UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E ENGENHARIA DO PETRÓLEO

CONTROLE PREDITIVO NEURAL APLICADO À PROCESSOS PETROQUÍMICOS

LUIZ HENRIQUE GOMES POPOFF

NATAL / 2009

LUIZ HENRIQUE GOMES POPOFF

CONTROLE PREDITIVO NEURAL APLICADO À PROCESSOS PETROQUÍMICOS

Dissertação submetida ao programa de Pós-Graduação em Ciência e Engenharia do Petróleo, da Universidade Federal do Rio Grande do Norte, como parte dos requisitos necessários para obtenção do titulo de Mestre em Ciência e Engenharia do Petróleo.

Orientador:
Prof. D.Sc. André Laurindo Maitelli

NATAL / 2009

LUIZ HENRIQUE GOMES POPOFF

CONTROLE PREDITIVO NEURAL APLICADO À PROCESSOS PETROQUÍMICOS

Dissertação submetida ao programa de Pós-Graduação em Ciência e Engenharia do Petróleo, da Universidade Federal do Rio Grande do Norte, como parte dos requisitos necessários para obtenção do titulo de Mestre em Ciência e Engenharia do Petróleo.

Aprovado em: ____/____/____

BANCA EXAMINADORA

_____

Prof. D. Sc. André Laurindo Maitelli

Orientador

UFRN – Universidade Federal do Rio Grande do Norte

_____

Prof. D. Sc. Fábio Meneghetti Ugulino de Araújo

UFRN – Universidade Federal do Rio Grande do Norte

_____

Prof. D. Sc. Oscar Gabriel Filho

UNP – Universidade Potiguar

RESUMO

A pesquisa tem como objetivo desenvolver uma estrutura de controle preditivo neural, com o intuito de controlar um processo de pH, caracterizado por ser um sistema SISO (*Single Input - Single Output*). O controle de pH é um processo de grande importância na indústria petroquímica, onde se deseja manter constante o nível de acidez de um produto ou neutralizar o afluente de uma planta de tratamento de fluidos. O processo de controle de pH exige robustez do sistema de controle, pois este processo pode ter ganho estático e dinâmica não-lineares. O controlador preditivo neural envolve duas outras teorias para o seu desenvolvimento, a primeira referente ao controle preditivo e a outra a redes neurais artificiais (RNA's). Este controlador pode ser dividido em dois blocos, um responsável pela identificação e outro pelo o cálculo do sinal de controle. Para realizar a identificação neural é utilizada uma RNA com arquitetura *feedforward* multicamadas com aprendizagem baseada na metodologia da Propagação Retroativa do Erro (*Error Back Propagation*). A partir de dados de entrada e saída da planta é iniciado o treinamento *offline* da rede. Dessa forma, os pesos sinápticos são ajustados e a rede está apta para representar o sistema com a máxima precisão possível. O modelo neural gerado é usado para predizer as saídas futuras do sistema, com isso o otimizador calcula uma série de ações de controle, através da minimização de uma função objetivo quadrática, fazendo com que a saída do processo siga um sinal de referência desejado. Foram desenvolvidos dois aplicativos, ambos na plataforma *Builder* C++, o primeiro realiza a identificação, via redes neurais e o segundo é responsável pelo controle do processo. As ferramentas aqui implementadas e aplicadas são genéricas, ambas permitem a aplicação da estrutura de controle a qualquer novo processo.

Palavras Chave: Rede Neural Artificial; Controle Avançado; Controle Preditivo;

ABSTRACT

The project's purpose is to develop a structure for the predictive neural control of the pH process, characterized as a SISO (Single Input - Single Output). The control of pH is an important process in petrochemical industries, where they want to keep a constant level of acidity of a product or neutralize the influx of a plant for treatment of fluids. The process control of pH requires a strong control system, as this process may have static gain and non-linear dynamics. The neural predictive controller involves two other theories for its development. The first relating to predictive control and the other the artificial neural networks (ANN's). This controller can be divided into two blocks, one for identification and the other by calculating the signal of control. To perform the neural identification, ANN is used with a multilayer feedforward architecture with learning methodology based on the Error Back Propagation. Starting with the plants input and output information the offline training of the network begins. Thus, the synaptic weights are adjusted and the network is able to represent the system with the maximum accuracy. The generated neural model is used to predict the future outputs of the system, and together with the optimizer, estimates a series of actions to control by minimizing a quadratic objective function, making the output of the process follow a desired reference signal. Two applications were developed, both in C ++ Builder platform, where the first makes the identification through neural networks and the other is responsible for controlling the process. The tools implemented here are generic and applied both to the implementation of the control structure of any new process.

Key words: Artificial Neural Network; Advanced Control; Predictive Control

AGRADECIMENTOS

A Deus, acima de tudo.

Ao meu orientador, Prof. André Laurindo Maitelli, a quem sou grato pela a orientação, atenção e confiança depositada.

Aos professores e funcionários constituem o Programa de Pós-Graduação em Ciência e Engenharia do Petróleo, que contribuíram de forma direta ou indireta para o desenvolvimento deste trabalho.

Aos professores e funcionários do Departamento de Engenharia Elétrica, por terem colaborado de forma essencial no meu desenvolvimento acadêmico e profissional.

DEDICATÓRIA

Aos meus amigos, que sempre me apoiaram nas horas alegres e difíceis da minha vida.

Aos meus irmãos, por sempre estarem ao meu lado nesta caminhada.

E principalmente aos meus pais, por todo amor, carinho e dedicação. Por sempre me apoiarem nas minhas escolhas e decisões. Por representarem tudo de bom na minha vida.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

*	Valor ótimo ou fixo
^	Valor estimado
x_p	Vetor de entrada do neurônio artificial
W_i	Vetor de pesos da rede neural artificial
w_i	Pesos sinápticos
$f(.)$	Função de ativação do neurônio artificial
v_j	Pré-ativação do neurônio artificial
k	Instante de amostragem
$e(k)$	Erro da saída do neurônio artificial no instante k
$\hat{e}(k)$	Erro estimado da saída do neurônio artificial no instante k
$y(k)$	Saída real da planta física no instante k
$\hat{y}(k)$	Saída da rede neural artificial no instante k
$u(k)$	Sinal de controle na entrada da planta física no instante k
$r(k)$	Sinal de referência no instante k
$E(n)$	Erro médio quadrado instantâneo
η	Parâmetro da taxa de aprendizado
β	Declividade ou coeficiente angular
$\gamma_j(n)$	Gradiente local
α	Constante de momento
NY	Horizonte de predição
NU	Horizonte de controle

$N1$	Horizonte mínimo de predição
J	Função objetivo
$\Delta u(k)$	Variação da ação de controle no instante k
λ	Ponderação à ação de controle
ρ	Passo de otimização
φ	Vetor regressor de entrada do modelo linear
$\hat{\theta}$	Vetor de parâmetros estimados
n	Ordem da planta
d	Atraso de transporte
n	Número de regressores referente ao sinal de saída
m	Número de regressores referente ao sinal de controle
w_1	Vetor peso das conexões entre a camada 1 e 2
w_2	Vetor peso das conexões entre a camada 2 e 3
$g(.)$	Função do comportamento dinâmico da planta
$Pnor$	Pontos normalizados
$Preal$	Ponto real (entradas e saídas)
$Pmin$	Valor mínimo de conjunto de dados
$Pmax$	Valor máximo de conjunto de dados
N	Número de neurônios na camada intermediária da RNA
b	Bias
Net_i	Saída estimada como função não linear
$\hat{Y}$	Vetor predito do sinal de saída
R	Vetor predito do sinal de referência
U	Vetor de ações de controle futuras

RNA	Rede Neural Artificial
MPC	Model Predictive Control
MPHC	Controle Preditivo Baseado em Modelo Heurístico
DMC	Controle por Matriz Dinâmica
MAC	Controle Baseado em Modelo
IMC	Controle com Modelo Interno
GPC	Controle Preditivo Generalizado
NNARX	Rede Neural Autoregressiva com Entradas Exógenas
NNFIR	Rede Neural com Resposta ao Impulso Finita
NNARMAX	Rede Neural Autoregressiva com Média Móvel e Entradas Exógenas
NNOE	Rede Neural com Erro na Saída
SISO	Entrada Única, Saída Única
MIMO	Entradas Múltiplas, Saídas Múltiplas

CAPÍTULO 1

INTRODUÇÃO

A cada dia é exigido um desempenho maior dos processos industriais, elevando, dessa forma, o nível da automação industrial. Porém, é importante que as operações se tornem mais seguras e estejam intimamente ligadas às questões ambientais e para que se possa atingir este patamar de qualidade, faz-se, ainda, necessário a utilização de novas técnicas de controle.

A análise e projeto dos sistemas de controle não-lineares têm recebido considerável atenção tanto no meio acadêmico como no industrial. Isto ocorre devido à insatisfação do desempenho de controladores lineares quando aplicados a plantas com acentuada não-linearidade ou plantas não-lineares atuando sobre uma ampla faixa de operação, (Santos, 2003).

Novas estratégias de identificação e controle para sistemas não-lineares são desenvolvidas devido aos avanços na teoria de sistemas não-lineares e também pelos novos métodos de identificação de modelos não-lineares.

Outro ponto que possibilita a implementação de novas técnicas é o grande avanço da tecnologia de computação digital. Com o desenvolvimento dos microprocessadores, que abre a possibilidade de se adotar estratégias mais elaboradas no projeto do controlador, tornando-os mais complexos e eficientes que os controladores PID tradicionais. Dessa forma, é viável a implementação digital dos controladores que utilizam modelos e algoritmos complexos.

Dentre as novas técnicas de controle, o controle preditivo baseado em modelo tem-se apresentado como uma das mais populares e eficientes estratégias de controle na indústria de processos.

De acordo com Santos (2003), isto ocorre porque muitos dos aspectos fundamentais em um projeto de controle industrial prático podem ser explorados em um controle preditivo baseado em modelo, como a trajetória de referência futura, previsão de perturbações e a

inclusão de restrições, verificando-se, assim, a flexibilidade de projeto desta técnica de controle.

Para atingir um desempenho superior do controle preditivo baseado em modelos lineares, é necessário representar o processo de forma mais fiel e eficiente. Desta forma, o controle preditivo não-linear se torna uma ferramenta de controle promissora para diversas áreas da engenharia, assim sua aplicação nas indústrias vem crescendo cada vez mais.

Com o intuito de representar um sistema da forma mais eficiente, técnicas lineares convencionais têm cedido cada vez mais espaço para técnicas não-lineares que conseguem capturar a dinâmica de sistemas complexos.

Desta forma, as aplicações de Redes Neurais Artificiais (RNAs) têm crescido rapidamente, com o objetivo de resolver problemas de predição e modelagem de sistemas. Tal fato deve-se à sua capacidade de aprender, com suficiente exatidão, o comportamento de um sistema, gerando modelos neurais para serem utilizados em projeto de controle não-linear.

As Redes Neurais Artificiais têm grandes aplicações em situações onde equações do modelo são desconhecidas ou somente informações parciais de estados do processo estão disponíveis.

Com uma rede neural devidamente treinada e representando um sistema de forma precisa é possível usar o modelo adquirido juntamente com o controle preditivo. Desta forma o modelo neural será capaz de realizar predições para que sejam geradas as leis de controle pelo otimizador, fazendo com que o sistema consiga atingir as referências desejadas.

Para analisar o desempenho do controlador preditivo neural é necessário realizar uma série de comparações com outros tipos de controladores, desta forma é possível analisar o seu desempenho e verificar as vantagens e desvantagens de sua utilização.

No capítulo 2, tem-se a teoria das Redes Neurais Artificiais, evidenciando seu funcionamento, métodos de treinamentos, arquiteturas existentes e uma revisão sobre o algoritmo de retropropagação do erro (*Error Backpropagation*).

No capítulo 3, procura-se dar mais ênfase ao controle preditivo, explicando a estrutura do controlador e caracterizando de forma abrangente algumas funções de otimização e regras usualmente aplicadas à atualização da ação de controle.

No capítulo 4, será realizada uma descrição do controle preditivo neural, explicando como as RNA's podem ser utilizadas para identificar processos, gerando um modelo neural e como é realizado o cálculo da ação de controle.

As aplicações do controlador e a descrição do processo ao qual ele é aplicado estão no capítulo 5. Ao final a conclusão e perspectivas futuras.

CAPÍTULO 2

REDES NEURAIS ARTIFICIAIS

2.1. Inteligência Computacional

A Inteligência Computacional envolve diversas áreas do conhecimento, tais como: ciência da computação, matemática, neurofisiologia, lingüística, dentre outras, com o objetivo de desenvolver sistemas capazes de imitar e/ou entender aspectos do pensamento humano.

As Redes Neurais Artificiais (RNA's) são sistemas computacionais com processamento altamente paralelo e distribuído, que apresentam a capacidade de aprender e armazenar conhecimento experimental.

O grande avanço na microinformática, constatado nas últimas décadas, vem possibilitando o desenvolvimento e aprimoramento de programas que procuram implementar as RNA's, ou seja, os princípios matemáticos nos quais se acredita estar baseada a inteligência humana. As RNA's são um campo com potencial elevado de aplicações, que ainda está em desenvolvimento.

A RNA que será empregada neste trabalho faz parte das técnicas de inteligência computacional que vêm sendo aplicadas a uma enorme gama de problemas com sucesso. As principais áreas de aplicação são: sistemas de controle, reconhecimento de padrões e identificação de funções. É necessário, no entanto esclarecer que as RNA's não são aplicadas a toda e qualquer tarefa computacional.

Dediquemo-nos agora a apresentar as Redes Neurais, tendo como intuito facilitar o entendimento das estratégias de identificação de sistemas.

2.2. Histórico das Redes Neurais Artificiais

As primeiras informações sobre as RNA's datam de 1943, com a publicação do artigo intitulado "*A Logical Calculus of the Ideas Immanent in Nevrous Activity*", cuja autoria é de Warren McCulloch e Walter Pitts. Esse artigo apresenta uma analogia entre células vivas e processos eletrônicos, construindo um modelo artificial do neurônio natural, onde esse possuía apenas uma saída, que era uma função de entrada da soma do valor de suas diversas entradas.

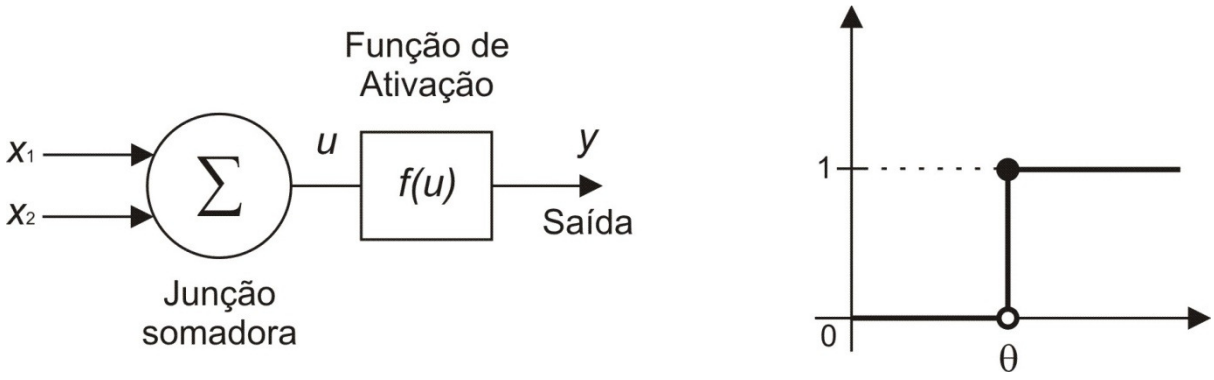

Figura 2.1. Modelo do neurônio de McCulloch-Pitts

Em 1949, Donald Hebb escreveu o livro "*The Organization of Behavior*", no qual ele demonstrou que a capacidade de aprendizagem das redes neurais vem da alteração sináptica, tornando-se assim o primeiro a propor uma lei de aprendizagem específica para as sinapses dos neurônios.

Já em 1958, Rosemblatt mostrou em seu livro, "*Principles of Neurodynamics*", o modelo dos *Perceptrons*, em que os neurônios estavam organizados em camadas de entrada e saída e os pesos das conexões eram adaptados com o intuito de se atingir a eficiência sináptica.

Acompanhando a evolução, em 1960 surgiu a rede com uma lei de aprendizado eficaz, denominada de *ADALINE* (*ADAptative LInear NEtwork*), e o *MADALINE* (*Many ADALINE*), propostas por Widrow e Hoff.

Os anos seguintes foram marcados por um entusiasmo exagerado de muitos pesquisadores, pois surgiram diversas publicações sobre máquinas tão poderosas quanto o cérebro humano, isso em um curto espaço de tempo.

Passados alguns anos, mais precisamente em 1969, Minsky e Papert mostraram no livro "*Perceptrons"*, como o *perceptron* com uma simples camada não poderia simular o comportamento de uma função *XOR* (ou-exclusivo). Além disso, generalizaram as limitações de um *perceptron* de única camada para sistemas de várias camadas. Após a publicação desta obra, iniciou o período chamado de "anos negros" das redes neurais.

A retomada das pesquisas na área das RNA's se deu com a publicação dos trabalhos de Hopfield em 1982, o qual descreveu a utilização de redes simétricas para otimização através de um algoritmo de aprendizagem que estabilizava uma rede binária simétrica com realimentação.

O *perceptron* foi validado com a elaboração do algoritmo de aprendizagem *backpropagation*, que possibilitou a implementação da terceira camada necessária para o aprendizado da função XOR. Fazendo uso da mesma rede de neurônios, como as utilizadas no *perceptron*, o *backpropagation* realiza o ajuste dos pesos sinápticos através da retropropagação do erro da camada de saída para as camadas anteriores.

O erro é o resultado da comparação entre a saída desejada (pré-definida) e a saída real da rede. Esta retropropagação possibilita a representação de funções não linearmente separáveis, permitindo o treinamento da função XOR.

2.3. Neurônio Artificial Clássico

O neurônio biológico é formado por um corpo celular que contém o núcleo da célula, por diversos dendritos, através dos quais impulsos elétricos são recebidos (receptor), e por um axônio, pelo qual impulsos elétricos são enviados (transmissor). O neurônio recebe os sinais de entrada através dos dendritos, processa-os no corpo celular, e transmite o resultado do processamento através do axônio. As conexões entre neurônios são efetuadas pelas sinapses, as quais são pontos de contato entre dendritos e axônios controlados por impulsos elétricos e reações químicas (Huamaní, 2003).

As redes neurais artificiais, como já citado anteriormente, são ferramentas computacionais pertencentes à abordagem conexionista da inteligência artificial. São formados por elementos de processamento, baseados no neurônio biológico e são denominados de neurônios artificiais.

Os neurônios artificiais são emulações dos neurônios biológicos, os quais recebem informações de sensores ou de outros neurônios artificiais, produzindo operações sobre estes dados, que passam o resultado para outros neurônios artificiais (Ender, 2002).

O primeiro modelo do neurônio artificial, proposto por McCulloch e Pitts tinha grandes limitações. Entre elas, destaca-se a sua saída binária. O funcionamento deste modelo pode ser descrito da seguinte maneira: se a soma ponderada dos sinais de entrada de um neurônio ultrapassar um determinado limiar a, então a saída $y = f(u)$ recebe valor 1 (um); se não, recebe valor 0 (zero). As entradas x_j do neurônio também são binárias.

O modelo atual do neurônio artificial considera um vetor de entrada X, não necessariamente binário, um operador de agregação genérico, vetor de saída Y, uma matriz de pesos sinápticos W e uma função de ativação $f(\cdot)$, que pode se apresentar de diversas formas.

Cada componente do vetor de entrada está ligado ao neurônio através de conexões, cujas intensidades são representadas pelos pesos sinápticos *W*. Usualmente o neurônio efetua uma soma ponderada entre os componentes do vetor de entrada e o vetor peso, obtendo assim a correspondente saída *Y*.

Há uma entrada fixa para um peso associado a cada neurônio denominado de bias, que é o limiar de ativação introduzido e tem o efeito de flexibilizar a ação da função de ativação, deslocando-a no plano x-y. A correspondente saída do neurônio é representada por:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + w_0 x_0\right) \qquad 2.1$$

onde $f(\cdot)$ é a função de ativação, w_0 é o peso sináptico correspondente à entrada fixa de polarização x_0. Usualmente utiliza-se $x_0 = +1$ ou $x_0 = -1$.

2.3.1. Funções de Ativação

Segundo McCulloch e Pitts (1943), um neurônio é ativado quando um conjunto de entradas lhe é aplicado e a função de ativação superar o valor do limiar. A forma da resposta é determinada pela função de ativação. Ela é utilizada para limitar ou gerar a amplitude de saída do sinal do neurônio, normalmente fornecendo uma característica não-linear ao neurônio, conforme a função de ativação escolhida pelo projetista. Existe uma série de funções de ativação possíveis para serem aplicadas nas redes neurais, para o desenvolvimento deste trabalho foram utilizadas as funções: linear, sigmóide unipolar; e sigmóide bipolar.

- Função Linear:

A expressão que representa esta função é dada a seguir:

$$f(NET_j) = \beta NET_j \qquad 2.2$$

onde β é uma constante de declividade da reta.

Sua derivada primeira é dada por:

$$f'(NET_j) = \beta \tag{2.3}$$

O gráfico da função linear pode ser observado na Figura 2.2.

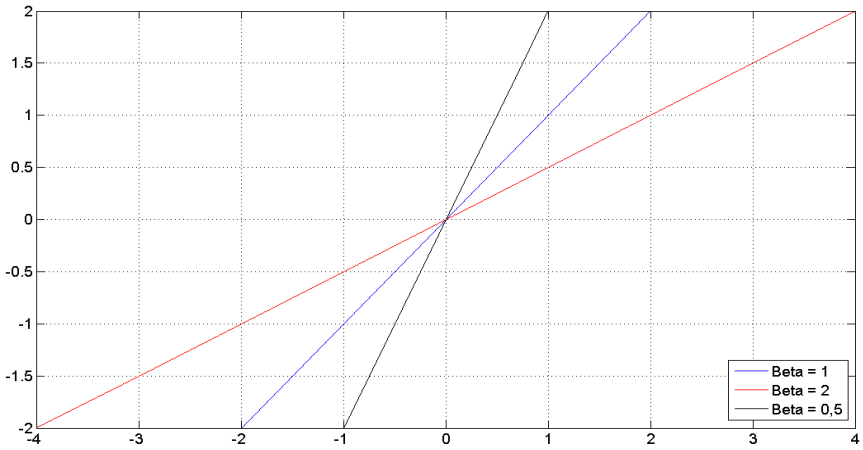

Figura 2.2. Função Linear

- Função Sigmóide Unipolar:

É o tipo de função mais utilizada no desenvolvimento das RNA's. Essa função permite que o mapeamento entrada-saída possa assumir características não-lineares. A sua expressão é expressa por:

$$f(NET_j) = \frac{1}{1 + e^{-\beta NET_j}} \tag{2.4}$$

A derivada primeira dessa função é dada pela equação 2.5.

$$f'(NET_j) = \beta f(NET_j)\left[1 - f(NET_j)\right] \tag{2.5}$$

A figura 2.3 representa o gráfico da função sigmóide unipolar.

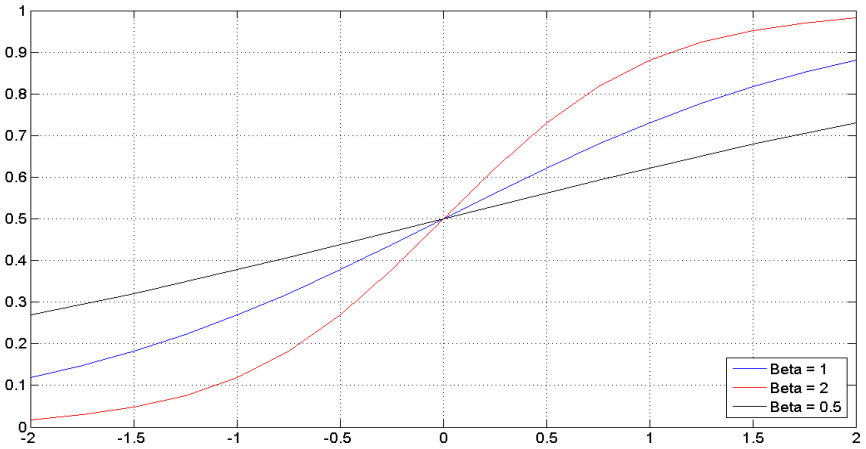

Figura 2.3. Função Sigmóide Unipolar

- Função Sigmóide Bipolar:

Esta função possui características semelhantes com a sigmóide unipolar, também denominada de Função Tangente Hiperbólica. Da mesma forma que a Função Sigmóide, ela permite um mapeamento entrada-saída com características não-lineares. Sua expressão é expressa por:

$$f(NET_j) = \frac{1 - e^{-\beta NET_j}}{1 + e^{-\beta NET_j}} \tag{2.6}$$

A derivada primeira da Função Sigmóide Bipolar é dada por:

$$f'(NET_j) = \beta \left[1 - \left(f(NET_j)\right)^2\right] \tag{2.7}$$

A figura 2.4 representa o gráfico da função sigmóide bipolar.

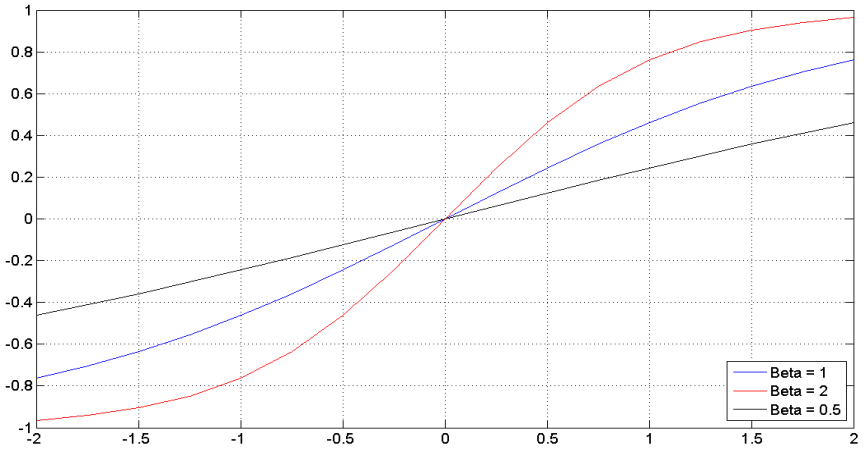

Figura 2.4. Função Sigmóide Bipolar

Segundo Rezende (1999), é bastante comum encontrar RNA´s onde os neurônios das camadas internas possuem funções sigmóides e nas camadas de saída funções lineares, com isso é possível preservar a não-linearidade da rede e é possível ter saídas com valores absolutos maiores do que um.

2.4. Métodos de Treinamento

Define-se aprendizado ou treinamento de redes neurais como o processo pelo qual os parâmetros livres ou adaptáveis da rede são ajustados via um processo contínuo de simulação do ambiente no qual a rede é inserida. Este processo pode ser classificado dentro de dois grandes grupos: Treinamento supervisionado e Treinamento não supervisionado (Maitelli e Gabriel, 2003).

2.4.1. Treinamento supervisionado

O treinamento supervisionado caracteriza-se pela disponibilidade de conhecimento ou informação sobre um sistema na forma de padrões entrada- saída (Haykin, 1994), formando um conjunto de treinamento.

Assim, para uma dada entrada a rede neural proporciona uma saída que é comparada a saída desejada do conjunto de treinamento. O treinamento da rede neural baseia-se no ajuste dos pesos sinápticos que é feito através do cálculo do erro da saída gerada pela rede e a saída desejada. Os ajustes são feitos de forma iterativa e prossegue até o erro atingir uma tolerância mínima previamente definida.

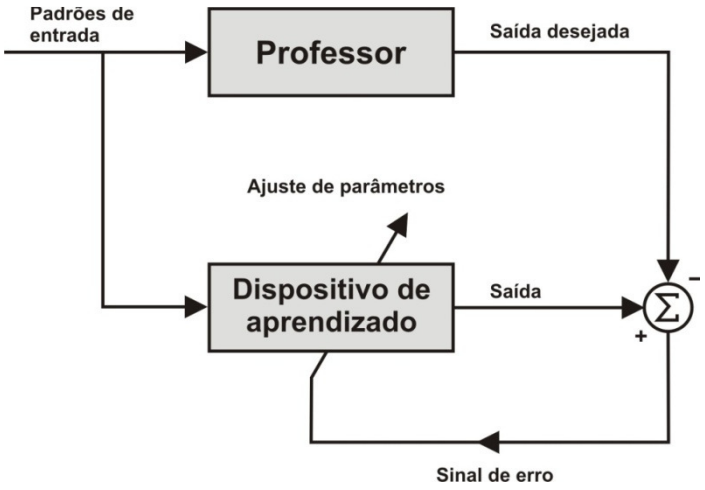

Figura 2.5. Treinamento supervisionado

2.4.2. Treinamento não-supervisionado

No treinamento não supervisionado ou auto-organizado, não existe um elemento externo que inspecione o processo, ou seja, não existe um conjunto específico de padrões a serem aprendidos pela rede neural, mas sim, uma medida da qualidade de representação que a rede requer para o aprendizado, o processo de aprendizagem da RNA é autônomo.

Os parâmetros da rede neural são ajustados em função desta medida. Uma vez que a rede consegue extrair algumas características dos dados de entrada, esta será capaz de gerar representações internas para classificar as características. Dentre as aplicações mais usuais destaca-se o reconhecimento de padrões.

Figura 2.6. Treinamento não supervisionado

2.5. Arquitetura da Rede

A arquitetura de uma rede neural é determinada pela forma em que são definidas as conexões entre os neurônios. Basicamente, uma rede é dividida em camadas de neurônios; geralmente, possuem uma camada de entrada, camadas intermediárias e uma camada de saída; em alguns casos, não existem camadas intermediárias. Podemos identificar três classes de arquitetura de redes neurais: redes *feedforward* de uma única camada, redes *feedforward* de múltiplas camadas e redes recorrentes.

2.5.1. Redes Neurais Feedforward de uma única entrada

Esta rede é formada apenas por uma camada de entrada e uma camada de saída. A propagação do sinal é dada no sentido da camada de entrada para a camada de saída, não ocorrendo ciclos. Esta rede é estritamente do tipo *feedforward* ou acíclica. Todo o processamento ocorre nos neurônios da camada de saída. Podendo ser utilizada para resolver problemas linearmente separáveis (Haykin, 1994).

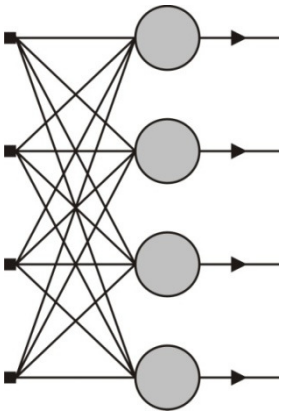

Figura 2.7. Redes Neurais Feedforward de uma única entrada

2.5.2. Redes Feedforward de Múltiplas Camadas

As redes *feedforward* de múltiplas camadas se distinguem da anterior pela presença de uma ou mais camadas ocultas ou intermediárias. As camadas intermediárias têm como função intervir entre a entrada e a saída da rede. A habilidade de os neurônios ocultos extraírem estatísticas de ordem elevada é particularmente valiosa quando o tamanho da camada de entrada é grande. Os sinais de saída de uma camada são utilizados como entradas da camada seguinte. Para esse trabalho foi utilizada uma rede *feedforward* de múltiplas camadas, sendo a rede também denominada de Perceptrons de Múltiplas Camadas, sendo que seu funcionamento será melhor explicado no seção 2.6.

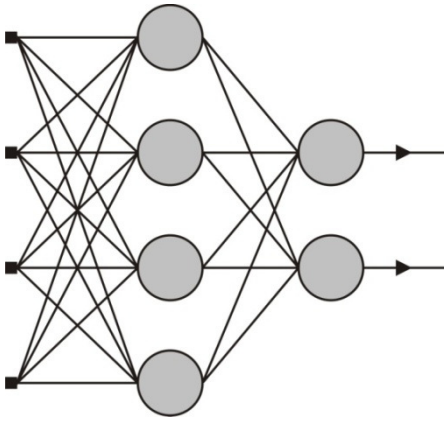

Figura 2.8. Redes Neurais Feedforward de múltiplas camadas

2.5.3. Redes Recorrentes

A rede neural recorrente tem por base uma rede *feedforward* onde é introduzido em sua estrutura pelo menos um laço de realimentação, ampliando sua potencialidade de modelagem de dados temporais ou espaciais. As realimentações consistem em saídas de neurônios de determinada camada que serão reintroduzidas como entradas de neurônios de camadas anteriores ou da própria, ou seja, um neurônio pode ser realimentado por sua própria

saída. Essas possibilidades fazem com que a arquitetura de uma rede recorrente possa tomar diversas formas (Gomes, 2005).

Os laços de realimentação envolvem o uso de ramos particulares compostos de elementos de atraso unitário, o que resulta em um comportamento dinâmico não-linear, admitindo-se que a rede neural contenha unidades não-lineares.

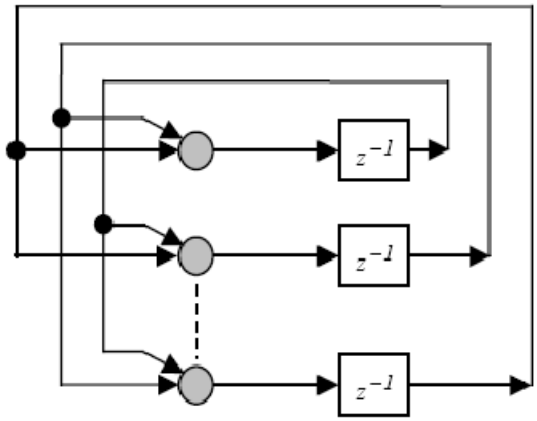

Figura 2.9. Redes Neurais Recorrentes

2.6. Perceptrons de Múltiplas Camadas

As redes neurais multicamadas tipicamente são formadas por: um conjunto de entrada, uma ou mais camadas ocultas e uma camada de saída.

Um aspecto a ser considerado é a escolha do número de camadas da rede neural artificial. Uma função contínua pode ser aproximada para qualquer grau de precisão usando uma rede neural, com três camadas, desde que haja um número suficiente de neurônios ativos na camada oculta (Ender, 2002).

Entretanto, para melhorar a exatidão dos resultados, normalmente aumenta-se o número de camadas internas, aumentando assim a complexidade da rede neural e por conseqüência, o esforço computacional. O principal problema decorrente deste procedimento

é que a rede neural com grande número de pesos tem uma forte tendência em representar bem os padrões de aprendizagem e ter um desempenho não apropriado com dados não usados na aprendizagem.

O treinamento de uma rede tem por objetivo ajustar seus pesos, de modo que ao aplicar um conjunto de dados de entrada seja produzido um conjunto desejado de saída. É necessário que cada valor de entrada esteja relacionado com um valor de saída, gerando os pares de treinamento. O ajuste dos pesos é feito através de algoritmos de aprendizagem. Para o desenvolvimento do trabalho foi utilizado o Algoritmo de Retropropagação do Erro (*Error Backpropagation*).

2.6.1. Algoritmo de Retropropagação do Erro (Backpropagation)

O algoritmo de aprendizagem por retropropagação do erro é um método de ajuste dos pesos sinápticos de uma rede multicamadas. Possui duas etapas distintas: primeira, quando é apresentado um padrão de entrada à rede e o fluxo é alimentado até a camada de saída (*forward*). Assim a saída estimada é comparada com a saída desejada, se a saída da rede não corresponder com a desejada, dentro de uma determinada exatidão, é feita uma correção nos pesos das conexões sinápticas, ajustando-os na direção oposta do gradiente do erro instantâneo. O ajuste é proporcional ao gradiente, segundo uma taxa de aprendizagem e é realizado da última camada em direção à camada de entrada (*backward*).

Esse algoritmo tem como objetivo minimizar a soma dos erros quadráticos e assim ajustar os pesos sinápticos da rede neural. O sinal de erro na saída do neurônio *j*, na iteração *n* é definido por:

$$e_j(n) = y_j(n) - \hat{y}_j(n) \qquad 2.8$$

onde *y* é a saída desejada e $\hat{y}$ a saída obtida da rede neural.

O funcional dos erros médios quadrados instantâneos (*E(n)*) é obtido somando-se os termos $\frac{1}{2}e_j^2(n)$ de todos os neurônios da camada de saída. Assim podemos escrever:

$$\mathrm{E(n)} = \frac{1}{2}\sum_{\mathrm{j}\in\mathrm{C}} \mathrm{e_j^2(n)} \tag{2.9}$$

em que o conjunto *C* inclui todos os neurônios da camada de saída da rede. A energia média dos erros médios quadrados instantâneos é obtida somando-se todos os *E(n)* para todas as *n* interações e dividindo o resultado pelo o número total de exemplos contidos no conjunto de treinamento, definido por *N*. Assim temos:

$$\mathrm{E_{med}} = \frac{1}{\mathrm{N}}\sum_{\mathrm{n}=1}^{\mathrm{N}} \mathrm{E(n)} \tag{2.10}$$

A função de ativação interna associada ao neurônio *j* pode ser definida da seguinte forma:

$$\mathrm{NET_j(n)} = \sum_{\mathrm{i}-0}^{\mathrm{m}} \mathrm{w_{ij}(n)y_i(n)} \tag{2.11}$$

onde *m* é o número total de entradas (excluindo o bias) aplicadas ao neurônio *j*. E o sinal de saída desse mesmo neurônio *j* é expresso da seguinte forma:

$$\mathrm{y_j(n)} = \mathrm{f_j(NET_j(n))} \tag{2.12}$$

Como foi dito anteriormente o objetivo do treinamento é minimizar *E(n)* em função do ajuste dos ganhos sinápticos $w_{ij}(n)$. Para isso utiliza-se o procedimento de minimização por gradiente descendente, percorrendo a rede da saída para a entrada, onde a correção nos pesos é definida pela "regra delta". A partir do Método do Gradiente, temos:

$$\mathrm{w_{ji}(n+1)} = \mathrm{w_{ji}(n)} + \Delta\mathrm{w_{ji}(n)} = \mathrm{w_{ji}(n)} - \eta\mathrm{(n)}\frac{\partial \mathrm{E(n)}}{\partial \mathrm{w_{ji}(n)}} \tag{2.13}$$

onde η é o parâmetro da taxa de aprendizado do algoritmo de retropropagação.

Pela regra da cadeia, temos:

$$\frac{\partial \mathrm{E(n)}}{\partial \mathrm{w_{ji}(n)}} = \frac{\partial \mathrm{E(n)}}{\partial \mathrm{e_j(n)}} \frac{\partial \mathrm{e_j(n)}}{\partial \mathrm{y_j(n)}} \frac{\partial \mathrm{y_j(n)}}{\partial \mathrm{v_j(n)}} \frac{\partial \mathrm{v_j(n)}}{\partial \mathrm{w_{ji}(n)}} \tag{2.14}$$

Com

$$\frac{\partial \mathrm{E(n)}}{\partial \mathrm{e_j(n)}} = \frac{\partial \mathrm{E}}{\partial \mathrm{e_j(n)}} \left(\sum_{\mathrm{j \in C}} \mathrm{e_j^2(n)} \right) = \mathrm{e_j(n)} \tag{2.15}$$

$$\frac{\partial \mathrm{e_j(n)}}{\partial \mathrm{y_j(n)}} = \frac{\partial}{\partial \mathrm{y_j(n)}} \big(\mathrm{d_j(n) - y_j(n)} \big) = -1 \tag{2.16}$$

$$\frac{\partial \mathrm{y_j(n)}}{\partial \mathrm{v_j(n)}} = \frac{\partial}{\partial \mathrm{v_j(n)}} \big(\mathrm{f_j(NET_j(n))} \big) = \mathrm{f_j'(NET_j(n))} \tag{2.17}$$

$$\frac{\partial \mathrm{NET_j(n)}}{\partial \mathrm{w_{ji}(n)}} = \frac{\partial}{\partial \mathrm{w_{ji}(n)}} \left(\sum_{\mathrm{i=1}}^{\mathrm{m}} \mathrm{w_{ji}(n) y_i(n)} \right) = \mathrm{y_i(n)} \tag{2.18}$$

Assim,

$$\frac{\partial \mathrm{E(n)}}{\partial \mathrm{w_{ji}(n)}} = -\mathrm{e_j(n) f_j'(NET_j(n)) y_i(n)} \tag{2.19}$$

e,

$$\Delta \mathrm{w_{ji}(n)} = -\eta \mathrm{(n)} \frac{\partial \mathrm{E(n)}}{\partial \mathrm{w_{ji}(n)}} = \eta \mathrm{(n) \gamma_j(n) y_i(n)} \tag{2.20}$$

A variável $\gamma_j(n)$ é o gradiente local. O cálculo desse gradiente local depende da camada em que se encontra o neurônio. Existe um cálculo para neurônios da camada de saída e outro para neurônios das demais camadas, conforme pode ser observado abaixo:

- O neurônio *j* sendo um nó de saída:

O gradiente local $\gamma_j(n)$ para o neurônio *j* é igual ao produto do sinal do erro $e_j(n)$ correspondente para aquele neurônio pela derivada $f_j'(v_j(n))$ da função de ativação associada (Haykin,1994)

$$\gamma_j(n) = -e_j(n) f_j'(NET_j(n)) \quad 2.21$$

- O neurônio j sendo um nó oculto:

O gradiente local é igual ao produto da derivada associada $f_j'(NET_j(n))$ pela média ponderada dos *γs* calculados para os neurônios na próxima camada, representada por *k*.

$$\gamma_j(n) = f_j'(NET_j(n)) \sum_k \gamma_k(n)\, w_{kj}(n) \quad 2.22$$

2.7. Considerações para o Treinamento

2.7.1. Taxa de Aprendizagem

De acordo com Huamaní (2003), o algoritmo de retropropagação fornece uma aproximação da trajetória no espaço dos pesos em direção ao mínimo da função objetivo (função do erro médio quadrático) computada via o método do gradiente descendente.

Ao diminuir o parâmetro da taxa de aprendizagem (η), as variações dos pesos sinápticos da rede, de uma iteração para outra serão menores e a trajetória no espaço de pesos será mais suave. Porém, se esse parâmetro for muito grande, poderá ocasionar grandes modificações nos pesos sinápticos podendo tornar a rede instável.

Um método de alterar a taxa de aprendizagem sem tornar a rede instável é incluir um termo de momento na regra delta.

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta(n) \gamma_j(n) y_i(n) \quad 2.23$$

onde α é chamado de constante de momento.

A utilização do momento no algoritmo de retropropagação ocasiona uma pequena modificação na atualização dos pesos, porém pode trazer benefícios no comportamento de

aprendizagem do algoritmo e também pode evitar que o processo de aprendizagem termine em um mínimo local na superfície de erro (Haykin, 1994).

Para o desenvolvimento da RNA é utilizada a técnica do η-adaptativo ou η-variável, sugerida em Gabriel (2004). Este artifício também pode ser utilizado para aumentar a velocidade de convergência das redes neurais. Ele consiste em variar a constante de aprendizagem durante a etapa de treinamento da rede, buscando otimizar o precesso de adaptação dos pesos sinápticos, como também para evitar a sua parada em um mínimo local.

2.7.2. Modos de Treinamento

Quando todo o conjunto de treinamento é apresentado a uma rede neural, obtém-se o que se denomina época. É recomendável tornar aleatória a ordem de apresentação dos exemplos de treinamento para cada época. Isto faz que com que a busca no espaço de pesos seja estocástica. Para um dado conjunto de treinamento, a aprendizagem por retropropagação pode ser aplicado de dois métodos diferentes:

- Modo seqüencial:

No aprendizado por retropropagação padrão, os pesos da rede são atualizados após a apresentação de cada exemplo de treinamento, ou seja, apenas o gradiente do correspondente padrão é calculado e utilizado para atualizar os pesos. Assim, por cada época, os pesos são ajustados N vezes.

- Modo por lote:

Neste modo de aprendizagem, o ajuste dos pesos é realizado após a apresentação de todo o conjunto de treinamento. Os erros são calculados para todos os padrões de treinamento

e a soma dos gradientes para cada padrão apresentado na época será utilizado para ajustar os pesos.

2.7.3. Critérios de Parada

Não existem critérios de parada recomendados para garantir que o algoritmo de retropropagação tenha convergido. Porém existem critérios que podem ser utilizados para encerrar o processo de treinamento. Dois critérios de parada podem ser utilizados:

- Considerando as propriedades de um mínimo local ou global da superfície de erro. Supondo que **w*** seja um mínimo local ou global, o vetor gradiente calculado em relação ao vetor de peso **w**, será nulo quando **w = w***. Assim, a convergência é obtida quando a norma Euclidiana do vetor gradiente assuma um valor suficientemente pequeno.
- O segundo critério de parada diz que o algoritmo de retropropagação converge quando a variação do erro médio quadrático por época seja menor do que uma tolerância previamente especificada.

Neste capítulo foram apresentados alguns fundamentos sobre Redes Neurais Artificiais, com ênfase nas diversas formas de arquiteturas existentes e no desenvolvimento do algoritmo de retropropagação do erro. Estes conceitos são importantes para o desenvolvimento de uma rede neural capaz de identificar sistemas, como é mostrado no capítulo 4.

CAPÍTULO 3

CONTROLE PREDITIVO

3.1. Controlador Preditivo

Este capítulo tem como objetivo explicar de forma geral o funcionamento e a estrutura do controlador preditivo, visando um entendimento inicial dessa ferramenta para que seja possível sua aplicação na área desejada.

Atualmente umas das mais populares e eficientes estratégias de controle na indústria é o controle preditivo baseado em modelo (MPC – *Model Predictive Control*). A sua utilização nesse trabalho é justificada pela sua capacidade de operação por grandes períodos de tempo sem intervenções freqüentes de especialistas. Muitos dos aspectos fundamentais em um projeto de controle industrial prático podem ser explorados em um controle preditivo baseado em modelo, como a trajetória de referência futura, previsão de perturbações e a possibilidade de inclusão de restrições, verificando a flexibilidade desta técnica de controle (Ogunnaike e Ray, 1994; Zambrano e Camacho, 2002).

O controle preditivo foi idealizado inicialmente para aplicações em sistemas de potência e na indústria petrolífera, porém, nos dias atuais esta ferramenta é empregada nas mais diversas áreas: regulação de tensão, controle de temperatura, pressão, nível e outros. Isto mostra a evolução prática destas estratégias e comprovando que em breve devem substituir a maioria dos controladores clássicos utilizados, que muitas vezes mostram-se ineficientes em ambientes complexos (Kwok, 1994; Rawlings, 2000).

O Controle Preditivo é assim denominado devido à forma como é realizado o cálculo da lei de controle. Este se baseia na predição do comportamento futuro do processo a ser controlado, através do uso de um modelo do mesmo. Dessa forma, a ação de controle é calculada através dos valores futuros preditos das variáveis controladas. Segundo Maitelli e Fontes (2005), os controladores preditivos comparados a outros métodos de controle, têm-se mostrado intrinsecamente robustos com relação a erros de modelagem.

3.2. Evolução Histórica

Estudos com relação ao controle preditivo foram iniciados no final da década de 50, mais precisamente em 1959 com Smith, que propôs um controlador baseado em um modelo de predição e uma lei de controle clássica.

Porém, o conceito dessa técnica de controle foi inicialmente desenvolvido de forma simultânea por Richalet, em 1978 e por Cutler e Ramaker, em 1979. Richalet apresentou um algoritmo que utiliza um modelo heurístico de resposta ao impulso para calcular a ação de controle, denominado Controle Preditivo Baseado em Modelo Heurístico (MPHC). Cutler propôs um controlador baseado na resposta ao degrau e foi denominado de Controle por Matriz Dinâmica (DMC). Depois disto vários outros métodos similares foram propostos.

A idéia básica do método é utilizar a resposta no domínio do tempo com uma perturbação degrau, calculando assim as futuras mudanças na variável manipulada, que irão minimizar um índice de desempenho.

Em 1982, Rouhani e Mehra desenvolveram o Algoritmo de Controle Baseado em Modelo (MAC). Garcia e Morari (1982) mostraram, no caso monovariável que vários esquemas de controle e o MPC possuem uma estrutura comum, denominada Controle com Modelo Interno (IMC).

Clarke em 1987 desenvolveu o Controle Preditivo Generalizado (GPC), em que preferiu utilizar um modelo paramétrico de entrada-saída, incluindo uma modelagem estruturada do ruído inserido na medição, o que o torna mais robusto a erros de modelagem, além do conceito do horizonte estendido de controle.

3.3. Estrutura do Controlador

Controle Preditivo Baseado em Modelo (MPC) é uma modalidade de controle, inerentemente discreta no tempo, que faz o uso de um modelo do processo a ser controlado para calcular as saídas do controlador. Em cada instante de tempo, este modelo é utilizado para predizer valores futuros das saídas do sistema, levando-as a seguir uma dada trajetória de referência. Essas predições são utilizadas na formulação de um problema de otimização, cuja solução determina a seqüência de entradas a serem aplicadas à planta de forma a minimizar uma função objetivo pré-determinada.

De acordo com Sansevero (2006), em geral esta função leva em conta tanto os custos associados às diferenças entre as saídas do sistema e suas referências quanto as associadas ao esforço de controle.

O conceito do algoritmo é representado pela figura abaixo, onde esta ilustra de maneira esquemática o seu funcionamento para o caso SISO (uma entrada e uma saída), facilitando o seu entendimento.

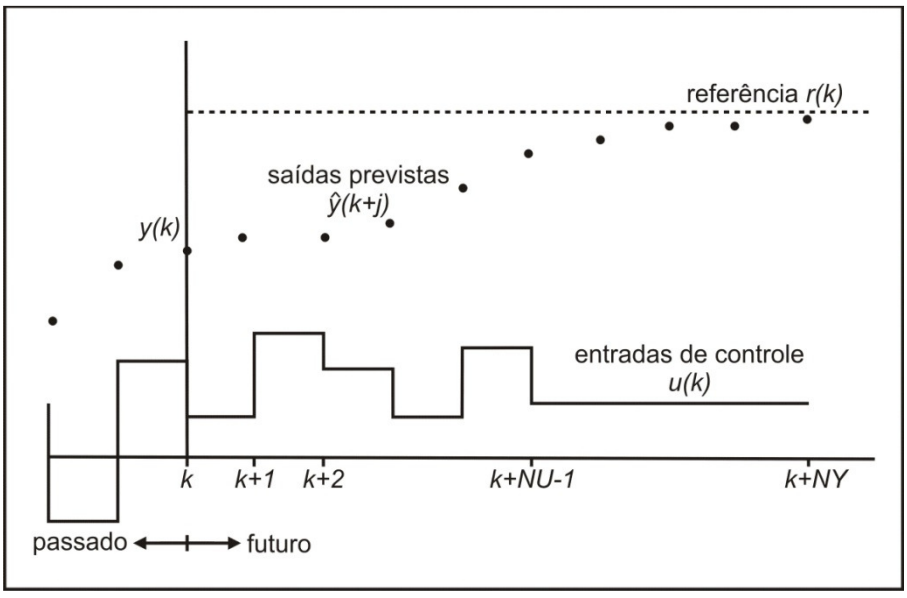

Figura 3.1. Controle Preditivo baseado em modelo

As variáveis *u(k)*, *ŷ(k)* e *r(k)* representam, respectivamente, os valores no instante atual k da variável manipulada, estimativa da variável controlada e do sinal de referência. Os valores atuais e futuros dessas variáveis são definidos pelos seguintes vetores:

$$\begin{cases} u = [u(k) \ \cdots \ u(k+NU-1)]^T \\ \hat{y} = [\hat{y}(k+1) \ \cdots \ \hat{y}(k+NY)]^T \\ r = [r(k+1) \ \cdots \ r(k+NY)]^T \end{cases} \quad 3.1$$

O parâmetro *NY* que determina o número de passos a frente a serem considerados na otimização é chamado de horizonte de predição. O parâmetro *NU* determina quantas vezes à variável de controle pode variar durante o horizonte de predição, sendo chamado de horizonte de controle. Obviamente, *NU* deve ser menor ou igual a *NY*.

No instante de tempo *k*, para qualquer estado presente e qualquer seqüência conhecida de entradas de controle, as saídas do sistema podem ser previstas utilizando-se o modelo do processo disponível. Em particular, estas previsões podem ser realizados ao longo de um horizonte de *NY* amostras de tempo, utilizando-se como entradas o estado atual e as entradas passadas, combinadas a um conjunto de *NU* entradas de controle futuras.

O algoritmo baseia-se no "Princípio do Horizonte Móvel", onde uma sequência de *NU* entradas de controle é obtida, mas somente o primeiro valor da seqüência é utilizado. Este processo é repetido no instante de tempo seguinte, utilizando as informações medidas mais recentes. O primeiro elemento da seqüência de controle obtida é aplicado ao processo, sendo desconsiderados os demais.

Utiliza-se uma função objetivo com o intuito de se quantificar o rastreamento da saída predita do processo em relação à trajetória de referência. Um exemplo simples de uma função objetivo pode ser vista abaixo. É importante ressaltar que a função relaciona as variáveis *y*, *u* e *r*.

$$J = \sum_{j=N1}^{NY} [\hat{y}(k+j) - r(k+j)]^2 + \lambda \sum_{i=1}^{NU} [u(k+i)]^2 \qquad 3.2$$

Contudo, resolver o problema expresso pela equação acima não é uma tarefa simples, em geral, pois é necessário resolver simultaneamente dois problemas. Primeiro, uma simulação para obter os valores previstos para as saídas do sistema. E posteriormente, calcular uma seqüência de ações de controle, que caracteriza um problema de minimização.

É importante ressaltar que a função objetivo ideal deve ser baseada em especificações de projeto, tais como: tempo de estabilização; *overshoot*; tempo de subida; margem de ganho e entre outros.

Desta forma, o problema de otimização resultante se tornaria de difícil solução. Por isso na maioria dos controladores preditivos utiliza-se como função objetivo uma função quadrática. Com isso as especificações de projeto devem ser transladadas para os parâmetros da função objetivo quadrática, para que as especificações mencionadas acima sejam atendidas no momento em que esta função estiver minimizada.

A estratégia de Controle Preditivo pode ser implementada pela estrutura básica da figura 3.2, onde os valores futuros das saídas da planta são preditos com base em valores presentes e passados e ainda nas futuras ações de controle ótimas propostas. Ações estas calculadas por um otimizador, levando em conta uma função objetivo que considera o erro de rastreamento futuro e restrições.

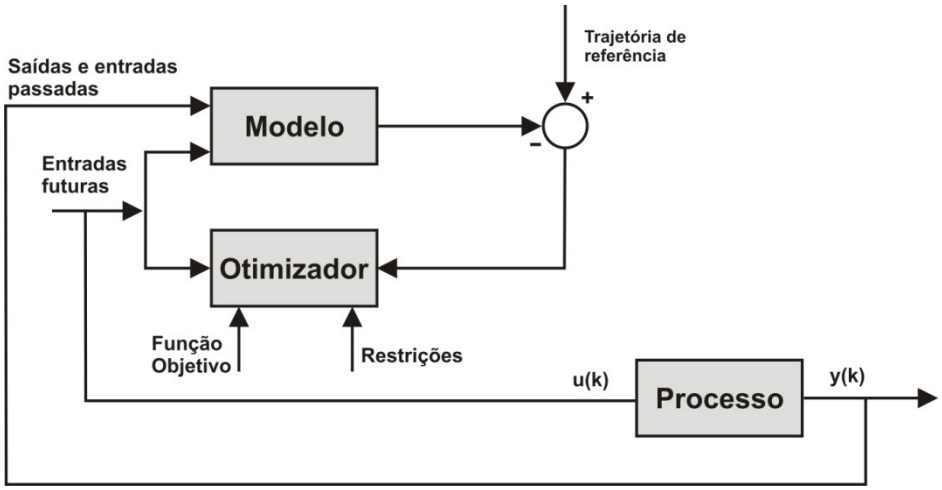

Figura 3.2. Estrutura básica de controle preditivo

3.4. Função Objetivo

A Função Objetivo, representada pelo índice *J*, é minimizada com o intuito de produzir uma lei de controle, que vai possuir características de acordo com a função escolhida.

A minimização dessa função gera uma seqüência de ações de controle em um determinado horizonte de predição. A seqüência de ações de controle em um determinado horizonte de predição é gerada a partir da minimização dessa função em relação a *u*. Essa seqüência deverá ser ótima, em relação à função objetivo.

Assim, os valores futuros da diferença entre a saída (*y*) e a referência (*r*) são minimizados, fazendo com que o processo acompanhe a trajetória de referência nos instantes de amostragem. Porém isso só ocorrerá se o modelo do processo utilizado representar de forma precisa o sistema. E também é importante lembrar que o sistema não deve estar sujeito a distúrbios e nem restrições. Se a seqüência de controle não conseguir fazer com que o processo atinja a trajetória de referência, haverá um erro, compensado no instante seguinte por um novo cálculo.

O exemplo de minimização do erro mais simples e usual de uma função objetivo é quando ela é realizada entre a saída da planta e o valor desejado. É expressa na forma:

$$J = y(k) - \hat{y}(k) = e(k) \tag{3.3}$$

onde:

y(k) – é a saída real da planta;

ŷ(k) – representa a saída estimada pelo modelo;

e(k) – é o erro de estimação;

k – é o instante de amostragem.

Outra função bastante usual e robusta para diversas aplicações se baseia no erro quadrático.

$$J = [y(k+1) - \hat{y}(k+1)]^2 = e(k+1)^2 \tag{3.4}$$

A expressão dada é denominada de Função Objetivo de Passo Único, pois é utilizada apenas uma previsão. Porém os controladores preditivos possuem modelos capazes de predizer *NY* passos à frente. Ao minimizar todo um vetor de erros preditos e não se restringir apenas a minimização de um único ponto é possível representa toda a trajetória das ações de controle futuras, no horizonte de predição. Podendo originar um melhor desempenho do sistema. A expressão que representa essa função pode ser observada abaixo.

$$J = \sum_{j=1}^{NY} [y(k+j) - \hat{y}(k+j)]^2 = \sum_{j=1}^{NY} e(k+j)^2 \tag{3.5}$$

Outro tipo de função objetivo mais complexa pode ser utilizado. Essa função além de minimizar o erro realiza uma ponderação do esforço de controle, garantindo assim que a lei de controle gerada apresente erro de regime nulo, mesmo não apresentando um ou mais integradores (Schnitman, 1998). É o caso do GPC (*Generalized Predictive Control*), que utiliza esse método. Essa função objetivo pode ser expressa da seguinte forma:

$$J = \sum_{j=N_1}^{NY} [r(k+j) - \hat{y}(k+j)]^2 + \sum_{i=1}^{NU} \lambda(j)[\Delta u(k+i)]^2 \qquad 3.6$$

em que:

r(k) – é sinal que representa a trajetória de referência;

Δu – é a variação da ação de controle, definida como u(k+i) – u(k+i-1);

λ – é a ponderação à ação de controle;

N_1 – é o horizonte mínimo de predição;

NU – é o horizonte de controle;

NY – é o horizonte máximo de predição.

A função objetivo pode estar sujeita a algumas restrições. Quando a ação de controle calculada excede o limite superior admissível ou o limite inferior pode ser utilizado às seguintes restrições:

$$\begin{gathered} y_{min} \leq \hat{y}(k+j) \leq y_{max}, j = 1, \dots, NY \\ u_{min} \leq u(k+j) \leq u_{max}, j = 1, \dots, NU \\ [u(k+j) - u(k+j+1)] \leq \Delta u_{max}, j = 1, \dots, NU \end{gathered} \qquad 3.7$$

3.5. Regras de atualização das ações de controle

Os controladores preditivos baseiam-se em regras de atualização das ações do controlador. Essas regras são expressões que calculam a variação da ação de controle atual em função dos índices a serem minimizados.

Os métodos de otimização podem ser classificados basicamente em relação à forma de utilização da informação sobre as derivadas das funções que são ou não avaliadas no método. Métodos de busca, que utilizam somente informação sobre a função, são mais adequados quando a função possui não linearidades e descontinuidades. Métodos que utilizam gradiente são mais eficientes quando a função possui primeira derivada contínua. Métodos de ordem superior, como o de Newton, utilizam informações de segunda derivada da função, a qual, quando numericamente calculada é de carga computacionalmente elevada.

Inúmeras regras podem ser utilizadas, porém nesse trabalho será feito uso da regra do gradiente descendente, a mesma base aplicada na regra delta para o treinamento da rede neural, mostrado capítulo 2. De acordo com Schnitman (1998), a atualização é realizada na direção oposta ao gradiente da função, procurando sempre o ponto de mínimo. A atualização da ação de controle é expressa da seguinte forma:

$$u(k+1) = u(k) - \rho \frac{\partial J}{\partial u(k)} \tag{3.8}$$

$$\Delta u(k+1) = -\rho \frac{\partial J}{\partial u(k)} \tag{3.9}$$

em que,

u(k) – é o valor da ação de controle no instante k;

Δ*u* – é a variação da ação de controle, definida como u(k+1) – u(k);

ρ – é o passo de otimização;

$\frac{\partial J}{\partial u(k)}$ – é a derivada da função objetivo em relação a ação de controle atual.

O vetor de ações de controle futuras pode ser expresso, em relação a um horizonte de predição *NU* da seguinte forma:

$$U(k) = \begin{bmatrix} u(k+1) \\ u(k+2) \\ \vdots \\ u(k+NU) \end{bmatrix} \tag{3.10}$$

Para derivada da função *J* é expressa de forma matricial através da Jacobiana.

$$\frac{\partial J}{\partial U\,(k)} = \begin{bmatrix} \dfrac{\partial J}{\partial u(k+1)} \\ \vdots \\ \dfrac{\partial J}{\partial u(k+NU)} \end{bmatrix} \tag{3.11}$$

Entre os métodos que utilizam informação de gradiente, os mais eficientes em termos computacionais são os de *Newton-Raphson*. Segundo Soloway (1996), este é expresso pela equação 3.12. Estes métodos constroem informação de tendência da função a cada iteração para formular um problema quadrático equacionado com a Hessiana. A solução envolve a sua inversão, onde o método de Newton calcula diretamente, o que é computacionalmente dispendioso. O método de *Newton-Raphson* evita este cálculo utilizando informação da tendência construída da função e aproximada iterativamente a inversa da Hessiana a cada situação.

$$u(k+1) = u(k) - \left(\frac{\partial^2 J}{\partial^2 u(k)}\right)^{-1} \frac{\partial J}{\partial u(k)} \tag{3.12}$$

A Hessiana pode ser expressa da seguinte forma:

$$\frac{\partial^2 J}{\partial U^2} = \begin{bmatrix} \dfrac{\partial^2 J}{\partial u(k+1)^2} & \dfrac{\partial^2 J}{\partial u(k+2)\partial u(k+1)} & \cdots & \dfrac{\partial^2 J}{\partial u(k+NU)\partial u(k+1)} \\ \dfrac{\partial^2 J}{\partial u(k+1)\partial u(k+2)} & \dfrac{\partial^2 J}{\partial u(k+2)^2} & \cdots & \dfrac{\partial^2 J}{\partial u(k+NU)\partial u(k+2)} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 J}{\partial u(k+1)\partial u(k+NU)} & \dfrac{\partial^2 J}{\partial u(k+2)\partial u(k+NU)} & \cdots & \dfrac{\partial^2 J}{\partial u(k+NU)^2} \end{bmatrix} \tag{3.13}$$

Este capítulo teve como finalidade apresentar alguns conceitos básicos sobre a técnica de controle preditivo. Estes conceitos juntamente com os apresentados sobre redes neurais artificiais serão importantes para a formulação da técnica do controle preditivo neural, como pode ser observado no capítulo 4 deste trabalho.

CAPÍTULO 4

CONTROLE PREDITIVO NEURAL

4.1. Controle Preditivo Neural

Os controladores preditivos neurais têm como característica realizar predição de valores futuros através do uso de um modelo neural. A rede neural devidamente dimensionada e treinada estará apta a representar de forma satisfatória o processo, sendo fiel a dinâmica do sistema.

A partir dos valores preditos, oriundos no modelo neural e de uma função objetivo bem definida, o otimizador será capaz de gerar uma lei de controle, calculando desta forma uma sequência de ações de controle com o intuito de estabilizar a saída do sistema na referência desejada.

É possível dividir o controle preditivo neural em dois blocos, o primeiro referente à RNA e a segunda ao otimizador. No diagrama da Figura 4.1 é apresentada uma estrutura geral para a realização dos controladores preditivos baseados nas RNA.

Em alguns casos, é possível utilizar um bloco para o modelo de referência, porém ao assumir que a RNA já está devidamente treinada e representa a dinâmica do processo com um erro satisfatório, não se faz necessário a utilização deste bloco.

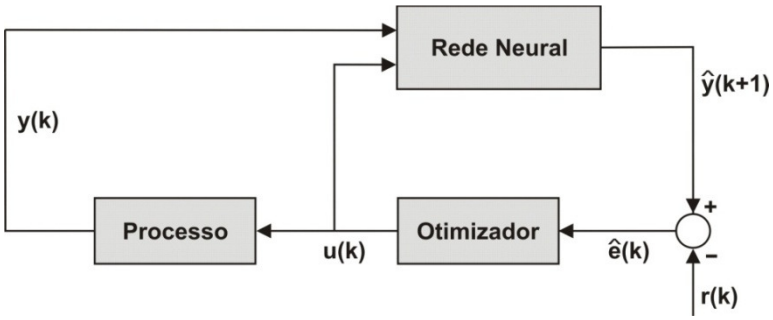

Figura 4.1. Estrutura básica do Controle Preditivo Neural

Nos itens a seguir será explicado o uso das redes neurais com o intuito de modelar as dinâmicas de um sistema e o desenvolvimento dos cálculos para a geração de ações de controle oriundas do otimizador.

4.2. Identificação através de Redes Neurais Artificiais

Define-se uma identificação de sistema como sendo a construção de um modelo matemático para descrever um sistema observado. Em engenharia de controle, identificação de sistemas é usada para determinar um modelo do sistema ou planta que se pretende controlar.

Neste contexto, os modelos dos sistemas descrevem o comportamento da planta em função do tempo *k*, enquanto é exposta ao controle e à influência de fatores externos. Neste tipo de procedimento é necessário identificar a estrutura das equações no modelo e os parâmetros que compõe este modelo ($\hat{\theta}$).

A técnica a ser utilizada para a identificação do sistema baseia-se nas redes neurais artificiais, já que as RNA's tornaram-se uma ferramenta para modelagem matemática não linear (Narendra e Parthasarathy, 1990). O objetivo de sua utilização é avaliar o comportamento do processo não-linear e gerar um modelo confiável e utilizá-lo em conjunto com o controle preditivo.

Segundo Maitelli e Gabriel (2003), no modelo entrada-saída, as entradas e as saídas da planta são representadas por suas respectivas derivadas, sendo que a ordem destas derivadas é que determinará o comportamento dinâmico da planta. A identificação torna possível conhecer a relação entre a entrada e saída do processo.

Para desenvolver um modelo baseado em RNA's será tomado como base o caso linear discreto. O seu modelo linear é dado pela equação 4.1.

$$y(k+d) = \varphi^T(k,d,n).\hat{\theta} + \hat{e}(k+d) \tag{4.1}$$

onde *y* é à saída da planta, φ é o vetor regressor de entrada do modelo linear, $\hat{\theta}$ é o vetor de parâmetros estimados da planta a ser identificada, $\hat{e}$ é o erro de estimação que está descrito na equação 4.2, *n* é ordem da planta e *d* é o atraso de transporte.

$$\hat{e}(k+d) = y(k+d) - \hat{y}(k+d) \tag{4.2}$$

Substituindo a equação do erro (Eq. 4.2) na equação 4.1, temos:

$$\hat{y}(k+d) = \varphi^T(k,d,n).\hat{\theta} \tag{4.3}$$

onde $\hat{y}$ é a saída estimada.

Com o modelo linear apresentado na equação 4.1 é possível desenvolver um modelo para o caso não-linear utilizando RNA. Esse modelo passa a ser descrito da seguinte forma:

$$y(k+d) = g[\varphi(k,d,n), W] + \hat{e}(k+d) \tag{4.4}$$

onde *g* é a função escolhida para modelar o comportamento dinâmico da planta e *W* é uma matriz contendo os pesos sinápticos da rede neural. Substituindo a equação do erro na expressão acima, temos:

$$\hat{y}(k+d) = g[\varphi(k,d,n), W] \tag{4.5}$$

As estruturas de modelos baseadas em rede neural, apropriadas para identificação de sistemas não-lineares, são generalizações das estruturas de modelo linear. Elas são caracterizadas por seu vetor de regressão, ou seja, por um vetor que contém as variáveis usadas para se estimar a saída do sistema. Algumas estruturas de modelo linear são: FIR (*Finite Impulse Response*), ARX (*AutoRegressive*, *eXternal input*), ARMAX (*AutoRegressive*, *Moving Average*, *eXternal input*), OE (*Output Error*) e SSIF (*State Space Innovations Form*). (Lucena, 2005)

De acordo com Nørgaard (2001), dependendo da escolha do vetor regressor podem ser utilizados diferentes modelos de estruturas, como: NNARX, NNFIR, NNARMAX, NNOE e NNSSIF.

O modelo linear NNARMAX (*Neural Networks AutoRegressive with Moving Averange and eXogeneous inputs* ou Rede Neural Autoregressiva com Média Móvel e Entradas Exógenas) tem como principal característica a realimentação dos erros de estimação, desta forma este modelo fica sujeito a problemas de estabilidade. De acordo com Nørgaard (2001), a análise de estabilidade desta estrutura é um problema mais complexo. Em algumas situações, é interessante considerar a estabilidade como uma propriedade local, o que significa afirmar que o modelo somente terá a sua estabilidade garantida quando operar dentro de uma determinada região, podendo ficar instável fora da região de operação. O vetor regressor φ é dado por:

$$\varphi_{NNARMAX}(k,d,n) = [y(k+d-1),\dots,y(k+d-n);\ u(k),u(k-1),\dots, u(k+d-n),\hat{e}(k+d-1),\dots,\hat{e}(k+d-n),-1] \tag{4.6}$$

em que *u* e *y* são a entrada e a saída da planta, respectivamente, $\hat{e}$ é o erro de estimação. A entrada extra, de valor fixo igual a -1 corresponde à entrada do limiar de operação (bias). De acordo com Gabriel (2004), este artifício matemático, reduz a estrutura de dados do programa de implementação computacional, por incluir o limiar de operação dentro da estrutura matricial dos pesos sinápticos.

Da mesma forma que o modelo linear NNARMAX, o modelo linear NNOE (*Neural Networks Output Error* ou Rede Neural Erro na Saída) possui realimentação de sua saída e com isso também está sujeito a problemas de estabilidade. A expressão matemática que define o vetor regressor deste modelo é dada por:

$$\varphi_{NNOE}(k,d,n) = [\hat{y}(k+d-1),\dots,\hat{y}(k+d-n);\ u(k),\ u(k-1),\dots,u(k+d-n),-1] \tag{4.7}$$

onde, *u* e *ŷ* são a entrada e a saída estimada da planta, respectivamente e novamente a entrada com valor fixo -1, corresponde à entrada do limiar de operação da rede neural (bias).

A rede NNFIR (*Neural Network with Finite Impulse Response* ou Rede Neural com Resposta ao Impulso Finita) pelo fato de não possuir realimentação da saída do modelo, é um modelo estável no sentido *BIBO* (*Bounded Input*, *Bounded Output* – entrada limitada, saída limitada), considerando que o vetor regressor e os pesos sinápticos têm valores finitos. De acordo com Gabriel (2004), esta é uma característica importante para a análise de estabilidade de sistemas não-lineares, devido a estes sistemas terem um comportamento mais complexo do que os sistemas lineares.

O modelo não-linear estendido NNFIR é definido pela expressão 4.5 e o vetor regressor (φ) é descrito da seguinte forma:

$$\varphi_{NNFIR}(k,d,n) = [u(k),u(k-1),\dots,u(k-n_u),-1] \tag{4.8}$$

em que *u* é a entrada da planta. E a entrada -1 representa novamente o à entrada do limiar de operação (bias).

O modelo NNARX (*Neural Network Autoregressive with Exogeneous Inputs* ou Rede Neural Autoregressiva com Entradas Exógenas) é formado por valores passados de saída e entrada do processo, em conjunto com os valores presentes, desta forma é possível capturar o comportamento dinâmico do sistema.

A estrutura NNARX trata-se também de um modelo estável no sentido *BIBO*, pelas mesmas razões explicadas no modelo NNFIR. A inexistência de problemas relativos à estabilidade faz deste modelo a escolha preferida quando o sistema a ser modelado é determinístico ou o nível de ruído não é significativo (Nørgaard, 2001). A expressão matemática do modelo não-linear estendido NNARX é dada pela expressão 4.5 e o vetor regressor (φ) é descrito pela seguinte expressão:

$$\varphi_{NNARX}(k,d,n) = [y(k+d-1),\dots,y(k+d-n);\ u(k),\ u(k-1),\dots,u(k+d-n),-1] \tag{4.9}$$

onde o valor -1 representa a entrada do bias na rede neural.

As estruturas NNARX e NNFIR são estáveis no sentido *BIBO*, isso ocorre devido não possuírem realimentação da saída estimada, diferenciando-as das demais estruturas comentadas anteriormente. Ambas asseguram a estabilidade do sistema, porém, a estrutura NNFIR não possui valores atrasados da saída da planta, exigindo um número maior de valores a ser usado no vetor de regressão, composto somente de valores atrasados da entrada da planta.

Por este motivo a estrutura NNARX será utilizada para realizar a identificação de sistema neste trabalho. A estrutura básica do modelo pode ser observada na figura 4.2.

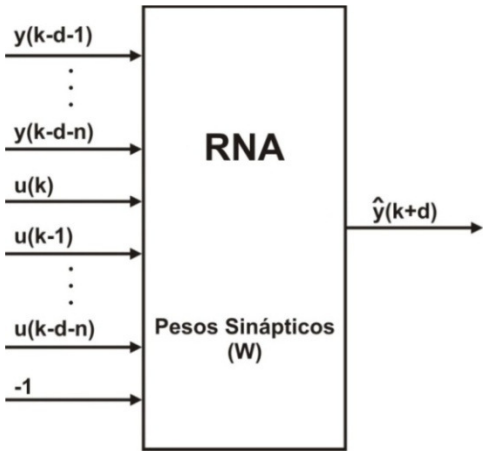

Figura 4.2. Modelo de estrutura NNARX

4.2.1. Normalização

Os dados apresentados a rede neural devem ser normalizados, com o objetivo de evitar que valores de ordem de grandeza distintos influenciem nos resultados. De acordo com Vieira (2005), a expressão adotada para normalização resulta num padrão limitado entre 0,1 e 0,9. Dessa forma é possível realizar uma pequena extrapolação dos dados e evitar a utilização de valores extremos (Vieira, 2002). A expressão utilizada foi:

$$P_{nor} = 0.1 + \left(\frac{P_{real} - P_{min}}{P_{max} - P_{min}}\right) \times 0.8 \qquad 4.10$$

onde:

P_{nor} – valor normalizado;

P_{real} – valor real (entradas e saídas);

P_{min} – valor mínimo de cada variável;

P_{max} – valor máximo de cada variável.

4.2.2. RNA utilizada como preditor

O comportamento futuro do processo ao longo de um horizonte de predição é determinado por um preditor, o qual, geralmente é representado por modelo analítico do sistema. Neste trabalho a rede neural artificial irá realizar o papel de preditor, predizendo o estado futuro da planta e descartando desta forma o modelo analítico.

A estrutura neural desenvolvida para o trabalho pode ser observada na figura 4.3. A rede neural em questão apresenta em sua arquitetura três camadas, sendo uma camada de entrada, uma intermediaria e uma camada de saída.

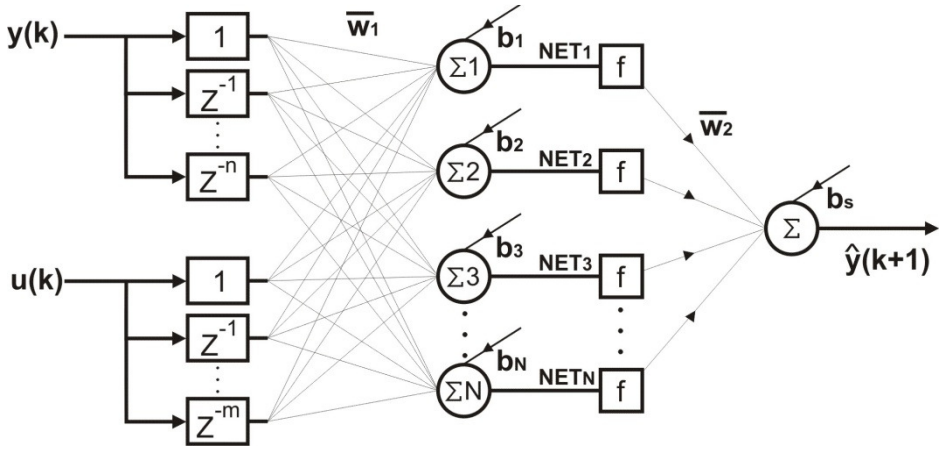

Figura 4.3. Estrutura da RNA utilizada

onde *N* representa o número de neurônios na camada intermediária, o *b* é o *bias* associado a cada neurônio e o *f* representa a função de ativação, no caso a utilizada para a camada oculta será a sigmóide bipolar. O termo *n* representa a quantidade de regressores de *y* (saída) e *m* representa o número de regressores de *u* (entrada). As matrizes de pesos w_1 e w_2 possuem dimensões de *Nx(n+m)* e *1xN* respectivamente e representam os pesos das conexões sinápticas entre as camadas entrada/intermediária e intermediária/saída.

Segundo Schnitman (1998), tem-se a seguinte expressão da saída predita para o modelo utilizado:

$$\hat{y}(k+1) = b + \sum_{i=1}^{N} w_2(1,i) f(NET_i) \tag{4.11}$$

O termo $f(NET_i)$ representa a saída estimada pela função não-linear, e depende dos valores anteriores das entradas e saídas do sistema. O valor de NET_i pode ser calculado pela seguinte expressão:

$$NETi = b(i,1) + \sum_{j=1}^{n} w_1(i,j) y(k-j+1) + \sum_{j=1}^{m} w_1(i,n+j) u(k-j+1) \tag{4.12}$$

As leis de controle que serão obtidas pelo otimizador, irão necessitar do cálculo do Jacobiano estimado da planta, a qual é fornecida pela derivada da saída estimada do processo em relação a entrada, qual seja:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \frac{\partial}{\partial u(k)}\left[b + \sum_{i=1}^{N} w_2(1,i) f(NET_i)\right] \tag{4.13}$$

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} w_2(1,i) f'(NET_i) \frac{\partial (NET_i)}{\partial u(k)} \tag{4.14}$$

onde *f'* pode ser representado da seguinte forma:

$$f' = \frac{\partial f}{\partial NET_i} \tag{4.15}$$

Em relação à equação 4.12, temos:

$$\begin{aligned} \frac{\partial (NETi)}{\partial u(k)} = \frac{\partial}{\partial u(k)}[b(i,1)] + \frac{\partial}{\partial u(k)}\left[\sum_{j=1}^{n} w_1(i,j) y(k-j+1)\right] \\ + \frac{\partial}{\partial u(k)}\left[\sum_{j=1}^{m} w_1(i,n+j) u(k-j+1)\right] \end{aligned} \tag{4.16}$$

Os valores passados das variáveis *y(k)* e *u(k)* não dependem de *u(k)*, desta forma, o somatório apenas não é nulo em *j = 1*. Assim:

$$\frac{\partial (NET_i)}{\partial u(k)} = w_1(i, n+1) \tag{4.17}$$

Desta forma, temos:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} w_2(1,i) f'(NET_i) w_1(i, n+1) \tag{4.18}$$

4.3. Otimizador

O otimizador de um controlador preditivo estabelece a estratégia do controle, através da minimização de um índice de desempenho *J* em relação às *NU* ações de controle, presentes e futuras, que são computadas de modo a minimizar uma função objetivo quadrática (Medeiros, 2005). Além de procurar reduzir o tempo de estabilização e o *overshoo*t, o otimizador busca minimizar a atuação do controle estabelecendo para isso uma solução de compromisso entre esses objetivos.

Como comentado anteriormente, a rede neural é utilizada como modelo do processo real e vai predizer as saídas do processo, desta forma o otimizador vai calcular as ações de controle com base na saída predita e não na saída real do sistema. O controle preditivo pode ser desenvolvido para realizar predições de 1 a *NY* passos à frente.

4.3.1. Controle Preditivo Neural NY passo(s) à frente

A função objetivo *J* a ser minimizado pelo controlador é representada da seguinte forma:

$$J = \frac{1}{2}\sum_{j=N_1}^{NY}[r(k+j) - \hat{y}(k+j)]^2 + \frac{1}{2}\lambda\sum_{i=0}^{NU}[\Delta u(k+i-1)]^2 \qquad 4.19$$

em que:

k – instante atual;

r – é sinal que representa a trajetória de referência;

Δu – é a variação da ação de controle, definida como u(k+j) – u(k+j-1);

λ – é a ponderação da ação de controle;

N_1 – é o horizonte mínimo de predição;

NU – é o horizonte de controle;

NY – é o horizonte máximo de predição.

onde,

$$r(k+j) - \hat{y}(k+j) = \hat{e}(k+j) \tag{4.20}$$

O método do gradiente decrescente, como dito no capítulo anterior, será utilizado como regra de atualização da ação de controle, desta forma, temos a seguinte expressão:

$$U(k+1) = U(k) - \rho \frac{\partial J}{\partial U(k)} \tag{4.21}$$

O parâmetro $\rho \in \mathbb{R}^+$ e pode ser definido como um passo de otimização. É um importante parâmetro para o desempenho do controlador, dependendo da maneira que for utilizado, pode causar lentidão ou instabilidade ao sistema. Para que isso seja evitado se faz necessário a utilização de um algoritmo que varie o valor desse parâmetro ao longo da atuação do controlador, buscando a otimização da ação do sinal de controle.

Como descrito no item 2.7.1 deste trabalho, este algoritmo foi proposto em Gabriel (2004), porém foi utilizado para variar a constante de aprendizagem η, durante o treinamento das redes neurais.

<u>Implementação</u>

Fazer ρ_aux1 = 0 antes do início da atuação do controlador
Fazer ρ_aux2 = ρ;

Se ρ_aux2<=1.04* ρ_aux1 Então Fazer
    Se ρ_aux2> ρ_aux1 Então Fazer
        ρ = ρ;
    Senão Fazer
        ρ = 1,05* ρ;
Senão Fazer
    ρ = 0.7* ρ;
FimSe

É sugerido que o passo de otimização seja iniciado com um valor pequeno, entre 0,001 e 0,1. Os fatores 1,04, 1,05 e 0,7 usados no algoritmo acima são sugeridos por Gabriel (2004).

O próximo passo é calcular a derivada da função *J* em relação a *u(k)*. Com isso, temos:

$$\frac{\partial J}{\partial u(k+h)} = \frac{1}{2}\sum_{j=N_1}^{NY}\left[-2.\hat{e}(k+j).\frac{\partial \hat{y}(k+j)}{\partial u(k+h)}\right] + \frac{1}{2}\lambda\sum_{i=0}^{NU} 2\frac{\partial \Delta u(k+i)}{\partial u(k+h)} \qquad 4.22$$

em que, $h = 1 \ldots NU$.

Simplificando, temos:

$$\frac{\partial J}{\partial u(k+h)} = -\sum_{j=N_1}^{NY}\left[\hat{e}(k+j).\frac{\partial \hat{y}(k+j)}{\partial u(k+h)}\right] + \lambda\sum_{i=0}^{NU}\frac{\partial \Delta u(k+i)}{\partial u(k+h)} \qquad 4.23$$

A variação do sinal de controle é definida da seguinte forma:

$$\Delta u(k) = u(k) - u(k-1) \qquad 4.24$$

Então,

$$\frac{\partial \Delta u(k+i)}{\partial u(k+h)} = \frac{\partial u(k+i)}{\partial u(k+h)} - \frac{\partial u(k+i-1)}{\partial u(k+h)} \qquad 4.25$$

De acordo com Zeng (2003), o termo $\partial \Delta u(k+NU)/\partial u(k+h)$ da equação 4.23 pode ser escrito em termos da função delta de Kronecker, assim:

$$\frac{\partial u(k+i)}{\partial u(k+h)} - \frac{\partial u(k+i-1)}{\partial u(k+h)} = \delta(h,i) - \delta(h,i-1) \qquad 4.26$$

onde a função delta de Kronecker é definida como:

$$\delta(h,i) = \begin{cases} 1 \; se \; h = i \\ 0 \; se \; h \neq i \end{cases} \qquad 4.27$$

O termo $\partial\hat{\mathrm{Y}}/\partial U(k)$ da equação 4.23 é uma matriz jacobiana e pode ser expandida na seguinte forma:

$$\frac{\partial\hat{\mathrm{Y}}}{\partial U(k)}=\begin{bmatrix} \dfrac{\partial\hat{y}(k+1)}{\partial u(k)} & 0 & \cdots & 0 \\ \dfrac{\partial\hat{y}(k+2)}{\partial u(k)} & \dfrac{\partial\hat{y}(k+2)}{\partial u(k+1)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial\hat{y}(k+NY)}{\partial u(k)} & \dfrac{\partial\hat{y}(k+NY)}{\partial u(k+1)} & \cdots & \dfrac{\partial\hat{y}(k+NY)}{\partial u(k+NU-1)} \end{bmatrix} \tag{4.28}$$

Para o cálculo recursivo da matriz jacobiana as equações 4.11, 4.12 e equação de controle são generalizadas, desta forma temos:

$$\hat{y}(k+NY)=b+\sum_{i=1}^{N} w_2(1,i)f[NET_i(NY)] \tag{4.29}$$

onde,

$$NET_i(NY)=b(i,1) + \sum_{j=1}^{n} w_1(i,j)y(k-j+NY)+\sum_{j=1}^{m} w_1(i,n+j)u(k-j+NY) \tag{4.30}$$

$$u(k+NU)=u(k+NU-1) - \rho\left[\left(-E\frac{\partial\hat{y}(k+NY)}{\partial u(k+NU-1)}\right)+\lambda\frac{\partial\Delta u(k+NU)}{\partial u(k+NU-1)}\right] \tag{4.31}$$

Generalizando o cálculo da diagonal principal da matriz jacobiana (Eq. 4.28), temos:

$$\frac{\partial\hat{y}(k+NY)}{\partial u(k+NU-1)}=\sum_{i=1}^{N} w_2(1,i)f'[NET_i(NY)]w_1(i,n+1) \tag{4.32}$$

Os elementos fora da diagonal principal podem ser calculados de forma recursiva, baseando-se no valor do elemento da diagonal principal que está mesma linha e nos elementos que estão acima, como descrito na equação 4.33.

$$A(i,j) = f\{A(i,i), A(k,j)\}, \text{com } k = 1 \text{ até } k = i - 1 \qquad 4.33$$

No fluxograma abaixo é possível visualizar o algoritmo do controle preditivo NY passos à frente, onde são apresentados todos os passos descritos ao longo deste capítulo, como: inicialização de variáveis, cálculo dos valores estimados do processo, da matriz Jacobiana e o cálculo da sequência de ação de controle.

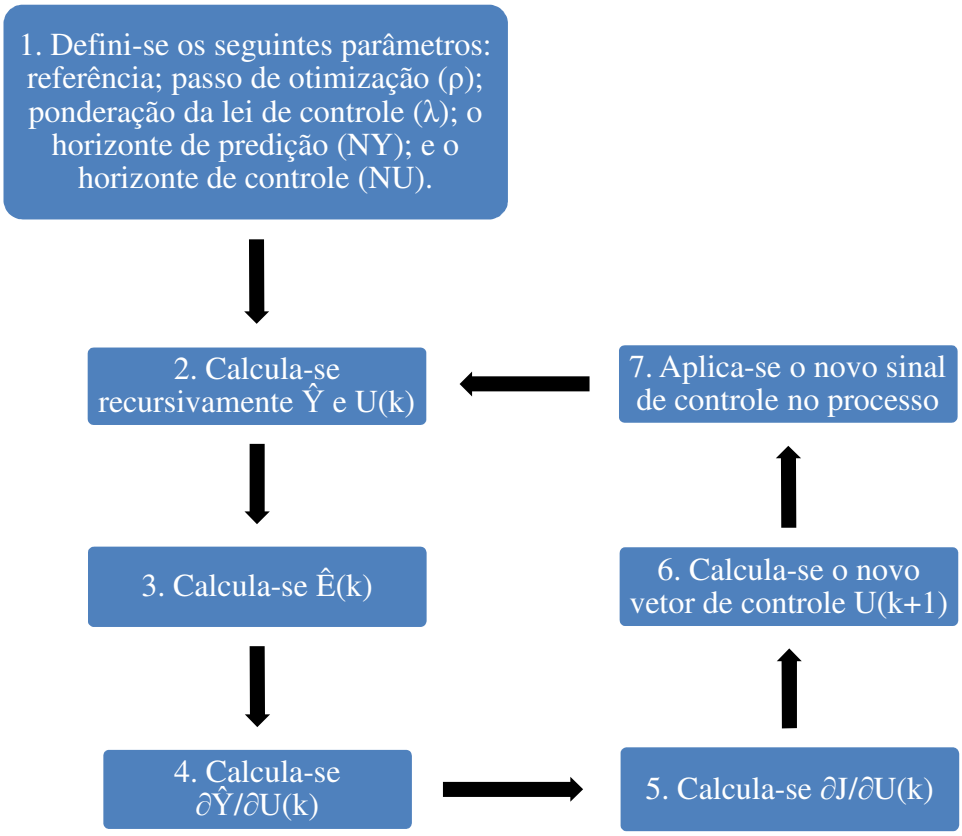

Figura 4.4. Fluxograma do Controle preditivo neural com NY passos à frente

Neste capítulo foi apresentada a teoria de controle preditivo neural, sendo mostrado como é realizada a identificação neural e o cálculo das ações de controle. O desenvolvimento

matemático foi baseado em Schnitman (1998), porém existem duas grandes diferenças, a primeira com relação à implementação de uma função objetivo mais complexa e a segunda com relação ao uso do passo de otimização de forma adaptativo.

CAPÍTULO 5

APLICAÇÃO E RESULTADOS

5.1. Implementação

O controlador preditivo neural foi aplicado em um processo de pH. Com esta implementação é possível analisar o desempenho do controlador. O processo de pH é um processo que traz grandes desafios por apresentar não-linearidade estática.

O sistema de controle de pH, utilizado neste trabalho, foi desenvolvido em Fontes et al. (2008) e utilizado em Vale et al. (2008). É um processo simulado computacionalmente no *software* Simulink e se aproxima de forma satisfatória do processo real.

Na primeira parte do capítulo foi comentado sobre o processo ao qual o controle foi empregado, explicando o seu funcionamento e o motivo pelo qual este tipo de processo representa um desafio para os projetos de controladores. Além do levantamento do modelo da planta é apresentado a modelagem e construção dos equipamentos que fazem parte do processo. Também foram feitas simulações em malha aberta do sistema com o intuito de adquirir um conjunto de dados para ser realizada uma identificação neural do mesmo.

Na seção seguinte, são apresentados os programas desenvolvidos para realizar a identificação neural e o controle preditivo neural, é possível observar as interfaces criadas e os parâmetros necessários para o seu funcionamento. Todos os programas foram desenvolvidos na plataforma C++ Builder.

Na última parte do capítulo, são realizados vários testes com o controlador no intuito de avaliar seu desempenho e são feitas comparações com outros controladores (PI e Preditivo escalonado). Desta forma é possível analisar as vantagens e desvantagens de sua utilização.

5.2. Processo de pH

O processo de pH é de grande importância em várias áreas da biotecnologia e indústria de processos químicos, além de ser essencial no tratamento de águas. Este sistema é utilizado na indústria petroquímica, com a finalidade de manter constante o nível de acidez de um produto, ou neutralizar o afluente de uma planta de tratamento de fluidos, por exemplo.

A elaboração de modelos para sistemas de neutralização e, especificamente, para o processo de pH tem-se tornado mais relevante devido à necessidade de melhorar a qualidade do produto ou de otimizar o processo de produção. Outro aspecto a ser citado é a importância de minimizar o impacto ambiental, no caso de uma estação de tratamento de esgotos.

De acordo com Fontes et al. (2008), o sistema de controle de pH representa um desafio para o projeto de controladores, por apresentar ganho estático e dinâmica não-linear, exigindo desta forma uma certa robustez do sistema de controle.

Por causa disso, controladores convencionais, como o PID, têm apresentado desvantagens no controle deste tipo de processo. Estes problemas podem ser melhores tratados com a utilização de controladores adaptativos, inteligentes, preditivos ou a combinação dessas técnicas (Vale et al., 2008).

Como pode ser observado na figura 5.1, a dinâmica de um reator tanque com agitador contínuo (CSTR – *Continuos Stirred-Tank Reactor*) no controle de pH, geralmente, consiste em duas vias de fluxo de corrente. Uma via com a vazão controlada de um ácido e a outra via com uma vazão de uma base que se deseja controlar o pH do sistema (Fontes et al., 2008).

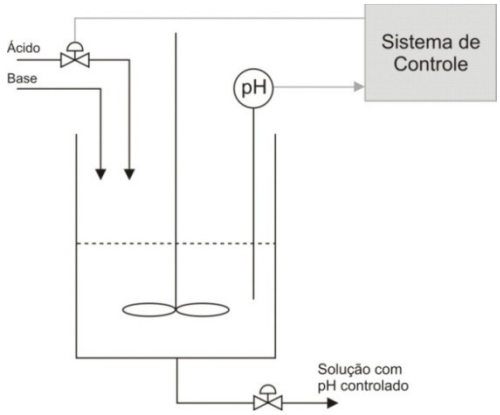

Figura 5.1. Diagrama da Planta de Neutralização de pH

O nível do CSTR é mantido constante, através do uso de um controle de nível, não descrito neste trabalho. O controle do pH é realizado através da mistura das substâncias por um agitador. Um sensor medidor de pH é instalado no tanque e é responsável por indicar o pH característico logo após a finalização da mistura.

5.2.1. Modelagem do sistema

No modelo desenvolvido para as simulações do processo de neutralização de pH, não foi considerado o controle de nível da planta. O modelo de Hammerstein é utilizado para representar o processo, onde a não linearidade estática antecede a dinâmica da planta.

Para que o sistema simulado se aproxime do real, se faz necessário introduzir na simulação alguns fatores, além do modelo da planta de pH, como a dinâmica do atuador e do sensor, que são equipamentos de grande importância no processo. O diagrama de blocos do sistema completo é mostrado na figura 5.2.

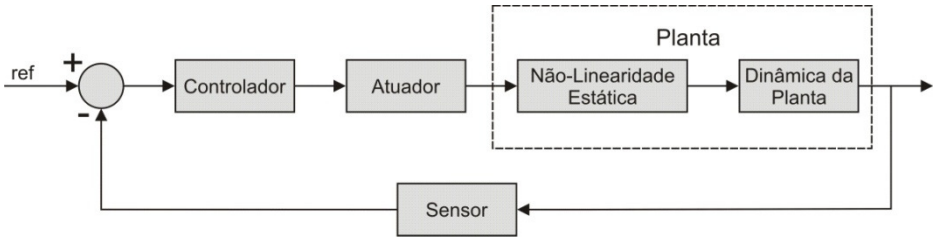

Figura 5.2. Diagrama de Blocos do Processo

Atuador (válvula):

O bloco do atuador é composto por uma função transferência para representar a sua dinâmica (vide equação 5.1) e também possui uma zona morta para simular a não-linearidade da válvula.

$$G_A(s) = \frac{1}{30s + 1} \qquad 5.1$$

Na saída do atuador, soma-se 50 a este sinal, pois foi considerado que para a condição de equilíbrio o pH é igual a 7. Assim, o desvio do sinal seria igual a zero, a válvula estaria aberta em 50% da sua abertura total, isto é na condição nominal.

Modelo de Hammerstein utilizado:

Para o desenvolvimento do modelo matemático do comportamento da planta foi utilizado o modelo de Hammerstein, extraído de Vall (2006). Este modelo consiste de um elemento com uma não-linearidade estática seguido pela dinâmica da planta, conforme a figura 5.3.

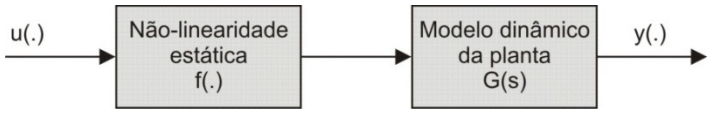

Figura 5.3. Modelo de Hammerstein

Essa não-linearidade foi modelada, com uma aproximação, como mostra a figura 5.4.

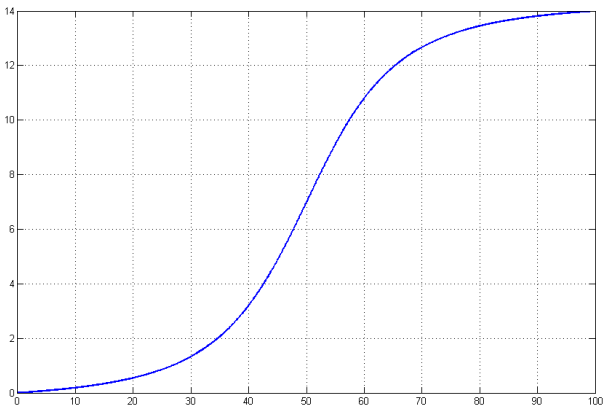

Figura 5.4. Não-linearidade estática simplificada

Algumas manipulações matemáticas foram realizadas para que em uma abertura de 50% da válvula o sinal de saída do modelo fosse igual 7 (pH neutro), considerado como ponto de equilíbrio do sistema. A não-linearidade é representada pela equação 5.2 e a dinâmica da planta pela equação 5.3.

$$y = 7\frac{0{,}02 \cdot u - 1}{\sqrt{0{,}1 + 0{,}9 \cdot (0{,}02 \cdot u - 1)^2}} \quad 5.2$$

$$G_P(s) = \frac{1}{200s + 1} \quad 5.3$$

Sensor:

O sensor é representado por uma função de transferência de primeira ordem (vide equação 5.4), sendo a constante de tempo igual a 10s e ganho igual a 1. O sinal da saída deste bloco representa o valor do pH (7 à 14). Para que o valor da saída do processo fique entre 0 e 100% é realizada uma normalização na saída do sensor.

$$G_S(s) = \frac{1}{10s + 1} \quad 5.4$$

5.2.2. Simulação em malha aberta

Para realizar a identificação neural é necessário obter um conjunto de pontos para o treinamento da rede neural. A aquisição destes dados é realizada com o sistema em malha aberta e são aplicados diversos degraus de diferentes amplitudes na entrada do processo.

A figura 5.5 mostra o sinal de entrada. Observando o sinal de saída do processo na figura 5.6, é possível perceber que essa variação no sinal de entrada é suficiente para cobrir quase toda a variação de pH (0 à 14).

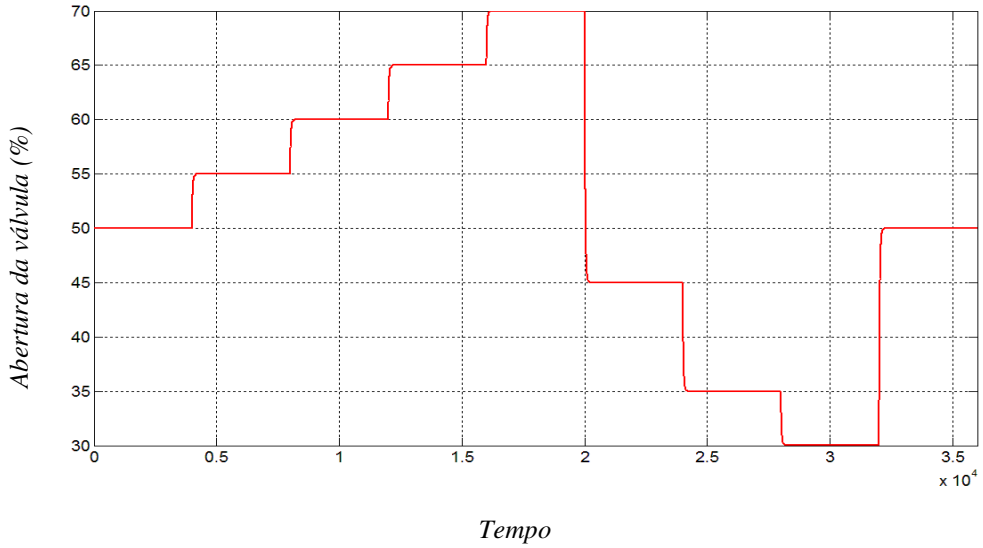

Figura 5.5. Sinal de controle em malha aberta

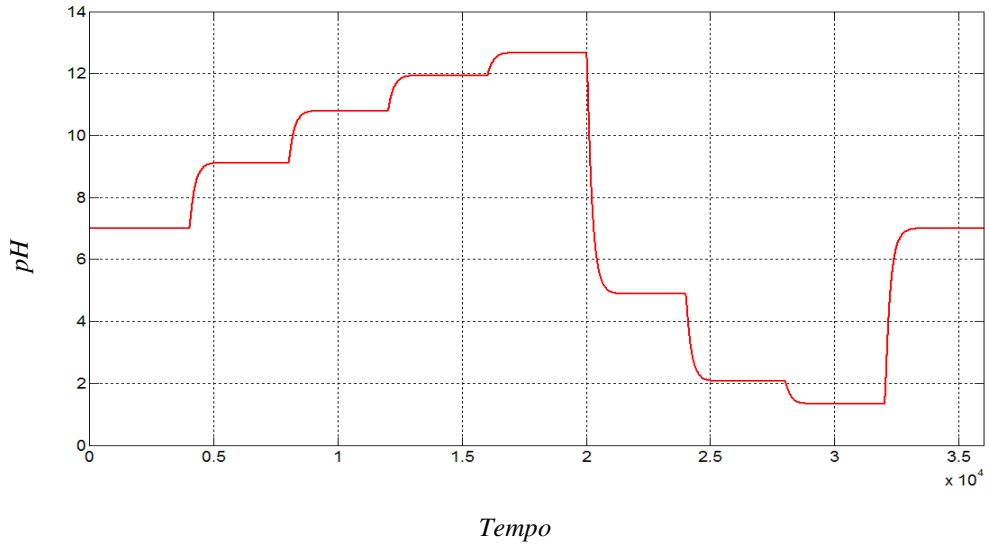

Figura 5.6. Saída do sistema em malha aberta

Ao final desta simulação, todos os dados gerados são armazenados em um arquivo e serão utilizados como dados de entrada na rede neural para realizar a identificação do sistema.

5.3. Interface

Os programas utilizados neste trabalho foram desenvolvidos na plataforma C++ Builder. A figura 5.7 mostra a interface do Identificador Neural, onde são ajustados todos os parâmetros da rede neural. Desta forma é possível realizar uma série de testes para que a

identificação do sistema seja a mais precisa possível. Os parâmetros de entrada para o funcionamento correto do identificador são:

- Quantidade de camadas da rede;
- Número de entradas e saídas do sistema;
- Número de regressores (valor utilizado tanto para as entradas como para as saídas);
- Quantidade de pontos para o treinamento (em porcentagem);
- Quantidade de neurônios e a função de ativação de cada camada;
- Se ativar o momento, entrar com o seu valor;
- Quantidade máxima de ciclos.

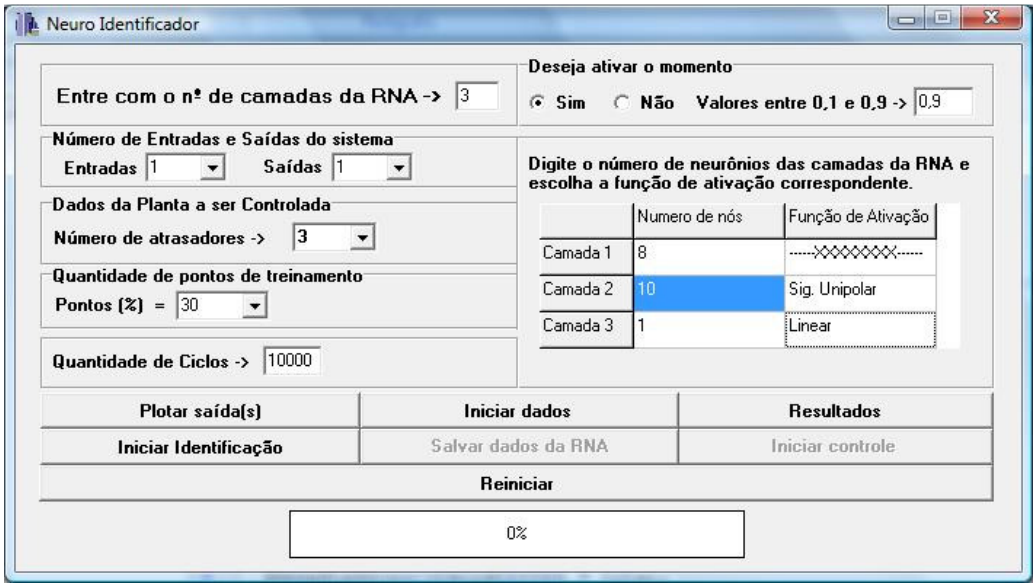

Figura 5.7. Interface do Identificador Neural

Depois de realizado a identificação neural é possível salvar os parâmetros utilizados e os pesos ajustados adequadamente para representar o sistema. Esses dados são salvos para serem usados posteriormente no controle preditivo neural.

A figura 5.8 mostra a interface do Controle Preditivo Neural. Da mesma forma que o identificador neural, na tela do controlador é necessário a inicialização de alguns parâmetros, que são:

- Horizonte de predição;
- Horizonte de Controle
- Ponderação do sinal de controle;
- Se deseja ativar o passo de otimização adaptativo ou que ele assuma um único valor ao longo do controle.
- Valor da referência desejada;

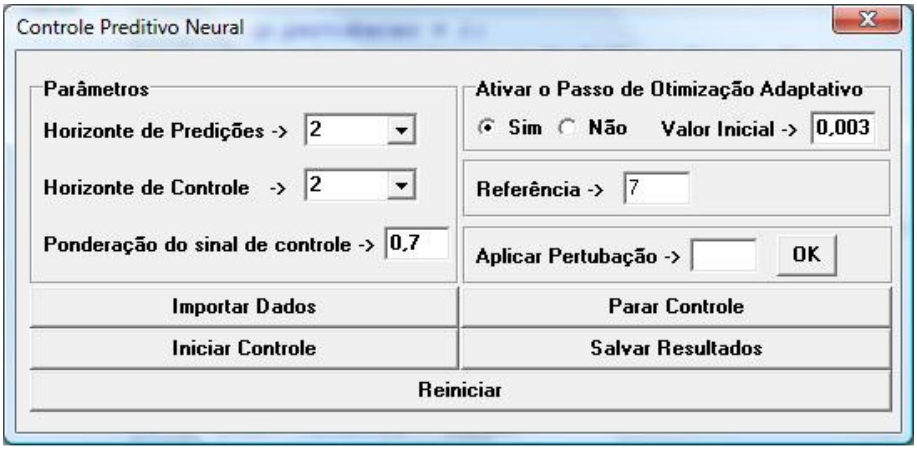

Figura 5.8. Interface do Controlador Preditivo Neural

Após o ajuste dos parâmetros descritos acima é necessário inicializar o arquivo onde estão os dados da rede neural, arquivo este que é criado no término da identificação neural. Como dito anteriormente esse arquivo possui todos os parâmetros da rede neural e a estrutura de pesos ajustados. É possível também aplicar uma perturbação na saída do sistema.

5.4. Identificação através das Redes Neurais Artificiais

Depois de obtido o conjunto de pares de entradas e saídas da simulação em malha aberta é possível iniciar o processo de identificação da planta. Na identificação neural devem ser feitos ajustes relacionados à arquitetura da RNA. A escolha da rede deve ser feita de forma que a mesma não seja demasiadamente simples, de modo a limitar a capacidade de identificação e predição da rede, e nem demasiadamente complexa, de modo a tornar o processo de treinamento excessivamente longo.

A arquitetura utilizada foi escolhida através de testes. Os melhores parâmetros utilizados na RNA para a identificação do sistema foram:

- 3 camadas, sendo, 1 de entrada, 1 oculta e outra de saída;
- 3 regressores;
- Arquitetura da RNA: 8 – 10 – 1;
- 1ª camada: função sigmóide bipolar; 2ª camada: função linear;
- Momento igual a 0,9;
- 80000 épocas;
- E um conjunto de treinamento com 2000 pares de entradas e saídas. 30% destes dados são utilizados para o treinamento da RNA e o restante é usado para a validação.

O comportamento do erro total, no treinamento da rede neural pode ser observado na figura 5.9. O erro assumiu, no final do treinamento, um valor aproximado de $4x10^{-4}$. A validação da RNA é realizada através dos 70% dos dados restantes. A figura 5.10 mostra o resultado da validação, onde o sinal verde representa a saída real e o sinal vermelho à saída da RNA.

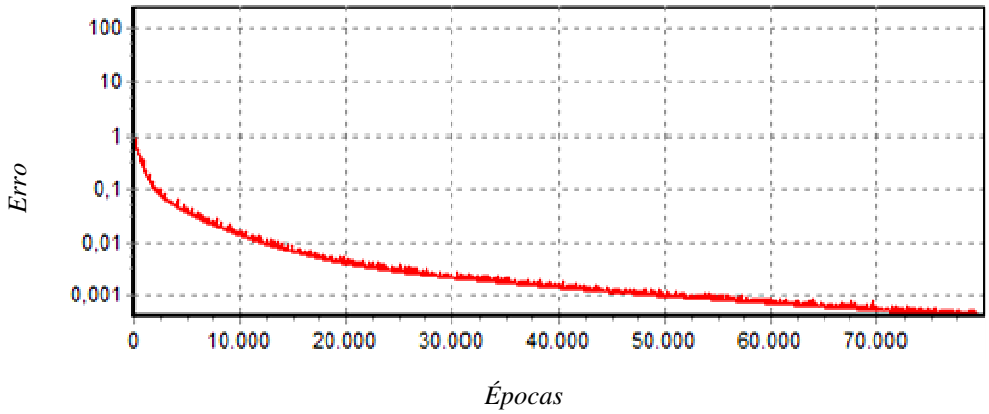

Figura 5.9. Sinal do erro total

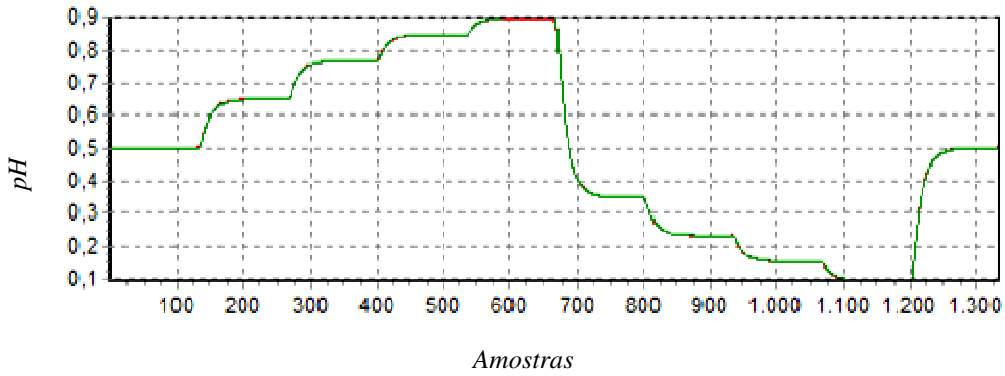

Figura 5.10. Sinal de saída da RNA

Ao analisar os gráficos apresentados é possível concluir que a rede neural foi capaz de identificar com um bom grau de exatidão o processo e desta forma é possível adotá-la como modelo neural do sistema.

É importante lembrar que os dados utilizados estão todos normalizados como foi citado no capítulo 5 desta dissertação, por isso os valores representados na figura 5.10 estão delimitados entre 0,9 e 0,1.

5.5. Controle Preditivo Neural

Com o modelo neural gerado e validado é possível utilizá-lo com controle preditivo e realizar os testes para analisá-lo. Os modelos neurais são utilizados em conjunto com o

controle preditivo com o intuito de predizer os valores de saída dos sistemas. Foram realizados vários testes, dentre eles: variação nos parâmetros do controlador, análise do passo de otimização e comparações com outros controladores. Os resultados obtidos são mostrados a seguir.

5.5.1. Simulação do controlador Preditivo Neural

A seguir será apresentado os resultados da implementação do Controlador Preditivo Neural. Para a mesma planta, foram realizados alguns testes.

Para este primeiro teste o tipo do sinal de referência utilizado é do tipo degrau, onde foram aplicados vários níveis de degrau de forma aleatória na entrada do processo. O primeiro resultado que será apresentado foi derivado do melhor ajuste encontrado dos parâmetros do controlador.

Os parâmetros utilizados foram:

- Horizonte de predição → NY = 3;
- Horizonte de controle → NU = 3;
- Ponderação do sinal de controle → $\lambda = 0{,}5$;
- O passo de otimização foi utilizado de forma adaptativa e o valor inicial igual a 0,005.

A figura 5.11 mostra o resultado obtido, onde o sinal azul representa a saída do processo e o vermelho representa o sinal de referência. O sistema parte do valor de pH igual a 7 pois é ponto de equilíbrio considerado. É possível observar que o erro de regime permanente assumiu valores da ordem de 10^{-2} a 10^{-3}.

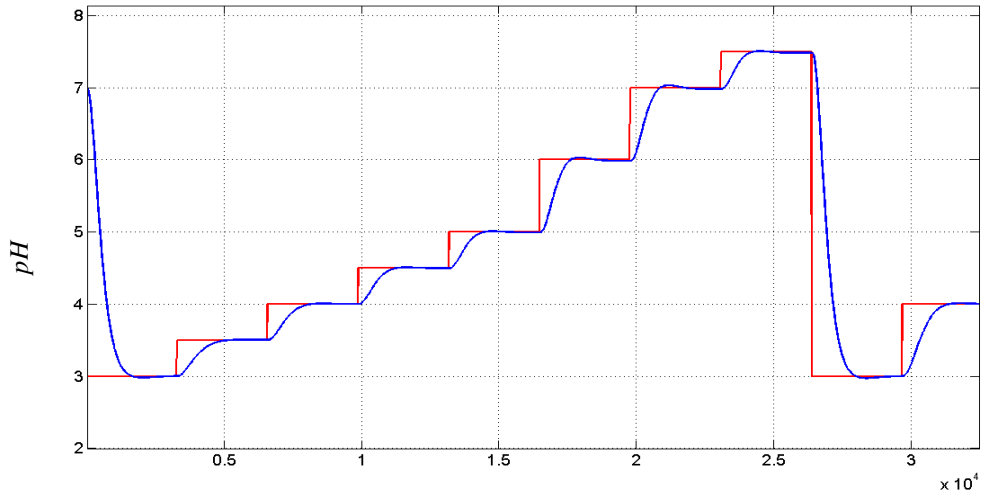

Figura 5.11. Sinal de saída da planta

O sinal de controle é mostrado na figura 5.12, onde a variação do sinal de controle apresentou um comportamento suave ao longo do processo.

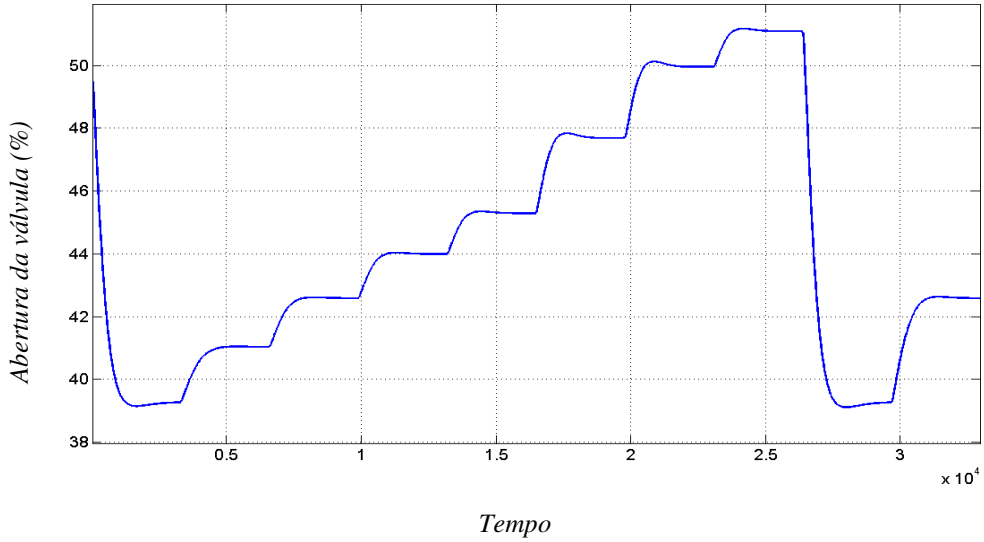

Figura 5.12. Sinal de controle

5.5.2. Análise da variação de parâmetros do controlador

Para os próximos testes, alguns parâmetros do controlador serão alterados no intuito de analisar o seu funcionamento. A referência utilizada para estes testes será constante e com valor de pH igual a 3.

A primeira análise será da ponderação do sinal de controle (λ). Foram realizadas três simulações: a primeira com o valor da ponderação igual a 0,6, a segunda com o valor de 0,7 e a terceira de 0,9. Todos os outros parâmetros permanecem os mesmos utilizados na primeira simulação, onde: NU e NY são iguais a 3 e o passo de otimização é usado de forma adaptativo. A figura 5.13 mostra os sinais de saída gerados por cada valor de λ e na figura 5.14 podem ser observados os sinais de controle.

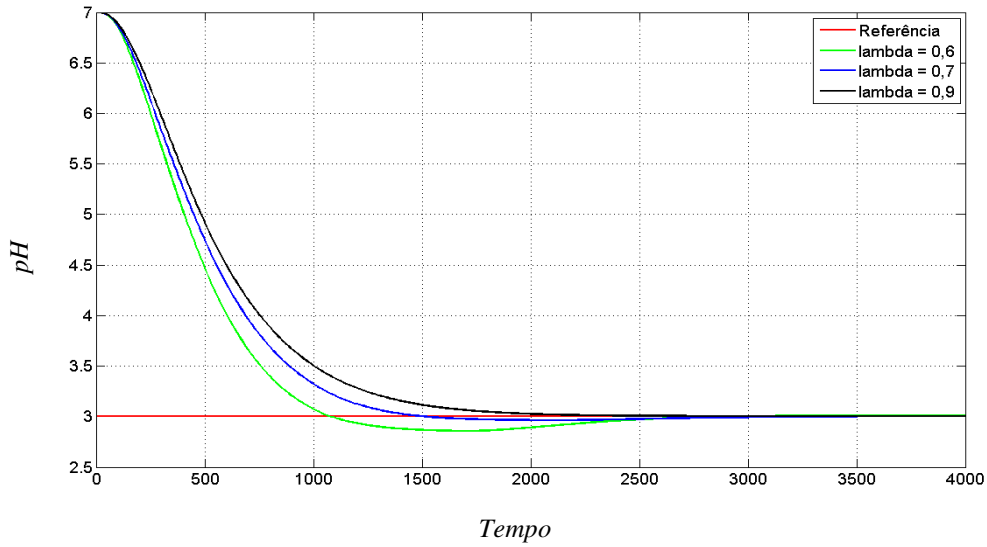

Figura 5.13. Comparação dos sinais de saída para diferentes valores de λ

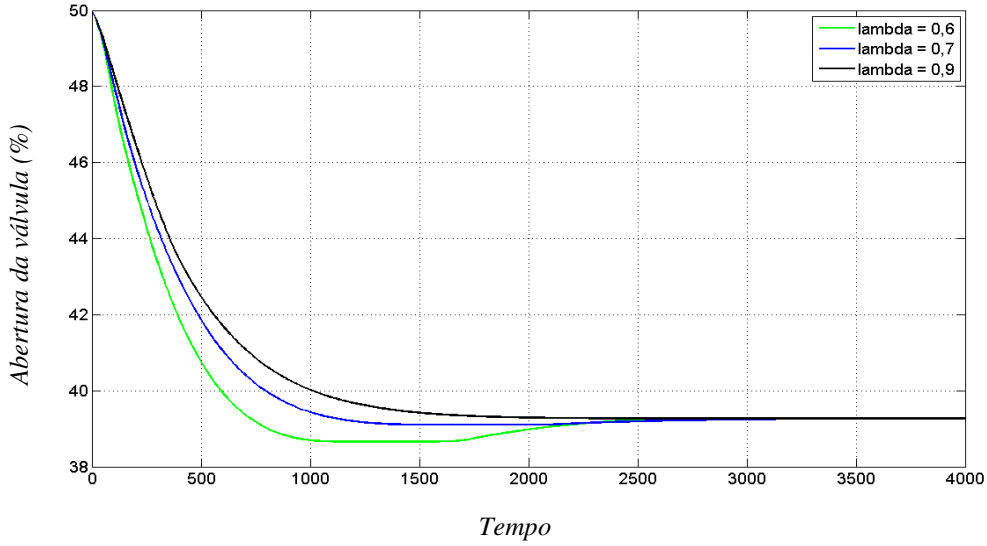

Figura 5.14. Sinais de controle para diferentes valores λ

De acordo com a função objetivo utilizada (Eq. 3.16), o valor do sinal de controle se torna mais agressivo ao diminuir a sua ponderação, isto pode ser comprovado através dos resultados obtidos acima. Quando é utilizado um valor de λ igual a 0,6 o sinal de controle se torna mais agressivo do que λ igual a 0,9, apresentando desta forma uma maior porcentagem de *overshoot* no sinal de saída.

Outra análise realizada é a variação do horizonte de predição e de controle. Foram feitas 3 variações, uma com NY e NU igual a 3 depois os seus valores foram alterados para 2 e por último foi utilizado 1 para os dois parâmetros, o valor da ponderação de controle foi igual a 0,8 e o passo de otimização foi usado de forma adaptativa. Os resultados obtidos podem ser observados nas figuras 5.15 (sinais de saída) e 5.16 (sinais de controle).

Após analisar os resultados é possível concluir que saída se tornou muito lenta ao se utilizar horizonte de igual a 1 predição, ao ponto de fazer com que o sistema não atinja o valor desejado no tempo estabelecido. Ao se utilizar horizonte de predição 3 o sistema estabilizou por volta de 1250 passos enquanto quando é usado duas predições o sistema apresenta uma resposta mais lenta, estabilizando no tempo de 1500 passos.

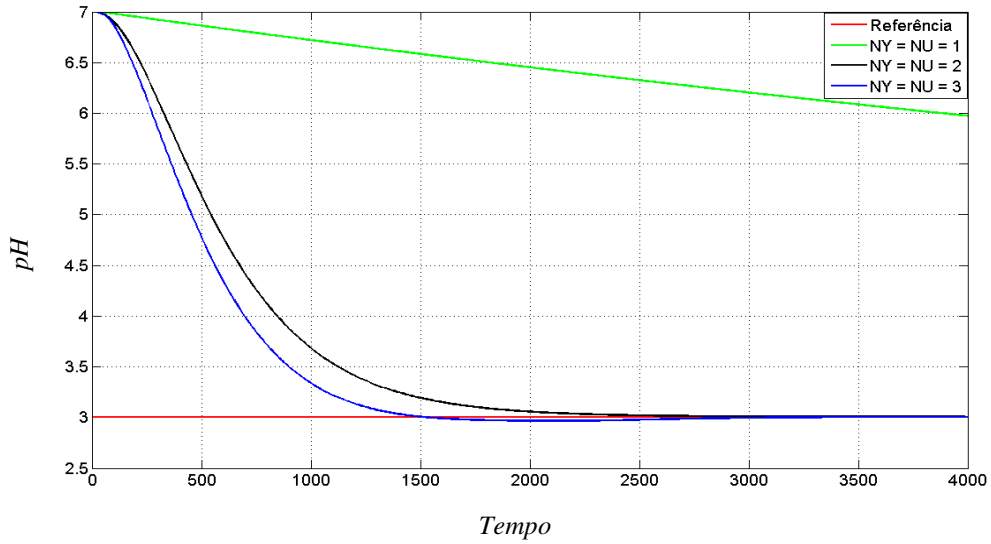

Figura 5.15. Comparação com os sinais de saída para diferentes horizontes

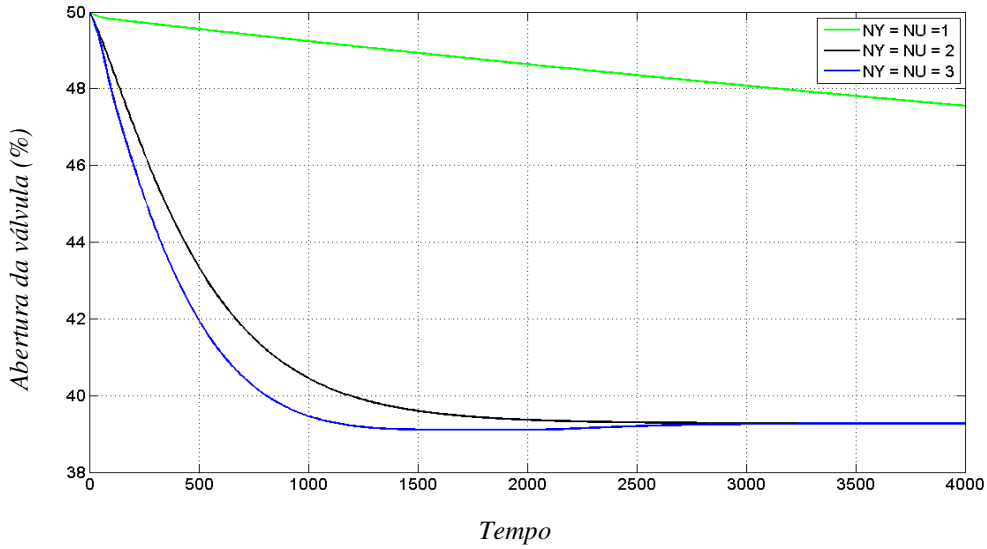

Figura 5.16. Sinais de controle para diferentes horizontes

5.5.3. Análise do passo de otimização

O passo de otimização (ρ) é um parâmetro de extrema importância para o desempenho do controlador. Como dito anteriormente esse parâmetro pode causar lentidão ou instabilidade ao sistema. Conforme descrito no item 4.3.1 deste trabalho foi utilizado um passo de otimização que sofre variação ao longo do controle, chamado de passo de otimização adaptativo.

Alguns testes foram realizados no intuito de analisar as vantagens de se utilizar este artifício. Na figura 5.17 o sinal azul representa a resposta do sistema quando se utiliza o passo de otimização de forma adaptativa, o sinal preto e verde representam a saída do sistema quando este parâmetro assume um valor constante, para o sinal preto ρ é igual a 0,001 e para o sinal verde ρ é igual a 0,002.

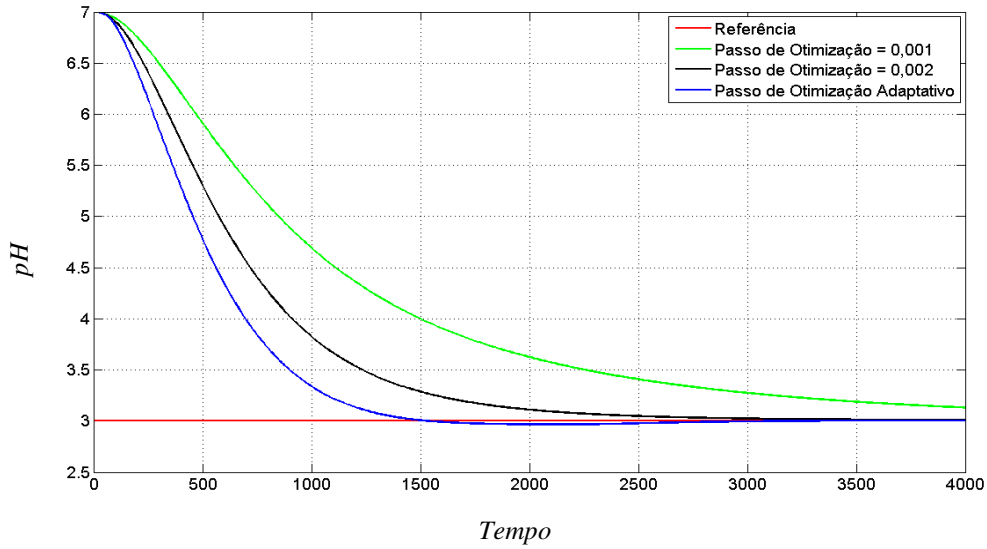

Figura 5.17. Resposta para diferentes passos de otimização

Analisando o resultado é possível concluir que este método leva vantagem em relação ao passo de otimização constante, pois conseguiu estabilizar a saída do sistema na referência desejada em um menor tempo.

O método do passo de otimização adaptativo mostrou vantagens ao ser utilizado e a importância deste parâmetro ter um ajuste preciso foi comprovada, pois ao aumentar o seu valor o sistema mostrou uma resposta mais rápida e ao continuar este aumento o sistema vai ser levado a instabilidade. Ao diminuí-lo a saída do sistema se comporta de forma mais lenta, fazendo com que o processo não atinja a referência desejada.

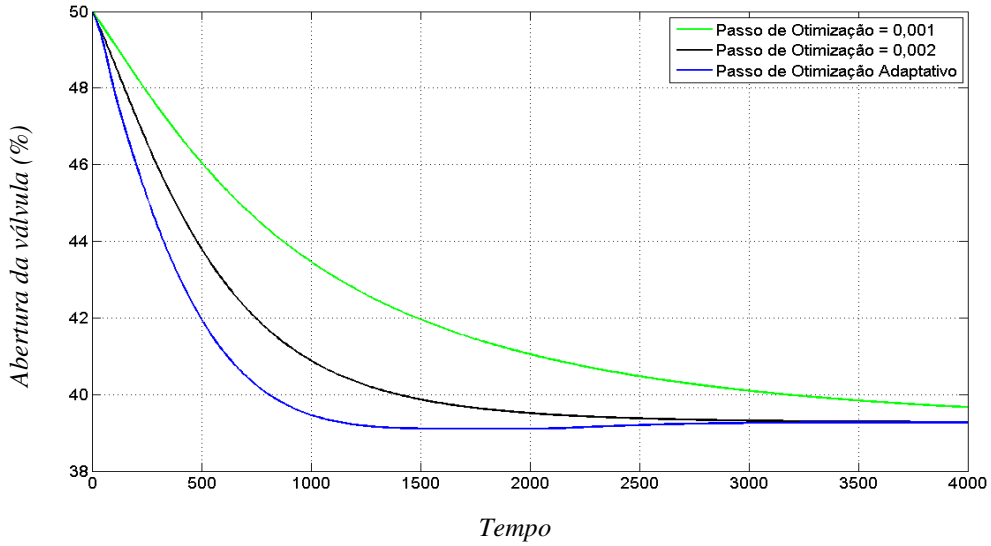

Figura 5.18. Sinais de controle para diferentes passos de otimização

5.5.4. Comparativo entre diferentes controladores

Para avaliar o desempenho do controlador é interessante realizar comparações de seu desempenho com o de outros tipos de controladores (Preditivo escalonado e PI). Desta forma é possível descrever as vantagens e as desvantagens do controlador Preditivo Neural sobre outros controladores.

Os controladores, PI e Preditivo escalonado, utilizados neste trabalho foram desenvolvidos em Fontes et al. (2008). Para as simulações foram utilizados as mesmas parametrizações descritos no trabalho citado anteriormente. Os parâmetros do controlador preditivo neural são os mesmos mencionados no item 5.5.1 deste trabalho com apenas uma alteração no valor da ponderação do sinal de controle (λ), que teve o seu valor foi ajustado para 0,67.

A figura 5.19 mostra simulações efetuadas com os três controladores citados anteriormente. Como pode ser observado ambos os controladores conseguem fazer com que o sinal de saída do processo atinja a referência desejada, no caso pH igual a 3, porém o controlador Preditivo Neural atua de forma mais suave e mais lenta do que os demais, fazendo com que o sistema demore um pouco mais para entrar em regime permanente. Desta forma este tipo de controle pode ser caracterizado por possuir variações de ordens bem menores do que os demais controladores.

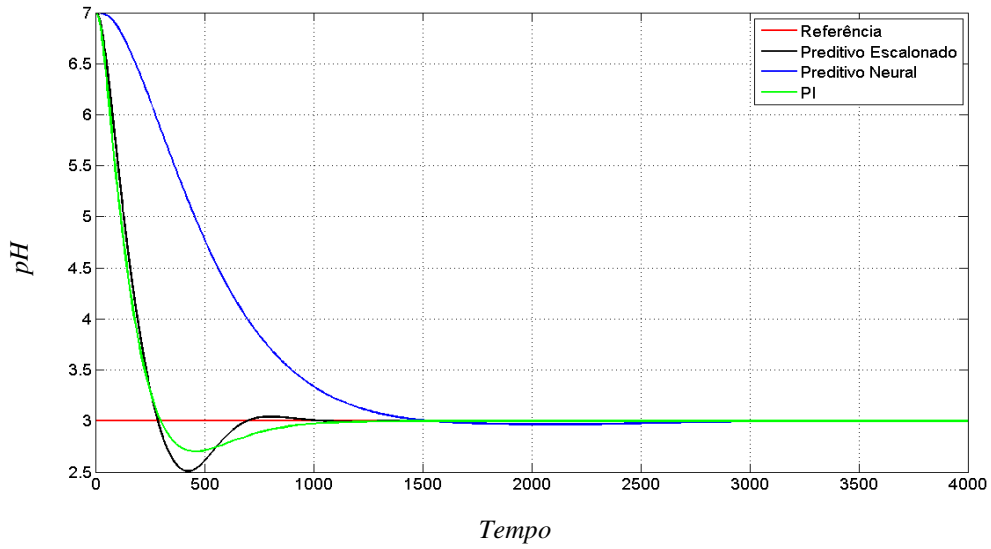

Figura 5.19. Sinais de saída (PI, Preditivo Escalonado e Preditivo Neural)

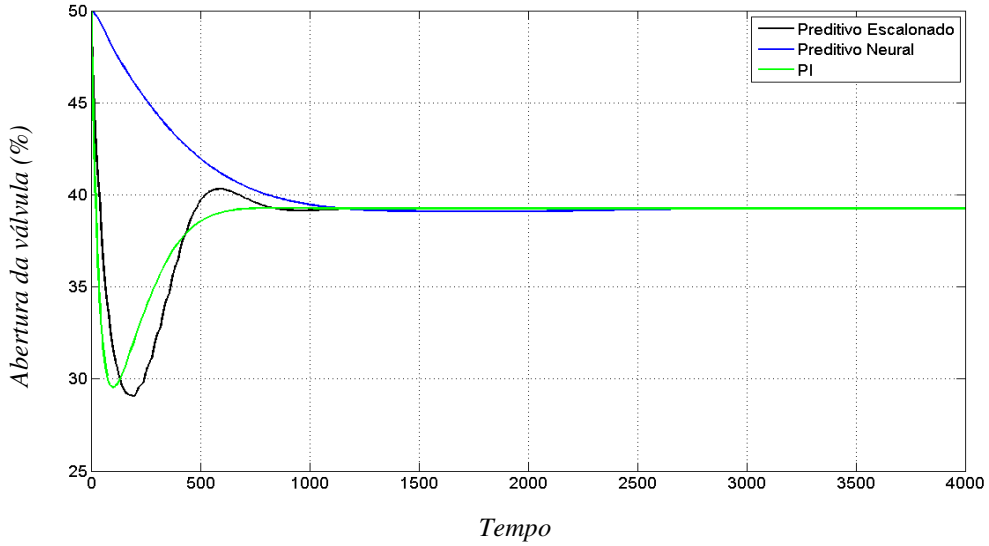

Figura 5.20. Sinais de controle (PI, Preditivo Escalonado e Preditivo Neural)

Para a avaliação do desempenho do controlador preditivo neural se faz necessário a utilização índices de desempenho, estes índices estão presentes em Goodhart et al. (1994) e são classificados da seguinte forma: média e a variância dos sinais de controle (u_m e u_{var}, respectivamente), a Integral do Erro Absoluto (IAE) e a Integral do Erro Absoluto com Ponderação de Tempo (ITAE). As equações dos indicadores de desempenho são definidos da seguinte forma:

- Esforço de controle médio (u_m):

$$u_m = \frac{1}{N}\sum_{i=1}^{N} u(i) \tag{5.5}$$

em que u(i) corresponde aos pontos da curva do esforço de controle e N corresponde ao número de pontos da curva de teste.

- Variância do esforço de controle (u_{var}):

$$u_{var} = \frac{1}{N}\sum_{i=1}^{N} [u(i) - u_m]^2 \tag{5.6}$$

em que u(i) e N são os mesmos do item anterior, e u_m está definido na equação 5.5.

- Integral do valor absoluto do erro (IAE):

A integral do valor absoluto do erro que penaliza o erro entre a referência e a saída controlada e sua expressão no domínio do tempo é dada por:

$$IAE = \int_{T_o}^{T} |e| dt \tag{5.7}$$

sendo T_o e T os limites de avaliação e |e| o erro absoluto sem ponderação do tempo.

- Integral do valor absoluto do erro com ponderação do tempo (ITAE):

A ITAE penaliza o erro absoluto ao longo do tempo, sua expressão é dada por:

$$ITAE = \int_{T_o}^{T} t.|e| dt \tag{5.8}$$

Sendo T_o e T os limites de avaliação e t.|e| o erro absoluto com ponderação do tempo.

A tabela 5.1 mostra todos os valores calculados a partir dos índices determinados acima.

Controlador	u_m	u_{var}	IAE	ITAE
PI	38,51	26,02	0.15	26,91
Preditivo Escalonado	38,69	5,52	0.17	28,15
Preditivo Neural	40,15	5,14	0.51	191,46

Tabela 5.1. Índices de desempenho

O controle preditivo neural obteve um bom desempenho com relação aos outros controladores (PI e Preditivo escalonado), apesar de apresentar um tempo de resposta maior em malha fechada, o controlador preditivo neural obteve um menor *overshoot*, uma menor variância do sinal de controle, desta forma é possível afirmar que a vantagem da utilização deste controlador esta na forma suave de sua resposta para este exemplo em particular.

Neste capítulo foram realizados diversos testes do controlador preditivo neural em um processo de neutralização de pH. Desta forma foi possível analisar o seu funcionamento e as vantagens e desvantagens de sua utilização. No próximo item do trabalho são discutidos os resultados obtidos e as perspectivas futuras para o aprimoramento deste tipo de controlador.

CONCLUSÃO E PERSPECTIVAS

O estudo realizado é focado no desenvolvimento de um Controlador Preditivo Neural. Este tipo de controle utiliza como modelo interno uma rede neural artificial, que é responsável pela identificação de sistemas, uma etapa considerada fundamental em aplicações de estratégias de controle avançado. A técnica de identificação neural de sistemas lineares e não-lineares tem mostrado resultados promissores.

O controlador foi aplicado a um sistema de neutralizador de pH, que fora apresentado no capítulo 5, um processo que traz grandes desafios por apresentar não-linearidade estática, como comentado anteriormente.

Inicialmente, para a realização dos testes de controle se faz necessário gerar um modelo neural do processo. Diversas arquiteturas de redes - mudança no número de nós na camada intermediária, no valor do momento e alteração na quantidade de regressores - foram utilizadas para atingir um resultado satisfatório. Assim, foi possível observar que a RNA foi capaz de representar os sistemas desejados de forma bastante precisa, gerando erros na ordem de 10^{-4} entre as saídas reais e as saídas do modelo neural.

Na primeira simulação os parâmetros do controlador foram ajustados de forma a originar os melhores resultados possíveis. O que se pode concluir é que o controle do processo foi realizado de forma satisfatória, onde o sistema conseguiu acompanhar as diversas referências desejadas.

Para análise do controlador foram feitas diversas mudanças nos seus parâmetros (horizonte de predição e de controle, e ponderação do sinal de controle e ajuste do passo de otimização) e foi possível observar a influência de cada um no resultado final do processo.

O emprego do passo de otimização adaptativo forneceu uma melhoria no controle do processo, visto que este parâmetro pode causar desde lentidão ao sistema até fazer com que

ele se torne instável e ao utilizá-lo na forma adaptativa, o seu valor se ajusta ao longo da simulação para proporcionar o melhor desempenho possível. Os resultados mostraram uma significativa melhora com a utilização desta técnica.

Ao comparar o controlador Preditivo Neural com o Preditivo escalonado e o PI, conclui-se que este controlador age de forma mais suave no processo, gerando uma variação de controle menos agressiva do que os outros controladores. Esta característica pode ser uma vantagem quando se deseja ter este tipo de comportamento do sinal de controle.

Uma desvantagem deste controlador sobre os outros é com relação ao número de parâmetros a serem ajustados, além do horizonte de predição, horizonte de controle e ponderação do sinal de controle é necessário ajustar os parâmetros da RNA, como: número de regressores, arquitetura da rede e número de épocas. Desta forma o ajuste do controlador se torna complexo, devido à grande quantidade parâmetros.

Os próximos passos a serem seguidos são a realização de melhoria no controlador para que possam alcançar resultados ainda melhores. Outra mudança a ser analisada e estudada é a relação do número de regressores com o valor do o horizonte de predição e controle, pois ao romper esta dependência será possível ter um maior grau de liberdade para os ajustes destes parâmetros (*NY*, *NU* e regressores). O último ponto a ser estudado é tornar a RNA mais genérica, para que seja possível escolher mais do que uma camada intermediária.

REFERÊNCIAS

CERQUEIRA, J. de J. F. *Identificação de Sistemas Dinâmicos Usando Redes Neurais Artificiais: Uma Aplicação a Manipuladores Robóticos.* Tese de Doutorado, Universidade Estadual de Campinas. São Paulo: Campinas, 2001.

EYNG, E., *Controle Feedforward Baseado em Redes Neurais Aplicado a Coluna de Absorção do Processo de Produção de Etanol.* Dissertação de Mestrado. Universidade Estadual de Campinas. São Paulo: Campinas, 2006.

FONTES, A. B., **SANTOS**, M. B., **SOUZA**, R. A. R., **ACHY**, A. R. A., **MAITELLI**, A. L., **CAMPOS**, M. C. M. M., *Técnicas de Controle Aplicadas em um Processo de Controle de pH*. Congresso Brasileiro de Automática. Minas Gerais: Juiz de Fora, 2008.

GABRIEL, O. F., *Contribuições à Análise de Robustez de Sistemas de Controle Usando Redes Neurais*. Tese de Doutorado, Universidade Federal do Rio Grande do Norte. Rio Grande do Norte: Natal, 2004.

GOODHART, S. G., **BURNHAM**, K. J., **JAMES**, D. J. G.,"Bilinear self-tuning control of a high temperature heat treatment plant". IEE Procedure, Control Theory Applications, vol. 141. n. 1, 1994.

HAYKIN, S. *Neural Networks*. Macmillan College Publishing Company. Canadá: Ontario, 1994.

JÚNIOR, F. L. C. *Controle Preditivo Utilizando um Modelo Nebuloso.* Dissertação de Mestrado. Universidade Estadual de Campinas. São Paulo: Campinas, 2000.

JÚNIOR, J. M. P. de M. *Redes Neurais Dinâmicas para Predição e Modelagem Não-Linear de Séries Temporais.* Dissertação de Mestrado. Universidade Federal do Ceará. Ceará: Fortaleza, 2006.

KWOK, K.-Y., SHAH, S.L. *Long-Range Predictive Control with a Terminal Matching Condition*, *Chemical Engineering Science*, vol. 49, n. 9, p. 1287-1300, 1994.

LIN, C.-T. & **LEE**, C. G. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice Hall Inc, 1996.

LUCENA, P. B., *Análise de um Controlador Baseado no Jacobiano Estimado da Planta Através de uma Rede Neural*. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte. Rio Grande do Norte: Natal, 2005.

MAITELLI, A. L., **GABRIEL**, O. F. *Controlador Híbrido Indireto Baseado em Redes Neurais – Parte I: Desenvolvimento e implementação.* VI Simpósio Brasileiro de Automação Inteligente. São Paulo: Bauru, 2003.

MAITELLI, A. L., **FONTES**, A. de B., *Apostila de Controle Preditivo*. Universidade Federal do Rio Grande do Norte. Rio Grande do Norte: Natal, 2005.

MAMANI, A. B. A., *Utilização de Redes Neurais no Controle da Velocidade de um Veículo Experimental*. Dissertação de Mestrado. Universidade Estadual de Campinas. São Paulo, 2004.

MEDEIROS, J. A. C. C., *Enxame de Particulas como Ferramentas de Otimização em Problemas Complexos de Engenharia Nuclear*. Tese de Doutorado. Universidade Federal do Rio de Janeiro. Rio de Janeiro: Rio de Janeiro, 2005.

MONTAGUE, G. A., **WILLIS**, M. J., **THAM**, M. T., **MORRIS**, A. J., *Artificial Neural Network Based Multivariables Presictive Control*. University of Newcastle. United Kingdom: Tyne.

NARENDRA, S. K., **PARTHASARATY**, K., *Identification and Control of Dynamical Systems Using Neural Networks*. IEEE Trans. on Neural Networks , vol. 1, n.1. March 1990.

NØRGAARD, M., **RVN**, O., **POULSE**, N.K. e **HANSEN**, L.K., *Neural Networks for Modelling and Control of Dynamic Systems*. Springer-Verlag London Limited. England: London, 2001.

OGUNNAIKE, B.A., **RAY**, W.H., *Process Dynamic, Modeling and Control*. Oxford University Press. USA: New York, 1994.

PALÚ, F., *Controle Preditivo de Colunas de Absorção com o Método de Controle por Matriz Dinâmica*. Tese de Doutorado. Universidade Estadual de Campinas, São Paulo: Campinas, 2001.

RAWLINGS, J.B. *Tutorial Overview of Model Predictive Control*. IEEE Control Systems Magazine, vol. 20, p. 38-52, 2000.

REZENDE, J. A. D., *Implementação em Tempo Real de um Esquema de Controle Adaptativo Neural*. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte, Rio Grande do Norte: Natal, 1999.

RUMELHART, D. E., **MCCLELLAND**, J. L. *Paralled Distributed Processing: Explorations in the Microstructure of cognition* . vol. 1: Foundations. Cambridge: The Mit Press, 1986.

SANTOS, J. E. S. Dos. *Controle Preditivo Não-Linear para Sistemas de Hammerstein.* Tese de Doutorado, Universidade Federal de Santa Catarina, Santa Catarina: Florianópolis, 2003.

SANSEVERO, G., *Controle Preditivo Baseado em Modelo para Turbo-Geradores Hidráulicos Tipo Francis.* Dissertação de Mestrado. Universidade Estadual de Campinas. São Paulo: Campinas, 2006.

SCHNITMAN, L., *Controladores Preditivos Baseado em Redes Neurais Artificiais*. Dissertação de Mestrado. Universidade Federal da Bahia. Bahia: Salvador, 1998.

SOLOWAY, D., **HALEY**, P., *Neural Generalized Predictive Control*. IEE Symposium on Inteligente Control. Dearborn, MI, 1996

SOUZA JR., M. B. De & PINTO, J. C., Lima. E. L., *Controlo of a chaotic polymerization reactor: a neural network based model predictive approach*. Polymer Engineering and Science. vol. 36, n. 4, p. 448-457, February, 1996.

SUN, H., LI, P., **ZHOU**, L., *A Strategy of Generalized Predictive Control Based on Neural Network*. Third Internacional Conference on Machine Learning and Cybernetics. China: Shanghai, 2004.

VALE, M. R. B. G., **FONSECA**, D. G. V., **MAITELLI**, A. L., **ARAÚJO**, F. M. U., *Controle Adaptativo por Modelo de Referência Aplicado em uma Planta de Neutralização de pH*. VIII Conferência Internacional de Aplicações Industriais. Minas Gerais: Poços de Caldas, 2008.

VALL, O., **RADHI**, M., *An approach to the closed loop identification of the Wiener systems with Variable Structure Controller using na Hybrid Neural model*. IEEE International Symposium on Industrial Electronics. vol. 4, p. 2654-2658, 2006.

VIEIRA, W. G., *Process identification with artificial neural networks apllied to experimental data from a continuos distillation column.* Congresso Brasileiro de Redes Neurais. Rio Grande do Norte: Natal, 2005.

VIEIRA, W. G., *FCC: Controle Preditivo e Identificação Via Redes Neurais*. Tese de Doutorado. Universidade Estadual de Campinas. São Paulo: Campinas, 2002.

ZAMBRANO, D., **CAMACHO**, E.F., *Application of MPC with Multiple Objective for a Solar Refrigeration Plant*. *IEEE Conference on Control Applications*. Escócia: Glascow, p. 1230-1235, 2002.

ZENG, G. M., **QIN**, X. S., **HE**, L., **HUANG**, G. H., **LIU**, H. L., **LIN**, Y. P., *A Neural Network Predictive Control System for Paper Mill Wastewater Treatment*. Engineering Applications of Artificial Intelligence. vol. 16, n. 2, p. 121-129, March, 2003.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo