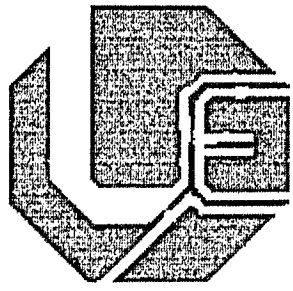
UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA



PROPOSTA DE APLICAÇÃO DAS REDES NEURAIS ARTIFICIAIS PARACONSISTENTES COMO CLASSIFICADOR DE SINAIS UTILIZANDO APROXIMAÇÃO FUNCIONAL

MAURICIO CONCEIÇÃO MARIO

MARÇO
2003

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PROPOSTA DE APLICAÇÃO DAS REDES NEURAIS ARTIFICIAIS PARACONSISTENTES COMO CLASSIFICADOR DE SINAIS UTILIZANDO APROXIMAÇÃO FUNCIONAL

Dissertação apresentada por Mauricio Conceição Mario à Universidade Federal de Uberlândia, para obtenção do título de Mestre em Engenharia Elétrica, aprovada em 25/03/2003 pela Banca Examinadora:

Professor Keiji Yamanaka, Ph.D (orientador)
Professor Haroldo Rodrigues de Azevedo, Dr. Eng. –Uniminas
Professor Gilberto Arantes Carrijo, Dr. Eng. - UFU
Professor João Inácio da Silva filho, Dr. Eng. (co-orientador)

PROPOSTA DE APLICAÇÃO DAS REDES NEURAIS ARTIFICIAIS PARACONSISTENTES COMO CLASSIFICADOR DE SINAIS UTILIZANDO APROXIMAÇÃO FUNCIONAL

MAURICIO CONCEIÇÃO MARIO

Dissertação apresentada por Mauricio Conceição Mario à Universidade Federal de Uberlândia para obtenção do título de Mestre em Engenharia Elétrica.

Prof. Keiji Yamanaka, Ph.D	**Prof. Alcimar Barbosa Soares, Ph.D**
Orientador	***Coordenador do Curso de Pós-Graduação***

RESUMO

Mario, M.C. Proposta de Aplicação das Redes Neurais Artificiais Paraconsistentes com Classificador de Sinais utilizando Aproximação Funcional, Uberlândia, 2003, 147p.

Este trabalho tem como objetivo desenvolver métodos para aplicações das Redes Neurais Artificiais Paraconsistentes -- RNAPs. Inicialmente são introduzidos tópicos sobre a Lógica Paraconsistente (com especial atenção para a célula paraconsistente e sua equação estrutural básica) e sobre as RNAPs (com destaque para a parte funcional das células que a compõem). A seguir é feita uma análise do comportamento da Célula Neural Artificial Paraconsistente de Aprendizagem, quando em sua entrada são inseridos padrões que representam valores discretos de um sinal; a partir destes resultados, evolui-se para a reprodução de um sinal por inteiro através de aproximação funcional. São, então, formadas Unidades Neurais Artificiais Paraconsistentes compostas de três tipos de células, capazes de aprender um padrão (sinal) e compará-lo com padrões que venham a ser testados pela rede, sendo que as saídas destas unidades são manipuladas de acordo com o grau de precisão que a aplicação exigir. O conjunto destas unidades neurais forma a Rede Neural Artificial Paraconsistente para Classificação de Sinais, capaz de aprender e armazenar vários tipos de padrões (sinais) e, quando inseridos padrões de teste na rede, classificá-los como pertencentes ou não à mesma.

A perspectiva é de que os resultados apresentados e a metodologia utilizada sirvam como divulgação para outros tipos de aplicação com as RNAPs.

Palavras Chaves:

Redes Neurais Artificiais Paraconsistentes, Classificação de Sinais, Aproximação Funcional.

ABSTRACT

Mario, M. C. Proposal of Application the Paraconsistents Artificials Neurals Networks with Signal Classifier useful Functional Approximation, Uberlândia, 2003, 147p.

This work has the objective of developping methods for applications of Paraconsistents Artificials Neurals Networks – PANNs. Firstly, it will be introduced some topics about Paraconsistent Logic (with prominence for the functional part of the cells that compose it) and also about PANNs (with prominence for the Paraconsistents Artificials Neurals Cells).

Later, it is made an analysis about the conduct of the Apprenticeship Paraconsistent Artificial Neural Cell, when it is inserted in its entrance patterns that represents discreet signal values. From this results, it develops for the entirely reproduction of a signal through functional approach. Then, it is formed the Paraconsistents Artificials Neurals Units, with three cells, capables to learn a pattern (signal) and also capable to compare these patterns with others introduced on the unit. The unit outlets are manipulated according to its application precision. The group of these units form the Paraconsistents Artificials Neurals Networks for Signals Classifiers, capable to learn and to store various types of patterns (signals) on the network, and also capable to classify these patterns as pertaining or not the same network when introduced patterns on it.

The perspective is that the results presented and the used methodology attend on the divulgation for others types applications with PANNs.

Key Words:

Paraconsistents Artificials Neurals Networks, Signals Classifiers, Functional Approximation.

PROPOSTA DE APLICAÇÃO DAS REDES NEURAIS ARTIFICIAIS PARACONSISTENTES COMO CLASSIFICADOR DE SINAIS UTILIZANDO APROXIMAÇÃO FUNCIONAL

SUMÁRIO

LISTA DE FIGURAS

página

LISTA DE FIGURAS

página

LISTA DE FIGURAS

página

LISTA DE FIGURAS

página

ACRÔNIMOS

LPA:	Lógica Paraconsistente Anotada
LPA2v:	Lógica Paraconsistente Anotada de Anotação de 2 valores
RNAP:	Rede Neural Artificial Paraconsistente
CNAP:	Célula Neural Artificial Paraconsistente
UNAP:	Unidade Neural Artificial Paraconsistente
EKG:	Eletrocardiograma
ECG:	Eletroencefalograma
EMG:	Eletromiograma
ENG:	Eletroneugrama
IA:	Inteligência Artificial
QUPC:	Quadrado Unitário no Plano Cartesiano
Gct:	Grau de Contradição
Gc:	Grau de Certeza
Vicc:	Valor inferior de controle de certeza
Vscc:	Valor superior de controle de certeza
Vsct:	Valor superior de controle de contradição
Vict:	Valor inferior de controle de contradição
CAPb:	Célula Artificial Paraconsistente básica
CNAPb:	Célula Neural Artificial Paraconsistente básica
EEB:	Equação Estrutural Básica
CNAPa:	Célula Neural Artificial Paraconsistente de aprendizagem
Fa:	Fator de Aprendizagem
UNAPCP:	Unidade Neural Artificial Paraconsistente de Comparação de Padrões
CNAPCLs:	Célula Neural Artificial Paraconsistente de Conexão Lógica Seletiva
CNAPd:	Célula Neural Artificial Paraconsistente de Decisão
OCR:	Optical Character Recognition
CNAPCa:	Célula Neural Artificial Paraconsistente de Conexão Lógica Analítica
UNAPCF:	Unidade Neural Artificial Paraconsistente de Classificação de Famílias

CAPÍTULO 1

Processamento e Classificação de Sinais

CAPÍTULO 1

Processamento e Classificação de Sinais

1.1 Introdução

A base teórica para a construção das Redes Neurais Paraconsistentes foi estabelecida pelos estudos das *lógicas não-clássicas*, mais precisamente a Lógica Paraconsistente Anotada [95], que é capaz de tratar sinais contraditórios sem trivialidade. Devido a esta característica de processar sinais recebidos em forma de anotações que podem trazer inconsistencias, a Lógica Paraconsistente Anotada [95] é indicada para ser utilizada em sistemas especialistas onde ocorrem situações *não triviais,* aumentando a precisão dos mesmos na tomada de decisão. Este trabalho faz uma adaptação da teoria de interpretação da Lógica Paraconsistente Anotada de 2 valores –LPA2v, apresentada em [Abe 92], aplicando-a a um sistema classificador de sinais. A expectativa é, que sob este tipo de aplicação, apareçam resultados satisfatórios devido à simplicidade na metodologia aplicada às células que compõem a rede, o que viria a demonstrar a sua elasticidade.
Comprovando-se estes resultados, estaria aberta a sua utilização para um nicho de aplicações que dependem de equacionamentos mais complexos.

Os objetivos da pesquisa, portanto, são:

a) Elaborar aplicações com as Redes Neurais Artificiais Paraconsistentes com base na sua estrutura teórica , objetivando implementações práticas, sendo que a ênfase neste trabalho são a utilização das redes na aproximação funcional para reprodução de sinais discretizados no tempo.

b) Construir algoritmo a partir das bases teóricas das Células Neurais Artificiais Paraconsistentes para ser aplicado em programas computacionais para aplicação em sistemas especialistas.

c) Propor aplicação do algoritmo para classificação de sinais elétricos discretizados no tempo.

d) Contribuir, a partir dos resultados obtidos, para que este trabalho sirva de referência nas comparações com outros métodos em aplicações de aproximação funcional.

Considerando que o trabalho tem a expectativa de que os resultados obtidos sirvam como referências para utilização das Redes Neurais Artificiais Paraconsistentes, a proposta é esquematizada conforme diagrama abaixo:

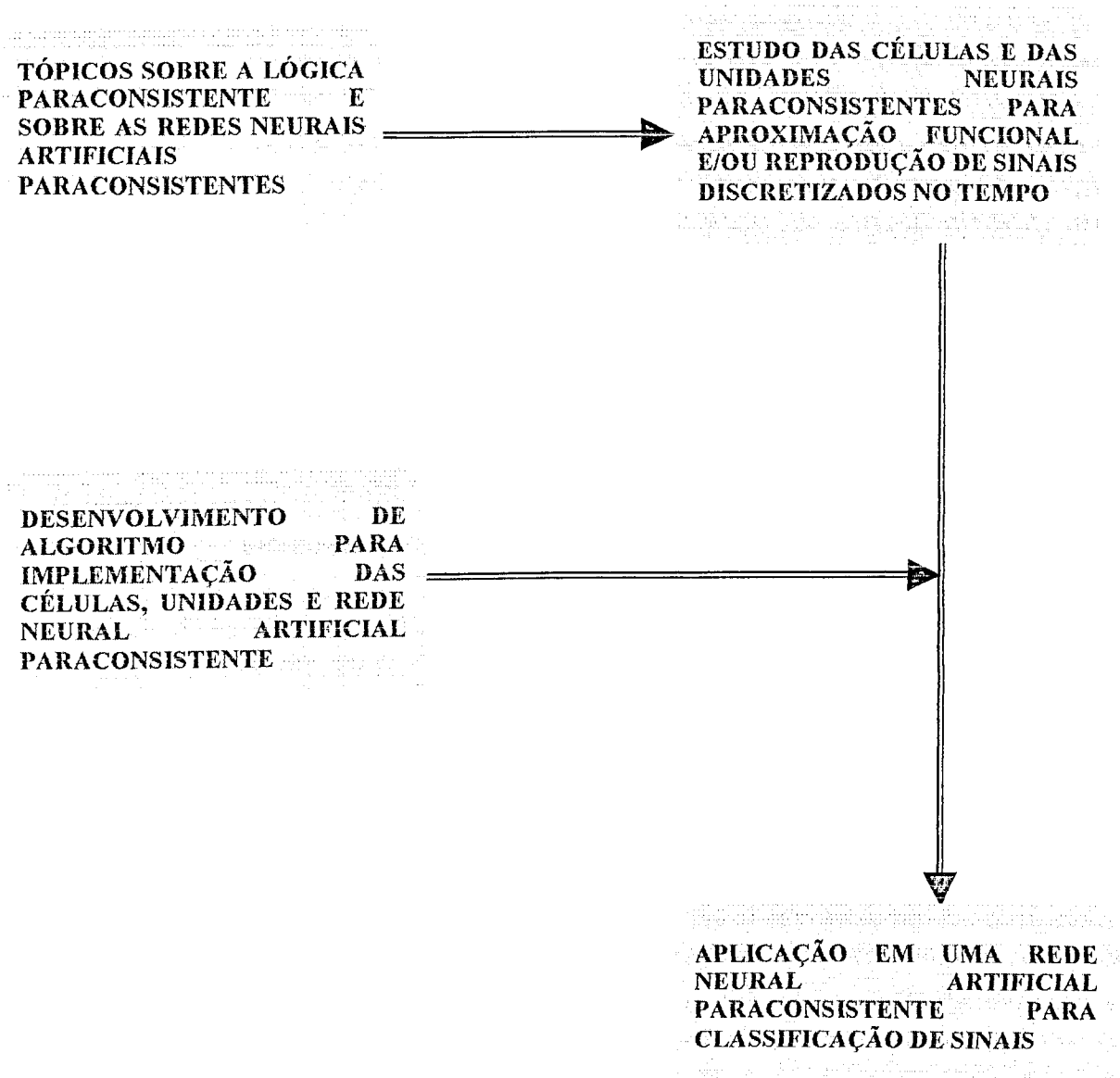


fig. 1.1: Esquema simplificado do trabalho de pesquisa

1.2. Sinais

Do ponto de vista de engenharia, sinais são funções ou seqüências que transportam informações de uma fonte de mensagens a um destinatário. As características específicas dos sinais dependem do canal de comunicações utilizado para o transporte do sinal. Um canal de comunicações é definido pelo tipo de distorção que introduz nos sinais, podendo esta ser do tipo: a) determinística linear (limitação da banda de freqüência dos sinais); b) determinística não linear (existência de saturações) ; c) aleatória (presença de ruídos) [8].

Em sistemas eletrônicos de comunicação a fonte geradora de informação, o canal de comunicação e o destinatário são elementos pré-definidos, geralmente com características bem especificadas. Em outras situações, como nos processos de medição em investigação científica, a fonte de mensagens e o canal de comunicações poderão estar apenas parcialmente caracterizados. Sinais bioelétricos como o eletrocardiograma (EKG), o eletroencefalograma (EEG), o eletromiograma (EMG) ou o eletroneugrama (ENG) são estudados há décadas com a finalidade de se extrair informação sobre estados patológicos de órgãos, sem que se tenha muitas vezes a certeza de que tal informação é de fato transportada por estes sinais. Por exemplo, os sinais que constituem a voz humana codificam uma variedade de informações: sobre a semântica do que está sendo dito, sobre a identidade do locutor, etc [8].

1.3 Classificação de Sinais

Dado o sinal representado pela equação 1.1:

$$\mathbf{y(t)=f(t,A)+n(t)} \qquad (1.1)$$

Supondo que o sinal y(t) possa pertencer a categorias $\{Cj\}^{M,\ j=1}$, conforme o valor do parâmetro A. Este é o caso por exemplo da tentativa de se diagnosticar condições cardíacas patológicas a partir de traçados de EKG. Para cada categoria diagnóstica j é escolhida uma coleção de traçados representativos:

$$\mathbf{\{y_{i,j}(t)\}^{Nj,\ i=1}\ j=1,2...M} \qquad (1.2)$$

Uma tentativa de classificação de sinais pode ser feita fazendo uma abordagem direta com técnicas de Redes Neurais, treinando-se uma rede neural multicamada N(W), esta rede neural multicamada terá como entradas todos os N pontos das séries temporais associadas a cada um dos sinais, e como saídas as M categorias em que se pretende classificá-los, como mostra a figura 1.2:

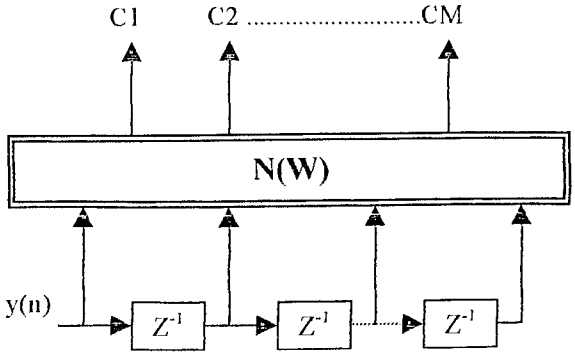


fig.1.2: Classificador Neural Direto de Sinais

Um problema para este tipo de abordagem é que cada série temporal é representada por cerca de 500 pontos amostrais para cada ciclo de EKG, o que impõe uma dimensão grande à rede, mesmo havendo poucas categorias de classificação [106].

Uma abordagem mais econômica é procurar reduzir a dimensão dos sinais a uma coleção menor de parâmetros que representem adequadamente os sinais originais e, implementar uma classificação neural para estes parâmetros.

Especificamente para a classificação de EKG, inúmeros métodos de parametrização foram propostos: medir amplitudes de pontos fiduciais (ondas P, Q, R, S, T, U) no traçado, áreas associadas, durações e separações [135]. Por um outro método, proposto a mais de duas décadas por Halliday [9], a coleção de todos os sinais é modelada por um processo aleatório Y(t) do qual cada traçado individual é uma particular função amostra. Este processo aleatório é em seguida representado por uma expansão ortonormal como o de Kahrunen-Loeve. Por esta expansão, determinam-se funções ortonormais $\{\Phi_i(t)\}$ chamadas de autofunções da função de covariança associada ao processo, tais que:

$$\int_0^t \Phi_i(t)\,\Phi_k(t)dt = \delta(i,k) \qquad (1.3)$$

$$y_k(t) = \lim_{n\to\infty} \sum_{i=1}^{n} a_{ki}\,\Phi_i(t) \qquad (1.4)$$

e os coeficientes associados a cada um dos sinais são calculados por:

$$a_{ki} = \int_0^t y_k(t)\,\Phi_i(t)\,dt \qquad (1.5)$$

Ordenando as médias destes coeficientes calculadas sobre toda a população de sinais em ordem decrescente dos seus valores absolutos, pode-se aproximar cada sinal individual apenas pelas L autofunções mais significativas:

$$y_k(t) = \sum_{i=1}^{L} a_{ki}\,\Phi_i(t) \qquad (1.6)$$

Deste modo, espera-se que o vetor de coeficientes $\mathbf{a_k}$ de dimensão L<<N represente adequadamente cada sinal com um erro de reconstrução suficientemente pequeno.

Pode-se agora buscar uma rede neural multicamada de dimensão mais viável, que, utilizando apenas estes coeficientes $\mathbf{a_k}$, classifique adequadamente os sinais nas J categorias, conforme representado nas figura 1.3 e 1.4. Zerbini [139] utilizou este procedimento para a classificação de traçados VCG's (vector cardiogramas).

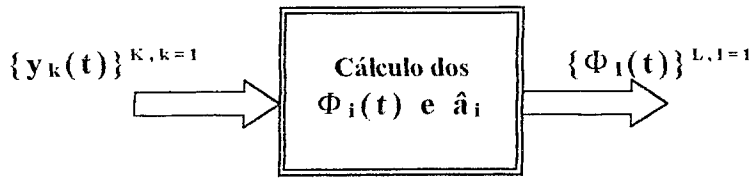


fig.1.3: Cálculo das autofunções usando toda a base de dados disponível

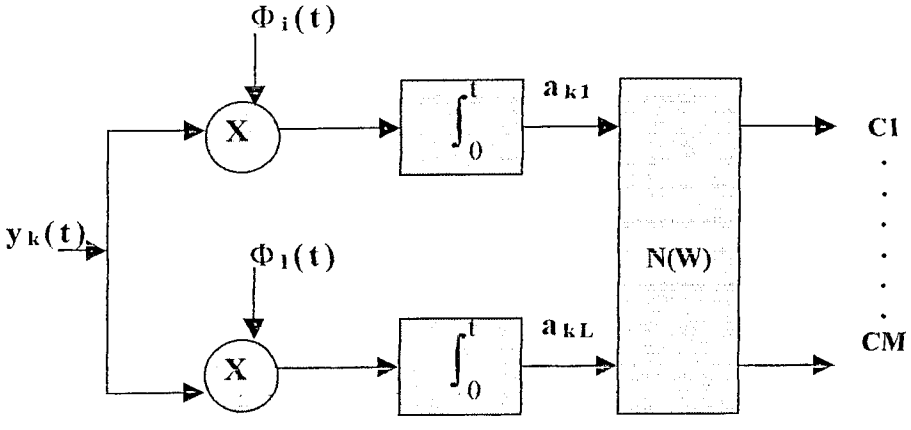


fig. 1.4: Cálculo dos coeficientes de cada um dos sinais para treinamento e posterior utilização do Classificador Neural

Esta metodologia que representa um dos paradigmas mais comuns para a utilização de redes neurais em processamento de sinais pode ser resumido da seguinte maneira:

Dado um conjunto de sinais que devem ser classificados em M categorias: $\{y_{i,j}(t)\}^{Nj,\ i=1}$ j=1,2...M, executam-se os seguintes passos:

1. Busca-se uma parametrização adequada. A particular parametrização escolhida determinará a eficiência e o desempenho do classificador.
2. Escolhe-se um conjunto de sinais representativos para o treinamento do classificador. Uma boa escolha resulta em um bom desempenho do classificador.
3. Este conjunto de sinais de treinamento é processado para gerar o conjunto de parâmetros definidos em 1. e que será o conjunto de treinamento para a rede multicamada.
4. Opta-se por uma arquitetura de rede: adaline, descriminadores, RHW ou FBR ou uma combinação destas. Avalia-se a dimensão das camadas.
5. Treina-se a rede.
6. Avalia-se o seu desempenho com um outro conjunto independente.

As aplicações de técnicas com redes neurais em processamento de sinais vem se concentrando principalmente na classificação de sinais em categorias. Resultados relevantes foram alcançados na classificação de sinais bioelétricos, particularmente EKG, EEG e EMG; na classificação de sinais de voz para reconhecimento de fonemas ou do locutor; na classificação para reconhecimento de ecos de sonar [106].

CAPÍTULO 2

Lógica Paraconsistente e Aplicações das Lógicas Não-Clássicas

CAPÍTULO 2

Lógica Paraconsistente e Aplicações das Lógicas Não-Clássicas

2.1 Introdução à Lógica Paraconsistente

A lógica paraconsistente[1] surgiu dos trabalhos formalizados por volta de 1948 de forma independente, pelo polonês Stanislaw Jàskowski e pelo brasileiro Newton C. A. Da Costa[2]. A lógica paraconsistente se destaca por infringir o princípio da não-contradição.

Diz-se que uma teoria dedutiva é *consistente* se não possuir teoremas contraditórios, um dos quais é a negação do outro. Caso contrário, a teoria diz-se *inconsistente* (ou *contraditória*). Uma teoria é chamada *trivial* se todas as fórmulas (ou sentenças) de sua linguagem forem nela demonstráveis; em hipótese contrária, diz-se *não-trivial* [94].

Analogamente, a definição aplica-se a sistemas de proposições, sistemas de informações etc, (levando-se em conta, o conjunto das conseqüências dos mesmos).

Se a lógica subjacente a uma teoria T é a lógica clássica ou alguma de suas extensões, T é inconsistente se e somente se for trivial. Portanto, para elaborar teorias ou sistemas de informação inconsistentes mas não-triviais, deve-se de recorrer a um tipo novo de lógica.

Lógica *paraconsistente* é uma lógica que pode servir de base para teorias inconsistentes e não-triviais.

Uma lógica é chamada *paracompleta* se pode funcionar como a lógica subjacente de teorias na qual há fórmulas tais que estas fórmulas e suas negações são ambas falsas. Uma teoria é chamada *paracompleta* se sua lógica subjacente é uma lógica paracompleta.

Como uma conseqüência, teorias paraconsistentes não satisfazem o princípio da não-contradição que pode ser expressa como segue: de duas proposições contraditórias (i.e., uma delas é negação do outro) uma deve ser falsa. Além disso, teorias paracompletas não satisfazem o princípio do terceiro excluído, formulado como se segue: de duas proposições contraditórias, uma deve ser verdadeira.

Finalmente, lógicas que são simultaneamente paraconsistentes e paracompletas chamam-se lógicas *não-aléticas* [94].

---

1 Nome dado pelo filósofo peruano Francisco Miró Quesada, em 1976, aos resultados dos estudos de Newton C. A.. da Costa.

2 Os cientistas S. Jàskowski e Newton C. A.. da Costa são considerados pela comunidade científica como os inventores da Lógica Paraconsistente.

Em algumas situações, quando necessitamos efetuar descrições mais próximas da realidade, a lógica clássica se mostra ineficaz ou mesmo impossibilitada de ser aplicada.

Exemplos de onde pode ser aplicada a lógica paraconsistente:

a) A proposição de que uma maçã é vermelha: a proposição pode ter uma conotação de verdadeira se a maçã for realmente vermelha, porém esta proposição pode ter uma conotação de aproximação de verdade, com um certo grau, se a maçã tiver um tom próximo a vermelho, ou uma conotação de falso com um certo grau de intensidade se a coloração tender a um tom verde. Na lógica clássica tal proposição não admitiria tons intermediários entre o vermelho e o verde, e ela assumiria somente conotações de verdadeiro ou falso.

b) Numa reunião de condomínio, para decidir uma reforma no prédio nem sempre as opiniões dos condônimos são unânimes. Se sempre houvesse unanimidade, facilitaria em muito a decisão do síndico. Alguns querem a reforma, outros não, gerando contradições. Outros nem mesmo têm opinião formada, gerando indefinições. A análise detalhada de todas as opiniões, contraditórias, indefinidas, contra e a favor pode originar buscas de outras informações para gerar uma decisão de aceitação ou não da reforma do prédio. A decisão tomada vai ser baseada nas evidencias trazidas pelas diferentes opiniões [95].

Alguns trabalhos, de cunho estritamente teórico, já sugeriram a lógica paraconsistente como uma boa solução para fazer tratamento de conhecimento incerto destas situações reais [10] [29] [33]. Atualmente vários centros importantes de pesquisa investigam formas de aplicação da lógica paraconsistente em projetos de sistemas especialistas que efetuem tratamentos adequados de situações não cobertas pela lógica clássica. A lógica paraconsistente encontrou várias aplicações em Inteligência Artificial (IA), programação lógica etc., mostrando-se de significado básico para a engenharia e ciência de computação [1] [4] [134].

Alguns resultados publicados demonstram que a análise de sinais utilizando a lógica paraconsistente anotada permite que vários problemas ocasionados por situações contraditórias e paracompletas podem ser tratados de uma maneira próxima da realidade através da consideração de evidências.

Os resultados de alguns trabalhos relevantes expostos na literatura especializada nos sugerem a possibilidade de aplicações diretas da Lógica Paraconsistente Anotada na área de Inteligência Artificial. Nestes trabalhos são encontradas promissoras pesquisas para construção de sistemas de controle e de programas aplicativos de simulação. Entre os trabalhos publicados destaca-se [71] onde métodos computacionais de análises paraconsistentes podem ser projetados através do algoritmo da Lógica Paraconsistente Anotada, denominado "Para-Analisador".

Os estudos posteriores apresentados em trabalhos que tratam de sistemas computacionais, permitem afirmar que o algoritmo pode ser implementado por *software*, como programa aplicativo e desenvolvido utilizando linguagem de computação convencional, proporcionando a aplicação da lógica paraconsistente em sistemas de controle e sistemas especialistas de inteligência artificial.
As aplicações das lógicas Não-Clássicas fazem com que Sistemas de controle de Robôs funcionem imitando o comportamento humano quando na resolução de situações do mundo real que envolve informações incompletas possíveis de gerar contradições.

2.2 Lógica Paraconsistente Anotada

Os estudos da Lógica Paraconsistente originaram outras classes de Lógicas Não-Clássicas, denominadas de **Lógicas Paraconsistentes Anotadas** [134].

A seguir são formalizados alguns conceitos da Lógica Proposicional Paraconsistente Anotada **Pτ:**

Seja um reticulado finito, denominado **reticulado de valores verdade**, $\tau = \{ T, v, f, \perp \}$, onde intuitivamente, as constantes anotacionais do reticulado de 4 anotações representam:

T = inconsistente

F = falso

V = verdadeiro

$\perp$ = paracompleto

O Reticulado é chamado de **Reticulado de Hasse**, e pode ser representativo da Lógica Paraconsistente Anotada, onde as anotações ou variáveis proposicionais são alocadas nas suas quatro arestas conforme a figura abaixo:

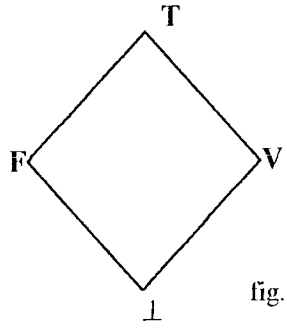


fig. 2.1: Reticulado de Hasse

O operador negação sobre τ é : $\sim : | \tau | \rightarrow | \tau |$, que define-se como:

$\sim (1) = 0$

$\sim (0) = 1$

$\sim (T) = T$

$\sim (\perp) = \perp$

Na lógica paraconsistente anotada a notação px pode ser entendida como "creio na proposição p com grau de crença de no máximo x, ou até ($\leq$ x) "; considera-se, portanto, o **grau de crença** como sendo uma constante anotacional do reticulado. Então, cada grau de crença atribuído à proposição é um valor que está contido no conjunto de valores composto pelas constantes anotacionais do reticulado $\{ T, V, F, \perp \}$.

De acordo com o que foi exposto, as variáveis proposicionais virão acompanhadas por um grau de crença, que dará uma conotação de "verdade", de "falsidade", de "inconsistência" ou de "paracompleteza" à proposição.

pT = a anotação ou grau de crença atribui uma conotação de inconsistente à proposição p.

p1 = a anotação ou grau de crença atribui uma conotação de verdade à proposição p.

p0 = a anotação ou grau de crença atribui uma conotação de falsidade à proposição p.

$p\perp$ = a anotação ou grau de crença atribui uma conotação de paracompleteza à proposição p.

No exemplo, considerou-se uma lógica com quatro anotações. No entanto uma lógica pode ter mais anotações, com consequências de que as proposições assumam novos valores ou valores intermediários entre os quatro apresentados. O reticulado representativo da Lógica Paraconsistente Anotada poderia também ter outros formatos.

2.3 Lógica Paraconsistente Anotada de anotação com dois valores – LPA2v

Uma nova interpretação da Lógica Paraconsistente Anotada pode ser vista quando a proposição vem com dois valores anotados. Na Lógica Paraconsistente Anotada de anotação com dois valores, para cada proposição associa-se dois valores de graus. O primeiro valor contido na anotação representa a evidência favorável à proposição p **(crença - μ1)**, e o segundo valor da anotação representa a evidência contrária à proposição p **(descrença - μ2)**.

Para o reticulado de Hasse com anotação de dois valores, tem-se:

$$\tau = \{ (\mu 1, \ \mu 2) \} \mid \mu 1, \ \mu 2 \in [\, 0,1 \,] \subset \Re \}, \qquad (2.1)$$

e se p é uma fórmula básica, o operador $\sim : | \tau | \rightarrow | \tau |$ é definido como:

$\sim [(\mu 1, \mu 2)] = (\mu 2, \mu 1)$, onde $\mu 1, \mu 2 \in \{ x \in \Re \mid 0 \leq x \leq 1 \}$, sendo $\mu 1, \mu 2$ uma anotação de p.
O reticulado de quatro vértices para a LPA2v é mostrado a seguir:

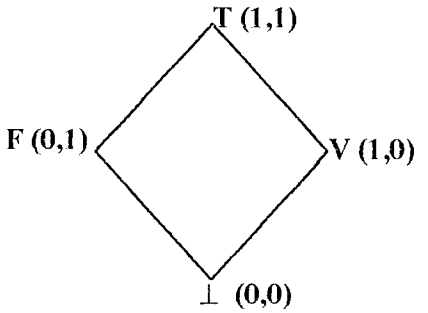


fig. 2.2: Reticulado LPA2v

Pode-se relacionar os estados lógicos extremos representados nos quatro vértices do reticulado com os valores dos graus de crença e descrença:

pT = $p(1,1) \Rightarrow$ a anotação composta pelos grau de crença e descrença atribui à proposição p uma leitura intuitiva que p é inconsistente.

$p1$ = $p(1,0) \Rightarrow$ a anotação composta pelos grau de crença e descrença atribui à proposição p uma leitura intuitiva que p é verdadeiro.

$p0$ = $p(0,1) \Rightarrow$ a anotação composta pelos grau de crença e descrença atribui à proposição p uma leitura intuitiva que p é falso.

$p\perp$= $p(0,0) \Rightarrow$ a anotação composta pelos grau de crença e descrença atribui à proposição p uma leitura intuitiva que p é paracompleto.

2.4 Análise da Lógica Paraconsistente Anotada de anotação com dois valores – LPA2v

Para uma análise paraconsistente na LPA2v, as anotações podem ser interpretadas como evidências e, portanto, quando o sistema paraconsistente projetado com esta lógica recebe informações contraditórias ou paracompletas, estas evidências em forma de graus de crença e descrença possuem papel importante na tomada de decisão.

Após a análise paraconsistente o estado lógico resultante na saída é que leva a uma conclusão na tomada de decisão. O estado lógico da saída é encontrado através do equacionamento entre os dois valores evidenciais que compõem a anotação ($\mu 1$, $\mu 2$), onde $\mu 1$, é o grau de crença atribuído à proposição e $\mu 2$ é o grau de descrença atribuído à proposição.

Na prática, quando na utilização da LPA2v, os graus de crença e descrença são considerados como informações de entrada do sistema e os estados lógicos representados nos vértices e nas regiões internas do reticulado são as saídas resultantes da análise paraconsistente.

Representação:

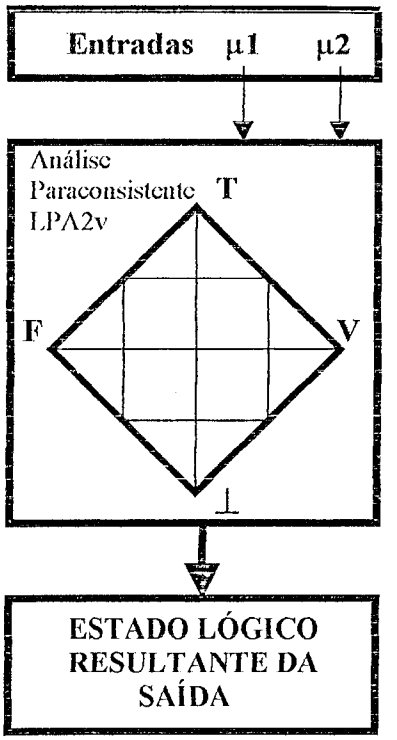


fig. 2.3: Representação da Análise Paraconsistente LPA2v

Na condição de alto grau de contradição o sistema solicita mais informações e à medida que vão chegando novas evidências (**μ1 , μ2**) para análise pode ir diminuindo as contradições, dando condições ao sistema de chegar a uma conclusão mais acertada.

2.4.1 Representação no Quadrado Unitário no Plano Cartesiano – QUPC

A LPA2v pode ser interpretada [95] conforme uma representação do reticulado em um quadrado unitário no plano cartesiano, como mostra a figura abaixo:

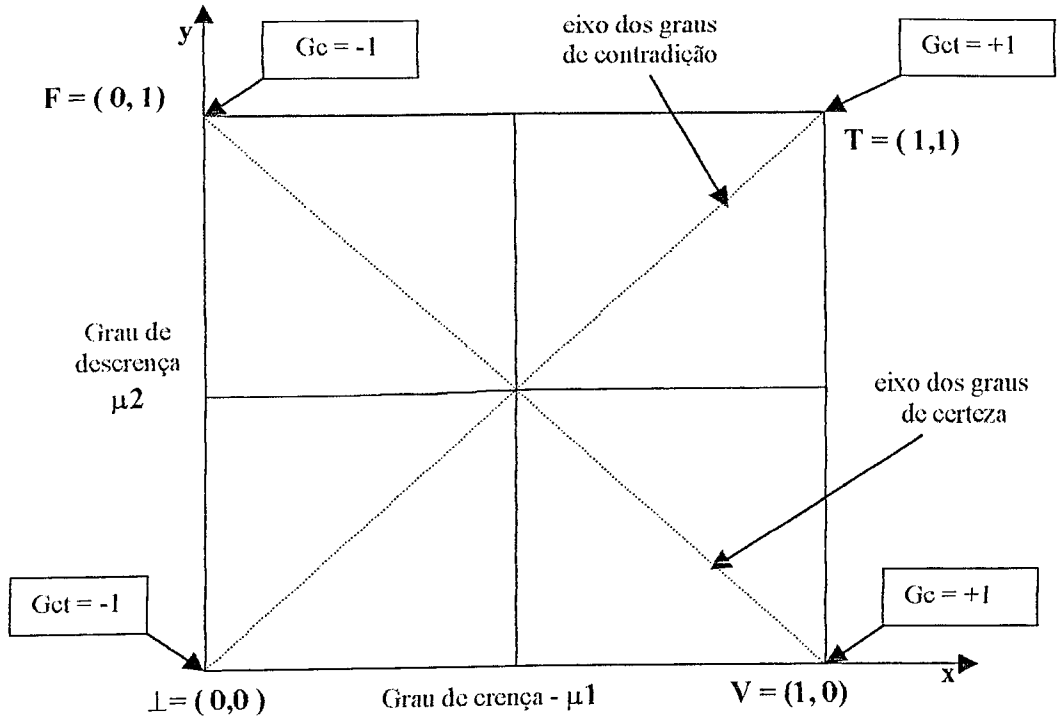


fig. 2.4: Quadrado Unitário no Plano Cartesiano - QUPC

Em uma análise da LPA2v os valores anotados vindos em forma de evidências são equacionados gerando graus de contradição "Gct" e graus de certeza "Gc". Estes dois novos valores são definidos como:

a) Gct = grau de contradição:

$$\mathbf{Gct = \mu 1 + \mu 2 - 1} \qquad (2.2)$$

Varia entre 1 e –1. No quadrado unitário do plano cartesiano da figura 2.4 vê-se que: Gct = -1 (no ponto ⊥= (0,0)) representa uma contradição máxima negativa e Gct = +1 (onde T = (1, 1)) representa uma contradição máxima positiva. Quanto mais a interpolação entre os graus de crença e descrença estiver próxima à reta FV, mais o resultado μ1 + μ2 se aproxima de 1, diminuindo o valor de Gct, o que representa uma menor contradição entre as informações de entrada.

b) Gc = grau de certeza:

$$\mathbf{Gc = \mu 1 - \mu 2} \qquad (2.3)$$

Varia entre 1 e –1. Quando Gc = -1 (no ponto F = (0, 1)), significa que tem-se uma certeza máxima da negação da proposição. Para Gc = +1 (no ponto V = (1, 0)), significa que tem-se uma certeza máxima da afirmação da proposição. Estes valores máximos significam informações consistentes. Quanto mais a interpolação entre os graus de crença e descrença estiver próxima à reta $\perp$T, mais o resultado $\mu 1 - \mu 2$ se aproxima de 0, diminuindo o valor de Gc, o que representa uma menor certeza entre as informações de entrada, porque significa uma maior coincidência nos valores dos graus de crença e descrença.

2.4.2 Representação no Reticulado LPA2v

Considerando os valores encontrados no quadrado unitário do plano cartesiano[95] pode-se estender a análise em uma representação de 2 eixos: um com os valores do grau de contradição e outro com os valores do grau de certeza. Estes dois eixos são sobrepostos de tal forma a serem comparados com o reticulado do LPA2v, conforme a figura abaixo:

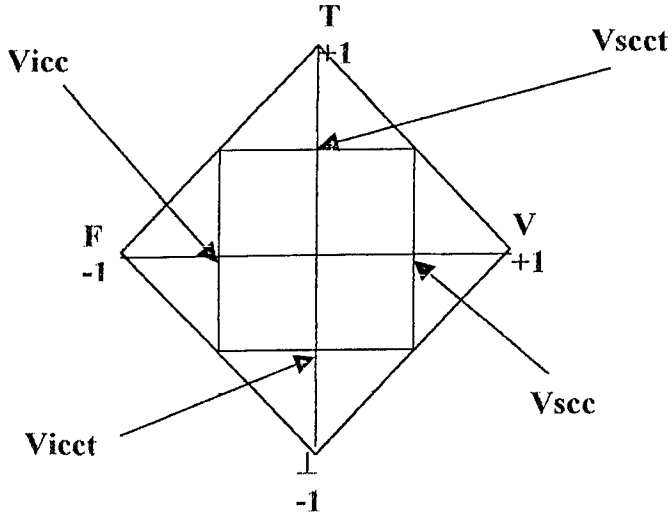


fig. 2.5: Representação do Reticulado LPA2v

O reticulado pode ser delimitado conforme a relação abaixo:

Vicc : Valor inferior de controle de Certeza (variando entre 0 e –1);

Vscc : Valor superior de controle de Certeza (variando entre 0 e +1);

Vsect : Valor superior de controle de Contradição (variando entre 0 e +1);

Vicct : Valor inferior de controle de Contradição (variando entre 0 e –1);

Os dois limites externos e arbitrários Vscc e Vicc determinam quando o grau de Certeza resultante é alto o suficiente para que a proposição analisada seja considerada totalmente verdadeira ou totalmente falsa.

Os dois limites externos e arbitrários Vscct e Vicct determinam quando o grau de Contradição resultante é alto o suficiente para que a proposição analisada seja considerada totalmente consistente ou totalmente inconsistente.

2.5 Algoritmo Para-Analisador simplificado.

Em uma descrição do reticulado da figura 2.5 pode-se obter o algoritmo da LPA2v denominado Para-Analisador. O algoritmo Para-Analisador simplificado é descrito como:

1. Entrar com os valores de Vicc, Vscc, Vicct e Vscct ;
2. Entrar com $\mu 1$ e $\mu 2$;
3. Calcular o Grau de Contradição : $Gct = (\mu 1 + \mu 2) - 1$
4. Calcular o Grau de Certeza : $Gc = \mu 1 - \mu 2$
5. Determinar os estados lógicos de saída:
    a) se $Gc \geq Vscc$, então $S1 = V$;
    b) se $Gc \leq Vicc$, então $S1 = F$;
    c) se $Gct \geq Vscct$, então $S1 = T$;
    d) se $Gct \leq Vicct$, então $S1 = \perp$;
    e) senão $S1 = I$ (indefinição)

onde S1 = saída discreta.

2.6 Operadores Lógicos da LPA2v:

Dada uma proposição anotada A: PA ($\mu 1^A$, $\mu 2^A$), onde:

$\mu 1^A$ = grau de crença da proposição A;

$\mu 2^A$ = grau de descrença da proposição A;

Dada uma proposição anotada B: PB ($\mu 1^B$, $\mu 2^B$), onde:

$\mu 1^B$ = grau de crença da proposição B;

$\mu 2^B$ = grau de descrença da proposição B;

E sendo na saída:

$\mu 1^R$ = grau de crença resultante na saída;

$\mu 2^R$ = grau de descrença resultante na saída;

Pode-se definir os seguintes operadores lógicos:

2.6.1 Operador Lógico OR

A maximização ou conjunção A $\wedge$ B entre as proposições A e B é obtida pelas condicionais:

Se $\mu 1^A \leq \mu 1^B$ então $\mu 1^R = \mu 1^B$

senão $\mu 1^R = \mu 1^A$

Se $\mu 2^A \leq \mu 2^B$ então $\mu 2^R = \mu 2^A$

senão $\mu 2^R = \mu 2^B$

Representação:

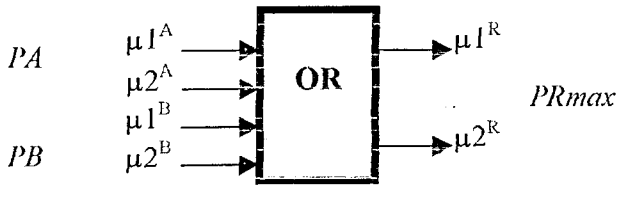


fig. 2.6: Operador Lógico OR

2.6.2 Operador Lógico AND

A minimização ou disjunção A $\vee$ B entre as proposições A e B é obtida pelas condicionais:

Se $\mu 1^A \geq \mu 1^B$ então $\mu 1^R = \mu 1^B$

senão $\mu 1^R = \mu 1^A$

Se $\mu 2^A \geq \mu 2^B$ então $\mu 2^R = \mu 2^A$

senão $\mu 2^R = \mu 2^B$

Representação:

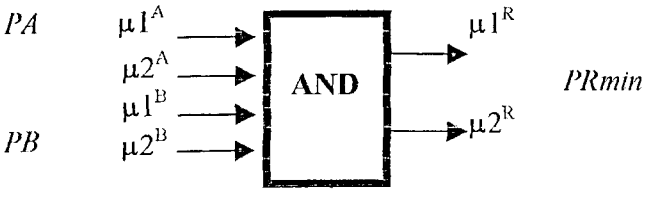


fig. 2.7: Operador Lógico AND

2.6.3 Operador Lógico NOT

A negação da proposição A é obtida pela ação de permuta entre o valor do grau de crença pelo de descrença:

$\mu 1^R = \mu 2^A$

e $\mu 2^R = \mu 1^A$

Representação:

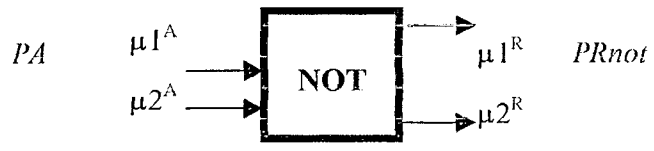


fig. 2.8: Operador Lógico NOT

2.7 Célula Artificial Paraconsistente Básica – CAPb

Conforme escrito em [95], em um sistema de análise paraconsistente pode-se considerar um algoritmo simplificado como uma célula neural artificial paraconsistente básica que vai gerar outros tipos de células que serão estudadas nos próximos capítulos. A figura e a descrição da célula artificial básica pode ser vista a seguir:

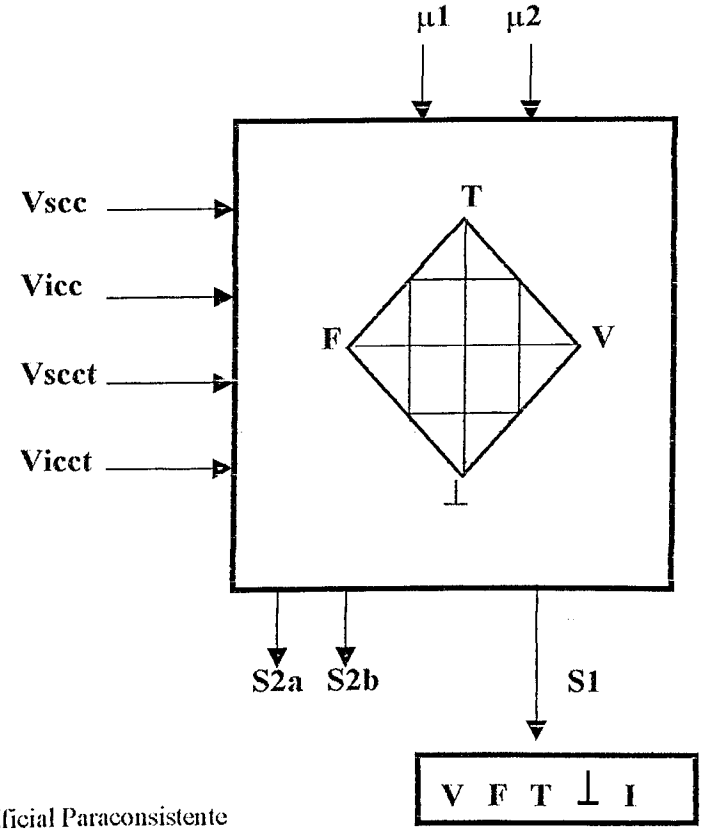


fig. 2.9: Célula Artificial Paraconsistente

Onde:

Entradas:

μ1 e μ2: Graus de Crença e Descrença da informação ($0 \leq \mu \leq 1$) ;

Vicc : Valor inferior de controle de Certeza ;

Vscc : Valor superior de controle de Certeza ;

Vscct : Valor superior de controle de Contradição ;

Vicct : Valor inferior de controle de Contradição ;

Saídas:

S2a = Gct : Grau de Contradição ;

S2b = Gc : Grau de Certeza ;

S1 : **T** = Inconsistente

F = Falso

V = Verdadeiro

$\perp$ = Paracompleto

I = Indefinição

2.8. Algumas aplicações utilizando Lógicas Não-Clássicas

2.8.1. Para-Sim: Simulador de Controle Lógico Paraconsistente

Consiste na descrição da implementação da Lógica Paraconsistente Aplicada a um Robô Móvel autônomo. O *Para-Sim* é um software Simulador do método de aplicação da Lógica Paraconsistente Anotada " LPA2v " em Robótica, e simula esta análise Paraconsistente aplicada ao robô.

O software simula o comportamento do robô através de variáveis como trajetória, posição e velocidade. Conclusivamente o trabalho ilustra a aplicação da Lógica Paraconsistente Anotada de anotação com dois valores LPA2v [134] em Robótica, demonstrando que a análise de sinais de informações sobre o ambiente utilizando o algoritmo " Para-Analisador " (será visto posteriormente no capítulo referente à Lógica Paraconsistente) consegue dar um tratamento eficiente a vários problemas ocasionados por situações contraditórias e paracompletas [96].

2.8.2. Teste de Validade para um ensaio por Eddy Current em Tubos de Geradores de Vapor Usando Lógica "Fuzzy" Paraconsistente

Baseando-se na lógica paraconsistente anotada é proposta uma metodologia para teste de validade de ensaios pôr "Eddy – Current". O objetivo é aumentar a credibilidade ou os índices de acertos dos diagnósticos das inspeções realizadas em tubos de geradores de vapor em usinas nucleares de eletricidade. Utiliza-se a forma "fuzzy" da lógica paraconsistente que permite a análise de consistência e da validade da informação. A implementação do método é feita através da "Toolbox Fuzzy" do programa Matlab, o que limita uma programação mais complexa. Um exemplo ilustrativo é apresentado para demonstrar a aplicabilidade do método. Conclui-se quanto a viabilidade do uso do método, porém funções de pertinência e regras de inferência necessitam ainda serem desenvolvidas [133].

2.8.3. Reconhecimento de Dígitos Manuscritos Usando Transformada de Hough e Redes Neurais

Linha de pesquisa na área de visão computacional, envolvendo o reconhecimento óptico de caracteres (OCR). A motivação para pesquisas nesta área tem sido as demandas por sistemas que sejam capazes de interpretar dados óticos consistindo de caracteres escritos manualmente ou por máquinas [137]. Uma aplicação de OCR imprescindível é a identificação e a separação automática de correspondências nas empresas de correios, através dos códigos de endereçamento postal [Shihari 91]. O objetivo do trabalho é avaliar a eficácia de técnicas baseadas na transformada de Hough e

redes neurais para fazer o reconhecimento ótico de dígitos manuscritos em campos de códigos de endereçamento postal de correspondências.

A técnica adotada consiste em digitalizar as imagens dos códigos manuscritos; detectar e afinar bordas dos dígitos; calcular a transformada de Hough das imagens das bordas dos dígitos; extrair descritores a partir do espaço de Hough; classificar os dígitos utilizando os descritores e redes neurais [107].

2.8.4. Concepção de um Sistema de Agentes Independentes Paraconsistente para o Tratamento de Cheques Bancários Brasileiros

Trabalho que faz parte do *Projeto Multicheck*[1], que define uma arquitetura de agentes autônomos para o tratamento automático de cheques bancários brasileiros manuscritos. A competência desses agentes é implementada em duas camadas. A primeira corresponde aos algoritmos de reconhecimento de padrões aplicados diretamente sobre segmentos de imagens. A segunda, corresponde aos mecanismos de raciocínio aplicados sobre as informações provenientes da primeira camada para validá-las ou interpretá-las. Essa interpretação envolve também informações provenientes de outros agentes que podem gerar inconsistências. Este problema é tratado adequadamente e naturalmente através de conceitos e operadores da lógica paraconsistente. O artigo trata da primeira camada aqui citada, e as informações referentes a segunda camada são obtidas a partir de uma base de dados simulada [50].

2.8.5. Um Modelo Neural de Predição Baseado em Redes MLP para Aplicação em WEB Mining

São obtidas informações de usuários de internet quando estes interagem com determinado site, e a partir daí o trabalho tem por objetivo o desenvolvimento de um agente de software e de métodos de análise do conjunto de padrões dos dados de comportamento de usuários, de forma a permitir predição sobre suas preferências e comportamentos, potencializando a navegação e exibição de informações relacionadas aos perfis destes usuários. O Agente utiliza uma Rede Neural Artificial com arquitetura Multi Layer Perceptron para classificação dos padrões. A partir da interação do usuário com o site e baseada em um conjunto de padrões de comportamento, é feita uma classificação e uma predição sobre a apresentação de banners relacionados às preferências dos usuários [121].

---

1. Projeto realizado na PUC-Paraná com apoio do governo (CNPq) no quadro de colaboração internacional entre l'École de Techonology Supérieure (ETS) do Canadá e a PUC-Paraná.

CAPÍTULO 3

Células, Unidades e Redes Neurais Artificiais Paraconsistentes

CAPÍTULO 3

Células, Unidades e Redes Neurais Artificiais Paraconsistentes

3.1 Aplicação da Lógica Paraconsistente em Redes Neurais

De acordo com estudos apresentados em [1] e [95] sobre Lógica Paraconsistente Anotada, sistemas baseados nesta lógica podem manipular em seu interior contradições ou inconsistências de modo não trivial. Nestes trabalhos foram elaborados um analisador paraconsistente e um para-control, de onde foram sugeridas várias aplicações ligadas à robótica.

Tais idéias inspiraram na confecção de um modelo de célula neural que verificou-se possuir propriedades que permitem aceitar e tratar sinais contraditórios combinando-se várias destas células, formando uma rede onde o conjunto de células interligadas pode armazenar dados em conflito e manipulá-los.

A unidade básica da rede é fundamentada em uma Célula Neural Artificial Paraconsistente. Várias destas Células Artificiais Paraconsistentes quando conectadas dão origem às Unidades Neurais Artificiais Paraconsistentes, que por sua vez quando agrupadas formam as **Redes Neurais Artificiais Paraconsistentes**:

3.2 Célula Neural Artificial Paraconsistente Básica - CNAPb

A figura abaixo mostra uma Célula Neural Artificial Paraconsistente básica – CNAPb com suas entradas e saídas;

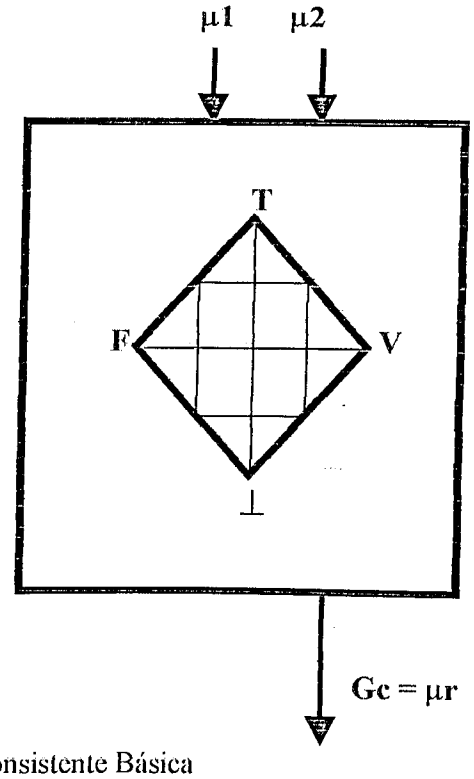


fig. 3.1: Célula Neural Artificial Paraconsistente Básica

Entradas: $\mu 1$ e $\mu 2$ $\Rightarrow$ Graus de Crença e Descrença da informação, onde $0 \leq \mu \leq 1$.

Na célula neural artificial paraconsistente básica o resultado da análise vem em forma de um valor analógico do grau de certeza (Gc = μr). O grau de certeza, dentro de uma rede, é transferido a outras células, neste caso como informação de entrada (grau de crença).

3.3 Equação Estrutural Básica - EEB

Quando várias CNAPb[s] estão interligadas, os valores de saída devem ser normalizados dentro de uma faixa entre [0, 1].

Conforme apresentado em [Da Silva Filho & Abe 01] as informações podem ser recebidas através de sensores entre as células que trazem sinais de valores reais entre 0 e 1, para serem considerados na forma de grau de credibilidade. A ponderação de valores e a troca de informações para processamento é feita pelas inúmeras conexões da rede através de sua **Equação Estrutural Básica (EEB):**

$$\mu r = \frac{\mu 1A - \mu 1Bc + 1}{2} \qquad (3.1)$$

Onde:

μr = Grau de Crença Resultante ;

$\mu 1A$ = Grau de Crença da célula A;

$\mu 1Bc$ = Grau de Crença Complementado da célula B;

3.4 Outros tipos de Células Neurais Artificiais Paraconsistentes

Cada um dos tipos de célula é originada do aperfeiçoamento do algoritmo Para-Analisador Simplificado, e terão diferentes funções, todas baseadas nas equações da Lógica Paraconsistente Anotada. A seguir são listadas algumas delas:

3.4.1 Célula Neural Artificial Paraconsistente de Conexão Analítica

Esta célula tem a função de fazer a análise analógica dos sinais e conectar todas as células componentes da rede. A descrição detalhada desta célula será feita no capítulo 8.2.1.

3.4.2 Célula Neural Artificial Paraconsistente de Conexão Lógica Simples

Esta célula tem a função de fazer a análise lógica dos sinais utilizando conectivos lógicos como o de maximização e minimização. A descrição desta célula (maximização) será descrita com detalhes no capítulo 6.1.1.

3.4.3 Célula Neural Artificial Paraconsistente de Conexão Lógica Seletiva

Esta célula tem a função de fazer a análise lógica dos sinais utilizando conectivos lógicos e, simultâneamente, selecionar qual de suas saídas deve ficar ativa (possui duas entradas e duas saídas distintas).

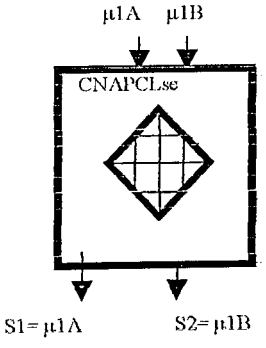


fig. 3.2.: célula de conexão lógica seletiva

$\mu r = [(\mu 1A - \mu 1B) + 1] / 2$

maximização:
se $\mu r > 1/2$ então $S1 = \mu 1A$ e $S2 = 1/2$
senão: $S1 = 1/2$ e $S2 = \mu 1B$

minimização:
se $\mu r > 1/2$ então $S1 = 1/2$ e $S2 = \mu 1B$
senão: $S1 = \mu 1A$ e $S2 = 1/2$

3.4.4 Célula Neural Artificial Paraconsistente de Passagem

Esta célula tem a função de direcionar o fluxo de informações para determinada região da rede.

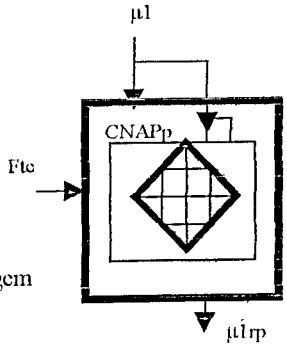


fig. 3.3.: célula de passagem

$\mu R = [(\mu 1A - \mu 1c) + 1] / 2$
$\mu 1c = 1 - \mu 2$
se $\mu R \geq Ftc$ então $\mu 1rp = \mu R$
senão $\mu 1rp = 1/2$

3.4.5 Célula Neural Artificial Paraconsistente de Complementação

Esta célula tem a função de fazer a complementação do valor em relação à unidade de qualquer sinal aplicado à sua entrada.

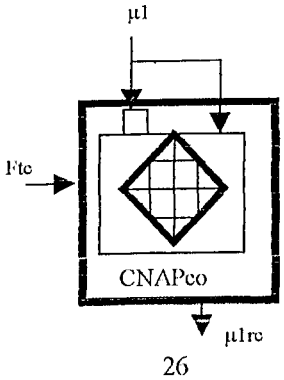


fig. 3.4.: célula de complementação

$\mu R = [(\mu 1c - \mu 1) + 1] / 2$
$\mu 1c = 1 - \mu 1$
se $\mu R \geq Ftc$ então $\mu 1rc = \mu R$
senão $\mu 1rc = 1/2$

3.4.6 Célula Neural Artificial Paraconsistente de Decisão

Esta célula tem a função de fazer a análise paraconsistente e determinar uma decisão baseada nos resultados da análise. A decisão vem na forma de três estados lógicos: verdadeiro, falso e indefinido. A descrição detalhada desta célula será feita no capítulo 6.1.2.

3.4.7 Célula Neural Artificial Paraconsistente de Aprendizagem

Esta célula tem a função de aprender e desaprender padrões que sejam aplicados respectivamente à sua entrada. A descrição detalhada desta célula será feita no capítulo 4.1.

3.4.8 Célula Neural Artificial Paraconsistente de Memorização

Esta célula tem a função de guardar os padrões aprendidos pela célula de aprendizagem num processo de funcionamento de aprendizagem/memorização.

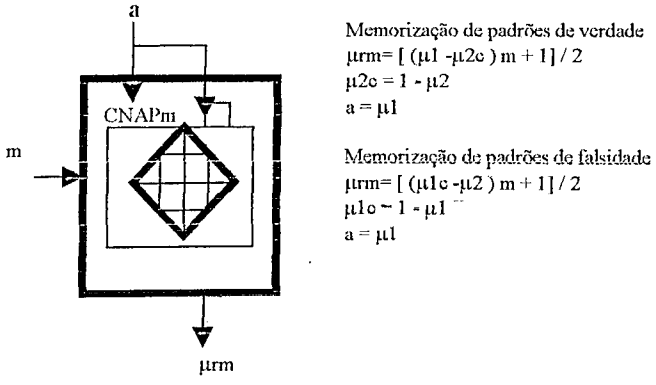


fig. 3.5.: célula de memorização

3.5 Unidades Neurais Artificiais Paraconsistentes UNAP^s

Segundo Da Silva Filho e Abe [94] as Unidades Neurais Artificiais Paraconsistentes (UNAP^s) podem ser comparadas aos blocos componentes estruturais da rede de neurônios biológicos do cérebro, compondo Sistemas Neurais de Raciocínio Lógico com várias finalidades, onde por meio de inferências são capazes de modelar as funções mentais do cérebro, fazendo inclusive tratamento de contradições. Estes blocos são interligados para que possam efetuar análises em conjunto, formando os Sistemas Neurais Artificiais Paraconsistentes de raciocínio lógico. Quando interligados desta forma o conjunto de UNAP^s apresenta funcionamento paralelo e distribuído, e que pode tratar sinais de forma analógica, permitindo elaboração de projetos de software aplicativos em linguagem de programação convencional.

3.6 Estrutura de uma Rede Neural Artificial Paraconsistente

A estrutura de uma Rede Neural Artificial Paraconsistente é composta hierarquicamente por estruturas menores, sendo que a base é formada pelas CNAP^s, que interligadas formam as Unidades Neurais Artificiais Paraconsistentes (UNAP^s). A interligação destas Unidades Neurais Artificiais Paraconsistentes formam por sua vez, as Redes Neurais Artificiais Paraconsistentes - RNAP^s.

Esta cadeia é representada através do esquema da figura 3.6:

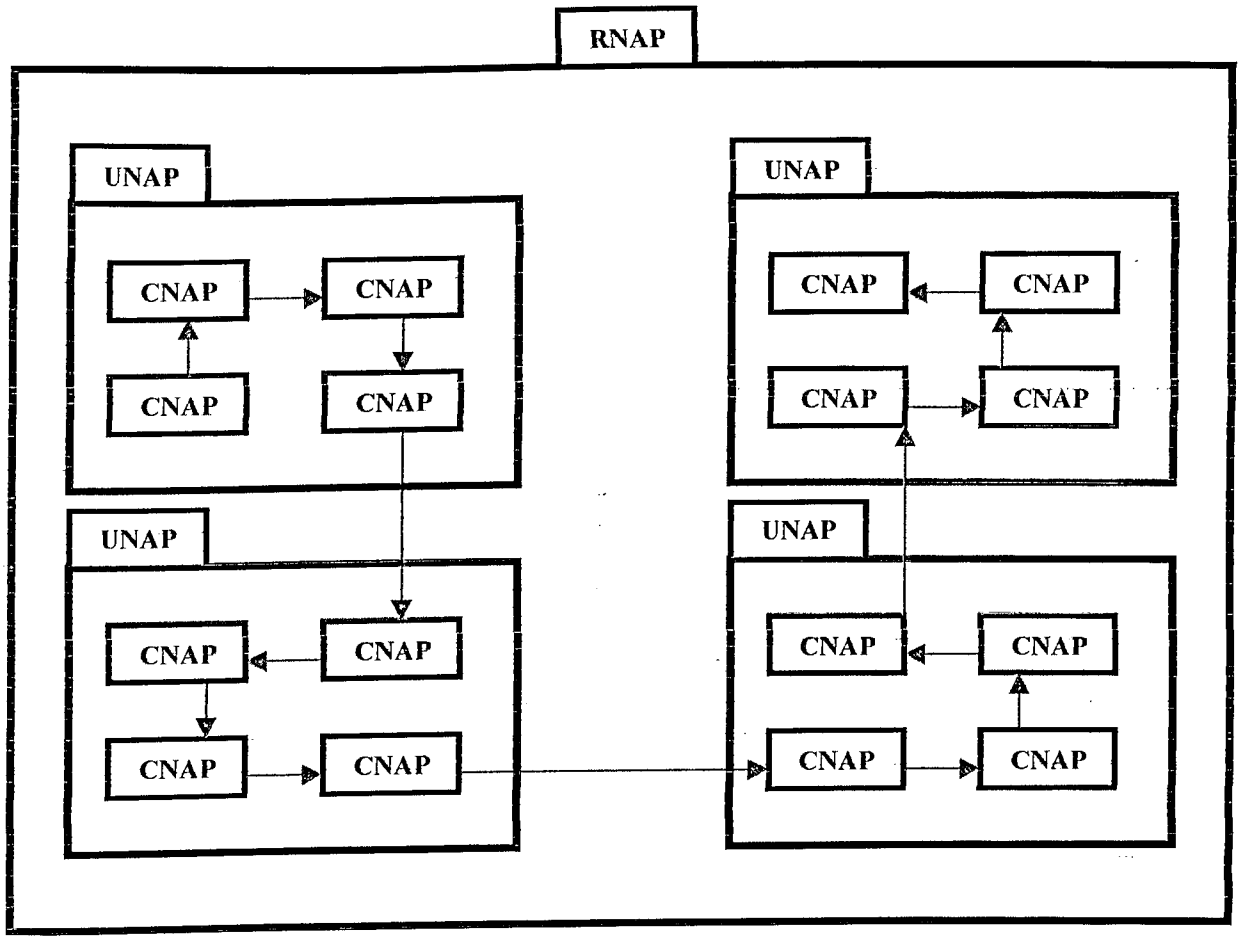


fig. 3.6: Estrutura de uma Rede Neural Artificial Paraconsistente

3.7 Características das Redes Neurais Artificiais Paraconsistentes

Segundo os estudos e ensaios elaborados em [94], as **Redes Neurais Artificiais Paraconsistentes** possuem as seguintes características:

- capacidade de aprendizagem
- desaprendizagem
- plasticidade
- robustez
- tolerância a falhas

Com base nesta teoria e resultados encontrados nos etudos e publicações já citados, é proposto neste trabalho uma aplicação das Redes Neurais Artificiais Paraconsistentes em um Sistema Classificador de Sinais através de aproximações funcionais.

Todas as Células Neurais Artificiais Paraconsistentes são fundamentadas em equações matemáticas que possibilitam a sua implementação através de hardware e software. Conforme os estudos feitos em [94] as CNAP^s apresentam características capazes de modelar certas funções do neurônio biológico. Desta maneira, estas propriedades são difundidas para as Unidades e Redes Neurais Paraconsistentes.

CAPÍTULO 4

Aplicações com a Célula Neural Artificial Paraconsistente de Aprendizagem - CNAPa

CAPÍTULO 4

Aplicações com a Célula Neural Artificial Paraconsistente de Aprendizagem - CNAPa

4.1 Célula Neural Artificial Paraconsistente de Aprendizagem – CNAPa

Conforme exposto em [94], uma célula Artificial Paraconsistente de Aprendizagem – CNAPa é o nome dado a um algoritmo originado da descrição da Célula Neural Artificial Paraconsistente apresentada no capítulo 3, item 3.2. A seguir é apresentado o algoritmo e sua representação simbólica:

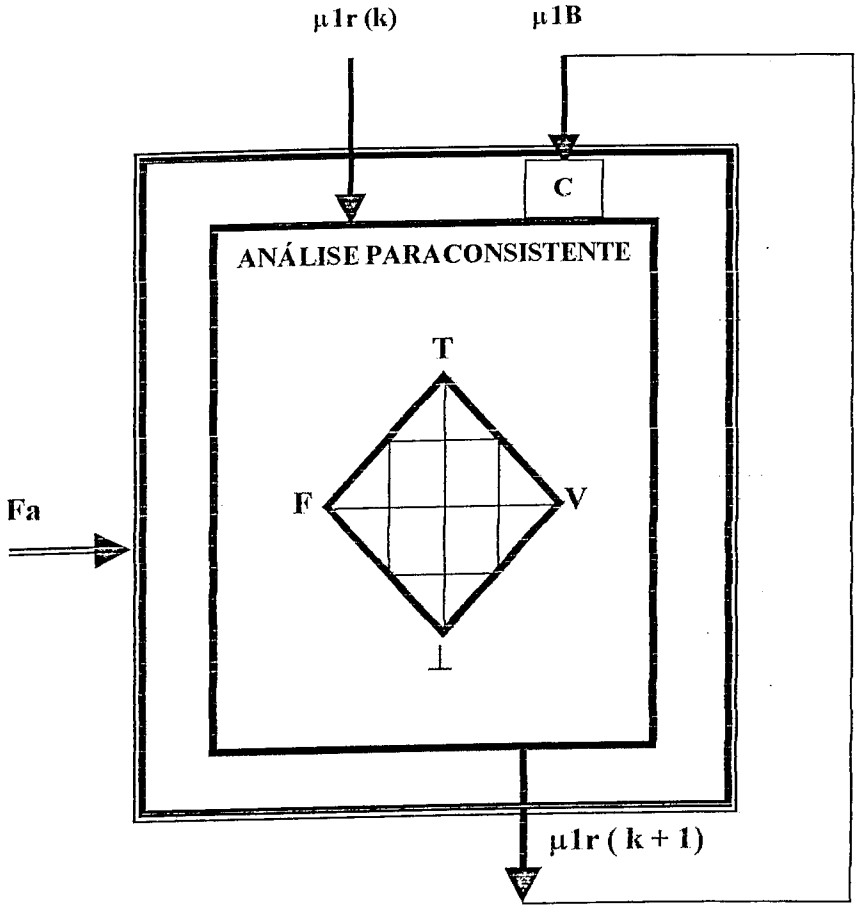


Fig. 4.1: Célula Neural Artificial Paraconsistente de Aprendizagem

Conforme estudos realizados em [94] as **CNAPa**[s] podem aprender qualquer valor real no intervalo fechado [0, 1] aplicado em sua entrada μ1r(k), como pode ser visto através da figura 4.2. O valor a ser "aprendido" é chamado de padrão **μ1r**, e é aplicado na entrada da célula. A saída da célula é realimentada à entrada complementada **C**. O valor inicial para a entrada complementada é 0.5, o que

equivale na LPA2v a um valor indefinido, portanto este será o valor usado na inicialização da célula de aprendizagem.

No processo de aprendizagem é introduzido um fator de aprendizagem – Fa, que é ajustado externamente, e que como pode ser visto nas equações do algoritmo, influencia em uma aprendizagem mais lenta ou mais rápida. Quando Fa = 1.0, a aprendizagem é otimizada.

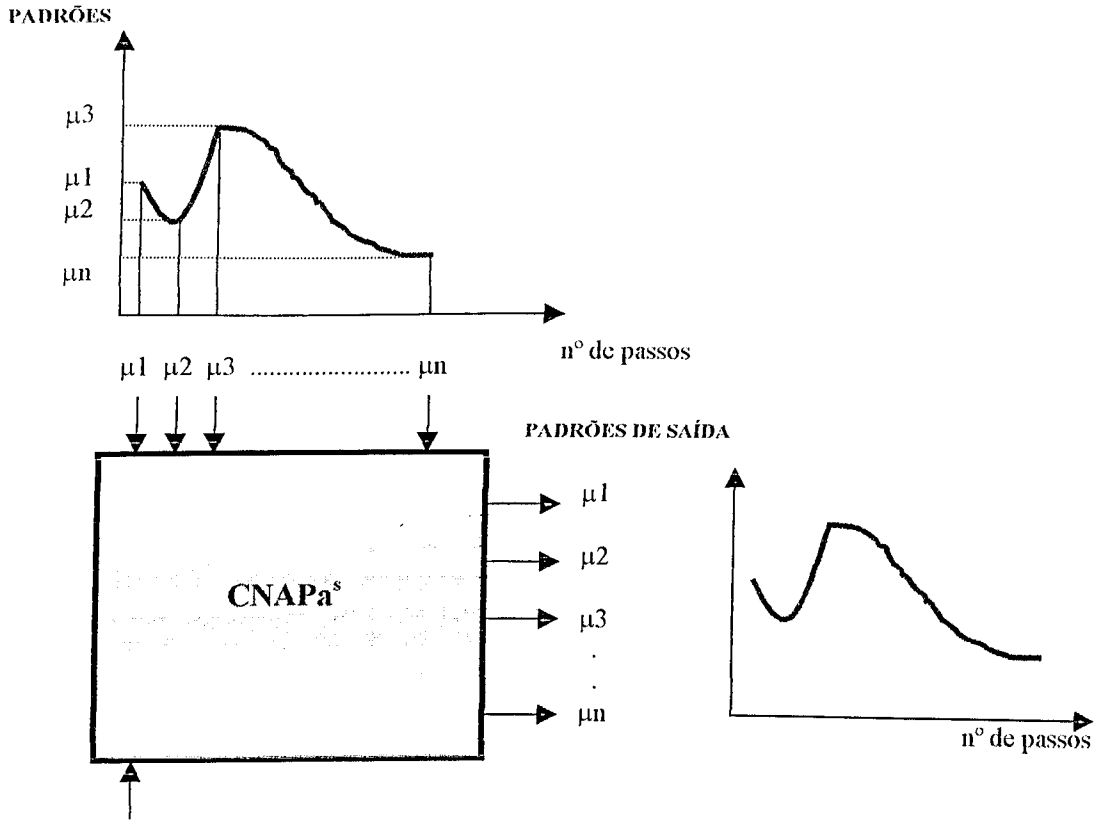
A equação de aprendizagem é definida como:

$$\mu 1r(k+1) = [\,(\mu 1r(k) - (1 - \mu 1B) \times Fa) + 1\,] / 2 \qquad (4.1)$$

Considera-se que o processo de aprendizado foi completado quando $\mu 1r(k+1) = \mu 1r(k)$.

4.2 Reprodução de um Sinal Discretizado a partir de uma Célula Neural Artificial Paraconsistente de Aprendizagem - CNAPa

Neste item será demonstrada a capacidade de aprendizado de uma CNAPa, onde a mesma reproduzirá um sinal elétrico discretizado no formato de matriz. Embora a rede permita tratar sistemas que contém inconsistências na sua análise, trata-se de uma aplicação onde os padrões de entrada da rede serão as amostras normalizadas entre os valores "0" e "1" do sinal a ser reproduzido (ver normalização no anexo 2). É possível demonstrar graficamente que à medida que o Fator de aprendizagem – **Fa** - da célula for tendendo para "1", que o sinal reproduzido na saída tende a se tornar uma cópia da entrada, como demonstrado no esquema abaixo:



Fator de Aprendizagem ⇒ 1

fig. 4.2: Esquema de reprodução de sinais através de CNAPa

Como ferramenta de software para demonstrar a aplicação, foi feita a programação utilizando o Delphi 4.0 e consequentemente a linguagem Object Pascal, além da ferramenta TeeChart do próprio Delphi.

4.3 Análise da aprendizagem de uma CNAPa em função do fator de aprendizagem Fa

Nessa seção será implementado algoritmo que permitirá o ensaio de uma CNAPa no aprendizado de funções discretizadas no formato de matriz, representando sinais. Os resultados irão definir o número de passos necessários para uma aprendizagem completa em função do valor do fator de aprendizagem – Fa. Os procedimentos e os resultados são expostos a seguir:
Considerando inicialmente o sinal abaixo cuja equação é:

$$\mathbf{sinal[\ i\ ] = (Sen\ (\ (i\ x\ Pi)\ /\ 180\ ) + 1)\ /\ 2} \qquad (4.2)$$

ou seja , um sinal senoidal normalizado com valores entre 0 e 1, no formato de matriz; este sinal mostrado na figura 4.3 será utilizado como entrada para os ensaios com a CNAPa:

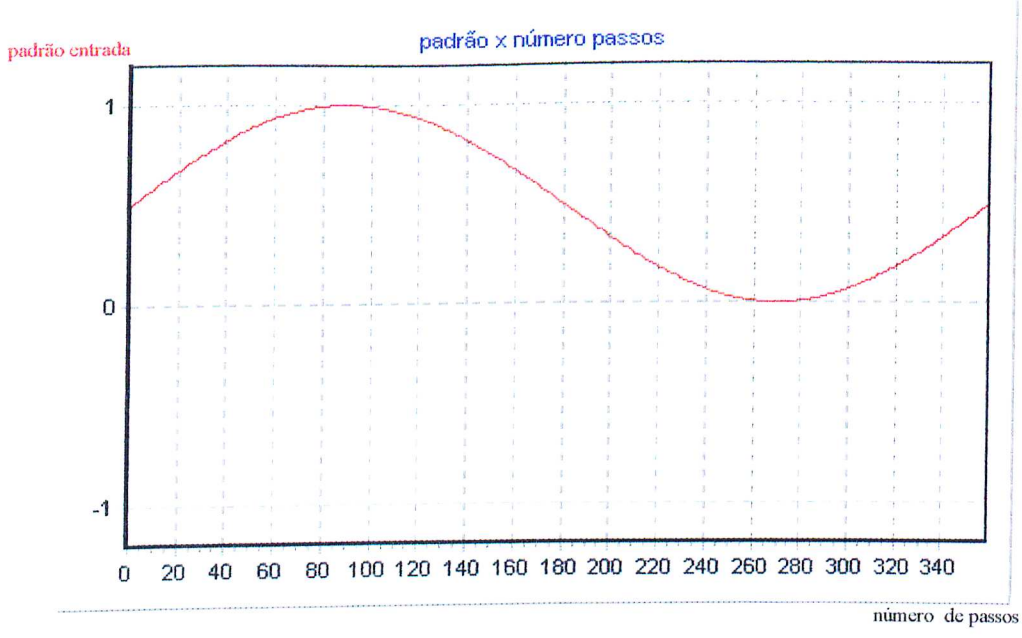


fig. 4.3: Sinal discretizado

Estabeleceu-se inicialmente neste ensaio que para testar a célula de aprendizagem, será utilizado um valor discreto do sinal, para i = 30; portanto, da equação 4.2 tem-se:

$$\mathbf{sinal[\ 30\ ] = (Sen\ (\ (i\ x\ Pi)\ /\ 180\ ) + 1)\ /\ 2 = 0.75}$$

O esquema da figura 4.4 representa de uma forma geral como será feita a análise da célula de aprendizagem:

ESQUEMA PARA ANÁLISE DA CÉLULA DE APRENDIZAGEM TENDO COMO ENTRADA O VALOR DE UM PADRÃO

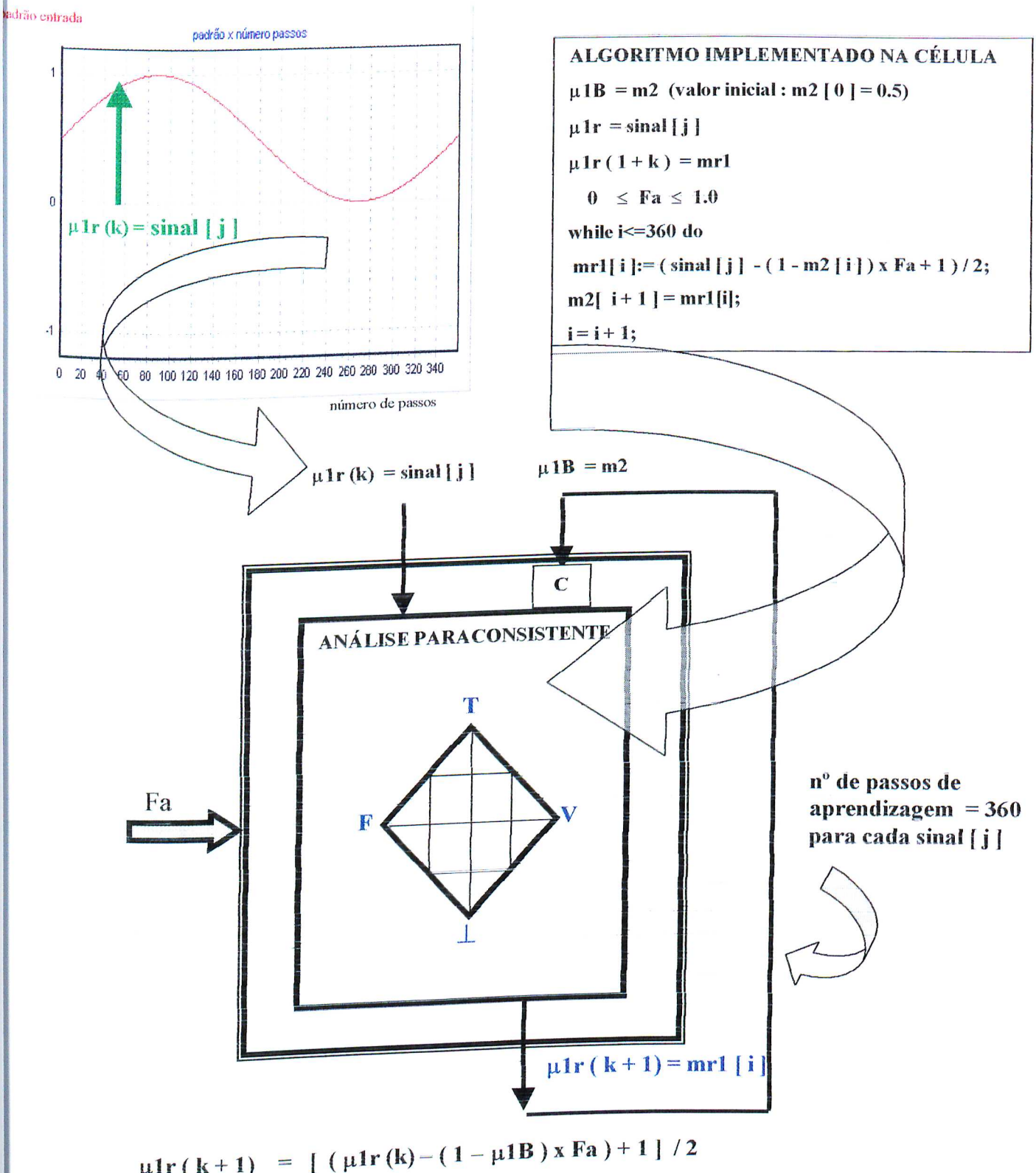


fig.4. 4: Esquema para análise do comportamento da Célula Neural Artificial Paraconsistente de Aprendizagem

4.3.1 Análise para Fa = 0.5

Considerando inicialmente Fa = 0.5 e o valor de inicial de μ1B = 0.5, aplica-se a equação de aprendizagem através do algoritmo abaixo:

μ1B = m2 (valor inicial : m2 [0] = 0.5)

μ1r = sinal [30] = 0.75

μ1r (1 + k) = mr1

Fa = 0.5

while i<=360 do

mr1[i]:= (sinal [30] - (1 - m2 [i]) x Fa + 1) / 2; (5.3)

m2[i + 1] = mr1[i];

i = i + 1;

A intersecção do valor de mr1 (padrão aprendido) com a curva original em um universo de 360 passos é mostrado a seguir:

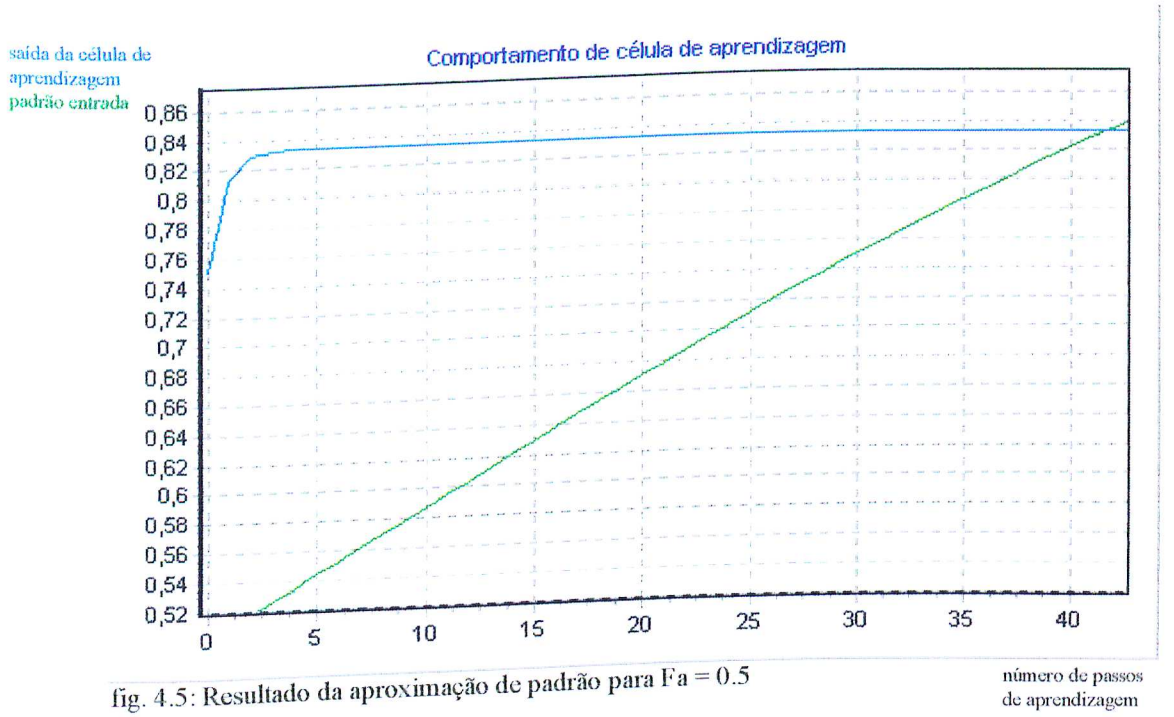


fig. 4.5: Resultado da aproximação de padrão para Fa = 0.5

Observa-se que **mr1 ~ 083** não estabiliza no valor desejado do padrão (0.75) com Fa = 0.5.

Repetindo a simulação para i = 90:

sinal[90] = (Sen ((i x Pi) / 180) + 1) / 2 = 1.0

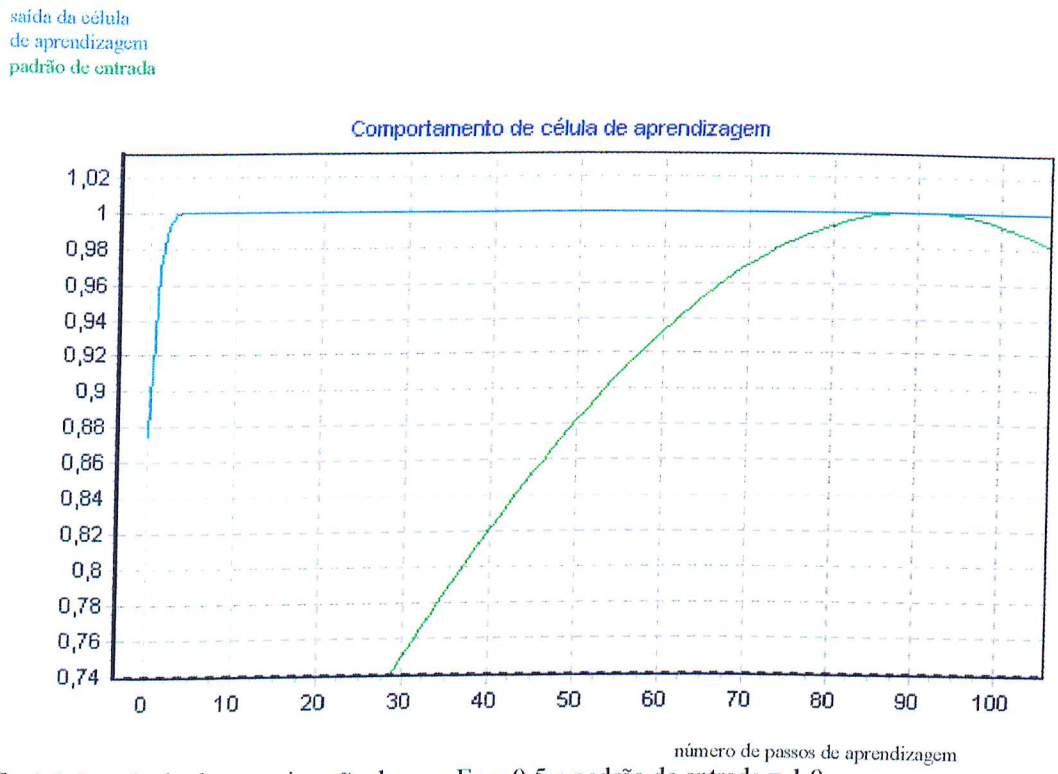


fig 4.6: Resultado da aproximação de para Fa = 0.5 e padrão de entrada = 1.0

Percebe-se que quando o padrão de entrada é 1.0, mesmo com fator de ajuste não maximizado, a célula aprende o padrão.

Expandindo a mesma figura, observa-se que a célula aprende o padrão em 06 passos:

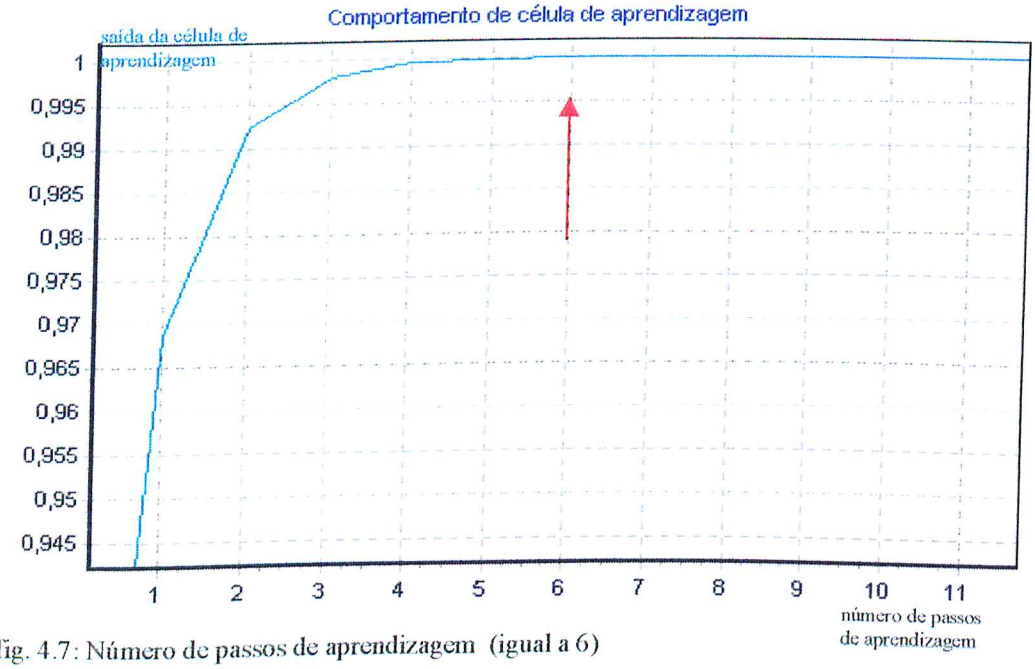


fig. 4.7: Número de passos de aprendizagem (igual a 6)

Estes resultados estão de acordo com os valores demonstrados no trabalho "Fundamentos das Redes Neurais Artificiais Paraconsistentes" [94].

4.3.2 Análise para Fa = 0.8

Verificando ainda a dependência da aprendizagem com o fator de aprendizagem, é feita agora nova análise com Fa = 0.8, e padrão de entrada igual a sinal [30]:

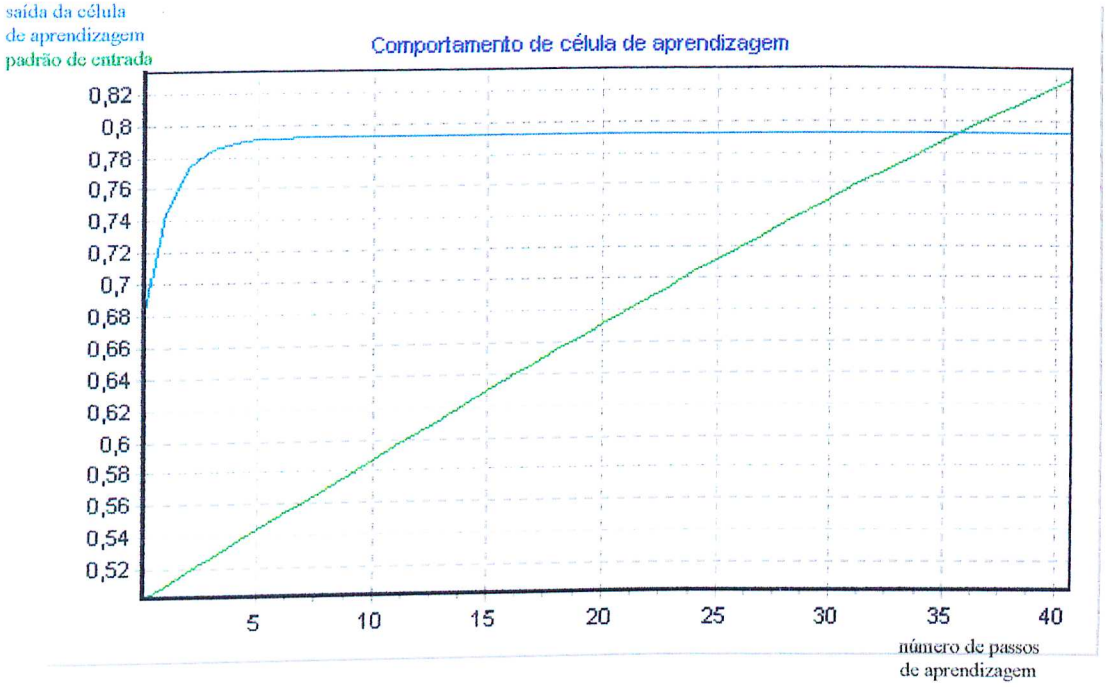


fig. 4.8: Resultado da aproximação de padrão para Fa = 0.8

Novamente **mr1 ~ 0.79** também não atinge o valor 0.75, porém a diferença do valor esperado é menor do que aquele apresentado para Fa = 0.5.

Simulando agora para **sinal [90] = 1.0** - (Fa = 0.8):

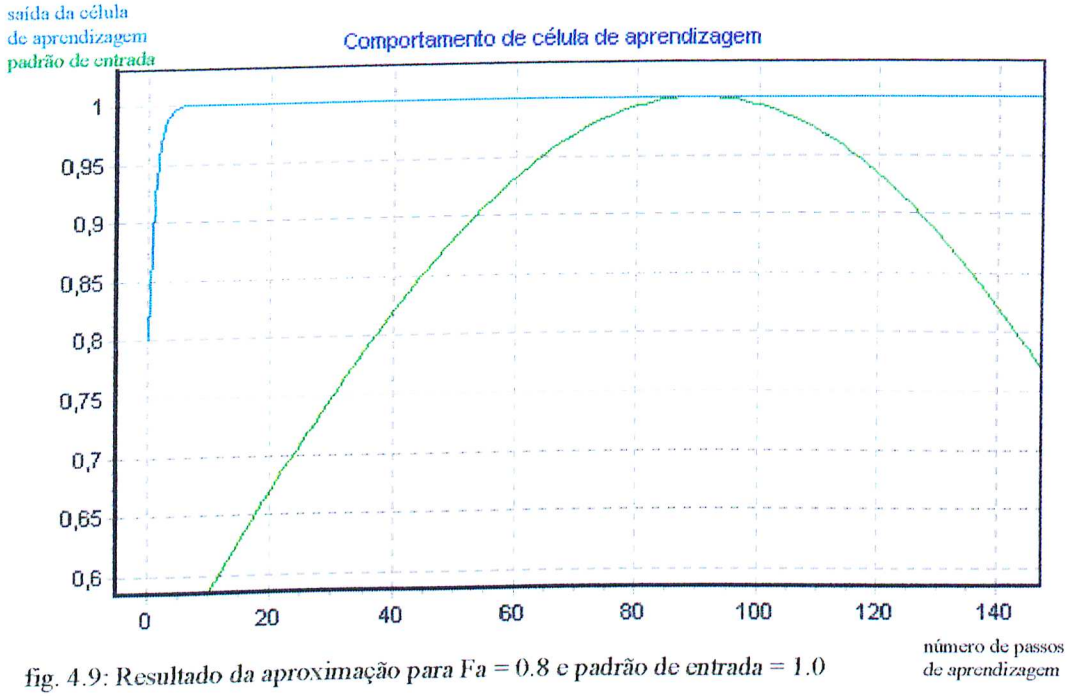


fig. 4.9: Resultado da aproximação para Fa = 0.8 e padrão de entrada = 1.0

A célula aprende o padrão em 08 passos de acordo com a figura expandida:

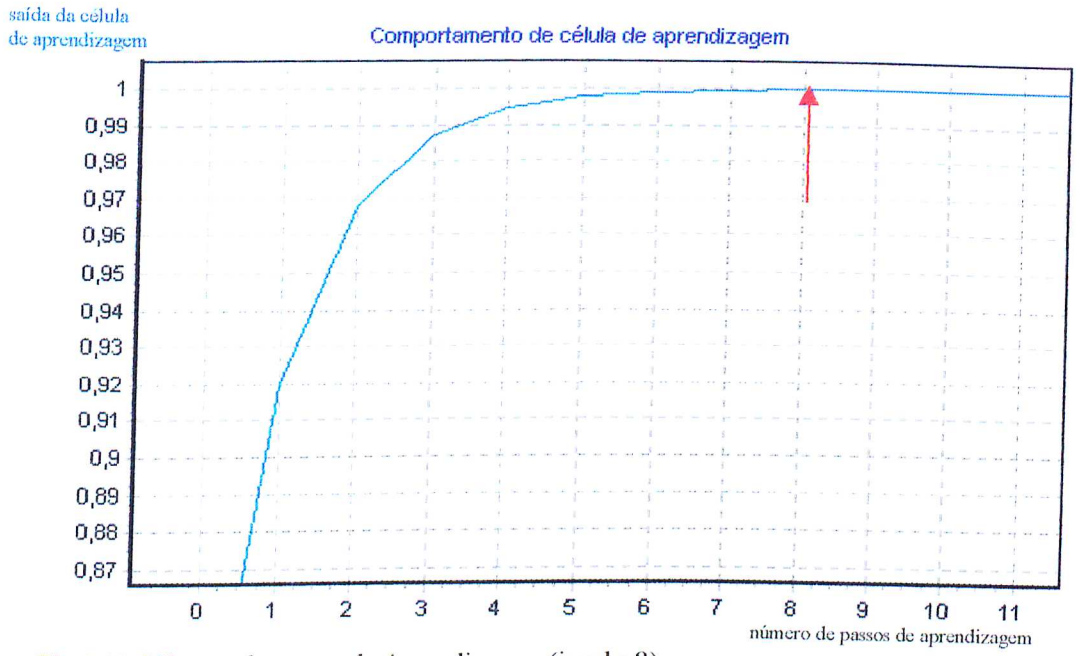


Fig 4.10: Número de passos de Aprendizagem (igual a 8)

4.3.3 Análise para Fa = 1.0

O ensaio agora é feito com o valor do fator de aprendizagem máximo Fa = 1.0.

Considerando i = 30: **sinal[30] = (Sen ((i x Pi) / 180) + 1) / 2 = 0.75**

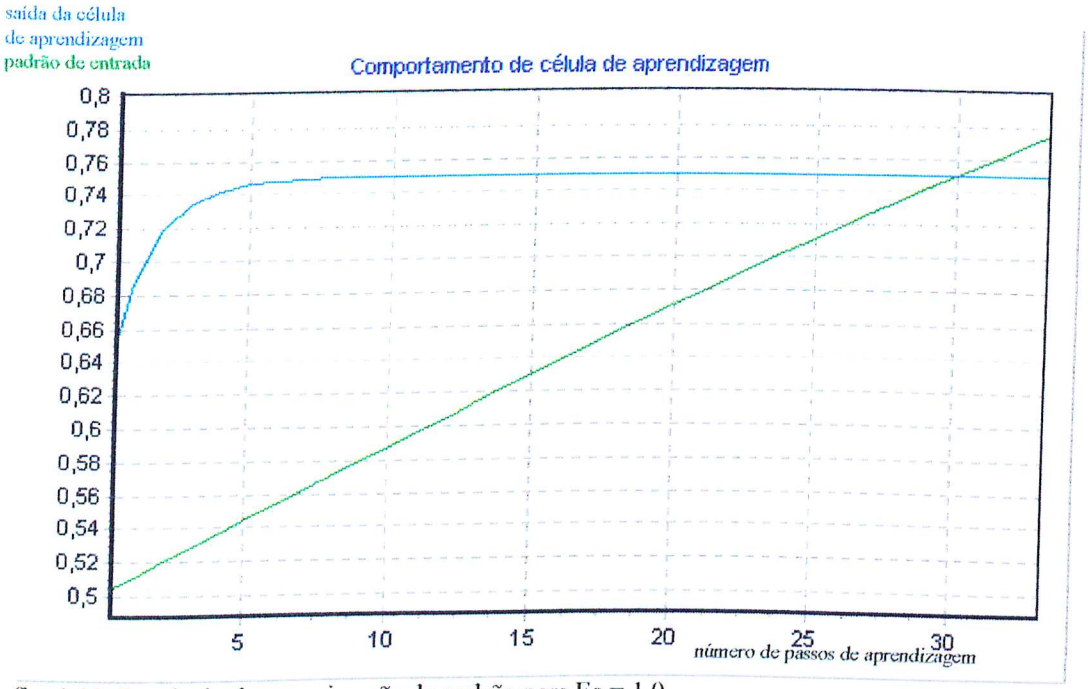


fig. 4.11: Resultado da aproximação de padrão para Fa = 1.0

No gráfico da figura 4.11 vê-se que quando Fa = 1.0, a célula aprendeu o padrão, mesmo que este seja diferente de 1.0. A quantidade de passos requerida para esta aproximação é 15, como pode ser visto na figura expandida:

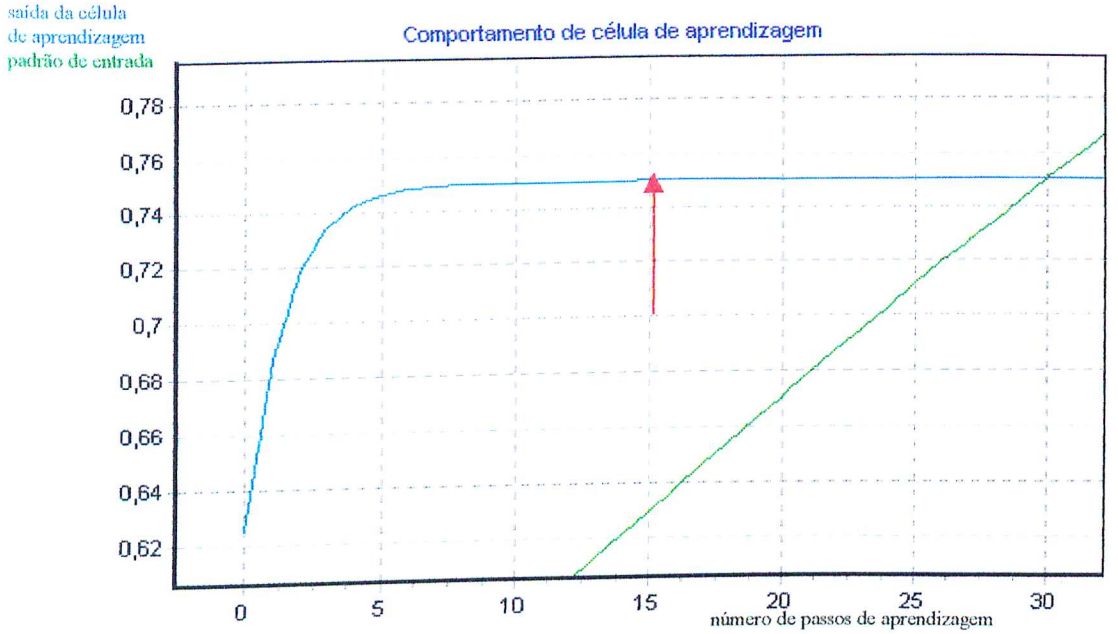


fig. 4.12: Número de passos de aprendizagem (igual a 15)

Repetindo o experimento para i = 90: **sinal [90] = 1.0.**

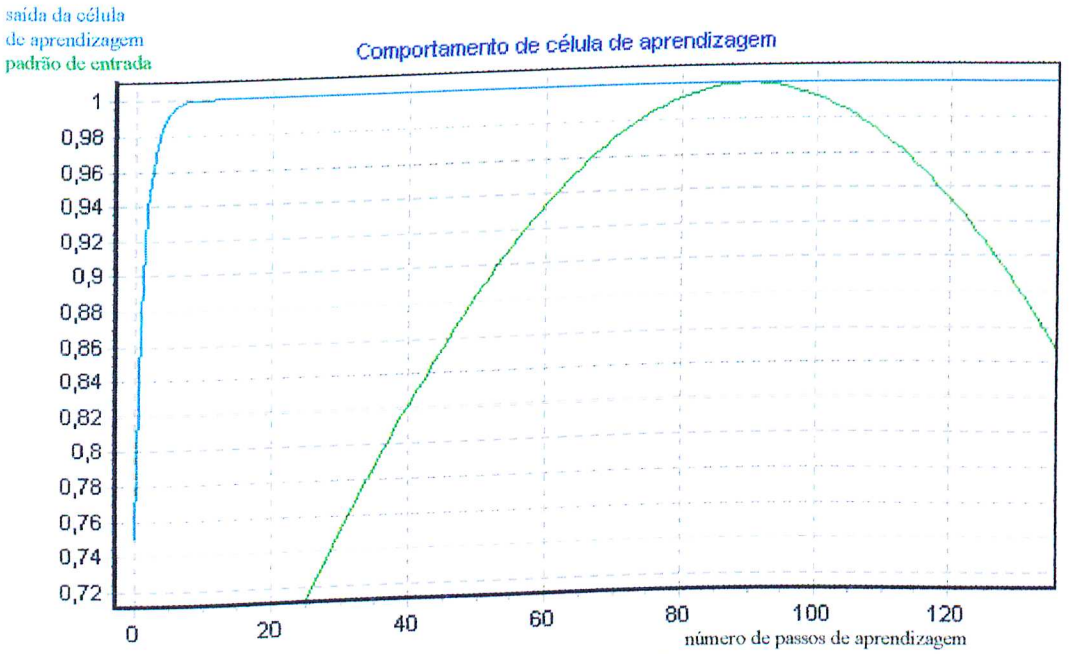


fig. 4.13: Resultado para Fa = 1.0 e padrão de entrada = 1.0.

Expandindo a figura, resulta que foram necessários 12 passos para o aprendizado:

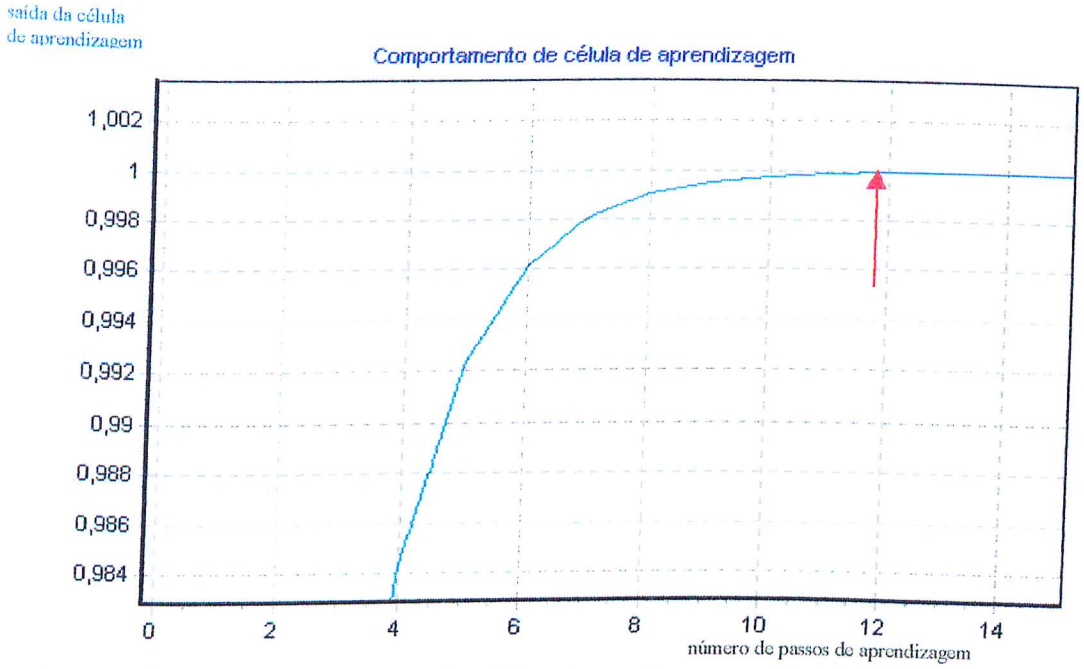


fig. 4.14: Número de passos de aprendizagem (igual a 12)

4.3.4 Conclusão sobre a dependência da aprendizagem com Fa

Este ensaio permitiu obter o comportamento de uma Célula Neural Artificial Paraconsistente de Aprendizagem em várias situações.

A partir dos resultados obtidos, conclui-se que quanto menor o fator de aprendizagem – Fa, menos passos são requeridos para se chegar à estabilização de um padrão na saída de uma célula de aprendizagem.

Verificou-se também que quanto mais próximo de 1.0 o valor de Fa, o padrão estabilizado na saída mais se aproxima do padrão aplicado na entrada.

Os valores obtidos e mostrados nos gráficos demonstram que a eficácia da célula de aprendizagem é maior quando o padrão de entrada se aproxima de 1.0.

4.4 Utilização da Célula de Aprendizagem para reprodução de sinais

A partir dos resultados obtitos em 4.3, nesta seção serão utilizados os conceitos da célula de aprendizagem com o objetivo de reproduzir um sinal discretizado na forma de matriz. Inicialmente serão utilizados os mesmos padrões do ensaio anterior, portanto a análise será feita a partir da equação 4.2, que dá origem ao sinal da figura 4.15:

$$\textbf{sinal[i]} = \textbf{(Sen ((i x Pi) / 180) + 1) / 2} \qquad (4.2)$$

com $0 \leq i \leq 360$;

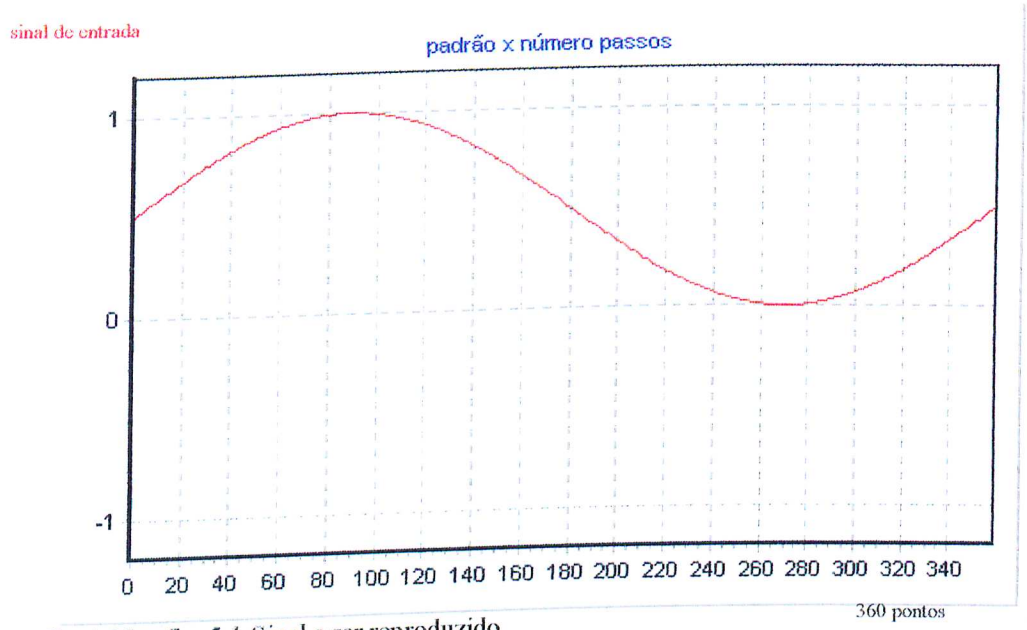


fig. 4.15 = fig. 5.1:Sinal a ser reproduzido

Considerando os mesmos valores iniciais:

μ1B = m2 (valor inicial : m2 [0] = 0.5)

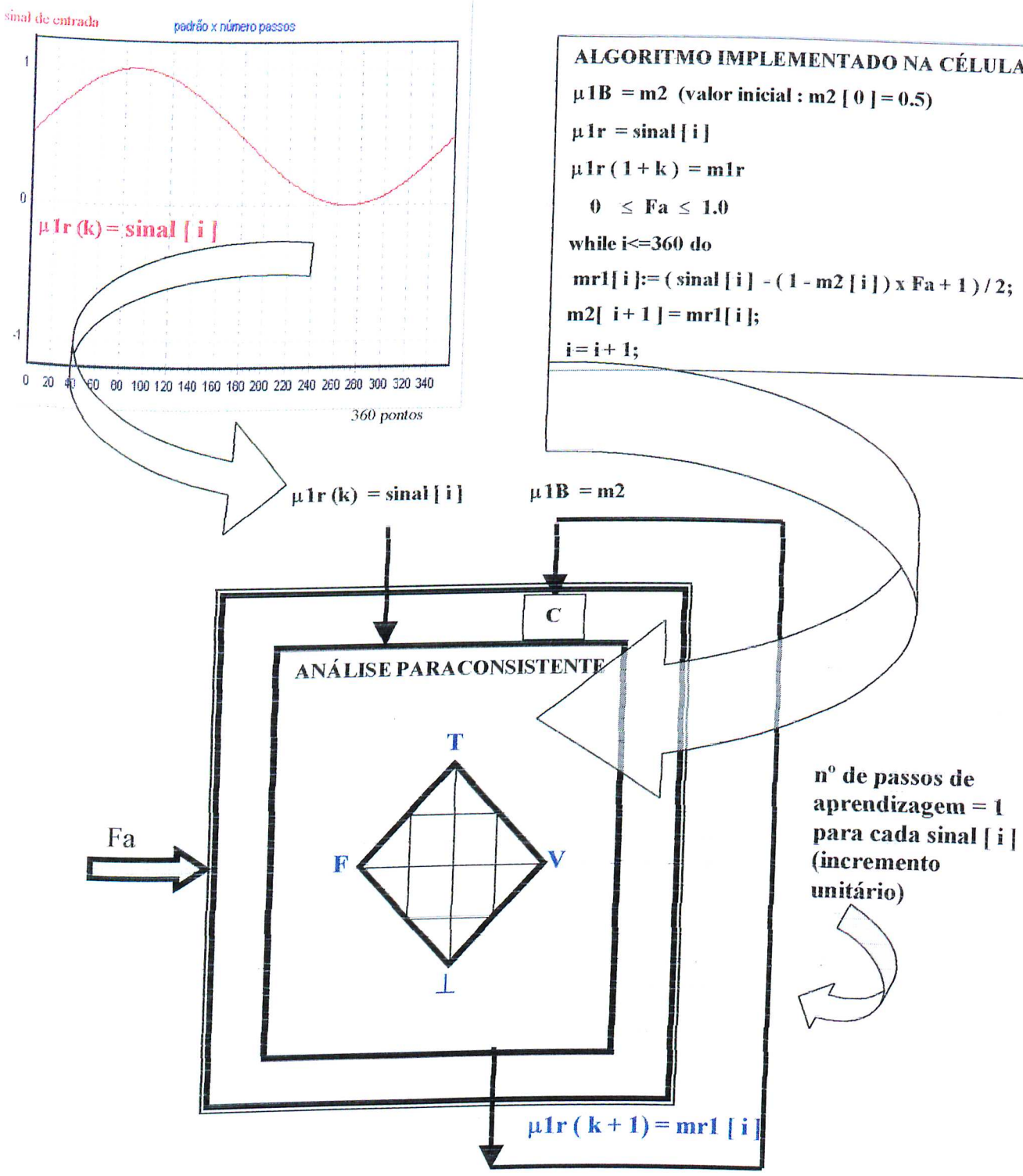
μ1r = sinal [i]

μ1r (1 + k) = m1r

while i <= 360 do

mr1[i] = (sinal [i] - (1 - m2 [i]) x Fa + 1) / 2;
m2[i + 1] = mr1[i];
i = i + 1;

ESQUEMA PARA REPRODUÇÃO DE SINAL COMPLETO (VÁRIOS PADRÕES) COM A CÉLULA DE APRENDIZAGEM



$$\mu 1r(k+1) = [\ (\mu 1r(k) - (1 - \mu 1B) \times Fa) + 1\ ] / 2$$

fig.4. 16: Esquema para reprodução de sinal usando a Célula Neural Artificial Paraconsistente de Aprendizagem

4.4.1 Reprodução do sinal para Fa = 0.5

Aplicando o sinal da equação 4.2 foi obtido resultado conforme mostrado na figura 4.17:

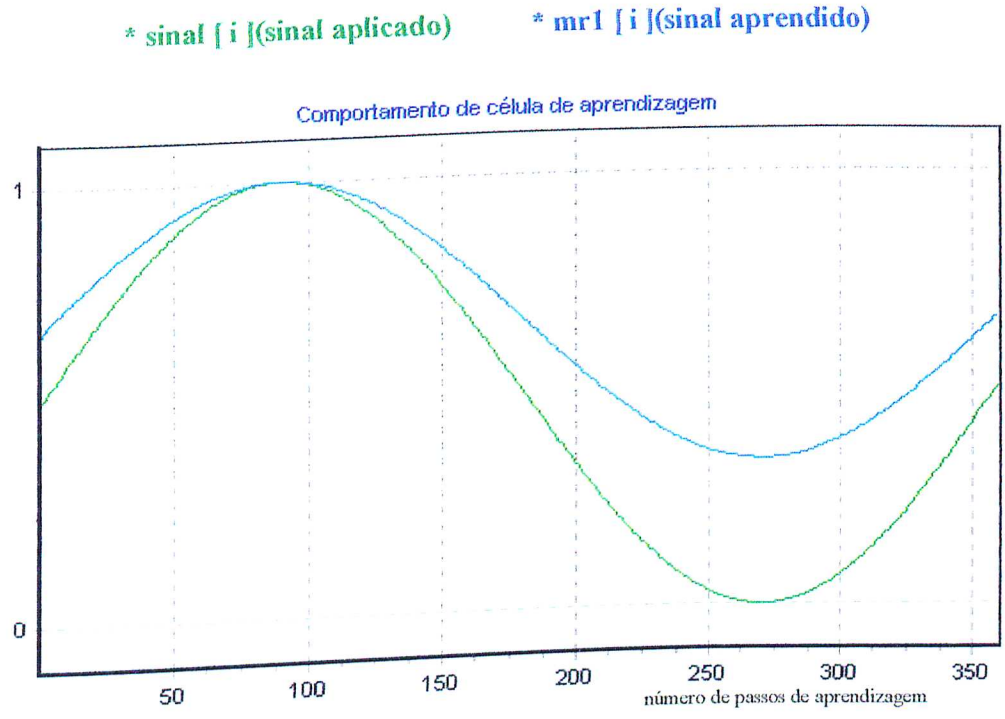


fig. 4.17: Reprodução de sinal com Fa = 0.5

Observa-se que, de acordo com o que foi concluído em 4.3.4, a reprodução mr1 [i] se aproxima do padrão sinal [i] para valores próximos de 1, e distancia-se quando os valores estão próximos de zero.

4.4.2 Reprodução do sinal para Fa = 0.8

A figura 4.18 mostra os resultados para um fator de aprendizagem agora aumentado para Fa = 0.8. Os resultados têm as mesmas características anteriores (fig. 4.17), porém percebe-se que a diferença entre o sinal padrão e a reprodução é menor:

*** sinal [i] (sinal aplicado)** *** mr1 [i] (sinal aprendido)**

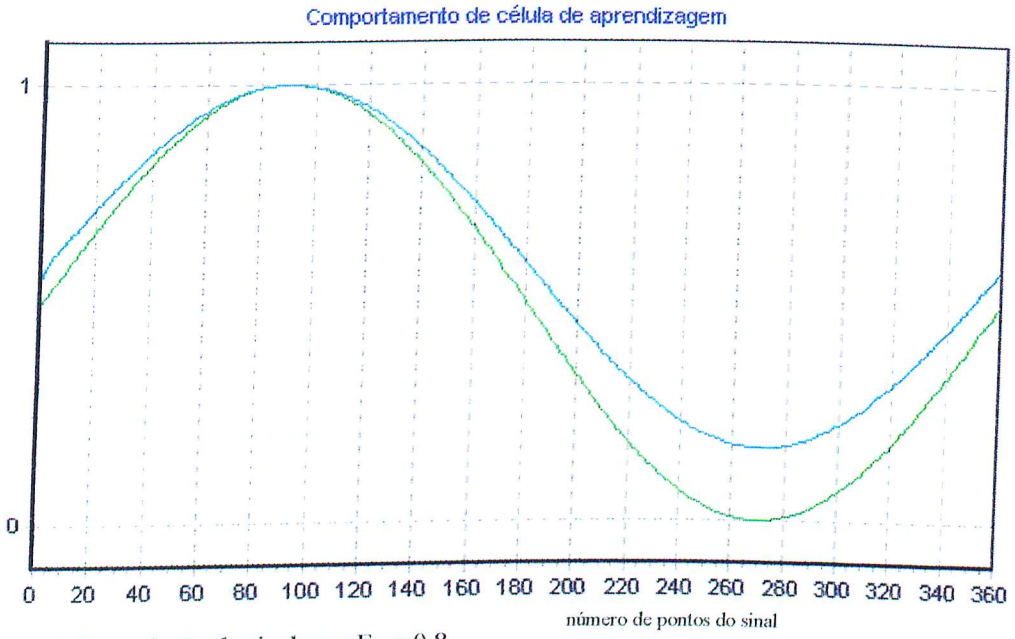


fig. 4.18: Reprodução de sinal para Fa = 0.8

4.4.3 Reprodução do Sinal para Fa = 1.0

O fator de aprendizagem é aumentado para Fa = 1.0 e feito novo ensaio. Verifica-se do gráfico que neste caso são obtidos os melhores resultados, com a reprodução muito aproximada do sinal. Os resultados estão mostrados na figura 4.19:

*** sinal [i] (sinal aplicado)** *** mr1 [i] (sinal aprendido)**

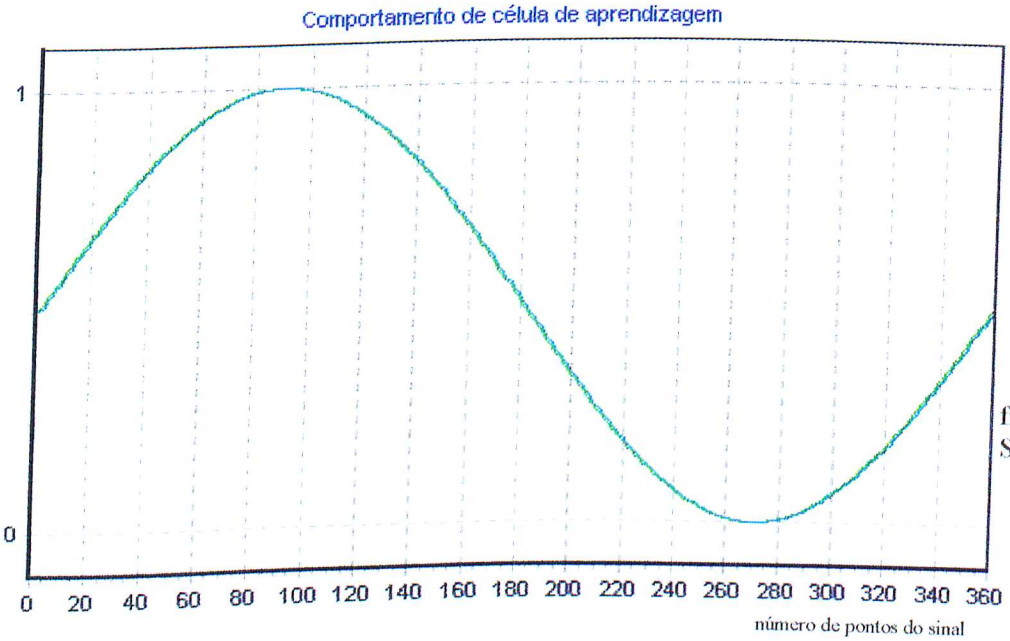


fig. 4.19: Reprodução de Sinal para Fa = 1.0

Expandindo a figura, nota-se que ainda persistem as diferenças, notóriamente para valores distantes de 1.0, porém a discrepância entre o sinal e a reprodução é mínima:

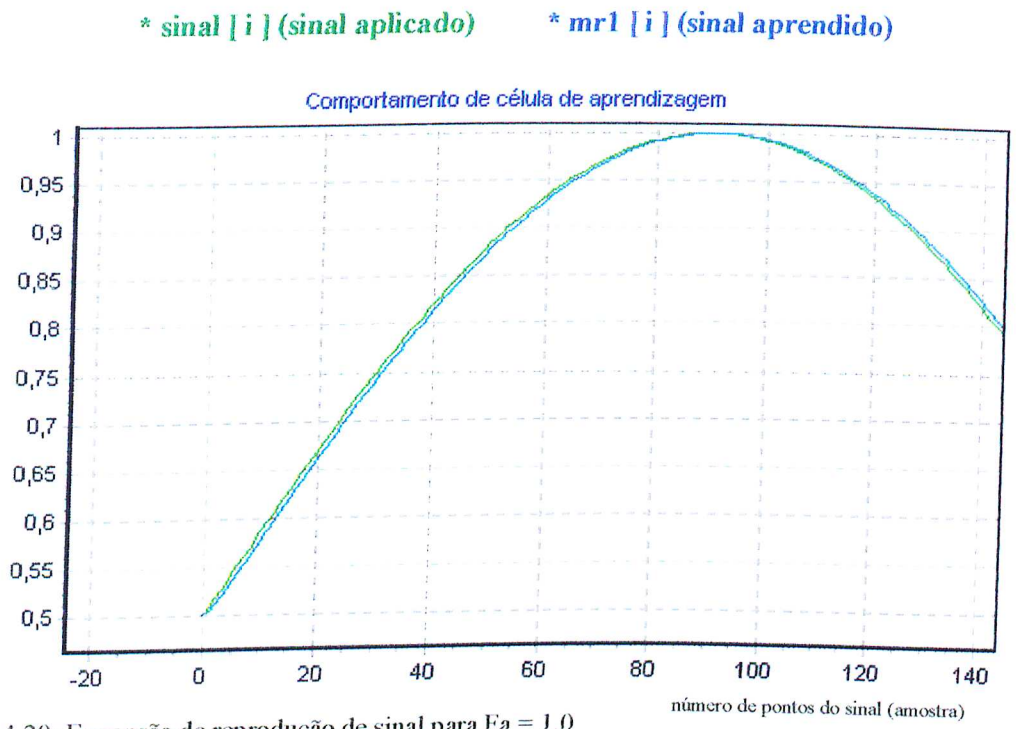


fig. 4.20: Expansão de reprodução de sinal para Fa = 1.0

Nas próximas figuras, para melhor visualização serão feitas novas expansões nos sinais, para melhor percepção das diferenças.

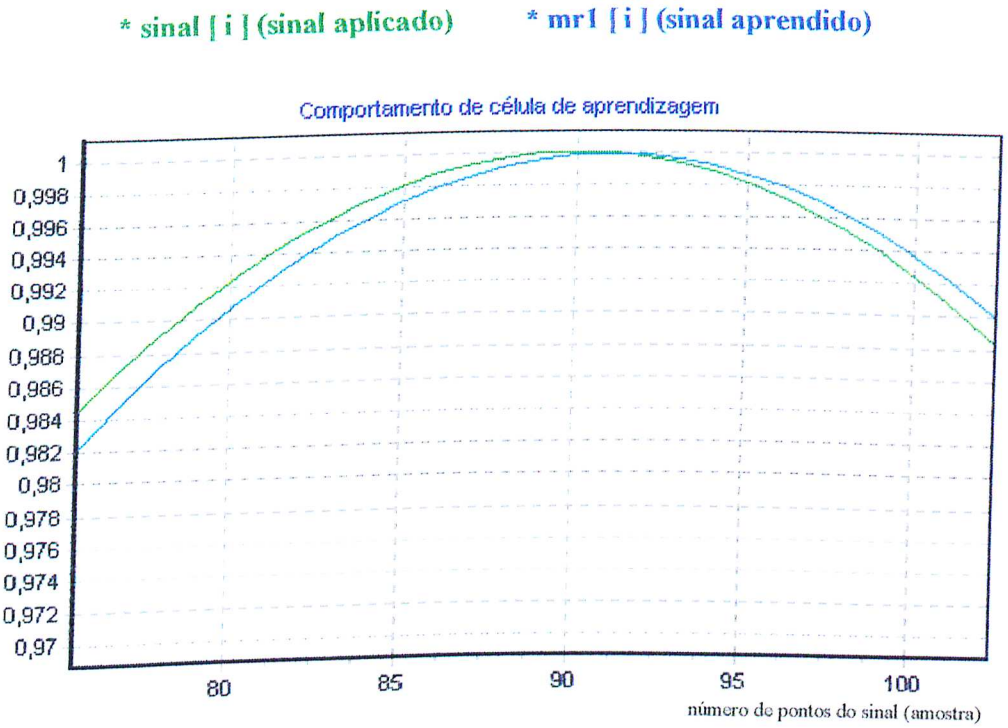


fig. 4.21: Expansão de reprodução de sinal para Fa = 1.0

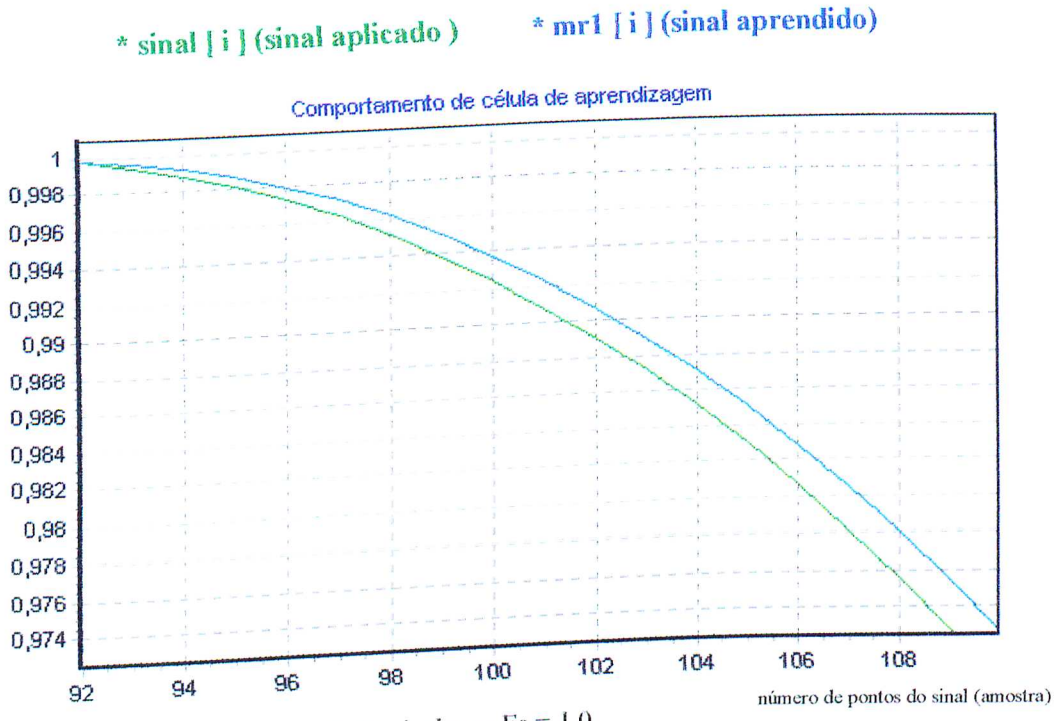


fig. 4.22: Expansão de reprodução de sinal para Fa = 1.0

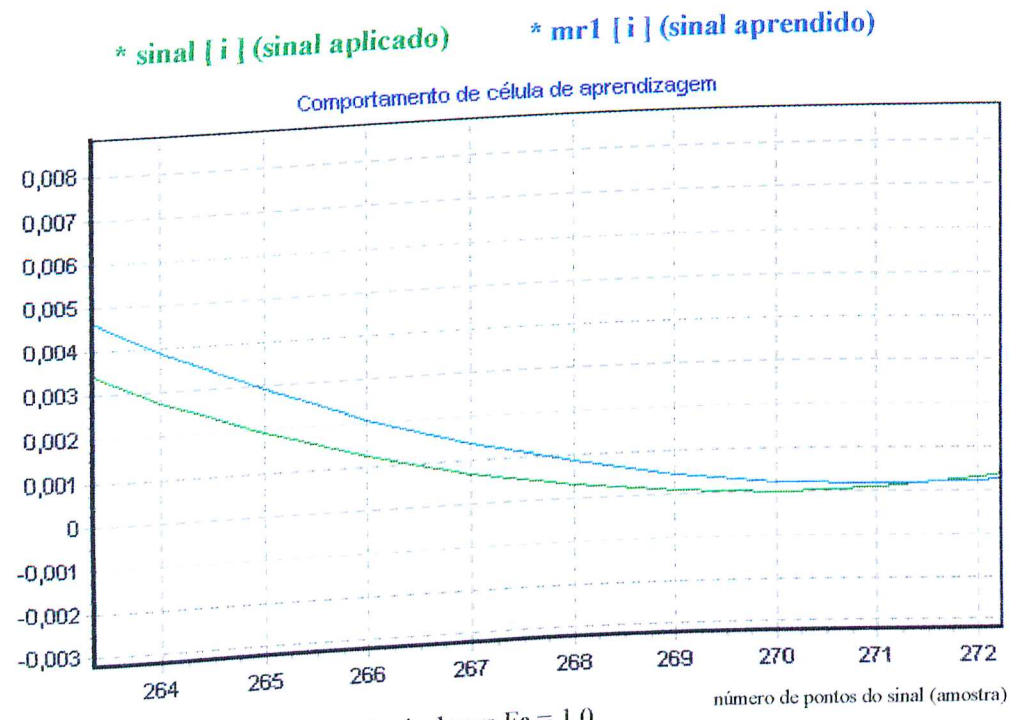


fig. 4.23: Expansão de reprodução de sinal para Fa = 1.0

4.4.4 Reprodução de Sinal Modulado

O ensaio de aprendizagem da CNAPa é feito agora com um sinal modulado. O sinal a ser reproduzido agora é do tipo:

sinal [i] = 0.5 + (Sen ((20 x i x Pi) / 180)) x (Cos ((i x Pi) / 180)) / 2 (4.4)

Através de um loop a partir de i foi construído o sinal:

for i = 0 to 360 do

sinal [i] = 0.5 + (Sen ((20 x i x Pi) / 180)) x (Cos ((i x Pi) / 180)) / 2

Sinal a ser reproduzido:

while i <= 360 do
mr1[i] = (sinal [i] - (1 - m2 [i]) x Fa + 1) / 2;
m2[i + 1] = mr1[i];
i = i + 1;

4.4.4.1 Fa = 0.8

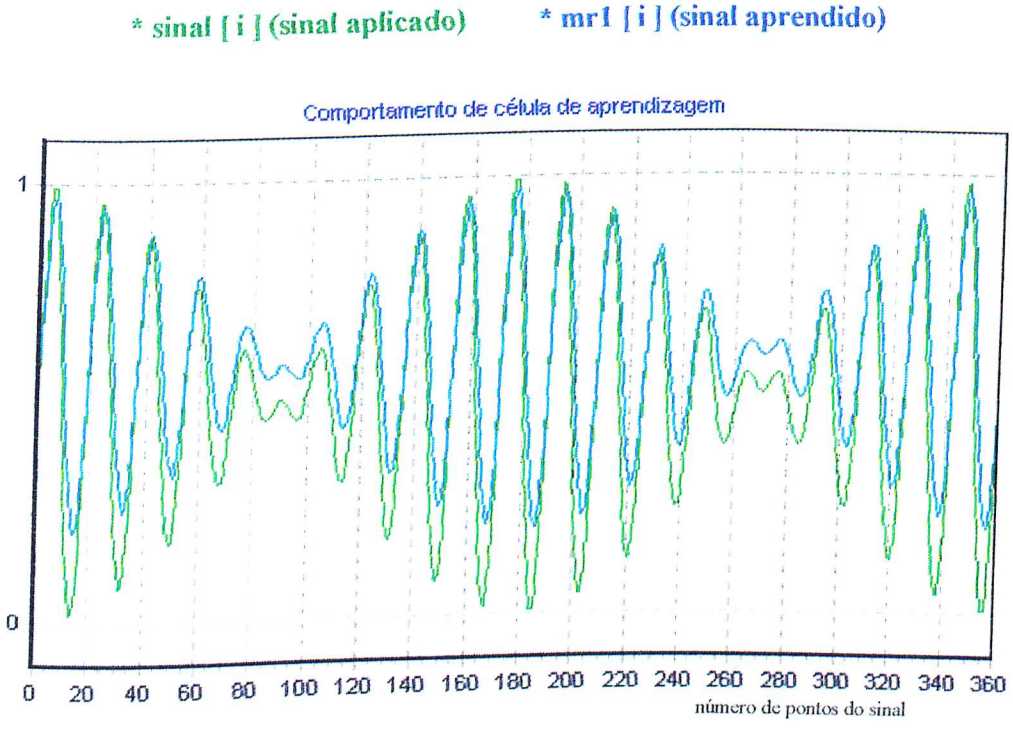


fig. 4.24: Reprodução de sinal modulado com Fa = 0.8

4.4.4.2 Fa = 1.0

Para o sinal modulado, quando o fator de aprendizagem – Fa = 1.0, o sinal é reproduzido na saída com diferenças mínimas.

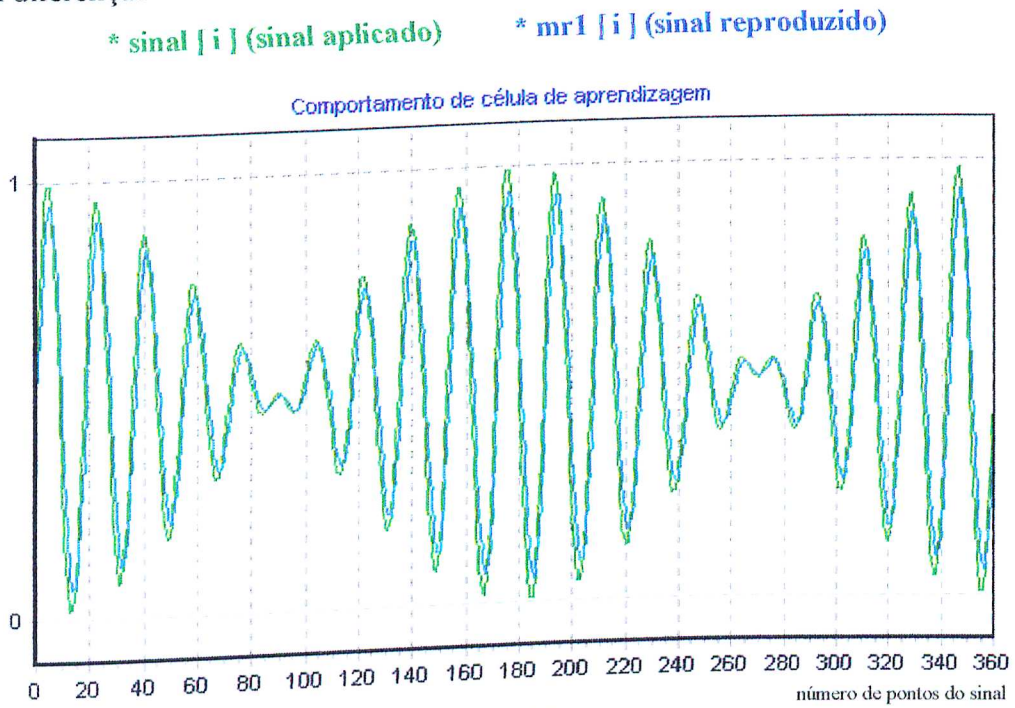


fig. 4.25: Reprodução de sinal modulado com Fa = 1.0

4.4.5 Reprodução de Sinal Randômico – Fa = 1.0

Neste caso o ensaio foi feito com um sinal randômico gerado pelo próprio software. Verifica-se na figura 4.26 que o sinal aprendido tende para os valores do sinal randômico original, sem no entanto, conseguir uma aproximação melhor devido à natureza aleatória do mesmo.

*** sinal randômico : sinal[i]:=Random(360)/400;** (4.5) *** sinal reproduzido**

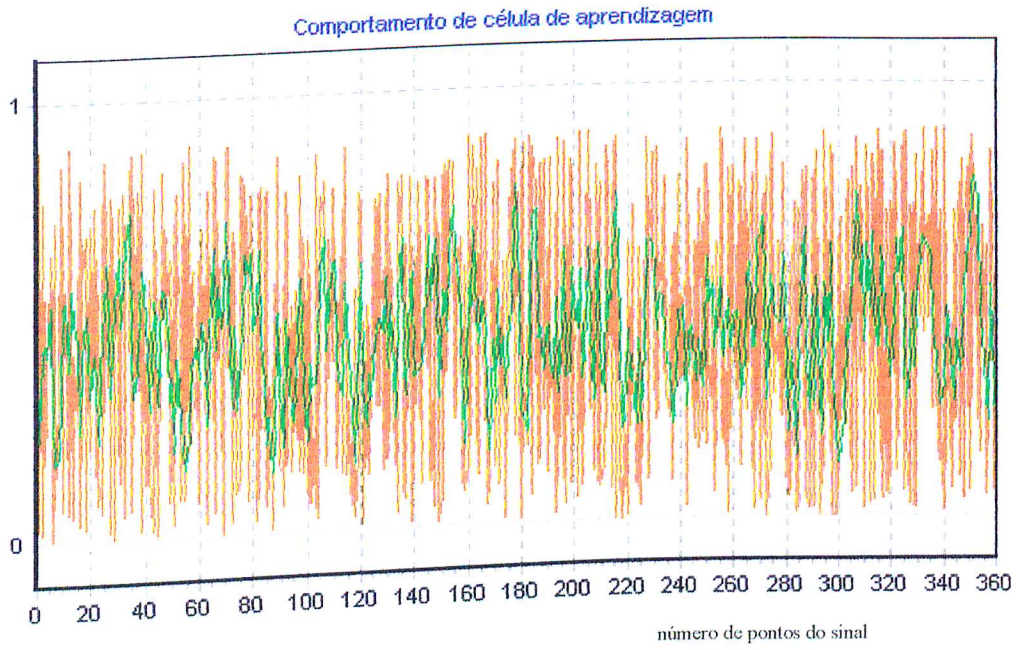


fig. 4.26: Reprodução de sinal randômico

*** sinal randômico : sinal[i]:=Random(360)/400;** (4.5) *** sinal reproduzido**

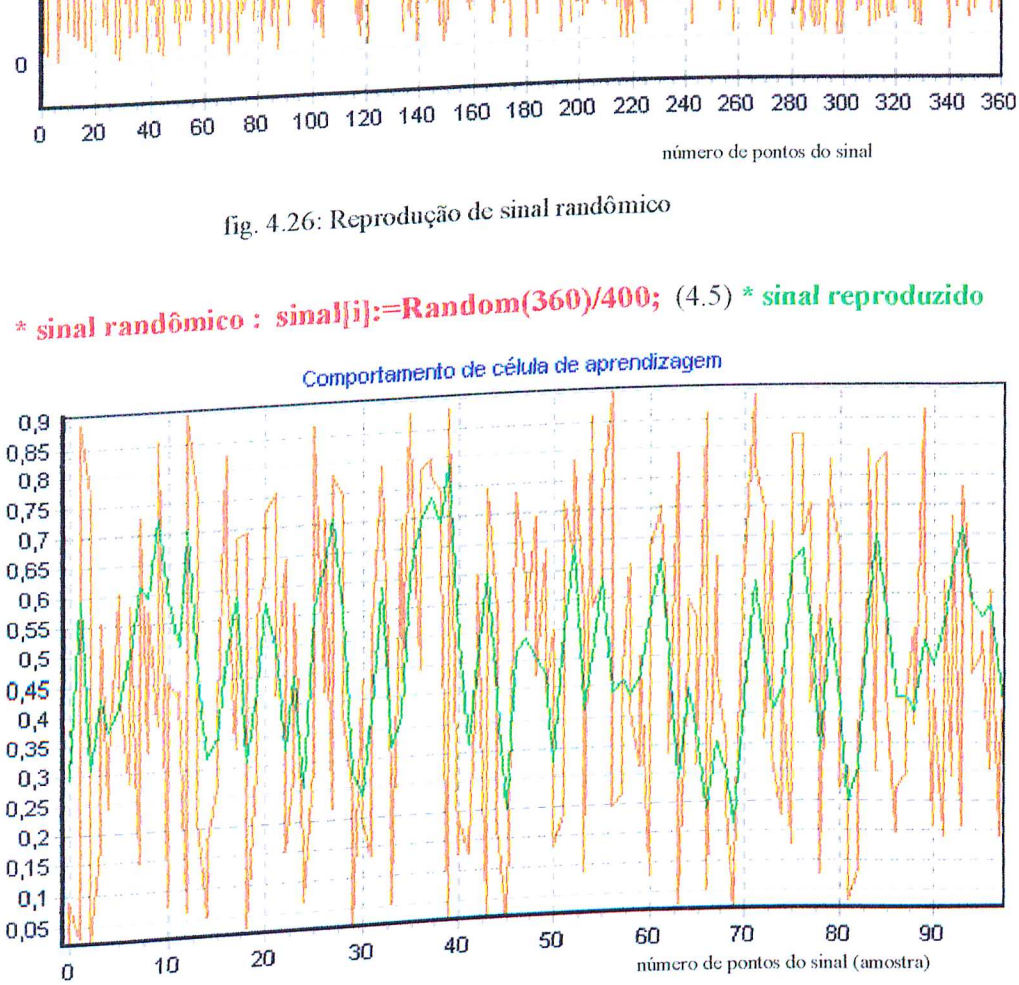


fig. 4.27: Reprodução de sinal randômico (amostra)

4.6 Considerações sobre a reprodução de sinais

Considerando os resultados obtidos nas simulações efetuadas neste capítulo, verificou-se que para a reprodução de sinais o fator de aprendizagem Fa deve ser 1; nestas condições, mesmo quando o padrão de entrada passa por somente um passo de aprendizagem, o resultado final de todo o sinal aprendido é satisfatório, principalmente quando o sinal é formado por características periódicas.

CAPÍTULO 5

Reprodução de Sinais através da Célula de Aprendizagem utilizando Aproximação Funcional

CAPÍTULO 5

Reprodução de Sinais através da Célula de Aprendizagem utilizando Aproximação Funcional

5.1 Técnica de Reprodução de Sinais e/ou Aproximação Funcional através de CNAPa:

No capítulo 04 a reprodução dos sinais foi baseada simplesmente na inserção de cada valor discretizado "sinal [i] " que compõe a matriz do sinal na equação da célula de aprendizagem. Naquele caso o valor de mr [i] de saída, na realidade, corresponde a apenas um passo de aproximação com o sinal [i , como demonstra o esquema da figura 5.1:

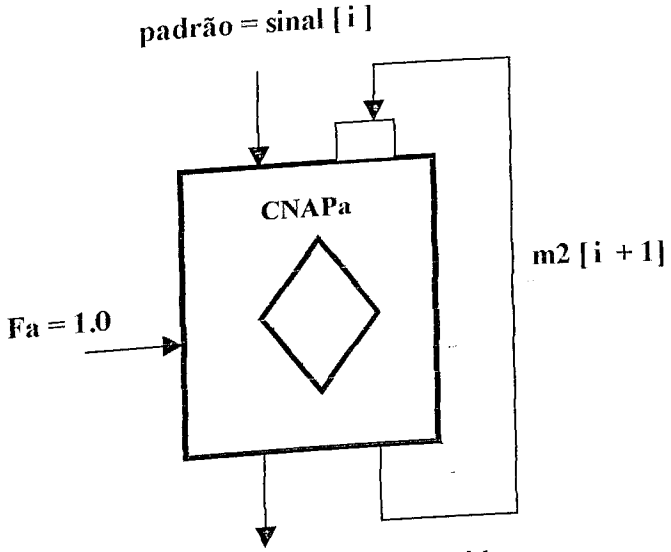


fig 5.1: Célula Neural Artificial Paraconsistente de Aprendizagem na reprodução de sinal

Verifica-se que o sinal reproduzido na saída é decorrente da somatória dos pontos **i** para sómente 01 passo. Embora os resultados para aproximação de sinais já fossem plenamente satistatórios, não estava sendo utilizado o conceito de realimentar a célula com **m2** até que a saída se igualasse à entrada. Será utilizada agora esta metodologia que estabelece uma aproximação funcional para a aprendizagem. De acordo com os resultados obtidos em 4.3.3, vimos que para Fa = 1.0 e padrão diferente de 1.0, o número de passos para que a saída se iguale ao padrão de entrada é 15.
A partir daí, a técnica utilizada para reprodução de sinais é treinar a saída com o mínimo de 20 passos, o que garantiria igualar com a entrada, e só a partir daí incrementar o valor de i, como demonstra fluxo da figura 5.2:

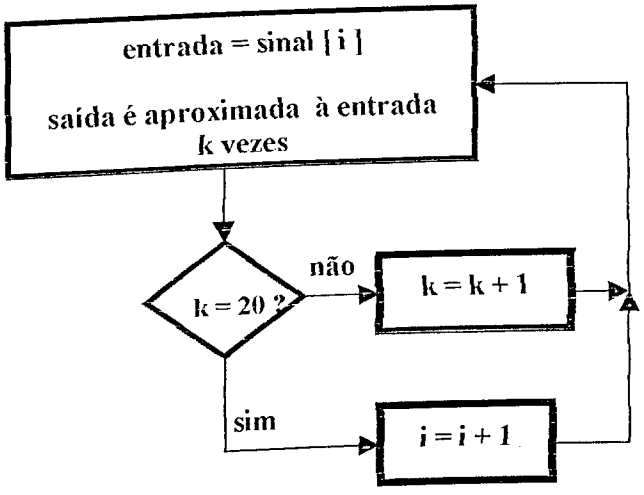


fig. 5.2: Lógica do algoritmo de reprodução de sinais

As linhas de código que geram o padrão de entrada para a análise de aproximação funcional é exposto abaixo:

```
i: Integer;
begin
for i = 0 to 360 do
begin
sinal [ i ]= (Sin(( i * Pi )/180 )+1) / 2;
end;
end;
```

O sinal padrão gerado pela função é:

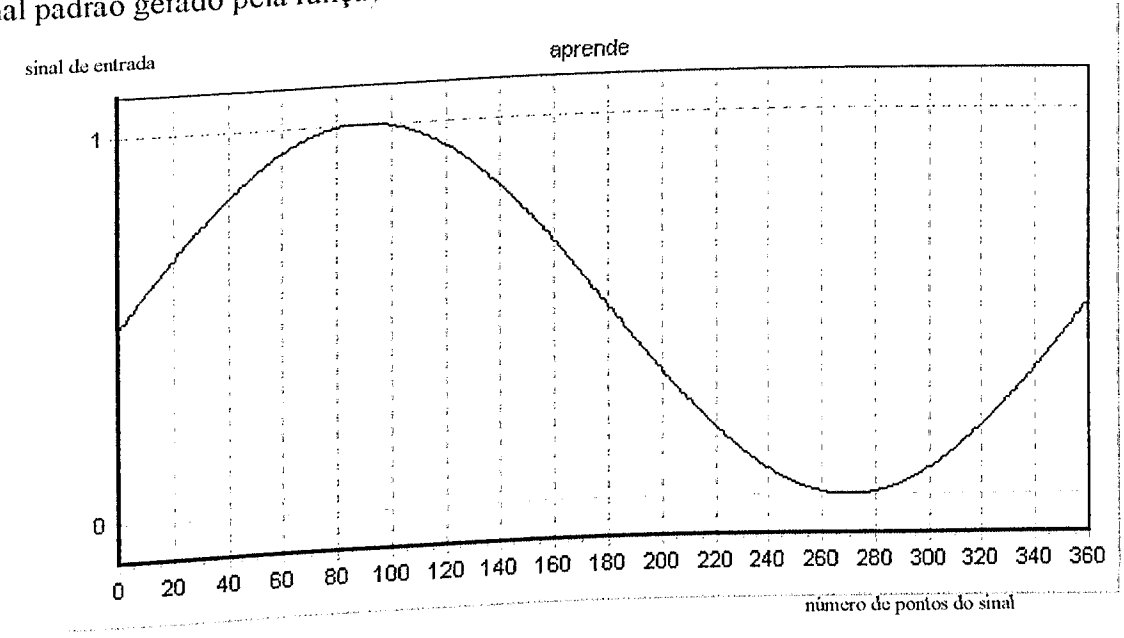


fig. 5.3: Sinal padrão de entrada

As linhas de código que reproduzem o esquema da figura 5.2 são descritas abaixo :

```
Fa = 1.0;
m2[0] = 0.5;
i = 0;

while i <=360 do
 begin
 k = 0;

  while k <= 20 do
    begin
    mr1[k] = ( sinal[i] - (1 - m2[k] ) * Fa + 1 ) / 2;
    m2[k+1] = mr1[k];
    k = k+1;
   end;

k=20;
mr2[i] = mr1[k];
i = i+1;

end;
```

O esquema que mostra o fluxo do ensaio de uma CNAPa para aproximação funcional é mostrado a seguir:

ESQUEMA PARA APROXIMAÇÃO FUNCIONAL USANDO CÉLULA DE APRENDIZAGEM

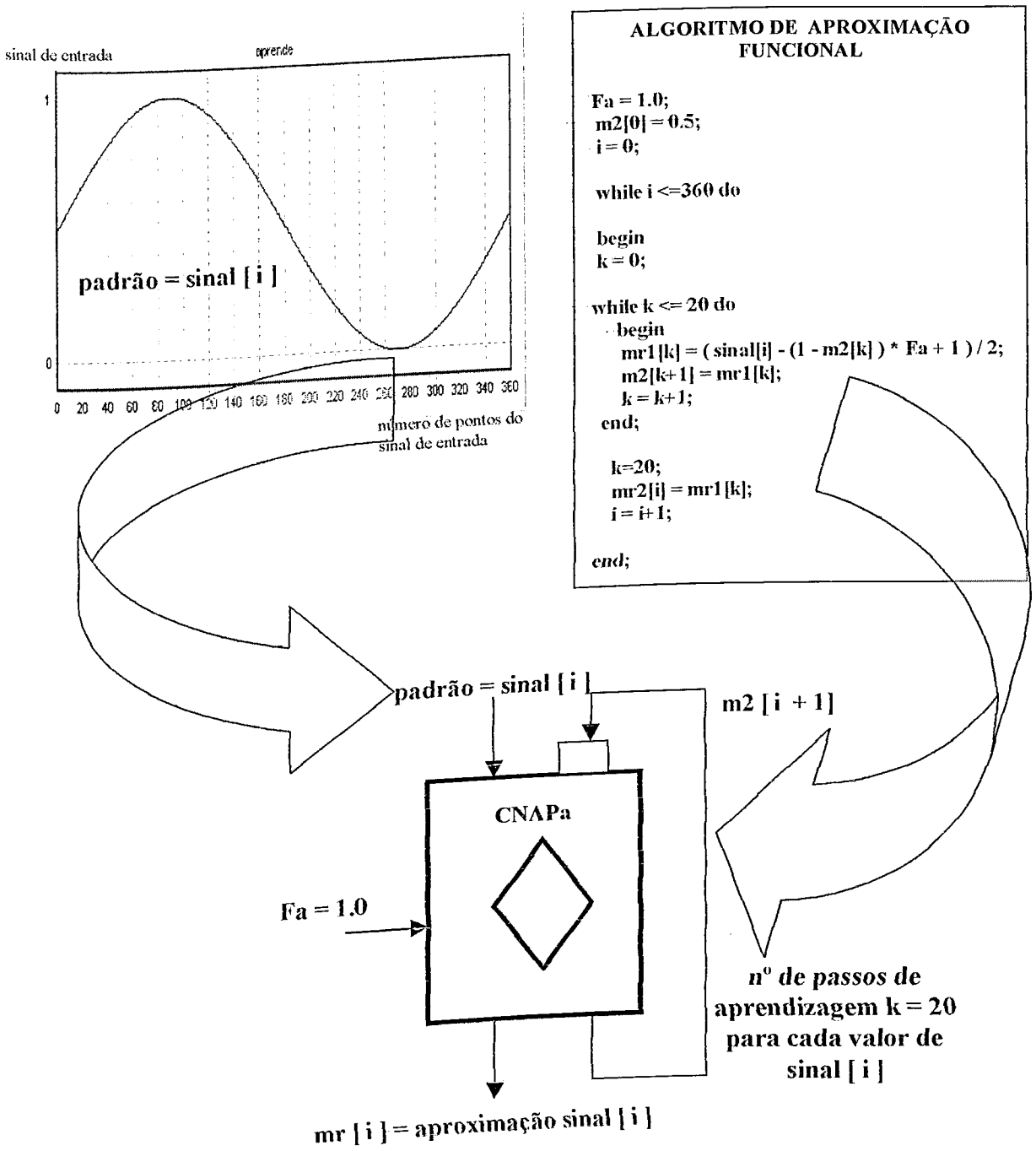


fig 5.4: Célula Neural Artificial Paraconsistente de Aprendizagem usada para aproximação funcional

O sinal de saída gerado quando é utilizada a técnica de aproximação funcional é mostrado na figura 5.5. Observa-se uma reprodução perfeita do padrão de entrada, onde não se distingue a princípio o sinal aprendido do sinal aplicado.

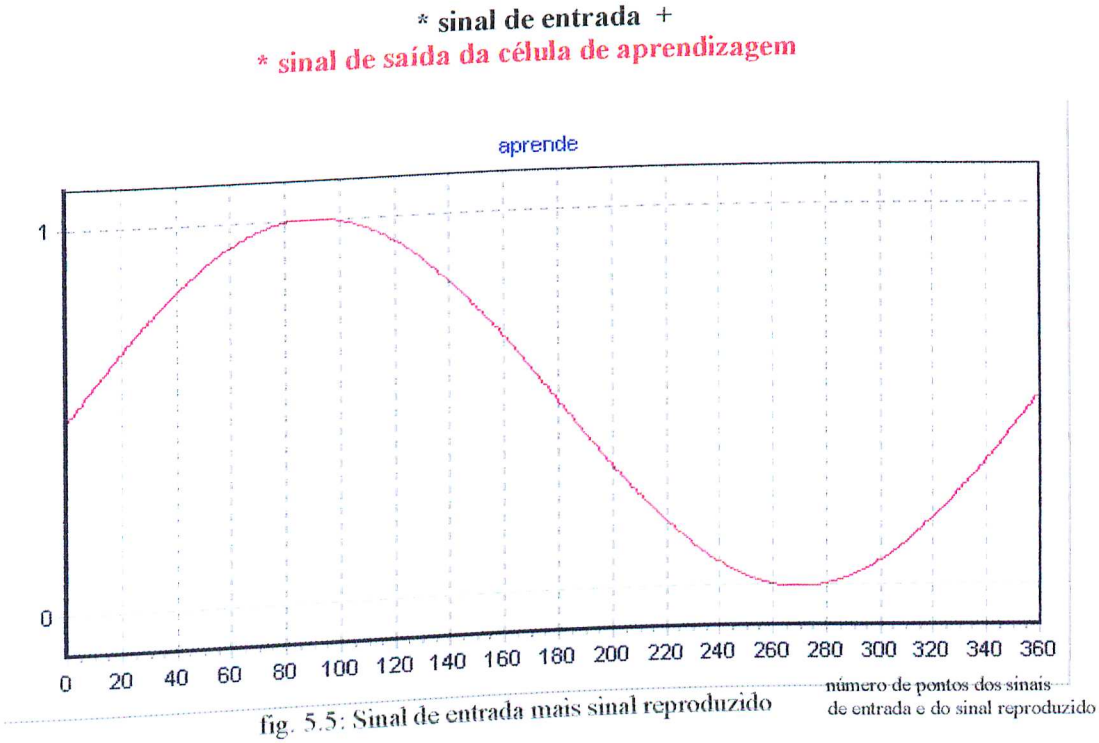


fig. 5.5: Sinal de entrada mais sinal reproduzido

Uma ampliação da figura 5.5 nos mostra um grau de aproximação de 100% entre a saída e a entrada:

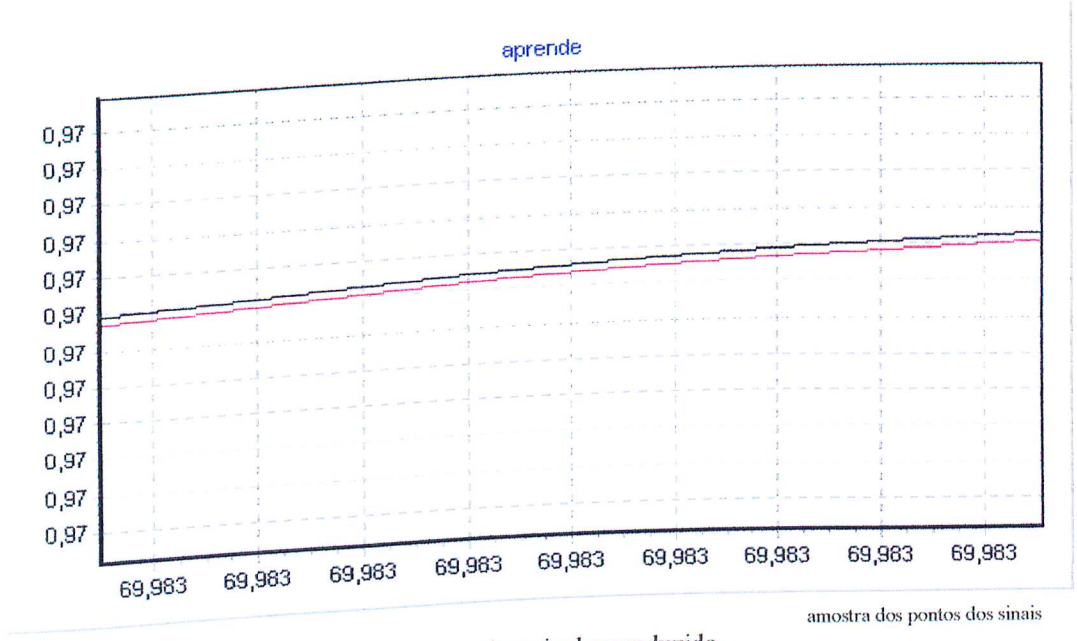


fig. 5.6: Expansão do sinal de entrada mais o sinal reproduzido

A mesma análise com aproximação funcional é feita agora para o sinal de entrada do tipo:

sinal[i]:= 0.5 + (Sin((20*i*Pi)/180))*(Cos((i*Pi)/180))/2; (5.1)

*** sinal de entrada**

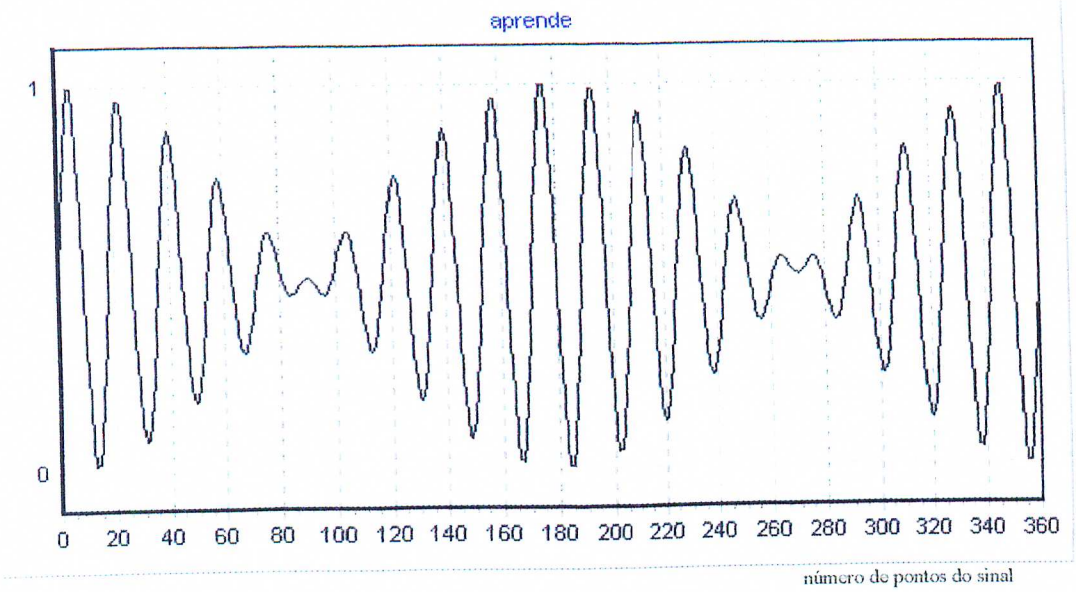


fig. 5.7: Sinal padrão de entrada modulado

O resultado, novamente com um grau de aproximação de 100% é mostrado na figura 5.8:

*** sinal de entrada + * sinal de saída**

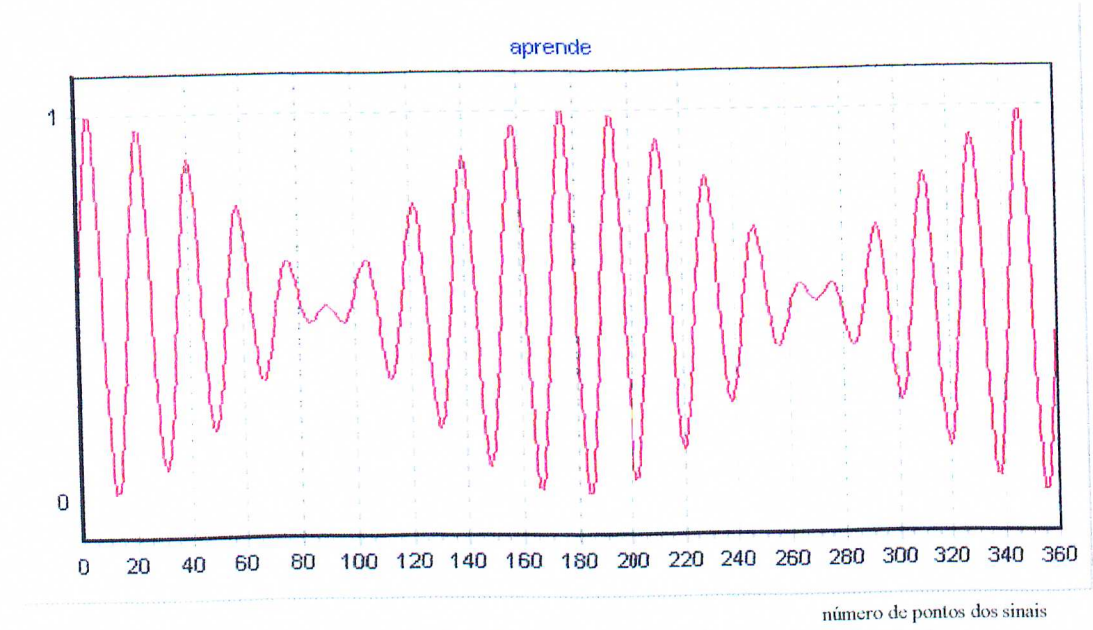


fig. 5.8: Sinal de entrada mais reprodução do sinal

Repetindo a análise com aproximação funcional para o sinal randômico:

sinal[i]:=Random(360)/400; (5.2)

tem-se o gráfico da figura 5.9:

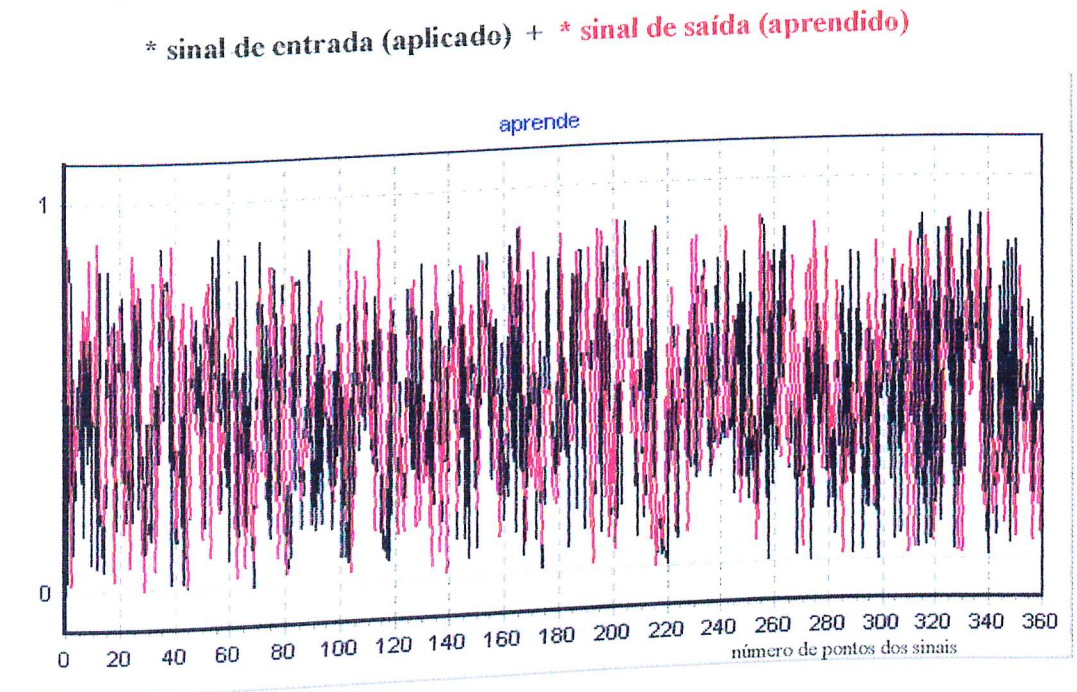


fig. 5.9: Reprodução de sinal randômico

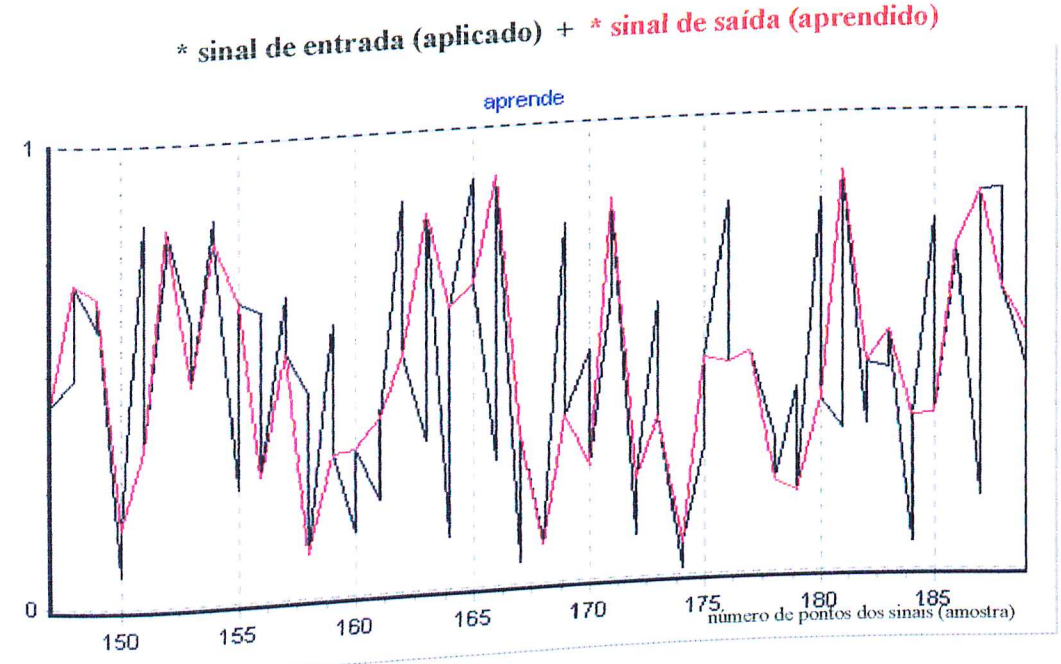


fig. 5.10: Reprodução de sinal randômico

Com a utilização da aproximação funcional houve uma maior proximidade entre o sinal aprendido e o sinal original.

5.2 Reprodução do sinal utilizando Fa = 0.5 e 20 passos de aprendizado:

sinal[i]:= (Sin((i*Pi)/180)+1)/2; (5.3)

*** sinal de entrada (aplicado) + * sinal de saída (aprendido)**

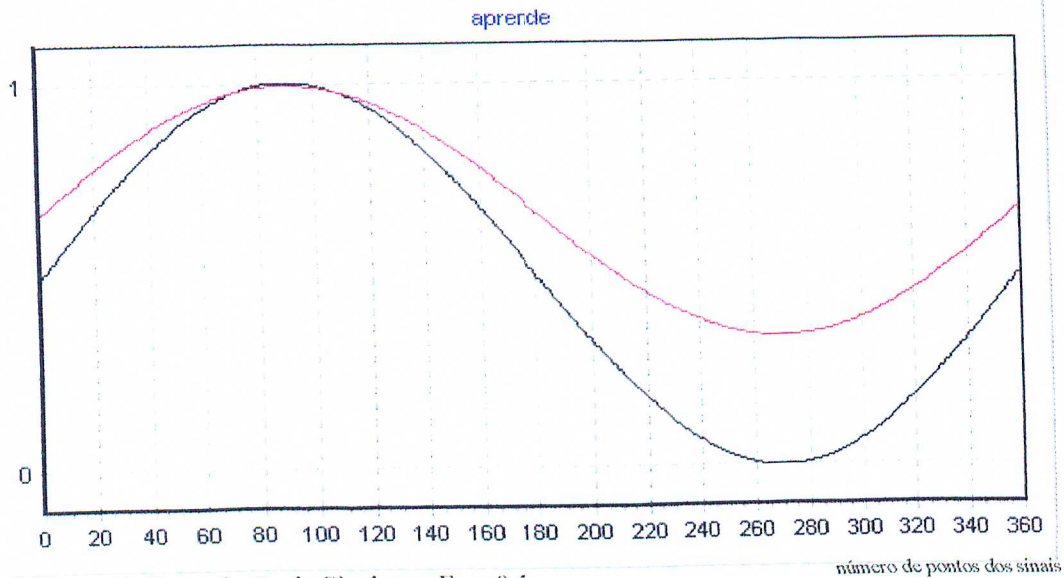


fig. 5.11: Reprodução de Sinal com Fa = 0.5

5.3 Reprodução do sinal utilizando Fa = 0.8 e 20 passos de aprendizado:

sinal[i]:= (Sin((i*Pi)/180)+1)/2;

*** sinal de entrada (aplicado) + * sinal de saída (aprendido)**

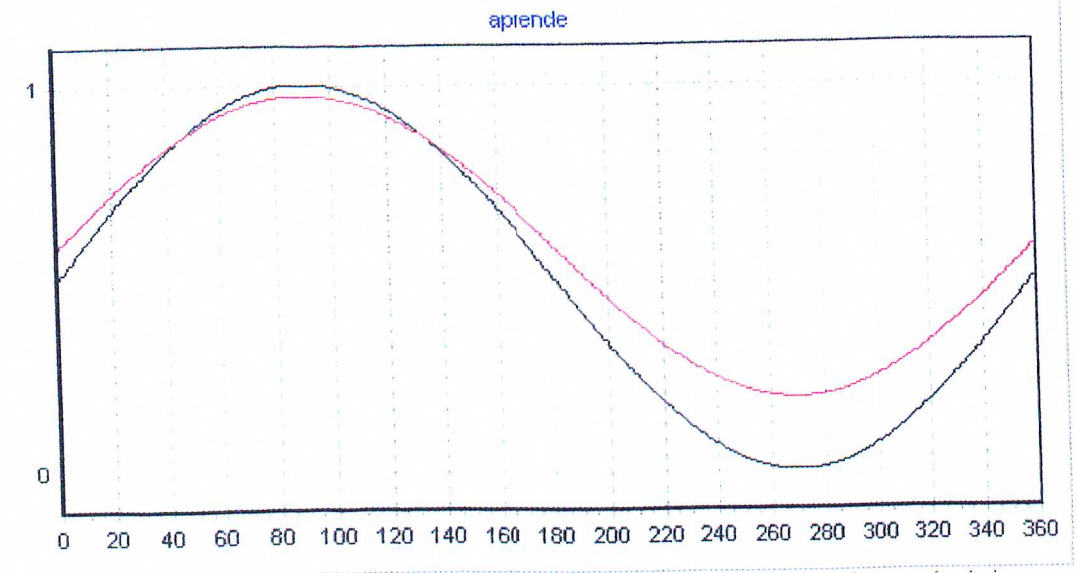


fig. 5.12: Reprodução de sinal com Fa = 0.8

5.4 Considerações sobre a reprodução de sinais utilizando 20 passos de aprendizagem:

A partir dos resultados obtidos no capítulo 3, de que para um Fa = 1 eram necessários 15 passos de aprendizagem para uma reprodução de um padrão, aplicando estas condições a vários padrões de uma matriz que constituem um sinal, conseguiu-se uma aproximação de 100% para sinais periódicos, e para os sinais com características randômicas a aproximação pode ser considerada como boa, pois o sinal reproduzido tende a acompanhar o sinal de entrada para a maioria dos pontos.

CAPÍTULO 6

Unidade Neural Artificial Paraconsistente de Comparação de Padrões

CAPÍTULO 6

Unidade Neural Artificial Paraconsistente de Comparação de Padrões

Em [3.5 e 3.6] foi visto que a interligação de Células Neurais Artificiais Paraconsistentes formam grupos denominados de Unidades Neurais Artificiais Paraconsistentes – UNAPs. Neste capítulo serão apresentadas as UNAPs que implementam uma comparação entre padrões (sinais).

6.1 Estrutura da Unidade Neural Artificial Paraconsistente de Comparação de Padrões - UNAPCP

A Unidade Neural Paraconsistente de Comparação de Padrões será formada por uma Célula Neural Artificial de Aprendizagem, por uma Célula Neural Artificial Paraconsistente de Conexão Lógica Seletiva de Maximização, citada no capítulo 3 (3.4.2), e por uma Célula Neural Artificial Paraconsistente de Decisão (cap. 3 – 3.4.6). O funcionamento e o algoritmo destas duas novas células serão detalhados nos próximos itens:

6.1.1 Funcionamento da Célula Neural Artificial Paraconsistente de Conexão Lógica Simples de Maximização:

A Célula Neural Artificial Paraconsistente de Conexão Lógica Simples - **CNAPCLs** – [94] tem a função de estabelecer conectivos lógicos entre sinais representativos de graus de crença. As células fazem os conectivos lógicos de maximização (OR) ou de minimização (AND), selecionando um dos sinais da entrada para ser conectado à saída. No caso da UNAPCP, a célula de conexão utilizada é a de maximização, que apresenta saída de acordo com o algoritmo apresentado:

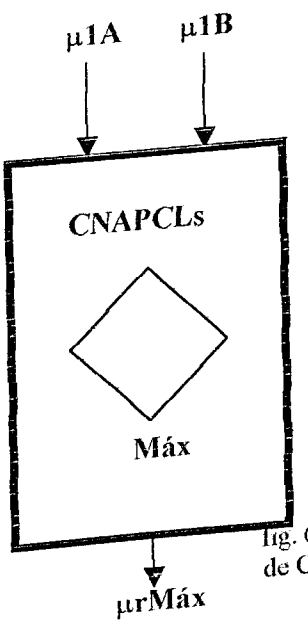


Fig. 6.1: Célula Neural Artificial Paraconsistente de Conexão Lógica Seletiva de Maximização

Algoritmo de Maximização

μ1A = Grau de Crença
μ1B = Grau de Descrença

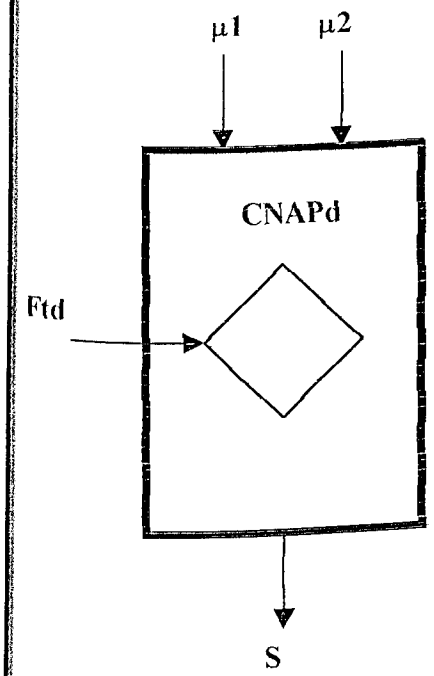
μr = Grau de Crença Resultante
μr = ((μ1A - μ1B) + 1) / 2 [E. E. B] (6.1)

Saída:

Se μr ≥ 1/2 , então μrMáx = μ1A
Senão : μrMáx = μ1B

6.1.2 Funcionamento da Célula Neural Artificial Paraconsistente de Decisão – CNAPd:

A Célula Neural Artificial Paraconsistente de Decisão - **CNAPd** [94] tem a função de trabalhar como nó de decisão nas RNAP's. A célula recebe em suas entradas dois sinais resultantes de análises efetuadas por outras células da rede, e apresenta na saída um resultado que representa uma conclusão da análise, como segue: saída = 1 – verdadeiro; saída = 0 – falso; saída = 0.5 – indefinido. O algoritmo e a representação da célula são apresentados a seguir:



Algoritmo de Decisão

$\mu 1$ = grau de crença de entrada: $0 \leq \mu 1 \leq 1$
$\mu 2$ = grau de descrença de entrada: $0 \leq \mu 2 \leq 1$
***Ftd* = *C1* = *Fator de Tolerância à Decisão*: $0 \leq C1 \leq 1$**
Gc = Grau de Certeza = $\mu 1 - \mu 2$
μR = Grau de Crença Resultante = $(\mu 1 - \mu 2 + 1)/2$ (6.2)
VLF = Valor Limite de Falsidade = $(1 - C1)/2$ (6.3)
VLV = Valor Limite de Verdade = $(1 + C1)/2$ (6.4)
Determinação do valor de saída:

Se: $VLF < \mu R < VLV$, então S = 0.5
Se: $\mu R \geq VLV$, então S = 1.0
Se: $\mu R \leq VLF$, *então* S = 0

fig. 6.2: Célula Neural Artificial Paraconsistente de Decisão

6.2 Funcionamento da Unidade Neural Artificial Paraconsistente de Comparação de Padrões - UNAPCP

A **UNAPCP** deverá inicialmente receber e aprender um padrão de referência através da Célula Neural Artificial Paraconsistente de Aprendizagem - CNAPap. A seguir, a saída desta célula, que representará o padrão aprendido, será uma das entradas de uma Célula Neural Artificial Paraconsistente de Conexão Lógica Seletiva de Maximização - CNAPLs. A outra entrada será composta por um padrão que se queira comparar com o padrão aprendido pela célula de aprendizagem. O esquema da **UNAPCP** é mostrado na figura 6.3:

Unidade Neural Artificial Paraconsistente de Comparação de Padrões:

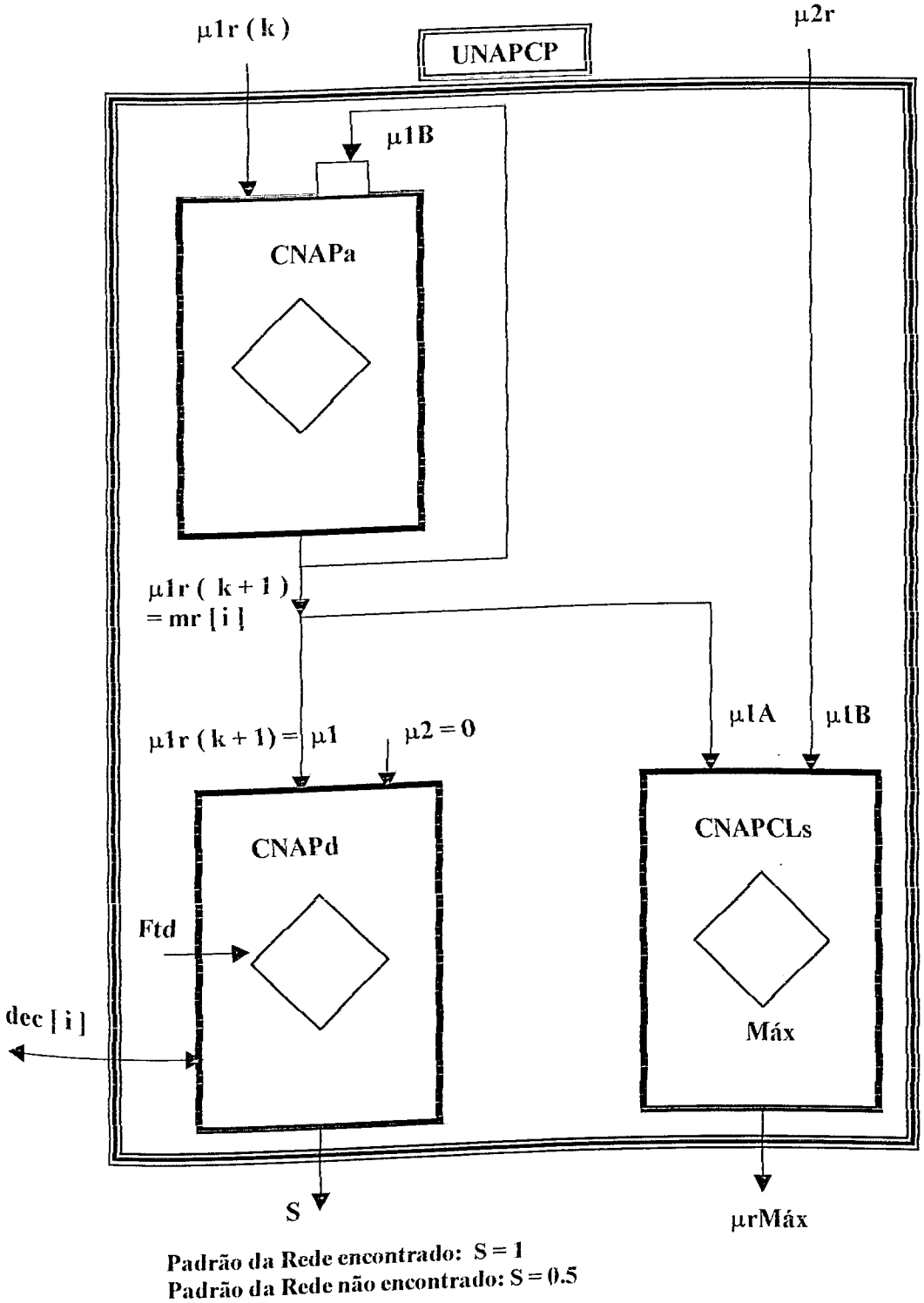


fig. 6.3: Unidade Neural Artificial Paraconsistente de Comparação de Padrões

No esquema da figura 6.3 **µ1r (k)** representa o padrão a ser aprendido pela unidade, **µ1r (k + 1)** será o padrão aprendido no final do processo, e **µ2r** o padrão que se quer comparar com **µ1r (k + 1)**. As saídas da **UNAPCP**, representadas por **S** e **µrMáx,** apresentarão respectivamente o valor de decisão e a repetição do valor aprendido. O valor de decisão conclui se o padrão inserido é igual ou não ao padrão da unidade, e a repetição do padrão aprendido pela unidade verifica se este é igual ao padrão inserido ou então é igual ao padrão de teste, se este não pertencer à rede.

6.3 Algoritmo da Célula de Aprendizagem da UNAPCP:

Inicialmente a célula de aprendizagem recebe um padrão ao qual ela vai aprender; o sinal referente a este padrão é gerado conforme o algoritmo abaixo:

```
procedure padrao01;
var
 i: Integer;
begin
  for i = 0 to 3600 do
  begin

padrao[i]:= 0.5 + (Sin((20*i*Pi)/180))*(Sin((100*i*Pi)/180))*(-Sin((600*i*Pi)/180))/2*Random(360)/400;

Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
  end;
end;
```

O sinal **padrao [i]** é gerado no formato de uma matriz com 360 pontos. Este sinal é mostrado a seguir, na figura 6.4:

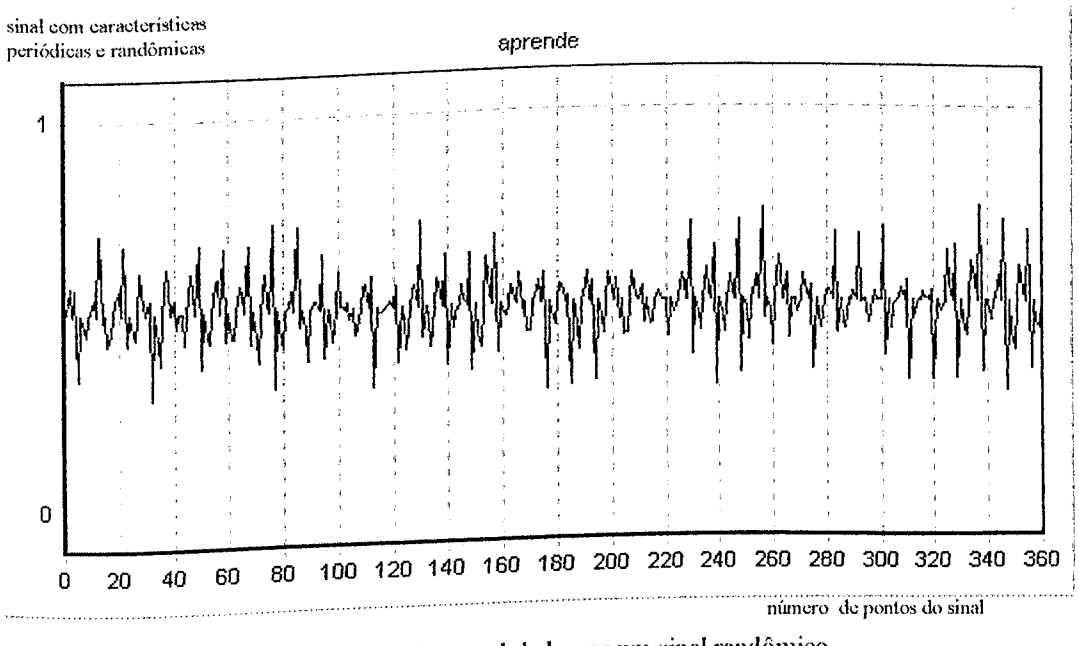


fig. 6.4: Simulação de um sinal periódico modulado por um sinal randômico

A seguir, a célula implementa a aprendizagem deste sinal através do seguinte código, conforme demonstrado no capítulo 5:

```
procedure sinal01Click;
var
i, k: Integer;

begin
Fa = 1.0;
m2[0] =0.5;
i = 0;

while i<=360 do
begin
 k=0;
 while k <= 20 do
  begin
   mr1[k] = ( padrao1[i] - (1 - m2[k] ) * Fa + 1 ) / 2;
   m2[k+1]= mr1[k];
   k = k+1;
  end;
 k=20;
 mr2[i] =mr1[k];
 i = i+1;
end;

for i=0 to 360 do
begin
 Chart1.Series[4].AddXY(i,mr2[i],'',clTeeColor);
end;
end;
```

Relacionando as variáveis envolvidas no algoritmo com o esquema da figura 6.3, tem-se:

μ1r (k) = padrao1 [i] (sinal de entrada) ;

μ1r (k + 1) = mr2 [i] (sinal aprendido) ;

μ1B = m2 [k] (realimentação da célula) ;

O fator de ajuste utilizado para a célula é Fa = 1.0, e o número de passos de aprendizagem é k = 20, o que de acordo com os resultados obtidos no capítulo 4 garante uma aprendizagem muito próxima de 100% para o sinal de entrada.

O sinal será aprendido com boa precisão, conforme mostra a figura 6.5 a seguir:

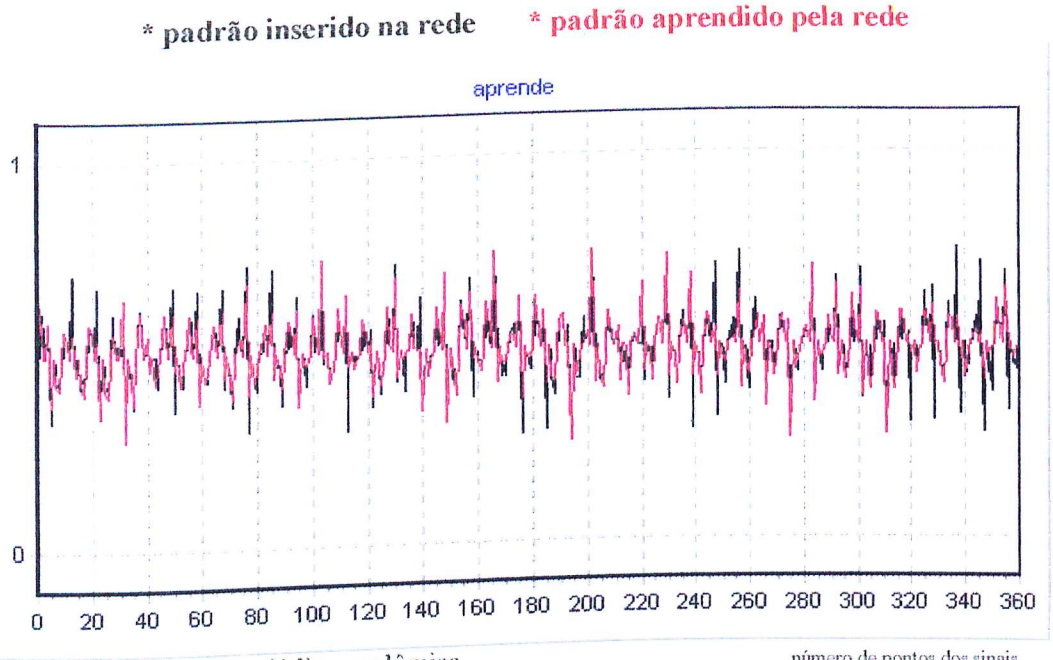


fig. 6.5: Reprodução de sinal periódico randômico

6.4 Algoritmo das Células de Conexão Lógica Simples de Maximização e Célula de Decisão da UNAPCP:

Estes algoritmos serão discutidos em conjunto pois no aplicativo usado na implementação dos ensaios estão inseridos dentro de uma mesma função chamada "sinalteste". Inicialmente são armazenados nas células de conexão os padrões de sinais que são residentes da rede, através de uma função aqui denominada de "armazena sinais". Nestes ensaios os sinais agora gerados têm características de uma matriz de 3600 pontos.

```
procedure armazenasinais;
var
i : Integer;
begin
for i = 0 to 3600 do
begin

padrao1[i] = (0.7 + (Sin((i*Pi)/180))*(-2*Sin((i*Pi)/180))*(Cos((i*Pi)/180))/2)*(Random(360)/400);

padrao2[i] = (0.7 + (Sin((1.0*i*Pi)/180))*(-2*Sin((1.0*i*Pi)/180))*(Cos((1.0*i*Pi)/180))/2)*(Random(360)/400);

padrao3[i] = 0.5 + (Sin((20*i*Pi)/180))*(Cos((i*Pi)/180))/2;

padrao4[i] = 0.5 + (0.99*Sin((20*i*Pi)/180))*(0.99*Cos((i*Pi)/180))/2;

end;
end;
```

A seguir, é implementada a função "sinalteste", que primeiro chama a função "armazenasinais" para dispor dos padrões ali gerados.

```
procedure sinalteste1;
var
i : Integer;

begin
armazenasinais;
sinal01;
sinal02;
sinal03;
sinal04;
```

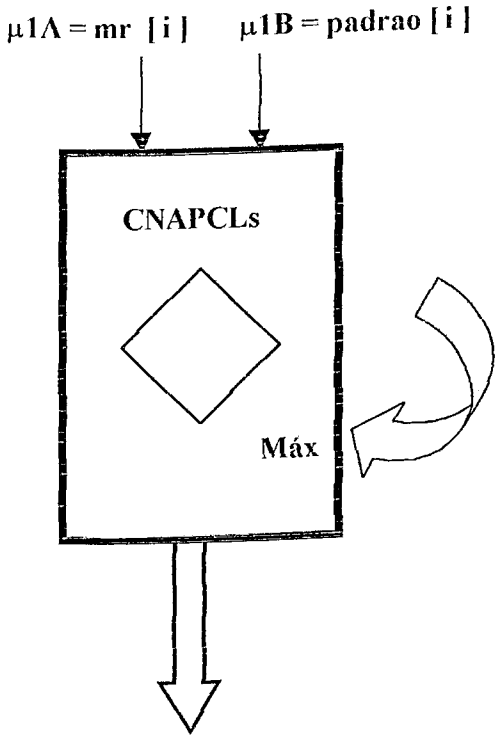
Na sequência, a função "sinalteste" implementará o algoritmo das células artificiais de conexão lógica de maximização (eq. 6.1) e de decisão (eq. 6.2); nas equações 6.5 do programa implementado, "teste" é o grau de crença resultante entre "mr" que representa o padrão aprendido, e "padrao" que representa o sinal que se quer verificar pertencente ou não à rede. Cada célula testa todos os padrões aprendidos com relação ao padrão que se quer verificar, como no algoritmo:

```
for i = 0 to 3600 do
 begin
  teste1[i] = ( (mr2[i] - padrao1[i])+1) / 2;    (6.5 )
  teste2[i] = ( (mr4[i] - padrao1[i])+1) / 2;
  teste3[i] = ( (mr6[i] - padrao1[i])+1) / 2;
  teste4[i] = ( (mr8[i] - padrao1[i])+1) / 2;
            ....
 teste5[i] = ( (mr2[i] – padrao2[i])+1) / 2;
            ....
            ....
  teste16[i] = ( (mr8[i] – padrao4[i])+1) / 2;
```

De acordo com a equação 6.5, pode-se perceber que se o sinal inserido "padrao [i]" pertencer à rede, o resultado do grau de crença resultante "teste [i]" será igual a 0.5. A partir daí, serão efetuados os testes para verificar se o resultado de **teste [i] = 0.5**, com uma faixa de tolerância que deve ser de acordo com a aplicação e a precisão desejada, verifica a correspondência do sinal inserido "padrao" com os sinais aprendidos "mr". No teste onde o resultado estiver dentro da faixa, equivale a dizer que o sinal inserido é praticamente igual àquele correspondente sinal aprendido pela rede.

O funcionamento simplificado das células de conexão lógica seletiva de maximização podem ser verificados através do esquema da figura 6.6:

ESQUEMA DE FUNCIONAMENTO DA CÉLULA DE CONEXÃO LÓGICA SELETIVA NA UNAPCP



TESTES EFETUADOS PELOS ALGORITMOS DAS CÉLULAS DE CONEXÃO LÓGICA MAX.

```
for i = 0 to 3600 do
 begin
// testes na célula da Unidade 1
  teste1[i] = ( (mr2[i] - padrao1[i])+1) / 2;
  teste5[i] = ( (mr2[i] - padrao2[i])+1) / 2;
  teste9[i] = ( (mr2[i] - padrao3[i])+1) / 2;
  teste13[i] = ( (mr2[i] - padrao4[i])+1) / 2;
  teste17[i] = ( (mr2[i] - padrao5[i])+1) / 2;
  teste21[i] = ( (mr2[i] - padrao6[i])+1) / 2;
       ....
// testes na célula da Unidade 2
teste2[i] = ( (mr4[i] – padrao1[i])+1) / 2;
teste6[i] = ( (mr4[i] – padrao2[i])+1) / 2;
            ....
             ....
 //testes na célula da Unidade 3
teste3[i] = ( (mr6[i] – padrao1[i])+1) / 2;
teste7[i] = ( (mr6[i] – padrao2[i])+1) / 2;
            ....
//testes na célula da Unidade 4
teste4[i] = ( (mr8[i] – padrao1[i])+1) / 2;
teste8[i] = ( (mr8[i] – padrao2[i])+1) / 2;
            ....

if (teste1 [i] >=0.49999) and (teste1 [i] <=0.50001)
then

 begin
 Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);

end
```

μrMáx = μ1A = mr [i], se teste [i] = 0,5 (padrao[i] pertence à rede)

μrMáx = μ1B = padrao [i], se teste [i] # 0,5 (padrao[i] não pertence à rede)

fig. 6.6: Funcionamento da Célula Neural Artificial Paraconsistente de Conexão Lógica Seletiva de Maximização na UNAPCP

Já as equações 6.6 e 6.7 mostradas a seguir são referentes à célula de conexão lógica de decisão:

$$\begin{aligned} &\mathbf{dec1[i] = (mr2[i]+1) / 2;} \\ &\mathbf{dec2[i] = (mr4[i]+1) / 2;} \\ &\mathbf{dec3[i] = (mr6[i]+1) / 2;} \\ &\mathbf{dec4[i] = (mr8[i]+1) / 2;} \end{aligned} \qquad (6.6)$$

$$\mathbf{vlv1[i] = (1+ padrao1[i]) / 2;} \qquad (6.7)$$

comparando com a equação 6.2 do algoritmo da célula de decisão, tem-se na eq. 6.6 que **dec [i]** equivale ao grau de crença resultante μr ; **mr [i]** equivale ao grau de crença de entrada $\mu 1$, com o grau de descrença $\mu 2$ nulo ($\mu 2 = 0$), **significando intuitivamente que não há necessidade de um grau de descrença na entrada pois se deseja apenas quantizar os valores do padrão inserido na rede**. Portanto, o grau de crença **dec [i]** resultante também será uma matriz, pois o que se está em análise é um sinal composto de vários pontos, sendo portanto uma variável com valor dinâmico. Nesse contexto, na eq. 6.7 tem-se **vlv [i]** equivalente a VLV, que é o valor limite de verdade; comparando com a equação 6.4, **C1** (fator de tolerância à decisão) é o próprio sinal **padrao1[i]**, e o valor limite de verdade também se torna uma matriz. Adaptando os critérios de tomada de decisão da célula para a aplicação de análise de sinais, tem-se:

Se: dec [i] < vlv [i], então S = 0.5 **(saída = indefinição)**

Se: dec [i] ≥ vlv [i], então S = 1.0 **(saída = verdadeiro)**

Analisando os critérios acima a partir da eq. 6.6, se o sinal analisado pertencer à rede tem-se que **padrao [i] = mr [i]**:

$$\begin{aligned} &\mathbf{dec[i] = (mr[i]+1) / 2;} \\ &\mathbf{vlv[i] = (1+ padrao[i]) / 2;} \end{aligned}$$

e pelo segundo critério, S = 1.0, já que forçosamente **dec [i] = vlv [i]**. Porém, se o sinal não pertencer à rede, **padrao [i] # mr [i]** e consequentemente **dec [i] # vlv [i]**. Desta forma, quando o sinal não pertencer à rede, cairá no critério **dec [i] < vlv [i], então S = 0.5**, que equivale a uma indefinição na LPA2v.

A sequência do algoritmo que implementa estes testes é mostrada a seguir:

```
if (teste10[i] >=0.49999) and (teste10[i] <=0.50001)  then

  begin
   Chart1.Series[0].AddXY(i,padrao[i],'',clTeeColor);
   if (dec1[i] >= vlv3[i]) then
   S3 =1;
   Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
  end


 else if  (teste11[i] >=0.49999) and (teste11[i] <=0.50001) then

begin
   Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
   if (dec2[i] >= vlv3[i]) then
   S3 =1;
   Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
  end

    else if  (teste12[i] >=0.49999) and (teste12[i] <=0.50001) then

  begin
   Chart1.Series[2].AddXY(i,padrao3[i],'',clTeeColor);
   if (dec3[i] >= vlv3[i]) then
   S3 =1;
   Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
  end

    else if  (teste13[i] >=0.49999) and (teste13[i] <=0.50001) then

    begin
     Chart1.Series[3].AddXY(i,padrao4[i],'',clTeeColor);
     if (dec3[i] >= vlv3[i]) then
     S3 =1;
     Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
    end

     else
      begin
      S3 =0.5;
      Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
     end
 end;
end;
```

Cabe ressaltar que, cada célula compara os padrões inseridos com o padrão armazenado em si mesma.

Um esquema demonstrando a parte funcional da célula de decisão é mostrado na figura 6.7:

ESQUEMA DE FUNCIONAMENTO DA CÉLULA DE DECISÃO NA UNAPCP

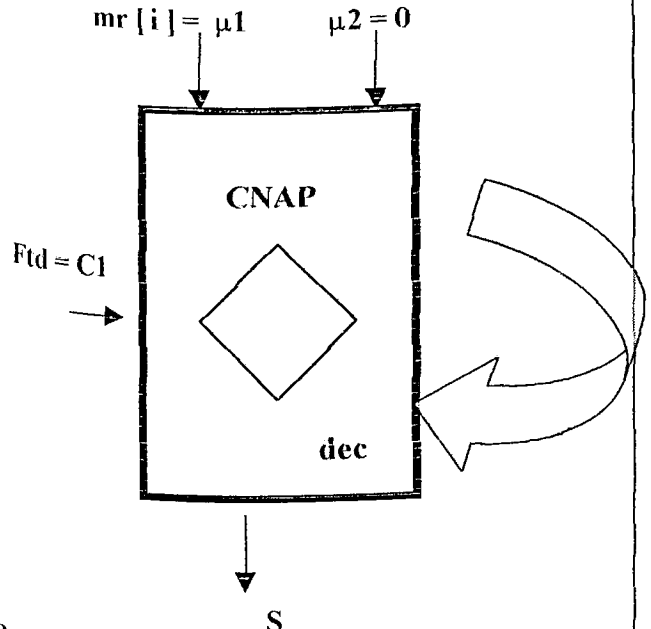
ADAPTAÇÃO DAS EQUAÇÕES DA CÉLULA DE DECISÃO

μR = Grau de Crença Resultante = $(\mu 1 - \mu 2 + 1) / 2$ = dec [i] com $\mu 1$ = mr [i] e $\mu 2 = 0$

VLV = Valor Limite de Verdade = $(1 + C1) / 2$ = vlv [i] com C1 = padrao [i]

Se: dec [i] < vlv [i], então S = 0.5 (saída = indefinição)

Se: dec [i] ≥ vlv [i], então S = 1.0 (saída = verdadeiro)



Padrão da Rede encontrado: S = 1

Padrão da Rede *não encontrado*: S = 0.5

fig. 6.7: Funcionamento da Célula Neural Artificial Paraconsistente de Decisão na UNAPCP

ALGORITMO DA CÉLULA DE DECISÃO NA *UNAPCP*

```
dec1[i] = ( mr2[i]+1) / 2;
dec2[i] = ( mr4[i]+1) / 2;
dec3[i] = ( mr6[i]+1) / 2;
dec4[i] = ( mr8[i]+1) / 2;

 vlv1[i] = (1+ padrao1[i]) / 2;

if (teste1[i] >=0.49999) and (teste1[i] <=0.50001)  then

  begin
Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
   if (dec1[i] >= vlv1[i]) then
   S1 =1;
   Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
  end


 else if  (teste2[i] >=0.49999) and (teste2[i] <=0.50001)
then

begin
Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
   if (dec2[i] >= vlv1[i]) then
   S1 =1;
   Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
  end
              ...
//repete teste para os padrões 3 e 4
              ...
else
      begin
      S1 =0.5;
      Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
     end
  end;
end;
```

6.5 Considerações sobre a Unidade Neural Artificial Paraconsistente de Comparação de Padrões

A UNAPCP possui então duas entradas, uma das quais receberá um sinal que será armazenado e aprendido pela mesma, e na outra entrada receberá um sinal o qual se quer verificar se o mesmo é semelhante (de acordo com critérios a serem definidos) com o sinal armazenado na unidade: se o sinal for semelhante, a célula de decisão fornecerá na saída valor 1 e a célula de conexão lógica simples terá como saída o próprio sinal aprendido e armazenado; caso contrário, a célula de decisão fornecerá na saída valor meio e a célula de conexão lógica simples fornecerá o padrão inserido para comparação.

CAPÍTULO 7

Rede Neural Artificial Paraconsistente para Classificação de Sinais

CAPÍTULO 7

Rede Neural Artificial Paraconsistente para Classificação de Sinais

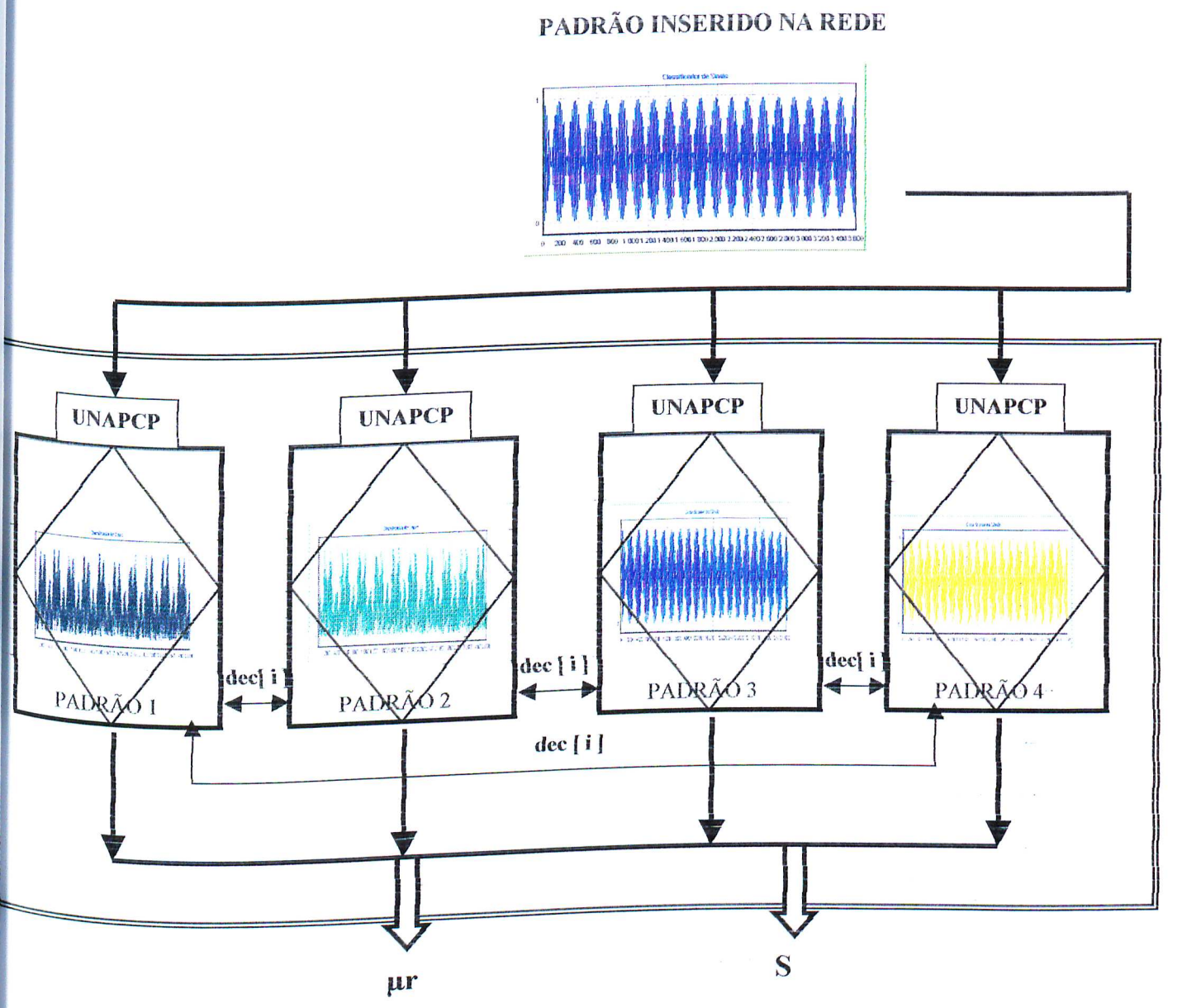
7.1 Classificação de Sinais

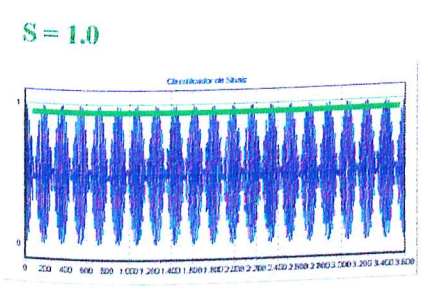
De acordo com [106], um sinal pode pertencer a várias categorias, e como exemplo pode-se citar o diagnóstico de condições cardíacas patológicas a partir de EKG, onde para cada categoria diagnóstica é escolhida uma coleção de traçados representativos. Aplicações de técnicas com redes neurais em processamento de sinais vêm se concentrando principalmente na classificação de sinais em categorias. Resultados relevantes foram alcançados na classificação de sinais bioelétricos, particularmente EKG, EEG e EMG; na classificação de sinais de voz para reconhecimento de fonemas ou do locutor; na classificação para reconhecimento de imagens e na classificação para reconhecimento de ecos sonar. Os problemas de classificação de sinais geralmente envolvem um problema associado de parametrização, referidos na literatura como pré-processamento ou filtragem.

7.2 Rede Neural Artificial Paraconsistente para Classificação de Sinais

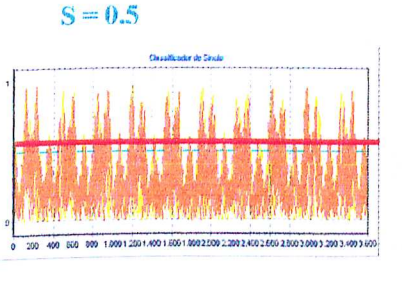
Neste trabalho a idéia básica é a utilização das características demonstradas nos ensaios de uma RNAP para classificar ou identificar sinais. A rede seria constituída por Unidades Neurais capazes de aprender determinado padrão ou sinal através de sua função matemática característica, armazenar este padrão ou sinal aprendido, e compará-lo com qualquer padrão externo que se queira identificar. A estrutura da Rede Neural Artificial Paraconsistente é apresentada em sua forma geral na figura 7.1 a seguir:

7.3 Estrutura da Rede Neural Artificial Paraconsistente para Classificação de Sinais





RESPOSTA DA REDE CASO O PADRÃO PERTENÇA À MESMA



RESPOSTA DA REDE CASO O PADRÃO NÃO PERTENÇA À MESMA

fig. 7.1: Estrutura da Rede Neural Artificial Paraconsistente para Classificação de Sinais

A figura 7.1, apresenta a proposta de uma rede com quatro Unidades Neurais Artificiais Paraconsistentes de Comparação de Padrões (6.2). Esta rede é capaz de armazenar quatro sinais diferentes. Se for inserido na rede um padrão semelhante ou igual ao aprendido pela rede, a saída da mesma apresentará este sinal mais o valor S = 1, que na LPA2v representa o grau de certeza; caso contrário, aparecerá na saída o valor S = 0.5, que na LPA2v representa o valor indefinido, mais o sinal inserido.

Os critérios para estabelecer se um sinal pertence ou não à rede já foram descritos no capítulo 7, e podem ser ajustados conforme a aplicação.

7.4 Resposta da Rede Neural Artificial Paraconsistente para Classificação de Sinais

A rede foi testada a partir de pequenas diferenças de amplitude e de freqüência, para sinais simplesmente periódicos e sinais com características randômicas; os padrões de teste com os resultados obtidos são apresentados a seguir:

7.4.1 Resposta da Rede a Sinais com Características Randômicas: diferenciação pela freqüência

Os padrões 1 e 2 apresentados respectivamente nas figuras 7.2 e 7.3 são dois sinais periódicos modulados por um sinal randômico, que podem simular um sinal bioelétrico ou um sinal de voz; a diferença entre os dois está apenas na freqüência, como mostram as equações 7.1 e 7.2, e as figuras 7.2 e 7.3, abaixo:

padrao1[i] = (0.7 + (Sin((i*Pi)/180))*(-2*Sin((i*Pi)/180))*(Cos((i*Pi)/180))/2)*(Random(360)/400); (7.1)

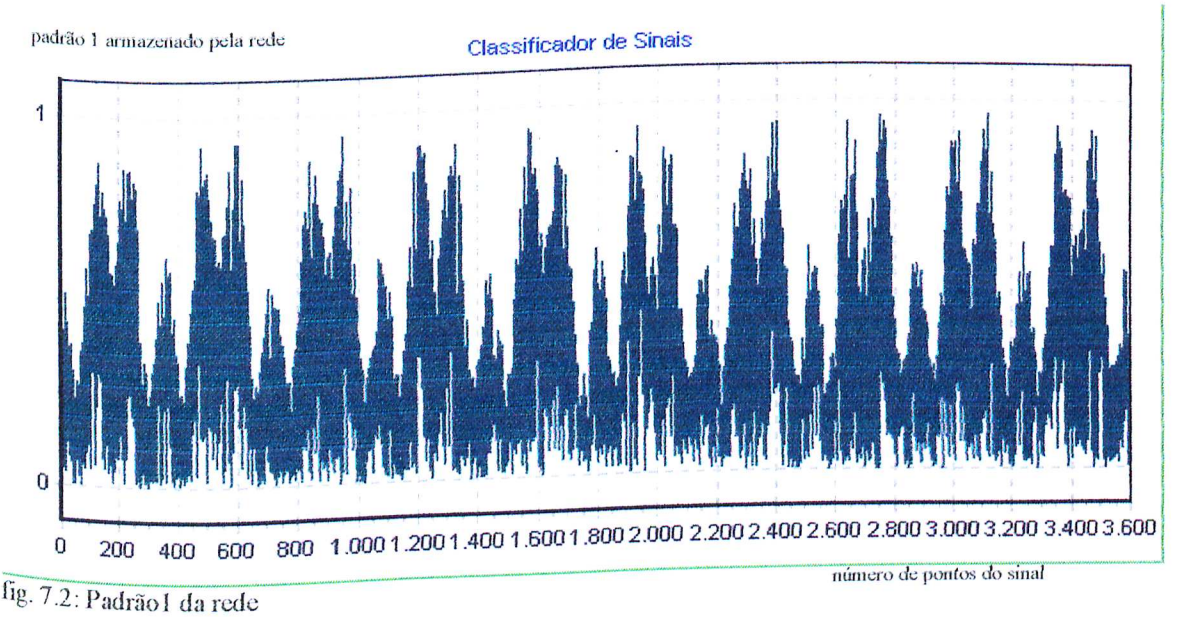


fig. 7.2: Padrão1 da rede

padrao2[i] = (0.7 + (Sin((0.99*i*Pi)/180))*(-2*Sin((0.99*i*Pi)/180))*(Cos((0.99*i*Pi)/180))/2)*(Random(360)/400);
(7.2)

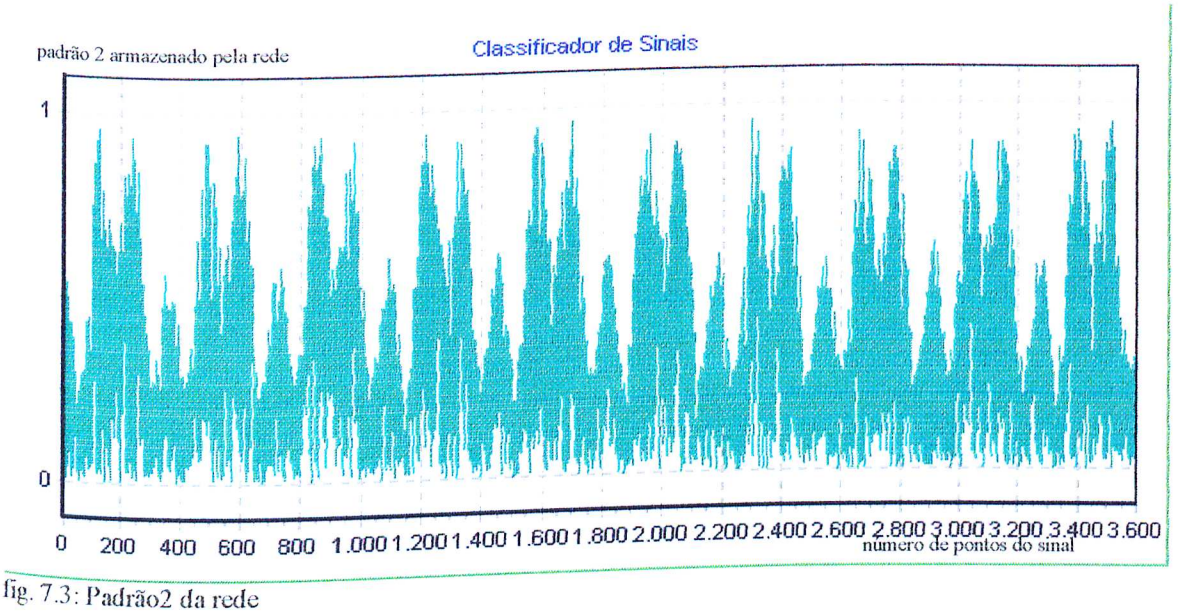


fig. 7.3: Padrão2 da rede

O algoritmo abaixo é a descrição da aprendizagem do padrão1 [i] utilizado no ensaio:

```
procedure TForm1.sinal01Click(Sender: TObject);
var
i, k: Integer;

begin

Fa = 1.0;
m2[0] = 0.5;
i = 0;

while i<=3600 do
begin
 k = 0;
 while k <= 20 do
  begin
   mr1[k] = ( padrao1[i] - (1 - m2[k] ) * Fa + 1 ) / 2;
   m2[k+1] = mr1[k];
   k = k+1;
  end;
 k =20;
 mr2[i] =mr1[k];
 i = i+1;
end;

for i = 0 to 3600 do
begin
 Chart1.Series[4].AddXY(i,mr2[i],'',clTeeColor);
end;
end;
```

Quando este algoritmo é processado, é gerado o sinal **mr2 [i]**, que corresponde ao sinal aprendido pela rede. Na figura 7.4, para uma melhor comparação, tem-se o sinal aprendido mr2 [i] sobreposto ao padrão1 [i]:

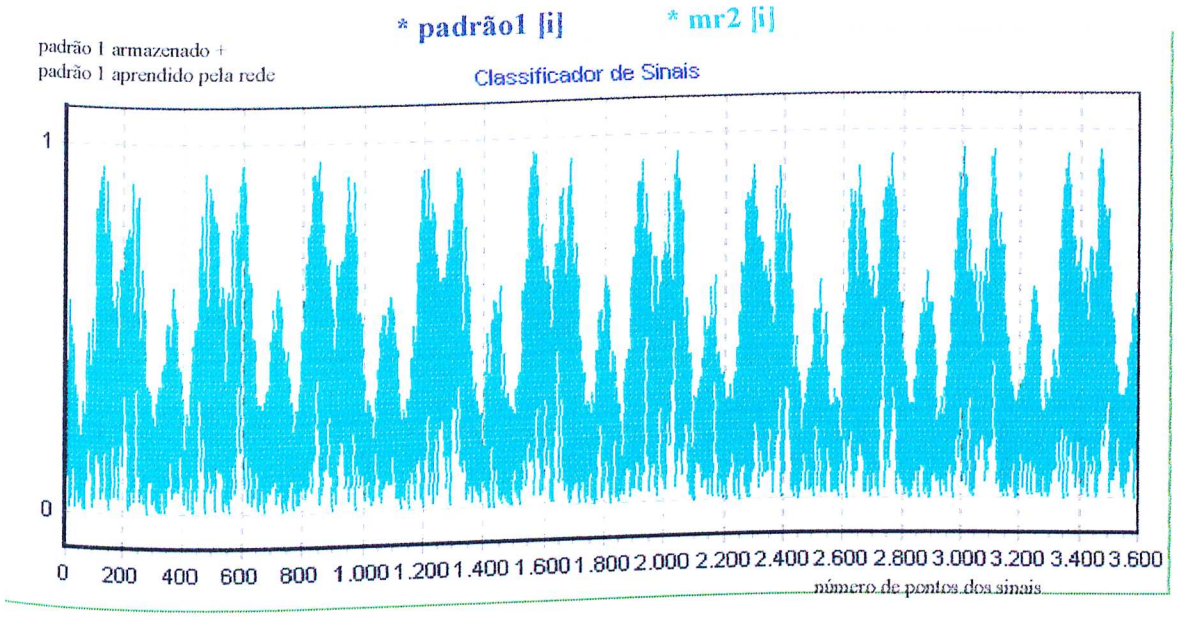


fig. 7.4: Sinal mr2 (aprendizado do padrão1) sobreposto ao padrão1

O algoritmo abaixo é a descrição da aprendizagem do padrão2 [i], utilizado no ensaio:

```
procedure TForm1.sinal02Click(Sender: TObject);
var
i, k: Integer;

begin

Fa = 1.0;
m3[0] = 0.5;
i = 0;

while i<=3600 do

begin
 k = 0;
 while k <= 20 do
  begin
   mr3[k] = ( padrao2[i] - (1 - m3[k] ) * Fa + 1 ) / 2;
   m3[k+1] = mr3[k];
   k = k+1;
  end;
  k = 20;
  mr4[i] =mr3[k];
  i = i+1;
 end;
```

```
for i = 0 to 3600 do

 begin
  Chart1.SeriesList.Series[5].AddXY(i,mr4[i],'',clTeeColor);
 end;
end;
```

Após a implementação deste algoritmo, é gerado o sinal de aprendizagem **mr4 [i]**, correspondente ao padrão2 [i]; a figura 7.5 representa mr4 [i] sobreposto ao padrão2 [i]:

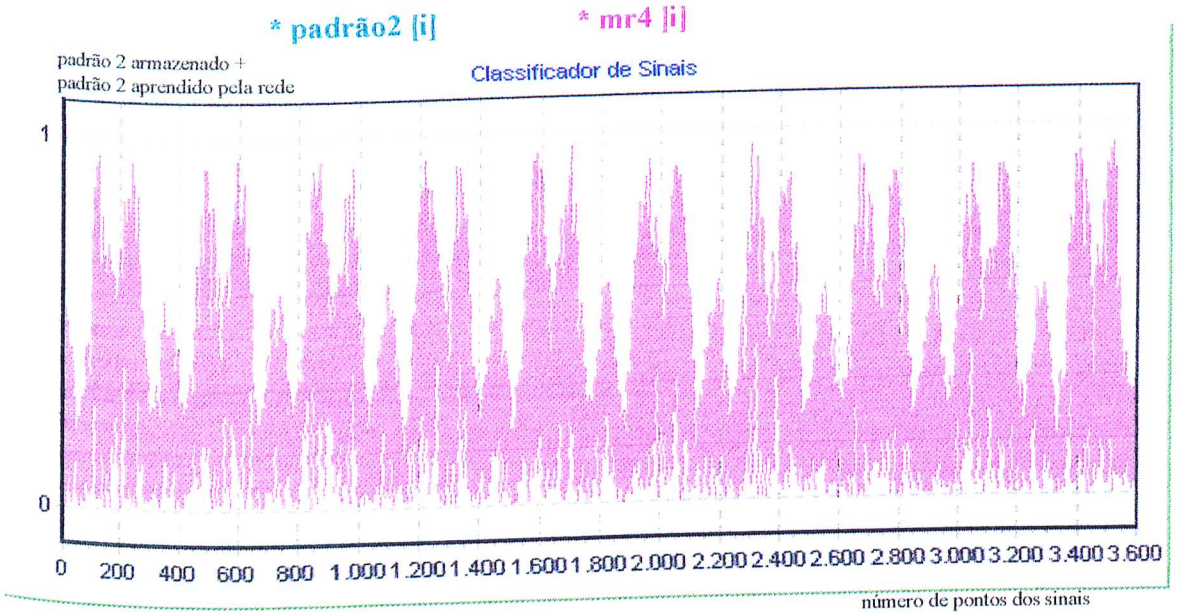


fig. 7.5: Sinal mr4 (aprendizado do padrão2) sobreposto ao padrão2

O código abaixo insere um sinal igual ao padrão1 [i] na rede, e verifica se este pertence à mesma:

```
procedure TForm1.sinalteste1Click(Sender: TObject);
var
i:Integer; S1: Real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
S1= 0;
 for i = 0 to 3600 do
  begin
  teste1[i] =((mr2[i]-padrao1[i])+1)/2;
  teste2[i] =((mr4[i]-padrao1[i])+1)/2;
  teste3[i] =((mr6[i]-padrao1[i])+1)/2;
  teste4[i] =((mr8[i]-padrao1[i])+1)/2;
```

```
dec1[i] =(mr2[i]+1)/2;
dec2[i] =(mr4[i]+1)/2;
dec3[i] =(mr6[i]+1)/2;
dec4[i] =(mr8[i]+1)/2;
vlv1[i] =(1+padrao1[i])/2;


if (teste1[i] >=0.49999) and (teste1[i] <=0.50001) then
begin
Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
if (dec1[i] >= vlv1[i]) then
begin
S1=1;
Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
end;
end

else if (teste2[i] >=0.49999) and (teste2[i] <=0.50001) then
begin
Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
if(dec2[i] >= vlv1[i]) then
begin
S1=1;
Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
end;
end

else if (teste3[i] >=0.49999) and (teste3[i] <=0.50001) then
begin
Chart1.Series[2].AddXY(i,padrao3[i],'',clTeeColor);
if (dec3[i] >= vlv1[i]) then
S1=1;
Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
end

else if (teste4[i] >=0.49999) and (teste4[i] <=0.50001)then
begin
Chart1.Series[3].AddXY(i,padrao4[i],'',clTeeColor);
if (dec4[i] >= vlv1[i]) then
begin
S1=1;
Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
end;
end

else
 begin
S1= 0.5;
Chart1.Series[10].AddXY(i,S1,'',clTeeColor);
 end
end;
end;
```

Neste caso, a resposta da rede à consulta se o padrão inserido é igual ao padrão 1, é apresentado conforme a figura 7.6:

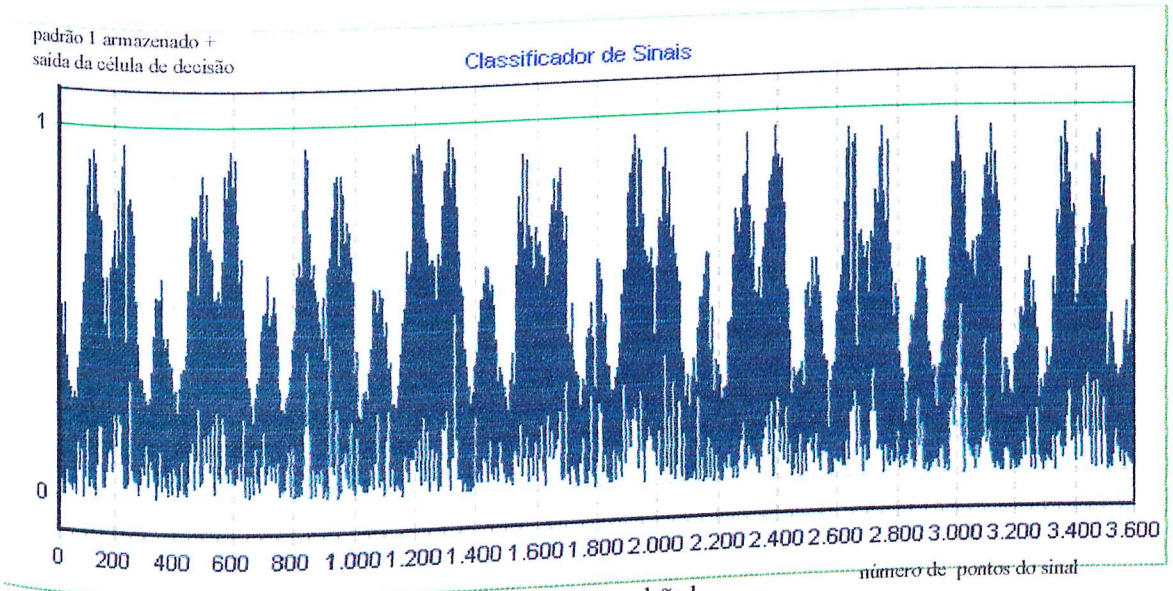


fig. 7.6: Resposta da rede para um sinal inserido igual ao padrão1

Em uma análise visual do gráfico da figura 7.6 vê-se que, embora o padrão1 [i] seja praticamente igual ao padrão2 [i], de acordo com os critérios estabelecidos, a rede foi capaz de diferenciar os dois, relacionando o padrão inserido com o padrão aprendido pela mesma com alto grau de precisão.

Análogamente, o código abaixo insere na rede um sinal igual ao padrão2 [i] e verifica se o mesmo pertence à rede:

```
procedure TForm1.sinalteste2Click(Sender: TObject);
var
i:Integer; S2: Real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
S2= 0;
 for i:=0 to 3600 do
  begin
  teste5[i] =((mr2[i]-padrao2[i])+1)/2;
  teste6[i] =((mr4[i]-padrao2[i])+1)/2;
  teste7[i] =((mr6[i]-padrao2[i])+1)/2;
  teste8[i] =((mr8[i]-padrao2[i])+1)/2;
  dec1[i] =(mr2[i]+1)/2;
  dec2[i] =(mr4[i]+1)/2;
  dec3[i] =(mr6[i]+1)/2;
  dec4[i] =(mr8[i]+1)/2;
  vlv2[i] =(1+padrao2[i])/2;
```

```
if (teste5[i] >=0.4999999999) and (teste5[i] <=0.5000000001) then

  begin
  Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
  if (dec1[i] >= vlv2[i]) then
  S2=1;
  Chart1.Series[10].AddXY(i,S2,'',clTeeColor);
  end
  else if (teste6[i] >=0.49999) and (teste6[i] <=0.50001) then
  begin
  Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
  if (dec2[i] >= vlv2[i]) then
  S2 =1;
  Chart1.Series[10].AddXY(i,S2,'',clTeeColor);
  end

   else if (teste7[i] >=0.49999) and (teste7[i] <=0.50001) then
   begin
   Chart1.Series[2].AddXY(i,padrao3[i],'',clTeeColor);
   if (dec3[i] >= vlv2[i]) then
   S2=1;
   Chart1.Series[10].AddXY(i,S2,'',clTeeColor);
   end

   else if (teste8[i] >=0.49999) and (teste8[i] <=0.50001) then
   begin
   Chart1.Series[3].AddXY(i,padrao4[i],'',clTeeColor);
   if (dec4[i] >= vlv2[i]) then
   S2=1;
   Chart1.Series[10].AddXY(i,S2,'',clTeeColor);
   end

   else
    begin
   S2=0.5;
   Chart1.Series[10].AddXY(i,S2,'',clTeeColor);
   end
 end;
end;
```

Depois da consulta, a saída da rede apresentará como resultado a figura 7.7:

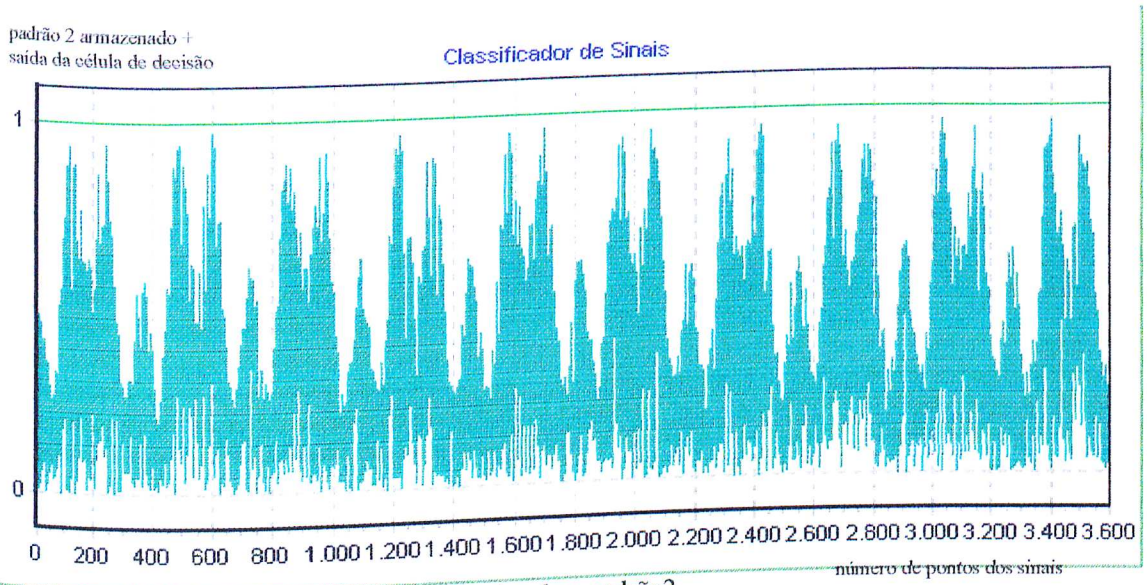


fig. 7.7: Resposta da rede para um sinal inserido igual ao padrão2

Novamente percebe-se a capacidade da rede de diferenciar sinais muito parecidos.

7.4.2 Resposta da Rede a Sinais com Características Periódicas e diferenciadas pela amplitude

Os padrões 3 e 4 aprendidos e armazenados pela rede são mostrados agora pelas equações 8.3 e 8.4, e os sinais gerados por ambos são mostrados nas figuras 7.8 e 7.9:

padrao3[i]= 0.5 + (Sin((20*i*Pi)/180))*(Cos((i*Pi)/180))/2;

(7.3)

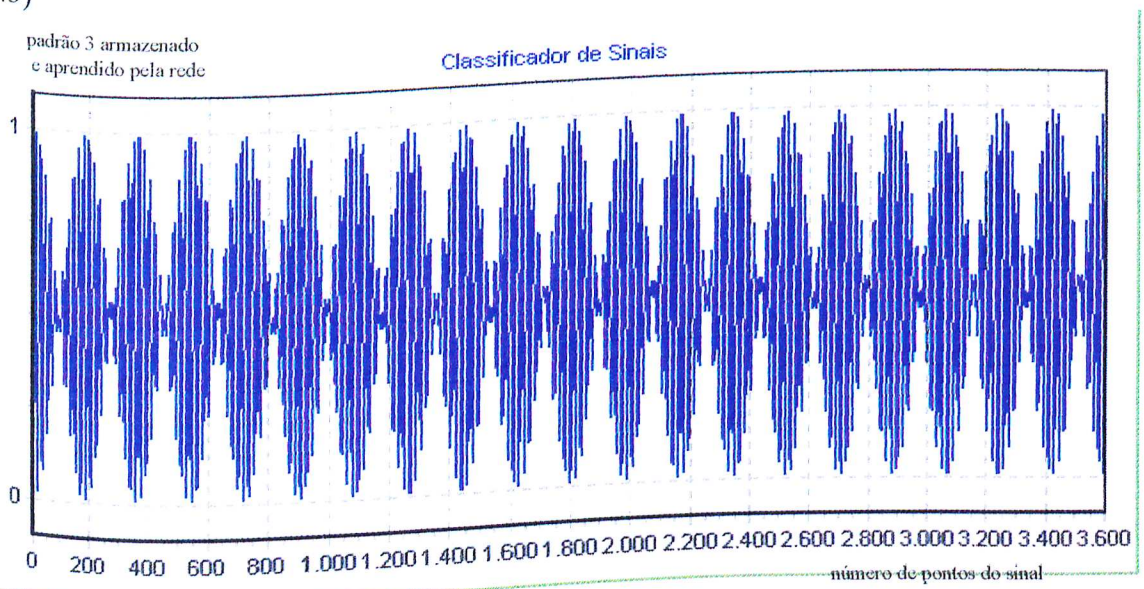


fig. 7.8: Padrão 3 da rede

padrao4[i] = 0.5 + (0.99*Sin((20*i*Pi)/180))*(0.99*Cos((i*Pi)/180))/2;
(7.4)

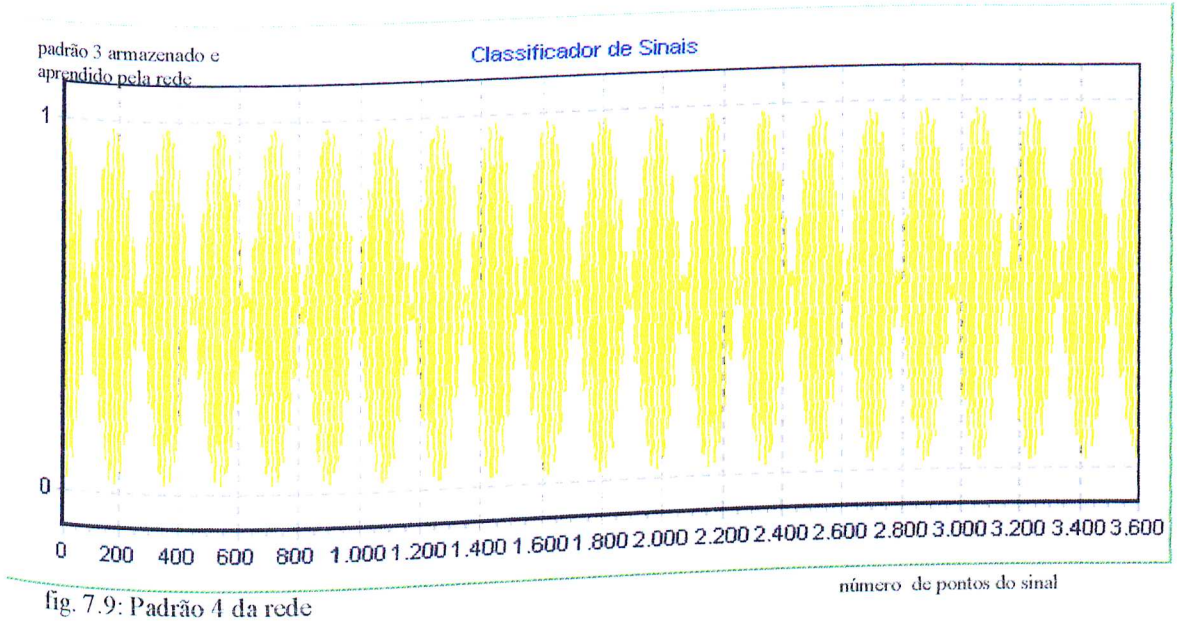


fig. 7.9: Padrão 4 da rede

Após o código a seguir ser processado, é gerado o sinal de aprendizagem do padrão3 [i], chamado **mr6 [i]**:

```
procedure TForm1.sinal03Click(Sender: TObject);
var
i, k: Integer;

begin

Fa:= 1.0;
m4[0]:=0.5;
i:=0;

while i<=3600 do

begin
 k = 0;
 while k <= 20 do
  begin
   mr5[k] = ( padrao3[i] - (1 - m4[k] ) * Fa + 1 ) / 2;
   m4[k+1] = mr5[k];
   k= k+1;
  end;
  k = 20;
  mr6[i] =mr5[k];
  i = i+1;
 end;
for i= 0 to 3600 do
begin
 Chart1.Series[6].AddXY(i,mr6[i],'',clTeeColor);
  end;
end;
```

Na figura 7.10 tem-se o sinal correspondente à aprendizagem **mr6 [i]** sobreposto ao padrão3 [i]:

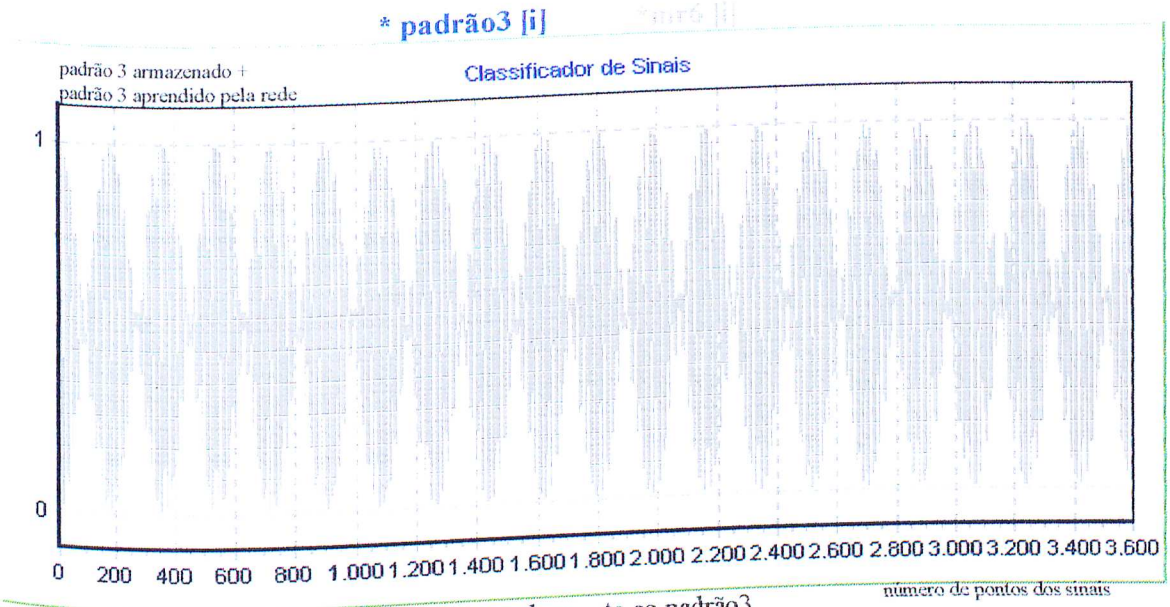


fig. 7.10: Sinal mr6 (aprendizado do padrão3) sobreposto ao padrão3

Após o código abaixo ser processado, é gerado o sinal de aprendizagem do padrão4 [i], chamado **mr8 [i]**:

```
procedure TForm1.sinal04Click(Sender: TObject);
var
i, k: Integer;

begin

Fa = 1.0;
m5[0] = 0.5;
i = 0;

while i<=3600 do
begin
 k= 0;
 while k <= 20 do
  begin
   mr7[k] = ( padrao4[i] - (1 - m5[k] ) * Fa + 1 ) / 2;
   m5[k+1] = mr7[k];
   k = k+1;
  end;
  k = 20;
  mr8[i] =mr7[k];
  i = i+1;
 end;
for i = 0 to 3600 do
begin
 Chart1.Series[7].AddXY(i,mr8[i],'',clTeeColor);

end;
end;
```

Na figura 7.11 tem-se o sinal correspondente à aprendizagem **mr8 [i]** sobreposto ao padrão4 [i]:

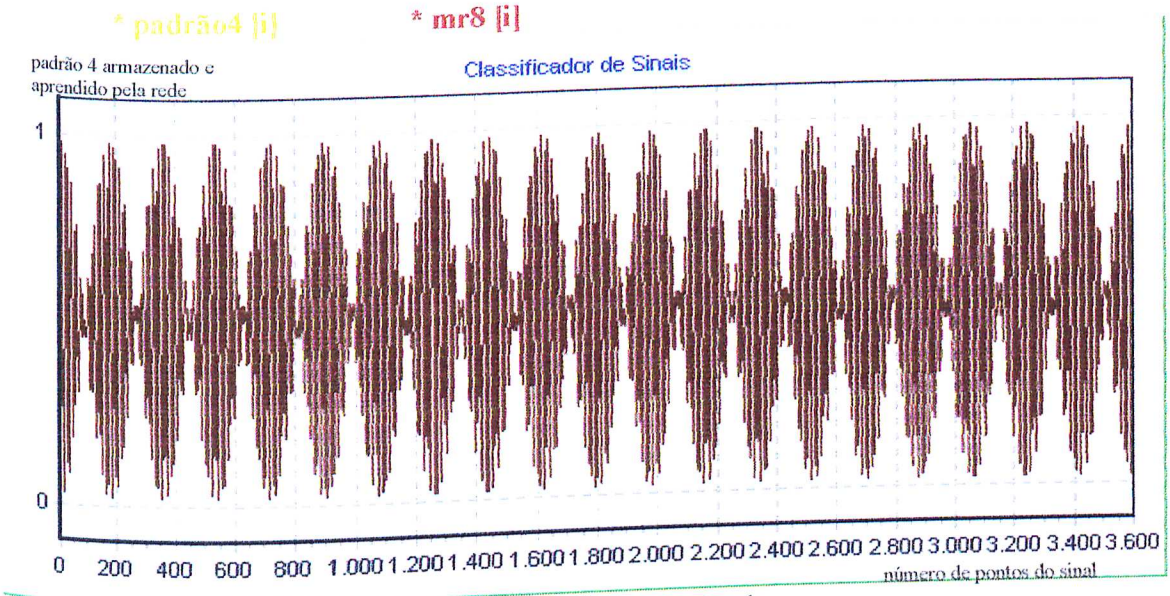


fig. 7.11: Sinal mr8 (aprendizado do padrão4) sobreposto ao padrão4

O código a seguir insere um sinal igual ao padrão3 [i] na rede, e verifica se este pertence à mesma:

```
procedure TForm1.sinalteste3Click(Sender: TObject);
var
i:Integer; S3:Real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
S3=0;
 for i=0 to 3600 do
  begin
  teste9[i] =((mr2[i]-padrao3[i])+1)/2;
  teste10[i] =((mr4[i]-padrao3[i])+1)/2;
  teste11[i] =((mr6[i]-padrao3[i])+1)/2;
  teste12[i] =((mr8[i]-padrao3[i])+1)/2;
  dec1[i] =(mr2[i]+1)/2;
  dec2[i] =(mr4[i]+1)/2;
  dec3[i] =(mr6[i]+1)/2;
  dec4[i] =(mr8[i]+1)/2;
  vlv3[i] =(1+padrao3[i])/2;

  if (teste9[i] >=0.49999) and (teste9[i] <=0.50001)  then
  begin
  Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
  if (dec1[i] >= vlv3[i]) then
  S3 =1;
  Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
  end
```

```
else if (teste10[i] >=0.49999) and (teste10[i] <=0.50001) then
begin
Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
if (dec2[i] >= vlv3[i]) then
S3 =1;
Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
end

else if (teste11[i] >=0.49999) and (teste11[i] <=0.50001) then
begin
Chart1.Series[2].AddXY(i,padrao3[i],'',clTeeColor);
if (dec3[i] >= vlv3[i]) then
S3 =1;
Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
end

else if (teste12[i] >=0.49999) and (teste12[i] <=0.50001) then
begin
Chart1.Series[3].AddXY(i,padrao4[i],'',clTeeColor);
if (dec3[i] >= vlv3[i]) then
S3 =1;
Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
end
else
 begin
S3 = 0.5;
Chart1.Series[10].AddXY(i,S3,'',clTeeColor);
end
 end;
end;
```

A resposta da rede ao sinal inserido será a demonstrada na figura 7.12, ou seja, saída S = 1 correspondendo que o sinal pertence à rede, e o próprio padrão correspondente. Apesar do padrão3 [i] ter diferença mínima de amplitude em relação ao padrão4 [i], por causa dos critérios adotados a rede identifica corretamente o padrão correspondente:

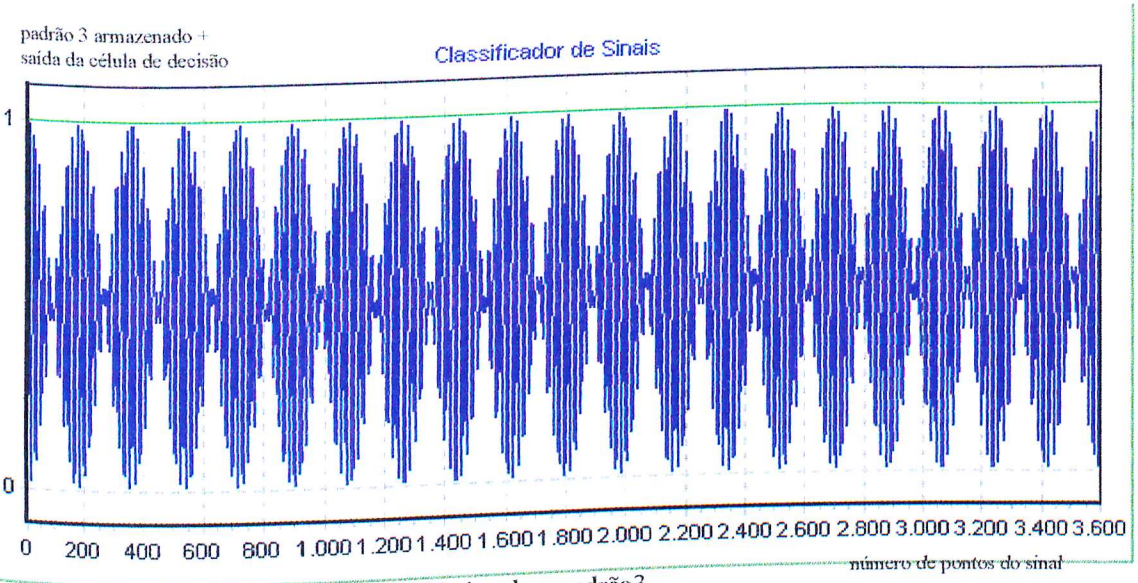


fig. 7.12: Resposta da rede a um sinal inserido igual ao padrão3

Análogamente, o código abaixo após ser processado insere na rede um sinal igual ao padrão4 [i] e verifica se este pertence à rede:

```
procedure TForm1.sinalteste4Click(Sender: TObject);
var
i:Integer; S4:Real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
S4 = 0;
 for i = 0 to 3600 do
  begin
  teste13[i] =((mr2[i]-padrao4[i])+1)/2;
  teste14[i] =((mr4[i]-padrao4[i])+1)/2;
  teste15[i] =((mr6[i]-padrao4[i])+1)/2;
  teste16[i] =((mr8[i]-padrao4[i])+1)/2;


  dec1[i] =(mr2[i]+1)/2;
  dec2[i] =(mr4[i]+1)/2;
  dec3[i] =(mr6[i]+1)/2;
  dec4[i] =(mr8[i]+1)/2;
  vlv4[i] =(1+padrao4[i])/2;

  if (teste13[i] >=0.49999) and (teste13[i] <=0.50001) then
  begin
  Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
  if (dec1[i] >= vlv4[i]) then
  S4=1;
  Chart1.Series[10].AddXY(i,S4,'',clTeeColor);
  end
```

```
else if (teste14[i] >=0.49999) and (teste14[i] <=0.50001) then
begin
Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
if (dec2[i] >= vlv4[i]) then
S4=1;
Chart1.Series[10].AddXY(i,S4,'',clTeeColor);
end

else if (teste15[i] >0.5) and (teste15[i] <0.5)then
begin
Chart1.Series[2].AddXY(i,padrao3[i],'',clTeeColor);
if (dec3[i] >= vlv4[i]) then
S4=1;
Chart1.Series[10].AddXY(i,S4,'',clTeeColor);
end

else if (teste16[i] >=0.49999) and (teste16[i] <=0.50001) then
begin
Chart1.Series[3].AddXY(i,padrao4[i],'',clTeeColor);
if (dec4[i] >= vlv4[i]) then
S4=1;
Chart1.Series[10].AddXY(i,S4,'',clTeeColor);
end

else
 begin
S4=0.5;
Chart1.Series[10].AddXY(i,S4,'',clTeeColor);

 end
 end;
end;
```

A resposta da rede é mostrada na figura 7.13, onde se percebe que a mesma foi capaz de identificar e reproduzir o padrão inserido:

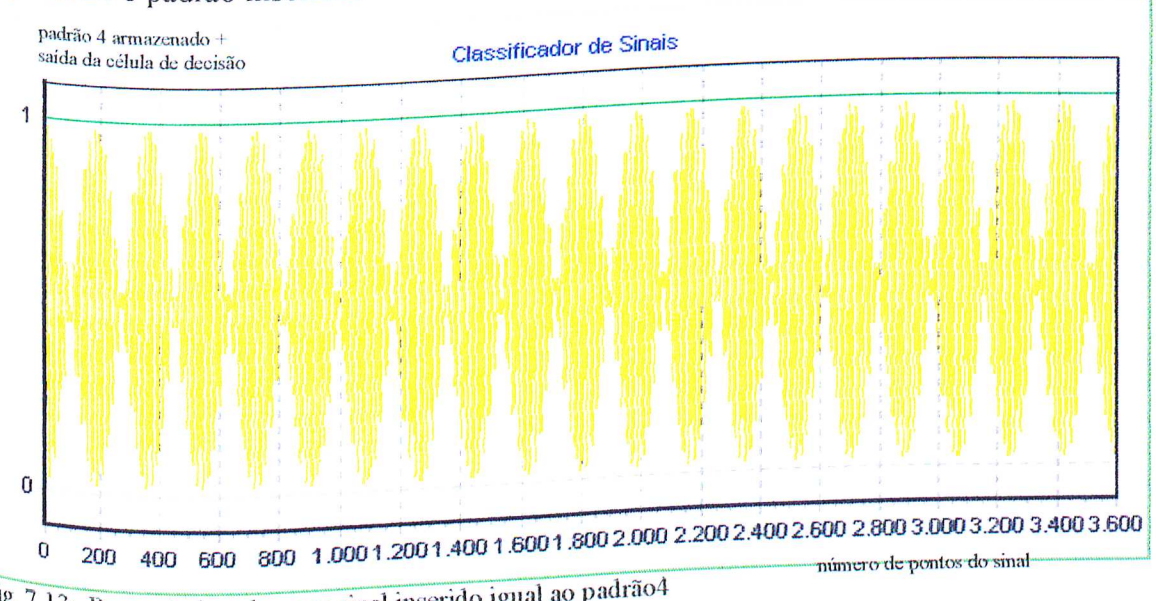


fig. 7.13: Resposta da rede a um sinal inserido igual ao padrão4

7.4.3 Resposta da Rede a Sinais não armazenados:

7.4.3.1 Resposta da Rede a Sinais Periódicos Randômicos não armazenados:

O código a seguir insere e verifica se pertence à rede o sinal **padrão5 [i]**, que difere dos padrões 1 e 2 apenas na freqüência:

```
padrao5[i] = (0.7 + (Sin((1.01*i*Pi)/180))*(-2*Sin((1.01*i*Pi)/180))*(Cos((1.01*i*Pi)/180))/2)*(Random(360)/400);
```
(7.5)

```
procedure TForm1.sinalteste5Click(Sender: TObject);
var
i:Integer; S5:Real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);

 for i = 0 to 3600 do
  begin
  teste17[i] =(mr2[i]-padrao5[i])+1/2;
  teste18[i] =(mr4[i]-padrao5[i])+1/2;
  teste19[i] =(mr6[i]-padrao5[i])+1/2;
  teste20[i] =(mr8[i]-padrao5[i])+1/2;
  dec1[i] =(mr2[i]+1)/2;
  dec2[i] =(mr4[i]+1)/2;
  dec3[i] =(mr6[i]+1)/2;
  dec4[i] =(mr8[i]+1)/2;
  vlv5[i] =(1+padrao5[i])/2;


if (teste17[i] >=0.49999) and (teste17[i] <=0.50001) then
   begin
   Chart1.Series[9].AddXY(i,padrao1[i],'',clTeeColor);
   if (dec1[i] >= vlv5[i]) then
   begin
   S5=1;
   Chart1.Series[10].AddXY(i,S5,'',clTeeColor);
   end;
   end

   else if  (teste18[i] >=0.49999) and (teste18[i] <=0.50001) then
   begin
   Chart1.Series[9].AddXY(i,padrao2[i],'',clTeeColor);
   if (dec2[i] >= vlv5[i]) then
   begin
   S5=1;
   Chart1.Series[10].AddXY(i,S5,'',clTeeColor);
   end;
   end
```

```
else if  (teste19[i] >=0.49999) and (teste19[i] <=0.50001) then
    begin
    Chart1.Series[9].AddXY(i,padrao3[i],'',clTeeColor);
    if (dec3[i] >= vlv5[i]) then
    begin
    S5 =1;
    Chart1.Series[10].AddXY(i,S5,'',clTeeColor);
    end;
    end

     else if  (teste20[i] >=0.49999) and (teste20[i] <=0.50001)then
     begin
     Chart1.Series[9].AddXY(i,padrao4[i],'',clTeeColor);
     if (dec4[i] >= vlv5[i]) then
     begin
     S5 =1;
     Chart1.Series[2].AddXY(i,S5,'',clTeeColor);
     end;
     end

     else
     begin
     S5:=0.5;
     Chart1.Series[11].AddXY(i,S5,'',clTeeColor);
    Chart1.Series[9].AddXY(i,padrao5[i],'',clTeeColor);
     end

  end;
end;
```

Como este sinal não pertence à rede, a capacidade seletiva da mesma proporciona que a resposta seja uma saída S = 0.5, representando uma indeterminação, e com a reprodução do padrão 5, como mostra a figura 7.14:

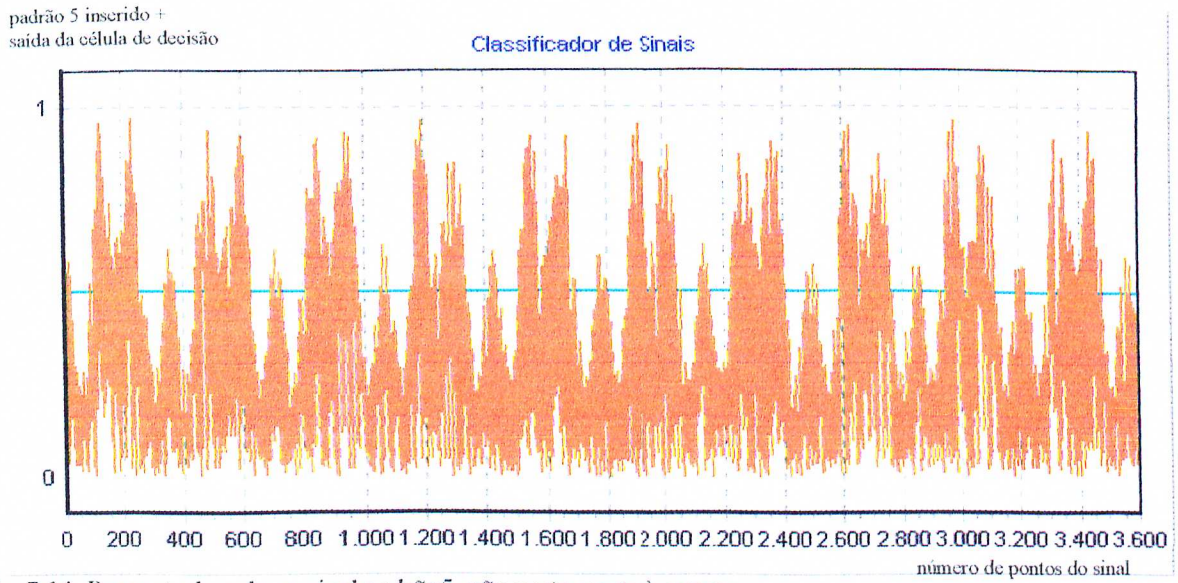


fig. 7.14: Resposta da rede ao sinal padrão5, não pertencente à mesma

Para demonstrar a eficácia da rede, são comparados graficamente os padrões 1 e 2, assim como as diferenças entre o padrão 5 em relação aos padrões 1 e 2; isto é demonstrado abaixo nas figuras 7.15, 7.16 e 7.17:

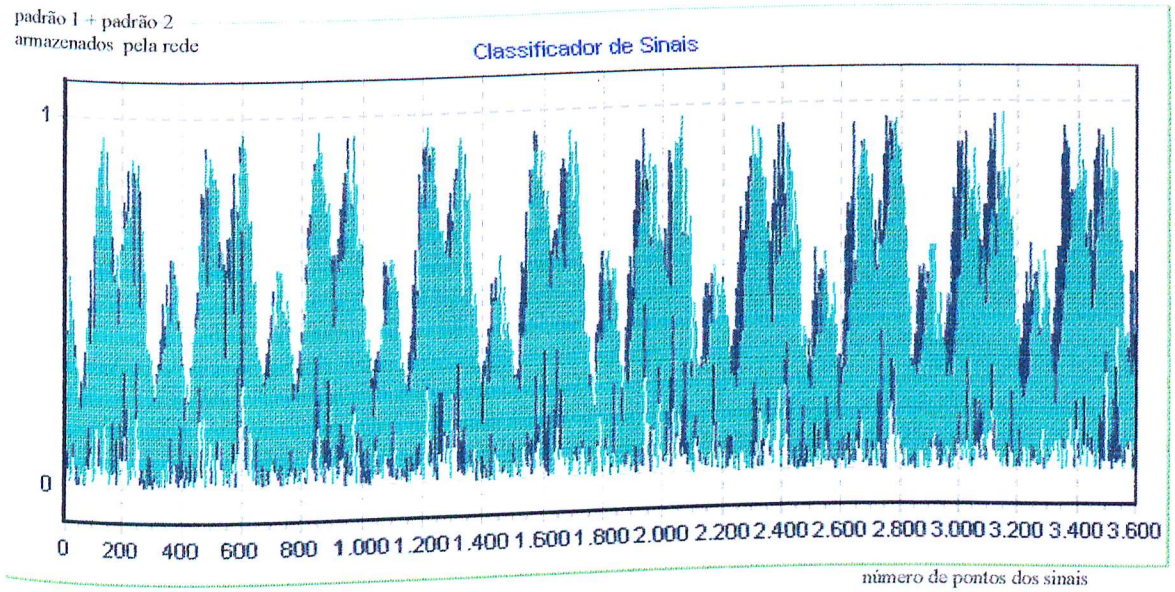


fig. 7.15: padrão 2 sobreposto ao padrão 1 da rede

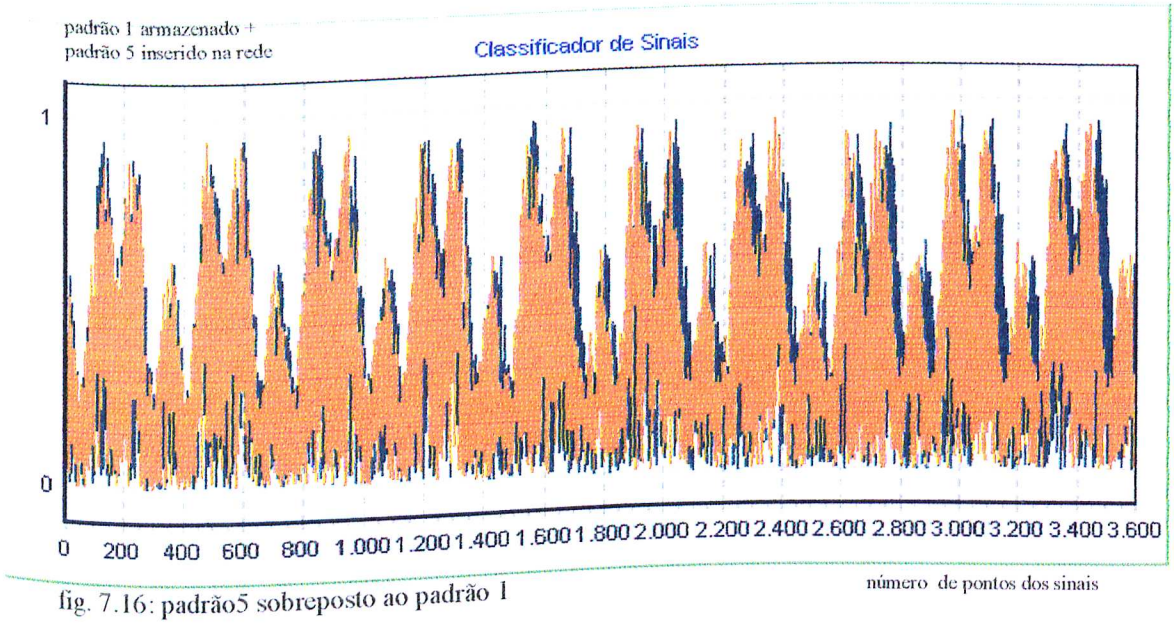


fig. 7.16: padrão5 sobreposto ao padrão 1

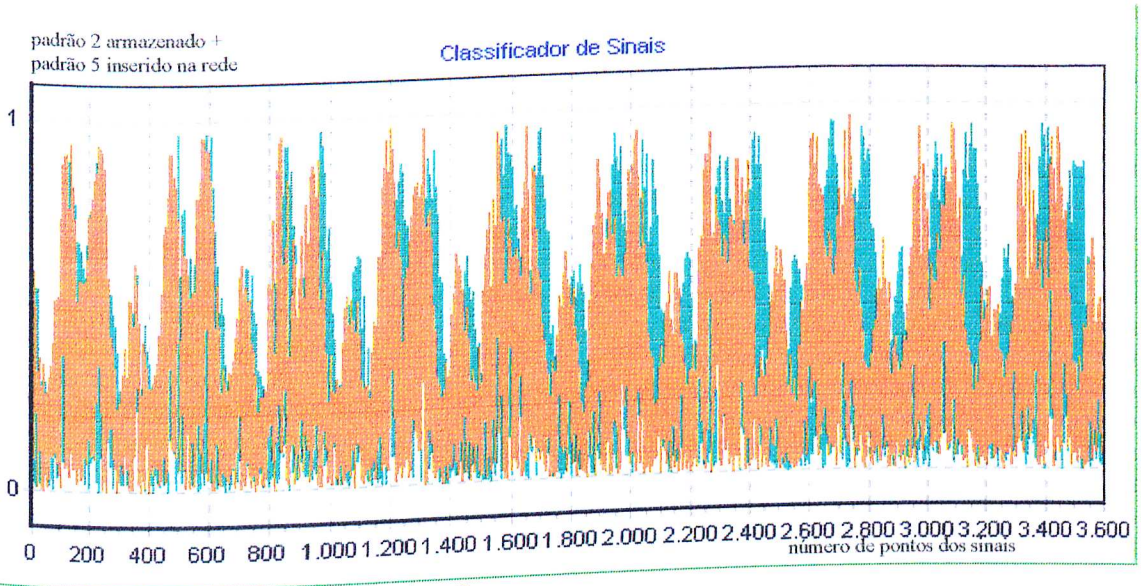


fig. 7.17: padrão 5 sobreposto ao padrão 2

Devido a esta capacidade da rede, pode-se pensar em aproveitar sua capacidade para identificar sinais com estas características, ou seja, sinais bioelétricos, fonemas, etc.

7.4.3.2 Resposta da Rede a Sinais Periódicos não armazenados:

Inserindo agora, através das linhas de código a seguir, um sinal **padrão6 [i]** que difere apenas em amplitude dos padrões 3 e 4:

```
padrao6[i] = 0.5 + (0.97*Sin((20*i*Pi)/180))*(0.97*Cos((i*Pi)/180))/2;
```
(7.6)

```
procedure TForm1.sinalteste6Click(Sender: TObject);
var
i:Integer; S6:Real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
```

```
for i:=0 to 3600 do
 begin

 teste21[i]=((mr2[i]-padrao6[i])+1)/2;
 teste22[i]=((mr4[i]-padrao6[i])+1)/2;
 teste23[i]=((mr6[i]-padrao6[i])+1)/2;
 teste24[i]=((mr8[i]-padrao6[i])+1)/2;
 dec1[i]=(mr2[i]+1)/2;
 dec2[i]=(mr4[i]+1)/2;
 dec3[i]=(mr6[i]+1)/2;
 dec4[i]=(mr8[i]+1)/2;
 vlv6[i]=(1+padrao6[i])/2;

if (teste21[i] >= 0.4999999) and (teste21[i] <= 0.5000001)then
  begin
  Chart1.Series[9].AddXY(i,padrao1[i],'',clTeeColor);
  if (dec1[i] > vlv6[i]) then
  begin
  S6 =1;
  Chart1.Series[10].AddXY(i,S6,'',clTeeColor);
  end;
  end

  else if  (teste22[i] >= 0.49999999) and (teste22[i] <= 0.50000001) then
  begin
  Chart1.Series[9].AddXY(i,padrao2[i],'',clTeeColor);
  if (dec2[i] > vlv6[i]) then
  begin
  S6 =1;
  Chart1.Series[10].AddXY(i,S6,'',clTeeColor);
  end;
  end

  else if  (teste23[i] >= 0.4999) and (teste23[i] <= 0.499999) then
  begin
  Chart1.Series[9].AddXY(i,padrao3[i],'',clTeeColor);
  if (dec3[i] > vlv6[i]) then
  begin
  S6 =1;
  Chart1.Series[10].AddXY(i,S6,'',clTeeColor);
  end;
  end

  else if  (teste24[i] >= 0.499999) and (teste24[i] <= 0.499999) then
  begin
  Chart1.Series[9].AddXY(i,padrao4[i],'',clTeeColor);
  if (dec4[i] > vlv6[i]) then
  begin
  S6 =1;
  Chart1.Series[10].AddXY(i,S6,'',clTeeColor);
  end;
  end
```

```
else
   begin
   S6 = 0.5;
   Chart1.Series[9].AddXY(i,S6,'',clTeeColor);
   Chart1.Series[10].AddXY(i,padrao6[i],'',clTeeColor);
   end
 end;
end;
```

Como o padrão 6 também não pertence à rede, a resposta será uma saída S = 0.5, representando uma indeterminação. A resposta também vem com a reprodução do padrão 6, como mostra a figura 7.18:

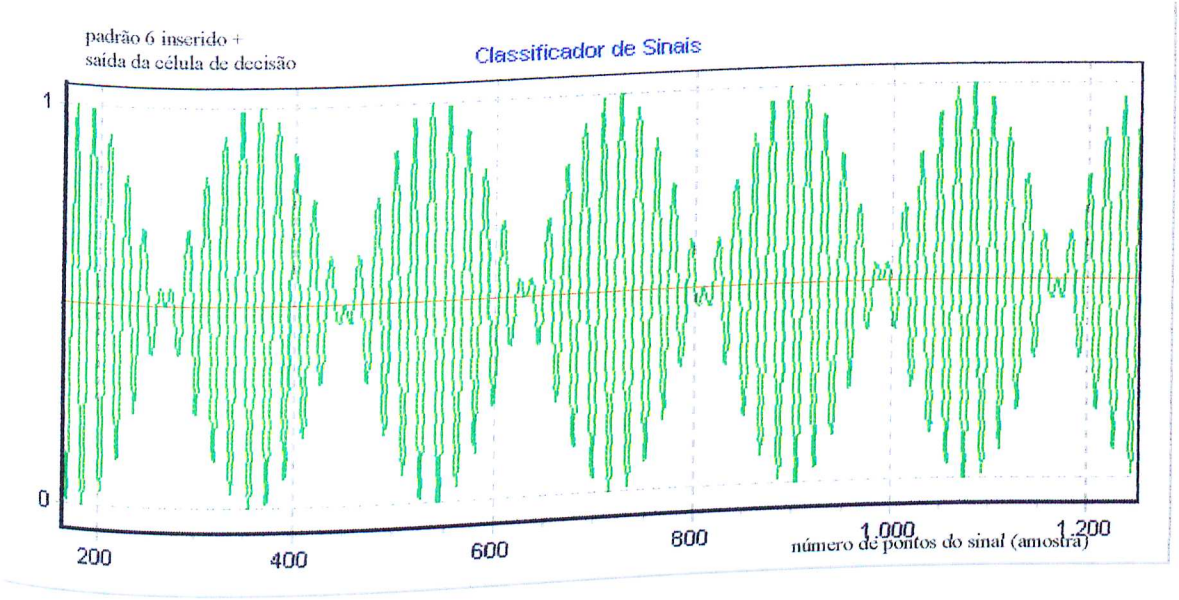


fig. 7.18: Resposta da rede ao sinal padrão 6, não pertencente à mesma

Para demonstrar a eficácia da rede, são comparados graficamente os padrões 3 e 4, assim como as diferenças entre o padrão 6 em relação aos padrões 3 e 4; isto é demonstrado abaixo nas figuras 7.19, 7.20 e 7.21; a rede foi capaz de discriminar os padrões 3, 4 e 6, sobrepostos.

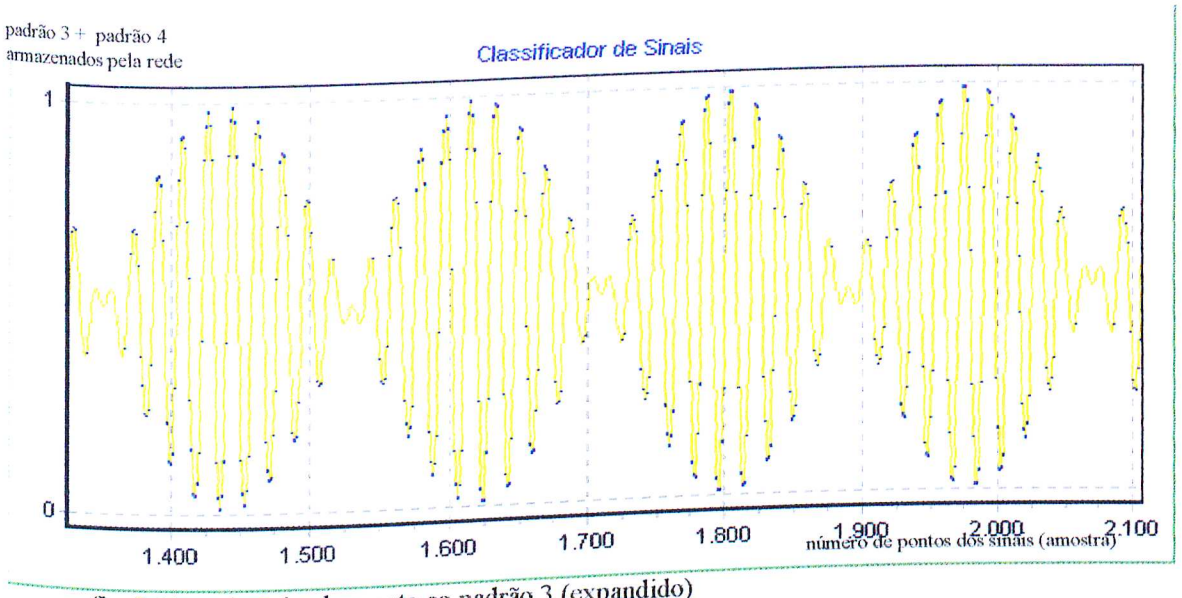


fig. 7.19: padrão 4 sobreposto ao padrão 3 (expandido)

Pode-se verificar que as diferenças de amplitudes são mínimas.

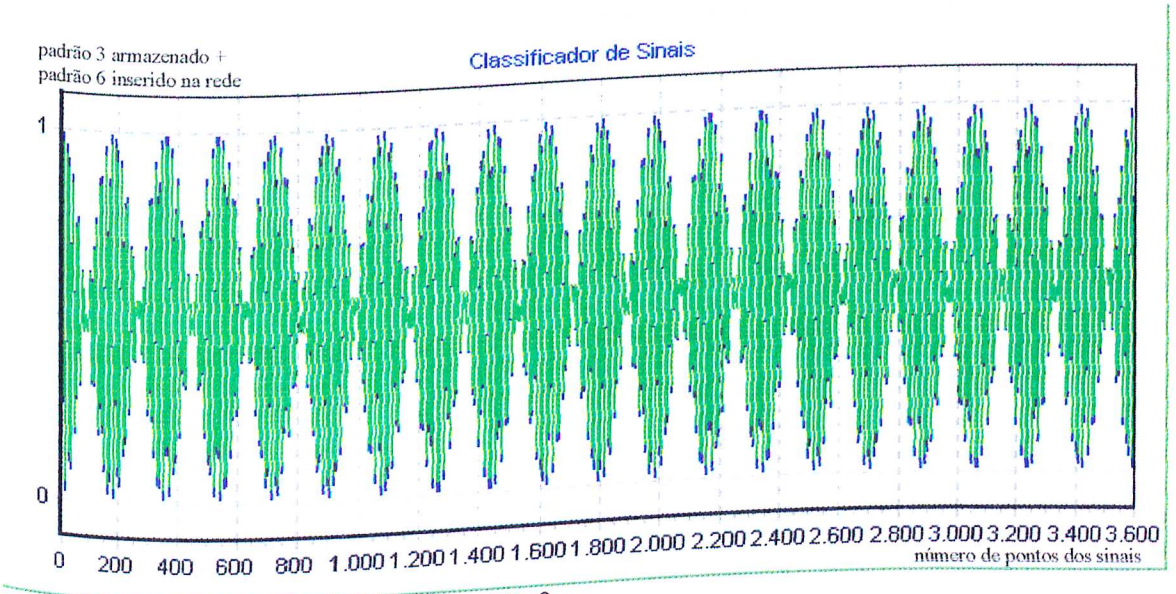


fig. 7.20: padrão 6 sobreposto ao padrão 3

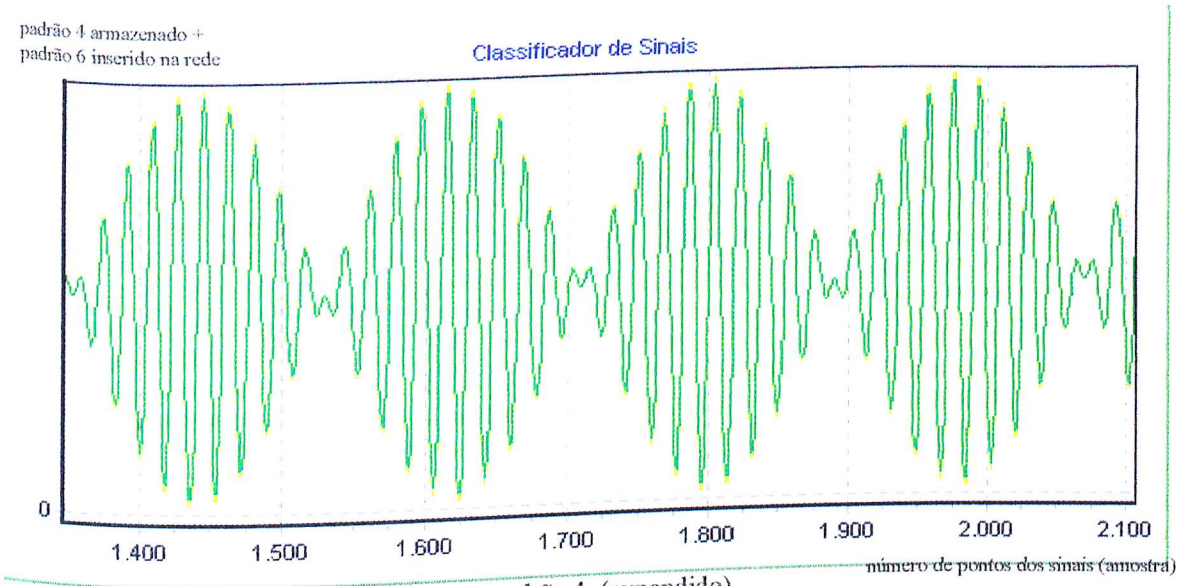


fig. 7.21: padrão 6 sobreposto ao padrão 4 (expandido)

Esta capacidade da rede de discriminar mínimas diferenças de amplitudes pode ser aplicada em sinais originados a partir de sensores ou conversores A/D, por exemplo.

7.5 Considerações sobre a RNAP para classificação de sinais

As propriedades das Unidades Neurais Artificiais Paraconsistentes de Comparação de Padrões - UNAPCP modelam o comportamento da RNAP para classificação de sinais, ou seja, como as saídas de todas as unidades são concorrentes e convergem para somente duas saídas na rede, que representam a saída de uma célula de decisão e de uma célula de conexão lógica simples. Prevalecerá na saída da rede, a saída da Unidade Neural que armazenar o padrão que mais se assemelhe ao padrão inserido na rede para comparação. Caso nenhuma unidade esteja próxima do padrão inserido, a saída da rede irá apresentar o padrão inserido e o valor meio referente à resposta negativa da células de decisão.

CAPÍTULO 8
Sistema Paraconsistente Classificador de Sinais por Famílias

CAPÍTULO 8

Sistema Paraconsistente Classificador de Sinais por Famílias

Utilizando as propriedades das UNAPs, foram implementados algoritmos capazes de identificar determinado sinal como tendo certas características de uma família de sinais, e portanto, classificá-lo como pertencente a esta família.

8.1 Utilização da Rede para classificar um sinal pertencente a uma família de sinais com características semelhantes

O objetivo enfocado até o momento foi o de discriminar sinais pertencentes ou não à rede, porém a rede também pode verificar se um sinal, mesmo não sendo exatamente igual aos armazenados, tenha características de uma família de sinais que façam parte do padrão da rede. Tal característica pode ser aproveitada, por exemplo, quando se faz reconhecimento de sinais através de uma OCR (Optical Caracter Recognition), onde um caracter escrito tem uma série de características que o definem como determinada letra de um alfabeto. No caso da rede aqui estudada pode-se, a partir de pequenas mudanças, diminuir a faixa de precisão nos testes efetuados nas células de Conexão Lógica Seletiva de Maximização, que correspondem ao grau de crença da mesma. Nesta célula foi verificado que quando um sinal testado era igual ao sinal armazenado e aprendido pela rede, este grau de crença tinha um valor próximo a 0,5. Se a faixa em torno desse valor for extendida, haverá maior tolerância em identificar um sinal como sendo próximo aos que estão armazenados. Como exemplo, tomamos a função **familia1**, que verifica se o sinal **padrao5** pertence à rede. Como a faixa agora foi aumentada, a resposta da rede será agora com a saída = 1.0 e mostrando também o padrao5, indicando que ele tem características dos padrões 1 e 2. No gráfico de saída, foram adicionados os sinais que têm características semelhantes ao padrão 5 (no caso os padrões 1 e 2), para que se possa identificar melhor a que família de sinais este pertence. O procedimento referente à função **familia1** é mostrado a seguir:

```
procedure TForm1.familia1Click(Sender: TObject);
var
i:Integer; S9:Real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
for i:=0 to 3600 do
  begin
  teste25[i]:=(mr2[i]-padrao5[i])+1/2;
```

```
teste26[i]:=(mr4[i]-padrao5[i])+1/2;
teste27[i]:=(mr6[i]-padrao5[i])+1/2;
teste28[i]:=(mr8[i]-padrao5[i])+1/2;
dec1[i]:=(mr2[i]+1)/2;
dec2[i]:=(mr4[i]+1)/2;
dec3[i]:=(mr6[i]+1)/2;
dec4[i]:=(mr8[i]+1)/2;
m3[i]:=dec1[i]-vlv7[i];
m4[i]:=dec2[i]-vlv7[i];
vlv7[i]:=(1+padrao5[i])/2;

 if (teste25[i] >= -0.1) and (teste25[i] <= 0.9999999) then
 begin
 Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
 if (m3[i] >= 0.5) and (m3[i] <= 0.99)then
 begin
 S9:=1;
 Chart1.Series[10].AddXY(i,S9,'',clTeeColor);
 Chart1.Series[9].AddXY(i,padrao5[i],'',clTeeColor);
 end;
end

 else if  (teste26[i] >= -0.4) and (teste26[i] <= 1.5) then
 begin
 Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
 if (m4[i] >= 0.5) and (m4[i] <= 0.99) then
 begin
 S9:=1;
 Chart1.Series[10].AddXY(i,S9,'',clTeeColor);
 Chart1.Series[9].AddXY(i,padrao5[i],'',clTeeColor);
 end;
 end
 else if  (teste27[i] >=0.49999) and (teste27[i] <=0.50001) then
 begin
 Chart1.Series[9].AddXY(i,padrao3[i],'',clTeeColor);
 if (dec3[i] >= vlv7[i]) then
 begin
 S9:=1;
 Chart1.Series[10].AddXY(i,S9,'',clTeeColor);
 end;
 end

  else if  (teste28[i] >=0.49999) and (teste28[i] <=0.50001)then
  begin
  Chart1.Series[9].AddXY(i,padrao4[i],'',clTeeColor);
  if (dec4[i] >= vlv7[i]) then
  begin
  S9:=1;
  Chart1.Series[2].AddXY(i,S9,'',clTeeColor);
  end;
  end

  else
  begin
```

```
    S9:=0.5;
    Chart1.Series[8].AddXY(i,S9,'',clTeeColor);
    Chart1.Series[9].AddXY(i,padrao5[i],'',clTeeColor);
    end

  end;
end;
```

Na figura 8.1 tem-se a resposta da rede depois de adaptada para reconhecer o padrão 5 como sendo da família de sinais dos padrões 1 e 2; aparecem o padrão 5, traços dos padrões 1 e 2, assim como a saída S = 1.0.

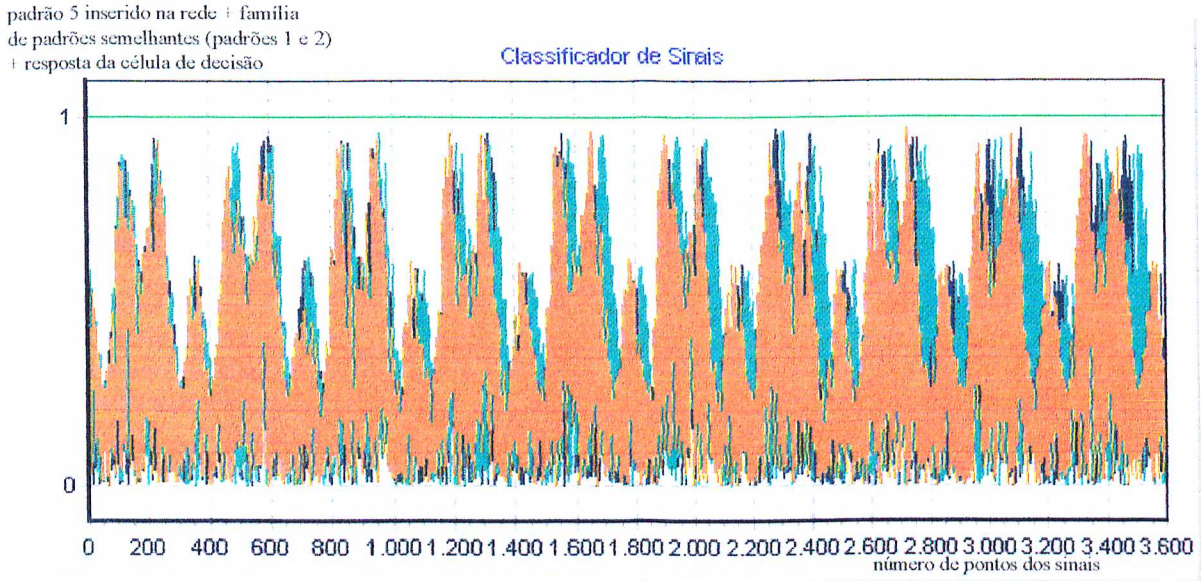


fig. 8.1: resposta da rede readptada para considerar o padrão5 como sendo da família dos padrões 1 e 2

Utilizando a mesma técnica, isto é, aumentando-se a faixa de tolerância dos testes 31 e 32 que correspondem aos graus de crença das células de conexão lógica seletiva de maximização, esta por sua vez, verifica se o sinal **padrao6** está dentro das características dos padrões aprendidos pela rede. Desta forma tem-se que a rede pode ser adaptada para identificá-lo como sendo pertencente a família de padrões 3 e 4. A função **família2** compara o padrão 6 com os padrões armazenados na rede, como demonstrado a seguir:

```
procedure TForm1.familia2Click(Sender: TObject);
var
i:Integer; S10:Real;
```

```
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
 for i:=0 to 3600 do
  begin
  teste29[i]:=((mr2[i]-padrao6[i])+1)/2;
  teste30[i]:=((mr4[i]-padrao6[i])+1)/2;
  teste31[i]:=((mr6[i]-padrao6[i])+1)/2;
  teste32[i]:=((mr8[i]-padrao6[i])+1)/2;
  dec1[i]:=(mr2[i]+1)/2;
  dec2[i]:=(mr4[i]+1)/2;
  dec3[i]:=(mr6[i]+1)/2;
  dec4[i]:=(mr8[i]+1)/2;
  vlv8[i]:=(1+padrao6[i])/2;
  m1[i]:=dec3[i]-vlv8[i];
  m2[i]:=dec4[i]-vlv8[i];

   if (teste29[i] >= 0.49999) and (teste29[i] <= 0.500001)then
  begin
  Chart1.Series[0].AddXY(i,padrao1[i],'',clTeeColor);
  if (dec1[i] > vlv8[i]) then
  begin
  S10:=1;
  Chart1.Series[7].AddXY(i,S10,'',clTeeColor);
  end;
  end

  else if  (teste30[i] >= 0.49999999) and (teste30[i] <= 0.50000001) then
  begin

  Chart1.Series[1].AddXY(i,padrao2[i],'',clTeeColor);
  if (dec2[i] > vlv8[i]) then
  begin
  S10:=1;
  Chart1.Series[7].AddXY(i,S10,'',clTeeColor);
  end;
  end

   else if  (teste31[i] >= 0.49) and (teste31[i] <= 0.501) then
  begin
  Chart1.Series[2].AddXY(i,padrao3[i],'',clTeeColor);
  if ( m2[i]>= -0.3 )and (m2[i] <= 0.2) then
  begin
  S10:=1;
  Chart1.Series[7].AddXY(i,S10,'',clTeeColor);
  Chart1.Series[10].AddXY(i,padrao6[i],'',clTeeColor);
  end;
  end

   else if  (teste32[i] >= 0.49) and (teste32[i] <= 0.51) then
   begin
```

```
   Chart1.Series[3].AddXY(i,padrao4[i],'',clTeeColor);
   if ( m2[i]>= -0.3 )and (m2[i] <= 0.2) then
   begin
   S10:=1;
   Chart1.Series[9].AddXY(i,S10,'',clTeeColor);
   Chart1.Series[10].AddXY(i,padrao6[i],'',clTeeColor);
   end;
   end

   else
   begin
   S10:=0.5;
   Chart1.Series[9].AddXY(i,S10,'',clTeeColor);
   Chart1.Series[10].AddXY(i,padrao6[i],'',clTeeColor);
   end
 end;
end;
```

A resposta da rede será então como demonstrado na figura 9.2, onde se observa que aparecem na figura o padrão 6 e traços dos padrões 3 e 4, assim como a saída S = 1.0, correspondendo ao grau de certeza máximo.

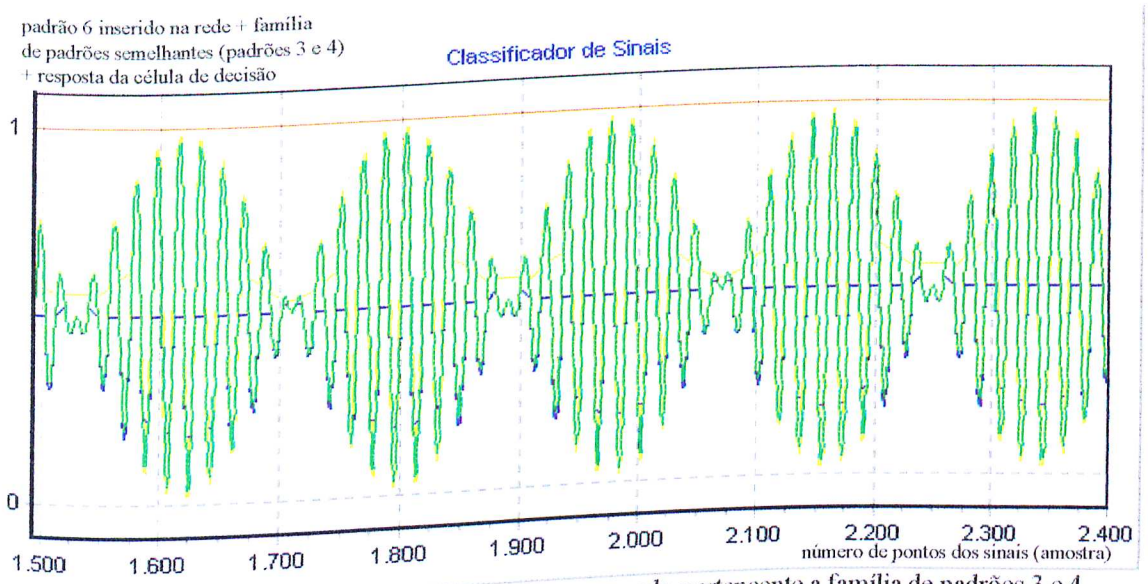


fig. 8.2: resposta da rede ao padrão 6, reconhecendo-o como sendo pertencente a família de padrões 3 e 4.

8.2 Utilização da Rede para classificar um sinal em uma família de sinais através da Célula Neural Artificial Paraconsistente de Conexão Lógica Analítica – CNAPCa:

Para que seja possível fazer uma classificação em famílias de características similares, foi inserido ao sistema a Célula Neural Artificial Paraconsistente de Conexão Lógica Analítica – CNAPCa, mostrada a seguir:

8.2.1. A Célula Neural Artificial Paraconsistente de Conexão Lógica Analítica – CNAPCa:

A Célula Neural Artificial Paraconsistente de Conexão Lógica Analítica tem a função de fazer a interligação entre células da Rede Neural Artificial Paraconsistente, associando graus de crença conforme os objetivos da análise. A célula de conexão é o elo de ligação que permite diferentes regiões de uma Rede Neural Artificial Paraconsistente efetuar o processamento de sinais de forma distribuída por meio de inúmeras conexões paralelas [94]. A figura 9.9 mostra o esquema da **CNAPCa**:

CNAPCa:

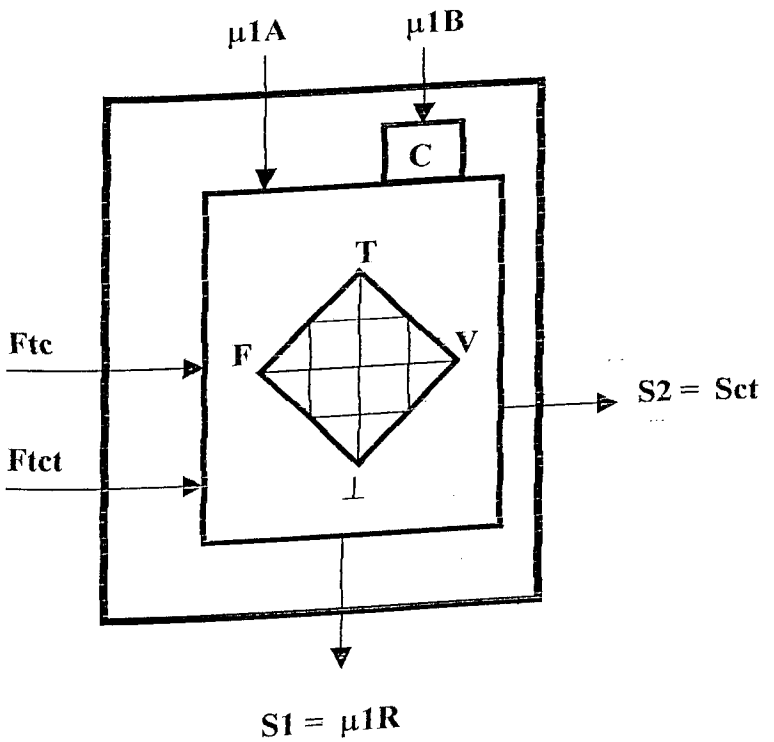


fig. 8.3: Célula Neural Artificial Paraconsistente de Conexão Lógica Analítica – CNAPCa:

O algoritmo da CNAPCa é mostrado a seguir:

Ftc = C1 - Fator de tolerância à certeza: $0 \leq C1 \leq 1$.
Ftct = C2 – Fator de tolerância à contradição: $0 \leq C2 \leq 1$.
$\mu 1A$ = Grau de crença de entrada: $0 \leq \mu 1 \leq 1$.
$\mu 1B$ = Grau de descrença de entrada: $0 \leq \mu 2 \leq 1$.

são calculados:

$\mu 1BC = 1 - \mu 1B$: complemento do grau de descrença (8.1)
$| Gct | = | \mu 1 + \mu 2C - 1 |$: valor do grau de contradição (8.2)
$| Gc | = | \mu 1 - \mu 2C |$: valor do grau de certeza (8.3)
Vicc = (1 – C1) / 2: valor do limite inferior de certeza (8.4)
Vscc = (1 + C1) / 2 : valor do limite superior de certeza (8.5)
$\mu 1r = (\mu 1A - \mu 1BC + 1) / 2$: valor do grau de crença resultante = EEB (8.6)

saídas:

se Vicc $\leq \mu 1r \leq$ Vscc : S1 = $\mu 1r$ e S2 = 0;
se $| Gct | \geq C2$ e $| Gct | \geq | Gc |$: S1 = ½ e S2 = $| Gct |$;
senão: S1 = ½ e S2 = 0;

8.2.2 Utilização da CNAPCa para classificar sinais:

Neste trabalho para que a célula de conexão lógica analítica classifique os sinais, utilizou-se como grau de crença de entrada a média dos padrões que constituem a família sob a qual se quer classificar um sinal inserido na rede. Como grau de descrença foi utilizado o valor um, o que dá continuidade à filosofia determinística da rede. A equação 9.7 mostra o cálculo do grau de crença resultante:

$$\mathbf{mr\ [i] = (((padraox\ [i] + padraoy\ [i]) / 2) - 0 + 1) / 2} \qquad (8.7)$$

A seguir são calculados os valores inferior (8.8) e superior (8.9) do valor limite de certeza; para tanto foi utilizado como fator de tolerância à certeza, o padrão a ser testado:

$$\mathbf{vicc\ [i] = (1 - padraoz\ [i]) / 2} \qquad (8.8)$$

$$\mathbf{vscc\ [i] = (1 + padraoz\ [i]) / 2} \qquad (8.9)$$

Em seguida são efetuados então os testes para se determinar as saídas: no primeiro teste é verificado se o grau de crença está entre os limites inferior e superior do valor limite de certeza. Em caso positivo, a saída será o grau de crença (que possui características da média dos padrões armazenados) e a saída S será zero (S = 0), indicando grau de contradição nulo. Estes testes são demonstrados a seguir:

```
if ( mr [i] > vicc [i] ) then
   begin
   if ( mr [i] < vscc [i] )then
    begin
    Chart1.Series[0].AddXY(i,mr[i],'',clTeeColor);
    S2:=0;
    Chart1.Series[9].AddXY(i,S2,'',clTeeColor);
   end
```

Se esta condição não for verdadeira, são testadas então as possibilidades do grau de crença ser menor que o grau de contradição e do grau de contradição ser maior que o fator de tolerância à contradição, que no caso assumirá o valor zero:

```
else if (((padraox[i]+padraoy[i])/2)<(((padraox[i]+padraoy[i])/2)-1)) and(((padraox[i]+padraoy[i])-1)>0)
then
  // grau de crença < grau de contradição   e    grau de contradição > fator tolerância à contradição

                begin
                 S1=0.5;                                                    ( 8. 10 )
                 S2=mr11[i]=(((padraox[i]+padraoy[i])/2)-1);
                 Chart1.Series[8].AddXY(i,S1,'',clTeeColor);
                 Chart1.Series[10].AddXY(i,mr11[i],'',clTeeColor);
                end
               end
// S2 = grau de contradição
```

Se nenhuma das condições anteriores for satisfeita, as saídas serão ½ e 0 respectivamente:

```
              else
               begin
               S1=0.5;
               S2[i] =0;
               Chart1.Series[8].AddXY(i,S1,'',clTeeColor);
                Chart1.Series[10].AddXY(i,S2[i],'',clTeeColor);
                end
              end;
            end;
```

8.3 Unidade Neural Artificial Paraconsistente de Classificação de Famílias - UNAPCF

Com a inclusão de uma unidade chamada de Unidade Neural Artificial Paraconsistente de Classificação de famílias, composta por duas células de conexão lógica analítica, estará se ampliando a capacidade da Rede adicionando a esta a UNAPCF, que alternativamente e simultaneamente poderia estar classificando e discriminando sinais. Inicialmente optou-se por fazer ensaios nesta unidade separadamente. Como exemplo esta unidade foi capacitada para analizar os padrões 5 e 6, respectivamente com as famílias de padrões 1/2 e 3/4. A figura 8.4 mostra esta unidade:

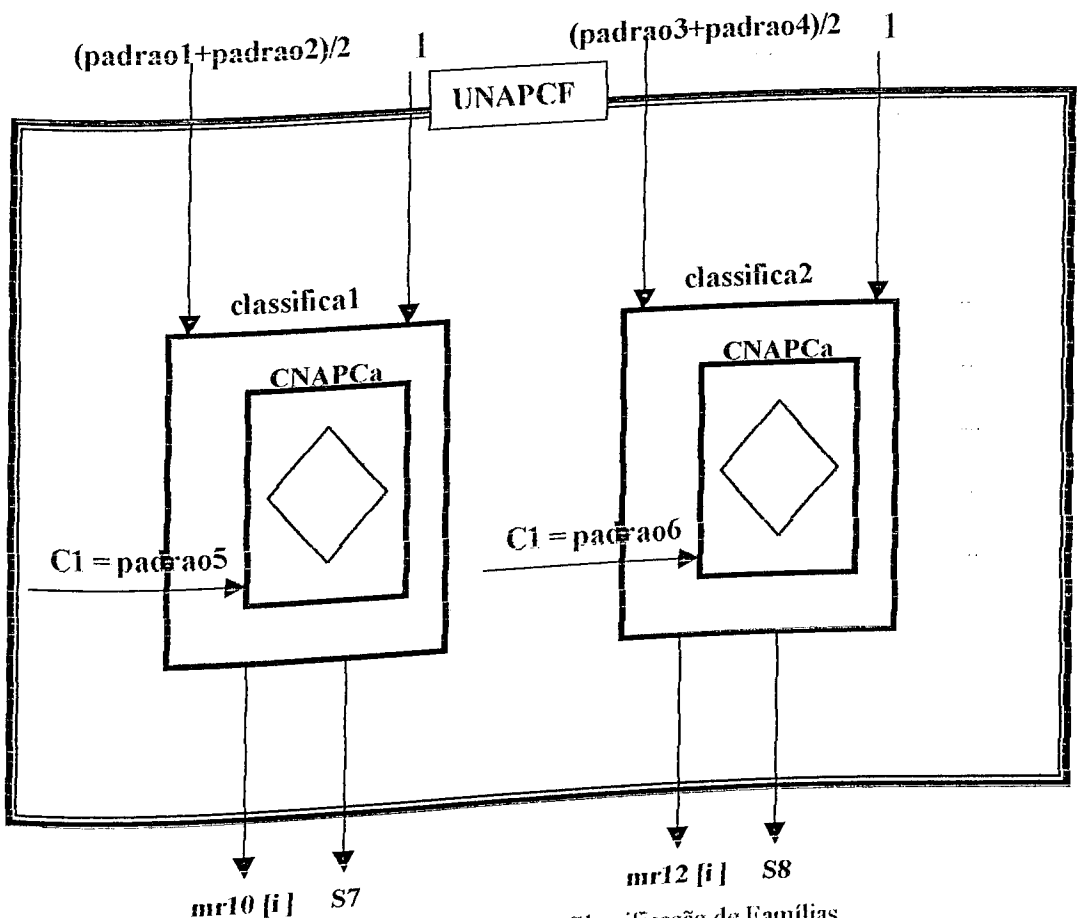


fig. 9.4: Unidade Neural Artificial Paraconsistente de Classificação de Famílias

De acordo com o que foi explanado no item 8.2.2, como C1 = padrão de teste inserido, então é quase certo que o grau de crença calculado (na verdade é a metade da média dos padrões da mesma família), estará entre o valor limite inferior e limite superior de verdade (que são calculados em função do padrão a ser testado), proporcionando uma saída mr [i] igual ao grau de crença e S = 0.

Simulando o funcionamento da UNAPCF, foram geradas duas funções, **classifica1** e **classifica2**, que testam respectivamente os padrões 5 (em relação à família dos padrões 1 e 2) e 6 (em relação à família dos padrões 3 e 4).

A seguir é mostrado o algoritmo da função **classifica1**:

```
procedure TForm1.classifica1Click(Sender: TObject);
var
i:Integer; S7: real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
 for i:=0 to 3600 do

begin
  mr10[i]:=(((padrao1[i]+padrao2[i])/2)-0+1)/2;//grau de crença = média dos padrões
  vicc1[i]:=(1-padrao5[i])/2;
  vscc1[i]:=(1+padrao5[i])/2;
   if (mr10[i] > vicc1[i]) then
   begin
    if (mr10[i] < vscc1[i])then
    begin
    Chart1.Series[0].AddXY(i,mr10[i],'',clTeeColor);
    S7:=0;
    Chart1.Series[9].AddXY(i,S7,'',clTeeColor);
    end
    else if (((padrao1[i]+padrao2[i])/2)<(((padrao1[i]+padrao2[i])/2)-1)) and((((padrao1[i]+padrao2[i])/2)-1)>0) then
  //grau de crença < grau de contradição  e  grau de contradição > fator tolerância à contradição
    begin
    S7:=0.5;
    mr11[i]:=(((padrao1[i]+padrao2[i])/2)-1);
    Chart1.Series[8].AddXY(i,S7,'',clTeeColor);
    Chart1.Series[10].AddXY(i,mr11[i],'',clTeeColor);
    end
   end
    else
    begin
    S7:=0.5;
    mr10[i]:=0;
    Chart1.Series[8].AddXY(i,S7,'',clTeeColor);
    Chart1.Series[10].AddXY(i,mr11[i],'',clTeeColor);
    end
  end;
end;
```

A resposta da rede é mostrada na figura 8.5:

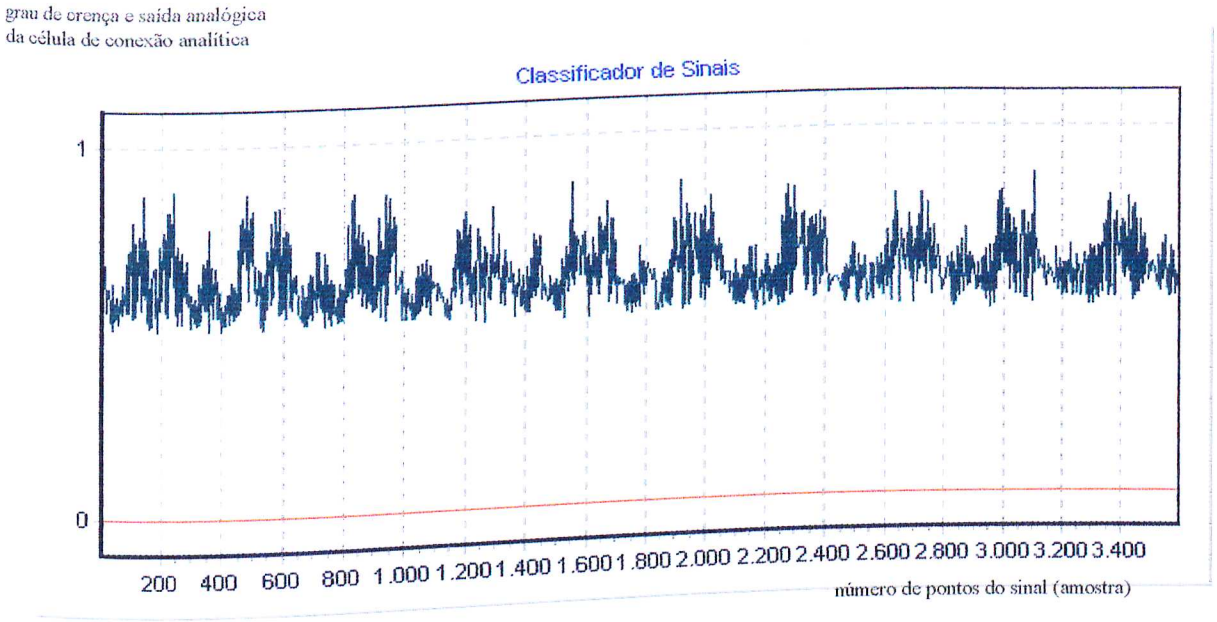


fig. 9.5: resposta da UNAPCF à verificação do padrão 5 em relação aos padrões 1 e 2.

Algoritmo da função **classifica2:**

```
procedure TForm1.classifica2Click(Sender: TObject);
var
i:Integer; S8: real;
begin
armazenasinais(Sender);
sinal01Click(Sender);
sinal02Click(Sender);
sinal03Click(Sender);
sinal04Click(Sender);
for i:=0 to 3600 do

 begin

  mr12[i]:=(((padrao3[i]+padrao4[i])/2)-0+1)/2;//grau de crença = média dos padrões
  vicc2[i]:=(1-padrao6[i])/2;        //grau de descrença = 0
  vscc2[i]:=(1+padrao6[i])/2;

  if (mr12[i] >= vicc2[i]) then
  begin
   if  (mr12[i] <= vscc2[i])then
   begin
   Chart1.Series[7].AddXY(i,mr12[i],'',clTeeColor);
   S8:=0;
   Chart1.Series[5].AddXY(i,S8,'',clTeeColor);
   end
   else if (((padrao3[i]+padrao4[i])/2)<(((padrao3[i]+padrao4[i])/2)-1)) and(((padrao3[i]+padrao4[i])/2)-1) > 0) then
//grau de crença < grau de contradição e grau de contradição > fator tolerância à contradição
```

```
begin
    S8:=0.5;
    mr13[i]:=(((padrao3[i]+padrao4[i])/2)-1);
    Chart1.Series[8].AddXY(i,S8,'',clTeeColor);
    Chart1.Series[10].AddXY(i,mr13[i],'',clTeeColor);
    end
   end
    else
    begin
    S8:=0.5;
    mr12[i]:=0;
    Chart1.Series[8].AddXY(i,S8,'',clTeeColor);
    Chart1.Series[10].AddXY(i,mr13[i],'',clTeeColor);
    end
  end;
end;

end.
```

A resposta da rede é mostrada na figura 8.6:

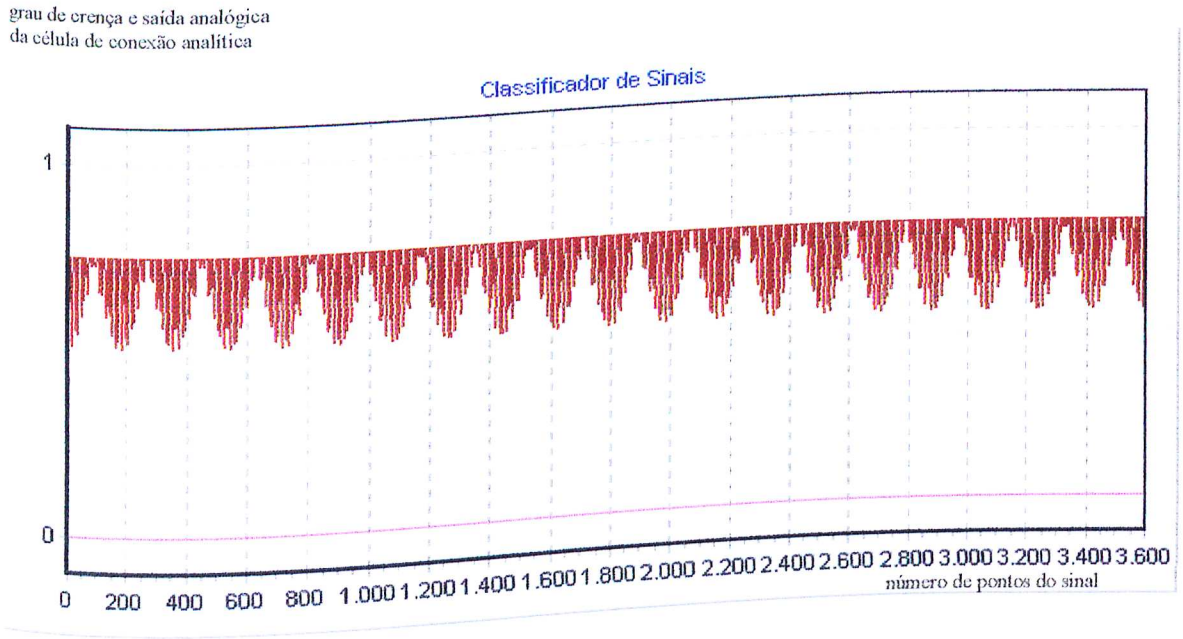


fig. 8.6: resposta da UNAPCF à verificação do padrão 6 em relação aos padrões 3 e 4.

8.3.1. Inclusão da Unidade Neural Artificial Paraconsistente de Classificação de Famílias na Rede Neural Artificial Paraconsistente de Classificação de Sinais:

Em comparação com a estrutura da rede mostrada na figura 7.1, a nova estrutura apenas inclui a **UNAPCF**, sem que hajam alterações do funcionamento das outras unidades neurais de comparação. A nova estrutura da rede é mostrada na figura 8.7:

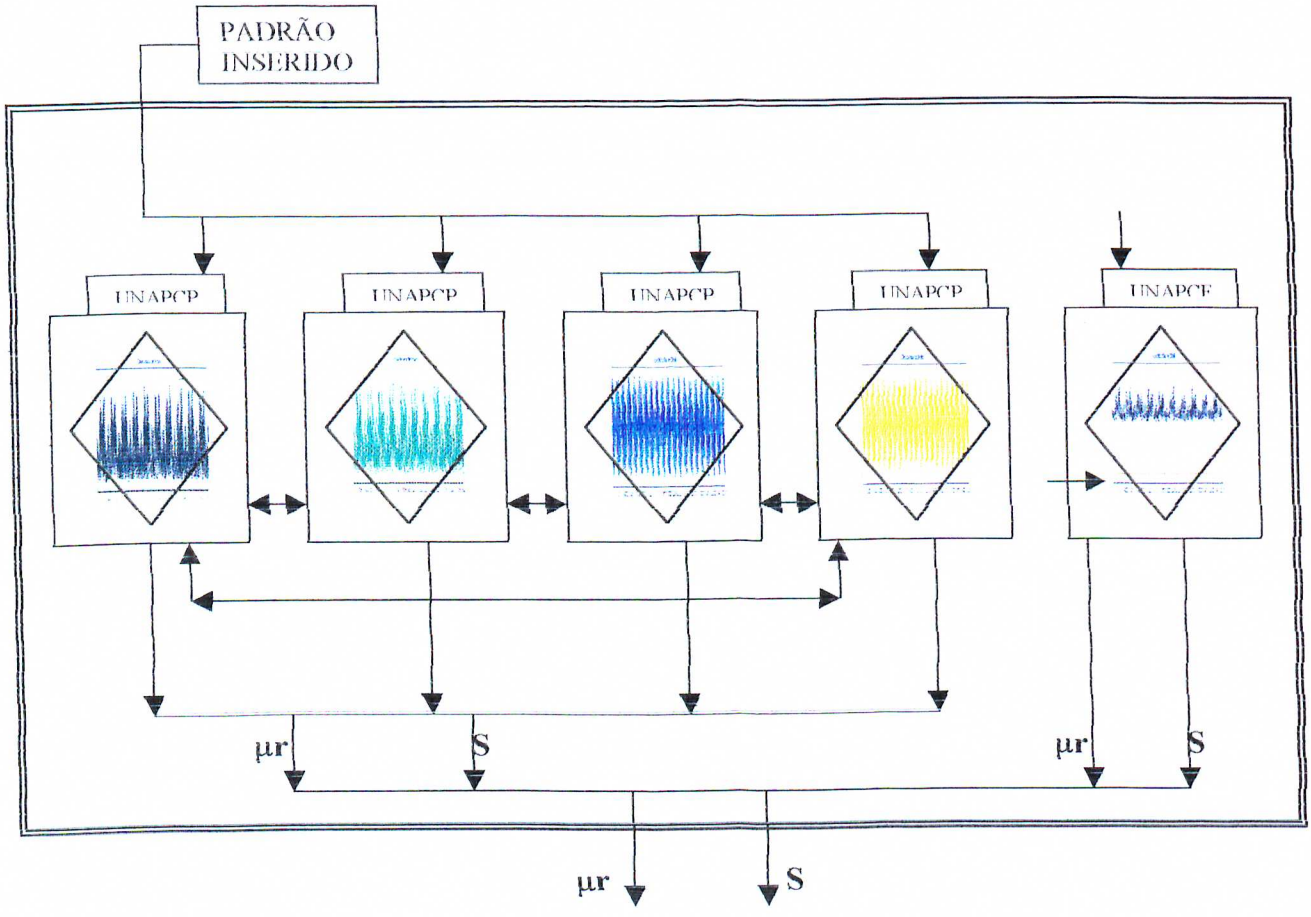


fig. 8.7: Rede Neural Artificial Paraconsistente de Comparação de Padrões com a inclusão da UNAPCF.

8.4 Considerações sobre a classificação de sinais de acordo com a semelhança entre as famílias de sinais:

A classificação por família de sinais pode ser efetuada como no capítulo 7, através das unidades neurais de comparação de padrões, mas com uma parametrização com uma faixa maior que em torno de **meio** nas funções **teste [ ]**, sendo neste caso a parametrização é feita de acordo com a aplicação da classificação de sinais. A classificação pode também ser efetuada pelas unidades neurais para classificação de famílias, e, neste caso, o critério para parametrização está diretamente ligado às características das próprias famílias, pois estas características servem agora como grau de crença de uma célula de conexão analítica, em comparação com o fator de tolerância à certeza, composto agora pela matriz que compões o sinal inserido na rede para comparação.

CAPÍTULO 9

Conclusões

CAPÍTULO 9

Conclusões

Neste trabalho, assim como ficou demonstrado nos capítulos anteriores, foram obtidos relevantes resultados que procura-se listar através das conclusões abaixo:

9.1 Dependência da aprendizagem de padrões com o fator de aprendizagem Fa

A partir dos resultados obtidos no capítulo 4, verificou-se que em uma célula de aprendizagem-CNAPa, quanto menor o fator de aprendizagem menos passos são requeridos para se chegar à estabilização de um padrão na saída.

Verificou-se também que quanto mais próximo de 1.0 estiver o valor de Fa, mais o padrão estabilizado na saída se aproximará do padrão de entrada.

Também verificou-se que a eficácia da célula de aprendizagem é maior quando o padrão de entrada se aproxima de 1.0.

9.2 Dependência da aproximação funcional com o número de passos e com o fator de aprendizagem

A partir dos resultados obtidos no capítulo 5, verificou-se que o número de passos suficiente para uma aproximação funcional ou reprodução de sinais com máxima precisão é a partir de 15. Nos experimentos efetuados foram utilizados 20 passos para se garantir total precisão, com fator de aprendizagem Fa = 1.0. Com a célula de aprendizagem ajustada desta forma, o funcionamento da mesma para reprodução de sinais teve maior eficácia.

Ainda no capítulo 6 foram efetuadas aproximações funcionais com apenas um passo (para Fa = 1.0), e os resultados obtidos foram bem razoáveis. Os resultados encontrados nos permite afirmar que comparados a outras técnicas de aproximação funcional utilizando redes neurais (Backpropagation), a Rede Neural Artificial Paraconsistente apresenta boa precisão.

9.3 Resposta da rede a sinais periódicos randômicos e somente periódicos

Conforme visto no capítulo 8, a rede foi capaz de discriminar os padrões periódicos com características randômicas ou simplesmente periódicos, apresentando eficácia na discriminação, o que nos permite identificar sinais com características aleatórias como sinais bioelétricos, ou então sinais com pequenas diferenças de amplitudes, como os manipulados por processadores digitais de sinais – DSPs.

9.4 Considerações Finais

O enfoque do trabalho foi o de implementar uma rede capaz de discriminar sinais com relativa precisão, diferenciando-os uns dos outros a partir de todas as suas características próprias, como frequência, amplitude e fase. Porém, como foi demonstrado ainda neste capítulo, a rede pode ser alterada para ao invés de discriminar, ser capaz de classificar um sinal e considerá-lo como sendo pertencente a uma família de sinais com características apenas semelhantes. Esta capacidade de discriminar e classificar sinais pode diversificar o seu uso para aplicações de reconhecimento de caracteres ou de imagens, por exemplo.

Este trabalho tem como objetivo principal identificar novas formas de aplicações possíveis com as Redes Neurais Artificiais Paraconsistentes, para que possam servir como alternativa a outros métodos convencionais já utilizados. Neste caso, o enfoque dos estudos foi para aproximações funcionais na reprodução de sinais, mas com os resultados obtidos abrem-se as perspectivas para muitas outras aplicações que possam ser testadas com resultados satisfatórios.

Referências

Referências

[1] ABE, J. M., Fundamentos da Lógica Anotada – Tese de Doutoramento, FFLCH/USP, 1992.

[2] ABE, J. M., On Annotated Model Theory, *Coleção Documentos IEA-USP*, série Lógica e Teoria da Ciência, n° 11, 29 pp, 1993.

[3] ABE, J.M., On Annotated Modal Logic, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 13, 19 p., 1993.

[4] ABE, J.M., Annotated Logics Qτ and ultraproducts, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 16, 1-10, 1994.

[5] ABE, J.M., On Annotated Modal Logic, *Mathematica Japonica*, 40, n. 3, pp. 553-560, 1994.

[6] ABE, J.M., Curry Algebras N_1, *Atti Acc. Lincei Rend. Fis.*, s.9, vol. 7, 125-128, 1996.

[7] ABE, J.M., A note on Curry Algebras Pτ, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 25, 7 p., 1997.

[8] ABE, J.M., Filters and ideals of a Pτ-Algebra, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 26, 7 p., 1997.

[9] ABE, J.M., A logical system for reasoning with inconsistency, Anais da 5ª *Reunião Anual da SBPN'97, Ciência e Cultura na Globalização - Novos Paradigmas*, 08 - 10 de agosto de 1997, Águas de Lindóia, SP, 196-201, 1997.

[10] ABE, J.M., Lógica Paraconsistente e Inteligência Artificial, *Coleção Cadernos de Estudos e Pesquisas - UNIP, Série: Estudos e Pesquisas*, n_o 1-004/97, Universidade Paulista, 28p, 1997.

[11] ABE, J.M., Some Aspects of Paraconsistent Systems and Applications, *Logique et Analyse*, 157(1997), 83-96.

[12] ABE, J.M., Uma algebrização dos sistemas anotados, Atas da 6ª *Reunião Anual da Sociedade Brasileira de Pesquisadores Nikkeis* – SBPN, 254-260, 1998.

[13] ABE, J.M., Incorporando Tempo em Raciocínio Paraconsistente, atas do VI Congresso Brasileiro de Filosofia, 5 a 10 de setembro de 1999, Instituto Brasileiro de Filosofia, Faculdade de Direito da Universidade de São Paulo, 1999.

[14] ABE, J.M., Curry algebras Pτ, a aparecer em *Logique et Analyse*, 1999.

[15] ABE, J.M., Um panorama da Lógica Atual, *Coleção Cadernos de Estudos e Pesquisas - UNIP, Série: Estudos e Pesquisas*, n_o 1-004/00, ISSN 1517-9230, Universidade Paulista, 28p, 2000.

[16] ABE, J.M., Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, São Paulo, SP – Brasil, Editor, ISBN 85-85795-29-8, Editôra Plêiade, 2000.

[17] ABE, J.M., An Algebraic Version of the Annotated Logics Pτ (Preliminary version), *Coleção Cadernos de Estudos e Pesquisas - UNIP, Série: Estudos e Pesquisas*, n_o 1-011/00, ISSN 1517-9230, Universidade Paulista, 10p, 2000.

[18] ABE, J.M. & S. AKAMA, Incorporating time in paraconsistent reasoning, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 37, 11p., 1998.

[19] ABE, J.M. & S. AKAMA, Paraconsistent annotated deontic logics, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 42, 13p., 1998.

[20] ABE, J.M. & S. AKAMA, Paraconsistent knowledge and multi-agents, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 41, 14p., 1998.

[21] ABE, J.M. & S. AKAMA, On some aspects of decidability of annotated systems, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 51, 14p., 1998.

[22] ABE, J.M. & S. AKAMA, A Logical System for Reasoning with Fuzziness and Inconsistencies, Proceedings of the IASTED *International Conference on Artificial Intelligence and Soft Computing* (ASC'99), August 9-12, Honolulu, Hawaii, USA, 221-225, 1999.

[23] ABE, J.M. & S. AKAMA, Annotated temporal logics $\Delta^*\tau$, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 53, 15p., 1999.

[24] ABE, J.M. & S. AKAMA, Discussive Logics $M\tau$, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 54, 15p., 1999.

[25] ABE, J.M. & S. AKAMA, Raciocínio Inconsistente e Multi-agentes, atas do VI Congresso Brasileiro de Filosofia, 5 a 10 de setembro de 1999, Instituto Brasileiro de Filosofia, Faculdade de Direito da Universidade de São Paulo, 1999.

[26] ABE, J.M. & S. AKAMA, Annotated logics $Q\tau$ and ultraproducts, a aparecer em *Logique et Analyse*, 1999.

[27] ABE, J.M. & S. AKAMA, On Jáskowski's Annotated Discursive Logic, Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, Editôra Plêiade, São Paulo, SP – Brasil, Editor: J.M. Abe, ISBN 85-85795-29-8, 445-458, 2000.

[28] ABE, J.M. & S. AKAMA, Paraconsistent annotated temporal logics $\Delta^*\tau$, aceito no International Joint Conference 7th IBERAMIA (the Ibero-American AI Conference) and 15th SBIA2000 (the Braziliam AI Symposium) Supported by SBC - Sociedade Brasileira de Computação, Brazil; AAIA - Associacion Argentina de Inteligencia Artificial; AEPIA - Associacion Espanola para Inteligencia Artificial; APPIA - Associacao Portuguesa para Inteligencia Artificial; AVINTA - Associacion Venezuelana de Inteligencia Artificial; SCCC - Sociedad Chilena de Ciencia de la Computacion; SMIA - Sociedad Mexicana de Inteligencia Artificial. Publication in the conference proceedings by Springer-Verlag in the Lecture Notes in Artificial Intelligence series

[29] ABE, J.M., S. AKAMA & J.I. DA SILVA FILHO, Modalities, paraconsistency, paracompleteness, and AI, Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, Editôra Plêiade, São Paulo, SP – Brasil, Editor: J.M. Abe, ISBN 85-85795-29-8, 459-470, 2000

[30] ABE, J.M., S. AKAMA & R. SYLVAN, General annotated logics, with an introduction to full accounting logic, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 39, 17p., 1998.

[31] ABE, J.M., J.P. ATTIE, A. SCALZITTI, E. TIHARU & F.S. YAMAMOTO, Introdução à Lógica Clássica, Editora Plêiade, 1999, 176 págs.

[32] ABE, J.M., B.C. ÁVILA & J.P.A. PRADO, Multi-agents and inconsistency, *International Conference on Computational Intelligence and Multimedia Applications* 1998, ISBN 981-023352-3,

H. Selvaraj & B. Verma Editors, World Scientific, (*ICCIMA'* 98, Proceedings of the 2nd International Conference), 137-142, 1998.

[33] ABE, J.M. & J.I. DA SILVA FILHO, Implementação de circuitos eletrônicos de funções lógicas paraconsistentes Radix N, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 22, 37 p., 1996.

[34] ABE, J.M. & J.I. DA SILVA FILHO, Inconsistency and Electronic Circuits, *Proceedings of The International ICSC Symposium on Engineering of Intelligent Systems* (EIS'98), Volume 3, Artificial Intelligence, Editor: E. Alpaydin, ICSC Academic Press International Computer Science Conventions Canada/Switzerland, ISBN 3-906454-12-6, 191-197, 1998.

[35] ABE, J.M., J.I. DA SILVA FILHO & K. NAKAMATSU, A Logical System for Reasoning with Inconsistent Deontic Modalities, submetido no Fifth International Conference on Computing Anticipatory Systems, CASYS'2001, Organized by the non-profit association CHAOS, Centre for Hyperincursion and Anticipation in Ordered Systems CHAOS asbl, Institut de Mathématique, Université de Liège, LIEGE, Belgium, Bélgica, 2000.

[36] ABE, J.M. & K. NAKAMATSU, Programação Lógica Paraconsistente e Raciocínio Não-monotônico, Defeasible e Default-Fuzzy, atas do VI Congresso Brasileiro de Filosofia, 5 a 10 de setembro de 1999, Instituto Brasileiro de Filosofia, Faculdade de Direito da Universidade de São Paulo, 1999.

[37] ABE, J.M., K. NAKAMATSU & B.C. ÁVILA, Paraconsistent Annotated Logic Programming, a aparecer no *International Journal of Computing Anticipatory Systems*, 1999.

[38] ABE, J. M. & N. PAPAVERO, Teoria Intuitiva dos Conjuntos, McGraw-Hill, Makron Books, São Paulo, 266 pp, 1991.

[39] ABE, J.M., J.P.A. PRADO & B.C. ÁVILA, On a class of paraconsistent multimodal systems for reasoning, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 24, 12 p., 1997.

[40] AKAMA, S. & J.M. ABE, Many-valued and annotated modal logics, IEEE 1998 *International Symposium on Multiple-Valued Logic* (ISMVL'98), Proceedings, pp. 114-119, Fukuoka, Japão, 1998.

[41] AKAMA, S. & J.M. ABE, Constructive logics as annotated logics, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 40, 9p., 1998.

[42] AKAMA, S. & J.M. ABE, Natural Deduction And General Annotated Logics, atas do *The First International Workshop on Labelled Deduction* (LD'98), Freiburg, Alemanha, 1-14, 1998. Também publicado na *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 49, 14p., 1998.

[43] AKAMA, S. & J.M. ABE, Many-valued and annotated modal logics (Extended version), *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 38, 10p., 1998.

[44] AKAMA, S. & J.M. ABE, Many-valued and annotated modal logics II, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 52, 14p., 1999.

[45] AKAMA, S. & J.M. ABE, Epistemic States In Paraconsistent Logic Programming, submetido no Workshop on Multi-Agent Systems in Logic Programming (ICLP'99), realizado conjuntamente com o International Conference on Logic Programming 1999, 29/11 a 04/12/1999, Las Cruces, Novo México, USA, 1999.

[46] AKAMA, S. & J.M. ABE, Fuzzy annotated logics, aceito no IPMU'2000.

[47] AKAMA, S. & J.M. ABE, Annotated Rules with Uncertainty in Expert Systems, aceito no Eighteenth IASTED International Conference on Applied Informatics (AI 2000) held February 14-17, 2000, Innsbruck, Austria.

[48] AKAMA, S. & J.M. ABE, Annotated logics and uncertainty, Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, Editôra Plêiade, São Paulo, SP – Brasil, Editor: J.M. Abe, ISBN 85-85795-29-8, 449-507, 2000.

[49] AKAMA, S. & J.M. ABE, Fuzzy annotated logics, Anais do 8th International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems, IPMU'2000, Organized by: Universidad Politécnica de Madrid (Spain), July 3-7, 2000, Madri, Espanha, Vol. 1, 504-508, 2000.

[50] ANGELOTTI, E.S., SCALABRIN, E.E., ÁVILA, B.C., BORTOLOZZI, F., Concepção de um Sistema de Agentes Independentes Paraconsistente para o Tratamento de Cheques Bancários Brasileiros -Anais do I Congresso de Lógica Aplicada à Tecnologia – Faculdade SENAC de Ciências Exatas e Tecnologia – LAPTEC'2000 – Editora Plêiade.

[51] ÁVILA, B.C., *Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar Exceções em Sistemas de Frames com Múltipla Herança*, tese de Doutoramento, Universidade de São Paulo, São Paulo, 1996.

[52] ÁVILA, B.C. & J.M. ABE, Inconsistencies, Exceptions, and Frame Systems, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 57, 25p., 1999.

[53] ÁVILA, B.C. & J.M. ABE, Handling Inconsistencies in Logic Programming, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 56, 22p., 1999.

[54] ÁVILA, B.C., J.M. ABE & J.P.A PRADO, Um sistema de frames utilizando programação lógica evidencial, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 27, 43p., 1997.

[55] ÁVILA, B.C., J.M. ABE & J.P.A PRADO, A utilização de redes de herança em representação de conhecimento, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 28, 18p., 1997.

[56] ÁVILA, B.C., J.M. ABE & J.P.A PRADO, Uma extensão da linguagem Prolog para suportar programação lógica evidencial, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 29, 43p., 1997.

[57] ÁVILA, B.C., J.M. ABE & J.P.A. PRADO, Reasoning in Paraconsistent Frame Systems, *The Second International Workshop on CSCW in Design*, P. Siriruchatapong Z. Lin & J. P. Barthes (Eds), International Academic Publishers, ISBN: 7-80003-412-7/TP.19, Bangkok, Thailand, pp. 239-244, 1997.

[58] ÁVILA, B.C., J.M. ABE & J.P.A. PRADO, ParaLog-e: A Paraconsistent Evidential Logic Programming Language, *XVII International Conference of the Chilean Computer Science Society, IEEE Computer Society Press*, pp 2-8, Valparaíso, Chile, Novembro, 1997.

[59] ÁVILA, B.C., J.M. ABE & J.P.A. PRADO, ParaLog-e: A Paraconsistent Logic Programming Language, *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA' 98*, 1998, ISBN 981-023352-3, H. Selvaraj & B. Verma Editors, World Scientific, Proceedings of the 2nd International Conference), 143-148, 1998.

[60] ÁVILA, B.C., J.M. ABE & J.P.A. PRADO, A Paraconsistent Logic Programming Language, *Proceedings of The International ICSC Symposium on Engineering of Intelligent Systems* (EIS'98), Volume 3, Artificial Intelligence, Editor: E. Alpaydin, ICSC Academic Press International Computer Science Conventions Canada/Switzerland, ISBN 3-906454-12-6, 281-287, 1998.

[61] BARROS, C.M., N.C.A. DA COSTA & J.M. ABE, Tópico de Teoria dos Sistemas Ordenados, Vol II: Sistemas de Curry, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, no 20, 132 p., 1995.

[62] BORESTEIN, J & Y. KOREN, "Obstacle avoidance with ultrasonic sensors" *IEEE Journal of Robotics and automation*, N.Y, 1988 .

[63] DA COSTA, N.C.A., *Ensaio sobre os Fundamentos da Lógica*, Hucitec, São Paulo, 1980.

[64] DA COSTA, N. C. A. & Henschen, L.J. & Lu, J.J. & Subrahmanian, V.S., Automatic Theorem Proving in Paraconsistent Logics: Theory and Implementation – Estudos Avançados – Coleção Documentos nº 3, 18p, USP, 1990.

[65] DA COSTA, N.C.A. & J.M. ABE, Inteligência Artificial Paraconsistente, atas do VI Congresso Brasileiro de Filosofia, 5 a 10 de setembro de 1999, Instituto Brasileiro de Filosofia, Faculdade de Direito da Universidade de São Paulo, 1999.

[66] DA COSTA, N.C.A. & J.M. ABE, Algumas Aplicações Recentes dos Sistemas Paraconsistentes à Inteligência Artificial e Robótica, a aparecer, 2000.

[67] DA COSTA, N.C.A. & J.M. ABE, Aspectos Sobre Aplicações dos Sistemas Paraconsistentes, Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, Editôra Plêiade, São Paulo, SP – Brasil, Editor: J.M. Abe, ISBN 85-85795-29-8, 559-571, 2000.

[68] DA COSTA, N.C.A., J.M. ABE, J.I. DA SILVA FILHO, A.C. MUROLO & C.F.S. LEITE, Lógica Paraconsistente Aplicada, ISBN 85-224-2218-4, Editôra Atlas, 214 págs., 1999.

[69] DA COSTA, N.C.A., J.M. ABE & V.S. SUBRAHMANIAN, Remarks on annotated logic, *Zeitschrift f. math. Logik und Grundlagen d. Math.* 37, pp 561-570, 1991.

[70] DA SILVA FILHO, J.I., Implementação de circuitos lógicos fundamentados em uma classe de Lógicas Paraconsistentes Anotadas, Dissertação de Mestrado-EPUSP, São Paulo, 1997.

[71] DA SILVA FILHO, J.I., Métodos de interpretação da Lógica Paraconsistente Anotada com anotação com dois valores LPA2v com construção de Algoritmo e implementação de Circuitos Eletrônicos, EPUSP, Tese de Doutoramento, São Paulo, 1999.

[72] DA SILVA FILHO, J.I. & J.M. ABE, Lógica paraconsistente anotada e circuitos de portas lógicas, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 23, 41p., 1997.

[73] DA SILVA FILHO, J.I. & J.M. ABE, Módulo analisador paraconsistente: uma proposta de circuito lógico paraconsistente, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 30, 18p., 1997.

[74] DA SILVA FILHO, J.I. & J.M. ABE, Parasensor: um sensor paraconsistente projetado para uso em sistema eletrônico embasado em lógica paraconsistente anotada, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 31, 18p., 1997.

[75] DA SILVA FILHO, J.I. & J.M. ABE, Algoritmo Para-analisador – Parte I: um algoritmo para tratamento de inconsistências em sistemas de controle, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 43, 18p., 1998.

[76] DA SILVA FILHO, J.I. & J.M. ABE, Algoritmo Para-analisador – Parte II: aplicação do operador negação (*NOT*) e dos conectivos da conjunção (*AND*) e da disjunção (*OR*) da lógica paraconsistente anotada de anotação com dois valores *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 44, 15p., 1998.

[77] DA SILVA FILHO, J.I. & J.M. ABE, Algoritmo Para-analisador – Parte III: propostas de aplicações do algoritmo da lógica paraconsistente anotada de anotação com dois valores – LPA2v em

sistemas especialistas de Inteligência Artificial, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 45, 15p., 1998.

[78] DA SILVA FILHO, J.I. & J.M. ABE, Algoritmo Para-analisador – Parte IV: propostas de aplicações do algoritmo da lógica paraconsistente anotada de anotação com dois valores – LPA2v em sistemas de controle de Robôs, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 46, 22p., 1998.

[79] DA SILVA FILHO, J.I. & J.M. ABE, Controlador lógico Para-Fuzzy – Parte I: Um novo método de controle híbrido utilizando lógica paraconsistente e lógica *fuzzy*, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 47, 20p., 1998.

[80] DA SILVA FILHO, J.I. & J.M. ABE, Controlador lógico Para-Fuzzy – Parte II: Um controlador híbrido indicado para tratamento de inconsistências utilizando lógica paraconsistente e lógica *fuzzy*, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 48, 22p., 1998.

[81] DA SILVA FILHO, J.I. & J.M. ABE, Para-Analyser and Inconsistencies in Control Systems, Proceedings of the IASTED *International Conference on Artificial Intelligence and Soft Computing* (ASC'99), August 9-12, Honolulu, Hawaii, USA, 78-85, 1999.

[82] DA SILVA FILHO, J.I. & J.M. ABE, Para-Fuzzy Logic Controller – Part I: A New Method of Hybrid Control Indicated for Treatment of Inconsistencies Designed with the Junction of the Paraconsistent Logic and Fuzzy Logic, Proceedings of the International ICSC Congress on Computational Intelligence Methods and Applications CIMA'99, Rochester Institute of Technology, RIT, Rochester, N.Y., USA, ISBN 3-906454-18-5, Editors: H. Bothe, E. Oja, E. Massad & C. Haefke, ICSC Academic Press, International Computer Science Conventions, Canada/Switzerland, 113-120, 1999.

[83] DA SILVA FILHO, J.I. & J.M. ABE, Para-Fuzzy Logic Controller – Part II: A Hybrid Logical Controlller Indicated for Treatment of Fuzziness and Inconsistencies, Proceedings of the International ICSC Congress on Computational Intelligence Methods and Applications CIMA'99, Rochester Institute of Technology, RIT, Rochester, N.Y., USA, ISBN 3-906454-18-5, Editors: H. Bothe, E. Oja, E. Massad & C. Haefke, ICSC Academic Press, International Computer Science Conventions, Canada/Switzerland, 106-112, 1999.

[84] DA SILVA FILHO, J.I. & J.M. ABE, Lógica Paraconsistente Aplicada à Robótica, atas do VI Congresso Brasileiro de Filosofia, 5 a 10 de setembro de 1999, Instituto Brasileiro de Filosofia, Faculdade de Direito da Universidade de São Paulo, 1999.

[85] DA SILVA FILHO, J.I. & J.M. ABE, " Contribuição da Lógica Paraconsistente ao Campo da Engenharia", atas do VI Congresso Brasileiro de Filosofia, 5 a 10 de setembro de 1999, Instituto Brasileiro de Filosofia, Faculdade de Direito da Universidade de São Paulo, 1999.

[86] DA SILVA FILHO, J.I. & J.M. ABE, Métodos de Aplicações da Lógica Paraconsistente Anotada, aceito na 7ª *Reunião Anual da Sociedade Brasileira de Pesquisadores Nikkeis* – SBPN, 29–31/07/1999, Londrina, PR, 1999.

[87] DA SILVA FILHO, J.I. & J.M. ABE, Paraconsistent Electronic Circuits, trabalho convidado, Fourth International Conference on Computing Anticipatory Systems, CASYS'2000, Fourth International Conference on Computing Anticipatory Systems, CASYS'2000, Organized by the non-profit association CHAOS, Centre for Hyperincursion and Anticipation in Ordered Systems CHAOS asbl, Institut de Mathématique, Université de Liège, LIEGE, Belgium, Bélgica, 7-12 de agosto de 2000, 2000.

[88] DA SILVA FILHO, J.I. & J.M. ABE, Paraconsistent Analyser Module, trabalho convidado, Fourth International Conference on Computing Anticipatory Systems, CASYS'2000, Fourth International Conference on Computing Anticipatory Systems, CASYS'2000, Organized by the non-profit association CHAOS, Centre for Hyperincursion and Anticipation in Ordered Systems CHAOS asbl, Institut de Mathématique, Université de Liège, LIEGE, Belgium, Bélgica, 7-12 de agosto de 2000, 2000.

[89] DA SILVA FILHO, J.I. & J.M. ABE, Emmy: an autonomous mobile robot, submetido para publicação nos anais do WCP'2000, Editora Marcel Dekker.

[90] DA SILVA FILHO, J.I., J.M. ABE & G.A. BONESSO, Simulating Inconsistencies in a Paraconsistent Logic Controller, submetido no Fifth International Conference on Computing Anticipatory Systems, CASYS'2001, Organized by the non-profit association CHAOS, Centre for Hyperincursion and Anticipation in Ordered Systems CHAOS asbl, Institut de Mathématique, Université de Liège, LIEGE, Belgium, Bélgica, 2000.

[91] DA SILVA FILHO, J.I., J.M. ABE & P.L. SANCHES, Circuitos de portas lógicas primitivas fundamentados em lógica paraconsistente anotada, *III WORKSHOP DE IBERSHIP*, 19-21 de fevereiro 1997, Departamento de Ingeniería Elétrica, CINVESTAV - IPN, México D.F., México, 227-237, 1997

[92] DA SILVA FILHO, J.I., J.M. ABE & P.L. SANCHES, Circuitos de portas lógicas primitivas implementados a partir de uma classe de lógicas paraconsistentes anotadas, *Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia Eletrônica*, ISSN 1413-2206, BT/PEE/9723, 13p.,1997.

[93] DA SILVA FILHO, J.I., J.M. ABE, C.R. TORRES, A.M. CÉSAR, M.C. MÁRIO, A.M. SANTOS, I.J. CANCINO Jr. & D.M. SALLES, Emmy: Robô Móvel Autônomo Paraconsistente – Protótipo 1, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 59, 17 págs, 1999.

[94] DA SILVA FILHO, J.I. & ABE, J.M., Fundamentos das Redes Neurais Artificiais Paraconsistentes – Destacando Aplicações em Neurocomputação , Ed. Arte Moderna, 2001.

[95] DA SILVA FILHO, J.I. ,Tese de Doutorado " Métodos de Aplicações da Lógica Paraconsistente Anotada de Anotação com dois valores – LPA2v com construção de algoritmo e implementação de circuitos eletrônicos", Escola Politécnica da USP, 1998.

[96] DA SILVA FILHO, J. I., BONESSO, G. A., ABE, J. M. -Para-Sim: Simulador de Controle Lógico Paraconsistente - Anais do I Congresso de Lógica Aplicada à Tecnologia – Faculdade SENAC de Ciências Exatas e Tecnologia – LAPTEC'2000 – Editora Plêiade.

[97] DA SILVA FILHO, J.I., G.A. BONESSO & J.M. ABE, Para-Sim: Simulador de Controle Lógico Paraconsistente, Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, Editôra Plêiade, São Paulo, SP – Brasil, Editor: J.M. Abe, ISBN 85-85795-29-8, 603-611, 2000.

[98] ENEMBRECK, F., B.C. ÁVILA & J.M. ABE, Um editor de conhecimento genérico em sistema de Frames paraconsistentes, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n$_o$ 55, 77p., 1999.

[99] ENRIQUES, F., *Per la Storia della Logica*, Zanichelli, Bolonha, 1922.

[Enriques & Santillana 32] ENRIQUES, F. & G. SANTILLANA, *Storia del Pensiero Scientifico*, vol. 1, Zanichelli, Bolonha, 1932.

[100] ENRIQUES, F. & G. SANTILLANA, *Compedio di Storia del Storia del Pensiero Scientifico*, Zanichelli, Bolonha, 1936.

[101] FISCHLER, M.A. & O. FIRSCHEIN, "*Intelligence The Eye, The Brain and The Computer*" Addison-Wesley Publishing Company, USA, 1987.

[102] HALLIDAY, J.S., The Characterization of Vectorcardiograms for Pattern Recognition – Master Thesis, MIT, Cambridge, 1973.

[103] HEBB, D. "*The Organization of Behavior*" Wiley, New York, 1949.

[104] ISÉKI, K. & J.M. ABE, A survey on BCK and BCI algebras, Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, Editôra Plêiade, São Paulo, SP – Brasil, Editor: J.M. Abe, ISBN 85-85795-29-8, 430-444, 2000.

[105] ISÉKI, K. & J.M. ABE, Lógica Matemática e Aplicações, a aparecer, 2000.

[106] KOVÁCS, ZSOLT LÁSZLO, Redes Neurais Artificiais – Fundamentos e Aplicações, collegium cognitio, 1996.

[107] MARANA, A. N., COSTA, P. V. C. – Reconhecimento de Dígitos Manuscritos Usando Transformada de Hough e Redes Neurais - Anais do I Congresso de Lógica Aplicada à Tecnologia – Faculdade SENAC de Ciências Exatas e Tecnologia – LAPTEC'2000 – Editora Plêiade.

[108] McCULLOCH, W & W. PITTS, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, 1943.

[109] NAKAMATSU, K. & J.M. ABE, Reasonings Based On Vector Annotated Logic Programs, atas do CIMCA'99, *International Conference on Computational Intelligence for Modelling Control and Automation*, Edited by M. Mohammadian, IOS Press – Ohmsha, ISBN 90 5199 474 5 (IOS Press), Netherlands, 396-403, 1999.

[110] NAKAMATSU, K., J.M. ABE & A. SUZUKI, An approximate reasoning in a framework of vector annotated logic programming, *The Vietnam-Japan Bilateral Symposium on Fuzzy Systems And Applications*, VJFUZZY' 98, Nguyen H. Phuong & Ario Ohsato (Eds), HaLong Bay, Vietnam, 521-528, 1998.

[111] NAKAMATSU, K., J.M. ABE & A. SUZUKI, Nonmonotonic, Defesiable, Default-Fuzzy & Paraconsistent Reasonings, em Aplicações de Programação Anotada, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 60, 13-22, 1999.

[112] NAKAMATSU, K., J.M. ABE & A. SUZUKI, Defesiable Reasoning Between Conflicting Agents Based on VALPSN, em Aplicações de Programação Anotada, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 60, 5-12, 1999.

[113] NAKAMATSU, K., J.M. ABE & A. SUZUKI, "Defeasible Reasoning Between Conflicting Agents Based on VALPSN", American Association for Artificial Intelligence - AAAI'99 Workshop on Agents' Conflicts, ISBN 1-57735-092-8, TR WS-99-08, AAAI Press – American Association for Artificial Intelligence, Menlo Park, California, USA, 20-27, 1999.

[114] NAKAMATSU, K., J.M. ABE & A. SUZUKI, Defesiable Reasoning Based on VALPSN and applications, anais do 3rd Australian Commonsense Reasoning Workshop, 12th Australian Joint Conference on Artificial Intelligence (AI'99), Sydney, Austrália, 114-130, 1999.

[115] NAKAMATSU, K., J.M. ABE & A. SUZUKI, Raciocínio automático em lógica deôntica "defeasible" paraconsistente, Atas do I Congresso de Lógica Aplicada à Tecnologia – LAPTEC'2000, Editôra Plêiade, São Paulo, SP – Brasil, Editor: J.M. Abe, ISBN 85-85795-29-8, 471-480, 2000.

[116] NAKAMATSU, K., J.M. ABE & A. SUZUKI, trabalho convidado, Fourth International Conference on Computing Anticipatory Systems, CASYS'2000, Organized by the non-profit association CHAOS, Centre for Hyperincursion and Anticipation in Ordered Systems CHAOS asbl, Institut de Mathématique, Université de Liège, LIEGE, Belgium, Bélgica, 7-12 de agosto de 2000, 2000.

[117] NAKAMATSU, K., J.M. ABE & A. SUZUKI, Annotated Semantics for Defeasible Deontic Reasoning, aceito no The Second International Conference on Rough Sets and Current Trends in Computing - RSCTC'2000, October 16-19, 2000, Banff, Canada, Conference Organization Conference Chair: Wojciech Ziarko, University of Regina, Canada Publication in the conference proceedings by Springer-Verlag in the Lecture Notes in Artificial Intelligence series.

[118] NAKAMATSU, K., Y. HASEGAWA, J.M. ABE & A. SUZUKI, A Framework for Intelligent Systems Based on Vector Annotated Logic Programs, IPMM'99 *The Second International Conference on Intelligent Processing and Manufacturing of Materials*, ISBN 0-7803-5489-3, Editors: J.A. Meech, M.M. Veiga, M.H. Smith & S.R. LeClair, IEEE Catalogue Number: 99EX296, Library of Congress Number: 99-61516, Honolulu, Hawaii, USA, 695-702, 1999.

[119] NAKAMATSU, K., Y. HASEGAWA, J.M. ABE & A. SUZUKI, A Framework for Intelligent Systems, em Aplicações de Programação Anotada, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 60, 23-30, 1999.

[120] NEGOITA, C.V. & D.A. RALESCU, *Applications of Fuzzy Sets to Systems Analysis*, John Wiley & Sons, 1975.

[121], NUNES, F., RODOPOULOS, C., -Um Modelo Neural de Predição Baseado em Redes MLP para Aplicação em WEB Mining, Logic, Artificial Intelligence and Robotics – vol.II – LAPTEC 2001 – Faculdade SENAC de Ciências e Tecnologia – Editora Plêiade.

[122] PRADO, J.P.A., *Uma Arquitetura em IA Baseada em Lógica Paraconsistente*, tese de Doutoramento, Universidade de São Paulo, 1996.

[123] PRADO, J.P.A. & J.M. ABE, Uma arquitetura para Inteligência Artificial Distribuída baseada em lógica paraconsistente anotada, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 33, 19p., 1998.

[124] PRADO, J.P.A. & J.M. ABE, Um Planejador Baseado em Lógica Paraconsistente. *2o Simpósio Brasileiro de Automação Inteligente*, Curitiba, pp. 177-178, 1995.

[125] PRADO, J.P.A., J.M. ABE & B.C. ÁVILA, Fundamentos da arquitetura *Blackboard*, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 35, 29p., 1998.

[126] PRADO, J.P.A., J.M. ABE & B.C. ÁVILA, Inteligência Artificial Distribuída: aspectos, *Coleção Documentos, Série Lógica e Teoria da Ciência*, IEA-USP, n_o 36, 35p., 1998.

[127] PRADO, J.P.A., J.M. ABE & M. RILLO, A paraconsistent planning system, In: International Conference on CAD/CAM, Robotics and Factories of the Future, 11, Pereira, 1995, 327-338.

[128] RESCONI, G. & J.M. ABE, Multilevel uncertainty logic, *Quaderni Del Seminario Matematico Di Brescia*, n_o 14/97, Universita Cattolica del Sacro Cuore e Università degli Studi di Brescia, 28p, Itália, 1997.

[129] ROSEMBLATT, "*Principles of Neurodynamics*", Spartan Books, NY, 1962.

[130] SIEBERT, W. "*Stimulus Transformation in Peripheral Auditory System in Recognizing Patterns*", Ed. Murray Eden, MIT Press, Cambridge, 1968.

[131] SHOENFIELD, J.R., *Mathematical Logic*, Addison-Wesley, 1967.

[132] SRIHARI, S. N., Recognition of handwritten and machine printed text for postal adress interpretation, Pattern Recog. Letters, 14(4), pp.291-302, 1991.

[133] SUN TING, D. K., MASOTTI, P. H. F, MESQUITA, R. N., Teste de Validade para um Ensaio por Eddy Current em Tubos de Geradores de Vapor Uando Lógica "Fuzzy" Paraconsistente – Anais do I Congresso de Lógica Aplicada à Tecnologia – Faculdade SENAC de Ciências Exatas e Tecnologia – LAPTEC'2000 – Editora Plêiade.

[134] SUBRAHMANIAN, V.S., On the semantics of quantitative Lógic Programs, Proc. 4th. IEEE Symposium on Logic Programming, Computer Society, press, Washington D.C., 1987.

[135] SUZUKI, Y., Self-Organizing QRS-Wave Recognition in ECG Unsing Neural Networks – IEEE Trsnd. On Neural Networks, 6, 1995.

[136] SYLVAN, R. & J.M. ABE, On general annotated logics, with an introduction to full accounting logics, *Bulletin of Symbolic Logic*, 2, 118-119, 1996.

[137] TAPPERT, C.C., SUEN, C. Y., and Wakara, C., The state of the art in on-line handwritten recognition, IEEE Trans. On Pattern Analysis and Machine Intelligence, 12(8), pp. 787-808, 1990.

[138] ZADEH, L.," Outline of a New Approach to the Analysis of Complex Systems and Decision Processes" – *IEEE Transaction on Systems, Mam and Cybernectics*, vol. SMC-3, No 1, p.p. 28-44, January, 1973.

[139] ZERBINI, R.C., Metodologia para a Classificação de VCG através de redes neurais – Dissertação de Doutoramento, Escola Politécnica da USP, 1993.

Anexos

Anexo 1:

APLICATIVO CLASSIFICADOR DE SINAIS

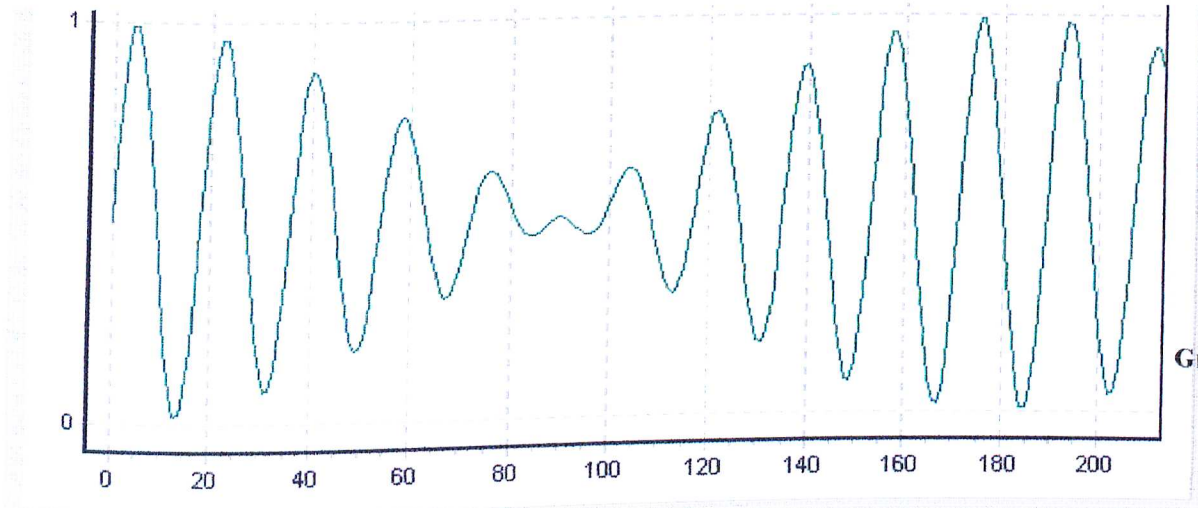


Gráfico 1

padrão1 | padrão2 | padrão3 | padrão4 | clear | grava | classifica1

sinal01 | sinal02 | sinal03 | sinal04 | familia1 | familia2 | classifica2

sinalteste1 padrao1[i]:= (0.7 + (Sin((i*Pi)/180))*(-2*Sin((i*Pi)/180))*(Cos((i*Pi)/180))/2)*(Random(360)/400);

sinalteste2 padrao2[i]:= (0.7 + (Sin((0.99*i*Pi)/180))*(-2*Sin((0.99*i*Pi)/180))*(Cos((0.99*i*Pi)/180))/2)*(Random(360)/400);

sinalteste3 padrao3[i]:= 0.5 + (Sin((20*i*Pi)/180))*(Cos((i*Pi)/180))/2;

sinalteste4 padrao4[i]:= 0.5 + (0.99*Sin((20*i*Pi)/180))*(0.99*Cos((i*Pi)/180))/2;

sinalteste5 padrao5[i]:= (0.7 + (Sin((1.01*i*Pi)/180))*(-2*Sin((1.01*i*Pi)/180))*(Cos((1.01*i*Pi)/180))/2)*(Random(360)/400);

sinalteste6 padrao6[i]:= 0.5 + (0.97*Sin((20*i*Pi)/180))*(0.97*Cos((i*Pi)/180))/2;

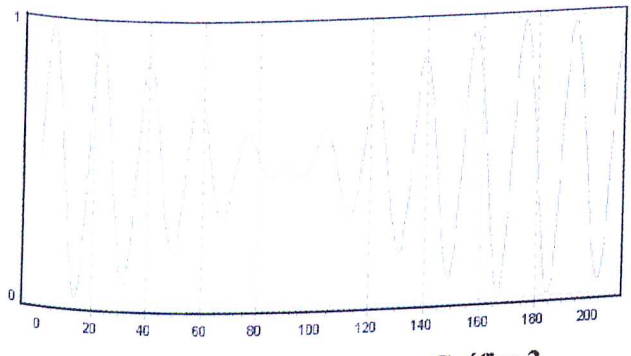


Gráfico 2

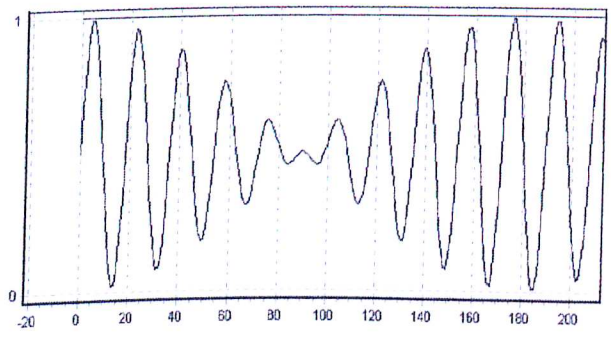


Gráfico 3

Funcionamento:

Botões **padrãox**: geram um sinal (as equações são mostradas nos labels) que são mostrados nos gráficos;

Botões **sinal0x**: geram sinais que correspondem à aprendizagem dos sinais anteriormente gerados (mr^s);

Botões **sinalteste0x**: inserem sinais de teste na rede (as equações são mostradas nos labels), e fazem com que verifique se pertencem ou não à mesma;

Botão **clear**: limpa os gráficos da tela;

Botão **grava**: grava a tela em um arquivo .bmp.

Botão **familiax:** iguais aos botões **sinalteste0x**, porém com uma faixa de teste mais larga, permitindo que a classificação se dê com um grau de tolerância maior.

Botão **classificax:** verifica se o sinal pertence a uma família de sinais armazenados, porém a célula utilizada nesse teste é a de conexão lógica analítica.

Exemplo para o **padrão3**:

a) apertando o botão **padrão3** será gerado o sinal padrao3, mostrado no gráfico1;

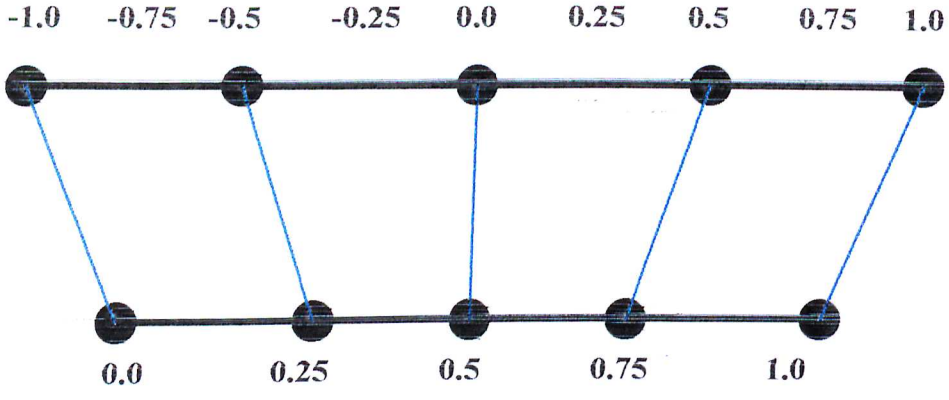
b) apertando o botão **sinal03** será gerado o sinal mr6 (aprendizado de padrao3), mostrado no gráfico2;

c) apertando o botão **sinalteste3**, a rede irá testar um sinal de teste igual ao padrao3, e portanto dará como resposta na saída o próprio padrao3, pois este pertence à rede, e saída S = 1, como mostra o gráfico3.

Anexo 2:

Lógica Paraconsistente: Conversão do Eixo de Graus de Certeza no Eixo dos Graus de Crença Resultante μr [Da Silva Filho & Abe – 2000].

Graus de certeza	Valores Resultantes	Significado Lógico
1	1	Verdadeiro
0.75	0.875	Quase-verdadeiro
0.5	0.75	Quase-indeterminado tendendo a verdadeiro
0.25	0.625	Quase-indeterminado
0.0	0.5	Indeterminado (valores indeterminados ou contraditórios)
-0.25	0.375	Quase-indeterminado
-0.5	0.25	Quase-indeterminado tendendo a falso
-0.75	0.125	Quase-falso
-1	0.0	Falso



Esta dissertação originou o trabalho **"Um Sistema Classificador de Sinais Baseado em Redes Neurais Artificiais Paraconsistentes"**, que foi submetido à participação dos congressos:

" The 6th World Multiconference on Systemics, Cybernetics and Informatics " [www.iiis.org/sci2002/], que será realizado em Orlando, Flórida, de 14 a 18 de Julho/2002.

"Laptec 2002 – III Congresso de Lógica Aplicada à Tecnologia "– Faculdade Senac de Ciências Exatas e Tecnologia. [www.sp.senac.br/laptec2002]