CONTROLADORES PREDITIVOS BASEADOS EM REDES NEURAIS ARTIFICIAIS

por

Leizer Schnitman

Dissertação de Mestrado apresentada

ao Departamento de Engenharia Elétrica

da Universidade Federal da Bahia

como parte dos requisitos para obtenção

do título de Mestre em Engenharia Elétrica

Outubro de 1998

CONTROLADORES PREDITIVOS BASEADOS EM REDES NEURAIS ARTIFICIAIS

por

Leizer Schnitman

Banca Examinadora:

Prof. Adhemar de Barros Fontes - Orientador

Prof. Dr. Caiuby Alves da Costa - DEE/UFBA

Prof. Dr. Luiz Pereira Calôba - COPPE/UFRJ

Outubro de 1998

Resumo

Este trabalho apresenta um estudo sobre Inteligência Artificial, mais especificamente a aplicação de Redes Neurais Artificiais em controle preditivo de sistemas dinâmicos não lineares. Além de uma abordagem geral sobre algumas técnicas de utilização de Inteligência Artificial, faz-se uma revisão histórica das Redes Neurais Artificiais e é dada uma ampla visão das diversas topologias registradas em bibliografia. Numa segunda etapa, mais específica, é feita uma abordagem das Redes Neurais Artificiais com um enfoque dado pelo ponto de vista da teoria de controle.

Em diagramas de blocos, são citadas diversas formas para a aplicação das Redes Neurais Artificiais na identificação e no controle de processos. Destaca-se em especial a estrutura para o controle preditivo, sob a qual desenvolve-se e realiza-se o algoritmo de controle NPC (*Neural Predictive Control*). Utiliza-se uma planta não linear para estudo e verificação de operação do NPC. Estabelece-se critérios comparativos de desempenho referenciando-se nos resultados obtidos com o controlador GPC (*Generalized Predictive Control*).

Baseando-se nos resultados práticos e nas características da planta à ser controlada, propõe-se uma adaptabilidade para a ponderação do controle em ambos os controladores, destacando-se a adaptabilidade proposta como uma contribuição significativa para futuros objetos de pesquisa.

Abstract

This job presents a study on Artificial Intelligence, specially basing on the Artificial Neural Nets for control, applied in nonlinear dynamic systems. Besides a general approach to some Artificial Intelligence techniques, it makes a historical revision of the Artificial Neural Nets and given a wide vision of several topology registered in bibliography. In a specific part, it makes an approach to Neural Nets based on the control theory approach.

In blocks diagram, several forms are mentioned to apply Neural Nets to identification and control processes. A especially approach is given to a general structure for predictive controllers and the controller algorithm is developed using this structure. A nonlinear plant is selected for study and apply the NPC (Neural Predictive Control). The results are compared with the GPC (Generalized Predictive Control) results.

Based on practical results and on plant characteristics, it is proposed an adaptability to ponder the control action , and it is used in both controllers. The proposed adaptability is an significant contribution for future research.

Dedicatória

Nos meus colegas, procuro experiências para compartilhar;

Nos meus professores, procuro orientação para ir além;

Nos meus amigos, procuro apoio e apoiar;

Naqueles que me criticam, procuro idéias novas para crescer;

Nos obstáculos do dia-a-dia, procuro a paz de espírito e sabedoria para lidar;

No meu filho, tenho a minha recompensa por viver;

Na minha esposa, nos meus pais e irmãs, não procuro nada. Encontro o amor e o incentivo para vencer e a compreensão para as minhas falhas. São eles à quem dedico este trabalho.

Agradecimentos

A realização deste trabalho não se daria sem a constante colaboração de todos os professores e funcionários que, direta ou indiretamente constituem o Departamento de Engenharia Elétrica da UFBA e mais especificamente àquele que tornou real o curso de Mestrado neste Departamento, o Prof. Dr Caiuby Alves da Costa.

Agradeço também a todos os meus professores e em particular ao meu orientador, Prof. Adhemar de Barros Fontes, pelo incentivo e apoio ao longo desta jornada. Agradeço também por ter certeza que sempre acreditaram em mim.

Abreviaturas

AGs	-	Algoritmos Genéticos
ARX	-	*AutoRegressive with eXogenous inputs*
ARMAX	-	*AutoRegressive with Moving Average and eXogenous inputs*
CE	-	Computação Evolucionária
DARPA	-	*Defense Advanced Research Projects Agency*
GPC	-	*Generalized Predictive Control*
IA	-	Inteligência Artificial
IFAC	-	*International Federation of Automatic Control*
MIMO	-	*Multiple Input Multiple Output*
MPC	-	*Model Predictive Control*
NARMAX	-	*Nonlinear ARMAX*
NPC	-	*Neural Predictive Control*
PID	-	Proporcional, Integral e Derivativo
RLS	-	*Recursive Least Square*
RNA	-	Redes Neurais Artificiais
RNAR	-	Redes Neurais Artificiais Recursivas
SISO	-	*Single Input Single Output*
TDNN	-	*Time Delay Neural Network*

Índice

Índice de Figuras

Capítulo 1

1. INTRODUÇÃO

O estudo sobre controle de processos dinâmicos não lineares têm sido objeto de diversos livros, artigos, publicações e muitos eventos para debate sobre o tema. De forma não menos importante, a Inteligência Artificial (IA) tem despertado interesse nas mais diversas áreas de aplicação. Este trabalho tem como objetivo aplicar as técnicas de IA na área de controle de processos, destacando-se a utilização de Redes Neurais Artificiais (RNA) no controle de sistemas dinâmicos não lineares.

A concepção de controle, mesmo que de forma intuitiva, faz parte do nosso cotidiano e tem as suas origens nos primórdios da humanidade. No entanto, a teoria de controle na sua forma atual, teve fundamentada as suas bases matemáticas apenas neste último século, abrangendo, inclusive, diversas áreas de especialização, das quais destacam-se o controle preditivo, controle adaptativo, controle robusto, controle multivariável, controle ótimo, entre outros.

Paralelamente ao estudo da teoria de controle, nas últimas décadas, o homem vem procurando inspiração na natureza para encontrar soluções de problemas complexos. A seleção natural, proposta por Charles Darwin, é um exemplo clássico de regras impostas pela natureza e que, inegavelmente, tem obtido sucesso ao longo de milhões de anos. Os resultados alcançados pela natureza vêm estimulando pesquisadores e cientistas a tentar reproduzir os seus métodos, criando inspiração para o campo da Inteligência Artificial.

As soluções de problemas científicos, em sua grande parte, podem ser generalizados como métodos de busca e otimização. O objetivo comum é encontrar uma melhor combinação de fatores que, conjunta ou isoladamente, propiciem uma solução adequada para o problema em questão. O *espaço de busca* é tecnicamente definido como sendo o conjunto de todas as combinações possíveis para os fatores que constituem o problema. Sem perda de generalidade, considerando-se apenas os problemas de maximização, dada uma função $f: R^n \rightarrow R$ e um espaço de busca $S \subseteq R^n$, o problema de otimização pode ser assim representado:

$$\text{maximizar } f(x) \mid x \in S \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

ou

$$\text{encontrar } x^* \mid f(x^*) \geq f(x), \forall x \in S \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

A função $f(.)$ pode ser derivável ou não, uni ou multidimensional, estacionária ou variante no tempo, assim como o espaço de busca *S* pode ser contínuo ou discreto, finito ou infinito, côncavo ou convexo. O problema é definido como multimodal quando há mais de um ponto de máximo, caracterizando, portanto, problemas que podem ser extremamente complexos, e para os quais são continuamente desenvolvidas técnicas na procura de soluções eficazes.

Entre as técnicas já desenvolvidas, os métodos analíticos de otimização baseiam-se no fato da função $f(.)$ ser conhecida e derivável ou que possa ser aproximada por uma função derivável, observando-se o grau de precisão desejado. Os métodos baseados em cálculo numérico são suficientes para a solução de problemas cujo espaço de busca é linear. Técnicas numéricas como a do *gradiente decrescente* ou *gradiente conjugado*, embora comumente utilizadas, apenas são capazes de encontrar ótimos locais, exigindo conhecimento do processo por parte do pesquisador, sendo geralmente ineficientes para a otimização de funções multimodais, como ilustrada na **Erro! Argumento de opção desconhecido.**.

Os métodos enumerativos de otimização varrem ponto a ponto todo o espaço de busca, identificando os pontos ótimos. Embora este procedimento, intuitivamente, pareça ser uma solução adequada, na prática é irrealizável, uma vez que a dimensão do espaço de busca normalmente é representado por um número muito grande ou mesmo infinito.

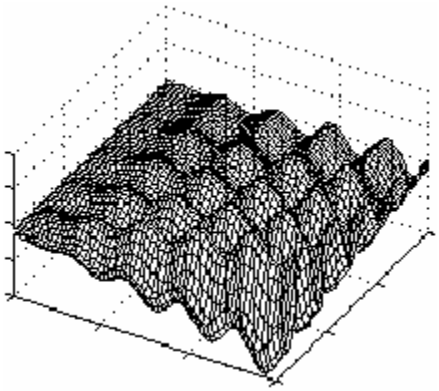

Figura **Erro! Argumento de opção desconhecido.**: Exemplo de uma função multimodal

Os métodos estatísticos como os Algoritmos Genéticos (AGs) empregam a idéia de busca probabilística dos pontos ótimos, dentro de regiões limitadas em um espaço de busca global. Um segundo exemplo muito claro desta aplicação é a técnica conhecida como *Simulated Annealing* a qual simula o resfriamento por etapas de ligas metálicas derretidas. A técnica baseia-se no fato de que, quando submetidas a altas temperaturas as moléculas da liga se comportam de modo aleatório, alternando entre estados de maior e menor energia continuamente. Sabe-se também que a medida que a temperatura baixa, cai a probabilidade das moléculas saltarem para outros níveis de energia. Assim, um processo de resfriamento bruto pode "congelar" as moléculas em estágios indesejáveis de energia, tornando a liga metálica frágil e quebradiça. Um resfriamento controlado, por sua vez, garante uma distribuição adequada das moléculas, caracterizando vantagens nas propriedades da liga. A aplicação da técnica para localização de mínimos globais é uma tentativa explícita de reproduzir procedimentos estatísticos já utilizados em outras áreas.

1.1 Motivação

Genericamente, a Inteligência Artificial apresenta estreitas relações com cálculos estatísticos, muito além das relações estabelecidas com os cálculos numéricos convencionais. Este fato ainda causa algumas resistências à aplicação destas técnicas, representadas por raros grupos de físicos e matemáticos.

Na engenharia, diferente da matemática, procura-se soluções adequadas dentro dos limites de um ambiente real. A solução matematicamente ideal nem sempre é realizável nos problemas reais, em que são considerados aspectos outros, dificilmente mensuráveis pelos modelos matemáticos, até então propostos. Assim, na busca de soluções adequadas e realizáveis, a engenharia encontra na IA uma excelente ferramenta de aplicação.

Por outro lado, o desenvolvimento dos microprocessadores e da sua respectiva capacidade de processamento, nos últimos 20 anos, vem tornando possível a utilização de microcomputadores nas mais diversas áreas de atuação. Na automação industrial, em específico, o microcomputador que normalmente é utilizado como equipamento de supervisão, apenas agora, em algumas empresas com maior interesse tecnológico, vem sendo utilizado para o controle de processos, caracterizando um verdadeiro desafio à ser vencido, especialmente pela indústria nacional.

O aumento da capacidade de processamento destes equipamentos também têm propiciado, a engenheiros de controle, o desenvolvimento e aplicação de algoritmos mais robustos e sofisticados, utilizando-se técnicas avançadas, convencionais e não convencionais para o controle de processos.

Assim sendo, a motivação deste trabalho é estudar um dos métodos de Inteligência Artificial, em específico as Redes Neurais Artificiais, buscando aplicá-las na área de controle de processos. Com esse objetivo, utiliza-se o ferramental hoje acessível através do uso de microcomputadores para a realização de algoritmos de controle mais avançados e procuram-se aplicar promissoras técnicas de controle não convencional baseadas na IA.

Desenvolve-se o trabalho em ambiente simulado numa plataforma PC, utilizando-se basicamente o *Simulink* e o *Matlab 5*, aplicativos desenvolvidos, dentre outros objetivos, para a simulação dinâmica de sistemas. Os algoritmos e conceitos aqui desenvolvidos são aplicáveis à sistemas reais em qualquer plataforma.

1.2 Estrutura do trabalho

No Capítulo 2 apresenta-se conceitos abrangentes da Inteligência Artificial, tecendo-se comentários gerais sobre alguns ramos da IA, em específico: os Algoritmos Genéticos (AGs), que representam uma das formas da Computação Evolucionária (CE); as Redes Neurais Artificiais (RNA); e a Lógica *Fuzzy*.

Em seguida, no Capítulo 3 detalha-se características mais específicas das RNA, enfocando-se alguns conceitos básicos, o histórico de aplicação das RNA e o seu renascimento. Ilustra-se a caracterização de uma RNA pela sua forma de aprendizagem e apresenta-se também as redes *feedforward* e *feedback*, a rede de *Kohonen* e redes com atrasos. Neste capítulo ainda aborda-se os conceitos de conjuntos de treinamento, validação e teste de uma RNA, sua forma de aprendizagem e um tratamento matemático sobre a regra delta, uma das mais populares regras de treinamento.

No Capítulo 4 cita-se as aplicações da RNA e reapresenta-se alguns conceitos sob o ponto de vista da teoria de controle. Adotando-se a notação aplicada à esta teoria, são apresentadas as equações características de uma RNA *feedforward*. Ilustra-se também, em diagramas de blocos, estruturas básicas para identificação e controle de sistemas baseados em RNA.

No Capítulo 5 procura-se dar ênfase ao controle preditivo, caracterizando de forma abrangente algumas funções de otimização e regras usualmente aplicadas à atualização da ação de controle. Seleciona-se um modelo de RNA padrão para aplicação e desenvolve-se as suas equações principais e de gradiente. Baseando-se na teoria de controladores preditivos e nas expressões algébricas da RNA, define-se uma função de otimização e uma

regra para atualização da ação de controle. Assim, desenvolve-se o algoritmo de controle preditivo NPC (*Neural Predictive Control*) para 1 e para T passos à frente.

No Capítulo 6 detalha-se uma planta não linear para aplicação e avaliação do desempenho dos controladores convencionais, através do GPC (*Generalized Predictive Control*) e não convencionais, através do NPC (*Neural Predictive Control*). Desenvolve-se as equações diferenciais que regem a dinâmica da planta e analisa-se a expressão geral da função de transferência em torno do ponto de equilíbrio. Detalha-se ainda um sensor, que é acrescentado ao sistema e ilustra-se um diagrama de blocos completo para simulação de operação da planta.

No Capítulo 7 define-se os parâmetros da RNA para modelagem da planta selecionada e destaca-se alguns aspectos para o treinamento rede proposta. Neste capítulo também traz-se os resultados obtidos com a simulação dos controladores estudados e aplicados à planta. Basicamente compara-se o desempenho do controlador NPC baseado na RNA proposta com o desempenho do GPC sob diversas condições de teste.

Finalmente, no último capítulo, apresenta-se algumas conclusões obtidas como fruto deste trabalho, assim como faz-se referências a continuidade dos trabalhos de pesquisa numa abordagem à perspectivas futuras.

Capítulo 2

2. INTELIGÊNCIA ARTIFICIAL

A aplicação de uma das formas de Inteligência Artificial na área de controle de processos, antes de mais nada nos leva a uma breve reflexão sobre a própria IA. O objetivo deste capítulo é conceituar aspectos gerais da IA, comentando-se sobre algumas formas de aplicação de técnicas de IA.

Tome-se por exemplo, um famoso jogo de imitação do qual participam 3 pessoas: um homem, uma mulher e um interrogador. Estando em um ambiente distinto, o interrogador realiza questões, à seu critério, que serão respondidas pelos outros dois participantes e, baseado nas suas respostas, este deve identificar quem é o homem e quem é a mulher.

"*Uma máquina pode pensar*?" . Buscando uma resposta a esta questão e baseado no jogo descrito, em 1950 Alan Turing propôs uma situação na qual uma máquina faria o papel do homem ou da mulher no jogo. Nesta proposta, a máquina seria considerada inteligente se o interrogador não conseguisse diferenciar as respostas da máquina das de um ser humano. Esta proposta é famosa e conhecida como teste de Turing.

Os dicionários de computação definem a Inteligência Artificial como:

> *"A aptidão ou capacidade de um dispositivo para desempenhar funções que são normalmente associadas à inteligência humana, tais como raciocínio, aprendizagem e auto-aperfeiçoamento".*

Assim, a IA não pode interpretada por uma técnica ou procedimento específico, mas sim, como uma série de técnicas distintas entre si, que contemplem um ou mais dos requisitos definidos. Sistemas que utilizam mais de uma linha de pesquisa da IA são usualmente definidos como Sistemas Híbridos.

As pesquisas sobre a IA nos últimos anos, têm se caracterizado pela inspiração em processos naturais, em que a própria natureza encontrou modelos complexos e adaptativos para problemas até então não modeláveis matematicamente, e com notável sucesso. Paradigmas da computação, tais como *Simulated Annealing*, Redes Neurais Artificiais, e Computação Evolucionária , são exemplos clássicos desta realidade.

2.1 Computação evolucionária - algoritmos genéticos

A Computação Evolucionária (CE) baseia-se na teoria da evolução de Darwin, assumindo-a como um processo adaptativo de otimização, sugerindo modelos em que populações de estruturas computacionais evoluem, resultando, em média, num melhor desempenho geral desta população com respeito a um dado problema. Metaforicamente, refletem a capacidade de adequação da população em relação ao ambiente, sobrevivendo e reproduzindo-se apenas indivíduos que foram capazes de se adaptar ao longo das gerações.

As técnicas de CE dão origem aos Algoritmos Evolucionários (AE), no entanto, a própria CE atualmente engloba um número crescente de métodos e formas de aplicação. Dentre outros destacam-se a Programação Evolucionária (PE), as Estratégias Evolucionárias (EEs) e os Algoritmos Genéticos (AGs), sendo este último, pela sua simplicidade de entendimento e realização, assim como pelo seu desempenho, uma das técnicas mais difundidas da CE.

Os AGs podem ser definidos como métodos computacionais de busca e otimização baseados nas formas de evolução natural e na genética. Assim, de uma forma probabilística e não determinística, os AGs tentam reproduzir a *inteligência* contida nestes modelos. Da genética, inclusive, é herdada a terminologia.

Dentre outras características, alguns aspectos básicos devem ser observados para desenvolvimento dos AGs que, diferentemente de outros métodos computacionais, não requerem o cálculo de derivadas ou fórmulas complexas. É importante destacar também que os AGs não operam no espaço de busca, mas sim, num espaço de soluções codificadas, assim como a solução não é definida por um indivíduo isolado, um ponto, mas sim por uma população, um conjunto de pontos.

O primeiro passo para a aplicação dos AGs em um problema qualquer, é codificar o espaço de busca, ou seja, cada possível solução $\boldsymbol{x}$ deve ser representada por uma sequência de símbolos $\boldsymbol{s}$ gerados a partir de um alfabeto $\boldsymbol{A}$ finito. Nos casos mais simples, é comum a utilização dos símbolos binários, sendo $\boldsymbol{A} = \{0, 1\}$. Assim, cada sequência $\boldsymbol{s}$ representa um indivíduo ou *cromossomo* e cada elemento de $\boldsymbol{s}$ é equivalente a um *gene*. Cada *gene* do indivíduo pode então assumir o valor de qualquer elemento do alfabeto $\boldsymbol{A}$, sendo assim denominados por *alelos*. Finalmente, a posição que um *gene* se encontra no *cromossomo* corresponde a um *locus gênico*, ou seja, corresponde ao índice dentro da sequência $\boldsymbol{s}$.

Definida uma população de indivíduos que representem pontos de partida dentro do espaço de busca, a solução para o problema proposto se dará através de iterações nas quais aplicam-se alguns conceitos de genética e resultados estatísticos. Cada iteração é definida como uma *geração*.

Os indivíduos das novas gerações[1] podem ser obtidos através de:

[1] Indivíduos bem adaptados podem sobreviver por várias gerações, caracterizando estratégias elitistas no processo de seleção.

- Recombinação: É a troca de fragmentos entre partes de cromossomos, emula o processo de *crossover*. É análoga à reprodução sexuada.

- Mutação: Representa a alteração no *alelo* do *gene* de forma aleatória.

Figura **Erro! Argumento de opção desconhecido.**: Processos de recombinação e mutação

Na maioria das aplicações, a população inicial de ***N*** indivíduos é gerada aleatoriamente ou através de processos heurísticos. Se na natureza é a variedade que caracteriza a evolução, sob o mesmo ponto de vista, é importante que a variedade da população inicial represente de forma adequada todo o espaço de busca. Uma população inicial indevidamente gerada, pode induzir a convergência da solução para um mínimo local ou, no mínimo, fazer aumentar em demasia o número de gerações necessárias à solução final.

De maneira similar a qualquer processo de otimização, quando da sua aplicação, devem ser definidas para os AGs as funções objetivo, também definidas como funções custo, que representarão a adequabilidade de cada indivíduo ao ambiente. Normalmente, quanto maior for o valor da função objetivo maiores são as chances deste indivíduo sobreviver e reproduzir-se no ambiente proposto e, pelos processos de recombinação e mutação, transferir as suas aptidões genéticas às novas gerações.

Na maioria dos AGs, por questões de simplicidade e facilidade de realização em computador, o número de indivíduos da população é geralmente fixo ou no mínimo finito. Assim, o surgimento de novos indivíduos torna necessária a eliminação de outros. O processo de seleção baseia-se no resultado da função objetivo, representando a adequabilidade de cada indivíduo, eliminando, estatisticamente, aqueles que menos se adaptam, numa típica aplicação da *lei dos mais fortes*.

Controladas as taxas de recombinação e mutação[2], após algumas gerações, a população de indivíduos converge para uma solução ótima. É observado que, neste procedimento, a solução não é representada por um único indivíduo mas, a princípio, por qualquer elemento da população.

2.2 As redes neurais artificiais

Do sucesso obtido pela natureza veio o desejo para produzir sistemas inteligentes artificiais. Nada mais estimulante que o estudo de sistemas sofisticados o suficiente para executar funções semelhantes as que o cérebro humano executa buscando, inclusive, aumentar nossa compreensão sobre este órgão. Assim, os modelos de Redes Neurais Artificiais baseiam-se nas cadeias biológicas de neurônios dos nossos cérebros.

O termo *Redes Neurais* pode induzir a compreensão de uma cadeia biológica cuja complexidade extrapola em muito os modelos matemáticos usualmente denominados de *Redes Neurais*. Assim, o termo adequado para descrição da ferramenta aqui estudada é *Rede Neural Artificial* (RNA) ou *Artificial Neural Network* (ANN), diferenciando-os dos modelos biológicos.

Pelo que se conhece (Tafner at all, 1995), o cérebro humano possui cerca de 10 bilhões de neurônios, caracterizando-se por serem as células humanas de maior complexidade estrutural e pelo fato de apenas desenvolverem-se durante o estágio embrionário da vida humana, permanecendo este número constante ao longo da vida do indivíduo, alterando apenas em volume e prolongamento de suas conexões com outras células.

[2] As taxas de recombinação e mutação são definidas pelo usuário, ainda não há, porém, regras definidas para a definição dos percentuais à serem aplicados.

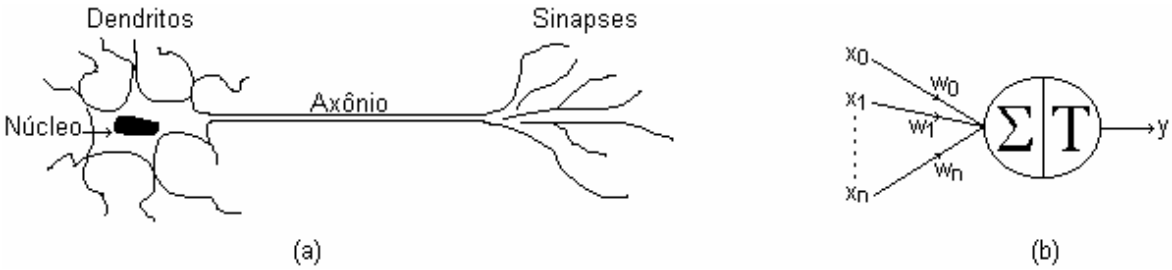

Figura **Erro! Argumento de opção desconhecido.**: Neurônio biológico (a) e artificial proposto por McCulloch e Pitts (b)

Um neurônio biológico, basicamente, capta os estímulos nas terminações nervosas denominadas de dendritos, e transmite para outras células através de canais denominados de axônios. O contato que ocorre entre dois neurônios é denominado de sinapse. Estas, caracterizam-se por serem polarizadas, ou seja, apenas um dos lados possui um neurotransmissor garantindo o fluxo unidirecional da informação neural.

Se o cérebro humano se limitasse a uma única sinapse, seria capaz de registrar apenas 2 estados, representados por número binários em {1,0}, ou ativo e inativo. Logo, duas sinapses poderiam representar 4 estados {00,01,10,11} e assim sucessivamente. Pelo que se sabe (Tafner at all, 1995), o cérebro humano possui 10^{10} neurônios, e cada um destes com a capacidade de criar até 10^4 sinapses. Fato que reforça a idéia de complexidade do cérebro humano.

Utilizando-se de técnicas convencionais, escrever um programa que realize uma ação diante de uma situação proposta, admite que todas as situações são conhecidas. Assume-se também que, conhecida a situação, de forma fundamentada, é possível definir a ação desejada dentro de um conjunto de ações possíveis.

Uma das características destacáveis do cérebro humano é a capacidade de aprendizagem. Diariamente somos expostos a novas situações às quais jamais nos haviam sido apresentadas e, diferente de um programa lógico combinacional, baseado no conhecimento sinapticamente acumulado, será tomada uma ação, possivelmente tão inédita quanto a própria situação proposta.

No equivalente matemático do neurônio biológico, ver **Erro! Argumento de opção desconhecido.**b, as entradas x_0 .. x_n representam os dendritos que prolongam-se do corpo da célula para outros neurônios onde receberão sinais através de um ponto de conexão denominado de sinapse. Os w_0 .. w_n representam os pesos sinápticos das respectivas entradas. As entradas são conduzidas para o corpo da célula, algumas com ação de excitação outras com ação inibidora para o disparo ou também definido como ativação da célula. Considerando-se o saldo entre os impulsos excitatórios e inibidores, ao exceder a um limiar, a célula dispara, enviando um sinal para outras células através do axônio. A **Erro! Argumento de opção desconhecido.** define a função de saída do neurônio proposto, na qual w_i representa o peso da conexão sináptica associado a entrada x_i, e θ_i é o limiar de ativação do neurônio.

$$y(t) = \begin{cases} 1 & se \quad \sum_{i=1}^{n} w_i x_i - \theta_i \geq 0 \\ 0 & se \quad \sum_{i=1}^{n} w_i x_i - \theta_i < 0 \end{cases}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

O modelo foi proposto por McCulloch e Pitts (McCulloch and Pitts, 1943). Em 1957 foi criado por Frank Rosenblatt o *Perceptron*, base atual das RNA dando origem a vários outros modelos, nos quais, a saída não limita-se aos valores binários 0 e 1.

A função de ativação, representada em **T** na **Erro! Argumento de opção desconhecido.**b, processa o resultado obtido pela função de somatório, transformando-a para a geração do sinal de saída. Funções de ativação atuam nas saídas das camadas escondidas de uma RNA, introduzindo as não linearidades ao sistema. Sendo a composição de funções lineares ainda uma função linear, a introdução de não linearidades aumenta a capacidade de representação das RNA, o que virá representar uma das suas características de maior destaque.

A maioria das funções não lineares poderiam ser aplicadas, porém, será visto adiante que uma das regras mais comuns para o aprendizado, baseado na retropropagação do erro e definida como *backpropagation*, assume que as funções devem ser diferenciáveis. Assim,

funções sigmóides como exponenciais, tangente hiperbólica ou mesmo a função identidade são comumente utilizadas.

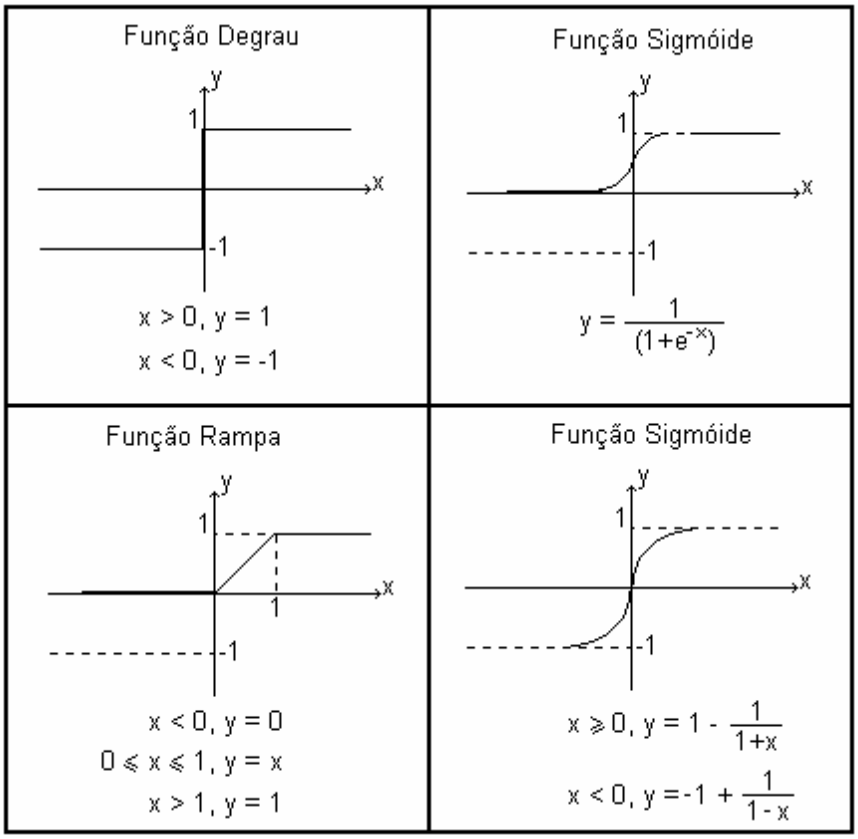

Figura **Erro! Argumento de opção desconhecido.**: Algumas funções de ativação usuais

Para os neurônios de saída, deve-se escolher uma função de ativação definida de acordo com a característica dos valores de saída do sistema. Funções de ativação limitadas são particularmente úteis quando os valores de saída também têm um alcance limitado. Geralmente, é comum utilizar-se a função identidade como função de ativação apenas dos neurônios que pertencem a camada de saída.

A conexão de diversos neurônios, de tipos iguais ou distintos, forma uma Rede Neural Artificial.

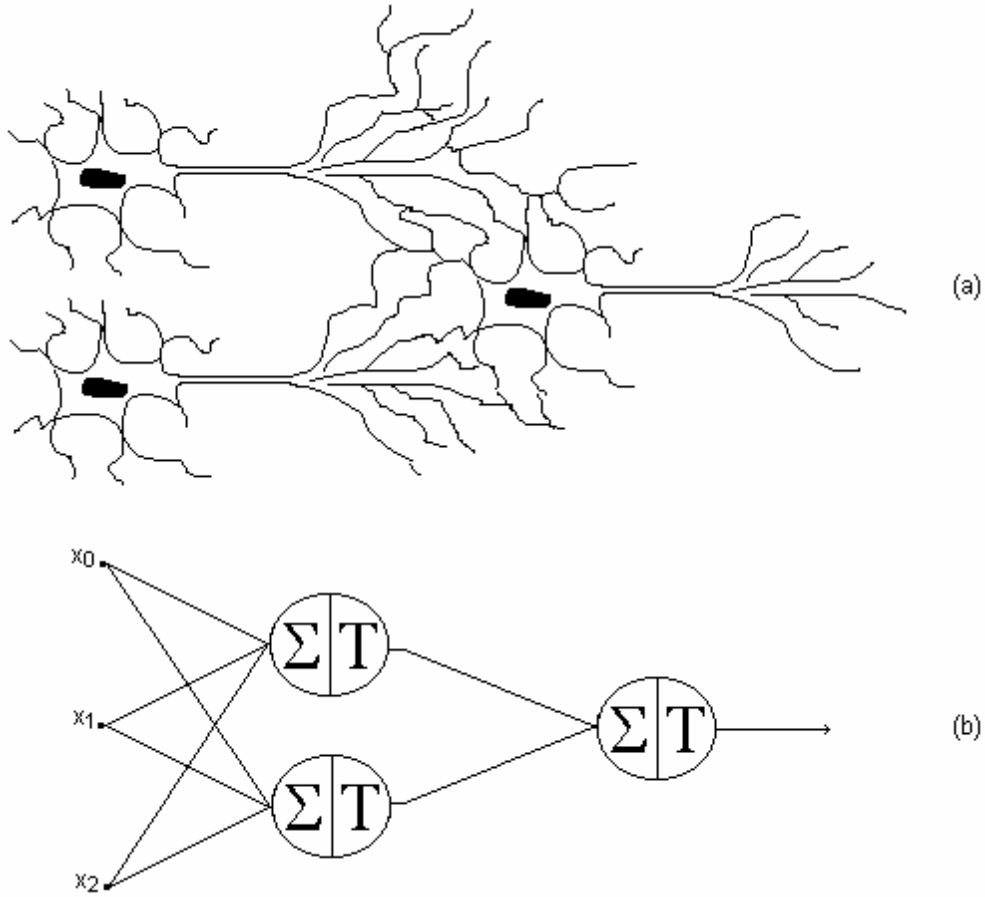

Figura **Erro! Argumento de opção desconhecido.**: Redes neurais biológicas (a) e artificiais (b)

Camadas escondidas de neurônios, assim como, neurônios de saída, normalmente apresentam um valor de polarização definido como *bias*, ver **Erro! Argumento de opção desconhecido.**, que é um valor constante à ser adicionado no somatório das entradas de cada neurônio. O termo *bias* pode ser tratado como um peso de conexão para um valor de entrada constante e igual a um. Assim, no processo de aprendizagem, o *bias* pode simplesmente ser tratado como qualquer outro peso.

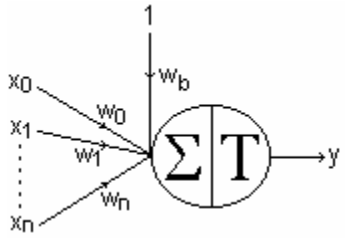

Figura **Erro! Argumento de opção desconhecido.**: Neurônio artificial com bias

É importante destacar que a nomenclatura para definição do número de camadas de uma RNA é diferente em diversas referências bibliográficas. Alguns autores não consideram as camadas de entrada/saída da RNA ou simplesmente não consideram a camada de entrada, por não ser esta associada a neurônios. Neste documento, a nomenclatura baseia-se na identificação mais usualmente encontrada na bibliografia consultada, e que referencia-se à camada de entrada como uma camada da RNA. Assim, o exemplo da RNA ilustrado na **Erro! Argumento de opção desconhecido.**b representa uma RNA de 3 camadas: uma camada de entrada, uma camada de saída e uma camada intermediária também definida como camada escondida..

Quanto ao aprendizado de uma RNA, este baseia-se em *regras de treinamento* por meio da qual são ajustados os pesos sinápticos. Assim, as RNA aprendem através de exemplos - como crianças aprendem reconhecer cachorros através de exemplos de cachorros - exibindo, inclusive, uma capacidade de generalização que extrapola os dados de treinamento - sem conhecer todas as raças de cachorro, a criança ainda os identifica como cachorros.

Em princípio uma RNA pode representar qualquer função real, ou seja, pode fazer tudo o que um computador digital normal pode fazer. Na prática, RNA são especialmente úteis para classificação e problemas de aproximação e mapeamento de funções, com especial destaque nos casos para os quais regras matemáticas já desenvolvidas não podem ser aplicadas. Genericamente, o mapeamento entre quaisquer espaços vetoriais pode ser aproximado, com uma precisão definida, através de uma RNA.

2.3 Lógica Fuzzy

A tradução mais próxima para a Lógica *Fuzzy* é Lógica Nebulosa e certamente não é a expressão que melhor define o conceito. Como *Fuzzy* é o termo que foi aceito pela comunidade acadêmica/científica, será a expressão citada ao longo desta dissertação.

A Lógica *Fuzzy* foi primeiramente proposta por Lotfi A. Zadeh na Universidade da California, Berkeley, em 1965. Na década de 80, o Japão já aplicava técnica *Fuzzy* em ambiente industrial, o que despertou o interesse de cientistas e pesquisadores americanos. Em 1987 ganhou adeptos quando, num evento internacional em Tókio-Japão, o pesquisador Takeshi Yamakawa apresentou uma solução *Fuzzy* para o problema do pêndulo invertido, um clássico da área de controle.

Assim como as Redes Neurais Artificiais, a Lógica *Fuzzy* vem ampliando a sua área de aplicação, dentro destas, destaca-se a área de controle de processos. Ainda não se fala em *controladores genéticos*, baseados nos AGs, porém, tão comum quanto encontrar controladores baseados em RNA, é encontrar controladores *Fuzzy*, motivo inclusive pelo qual caracteriza-se como um tema de grande interesse desta dissertação. A Lógica *Fuzzy*, em resumo, busca tratar processos digitais como analógicos dando *pesos* às *verdades* digitais.

Para exemplificação, considera-se um sistema de freio que toma decisões baseado na temperatura, velocidade e aceleração entre outras variáveis do sistema. A temperatura neste sistema, por exemplo, pode ser dividida em faixas de estados: *frio*, *fresco*, *natural*, *morno*, *quente*, *muito quente*. A transição de um estado para o próximo, porém, é difícil de ser definida. A **Erro! Argumento de opção desconhecido.** ilustra o exemplo.

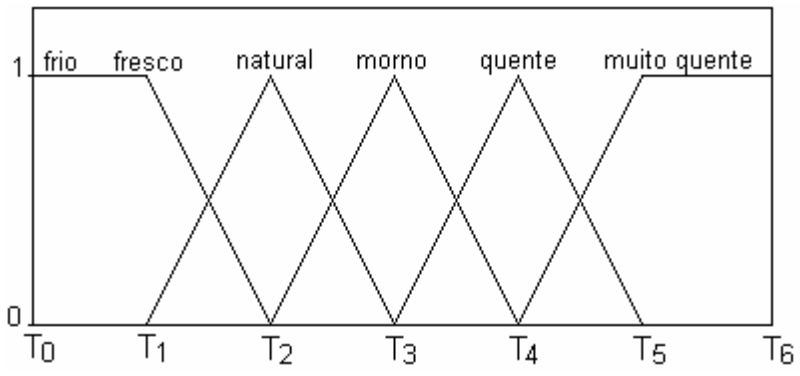

Figura **Erro! Argumento de opção desconhecido.**: Exemplo Fuzzy para temperaturas

Pode-se fixar um limite arbitrário para dividir *morno* de *quente*, mas isto resultaria em uma descontinuidade no momento em que o valor ultrapassa o limite entre os estados.

A Lógica *Fuzzy* procura eliminar estas descontinuidades criando estados *Fuzzy*, ou nebulosos, permitindo uma mudança gradual e contínua entre estados. É definida uma *função de sociedade* que resulta em valores decrescentes na faixa à qual o valor da variável pertence e valores crescentes na faixa para a qual o valor da variável se desloca. Este procedimento permite que, a qualquer instante, o valor da temperatura possa pertencer à mais de uma classe (*morno* e *quente*), com pesos distintos.

As variáveis de entrada em um sistema de controle *Fuzzy* são normalmente mapeadas por um conjunto *funções de sociedade*. As *funções de sociedade* usualmente tem formas triangulares, trapezoidais, ou outras, no entanto, na prática, o número de faixas tem se mostrado mais relevante que a especificidade da função utilizada.

Mapeando as variáveis de entrada com *funções de sociedade*, o sistema está apto ao exercício do controle: “SE a temperatura do freio está morna, a velocidade é constante e lenta, ENTÃO diminua levemente a pressão do freio”. Caracterizando-se a temperatura, a velocidade e a aceleração como variáveis de entrada e a pressão do freio como variável de saída.

Há também ainda muita confusão entre a Lógica *Fuzzy* e uma simples aplicação de regras lógicas, no entanto, é importante destacar que a decisão *Fuzzy* não é baseada numa regra, mas num conjunto de regras dando *pesos* às *verdades*.

Há vários modos diferentes para definir o resultado de uma regra, mas um dos mais comuns e mais simples é o denominado *max-min*, que é um método de conclusão no qual a saída da *função de sociedade* determina o *peso* da afirmação baseando-se nas regras definidas. Regras podem ser aplicadas numa forma sequêncial, por software, ou paralela, por hardware, e há diversas teorias sobre vantagens/desvantagens de cada um destes métodos.

Assim, os controladores Fuzzy são conceitualmente muito simples, constituídos basicamente de três estágios:

a) Entrada: mapeia as entradas segundo *funções de sociedade* definidas.
b) Processo: aplica conjunto de regras definidas gerando resultados específicos para cada uma destas, permitindo, em seguida, uma combinação entre resultados de várias regras.
c) Saída: converte o resultado obtido para um sinal específico de atuação/controle.

O processo, segundo estágio da Lógica *Fuzzy*, normalmente baseia-se em regras na forma SE-ENTÃO. Um sistema completo deve conter um número significativo de regras de forma a contemplar todo o espaço de análise. É importante considerar também que numa simples regra do tipo: "SE X é alto ENTÃO Y é baixo", o valor de X não tem influência definitiva no valor de Y, todas as outras regras devem ser avaliadas para uma conclusão final do valor de Y.

Há ainda casos em que podem-se criar adjetivos como: *perto*, *longe*, *pouco*, *muito*, *exagerado*, e outros. Nestes casos, é importante definir o *peso* destas medidas, como *muito* pode representar elevar ao quadrado e *exagerado* pode representar elevar ao cubo.

Capítulo 3

3. ESTUDO DAS REDES NEURAIS ARTIFICIAIS

Neste capítulo aborda-se com mais detalhes alguns conceitos e características específicas das RNA, contemplando um breve histórico e uma descrição geral de estruturas das RNA e formas de treinamento. São apresentadas redes *feedforward* e *feedback* e dado um destaque específico para as redes de Kohonen e redes com retardos. No final deste capítulo faz-se uma breve abordagem matemática sobre a regra delta, um dos mais difundidos algoritmos para treinamento das RNA.

3.1 Definição de uma RNA

Por ser uma ferramenta matemática recente e ainda em desenvolvimento, não há definição universal aceita de uma RNA. Sem uma definição formal, porém, converge-se para a afirmativa que uma RNA é uma cadeia de muitas unidades de processadores simples, na qual, cada um contém uma pequena quantia de memória local. As unidades são conectadas através de canais de comunicação que transportam dados numéricos. Cada unidade só opera os dados locais e os dados recebidos pelos canais de comunicação. Por tratar-se de processadores independentes, as RNA de uma forma geral apresentam grande potencial para paralelismo.

A definição de uma RNA ainda caracteriza uma diversidade de opiniões, e é de consenso que ainda não há uma que agrade a todos, motivo pelo qual muitos livros didáticos envolvendo RNA ainda não a definam explicitamente.

Segundo Haykin (1994 - pg.02):

> *"Uma rede neural é um processador distribuído, maciçamente paralelo, que tem uma propensão natural por armazenar conhecimento experimental, tornando-o disponível para uso. Se assemelha ao cérebro em dois aspectos:*
>
> *I. O conhecimento é adquirido pela rede por um processo de aprendizagem.*
>
> *II. São usadas conexões entre os neurônios, forças conhecidas como pesos de sinapses, para armazenar o conhecimento."*

Segundo Nigrin (1993 - pg.11):

> *"Uma rede neural é um circuito composto de um número muito grande de elementos de processo simples que são baseados numa estrutura neural. Cada elemento só opera em informação local e de forma assíncrona. Assim, não há nenhum relógio de sistema global."*

E de acordo com Zurada (1992 - pg. xv):

> *"Sistemas de redes neurais artificiais são sistemas celulares físicos que podem adquirir, armazenar, e utilizar o conhecimento experimental."*

Apesar das aparências, há considerável superposição entre o estudo das RNA e a estatística. Enquanto a estatística preocupa-se com análise de dados, numa terminologia aplicável pelas RNA, conclusão estatística significa aprender a generalizar a partir de conjuntos de dados contaminados com ruído. Evidentemente algumas RNA não se preocupam com análise de dados, a exemplo das que pretendem modelar sistemas biológicos, nestes casos, pouco tendo à ver com a estatística. Mas a maioria das RNA que

podem aprender a generalizar a partir de dados ruidosos, tem comportamento similar aos métodos estatísticos. Diversos métodos comumente utilizados para ajuste de modelos não lineares, como Levenberg-Marquardt e algoritmos de gradiente conjugados, podem agora ser utilizados no treinamento das RNA.

3.2 O perceptron e a função XOR

Quando da proposta do Perceptron em 1957 por Rosenblatt, e dada a sua capacidade de solução adaptativa de problemas, cresceu vertiginosamente o interesse de cientistas e pesquisadores, iniciando-se uma série de trabalhos na área da Inteligência Artificial com base nas RNA. Posteriormente a este fato, Marvin Minsky e Seymor Papert, dois respeitáveis cientistas da época, publicaram um livro intitulado *Perceptron*, no qual demonstravam as limitações dos perceptrons propostos por Rosenblatt, provando que, uma RNA com uma camada perceptron, não seria capaz de resolver um problema XOR, convencendo a maioria dos pesquisadores da incapacidade desta RNA de resolver problemas mais complexos.

Felizmente, nem todos os pesquisadores foram atingidos por esta teoria, apesar de que, durante longos anos, nada foi publicado na área. Apenas em 1982 com a apresentação de um trabalho para a *National Academy of Sciences*, por John Hopfield, renasce o interesse pelas RNA.

Matematicamente, diz-se que um neurônio perceptron só esta apto a resolver problemas linearmente separáveis. A função XOR é representada na **Erro! Argumento de opção desconhecido.**, na qual x_0 e x_1 representam as suas entradas e y a sua saída.

E uma RNA com duas entradas, uma saída e uma única camada perceptron, é representada na **Erro! Argumento de opção desconhecido.**.

x_0	x_1	y
0	0	0
0	1	1
1	0	1
1	1	0

Tabela **Erro! Argumento de opção desconhecido.**: Função XOR

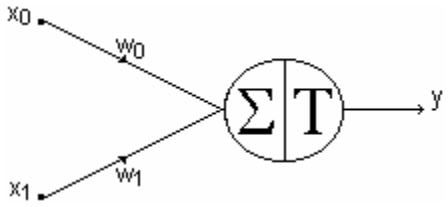

Figura **Erro! Argumento de opção desconhecido.**: RNA com 2 entradas, 1 saída e uma única camada perceptron

Mapeando todas as combinações possíveis de x_0 e x_1, percebe-se que não há valores para w_0 e w_1 que definam uma reta que possa agrupar as saídas desejadas. O que ratifica a afirmação destas não serem linearmente separáveis, confirmando o fato desta rede não conseguir resolver o problema da função XOR.

Uma forma clássica para a resolução deste problema é admitir uma rede neural perceptron com mais de uma camada, conforme ilustrado na **Erro! Argumento de opção desconhecido.**. Neste caso, assume-se que cada neurônio tem o limite de 1, ou seja, o neurônio disparará apenas se a soma das entradas totalizar valores iguais ou superiores a 1.

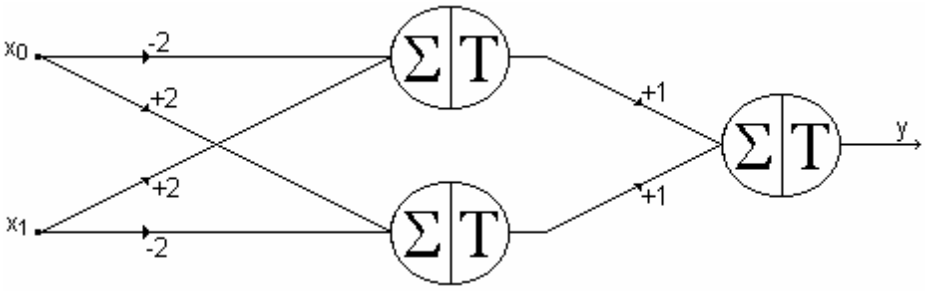

Figura **Erro! Argumento de opção desconhecido.**: RNA com 2 entradas, 1 saída e 2 camadas perceptron

Admitindo-se os pesos já treinados conforme **Erro! Argumento de opção desconhecido.** e, executando-se a rede então proposta, obtém-se a tabela do XOR.

3.3 Caracterização de uma RNA pela forma de aprendizado

Retomadas as pesquisas na área, diversas formas de RNA já foram propostas, não se podendo afirmar um número definido, assim como, sabe-se que o tema ainda está distante de um conceito final. Nos diversos modelos propostos, doais aspectos básicos caracterizam as redes: sua forma de aprendizagem, supervisionada ou não; sua topologia, com ou sem realimentação. Assim, a aprendizagem de uma RNA é intimamente relacionada com a respectiva topologia da rede.

3.3.1 Aprendizado supervisionado

Nesta forma de aprendizagem há um conjunto de referências que norteiam a RNA como proceder ou que simplesmente indicam qual deveria ser o comportamento esperado. Nesta fase, serão apresentados à rede diversos padrões de pares de entrada/saída conhecidos, usualmente definidos como conjunto de treinamento.

Assim, sempre que fornecermos um vetor de entrada conhecido, a avaliação do erro cometido entre a saída real e a saída teórica esperada, propicia à rede, se necessário, um procedimento para o seu treinamento. O treinamento baseia-se em algoritmos para ajuste dos pesos sinápticos da conexão entre os neurônios e normalmente é encerrado quando para todos os pares de entrada/saída do conjunto de treinamento o erro é menor que uma tolerância definida.

Segundo (IFAC, 1998), são exemplos de topologias de rede para aprendizado supervisionado:

1. Redes sem realimentação:

a)Perceptron
b)Adaline, Madaline

c)Backpropagation (BP)
d)Cauchy Machine (CM)
e)Adaptive Heuristic Critic (AHC)
f)Time Delay Neural Network (TDNN)
g)Associative Reward Penalty (ARP)
h)Avalanche Matched Filter (AMF)
i)Backpercolation (Perc)
j)Artmap
k)Adaptive Logic Network (ALN)
l)Cascade Correlation (CasCor)
m)Extended Kalman Filter (EKF)
n)Learning Vector Quantization (LVQ)
o)Probabilistic Neural Network(PNN)
p)General Regression Neural Net (GRNN)

2) Redes com Realimentação:

a)Brain-State-in-a-Box (BSB)
b)Fuzzy Congitive Map (FCM)
c)Boltzmann Machine (BM)
d)Mean Field Annealing (MFT)
e)Recurrent Cascade Correlation (RCC)
f)Backpropagation through time (BPTT)
g)Real-time recurrent learning (RTRL)
h)Recurrent Extended Kalman Filter(EKF)

3.3.2 Aprendizado não supervisionado

Num aprendizado não supervisionado, o processo de aprendizagem da RNA é autônomo. O sistema ignora os valores de saída para o treinamento, baseando-se nos dados de entrada passados e nos erros cometidos, tomando-os como referência para os passos seguintes. A rede busca caracterizar a entrada atual, num processo de classificação, em cima de critérios desenvolvidos pela aprendizagem das entradas passadas. A auto-organização de uma forma geral, envolve um processo de competição entre os neurônios, basicamente utilizando-os como neurônios classificadores. Dentre as aplicações mais usuais destaca-se o reconhecimento de padrões.

Segundo (IFAC, 1998), são exemplos de topologias de rede para aprendizado não supervisionado:

1) Redes sem Realimentação:

a)Learning Matrix (LM)
b)Driver-Reinforcement Learning (DR)
c)Linear Associative Memory (LAM)
d)Counterprogation (CPN)
e)Fuzzy Associative Memory (FAM)
f)Optimal Linear Associative Memory (OLAM)
g)Sparse Distributed Associative Memory (SDM)

2) Redes com Realimentação:

a)Additive Grossberg (AG)
b)Shunting Grossberg (SG)
c)Discrete Hopfield (DH)
d)Continuous Hopfield (CH)
e)Competitive learning
f)Kohonen Self-organizing Map/ Topology-preserving map (SOM/TPM)
g)Discrete Bidirectional Associative Memory (BAM)
h)Temporal Associative Memory (TAM)
i)Adaptive Bidirectional Associative Memory (ABAM)
j)Binary Adaptive Resonance Theory (ART1)
k)Analog Adaptive Resonance Theory (ART2, ART2a)

Evidentemente, não faz parte da proposta deste trabalho a análise e compreensão de cada modelo de RNA aqui apresentado, mesmo porque, de uma forma geral, a aplicação de cada topologia busca soluções adequadas para determinados tipos de problemas, não sendo aplicáveis em diversos outros casos. Procura-se então compreender basicamente duas topologias clássicas de RNA, aplicáveis de forma genérica ao reconhecimento de padrões e ao mapeamento de funções não lineares, nesta segunda, dando-se ênfase a aplicabilidade em controle de processos dinâmicos não lineares.

3.4 As redes de Kohonen

Por ter proposto uma variedade de topologias de RNA, Teuvo Kohonen é um dos cientistas mais famosos em neurocomputação. Assim, uma rede de Kohonen (Kohonen, 1989), é a nomenclatura aplicada para algumas estruturas de rede, todas caracterizadas pela competitividade, ou seja, pelo fato de apenas uma das saídas estar habilitada por vez. Entre outras aplicações, é utilizada com respeitável desempenho no reconhecimento de padrões. Estes modelos baseiam-se na idéia de que o cérebro humano, apesar da interconexão dos neurônios biológicos, apresenta regiões de maior atividade cerebral em função da atividade simultaneamente exercida.

Genericamente, uma rede de Kohonen é uma RNA tipo *feedforward* de treinamento não supervisionado e admite apenas duas camadas, basicamente as camadas de entrada e saída, não caracterizando-se limites para o número de entradas/saídas. Nesta topologia, cada neurônio representa uma saída, sendo indiferente referenciar-se ao número de neurônios ou ao número de saídas desta RNA. Um aspecto importante é que, neste caso, a rede é *fully connected* ou amplamente conectada, ou seja, todos os canais de entrada são conectados à todas as saídas.

Na inicialização de uma rede de Kohonen, os pesos sinápticos são geralmente aleatórios mas de amplitude baixa. Quando um primeiro sinal de entrada carregando uma informação qualquer é apresentado à rede, entre ***N*** saídas da rede, uma única deve ser ativada, é a denominada vencedora; assim, apenas esta saída será treinada, tendo seus pesos sinápticos ajustados e, em alguns casos, também as saídas adjacentes. Neste procedimento procura-se deixar clara a predominância da saída vencedora para as condições da entrada proposta. É considerada a saída vencedora aquela que apresentar a menor distância euclidiana do valor de saída. E, por definição, a distância euclidiana é dada por:

$$d_j = \sum_{i=1}^{n} \left(w_{i,j} . x_i \right)^2 \qquad \text{(Eq. Erro! Argumento de opção descon}$$

hecido.)

em que: N = número de neurônios de saída

n = número de entradas na rede

j = indexador do neurônio de saída (1..N)

i = indexador da entrada (1..n)

$w_{i,j}$ = peso sináptico que conecta a entrada i ao neurônio j

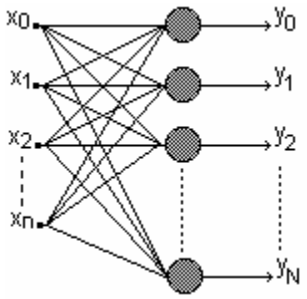

Figura **Erro! Argumento de opção desconhecido.**: Típica rede de Kohonen

Considera-se que a rede está treinada quando já lhe foi apresentado todo o conjunto de treinamento, e atingidos os critérios de erro definidos. De forma similar a qualquer outra RNA, uma fase de testes precede o treinamento. Nesta segunda fase, não há mais ajustes dos pesos sinápticos e entradas de caráter inédito devem ser apresentadas à rede. A rede é considerada como adequadamente treinada quando os resultados obtidos através dos testes atendem um critério de validação, como, por exemplo, o erro quadrático médio de todo o conjunto ser menor que uma tolerância definida.

3.5 Redes feedforward e feedback

Usualmente, as RNA apresentam a característica de *fully connected*, ou seja, todos os neurônios de determinada camada têm suas saídas conectadas a todos os neurônios da camada seguinte. Na sua forma padrão, ver **Erro! Argumento de opção desconhecido.**a, define-se esta RNA como uma rede *feedforward*, ou seja, o sinal é propagado entre camadas sempre no sentido progressivo.

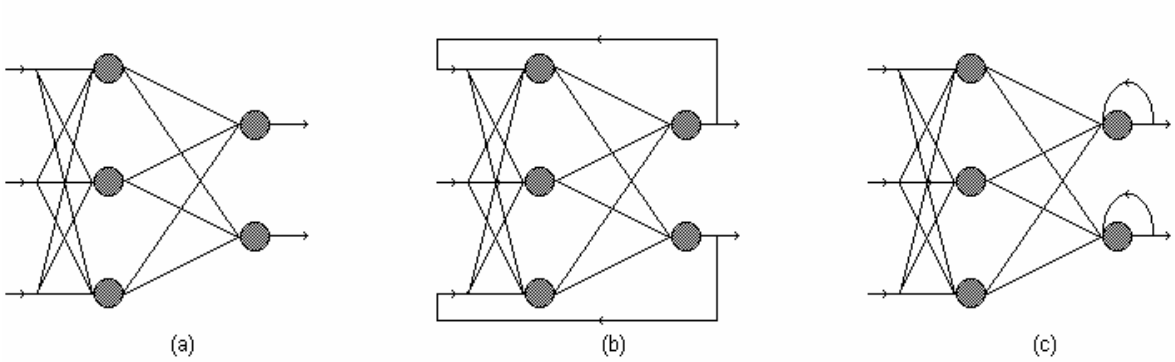

Figura **Erro! Argumento de opção desconhecido.**: Redes feedforward e feedback

Uma RNA que possua camadas em que a saída de um neurônio é entrada para neurônios de camadas anteriores, **Erro! Argumento de opção desconhecido.**b, ou mesmo para o próprio neurônio, **Erro! Argumento de opção desconhecido.**c, são definidas como redes *feedback*, ou seja, realimentadas.

RNA Recursivas (RNAR), ou Recursive Artificial Neural Networks (RANN), aumentam a área de atuação das RNA, nas quais, diferentemente de um simples sistema combinacional, a saída de um neurônio é função dos dados atuais de entrada e dos valores passados de entrada/saída, o que torna a rede capaz de modelar sistemas dinâmicos, lineares ou não, tema principal desta dissertação.

3.6 RNA com atrasos

A bibliografia é não é única quanto a definição da recursividade ou não de uma RNA na qual o vetor de entrada á composto por valores passados da própria entrada e saída da rede. Por definição, esta topologia é denominada de TDNN (*Time Delay Neural Network*) e uma típica estrutura de uma TDNN é ilustrada na **Erro! Argumento de opção desconhecido.**. Nesta dissertação, admite-se a TDNN como uma RNA *feedforward*.

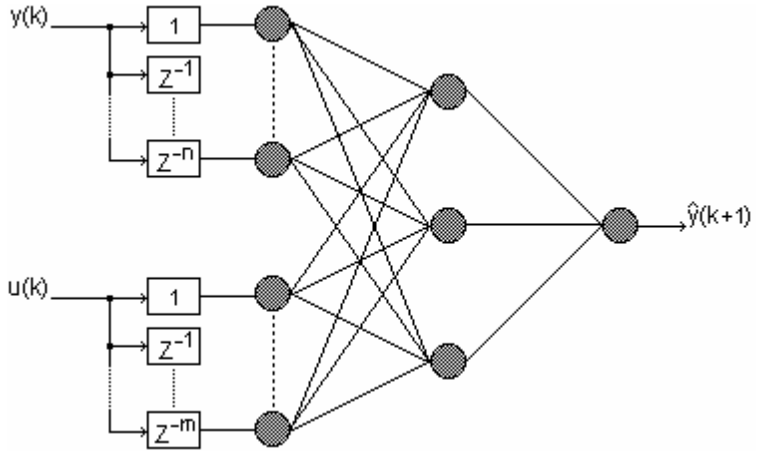

Figura **Erro! Argumento de opção desconhecido.**: Estrutura de uma TDNN

A recursividade e/ou a consideração de valores passados de entrada/saída, aumentam a capacidade de aprendizagem das RNA, assim como, possibilitam capturar a dinâmica do sistema. Assim, as RNAR e as TDNN caracterizam-se por serem ferramentas adequadas para aplicações práticas em controle de sistemas dinâmicos.

3.7 Conjuntos de treinamento, validação e teste

Para o treinamento de uma rede no modo supervisionado, admite-se conhecimento prévio de conjuntos de entrada/saída do sistema à ser modelado pela RNA. Em estudos nos quais o modelo matemático de um sistema é conhecido, a geração de dados simulados para treinamento é simples. Neste caso, porém, sendo conhecida a função, a aplicação da IA, de uma forma geral, não necessariamente caracterizará a melhor solução, uma vez já desenvolvido o ferramental teórico para solução do sistema já modelado. Nas aplicações práticas, de uma forma geral, os dados são levantados diretamente do campo.

A metodologia usualmente aplicada para aprendizagem das RNA é a divisão do conjunto de amostras em conjuntos de treinamento e validação. O treinamento corresponde a fase de ajuste dos pesos sinápticos, mediante apresentação dos seus pares de entrada/saída, ao passo em que o procedimento de validação, mediante um sinal de entrada, apenas avalia o desempenho da saída real referenciando-se na saída esperada. Admitindo-se que os pares de entrada/saída do conjunto de validação jamais foram apresentados à rede na fase de

treinamento, a rede é válida se os erros obtidos nesta segunda fase forem inferiores a uma tolerância pré-definida. É importante observar que, na fase de validação, não há o ajuste dos pesos sinápticos.

Percebe-se assim que a diferenciação entre os conjuntos de teste e treinamento é de fundamental importância para a validação do modelo, no entanto, ainda não foram estabelecidos critérios que possam definir de forma unânime a classificação dos conjuntos.

Apesar da literatura referenciar-se apenas a conjuntos de treinamento e validação, alguns autores ainda são mais específicos, destacando-se Ripley (1996, pg. 354), com as definições:

a. Conjunto de Treinamento:

> "*Um conjunto de exemplos utilizado para aprendizagem, isto é, que propicie o ajuste correto dos parâmetros (pesos) do classificador.*"

b. Conjunto de Validação:

> "*Um conjunto de exemplos que faça o ajuste fino dos parâmetros de um classificador.*"

c. Conjunto de Teste:

> "*Um conjunto de exemplos utilizado apenas para avaliar o desempenho e robustez de um classificador completamente especificado.*"

Segundo Bishop (1995, pg. 372), é proposta a seguinte explicação:

> "*Considerando que nossa meta é achar a RNA que apresente o melhor desempenho com dados novos, a aproximação mais simples para a comparação de diferentes RNA é avaliar a função de erro utilizando-se de*

dados que não foram utilizados no treinamento. Diversas redes são treinadas pela minimização de uma função de erro apropriada referenciando-se a um conjunto de treinamento. O desempenho das RNA é comparado pela avaliação da função de erro quando utiliza-se um conjunto de validação independente, pré-fixado. Considera-se que este procedimento deve relevar as tendências do conjunto de treinamento/validação, assim, o desempenho da RNA selecionada deve ser ratificado avaliando-se o desempenho com base num terceiro conjunto, definido como conjunto de teste."

3.8 Aprendizagem da RNA

O treinamento de uma RNA é, na maioria dos casos, um exercício de otimização numérica e, normalmente, de uma função objetivo não linear. Há séculos, são estudados e desenvolvidos métodos de otimização não linear, resultando numa vasta literatura disponível em campos como análise numérica, pesquisas operacionais e computação estatística. Assim, não se pode definir um método único, ideal para a otimização não linear. O método é selecionado baseando-se nas características do problema à ser resolvido.

O diagrama de blocos da **Erro! Argumento de opção desconhecido.** caracteriza de forma generalizada um dos procedimentos mais comuns para o treinamento dos pesos sinápticos da RNA.

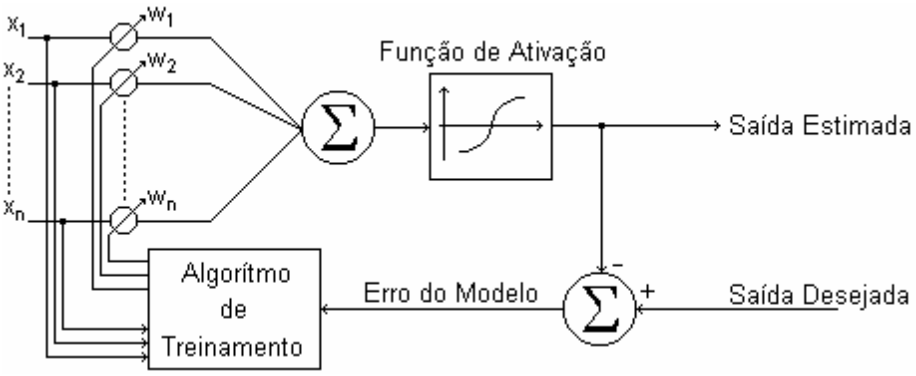

Figura **Erro! Argumento de opção desconhecido.**: Diagrama de blocos generalizado para treinamento dos pesos da RNA

Conforme (IFAC, 1998), soluções específicas podem ser generalizadas para dois grupos básicos de problemas:

a) Para funções de objetivo que não são continuamente diferenciáveis, os algoritmos de Nelder-Mead e Simulated Annealing caracterizam soluções adequadas.

b) Para funções de objetivo com segunda derivada contínua, são definidos três tipos gerais de algoritmos, caracterizando-se por resultados eficazes para a maioria dos casos práticos:

- *Para um número pequeno de pesos, são eficientes os algoritmos de Newton estabilizado, Gauss-Newton e Levenberg-Marquardt.*
- *Para um número moderado de pesos, variações do algoritmo de quasi-Newton são eficientes.*
- *Para um número grande de pesos, variações de algoritmos de conjugado gradiente são eficientes.*

Há inclusive sugestões no IFAC (1998) quanto ao desempenho usualmente superior do algoritmo de Levenberg-Marquardt (Press at all, 1990), (Grace, 1992). Fato que se observa nas aplicações práticas realizadas (Schnitman e Fontes, 1998).

Destaca-se ainda que os métodos propostos, apesar de caracterizarem a generalização para aplicabilidade de algoritmos definidos, representam procedimentos de busca de mínimos locais e nenhum deste assegura a localização de um mínimo global. Há uma variedade de combinações de métodos que possibilitam a otimização global, destacam-se, porém, aplicações de Inteligência Artificial como o *Simulated Annealing* e os Algoritmos Genéticos.

Um dos algoritmos mais difundidos para o treinamento das RNA é o algoritmo de *backpropagation* (Chauvim and Rumelhart, 1995). *Backpropagation* é a terminologia utilizada para especificar a retropropagação do erro. No sentido exato, *backpropagation* se

refere ao procedimento para calcular o gradiente do erro para uma rede *feedforward*. *Backpropagation standard*, nossa típica RNA. Na literatura, a retropropagação do erro de saída de uma RNA baseada no gradiente negativo é também conhecida como regra delta, algoritmo de treinamento popularizado por Rumelhart (Rumelhart and McClelland, 1986).

Admite-se um neurônio de uma RNA genérica, ilustrado na **Erro! Argumento de opção desconhecido.**a e o respectivo modelo reverso na **Erro! Argumento de opção desconhecido.**b.

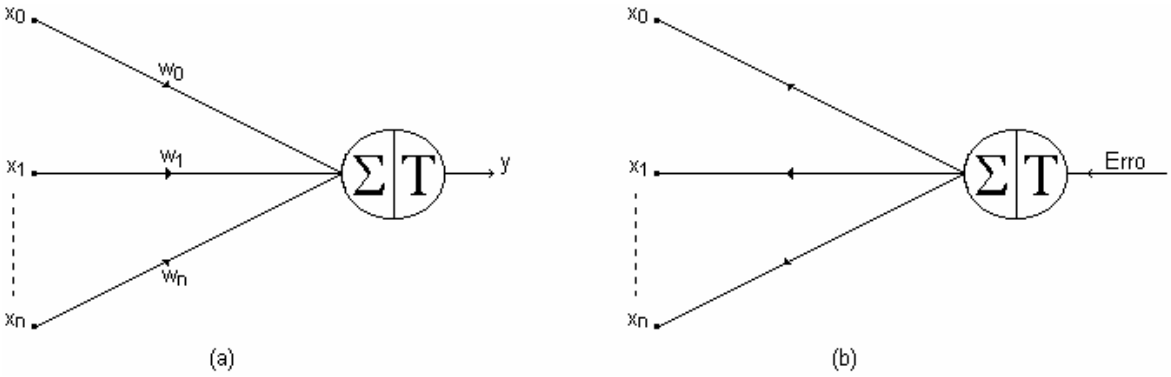

Figura **Erro! Argumento de opção desconhecido.**: RNA genérica (a) e modelo reverso (b) para dedução da regra delta

A expressão para atualização dos pesos sinápticos é dada por:

$$w = w + \Delta w \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que $w =$ vetor de pesos sinápticos,

$\Delta w =$ incremento a ser dado nos pesos sinápticos

$$\Delta w = -\alpha . \frac{\partial e^2}{\partial w} \qquad \text{(Eq. Erro! Argumento de opção descon}$$

hecido.)

em que $e =$ erro de saída da RNA

$\alpha =$ passo de aprendizagem

e na qual a derivada parcial do erro quadrático em relação aos pesos representa a própria regra delta. Desenvolve-se então a derivada parcial:

$$\frac{\partial e^2}{\partial w} = 2.e.\frac{\partial e}{\partial w} = 2.e.\frac{\partial (y - \hat{y})}{\partial w}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

em que $y =$ saída real, $\hat{y} =$ saída estimada pela RNA e $(y - \hat{y})$ representa o erro de estimação da saída da RNA, assim, tem-se que $\frac{\partial y}{\partial w} = 0$, logo:

$$\frac{\partial e^2}{\partial w} = -2.e.\frac{\partial \hat{y}}{\partial w} = -2.e.\frac{\partial \hat{y}}{\partial u}.\frac{\partial u}{\partial w}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

com

$$u = \sum_{i=1}^{n} w_i . x_i \quad \text{e} \quad \frac{\partial u}{\partial w} = x_i$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Definindo-se $\frac{\partial \hat{y}}{\partial u} = Gr$ (ganho reverso), obtém-se que:

$$\frac{\partial e^2}{\partial w} = -2.e.x_i.Gr$$

(Eq. **Erro! Argumento de opção**

descon hecido.)

assim sendo,

$$\Delta w = 2.\alpha.e.x_i.Gr \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Observa-se claramente que o ganho reverso é dado pelo gradiente da função de saída estimada, sendo um dos motivos pelos quais, na aplicação da regra delta, a função de ativação deve ser diferenciável. Para as funções de ativação mais usuais, a expressão da derivada é conhecida e de fácil realização. Nota-se também que a mudança dos pesos é realizada iterativamente, a cada aquisição de dados, formando um par de entradas/saídas, possibilitando a sua realização *on-line* ou *off-line*.

Uma das características mais determinantes para o tempo de convergência do algoritmo de *backpropagation*, é o passo de aprendizagem α. Conforme **Erro! Argumento de opção desconhecido.**, baixos valores do passo de aprendizagem, Δw pequeno demais, tornam a convergência do algoritmo muito lenta, da mesma forma que, para maiores passos de aprendizagem, Δw grande, a rede que, em tese, tenderia a minimizar o erro mais rapidamente, pode não ter garantida a sua convergência. Neste segundo caso, podendo inclusive divergir para passos de aprendizados exagerados. Em muitas aplicações práticas, a convergência é obtida através da redução progressiva da taxa de aprendizagem, numa metodologia chamada de aproximação estocástica.

Técnicas variantes da *backpropagation* são propostas, a maioria, porém, a literatura considera falha quando assume a magnitude da mudança nos pesos em função amplitude do gradiente. Muitos algoritmos tentam adaptar a taxa de aprendizagem, no entanto, qualquer algoritmo que multiplica a taxa de aprendizagem pelo gradiente para calcular a mudança nos pesos, produz comportamento irregular quando o gradiente muda abruptamente.

Entre outras propostas, o Quickprop (Fahlman, 1989) e RPROP (Riedmiller and Braun, 1993) são algoritmos que não têm dependência exclusiva na magnitude do gradiente, tendo sido registrados resultados satisfatórios (IFAC, 1998) no qual, segundo o IFAC (1998), algoritmos de otimização não apenas usam o gradiente mas também a derivada de segunda ordem, ou combinação destas, para obter um ritmo adequado de aprendizagem. Também há uma variante de aproximação estocástica chamada *média iterativa* ou *média Polyak* (Kushner and Yin, 1997), que propõe uma taxa de convergência mantendo uma média corrente dos valores de peso.

Capítulo 4

4. REDES NEURAIS ARTIFICIAIS EM CONTROLE

As RNA tem sido aplicada nas mais diversas áreas de interesse, dentre outras, pode-se destacar:

- Análise do aprendizado de sistemas inteligentes
- Regressão não linear e classificação de modelos
- Processamento de sinal e controle
- Modelagem de fenômenos físicos

Neste capítulo serão abordados alguns aspectos matemáticos das RNA, adotando, inclusive, a notação formal da teoria de controle. É dado um enfoque no neurônio, unidade de processamento de uma RNA e expandem-se os conceitos para uma rede. Baseando-se nas RNA, são apresentadas estruturas em diagrama de blocos para identificação de sistemas assim como estruturas para realização de blocos controladores.

4.1 Características das RNA

Para que se possa traçar um comparativo entre a aplicação das RNA e dos métodos convencionais aplicados à controle de processos, é importante que se destaquem algumas características das RNA, que representam aspectos positivos deste ferramental:

a) *Sistemas não lineares*: a aplicação das RNA em identificação e controle de sistemas não lineares tem apresentado resultados promissores, baseados na capacidade das RNA de mapear funções aleatórias, não lineares.

b) *Processamento paralelo distribuído*: as RNA têm uma estrutura altamente paralela que resulta de forma natural num processamento paralelo. Uma vez que o neurônio matemático, elemento de processamento numa RNA, é de fácil realização, a realização de redes de processamento paralelas não apresenta dificuldades para execução. A simplicidade matemática, inclusive, além de facilitar o desenvolvimento de algoritmos, minimiza erros de procedimento.

c) *Velocidade de processamento*: por apresentar um elemento processador simples e uma arquitetura paralela, o processamento de uma informação é naturalmente rápido.

d) *Tolerância a falha*: a característica de distribuição da informação torna o sistema tolerante a falha ou, no mínimo, mais eficaz que os métodos convencionais. É comum a realização de testes nos quais, após devidamente treinada, elimina-se uma ou mais unidades de processamento da RNA sem que haja comprometimento significativo na estatística dos resultados de saída (Calôba, 1997).

e) *Aprendizado e adaptabilidade*: as RNA são treinadas mediante registro de dados obtidos do sistema em estudo. Após validação e testes, quando da aquisição de dados em tempo real, a RNA pode ainda aprender mediante situações que não foram abordadas no treinamento, caracterizando uma adaptabilidade *on-line*.

f) *Sistemas multivariáveis*: a RNA é, por natureza, um sistema que processa múltiplas entradas e múltiplas saídas, sendo aplicadas de maneira eficaz à sistemas MIMO.

Através da teoria de controle, já foram desenvolvidos diversos métodos para análise de sistemas não lineares, em geral, porém, baseando-se em aproximações lineares. Assim, do ponto de vista da teoria de controle, a habilidade das RNA em operar com sistemas não lineares representa uma das suas características mais promissoras.

4.2 Um neurônio sob o ponto de vista de controle

A seguir, descreve-se o modelo básico de um neurônio, elemento processador numa RNA, buscando-se aproximar os seus conceitos à teoria de controle, em que $y_1..y_N$ são valores passados das saídas do sistema, $u_1..u_M$ são valores passados das entradas do sistema, caracterizando *N* autoregressores e *M* regressores exógenos e w_i é o *bias* associado ao neurônio.

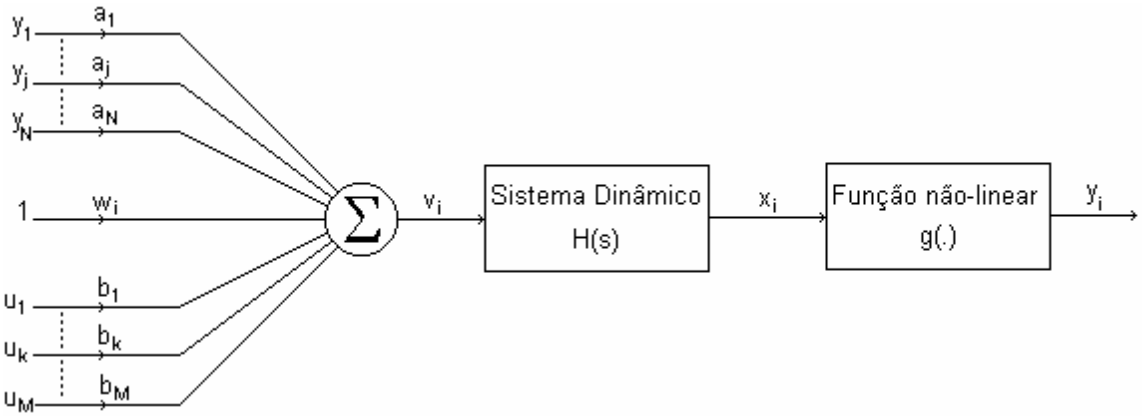

Figura **Erro! Argumento de opção desconhecido.**: Neurônio - elemento básico de processamento numa RNA

Baseando-se na **Erro! Argumento de opção desconhecido.** considera-se basicamente três componentes para análise: o somatório dos pesos, um sistema dinâmico linear, uma função não linear.

4.2.1 Somatório dos pesos

O somatório dos pesos na entrada do elemento processador representa-se na forma:

$$v_i(t) = w_i + \sum_{j=1}^{N} a_{ij}.y_j(t) + \sum_{k=1}^{M} b_{ik}.u_k(t) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que v_i representa o somatório dos pesos no qual $\left[u_1,...,u_k,...,u_M\right]$ é um vetor de entradas externas de dimensão 1xM e pesos b_{ik}, $\left[y_1,...,y_j,...,y_N\right]$ é um vetor de entrada composto por valores passados de saídas do sistema de dimensão 1xN e pesos a_{ij}. O peso associado a polarização, ou *bias*, é representado por w_i. Assim, matricialmente pode-se escrever:

$$v(t) = A.y(t) + B.u(t) + w \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que A é uma matriz NxN composta pelos elementos a_{ij}, B é uma matriz NxM composta pelos elementos b_{ik}. A constante w, representa o valor de polarização do neurônio.

4.2.2 Sistema dinâmico linear

Na **Erro! Argumento de opção desconhecido.**, o sistema dinâmico linear é um sistema SISO (*Sigle Input Single Output*), com uma única entrada e uma única saída, representadas respectivamente por v_i e x_i. Assim, aplicando-se a transformada de Laplace nas funções de entrada e saída, a função de transferência $H(s)$ pode ser expressa por:

$$H(s) = \frac{\overline{x}_i(s)}{\overline{v}_i(s)} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

na qual a barra representa a transformada de Laplace da função.

Representando-se $h(t)$ como a anti-transformada de Laplace de $H(s)$, no domínio do tempo, pode-se então representar a convolução:

$$x_i(t) = \int_{-\infty}^{t} h(t-t').vi(t').dt' \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Destaca-se na **Erro! Argumento de opção desconhecido.**, algumas funções de transferência comumente utilizadas.

Domínio da Frequência	Domínio do Tempo	Equação Diferencial
$H(s) = 1$	$h(t) = \delta(t)$	$x_i(t) = v_i(t)$
$H(s) = \frac{1}{s}$	$h(t) = \begin{cases} 0, & t < 0 \\ 1, & t \geq 0 \end{cases}$	$\dot{x}_i(t) = v_i(t)$
$H(s) = \frac{1}{Ts+1}$	$h(t) = \frac{1}{T}.e^{-t/T}$	$T.\dot{x}_i(t) + x_i(t) = v_i(t)$
$H(s) = \frac{1}{\alpha_0 s + \alpha_1}$	$h(t) = \frac{1}{\alpha_0}.e^{-(\alpha_1/\alpha_0).t}$	$\alpha_0 \dot{x}_i(t) + \alpha_1 x_i(t) = v_i(t)$
$H(s) = e^{-sT}$	$h(t) = \delta(t-T)$	$x_i(t) = v_i(t-T)$

Tabela **Erro! Argumento de opção desconhecido.**: Funções de transferência e transformadas de Laplace

na qual δ representa a função delta de Dirac.

4.2.3 Função não linear

Uma função não linear $g(.)$, mapeia a entrada x_i numa saída y_i, assim, pode ser expressa como:

$$y_i = g(x_i) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Genericamente, as funções não lineares utilizadas podem apresentar as características mais diversas. Usualmente, as limitações para estas funções são consequência das características do algoritmo de treinamento, a exemplo do *backpropagation* que assume ser esta função diferenciável.

A tabela seguinte ilustra algumas propriedades de funções não lineares usualmente utilizadas.

Função	Fórmula	Características
Degrau	+1 se $x > 0$, 0 se $x \leq 0$	Não diferenciável, forma de degrau, positiva
Degrau c/ média zero	+1 se $x > 0$, -1 se $x \leq 0$	Não diferenciável, forma de degrau, média zero
Sigmóide	$1 \big/ \left(1 + e^{-x}\right)$	Diferenciável, forma de degrau, positiva
Tangente Hiperbólica	$tanh(x)$	Diferenciável, forma de degrau, média zero
Gaussiana	$e^{\left(-x^2/\sigma^2\right)}$	Diferenciável, forma de impulso

Tabela **Erro! Argumento de opção desconhecido.**: Características de funções não lineares usuais

À título de exemplo, admite-se um neurônio linear, $H(s) = 1$. Baseando-se nas **Erro! Argumento de opção desconhecido.**, **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.**, pode-se escrever as equações que regem este modelo na forma:

$$x(t) = A.y(t) + B.u(t) + w \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$y(t) = g(x(t)) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Assim como, admitindo-se um neurônio com uma função de transferência de primeira ordem, $H(s) = \frac{1}{Ts+1}$, as equações que regem o modelo podem ser escritas na forma:

$$T\dot{x}(t) + x(t) = A.y(t) + B.u(t) + w \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$y(t) = g(x(t)) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

4.3 Uma RNA sob o ponto de vista de controle

Uma unidade de processamento é extremamente limitada, no entanto, a conexão entre várias unidades lhe atribui uma maior capacidade de processamento e grande facilidade para mapear funções complexas não lineares. Matematicamente, a conexão das unidades

de neurônios descritos na seção anterior, nas suas diversas combinações, geram as equações que regem a RNA formada.

Nos modelos matemáticos utilizados, uma RNA caracteriza-se por camadas consecutivas de neurônios, nas quais cada neurônio é conectado à todos os neurônios da camada anterior, excetuando-se a segunda camada, cujos neurônios são conectados à todas as entradas do sistema.

Para efeito ilustrativo, admite-se uma RNA *feedforward* com 4 camadas, cada uma destas com ***N*** neurônios lineares e função de transferência $H(s) = 1$

Para a primeira camada tem-se que:

$$x_1(t) = B_1.u_1(t) + w_1 \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$y_1(t) = g(x_1(t)) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e para as camadas seguintes tem-se que:

$$x_2(t) = A_2.y_1(t) + w_2 \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$y_2(t) = g(x_2(t)) \qquad \text{(Eq. Erro! Argumento de opção descon}$$

hecido.)

e

$$x_3(t) = A_3 . y_2(t) + w_3 \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$y_3(t) = g(x_3(t)) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

que pode-se representar matricialmente na forma:

$$\begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} = A. \begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix} + B. \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e as estruturas das matrizes A e B são conhecidas:

$$A = \begin{bmatrix} 0_{NN} & 0_{NN} & 0_{NN} \\ A_2 & 0_{NN} & 0_{NN} \\ 0_{NN} & A_3 & 0_{NN} \end{bmatrix} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$B = \begin{bmatrix} B_1 & 0_{NM} & 0_{NM} \\ 0_{NM} & 0_{NM} & 0_{NM} \\ 0_{NM} & 0_{NM} & 0_{NM} \end{bmatrix} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Nas quais 0_{NN} e 0_{NM} são matrizes nulas de dimensão NxN e NxM respectivamente. B_1 é uma matriz de dimensão NxM que representa os pesos sinápticos da camada de entrada. A_2

e A_3 são matrizes de dimensão *N*x*N* que representam os pesos sinápticos da conexão entre as camadas 2-3 e 3-4 respectivamente.

Observe que, na forma apresentada, a dinâmica do sistema é parte integrante do neurônio, o que atribui à rede a capacidade de modelagem de sistemas dinâmicos. É importante destacar-se que, na RNA proposta para estudo, não foram considerados neurônios com dinâmica própria. A capacidade de modelar sistemas dinâmicos se dá por utilizar-se uma TDNN (*Time Delay Neural Network*), ou seja, uma RNA cujas entradas são compostas por valores passados da própria entrada e saída de rede.

4.4 Identificação de sistemas usando RNA

Admite-se um modelo não linear expresso pela equação:

$$y(k+1) = f\left[y(k), y(k-1), .., y(k-n), u(k), u(k-1), .., u(k-m)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que *f* é uma função não linear dos valores passados da saída *[y(k), y(k-1),..., y(k-n)]* e dos valores passados da entrada *[u(k), u(k-1),..., u(k-m)]* do sistema.

Para a modelagem de sistemas dinâmicos, existem duas configurações básicas para disposição das RNA: um modelo direto e um modelo inverso (Brown and Harris, 1994; Hunt at all, 1992; Narendra and Parthasarathy, 1990; Narendra, 1996; Widrow at all, 1981; Widrow and Stearns, 1985; Widrow, 1986).

4.4.1 Modelo direto

Nesta estrutura, a RNA é disposta em paralelo com o sistema a ser identificado e o erro entre a saída real e a saída estimada é utilizado para o treinamento da rede, numa aplicação clássica do algoritmo de *backpropagation*.

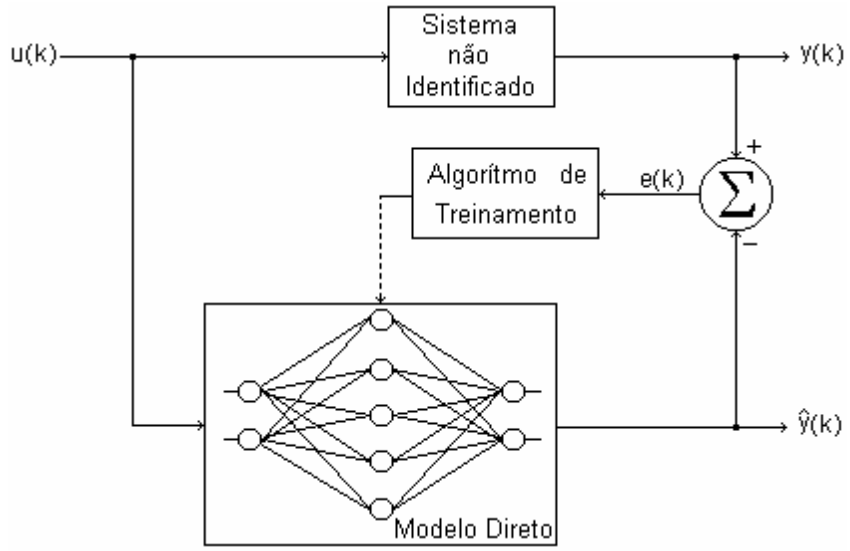

Figura **Erro! Argumento de opção desconhecido.**: Identificação - Modelo Direto

4.4.2 Modelo inverso

Amplamente aplicado à área de controle, o modelo inverso admite a entrada da RNA como sendo a saída da planta real. Neste caso, a saída da RNA deverá ser treinada para apresentar os respectivos valores de entrada dos pares de entrada/saída admitidos para o treinamento, o que caracteriza de forma clara a aprendizagem de um modelo inverso.

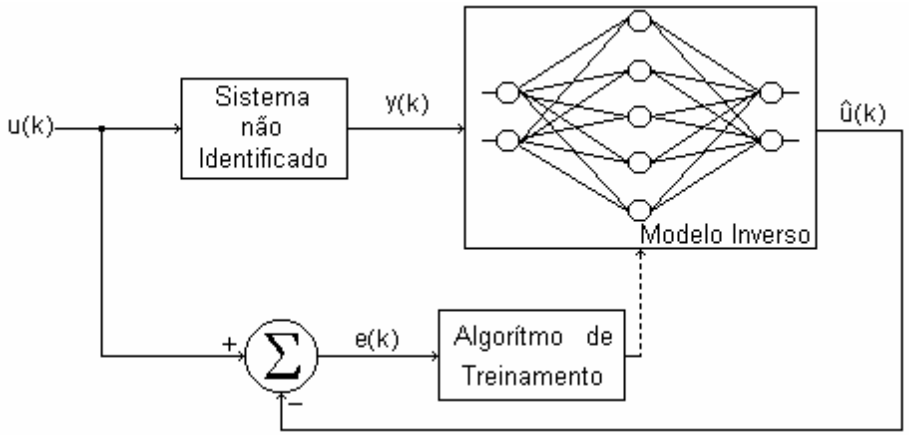

Figura **Erro! Argumento de opção desconhecido.**: Identificação - Modelo Inverso

Para o modelo inverso utilizado, existem alguns cuidados a serem tomados. Destacam-se especialmente situações nas quais o mapeamento não é biunívoco, podendo-se obter modelos inversos incorretos.

4.5 Estruturas de controladores usando RNA

4.5.1 Controle direto e controle inverso

A literatura cita diversas estruturas padrões de controladores baseados em RNA (Brown and Harris, 1994; Hunt at all, 1992; Narendra and Parthasarathy, 1990; Narendra, 1996; Widrow at all, 1981; Widrow and Stearns, 1985; Widrow, 1986; Widrow and Walach, 1996), dentre as quais destaca-se as mais genéricas para ilustração. De uma forma mais ampla, porém, a forma de atuação de controle, assim como os métodos de identificação, também podem ser classificadas como controle direto ou inverso.

Controle direto: Há várias situações em que uma ação humana provém uma realimentação para a ação de controle em determinada tarefa na qual é difícil o controle automático baseando-se em técnicas convencionais. Em algumas situações, porém, é possível projetar um controlador que procure imitar a ação humana caracterizando-se potencialmente como uma aplicação ideal para as RNA. O treinamento da RNA associa-se a um modelo direto, no qual, a ação no sinal de entrada da rede deverá levar à entrada da planta um sinal de controle correspondente a ação humana.

Controle inverso: Extremamente comum em aplicações como robótica (Miller at all, 1990), este modelo procura um mapeamento direto entre a resposta desejada e a respectiva ação de controle (Widrow and Walach, 1996). Esta abordagem também baseia-se na utilização do modelo inverso, já treinado, para uma RNA modelo.

A **Erro! Argumento de opção desconhecido.** e a **Erro! Argumento de opção desconhecido.** a seguir, ilustram em diagrama de blocos a concepção básica dos controladores diretos e inversos.

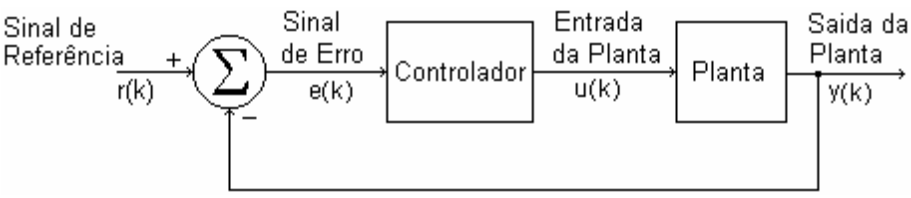

Figura **Erro! Argumento de opção desconhecido.**: Controle direto - realimentação convencional

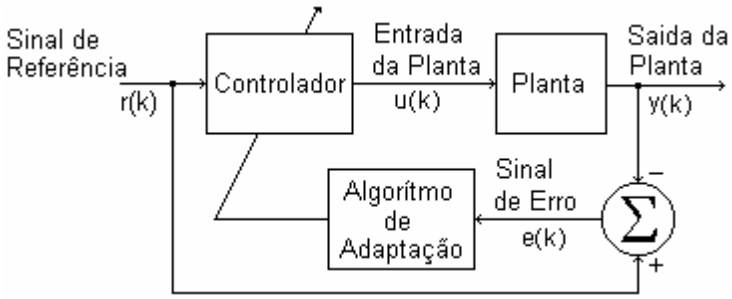

Figura **Erro! Argumento de opção desconhecido.**: Conceito básico do controle inverso

4.5.2 Diagrama de blocos de controladores

a) Controle baseado em modelo de referência

Estes controladores baseiam-se num modelo de referência capaz de gerar pares de entrada/saída desejados para o sistema. O objetivo é fazer a planta responder de forma semelhante ao modelo.

A **Erro! Argumento de opção desconhecido.** ilustra um diagrama de blocos básico para a realização deste controlador.

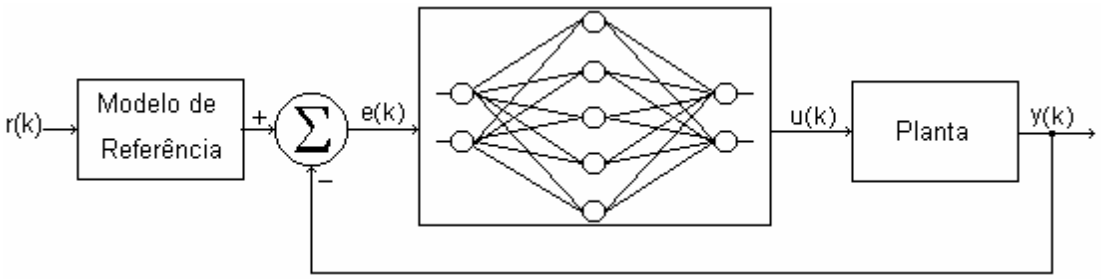

Figura **Erro! Argumento de opção desconhecido.**: Controlador baseado em modelo de referência

Observa-se que o diagrama de blocos de representação para o controlador não se limita à aplicação com as RNA, mas sim a uma estrutura geral de controladores baseados em modelo. Um estudo mais detalhado sobre controladores baseados em modelo de referência pode ser encontrado em Astrom and Wittenmark (1995, Capítulo 5).

b) Controle baseado em modelo interno

Nesta forma de controle, um modelo de referência é colocado em paralelo com a planta real e a diferença entre a resposta real e a resposta do modelo é realimentada para o controlador. O filtro tem por objetivo dar maior robustez ao controlador. O controlador neural responsável por oferecer à entrada da planta um sinal de controle capaz de gerar a saída desejada.

Em Canney and Morari (1986) apresenta-se uma abordagem geral sobre controladores baseados em modelo interno.

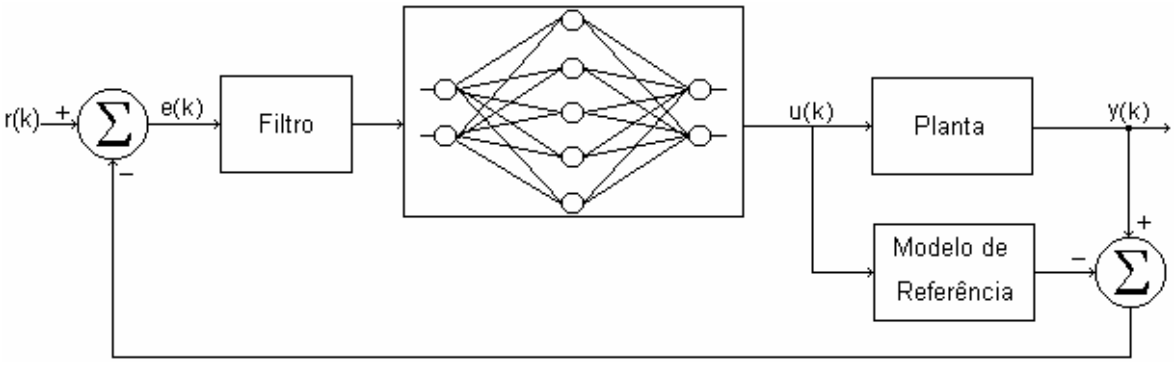

Figura **Erro! Argumento de opção desconhecido.**: Controlador baseado em modelo interno

c) Controle baseado em RNA recursiva

Nesta estrutura, a rede é literalmente treinada para operar como controlador, ou seja, dado um sinal de referência, a RNAR gera um sinal de controle para a planta de forma a forçar que a saída da planta venha a convergir para o valor de referência.

Usualmente, as redes recebem direta ou indiretamente os valores de entrada e saída da planta ou os erros associados a estes. Observa-se que, no caso específico, a RNAR recebe o sinal de saída da planta e é a sua realimentação interna que leva às entradas da rede o sinal de entrada da planta.

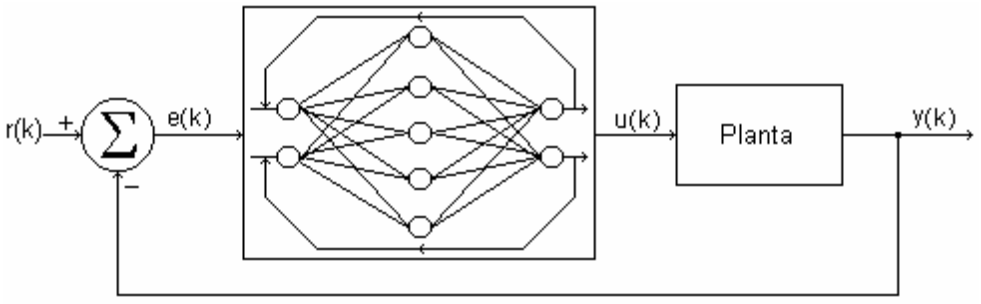

Figura **Erro! Argumento de opção desconhecido.**: Controlador Neural baseado em RNAR

d) Controle ótimo baseado em RNA

A estrutura ilustrada na **Erro! Argumento de opção desconhecido.** caracteriza uma opção comum para utilização de controladores baseados em RNA para soluções em controle ótimo. Neste caso, usualmente o bloco de otimização constitui-se de uma rede competitiva, tipicamente uma rede de Kohonen com realimentação interna de forma a permitir-lhe a ponderação da dinâmica do sistema. Assim, em função da dinâmica captada, a rede de otimização seleciona uma única saída ativa, indicando à rede de controle a respectiva condição de operação, para processamento da ação de controle à ser tomada.

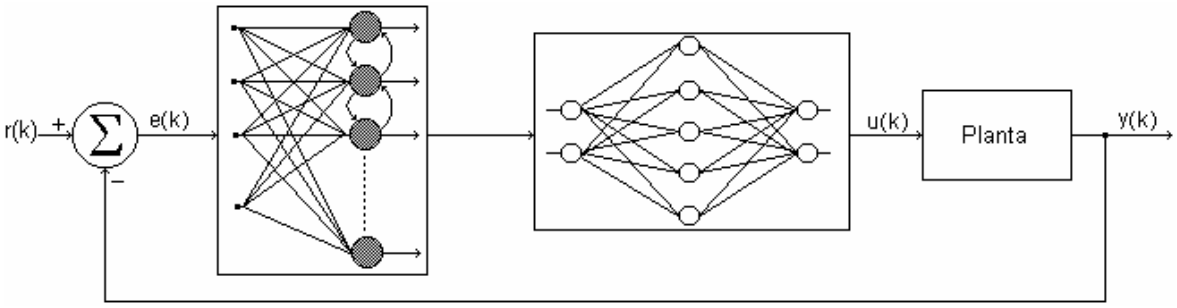

Figura **Erro! Argumento de opção desconhecido.**: Estrutura para controle ótimo baseado em RNA

e) Controle preditivo baseado nas RNA

Realizar a predição de valores futuros através de um modelo e, com base nos valores preditos, numa função otimização e numa lei de controle, gerar uma ação futura para o controlador, é o que caracteriza os controladores preditivos.

No diagrama de blocos da **Erro! Argumento de opção desconhecido.** é apresentada uma estrutura geral para a realização dos controladores preditivos baseados nas RNA. Em alguns casos, ainda pode ser inserido um segundo bloco controlador neural que atua na saída do bloco de otimização. Da mesma forma, nem todos os casos contemplam um modelo de referência, assumindo-se que a RNA já esta devidamente treinada e representa com erro satisfatório a dinâmica da planta à ser controlada.

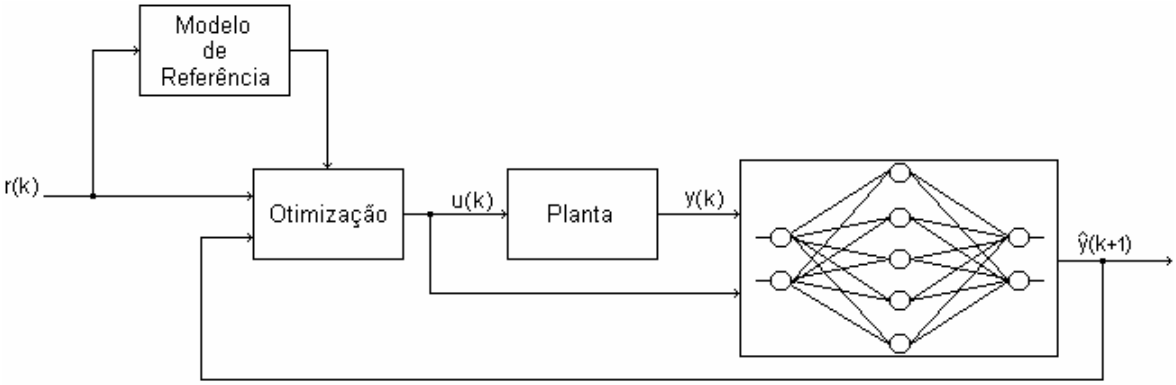

Figura **Erro! Argumento de opção desconhecido.**: Diagrama de blocos de um controlador preditivo neural

Capítulo 5

5. CONTROLE PREDITIVO

O controle preditivo baseado em modelo (MPC - *Model Predictive Control*) em resumo, caracteriza-se por calcular ações de controle futuras baseadas em valores de saída preditos por um modelo, com vasta literatura na área (Clarke, 1987; Astrom and Wittenmark 1995; Garcia et all, 1989; Menezes, 1993; Arnaldo, 1998) e de grande interesse acadêmico e industrial. Neste capítulo, busca-se desenvolver os conceitos de controle preditivo baseando-se em modelos de controladores neurais conforme diagrama da **Erro! Argumento de opção desconhecido.**, ilustrada no Capítulo 4. Caracteriza-se as funções de otimização e leis de controle usualmente aplicadas aos controladores preditivos. No Capítulos 3 e 4 apresentou-se algumas características gerais das RNA e algumas estruturas para aplicação em identificação e controle de sistemas dinâmicos. Neste capítulo, também desenvolve-se as equações características da RNA utilizadas no desenvolvimento prático deste trabalho. Apresenta-se suas equações básicas e de gradiente à serem utilizadas no projeto do controlador. Baseando-se na capacidade de predição da rede, numa função de otimização definida e numa regra de atualização da ação de controle, desenvolve-se também os algoritmos de controle NPC.

5.1 Funções de otimização

A função de otimização, normalmente representada pelo índice *J*, representa a função que procura-se minimizar através das ações de controle. De uma forma bastante intuitiva, a

minimização do erro entre a saída da planta e o valor desejado é o exemplo mais simples e usual de uma função de otimização, e é expressa na forma:

$$J = y(k) - \hat{y}(k) = e(k) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que:

$y(k)$ representa a saída real da planta

$\hat{y}(k)$ representa a saída estimada pelo modelo

$e(k)$ representa o erro de estimação

k é o instante de amostragem

Uma das funções de otimização mais usuais e bastante robusta para muitas aplicações, baseia-se no erro quadrático, ou seja:

$$J = \left[y(k) - \hat{y}(k)\right]^2 = \left[e(k)\right]^2 \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

As funções de otimização, porém, podem tomar formas das mais variadas possíveis, buscando a minimização de funções mais complexas. Para o caso da aplicação de controladores preditivos, cujos modelos são capazes de predizer N passos à frente, a simples aplicação de um critério quadrático pode representar desempenhos satisfatórios, admitindo-se que a função de otimização não se restringe a um único ponto, mas a minimização de todo um vetor de erros preditos, o que representa otimizar toda a trajetória das ações de controle futuras num horizonte de N passos à frente.

$$J = \sum_{j=1}^{N}\left[y(k+j) - \hat{y}(k+j)\right]^2 = \sum_{j=1}^{N}\left[e(k+j)\right]^2 \quad \text{(Eq. Erro! Argumento de opção}$$

descon hecido.)

Funções de otimização ainda mais complexas podem, além de minimizar o erro, buscar a ponderação do esforço de controle, caso específico do GPC (*Generalized Predictive Control*), no qual, o índice *J* da função de otimização pode ser expresso como:

$$J = \sum_{j=N_1}^{N_Y}\left[r(k+j) - \hat{y}(k+j)\right]^2 + \sum_{j=1}^{N_U}\lambda(j).\left[\Delta u(k+j)\right]^2 \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que:

$r(k)$ - é o sinal externo de referência à ser rastreado.

Δu - é a atualização no valor da ação de controle definida como $u(k+j) - u(k+j-1)$.

λ - é uma ponderação à ação de controle, limitando a ação do controlador.

N_1 - é o horizonte mínimo de predição.

N_U - é o horizonte de controle.

N_Y - é o horizonte máximo de predição.

(ver Apêndice A)

O objetivo da função é minimizar o índice *J*, usualmente com respeito às ações de controle, ou seja, procurar o(s) ponto(s) nos quais a primeira derivada é nula.

5.2 Regras de atualização das ações de controle

Além de um índice *J* à ser minimizado, os controladores preditivos baseiam-se em regras de atualização das ações do controlador, ou seja, expressões que regem o incremento na ação de controle atual em função dos índices à serem minimizados.

Uma das regras mais comuns e difundidas é a regra do gradiente decrescente, na qual, a atualização é feita na direção do gradiente negativo da função, procurando sempre o ponto de mínimo - é inclusive a base da regra delta, aplicada no método de treinamento com o algoritmo de *backpropagation*, conforme Capítulo 4.

Uma das maiores restrições para esta regra é o fato de ser não ser aplicável em funções multimodais, podendo o algoritmo convergir para mínimos locais. Assim sendo, não se caracteriza por ser um método de busca de mínimo global. Diversas adaptações são propostas, inclusive algoritmos híbridos que, numa primeira etapa utilizam-se de métodos de busca de mínimos globais, como os AGs, para depois aplicar o método do gradiente decrescente, assegurando a localização do mínimo global com margem de erro definida pelo passo do gradiente decrescente.

A atualização da ação de controle pela regra do gradiente decrescente pode ser expressa na forma:

$$u(k+1) = u(k) - \lambda . \frac{\partial J}{\partial u(k)} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que

$u(k)$ - é o valor da ação de controle no instante k.

λ - é uma ponderação à ação de controle, limitando a ação do controlador.

$\frac{\partial J}{\partial u(k)}$ - é a primeira derivada do índice J, com relação a ação de controle atual.

Admitindo-se um horizonte de predição *Nu*, pode-se escrever um vetor de ações de controle futuras na forma:

$$\vec{U}(k) = \begin{bmatrix} u(k+1) \\ u(k+2) \\ \vdots \\ u(k+Nu) \end{bmatrix}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Assim, a expressão da derivada pode também ser expressa na forma matricial, através da Jacobiana:

$$\frac{\partial J}{\partial \vec{U}(k)} = \begin{bmatrix} \frac{\partial J}{\partial u(k+1)} \\ \frac{\partial J}{\partial u(k+2)} \\ \vdots \\ \frac{\partial J}{\partial u(k+Nu)} \end{bmatrix}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Algumas regras de atualização baseiam-se na primeira e também na segunda derivada da função. Uma das mais difundidas é a regra de Newton-Raphson que pode ser expressa na forma:

$$u(k+1) = u(k) - \left(\frac{\partial^2 J}{\partial^2 u(k)} \right)^{-1} \cdot \frac{\partial J}{\partial u(k)}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

na qual, pode-se escrever matricialmente a expressão da segunda derivada na forma da Hessiana:

$$\frac{\partial^2 J}{\partial \vec{U}^2} = \begin{bmatrix} \frac{\partial^2 J}{\partial u(k+1)^2} & \frac{\partial^2 J}{\partial u(k+2).\partial u(k+1)} & \cdots & \frac{\partial^2 J}{\partial u(k+Nu).\partial u(k+1)} \\ \frac{\partial^2 J}{\partial u(k+1).\partial u(k+2)} & \frac{\partial^2 J}{\partial u(k+2)^2} & \cdots & \frac{\partial^2 J}{\partial u(k+Nu).\partial u(k+2)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial u(k+1).\partial u(k+Nu)} & \frac{\partial^2 J}{\partial u(k+2).\partial u(k+Nu)} & \cdots & \frac{\partial^2 J}{\partial u(k+Nu)^2} \end{bmatrix}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

O desenvolvimento de controladores preditivos baseados em RNA, com aplicações destas regras de atualização e funções de otimização, podem ser encontrados em Soloway and Haley (1997), Schnitman e Fontes (1998), Tan and Cauwenberghe (1996), Noriega and Wang (1998).

5.3 Estrutura da RNA selecionada para estudo e aplicação

É provado que as RNA, com apenas uma única camada escondida e utilizando-se de funções de ativação tipo sigmóide, são estimadores universais de classificações binárias (Farago and Lugosi, 1993), (Devroye; Gyorfi and Lugosi, 1996). Também citado em Tan and Cauwenberghe (1996) e Hecht-Nielsen (1990), é provado que uma rede neural, com apenas uma única camada escondida, tem capacidade de representar qualquer de função de $R^n \rightarrow R^m$, limitada apenas pelo número de neurônios na camada intermediária. Assim, apesar das expressões matemáticas serem genéricas, nesta dissertação, trabalha-se sempre com uma RNA de 3 camadas, cuja topologia básica é a referência dos trabalhos desenvolvidos.

5.3.1 Equações gerais utilizadas

Admite-se um modelo real, não linear, expresso na forma:

$$y(k+1) = f\left[y(k), y(k-1), .., y(k-n), u(k), u(k-1), .., u(k-m)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que *f* é uma função não linear dos valores passados da saída *[y(k), y(k-1),..., y(k-n)]* e dos valores passados da entrada *[u(k), u(k-1),..., u(k-m)]* do sistema.

É sabido que para representarmos sistemas dinâmicos, é necessário o efeito de realimentação na RNA, caracterizando a aplicação das RNAR. Com referências nos

modelos ARX (AutoRegressive with eXogenous inputs), opta-se por associar os sinais de entrada da rede aos próprios valores de entrada e saída passadas, o que caracteriza especificamente uma TDNN. As variáveis *n* e *m* são associadas ao número de auto-regressores e ao número de regressores exógenos respectivamente.

Assim, utilizando-se de uma rede com três camadas, o trabalho desenvolvido baseia-se na estrutura neural ilustrada na **Erro! Argumento de opção desconhecido.**.

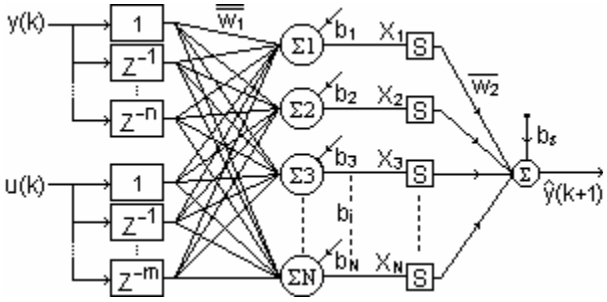

Figura **Erro! Argumento de opção desconhecido.**: Estrutura da TDNN utilizada

em que *N* representa o número de neurônios na camada escondida, os b_i são as polarizações associadas a cada neurônio desta camada e o bloco *S* representa uma função de ativação sigmóide, não linear, aplicada na saída de cada neurônio. O b_s representa a polarização associada ao neurônio linear da camada de saída. A matrizes de pesos w_1 e w_2 têm dimensões *N* x (*n*+*m*) e 1 x *N* respectivamente e representam os pesos das conexões sinápticas entre as camada 1-2 e 2-3.

Para o modelo utilizado, tem-se genericamente a expressão:

$$\hat{y}(k+1) = bs + \sum_{i=1}^{N} w_2(1,i).S(Xi) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que

$$Xi = b(i,1) + \sum_{j=1}^{n} w_1(i,j).y(k-j+1) + \sum_{j=1}^{m} w_1(i,n+j).u(k-j+1) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

a qual representa a saída estimada $\hat{y}(k+1)$ como função não linear, dos valores passados das entradas e saídas do sistema.

5.3.2 Equações da derivada

De uma forma geral, a aplicação de uma lei de controle requer o cálculo de derivadas da expressão de saída em relação à entrada do processo. Baseando-se na **Erro! Argumento de opção desconhecido.** e derivando-a em relação a entrada $u(k)$, para o modelo de RNA utilizado, pode-se generalizar a expressão:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \frac{\partial}{\partial u(k)}\left[bs + \sum_{i=1}^{N} w_2(1,i).S(Xi) \right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

ou, utilizando-se o conceito de derivadas sucessivas pode-se reescrever:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} w_2(1,i).S'(Xi).\frac{\partial Xi}{\partial u(k)} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Na qual S' representa a primeira derivada da função de ativação utilizada, ou $S' = \frac{\partial S}{\partial Xi}$

Baseando-se na **Erro! Argumento de opção desconhecido.**, pode-se expandir a expressão $\frac{\partial Xi}{\partial u(k)}$, na forma:

$$\frac{\partial Xi}{\partial u(k)} = \frac{\partial}{\partial u(k)}\{b(i,1)\} + \frac{\partial}{\partial u(k)}\left\{\sum_{j=1}^{n} w_1(i,j).y(k-j+1)\right\} + \frac{\partial}{\partial u(k)}\left\{\sum_{j=1}^{m} w_1(i,n+j).u(k-j+1)\right\} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Observa-se que os termos *y(k-1), y(k-2),.., y(k-n),* assim como os termos *u(k-1), u(k-2),..., u(k-m),* são valores passados, logo, não dependem de *u(k),* desta forma, o somatório apenas não é nulo em *j = 1*, no termo *u(k)* cuja derivada é 1. Assim:

$$\frac{\partial Xi}{\partial u(k)} = w_1(i,n+1) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

substituindo-se na **Erro! Argumento de opção desconhecido.**, tem-se que:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} w_2(1,i).S'(Xi).w_1(i,n+1) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

A **Erro! Argumento de opção desconhecido.** generaliza a expressão de $\frac{\partial \hat{y}(k+1)}{\partial u(k)}$ para a TDNN utilizada.

5.4 Controlador preditivo neural 1 passo à frente

Um controlador preditivo, por definição, toma ações de controle em função de valores preditos para o sistema

A utilização de algoritmo de treinamento com retropropagação do erro (Chauvim and Rumelhart, 1995) garante que a rede neural, devidamente dimensionada e treinada, pode representar de forma satisfatória o modelo dinâmico proposto. Como resultado, algoritmos de controle podem ser realizados baseando-se na predição de valores de saída, buscando fazer acompanhar um sinal de referência:

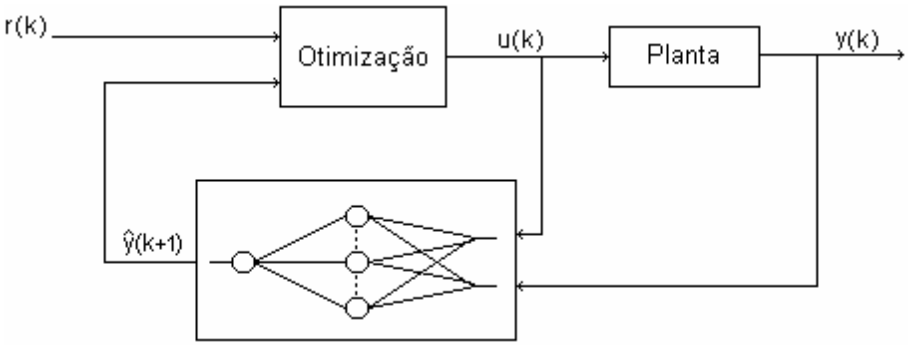

Figura **Erro! Argumento de opção desconhecido.**: Diagrama de blocos para o controlador NPC utilizado

em que:

k = instante discreto.
r(k) = sinal de referência
u(k) = sinal de controle
y(k) = sinal de saída da planta
$\hat{y}$ (k+1) = sinal predito de saída um passo à frente
e(k) = sinal de erro entre a referência e a saída estimada

É importante observar-se que a ação de controle baseia-se na saída predita e não na saída real, ou seja, o modelo considera uma TDNN devidamente treinada e capaz de representar satisfatoriamente a planta, eliminando-se o modelo de referência conforme ilustrado

anteriormente na **Erro! Argumento de opção desconhecido.**, pg. 61. O dimensionamento da TDNN e seu treinamento são abordados no Capítulo 7.

Um controlador em sua concepção geral, busca minimizar um índice, considerando aspectos como erro, esforço de controle, etc. Define-se então para o controlador proposto, um índice *J* à ser minimizado, e que é expresso na forma:

$$J = \frac{1}{2} e^2(k+1) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

no qual:

$$e(k+1) = r(k+1) - \hat{y}(k+1) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Para regra de atualização da ação de controle, baseia-se no gradiente decrescente, assim, tem-se que:

$$u(k+1) = u(k) - \lambda \frac{\partial J}{\partial u(k)} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

sendo λ uma ponderação à ação de controle. Logo, das **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.**:

$$\frac{\partial J}{\partial u(k)} = \frac{1}{2}\left[-2.e(k+1).\frac{\partial \hat{y}(k+1)}{\partial u(k)}\right] = -e(k+1).\frac{\partial \hat{y}(k+1)}{\partial u(k)} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Da **Erro! Argumento de opção desconhecido.**, para o modelo da TDNN proposto, já é conhecida a expressão da derivada, assim:

$$\frac{\partial J}{\partial u(k)} = -e(k+1).\sum_{i=1}^{N} w_2(1,i).S'(Xi).w_1(i,n+1) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Com os resultados obtidos, retornando-se à **Erro! Argumento de opção desconhecido.**, pode-se escrever a lei de controle na forma:

$$u(k+1) = u(k) + \lambda.e(k+1).\left[\sum_{i=1}^{N} w_2(1,i).S'(Xi).w_1(i,n+1)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

E a realização do algoritmo de controle realiza-se nos seguintes passos:

1. Define-se a ponderação de controle (λ).
2. Calcula-se $\hat{y}(k+1)$ usando **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.**.
3. Calcula-se $e(k+1)$ usando **Erro! Argumento de opção desconhecido.**.
4. Calcula-se o novo sinal de controle com **Erro! Argumento de opção desconhecido.**.
5. Aplica-se o novo sinal de controle no sistema.
6. Retorna-se ao passo 2.

5.5 Controlador preditivo neural T passos à frente

O algoritmo proposto na seção anterior pode ser melhorado utilizando-se as técnicas de controle preditivo e considerando-se a capacidade de predição T passos à frente pela TDNN proposta. Assim, valores escalares, antes associados a um único ponto, agora podem ser expressos sob forma de vetores preditos.

$$\vec{R} = \left[r(k+1), r(k+2), \ldots, r(k+T) \right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$\vec{Ye} = \left[\hat{y}(k+1), \hat{y}(k+2), \ldots, \hat{y}(k+T) \right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$\vec{U} = \left[u(k), u(k+1), \ldots, u(k+T-1) \right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

obtendo:

$$\vec{E} = \left[e(k+1), e(k+2), \ldots, e(k+T) \right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Assim, o índice à ser minimizado pode ser expresso na forma:

$$J = \frac{1}{2}\left[\vec{E}.\vec{E}^{T}\right] \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Aplicando-se a regra do gradiente decrescente e, de forma análoga a **Erro! Argumento de opção desconhecido.**, tem-se que:

$$\vec{U}(k+1) = \vec{U}(k) - \lambda \frac{\partial J}{\partial \vec{U}(k)} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que

$$\frac{\partial J}{\partial \vec{U}(k)} = -\vec{E}.\frac{\partial \vec{Y}e}{\partial \vec{U}(k)} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

sendo $\frac{\partial \vec{Y}e}{\partial \vec{U}(k)}$ uma matriz jacobiana obtida derivando-se **Erro! Argumento de opção desconhecido.** em relação a **Erro! Argumento de opção desconhecido.** e que pode ser expandida na forma:

$$\frac{\partial \vec{Y}e}{\partial \vec{U}(k)} = \begin{bmatrix} \frac{\partial \hat{y}(k+1)}{\partial u(k)} & 0 & \cdots & 0 \\ \frac{\partial \hat{y}(k+2)}{\partial u(k)} & \frac{\partial \hat{y}(k+2)}{\partial u(k+1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}(k+T)}{\partial u(k)} & \frac{\partial \hat{y}(k+T)}{\partial u(k+1)} & \cdots & \frac{\partial \hat{y}(k+T)}{\partial u(k+T-1)} \end{bmatrix} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

5.5.1 Cálculo recursivo da matriz jacobiana

Observa-se que das **Erro! Argumento de opção desconhecido.**, **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.**, pode-se generalizar:

$$\hat{y}(k+T) = bs + \sum_{i=1}^{N} w_2(1,i).S\left[Xi(T)\right] \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que

$$Xi(T) = b(i,1) + \sum_{j=1}^{n} w_1(i,j).y(k-j+T) + \sum_{j=1}^{m} w_1(i,n+j).u(k-j+T) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$u(k+T) = u(k+T-1) + \lambda.e(k+T).\left[\sum_{i=1}^{N} w_2(1,i).S'\left[Xi(T)\right].w_1(i,n+1)\right] \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Da **Erro! Argumento de opção desconhecido.**, observa-se que o cálculo da diagonal principal da matriz das derivadas **Erro! Argumento de opção desconhecido.** pode ser generalizado para:

$$\frac{\partial \hat{y}(k+T)}{\partial u(k+T-1)} = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(T)\right].w_1(i,n+1) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Os elementos fora da diagonal principal, podem ser calculados de forma recursiva, baseando-se no valor do elemento da diagonal que está na mesma linha e nos elementos que estão acima.

$$A(i,j) = f\left\{A(i,i), A(k,j)\right\}, \text{ com } k = 1 \text{ até } k = i\text{-}1 \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

gerando-se uma forma recursiva para cálculo da matriz Jacobiana a partir dos elementos da diagonal principal, já conhecidos.

Á título de exemplo, admite-se uma predição 3 passos à frente para verificar-se a lei de formação:

Para T = 1

$$\hat{y}(k+1) = bs + \sum_{i=1}^{N} w_2(1,i).S\left[Xi(1)\right] \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$Xi(1) = b(i,1) + \sum_{j=1}^{n} w_1(i,j).y(k-j+1) + \sum_{j=1}^{m} w_1(i,n+j).u(k-j+1) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

assim

$$Ja(1,1) = \frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(1)\right].w_1(i,n+1) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Para T = 2

$$\hat{y}(k+2) = bs + \sum_{i=1}^{N} w_2(1,i).S\left[Xi(2)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$Xi(2) = b(i,1) + \sum_{j=1}^{n} w_1(i,j).y(k-j+2) + \sum_{j=1}^{m} w_1(i,n+j).u(k-j+2) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

assim

$$Ja(2,2) = \frac{\partial \hat{y}(k+2)}{\partial u(k+1)} = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(2)\right].w_1(i,n+1) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$Ja(2,1) = \frac{\partial \hat{y}(k+2)}{\partial u(k)} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$Ja(2,1) = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(2)\right].\left[w_1(i,n+1).\frac{\partial \hat{y}(k+1)}{\partial u(k)} + w_1(i,n+2)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$Ja(2,1) = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(2)\right].\left[w_1(i,n+1).Ja(1,1) + w_1(i,n+2)\right] \quad \text{(Eq. Erro! Argumento de opção desconh}$$

ecido.)

Para T = 3

$$\hat{y}(k+3) = bs + \sum_{i=1}^{N} w_2(1,i).S\left[Xi(3)\right]$$

(Eq. **Erro! Argumento de opção desconhecido.**)

$$Xi(3) = b(i,1) + \sum_{j=1}^{n} w_1(i,j).y(k-j+3) + \sum_{j=1}^{m} w_1(i,n+j).u(k-j+3)$$

(Eq. **Erro! Argumento de opção desconhecido.**)

assim

$$Ja(3,3) = \frac{\partial \hat{y}(k+3)}{\partial u(k+2)} = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(3)\right].w_1(i,n+1)$$

(Eq. **Erro! Argumento de opção desconhecido.**)

e

$$Ja(3,2) = \frac{\partial \hat{y}(k+3)}{\partial u(k+1)}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

$$Ja(3,2) = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(3)\right].\left[w_1(i,n+1).\frac{\partial \hat{y}(k+2)}{\partial u(k+1)} + w_1(i,n+2)\right]$$

(Eq. **Erro! Argumento de opção desconhecido.**)

$$Ja(3,2) = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(3)\right].\left[w_1(i,n+1).Ja(2,2) + w_1(i,n+2)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$Ja(3,1) = \frac{\partial \hat{y}(k+3)}{\partial u(k)} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$Ja(3,1) = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(3)\right].\left\{w_1(i,n+1).\frac{\partial \hat{y}(k+2)}{\partial u(k)} + w_1(i,n+2).\frac{\partial \hat{y}(k+1)}{\partial u(k)} + w_1(i,n+3)\right\} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$Ja(3,1) = \sum_{i=1}^{N} w_2(1,i).S'\left[Xi(3)\right].\left\{w_1(i,n+1).Ja(2,1) + w_1(i,n+2).Ja(1,1) + w_1(i,n+3)\right\} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Observa-se o último termo da **Erro! Argumento de opção desconhecido.**, em $w_1(i,n+3)$, e conclui-se que para predição de T passos à frente, precisa-se de, no mínimo, T regressores exógenos para montar-se a matriz w_1.

E a realização do novo algoritmo de controle com predição T passos à frente realiza-se nos seguintes passos:

1. Define-se a ponderação de controle (λ) e o horizonte de predição (T).
2. Com **Erro! Argumento de opção desconhecido.**, **Erro! Argumento de opção desconhecido.** calcula-se recursivamente $\vec{Y}e$ e $\vec{U}_k$.
3. Calcula-se $\vec{E}$ usando **Erro! Argumento de opção desconhecido.**.

4. Na forma proposta no item 5.5.1, calcula-se $\frac{\partial \vec{Y}e}{\partial \vec{U}(k)}$.
5. Calcula-se $\frac{\partial J}{\partial \vec{U}(k)}$ usando **Erro! Argumento de opção desconhecido.**.
6. Com **Erro! Argumento de opção desconhecido.** calcula-se o novo vetor de controle $\vec{U}_{k+1}$.
7. Aplica-se o novo sinal de controle no sistema.
8. Retorna-se ao passo 2.

Capítulo 6

6. ESTUDO DA PLANTA MODELO

A seleção de uma planta para atuação dos controladores é de grande importância para caracterizar o desempenho de cada um dos algoritmos realizados. Nesta seleção, procura-se uma planta com acentuadas características de não linearidade que tornem complexa a ação de controle baseada em controladores tipo PID convencionais. Neste capítulo, apresenta-se uma planta para estudo, levanta-se as equações diferenciais que regem o sistema e detalha-se todos os aspectos matemáticos necessários ao entendimento do processo. No final, propõe-se um diagrama para simulação de operação da planta.

6.1 Planta modelo para o controle

Para simulação e controle, seleciona-se então um tanque cônico, caracterizando-se por um sistema com grandes não linearidades, e com objetivo de controle de nível. Tem-se como variável manipulada a vazão de entrada de líquido no tanque. Assim, para análise do desempenho dos controladores, seleciona-se a planta, não linear, conforme a **Erro! Argumento de opção desconhecido.**.

A proposta é controlar o nível do tanque atuando-se na vazão de entrada q_i. Considera-se sempre aberta a válvula de saída para o fluxo $q_{o,}$ que também apresenta característica não linear do tipo $q_o = K\sqrt{h}$, acentuando-se ainda mais as não linearidades deste sistema.

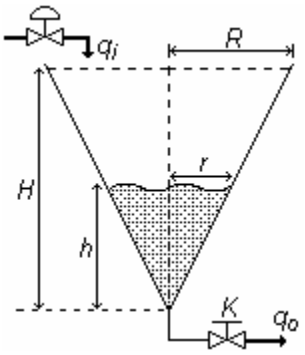

Figura **Erro! Argumento de opção desconhecido.**: Planta modelo para simulação e controle

Em que:

H = altura do tanque

R = raio do tanque

h = nível do líquido

q_i = vazão de entrada

q_o = vazão de saída

K = constante da válvula

6.2 Equações diferenciais que regem o sistema

Partindo-se da relação de equivalência dos triângulos: $\frac{H}{R} = \frac{h}{r}$, tem-se que:

$$r = \frac{R.h}{H}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Sabe-se também que o volume de um cone é dado pela equação:

$$V = \frac{1}{3}\pi.r^2.h$$ (Eq. **Erro! Argumento de opção**

desconhecido.)

Substitui-se a **Erro! Argumento de opção desconhecido.** na **Erro! Argumento de opção desconhecido.**, tem-se então que:

$$V = \frac{1}{3}\pi.\left(\frac{R.h}{H}\right)^2.h = \frac{1}{3}\pi.\left(\frac{R}{H}\right)^2.h^3$$

(Eq. **Erro! Argumento de opção desconhecido.**)

E, genericamente, tem-se que:

$$\frac{dV}{dt} = \pi.\left(\frac{R}{H}\right)^2.h^2.\frac{dh}{dt}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Define-se a constante α como sendo:

$$\alpha = \pi.\left(\frac{R}{H}\right)^2$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Assim, pode-se escrever:

$$\frac{dV}{dt} = \alpha.h^2.\frac{dh}{dt}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

6.2.1 Quanto ao balanço de massa

Do balanço de massa do sistema, pode-se escrever a equação:

$$\delta_i . q_i - \delta_o . q_o = \frac{d(\delta . V)}{dt} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que $\delta = \delta i = \delta o =$ densidade do líquido, que, para o sistema proposto, considera-se constante. Assim, tem-se que:

$$q_i - q_o = \frac{dV}{dt} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Por definição, utiliza-se: $q_i = u$ $\rightarrow$ variável à ser manipulada

$q_o = K.\sqrt{h}$ $\rightarrow$ característica da válvula, não linear

Substituindo-se as variáveis nas **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.**, tem-se que:

$$u - K.\sqrt{h} = \frac{dV}{dt} = \alpha . h^2 . \frac{dh}{dt} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Rearrumando-se a equação, tem-se:

$$\frac{dh}{dt} = \frac{u - K.\sqrt{h}}{\alpha . h^2} = \frac{u.h^{-2}}{\alpha} - \frac{K.h^{-3/2}}{\alpha} \quad \text{(Eq. Erro! Argumento de opção}$$

desconhecido.)

Assim, obtém-se a equação diferencial que rege o sistema:

$$\frac{dh}{dt} = \frac{u.h^{-2}}{\alpha} - \frac{K.h^{-3/2}}{\alpha}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

6.3 Dimensionamento das constantes da planta

O tanque tem suas dimensões definidas baseando-se em valores fictícios, observando-se, porém, valores dentro de um limite de exequibilidade. Apesar do modelo ser o mais realístico possível, assume-se valores para as constantes do tanque - α, e da válvula - K, de forma a obter-se pequenos tempos para subida/descida do nível do tanque, representando uma grande capacidade de vazão de entrada e de saída. As variáveis trabalhadas são adimensionais e o procedimento adotado para o dimensionamento das constantes procura tornar mínimo o tempo de simulação em computador, assim como, em tese, tornar mais difícil a ação do controlador.

Define-se então um tanque cônico com altura $H = 10$ e raio máximo de $R = 5$.

Assim, pela **Erro! Argumento de opção desconhecido.**, tem-se definido o valor da constante do tanque:

$$\alpha = \pi.\left(\frac{R}{H}\right)^2 = \pi.\left(\frac{5}{10}\right)^2 = \frac{\pi}{4}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Para característica da válvula de saída e, a fim de permitir uma grande capacidade de vazão, admite-se $K = 10$, definindo-se então a constante da válvula.

6.4 Condição de equilíbrio

Da condição de equilíbrio, tem-se que:

$$\frac{dh}{dt} = 0 \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Logo, da **Erro! Argumento de opção desconhecido.**, tem-se que, em equilíbrio: $\bar{u} = K\sqrt{\bar{h}}$ **.**

Admite-se um ponto de operação não nulo, assim, para ponto de partida do sistema faz-se $\bar{h} = 3$, e, pela **Erro! Argumento de opção desconhecido.**, tem-se que, no ponto de operação, a vazão é dada por:

$$\bar{u} = K.\sqrt{\bar{h}} = 10\sqrt{3} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

6.5 Linearização em torno do ponto de equilíbrio

A linearização não é essencial aos controladores NPC propostos, no entanto, com a linearização em torno do ponto de equilíbrio, permite-se uma noção exata, matemática, da dinâmica do sistema, caracterizando, inclusive, uma variação muito grande na posição do pólo dominante, o que indica ao engenheiro de controle a não simplicidade do controle desta planta.

Conhecendo-se a equação geral, representada na **Erro! Argumento de opção desconhecido.**, e admitindo-se que no ponto de equilíbrio as condições:

$$\frac{dh}{dt} = 0 \quad \text{e} \quad \bar{h} = constante \tag{Eq. Erro! Argumento de opção desconhecido.}$$

tem-se então que:

$$\bar{u} = K.\sqrt{\bar{h}} \tag{Eq. Erro! Argumento de opção desconhecido.}$$

Admite-se funções auxiliares f_1 e f_2 e, linearizando-se em torno do ponto de equilíbrio. Tem-se então que:

$$f_1(h) = h^{-3/2} \cong \bar{h}^{-3/2} + (h - \bar{h}).\left[-\frac{3}{2}.h^{-5/2} \right]\Bigg|_{h=\bar{h}} \tag{Eq. Erro! Argumento de opção desconhecido.}$$

$$f_1(h) \cong \bar{h}^{-3/2} - \frac{3}{2}\bar{h}^{-5/2}.(h - \bar{h}) \tag{Eq. Erro! Argumento de opção desconhecido.}$$

e

$$f_2(h,u) = h^{-2}.u \cong \bar{h}^{-2}\bar{u} + \left[(h - \bar{h}).\left[u.(-2).h^{-3} \right] + (u - \bar{u}).h^{-2} \right]\Bigg|_{\substack{h=\bar{h} \\ u=\bar{u}}} \tag{Eq. Erro! Argumento de opção desconhecido.}$$

$$f_2(h,u) \cong \bar{h}^{-2}\bar{u} - 2.(h-\bar{h}).\bar{u}.\bar{h}^{-3} + (u-\bar{u}).\bar{h}^{-2}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Substitui-se então as **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.** na **Erro! Argumento de opção desconhecido.**. Assim, tem-se que:

$$\frac{dh}{dt} \cong \frac{1}{\alpha}\left\{\bar{h}^{-2}\bar{u} - 2(h-\bar{h})\bar{u}\bar{h}^{-3} + (u-\bar{u})\bar{h}^{-2} - K\left[\bar{h}^{-3/2} - \frac{3}{2}(h-\bar{h})\bar{h}^{-5/2}\right]\right\}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Assim sendo, tem-se:

$$\frac{d(h-\bar{h})}{dt} \cong \frac{1}{\alpha}\left\{-2(h-\bar{h})\bar{u}\bar{h}^{-3} + (u-\bar{u})\bar{h}^{-2} + \frac{3}{2}K(h-\bar{h})\bar{h}^{-5/2}\right\}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Para simplificação, faz-se:

$$H = h - \bar{h} \qquad e \qquad U = u - \bar{u}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Reescrevendo-se a expressão, tem-se:

$$\frac{dH}{dt} \cong \frac{1}{\alpha}\left\{-2.H.\bar{u}\bar{h}^{-3} + U.\bar{h}^{-2} + \frac{3}{2}K.H.\bar{h}^{-5/2}\right\}$$ (Eq. **Erro! Argumento de opção desconh**

ecido.)

ou ainda,

$$\frac{dH}{dt} \cong \underbrace{\frac{1}{\alpha}\left\{-2.\bar{u}\bar{h}^{-3} + \frac{3}{2}K.\bar{h}^{-5/2}\right\}}_{A}.H + \underbrace{\frac{1}{\alpha}\left\{\bar{h}^{-2}\right\}}_{B}.U$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Sendo A e B constantes, para um ponto de equilíbrio definido, pode-se escrever a equação na forma:

$$\frac{dH}{dt} \cong A.H + B.U$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Aplicando-se a transformada de Laplace para solução da equação diferencial, tem-se que:

$$(s - A).H(s) = B.U(s)$$

(Eq. **Erro! Argumento de opção desconhecido.**)

ou ainda:

$$\frac{H(s)}{U(s)} = \frac{B}{s - A} = \frac{\frac{1}{\alpha}\left\{\bar{h}^{-2}\right\}}{s - \frac{1}{\alpha}\left\{-2.\bar{u}\bar{h}^{-3} + \frac{3}{2}K.\bar{h}^{-5/2}\right\}}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Sabe-se também que $\bar{u} = K.\sqrt{\bar{h}} = K.\bar{h}^{1/2}$. Substituindo-se na equação, tem-se que:

$$\frac{H(s)}{U(s)} = \frac{\frac{1}{\alpha}\left\{\bar{h}^{-2}\right\}}{s - \frac{1}{\alpha}\left\{-2\left(K.\bar{h}^{1/2}\right)\bar{h}^{-3} + \frac{3}{2}K.\bar{h}^{-5/2}\right\}} = \frac{\frac{1}{\alpha}\left\{\bar{h}^{-2}\right\}}{s + \frac{1}{2}\frac{K}{\alpha}.\bar{h}^{-5/2}}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Simplificando-se com os valores definidos de $K = 10$ e $\alpha = \frac{\pi}{4}$, tem-se que:

$$\frac{H(s)}{U(s)} = \frac{\frac{4}{\pi}\left(\bar{h}^{-2}\right)}{s + \frac{20}{\pi}\left(\bar{h}^{-5/2}\right)}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

ou seja, o ganho estático e o pólo da planta são funções do ponto de equilíbrio $\bar{h}$. E, observa-se que o pólo da planta varia de 6.37 para $\bar{h} = 1$ até 0.02 para $\bar{h} = 10$, confirmando-se a dinâmica do sistema e caracterizando-se a complexidade da realização de controladores PID convencionais.

6.6 Sensor de nível

A fim de aproximar mais ainda a planta da realidade, acrescenta-se ao sistema um sensor de nível, representado por um sistema de primeira ordem, com constante de tempo τ e ganho estático K_s. Considera-se h_r a altura real do líquido e h é a altura medida pelo sensor. Assim, o sensor pode ser expresso pela equação:

$$\frac{h}{h_r} = \frac{K_s}{\tau s + 1}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

e pode ser representado pelo seguinte diagrama de blocos:

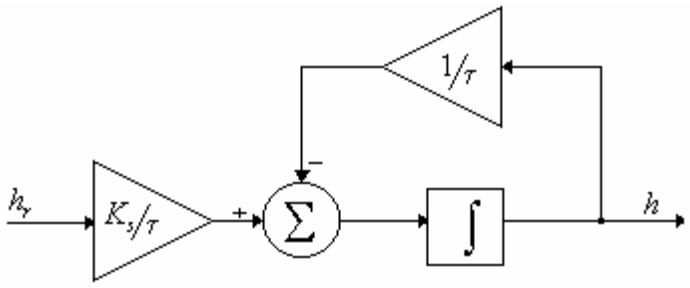

Figura **Erro! Argumento de opção desconhecido.**: Diagrama de blocos do sensor

Para medição eficaz, sabe-se que o sensor deve ter uma dinâmica rápida para captar bem a dinâmica da planta. Sabe-se também que o pólo dominante da planta varia de acordo com **Erro! Argumento de opção desconhecido.**.

Como não se deve operar a planta com tanque vazio, considera-se a sua dinâmica mais rápida para $\bar{h} = 2$, assim, neste ponto, tem-se que o pólo da planta fica em 1,13. Assume-se então o pólo do sensor em 10, caracterizando-se, na pior situação, uma dinâmica quase dez vezes mais rápida que a planta. Assim, para o sensor, tem-se que $\tau = 0.1$. Admite-se o ganho estático unitário para o sensor, ou seja, $K_s = 1$.

6.7 Simulação de operação da planta

Com base na **Erro! Argumento de opção desconhecido.** que rege o sistema, utilizando-se do programa SIMULINK para simulação dinâmica, realiza-se o diagrama de blocos da **Erro! Argumento de opção desconhecido.** para verificação de operação da planta:

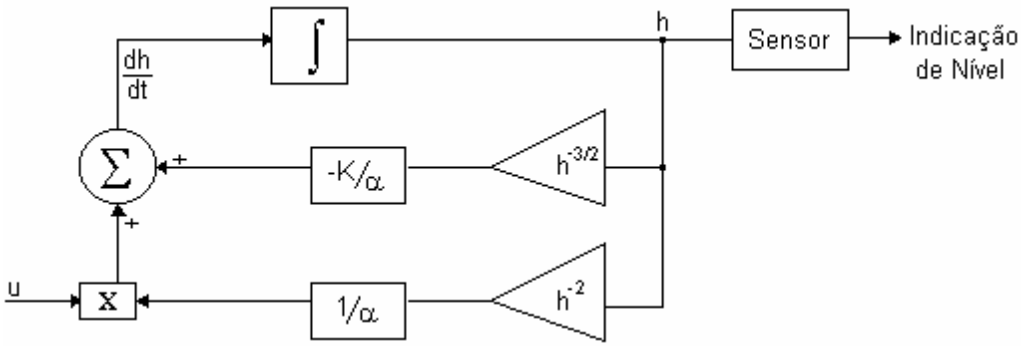

Figura **Erro! Argumento de opção desconhecido.**: Diagrama de simulação de operação da planta

Na simulação, obtém-se resultados absolutamente coerentes com a previsão teórica e considera-se a planta pronta para atuação dos controladores.

Observa-se também que ao excitar a planta com degraus em torno do ponto de equilíbrio, o tempo de estabilização muda em função da amplitude do degrau. Verifica-se também que o tempo de estabilização para um desvio de 5% em $h = 3m$ é aproximadamente 10 vezes menor que o tempo de estabilização para um desvio de 5% em $h = 8m$. Estes, entre outros testes, deixam bem caracterizada a não linearidade do sistema.

Capítulo 7

7. TREINAMENTO, APLICAÇÃO E RESULTADOS

Neste capítulo apresenta-se os resultados obtidos com a realização do algoritmo NPC. Primeiramente, cita-se alguns critérios utilizados para configuração e treinamento da TDNN selecionada e são dimensionados os parâmetros desta TDNN. Admitindo-se a rede devidamente treinada, realizam-se os controladores preditivos neurais - NPC, e convencionais - GPC, para avaliação do desempenho. Apresenta-se os resultados obtidos. Com base nos resultados e, independentemente dos algoritmos de controle, percebe-se pela característica do sistema que a adaptabilidade da ponderação da ação de controle pode contribuir significativamente para uma melhor sintonia dos controladores. No final deste capítulo, apresenta-se os resultados obtidos considerando-se a adaptabilidade da ponderação da ação de controle como função do polo dominante da planta linearizada. Percebe-se claramente a melhoria do desempenho dos controladores com a adaptabilidade proposta e destaca-se nesta uma contribuição científica, ainda muito à ser explorada.

7.1 Parâmetros da RNA

Conforme visto nos capítulos anteriores, os trabalhos baseiam-se numa TDNN de três camadas, cuja caracterização se dá pelo número de auto-regressores: entradas associadas a valores passados da saída da rede; pelo número de regressores exógenos: entradas associadas a valores passados de entradas externas à rede; e pelo número de neurônios na camada intermediária.

A princípio, realiza-se testes para a seleção da estrutura da rede. Com a realização do diagrama de blocos da **Erro! Argumento de opção desconhecido.**, gera-se três conjuntos de dados sintéticos para treinamento de quatro estruturas de rede distintas, e analisa-se o seu desempenho ao longo de um mesmo período de treinamento. A **Erro! Argumento de opção desconhecido.** a seguir completa os resultados obtidos.

Ti	Rede	PesoIni	SSE	ΔSSE
T1	4/4/6	rand(1)	4.37×10^{-5}	5.39×10^{-7}
T2	4/4/6	rand(1)	6.97×10^{-4}	1.14×10^{-4}
T3	4/4/6	rand(1)	8.52×10^{-4}	1.28×10^{-5}
T1	3/3/5	rand(2)	4.12×10^{-3}	3.90×10^{-5}
T2	3/3/5	rand(2)	8.49×10^{-3}	3.12×10^{-5}
T3	3/3/5	rand(2)	3.03×10^{-3}	6.34×10^{-5}
T1	4/2/8	rand(3)	3.54×10^{-3}	6.00×10^{-5}
T2	4/2/8	rand(3)	7.07×10^{-4}	5.25×10^{-5}
T3	4/2/8	rand(3)	4.02×10^{-5}	5.03×10^{-6}
T1	4/4/8	rand(4)	3.40×10^{-4}	5.82×10^{-6}
T2	4/4/8	rand(4)	2.37×10^{-3}	5.38×10^{-6}
T3	4/4/8	rand(4)	1.04×10^{-3}	2.00×10^{-6}

Tabela **Erro! Argumento de opção desconhecido.**: Análise do desempenho da estrutura da TDNN

na qual:

Ti - indica qual dos 3 conjuntos (T1,T2 ou T3) foi utilizado para o treinamento

n/m/N - correspondem ao número de auto-regressores, ao número de regressores exógenos e ao número de neurônios na camada intermediária, respectivamente.

PesoIni - apenas indica que cada conjunto de teste teve seus pesos inicializados aleatoriamente, porém, com valores comuns à cada estrutura neural analisada.

SSE - é o somatório do erro quadrático entre os valores estimados e os valores reais.

ΔSSE - corresponde a variação do SSE nas últimas 20 épocas de treinamento. Este número apenas indica se o treinamento foi encerrado num momento em que a tendência de queda do erro era intensa, não sendo, necessariamente, um dado conclusivo.

De uma forma geral, através dos resultados obtidos nos testes, quaisquer das estruturas de rede propostas é candidata a modelar com sucesso a planta selecionada. A fim de não incorrer em erro, selecionando-se um modelo que futuramente não consiga modelar a planta proposta, assim como, para evitar o superdimensionamento da rede, gerando processamento em excesso, logo, maiores tempos de simulação, opta-se pela rede na estrutura 4/4/6 para o prosseguimento da nossa análise.

7.2 Algoritmo de treinamento da RNA

Existem diversas formas de treinamento de uma RNA, com desempenhos distintos para diferentes aplicações, assim, é importante selecionar-se um algoritmo adequado, que garanta um desempenho eficaz ao treinamento da TDNN, observando-se principalmente os aspectos de tempos de aprendizagem/simulação. Realiza-se testes com a retropropagação do erro (Chauvim and Rumelhart, 1995; Demuth and Beale, 1994; Haykin, 1994) realizado com momentum e uma segunda versão realizada com o método de otimização proposto por Levenberg-Marquardt (Demuth and Beale, 1994; Grace, 1992; Press at all, 1990).

A fim de assegurar-se da mesma condição inicial, faz-se a inicialização dos pesos de forma aleatória, mantida, porém, para todos os treinamentos propostos. Submete-se as redes a três conjuntos dados de treinamento e os resultados obtidos completam a **Erro! Argumento de opção desconhecido.** seguinte:

Ti	Treinamento	PesoIni	SSE	ΔSSE
T1	Backpropagation	rand(1)	69,20	2.03×10^{-1}
T2	Backpropagation	rand(1)	65,16	1.93×10^{-2}
T3	Backpropagation	rand(1)	136,12	$6,00 \times 10^{-3}$
T1	Lev.-Marq	rand(1)	4.94×10^{-5}	9.05×10^{-6}
T2	Lev.-Marq	rand(1)	5.20×10^{-3}	3.30×10^{-5}
T3	Lev.-Marq	rand(1)	5.28×10^{-3}	1.74×10^{-5}

Tabela **Erro! Argumento de opção desconhecido.**: Análise do algoritmo de treinamento

É natural admitir-se que resultados distintos podem ser obtidos alternando-se as taxas de aprendizagem para os algoritmos de *backpropagation* com *momentum*. Assim, a **Erro! Argumento de opção desconhecido.** anterior representa apenas um caso específico da análise do treinamento, que, nas condições propostas e para o sistema em estudo, encontra no algoritmo de Levenberg-Marquardt um desempenho eficaz, satisfatório à nossa aplicação.

Assim, a TDNN utilizada é composta por apenas 6 neurônios na camada escondida: $N=6$; 4 auto-regressores: $n=4$; 4 regressores exógenos: $m=4$; e usa a tangente hiperbólica como função de ativação, e o treinamento baseia-se no algoritmo de Levenberg-Marquardt.

Após o devido treinamento, obtém-se os valores para os pesos e polarizações da rede:

$$w_1 = \begin{vmatrix} 1.1102 & 9.2980 & 5.7334 & -10.4826 & -0.0458 & 0.0654 & -0.1478 & -0.0222 \\ 0.9527 & -5.5380 & -0.3421 & -0.8649 & -0.3673 & 0.0104 & 0.0157 & 0.0882 \\ 2.1366 & -2.0983 & -0.1363 & 0.3950 & 0.0040 & 0.0158 & -0.0065 & -0.0028 \\ -0.7804 & -0.1887 & -1.4589 & -1.2662 & -0.2559 & 0.0156 & -0.2192 & -0.8067 \\ 0.0580 & 0.0258 & -0.0226 & 0.0018 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 1.2056 & -1.2302 & 0.0912 & 0.1288 & 0.0034 & 0.0043 & -0.0024 & -0.0009 \end{vmatrix}$$

$b_1 = [\ -8.3618 \quad 15.2004 \quad 1.4663 \quad -1.3372 \quad -0.6007 \quad 1.7675\ ]^T$

$w_2 = [\ 2.7657 \quad 2.8091 \quad -13.7712 \quad 38.6162 \quad 15.4203 \quad 48.0045\ ]$

$bs = 13.9752$

Uma análise mais detalhada sobre o treinamento desta RNA encontra-se em Schnitman e Fontes (1998). A **Erro! Argumento de opção desconhecido.** ilustra um dos resultados obtidos após o treinamento da TDNN.

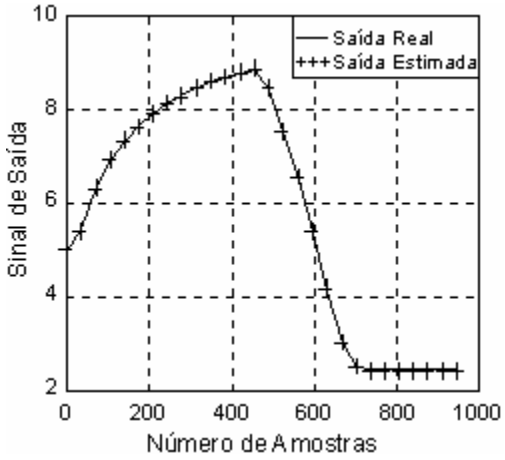

Figura **Erro! Argumento de opção desconhecido.**: Desempenho da TDNN após treinamento

7.3 Resultados obtidos

Realizam-se os controladores através do software Matlab versão 5.0 com Simulink, baseando-se nos modelos e algoritmos desenvolvidos nos Capítulos 5 e 6.

Submete-se a planta, sob ação dos algoritmos de controle NPC e GPC, a um perfil de entrada de grande amplitude. Os resultados obtidos e os respectivos esforços de controle são ilustrados na **Erro! Argumento de opção desconhecido.**.

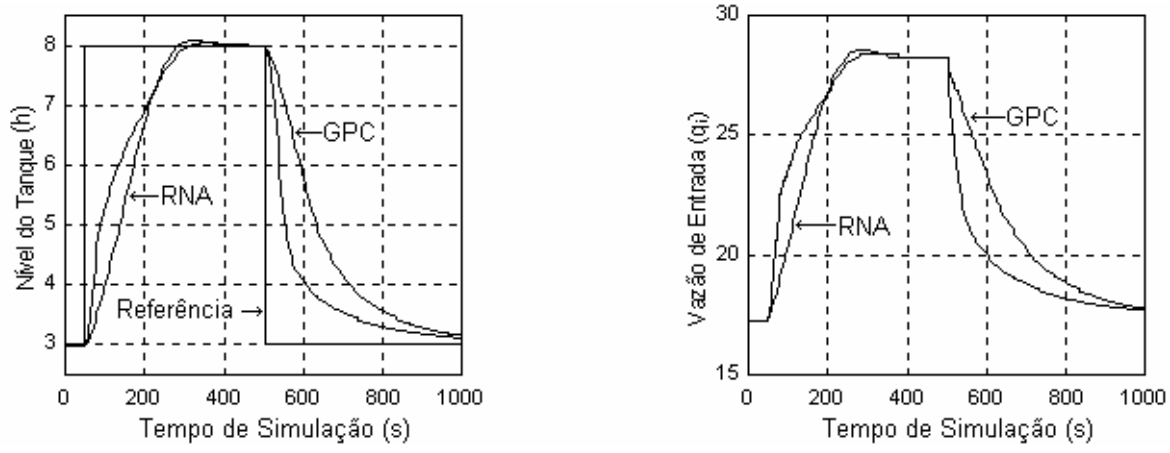

Figura **Erro! Argumento de opção desconhecido.**: Desempenho dos algoritmos de controle e respectivos esforços de controle

Em ambos os casos, observa-se um desempenho bastante satisfatório, que, para a planta proposta, não seria alcançado por controladores convencionais. Verifica-se também a estabilidade das ações de controle.

7.4 Adaptabilidade da ponderação da ação de controle

Pelas características do sistema à ser controlado, já estudado no Capítulo 6 e, com os resultados da realização clássica dos controladores, observa-se que o valor de saída da planta apresenta comportamento distinto para a subida e para a descida do sinal de referência.

Observa-se na **Erro! Argumento de opção desconhecido.** que, na aplicação de ambos os controladores, o peso da ponderação de controle (λ) é pequeno o suficiente para gerar uma sobrecarga na saída quando da subida do sinal de referência, sendo, ao mesmo tempo, grande demais para garantir um tempo de estabilização eficaz na saída da planta, quando da descida do sinal de referência.

Para melhor desempenho do controle para a planta em estudo, é proposta ainda uma sintonia automática do peso da ponderação de controle (λ), de forma a compensar a dinâmica do processo.

Gera-se algumas regras empíricas com resultados práticos satisfatórios, porém, obtém-se resultados mais adequados quando a sintonia é feita em função da posição do pólo dominante, conforme **Erro! Argumento de opção desconhecido.**. Assim, conhecendo-se a curva de adaptação, para aplicação nos algoritmos propostos, sintoniza-se λ_0 e faz-se:

$$\lambda = \lambda_0 . \left[\frac{20}{\pi} . \left(y(k) \right)^{-5/2} \right]^{-1} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

A **Erro! Argumento de opção desconhecido.** ilustra os resultados obtidos pelos controladores considerando-se a adaptabilidade baseada na variação do pólo dominante da planta linearizada.

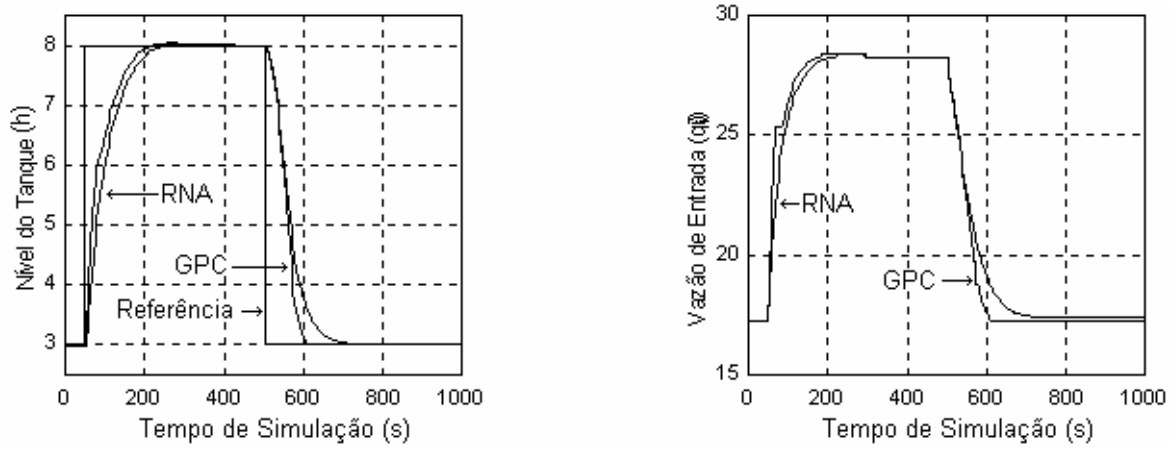

Figura **Erro! Argumento de opção desconhecido.**: Resultados obtidos com sintonia automática dos controladores

Os resultados obtidos apresentam uma grande melhoria no desempenho dos controladores, assim como, para a planta estudada, caracterizam uma proposta adequada para a adaptabilidade de λ em função da posição do pólo dominante.

Capítulo 8

8. CONCLUSÕES E TRABALHOS FUTUROS

É evidente que o fruto deste estudo é muito mais complexo e abrangente do que qualquer explicação escrita que se possa dar, neste capítulo, porém, tenta-se resumir de forma clara algumas conclusões obtidas ao longo do desenvolvimento dos trabalhos e que também baseiam-se nos resultados obtidos em simulação. Uma das conclusões mais evidentes, caracteriza uma fase de pleno desenvolvimento de pesquisas científicas e aplicações industriais, baseadas nas técnicas de Inteligência Artificial, aplicadas à área de controle de processos, o que, em tese, abre um leque muito grande de perspectivas futuras.

8.1 Conclusões

Os resultados obtidos caracterizam um desempenho bastante eficaz na ação de controle exercida pelo NPC, comparável ao GPC, ao longo das simulações.

Os algoritmos são considerados estáveis, sob restrições do valor do peso da ponderação de controle λ, ou seja, grande liberdade a ação de controle, Δu maiores, pode levar ambos os controladores à instabilidade.

8.1.1 Quanto ao esforço computacional

A estimação RLS (*Recursive Least Square)* aplicada ao GPC, permite um desempenho bastante eficaz do controlador, caracterizada pela adaptabilidade dos parâmetros quando da mudança de nível no tanque. A estimação *on-line*, por sua vez, reflete num maior esforço computacional deste algoritmo. A TDNN exige um treinamento prévio para operação, para a realização do controlador, porém, o fato da TDNN já estar treinada para toda a faixa de operação, minimiza o seu esforço computacional.

8.1.2 Não linearidades

O GPC baseia-se em sistemas lineares, porém, através da estimação RLS *on-line* procura linearizar o sistema em torno do ponto de operação, que, realizado dinamicamente, permite uma ação eficaz do controlador em sistemas não lineares. Uma RNA modela naturalmente as não linearidades do sistema, permitindo que, no caso proposto, uma rede devidamente treinada represente de forma satisfatória o sistema em toda a faixa de operação.

8.1.3 Adaptabilidade

O GPC pode ser operado numa configuração não adaptativa. Porém, dada a sua natureza linear e, pela planta proposta, para o GPC representar uma ampla faixa de operação de um sistema não linear, é necessária a sua operação de forma adaptativa, o que o permite ajustar-se as não linearidades do sistema.

Uma RNA pode ser operada de forma adaptativa, com aprendizagem *on-line*, procurando melhorar o desempenho do controlador. A RNA tal qual proposta, é treinada para representar o sistema em toda a sua faixa útil de operação, ou seja, não caracterizou-se necessária a adaptabilidade *on-line*, uma vez que, para os testes realizados, a rede treinada já representa satisfatoriamente o modelo em toda a faixa de operação. Aspecto que diferencia principalmente o esforço computacional.

8.1.4 O engenheiro de controle

Apesar das técnicas avançadas, convencionais e não convencionais aplicadas, fica clara a importância do engenheiro de controle, quando, por sensibilidade, através dos resultados obtidos com os controladores, encontra-se uma forma de adaptação do peso da ponderação de controle baseando-se na dinâmica do sistema. Observa-se claramente uma melhoria no desempenho de ambos os controladores para a adaptabilidade proposta em função do pólo dominante da planta linearizada. A proposta pode ser caracterizada como uma contribuição acadêmico/científica e destaca-se como um tema de interesse, à ser explorado em trabalhos futuros.

8.1.5 Desempenho dos controladores

De uma forma geral, ambos os controladores demostram um desempenho eficaz para o controle da planta proposta. Há praticamente uma equivalência nos resultados obtidos, observando-se, apesar de irrelevante, uma melhor operação da RNA quando não é feita a sintonia automática. Com a proposta de adaptabilidade dos pesos da ponderação de controle, de forma inversa, o desempenho do GPC é ligeiramente superior.

8.2 Perspectivas futuras

O desenvolvimento deste trabalho é sem dúvida compensador e, baseando-se nas referências bibliográficas consultadas, é claro o estado ainda precoce da aplicação de técnicas de Inteligência Artificial no controle de processos, o que, de forma natural, abre um grande leque de perspectivas futuras.

Como fruto do material estudado percebe-se que na área de controle preditivo baseado nas RNA, em específico, ainda há muito a fazer e dentre outros trabalhos cita-se:

1. Realização prática de todo o material teórico desenvolvido em ambiente simulado.
2. Desenvolvimento das diversas regras de atualização da ação de controle para análise e comparação de desempenho quando realizadas num NPC.

3. Desenvolvimento genérico das expressões para realização recursiva da matriz Jacobiana para outros índices de otimização comuns a controladores preditivos.
4. Desenvolvimento genérico das expressões para realização recursiva do cálculo da matriz Hessiana para os diversos índices de otimização propostos.
5. Baseando-se nos itens anteriores, a realização dos controladores NPC com uma variedade de índices e de leis de atualização da ação de controle, permitirá uma ampla avaliação dos controladores preditivos, seus índices de otimização e suas regras de atualização.
6. Explorar-se outras estruturas de controle, além do controle preditivo, avaliando-se o desempenho quando comparado ao controle preditivo e ao equivalente convencional.
7. Desenvolver a teoria da adaptabilidade da ponderação do controle em função da dinâmica da planta à ser controlada.
8. Aplicar as técnicas desenvolvidas de controle preditivo em sistemas multivariáveis.

A aplicação de outras técnicas de Inteligência Artificial também caracteriza perspectivas futuras, entre outras aplicações cita-se:

1. Utilização de técnicas de busca de mínimos globais, como os AGs e técnicas de *Simulated Annealing* para treinamento das RNA, criando-se algoritmos híbridos
2. Desenvolver controladores baseados na *Lógica Fuzzy* ou *NeuralFuzzy*.

Por ser a RNA uma ferramenta de vasta utilidade, aplicações que não se limitem ao controle de processos podem caracterizar pesquisas futuras. Na perspectiva de continuidade dos trabalhos de pesquisa até então desenvolvidos, porém, a conversão dos algoritmos para uma linguagem de programação convencional, pode, a curto prazo, permitir a aplicação das técnicas de controle estudadas, não mais em ambientes simulados, mas sim, em ambientes reais, caracterizando-se aplicações cujo interesse possa extrapolar o meio acadêmico, buscando, à médio prazo, o desenvolvimento de controladores neurais de interesse industrial.

REFERÊNCIAS BIBLIOGRÁFICAS

Ahmed, M.S. and Anjum, M.F.: "*Neural-net-based direct self-tuning control of nonlinear plants*". International Journal of Control. Vol. 66, nº 1, pg. 85-104. 1997.

Arnaldo, M.C.M.: "*Avaliação comparativa de desempenho entre controladores preditivos adaptativos GPC e DMC*". Dissertação de Mestrado do curso de Pós Graduação em Eng. Elétrica da UFBA. 1998.

Astrom, K.J. and Wittenmark, B.: "*Adaptive control*". Addison-Wesley Publishing Company. 1995.

Bishop, C.M: "*Neural network for pattern recognition*". Oxford University Press. Oxford, 1995.

Brown, M.D.; Lightbody, G. and Irwin, G.W.: "*Nonlinear internal model control using local model networks*". IEE Proceedings. Control Theory Application. Vol. 144, nº 6, pg. 505-514. November, 1997.

Brown, M. and Harris, C.: "*Neurofuzzy adaptive modelling and control*". Prentice Hall. 1994.

Calôba, L.P.: "*Notas do curso de redes neurais artificiais*". Curso de Mestrado em Eng. Elétrica da UFBA. 1997.

Candy, J.V.: "*Signal processing: the modern approach*". McGraw Hill. 1988.

Canney, W.M. and Morari, M.: "*Experimental study of internal model control*". Industrial Eng. Chem. Process Des. Dev. Vol 25, nº 1, pg 102-108. 1986.

Chapra, S.C. and Canale, R,P.: "*Numerical methods for engineers*". McGraw Hill. 1988.

Chauvim, Y. and Rumelhart, D.E.: "*Backpropagation theory: architectures and applications*". Laurence Erlbaum Associates Inc. 1995.

Chen, C.L. and Chang, F.Y.: "*Design and analisis of neural/fuzzy variable structural PID control systems*". IEE Proceedings. Control Theory Application. Vol. 143 nº 2, pg. 200-208. March, 1996.

Cichocki A. and Unbehauen R.: "*Neural networks for optimization and signal processing*". John Wiley & Sons Ltd. & B.G. Teubner. 1995.

Clarke, D.W.; Mohtadi, C. and Tuffs, P.S.: "*Generalized predictive control. Part I: the basic algorithm*". Automática, Vol. 23 nº 2, pg. 137-148. 1987.

Clarke, D.W.; Mohtadi, C. and Tuffs, P.S.: "*Generalized predictive control. Part II: extensions and interpretations*". Automática, Vol. 23 nº 2, pg. 149-160. 1987.

Coelho, L.S.: "*Metodologias da inteligência computacional em identificação e controle de processos: abordagem nebulosa, evolutiva e neural*". Dissertação de Mestrado do curso de Pós Graduação em Eng. Elétrica da UFSC. 1997.

Demuth, H. and Beale, M.: "*Neural network toolbox*". Math Works Inc. 1994.

Devroye, L.; Gyorfy, L. and Lugosi, G.: "*A probabilistic theory of pattern recognition*". Springer. NY, 1996.

Duwaish, H.AI; Karim, M.N. and Chandrasekar, V.: "*Use of multilayerfeedforward neural networks in identification and control of Wiener model*". IEE Proceedings. Control Theory Application. Vol. 143, nº 3, pg. 255-258. May, 1996.

Fahlman, S.E.: "*Faster-learning variations on back-propagation: an empirical study*". Proceedings of the 1988 Connectionist Models Summer Scholl, Morgan Kaufmann, pg. 38-51.1989.

Farago, A. and Lugosi, G.: "*Strong universal consistency of neural network classifiers*". IEEE Transactions on Information Theory. Vol. 39, pg. 1146-1151. 1993.

Garcia, C.E.; Prett, D.M. and Morari, M.:"*Model predictive control: theory and practice - a survey*". IFAC Workshop on Model Based Control, Atlanta-USA, pg 335-348. June, 1989.

Goebel, G.: "*An introduction to fuzzy control systems*". Página da World Wide Web, http://www.isis.ecs.soton.ac.uk/research/nfinfo/fuzzycontrol.html. 1998.

Grace, A.: "*Otimization Toolbox: for use with matlab*". Math Works Inc. 1992.

Hanselman, D. and Littlefield B.: "*Mastering matlab: a comprehensive tutorial and reference*". Prentice Hall. 1996.

Haykin, S.: "*Neural Networks: a comprehensive foundation*". Macmillian College Publishing Company Inc. 1994.

Hecht-Nielsen, R.: "*Neurocomputing*". University of California Inc. 1990.

Hunt, K.J.; Sbarbaro, D.; Zbikowski, R. and Gawthrop, P.J.: "*Neural networks for control systems - a survey*". Automática. Vol. 28 nº 6. pg. 1083-1112. 1992.

Hyland, D.C.; Collins, E.G.Jr.; Haddad, W.M. and Hunter, D.L.: "*Neural network system identification for improved noise rejection*". International Journal of Control. Vol. 68, nº 2, pg. 233-258. 1997.

IFAC: "*International Federation of Automatic Control*". Página na World Wide Web, ftp:\\ftp.sas.com/pub/neural/FAQ.html. 1998.

Ifeachor, E.C and Jervis, B.W.: "*Digital signal processing: a practical approach*". Addison Wesley. 1993.

Jagannathan, S.; Lewis, F.L. and Pastravanu, O.: "*Discrete-time model reference adaptive control of nonlinear dynamical systems using neural networks*". International Journal of Control. Vol. 64, nº 2, pg. 217-239. 1996.

Jang, J.R. and SUN, C.: "*Neuro-fuzzy modeling and control*". Proceedings of the IEEE. Vol.83, nº 3, pg. 378-406. 1995.

Kim, Y.H.; Lewis, F.L. and Abdallah, C.T.: "*A dynamic recurrent neural-network-based adaptive observer for a class of nonlinear systems*". Automática, Vol. 33, nº 8, pg. 1539-1543. 1997.

Klinguelfus, M.C.: "*Ambiente integrado para síntese de controladores neurais adaptativos*". Dissertação de Mestrado do curso de Pós Graduação em Eng. Elétrica da UFSC. 1996.

Kohonen, T.: "*Self-organization and associative memory*". Springer-Verlag, USA. 1989.

Krauss, T.P.; Shure, L. and Little J.N.: "*Signal processing toolbox: for use with matlab*". Math Works Inc. 1992.

Kuo, C. B. and Hanselman, D.C.: "*Matlab tools for control system analysis and design*". Prentice Hall, Inc.1994.

Kushner, H.J. and Yin, G.: "*Stochastic Approximation Algorithms and Applications*". Springer-Verlag, NY. 1997.

Leu, Y.; Lee, T. and Wang, W.: "*On-line tunning of fuzzy-neural network for adaptive control of nonlinear dynamical systems*". IEEE Transactions on System, Man and Cybernetics. Vol 27, nº 6, pg. 1034-1043. December, 1997.

Lindfield, G. and Penny, J.: "*Numerical methods using matlab*". Ellis Horwood. 1995.

Marple, S.L. Jr.: "*Digital spectral analysis*". Prentice Hall, Eglewood Cliffs, NJ. 1987.

Masters, T.: "*Advanced algorithms for neural networks: A C++ sourcebook*". John Wiley and Sons, NY. 1995.

McCulloch, W.S and Pitts, W.H.: "*A logical calculus of ideas immanent in nervous activity*". Bull Math Biophys, Vol.5, pg.115-133. 1943.

Mendel, J.M. : "*Fuzzy logic systems for engineering: a tutorial*". Proceedings of the IEEE. Vol. 83, nº 3, pg. 345-377. March, 1995.

Menezes, M.A.S.:"*Implementação de um controlador GPC adaptativo aplicado a processos industriais*". Dissertação de Mestrado do curso de Pós Graduação em Eng. Elétrica da UFSC. 1993.

Miller, W.T.; Sutton, R.S. and Werbos, P.J.: "*Neural networks for control*". MIT Press, Cambridge, MA. 1990.

Mills, P.M.; Zomaya, A.Y. and Tadé, M.O.: "*Adaptive model-based control using neural networks*". International Journal of Control. Vol. 60, nº 6, pg. 1163-1192. 1994.

Narendra, K.S. and Parthasaranthy, K.: "*Identification and control of dynamical systems using neural networks*". IEEE Transactions on Neural Networks. Vol 1, nº 1, pg. 4-27, 1990.

Narendra, K.S.: "*Neural networks for control: theory and practice*". Proceedings of the IEEE. Vol.84, nº 10, pg. 1385-1406. 1996.

Narendra, K.S. and Mukhopadhyay, S.: "*Adaptive control using neural networks and approximate models*". IEEE Transactions on Neural Networks. Vol. 8, nº 3, pg. 475-485. May, 1997.

Nigrin, A: "*Neural networks for pattern recognition*". MIT Press. Cambridge.1993.

Noriega, J.R. and Wang, H.: "*A direct adaptive neural-network control for unknown nonlinear systems and its application*". IEEE Transactions on Neural Networks. Vol 9, nº 1, pg. 27-33. 1998.

Press, W.H.; Teukolsky, S.A.; Vetterling, W.T.; Flannery, B.P.: "*Numerical reciples in fortran*". Cambridge University Press. 1990.

Rice, J.R.: "*Matrix computations & mathematical software*". McGraw Hill. 1983.

Ripley, B.D: "*Pattern recognition and neural networks*". Cambridge University Press. Cambrige, 1996.

Richalet, J.; Rault, A.; Testud, J.L. and Papon, J.: "*Model predictive heuristic control: application to industrial processes*". Automática, Vol. 14, pg. 413-418. 1978.

Riedmiller, M. and Braun, H.: "*A direct adaptive method for faster backpropagation learning: the rprop algorithm*", Proceedings of the IEEE International Conference on Neural Networks. 1993.

Rumelhart, D.E. and McClelland, J.L.: "*Paralel Distributed Processing*". MIT Press. 1986.

Santharam, G. and Sastry, P.S.: "*A reinforcement learning neural network for adaptive control of Markov chains*". IEEE Transactions on Systems, Man and Cybernetics. Vol. 27, nº 5, pg. 588-600. September, 1997.

Schnitman, L. e Fontes, A.B.: "*Predictive and adaptive nonlinear dynamic control system using neural networks*". V Simpósio Brasileiro de Redes Neurais. Minas Gerais, Dezembro.1998.

Schnitman, L. e Fontes, A.B.: "*Análise do treinamento de uma rede neural para modelagem de sistemas dinâmicos*". Revista Ciência e Engenharia. 1998.

Schnitman, L; Arnaldo, M.C.M. e Fontes, A.B.: "*Controladores preditivos: GPC, DMC, Redes Neurais*". Seminário de Instrumentação e Automação, Instituto Brasileiro de Petróleo. Rio de Janeiro, Outubro.1998.

Soloway, D. and Haley, P.: "*Neural generalized predictive control: a Newton-Raphson implementation*". NASA Technical Memorandum 110244. February, 1997.

Stemmer, M.R. and Klinguelfus, M.C.: "*Implementation and test of an integrated environment for adaptive neural control*". 13 th International Conference on Applications of Intelligent Software Systems in Power Plant, Process Plant and Structural Engineering. São Paulo - Brasil, 1995.

Stemmer, M.R. e Klinguelfus, M.C.: "*Implementação de um abiente integrado para controle neural adaptativo*". II Congresso Brasileiro de Redes Neurais. Curitiba - Brasil, 1995.

Tafner, M.A.; Xerez, M. e Rodrigues Filho, I.W.: "*Redes neurais artificiais - introdução e princípios de neurocomputação*". Eko, Blumenau. 1995.

Tan, Y. and Cauwenberghe, A.V.: "*Nonlinear one-step-ahead control using neural networks: control strategy and stability design*". Automática, Vol. 32 pg.1701-1706, nº12. 1996.

Tanomaru, Júlio: "*Motivação, fundamentos e aplicações de algoritmos genéticos*". Proc. II Congresso Brasileiro de Redes Neurais, Curitiba. 1995.

Thomas, D.E. and Armstrong-Hélouvry, B.: "*Fuzzy logic control - a taxonomy of demonstrated benefits*". Proceedings of the IEEE, Vol. 83, Nº 3. March, 1995.

Widrow, B.; Shur, D. and Shaffer, S.: "*On adaptive inverse control*". Fifteenth Asilomar Conference on Circuits, Systems and Computers, pg. 185-189. November, 1981.

Widrow, B. and Stearns, S.D.: "*Adaptive signal processing*". Prentice Hall, Inc. 1985.

Widrow, B.: "*Adaptive Inverse Control*". Adaptive Systems in Control and Signal Processing, pg. 1-5. July, 1986.

Widrow, B. and Lehr, M.A.: "*30 years of adaptive neural networks: Perceptron, Madaline and backpropagation*". Proceedings of the IEEE, Vol. 78, Nº 9, pg. 1415-1441. September, 1990.

Widrow, B. and Lehr, M.A.: "*Adaptive neural networks and their applications*". International Journal of Intelligent Systems. Vol. 8, pg. 453-507. 1993.

Widrow, B.; Rumelhart, D.E. and Lehr, M.A.: "*The basic ideas in neural networks*". Communications of the ACM, Vol. 37, Nº 3, pg. 87-92. March, 1994.

Widrow, B.; Rumelhart, D.E. and Lehr, M.A.: "*Neural networks: applications in industry, business and science*". Communications of the ACM, Vol. 37, Nº 3, pg. 93-105. March, 1994.

Widrow, B. and. Walach, E.: "*Adaptive Inverse Control*". Prentice Hall. 1996.

Willis, M.J.; Montague, G.A.; Di Massimo, C.; Tham, M.T. and Morris, A.J.: "*Artificial neural networks in process estimation and control*". Automática. Vol. 28 nº 6. pg. 1181-1187. 1992.

Yu, S.H. and Annaswamy, A.M.: "*Adaptive control of nonlinear dynamic systems using θ-adaptive neural networks*". Automática, Vol. 33, nº 11, pg. 1975-1995. 1997

Zurada, J.M.: "*Intruduction to artificial neural systems*". PWS Publishing Company. Boston. 1992.

APÊNDICE ERRO! ARGUMENTO DE OPÇÃO DESCONHECIDO.

CONTROLE PREDITIVO GENERALIZADO

Para avaliação dos controladores baseados nas RNA, é importante que se gerem parâmetros confiáveis que possam, de fato, refletir num desempenho eficaz, comparando-se aos resultados obtidos por controladores avançados / convencionais.

O controlador GPC (Clarke, 1987), com uma farta literatura, vem sendo amplamente estudado por pesquisadores e cientistas também tornando-se conhecido, apesar de não tanto difundido, na indústria brasileira. Assim sendo, adota-se este controlador para que se possa realizar uma análise crítica entre o desempenho de controladores baseados em RNA e num controlador avançado e robusto, aplicável a sistemas dinâmicos não lineares com desempenho eficaz.

O GPC, assim como o NPC proposto, pertencem a uma família de controladores da classe MPC (*Model Predictive Control*), ou seja, controladores preditivos baseados em modelo que, em sua essência, adotam critérios similares.

Para a predição, é necessário um modelo matemático. A diversidade de algoritmos MPC deve-se de uma forma geral aos procedimentos distintos para obtenção dos modelos e na forma de cálculo das respectivas ações de controle.

O GPC, na sua forma básica, foi proposto por D.W. Clarke (1987), e baseia-se num modelo paramétrico de entrada/saída, possibilitando, inclusive, a inserção de ruído de medição, tornando-o bastante robusto quanto aos erros de modelagem. Por ser o GPC um controlador preditivo eficaz, como também por representar de uma forma mais ampla os conceitos do controle preditivo, os resultados experimentais do desempenho deste controlador são referências para os resultados obtidos através dos controladores NPC, no controle da planta estudada.

A.1 O Preditor de Diophantine

Conhecido o modelo, o objetivo de um preditor é, num instante *k*, predizer o comportamento futuro do processo, num horizonte de predição *k+j*. O controlador, por sua vez, forçará ações de controle futuras de modo a minimizar um índice de otimização proposto.

Admite-se então um modelo na forma:

$$A(z^{-1}).y(k) = (z^{-d})B(z^{-1}).u(k-1) + C(z^{-1}).w(k) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que (z^{-1}) é um operador de deslocamento temporal e *A*,*B* e *C* são polinômios em (z^{-1}); $y(k)$ é a saída do sistema no instante *k*; $u(k)$ é a entrada externa do sistema no instante *k*; $w(k)$ é o valor do ruído de densidade espectral constante e média nula (ruído branco) no instante *k*, e *d* é um retardo que, se conhecido, pode ser explícito, senão, pode estar contido nos coeficientes do polinômio *B*.

Pode-se reescrever as equações do modelo proposto na forma:

$$\hat{y}(k+1) = (z^{-d})\frac{B(z^{-1})}{A(z^{-1})}.u(k) + \frac{C(z^{-1})}{A(z^{-1})}.\hat{w}(k+1) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Assim, fica explícita a necessidade de determinação de $\hat{w}(k+1)$ para predição de $\hat{y}(k+1)$.

Sendo $w(k)$ definido como um ruído gaussiano de média zero (ruído branco), e a esperança matemática de um sinal aleatório de média nula sendo um valor que tende a zero, pode-se dizer que a melhor estimativa futura para $\hat{w}(k+1)$ é o valor zero.

Nem todos os preditores assumem esta posição, atribuindo ao valor $\hat{w}(k+1)$ o mesmo valor obtido no instante anterior. Este procedimento não caracteriza melhorias na estimação e assume erros maiores que aqueles obtidos se fosse assumido um valor nulo.

Uma terceira linha de predição, baseia-se na parte determinística do ruído, realizando a predição de $\hat{w}(k+1)$ nos dados passados de valores do ruído medido. Este preditor baseia-se na Equação de Diophantine para separar valores passados e futuros da variável aleatória. É o preditor no qual se baseia o GPC, o que lhe proporciona robustez no controle de plantas submetidas a ruídos estruturados.

Assim, a Equação de Diophantine tem por objetivo numa predição *j* passos à frente, separar os valores passados e futuros da parcela de média móvel. Da **Erro! Argumento de opção desconhecido.**, pode-se retirar a expressão:

$$\frac{C(z^{-1})}{A(z^{-1})} = \left[F(z^{-1}) + z^{-j}.\frac{G(z^{-1})}{A(z^{-1})} \right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

ou

$$C(z^{-1}) = A(z^{-1}).F(z^{-1}) + z^{-j}.G(z^{-1}) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

que são expressões nas quais:

$G(z^{-1}).w(k+j)$ representa os valores passados de $w(k+j)$

$F(z^{-1}).w(k+j)$ representa os valores futuros de $w(k+j)$

e os polinômios *A,C,F* e *G* são definidos como sendo:

$$A(z^{-1}) = 1 + a_1.z^{-1} + a_2.z^{-2} + ... + a_{na}.z^{-na};$$

$$C(z^{-1}) = c_1 + c_2.z^{-1} + ... + c_{nc}.z^{-nc};$$

$$F(z^{-1}) = f_1 + f_2.z^{-1} + ... + f_j.z^{-j+1};$$

$$G(z^{-1}) = g_1 + g_2.z^{-1} + ... + g_{na}.z^{-na+1};$$

Substituindo as expressões expandidas na **Erro! Argumento de opção desconhecido.**, e igualando-se os termos em z^{-1}, deduzem-se as expressões:

$$f_i = c_i - \sum_{r=1}^{i-1} f_r.a_{i-r+1} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$g_i = c_{j+i} - \sum_{r=1}^{j+i-1} f_r . a_{j+i-r+1} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Destaca-se que os coeficientes dos polinômios *A* e *C* são obtidos pelo estimador de mínimos quadrados, conforme **Erro! Argumento de opção desconhecido.**. Levanta-se assim os coeficientes dos polinômios *F* e *G*.

A.2 Modelo CARIMA

Admite-se um modelo CARIMA para predição, conforme a **Erro! Argumento de opção desconhecido.**, e admitindo-se a igualdade de Diophantine da **Erro! Argumento de opção desconhecido.**, pode-se representar o modelo pela equação:

$$\Delta\hat{y}(k+j) = (z^{-d}).\frac{B(z^{-1})}{A(z^{-1})}.\Delta u(k+j-1) + \left[F_j(z^{-1}) + z^{-j}.\frac{G_j(z^{-1})}{A(z^{-1})}\right].\hat{w}(k+j) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e assumindo-se que $F_j(z^{-1}).\hat{w}(k+j)$ é composto unicamente por valores futuros de um ruído branco, admite-se então um valor nulo como a melhor predição. Assim, pode-se reescrever a expressão:

$$\Delta\hat{y}(k+j) = (z^{-d}).\frac{B(z^{-1})}{A(z^{-1})}.\Delta u(k+j-1) + z^{-j}.\frac{G_j(z^{-1})}{A(z^{-1})}.\hat{w}(k+j) \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

ou

$$\Delta\hat{y}(k+j) = (z^{-d}).\frac{B(z^{-1})}{A(z^{-1})}.\Delta u(k+j-1) + \frac{G_j(z^{-1})}{A(z^{-1})}.\hat{w}(k) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Mas, da expressão geral de um modelo CARIMA, tem-se que:

$$w(k) = \frac{A(z^{-1}).\Delta y(k) - \left(z^{-d}\right).B(z^{-1}).\Delta u(k-1)}{C(z^{-1})} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

substituindo-se na equação **Erro! Argumento de opção desconhecido.**, tem-se então que:

$$\Delta\hat{y}(k+j) = (z^{-d}).\frac{B(z^{-1})}{A(z^{-1})}.\Delta u(k+j-1) + \frac{G_j(z^{-1})}{A(z^{-1})}.\frac{A(z^{-1}).\Delta y(k) - \left(z^{-d}\right).B(z^{-1}).\Delta u(k-1)}{C(z^{-1})} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

que pode ser reescrita na forma:

$$\Delta\hat{y}(k+j) = \frac{G_j(z^{-1})}{C(z^{-1})}.\Delta y(k) + (z^{-d}).\frac{B(z^{-1})}{C(z^{-1})}\underbrace{\left[\frac{C(z^{-1})}{A(z^{-1})} - (z^{-j}).\frac{G_j(z^{-1})}{A(z^{-1})}\right]}_{F_j(z^{-1})}.\Delta u(k+j-1) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Tem-se assim a equação clássica de predição do GPC:

$$\Delta\hat{y}(k+j) = \frac{G_j(z^{-1})}{C(z^{-1})}.\Delta y(k) + (z^{-j}).\frac{B(z^{-1}).F_j(z^{-1})}{C(z^{-1})}.\Delta u(k+j-d-1) \quad \text{(Eq. Erro! Argumento de opção desconh}$$

ecido.)

ou

$$C(z^{-1}).\Delta\hat{y}(k+j) = G_j(z^{-1}).\Delta y(k) + (z^{-j}).B(z^{-1}).F_j(z^{-1}).\Delta u(k+j-d-1)$$

(Eq. **Erro! Argumento de opção desconhecido.**)

A.3 Modelo para o GPC utilizado

Admite-se um modelo paramétrico de um sistema dinâmico, representado por:

$$y(k+1) = f\left[y(k), y(k-1), .., y(k-n), u(k), u(k-1), .., u(k-m)\right]$$

(Eq. **Erro! Argumento de opção desconhecido.**)

em que *f* é uma função dos valores passados da saída [*y*(*k*), *y*(*k*-1),..., *y*(*k*-*n*)] e dos valores passados da entrada [*u*(*k*), *u*(*k*-1),..., *u*(*k*-m)] do sistema. As diferentes escolhas de *f, n* e *m*, dão origem aos diversos modelos paramétricos disponíveis na literatura.

Utiliza-se a versão do GPC baseada num modelo ARMAX incremental, obtido a partir da **Erro! Argumento de opção desconhecido.**. Neste caso, a função ***f*** pode ser representada através da **Erro! Argumento de opção desconhecido.**, na qual os diversos pesos ***ai*** e ***bi*** ponderam, respectivamente, os valores passados da saída e da entrada do processo. O modelo escolhido é apropriado para processos não ruidosos e é descrito pela **Erro! Argumento de opção desconhecido.**.

$$\Delta\hat{y}_{(k+1)} = \sum_{i=1}^{n} a_i.\Delta y_{(k-i+1)} + \sum_{i=1}^{m} b_i.\Delta u_{(k-i+1)}$$

(Eq. **Erro! Argumento de opção desconh**

ecido.)

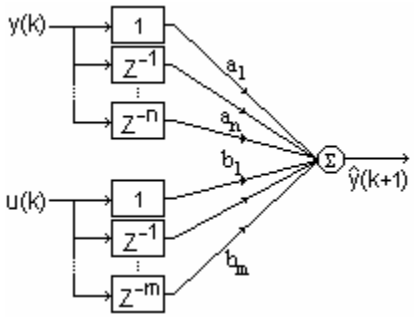

Figura **Erro! Argumento de opção desconhecido.**: Modelo paramétrico para o GPC

O operador Δ constitui-se do polinômio $(1-z^{-1})$, que aplicado às variáveis ***y*** e ***u***, transforma-as na diferença existente entre os valores atual e passado.

Por tratar-se de um modelo linear em torno do ponto de operação, sabe-se que este não retrata o comportamento de um processo não linear a medida em que há um afastamento do ponto de operação. Assim, utiliza-se um estimador de mínimos quadrados recursivos para a atualização dos parâmetros do modelo quando da mudança do ponto de operação. A estimação *on-line* torna o algoritmo adaptativo.

Como o modelo é linear e causal, pode-se considerar o valor predito como a soma de duas parcelas:

> Parcela devido a resposta livre do processo $(\hat{Y}l)$, calculada através da **Erro! Argumento de opção desconhecido.**, utilizando os valores passados já conhecidos e supondo uma sequência nula de ações de controle futuras.
>
> Parcela devido a resposta forçada $(\hat{Y}d)$, obtida a partir da consideração de condição inicial nula, sujeita a uma sequência de ações de controle futuras.

A contribuição forçada ($\hat{Y}d$) é função das ações de controle futuras, ainda desconhecidas, conforme **Erro! Argumento de opção desconhecido.**. A matriz dinâmica (*H)* é obtida por

comparação entre as **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.**.

$$\hat{Y}d = H.\Delta U \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

A.4 Índice de desempenho e expressão do vetor de ações futuras

Para o GPC, calcula-se as ações de controle futuras de forma a minimizar um critério de desempenho. Define-se então:

$$J = \sum_{i=N1}^{NY}\left(r_{(k+j)} - \hat{y}_{(k+j)}\right)^2 + \sum_{i=1}^{NU}\lambda.\Delta u^2{}_{(k+j)} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que:

r é o sinal de referência para o controlador.
N1 = horizonte mínimo de predição
NY= horizonte máximo de predição
NU= horizonte de controle
λ = ponderação de controle

A característica linear dos modelos utilizados permite a obtenção de uma expressão analítica para o vetor de ações de controle futuras:

$$\Delta\vec{U} = \left(H^T.H + \lambda.I\right)^{-1}.H^T.\left(\vec{R} - \hat{Y}l\right) \qquad \text{(Eq. Erro! Argumento de opção desconh}$$

ecido.)

em que:

$$\vec{R} = \left[r(k+1), r(k+2), ..., r(k+NY)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e

$$\Delta\vec{U} = \left[\Delta u(k+1), \Delta u(k+2), ..., \Delta u(k+NU)\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

APÊNDICE ERRO! ARGUMENTO DE OPÇÃO DESCONHECIDO.

ESTIMADORES PARAMÉTRICOS

O controle de sistemas dinâmicos passa pelo processo de identificação dos sistemas a serem controlados, de forma a permitir a realização adequada da ação de controle. Neste procedimento, podem ser aplicadas técnicas convencionais, como o método dos mínimos quadrados, ou não convencionais como as RNA. Os estimadores podem ser realizáveis *on-line* ou *off-line*, recursivos ou não.

Estimadores paramétricos para identificação de sistemas, podem basear-se genericamente num modelo

$$A(z^{-1}).y(k) = B(z^{-1}).u(k) + C(z^{-1}).w(k) \quad \text{(Eq. **Erro! Argumento de opção desconhecido.**)}$$

ou

$$A(z^{-1}).y(k) = (z^{-d})B(z^{-1}).u(k) + C(z^{-1}).w(k) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Em que (z^{-1}) é um operador de deslocamento temporal e *A,B* e *C* são polinômios em (z^{-1}); $y(k)$ são as saídas do sistema; $u(k)$ são entradas externas; $w(k)$ é um ruído de densidade espectral constante e média nula (ruído branco) e *d* é um retardo que, se conhecido, pode ser expresso explicitamente como na **Erro! Argumento de opção desconhecido.**.

As características dos polinômios *A,B* e *C* dão nomes aos modelos:

Modelo	Equação Geral	Observação
AR	$A(z^{-1}).y(k) = \lambda.w(k)$	λ = constante
ARX	$A(z^{-1}).y(k) = B(z^{-1}).u(k)$ ou $A(z^{-1}).y(k) = (z^{-d}).B(z^{-1}).u(k)$	*d* = retardo
ARMA	$A(z^{-1}).y(k) = C(z^{-1}).w(k)$	
ARMAX	$A(z^{-1}).y(k) = B(z^{-1}).u(k) + C(z^{-1}).w(k)$ ou $A(z^{-1}).y(k) = (z^{-d}).B(z^{-1}).u(k) + C(z^{-1}).w(k)$	*d* = retardo
CARIMA	$A(z^{-1}).\Delta y(k) = B(z^{-1}).\Delta u(k) + C(z^{-1}).w(k)$	

Tabela **Erro! Argumento de opção desconhecido.**: Modelagem paramétrica

Um modelo ARMAX genérico, pode ser representado no diagrama de blocos ilustrado na **Erro! Argumento de opção desconhecido.**.

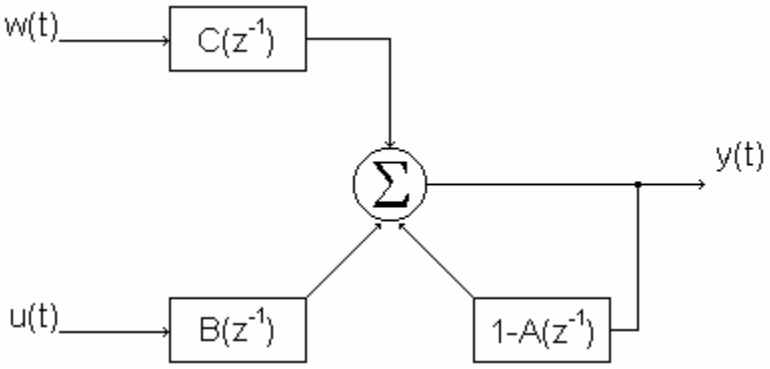

Figura **Erro! Argumento de opção desconhecido.**: Diagrama de blocos de um modelo ARMAX genérico

B.1 Estimador de mínimos quadrados

Por ser uma das técnicas mais difundidas e de fácil realização, adota-se o RLS como método básico de estimação paramétrica. As expressões seguintes baseiam-se genericamente num modelo ARMAX, aplicando-se também ao modelo CARIMA, utilizado pelo GPC, diferenciando-se apenas pelo valor incremental..

Considera-se então um sistema representado por:

$$A(z^{-1}).y(k) = B(z^{-1}).u(k) + C(z^{-1}).w(k) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

ou

$$(1 + a_1.z^{-1} + a_2.z^{-2} + ... + a_{na}.z^{-na}).y(k) = (b_1.z^{-1} + b_2.z^{-2} + ... + b_{nb}.z^{-nb}).u(k) + (c_1.z^{-1} + c_2.z^{-2} + ... + c_{nc}.z^{-nc}).w(k) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que n_a, n_b e n_c definem as ordens dos polinômios A,B e C e são definidos como o número de auto-regressores, o número de regressores exógenos e o número de regressores de média móvel respectivamente. Os vetores $[a_1, a_2, ..., a_{na}]$, $[b_1, b_2, ..., b_{nb}]$ e $[c_1, c_2, ..., c_{nc}]$ são os coeficientes dos polinômios. Assim, pode-se reescrever a **Erro! Argumento de opção desconhecido.** na forma:

$$\begin{aligned} y(k) = &-\left[a_1.y(k-1) + a_2.y(k-2) + ... + a_{na}.y(k-n_a)\right] \\ &+\left[b_1.u(k-1) + b_2.u(k-2) + ... + b_{nb}.u(k-n_b)\right] \\ &+\left[c_1.w(k-1) + c_2.w(k-2) + ... + c_{nc}.w(k-n_c)\right] \end{aligned} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

ou, matricialmente:

$$y(k) = \begin{bmatrix} -y(k-1) & \ldots -y(k-n_a) & u(k-1) & \ldots u(k-n_b) & w(k-1) & w(k-n_c) \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_{na} \\ b_1 \\ \vdots \\ b_{nb} \\ c_1 \\ \vdots \\ c_{nc} \end{bmatrix}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Define-se:

$$\varphi(k) = \begin{bmatrix} -y(k-1) & \ldots -y(k-n_a) & u(k-1) & \ldots u(k-n_b) & w(k-1) & \ldots w(k-n_c) \end{bmatrix}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

e

$$\theta = \begin{bmatrix} a_1 & \cdots & a_{na} & b_1 & \cdots & b_{nb} & c1 & \cdots & c_{nc} \end{bmatrix}^T$$ (Eq. **Erro! Argumento de opção desconhecido.**)

em que θ é o vetor de parâmetros a determinar e φ é denominado vetor de regressão.

Assim, escreve-se:

$$y(k) = \varphi(k).\theta$$ (Eq. **Erro! Argumento de opção**

desconhecido.)

Define-se como $e(k)$ o erro entre o valor predito e o valor real:

$$e(k) = y(k) - \hat{y}(k) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Admite-se um índice J à ser minimizado para um universo de N amostras e aplicando-se o critério quadrático pode-se representar J através da equação:

$$J = \frac{1}{2}.\sum_{k=1}^{N} e(k)^2 = \frac{1}{2}.e(k).e(k)^T \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

O problema dos mínimos quadrados é basicamente minimizar o índice J. E, a solução do problema é obtida com $\frac{\partial J}{\partial \theta} = 0$.

Considerando-se os vetores:

$$\Phi = \begin{bmatrix} \varphi_1 & \varphi_2 & \cdots & \varphi_N \end{bmatrix} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$Y = \begin{bmatrix} y_1 & y_2 & \cdots & y_N \end{bmatrix} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

$$\widehat{Y} = \begin{bmatrix} \widehat{y}_1 & \widehat{y}_2 & \cdots & \widehat{y}_N \end{bmatrix}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

$$E = \begin{bmatrix} e_1 & e_2 & \cdots & e_N \end{bmatrix}$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Tem-se que:

$$J = E.E^T = (Y - \widehat{Y}).(Y - \widehat{Y})^T = (Y - \Phi.\theta).(Y - \Phi.\theta)^T$$ (Eq. **Erro! Argumento de opção desconhecido.**)

$$J = Y^T.Y - \theta^T\Phi^T Y - Y^T\Phi\theta + \theta^T\Phi^T.\Phi\theta$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Para minimizar *J*, faz-se:

$$\frac{\partial J}{\partial \theta} = \left[\frac{\partial J}{\partial \theta_1} \quad \frac{\partial J}{\partial \theta_2} \quad \cdots \quad \frac{\partial J}{\partial \theta_{na+nb+nc}} \right]$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Aplicando-se a derivada parcial na **Erro! Argumento de opção desconhecido.** tem-se que:

$$\frac{\partial J}{\partial \theta} = -2.\Phi^T Y + 2.\Phi^T \Phi\theta = 0$$ (Eq. **Erro! Argumento de opção desconhecido.**)

obtendo-se

$$\theta = \left[\Phi^T \Phi\right]^{-1}.\Phi^T Y$$ (Eq. **Erro! Argumento de opção desconhecido.**)

que é a equação geral do estimador de mínimos quadrados.

B.2 Mínimos quadrados recursivos - RLS

Uma das características indesejáveis do simples estimador de mínimos quadrados, porém, é a necessidade de inversão da matriz $\left[\Phi^T \Phi\right]$, atualizada a cada amostra. A realização recursiva do algoritmo de mínimos quadrados, busca minimizar o esforço computacional no processo de estimação.

A recursividade busca representar um incremento a ser dado aos parâmetros estimados, em função dos parâmetros do passo anterior.

Da dedução de mínimos quadrados, tem-se que:

$$Y(k) = \Phi\,(k).\theta(k)$$ (Eq. **Erro! Argumento de opção desconhecido.**)

Admite-se um sistema com *m* parâmetros e *n* amostras. Esta mesma expressão pode então ser escrita matricialmente na forma:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1m} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{n1} & \varphi_{n2} & \cdots & \varphi_{nm} \end{bmatrix} . \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_m \end{bmatrix}, \quad \text{para } k = 1..n \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

As expressões acima, segundo o critério dos mínimos quadrados, fornecem a estimação dos parâmetros $\hat{\theta}_n$ em função dos dados adquiridos até então.

Admite-se então a chegada de uma nova amostra y_{n+1}. O objetivo da recursividade é obter o novo vetor de parâmetros $\hat{\theta}_{n+1}$ em função do vetor já calculado $\hat{\theta}_n$.

Assim, pode-se escrever:

$$Y_{n+1} = \Phi_{n+1} . \theta_{n+1} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

em que:

$$Y_{n+1} = \begin{bmatrix} Y_n \\ y_{n+1} \end{bmatrix} \quad \text{e} \quad \Phi_{n+1} = \begin{bmatrix} \Phi_n \\ \varphi_{n+1} \end{bmatrix} \qquad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Aplicando-se a equação dos mínimos quadrados, tem-se que:

$$\theta_{n+1} = \left[\Phi^T{}_{n+1} . \Phi_{n+1} \right]^{-1} . \Phi^T{}_{n+1} Y_{n+1} \qquad \text{(Eq. Erro! Argumento de}$$

opção desconhecido.)

mas, $\left[\Phi^T{}_{n+1}.\Phi_{n+1}\right]$, pode ser escrito como:

$$\Phi^T{}_{n+1}.\Phi_{n+1} = \left[\Phi^T{}_n \quad \varphi_{n+1}\right].\begin{bmatrix}\Phi_n \\ \varphi^T{}_{n+1}\end{bmatrix} = \Phi^T{}_n.\Phi_n + \varphi_{n+1}.\varphi^T{}_{n+1}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

e $\left[\Phi^T{}_{n+1}.Y_{n+1}\right]$, pode também ser escrito como:

$$\Phi^T{}_{n+1}.Y_{n+1} = \Phi^T{}_n.Y_n + \varphi^T{}_{n+1}.y_{n+1}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

que representam sob forma de soma, o passado até a amostra *n* e a atualização com a amostra *n*+1.

Para simplificação matemática, faz-se:

$$\left(\Phi^T{}_n.\Phi_n\right)^{-1} = P_n$$

(Eq. **Erro! Argumento de opção desconhecido.**)

e

$$\left(\Phi^T{}_n.Y_n\right) = Q_n$$

(Eq. **Erro! Argumento de opção desconh**

ecido.)

de forma que:

$$\hat{\theta}_n = P_n.Q_{n+} \qquad \text{e} \qquad \hat{\theta}_{n+1} = P_{n+1}.Q_{n+1}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

e ainda:

$$\left(P_{n+1}\right)^{-1} = \left(P_n\right)^{-1} + \varphi_{n+1}.\varphi^T{}_{n+1}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

e

$$\left(Q_{n+1}\right) = \left(Q_n\right) + \varphi_{n+1}.y_{n+1}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

Aplicando-se o Lema da Inversão, expresso na forma:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B.\left(C^{-1} + DA^{-1}B\right)^{-1}.DA^{-1}$$

(Eq. **Erro! Argumento de opção desconhecido.**)

e fazendo:

$$A = P_n^{-1} \quad / \quad B = \varphi_{n+1} \quad / \quad C = 1 \text{ (escalar)} \quad / \quad D = \varphi^T{}_{n+1}$$

tem-se então que:

$$P_{n+1} = P_n.\left[I - \varphi_{n+1}.\left(1 + \varphi^T{}_{n+1}.P_n.\varphi_{n+1}\right)^{-1}.\varphi^T{}_{n+1}.P_n\right] \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Define-se o erro de estimação:

$$e_{n+1} = y_{n+1} - \varphi^T{}_{n+1}.\hat{\theta}_n \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

e substituindo-se o valor de y_{n+1} na **Erro! Argumento de opção desconhecido.**, pode-se então escrever:

$$\left(Q_{n+1}\right) = \left(Q_n\right) + \varphi_{n+1}.\left(e_{n+1} + \varphi^T{}_{n+1}.\hat{\theta}_n\right) \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Substitui-se as **Erro! Argumento de opção desconhecido.** e **Erro! Argumento de opção desconhecido.** na **Erro! Argumento de opção desconhecido.**, tem-se então que:

$$\hat{\theta}_{n+1} = \hat{\theta}_n + P_{n+1}.\varphi_{n+1}.e_{n+1} \quad \text{(Eq. Erro! Argumento de opção desconhecido.)}$$

Note que a atualização dos parâmetros estimados não requer a cada passo uma inversão matricial. Assim, o algoritmo de estimação recursiva dos mínimos quadrados (RLS) pode ser realizado nos seguintes passos:

1. Inicializar o vetor de parâmetros: $\hat{\theta}$;
2. Adquirir nova amostra: y_{n+1};
3. Com **Erro! Argumento de opção desconhecido.**, montar do vetor de regressão: φ_{n+1};
4. Obtenção do erro através da **Erro! Argumento de opção desconhecido.**: $e_{n+1} = y_{n+1} - \varphi^T{}_{n+1}.\hat{\theta}_n$;
5. Com **Erro! Argumento de opção desconhecido.**, atualizar matriz $P_{n+1} = P_n.\left[I - \varphi_{n+1}.\left(1 + \varphi^T{}_{n+1}.P_n.\varphi_{n+1}\right)^{-1}.\varphi^T{}_{n+1}.P_n\right]$;
6. Usar **Erro! Argumento de opção desconhecido.** para atualização do vetor de parâmetros: $\hat{\theta}_{n+1} = \hat{\theta}_n + P_{n+1}.\varphi_{n+1}.e_{n+1}$;
7. Retorno ao passo 2.

B.3 Diagrama de blocos para estimação paramétrica

O estimador RLS deve ter como referência a ordem do modelo (n_a,n_b,n_c), o fator de esquecimento (λ) e o vetor de parâmetros iniciais. A **Erro! Argumento de opção desconhecido.** a seguir, ilustra um diagrama de blocos de simulação para a estimação paramétrica de uma planta linear não identificada.

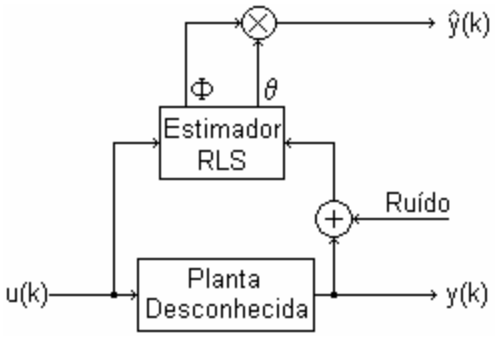

Figura **Erro! Argumento de opção desconhecido.**: Diagrama de blocos para estimação RLS

Observe que a presença de ruído, não sendo um ruído branco, torna necessária a introdução dos coeficientes do polinômio *C*, caracterizando a necessidade dos coeficientes de média móvel para modelagem do ruído.