

unesp

UNIVERSIDADE ESTADUAL PAULISTA
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DISSERTAÇÃO DE MESTRADO
EM
ENGENHARIA ELÉTRICA

1210001012



Nº 7

CONTRIBUIÇÕES AO CONTROLE DE SISTEMAS
UTILIZANDO REDES NEURONAIS ARTIFICIAIS
ALUNO: DÉCIO LAMAS
ORIENTADOR: MARCELO CARVALHO MINHOTO TEIXEIRA

Te.1012

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
- FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA -
FEIS/UNESP

CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA
STRICTO SENSU
ÁREA DE AUTOMAÇÃO

CONTRIBUIÇÕES AO CONTROLE DE SISTEMAS UTILIZANDO REDES NEURONAIS ARTIFICIAIS

"Dissertação submetida à Faculdade de Engenharia de Ilha Solteira - Unesp, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica, junto ao Curso de Pós-Graduação em Engenharia Elétrica."

Orientador: Prof. Dr. Marcelo C. Minhoto Teixeira

Aluno: Engº Décio Lamas

Ilha Solteira - SP
11/1995

BCpIS
FEIS-UNESP

Ao meu orientador Prof. Dr. Marcelo C. M. Teixeira, pela maneira paciente com que me tratou nos momentos de incertezas, me proporcionando sempre muita autoconfiança através de um relacionamento amigo e de liberdade, durante todo o desenvolvimento do trabalho.

Ao Departamento de Engenharia Elétrica pela infra-estrutura oferecida, principalmente no que se refere ao número de micro-computadores disponíveis a qualquer momento, possibilitando o perfeito cumprimento das tarefas segundo o cronograma traçado por nós.

À todos os professores que, direta ou indiretamente, nos passaram conhecimentos teóricos, ou experiências, durante todo o curso. Um agradecimento especial ao Prof. Dr. Carlos R. Minussi, que como Coordenador do Curso de Pós-Graduação, ofereceu totais condições para que eu pudesse ingressar neste curso, que é voltado para Engenharia Elétrica, sendo eu formado à nível de graduação em Engenharia Mecânica.

À UNESP, pelo apoio financeiro no primeiro semestre, quando nos contemplou com uma bolsa de estudos, através de seu programa de ajuda a alunos dos cursos de Pós-Graduação.

À FAPESP, também pelo apoio financeiro, oferecido durante 2 (dois) anos, possibilitando o perfeito andamento do curso conforme planejamento inicial.

À minha esposa, Prof^a Rita de C. P. Lamas, pelo apoio, tanto de caráter moral, como de caráter técnico.

Aos meus filhos, Tiago, André e Bruno pela compreensão de alguns momentos importantes de suas vidas, que possivelmente estive ausente, mas que se fizeram necessário para realização deste trabalho, de uma maneira satisfatória.

CONTRIBUIÇÕES AO CONTROLE DE SISTEMAS UTILIZANDO REDES NEURONAIS ARTIFICIAIS

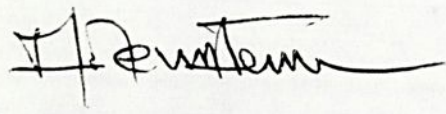
DÉCIO LAMAS

Dissertação submetida à Faculdade de Engenharia de Ilha Solteira - Unesp, como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica, junto ao Curso de Pós-Graduação em Engenharia Elétrica.


CARLOS ROBERTO MINUSSI, Dr. Sc. - Coordenador

BANCA EXAMINADORA:


MARCELO C. MINHOTO TEIXEIRA, Dr. Sc. - Orientador


ADEMAR FERREIRA, Dr. Sc.


JOSÉ PAULO GARCIA, Dr. Sc.



1210001012-1210001012-1210001012

UNESP - "CARLOS DE ILHA SOLTEIRA"	
SERVIÇO DE REGISTRO E DOCUMENTAÇÃO	
DATA DE RECEBIMENTO	DATA DE TURNO
01.3.96	29.3.96
REGISTRO	TURNO
Ailga	Tc. 1012
RECEBIMENTO	ESPECIFICAÇÃO
Recebido R\$ 10,00 Autp	

RNA	Rede Neuronal Artificial;
RNB	Rede Neural Biológica;
IA	Inteligência Artificial;
EQM	Erro Quadrático médio;
BA	Algoritmo "Backpropagation";
FE	Filtro de estados;
FT	Função de Transferência;
W_{ij}^k	Peso que liga o i-ésimo neurônio da k-ésima coluna com o j-ésimo neurônio da camada anterior (k-1);
X_i^j	Saída do i-ésimo neurônio da j-ésima camada;
S_i^j	Resultado da soma de todas as entradas multiplicadas pelos respectivos pesos, do i-ésimo neurônio da j-ésima camada;
$\Gamma(\cdot)$	Função não-linear utilizada nos neurônios;
$K(j)$	É o número de neurônios da j-ésima camada;
$y_{d,i}$	I-ésima saída desejada;



y_i	I-ésima saída fornecida pela RNA;
E_i	Erro entre a i-ésima saída desejada e a i-ésima saída fornecida pela RNA;
$u_i(k)$	I-ésima entrada fornecida ao sistema;
∇	Gradiente de uma função;
μ	Passo de iteração;
δ_i^j	Erro derivativo referente ao i-ésimo neurônio da j-ésima camada;
η	Fator (ou termo) de momento.



CAPÍTULO I - INTRODUÇÃO	01
CAPÍTULO II - INFORMAÇÕES BÁSICAS SOBRE RNA	05
II.1 - Introdução	05
II.2 - Histórico	05
II.3 - Estruturas das RNA's	11
II.3.1 - "Perceptrons"	14
II.3.2 - "Feedforward"	14
II.3.3 - "Counterpropagation"	16
II.3.4 - Métodos Estatísticos	16
II.3.5 - Hopfield	17
II.3.6 - Memória Associativa	17
II.4 - Conclusões	17
CAPÍTULO III - PROPOSTAS PARA MELHORIA DO BA	18
III.1 - Introdução	18
III.2 - Métodos Encontrados na Literatura	18
III.3 - Concepção de um Novo Método	21
III.4 - Conclusões	27



CAPÍTULO IV - APLICAÇÕES DO MÉTODO PROPOSTO	28
IV.1 - Introdução	28
IV.2 - Resultados das Simulações	28
IV.2.1 - Exemplo-1	29
IV.2.2 - Exemplo-2	30
IV.2.3 - Exemplo-3	31
IV.2.3.1 - Exemplo-3.1	33
IV.2.3.2 - Exemplo-3.2	34
IV.3 - Conclusões	35
 CAPÍTULO V - SISTEMAS DE CONTROLE UTILIZANDO RNA's	 36
V.1 - Introdução	36
V.2 - Controle de Plantas Dinâmicas	36
V.3 - RNA's em Controle	38
V.4 - Conclusões	44
 CAPÍTULO VI - EXEMPLOS DE CONTROLE UTILIZANDO RNA'S	 45
VI.1 - Introdução	45
VI.2 - Modelagem do Sistema Dinâmico	45
VI.3 - Sistema Dinâmico Linearizado	47
VI.3.1 - Simulações Com a Planta Linearizada	49
VI.3.1.1 - Exemplo-4	50
VI.3.1.2 - Exemplo-5	51
VI.3.1.3 - Exemplo-6	52
VI.3.2 - Conclusões	53
VI.4 - Sistema Dinâmico Não-Linearizado	53
VI.4.1 - Simulações Com a Planta Não-Linearizada	56
VI.4.1.1 - Exemplo-7	57
VI.4.1.2 - Exemplo-8	58
VI.4.1.3 - Exemplo-9	59
VI.4.2 - Conclusões	60



CAPÍTULO VII - CONCLUSÕES GERAIS	61
REFERÊNCIAS BIBLIOGRÁFICAS	63
APÊNDICE - I - ALGORITMO "BACKPROPAGATION" ORIGINAL	A-I.1
APÊNDICE - II - ALGORITMO "BACKPROPAGATION" COM MOMENTO	A-II.1
APÊNDICE - III - ALGORITMO "BACKPROPAGATION" MODIFICADO	A-III.1
APÊNDICE - IV - ARTIGO PUBLICADO EM CONGRESSO (em Inglês)	A-IV.1
APÊNDICE - V - FLUXOGRAMA RESUMIDO DO "SOFTWARE" UTILIZADO	A-V.1



Este trabalho estuda Redes Neurais Artificiais (RNA) e suas aplicações em Controle Automático de Sistemas.

O Algoritmo "Backpropagation" (BA) é o método "standard" para treinamento de RNA's multicamadas, porém sua convergência normalmente é muito lenta e pode parar em um mínimo local.

Nós apresentamos um novo método para treinamento de RNA's utilizando o BA, inspirado no Construtivismo, um método de alfabetização proposto por Emília Ferrero, baseado na filosofia de Piaget.

Os resultados das simulações, em problemas de identificação de funções, mostram que a configuração proposta, usualmente, obtém um menor erro quadrático médio final, quando comparado com o "standard" BA e com o BA utilizando o termo de momento.

É apresentada uma aplicação da configuração proposta, no controle inverso de um pêndulo forçado de segunda ordem.



This work studies Artificial Neural Network and its applications in automatic control systems.

The Backpropagation Algorithm is the standard method for training multilayer Artificial Neural Network although it converges very slowly and can stop in a local minimum.

We present a new method for neural network training using the Backpropagation Algorithm inspired on constructivism, an alphabetization method proposed by Emília Ferrero based on Piaget philosophy.

Simulation results in identification of functions problems, show that the proposed configuration, usually, obtains a lower final mean square error, when compared with the standard BA and with the BA with momentum factor.

An application of the proposed configuration in the inverse control of a second order forced pendulum is presented.



INTRODUÇÃO

A natureza há bilhões de anos rege o universo de uma forma perfeita, através de suas leis infalíveis. Ela controla todos os seus movimentos de forma tão fascinante que o homem por mais que possa tentar, jamais conseguirá decifrar todos os seus enigmas, e nunca conseguirá substituí-la nesta tarefa de controlar este sistema dinâmico tão complexo.

Em outras épocas o homem se auto classificou, assumindo a posição de centro do mundo, até que Galileu destronou-o do centro do universo e Darwin destituiu-o de sua superioridade biológica.

Atualmente a Inteligência Artificial (IA) está sendo estudada e classificada como o maior desafio que o homem já enfrentou. Porém não se pode acreditar que a IA possa privar o homem de sua exclusividade criadora. O que se espera de uma máquina "inteligente" é apenas que consiga resolver problemas muito específicos, processando informações, como faz o cérebro humano, as quais seriam empregadas nos casos onde o homem por algum motivo se acha incapacitado de estar presente.

Porém, para isto o homem necessita antes de mais nada conhecer melhor a si próprio para que possa construir e dominar tais equipamentos, e esta tarefa é bastante árdua, demorada, e com alto custo financeiro.

Com a revolução industrial do século XVIII o homem percebeu que havia uma necessidade bem mais simples do que aquela de controlar o universo. Ele teria que controlar



pequenas máquinas, ou pequenos processos para obter os resultados que atendessem as novas necessidades. Porém com a velocidade espantosa do progresso, da tecnologia, não demorou para o homem perceber que haviam certos tipos de controle impossíveis de serem executados por ele, e passou a tentar embutir nas máquinas as ações de controle inicialmente executadas pelo próprio homem.

Nesta fase o homem começou a se preocupar em estabelecer regras matemáticas e físicas para serem seguidas pelos controladores automáticos de sistemas. Assim nasceu um novo campo de estudo muito importante no contexto da indústria moderna, que é o "Controle Automático de Sistemas".

O primeiro trabalho nesta linha, registrado de valor prático e científico, foi o "Controlador Centrífugo de Velocidade" criado por James Watt, concebido para executar o controle de velocidade de uma máquina automaticamente [01].

Este trabalho foi tão relevante que a idéia contida naquele tipo de controlador vem sendo utilizado até nos dias de hoje, tanto na indústria como didaticamente.

A partir desta data muitos estudos nesta área foram efetuados, e muitos outros trabalhos importantes foram registrados. No entanto, até a metade deste século, o controle automático de sistemas ficou restrito basicamente a sistemas que pudessem ser simplificados a ponto de serem descritos por equações matemáticas lineares e invariantes no tempo. É o chamado "Controle Clássico".

Foram criados importantes métodos e técnicas, porém a hipótese de linearização dos modelos matemáticos sempre esteve presente no projeto ou análise dos sistemas a serem controlados.

Com a crescente necessidade de sistemas cada vez mais complexos, e também a necessidade de modelos que pudessem representar os problemas físicos de forma mais realista, os cientistas desenvolveram novas técnicas, atingindo uma nova fase nesta área, que é o chamado "Controle Moderno".

Através das técnicas do Controle Moderno, problemas não lineares, de múltiplas entradas e múltiplas saídas e/ou variantes no tempo, podem ser analisados.

O computador foi introduzido nesta área como a ferramenta mais potente e decisiva para a implementação prática dos controladores.

Inicialmente foram os computadores analógicos, com uma incrível rapidez nas operações matemáticas, e em seguida os computadores digitais em sua versão mais utilizada em



forma de micro-computadores ou computadores pessoais com grande capacidade de memória, facilidades na programação e alterações na programação, e o mais importante, com um custo de aquisição e manutenção cada vez menores, devido ao crescente desenvolvimento da tecnologia digital. Este tipo de máquina vem proporcionando também, tempos de processamento cada vez menores.

Porém as exigências do mundo de hoje com o avanço tecnológico que não para, sugere sistemas dinâmicos cada vez mais complexos, dificultando, a cada novo problema, a possibilidade de controlá-lo baseando-se exclusivamente em modelos matemáticos, pois não se consegue equacioná-los através dos princípios físicos conhecidos, de uma forma desejada, para representá-los satisfatoriamente em suas mais diversas aplicações.

Entramos então em uma terceira etapa da teoria de controle de sistemas, que poderia ser denominada "Controle Inteligente" que estaria exatamente preocupado em analisar os diversos casos em que o computador precisa interagir de forma intensa com o mundo real. E como sabemos, não se consegue descrever o mundo real por equações matemáticas exatas e isto inviabiliza o uso de máquinas ou procedimentos baseados na arquitetura de *Von Neuman*.

Alguns exemplos clássicos deste tipo de problema são: reconhecimento de voz e de imagem, classificação e identificação de padrões, controle da atividade motora, etc [02].

Observando os exemplos colocados, percebe-se que o homem soluciona estes problemas sem nenhum esforço. Por exemplo, uma criança de alguns meses já possui o poder de reconhecer voz e imagem, e com alguns anos consegue andar de bicicleta sem conhecer as leis físicas envolvidas naquelas ações. Ao tentarmos executar tais ações utilizando processos convencionais de computação teremos grandes problemas, justamente porque é impossível armazenar todas as informações necessárias em uma máquina, com cálculos complexos, elevando o custo computacional.

Com o objetivo de atender a crescente necessidade de se analisar tais sistemas, é que o engenheiro de controle ("controlista"), vem buscando uma alternativa na IA. Existe hoje uma "verdadeira revolução da informática", tendo à frente os países desenvolvidos (principalmente Japão e USA) aplicando enormes quantidades de recursos nesta área, principalmente na busca do computador neural, deixando os controlistas confiantes nas resoluções de tais problemas para um futuro bastante próximo.

Dentre as diversas áreas da IA para resolução dos problemas mais complexos no mundo do controle automático de sistemas, foi justamente o paradigma envolvendo Redes Neurais



Graduação.

Os objetivos principais deste trabalho, foram:

- # Apresentar uma introdução ao assunto de RNA's, e como elas podem ser aplicadas em controle automático de sistemas dinâmicos;
- # Apresentar uma melhoria ao BA através de um estudo heurístico, baseado nos paradigmas do construtivismo;
- # Apresentar aplicações de RNA's em controle automático de sistemas utilizando o conceito de controle inverso.



INFORMAÇÕES BÁSICA SOBRE RNA's

II.1 - INTRODUÇÃO

Este capítulo tem o objetivo de apresentar uma introdução teórica sobre Redes Neurais Artificiais (RNA) oferecendo ao leitor informações básicas para a perfeita compreensão dos próximos capítulos.

Um maior detalhamento do assunto pode ser encontrado na abundante literatura existente, como nos artigos clássicos [03], [04] e [05], ou em livros como [06].

II.2 - HISTÓRICO

O processo de modernização do mundo, vem exigindo tempos de processamento cada vez menores para qualquer atividade. O mundo moderno teve suas "distâncias" diminuídas, em função dos meios de transportes, e tem na indústria atividades extremamente delicadas ou complicadas, exigindo rapidez e precisão ao mesmo tempo. Na área da pesquisa tem-se os satélites, os ônibus espaciais, em fim poderia ser listado aqui uma enormidade de atividades em que o homem vem exigindo de si mesmo um grande desdobramento no sentido de aumentar seus conhecimentos a fim de atender essas novas necessidades.



É pensando nesta modernização que há muito, o homem tenta construir máquinas que possam substituí-lo em atividades complexas, as quais ele próprio levaria um tempo inaceitavelmente grande, ou até mesmo, não conseguiria realizar de forma satisfatória.

Neste sentido foram, sem dúvidas, as máquinas baseadas em arquitetura *Von Neuman*, o maior avanço já conseguido até o presente. Essas máquinas foram extensivamente estudadas e vem sendo progressivamente desenvolvidas no modo digital (processamento discreto), se transformando de enormes máquinas, nos atualmente tão conhecidos micro-computadores pessoais, ou simplesmente "micros" como são conhecidos.

O uso dessas máquinas se tornou obrigatório em todas os níveis da sociedade produtiva. Em um nível mais específico como os intelectuais, pesquisadores ou executivos, é necessário hoje que cada indivíduo tenha seu próprio exemplar, como se fosse um livro básico.

Diante do constante desenvolvimento, principalmente dos materiais eletrônicos que compõem os micros, quase sempre acompanhado de uma diminuição do custo, fica fácil arriscar uma previsão de que dentro de um curto prazo se terá pelo menos um micro-computador para cada casa, como se tem hoje o aparelho de televisão, por exemplo.

Estes tipos de máquinas resolvem problemas de forma exata, realizando milhões de operações matemáticas por segundo, ou considerando-se os chamados supercomputadores, bilhões de operações por segundo.

Porém há muitas décadas, antes mesmo da informática ter alcançado tal avanço, já se questionava como teria que ser uma máquina para resolver problemas básicos resolvidos naturalmente pelo homem como por exemplo reconhecimento de imagens. A pergunta seria: qual o algoritmo necessário para que uma máquina possa ser programada para reconhecer um rosto, com a mesma simplicidade que faz o cérebro ?

A motivação principal seria uma máquina com a velocidade de processamento dos computadores convencionais, ou mais rápidos, resolvendo aqueles problemas básicos com a mesma naturalidade do homem.

O primeiro desafio seria conhecer o cérebro e a dinâmica de seu processamento.

O cérebro humano é sem dúvida, uma potente "máquina" de tratamento de informações, e talvez quando o homem for capaz de analisar meticulosamente seu funcionamento será capaz também de criar programas equivalentes de tratamento dos símbolos pelo computador. Porém como se sabe o avião não bate as asas para voar, portanto, não se pode esperar que o computador tenha algum dia "intuição" e "imaginação" como o homem.



No entanto a máquina pode ser muito mais eficaz do que o homem para solucionar problemas específicos, pois enquanto no cérebro a velocidade de processamento está na ordem de milisegundos, nas máquinas está na ordem de nonosegundos, portanto, muito mais rápidos.

Neste sentido, o que se pode registrar, é que os avanços na área biológica foram também espantosos nas últimas décadas, e com o aprofundamento dos estudos na área neurológica, a área moderna da eletrônica aplicada à informática, ou a neurocomputação, pode tirar proveito, no sentido de obter informações mais precisas do funcionamento cerebral, aproximando seus modelos com uma acuracidade cada vez maior.

Logicamente está longe de se conhecer totalmente o funcionamento cerebral, mas já se pode dizer que com o que já se tem, muitos estudos e trabalhos na área da neurocomputação podem ser efetuados.

As primeiras tentativas de modelagem do cérebro foram feitas com base nos paradigmas cognitivos, que tentavam explicá-lo como um conjunto de procedimentos heurísticos processados através da representação simbólica, fazendo bastante uso da automação. Este conceito é empregado, também dentro da área de IA na sub-área denominada sistemas especialistas, onde se tenta passar para um algoritmo, o conhecimento de um especialista humano, conhecedor profundo das atividades a serem desenvolvidas.

O especialista forma um banco de dados com seu conhecimento (uma base de conhecimentos) e a máquina acessa essas informações à medida que vai necessitando. Através de regras que determinam seu modo de busca, ela consegue atuar, imitando as atitudes que teria o homem para aquela determinada situação.

Esse modelo deparou com muitos problemas, principalmente na tentativa de se representar o conhecimento de maneira que o mesmo pudesse ser acessado para ser combinado com outros, de forma que pudesse propor novas idéias, o que seria a síntese da inteligência. É um modelo que se apresentou eficiente apenas para resolver problemas conhecidos a priori, não tendo a capacidade de aprender, ou de guardar a experiência "vivida" anteriormente.

Apesar dos problemas, este tipo de máquina ocupa um espaço muito grande na indústria, sendo muito utilizado nas máquinas automáticas empregadas nos processos industriais, como usinagem e processos químicos, e está substituindo em muitos casos operadores humanos. Por isso, continuam sendo chamadas de máquinas "inteligente" e estão classificadas dentro da área de IA.

Diante das dificuldades procurou-se novos modelos para o cérebro, chegando aos



paradigmas conexionistas, de onde se baseia toda a teoria de RNA, que tem como idéia principal um número muito grande de processadores simples, conectados entre si, e que possuem a capacidade ou o poder de aprender, ou utilizar informações das experiências "vividas" anteriormente para resolver problema novos.

No início dos trabalhos havia uma dificuldade na comunicação entre as disciplinas envolvidas, ou seja, controle, ciências da computação (incluindo IA) e neurobiologia, seguiam por caminhos próprios [07] sem a interligação necessária. Isto proporcionou por muito tempo uma dificuldade na comunicação entre os pesquisadores das respectivas áreas. Atualmente esta distância foi bastante diminuída graças à importantes trabalhos publicados, como [07] e [08], que apresentam as idéias básicas e as técnicas envolvidas em RNA's em uma linguagem familiar, utilizando uma notação de engenharia de controle, e a homogeneização entre os pesquisadores das diversas áreas.

Os elementos básicos da arquitetura conexionista são chamados também de neurônios, pela analogia com a neurofisiologia, porém, outros nomes também são utilizados, como "perceptron" em [09] ou "adaline" em [10].

Na figura - 01 pode ser visto um modelo básico de neurônio.

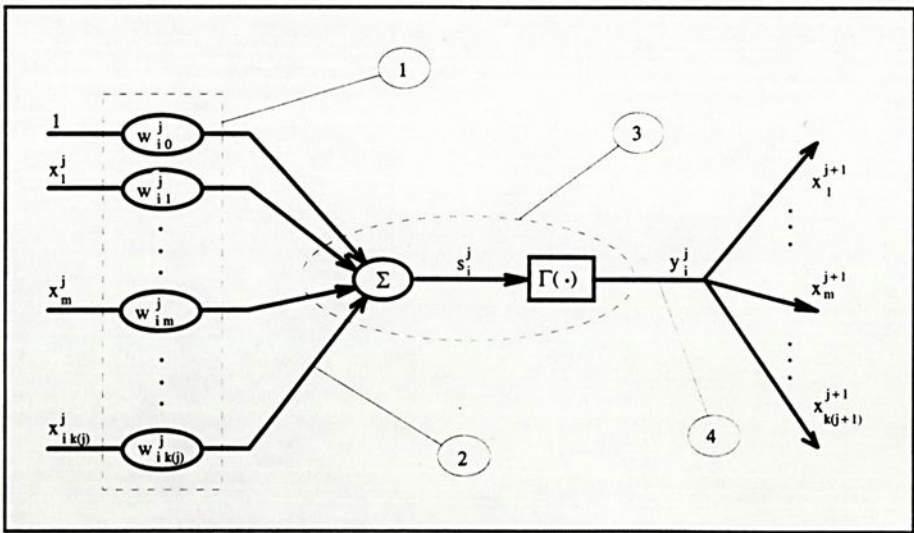


Fig. 01 - Modelo do neurônio de McCULLOCH-PITTS.

Este modelo é conhecido como modelo de McCULLOCH-PITTS, modelado em 1943 por W. S. McCulloch and W.H. Pitts [11].

O modelo descreve o neurônio dividindo-o em quatro partes básicas, que podem ser relacionadas com o neurônio biológico, como segue:

- (1) Parâmetros ajustáveis (denominados normalmente por pesos) os quais dão a intensidade do fluxo de informações entre os neurônios da rede, correspondem às sinapses ;
- (2) Ligações anteriores, que recebem as informações (entradas) a serem processadas, e correspondem aos dendritos;
- (3) Dois componentes de processamento, um somatório linear e uma função não linear, representando o corpo celular;
- (4) Ligações posteriores, que se ramificam, e possuem a função de fornecer informações (saídas) depois de processadas. São equivalentes aos axônios.

Segundo os neurobiologistas, o cérebro humano é constituído de bilhões de neurônios porém não são todos interligados entre si. Alguns cientistas da área acreditam que um neurônio interliga com 100 (cem), outros com 10.000 (dez mil) neurônios diferentes [12], o que significa que cada célula recebe informações de um número relativamente pequeno de outras.

Outro fato comprovado é que os neurônios localizados em partes diferentes do sistema nervoso tem funções diferentes e conseqüentemente possuem características diferentes [07].

Em [13] é citado por exemplo que os neurônios do sistema motor ocular possuem características "sigmoidal" enquanto aqueles localizados na área visual possuem características "Gaussian". Ou seja, os primeiros podem ser modelados matematicamente utilizando uma função tipo sigmóide, enquanto os outros por uma função de Gauss.

Isto mostra que as RNA's devem ser organizadas em seu modo de fluxo de informações e podem ser utilizados diferentes tipos de modelos de neurônios em uma mesma estrutura.

As funções não lineares representadas na Fig. 01 por $\Gamma(\cdot)$, utilizadas usualmente na literatura com mais frequência, estão apresentadas na tabela-I.



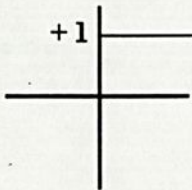
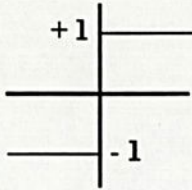
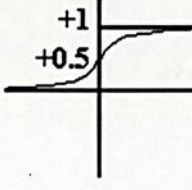
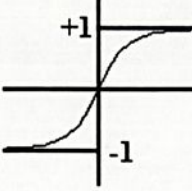
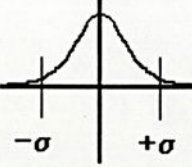
Nome	Fórmula	Curva	Características
Threshold	$+1$ se $x > 0$, 0 se $x \leq 0$.		Não diferenciável, degrau positiva.
Threshold	$+1$ se $x > 0$, -1 se $x \leq 0$.		Não diferenciável, degrau com valor médio zero.
Sigmóide	$\frac{1}{1 + e^{-x}}$		Diferenciável, degrau positiva.
Tangente hiperbólica	$\tanh(x)$		Diferenciável, degrau com valor médio zero.
Gaussian	$e^{(-x^2/\sigma^2)}$		Diferenciável, pulso.

Tabela-I - Funções não-lineares usadas com mais frequência nos modelos de neurônios.

Para alguns algoritmos de adaptação, como o "backpropagation", por exemplo, são necessárias funções diferenciáveis, outros necessitam de funções descontínuas que resultam em valores binários. Cada tipo de função possui então suas vantagens e desvantagens que o projetista da RNA tem que ter habilidade para poder analisar e escolher para cada caso, aquela que melhor

se encaixa no problema estudado.

Em uma RNA o que se observa é que cada neurônio por si só não possui grande capacidade em termos computacionais ou de representação. Porém a conexão entre eles é que proporciona uma alta capacidade para relacionar variáveis, fornecendo resolução para diversos tipos de processo.

O fato de que os neurônios biológicos não estão todos interligados entre si, é particularmente importante para a construção da rede artificial, pois se a saída de qualquer neurônio "i" estiver interligado com a entrada de qualquer neurônio anterior "j", haveriam malhas de realimentação, que certamente resultariam em problemas de estabilidade na hora do treinamento e/ou processamento.

II.3 - ESTRUTURAS DE RNA's

Em função do gigantesco número de pesquisadores trabalhando nesta área existem hoje uma infinidade de trabalhos publicados e consequentemente diversas estruturas e algoritmos de treinamento para as RNA's.

Suriram na década de 80 (oitenta) principalmente várias estruturas novas, algumas baseadas em novos pontos teóricos (redes de HOPFIELD) outras combinando técnicas já existente (redes COUNTERPROPAGATION), porém cada uma fornecendo, sob seu ponto de vista, melhorias na performance de treinamento.

Sendo um assunto que ainda pode ser considerado novo, certamente está carente de definições e teorias básicas que possam sustentar amplas generalizações como se espera de um assunto que se mostra tão promissor com possibilidades de aplicações em todas as ciências.

A principal classificação do assunto, divide as RNA's em dois grandes grupos que são, as redes não-recorrentes e recorrentes.

A característica principal das redes não-recorrentes é que as entradas, saídas e fluxo de informações são sempre bem definidos, garantindo a estabilidade da RNA na fase de treinamento.

Nestas redes os elementos básicos (neurônios) são posicionados em camadas, e os neurônios pertencentes a uma camada podem se comunicar somente com os neurônios pertencentes a camada seguinte.

A entrada da rede $X = [x_1, x_2, \dots, x_n]^T$ se comunica apenas com a primeira camada,



e a saída $Y = [y_1, y_2, \dots, y_m]^T$ apenas com a última camada, e entre a camada de entrada e a camada de saída existem uma ou mais camadas denominadas camadas intermediárias.

Estas camadas intermediárias servem para dar à rede uma representação interna independente dos sinais externos. É esta representação interna que permite à rede ter um razoável poder de generalização, e consequentemente, uma boa performance quando é apresentado como entrada, algo diferente daqueles padrões utilizados para o treinamento.

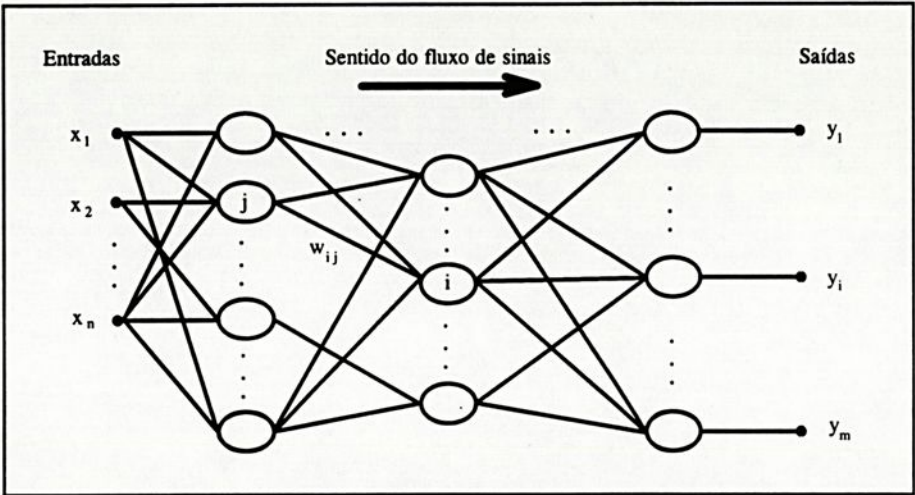


Fig. 02 - Esquema geral de uma RNA do tipo não-recorrente.

Este tipo de estrutura ficou por algum tempo sem aplicações práticas porque não havia nenhum método adequado para ajustar os parâmetros variáveis (pesos) das camadas intermediárias, para que a rede pudesse armazenar informações, ou seja, ser educada.

Nesta fase, ainda na década de sessenta (60) foi publicado o trabalho de Minsky e Papert [14], provando matematicamente a impossibilidade de se fazer o treinamento ou a adaptação dos pesos para redes "feedforward" multicamadas.

No início da década de 70 (setenta) Werbos em sua tese de mestrado [15] desenvolveu o método de treinamento denominado "backpropagation" (Apêndice-I), que inicialmente permaneceu sem o devido reconhecimento da comunidade científica, e, somente uma década depois é que esta técnica de adaptação dos pesos foi devidamente reconhecida, quando outros estudiosos publicaram trabalhos "redescobrimdo" o método [06].

Atualmente as RNA do tipo não-recorrentes treinadas através do algoritmo



"backpropagation" representam em torno de 90% (noventa por cento) dos trabalhos da área. Existe um interesse muito grande atualmente, por parte dos engenheiros de controle, neste tipo de rede, para aplicações em sistemas não-lineares, pois estas redes desempenham excelentes mapeamentos de funções não-lineares. Redes com uma camada intermediária se mostram hábeis para efetuar mapeamentos não-lineares, porém, utilizando um número maior de camadas intermediária, podemos ter a vantagem de uma diminuição considerável no número total de neurônios intermediários [16] e consequentemente do número total de neurônios da rede.

As redes do tipo recorrentes tem como principal característica o fato de possuir um caminho "feedback" que liga as saídas anteriores da RNA às suas entradas atuais, proporcionando desta forma um comportamento dinâmico da RNA. Porém em muitos casos este comportamento dinâmico pode levar a instabilidade da RNA.

A Fig. 03 mostra um esquema deste tipo de RNA.

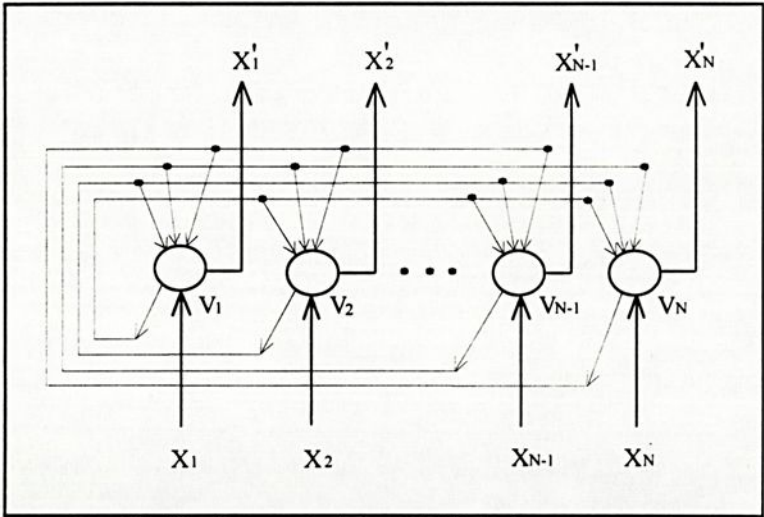


Fig. 03 - Esquema geral de uma RNA do tipo recorrente.

Onde:

- $X_i \Rightarrow$ indica o valor inicial do i-ésimo neurônio;
- $V_i \Rightarrow$ indica o estado do i-ésimo neurônio;
- $X'_i \Rightarrow$ indica o valor da saída do i-ésimo neurônio depois da convergência.

Muitos pesquisadores trabalharam tentando definir quais RNA's possuem comportamento estável e quais são instáveis. A publicação de um teorema por Cohen e Grossberg em 1983, que teoricamente assegura a divisão das RNA's recorrentes em estáveis e instáveis, foi a porta para pesquisas futuras nesta linha [06].

A seguir são apresentadas algumas das principais estruturas encontradas na literatura.



IL3.1 - "PERCEPTRONS"

Esta estrutura é a mais simples que existe, e também a primeira RNA estudada. Trata-se de uma RNA não-recorrente, consiste de uma única camada com um único neurônio, que é conectado através de pesos à um conjunto de entradas. A função de ativação é do tipo "threshold", e o treinamento é supervisionado, utilizando normalmente a regra delta como algoritmo de treinamento.

Este tipo de RNA foi utilizado inicialmente para reconhecimento de padrões, sendo que o maior desenvolvimento foi no início da década de 60 (sessenta) [09], [10] e de [17] a [19].

IL3.2 - "FEEDFORWARD"

Este modelo é o que melhor representa uma RNB. É uma rede não-recorrente, utilizando como função de ativação, funções contínuas, do tipo sigmóide ou tangente hiperbólica. O treinamento é supervisionado e o algoritmo utilizado é o "Backpropagation".

Dentro desta estrutura estão sem dúvidas os resultados mais contundentes já conseguidos até o momento para aplicações práticas de RNA's.

A nível de controle, o fato de que estas redes possuem originariamente comportamento estático, aparentemente seria difícil sua utilização em sistemas dinâmicos.

Porém apareceram muitas estruturas utilizando RNA's não-recorrentes que foram utilizadas com sucesso junto a teoria de controle de sistemas quebrando esta barreira.

Foi proposto um algoritmo de treinamento denominado "backpropagation dinâmico", colocando um caminho "feedback" realimentando a rede, funcionando como um filtro de estados e dando um comportamento dinâmico a rede de [20] a [22].

Esta estrutura conhecida como RNA's generalizadas pode ser sintetizada na idéia básica conforme apresentado esquematicamente na Fig. 04.



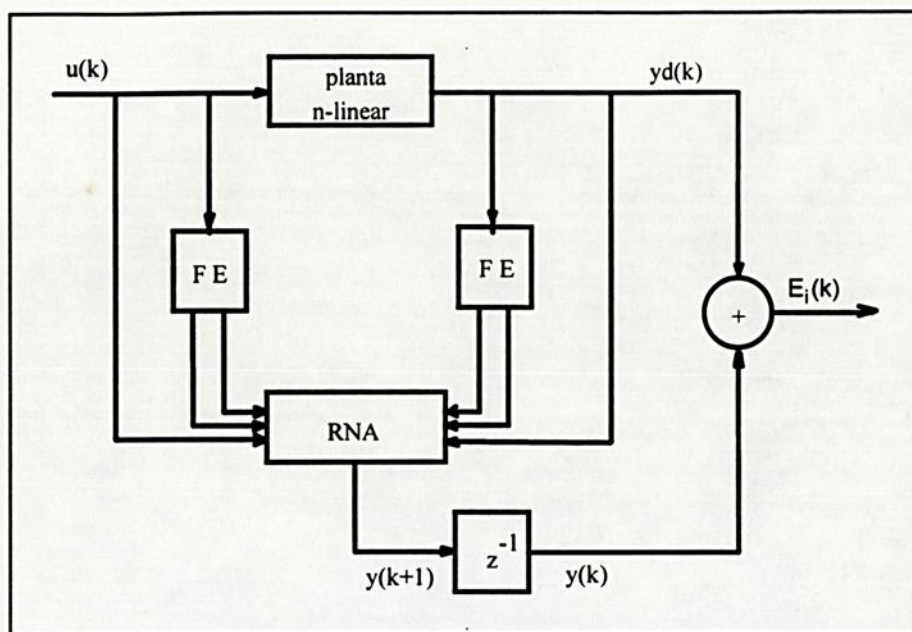


Fig. 04 - Estrutura geral de RNA's generalizadas.

Também já existem algumas propostas para transformar as redes não-recorrentes (estáticas) em sistemas dinâmicos. Em [12] foi proposto um algoritmo de treinamento denominado "Backpropagation Throught Time" que transforma as redes não-recorrentes de estáticas em dinâmicas, colocando memória no interior da rede de forma que os valores atrasados dos parâmetros de cada neurônio (entradas, saídas e pesos) possam ser utilizados no cálculo atual, Fig. 05.

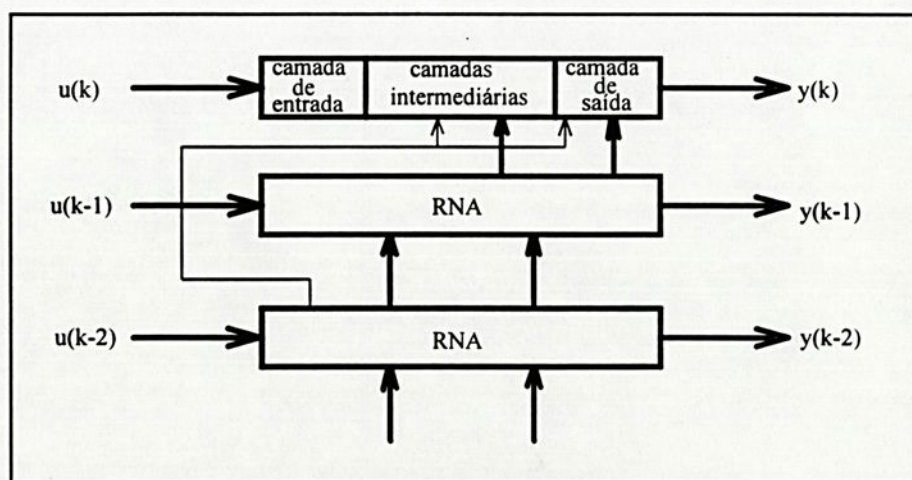


Fig. 05 - Estrutura geral de RNA's com memória.

A principal dificuldade das RNA's tipo "feedforward" está no tempo de treinamento que normalmente é exageradamente grande, principalmente para aplicações de modo "on line". Muitos trabalhos no sentido de melhorar o BA principalmente tentando minimizar este tempo de treinamento foram publicados nos últimos tempos [23] a [31]. Porém, pelo levantamento bibliográfico efetuado, os melhores resultados foram obtidos com a técnica do fator de momento [02] e [29].

IL3.3 - "COUNTERPROPAGATION"

Esta estrutura é resultante da composição das redes de Kohonen [32] e de Grossberg de [33], juntas conseguem propriedades que não possuem individualmente.

É uma concepção baseada na idéia de que o cérebro possui vários módulos, interligados em forma de cascata, onde cada módulo é especialista em resolver problemas específicos, dando a solução final desejada para o problema global.

IL3.4 - MÉTODOS ESTATÍSTICOS

Os métodos estatísticos podem ser utilizados tanto na fase de treinamento, como no monitoramento das saídas de uma RNA treinada anteriormente por outros métodos.

Os itens anteriores são usualmente treinados por métodos determinísticos, onde o procedimento é ajustar os pesos da RNA passo-a-passo, considerando os valores atuais dos pesos, das entradas, saídas e os valores desejados das saídas.

No método estatístico normalmente é adotado algum procedimento pseudorandômico para alterar os pesos, considerando as alterações que resultem em melhorias no comportamento da RNA, comparando a saída da RNA com a saída desejada.

É um conceito que pode ser utilizado em conjunto com outras técnicas, principalmente quando o treinamento apresenta problemas de mínimo local.



II.3.5 - HOPFIELD

É o principal tipo de RNA pertencente à classe recorrente com um certo potencial para ser utilizado em sistemas dinâmicos, justamente pelo seu comportamento interno que é dinâmico.

Esta nomenclatura, se deve aos importantes trabalhos teóricos e também de aplicações publicados por John Hopfield na década de 80, nesta linha.

A função de ativação utilizada inicialmente, foi do tipo "threshold", porém, podem ser utilizadas também as funções contínuas. Existem também as combinações deste tipo de RNA com processos estatísticos.

II.3.6 - MEMÓRIAS ASSOCIATIVAS

É uma concepção baseado no fato de que a memória humana em muitos casos recorre a um determinado acontecimento para lembrar-se de outro, ou seja, associa um fato a outro.

É um tipo muito utilizado para reconhecimento de padrões, a RNA consegue identificar padrões a partir de informações com ruídos, por exemplo.

Em controle de sistemas, atualmente esta concepção de RNA's vem ganhando espaço principalmente através da estrutura de redes denominada CMAC [34] a [40].

II.4 - CONCLUSÕES

O fator fundamental para que se iniciasse as pesquisas na área de RNA's, foi sem dúvidas, a modelagem do neurônio biológico conseguido na década de 40 (quarenta). Porém, a descoberta do BA foi o que proporcionou esta verdadeira "explosão" que estamos vivendo hoje, nesta linha, em termos de trabalhos publicados. Podendo ser considerado um dos assuntos mais promissores pela sua interdisciplinaridade, despertando interesse em pesquisadores do mundo todo.

Embora o objetivo principal seja comum, "imitar" o comportamento do cérebro, muitas estruturas e algoritmos diferentes são utilizados para isso.



PROPOSTAS PARA MELHORIA DO BA

III.1 - INTRODUÇÃO

O algoritmo BA é o método "standard" para treinamento de RNA's com múltiplas camadas, porém sua convergência é muito lenta e também pode parar em um mínimo local. Alguns métodos encontrados na literatura com o objetivo de minimizar tais problemas, são citados. Apenas com a pretensão de registrar a existência destes métodos, são apresentados resumos rápidos dando uma idéia geral desta linha de pesquisa.

É proposto em seguida um novo método, também com o objetivo de melhorar o BA. O método foi inspirado no construtivismo, uma nova metodologia de alfabetização, proposto por Emília Ferrero, baseado na filosofia de Piaget.

III.2 - MÉTODOS ENCONTRADOS NA LITERATURA

O BA normalmente citado como responsável pela retomada dos estudos na área de RNA é também frequentemente considerado, um algoritmo lento na convergência, além dos problemas quanto às especificações dos parâmetros envolvidos no algoritmo (taxa de aprendizado, valor dos pesos iniciais, etc) e no dimensionamento da RNA (N° de camadas, N° de neurônios em cada camada, etc).



Não existem métodos que resolvam estes problemas com a eficiência desejada. Os cientistas esperam encontrar soluções em breve, ou seja, a RNA deverá possuir a característica de auto-organização, e para isto, serão necessárias novas técnicas.

Com este objetivo estão sendo publicados muitos trabalhos, principalmente procurando novas alternativas para se especificar a taxa de aprendizado e o termo de momento, de maneira variável e com ajuste automático, tirando do projetista a tarefa de adotar estes valores através do método de tentativa e erro.

Estas pesquisas podem ser divididas em duas categorias: a primeira envolve o desenvolvimento de novas técnicas (variação da taxa de aprendizado, termo de momento, "rescaling" variáveis, etc); a segunda envolve técnicas de otimização numérica "standard" buscando melhorias para o BA [31].

Antes de apresentar o método que desenvolvemos, colocamos alguns resumos rápidos para que se tenha uma idéia do que está sendo proposto na literatura com o objetivo de melhorar o treinamento das RNA's do tipo "feedforward". Logicamente estes resumos não tem como objetivo detalhar cada um dos trabalhos, simplesmente estão sendo apresentados para que o leitor tenha uma idéia, mesmo que bastante superficial, sobre os trabalhos desenvolvidos nesta linha. Também servirá para mostrar que o trabalho que desenvolvemos partiu de uma idéia totalmente nova e esperamos que com o aperfeiçoamento de alguns pontos, possa fornecer grandes contribuições ao assunto.

A contribuição mais significativa em termos de acelerar o tempo de treinamento do BA foi sem dúvidas a técnica do termo de momento [02]. Esta técnica inclui um termo de proporcionalidade referente a cada componente do vetor peso, calculados em algumas iterações anteriores ($k-n$), para efetuar o cálculo da k -ésima iteração, para cada peso. O mais usual é considerar $n=1$, ou seja, considera-se somente o termo de proporcionalidade referente à iteração anterior (Apêndice - II). Porém esta técnica introduziu mais um parâmetro a ser definido pelo projetista (o fator de momento η), é realmente um método eficiente para muitos problemas, mas apresenta um efeito pequeno ou negativo em outros [06].

Em [28] é apresentado um algoritmo que faz o termo de momento variável, com adaptação sendo feita automaticamente para cada passo de iteração. A idéia é trabalhar com o gradiente ótimo em cada iteração. Em função da complexidade da estrutura não-linear, é muito difícil obter o vetor do gradiente ótimo, por métodos analíticos, porém pode ser provado que os

vetores do gradiente ótimo são ortogonais em duas iterações consecutivas [28]. Baseado nesta afirmação é utilizado o método de ortogonalização de Gram-Shmidt, para assegurar a ortogonalidade dos dois vetores sucessivos para cada iteração, o que resulta em algo equivalente a colocação de um termo de momento adaptativo adicionado ao BA "standard".

Foi proposto em [25] um ajuste dinâmico dos valores da taxa de aprendizado e termo de momento durante o aprendizado. Estes valores são ditos dependentes do erro e da derivada da função de ativação para cada elemento, isto permite que os valores dos parâmetros sejam distintos e assumam valores ótimos para cada elemento a ser reajustado.

Em [31] foi apresentado o algoritmo denominado "Marquardt Algorithm" como um novo algoritmo para o treinamento de RNA's tipo "feedforward". A idéia é utilizar o fato de que a função objetivo a ser minimizada é uma função composta pela soma dos quadrados de outras funções não-lineares. A minimização de funções objetivo deste tipo sugere a utilização do método dos mínimos quadrados não-linear, que é utilizada no método apresentado. Este método de minimização de funções objetivo desconhecidas, mais conhecido como "Marquardt-Levenberg", é uma modificação do método de Newton.

Foi apresentado em [27] uma análise da influência negativa que o fator de momento pode ter no processo de convergência das RNA's. Em seguida é proposto um algoritmo, fazendo com que o fator de momento sofra uma atenuação estocástica. A idéia é fazer com que o fator de momento "caminhe" para zero de forma suave ao mesmo tempo que o treinamento é efetuado, de tal forma que, quando a convergência for alcançada o termo de momento será zero.

Em [30] foi apresentado uma regra para seleção de um coeficiente de aprendizado ótimo para o BA. Segundo o autor existe uma relação de dependência entre o coeficiente de aprendizado e o número de padrões, ou classes de padrões, encontrados no conjunto de padrões utilizado para o treinamento.

Foi apresentado em [24] um método para acelerar o treinamento feito pelo BA utilizando o termo de momento variável. A variação do termo de momento é feito através de uma técnica de otimização conhecida como método das tangentes paralelas.

Em [23] foi apresentado um algoritmo de treinamento que utiliza a idéia de minimizar o erro entre a saída desejada e a atual, com relação ao vetor de entrada. No BA standard se procura minimizar este erro também, porém, com relação ao vetor peso.

III.3 - CONCEPÇÃO DE UM NOVO MÉTODO

Baseados na insistente posição de que RNA's "imitam" o cérebro humano em seu modo de aprender e armazenar informações, tivemos a idéia de pesquisar como se comportam as crianças e quais os melhores métodos aplicados no início do aprendizado escolar. Pois, o que vemos, é que as crianças estão sendo alfabetizadas cada vez mais cedo e com mais facilidade.

Não sendo esta nossa área de atuação específica, resolvemos analisar o que estava acontecendo com nossos próprios filhos e as escolas que frequentavam, e tivemos assim contato com uma moderna metodologia proposta para o ensino, que vem sendo cada vez mais utilizadas nas escolas e pré-escolas do Brasil, o CONSTRUTIVISMO.

Segundo as pesquisas de Emília Ferrero, que se inspirou nas teorias do filósofo Piaget, convencionou-se chamar esta nova proposta de ensino de "construtivismo", porque a mesma oferece meios para que o próprio indivíduo construa seu conhecimento, com base na interação do sujeito com o meio [41].

Considerando este estilo de aprendizado, fortaleceu-se mais a idéia de um aprendizado gradativo para as RNA's, utilizando os paradigmas do construtivismo para treiná-las.

O objetivo então seria aumentar o grau de detalhamento das informações à medida que a RNA fosse aprendendo, de forma que se possa "respeitar" e aproveitar o quanto ela já aprendeu em um passo anterior.

Destas observações surgiram algumas idéias, e uma delas em particular pareceu-nos bastante interessante: a RNA poderia iniciar o aprendizado de forma superficial com a utilização de um filtro com parâmetros variáveis. À medida que houvesse informações confirmando o aprendizado, a influência do filtro poderia ser diminuída através da variação do parâmetro do filtro.

A influência de um filtro de primeira ordem, com o parâmetro variando do valor máximo (0.99), até o valor mínimo (0.0), pode ser entendida melhor, observando as Fig's. de 06 a 09.

Nas Fig's de 06 a 08 são apresentados diferentes resultados, representando o



comportamento da função $g(f(x))$, que, nada mais é do que, o filtro colocado na saída da função $f(x)$.

A Fig. 09 apresenta o comportamento da função $f(x)$, como realmente ela é, com bastante ruído. Neste momento não existe qualquer influência do filtro, ou seja, o filtro se comporta como um filtro "passa-tudo", é como se ele não existisse.

Cada figura, é portanto, o resultado de um valor diferente do parâmetro do filtro. A Fig. 06 é resultado da colocação de um filtro muito próximo de um "passa-nada", que seria o filtro com valor do parâmetro igual a 1.0 (um), as Fig's 07 e 08 são passos intermediários, entre os valores mínimo e máximo do parâmetro do filtro.

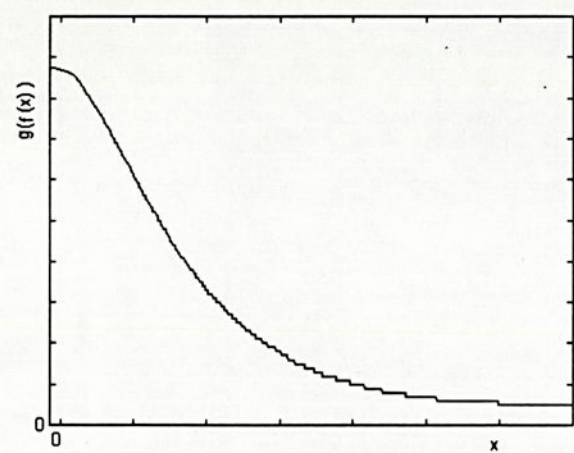


Fig. 06 - Saída do filtro com parâmetro 0.99.

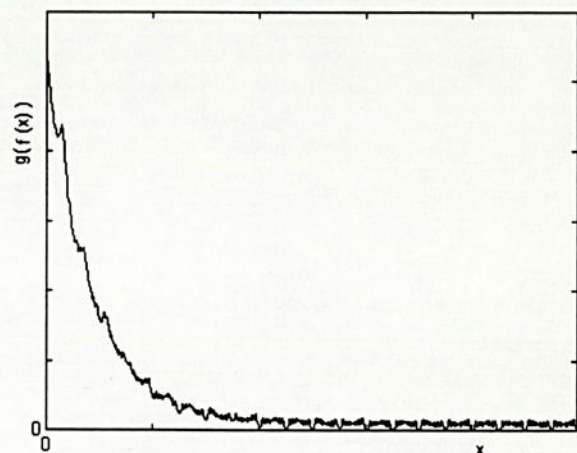


Fig. 07 - Saída do filtro com parâmetro 0.8.

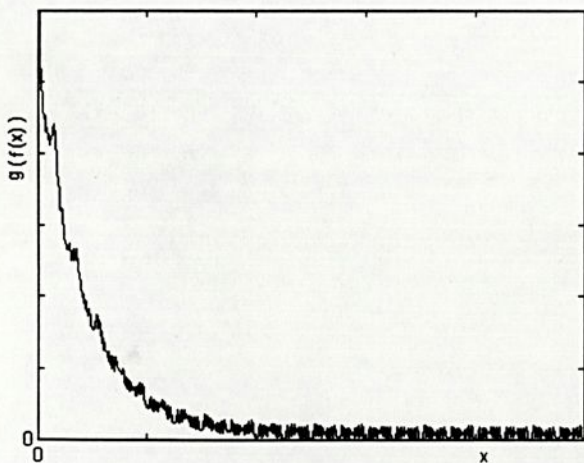


Fig. 08 - Saída do filtro com parâmetro 0.2.

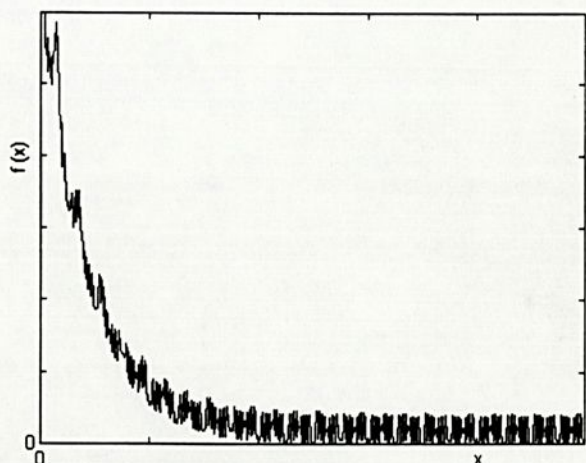


Fig. 09 - Saída da função sem o filtro.

A idéia então seria utilizar o filtro, de maneira que no início seria dada a RNA uma informação mais detalhada do que realmente é, e a medida que a RNA for aprendendo, as informações vão caminhando em direção à realidade desejada.

Definimos uma estrutura, que pode ser vista na Fig. 10, onde o objetivo é fazer a RNA reproduzir a função $f(\cdot)$, desconhecida. O erro após ser calculado, passa por um filtro passa-baixa de parâmetros variáveis, que tem a função de amenizar as oscilações bruscas existentes na curva do erro. Como o filtro possui estrutura variável, o que se pretende é utilizar uma informação de erro mais branda para o início do aprendizado, conforme já comentado.

Esta estrutura necessita da informação de quando e quanto variar o parâmetro do filtro, para que seja favorável ao aprendizado.

No construtivismo, o aprendiz somente passa para uma etapa seguinte do aprendizado, quando já domina a etapa atual. Para o caso do treinamento das RNA's equivale a dizer que o parâmetro do filtro deve ser reajustado somente quando há garantias de que ela já aprendeu o estágio atual.

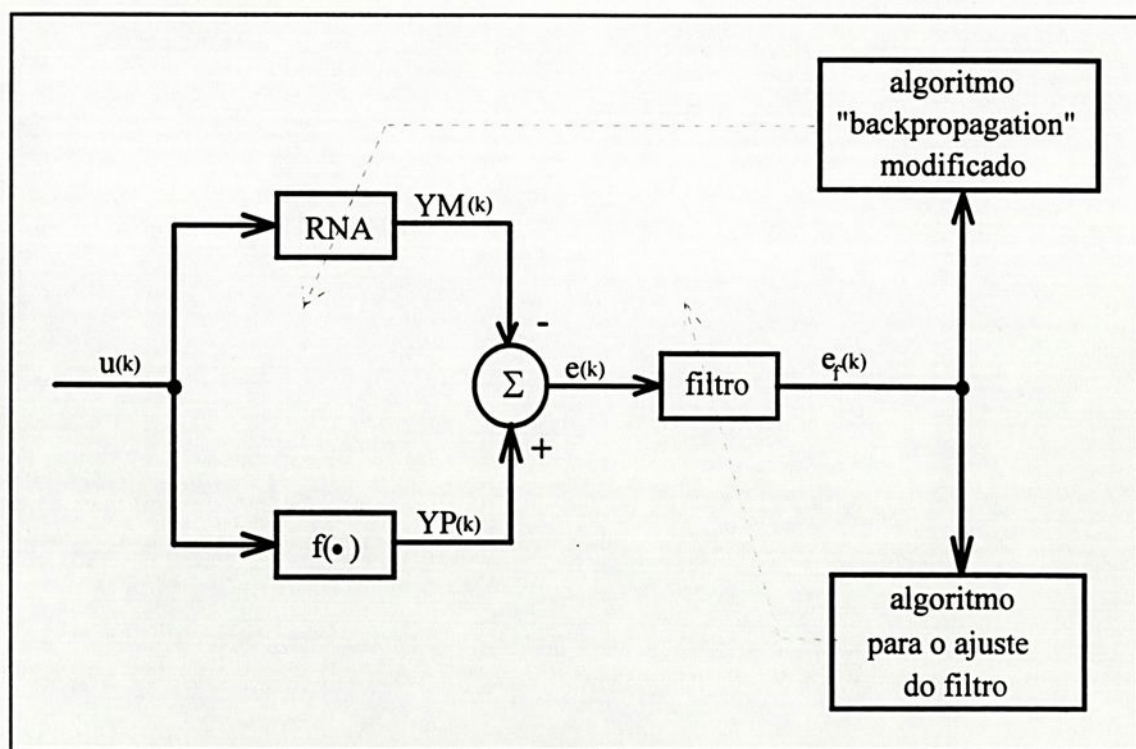


Fig. 10 - Estrutura de aprendizado do método proposto

A primeira idéia foi simplesmente analisar a diferença entre a média dos módulos de dois erros filtrados entre uma iteração e outra, de maneira consecutiva. Obviamente que, se a média do módulo do erro filtrado atual for menor do que a anterior, a rede está aprendendo e portanto está apta a receber informações com um detalhamento maior, ou seja, está pronta para se aprofundar mais no aprendizado.

O filtro utilizado nas simulações foi linear e de primeira ordem. A função transferência discreta (utilizando a transformada "z") é a seguinte:

$$G(z) = \frac{(1 - e^{-aT})z}{z - e^{-aT}}, \quad (01)$$

definindo:

$$b_p \triangleq e^{-aT}, \quad (02)$$

$$G(z) \triangleq \frac{E_f(z)}{E(z)}, \quad (03)$$

onde: b_p é o parâmetro variável do filtro;

$E_f(z)$ é a transformada-z da saída do filtro (Fig. 10);

$E(z)$ é a transformada-z da entrada do filtro (Fig. 10).

A equação (01) pode ser então reescrita como:

$$\frac{E_f(z)}{E(z)} = \frac{(1 - b_p)z}{z - b_p}. \quad (04)$$

Aplicando a anti-transformada-z, pode-se escrever a equação (04) na forma de equação diferença, como segue:

$$e_f(k+1) = b_p e_f(k) + (1-b_p) e(k+1), \quad (05)$$

onde: $e_f(k+1)$ é a saída do filtro (erro filtrado) no instante $(k+1)$;

$e_f(k)$ é, logicamente, a saída do filtro (erro filtrado) no instante anterior (k) ;

$e(k+1)$ é a entrada do filtro (erro real) no instante $(k+1)$.

b_p é o parâmetro do filtro no p -ésimo ciclo de apresentação das entradas padrões;

Analizando a equação (05), percebe-se que o parâmetro " b_p ", deverá variar no intervalo $(1, 0]$. Quando " $b_p = 0$ " o erro filtrado $e_f(k+1)$, é o próprio erro real calculado naquele instante, entre a saída desejada (YP) e a saída da RNA (YM). O filtro seria do tipo passa-tudo e, " b_p " não pode assumir o valor 1 (um), porque neste caso o erro filtrado nunca iria se alterar e, seria um filtro do tipo "passa-nada".

Com isso, no regime se teria o erro filtrado igual ao erro real e espera-se que a RNA neste momento tenha conseguido identificar a planta.

Conforme já citado anteriormente, para o ajuste do parâmetro necessita-se da informação se a RNA está aprendendo ou não.

A maior dificuldade deste método está, portanto, no ajuste de " b_p ", ou seja, quando e como devemos ajustar " b_p ".

Nós sabemos que o ajuste do " b_p " deve ser feito quando a RNA aprende. Existem diversos caminhos para obter esta informação e nós usamos a diferença da média do módulo do erro filtrado $(|e_f(k)|)$ para dois ciclos consecutivos de apresentações de todos os valores distintos do conjunto de entradas padrões.

Durante as simulações nós notamos que o ajuste de " b_p " deveria ser menor que os ajustes dos pesos.

Nós utilizamos a seguinte lei:

$$b_{p+1} = b_p - \frac{\gamma}{2} \left[1 - \text{sgn} \left(M_p(e_f) - M_{p-1}(e_f) \right) \right], \quad (06)$$



onde: γ é o parâmetro de ajuste de de " b_p ", $\gamma > 0$ e constante;

$M_p(e_f)$ é a média de $|e_f|$ para o p-ésimo ciclo de apresentação de todos os valores distintos do conjunto de entradas padrões;

$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0. \end{cases}$$

O valor de γ foi obtido através de tentativas e erros, tal que $b_p = 0$ algumas iterações antes da estabilização do mínimo erro quadrático ocorrer.

Algumas técnicas para automatizar este ajuste estão sendo estudadas, e serão propostas em trabalhos futuros.

A principal idéia é fazer um controlador através de lógica "fuzzy" para ajustar o parâmetro " b_p ", utilizando as experiências que tivemos durante as simulações.

Para este trabalho, o ajuste dos pesos na Fig.10, considerando uma única saída, é feito utilizando o algoritmo BA para a seguinte função energia:

$$E_f(k) = \sum_{i=k-(m-1)}^k e_f^2(i) , \quad (07)$$

onde: m é o número de termos calculados anteriormente, ou seja, nas iterações anteriores, que se queira considerar.

Em nosso caso, consideramos $m=1$, ou seja, consideramos apenas o termo calculado na k -ésima iteração para definir a função energia. Portanto, podemos eliminar a notação de somatório.

Tendo que o gradiente é dado pela derivada da função energia com relação aos pesos, substituindo (05) em (07), e considerando " b_p " constante durante a apresentação de um ciclo completo de todas as entradas padrões, podemos escrever cada componente do novo gradiente, conforme segue:



$$\frac{\partial E_f(k)}{\partial w_{ij}(k)} = 2 e_f(k) (1 - b_p) \frac{\partial e(k)}{\partial w_{ij}(k)} , \quad (08)$$

onde $w_{ij}(k)$ é um peso da RNA, na k -ésima iteração. Então, nós usamos (08) no BA para ajustar os pesos durante o treinamento, da RNA envolvida.

Utilizando esta nova função energia, dado por (07), e consequentemente o novo gradiente, cujas componentes são dadas por (08), a dedução matemática (Apêndice - III) mostra que apenas o erro derivativo sofrerá uma pequena alteração, para a última camada,

$$\text{BA "standard"} \quad \Rightarrow \quad \delta_i^l = e(k) \Gamma(s_i^l) ,$$

$$\text{BA modificado} \quad \Rightarrow \quad \delta_i^l = e_f(k) (1-b_p) \Gamma(s_i^l) .$$

III.4 CONCLUSÕES

Existe uma preocupação com relação à otimização do BA, que pode ser notada através dos trabalhos publicados nesta área. Isto é perfeitamente justificável em função da enorme importância que este algoritmo de treinamento possui.

Muitos pesquisadores acreditando no potencial deste algoritmo, continuam a estudá-lo, buscando alternativas de otimização numérica dos conceitos já existentes, e desenvolvimento de novas técnicas.

Nesta linha estamos propondo uma pequena alteração no BA, utilizando conceitos de uma nova metodologia de ensino, que vem sendo utilizada na alfabetização de crianças, com muito sucesso. Os resultados dos estudos deste e do próximo capítulo, foram apresentados no *"38TH MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS"* promovido pelo IEEE (Apêndice - IV).



APLICAÇÕES DO MÉTODO PROPOSTO

IV.1 - INTRODUÇÃO

A configuração proposta para o treinamento de RNA's, foi utilizada em problemas de identificação de funções não-lineares, através de simulações em computador digital.

Os resultados das simulações mostram que a configuração proposta usualmente, obtém um menor erro quadrático final médio, quando comparado com o BA "standard" e com BA com o fator de momento.

IV.2 - RESULTADOS DAS SIMULAÇÕES

Para as simulações elaboramos nosso próprio "software" (fluxograma resumido no Apêndice - V). Este programa foi elaborado inicialmente para resolver problemas de identificação de funções, utilizando o BA "standard" ou o BA com termo de momento.

Para a programação foi necessário um total detalhamento das equações e variáveis envolvidas, que necessitam de uma complexa notação (Apêndice - I).

O fato de termos elaborado o programa possibilitou uma maior familiarização com o BA, e permitiu que se fizesse a alteração proposta no item III.3, com mais facilidades. Esta alteração foi incluída no programa como uma alternativa de execução (subrotina)



O método proposto, o método BA "standard" e o BA com o fator de momento foram aplicados na identificação de funções consideradas desconhecidas de forma que fosse possível a comparação dos resultados obtidos em cada um dos métodos.

Extensivas simulações numéricas foram feitas de forma que os resultados apresentados são os melhores conseguidos para cada caso. Isto é, para cada método e para cada função, foram utilizados os parâmetros que forneceram seus melhores resultados.

As funções a serem identificadas em cada exemplo representam plantas dinâmicas. Estas funções são utilizadas para gerar as saídas desejadas segundo as entradas escolhidas, formando o conjunto de pares entradas/saídas padrões, utilizado para o treinamento.

Em casos reais de identificação, esse conjunto padrão, seria obtido através de sensores colocados na entrada e saída da planta.

IV.2.1 - EXEMPLO 1 - Considere a função dada por:

$$f[u(k)] = 0.6\sin[\pi u(k)] + 0.3\sin[3\pi u(k)] + 0.1\sin[5\pi u(k)], \quad (09)$$

com $u(k) \in [-1.0, 1.0]$.

Foi utilizada uma RNA de três camadas, com a seguinte distribuição de neurônios:

7 na primeira camada;

5 na segunda;

1 na terceira camada.

Caso 1 - BA "standard" com coeficiente de aprendizado $\mu = 0.02$, termo de momento $\eta = 0$;

Caso 2 - BA com fator de momento utilizando coeficiente de aprendizado $\mu = 0.1$, termo de momento $\eta = 0.8$;

Caso 3 - BA utilizando o erro filtrado (método proposto) com coeficiente de aprendizado $\mu=0.05$, termo de momento $\eta = 0$ e coeficiente para ajuste do parâmetro do filtro $\gamma=0.003$.

A Fig. 11 apresenta o erro quadrático médio para cada caso e a Fig. 12 apresenta a saída desejada e a saída obtida pela RNA para cada caso.



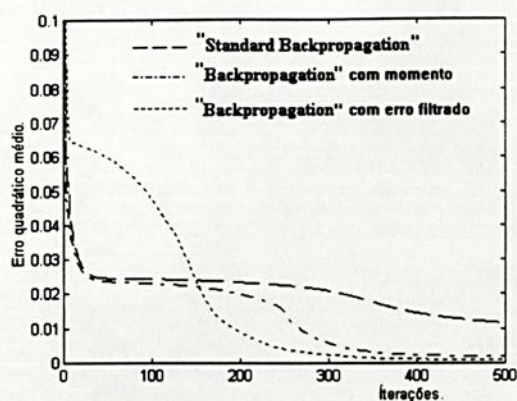


Fig. 11 - Performance dos três métodos.

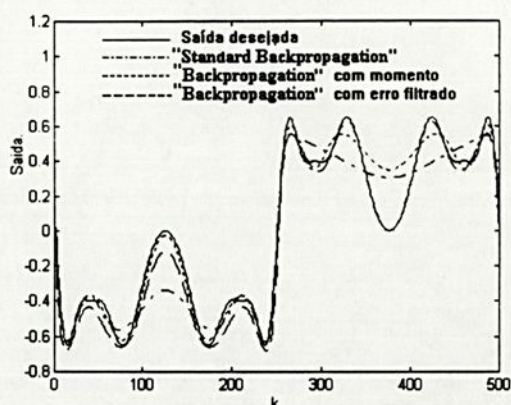


Fig. 12 - Comparação dos resultados obtidos pelos três métodos.

Pode ser observado das figuras, que o método proposto apresenta o melhor resultado final, e neste caso também convergiu mais rapidamente.

IV.2.2 - EXEMPLO 2 - Considere a função dada por:

$$f[u(k)] = 0.80 \sin[\pi u(k)(1.41u(k) + 1.59)] - 0.05u^2(k), \quad (10)$$

com, $u(k) \in [-1.0, 1.0]$.

Para este exemplo foi utilizada uma RNA também com três camadas, com a seguinte distribuição de neurônios:

- 5 na primeira camada;
- 3 na segunda camada;
- 1 na terceira camada.

- Caso 1** - BA "standard" com coeficiente de aprendizado $\mu = 0.07$ e termo de momento $\eta = 0$;
- Caso 2** - BA com fator de momento utilizando coeficiente de aprendizado $\mu = 0.05$ e termo de momento $\eta = 0.6$;
- Caso 3** - BA utilizando erro filtrado com coeficiente de aprendizado $\mu = 0.05$, termo de momento $\eta = 0$ e coeficiente para ajuste do parâmetro do filtro $\gamma = 0.003$.

A Fig. 13 apresenta o erro quadrático médio para cada caso, e a Fig. 14 apresenta a saída desejada e a saída obtida pela RNA para cada caso.

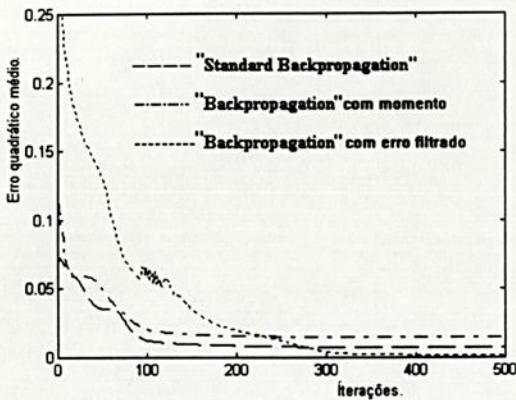


Fig. 13 - Performance dos três métodos.

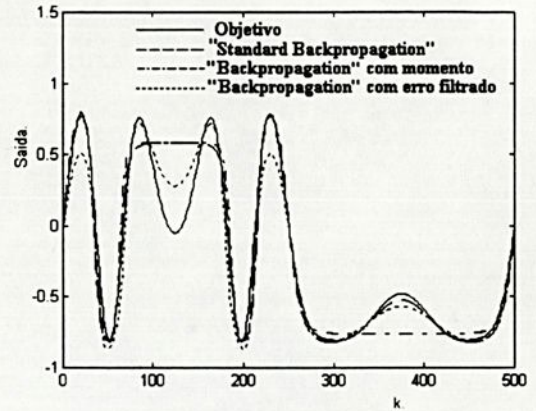


Fig. 14 - Comparação dos resultados obtidos pelos três métodos.

Pode ser observado das figuras que o método proposto apresenta o melhor desempenho final também neste caso.

IV.2.3 - EXEMPLO 3 - Considere a função dada por:

$$f(x, y) = \sin\left(\frac{2\pi x}{x_{\max}}\right) \sin\left(\frac{2\pi y}{y_{\max}}\right), \quad (11)$$

com $0.0 \leq x, y \leq 19.0$ na fase de treinamento e $0.0 \leq x, y \leq 49.0$, para verificação do desempenho de cada método.

- Caso 1 -** BA "standard" com coeficiente de aprendizado $\mu = 0.02$ e termo de momento $\eta = 0$;
- Caso 2 -** BA com fator de momento, utilizando coeficiente de aprendizado $\mu = 0.1$ e termo de momento $\eta = 0.8$;
- Caso 3 -** BA utilizando erro filtrado, com coeficiente de aprendizado $\mu = 0.05$, termo de momento $\eta = 0$ e coeficiente para ajuste do parâmetro do filtro $\gamma = 0.003$.

A Fig. 15 apresenta a saída desejada gerada pela equação (11).

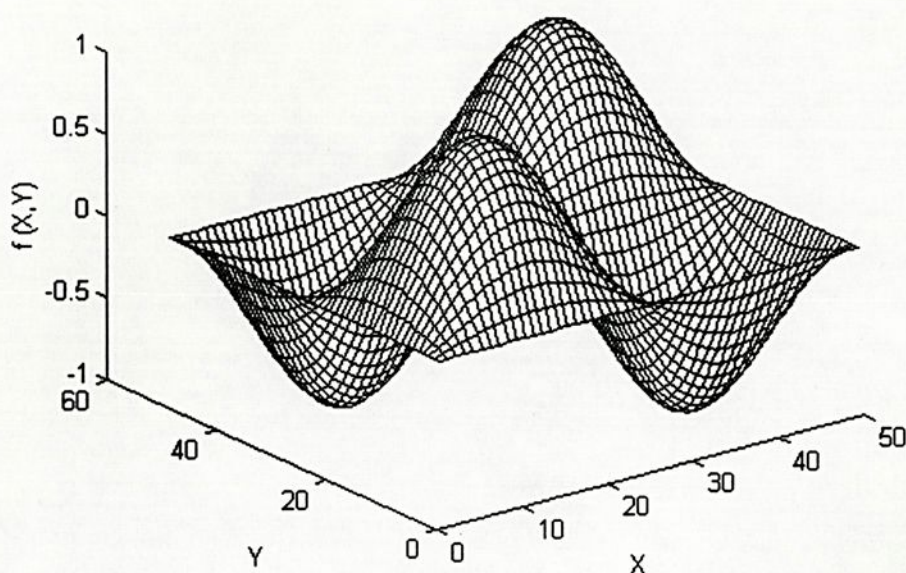


Fig. 15 - Grafico da função objetivo a ser identificada.

Este exemplo foi dividido em dois sub-ítems IV.2.3.1 e IV.2.3.2, sendo que no primeiro caso, a RNA escolhida está sub-dimensionada para resolver tal problema. Isto foi feito propositadamente com o objetivo de mostrar uma característica do método proposto, que observamos durante a fase de simulações dos exemplos. O método proposto, usualmente, utiliza melhor o potencial da RNA.

No segundo caso, agora com uma RNA maior, portanto melhor dimensionada para resolver o problema, para mostrar que uma uma equivalência dos resultados apresentados pelos três métodos.

Ressaltamos que para ambos os sub-ítems, o número de pares padrões (400) utilizados para o treinamento é relativamente pequeno.

IV.2.3.1 - EXEMPLO 3.1 - Neste primeiro caso foi utilizado uma RNA de duas camadas com a seguinte distribuição de neurônios:

5 na primeira camada;

1 na segunda camada.

A Fig. 16 apresenta o erro quadrático médio para cada caso e a Fig. 17, 18 e 19 apresenta a saída da RNA para cada caso.

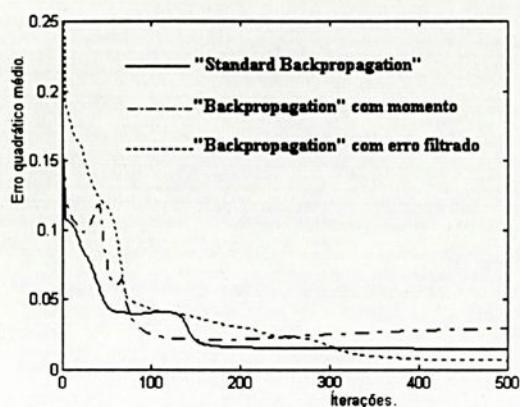


Fig. 16 - Performance dos três métodos.

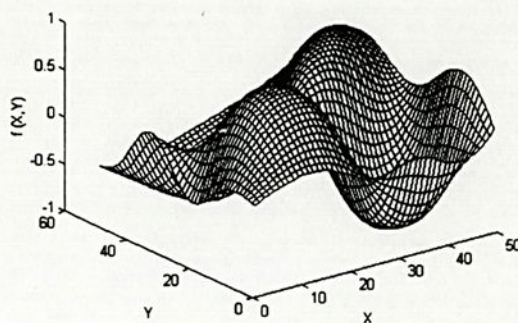


Fig. 17 - Saída da RNA treinada pelo BA "standard".

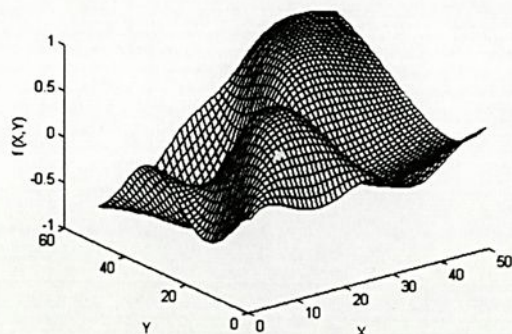


Fig. 18 - Saída da RNA treinada pelo BA com momento.

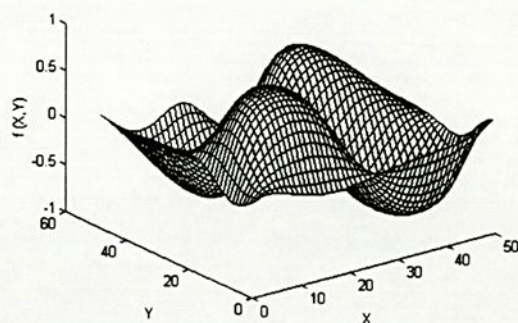


Fig. 19 - Saída da RNA treinada pelo BA com erro filtrado.

Pode ser notado da Fig. 16, que o método proposto apresenta um melhor desempenho final, para esta RNA escolhida. O melhor resultado, também pode ser confirmado ao obervarmos as Fig. 17, 18 e 19. As figuras mostram que, com a RNA utilizada, foram conseguidas identificações muito pobres, logicamente porque a RNA utilizada, está subdimensionada.

O objetivo deste exemplo foi justamente colocar a idéia de que o método proposto consegue extrair da RNA todo o seu potencial.

IV.2.3.2 - EXEMPLO 3.2 - Nós utilizamos agora uma RNA de três camadas com a seguinte distribuição de neurônios:

10 na primeira camada;

10 na segunda camada;

1 na terceira camada.

A Fig. 20 apresenta o erro quadrático médio para cada caso, e as Fig's. 21, 22 e 23 apresentam a saída da RNA para cada caso.

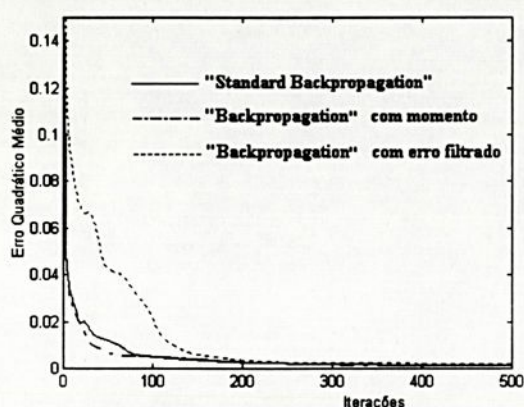


Fig. 20 - Performance dos três métodos.

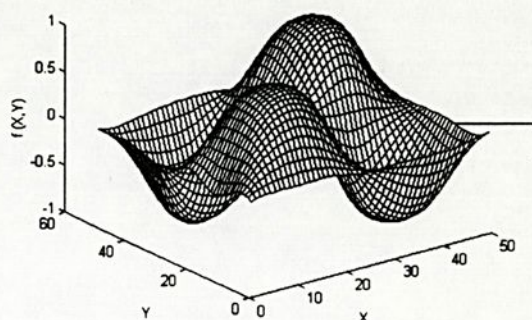


Fig. 21 - Saída da RNA treinada pelo BA "standard".

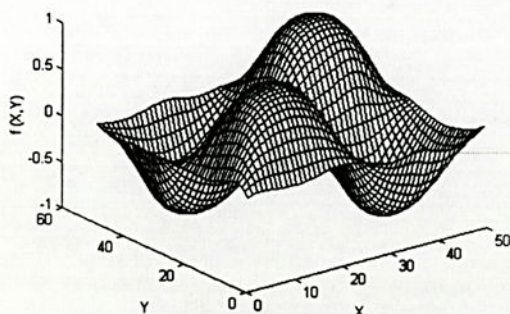


Fig. 22 - Saída da RNA treinada pelo BA com momento.

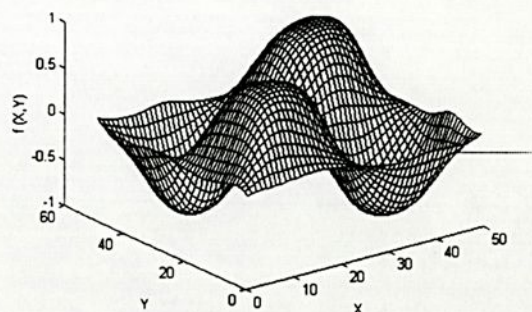


Fig. 23 - Saída da RNA treinada pelo BA com erro filtrado.

Observamos neste caso que os três métodos comparados, apresentam resultados finais semelhantes, e muito melhores do que no exemplo 3.1 em função da RNA estar melhor dimensionada neste caso.

IV.3 - CONCLUSÕES

As simulações mostraram que em alguns casos o método proposto é consideravelmente melhor, e que em outros acaba se igualando com os outros métodos.

Percebemos pelo exemplo 3, que a técnica do termo de momento pode até dificultar o aprendizado, o que parece não ocorrer com o método proposto. Isto se deve ao fato de que o filtro utilizado possui parâmetro variável, se anulando ao final do processo de treinamento.

Podemos dizer que, na pior das hipóteses quando o método proposto não ajuda, ele também não atrapalha.

Outro fator importante é o tempo de treinamento exigido para cada método.

As simulações foram efetuadas considerando sempre um número máximo de iterações (500), e não o erro quadrático mínimo, ou o tempo de treinamento. Em função disto, fizemos uma comparação do tempo gasto, para cada método efetuar o número máximo de iterações. O resultado foi muito positivo para o método proposto, pois, enquanto a técnica do fator de momento necessita cerca de 20% (vinte por cento) a mais do que o BA "standard", o novo método gasta em torno de 3% (três por cento) apenas, mais do que o BA "standard", em termos de tempo.

Isto já era esperado, porque a técnica do fator de momento, executa algumas operações matemáticas a mais para o ajuste de cada peso individualmente, enquanto que o novo método somente executa operações matemáticas a mais, ao final da apresentação do conjunto todo de entradas padrões, para ajustar apenas um parâmetro: b_p do filtro.



SISTEMAS DE CONTROLE UTILIZANDO RNA's

V.1 - INTRODUÇÃO

Este capítulo traz uma análise rápida, porém necessária, sobre alguns motivos que justificam a aplicação de RNA's em sistemas de controle.

Veremos que RNA's em suas diferentes estruturas são fortes ferramentas em potencial para modelagem, identificação e controle de sistemas não-lineares de múltiplas entradas e múltiplas saídas, onde são mais comumente utilizadas.

V.2 - CONTROLE DE PLANTAS DINÂMICAS

Controlar uma planta significa, alterar seu comportamento para que possa atingir certos objetivos desejados, sem alterar sua estrutura ou seus parâmetros. Para isso, se coloca normalmente componentes interligados à planta que possam alterar as entradas recebidas pela planta ou suas saídas.

Um dos principais problemas nesta área, é a estabilização de plantas que não se apresentam estáveis. Uma planta pode ser considerada estável se, dadas entradas comportadas, ou seja, dentro de uma região de interesse, a saída também se manterá de maneira comportada, ou seja, dentro de uma região de interesse para a saída.



Aqui estaremos mais preocupados em controlar plantas já estáveis. Neste tipo de problema, o projetista tem que fazer com que a planta forneça respostas com comportamento desejável dentro dos critérios estabelecidos pelo usuário.

A resposta temporal de um sistema dinâmico pode ser dividido em duas partes:

- resposta transitória;
- resposta de regime;

Em estudos de sistemas de controle, define-se regime permanente como a resposta obtida quando o tempo tende para o infinito.

Frequentemente, as características de desempenho em um sistema de controle são especificadas em termos da resposta transitória para uma entrada do tipo degrau unitário, pois esta entrada é fácil de se gerar e é suficientemente severa, em outras palavras, se a resposta a uma entrada degrau é conhecida, é matematicamente possível encontrar a resposta para qualquer outra entrada que o sistema possa ser submetido na prática [01].

A resposta transitória de um sistema para uma entrada degrau unitário depende das condições iniciais. Para conveniência na comparação de respostas transitórias de vários sistemas, costuma-se usar a condição inicial padrão, de que o sistema está inicialmente em repouso com a saída e todas as suas derivadas nulas. Desta forma as características de resposta transitória podem ser facilmente comparadas.

Na prática, a parte transitória da resposta pode apresentar oscilações amortecidas antes de alcançar o estado estacionário (regime permanente).

A seguir, na Fig.24 estão apresentadas as variáveis que normalmente estão sendo analisadas em problemas de controle, de onde se percebe que a parte transitória da resposta é justamente, a mais complicada para se controlar, pelo fato de que muitas variáveis estão envolvidas.

As características de desempenho mostradas na Fig. 24, são função dos parâmetros ξ (coeficiente de amortecimento) e ω_n (frequência natural) da planta. O coeficiente de amortecimento influencia predominantemente no valor de sobre-sinal máximo da resposta ao degrau da planta, e a frequência natural influencia com maior intensidade no tempo de subida da resposta. Entretanto, todas as especificações de resposta transitória sofrem em conjunto com as variações destes parâmetros.

É importante ressaltar que as especificações de sobre-sinal máximo e tempo de subida são conflitantes, ou seja, se temos um baixo tempo de subida, teremos um sobre-sinal máximo relativamente grande, e se obtemos um sinal de resposta com um mínimo de sobre-sinal máximo

acarretará um tempo de subida alto, considerando a frequência natural (ω_n) do sistema constante. Ficando portanto, a cargo do projetista de controle a tarefa de como proceder para alcançar os objetivos desejados pelo usuário, da maneira mais satisfatória possível.

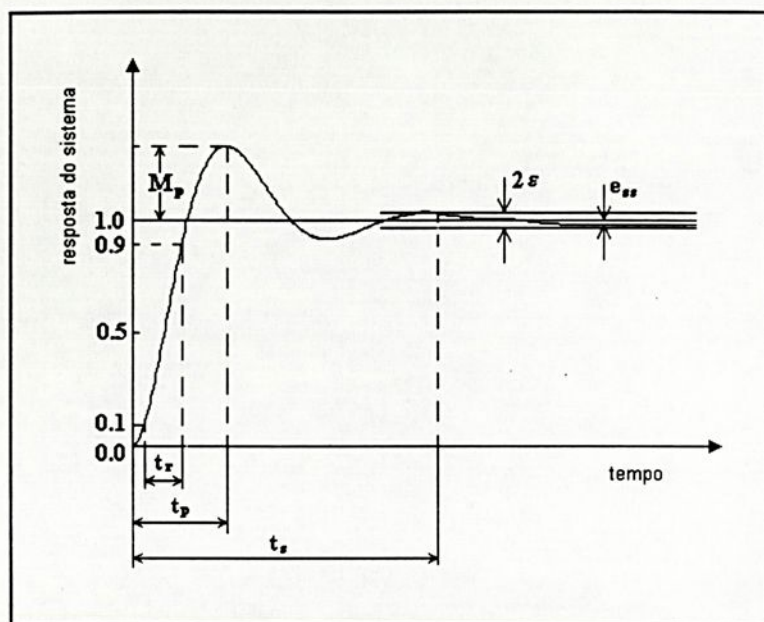


Fig. 24 - Curva de resposta a degrau unitário mostrando M_p , t_r , t_p , t_s , e_{ss} .

- M_p é o sobre-sinal máximo;
- t_r é o tempo de subida;
- t_p é o instante de pico;
- t_s é o tempo de acomodação;
- e_{ss} é o erro de regime;
- ϵ é a tolerância permitida.

V.3 - RNA's EM CONTROLE

Algumas características de RNA's se apresentam particularmente importante para aplicações em problemas de controle, na fase de identificação, de modelagem ou no controle propriamente dito.

Não linearidade

O fato de que RNA's teoricamente efetua mapeamentos não-lineares, faz delas uma potente ferramenta para resolver problemas de controle no domínio não-linear, área esta, que se apresenta com uma certa dificuldade para ser tratada utilizando métodos convencionais do controle moderno.

Paralelismo e alta conectividade

As estruturas das RNA's são normalmente formadas através da conexão entre seus elementos básicos, os neurônios, de modo paralelo e com alto grau de conectividade. Isto sugere considerar que RNA's seja uma implementação paralela por si só. Com esta estrutura, é de se esperar que as tolerâncias envolvidas, sejam menores do que os esquemas convencionais [07].

Capacidade de aprender

RNA's possuem a capacidade de armazenar informações e utilizar tais informações para generalizar respostas a partir do treinamento feito a partir de um determinado conjunto de padrões, ou aprender de forma "on line", sendo excelentes ferramentas para o controle adaptativo.

Multivariáveis

RNA's possuem sempre a capacidade de processar muitas entradas, obtendo muitas saídas, de maneira simultânea, sendo indicadas portanto para sistemas multivariáveis.

Fusão de dados

As RNA's podem operar com dados de maneira quantitativa (engenharia tradicional $\Rightarrow$ numérico), ou qualitativa (inteligência artificial $\Rightarrow$ símbolo), simultaneamente.

Implementação

Já existem alguns trabalhos no sentido de implementação a nível de "hardware", o que aumentaria muito a velocidade de processamento dos sistemas envolvidos [44] a [47].

Controle Inverso

Para os "controlistas", a habilidade que RNA's do tipo "feedforward" possui para trabalhar com sistemas não-lineares, é o fator mais importante para usá-las como ferramentas nesta área. Podendo inclusive ser traçado um paralelo, afirmando que da mesma maneira que as FT's oferecem representações genéricas para modelos lineares de "caixas-pretas", RNA's potencialmente oferecem representações genéricas para modelos não-lineares de "caixas-pretas" [07].

Esta capacidade que RNA's multicamadas possui para realizar mapeamentos não-lineares, agora associada, com a capacidade de representação dinâmica, possibilitou a esta estrutura ser considerada uma gigantesca ferramenta na área de identificação e controle de sistemas dinâmicos



não-lineares desconhecidos matematicamente [07].

A técnica de controle escolhida para ser utilizada em nosso problema, foi o Controle Inverso. A idéia embutida é muito simples, embora no controle convencional esta técnica não seja muito utilizada ou muito divulgada, pode ganhar mais espaço quando utilizada em conjunto com RNA's.

A aplicação da técnica do Controle Inverso com RNA's já foi discutida em alguns artigos [08], [43] e [44], porém, em nenhum destes trabalhos encontramos exemplos envolvendo plantas de natureza não-linear e de segunda ordem, conforme apresentados a seguir, no capítulo VI.

A idéia básica está sintetizada nas Fig's. 25 e 26, onde são apresentados os diagramas de blocos para as duas fases do Controle Inverso, onde $s(k)$ é o vetor de estado da planta no instante "k".

Inicialmente tem-se a fase de projeto, ou de treinamento. É nesta fase que a RNA identifica a dinâmica inversa da planta, esquema apresentado na Fig. 25.

Foi inserido um ganho "2", com objetivo de amplificar $u(k)$, que foi restrito ao intervalo $[-1, 1]$.

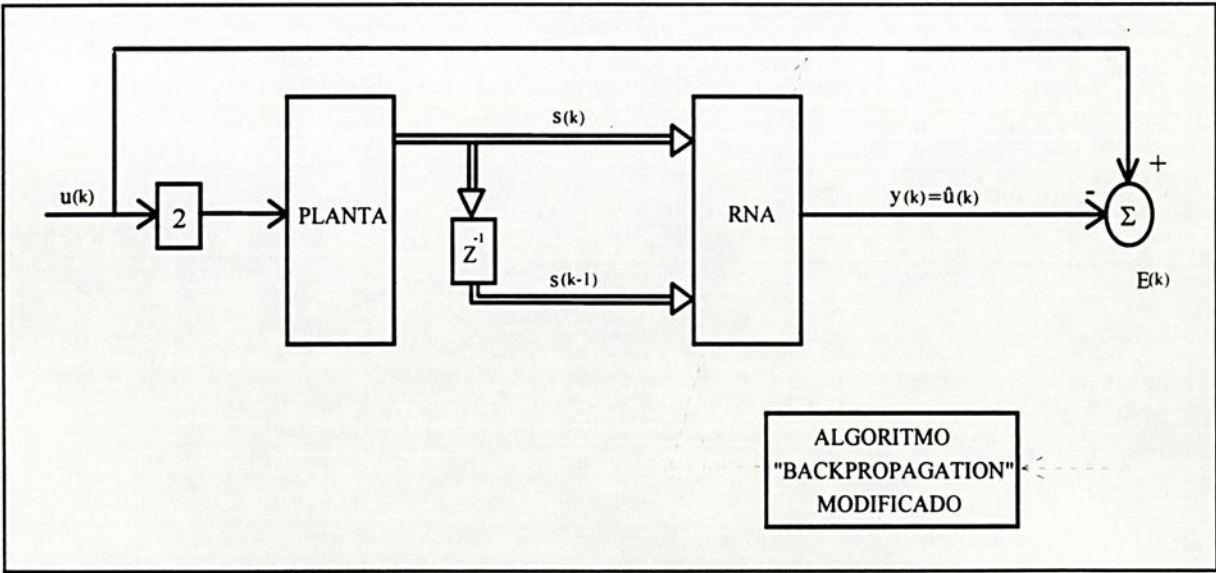


Fig. 25- Diagrama de blocos do esquema de treinamento da RNA para o Controle Inverso.

A seguir tem-se a fase de aplicação, ou de controle. Nesta fase a RNA treinada é

utilizada para executar o controle da planta, através de um modelo de referência, conforme esquema apresentado na Fig. 26.

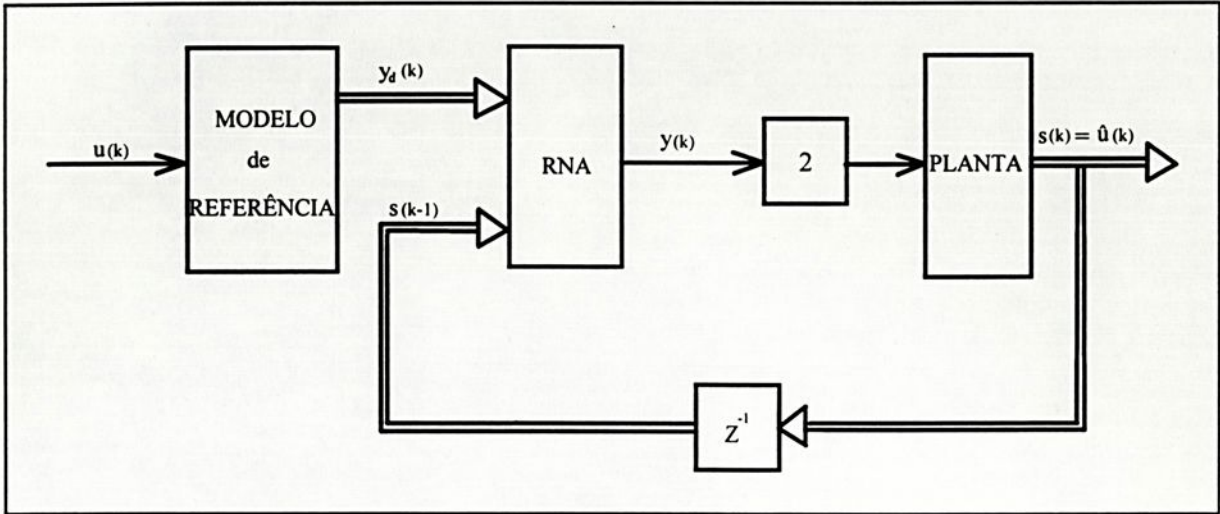


Fig. 26 - Diagrama de blocos do esquema de Controle Inverso da planta com modelo de referência.

Na fase de treinamento ou de projeto, foi considerada a hipótese que garante a existência da função inversa da planta.

Na fase de projeto do controlador, são apresentadas à planta um determinado número de entradas " $u(k)$ ". Esta fornecerá então as respectivas saídas (estados), que são apresentadas à RNA como entradas; a RNA por sua vez, fornecerá saídas " $\hat{u}(k)$ " que deverão aproximar-se dos valores das entradas fornecidas à planta, através do algoritmo de treinamento da RNA.

Considerando-se que ao final do treinamento se consiga $u(k) = \hat{u}(k)$, esquematicamente tem-se:

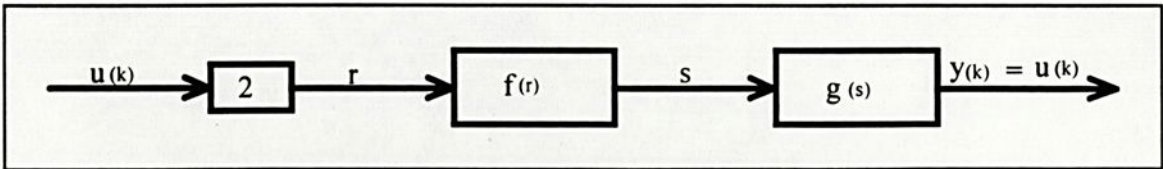


Fig. 27 - Diagrama de blocos das funções matemática envolvidas no treinamento da RNA.

onde:

$u[k]$ é o sinal de entradas externas,

2 é um ganho colocado de forma a duplicar o sinal de entrada,

$r = 2u[k]$ é a entrada efetiva da planta,

$f[.]$ é uma função inversível desconhecida, e representa a planta dinâmica,

$s = f[2u[k]]$ é a saída da planta,

$g[.]$ representa a função de mapeamento da RNA.

Matematicamente tem-se:

$$u(k) \xrightarrow{h} h(u(k)) \xrightarrow{f} f(h(u(k))) \xrightarrow{g} g(f(h(u(k)))) = u(k),$$

ou ainda,

$$g \circ (f \circ h)(u) = u.$$

(12)

Pode ser mostrado analiticamente que a função g é a inversa de $(f \circ h)$, como segue. Sabe-se que $(f \circ h)$ possui inversa, pois é composta de funções inversíveis.

Seja $p = (f \circ h)^{-1}$ esta função inversa, então tem-se que

$$(f \circ h) \circ p(x) = x,$$

(13)

$$p \circ (f \circ h)(x) = x.$$

(14)

Supondo que a função g não seja a inversa de $(f \circ h)$, ou seja, g é diferente de p , então deve existir um ponto x_0 tal que $g(x_0) \neq p(x_0)$.

Fazendo $p(x_0) = y_0$ pode-se escrever,

$$(f \circ h)(y_0) = (f \circ h)(p(x_0)) = ((f \circ h) \circ p)(x_0) = x_0.$$

(15)

Desta maneira,

$$g(x_0) = g((f \circ h)(y_0)) = (g \circ (f \circ h))(y_0) = y_0.$$

(16)

Este resultado é uma contradição com o fato de que $p(x_0) \neq g(x_0)$, logo, $g = p$.

Portanto quando a RNA for treinada, e segundo o diagrama da Fig. 25, o erro estiver próximo de 0 (zero), pode-se dizer que seus parâmetros estão ajustados de forma que ela possa fornecer um mapeamento semelhante a inversa da planta, tão próximo quanto menor for o erro desejado.

Observa-se que o treinamento é feito de forma "off-line", permitindo uma certa liberdade quanto ao tempo gasto para tal. Depois de treinada a RNA será então conectada no sistema, conforme Fig. 26, e nesta fase sim é importante que o tempo de resposta da RNA seja compatível com as constantes de tempo do sistema.

V.4 - CONCLUSÕES

As características de desempenho desejadas para a planta, serão definidas inicialmente para o modelo de referência, e o controlador (RNA), fará com que a planta obedeça aquelas exigências.

O controle através da dinâmica inversa, pode ser considerado como sendo de uma simplicidade muito grande, tanto na parte matemática como na idéia conceitual em si.

Um problema, é que esta técnica, exige sempre a hipótese de que o sistema analisado admite inversa, isto pode complicar já que o objetivo é trabalhar com sistemas cujos modelos matemáticos são desconhecidos. Outro problema é conseguir o treinamento de plantas que se desestabilizam rapidamente, pois isto dificulta muito o treinamento da RNA para que ela possa identificar a inversa.



EXEMPLOS DE CONTROLE UTILIZANDO RNA's

VI.1 - INTRODUÇÃO

Este capítulo apresenta aplicações de RNA em controle, utilizando a técnica da dinâmica inversa, sob várias condições.

A RNA treinada para fornecer a dinâmica inversa da planta, acoplada à planta, deverá seguir as trajetórias determinadas por um modelo de referência.

Será utilizado o modelo matemático de um pêndulo forçado.

VI.2 - MODELAGEM DO SISTEMA DINÂMICO

Apesar de que o objetivo principal deste exemplo seria mostrar a aplicação de técnicas de RNA's em controle de sistemas dinâmicos desconhecidos, escolhemos um sistema conhecido, para gerar os pares de padrões entradas/saídas. Na prática estes valores seriam medidos fisicamente do sistema estudado, sem conhecer a função matemática envolvida.

O sistema dinâmico (planta) utilizado que pode ser visto esquematicamente na Fig. 28, é denominado pêndulo forçado.

É um sistema estável, porém com respostas de posição e velocidade muito lentas, e que supostamente não é satisfatório para uma determinada aplicação, sendo necessário então a



inclusão de um controlador que permita ao sistema dinâmico fornecer um comportamento desejado, segundo um modelo de referência.

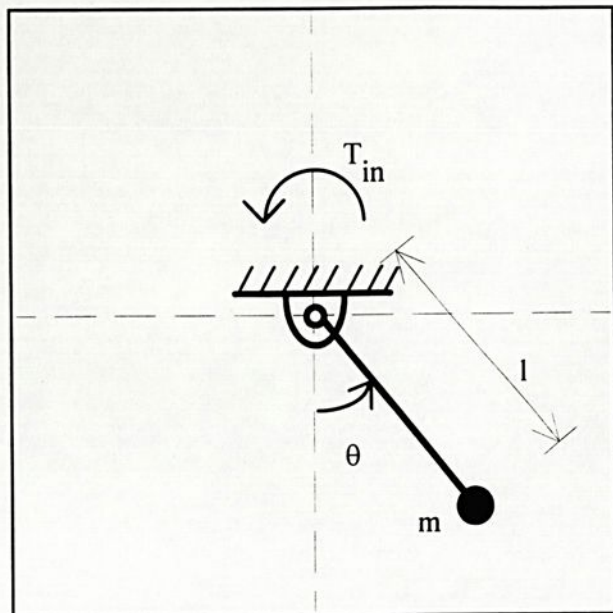


Fig. 28 - Esquema do modelo da planta utilizada para os exemplos.

O modelo matemático para este sistema é completamente definido pelas equações 17 e 18, apresentadas a seguir,

$$\dot{\omega} = -\frac{\xi}{ml^2}\omega - \Omega^2 \sin\theta + T_{in} \frac{1}{ml^2} , \quad (17)$$

$$\dot{\theta} = \omega . \quad (18)$$

onde,

θ é a variável de posição,

ω é a variável de velocidade ($\omega = d\theta/dt$),

T_{in} é o torque externo (especificado pelo projetista),

ξ é o coeficiente de amortecimento do sistema ($\xi = 0.25$),

m é a massa do sistema, considerada concentrada na esfera do pêndulo ($m = 0.5$),

l é o comprimento do pêndulo ($l = 0.5$),

Ω é a frequência de oscilação ($(g/l)^{1/2}$).

g é a aceleração gravitacional ($g = 10.0$).

Observação: todas as unidades estão no sistema MKS.

Para as simulações o problema foi dividido em duas etapas, sendo que na primeira o modelo matemático foi linearizado, de modo que pudesse ser facilmente discretizado através do programa "MATLAB", que utiliza a transformada-z, um conceito definido somente para equações dinâmicas lineares. Isto facilitaria as primeiras análises dos resultados das simulações computacionais, proporcionando uma maior familiarização com este tipo de problema. Em uma segunda fase passou-se a trabalhar com o modelo matemático original, ou seja, não-linear, sendo necessário portanto, fazer uso do método de EULER, para as simulações no computador digital.

Com isto foi possível analisar como se comporta a RNA envolvida quando se trata de plantas lineares e não-lineares.

VL3 - SISTEMA DINÂMICO LINEARIZADO

Inicialmente definimos o seguinte vetor de estado,

$$X = \begin{bmatrix} \omega \\ \theta \end{bmatrix},$$

(19)



em seguida efetuou-se a linearização do modelo matemático em torno do ponto $(\omega_0, \theta_0) = (0, 0)$, que representa a situação de origem com o pêndulo em repouso na posição de descanso, obtendo assim a seguinte equação de estado linearizada,

$$\dot{X} = \begin{bmatrix} -\frac{\xi}{ml^2} & -\Omega^2 \cos \theta_0 \\ 1 & 0 \end{bmatrix} X + \begin{bmatrix} \frac{1}{ml^2} \\ 0 \end{bmatrix} T_{in} , \quad (20)$$

e a saída é dada por,

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} X . \quad (21)$$

Utilizando um período de amostragem $T_A = 0.01$ segundos, efetuamos a discretização do modelo linearizado, obtendo equação discreta, ou de diferença, mostrada a seguir, com um conversor D/A usando um bloqueador de ordem zero na entrada da planta e um conversor A/D na saída da planta,

$$X[k+1] = \begin{bmatrix} 0.9792 & -0.1979 \\ 0.0099 & 0.9990 \end{bmatrix} X[k] + \begin{bmatrix} 0.0792 \\ 0.0004 \end{bmatrix} T_{in}[k] . \quad (22)$$

Nesta primeira etapa, com o modelo linearizado, por coerência, todas as análises foram feitas para ângulos pequenos, onde a não linearidade da planta não possui tanta influência.

Trabalhou-se então com a variável de posição, ângulo, assumindo valores de no

máximo de 20° (vinte graus), em regime.

A planta linearizada, quando excitada por entradas degrau, apresenta o comportamento conforme pode ser observado pelas Fig's. 29 e 30, apresentadas a seguir.

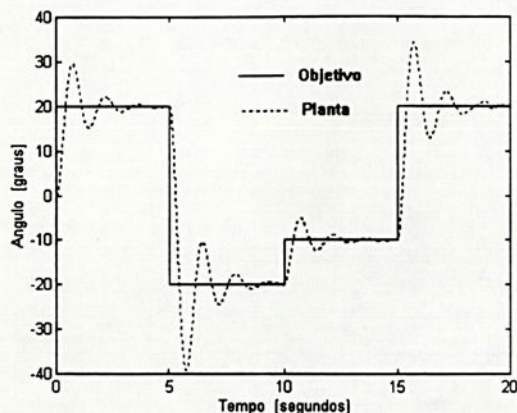


Fig. 29 - Comportamento da resposta de posição da planta linearizada.

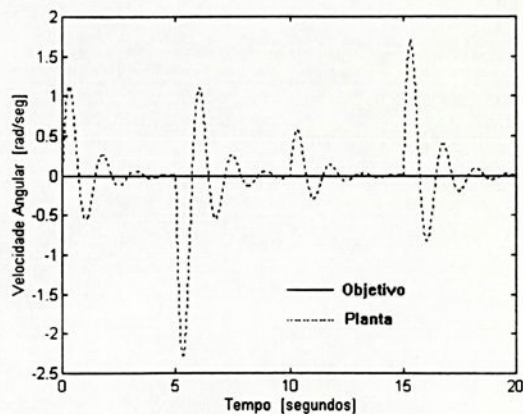


Fig. 30 - Comportamento da resposta de velocidade da planta linearizada.

VL3.1 - SIMULAÇÕES COM A PLANTA LINEARIZADA

As simulações foram feitas utilizando modelos de referências com diferentes comportamento, variando para isto o valor do parâmetro que define o coeficiente de amortecimento " ξ " do referido modelo. Em todas as simulações, as condições de controle são bastantes severas, alterando as entradas degrau de forma a dificultar cada vez mais o controle exercido pelo controlador neuronal.

O treinamento foi feito durante aproximadamente 20 (vinte) minutos, utilizando 500 pares padrões gerados pela aplicação de torques aleatórios, partindo de condições iniciais nulas, com 500 iterações, em um microcomputador PC-486-DX2.

O erro quadrático médio final ficou em torno de 0.0013.

A RNA utilizada possui 41 (quarenta e um neurônios) distribuidos como segue:

- 20 (vinte) na primeira camada,
- 20 (vinte) na segunda camada,
- 1 (um) na camada de saída.

Algoritmo utilizado para o treinamento foi o "backpropagation" com o erro filtrado, com taxa de aprendizado $\mu = 0.05$ e coeficiente para ajuste do parâmetro do filtro $\gamma = 0.003$.

VL3.1.1 EXEMPLO-4 $\Rightarrow$ Controle da planta linearizada, com modelo de referência utilizando coeficiente de amortecimento $\xi = 0.5$.

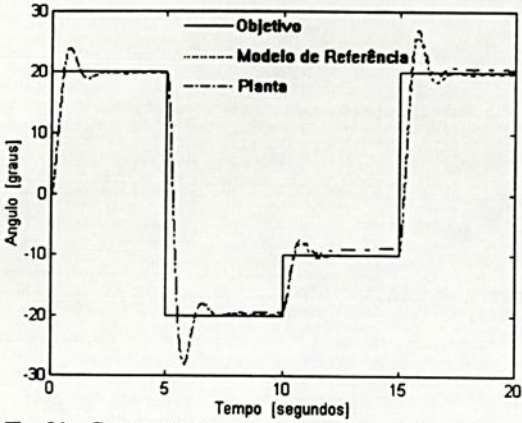


Fig. 31 - Comportamento da resposta de posição da planta depois de controlada.

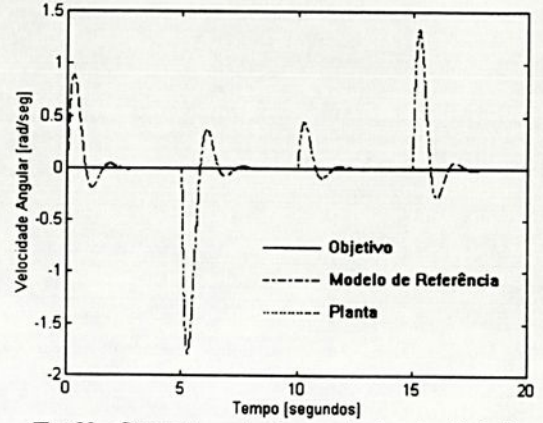


Fig. 32 - Comportamento da resposta de velocidade da planta depois de controlada.

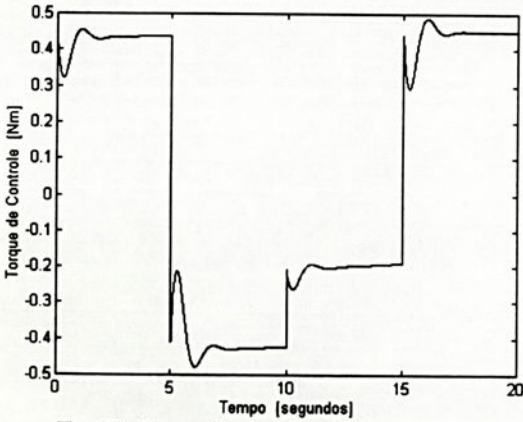


Fig. 33 - Torque de controle gerado pela RNA.

O resultado foi bastante satisfatório para este caso, podemos notar que a planta segue a trajetória do modelo de referência perfeitamente.

VL3.1.2 - EXEMPLO-5 ⇒ Controle da planta com modelo de referência utilizando coeficiente de amortecimento $\xi=0.8$.

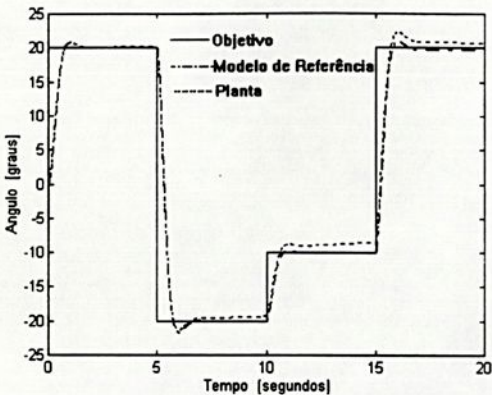


Fig. 34 - Comportamento da resposta de posição da planta depois de controlada.

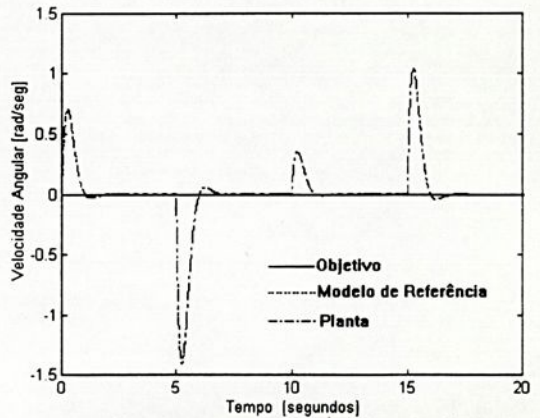


Fig. 35 - Comportamento da resposta de velocidade da planta depois de controlada.

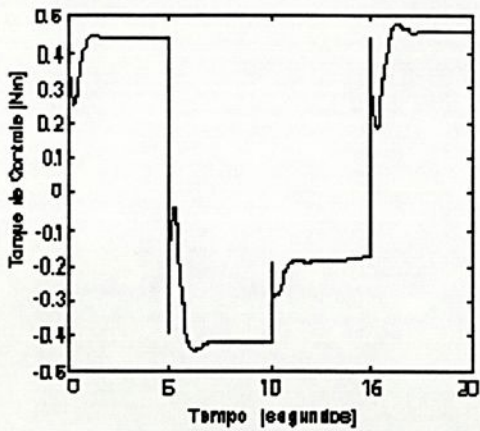


Fig. 36 - Torque de controle gerado pela RNA.

Também neste caso as respostas se mostraram bastante satisfatórias, agora com um modelo mais rígido, diminuindo o sobre-sinal máximo.

VL3.1.3 - EXEMPLO-6 $\Rightarrow$ Controle da planta com modelo de referência utilizando coeficiente de amortecimento $\xi=1.0$.

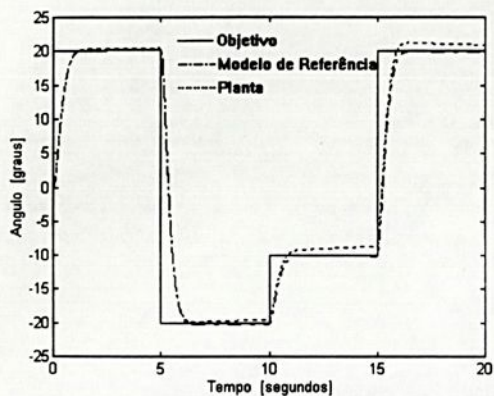


Fig. 37 - Comportamento da resposta de posição da planta depois de controlada.

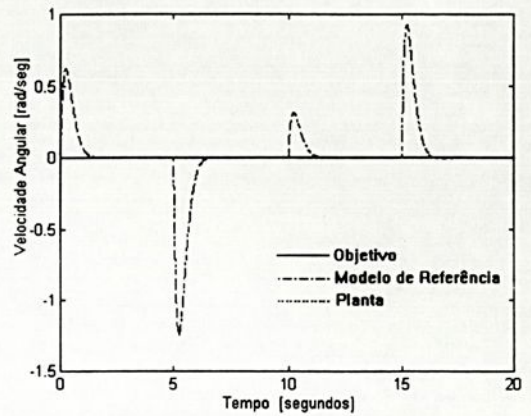


Fig. 38 - Comportamento da resposta de velocidade da planta depois de controlada.

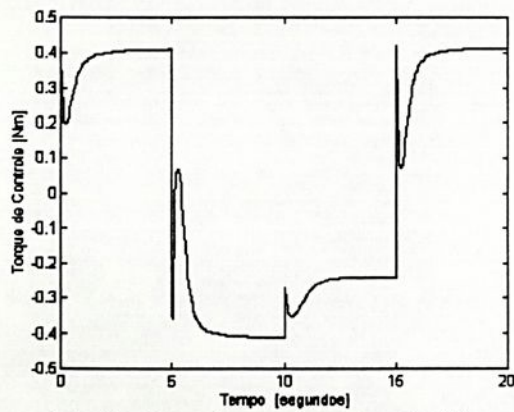


Fig. 37 - Torque de controle gerado pela RNA.

Este caso é bastante interessante, pois transformou o comportamento da resposta da planta que era bastante lento (tempo de acomodação grande), em uma resposta rápida.

VL3.2 - CONCLUSÕES

Observamos que o Controle Inverso aplicado através de RNA's, resultou em grande sucesso, para os casos onde os ângulos sofrem uma variação relativamente pequena, permanecendo dentro de uma faixa que permita a linearização do sistema.

Em todas as simulações observou-se um sucesso absoluto no transitório, e um pequeno erro de regime.

Observou-se também que a variável de velocidade é mais facilmente controlada do que a variável de posição, que apresenta um erro de regime.

O erro de regime encontrado nos gráficos de variável de posição é constante e da ordem de 4% (quatro por cento). Não aparece nos gráficos da variável velocidade porque esta é a derivada da variável de posição, sendo o erro constante, sua participação na derivada é nula.

VL4 - SISTEMA DINÂMICO NÃO-LINEARIZADO

Para esta segunda parte considerou-se o sistema original, ou seja não-linear e de segunda ordem.

Nesta parte é que se encontra o resultado mais interessante, pois se consegue trabalhar de uma maneira muito simples, mesmo se tratando de um sistema não-linear, alcançando os mesmos resultados obtidos para sistemas linearizados.

Neste caso, como o sistema é não-linear, não é possível utilizar o método da transformada-z, para obter as equações discretizadas para as simulações. Utilizou-se então o método de discretização de EULER, que apesar de ser aproximado apresenta resultados satisfatórios para as aplicações feitas aqui.

Neste caso foi utilizado um período de amostragem $T_E=0.001$ segundos para a discretização de EULER e $T_A=0.01$ segundos para o aprendizado da RNA.



Fazendo $t = k(T_E)$, para $k = 0, 1, 2, \dots$, podemos escrever:

$$\dot{\omega}(k(T_E)) \cong \frac{\omega((k+1)(T_E)) - \omega(k(T_E))}{T_E} . \quad (23)$$

Substituindo (23) em (17), temos:

$$\omega((k+1)(T_E)) = T_E \left(-\xi \omega(k(T_E)) - \Omega^2 \sin \theta((k(T_E))) + \frac{T_{in}(k(T_E))}{ml^2} \right) + \omega(k(T_E)) . \quad (24)$$

Analogamente para θ , temos

$$\dot{\theta}(k(T_E)) = \frac{\theta((k+1)(T_E)) - \theta(k(T_E))}{T_E} , \quad (25)$$

por (18),

$$\omega(k(T_E)) = \frac{\theta((k+1)(T_E)) - \theta(k(T_E))}{T_E} , \quad (26)$$

ou seja,

$$\theta((k+1)(T_E)) = T_E (\omega(K(T_E))) + \theta(k(T_E)) . \quad (27)$$



Substituindo agora, os respectivos valores dos parâmetros do modelo estudado, em (24) e (27),

$$\omega((k+1)(T_E)) = 0.001 \left(-0.25 \omega(k(T_E)) - 20.0 \sin \theta(k(T_E)) + 8.0 T_{in}(k(T_E)) \right) + \omega(k(T_E)) , \quad (28)$$

$$\theta((k+1)(T_E)) = 0.001 (\omega(k(T_E))) + \theta(k(T_E)) . \quad (29)$$

O comportamento da resposta do sistema, antes de ser colocado o controlador, está mostrada nas Fig's. de 40 a 43.

Sendo que as Fig's. 40 e 41 são para ângulos máximos de regime desejados de 50°

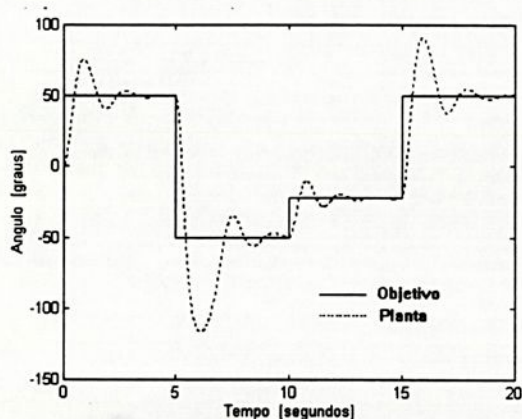


Fig. 40 - Resposta de posição da planta não-linearizada para ângulos pequenos.

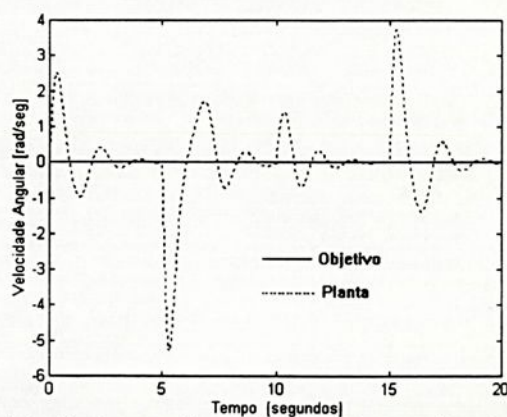


Fig. 41 - Resposta de velocidade da planta não-linearizada para ângulos pequenos.

Já nas Fig's. 42 e 43 são apresentados o comportamento da planta para ângulos máximos de regime desejados de 80° (oitenta graus).

Observa-se que para o caso onde se pretende levar o pêndulo para a posição de 80° (oitenta graus), o sistema torna-se instável, em função do valor do sobre-sinal máximo elevado, cerca de 45% (quarenta e cinco por cento).

Este fato faz com que o valor do ângulo ultrapasse os 90° (noventa graus), e quando isto ocorre o torque constante aplicado para levar o pêndulo até a posição desejada, passa a ser maior do que o torque causado pelo peso e atrito do pêndulo que é proporcional ao seno do ângulo.

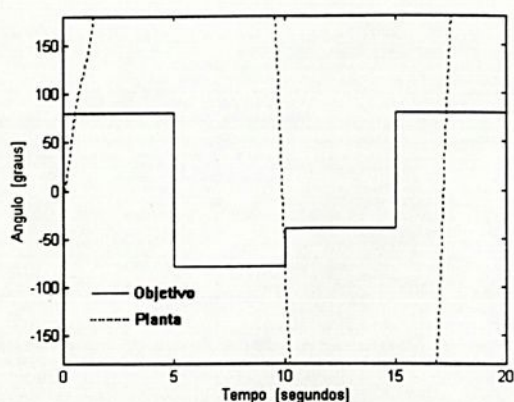


Fig. 42 - Resposta de velocidade da planta não-linearizada para ângulos grandes.

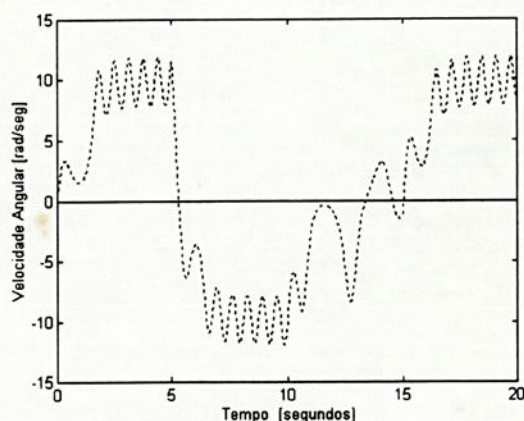


Fig. 43- Resposta de velocidade da planta não-linearizada para ângulos grandes.

VL4.1 - SIMULAÇÕES COM A PLANTA NÃO-LINEARIZADA

As simulações foram feitas utilizando modelos de referências com diferentes comportamento, variando para isto o valor do parâmetro que define o coeficiente de amortecimento " ξ " do referido modelo.

Em todas as simulações, as condições de controle são bastantes severas, alterando as entradas degrau de forma a dificultar cada vez mais o controle exercido pelo controlador neuronal.

O treinamento foi feito durante aproximadamente 20 (vinte) minutos, utilizando 500 pares padrões, gerados aleatoriamente, com 500 iterações. O erro quadrático médio final ficou em torno de 0.0014, nas mesmas condições do caso linear.

A RNA utilizada possui 41 (quarenta e um neurônios) distribuídos como segue:

- 20 (vinte) na primeira camada,
- 20 (vinte) na segunda camada,
- 1 (um) na camada de saída.

Algoritmo utilizado para o treinamento foi o "backpropagation" com o erro filtrado, com taxa de aprendizado $\mu = 0.05$ e coeficiente para variação do parâmetro do filtro $\gamma = 0.003$.

VL4.1.1 - EXEMPLO 7 $\Rightarrow$

Controle da planta com modelo de referência utilizando coeficiente de amortecimento $\xi=0.5$.

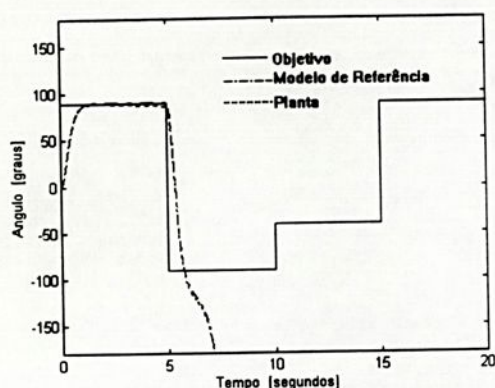


Fig. 44 - Resposta de posição da planta depois de controlada.

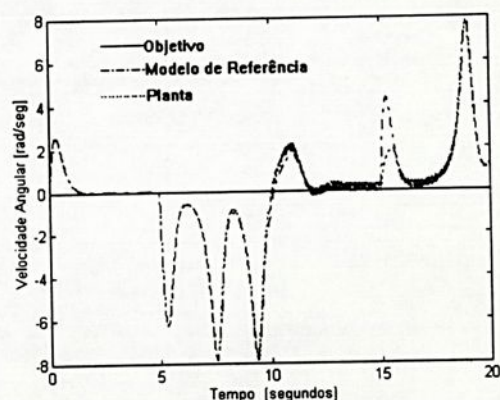


Fig. 45 - Resposta de velocidade da planta depois de controlada.

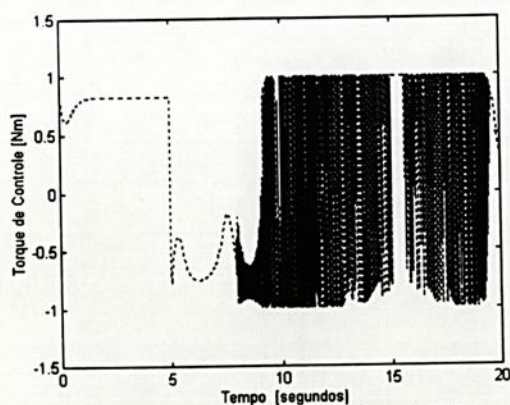


Fig. 46 - Torque de controle gerado pela RNA.

A planta seguiu perfeitamente a trajetória do modelo de referência. Pelo gráfico de posição não se consegue saber o que ocorreu depois, mas pelo gráfico das velocidades percebemos que a planta seguiu o modelo. Porém nem mesmo o modelo escolhido conseguiu estabilizar-se em algum ponto desejado, devido as condições severas impostas pelas variações bruscas nas posições desejadas.

VL4.1.2 - EXEMPLO 8 $\Rightarrow$

Controle da planta com modelo de referência utilizando coeficiente de amortecimento $\xi=0.8$.

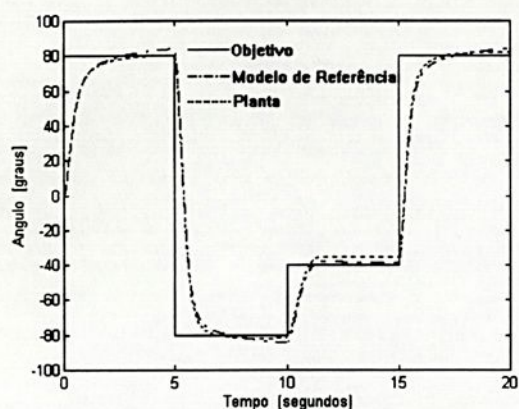


Fig. 47 - Resposta de posição da planta depois de controlada.

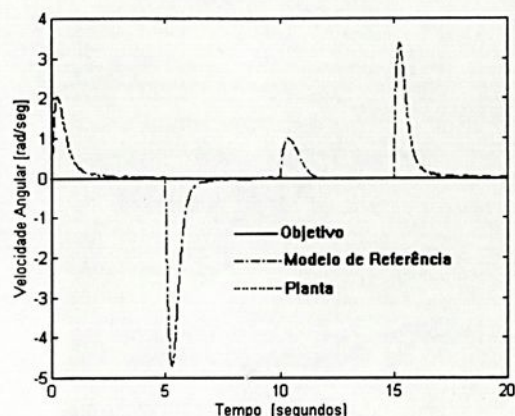


Fig. 48 - Resposta de velocidade da planta depois de controlada.

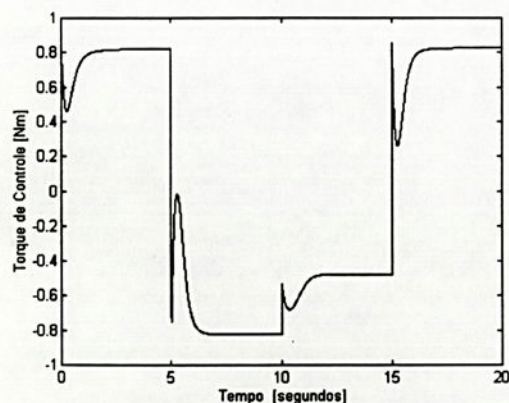


Fig. 49 - Torque de controle gerado pela RNA.

Neste exemplo percebemos que o comportamento da planta foi bastante satisfatório na região de trabalho solicitada.

Notamos que sem o controlador a planta teria um comportamento instável nesta região, como foi visto nas Figs 42 e 43.

VL4.1.3 - EXEMPLO 9 $\Rightarrow$ Controle da planta com modelo de referência utilizando coeficiente de amortecimento $\xi=1.0$.

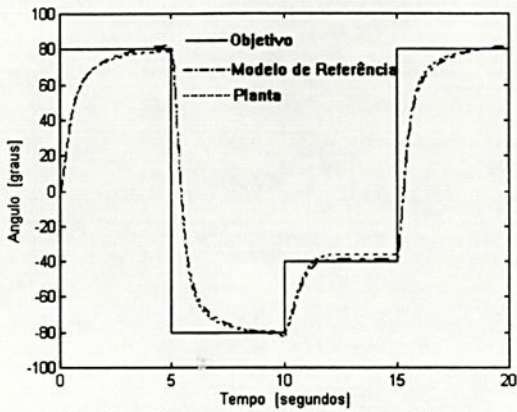


Fig. 50 - Resposta de posição da planta depois de controlada.

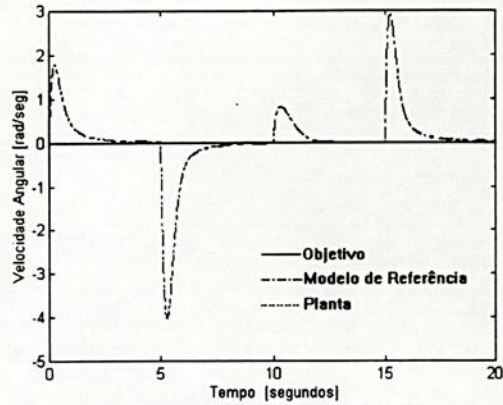


Fig. 51 - Resposta de velocidade da planta depois de controlada.

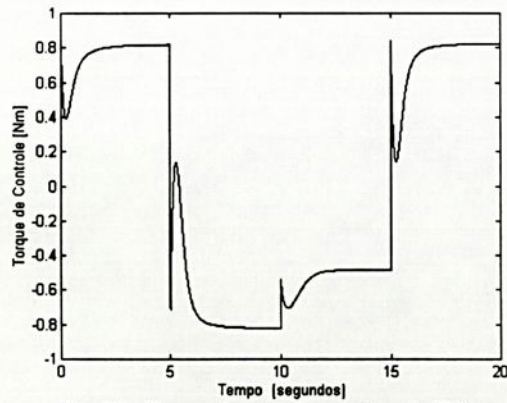


Fig. 52 - Torque de controle gerado pela RNA.

Este é também um resultado muito importante. A planta que apresentava um sobre-sinal máximo relativamente grande, foi para a posição desejada de maneira direta, com sobre-sinal nulo.

VL4.2 - CONCLUSÕES

Neste caso a planta teve que trabalhar com valores de ângulos relativamente grandes, caindo em regiões onde sabemos que não é possível considerar um modelo linear para representar a planta.

Os resultados apresentados foram excelentes, pois apesar de considerarmos o sistema desconhecido, sabemos que se trata de um sistema de segunda ordem e não-linear.

É importante ressaltar que na técnica do controle inverso, a RNA identifica diretamente o comportamento da dinâmica inversa, não tendo a necessidade de identificar a priori a dinâmica da planta.

Nestes exemplos também foi necessário considerar a hipótese de que a planta admite inversa.



CONCLUSÕES GERAIS

O trabalho apresenta inicialmente algumas noções gerais sobre Redes Neurais Artificiais, importantes para alguém que queira iniciar estudos nesta área. Em seguida é apresentada uma alteração no Algoritmo "Backpropagation", com o objetivo de melhorá-lo. Este algoritmo em sua forma "standard" é um método básico para o treinamento de RNA's, e considerado por muitos pesquisadores, o maior responsável pelo ressurgimento do assunto, despertando interesse em tantos cientistas, no mundo todo, das mais diversas áreas e de uma maneira simultânea.

Alguns resultados de simulações, de problemas envolvendo identificação de funções não-lineares são apresentados. Em todos os problemas estudados são apresentados resultados obtidos através do BA "standard", do BA utilizando o termo de momento e do método proposto.

Através da comparação dos resultados obtidos para cada método analisado, podemos notar a potencialidade do método proposto.

O trabalho apresenta também algumas noções importantes sobre Controle Automático de Sistemas Dinâmicos utilizado RNA's, através da técnica do Controle Inverso com modelo de referência.

São apresentados, alguns resultados de simulações, onde a RNA atua como o controlador de uma planta de segunda ordem, não-linear, que supostamente não possui modelo matemático conhecido.

Os resultados obtidos foram muito positivos, mostrando que o assunto é bastante



promissor, e com aplicabilidade prática.

Como sugestões para possíveis trabalhos a serem desenvolvidos a partir deste, temos:

- # A automatização do coeficiente (γ), utilizado no ajuste do parâmetro do filtro (b_p), é um ponto importante para o sucesso do método, pois, uma vez isto feito, o projetista somente teria que especificar a taxa de aprendizado a ser utilizado em cada treinamento, o que facilitaria muito o uso do método;
- # A comparação do método proposto, com outros métodos encontrados na literatura com o mesmo objetivo, de melhorar o comportamento do BA, analisando vantagens e desvantagens de cada um, também é um trabalho bastante interessante a ser desenvolvido;
- # Aplicações de RNA's junto com a técnica do controle inverso, em problemas de plantas instáveis, como por exemplo o pêndulo invertido. Nestes problemas existe a dificuldade para a identificação da dinâmica inversa da planta, justamente porque em apenas alguns décimos de segundo a planta se desestabiliza dificultando muito o treinamento da RNA;
- # Comparação dos resultados obtidos com a aplicação de RNA's e técnicas do controle moderno, principalmente para sistemas não-lineares, e instáveis.



REFERÊNCIAS BIBLIOGRÁFICAS

- [01] K. OGATA, "Modern Control Engineering", Prentice-Hall International Editions, Second Edition, 1990;
- [02] D. E. RUMELHART, G. E. HINTON and R. J. WLLIANS, "Learning Internal Representations by Error Propagation". In Parallel distributed processing, vol.1, Combridge, MA: Mit Press, 1986.
- [03] R. P. LIPPMANN, "An Introduction to Computing With Neural Nets", IEEE ASSP Magazine, Apr. 1987, vol. 3, N^o 4, p. 4-22;
- [04] B. WIDROW and M. LEHR, "30 Years of Adaptive Neural Networks: Perceptron, Madaline and Backpropagation", proceedings of the IEEE, vol. 78, N^o 9, setember, 1990, p. 1415-1452;
- [05] G. A. CARPENTER, "Neural Networks Models for Pattern Recognition and Associative Memory", Neural Networks, vol. 2, 1989, p. 243-257;
- [06] P. D. WASSERMAN, "Neural Computing: Theory and Prattice", Library of Congress Cataloging-in-Publication Data, 1989.



- [07] K. J. HUNT, D. SBARBARO, R. ZBIKOWSKI and P. J. GAWTHROP, "Neural Networks for Control Systems - A Survey", *Automatica*, Vol. 28, No. 6, 1992.
- [08] T. FUKUDA and T. SHIBATA, "Theory and Applications of Industrial Control Systems", *IEEE Trans. on Ind. Electronics*, vol. 39, N° 6, December, 1992.
- [09] R. ROSEMBLANTT, "Principles of Neuodynamics", N. York, Spartan Books, 1959;
- [10] WIDROW, B. and M. E. HOFF, "Adaptative Switching Circuits"; 1960 IRE WESCON Conv. Record, part 4, p. 96-104, August 1960;
- [11] W. S. McCULLOCH and W. H. PITTS "A Logical Calculus of the Ideas Immanent in Nervous activity", *Bull. Math. Biophysics*, vol. 5, p. 115-123, 1943.
- [12] P. J. WERBOS, "Backpropagation Through Time: What It Does and How to Do It", *Proceedings of the IEEE*, vol. 78, N° 10, October 1990, p. 1550-1560;
- [13] H. D. BALLARD, "Cortical Connections and Parallel Processing: Structure and Function", In M. Arbib and Homson (Eds); *Vision, Brain and Cooperative Computation*, 1988, p. 563-621. MIT Press, Cambridge, Ma;
- [14] M. L. MINSKY and S. PAPERT, "Perceptrons", Cambridge, MA: MIT Press, 1969;
- [15] P. J. WERBOS, "Beyond Regression: Newtools for Prediction and Analysis in the Behavioral Sciences", Masters Thesis, Harvard University, 1974;
- [16] F. C. CHEN, "Backpropagation Neural Networks for Nonlinear Self-Tuning Adaptive Control", *IEEE Control Systems Magazine*, April, 1990, p. 44-48;
- [17] B. WIDROW, "The Speed of Adaptation in Adaptive Control Systems", American Rocket Society Guidance Control and Navigation Conference, 1961;



- [18] B. WIDROW, "A Statistical Theory of Adaptation. Adaptive Control Systems", Pergamon Press, N.Y., 1963;
- [19] B. WIDROW and J. B. ANGELL, "Reliable, Trainable Networks for Computing and Control", Aerospace Engineering 21:78-123, 1962;
- [20] K. S. NARENDRA and PARTHASARATHY, "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Trans. on Neural Networks, vol. 1, Nº 1, march 1990.
- [21] K. S. NARENDRA and S. MUKHOPADHYAY, "Intelligent Control Using Neural Networks", IEEE Control Systems, april 1992.
- [22] K. S. NARENDRA and S. MUKHOPADHYAY, "Disturbance Rejection in Nonlinear Systems Using Neural Networks", IEEE Trans. on Neural Networks, vol. 4, Nº 1, january 1993.
- [23] R. S. SCALERO and N. TEPEDELENLIOGLU, "A Fast New Algorithm for Training Feedforward Neural Networks", IEEE Trans. on Signal Processing, vol. 40, Nº 1, 1992.
- [24] G. QIU, M. R. VARLEY and T. J. TERREL, "Accelerated Training of Backpropagation Networks by Using Adaptive Momentum Step", Electronics Letters, vol. 28, Nº 4, 1992.
- [25] R. PEDONE and D. PARISI, "Learning the Learning Parameters", IEEE International Joint Conference on Neural Networks, p. 2033-2037, 1991.
- [26] M. K. MICHAEL, "A Method for Self-Determination of Adaptive Learning Rates in Back Propagation", Neural Networks, vol. 4, p. 371-379, 1991.
- [27] Y. JIA and Y. DALI, "Analysis of the Misadjustment of BP Networks and an Improved Algorithm", Proceedings of the IEEE International Symposium on Circuits and Systems, 1993.



- [28] X. YU, N. K. LOH and W. C. MILLER, " A New Acceleration Technique for the Backpropagation Algorithm", IEEE International Conference on Neural Networks, 1993.
- [29] V. V. PHANSALKAR and P. S. SASTRY, "Analysis of the Back-Propagation Algorithm with Momentum", IEEE Trans. on Neural Networks, vol. 5, Nº 3, 1994.
- [30] A. C. HARRY and T. L. OLIVIER, "Learning Coefficient Dependence on Training Set Size", Neural Networks, vol. 5, p. 283-288, 1992.
- [31] M. T. HAGAN and M. B. MENHAJ, "Training Feedforward Networks with the Marquardt Algorithm", IEEE Trans. on Neural Networks, vol 5, Nº 6, November 1994.
- [32] T. KOHONEN, "Self-Organization and Associative Memory", 2d ed. New York: Springer- Verlag, 1988.
- [33] S. GROSSBERG, "Some Networks that Can Learn, Remember and Reproduce any Number of Complicated Space-Time Patterns", Journal of Mathematics and Mechanics, 1969.
- [34] T. J. SEJNOWSKI and C. R. ROSENBERG, "Parallel Networks that Learn to Pronounce English text", Complex Systems, 1987.
- [35] H. TOLLE and E. ERSÜ, "Lecture Notes in Control and Information Sciences", Edited by M. Thoma and A. Wyner, 1992;
- [36] W. T. MILLER, F. H. GLANZ and L. G. KRAFT, "CMAC: An Associative Neural Network Alternative to Backpropagation", reprinted from Proc. IEEE, vol 78, Nº 10, Oct. 1990, pp. 1561-1567.
- [37] L. G. KRAFT and D. P. CAMPAGNA, "A Comparasion Between CMAC Neural Network Control and Two Traditional Adaptive Control Systems", IEEE Control Systems Magazine, April, 1990.



- [38] Y. WONG and A. SIDERIS, "Learning Convergence in the Cerebellar Model Articulation Controller", IEEE Transactions on Neural Networks, vol. 1, January, 1992.
- [39] P. C. PARKS and J. MILITZER, "A Comparison of Five Algorithms for the Training of CMAC Memories for Learning Control Systems", Automatica, vol. 28, Nº 5, pp 1027-1035, 1992.
- [40] S. H. LANE, D. A. HANDELMAN and J. J. GELFAND, "Theory and Development of Higher-Order CMAC Neural Networks, IEEE Control Systems, April 1992.
- [41] E. FERRERO e A. TEBEROWSKY, "Psicogênese da Língua Escrita", Porto Alegre-RS, Edit. Artes Médicas, 1985.
- [42] H. J. SIRA-RAMÍREZ and S.H. ZAK, "The Adaptation of Perceptrons with Applications to Inverse Dynamics Identification of Unknown Dynamic Systems", IEEE Trans. on Systems, Man and Cybernetics, vol. 21, Nº 3, May/June, 1991.
- [43] B. WIDROW and R. WINTER, "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition", IEEE Computer, vol. 21, Nº 3, pp. 25-39, Mar, 1988.
- [44] H. P. GRAF, L. D. JACKEL and W. E. HUBBARD, "Implementation of Neural Network Model", IEEE Computer, March 1988, pp. 41-49.
- [45] H. P. GRAF and L. D. JACKEL, "Analog Electronic Neural Network Circuits", IEEE Circuits and Devices Mag., July 1989, pp. 44-55.
- [46] J. ZURADA, "Implementation of Neural Networks", IEEE Circuits and Devices Mag., September 1992, pp. 36-41.
- [47] G. L. DEMPSEY and E. S. McVEY, "A Hough Transform System Based on Neural Networks", IEEE Trans. on Industrial Electronics, vol. 39, Nº 6, December, 1992.



APÊNDICE - I

ALGORITMO " BACKPROPAGATION " ORIGINAL

A-I.1



Para facilitar o entendimento, o equacionamento será feito em duas etapas. A primeira estudará a última camada, e a segunda estudará as demais. No final da demonstração, será concluído que as fórmulas matemáticas envolvidas podem ser generalizadas de forma que possam ser utilizadas para qualquer elemento de qualquer camada.

O motivo de se analisar inicialmente as camadas de forma separada, está ligado ao fato de que para RNA de múltiplas camadas, não se conhece as saídas desejadas dos elementos das camadas intermediárias. O "professor" no caso, somente tem conhecimento do que deseja nas saídas dos elementos da última camada. E como se sabe, é esta informação que será transmitida para os outros neurônios para que possam modificar os valores dos parâmetros ajustáveis (pesos) adequadamente.

Inicialmente define-se uma função custo que se deve minimizar, buscando com isto melhorar o desempenho do sistema. Esta função é do tipo erro quadrático e pode ser calculada utilizando as saídas dos elementos da última camada, conforme definido a seguir:

$$E^2 = \sum_i^{k(l)} E_i^2 . \quad (A-I.1)$$

O erro E_i é calculado sempre entre a saída da rede e a respectiva saída desejada, sendo E_i^2 o produto escalar de E_i por E_i . Para uma melhor interpretação pode-se observar a Fig. A1, que representa os elementos da última camada.

É mostrado esquematicamente como é feito o cálculo dos erros de cada saída, e como se inicia a propagação desta informação às demais camadas.



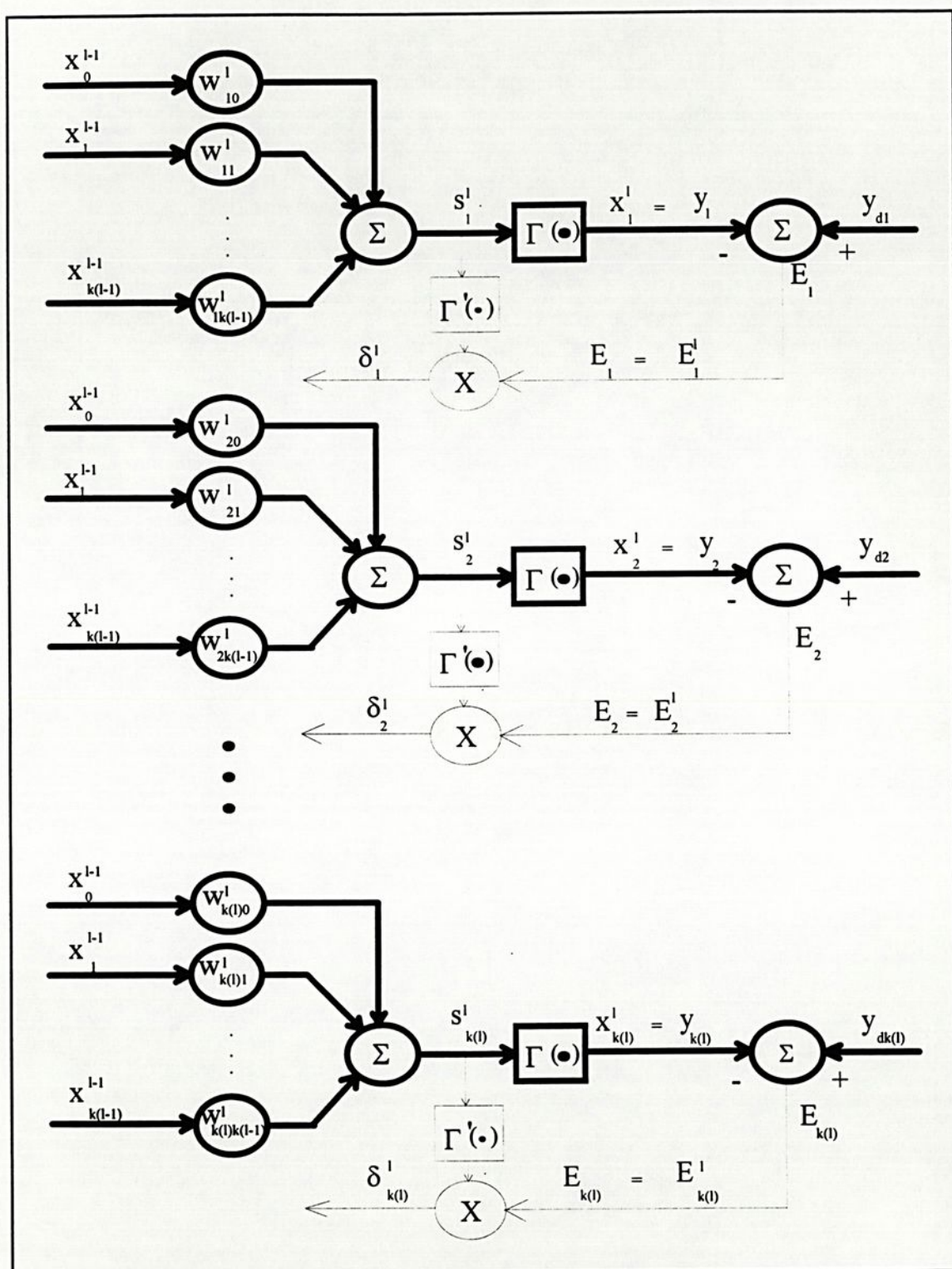


Fig. A1 - Representação esquemática dos neurônios da última camada.

Observando-se a Fig. A1 e utilizando a definição dada em (A-I.1), percebe-se que o erro quadrático total pode ser calculado como segue:

$$E^2 = \sum_j^{k(l)} (y_{d j} - y_j)^2 . \quad (A-I.2)$$

Ainda da Fig. A1 tem-se que o vetor Y é função do vetor S, que é função do vetor W, ou seja:

$$E^2 = E^2 (Y (S (W))) . \quad (A-I.3)$$

O ajuste dos pesos pelo algoritmo "Backpropagation" é dado pelo método do gradiente descendente, utilizando o vetor do gradiente instantâneo na direção contrária, ou seja:

$$W_{(atual)} = W_{(anterior)} + \mu (- \nabla) , \quad (A-I.4)$$

onde:

$$\nabla = \frac{\partial E^2}{\partial W_{(atual)}} , \quad (A-I.5)$$

e μ é uma constante.

O objetivo então é determinar a derivada parcial do erro quadrático com relação ao vetor peso, para que o gradiente possa ser calculado e utilizado para o ajuste dos respectivos pesos da RNA.



Por (A-I.3) e pela regra da cadeia,

$$\frac{\partial E^2}{\partial W} = \frac{\partial E^2}{\partial Y} \cdot \frac{\partial Y}{\partial S} \cdot \frac{\partial S}{\partial W} .$$

(A-I.6)

Na equação (A-I.6) todas as variáveis são grandezas vetoriais. Detalhando, pode-se escrever cada componente do gradiente, conforme segue:

$$\frac{\partial E^2}{\partial w_{ij}^l} = \frac{\partial E^2}{\partial y_i} \cdot \frac{\partial y_i}{\partial s_i^l} \cdot \frac{\partial s_i^l}{\partial w_{ij}^l} .$$

(A-I.7)

Derivando a equação (A-I.2) com relação a y_i tem-se o primeiro termo da equação (A-I.7),

$$\frac{\partial E^2}{\partial y_i} = -2 \cdot (y_{di} - y_i) .$$

(A-I.8)

Observando a Fig. A1,

$$y_i = x_i^l = \Gamma(s_i^l) ,$$

(A-I.9)

de onde se pode calcular o segundo termo da equação (A-I.7),

$$\frac{\partial y_i}{\partial s_i^l} = \Gamma'(s_i^l) .$$

(A-I.10)



Para facilitar a dedução do terceiro termo da equação (A-I.7), a Fig. A1 deve ser observada com atenção mais uma vez, pois fica fácil ver que:

$$s_i^l = \sum_{j=0}^{k(l)} w_{ij}^l \cdot x_j^{l-1}, \quad (\text{A-I.11})$$

Portanto pode-se calcular o terceiro termo facilmente, necessitando para isso apenas derivar a equação (A-I.11) em relação aos parâmetros pesos, conforme segue:

$$\frac{\partial s_{ij}^l}{\partial w_{ij}^l} = \frac{\partial}{\partial w_{ij}^l} \left(\sum_{j=0}^{k(l)} w_{ij}^l \cdot x_j^{l-1} \right), \quad (\text{A-I.12})$$

resultando portanto:

$$\frac{\partial s_{ij}^l}{\partial w_{ij}^l} = x_j^{l-1}. \quad (\text{A-I.13})$$

Substituindo as equações (A-I.8), (A-I.10) e (A-I.13) na equação (A-I.7), tem-se:

$$\frac{\partial E^2}{\partial w_{ij}^l} = -2 \cdot (y_{di} - y_i) \cdot \Gamma'(s_i^l) \cdot x_j^{l-1}. \quad (\text{A-I.14})$$

Definindo o erro derivativo como:

$$\delta_i^l \triangleq -\frac{1}{2} \cdot \left(\frac{\partial E^2}{\partial s_i^l} \right), \quad (\text{A-I.15})$$



utilizando novamente a regra da cadeia, tem-se:

$$\delta_i^l = -\frac{1}{2} \cdot \left(\frac{\partial E^2}{\partial y_i} \cdot \frac{\partial y_i}{\partial s_i^l} \right).$$

(A-I.16)

Agora substituindo (A-I.8) e (A-I.10) em (A-I.16), tem-se:

$$\delta_i^l = (y_{di} - y_i) \cdot \Gamma'(s_i^l).$$

(A-I.17)

Substituindo (A-I.17) em (A-I.14), tem-se finalmente o gradiente, conforme segue:

$$\frac{\partial E^2}{\partial w_{ij}^l} = -2 \cdot \delta_i^l \cdot x_j^{l-1}.$$

(A-I.18)

Assim, com o gradiente calculado por (A-I.18), tem-se condições de fazer adaptação (um passo de iteração) para os pesos da última camada. Para isso basta substituir (A-I.18) em (A-I.4), resultando a equação (A-I.19):

$$w_{ij}^l \text{ (atual)} = w_{ij}^l \text{ (anterior)} + 2 \cdot \mu \cdot \delta_i^l \cdot x_j^{l-1}.$$

(A-I.19)

Para as demais camadas (as intermediárias) não é possível calcular o erro diretamente, pois não se conhece as saídas desejadas naqueles pontos. Porém para se calcular o gradiente, utilizado nas iterações, é suficiente que se conheça as derivadas parciais do erro com relação aos parâmetros a serem adaptados [04].

A seguir, na Fig. A2, está sendo mostrado uma camada de neurônios intermediária qualquer (m), esquematicamente.

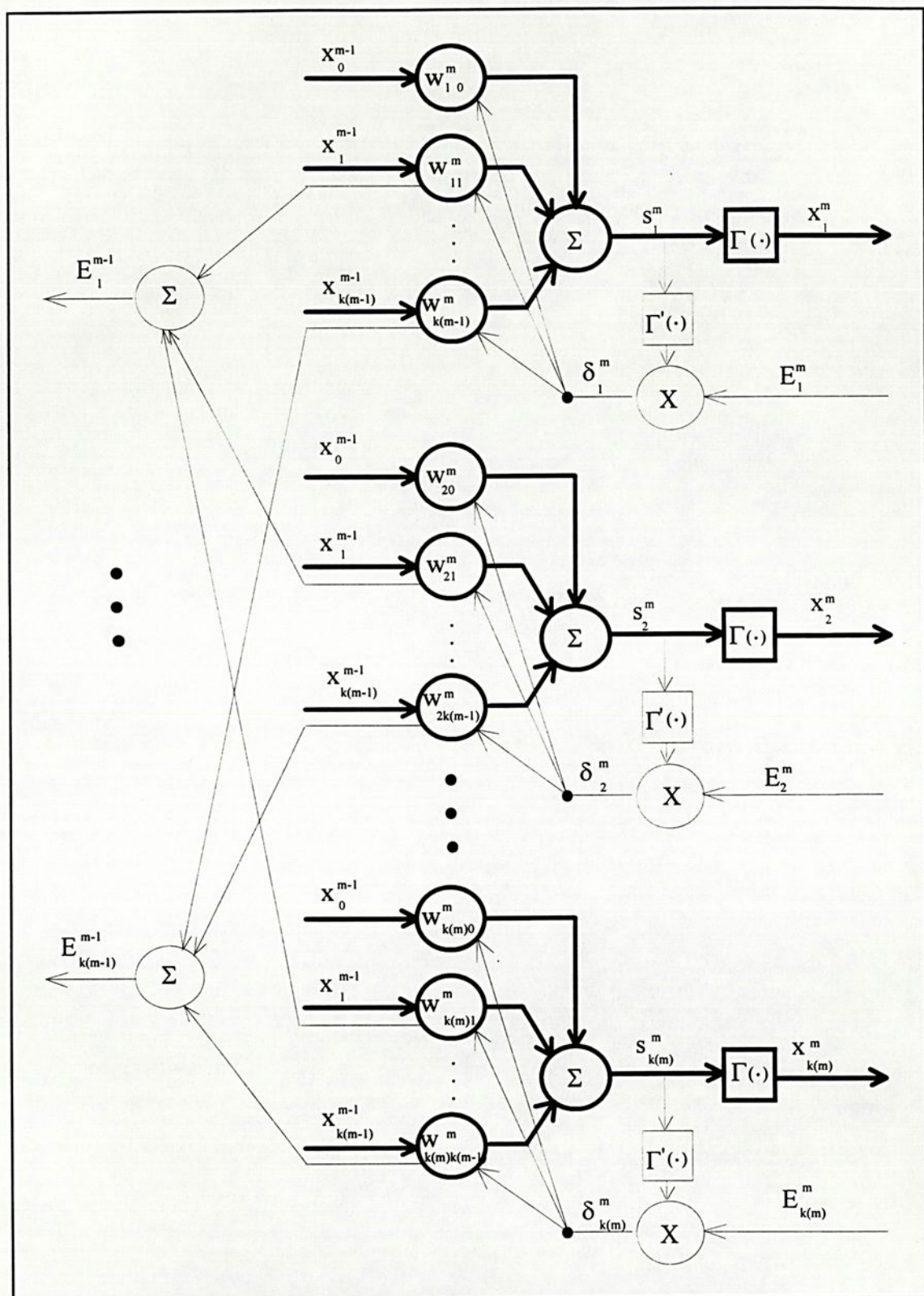


Fig. A2 - Representação esquemática dos neurônios de uma camada intermediária (m)

Para uma camada intermediária qualquer (m) pode-se notar que E^2 é uma função que pode ser vista de diversas maneiras, pois é dependente de vários parâmetros. Como pode ser observado pela Fig. A2, as três equações (A-I.20.a), (A-I.20.b) e (A-I.20.c) escritas abaixo, são válidas, onde todas as variáveis são grandezas vetoriais.

$$E^2 = E^2 \left(X^m \left(S^m \left(W^m \right) \right) \right), \quad (A-I.20.a)$$

$$E^2 = E^2 \left(S^{m+1} \left(X^m \right) \right), \quad (A-I.20.b)$$

$$E^2 = E^2 \left(S^{m+1} \left(S^m \right) \right). \quad (A-I.20.c)$$

Aplicando a regra da cadeia em (A-I.20.a), tem-se:

$$\frac{\partial E^2}{\partial W^m} = \frac{\partial E^2}{\partial X^m} \frac{\partial X^m}{\partial S^m} \frac{\partial S^m}{\partial W^m}. \quad (A-I.21)$$

Novamente, pode-se detalhar e escrever cada componente do gradiente, conforme segue:

$$\frac{\partial E^2}{\partial w_{ij}^m} = \frac{\partial E^2}{\partial x_i^m} \frac{\partial x_i^m}{\partial s_i^m} \frac{\partial s_i^m}{\partial w_{ij}^m}. \quad (A-I.22)$$



Analogamente ao que foi feito em (A-I.10) e (A-I.13), e considerando que:

$$s_i^m = \sum_{j=0}^{k(m)} w_{ij}^m x_j^{(m-1)}, \quad (A-I.23)$$

tem-se que:

$$\frac{\partial E^2}{\partial w_{ij}^m} = \frac{\partial E^2}{\partial x_i^m} \Gamma(s_i^m) x_j^{m-1}. \quad (A-I.24)$$

Para que (A-I.24) esteja totalmente determinado é necessário calcular então o termo

$$\partial E^2 / \partial x_i^m.$$

Efetuada a derivada na equação (A-I.20.b),

$$\frac{\partial E^2}{\partial x_i^m} = \sum_{t=1}^{k(m+1)} \frac{\partial E^2}{\partial s_t^{m+1}} \frac{\partial s_t^{m+1}}{\partial x_i^m}. \quad (A-I.25)$$

Utilizando a definição de erro derivativo, dado pela equação (A-I.15), pode-se escrever para a camada intermediária $(m + 1)$, a seguinte equação:

$$\delta_t^{m+1} = -\frac{1}{2} \left(\frac{\partial E^2}{\partial s_t^{m+1}} \right), \quad (A-I.26)$$



logo:

$$\frac{\partial E^2}{\partial x_i^m} = \sum_{t=1}^{k(m+1)} \left(-2 \delta_t^{m+1} w_{ti}^{m+1} \right) .$$

(A-I.27)

Substituindo (A-I.27) em (A-I.24), tem-se:

$$\frac{\partial E^2}{\partial w_{ij}^m} = -2 \sum_{t=1}^{k(m+1)} \left(\delta_t^{m+1} w_{ti}^{m+1} \right) \Gamma'(s_i^m) x_j^{m-1} .$$

(A-I.28)

Para que se possa generalizar a equação obtida para o cálculo da derivada do erro quadrático (E^2) em relação aos pesos, é necessário mostrar que:

$$\delta_i^m = \left(\sum_{t=1}^{k(m+1)} \delta_t^{m+1} w_{ti}^{m+1} \right) \Gamma'(s_i^m) ,$$

(A-I.29)

ou seja, deve ser mostrado que o erro derivativo que foi definido pela equação (A-I.15), para a última camada, é igual a equação (A-I.29) quando aplicado para qualquer camada intermediária (m).

Analisando (A-I.20.c), percebe-se que:

$$\frac{\partial E^2}{\partial s_i^m} = \sum_{t=1}^{k(m+1)} \frac{\partial E^2}{\partial s_t^{m+1}} \frac{\partial s_t^{m+1}}{\partial s_i^m} .$$

(A-I.30)

Tendo que:

$$\delta_t^{m+1} = -\frac{1}{2} \frac{\partial E^2}{\partial s_t^{m+1}} \quad \text{ou} \quad \frac{\partial E^2}{\partial s_t^{m+1}} = -2 \delta_t^{m+1} ,$$

(A-I.31)



e

$$\frac{\partial s_t^{m+1}}{\partial s_i^m} = \frac{\partial s_t^{m+1}}{\partial x_i^m} \frac{\partial x_i^m}{\partial s_i^m}, \quad (\text{A-I.32})$$

onde:

$$\frac{\partial s_t^{m+1}}{\partial x_i^m} = w_{ti}^{m+1}, \quad (\text{A-I.33})$$

e

$$\frac{\partial x_i^m}{\partial s_i^m} = \Gamma(s_i^m), \quad (\text{A-I.34})$$

e, escrevendo A-I.32 como

$$\frac{\partial s_t^{m+1}}{\partial s_i^m} = w_{ti}^{m+1} \Gamma(s_i^m), \quad (\text{A-I.35})$$

substituindo (A-I.31) e (A-I.35) em (A-I.30),

$$\frac{\partial E^2}{\partial s_i^m} = \sum_{t=1}^{K(m+1)} \left(-2 \delta_t^{m+1} w_{ti}^{m+1} \right) \Gamma(s_i^m). \quad (\text{A-I.36})$$

Assim,



$$\delta_i^m = -\frac{1}{2} \sum_{t=1}^{k(m+1)} (-2 \delta_t^{m+1} w_{ti}^{m+1}) \Gamma'(s_i^m) .$$

(A-I.37)

Portanto,

$$\delta_i^m = \sum_{t=1}^{k(m+1)} \delta_t^{m+1} w_{ti}^{m+1} \Gamma'(s_i^m) .$$

(A-I.38)

Substituindo (A-I.38) em (A-I.28), tem-se:

$$\frac{\partial E^2}{\partial w_{ij}^m} = -2 \delta_i^m x_j^{m-1} .$$

(A-I.39)

De (A-I.18) e (A-I.39), pode-se concluir que para qualquer camada "r", tem-se

$$\frac{\partial E^2}{\partial w_{ij}^r} = -2 \delta_i^r x_j^{r-1} .$$

(A-I.40)

Assim a atualização dos pesos de qualquer camada da rede é dado pela seguinte regra:

$$w_{ij}^r(\text{atual}) = w_{ij}^r(\text{anterior}) + 2 \mu \delta_i^r x_j^{r-1} .$$

(A-I.41)



APÊNDICE - II

ALGORITMO "BACKPROPAGATION" COM MOMENTO

A-II.1



A técnica de momento é uma variante do BA, e muito utilizada no treinamento de RNA's multicamadas, com o objetivo de minimizar o tempo gasto para alcançar a convergência desejada.

Esta técnica inclui no cálculo de cada componente do vetor peso, para cada iteração, um termo de proporcionalidade referente aos valores dos respectivos pesos em iterações anteriores. Isto é, a equação apresentada no apêndice-I (A-I.41), para o cálculo das componentes do vetor peso, para cada iteração no BA "standard", é substituída pelas equações (A-II.1) e (A-II.2), apresentadas a seguir:

$$\Delta w_{ij}^r(\text{atual}) = 2\mu(1-\eta)\delta_i^r x_j^{r-1} + \eta \Delta w_{ij}^r(\text{anterior}) , \quad (\text{A-II.1})$$

$$w_{ij}^r(\text{atual}) = w_{ij}^r(\text{anterior}) + \Delta w_{ij}^r(\text{atual}) , \quad (\text{A-II.2})$$

onde o termo de momento pertence à $[0, 1)$, e na prática usualmente assume valores de 0.8 a 0.9.

As equações acima estão escritas considerando apenas a proporcionalidade referente a iteração imediatamente anterior, que é o mais usual. Teóricamente pode ser considerado o fator de proporcionalidade de várias iterações anteriores, porém isto exigiria mais memórias e um maior tempo computacional, uma vez que o cálculo considerado é efetuado para cada componente do vetor peso, em todas as iterações.



APÊNDICE -III

ALGORITMO "BACKPROPAGATION" MODIFICADO

A-III.1



O método proposto utiliza a função energia definida com o erro filtrado. Considerando uma RNA com apenas uma única saída, e o erro de uma única iteração, a nova função energia para o BA modificado, é dado por:

$$E_f^2 = e_f^2(k) . \quad (A-III.1)$$

Sabendo que o erro filtrado ($e_f(k)$) é dado por,

$$e_f(k) = b_p e_f(k-1) + (1 - b_p) e(k) , \quad (A-III.2)$$

podemos calcular cada componente do gradiente, da última camada (1), conforme segue:

$$\frac{\partial E_f^2}{\partial w_{ij}^1(k)} = 2 e_f(k) (1 - b_p) \frac{\partial e(k)}{\partial w_{ij}^1(k)} . \quad (A-III.3)$$

Sabendo que:

$$e(k) = (y_d(k) - y(k)) , \quad (A-III.4)$$

podemos obter:

$$\frac{\partial e(k)}{\partial w_{ij}^1(k)} = \frac{\partial e(k)}{\partial y(k)} \frac{\partial y(k)}{\partial s_1^1(k)} \frac{\partial s_1^1(k)}{\partial w_{ij}^1(k)} , \quad (A-III.5)$$

ou ainda por (A-I.10) e (A-I.13) encontradas no apêndice-I,



$$\frac{\partial e(k)}{\partial w_{ij}^l(k)} = - \Gamma'(s_i^l(k)) x_j^{(l-1)}(k) ,$$

(A-III.6)

assim:

$$\frac{\partial E_f^2(k)}{\partial w_{ij}^l(k)} = - 2 e_f(k) (1 - b_p) \Gamma'(s_i^l(k)) x_j^{(l-1)}(k) .$$

(A-III.7)

Tendo que:

$$\delta_i^l(k) = - \frac{1}{2} \frac{\partial E_f^2(k)}{\partial s_i^l(k)} ,$$

(A-III.8)

e utilizando (A-III.1), (A-III.2) e (A-I.10), sendo esta última do apêndice-I, podemos escrever:

$$\delta_i^l(k) = e_f(k) (1 - b_p) \Gamma'(s_i^l(k)) .$$

(A-III.9)

Substituindo (A-III.9) em (A-III.7), temos:

$$\frac{\partial E_f^2(k)}{\partial w_{ij}^l(k)} = - 2 \delta_i^l(k) x_j^{(l-1)}(k) .$$

(A-III.10)

Então a atualização de cada peso da última camada da RNA, é dada por:



$$w_{ij}^l(k) = w_{ij}^l(k-1) + 2\delta_i^l(k)x_j^{(l-1)}(k) .$$

(A-III.11)

Observamos que (A-III.11) é análoga a equação (A-I.19) encontrada no apêndice-I, a diferença está no erro derivativo (δ), equação (A-I.17) para o caso "standard" e (A-III.9) para o caso do método proposto.

Com relação as demais camadas, verificamos que é análogo ao BA "standard", ou seja, as equações de (A-I.20.a) a (A-I.20.c) encontradas no apêndice-I, são válidas também para o caso do erro filtrado.

Sendo assim, procedendo de maneira análoga ao apêndice-I, chegaremos aos mesmos resultados, tanto para o erro derivativo como para o ajuste final dos pesos, para as demais camadas, conforme pode ser verificado abaixo.

Analogamente ao procedimento tomado para o caso do BA "standard", temos para uma camada intermediária qualquer (m), que E_f^2 é uma função que pode ser vista de diversos maneiras por ser dependente de vários parâmetros.

Temos:

$$E_f^2 = E_f^2 \left(X^m \left(S^m \left(W^m \right) \right) \right),$$

(A-III.12.a)

$$E_f^2 = E_f^2 \left(S^{m+1} \left(X^m \right) \right),$$

(A-III.12.b)

$$E_f^2 = E_f^2 \left(S^{m+1} \left(S^m \right) \right) .$$

(A-III.12.c)

Aplicando a regra da cadeia em (A-III.12.a), tem-se:



$$\frac{\partial E_f^2}{\partial W^m} = \frac{\partial E_f^2}{\partial X^m} \frac{\partial X^m}{\partial S^m} \frac{\partial S^m}{\partial W^m} .$$

(A-III.13)

Novamente, pode-se detalhar e escrever cada componente do gradiente, conforme segue:

$$\frac{\partial E_f^2}{\partial w_{ij}^m} = \frac{\partial E_f^2}{\partial x_i^m} \frac{\partial x_i^m}{\partial s_i^m} \frac{\partial s_i^m}{\partial w_{ij}^m} .$$

(A-III.14)

Analogamente ao que foi feito em (A-I.10) e (A-I.13), e considerando que:

$$s_i^m = \sum_{j=0}^{k(m)} w_{ij}^m x_j^{(m-1)} ,$$

(A-III.15)

tem-se que:

$$\frac{\partial E_f^2}{\partial w_{ij}^m} = \frac{\partial E_f^2}{\partial x_i^m} \Gamma(s_i^m) x_j^{m-1} .$$

(A-III.16)

Para que (A-III.16) esteja totalmente determinado é necessário calcular então o termo

$$\partial E_f^2 / \partial x_i^m .$$



Efetuada a derivada na equação (A-III.12.b),

$$\frac{\partial E_f^2}{\partial x_i^m} = \sum_{t=1}^{k(m+1)} \frac{\partial E_f^2}{\partial s_t^{m+1}} \frac{\partial s_t^{m+1}}{\partial x_i^m} .$$

(A-III.17)

Utilizando a definição de erro derivativo, dado pela equação (A-III.8), pode-se escrever para a camada intermediária $(m + 1)$, a seguinte equação:

$$\delta_t^{m+1} = -\frac{1}{2} \left(\frac{\partial E_f^2}{\partial s_t^{m+1}} \right) ,$$

(A-III.18)

logo:

$$\frac{\partial E_f^2}{\partial x_i^m} = \sum_{t=1}^{k(m+1)} \left(-2 \delta_t^{m+1} w_{ti}^{m+1} \right) .$$

(A-III.19)

Substituindo (A-III.19) em (A-III.16), tem-se:

$$\frac{\partial E_f^2}{\partial w_{ij}^m} = -2 \sum_{t=1}^{k(m+1)} \left(\delta_t^{m+1} w_{ti}^{m+1} \right) \Gamma'(s_i^m) x_j^{m-1} .$$

(A-III.20)

Para que se possa generalizar a equação obtida para o cálculo da derivada do erro quadrático (E_f^2) em relação aos pesos, é necessário mostrar que:

$$\delta_i^m = \left(\sum_{t=1}^{k(m+1)} \delta_t^{m+1} w_{ti}^{m+1} \right) \Gamma'(s_i^m) ,$$

(A-III.21)



Analisando (A-III.12.c), percebe-se que:

$$\frac{\partial E_f^2}{\partial s_i^m} = \sum_{t=1}^{k(m+1)} \frac{\partial E_f^2}{\partial s_t^{m+1}} \frac{\partial s_t^{m+1}}{\partial s_i^m} .$$

(A-III.22)

Tendo que:

$$\delta_t^{m+1} = - \frac{1}{2} \frac{\partial E_f^2}{\partial s_t^{m+1}} \quad \text{ou} \quad \frac{\partial E_f^2}{\partial s_t^{m+1}} = - 2 \delta_t^{m+1} ,$$

(A-III.23)

e

$$\frac{\partial s_t^{m+1}}{\partial s_i^m} = \frac{\partial s_t^{m+1}}{\partial x_i^m} \frac{\partial x_i^m}{\partial s_i^m} ,$$

(A-III.24)

onde:

$$\frac{\partial s_t^{m+1}}{\partial x_i^m} = w_{ti}^{m+1} ,$$

(A-III.25)

e

$$\frac{\partial x_i^m}{\partial s_i^m} = \Gamma(s_i^m) ,$$

(A-III.26)



e, escrevendo (A-III.24) como:

$$\frac{\partial s_t^{m+1}}{\partial s_i^m} = w_{ti}^{m+1} \Gamma(s_i^m), \quad (\text{A-III.27})$$

substituindo (A-III.27) e (A-I.23) em (A-III.22),

$$\frac{\partial E_f^2}{\partial s_i^m} = \sum_{t=1}^{K(m+1)} (-2 \delta_t^{m+1} w_{ti}^{m+1}) \Gamma'(s_i^m). \quad (\text{A-III.28})$$

Assim,

$$\delta_i^m = -\frac{1}{2} \sum_{t=1}^{K(m+1)} (-2 \delta_t^{m+1} w_{ti}^{m+1}) \Gamma'(s_i^m). \quad (\text{A-III.29})$$

Portanto,

$$\delta_i^m = \sum_{t=1}^{K(m+1)} \delta_t^{m+1} w_{ti}^{m+1} \Gamma'(s_i^m). \quad (\text{A-III.30})$$

Substituindo (A-III.30) em (A-III.20), tem-se:

$$\frac{\partial E_f^2}{\partial w_{ij}^m} = -2 \delta_i^m x_j^{m-1}. \quad (\text{A-III.31})$$

De (A-III.10) e (A-III.31), pode-se concluir que para qualquer camada "r", tem-se



$$\frac{\partial E_f^2}{\partial w_{ij}^r} = -2 \delta_i^r x_j^{r-1} .$$

(A-III.32)

Assim a atualização dos pesos de qualquer camada da rede é dado pela seguinte regra:

$$w_{ij}^{r(\text{atual})} = w_{ij}^{r(\text{anterior})} + 2 \mu \delta_i^r x_j^{r-1} .$$

(A-III.33)



APÊNDICE - IV

*ARTIGO PUBLICADO NO "38TH MIDWEST SYMPOSIUM ON CIRCUITS
AND SYSTEMS" PROMOVIDO PELO IEEE, REALIZADO NO RIO DE JANEIRO
EM AGOSTO DE 1995*



NEURAL NETWORKS TRAINING USING THE CONSTRUCTIVISM PARADIGMS

Marcelo Carvalho Minhoto Teixeira and Décio Lamas

Faculdade de Engenharia de Ilha Solteira - FEIS - UNESP
- Department of Electrical Engineering -
Av. Brasil, 56 - 15385-000 Ilha Solteira, SP - Brazil

ABSTRACT

The Backpropagation Algorithm (BA) is the standard method for training multilayer Artificial Neural Networks (ANN), although it converges very slowly and can stop in a local minimum. We present a new method for neural network training using the BA inspired on constructivism, an alphabetization method proposed by Emilia Ferreiro based on Piaget philosophy. Simulation results show that the proposed configuration usually obtains a lower final mean square error, when compared with the standard BA and with the BA with momentum factor.

1. INTRODUCTION

The multilayer Artificial Neural Networks (ANN) are very used currently. The Backpropagation Algorithm (BA) has become the standard algorithm for training multilayer ANN.

The BA was introduced by Werbos in 1974 and it was responsible by an explosive growth in pure and applied research related to Neural Networks [1].

Although the BA is capable of providing reasonable estimates of the weighting vectors, this algorithm converges very slowly.

There are papers [2] [3] [4], that propose new acceleration techniques for the BA. The standard technique used for the BA acceleration is the momentum term [5]. Another problem of the BA is that the learning may stop in a local minimum. This paper presents a new technique for the BA. The idea is to use the paradigms of the constructivism to improve the learning.

The constructivism is a new alphabetization method, proposed by Emilia Ferreiro, based on Piaget philosophy, that is being used with success in some Brazilian schools [6].

In the constructivism alphabetization, the children learn initially the general things, and then, the details.

This idea, applied in ANN, can be implemented supplying initially only the low frequencies of the error between the ANN output and the desired output. When the ANN is learning we increase the details of the error, supplying continuously the high frequencies.

We used an adaptive low-pass filter, where the bandwidth increases with the learning and the filtering error is used to adjust the ANN.

Simulations using the standard BA and the new proposed algorithm are presented. The examples show that the proposed method usually works better than the standard BA and the BA with momentum factor, with respect to the final mean-square error.

2. THE LEARNING STRUCTURE OF THE PROPOSED METHOD

Fig. 1 presents the structure of the new learning method.

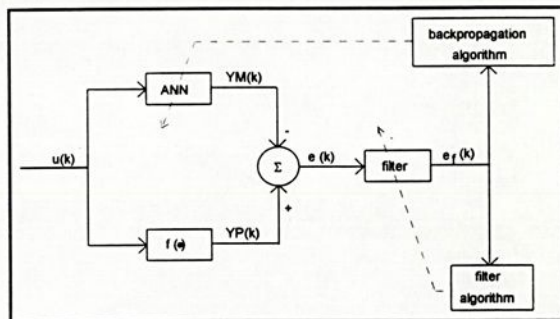


Fig. 1 - Diagram of the learning structure.

We choose a first order filter, given by:

$$e_f(k+1) = b e_f(k) + (1-b) e(k+1), \quad (1)$$

where:

- $e_f(k+1)$ is the output of the filter (filtering error) in the iteration $(k+1)$;
- $e(k+1)$ is the input of the filter and the error between the ANN and the desired output in the iteration $(k+1)$;

When $b = 0$, $e_f(k+1) = e(k+1)$, and we have an all-pass filter. When $b = 1$, $e_f(k+1) = e_f(k)$, and we have a nothing-pass filter.

Then, we will use $b \in (0, 1]$, and b will decrease from $b_{\max} > 0$ to zero, when the ANN learns.

When and how to adjust b are the major difficulties of this method. We know that the adjust of b is made when the ANN learns. There are several ways to obtain this information and we used the difference of the average of $|e_f(k)|$ for two consecutive periodical presentations of all

distinct values of $u(k)$ in Fig. 1.

During the simulations, we note that the adjust of b must be slower than the ANN weights adjust. We used the following law:

$$b(p+1) = b(p) - \frac{\gamma}{2} [1 - \text{sgn}(M_p(e_f) - M_{p-1}(e_f))], \quad (2)$$

where:

- γ is the adjust parameter of b , $\gamma > 0$ and constant;
- $M_p(e_f)$ is the average of $|e_f|$ for the periodical presentation number "p" of all distinct values of $u(k)$ in Fig. 1;
- $b(p)$ is the filter parameter in the periodical presentation p, and

$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0. \end{cases} \quad (3)$$

The value of γ is obtained by a trial and error procedure, but such that $b=0$ few iterations before of the stabilization of the mean square error. We are studying techniques for an automatic determination of γ , and we already have some results.

The adjust of the weights in Fig. 1 is made using the BA for the energy function

$$E_f = \sum_{k=m+1}^{m+n} e_f(k) \quad (4)$$

We adopted in this paper, $n = 1$.

Then, from (1) and considering that b is constant in one presentation of all data for training we have:

$$\frac{\partial E_f}{\partial w_{ij}} = \sum_{k=m+1}^{m+n} 2e_f(k)(1-b) \frac{\partial e(k)}{\partial w_{ij}}, \quad (5)$$

where w_{ij} is one weight of the ANN. Then, we use (5) in BA to adjust the weights of the ANN in fig.1.

The simulation results show the potentiality of this method.

3. SIMULATION RESULTS

This new method was applied in the identification of unknown functions.

Extensive numerical simulations have been carried and we will present the best results for each case:

Example 1 - Consider the function given by:

$$f[u(k)] = 0.6\sin[\pi u(k)] + 0.3\sin[3\pi u(k)] + 0.1\sin[5\pi u(k)], \quad (6)$$

with, $u(k) \in [-1.0, 1.0]$, and a three-layer neural network with 7,5 and 1 neurons, in the first, second and third layer, respectively.

Case 1- BA (standard) with learning rate $\mu=0.02$, momentum term $\eta=0$.

Case 2- BA with learning rate $\mu=0.1$, momentum term $\eta=0.8$

Case 3- BA using filtering error with learning rate $\mu=0.05$, momentum term $\eta=0$, adjust term of b , $\gamma=0.003$.

Fig. 2 presents the mean-square errors for each case. Fig. 3 present the desired curve and the ANN output for each case.

We note, from Fig. 2, that the proposed method presented the best final performance.

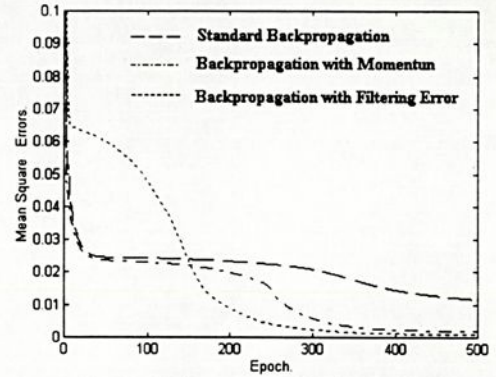


Fig. 2 - Performance of the three methods.

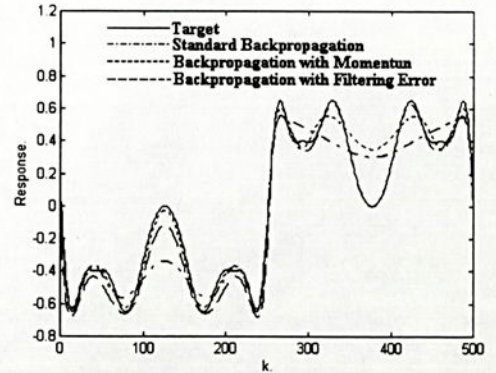


Fig.3 - Comparison of the three methods.

Example 2 - Consider the function given by:

$$f[u(k)] = 0.80\sin[\pi u(k)](1.41u(k) + 1.59) - 0.05u^2(k), \quad (7)$$

with, $u(k) \in [-1.0, 1.0]$, and a three-layer neural network



with 5,3 and 1 neurons, in the first, second and third layer, respectively.

- Case 1-** BA (standard) with learning rate $\mu=0.07$, momentum term $\eta=0$.
- Case 2-** BA with learning rate $\mu=0.05$, momentum term $\eta=0.6$.
- Case 3-** BA using filtering error with learning rate $\mu=0.05$, momentum term $\eta=0$, adjust term of b, $\gamma=0.003$.

Fig. 4 presents the mean-squared errors for each case. Fig. 5 presents the desired curve and the ANN output for each case. We note, from Fig. 4, that the proposed method presented the best final performance.

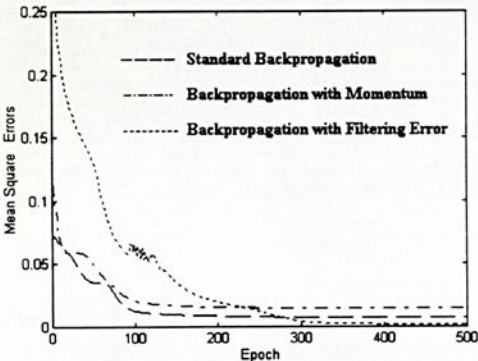


Fig. 4 - Performance of the three methods.

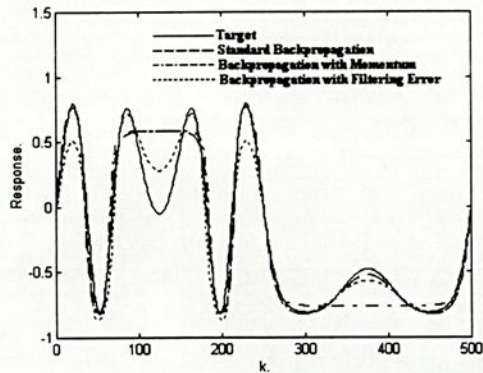


Fig. 5 - Comparison of the three methods.

Example 3 - Consider the function given by:

$$f(x,y) = \sin\left(\frac{2 \pi x}{x_{\max}}\right) \sin\left(\frac{2 \pi y}{y_{\max}}\right), \tag{8}$$

with $0.0 \leq x,y \leq 19.0$ in the learning and $0.0 \leq x,y \leq 49.0$, in the verification of the learning.

- Case 1-** BA (standard) with learning rate $\mu=0.02$, momentum term $\eta=0$.
- Case 2-** BA with learning rate $\mu=0.1$, momentum term $\eta=0.8$.
- Case 3-** BA using filtering error with learning rate $\mu=0.05$, momentum term $\eta=0$, adjust term of b, $\gamma=0.003$. Fig. 6 presents the plot of (8).

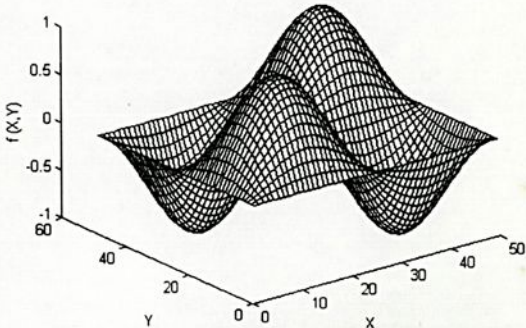


Fig. 6 - Plot of $f(x,y)$ given in (8).

Example 3.1 - The neural network with two-layer, with 5 and 1 neurons, in the first and second layer, respectively. Fig. 7 presents the mean squared errors for each case. Fig. 8, 9 and 10 present the ANN output for each case. We note, from Fig. 7, that the proposed method presented the best final performance.

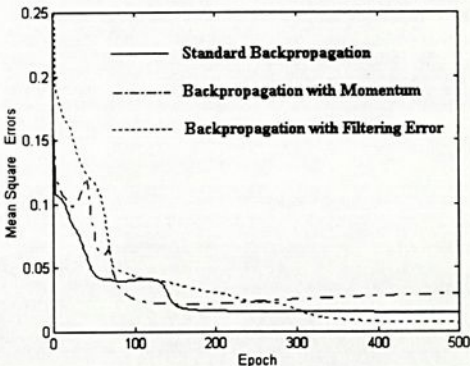


Fig. 7 - Performance of the three methods.



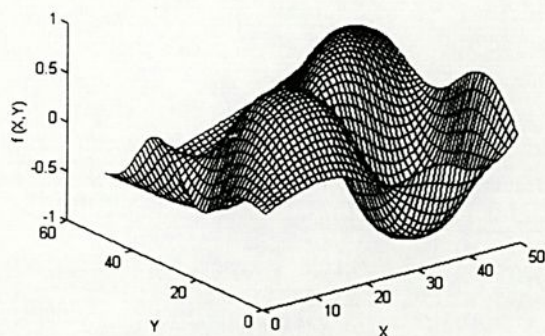


Fig. 8 - ANN output for the case 1.

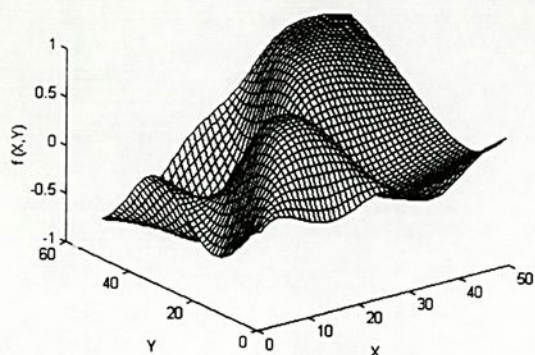


Fig. 9 - ANN output for the case 2.

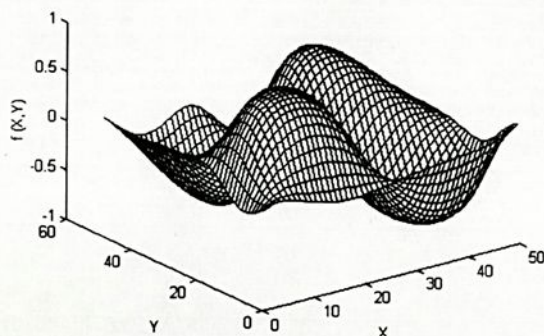


Fig. 10 - ANN output for the case 3.

Example 3.2 - We use now a neural network three-layer, with 10,10 and 1 neurons in the first, second and third layer, respectively.

Fig. 11 presents the-mean square error for each case. In this example, the three methods presented the same final performance, described in Fig. 12.

4. CONCLUSIONS

A new method for neural networks training using the BA and inspired on constructivism was presented.

Extensive simulation results show that, usually, the proposed configuration allows a lower final mean square error, when compared with the standard BA and with the BA with momentum factor. Future works will be the comparison with other methods and the automatization of the filter parameter "b" adjust in (1).

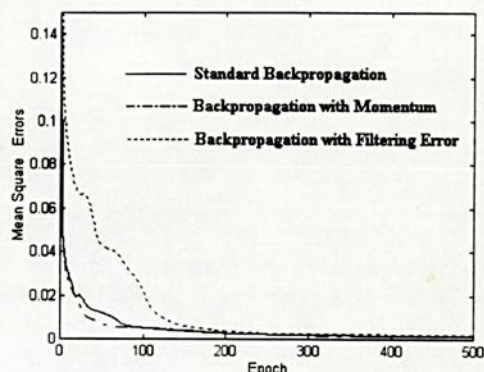


Fig. 11 - Performance of the three methods.

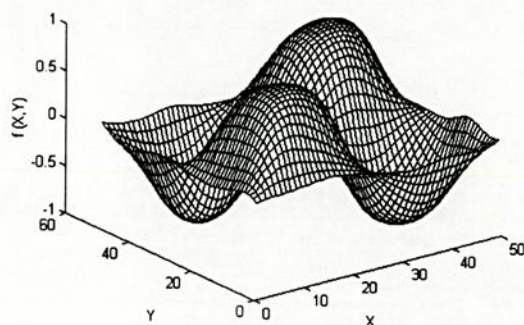


Fig. 12 - ANN output for the case 3.

5. ACKNOWLEDGEMENT

We would like to thank FAPESP and CNPq for the financial support.

6. REFERENCES

- [1] P. D. WASSERMAN, "Neural Computing: Theory and Practice", Library of Congress Cataloging in Publication Data, 1989.
- [2] X. YU, N. K. LOH and W. C. MILLER, "A New Acceleration Technique for the Backpropagation Algorithm", IEEE International Conference on Neural Networks, pp. 1157-11-61, 1993.
- [3] M. K. WEIR, "A Method for Self-Determination of Adaptive Learning Rates in Backpropagation", Neural Networks, vol. 4, pp. 371-379, 1991.
- [4] G. QIU, M.R. VARLEY and T. J. TERRELL, "Accelerated Training of Backpropagation Networks by Using Adaptive Momentum Step", Electronics Letters, vol. 28, pp. 377-379, N° 4, 13th February, 1992.
- [5] D. E. RUMELHART, G. E. HINTON and R. J. WILLIAMS, "Learning Internal Representations by Error Propagation". In Parallel Distributed Processing, vol. 1, Cambridge, MA: Mit Press, 1986.
- [6] E. FERREIRO and A. TEBEROWSKY, "Psicogênese da Língua Escrita", Porto Alegre-RS, Edit. Artes Médicas, 1985 (in Portuguese).

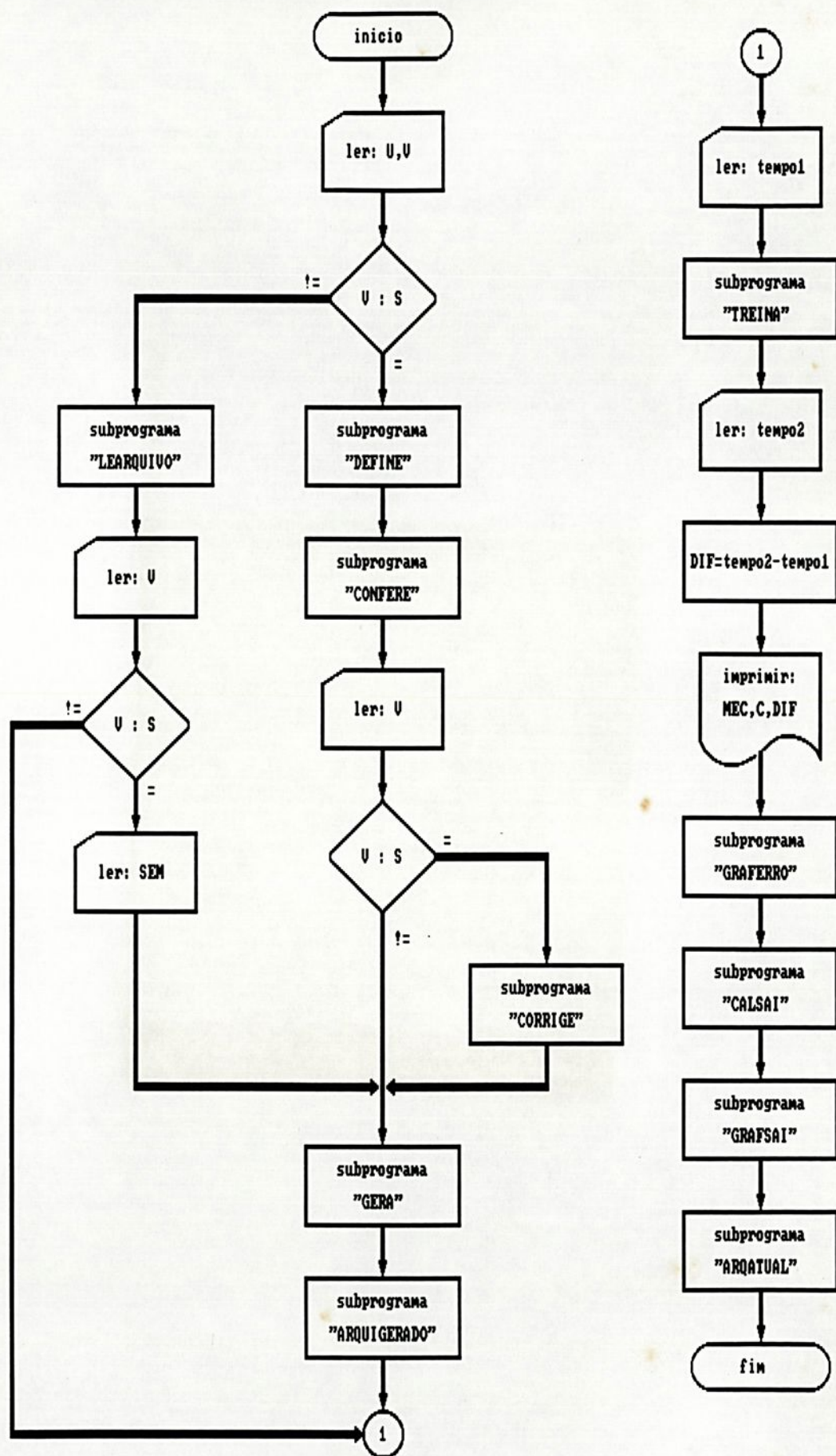
APÊNDICE - V

FLUXOGRAMA RESUMIDO DO "SOFTWARE" UTILIZADO

A-V.1



ESTE FLUXOGRAMA É REFERENTE AO PROGRAMA DENOMINADO "SIMULADOR COM REDES NEURONAIS" DESENVOLVIDO EM LINGUAGEM "C" PELO ENGO. DECIO LAMAS, SENDO UMA ATIVIDADE DO CURSO DE POS GRADUACAO EM ENGENHARIA DA FEIS-UNESP EM MARÇO/94. ESTE PROGRAMA FAZ SIMULACOES RESOLVENDO PROBLEMAS DE IDENTIFICACAO E CONTROLE DE SISTEMAS DINAMICOS NAO-LINEARES COM PARAMETROS DESCONHECIDOS, UTILIZANDO O ALGORITMO "BACKPROPAGATION" DINAMICO PARA O AJUSTE DOS PARAMETROS DAS REDES NEURONAIS ARTIFICIAIS ENVOLVIDAS.



A-V.2

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
AV. BRASIL, 56 - CAIXA POSTAL 31
15378-000 ILHA SOLTEIRA - SP

