PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA

IMPLEMENTAÇÃO DE UM NEURO-CONTROLADOR PREDITIVO COM OTIMIZAÇÃO POR SEÇÃO ÁUREA APLICADO EM UM PROCESSO DE NEUTRALIZAÇÃO DE pH

Silvano Fonseca Paganoto

Dissertação submetida à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia do Centro Universitário do Leste de Minas Gerais, como parte dos requisitos necessários à obtenção do grau de Mestre em Engenharia Industrial.
Área de Concentração: Processos Industriais

Orientador: Dr. Roselito de Albuquerque Teixeira

Coronel Fabriciano, Dezembro de 2008

PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA

IMPLEMENTAÇÃO DE UM NEURO-CONTROLADOR PREDITIVO COM OTIMIZAÇÃO POR SEÇÃO ÁUREA APLICADO EM UM PROCESSO DE NEUTRALIZAÇÃO DE pH

Silvano Fonseca Paganoto

Banca:

Prof. Dr. Roselito de Albuquerque Teixeira - UnilesteMG (orientador)
Prof. Dr. Marcelo Vieira Corrêa - UnilesteMG
Prof. Dr. Dair José de Oliveira - UnilesteMG
Prof. Dr. Wilian Soares Lacerda - Universidade Federal de Lavras, UFLA

Ficha catalográfica

P131i Paganoto, Silvano Fonseca
Implementação de um neuro-controlador preditivo com otimização por seção áurea aplicado em um processo de neutralização de pH / Silvano Fonseca Paganoto. — 2008.
xxvi, 108 f. : il.

Dissertação (mestrado) - Centro Universitário do Leste de Minas Gerais - Unileste-MG, 2008.
"Orientação: Roselito de Albuquerque Teixeira"

1. Redes Neurais Artificiais (RNA). 2. Sistemas dinâmicos. I. Titulo.

CDU -004.032.26

À minha Esposa e meus Pais

Agradecimentos

Agradeço a Tatiana, minha esposa, pelo companheirismo, amor e dedicação incondicional, sendo um pilar de sustentação e ao mesmo tempo um refúgio nos momentos difíceis passados durante esta jornada.

Agradeço aos amigos e familiares que indiretamente estiveram contribuindo e torcendo para a realização deste sonho.

Agradeço aos professores Dr. Roselito, Dr. Marcelo, Dr. Figueiredo, Dr. Esly e Dra. Andréa, pelo empenho e dedicação ao transmitir seus conhecimentos e estarem sempre dispostos a contribuir na minha formação acadêmica.

Agradeço a Rodrigo, Willian e Gláucio pelo apoio nos ajustes, no preparo de soluções e na realização dos testes com os neuro-controladores na planta piloto de neutralização de pH.

Agradeço aos colegas do mestrado por serem elos de motivação nos momentos de desânimo, pelo intercâmbio de conhecimentos e pelo auxílio durante a realização das pesquisas em todas as disciplinas.

Enfim, agradeço a Deus por ser o autor de tantas maravilhas em minha vida, por permitir todos estes acontecimentos durante este período de mestrado e por conceder que mais uma graça fosse alcançada.

"O único homem que está isento de erros, é aquele que não arrisca acertar."

Albert Einstein

Resumo

Em controle de sistemas dinâmicos não-lineares é comum o emprego de técnicas de linearização. Contudo, em processos com necessidades operacionais complexas, dinâmicas e com maior gama de funcionamento, o controle baseado em sistemas linearizados nem sempre apresenta resultados satisfatórios. Objetivando a busca de soluções para esse problema de controle, apresenta-se esta pesquisa sobre o controle de sistemas dinâmicos por meio do uso de Redes Neurais Artificiais (RNA). Devido à característica de serem mapeadoras universais de funções e de serem passivas de treinamento, as RNAs proporcionam diversas formas de implementações em sistema de controle, caracterizadas como Neuro-controladores, sendo os mais difundidos o Neuro-controlador por Modelo Inverso e o Neuro-controlador Preditivo. De forma a ter uma fundamentação teórica é realizada uma introdução sobre RNA destacando a arquitetura, o processo de aprendizagem, mecanismos de aumento de generalização e aplicação em identificação de sistemas dinâmicos, bem como a sua utilização na arquitetura de controle, abordando-as em Neuro-controladores por Modelo Inverso e Neuro-controladores Preditivo, sendo este explorado com maior profundidade. Pretendendo-se eliminar pontos de fragilidades em neuro-controladores encontrados na literatura é proposto um Neuro-controlador Preditivo que utiliza o método de otimização por eliminação de regiões e busca por seção áurea. A análise de desempenho do neuro-controlador proposto é obtida por meio de testes em processo simulado e em processo real, por meio de uma planta piloto de neutralização de pH. Os resultados dos testes de rastreabilidade e rejeição de distúrbios indicam que o neuro-controlador proposto apresenta evidências de um melhor desempenho comparado a algumas técnicas de controle, além de possuir simplicidade de parametrização.

Abstract

In the control of nonlinear dynamic systems, the use of linearization techniques is very common. However, in cases of complex and dynamic operational needs with greater range of operating systems, the control based on linearized systems doesn't always show satisfactory results. Searching for solutions to this control problem, this work aims at researching the control of dynamic systems through the use of Artificial Neural Networks (ANN). Due to the characteristic of being universal mappers of functions and of being training passive, the ANNs provide various forms of implementations in control system, known as Neuro-controllers, and the most widespread are the Inverse Model Neuro-controller and the Predictive Neuro-controller. In order to have a theoretical basis, an introduction to ANN is being conducted, highlighting the architecture, the learning process, the mechanisms of increasing and widespread use in identification of dynamical systems, as well as their use in the control architecture, addressing them in Inverse Model Neuro-controllers and Predictive Neuro-controllers, which is explored in greater depth. Aiming at eliminating points of weakness in neuro-controllers found in literature, a Predictive Neuro-controller that uses the method of optimization by eliminating regions and search for golden section is proposed. The review of its performance is obtained by testing it in real and simulated processes, through a pilot plant of pH neutralization. The test results of tracking and rejection of disturbances indicate that the proposed neuro-controller seems to have an improved performance compared to some techniques of control, besides having simplicity of parameterization.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

N	Identificação para neurônio
x	Sinal de entrada do neurônio
j	Entrada do neurônio
w_{Nj}	Peso sináptico da entrada j do neurônio N
$\sum$	Somtório
v	Nível de ativação interna ou campo local induzido
v_N	Nível de ativação interna ou campo local induzido do neurônio N
φ	Função de ativação do neurônio
y_N	Saída do neurônio N
k	Iteração
φ'	Derivada da função de ativação do neurônio
$\Delta w_{Nj}\,(k)$	Correção no peso do neurônio N na iteração k
η	Taxa de aprendizagem
$\delta_N\,(k)$	Gradiente local do neurônio N na iteração k
$\mathbf{x}$	Vetor de entradas do neurônio
$\mathbf{W}$	Matriz de peso do neurônios
$\varepsilon_s(\mathbf{W})$	Função de custo (treinamento de RNAs)
$\lambda\varepsilon_c(\mathbf{W})$	Termo de punição ou penalidade
$\mu(\mathbf{x})$	Função de ponderação
∂	Derivada parcial
$F(\mathbf{x},\mathbf{W})$	Mapeamento de entrada-saída realizado pelo modelo
$g\,(w)$	Vetor gradiente avaliado em w
$H\,(w)$	Matriz *Hessiana* avaliado em w
A^T	Transposta de A
$\mathcal{F}$	Função não-linear
$u(k)$	Ação de controle no instante k
z^{-1}	Operador de atraso

n	Memória de linha de atraso derivada para y
m	Memória de linha de atraso derivada para u
$y(k)$	Saída do processo no instante k
$\hat{y}(k+1)$	Saída estimada do processo para o instante $k+1$
$r(k+1)$	Sinal de referência para o instante $k+1$
$e(k)$	Erro no instante k
$\hat{e}_c(k+1)$	Erro do controlador estimado para o instante $k+1$
ξ	Resíduo
$\mathbf{W_1}$	Matriz de pesos dos neurônios da camada escondida
$\mathbf{w_2}$	Vetor de pesos dos neurônios da camada de saída
$\mathbf{b}$	Vetor dos termos de polarização dos neurônios da camada escondida
b_s	Termo de polarização do neurônio da camada de saída
∇	Operador diferenciação (Gradiente)
$\lambda(k)$	Tamanho do passo (Gradiente descendente)
a	Limite inferior da região de incerteza (otimização)
b	Limite superior da região de incerteza (otimização)
L	Dimensão da região de incerteza ($b-a$)
f_m	Função que represente um modelo de referência
$d(k)$	Distúrbio no instante k
Q	Vazão

Siglas e Abreviações

ARX	Modelo linear auto-regressivo com entradas exógenas (*Auto Regressive model with eXogenous inputs*)
MIMO	Múltiplas Entradas e Múltiplas Saídas (*Multi-Input Multi-Ouput*)
MLP	Redes neurais multi-camadas (*Multilayer Perceptron*)
MSE	Média do somatório dos erros quadráticos (*Mean Squared Errors*)
PID	Proporcional Integral Derivativo
RBF	Redes de funções de base radiais (*Radial Basis Function*)
RNA	Rede Neural Artificial
SISO	Uma Entrada e Uma Saída (*Single-Input Single-Output*)
TDNN	Redes Neurais com Atraso no Tempo (*Time Delay Neural Network*)
USB	Barramento Serial Universal (*Universal Serial Bus*)

Capítulo 1

Introdução

Em controle de sistemas dinâmicos não-lineares é comum o emprego de técnicas de linearização (Isidori, 2000; Khalil, 2002), pois muitos sistemas não-lineares podem ser representados por um sistema linear sem perda significativa de suas propriedades (Schwarzenbach e Gill, 1992). Contudo, exigências ambientais, uso de insumos, exigências de qualidade dos produtos, flexibilidade na produção e outros ítens exigem do processo condições operacionais mais complexas, dinâmicas e com maior gama de funcionamento. Nestes casos, o controle baseado em sistemas linearizados nem sempre apresenta resultado satisfatório (Te Braake et al., 1998).

Alvarez et al. (2001) também comentam que projetar controladores para sistemas não-lineares pelo desenvolvimento analítico de controladores convencionais (Ex.: controlador Proporcional Integral Derivativo - PID) não é garantia de funcionamento satisfatório em toda a faixa de operação.

Em virtude desses problemas, existe um crescente interesse em controladores que são baseados na teoria de controle não-linear, métodos que assumem o conhecimento da forma de controle (controlador por lógica nebulosa ou *fuzzy*), métodos que assumem o conhecimento do comportamento do sistema (controladores preditivos baseado em modelo) e métodos que não incorporam conhecimento do controle nem do sistema, como controladores com Aprendizagem por reforço (Riedmiller, 1999).

Riedmiller (1994) diz que basicamente um controlador se diferencia em tipo e qualidades de informações incorporadas à priori. Como as Redes Neurais Artificiais (RNAs) possuem características de serem mapeadoras universais de funções e são passivas de treinamento, estas são de grande importância para uso em controle de sistemas (Hunt et al., 1992; Fukuda e Shibata, 1992).

Dentre as diversas arquiteturas que as RNAs apresentam, o desenvolvedor pode escolher a que melhor atenda à sua aplicação. Por exemplo, em aplicação de modelagem de sistemas dinâmicos, a arquitetura chamada de Redes Neurais Recorrente de Entrada-Saída é amplamente usada (Haykin, 1999).

Em um sistema de controle que faz uso de uma RNA, o controlador recebe o nome de Neuro-controlador e dependendo de sua disposição dentro da arquitetura de controle este é dividido em Neuro-controlador por Modelo Inverso e Neuro-controlador Preditivo.

Como mencionado anteriormente, para o controle de sistemas dinâmicos não-lineares com operações críticas, é justo projetar um sistema de controle baseado em teoria não-linear. Assim, os neuro-controladores, por possuir uma RNA em sua arquitetura, apresentam propriedades favoráveis ao emprego nesse tipo de sistemas.

No seguimento de Neuro-Controladores, Narendra e Mukhopadhyay (1994) e Sanner e Slotine (1995) apresentam controladores adaptativos multivariáveis usando RNA. Liu et al. (1999) apresentam um neuro-controlador adaptativo com RNA do tipo RBF com capacidade de variar o número de funções de base radial para evitar super-ajuste durante a adaptação, garantindo estabilidade e convergência do sistema de controle por meio da teoria de estabilidade de Lyapunov. Warwick et al. (2000) propõem uma arquitetura de neuro-controlador robusto e adaptativo multivariável.

Constantin (2003) apresenta técnicas de adaptação para neuro-controladores aplicado em sistemas não-lineares e Ge et al. (2004) expõe um tipo de controlador adaptativo para sistemas com atraso no tempo desconhecido. Assim, não é necessário incorporar informação à priori na RNA sobre a quantidade de atraso. Pozas (2005) aplica controladores preditivos não-lineares usando RNA em processos de refino de petróleo.

Boukabou e Mansouri (2005) propõem um neuro-controlador preditivo para controle de sistema caótico com o estabelecimento de que a dinâmica de qualquer sistema não-linear é aproximadamente a de um sistema linear em torno de um ponto fixo. Žilková et al. (2006) aplica um neuro-controlador por modelo inverso em um sistema de controle de velocidade para motores de indução. Ying et al. (2007) apresentam um neuro-controlador preditivo com otimização conhecida como *Tent-Map Chaos Optimization* que apresenta melhor resultado que o neuro-controlador preditivo com método de otimização por Gradiente Descendente ou com Quasi-Newton.

No cenário de controle não-linear, o processo de neutralização de pH tem despertado muito interesse, devido ao alto grau de dificuldade de controle (Wilson e Wylupek, 1969), ocasionando o emprego de técnicas de controle não-linear, sobretudo os neuro-controladores.

Nahas et al. (1992) aplicam um neuro-controlador com modelo interno ao processo de neutralização de pH em um reator contínuo perfeitamente agitado (*CSTR - Continuous Stirred Tank Reactor*) em conjunto a um preditor de Smith para compensação do tempo morto. Doherty et al. (1995) aplicam um neuro-controlador preditivo em processo semelhante.

Doherty (2000) apresenta uma comparação entre técnicas de controle que faz uso de RNAs e controladores lineares aplicados ao processo de neutralização de pH. Seus resultados indicam um desempenho superior para o neuro-controlador.

Syafiie et al. (2005) propõe uma alteração na política de aprendizagem por reforço do neuro-controlador que eleva o desempenho do controlador quando aplicado ao processo de neutralização de pH.

Douratsos e Gomm (2007) apresentam resultados de neuro-controladores adaptativos aplicados em um processo de neutralização de pH e os comparam com o de técnica de controle linear. É ressaltado a superioridade do desempenho do neuro-controlador, mesmo durante a fase de adaptação.

Nota-se portanto, o crescimento da utilização de neuro-controladores em diversos seguimentos, isto devido à diversidade de arquiteturas que estes permitem, bem como aliado ao seu potencial de controle, motivando assim, seu estudo e implementação em controle de processos dinâmicos, como apresentado no decorrer deste trabalho.

Assim, inicialmente é realizado uma introdução sobre RNA e sua utilização em modelagem de processo dinâmico seguido de noções e arquiteturas de neuro-controladores, que, por conseguinte, culminam na proposta de um neuro-controlador preditivo com otimizador por método de eliminação de regiões, dotado de busca por seção áurea, cujo desempenho é avaliado em controle de processos simulados e, em uma planta piloto de neutralização de pH. Seus resultados também são comparados aos do neuro-controlador preditivo com método de otimização por gradiente descendente e controlador PID.

1.1 Objetivos

Pesquisar sobre controle de sistemas dinâmicos por meio do uso de redes neurais artificiais, focando no desenvolvimento e testes do neuro-controlador preditivo, bem como sua variação dotada de modelo de referência.

E em decorrência desse, surgem os seguintes objetivos secundários:

- Propor um neuro-controlador preditivo com elemento de otimização por seção-áurea;
- Desenvolver uma plataforma de testes;
- Realizar testes de algoritmos de controle em sistemas simulados e reais e compará-los entre si.

1.2 Organização do Texto

Este texto está organizado em:

- **Capítulo 2: Redes neurais artificiais:** Apresenta a fundamentação sobre as redes neurais artificiais (RNA) e sua aplicação em identificação de sistemas dinâmicos. Modelo, arquiteturas, funções de ativação de neurônios, processos de aprendizagem e mecanismo de ampliação de generalização compõe esta fundamentação. Focado na identificação de sistemas dinâmicos, também apresenta a conceituação sobre tipos e modelagem de sistemas com RNA, por meio de rede recorrente de entrada-saída.
- **Capítulo 3: Neuro-Controladores:** Apresenta a conceituação de neuro-controlados e suas arquiteturas. Inicialmente são mostradas informações sobre controle de sistemas dinâmicos bem como o uso de RNAs nestes sistemas. Em seguida é exposto o neuro-controlador por modelo inverso focando em sua arquitetura e obtenção da RNA. Por fim, é detalhado o neuro-controlador preditivo expondo sua arquitetura, métodos de otimização e finalizando com um aprofundamento da aplicação do método de otimização por gradiente descendente em neuro-controlador preditivo.
- **Capítulo 4: Neuro-Controlador Proposto:** Apresenta uma introdução sobre o método de otimização por eliminação de região e seção áurea, seguido da conceituação do neuro-controlador proposto. Destaca-se também a arquitetura do neuro-controlador proposto dotado de rede recorrente de entrada-saída, bem como sua configuração com modelo de referência.
- **Capítulo 5: Estudo de Casos, Resultados e Discussões:** Apresenta a avaliação de desempenho do neuro-controlador proposto frente ao neuro-controlador preditivo com otimizador por gradiente descendente e controlador PID. E, os testes realizados em sistema simulado não-linear variante no tempo e processo real (planta piloto de neutralização de pH) em cenários de rastreabilidade e rejeição de distúrbios.

- **Capítulo 6: Conclusões:** Apresenta as conclusões sobre a utilização de neuro-controladores e sobre o neuro-controlador proposto, e também as sugestões para trabalhos futuros.

Capítulo 2

Redes Neurais Artificiais

2.1 Introdução

Este capítulo apresenta uma visão geral sobre as Redes Neurais Artificiais (RNAs), objetivando o fornecimento de subsídios para sua utilização em controle de sistemas, expondo sua conceituação, bem como, particularidades inerentes de sua utilização.

A capacidade de aprendizado das RNAs e a possibilidade de adaptação ao processo motivam o seu uso em sistemas de controle. No entanto, é necessário conhecer os conceitos básicos sobre RNAs, tais como: o algoritmo de treinamento e as estratégias, para garantir uma boa generalização pós-treinamento e aplicabilidade em identificação de sistema dinâmicos não-lineares. Estes e outros tópicos serão detalhados nas seções seguintes deste capítulo.

2.2 Fundamentação teórica

A aplicação de redes neurais artificiais tem sido cada vez mais exploradas em diversos segmentos, nos quais se destacam: classificação de padrões, aproximação de funções e predição e controle de processos, isto devido a uma série de características, tais como:

- não-linearidade;
- mapeamento de entrada-saída;
- adaptabilidade;

- generalização.

Como as redes neurais são modelos matemáticos inspirados no conexionismo do cérebro humano, elas apresentam um sistema de processamento paralelo e distribuído, sendo os neurônios as unidades de processamento capazes de armazenar conhecimento experimental tornando-o disponível para o uso (Haykin, 1999).

O conhecimento experimental é obtido por meio do treinamento. Nesta etapa, os padrões são apresentados à RNA, de forma que a mesma busque obter a capacidade de generalização, ou seja, uma vez cessado o treinamento, a RNA deverá produzir respostas satisfatórias mesmo quando apresentados padrões que não foram utilizados durante o treinamento.

Desta forma, o funcionamento de uma rede neural é semelhante ao de um cérebro humano, nos seguintes aspectos:

- o conhecimento é adquirido pela RNA a partir de seu ambiente, por meio de um processo de aprendizagem;
- forças de conexões entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

Através do processo de aprendizagem, os pesos sinápticos dos neurônios são ajustados, pesos estes que definem o grau de abstração do conhecimento presente no ambiente (padrões). Uma vez que a RNA passou por esse processo de aprendizagem, é dito que a RNA está treinada para operar nesse ambiente. Caso ocorram alterações no ambiente, dentro de certos limites, a RNA pode ser re-treinada para continuar em operação.

2.3 Modelo de um neurônio

O neurônio artificial é uma aproximação simplificada, através de funções matemáticas, de um neurônio biológico, sendo este a unidade fundamental de processamento de informação de uma rede neural (Haykin, 1999). A Figura 2.1 mostra o modelo de um neurônio artificial, podendo ser identificados três elementos básicos, sendo:

1. **sinapses** - ou conexões de entrada, caracterizadas por pesos ou forças próprias. Sendo assim, um sinal x_j na entrada da sinapse j conectada ao neurônio N é multiplicado pelo peso sináptico w_{Nj};

2. **junção de soma** - responsável pela combinação aditiva dos sinais de entrada, realizando a soma ponderada dos sinais de entrada;

3. **função de ativação** - pode ser linear ou não-linear; no segundo caso geralmente restringe a amplitude de saída do neurônio.

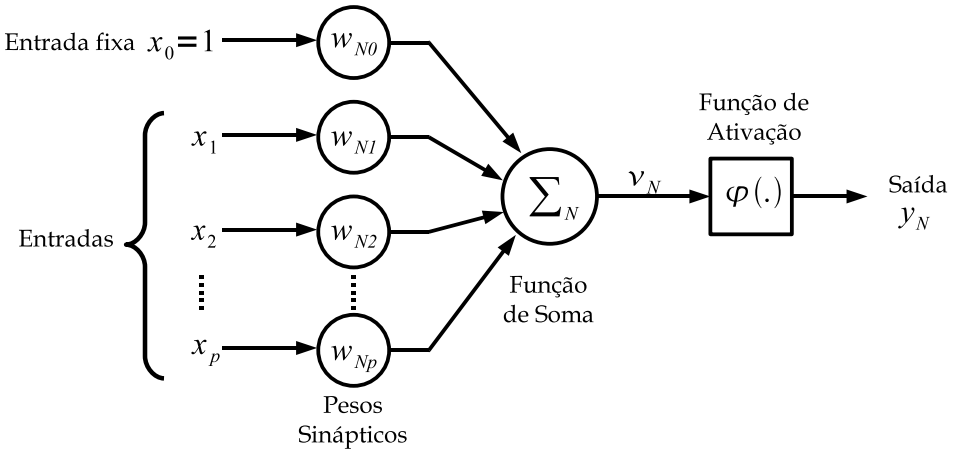

Figura 2.1: Modelo de um neurônio.

O modelo do neurônio da Figura 2.1 pode ser descrito matematicamente pelas Equações 2.1 e 2.2

$$v_N = \sum_{j=0}^{p} w_{Nj} x_j \tag{2.1}$$

$$y_N = \varphi(v_N) \tag{2.2}$$

sendo $x_0, x_1, ..., x_p$ os sinais de entrada, $w_{N0}, w_{N1}, ..., w_{Np}$ os pesos sinápticos do neurônio N, v_N o nível de ativação interna ou potencial de ativação do neurônio N, $\varphi(.)$ a função de ativação e y_N é a saída do neurônio N.

Verifica-se também na Figura 2.1 a presença de $x_0 = +1$, a qual é atribuído o nome de *Termo de Polarização*, cuja função, juntamente com o peso w_{N0} a ela associada, tem o efeito de transladar a função de ativação em torno da origem, fazendo com que a ativação interna v_N do neurônio não seja nula quando todas as demais entradas $x_0, x_1, ..., x_p$ forem nulas.

2.4 Funções de ativação

A função de ativação $\varphi(\cdot)$ determina a saída do neurônio em termos do seu nível de ativação interna v_N, conhecido também como campo local induzido, sendo v_N dado pelo somatório das entradas ponderadas por seus respectivos pesos. Isso dá ao neurônio a capacidade de operação em faixa mais ampla, e em alguns casos até a não-linearidade. As funções de ativação mais comuns são enumeradas abaixo:

1. **Função limiar** - A saída do neurônio é dada pela Equação 2.3 tendo sua representação gráfica na Figura 2.2(a),
$$\varphi(v) = \begin{cases} +1, & \text{para } v \geq 0 \\ -1, & \text{para } v < 0 \end{cases} \tag{2.3}$$

2. **Função linear** - A saída do neurônio é dada pela Equação 2.4, tendo sua representação gráfica na Figura 2.2(b),
$$\varphi(v) = \alpha v, \tag{2.4}$$
sendo α um número real que define a saída linear para os valores de v.

3. **Função sigmoidal tangente hiperbólica** - A saída do neurônio é dada pela Equação 2.5, tendo sua representação gráfica na Figura 2.2(c),
$$\varphi(v) = \frac{1 - e^{-2v}}{1 + e^{-2v}}. \tag{2.5}$$

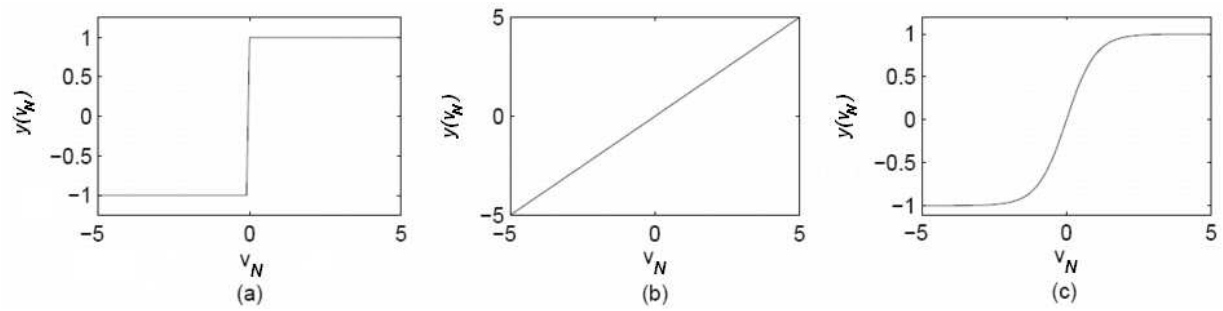

Figura 2.2: Funções de ativação: (a) função limiar, (b) função linear, (c) função sigmoidal tangente hiperbólica.

Observe que para exemplificar as funções de ativação limiar e sigmoidal tangente hiperbólica foi considerada a faixa de variação entre $\pm$ 1. Porém, em algumas situações se faz necessário o uso da faixa de variação entre 0 e 1.

2.5 Arquiteturas e topologia das RNAs

A arquitetura e a topologia definem como os neurônios de uma RNA estarão estruturados e conectados uns aos outros, bem como determinando o fluxo entre eles. Pode-se identificar, segundo Haykin (1999), que a arquitetura das RNAs está divida em:

- Número de camadas:
  - Uma camada (*Ex. Perceptron, Adaline*);
  - Multi-camadas (*Ex. MLP*);
- Número de neurônios em cada camada;
- Conectividade:
  - Completamente conectada;
  - Parcialmente conectada;
  - Localmente conectada.

A topologia de uma RNA está relacionada com a forma de como o fluxo de troca de dados ocorre entre os neurônios, também chamado de arranjo de conexões, podendo apresentar as seguintes formas:

- Redes com Alimentação para Frente (*Feedforward*);
- Redes Recorrentes ou Realimentadas.

Braga et al. (2000) comentam que as redes com Alimentação para Frente (*Feedforward*) possuem características ideais para o mapeamento estático, enquanto as Redes Recorrentes ou Realimentadas apresentam desempenho superior para processamento temporal.

Segundo Iyoda (2000), a camada de entrada não é considerada na contagem do número de camadas, pois esta tem apenas a função de distribuir cada uma das entradas da rede a todos os neurônios da camada seguinte, sem as modificar. Portanto, o número de camadas de uma RNA é definido somente pelas camadas que contém nós computacionais. A Figura 2.3 ilustra uma rede *feedforward* com somente uma camada.

As RNAs que possuem mais de uma camada com nós computacionais apresentam uma capacidade de processamento não-linear elevada, isso devido ao maior número de sinapse entre

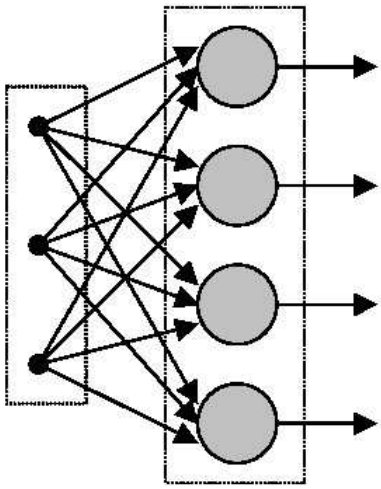

Figura 2.3: RNA *Feedforward* de uma única camada.

as camadas. A camada posicionada entre a camada de entrada e a camada de saída recebe o nome de *camada escondida*. Na Figura 2.4 é ilustrada uma RNA *feedforward* com uma camada escondida e totalmente conectada.

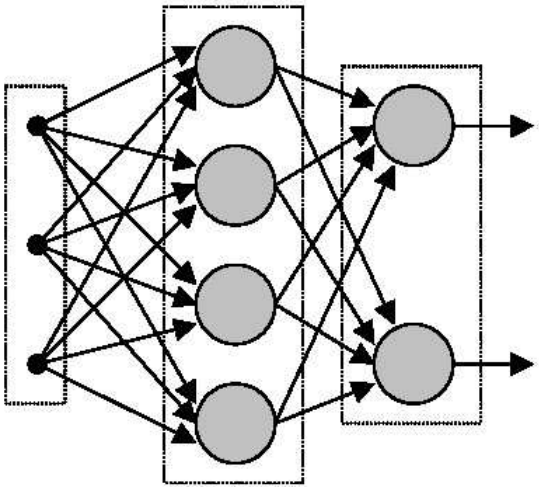

Figura 2.4: RNA *feedforward* com uma camada escondida e totalmente conectada.

As redes recorrentes, por apresentarem um laço de realimentação, aumentam significativamente a capacidade de aprendizagem e o desempenho da RNA para o processamento temporal (Haykin, 1999), sendo úteis à identificação e controle de sistemas dinâmicos. Tais características serão exploradas com mais detalhes na Seção 2.8. A Figura 2.5 apresenta uma RNA recorrente com operadores de atraso unitário.

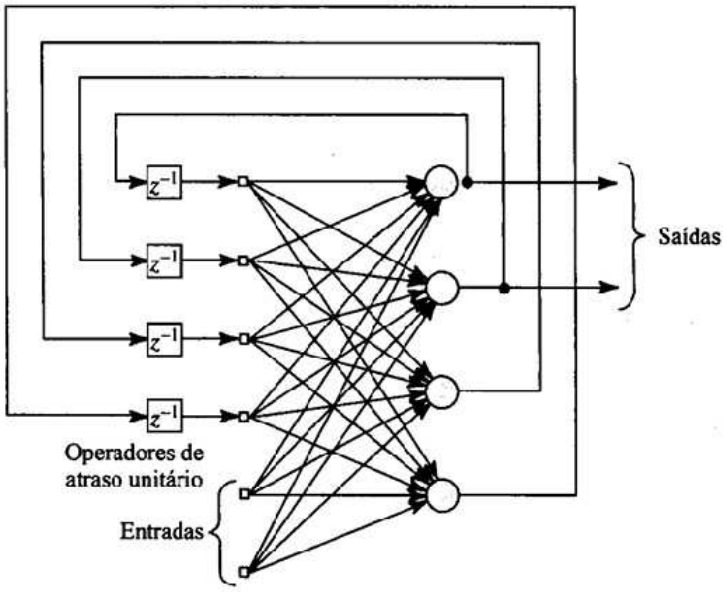

Figura 2.5: RNA recorrente com operadores de atraso unitário.

2.6 Processo de apredizagem das RNAs

Conforme mencionado no início da seção, o processo de aprendizagem define o armazenamento de conhecimento de uma RNA por meio da modificação dos pesos sinápticos, os quais são modificados iterativamente.

Para Haykin (1999) o processo de aprendizagem de uma RNA é definido como: Processo pelo qual os parâmetros livres de uma rede neural são adaptados, através de estímulos do ambiente no qual a mesma está inserida. O tipo de aprendizagem é determinado pela maneira na qual a modificação dos parâmetros ocorre, dado pelo processo de treinamento, que consiste no conjunto de regras bem definidas que leva ao aprendizado da RNA.

Dentre o paradigma de aprendizagem, somente o método de aprendizagem supervisionada será abordado neste trabalho, visto que este é o método utilizado nos algoritmos de abstração de conhecimento usados nos arranjos dos controladores neurais expostos nos capítulos 3 e 4.

O método de aprendizado supervisonado é caracterizado pela presença de um supervisor externo (professor), que após a apresentação dos padrões (entradas e as saídas desejadas) avalia a resposta da rede e ajusta os pesos sinápticos (Braga et al., 2000). Na Figura 2.6 é apresentado um diagrama de blocos da aprendizagem supervisionada.

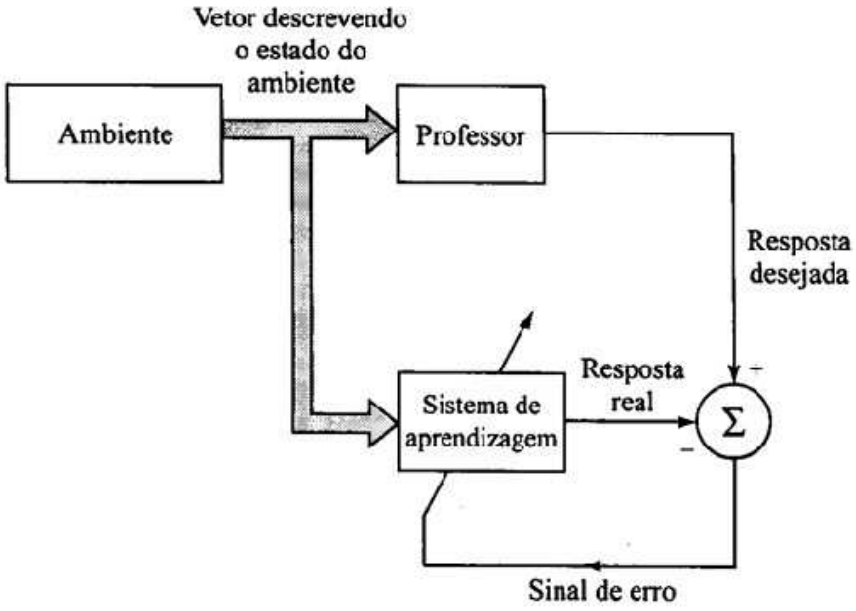

Figura 2.6: Aprendizagem supervisionada.

Os algoritmos para aprendizado supervisionado mais difundidos são a regra delta (Widrow e Hoff, 1960) e o algoritmo de retropropagação do erro (*error back-propagation*) (Rumelhart et al., 1986), sendo este último uma generalização da regra delta para RNAs de múltiplas camadas, o qual terá enfoque neste trabalho.

2.6.1 Algoritmo *error back-propagation*

O algoritmo *error back-propagation* (Rumelhart et al., 1986) é assim nomeado devido ao mecanismo de retropropagação do erro após a apresentação dos padrões, visando ajustar os pesos da rede neural. Esse é um dos algoritmos mais utilizado no treinamento de redes neurais multi-camadas do tipo MLP (*Multilayer Perceptron*) com uma ou mais camadas escondidas.

O treinamento de uma RNA através do algoritmo *error back-propagation* ocorre em dois passos (Rumelhart et al., 1986):

- **Fase propagação:** utilizada para definir a saída da rede para um dado padrão de entrada sendo que os pesos são mantidos fixos, e o sentido do fluxo é da entrada para saída;
- **Fase retropropagação:** utiliza a saída desejada e a saída calculada pela rede na fase de propagação, para a realização do ajuste dos pesos das conexões da rede. Nesta fase o fluxo do sinal de erro é inverso ao da fase de propagação.

O cálculo dos ajustes nos pesos pelo algoritmo de retropropagação do erro é dado pelas seguintes relações:

1. Cálculo da correção dos pesos, proveniente da regra delta (Widrow e Hoff, 1960):

$$\Delta w_{Nj}(k) = \eta \delta_N(k) x_j(k), \tag{2.6}$$

sendo:

$\Delta w_{Nj}(k)$	Correção no peso do neurônio N na iteração k;
η	Taxa de aprendizagem;
$\delta_N(k)$	Gradiente local do neurônio N na iteração k;
$x_j(k)$	Sinal da entrada j do neurônio N na iteração k.

2. Cálculo do gradiente local:

   - Para um neurônio N da camada de saída:

$$\delta_N(k) = e_N(k) \varphi'_N(v_N(k)), \tag{2.7}$$

     sendo $e_N(k)$ o erro entre a saída do neurônio N e a saída desejada na iteração k, $\varphi'_N(v_N(k))$ a derivada da função de ativação do neurônio N em relação à saída linear do neurônio, $v_N(k)$, na iteração k.

   - Para um neurônio N da camada intermediária:

$$\delta_N(k) = \varphi'_N(v_N(k)) \sum_p \delta_p(k) w_{pN}(k), \tag{2.8}$$

     sendo $\varphi'_N(v_N(k))$ a derivada da função de ativação do neurônio N em relação à saída linear do neurônio na iteração k e $\sum_p \delta_p(k) w_{pN}(k)$ a soma ponderada dos gradientes locais da camada seguinte na iteração k.

As funções de ativação utilizadas em uma rede MLP devem ser diferenciáveis, isto devido ao aparecimento do termo $\varphi'_N(v_N(k))$ nas Equações 2.7 e 2.8 do cálculo do gradiente local.

O desenvolvimento completo do algoritmo *back-propagation* de Rumelhart et al. (1986), pode ser visto em detalhes em Haykin (1999) e em Braga et al. (2000).

2.6.2 A taxa de aprendizado

Por definição, o algoritmo *back-propagation* faz uso do método de otimização *Gradiente Descendente* (*steepest descent*) para cálculo espacial da trajetória dos pesos. A taxa de aprendizado η define o tamanho do passo na busca da minimização do erro. Dessa forma, quanto menor η, menor será o ajuste dos pesos Δw, consequentemente mais lento será o processo de treinamento a cada iteração. Por outro lado, valores elevados de η podem instabilizar o algoritmo, uma vez que este pode ultrapassar o ponto ótimo ocasionado por um passo muito grande.

A maneira mais comum de minimização do problema da instabilidade é o acréscimo de um termo na regra de aprendizagem, chamada de *momento*, como mostrado na Equação 2.9

$$\Delta w_{Nj}(k) = \eta \delta_N(k) x_j(k) + \alpha \Delta w_{Nj}(k-1), \tag{2.9}$$

sendo α chamada de constante de momento. Nesse caso, a Equação 2.9 é chamada de regra delta generalizada com momento. Segundo Braga et al. (2000), o termo de momento, além de estabilizar o algoritmo, aumenta a velocidade de aprendizagem em regiões planas da superfície de erro e pode também retirar a rede de mínimos locais.

2.6.3 Modos de treinamento seqüencial e por lote

Durante o treinamento de uma RNA, o processo no qual todos os elementos de um conjunto de padrões são apresentados a uma rede, chama-se **época**. Este processo se dá por dois modos distintos no algoritmo *back-propagation* (Haykin, 1999):

- **Modo seqüencial**, que é também chamado de modo *online*, no qual o ajuste de pesos é realizado após a apresentação de cada padrão à rede.

- **Modo por lote ou batelada**, que é também chamado de modo *batch*, no qual o ajuste de pesos é realizado após a apresentação de todos os padrões à rede.

No modo seqüencial um conjunto de dados para treinamento com m padrões, ao final de uma época terá realizado m ajustes nos pesos, e no modo batelada o mesmo conjunto terá realizado apenas 1 ajuste nos pesos, porém, este ajuste leva em conta os erros obtidos em todos os padrões.

2.6.4 Critérios de parada

Durante o processo de treinamento de uma RNA, em algum momento deve-se cessar o treinamento, mas identificar qual é momento correto de parar o treinamento não está formalizado na literatura (Haykin, 1999). No entanto, nota-se que alguns critérios de parada, mesmo que de ordem prática, são adotados e apresentam resultados satisfatórios. Alguns dos critérios são:

- *pelo valor da norma euclidiana do vetor gradiente:* o algoritmo converge quando a norma euclidiana do vetor gradiente atinge um limiar especificado;
- *pelo valor da taxa de variação do erro médio quadrático:* o algoritmo converge quando a taxa de variação do erro médio quadrático por época for suficientemente pequena;
- *pela capacidade de generalização da rede:* neste caso deve ser usado um conjunto de padrões, segregado do conjunto total de padrões, para validação.

2.6.5 Métodos de segunda ordem

Treinar uma rede neural trata-se de um problema de otimização, cuja função de custo a ser minimizada é função do erro, calculado pela diferença entre a saída desejada (padrões) e a resposta da saída da RNA. No entanto, Haykin (1999) no desenvolvimento do algoritmo *back-propagation* aponta que o treinamento de redes neurais multi-camadas torna-se um problema de otimização não-linear. E, vários métodos de otimização não-lineares encontrados na literatura podem ser aplicados ao problema de treinamento de redes neurais, vistos a diante.

Conforme indicado na Seção 2.6.1, o algoritmo *back-propagation* é uma implementação baseada no método do gradiente descendente, cujo vetor de parâmetros (pesos) é ajustado na direção oposta ao do vetor gradiente do erro. O método do gradiente é classificado como um método indireto de primeira ordem, já que utiliza apenas a informação do gradiente (primeira derivada) da função de custo para o ajuste dos pesos da rede. Os métodos de primeira ordem são conhecidos por serem ineficientes no tratamento de problemas de larga escala, pois apresentam taxas de convergência muito lentas, especialmente em regiões próximas a mínimos locais (Luenberger, 1984; Bazaraa et al., 1993).

Do ponto de vista da direção de busca, o método do gradiente pode ser interpretado como sendo ortogonal a uma aproximação linear da função de custo em determinado ponto (Edgar e Himmelblau, 1988). Nos métodos indiretos de segunda ordem, além do vetor gradiente da função objetivo, faz-se também o uso da matriz *Hessiana* (matriz de derivadas de segunda ordem) da mesma. Na literatura referente à otimização não-linear, uma classe de algoritmos de

segunda ordem é apontada como apropriada para os problemas de larga escala (Silva, 1998). No entanto, alguns métodos de segunda ordem também apresentam desvantagens, sendo a principal delas o alto custo computacional associado ao cálculo e armazenamento da matriz *Hessiana*.

O algoritmo proposto por *Levemberg Marquardt* é um dos algoritmos de segunda ordem mais rápidos para o treinamento de RNAs de tamanho moderado (Jones et al., 2005), sendo este uma variação do método de Newton (Edgar e Himmelblau, 1988), que aproxima localmente a superfície de erro por uma função quadrática. Nesse algoritmo, o cálculo da matriz *Hessiana* é simplificado, usando apenas a matriz *Jacobiana* (Matriz de derivadas de primeira ordem com relação aos pesos e termos de polarização da RNA).

2.7 Generalização em RNAs

A generalização é a capacidade de uma RNA, devidamente treinada, responder coerentemente a padrões desconhecidos. Uma boa capacidade de generalização é conseguida se alguns fatores são levados em consideração, como por exemplo, o número de padrões utilizados no processo de treinamento e a capacidade do modelo de se ajustar a estes dados.

A arquitetura da rede neural e a complexidade física do problema abordado também determinam a qualidade da generalização, frente a representatividade estatística do conjunto de padrões selecionado para o treinamento (Teixeira, 2001), ou seja, somente é possível extrair uma informação se ela estiver contida dentre os padrões de treinamento e se a RNA está estruturalmente dimensionada para extrair essa informação.

No entanto, dimensionar o conjunto de padrões e a arquitetura da rede, não é uma tarefa simples, visto que não existem regras que definem como se deve proceder para garantir boa generalização. A capacidade de generalização não é uma propriedade inerente às RNAs, ou seja, ela não é facilmente obtida simplesmente submetendo a rede à fase de treinamento , sendo necessário uma verificação da qualidade de generalização em um processo posterior ao treinamento chamado de validação (Teixeira, 2001).

O processo de validação tem por finalidade testar a generalização da RNA com um conjunto de padrões não apresentado à rede durante o treinamento. Sendo este, um instrumento de grande importância para determinar se é necessário um novo treinamento e/ou redimensionamento da rede.

De forma a exemplificar a generalização de uma rede, as Figuras 2.7, 2.8 e 2.9 apresentam os gráficos da função geradora, senóide que varia de 0 a 2π, a função geradora contaminada por um ruído de média 0 (zero) e desvio padrão 1 (um) chamada de padrões de treinamento e, a

resposta (saída) da RNA após o treinamento.

Ao final do treinamento uma RNA pode apresentar sua resposta em três situações distintas, a saber:

- a Figura 2.7 representa que a complexidade do problema a ser identificado é maior que a capacidade da RNA, ocorrendo um sub-ajuste aos dados de treinamento, sendo comumente chamado de *underfitting*.
- quando a complexidade da RNA supera a complexidade do problema, ocorre o superajuste ou *overfitting*, apresentando o comportamento conforme a Figura 2.8.
- a Figura 2.9 ilustra a resposta de uma rede cuja complexidade se equipara ao do problema.

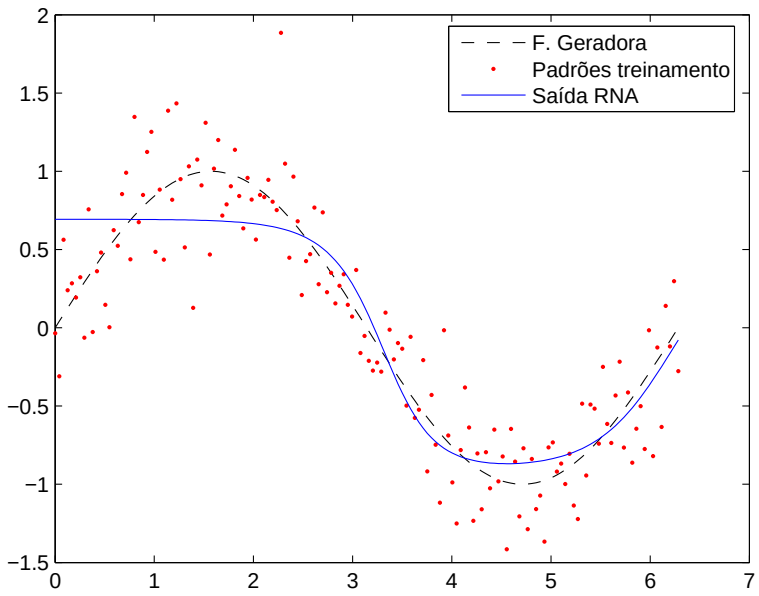

Figura 2.7: Problema do ajuste do modelo - *Underfitting*

Como pode ser visto, os fenômenos *underfitting* e *overfitting* afetam a capacidade de generalização das RNAs, sendo que o *underfitting* pode também ser causado por um número pequeno de épocas de treinamento, assim como o *overfitting* pode ser causado por um número elevado de épocas de treinamento.

Visando um aumento da capacidade de generalização das RNA algumas estratégias são apresentadas na literatura, destacando-se o algoritmo de *Early Stopping* (Weigend et al., 990b),

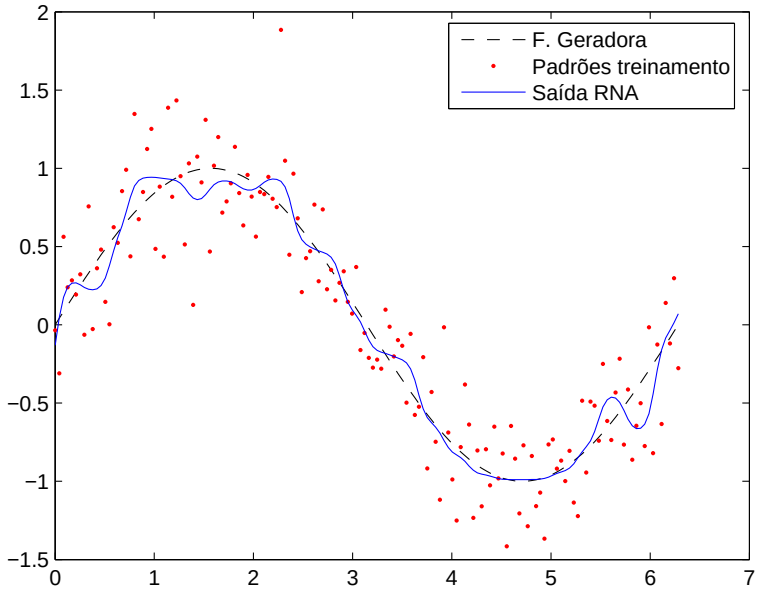

Figura 2.8: Problema do ajuste do modelo - *Overfitting*

o de Regularização da Complexidade (Chauvin, 1989; Hanson e Pratt, 1989; Weigend et al., 990a), o de *Pruning* (LeCun et al., 1990; Hassibi et al., 1993). Nas seções seguintes deste capítulo serão apresentados algumas destas.

2.7.1 Parada antecipada do treinamento

A parada antecipada do treinamento (*Early Stopping*) é uma técnica de treinamento na qual a cada época se faz uma teste de validação para checar a generalização da rede. Proposto por Weigend et al. (990b), o *Early Stopping* faz o uso de dois conjuntos de padrões, um para treinamento e outro para validação. Ambos possuem a mesma representatividade estatística, ou seja, os conjuntos são formados por valores cuja análise estatística seja semelhante (média, variância, desvio padrão, etc.).

A cada época do treinamento é avaliado se o comportamento do erro de validação é monotonicamente decrescente, e, caso não seja, o treinamento é encerrado. Sendo esta a origem do nome da técnica, garantindo assim que os pesos da rede sejam ajustados corretamente (Weigend et al., 990b). A Figura 2.10 mostra o comportamento dos erros de treinamento e de validação no treinamento com dados ruidosos.

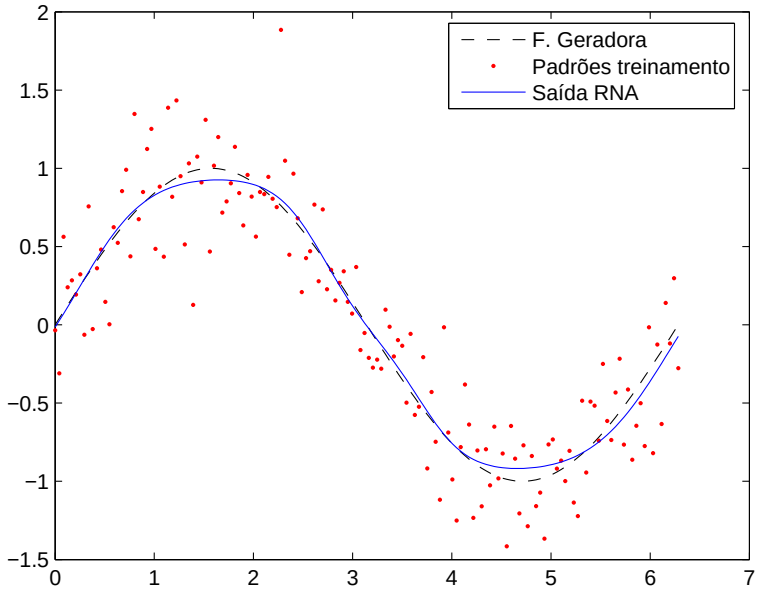

Figura 2.9: Problema do ajuste do modelo - Ajuste Adequado

Através desta técnica, durante o processo de treinamento, é possível identificar quando a RNA começa apresentar características de adaptação aos padrões de treinamento (memorização dos dados de treinamento) e grande erro de validação. Também elimina o tempo de treinamento onde a rede diminui o erro perante aos padrões de treinamento mas entra em uma fase de degradação da generalização com o passar das épocas de treinamento (Haykin, 1999). No entanto, em situações onde o conjunto de padrões é limitado (pequeno), a técnica pode culminar em um resultado não satisfatório.

2.7.2 Regularização da Complexidade

O algoritmo de regularização tem por finalidade regular a complexidade do modelo, adicionando à função de custo do erro a ser minimizado uma função de penalidade (Chauvin, 1989; Hanson e Pratt, 1989; Weigend et al., 990a). No objetivo inicial do treinamento das RNAs, a função de custo é dada por:

$$J = \varepsilon_s(\mathbf{W}) \tag{2.10}$$

De forma a aumentar a capacidade de generalização por este método, a função de custo J

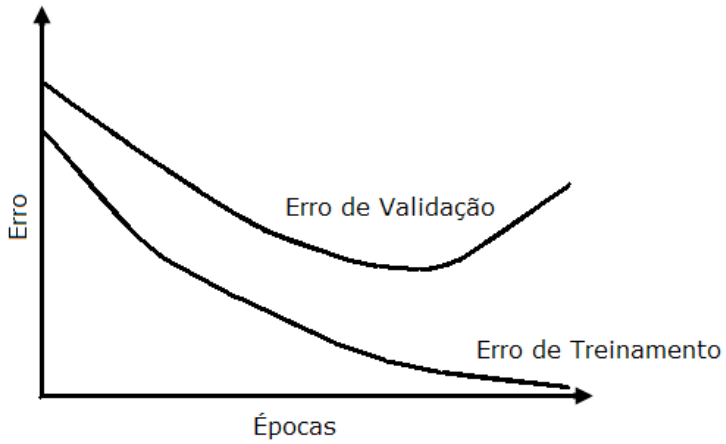

Figura 2.10: Comportamento dos erros de treinamento e validação no treinamento com dados ruidosos.

é modificada adicionando-se um termo de punição ou penalidade conforme a Equação (2.11).

$$J = \varepsilon_s(\mathbf{W}) + \lambda\varepsilon_c(\mathbf{W}) \tag{2.11}$$

que em situação genérica o termo de punição $\varepsilon_c(\mathbf{W})$é a integral da suavização da K-ésima ordem dado por:

$$\varepsilon_c(\mathbf{W}) = \frac{1}{2}\int \left\| \frac{\partial^K}{\partial \mathbf{x}^K} F(\mathbf{x},\mathbf{W}) \right\|^2 \mu(\mathbf{x}) d\mathbf{x} \tag{2.12}$$

sendo $F(\mathbf{x},\mathbf{W})$ é o mapeamento de entrada-saída realizado pelo modelo, e $\mu(\mathbf{x})$ é a função de ponderação que determina a região do espaço de entrada sobre a qual $F(\mathbf{x},\mathbf{W})$ deve ser suave (Haykin, 1999).

Desta forma, o objetivo é tornar pequena a K-ésima derivada de $F(\mathbf{x},\mathbf{W})$ em relação ao vetor de entrada $\mathbf{x}$. Assim, quanto maior for o valor escolhido para K, menos complexa se tornará a função $F(\mathbf{x},\mathbf{W})$ aumentando a capacidade de generalização da RNA.

Chauvin (1989); Hanson e Pratt (1989); Weigend et al. (990a) alertam que o termo de penalidade pode criar mínimos locais adicionais e um aumento no tempo de treinamento, sendo requerido um balanço o entre o termo de erro e o termo de penalidade.

2.7.3 Algoritmo de *pruning*

Os algoritmos de *pruning* têm por finalidade alterar a estrutura e/ou topologia da RNA durante o processo de treinamento através da eliminação de neurônios e/ou conexões.

Geralmente, um algoritmo de *pruning* usa um termo de penalidade associado à função de custo para identificar qual neurônio ou conexão está sendo pouco estimulado. O algoritmo de treinamento com decaimento dos pesos (*weight decay*) (Hinton, 1989), além de reduzir a complexidade do modelo por meio do decaimento dos pesos (parâmetros livres), auxilia também na identificação dos elementos que menos afetam a função de minimização do erro.

Os algoritmos *Optimal brain damage* (LeCun et al., 1990) e *Optimal brain surgeon* (Hassibi et al., 1993) são os algoritmos de *pruning* mais difundidos na literatura.

No algoritmo de *pruning Optimal brain surgeon* (OBS), inicialmente treina-se uma rede superdimensionada para o problema em questão. Após o treinamento da RNA é realizado o *cálculo de saliências*, o qual indica a influência de cada peso na função de custo. De posse desta informação, os pesos que apresentarem menores saliências são eliminados, modificando-se a estrutura da RNA, o que ocasiona a necessidade de um novo treinamento.

No cálculo da saliência é utilizado uma aproximação, pela série de Taylor, da função de custo, pela qual pode-se predizer o efeito de uma perturbação no vetor de pesos. Ou seja, sendo a função de custo $\varepsilon(w)$ pode-se predizer o efeito causado por uma perturbação Δw no vetor de parâmetros w (pesos), descrita na Equação 2.13:

$$\varepsilon(w+\Delta w) = \varepsilon(w) + g^T(w)\,\Delta w + \frac{1}{2}\Delta w^T H(w)\,\Delta w + O\left(||\Delta w||^3\right), \tag{2.13}$$

sendo $g(w)$ o vetor gradiente avaliado em w e $H(w)$ a matriz *Hessiana* também avaliada em w. Por meio desta equação, o objetivo do método é identificar um conjunto de pesos w que ao serem eliminados promova um incremento mínimo na função $\varepsilon(w)$.

Nota-se que o cálculo da matriz *Hessiana* causa um elevado custo computacional. Visando reduzir esse custo, foi elaborado o algoritmo *Optimal brain damage* (OBD), o qual faz uma aproximação da matriz *Hessiana*, considerando-a como uma matriz diagonal. Segundo Haykin (1999) pode-se considerar que o algoritmo OBD é um um caso particular do algoritmo OBS.

Mais detalhes sobre a implementação dos algoritmos OBD e OBS, podem ser vistos em LeCun et al. (1990), Hassibi et al. (1993) e Haykin (1999).

2.8 Identificação de sistema dinâmico com RNA

No processo de identificação de um sistema é necessário definir e dimensionar a representação matemática que melhor se aplica a este. Devido a esse motivo, a Seção 2.8.1 faz uma breve introdução à identificação de sistemas, destacando os tipos de modelos e processo de

identificação (modelagem).

A Seção 2.8.2 apresenta o uso de Redes Neurais Recorrente de Entrada-Saída em identificação de sistemas dinâmicos, servindo de base para a aplicação de técnicas de controle no Capítulo 3.

2.8.1 Identificação de sistemas

A identificação de um processo é um método que determina quais são as relações entre entrada e saída, obtidos a partir de sinais do processo e/ou do conhecimento da física envolvida no sistema (Pozas, 2005), pertencente a área de *Modelagem matemática*. Aguirre (2004) define Modelagem matemática como sendo a área do conhecimento que estuda as maneiras de desenvolver e implementar modelos (matemáticos) de sistemas reais.

Um modelo de processo resume-se então, em uma função que explica as relações de causa e efeito entre entradas e saídas de um sistema (Ansari e Tadé, 2001), podendo-se agrupar os modelos disponíveis basicamente em três grandes categorias:

- Modelo empírico, também conhecido como modelo caixa-preta;
- Modelo fenomenológico, também conhecido como modelo caixa-banca;
- Modelo caixa-cinza (semi-físico ou empírico com uso de informação auxiliar).

Dentre estas categorias, os modelos são classificados, basicamente, quanto às seguintes propriedades:

- Dinâmica do processo (Estáticos ou Dinâmicos);
- Tipo de sinal (Discretos ou Contínuos)
- Linearidade do processo (Linear ou Não-Linear);
- Variância no tempo (Variante ou Invariante no tempo);
- Quantidade de entradas e saídas (Monovariáveis ou Multivariáveis).

2.8.2 Redes Neurais Recorrentes de Entrada-Saída

As redes neurais são capazes de representar modelos não-lineares complexos (Haykin, 1999), e a arquitetura chamada de redes neurais recorrente de entrada-saída são amplamente usadas para modelagem de sistema dinâmicos (Schnitman e Fontes, 1999), devido à incorporação do comportamento temporal presente entre entradas e saídas da própria RNA.

A rede neural recorrente de entrada-saída usa os valores passados de sua própria saída e entrada realimentando os neurônios para responder dinamicamente aos estímulos aplicados na entrada.

De forma a exemplificar a aplicação deste conceito, considere um modelo não-linear de um sistema que seja expresso pela Equação 2.14.

$$y(k+1) = \mathcal{F}\big(y(k),y(k-1),...,y(k-n),u(k),u(k-1),...,u(k-m)\big) \tag{2.14}$$

sendo $\mathcal{F}$ uma função não-linear entre a saída $[y(k), y(k-1),..., y(k-n)]$ e a entrada $[u(k),u(k-1),..., u(k-m)]$, n e m representam o número de atraso em y e em u, respectivamente.

Nota-se que este é um sistema dinâmico, caracterizado pela presença do uso de valores passados tanto da entrada quanto da saída, ou seja, pela dependência temporal. Haykin (1999) expõe que na modelagem de qualquer sistema dinâmico por meio de RNAs usa-se estruturas chamadas de Modelos Neurodinâmicos, pois estes são compostos por elementos em que a dependência temporal está presente.

Pertencente a modelos neurodinâmicos, as redes neurais recorrentes de entrada-saída compartilham das seguintes características (Haykin, 1999):

- Incorporam um perceptron de múltiplas camadas estático ou parte dele;
- Exploram a capacidade de mapeamento não-linear do perceptron de múltiplas camadas;
- Camada de entrada do perceptron de múltiplas camadas composta por memória de linha de atraso derivada com tamanho parametrizável, tanto para a entrada do modelo quanto para a saída do mesmo.

Sendo $u(k)$ o valor presente da entrada do modelo e o valor correspondente da saída do modelo representado por $y(k+1)$ mostra que a saída está adiantada em relação a entrada por uma unidade de tempo. Assim, o sistema expresso pela Equação 2.14 pode ser modelado por uma RNA recorrente de entrada-saída como exposto na Figura 2.11.

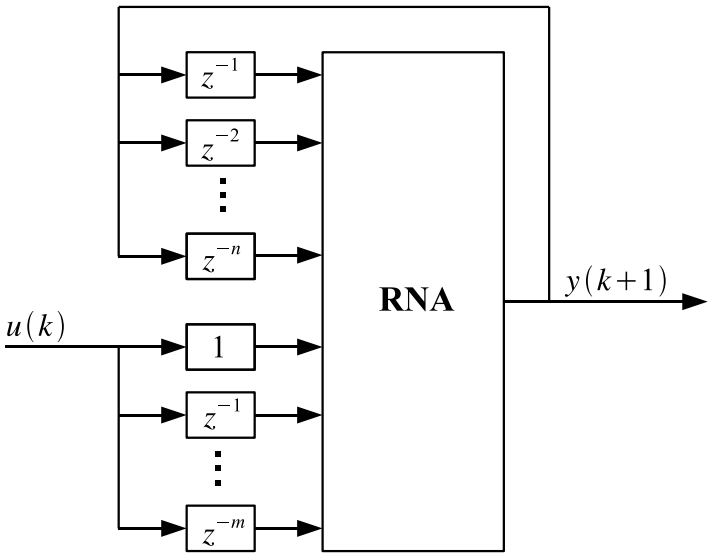

Figura 2.11: Representação de uma rede neural recorrente de entrada-saída.

Na RNA exposta na Figura 2.11, o sinal aplicado a sua camada de entrada está associado à:

- valores presente e passado da entrada do modelo (entradas exógenas ordinárias de fora da rede), e

- valores passados de sua própria saída, sobre os quais é feita a regressão da saída do modelo $y(k+1)$,

tendo o operador de atraso representando por z^{-1}, sendo n e m a dimensão da memória de linha de atraso derivada para y e u, respectivamente.

Haykin (1999) relata que a dimensão da memória de linha de atraso derivada, tanto para y como para u, está relacionada à ordem do sistema a ser modelado. Em processo de identificação de sistema, devido ao não conhecimento à priori da complexidade do sistema em questão, Aguirre (2004) salienta sobre a dificuldade de determinação dos parâmetros da RNA (estrutura, número de neurônios e função de ativação). No entanto, como em qualquer outra representação matemática, uma vez dimensionada de forma inadequada, a mesma poderá apresentar diversos problemas dinâmicos.

2.9 Comentários Finais

Este capítulo teve como principal objetivo destacar a teoria sobre redes neurais artificiais (RNA), desde a sua fundamentacão teórica até sua aplicação em modelagem de sistemas dinâmicos.

Na Seção 2.2 foi exposta uma introdução sobre as RNAS, destacando suas características, abstração de conhecimento, vantagens comparadas como outras representações matemáticas, entre outros. O modelo matemático de um neurônio, seus elementos básicos e suas diversas funções de ativação estão contidos nas Seções 2.3 e 2.4.

Com o objetivo de ampliar a capacidade de aprendizado, vários neurônios podem ser conectados entre si, formando assim uma rede neural. A forma como os neurônios estão dispostos e conectados uns aos outros pode ser vista na Seção 2.5.

O processo como se dá o aprendizado e o algoritmo de treinamento foi exposto na Seção 2.6. Visando que a RNA responda satisfatoriamente a padrões não conhecidos, a Seção 2.7 expõe os diversos mecanismos usados na ampliação da generalização das RNAs.

Por fim, a Seção 2.8 apresenta uma breve conceituação sobre sistemas dinâmicos e como identificá-los por meio da utilização de redes neurais recorrentes de entrada-saída.

Capítulo 3

Neuro-Controladores

3.1 Introdução

Este capítulo apresenta os métodos mais comuns de utilização de RNAs para realizar controle de sistemas, enfatizando sua disposição dentro da estrutura de controle. Segundo Schnitman e Fontes (1999) um Neuro-Controlador é caracterizado pelo uso de uma RNA localizado no sistema de controle, sendo o próprio controlador ou um modelo do processo usado para inferir uma ação de controle. Dentro dessa abordagem, é feito na Seção 3.2 uma exposição sobre os conceitos e fundamentos usados em controle de sistemas dinâmicos, objetivando um embasamento para utilização de RNA no controle desses. Nas Seções 3.3 e 3.4 são explorados os controles baseados em RNAs, abordando com mais profundidade teórica o Neuro-Controlador Preditivo, sendo este a base para o desenvolvimento do controlador proposto no Capítulo 4.

3.2 Controle de sistema dinâmico

Segundo Rivals e Personnaz (1996), um sistema de controle consiste em um processo a ser controlado e um dispositivo de controle, o qual impõe um comportamento desejado para o processo. Caso o processo apresente comportamento não-linear, o grau de dificuldade em controlá-lo aumenta, necessitando de controladores que contenham em sua arquitetura mecanismo para lidar com a não-linearidade.

3.2.1 Escolha do controlador

A escolha do tipo de controlador a ser aplicado em um processo dinâmico é proveniente da definição dos seguintes critérios, considerados os mais importantes:

- Estabilidade;
- Tempo de acomodação menor possível;
- Mínimo sobre-sinal (*Overshoot*);
- Pequeno erro em regime estacionário;
- Baixa amplitude/energia de controle.

Estabelecido os critérios acima, a estrutura de controle a ser escolhida pertencerá basicamente a um dos dois grupos de sistemas de controle:

- *Sistema de controle por simples realimentação*, sendo o dispositivo de controle caracterizado somente pelo uso do controlador; e
- *Sistema de controle por Modelo Interno - IM*, caracterizado pelo dispositivo de controle ser constituído por um controlador e um modelo do processo.

As representações desses grupos podem ser vistas nas Figuras 3.1 e 3.2, respectivamente. Nessas figuras pode-se identificar o sinal de referência r, a presença de distúrbios na saída, caracterizado por uma eventualidade inerente ao processo, e a saída do processo y_p.

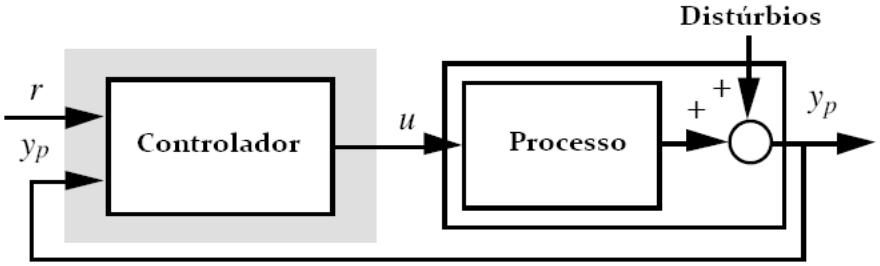

Figura 3.1: Representação de um sistema de controle por simples realimentação. Fonte: Rivals e Personnaz (2000).

No sistema de controle por simples realimentação apresentado na Figura 3.1, a saída do processo y_p realimenta diretamente o controlador. Já em um sistema de controle por modelo

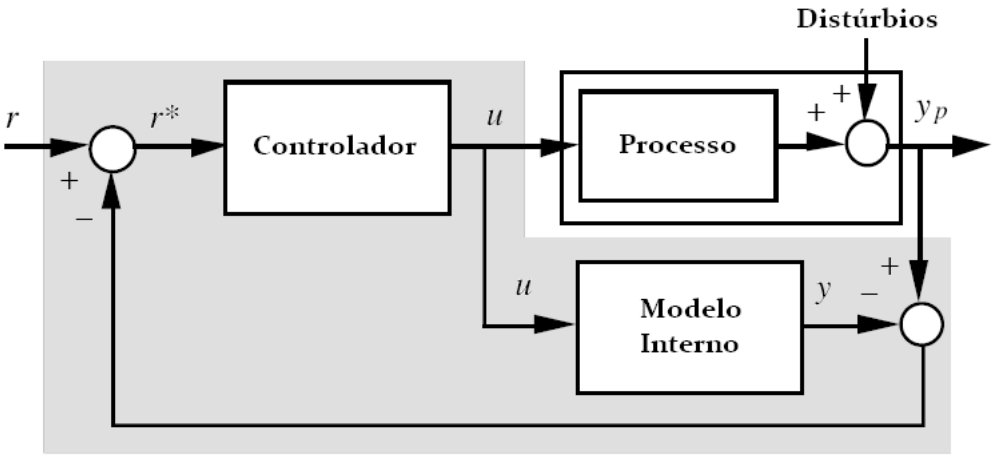

Figura 3.2: Representação de um sistema de controle por modelo interno. Fonte: Rivals e Personnaz (2000).

interno, Figura 3.2, é utilizado um modelo explícito do processo a ser controlado, sendo que a realimentação para o controlador é função desse modelo.

3.2.2 RNA na Arquitetura de Controle

Caso o sistema de controle faça uso de redes neurais artificiais, seja ele por simples realimentação ou modelo interno, a RNA pode assumir funções específicas dependendo de sua disposição na arquitetura do controlador. Agarwal (1997) classifica os controladores baseado em RNA, segundo sua arquitetura, em:

- **Arquitetura Direta**: Cuja RNA é o próprio controlador.
- **Arquitetura Indireta**: O controlador não é RNA, mas a utiliza como modelo do processo para determinar a próxima ação de controle.

Nota-se que em um sistema de controle baseado em RNA, usando a arquitetura direta, o controlador é do tipo simples realimentação, mas se é usado a arquitetura indireta, então tem-se um sistema de controle por modelo interno. Assim, se em um sistema de controle é feito o uso de RNA em sua arquitetura, este é comumente chamado de **Neuro-Controlador** ou **Controlador Neural**.

A disposição da RNA em um neuro-controlador determina seu comportamento, e a cada arranjo, um novo nome, sendo os mais conhecidos o Neuro-Controlador por Modelo Inverso e

Neuro-Controlador Preditivo, possibilitando ainda o uso de mecanismo de adaptação *on-line* e modelo de referência.

Tendo em vista essa gama de aplicação de RNA em controle de sistemas, o presente trabalho se atem somente ao controle monovariável, dando maior enfoque aos controladores que usam a RNA como Modelo Interno. No entanto, se faz necessário a caracterização do controlador por simples realimentação usando RNA de forma a possibilitar um paralelo entre os dois grupos de neuro-controladores.

3.3 Neuro-Controlador por Modelo Inverso

O Neuro-Controlador por Modelo Inverso pertence à classe de Neuro-Controlador por simples realimentação, constituído, basicamente, por um modelo inverso (RNA) em série com o processo, sendo um sistema de controle arquitetado de forma direta (Miller et al., 1990).

Tal característica motivou a pesquisa e aplicação deste tipo de neuro-controlador em sistema de fornos de aquecimento (Dias e Mota, 2000), em controle de movimentação de robôs (Patiño et al., 2002) e em processo de decapagem em usina siderúrgica (Daosud et al., 2005).

3.3.1 Arquitetura do Neuro-Controlador por Modelo Inverso

Conforme mencionado, o neuro-controlador por modelo inverso constitui-se de uma RNA (treinada para ser o modelo inverso do processo) em série com o próprio processo a ser controlado. A Figura 3.3 representa essa arquitetura de controle.

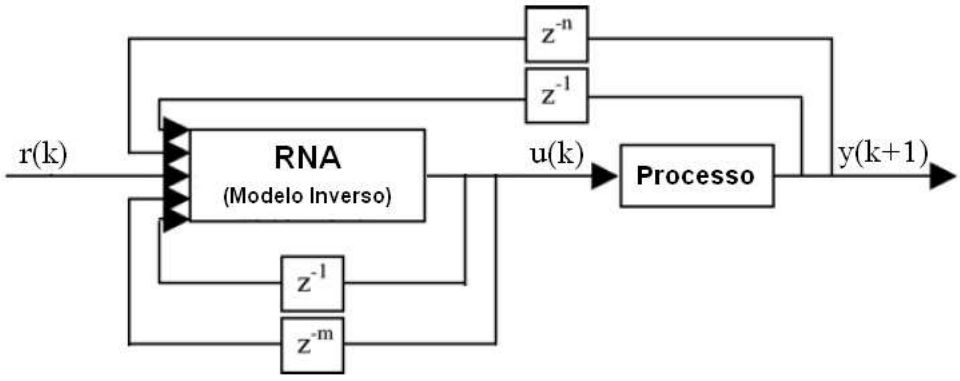

Figura 3.3: Representação um neuro-controlador por modelo inverso.

Na Figura 3.3, a entrada do neuro-controlador (RNA - modelo inverso) é composto pelo si-

nal de referência $r(k)$ associado aos valores passados das próprias ações de controle u e estados do processo y, sendo o operador de atraso representando por z^{-1}.

Miller et al. (1990), Rivals e Personnaz (1996) comentam que se a RNA deste controlador é obtida pelo método de treinamento de modelo inverso especializado (ver Sub-seção 3.3.2), o neuro-controlador apresenta as seguintes características:

- Se o processo e o controlador são estáveis, se o modelo direto é uma boa aproximação do processo, se o controlador é o inverso do modelo direto, e se não há distúrbios, então o controle perfeito é alcançado;
- Se o ganho do controlador em regime estacionário é igual ao ganho do inverso do modelo direto em regime estacionário, e se o sistema de controle é estável com este controlador, então o erro nulo de posição do controlador é obtido para o sinal de referência constante com ou sem distúrbios na saída.

3.3.2 Obtenção do modelo inverso

O modelo inverso é obtido, em linhas gerais, pelo treinamento da RNA tomando a saída do do processo como entrada da RNA e entrada do processo como saída da RNA, ou seja, a rede neural apresentará um comportamento inverso ao do processo após o treinamento. A Figura 3.4 representa esta forma de treinamento.

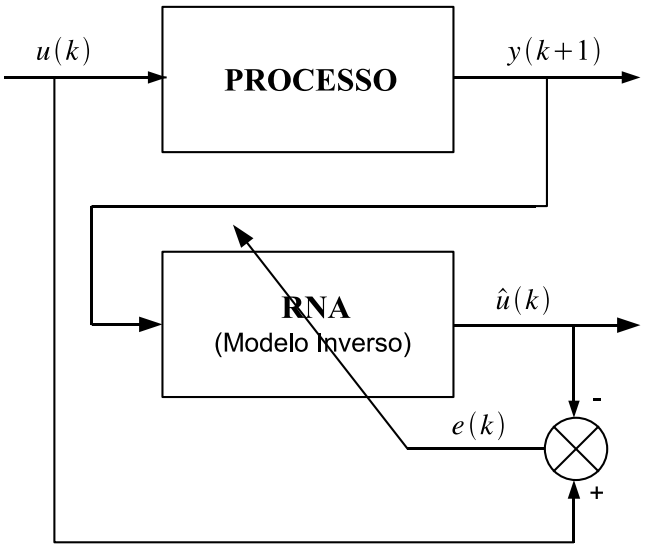

Figura 3.4: Representação de treinamento inverso de uma RNA.

No entanto, apesar de simples, esta abordagem tem algumas desvantagens (Psaltis et al., 1988):

- O objetivo do procedimento de aprendizagem não é direto. Como o modelo não é treinado na mesma situação em que será utilizado após o treinamento, a estrutura de treinamento é dita não ser direcionada ao objetivo;

- A variável de saída do processo, especificada para o treinamento da RNA, será substituída pelo sinal de referência quando a RNA for colocada em situação de controle, o que não garante que sua saída será ação de controle adequada.

De forma a evitar esses problemas, Psaltis et al. (1988) apresenta outro método de treinamento para obter o modelo inverso, conhecido como treinamento de modelo inverso especializado, sendo esse apresentado na Figura 3.5.

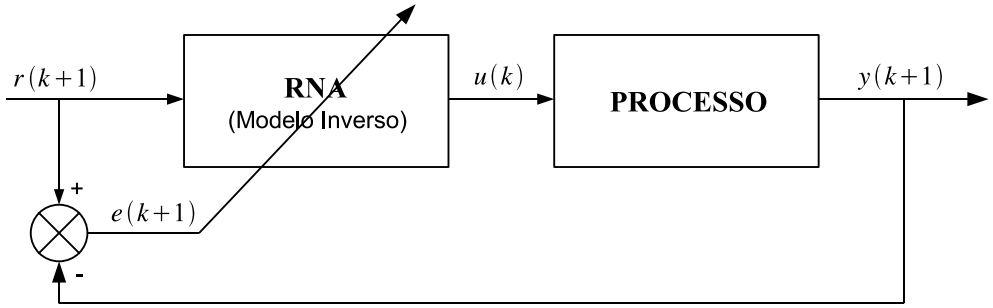

Figura 3.5: Representação de treinamento de modelo inverso especializado de uma RNA.

Nota-se que o modelo inverso está conectado em série com o processo. Assim, o sinal de erro usado no treinamento da RNA é formado pela relação entre o sinal de referência e a saída do processo. Em situações práticas, um modelo direto do processo (RNA ou outra representação matemática) deve ser obtido a priori e usá-lo no treinamento da RNA que será o modelo inverso do processo. Isto se faz necessário para conhecer a dinâmica do processo e facilitar os testes do sistema de controle.

Para um maior aprofundamento sobre o neuro-controlador por modelo inverso e a forma de treinamento da RNA, recomenda-se a leitura dos trabalhos de Psaltis et al. (1988), Miller et al. (1990), Hunt et al. (1992) e Rivals e Personnaz (1996).

3.4 Neuro-controladores Preditivos

Os neuro-controladores preditivos pertencem à classe dos Controladores Preditivos baseados em Modelos (*Model Predictive Control* - MPC), cujo modelo interno é uma RNA. Desta forma, todo conceito de MPC pode ser aplicado aos neuro-controladores.

Visando um melhor entendimento sobre MPC, a primeira parte desta seção mostra os seus conceitos básicos, facilitando a exploração dos neuro-controladores preditivos em seguida.

3.4.1 Controladores Preditivos Baseados em Modelos

Controladores Preditivos baseados em Modelos referem-se a uma classe de algoritmos computacionais para controle que utilizam um modelo explícito do processo para predizer a resposta futura da planta. Sendo assim, a cada iteração o MPC otimiza o comportamento futuro da planta através da seqüencia de ações de controle (Qin e Badgwell, 2003).

O MPC pertence ao grupo de controladores baseados em Modelo Interno (*Internal Model - IM*) o qual é oriundo da conseqüência da seguinte suposição (Rivals e Personnaz, 1996): Se o processo e o controlador são estáveis, e se o IM é perfeito, então o sistema é estável.

Sendo assim, o controle adequado é alcançado por um mecanismo de otimização que avalia a diferença entre a saída do IM e a saída real da planta, ou usa o IM para avaliar seu estado futuro imposta por uma ação de controle no presente.

Os procedimentos para o desenvolvimento e aplicação no processo de um MPC geralmente seguem os seguintes passos:

1. Definir os objetivos iniciais de controle;
2. Definir o tamanho do problema e determinar as variáveis (entradas e saídas do processo) relevantes;
3. Excita-se sistematicamente o processo coletando as variáveis definidas previamente;
4. Extrair um modelo dinâmico da base de dados coletada, que represente bem o processo que se queira controlar, sendo este realizado por meio da identificação, validação e testes;
5. Parametrizar o MPC e testar o controlador usando uma simulação em malha fechada para verificar seu desempenho;

6. Carregar o MPC no dispositivo de controle definitivo, testar a predição do modelo e refinar a sintonia do controlador, quando necessário.

Segundo Qin e Badgwell (2003), é notável o crescimento dos controladores do tipo MPC, visto que seu uso tem sido aplicado em plataformas comercializadas por grandes empresas do seguimento (Ex.: *Honeywell Hi-Spec*, *Aspen Tech*), os quais possuem ferramentas tanto de simulação quanto de sintonia em malha fechada.

Quando se fala em controlador preditivo é necessário estipular quanto tempo a frente se quer predizer, sendo este conhecido como horizonte de predição. Em sistemas discreto, o horizonte de predição é determinado pelo número de passos, o qual é múltiplo do período de amostragem (ou tempo de discretização) do sistema.

3.4.2 Arquitetura do Neuro-Controlador Preditivo

O neuro-controlador preditivo usa um modelo do processo dado por uma RNA e um algoritmo de otimização. O algoritmo de otimização usa a resposta da RNA, que representa o processo, e infere uma ação de controle no presente visando que a saída $\hat{y}(k+1)$ seja igual a referência $r(k+1)$. Na Figura 3.6 é apresentada a arquitetura clássica de um neuro-controlador preditivo.

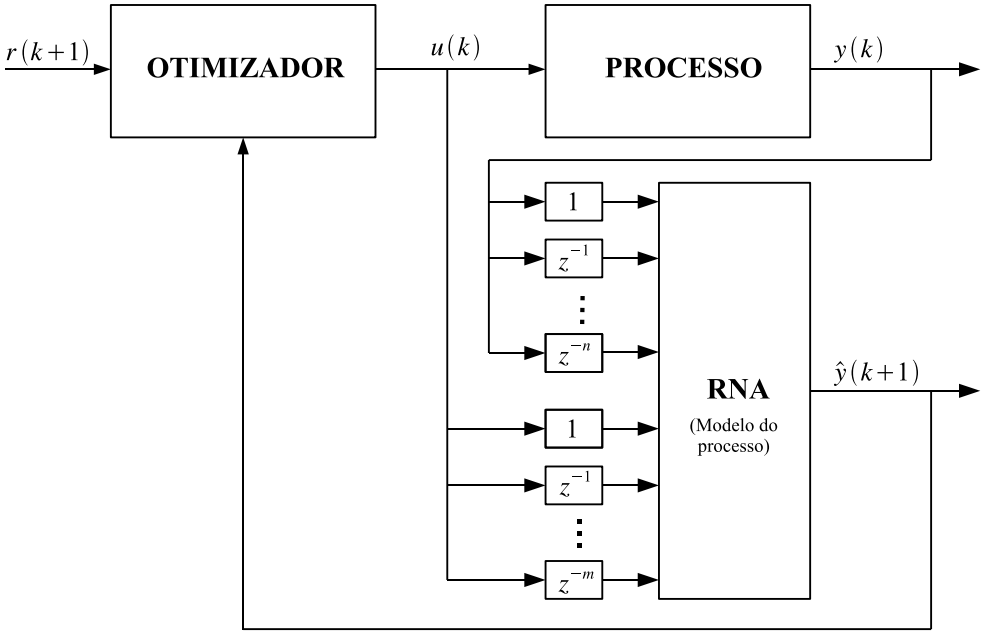

Figura 3.6: Arquitetura clássica de um neuro-controlador preditivo.

Na Figura 3.6 é possível notar a presença do operador de atraso z^{-1}, para denotar a memória de linha de atraso derivada de ordem n e m da RNA, tanto para a entrada quanto para a saída do processo. Nota-se que esta RNA é semelhante a uma rede neural recorrente de entrada-saída, porém, usando os valores da saída do próprio processo. Mais detalhes sobre este tipo de RNA podem ser vistos na Seção 2.8.2.

3.4.3 RNA do Neuro-Controlador Preditivo

A RNA para um neuro-controlador preditivo é dimensionada de acordo com a complexidade do processo que se queira controlar, podendo variar em quantidade de camadas da rede, em número de neurônio por camada e em tipo de função de ativação (Miller et al., 1990).

Cybenko (1989) e Funahashi (1989) afirmam que uma RNA com apenas uma camada escondida é capaz de representar qualquer função contínua, limitado somente ao número de neurônios da camada escondida. Tan e Cauwenberghe (1996) usam esta afirmação na proposta de neuro-controlador preditivo de um passo à frente para controlar sistemas com atraso de transporte.

Tendo em vista estas características, a arquitetura da RNA adotada neste trabalho é composta por somente uma camada escondida, com função de ativação do tipo sigmoidal tangente hiperbólica para neurônios da camada escondida e função de ativação do tipo linear para o neurônio da camada de saída. Assim, esta RNA pode ser representada conforme a Figura 3.7.

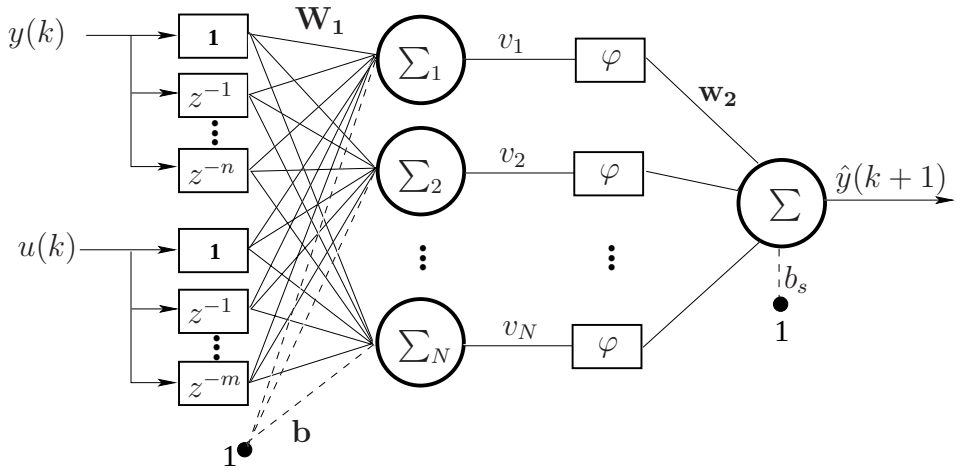

Figura 3.7: Arquitetura da RNA para neuro-controlador preditivo.

Analisando a RNA apresentada na Figura 3.7, pode-se identificar os seguintes elementos:

$u(k)$	ação de controle aplicada no processo no instante k;
$y(k)$	resposta do processo no instante k;
$\hat{y}(k+1)$	resposta estimada pela RNA a uma ação de controle aplicada no instante k;
z^{-1}	operador de atraso;
n	número de atrasos em y;
m	número de atrasos em u;
$v_1, v_2,...,v_N$	campo local induzido do neurônio $1, 2,..., N$ da camada escondida;
N	número de neurônios da camada escondida;
φ	função de ativação dos neurônios da camada escondida;
$\mathbf{W_1}$	matriz de pesos dos neurônios da camada escondida com dimensão $(N,n+m)$;
$\mathbf{w_2}$	vetor de pesos dos neurônios da camada de saída com dimensão $(1,N)$;
$\mathbf{b}$	vetor dos termos de polarização dos neurônios da camada escondida com dimensão $(N,1)$;
b_s	termo de polarização do neurônio da camada de saída;

Com base nestas informações, a representação matemática desta RNA é dada pela equação (3.1) e (3.2).

$$\hat{y}(k+1) = b_s + \sum_{i=1}^{N} \mathbf{w_2}(1,i).\varphi(v_i) \tag{3.1}$$

sendo

$$v_i = b(i,1) + \sum_{j=1}^{n} \mathbf{W_1}(i,j).y(k-j+1) + \sum_{j=1}^{m} \mathbf{W_1}(i,n+j).u(k-j+1) \tag{3.2}$$

3.4.4 Algoritmo de otimização do Neuro-Controlador Preditivo

Antes de aprofundar no algoritmo de otimização do neuro-controlador preditivo, alguns conceitos sobre otimização de processo devem estar claros. Desta forma, é feito uma breve abordagem sobre otimização de processos e o método do gradiente descendente. Em seguida, esse método de otimização é aplicado em um neuro-controlador preditivo.

- **Formulação do Problema de Otimização**

Para a formulação do problema é necessário um modelo matemático que correlacione as variáveis do processo. Deve-se então, determinar uma expressão matemática entre as variáveis do processo que quantifique a adequação das soluções possíveis.

A formulação do problema de otimização envolve:

- Pelo menos uma função objetivo (quantificação matemática da adequação de uma solução);
- Restrições de igualdade;
- Restrições de desigualdade.

A função objetivo, também conhecida como função de custo, é a função que se deseja minimizar ou maximizar. Neste trabalho a notação adotada para uma função objetivo é representada por $J(\mathbf{u})$, sendo $\mathbf{u}$ o vetor das variáveis que compõe o problema a ser minimizado.

Edgar e Himmelblau (1988) sugerem que os passos para otimizar processos devem seguir basicamente a seguinte ordem:

1. Analisar o processo, fazendo uma lista de todas as variáveis e características de interesse;
2. Determinar o critério de otimização e construir a função objetivo como função das variáveis previamente levantadas. Neste passo, os coeficientes que compõem a função devem ser arbitrados;
3. Desenvolver o modelo do processo e incluir os coeficientes da função objetivo;
4. Determinar todas as relações de igualdade e desigualdade que em geral são provenientes de princípios físicos (balanços de massa, energia, quantidade de movimento), relações empíricas, conceitos implícitos e restrições externas;
5. Caso a dimensão e/ou complexidade do problema seja muito grande:
   - Divida-o em partes que possam ser resolvidas;
   - Simplifique o modelo ou a função objetivo;
6. Aplicar a técnica de otimização;
7. Checar a resposta e a sensibilidade do resultado com os coeficientes da função objetivo ou com as suposições adotadas.

Uma vez que se tenha estabelecido a função de custo e a dimensão do problema, este se enquadrará em um problema de Otimização Unidimensional (quando a função de custo contém

apenas uma única variável) ou Otimização Multidimensional (quando a função de custo contém mais de uma variável) cuja solução é dada pelo emprego do método de otimização, sendo este dividido em:

- **Métodos diretos**: utilizam apenas o cômputo da função objetivo (O valor da função é comparado em diversos pontos de avaliação na busca pelo extremo);
- **Métodos indiretos**: utiliza-se a condição necessária de um extremo (A expressão analítica da derivada é utilizada para o cômputo da mesma, sendo também usado o cômputo da função objetivo).

Existem vários métodos de otimização, sendo eles diretos ou indiretos para solução de problemas unidimensionais ou multidimensionais. Desta forma, os métodos de otimização mais comuns podem ser classificados da seguinte forma (Reklaitis et al., 1983; Edgar e Himmelblau, 1988):

- Otimização Unidimensional Sem Restrições
  - Métodos Indiretos
    1. Método de Newton
    2. Método de Quasi-Newton
    3. Método da Secante
  - Métodos Diretos
    1. Método de Eliminação de Região
    2. Métodos por Aproximação Polinomial - Interpolação Quadrática
    3. Métodos por Aproximação Polinomial - Interpolação Cúbica
- Otimização Multidimensional Sem Restrições
  - Métodos Indiretos
    1. Método do Gradiente ou Método do Gradiente Descendente
    2. Método do Gradiente Conjugado
    3. Método de Newton
    4. Método de Levenberg-Marquardt
    5. Método da Secante ou Quasi-Newton

- Métodos Diretos
  1. Busca Aleatória
  2. Grade de Busca
  3. Busca Unidimensional
  4. Método Simplex ou do Poliedro Flexível

Uma vez que o objetivo deste capítulo é explorar o neuro-controlador preditivo, somente o método do gradiente descendente será detalhado nesta seção, pois, este é comumente empregado em controlador preditivo. No entanto, para entendimento sobre a forma e implementação de cada um desses métodos, recomenda-se a leitura de Reklaitis et al. (1983), Luenberger (1984), Edgar e Himmelblau (1988) e Bazaraa et al. (1993).

- **Método do Gradiente Descendente**

O método de gradiente descendente é um algoritmo de otimização usado para encontrar um mínimo local de uma função usando o gradiente da função objetivo.

Seja a função $J(\mathbf{u})$ cujo $\mathbf{u}$ representa o vetor de variáveis u_1,u_2,...,u_n, o gradiente da função $J(\mathbf{u})$ com o respectivo vetor de variáveis $\mathbf{u}$ é denotado por $\nabla J(\mathbf{u})$, sendo ∇ o operador diferenciação. Por definição, então, o gradiente é a derivada parcial da função $J(\mathbf{u})$ em relação ao seu vetor de variáveis $\mathbf{u}$, escrito conforme a equação (3.3).

$$\nabla J(\mathbf{u}) = \left(\frac{\partial J}{\partial u_1}, \frac{\partial J}{\partial u_2}, \ldots, \frac{\partial J}{\partial u_n} \right) \tag{3.3}$$

Nota-se também, que por definição, $\nabla J(\mathbf{u})$ é ortogonal ao contorno do ponto dado por $J(\mathbf{u})$ (Edgar e Himmelblau, 1988). Desta forma, o método de otimização por gradiente descendente usa esta ortogonalidade como direção de busca, com passo definido por:

$$\begin{aligned} \mathbf{u}(k+1) &= \mathbf{u}(k) + \Delta\mathbf{u}(k) \\ &= \mathbf{u}(k) + \lambda(k)\mathbf{s}(k) \\ &= \mathbf{u}(k) - \lambda(k)\nabla f(\mathbf{u}(k)) \end{aligned} \tag{3.4}$$

sendo:
$\Delta\mathbf{u}(k)$ o vetor de $\mathbf{u}(k)$ para $\mathbf{u}(k+1)$
$\mathbf{s}(k)$ a direção de busca (a direção do gradiente descendente)

$\lambda(k)$ um escalar que determina o tamanho do passo na direção $\mathbf{s}(k)$

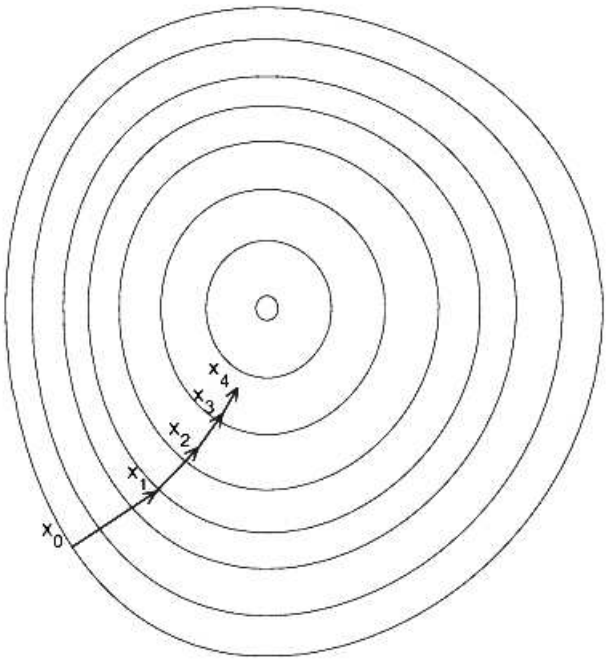

Figura 3.8: Representação gráfica do gradiente da função $J(\mathbf{u})$.

Nota-se que ocorre a inversão do sinal na terceira linha da Equação (3.4), isto é devido ao problema de otimização ter objetivo de minimização da função de custo. Portanto, a representação gráfica gradiente da função $J(\mathbf{u})$ pode ser visto na Figura 3.8, sendo que o sentido do vetor de minimização é ortogonal as isolinhas e para o centro.

Se $J(\mathbf{u})$ for a função objetivo do problema de otimização, pode-se então dizer que o método de gradiente descendente usa somente o cálculo da primeira derivada da função objetivo e a cada iteração k essa função é minimizada com tamanho do passo definido por $\lambda(k)$. Se este passo for constante tem-se:

$$\mathbf{u}(k+1) = \mathbf{u}(k) - \lambda \nabla J(\mathbf{u}(k)) \tag{3.5}$$

3.4.5 Gradiente Descendente para Neuro-Controlador Preditivo

Em um sistema de controle, o principal objetivo é que o erro entre o sinal de referência e a resposta do processo seja mínimo no instante seguinte $(k+1)$, mediante a uma alteração do sinal de referência ou percepção de um distúrbio, equação (3.6). De forma a atingir o objetivo,

uma ação de controle u é escolhida e aplicada ao processo no instante k.

$$e(k+1) = r(k+1) - \hat{y}(k+1) \tag{3.6}$$

sendo:
$e(k+1)$ o erro esperado;
$r(k+1)$ o sinal de referência; e
$\hat{y}(k+1)$ a resposta estimada do processo (Equação (3.1)).

Seja J a função de custo do problema de otimização para o controlador que equacione a minimização do erro esperado, como por exemplo o *erro quadrático* dado por:

$$J = \frac{1}{2}e^2(k+1) \tag{3.7}$$

Como J é função de $e(k+1)$ que função de $\hat{y}(k+1)$ que função de $u(k)$, logo J é função de $u(k)$. Portanto, aplicando essa função à Equação (3.5), na forma escalar, tem-se o método do gradiente descendente aplicado na busca da ação de controle, dado por:

$$u(k+1) = u(k) - \lambda \nabla J(u(k)) \tag{3.8}$$

ou na forma diferenciada e expandida pelas substituições das Equações (3.6) e (3.7)

$$\begin{aligned}
u(k+1) &= u(k) - \lambda \frac{\partial J}{\partial u(k)} \\
&= u(k) - \lambda \frac{\partial}{\partial u(k)} \left[\frac{1}{2} e^2(k+1) \right] \\
&= u(k) - \lambda \frac{\partial}{\partial u(k)} \left[\frac{1}{2} [r(k+1) - \hat{y}(k+1)]^2 \right] \\
&= u(k) - \lambda \frac{1}{2} \left[-2e(k+1) \frac{\partial \hat{y}(k+1)}{\partial u(k)} \right] \\
&= u(k) + \lambda e(k+1) \frac{\partial \hat{y}(k+1)}{\partial u(k)}
\end{aligned} \tag{3.9}$$

A representação matemática da RNA do modelo interno presente no neuro-controlador

preditivo é dada pela Equação (3.1). Diferenciando essa equação em relação a $u(k)$, tem-se:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \frac{\partial}{\partial u(k)} \left[b_s + \sum_{i=1}^{N} \mathbf{w}_2(1,i).\varphi(v_i) \right] \tag{3.10}$$

podendo ser reescrita como

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} \mathbf{w}_2(1,i).\varphi'(v_i).\frac{\partial v_i}{\partial u(k)} \tag{3.11}$$

sendo

$$\varphi' = \frac{d\varphi}{dv_i} \tag{3.12}$$

e

$$\begin{aligned} \frac{\partial v_i}{\partial u(k)} &= \frac{\partial}{\partial u(k)} \left[b(i,1) \right] + \frac{\partial}{\partial u(k)} \left[\sum_{j=1}^{n} \mathbf{W_1}(i,j).y(k-j+1) \right] \\ &+ \frac{\partial}{\partial u(k)} \left[\sum_{j=1}^{m} \mathbf{W_1}(i,n+j).u(k-j+1) \right] \end{aligned} \tag{3.13}$$

Nota-se que os termos $y(k-1)$,$y(k-2)$,...,$y(k-n)$, bem como os termos $u(k-1)$,$u(k-2)$,...,$u(k-m)$ são valores passados e independem de $u(k)$. Dessa forma o somatório é sempre nulo, exceto para $j=1$ cujo valor será 1. Portanto a equação (3.13) pode ser simplificada para:

$$\frac{\partial v_i}{\partial u(k)} = \mathbf{W_1}(i,n+1) \tag{3.14}$$

que substituída na Equação (3.11) resulta em:

$$\frac{\partial \hat{y}(k+1)}{\partial u(k)} = \sum_{i=1}^{N} \mathbf{w}_2(1,i).\varphi'(v_i).\mathbf{W_1}(i,n+1) \tag{3.15}$$

Pode-se dizer então, que a Equação (3.15) é a expressão diferencial generalizada para a RNA do neuro-controlador preditivo, que substituída na equação (3.9) produz a equação final de atualização da ação de controle, dado por:

$$u(k+1) = u(k) + \lambda e(k+1) \left[\sum_{i=1}^{N} \mathbf{w}_2(1,i).\varphi'(v_i).\mathbf{W_1}(i,n+1) \right] \tag{3.16}$$

O algoritmo do neuro-controlador preditivo usando o método de otimização por gradiente descendente para minimização da função de custo conhecida como erro quadrático é dada pela implementação dos seguintes passos (Schnitman e Fontes, 1999):

1. selecionar o tamanho do passo λ;
2. calcular $\hat{y}(k+1)$ por meio das Equações (3.1) e (3.2);
3. computar o erro $e(k)$ por meio da Equação (3.6);
4. calcular a nova ação de controle $u(k+1)$ por meio da Equação (3.16);
5. aplicar $u(k+1)$ na entrada do processo;
6. retornar ao passo 2.

Schnitman e Fontes (1999) comentam que λ é selecionado empiricamente, mas propõem um método para adaptação dinâmica de λ baseado no pólo dominante do sistema. Este, por sua vez, apresenta um melhor desempenho comparados a λ estático, porém os testes foram realizados somente em sistemas linearizados.

Quando λ assume valor muito baixo ou alto, o controlador pode levar o sistema à instabilidade ou não responder dinamicamente à sua excitação (Assis, 2007).

3.5 Comentários Finais

Este capítulo apresentou conceitos relacionados ao controle dinâmico e formas de utilização de redes neurais artificiais em controle de sistemas. Foram abordados os controladores por modelo inverso e preditivo usando RNA, os quais são as formas mais comuns de neuro-controladores.

Na Seção 3.2 foram apresentados os critérios adotados na escolha da estrutura do controlador, sendo ela: por simples realimentação ou com modelo interno, e a aplicação da RNA na arquitetura de controle, caracterizado pela forma direta ou indireta.

A Seção 3.3 abordou o neuro-controlador por modelo inverso, focando em sua arquitetura e os métodos de obtenção da RNA para este tipo de controlador.

O neuro-controlador preditivo abordado na Seção 3.4, teve sua conceituação oriunda no MPC, assim, nessa seção foi apresentado o MCP de forma simplificada, fornecendo um enten-

dimento sobre a arquitetura e RNA do neuro-controlador predito. Em seguida, foram apresentados os mecanismos de otimização, culminando no desenvolvimento do método de otimização por gradiente descendente, bem como sua aplicação em neuro-controlador preditivo.

Capítulo 4

Neuro-controlador proposto

4.1 Introdução

Pertencente à classe dos neuro-controladores preditivos, é proposto neste capítulo um neuro-controlador que usa o método de otimização unidimensional conhecido como Método de Eliminação de Região. Existem vários mecanismos de busca para reduzir a região em que o mínimo da função se encontra, no qual se destaca a Busca por seção Áurea.

Desta forma, no presente capítulo, esse método de otimização e os três mecanismos de busca mais conhecidos são apresentados. Por fim, é apresentado a arquitetura do neuro-controlador proposto, convenções adotadas e suas variações com modelo de referência.

4.2 Otimização: Método de Eliminação de Região

Dentre os métodos de otimização listados na Seção 3.4.4, o Método de Eliminação de Região, também conhecido como Método por Diminuição da Região de Busca, apresenta grandes vantagens por ser um método direto de otimização, caracterizado principalmente por não usar derivadas da função objetivo (Edgar e Himmelblau, 1988).

No entanto, esse método somente pode ser aplicado em problemas de otimização cuja função objetivo seja unidimensional. Como a aplicação do neuro-controlador proposto se destina a controle de sistemas do tipo SISO, essa limitação não é relevante.

O método de eliminação de região consiste em diminuir a região de incerteza de forma

iterativa, objetivando que a região final seja pequena e contenha a solução ótima. Seja uma função objetivo $J(u)$ a ser minimizada e a região de incerteza definida por dois pontos a e b sendo $a < b$.

Assumindo que o mínimo da função objetivo seja definido por qualquer valor u tal que $a \leq u \leq b$, escolhe-se dois pontos u_1 e u_2, sendo $u_1 \leq u_2$, compreendido entre a e b, e que dividem a região em três partes, como pode ser visto na Figura 4.1.

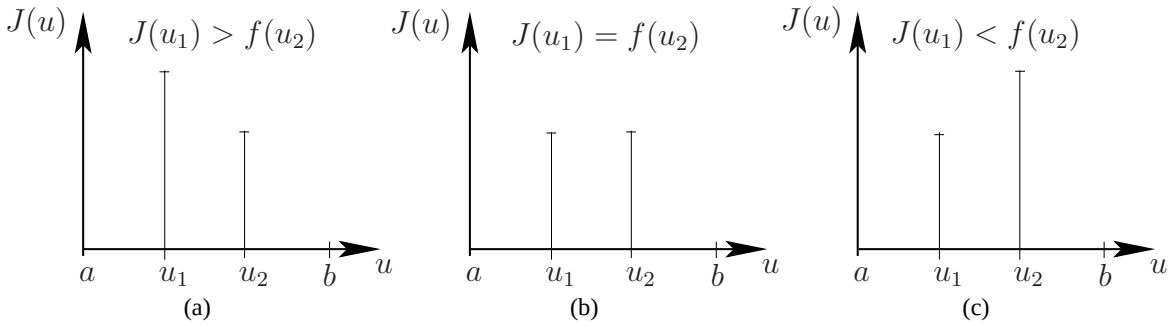

Figura 4.1: Região de incerteza dividida em três partes e possíveis valores para $J(u_1)$ e $J(u_2)$.

O mecanismo de redução da região de incerteza se dá por meio da eliminação de uma porção da região definida pela avaliação de $J(u_1)$ e $J(u_2)$ obedecendo os seguintes critérios (Edgar e Himmelblau, 1988):

- Se $J(u_1) > J(u_2)$ elimina-se toda região a esquerda de u_1 e, o novo a é igual u_1;
- Se $J(u_1) < J(u_2)$ elimina-se toda região a direita de u_2 e, o novo b é igual u_2;
- Se $J(u_1) = J(u_2)$ o valor ótimo u^* está entre u_1 e u_2, por questões computacionais (erro de arredondamento) uma boa prática é eliminar toda região a direita u_2 e, o novo b é igual u_2;

É importante notar que esse método quando aplicado a função multimodal poderá identificar um mínimo local como sendo o mínimo global para a função objetivo.

Como se trata de um método iterativo, toda vez que se elimina uma parte da região de incerteza, uma nova locação para os pontos u_1 e u_2 deve ser realizada. Este processo é realizado iterativamente até que pelo menos uma das condições seja atingida:

- O número máximo de iterações seja excedido;

- A diferença entre b e a seja menor ou igual ao valor da tolerância aceitável (L).

Atingindo pelo menos uma dessas condições, tanto a quanto b podem ser adotados como sendo o u^* para a função objetivo $J(u)$, pois admite-se que o novo intervalo definido por a e b contem a solução ótima. Como a tolerância L é uma valor muito pequeno, considera-se que um dos extremos é a própria solução ótima para a função objetivo.

O ponto principal do método de eliminação de região é caracterizado pelo mecanismo de localização dos novos u_1 e u_2 após a eliminação de um parte da região de incerteza. Esses mecanismos são chamados de Algoritmos de Busca, que se diferenciam pela forma de alocação dos novos u_1 e u_2, podendo ser resumidos em:

- Busca de Dois Pontos com Intervalos Iguais;
- Busca Dicotômica;
- Busca por Seção áurea;

os quais são apresentados nas próximas seções.

4.2.1 Busca de Dois Pontos com Intervalos Iguais

Este tipo de busca é definido pela igualdade de tamanho dos três intervalos alocados dentro da região de incerteza, ou seja, $u_1 - a = u_2 - u_1 = b - u_2$. Como a cada iteração k, é eliminado um destes intervalos, então pode-se afirmar que neste método de busca a região que contem u^* é reduzida a $\frac{1}{3}$ da região prevista na iteração anterior.

Considerando $L_0 = b - a$ na iteração $k = 0$, é possível afirmar que o tamanho do intervalo que contém u^* na iteração k é dado por:

$$L_k = \left(\frac{2}{3}\right)^k L_0 \tag{4.1}$$

Como em cada iteração é feita uma nova avaliação da função objetivo para o novo valor de u_1 e outra para o valor de u_2, e a definição do novo intervalo necessita de duas avaliações da função objetivo adicionais, a quantidade de avaliação da função objetivo ao longo de k iterações é dada por:

$$N_f = 2k + 2 \tag{4.2}$$

4.2.2 Busca Dicotômica

O método de busca dicotômica ou Bisseção (Corliss, 1977), consiste em avaliar a função em dois pontos u_1 e u_2, semelhante ao método de busca anterior, porém os pontos u_1 e u_2 estão infinitesimalmente separados um do outro, e estão alocados próximos ao centro da região de incerteza.

Como a distância entre u_1 e u_2 é muito pequena, pode-se dizer que a cada iteração a região que contém u^* é reduzida à metade da região dada na iteração anterior.

Assim, considerando $L_0 = b - a$ na iteração $k = 0$, por aproximação, pode-se dizer que o tamanho do intervalo que contém u^* na iteração k é dado por:

$$L_k = \left(\frac{1}{2}\right)^k L_0 \tag{4.3}$$

sendo que a quantidade de avaliação da função objetivo ao longo de k é expresso por:

$$N_f = 2k + 2 \tag{4.4}$$

caracterizado pelo mesmo conceito de avaliação da função objetivo do método de busca de dois pontos com intervalos iguais.

4.2.3 Busca por Seção Áurea

A estratégia empregada no método de busca por seção áurea consiste em locar dois pontos no interior da região de incerteza, tal que o intervalo eliminado em uma iteração tenha mesma proporção no intervalo total restante, não levando mais em consideração o tamanho do intervalo restante conforme nos métodos de busca anteriores (Edgar e Himmelblau, 1988).

Desta forma, a cada iteração somente um novo ponto tem que ser calculado, diferentemente do método de busca de dois pontos com intervalos iguais e busca dicotômica, visto que a localização de um dos pontos, u_1 ou u_2, permanecerá na nova iteração juntamente com o resultado da avaliação objetivo para seu valor.

Com este propósito, divide-se o intervalo $(b-a)$ em duas seções de tamanho c e d, conforme a Figura 4.2, de modo que a relação entre os tamanhos de c e d seja obrigatoriamente definida por:

$$\frac{c+d}{d} = \frac{d}{c} = \varphi \tag{4.5}$$

sendo φ uma constante conhecida como Número Áureo cujo valor é 1,618 (arredondado a três casas decimais).

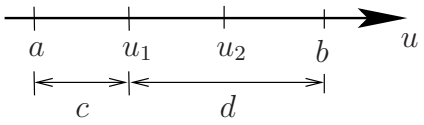

Figura 4.2: Divisão do intervalo $(b-a)$ em duas seções, c e d.

Para determinar os valores de c e d, Edgar e Himmelblau (1988) consideram $c+d=1$, sendo d a maior seção. Assim pode-se reescrever a Equação (4.5) da seguinte forma:

$$\frac{1}{d} = \frac{d}{c} \tag{4.6}$$

ou

$$\frac{1}{1-c} = \frac{1-c}{c} \tag{4.7}$$

que rearranjando, tem-se:

$$c^2 - 3c + 1 = 0 \tag{4.8}$$

cujas raízes são:

$$c = \frac{3 \pm \sqrt{5}}{2} = \left\{ \begin{array}{c} 2.618 \\ 0.382 \end{array} \right\} \tag{4.9}$$

como 2.618 é maior que intervalo inicialmente adotado $(c+d=1)$, somente o valor 0.382 é aceitável, portanto, o tamanho de cada uma das seções pertencente ao intervalo é:

$$c = \frac{3-\sqrt{5}}{2} \approx 0.382 \tag{4.10}$$

$$d = 1 - \frac{3-\sqrt{5}}{2} \approx 0.618 \tag{4.11}$$

Conforme mencionado anteriormente, somente uma nova localização do ponto u_1 ou u_2 deve ser feita a cada iteração, caracterizado pelas seguintes condições na iteração k:

- se $J(u_1) > J(u_2)$ então: $a \Leftarrow u_1$, $u_1 \Leftarrow u_2$, $J(u_1) \Leftarrow J(u_2)$ sendo apenas necessário calcular o novo $u_2 = a + d(b-a)$ e $J(u_2)$;
- Caso contrário: $b \Leftarrow u_2$, $u_2 \Leftarrow u_1$, $J(u_2) \Leftarrow J(u_1)$ sendo apenas necessário calcular o novo $u_1 = a + c(b-a)$ e $J(u_1)$;

Podendo-se dizer então, que o tamanho aproximado do intervalo que contém u^* na iteração k considerando $L_0 = b - a$ na iteração $k = 0$ é dado por:

$$L_k = (0.618)^k \, L_0 \tag{4.12}$$

pois a cada iteração é eliminado uma seção de tamanho c.

Como em cada iteração é feito somente uma única avaliação da função objetivo, para o novo valor de u_1 ou u_2, a quantidade de avaliação da função objetivo ao longo de k é composta por:

$$N_f = k + 2 \tag{4.13}$$

Portanto, nota-se que o método de busca por seção áurea apresenta desempenho superior comparado aos outros métodos de busca indicados anteriormente, visto que a cada iteração somente é necessário fazer uma avaliação da função objetivo.

Devido a estas características, o mecanismo de busca por seção áurea é o escolhido para compor o bloco de otimização do neuro-controlador proposto na Seção 4.3. O Algoritmo 1 no Apêndice A, exemplifica a implementação do mecanismo de busca por seção áurea.

4.3 Neuro-Controlador e Otimização com Método de Busca por Seção Áurea

O neuro-controlador preditivo apresenta em sua arquitetura um bloco de otimização, no qual diversas técnicas de otimização podem ser empregadas. Sendo mais comum o emprego do método de gradiente descendente, visto na Seção 3.4.5. No entanto, esse neuro-controlador contém algumas inconveniências, destacando-se:

- Velocidade de convergência lenta - inerente ao método de otimização (Bazaraa et al., 1993);
- Necessidade de determinar o valor de λ empiricamente - inerente à arquitetura do neuro-controlador (Schnitman e Fontes, 1999).

Visando eliminar tais inconvenientes, esta seção apresenta uma proposta de neuro-controlador preditivo, cujo bloco otimizador é formado pelo método de eliminação de região com busca por

seção áurea.

4.3.1 Arquitetura Fundamental do Neuro-Controlador Proposto

A arquitetura inicial do neuro-controlador proposto não diverge, em primeiro momento, da arquitetura clássica de neuro-controlador preditivo citada na seção 3.4, representada pela Figura 4.3.

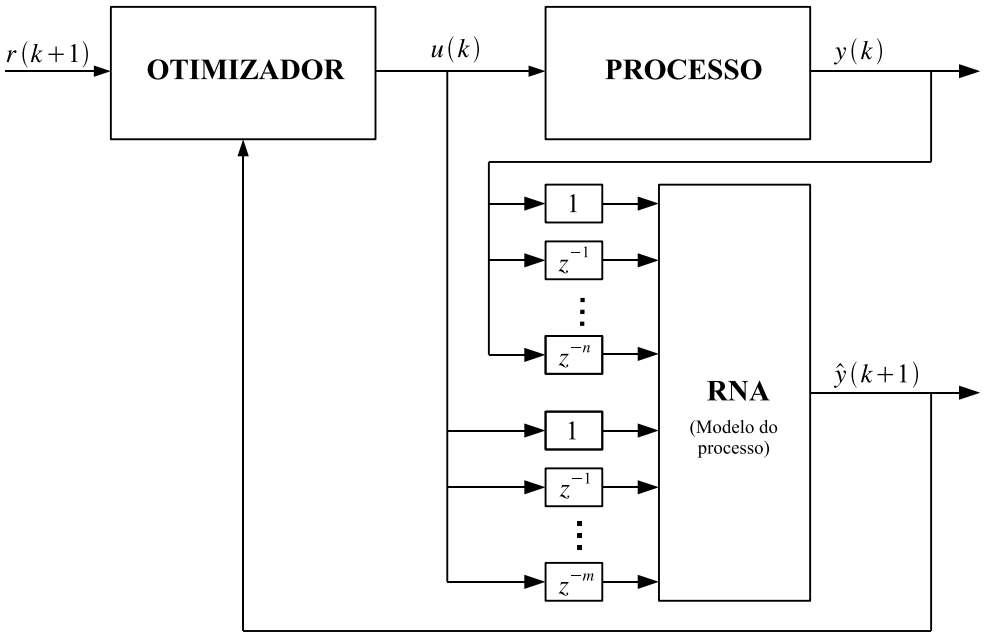

Figura 4.3: Arquitetura fundamental do neuro-controlador proposto.

Pode-se identificar na Figura 4.3 a presença dos seguintes sinais:

$r(k+1)$	sinal de referência desejado para o comportamento do processo no instante seguinte;
$u(k)$	ação de controle aplicada no processo no instante k;
$y(k)$	resposta do processo no instante k;
$\hat{y}(k+1)$	resposta do processo estimada pela RNA para o instante $k+1$;
z^{-1} ... z^{-n}	memória de linha de atraso derivada para y de ordem n;
z^{-1} ... z^{-m}	memória de linha de atraso derivada para u de ordem m;

4.3.2 Conceito do Neuro-Controlador Proposto

Considere que a RNA do modelo interno (modelo do processo) da Figura 4.3 seja uma boa representação matemática do processo, e que seja uma rede neural recorrente de entrada-saída (ver Seção 2.8.2). Assim, pode-se dizer que a saída $\hat{y}(k+1)$ é a resposta do processo no instante futuro predito pela RNA, sendo esta influenciada por $u(k)$ e $y(k)$ juntamente a seus valores passados, podendo ser equacionado como:

$$\hat{y}(k+1) = \mathcal{F}(\mathbf{x}) \tag{4.14}$$

sendo $\mathcal{F}$ a função mapeada pela RNA e $\mathbf{x}$ o vetor de entrada da mesma composto de:

$$\mathbf{x} = [y(k),y(k-1),...,y(k-n),u(k),u(k-1),...,u(k-m)] \tag{4.15}$$

Em um sistema de controle, o objetivo é impor ao processo um determinado comportamento, sendo este regido pelo sinal de referência $r(k+1)$. Note-se que o comportamento desejado é para o instante $k+1$. Assim, o controlador deve calcular uma ação de controle e aplicá-la no processo no instante k para que se atenda o objetivo, cuja ação de controle é calculada em função da diferença entre o sinal de referência e a resposta do processo.

Com esse intuito, se a RNA é uma boa representação matemática do processo, então a resposta do processo estimada pela RNA $\hat{y}(k+1)$ é uma boa aproximação para a resposta real do processo no instante $k+1$, logo pode-se usar a resposta estimada pela RNA para calcular o erro do controlador no instante $k+1$, sendo este dado por:

$$\hat{e}_c(k+1) = r(k+1) - \hat{y}(k+1) \tag{4.16}$$

Repare na Equação (4.14) que o valor $\hat{y}(k+1)$ é definido no instante k, sendo função de $y(k)$ e seus valores passados em conjunto de $u(k)$ e seus valores passados. Nota-se também que $y(k)$ é o valor atual lido do próprio processo, e neste instante apenas o valor de $u(k)$ não está definido. Portanto, o controlador deve calcular o valor $u(k)$ tal que minimize a diferença entre o sinal de referência e a resposta estimada do processo. Então, assumindo que

$$J = \mid \hat{e}_c(k+1) \mid \tag{4.17}$$

seja a função de custo e $u(k)$ a variável manipulada, um problema de otimização pode ser montado. Assim, se uma ação de controle $u(k)$ for calculada no instante atual, e esta minimize a diferença entre $r(k+1)$ e $\hat{y}(k+1)$ e, se esta mesma ação de controle for aplicada ao processo,

então a resposta do processo para instante $k+1$ tenderá a $r(k+1)$.

No entanto, sabe-se que um modelo matemático, seja RNA ou outra representação matemática, é uma aproximação do processo, passivos de pequenos erros de modelagem, influenciados por:

- ruído presente nos dados de treinamento;
- representatividades estatística dos dados de treinamento;
- escolha de uma estrutura adequada para o modelo;

Levando-se em consideração que a RNA é uma aproximação do processo passiva destes pequenos erros, uma diferença de resposta entre o modelo e processo sempre existirá, sobretudo quando o processo é submetido à distúrbios, resultando em aparecimento de erro estacionário.

Com o intuito de minimizar esta diferença, é possível calcular o quanto que a resposta do modelo diverge do processo real, ou seja, calcular o resíduo na iteração k. Sendo representado por ξ, o resíduo é expresso pela Equação (4.18).

$$\xi(k) = y(k) - \hat{y}(k) \tag{4.18}$$

sendo $\hat{y}(k)$ o valor predito pela RNA da resposta do processo na iteração k.

Assumindo que no instante $k+1$ os valores dos elementos do vetor **x**, definidos pela Equação (4.15), sejam idênticos aos do instante k, o resíduo $\xi(k+1)$ pode ser considerando igual ao do instante k. Logo, $\xi(k)$ pode ser incluído no cálculo do erro do controlador, Equação (4.16), resultando em:

$$\hat{e}_c(k+1) = r(k+1) - [\hat{y}(k+1) + \xi(k)] \tag{4.19}$$

Assim, a Equação (4.17) pode ser reescrita como:

$$J = \mid r(k+1) - [\hat{y}(k+1) + \xi(k)] \mid \tag{4.20}$$

para que o cálculo da ação de controle leve em consideração a diferença entre o valores preditos pela RNA e os valores reais do processo.

Como a estimativa de erro do neuro-controlador tende a diminuir, seu desempenho tende a aumentar, principalmente em operações de sistemas sujeito a uma grande quantidade de distúrbios. Pois, a presença de um distúrbio no processo provoca imediatamente, por meio de ξ, uma modificação no valor do cálculo da ação de controle.

Desta forma, o neuro-controlador proposto realiza, em primeiro momento, a quantificação do resíduo, e em seguida otimiza o valor de $u(k)$ para minimizar a função de custo expressa pela Equação (4.20), sendo que a otimização do valor de $u(k)$ é realizada pelo método de eliminação de regiões com busca por seção áurea. A arquitetura final do neuro-controlador proposto, contemplando a adição do resíduo na função de custos a ser minimizada, pode ser vista na Figura 4.4.

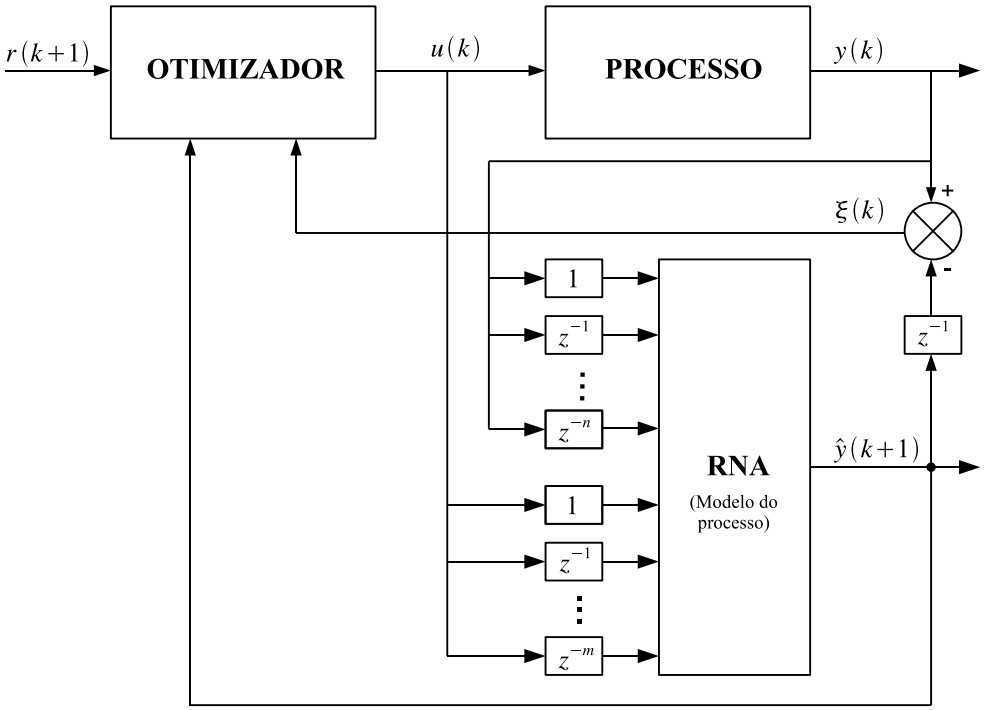

Figura 4.4: Arquitetura do neuro-controlador proposto.

4.3.3 Algoritmo do Neuro-Controlador Proposto

Assumindo que os valores possíveis para ação de controle $u(k)$ do neuro-controlador proposto estejam contidas no universo limitado por a e b. Determina-se que esses são respectivamente o limite inferior e superior da região de busca a ser usado pelo método de otimização. Como se deseja encontrar $u(k)$, pelo método de busca por seção áurea, que através da Equação (4.15) seguido da Equação (4.14) minimize a função de custo descrita na Equação (4.20), então são avaliados dois pontos que pertença a região de busca $(b-a)$, dado por:

$$u_1(k) = a + c(b-a) \tag{4.21}$$

$$u_2(k) = a + d(b-a) \tag{4.22}$$

sendo c e d constantes calculadas conforme as Equações (4.10) e (4.11), respectivamente.

Substituindo $u_1(k)$ e $u_2(k)$ na Equação (4.15) , dois novos vetores são produzidos:

$$\mathbf{x_1} = [y(k), y(k-1),..., y(k-n),u_1(k), u(k-1),..., u(k-m)] \tag{4.23}$$
$$\mathbf{x_2} = [y(k), y(k-1),..., y(k-n),u_2(k), u(k-1),..., u(k-m)] \tag{4.24}$$

que implicam em duas estimativas de estado do processo dado pela Equação (4.14):

$$\hat{y}_1(k+1) = \mathcal{F}(\mathbf{x_1}) \tag{4.25}$$
$$\hat{y}_2(k+1) = \mathcal{F}(\mathbf{x_2}) \tag{4.26}$$

e conseqüentemente, dois erros de controle prováveis a serem minimizados pela função objetivo da Equação (4.20):

$$J_1 =\mid r(k+1) - [\hat{y}_1(k+1) + \xi(k)] \mid \tag{4.27}$$
$$J_2 =\mid r(k+1) - [\hat{y}_2(k+1) + \xi(k)] \mid \tag{4.28}$$

que por meio do método de busca por seção áurea são avaliados J_1 e J_2 definindo qual ação de controle, $u_1(k)$ ou $u_2(k)$, deverá ser eliminado de forma a diminuir a região de busca.

É sabido que, no método de busca por seção áurea, a cada iteração um novo valor de a ou b é calculado de forma que o região de busca seja reduzida. Assim, quando a região de busca for reduzida a um valor aceitável (tolerância) ou o algoritmo de otimização atinja uma quantidade de iteração superior ao número desejado, a ação de controle $u(k)$ pode ser definida pelo limite inferior da região de busca restante, ou seja, $u(k) = a$, sendo essa aplicada ao processo.

Portanto, com a arquitetura de controle vista na Figura 4.4 e a implementação completa do neuro-controlador proposto é listada no Algoritmo 2 presente no Apêndice A.

4.3.4 Acrescentando Modelo de Referência ao Neuro-Controlador Proposto

O sobre-sinal da saída causado instantes após a mudança brusca de patamar do sinal de referência, pode torna-se um problema indesejável, prejudicando o desempenho do controlador. Segundo Kasparian e Batur (1998) um modelo de referência determina a trajetória desejada que a saída do processo deverá seguir dada uma mudança no sinal de referência (*set-point*) e, tal modelo de referência deve ser preferencialmente de primeira ordem e estável.

Assim, de forma a atenuar o impacto na saída do processo causado pela mudança brusca

do sinal de referência, pode-se aplicá-lo à entrada de um modelo de referência e a saída deste à entrada do controlador. Desta forma, o modelo de referência aplicado ao neuro-controlador pode ser visto na Figura 4.5.

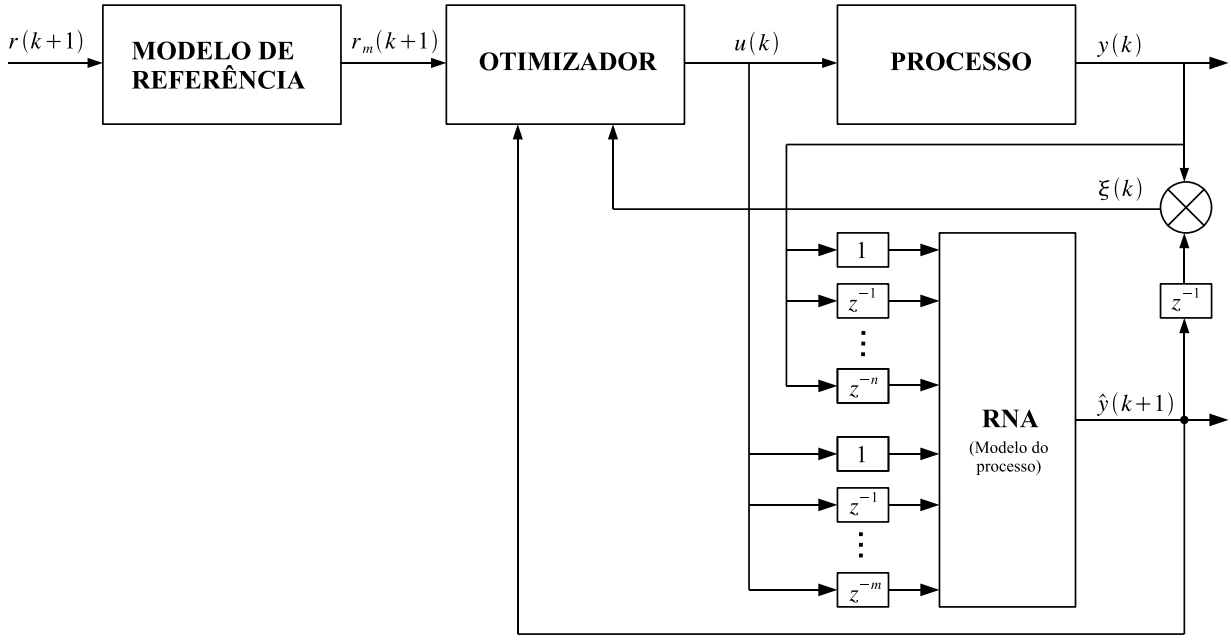

Figura 4.5: Arquitetura do neuro-controlador proposto com modelo de referência.

Considerando que a saída do modelo de referência seja dado por $r_m(k+1) = f_m(r(k+1))$, sendo f_m uma função que represente o modelo de referência e $r(k+1)$ a referência propriamente dita, a implementação completa do neuro-controlador proposto com modelo de referência pode ser listada como apresentado no Algoritmo 3 presente no Apêndice A.

4.4 Comentários Finais

O neuro-controlador proposto apresentado neste capítulo, faz uso do método de eliminação de região como otimizador. Dentro deste método de otimização o mecanismo de busca por seção áurea foi adotado, sendo este escolhido por apresentar eficiência superior aos demais deste método.

Na Seção 4.2 foi apresentado o método de otimização por eliminação de região, destacando sua facilidade por ser um método direto. Como mecanismos de busca foram evidenciados, a busca de dois pontos com intervalos iguais, a busca dicotômica e busca por seção áurea, sendo

ressaltado que a busca por seção áurea é a mais eficiente.

A arquitetura, conceituação e o algoritmo do neuro-controlador proposto foi apresentado na Seção 4.3. Nesta seção, destacou-se a formulação da função de custo dotada de mecanismo para quantificar dinamicamente o desvio de modelagem do processo e a justificativa de sua utilização. Foi também evidenciado a utilização e benefícios da introdução de um modelo de referência na arquitetura de controle.

Capítulo 5

Estudo de Casos, Resultados e Discussões

5.1 Introdução

Com o intuito de comparar e validar o neuro-controlador proposto, foi desenvolvida uma plataforma de testes que avalia o desempenho do controlador aplicado em modelos simulados e processo real (planta piloto de neutralização de pH), sendo os resultados apresentados neste capítulo.

De forma a organizar o estudo de casos, este capítulo está dividido em duas seções, uma para testes em sistemas simulados e outra para testes na planta real. Nelas serão descritos o processo, a obtenção na RNA e os resultados dos testes.

É também apresentado uma comparação de desempenho entre o neuro-controlador proposto, neuro-controlador com gradiente descendente e controlador PID para os testes realizados, sendo que para os testes em modelo simulado o uso do modelo de referência é empregado.

5.2 Sistema Simulado

Durante a primeira fase do desenvolvimento do neuro-controlador proposto, partiu-se de testes com modelos de processo lineares de primeira ordem até se chegar a modelos não-lineares invariantes no tempo. Os testes de rastreabilidade e rejeição de distúrbios bem como aplicação ou não de modelo de referência ao neuro-controlador foram colocados a prova em todos os modelos.

De forma a facilitar a execução dos testes, foi desenvolvido em MatLab® uma plataforma para realização de testes, sendo esta apresentada na Apêndice B, que mais tarde veio também a controlar a comunicação com a planta piloto de neutralização de pH.

Assim, dentre os modelos testados nesta seção, somente serão expostos os testes com o modelo que apresentou maior dificuldade de controle: o modelo de um processo não-linear invariante no tempo.

5.2.1 Descrição do Sistema

O modelo apresentado na Equação (5.1) foi um modelo originalmente usado por Chen et al. (1990) para teste de identificação de sistemas com redes neurais, sendo mais tarde usado por Kasparian e Batur (1998) para teste de neuro-controlador. Assim, este será o modelo de um processo não-linear invariante no tempo adotado para testes dos controladores.

$$\begin{aligned} y(k) &= \left[0{,}8 - 0{,}5e^{-y^2(k-1)}\right] y(k-1) - \left[0{,}3 + 0{,}9e^{-y^2(k-1)}\right] y(k-2) + \\ &\quad u(k-1) + 0{,}2u(k-2) + 0{,}1u(k-1)u(k-2) \end{aligned} \tag{5.1}$$

A Figura 5.1 representa a excitação em malha aberta desse modelo para valores de referência aleatórios compreendidos entre 0 e 1.

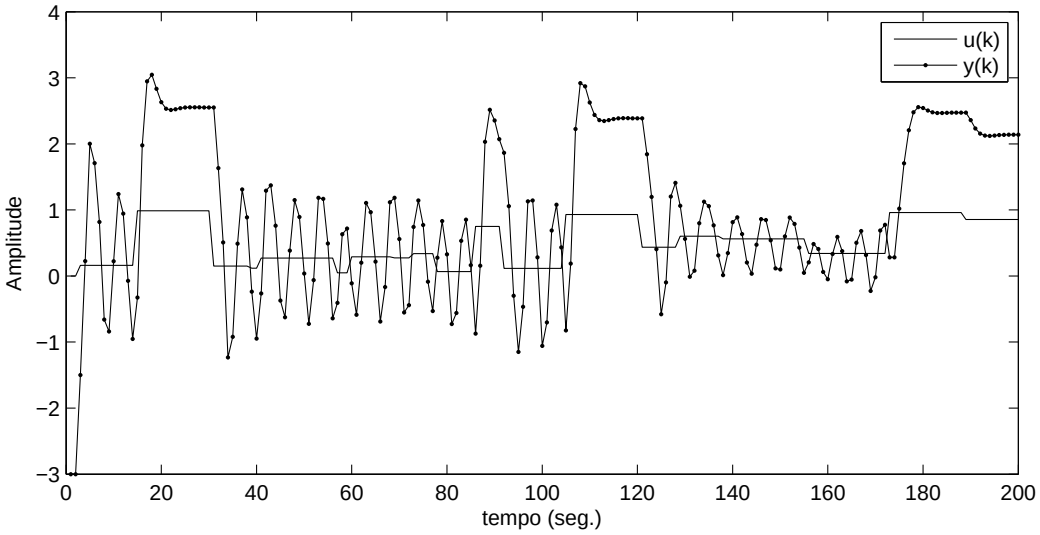

Figura 5.1: Excitação em malha aberta do modelo não-linear invariante no tempo da Equação (5.1), sendo $u(k)$ o sinal aplicado e $y(k)$ a resposta do modelo.

5.2.2 Obtenção da RNA

Como se trata de um neuro-controlador preditivo, que por sua vez faz uso de um modelo interno, a obtenção da RNA se dá pela identificação do processo em questão. Assim, os seguintes passos são seguidos:

- Coleta de dados por meio de simulação do modelo - Equação (5.1);
- Análise dos dados e dimensionamento RNA (arquitetura, função de ativação, etc.);
- Treinamento;
- Validação;
- Testes.

Durante o processo de coleta de dados, a simulação do processo foi realizada em malha aberta. Foi coletado um conjunto de dados para treinamento, validação e testes, sendo que, para cada conjunto, foi realizada uma simulação da Equação (5.1).

De posse dessas massas de dados, iniciou-se o projeto da RNA. Conforme indicado na Seção 3.4.3, neste trabalho é abordado o uso de RNA com apenas uma camada escondida, visto

que esta é capaz de mapear qualquer função contínua. Dessa forma, várias arquiteturas foram avaliadas, em termos de número de neurônio na camada escondida e tamanho de memória de linha de atraso derivada tanto para $y(k)$ quanto para $u(k)$.

Contudo, somente é apresentado a RNA que apresentou melhor desempenho no teste de validação, sendo esta do tipo MLP e treinada previamente antes de ser colocada em operação no neuro-controlador. Esta RNA possui as seguintes características:

- Somente uma camada escondida com quatro neurônios;
- Uma memória de linha de atraso derivada de ordem 2 para a saída $y(k)$;
- Uma memória de linha de atraso derivada de ordem 2 para a entrada $u(k)$;
- Funções de ativação do tipo sigmoidal tangente hiperbólica e linear para os neurônios ocultos e de saída, respectivamente.

Sendo que seu processo de treinamento se deu pelo uso de:

- Modo Treinamento em Lote ou batelada;
- Algoritmo de treinamento: *Levenberg-Marquardt Backpropagation*;
- Estratégia de ampliação da generalização: Parada antecipada (*Early Stopping*) + regularização bayesiana.

Durante o treinamento foram utilizados um conjunto de 5000 (cinco mil) amostras para o treinamento e outro com 2000 (duas mil) amostras para validação. Um conjunto de 200 (duzentas) amostras foi usado para o teste da RNA em regime de predição livre ao término do treinamento e validação, sendo este apresentado na Figura 5.2.

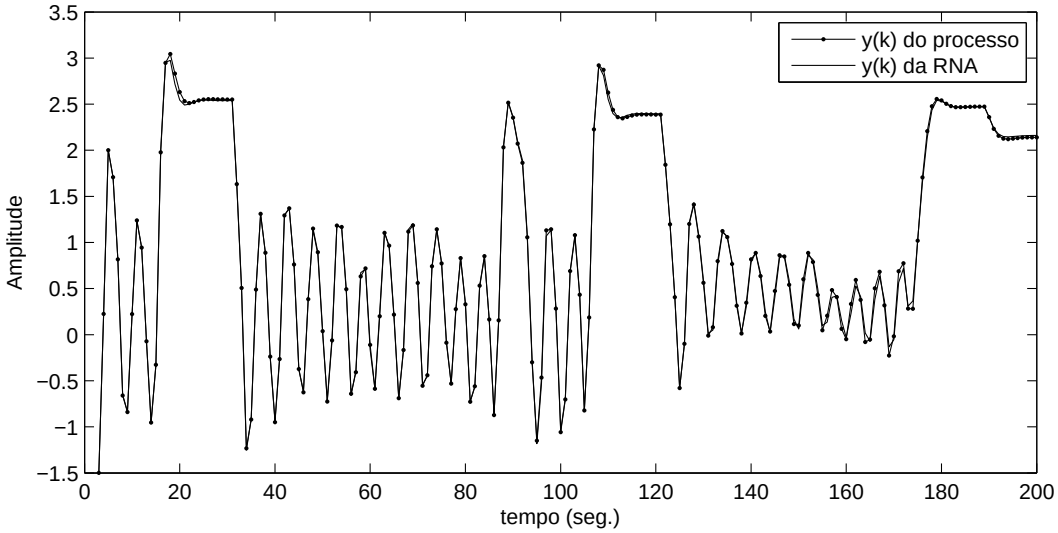

Figura 5.2: Teste da RNA em regime de predição livre.

Na Figura 5.3 pode ser vista a diferença entre a resposta do processo e a resposta da RNA, o resíduo ξ.

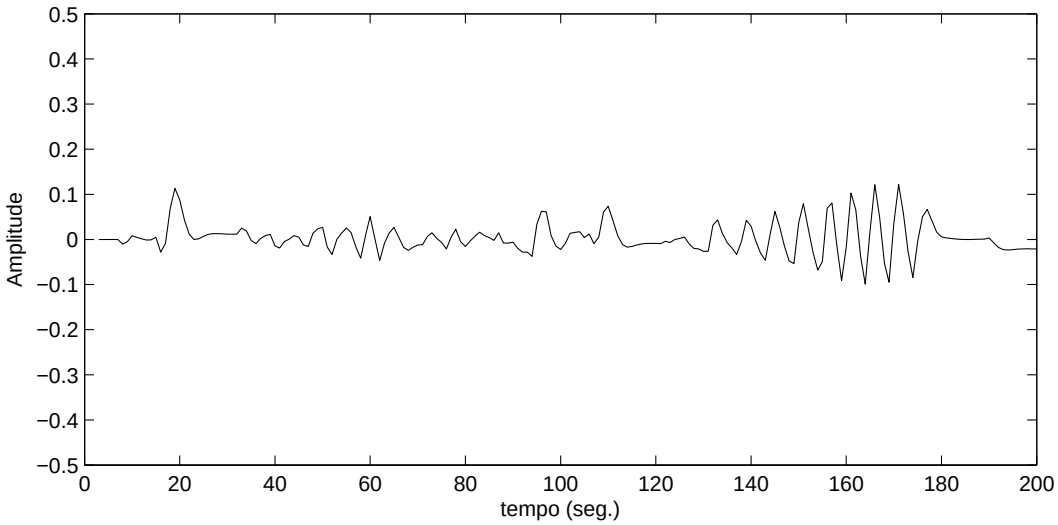

Figura 5.3: Resíduo ξ - Diferença entre resposta do processo e resposta da RNA.

Após a obtenção do modelo, o neuro-controlador proposto foi implementado conforme a Figura 4.4 e, os resultados obtidos podem ser vistos na seção seguinte.

5.2.3 Resultados e Discussões

Esta seção destaca os resultados alcançados com o neuro-controlador proposto e o compara com o neuro-controlador preditivo com otimização por gradiente descendente e também com o controlador clássico PID (Callender e Stevenson, 1939; Ogata, 1997). A RNA obtida na Seção 5.2.2 foi usada como modelo interno para os dois neuro-controladores, isto foi feito objetivando comparar o desempenho deles com a mesma RNA.

Dois tipos de testes são avaliados para compor o desempenho dos controladores: rastreabilidade e rejeição de distúrbios, sendo estes realizados sob a arquitetura de controle sem modelo de referência e com modelo de referência.

A parametrização dos controladores foi:

- Para o neuro-controlador preditivo com otimizador por gradiente descendente, o valor de λ foi determinado empiricamente, sendo $\lambda = 0{,}06$;
- Para o controlador PID, seus ganhos foram: Proporcional $KP = 0{,}06$, Integral $KI = 0{,}3$ e Derivativo $KD = 0{,}8$;
- Para o neuro-controlador proposto foi definido o Número máximo de iterações $N = 10$ e a Tolerância (região final que contenha $u(k)$ ótimo) $tol = 1\%$ do universo de busca.

Para todos os controladores foi determinado que a ação de controle $u(k)$ deveria estar compreendida entre 0 e 1.

De forma a padronizar a exibição dos resultados dos testes realizados com os controladores, toda figura com esta finalidade é composta por três gráficos, a saber:

- Gráfico (a): Comportamento do processo, sendo $r(k)$ a referência e $y(k)$ a resposta do processo;
- Gráfico (b): Ação de controle e distúrbios, sendo $u(k)$ a ação de controle e $d(k)$ o distúrbio;
- Gráfico (c): Integral do Erro Absoluto (IAE) calculado segundo a função $IAE(k) = IAE(k-1) + |r(k) - y(k)|$, sendo apresentado em destaque o valor final obtido no teste.

Nos testes de rastreabilidade o sinal de referência foi gerado aleatoriamente entre 0 e 1, e não houve a presença de distúrbios.

Nos testes de rejeição de distúrbios, o sinal de referência foi mantido em $0{,}5$ e os distúrbios, representado por $d(k)$, foram gerados aleatoriamente entre -$0{,}5$ e 0,5 e adicionados a resposta do processo antes de ser apresentado ao controlador, ou seja, $y(k) = d(k) + y(k)$.

O modelo de referência adotado nos testes é caracterizado como um sistema de primeira ordem e estável (Kasparian e Batur, 1998), sendo este apresentado na Equação (5.2).

$$r_m(k+1) = 0{,}6r(k+1) + 0{,}4r_m(k); \tag{5.2}$$

Todos os resultados dos testes expostos em seguida, foram realizados utilizando a plataforma de testes apresentada no Apêndice B.

- **Teste de rastreabilidade (sem Modelo de Referência)**

Os resultados dos testes de rastreabilidade realizados com os controladores sem o uso de modelo de referência são apresentados nas figuras 5.4, 5.5 e 5.6 referentes ao neuro-controlador preditivo com otimizador por gradiente descendente, controlador PID e neuro-controlador proposto, respectivamente.

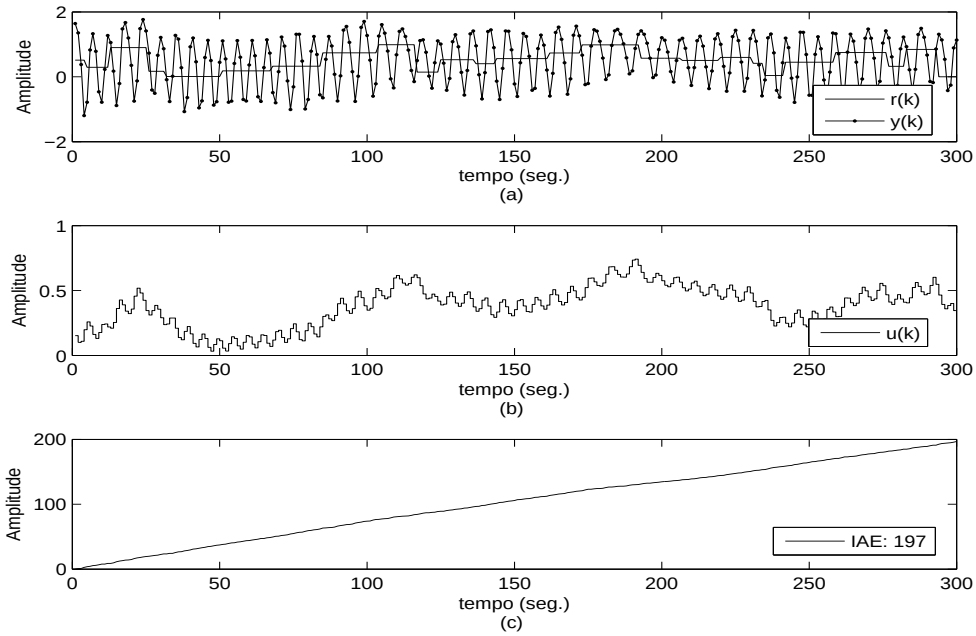

Figura 5.4: Sistema simulado: resultado do teste de rastreabilidade com neuro-controlador preditivo com otimizador por gradiente descendente sem modelo de referência

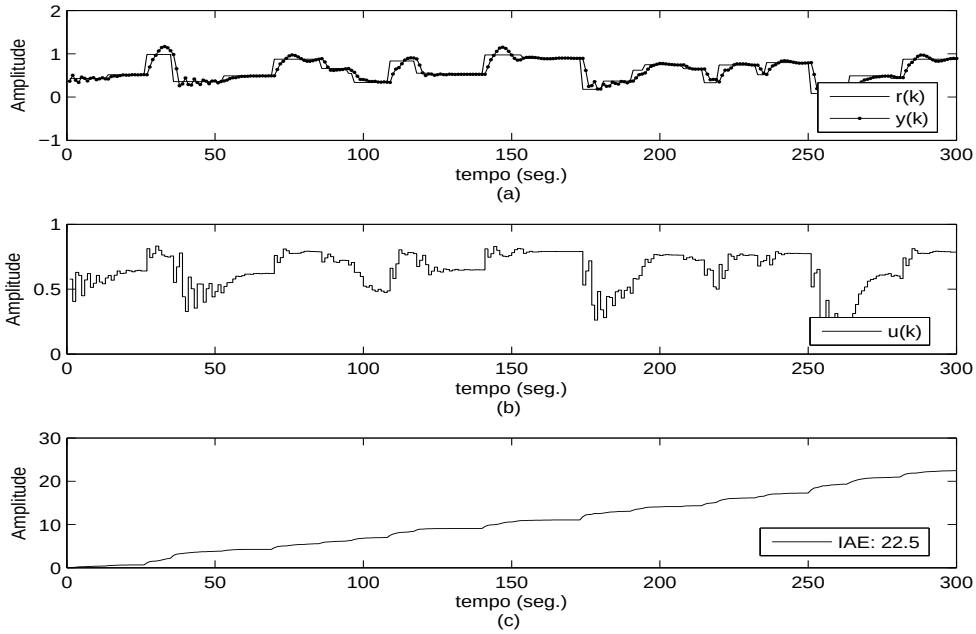

Figura 5.5: Sistema simulado: resultado do teste de rastreabilidade com controlador PID sem modelo de referência.

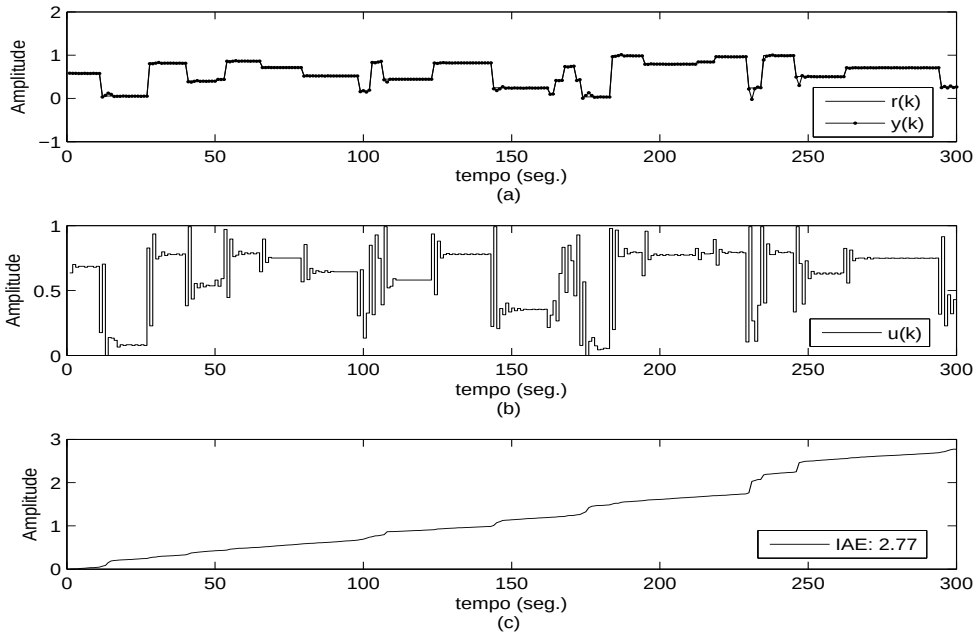

Figura 5.6: Sistema simulado: resultado do teste de rastreabilidade com neuro-controlador proposto sem modelo de referência.

Foi evidenciado que o neuro-controlador com otimizador por gradiente descendente não consegue estabilizar o processo, isto porque suas ações de controle são alteradas de forma bem suave. Percebe-se também que o controlador PID mesmo tendo resultado inferior ao do neuro-controlador proposto, tem ações de controle mais suaves. No entanto, o neuro-controlador proposto apresenta menores níveis de sobre-sinal e estabilização mais rápida.

- **Teste de rastreabilidade (com Modelo de Referência)**

Os resultados dos testes de rastreabilidade realizados com os controladores usando modelo de referência são apresentados nas Figuras 5.7, 5.8 e 5.9 referentes ao neuro-controlador preditivo com otimizador por gradiente descendente, controlador PID e neuro-controlador proposto, respectivamente.

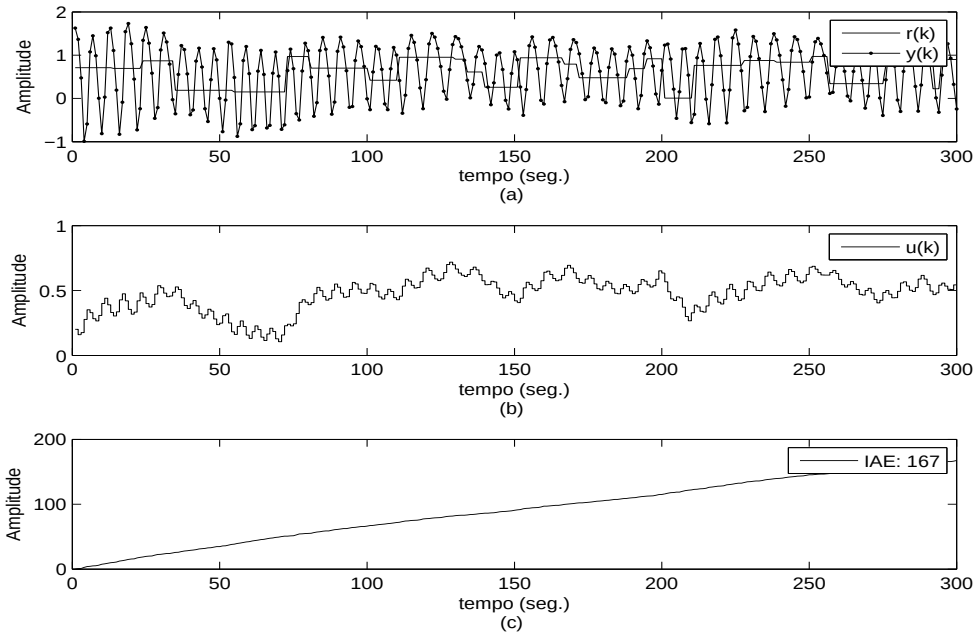

Figura 5.7: Sistema simulado: resultado do teste de rastreabilidade do neuro-controlador preditivo com otimizador por gradiente descendente dotado de modelo de referência.

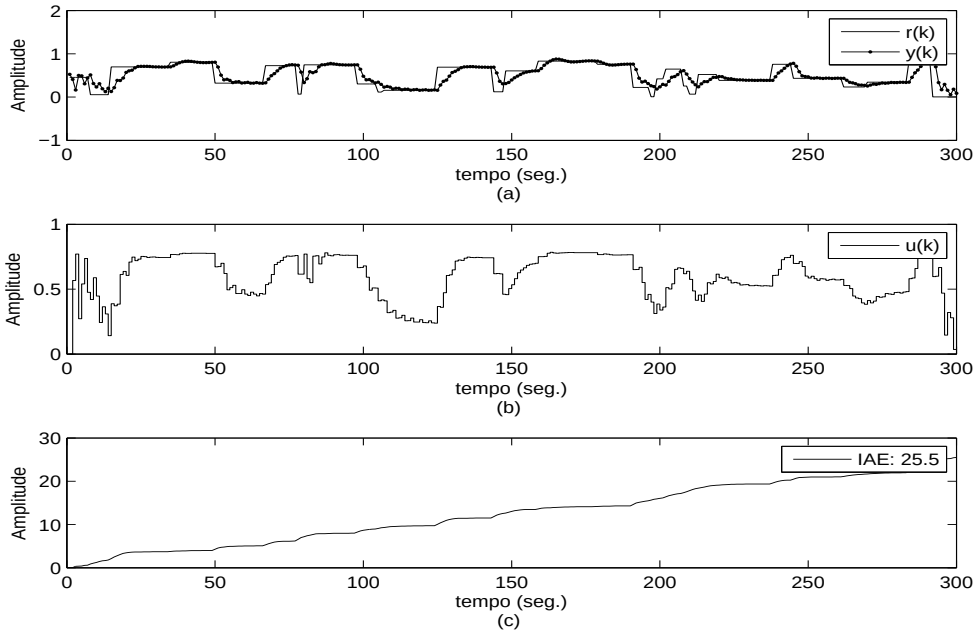

Figura 5.8: Sistema simulado: resultado do teste de rastreabilidade do controlador PID dotado de modelo de referência.

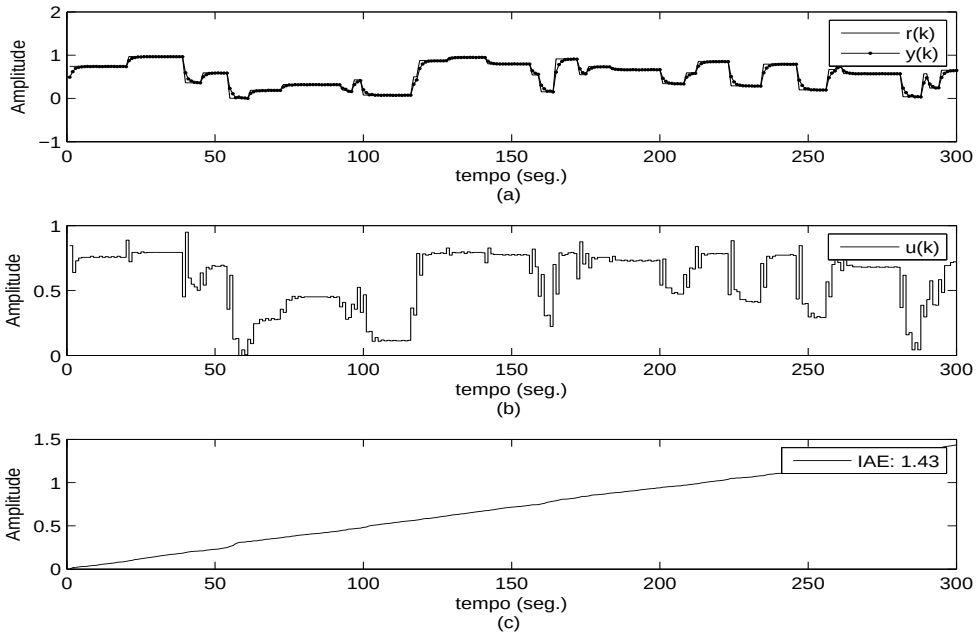

Figura 5.9: Sistema simulado: resultado do teste de rastreabilidade do neuro-controlador proposto dotado de modelo de referência.

A inserção de um modelo de referência não melhora o desempenho do neuro-controlador com otimizador por gradiente descendente. Percebe-se que para o controlador PID o sobre-sinal é reduzido consideravelmente, mas, o tempo de acomodação aumenta. Para o neuro-controlador proposto, ocorre uma perda de velocidade de estabilização, mas, a ação de controle se torna mais suave.

- **Teste de rejeição de distúrbios (sem Modelo de Referência)**

Os resultados dos testes de rejeição de distúrbios realizados com os controladores sem uso de modelo de referência são apresentados nas Figuras 5.10, 5.11 e 5.12 referentes ao neuro-controlador preditivo com otimizador por gradiente descendente, controlador PID e neuro-controlador proposto, respectivamente.

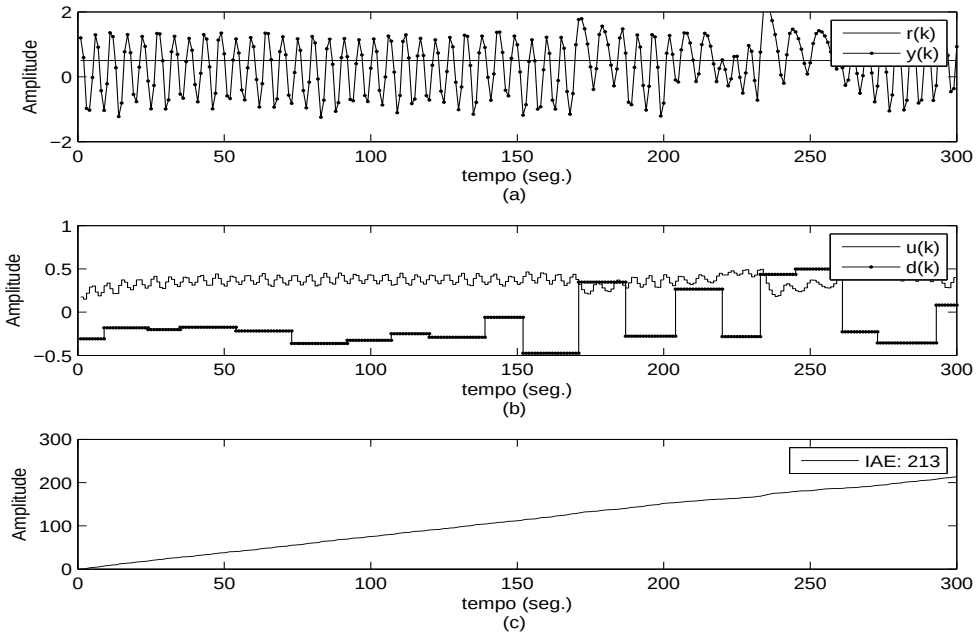

Figura 5.10: Sistema simulado: resultado do teste de rejeição de distúrbios do neuro-controlador preditivo com otimizador por gradiente descendente sem modelo de referência.

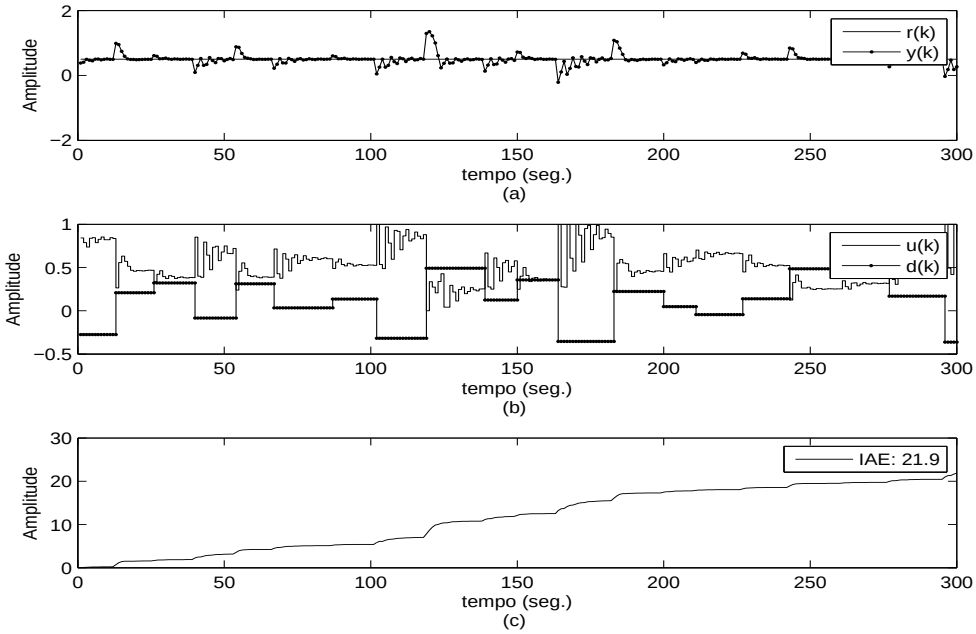

Figura 5.11: Sistema simulado: resultado do teste de rejeição de distúrbios do controlador PID sem modelo de referência.

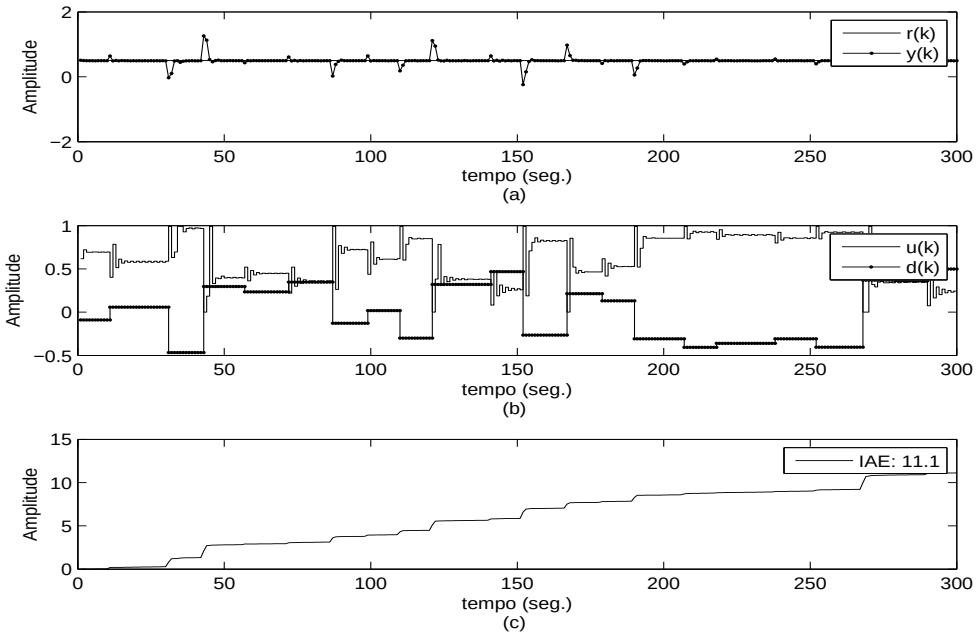

Figura 5.12: Sistema simulado: resultado do teste de rejeição de distúrbios do neuro-controlador proposto sem modelo de referência.

Percebe-se que o neuro-controlador com otimizador por gradiente descendente não consegue rejeitar os distúrbios e estabilizar o processo. Nota-se também a presença de oscilações consideráveis em y, no teste com o controlador PID no momento de alteração do distúrbio. Para o neuro-controlador proposto, têm-se menores níveis de oscilação em y e estabilização mais rápida, porém com alteração brusca da ação de controle.

- **Observações**

Em cenários de rejeição de distúrbios, o modelo de referência não influencia no desempenho dos controladores, pois o sinal de referência é mantido constante, logo: $r_m(k+1) = r(k+1)$. Assim, os resultados dos testes realizados nesse cenário se mantêm idênticos ao do sem modelo de referência, razão pela qual não são apresentados.

Nota-se que em todos os testes o neuro-controlador proposto apresenta melhor desempenho que os demais. Sua ação de controle em cada iteração acarreta em um resposta mais rápida do sistema, isto devido a capacidade de mudar o valor da ação de controle de um patamar para outro bruscamente no decorrer do tempo.

A adição de um modelo de referência faz com que os controladores suavizem sua ação de controle somente no momento da mudança do sinal de referência, sendo possível notar que o desempenho do controlador aumenta.

A Tabela 5.1 apresenta um quadro comparativo do IAE (Integral do Erro Absoluto) dos controladores obtidos ao final dos testes, tanto para o cenário de rastreabilidade quanto para a rejeição de distúrbios.

Tabela 5.1: IAE (Integral do Erro Absoluto) dos controladores: Sistema não-linear invariante no tempo (simulado)

Teste	Neuro-controlador preditivo com otimizador por gradiente descendente	Controlador PID	Neuro-controlador proposto
Rastreabilidade sem modelo de ref.	197	22,5	2,77
Rastreabilidade com modelo de ref.	167	25,5	1,43
Rejeição de distúrbios sem modelo de ref.	213	21,9	11,1

5.3 Planta piloto de neutralização de pH

Entre os mais diversos processos não-lineares, destaca-se o processo de neutralização de pH, sendo este referência em controles de processos químicos, pois apresenta grande variação de comportamento oriundo de pequenas mudanças nas condições do processo (Lee et al., 2001; Yoo et al., 2004).

Objetivando demonstrar o desempenho dos controladores em processo real que apresenta alto grau de não-linearidade, foram realizados testes em uma planta piloto de neutralização de pH instalada no laboratório de Modelagem Otimização e Controle de Processos - MOCP, do Centro Universitário do Leste de Minas Gerais - UnilesteMG, Coronel Fabriciano (MG) - Brasil.

Esta planta foi construída com fomento do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, com o objetivo de ser uma plataforma para realização de testes de identificação de processos e controle mono e multi-variáveis (Campos, 2007).

5.3.1 Potencial Hidrogeniônico - pH

As concentrações *hidrogeniônica* [H^+] e *hidroxiliônica* [OH^-] em uma solução estão inversamente correlacionadas, ou seja, o aumento de uma acarreta a diminuição da outra e vice-versa. Assim, em uma solução ácida há mais íons de H^+ do que íons de OH^- e o inverso para soluções alcalinas, e quando suas concentrações são iguais, a solução é dita neutra (Ylén, 2001).

O potencial hidrogeniônico (pH) avalia o nível de acidez de uma solução, cujo valor por convenção, varia entre 0 a 14, calculado conforme Equação (5.3).

$$pH = -log_{10}\{H+\} \tag{5.3}$$

Através da Equação (5.3), a classificação de uma solução, do ponto de vista do potencial hidrogeniônico (pH), estando a uma temperatura de 25^oC é dada por (Ylén, 2001):

- $pH < 7 \Rightarrow$ ácido
- $pH = 7 \Rightarrow$ neutro
- $pH > 7 \Rightarrow$ alcalino

5.3.2 Solução tampão

São soluções que atenuam a variação dos valores de pH (ácido ou alcalino), mantendo-o aproximadamente constante, mesmo com adição de pequenas quantidades de ácidos ou bases (Ylén, 2001). As soluções tampão são geralmente formadas por um ácido fraco e um sal desse ácido, ou, então, por uma base fraca e um sal dessa base (Lehninger et al., 1995). Em situações práticas (Ex.: processos bioquímicos) onde o valor do pH deve ser mantido perto de um valor ótimo, uma solução tampão é adicionada ao processo (Marzzoco e Torres, 2007).

5.3.3 Descrição da Planta

Objetivando a neutralização do pH, na planta piloto, duas ou mais soluções são colocadas continuamente em um reator por meio de bombas dosadoras, sendo que estas e a sonda de medição do pH estão conectadas a um microcomputador via interface de comunicação USB.

Esta planta piloto de neutralização de pH, usada para realização de testes dos controladores, é constituída basicamente pelos seguintes elementos:

- três bombas peristálticas para dosagem de reagentes com sinal de controle de 4 a $20mA$;
- um agitador magnético;
- um medidor de pH com precisão de $0{,}01pH$ com transmissor de 4 a $20mA$;
- um tanque para reagente ácido ($60l$);
- um tanque para solução base ($60l$);
- um tanque para solução tampão ($60l$);
- um reator ($2{,}1l$);
- um tanque coletor ($140l$);
- duas placas de aquisição de dados USB:
  - modelo NI USB-6008;
  - oito canais de entradas analógicas com resolução de 12 (doze) bits;
  - dois canais de saídas analógicas com resolução de 12 (doze) bits;
  - doze canais de entradas ou saídas digitais configuráveis;

  - taxa de amostragem de 10kS/s;

- um microcomputador:
  - processador Intel® Core^{TM} 2 Duo de 1,66GHz;
  - memória RAM de 1,0 (um) Gbytes;
  - HD (Disco Rígido) de 80 (oitenta) Gbytes;
  - placa de vídeo de 256 (duzentos e cinqüenta e seis) Mbytes;
  - quatro portas USB;

A Figura 5.14 apresenta a foto da planta piloto de neutralização de pH e, a Figura 5.14 apresenta seu diagrama funcional. Mais detalhes sobre a construção, croqui estrutural, especificação detalhada dos materiais, métodos de calibração e preparação de soluções podem ser vistos em Campos (2007).

Figura 5.13: Foto da planta piloto de neutralização de pH.

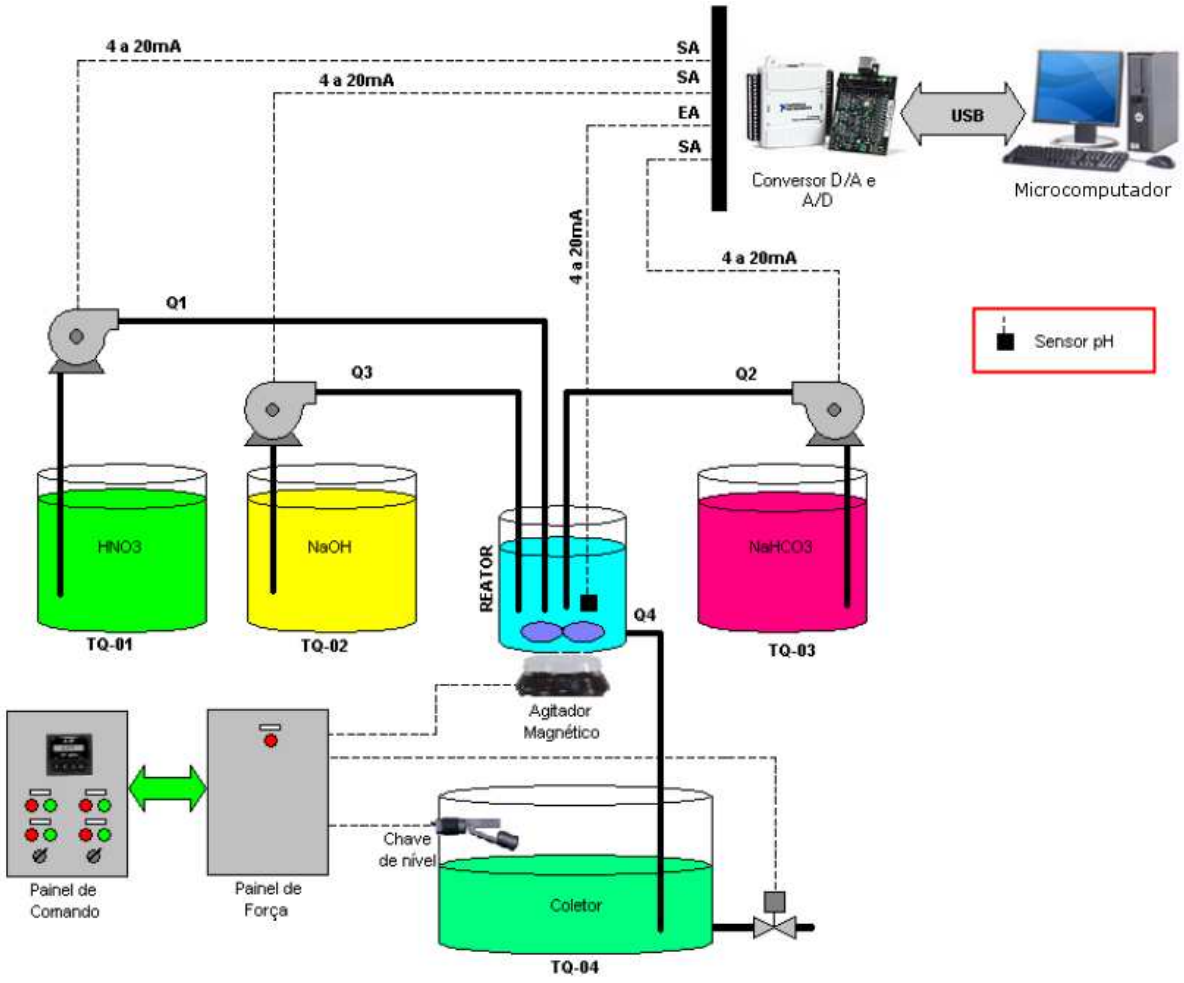

Figura 5.14: Diagrama funcional da planta piloto de neutralização de pH. Fonte: Campos (2007).

As soluções usadas nos testes dos controladores foram preparadas seguindo o procedimento de Campos (2007), exceto para a solução de ácido nítrico (HNO_3) que foi substituída por uma solução de ácido clorídrico (HCl).

Portanto, o reator foi alimentado ,continuamente, com as seguintes vazões:

$$\begin{aligned} Q1 &\Rightarrow HCl \rightarrow H^+ + Cl^- \\ Q2 &\Rightarrow NaHCO_3 \rightarrow Na^+ + HCO_3^- \\ Q3 &\Rightarrow NaOH \rightarrow Na^+ + OH^- \end{aligned}$$

5.3.4 Obtenção da RNA

A obtenção da RNA foi realizada de forma semelhante à do processo simulado. Como se trata de um neuro-controlador preditivo, que por sua vez faz uso de um modelo interno, a obtenção da RNA se dá pela identificação do processo em questão. Assim, os seguintes passos foram seguidos:

- Coleta de dados do processo por meio das placas de aquisição de dados USB;
- Análise dos dados e projeto da RNA (arquitetura, função de ativação, etc.);
- Treinamento;
- Validação;
- Testes

Mantendo-se constante a vazão da solução tampão ($NaHCO_3$) igual a $Q2 = 0{,}1ml/seg$ e a solução de ácido clorídrico (HCl) igual a $Q1 = 2{,}74ml/seg$, a coleta de dados se deu pela leitura da sonda de medição de pH e da vazão de hidróxido de sódio aplicada ao processo em malha aberta, sendo que a vazão $Q3$, hidróxido de sódio ($NaOH$), foi persistentemete variada entre valores de $0ml/seg$ a $4{,}4ml/seg$.

A taxa de aquisição de dados foi de 1 (uma) amostra por segundo e durante a avaliação do conjunto de dados. Através da auto-covariância dos sinais de pH, observou ser necessária uma decimação de 20 (vinte) amostras. Esta avaliação segue os procedimentos propostos por Aguirre (2004). Assim, para o treinamento da RNA e todos os testes dos controladores utilizou-se 1 (uma) amostra a cada 20 (vinte) segundos. A Figura 5.15 apresenta a massa após a decimação.

As coletas de dados para treinamento, validação e testes foram realizadas a partir da excitação da planta em tempos e períodos distintos. Então, iniciou-se o projeto da RNA, abordando o uso de RNA do tipo MLP com apenas uma camada escondida, devido aos fatores já mencionados na Seção 3.4.3.

Dessa forma, várias arquiteturas foram avaliadas, em termos de número de neurônios na camada escondida e tamanho de memória de linha de atraso derivada tanto para $y(k)$ quanto para $u(k)$. Contudo, somente é apresentado a RNA que apresentou melhor desempenho no teste de validação, cujas características são:

- Somente uma camada escondida com três neurônios;

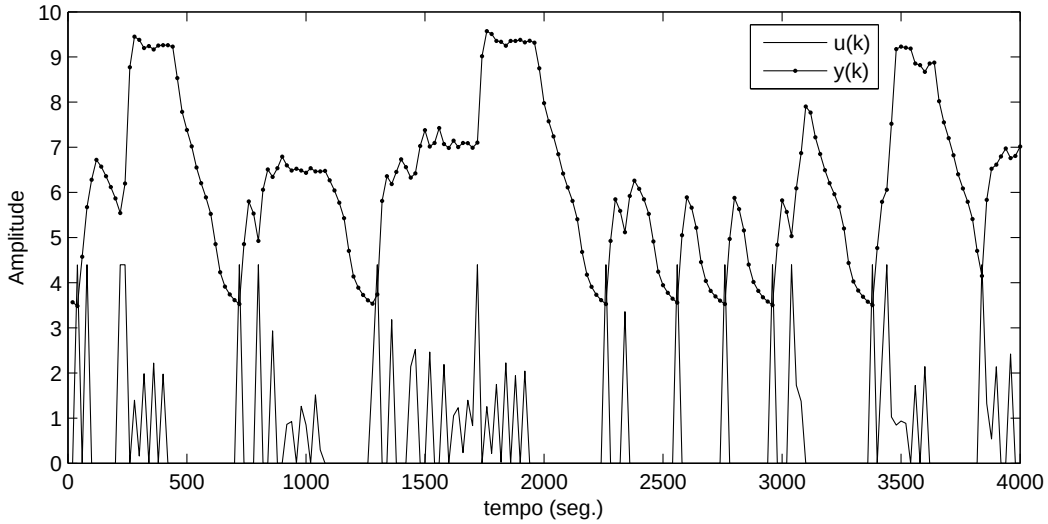

Figura 5.15: Excitação em malha aberta da planta piloto de neutralização de pH, sendo $u(k)$ a vazão de $NaOH$ e $y(k)$ o pH.

- Uma memória de linha de atraso derivada de ordem 3 para a saída $y(k)$;
- Uma memória de linha de atraso derivada de ordem 2 para a entrada $u(k)$;
- Funções de ativação do tipo sigmoidal tangente hiperbólica e linear para os neurônios ocultos e de saída, respectivamente.

Seu processo de treinamento se deu pelo uso de:

- Modo Treinamento em Lote ou batelada;
- Algoritmo de treinamento: *Levenberg-Marquardt Backpropagation*;
- Estratégia de ampliação da generalização: Parada antecipada (*Early Stopping*) + regularização bayesiana.

Durante o treinamento foram utilizados um conjunto de 320 (trezentas e vinte) amostras para o treinamento e 300 (trezentas) amostras para validação, sendo estes coletados de forma independente no processo. O conjunto de 200 (duzentas) amostras, visto na Figura 5.15, foi usado para o teste final da RNA com predição livre, sendo este apresentado na Figura 5.16.

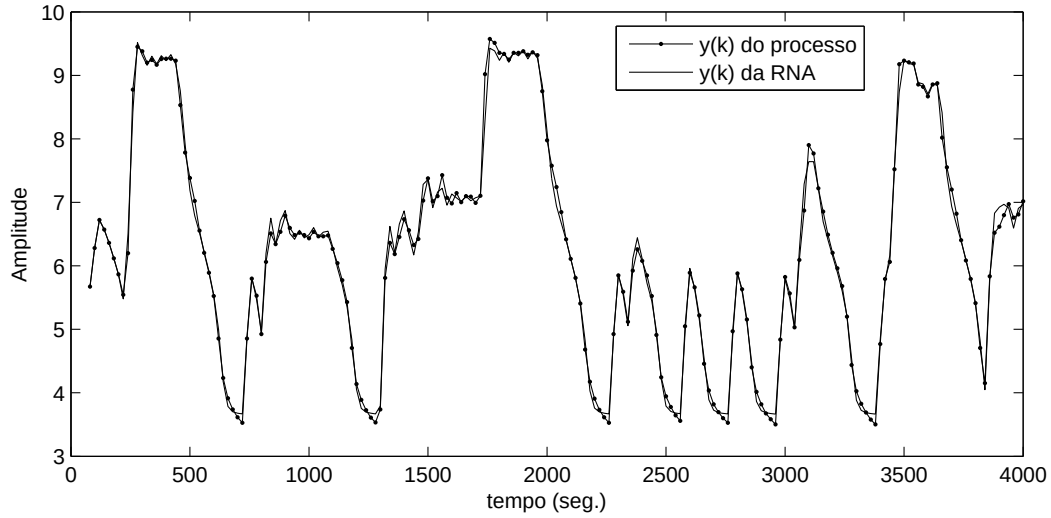

Figura 5.16: Teste de validação da RNA.

Como se pode notar na Figura 5.16 ocorre uma pequena diferença entre a resposta do processo e a resposta da RNA, que são os resíduos ξ. Isto ocorre devido a erro de modelagem e também a presença de ruídos nas medições. A Figura 5.17 apresenta os resíduos ξ.

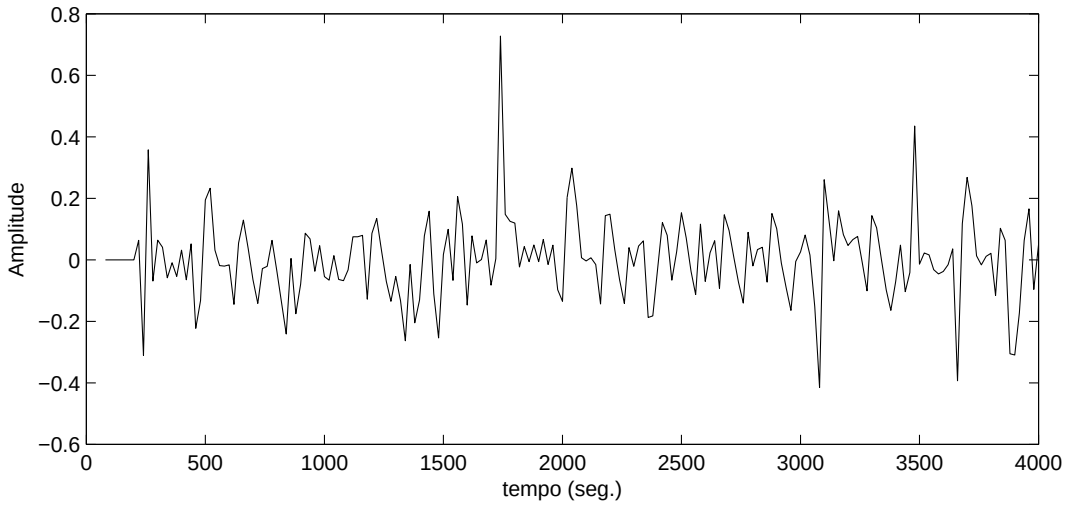

Figura 5.17: Resíduo ξ - Diferença entre resposta do processo e resposta da RNA.

Após a obtenção do modelo, o neuro-controlador proposto foi implementado conforme a Figura 4.4. Os resultados obtidos podem serem vistos na seção seguinte.

5.3.5 Resultados e Discussões

Esta seção destaca os resultados alcançados com neuro-controlador proposto e o compara com o neuro-controlador preditivo com otimização por gradiente descendente e também com o controlador clássico PID. A RNA obtida na Seção 5.3.4 é a usada como modelo interno para os dois neuro-controladores, isto é feito objetivando comparar o desempenho deles utilizando a mesma RNA.

Dois tipos de testes são avaliados para compor o desempenho dos controladores: rastreabilidade e rejeição de distúrbios, sendo estes realizados sob a arquitetura de controle sem modelo de referência.

A parametrização dos controladores foi:

- Para o neuro-controlador preditivo com otimizador por gradiente descendente, o valor de λ foi determinado empiricamente, sendo $\lambda = 2$;
- Para o controlador PID, seus ganhos foram: Proporcional $KP = 1{,}48$, Integral $KI = 0{,}41$ e Derivativo $KD = 1{,}48$;
- Para o neuro-controlador proposto foi definido o Número máximo de iterações $N = 10$ e a Tolerância (região final que contenha $u(k)$ ótimo) $tol = 1\%$ do universo de busca.

Para todos os controladores foi determinado que a ação de controle $u(k)$ seria a manipulação da vazão $Q3$, hidróxido de sódio ($NaOH$), e deveria estar compreendida entre $0ml/seg$ e $4{,}4ml/seg$, exceto para o neuro-controlador preditivo com otimizador por gradiente descendente.

De forma a padronizar a exibição dos resultados dos testes realizados com os controladores, toda figura com esta finalidade é composta por três gráficos, a saber:

- Gráfico (a): Comportamento do processo, sendo $r(k)$ a referência e $y(k)$ a resposta do processo;
- Gráfico (b): Ação de controle e distúrbios, sendo $u(k)$ a ação de controle dada pela vazão $Q3$ de $NaOH$ e $d(k)$ a variação da vazão $Q1$ de HCl, caracterizada com distúrbio;
- Gráfico (c): Integral do Erro Absoluto (IAE) calculado segundo a função $IAE(k) = IAE(k-1) + |r(k) - y(k)|$, sendo apresentado em destaque o valor final obtido no teste.

Nos testes de rastreabilidade o sinal de referência (pH) foi gerado aleatoriamente entre $4{,}0$ e $10{,}0$. Nesses testes não houve a presença de distúrbios.

Nos testes de rejeição de distúrbios, o sinal de referência (pH) foi mantido em $7{,}0$ e os distúrbios foram gerados aplicando aleatoriamente uma variação na vazão $Q1$ de ácido clorídrico (HCl) compreendida entre $-0{,}5ml/seg$ e $0{,}5ml/seg$ sobre sua vazão nominal de $2{,}74ml/seg$.

Todos os resultados dos testes expostos em seguida foram obtidos utilizando a plataforma de testes apresentada no Apêndice B.

- **Teste de rastreabilidade**

Os resultados dos testes de rastreabilidade realizados com os controladores são apresentados nas Figuras 5.18, 5.19 e 5.20 referente ao neuro-controlador preditivo com otimizador por gradiente descendente, controlador PID e neuro-controlador proposto, respectivamente.

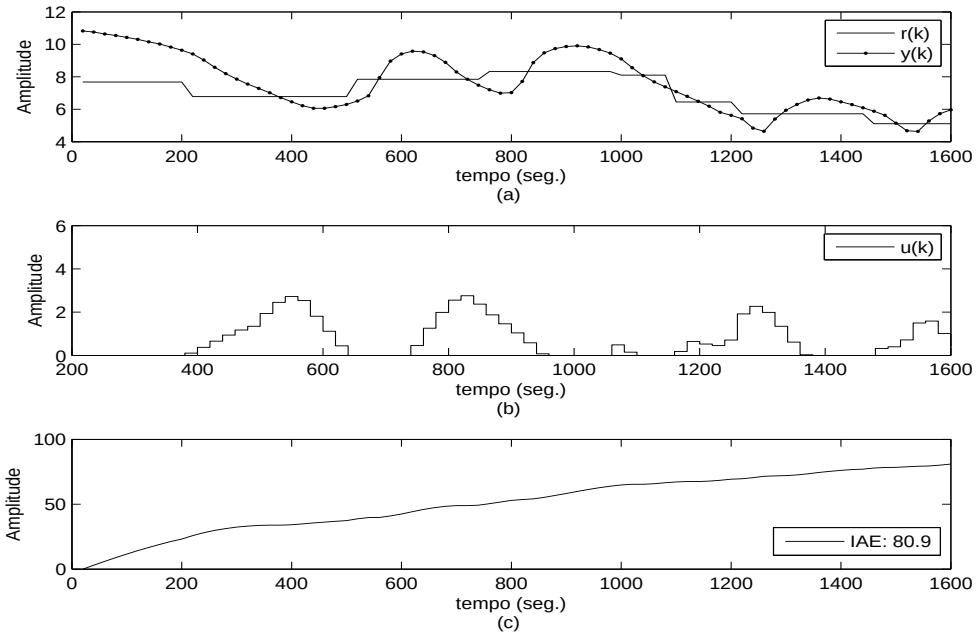

Figura 5.18: Processo de neutralização de pH: Resultado do teste de rastreabilidade do neuro-controlador preditivo com otimizador por gradiente descendente.

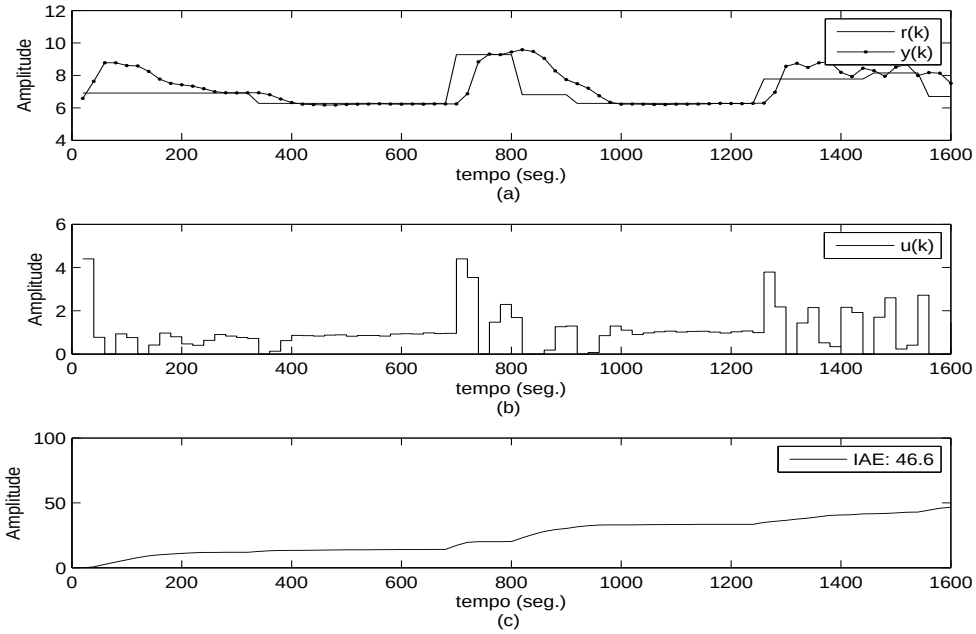

Figura 5.19: Processo de neutralização de pH: Resultado do teste de rastreabilidade do controlador PID.

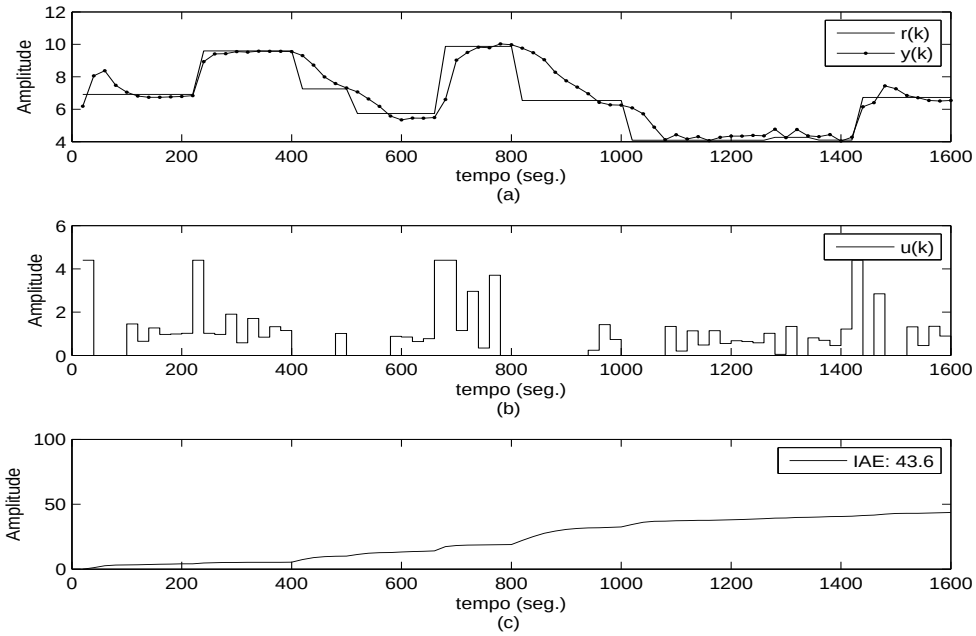

Figura 5.20: Processo de neutralização de pH: Resultado do teste de rastreabilidade do neuro-controlador proposto.

Nota-se claramente que o neuro-controlador com método otimização por gradiente descendente não consegue estabilizar o processo, isto porque suas ações de controle são alteradas de forma bem suave, ou seja, a amplitude entre as ações de controle é pequena.

Percebe-se também que o controlador PID mesmo tendo resultado próximo ao do neuro-controlador proposto, não é capaz de responder satisfatoriamente em toda faixa de operação, ficando mais evidente após 1200 segundos de operação (Figura 5.19), visto que o valor de referência fica em torno de pH 8 por um tempo considerável.

O neuro-controlador proposto, a cada iteração aplica ao processo uma ação de controle mais eficaz, pois percebe-se na Figura 5.20 que a amplitude entre as ações de controle atinge valores em toda faixa de operação.

- **Teste de rejeição de distúrbios**

Os resultados dos testes de rejeição de distúrbios realizados com os controladores são apresentados nas Figuras 5.21 e 5.22 referentes ao controlador PID e neuro-controlador proposto, respectivamente.

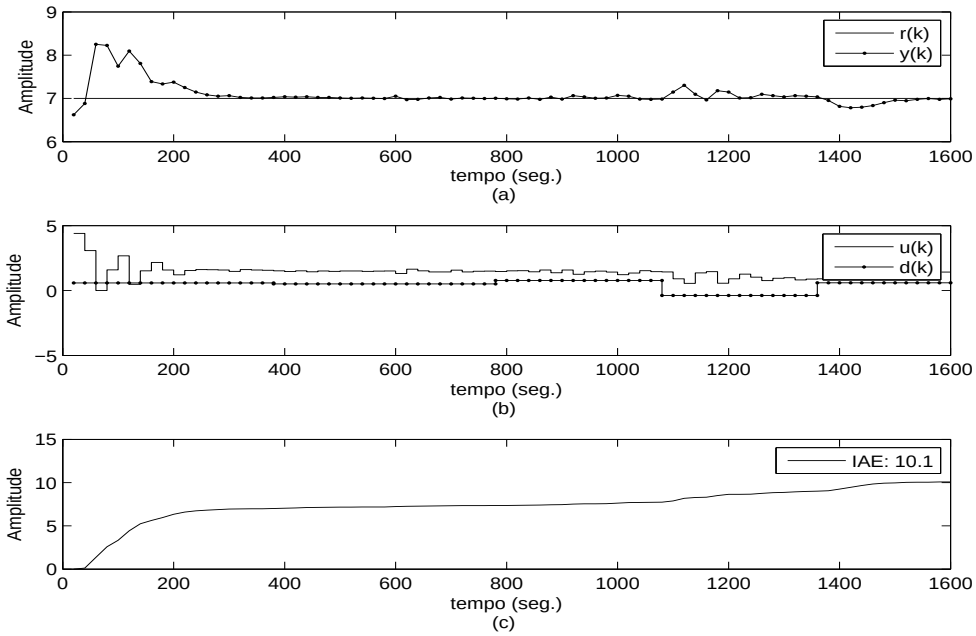

Figura 5.21: Processo de neutralização de pH: Resultado do teste de rejeição de distúrbios do controlador PID.

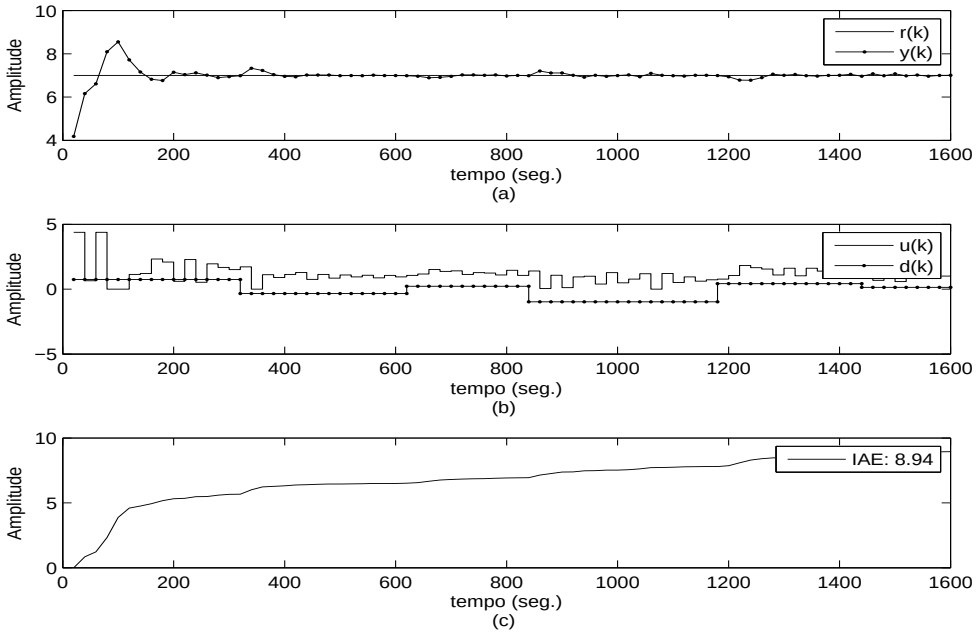

Figura 5.22: Processo de neutralização de pH: Resultado do teste de rejeição de distúrbios do neuro-controlador proposto.

Em virtude do baixo desempenho do neuro-controlador com otimizador por gradiente descendente nos teste de rastreabilidade, não foi realizado teste rejeição de distúrbios para o mesmo.

Percebe-se nas Figuras 5.21 e 5.22 que o neuro-controlador proposto possui uma melhor resposta na rejeição os distúrbios, impostos pela variação de $Q1$ de ácido clorídrico (HCl), comparado ao PID.

Isso deve-se ao fato da diferença entre a saída do modelo (RNA) e do processo tender a aumentar na presença de distúrbios e, como o resíduo é usado no bloco de otimização do neuro-controlador proposto, a sua ação de controle já considera tal diferença, o que resulta em um melhor de desempenho.

No entanto, nota-se que o esforço de controle apresentado pelo controlador PID é menor, sendo essa uma boa característica para operação em processo em que seu regime dinâmico, exige uma taxa de variação da ação de controle menor.

- **Observações**

A Tabela 5.2 apresenta o Integral do Erro Absoluto (IAE) dos controladores obtido ao final

dos testes. Nota-se que o neuro-controlador proposto apresenta um desempenho superior aos demais, e que o neuro-controlador com otimizador por gradiente descendente não realizou controle sobre o sistema, semelhante aos resultados dos testes realizados em sistemas simulados.

Tabela 5.2: IAE (Integral do Erro Absoluto) dos controladores: Planta de neutralização de pH

Teste	Neuro-controlador preditivo com otimizador por gradiente descendente	Controlador PID	Neuro-controlador proposto
Rastreabilidade sem modelo de ref.	80,9	46,6	43,3
Rejeição de distúrbios sem modelo de ref.	-	10,1	8,94

5.4 Discussões Gerais - Sistema simulado e planta de neutralização de pH

Para a determinação dos parâmetros (sintonia) do controlador PID, em primeiro momento, tentou-se usar o método de sintonia λ (Dahlin, 1968; Åström e Hägglund, 2005). No entanto, devido ao alto grau de não-linearidade do processo, os parâmetros gerados por este método apresentaram desempenho insatisfatório - fora da faixa em que foi sintonizado, sendo que para a planta piloto esta faixa foi compreendida entre o pH 6 e 8. Assim, foi necessário realizar um ajuste empírico para que o mesmo não entrasse em instabilidade - fora da faixa de ajuste, que por conseguinte surtiu um efeito positivo. Como se trata de ajuste por "tentativa e erro", sendo um processo de neutralização de pH relativamente lento, foi desprendido um tempo considerável nesta sintonia.

Para o neuro-controlador preditivo com otimizador por gradiente descendente, conforme Schnitman e Fontes (1999), a determinação de seus parâmetros é empírica, o que demandou de um maior esforço, principalmente no processo neutralização de pH. Nota-se que para todos os casos, mesmo depois de encontrada a melhor sintonia para este neuro-controlador, o mesmo apresentou o maior IAE entre os três controladores testados.

A sintonia do neuro-controlador proposto foi definida utilizando a equação que determina o tamanho do intervalo de busca após k iterações do método de otimização por eliminação de regiões com busca por seção áurea. Edgar e Himmelblau (1988) definem, para este método de busca, que a tolerância é dada por $L_k = (0{,}618)^k \, L_0$, sendo L_0 a região de busca inicial. Desta

forma, independente do processo a ser controlado, assumindo um valor de tolerância têm-se o número máximo de iterações. A tolerância determina a precisão da ação de controle, que nos testes, 1% da região de busca foi suficiente para que o neuro-controlador proposto apresentasse o menor IAE no teste de rastreabilidade entre o controladores testados, mesmo sujeito a uma maior variabilidade de amplitude do sinal de referência em comparação com os demais. No teste de rejeição de distúrbios na planta de neutralização de pH, o resultado encontrado é bastante satisfatório, pois submetido a uma variação de aproximadamente 40% na vazão Q_1, solução de ácido clorídrico (HCl), o neuro controlador proposto estabiliza o processo rapidamente.

Nota-se tanto no neuro-controlador proposto quanto no controlador PID que a mudança de patamar da ação de controle $u(k)$ ocorre de forma mais brusca e veloz, enquanto no neuro-controlador com método de otimização por gradiente descendente é mas suave e lenta. Percebe-se também que o controlador PID mesmo tendo resultado próximo ao do neuro-controlador proposto, não é capaz de responder satisfatoriamente em toda faixa de operação, como pode ser visto na Figura 5.20.

5.5 Comentários Finais

Objetivando colocar em prova o desempenho do neuro-controlador proposto, foram apresentados neste capítulo os resultados dos testes realizados em processo simulado e real (planta piloto de neutralização de pH) em cenários de rastreabilidade e rejeição de distúrbios. Também uma comparação do resultado do neuro-controlador proposto com controlador PID e neuro-controlador preditivo com otimização por gradiente descendente.

Na Seção 5.2 foi exposto o modelo de um sistema não-linear variante no tempo usado como processo simulado a ser controlado. Foi também apresentado o mecanismo de obtenção da RNA, para uso nos neuro-controladores, seguido de testes com os controladores sem modelo de referência e com modelo de referência em cenários de rastreabilidade e rejeição de distúrbios.

Os resultados dos testes dos controladores aplicados em um sistema real, planta piloto de neutralização de pH, foram apresentados na Seção 5.3. São destacados nesta seção as características do processo, a obtenção da RNA para uso nos neuro-controladores, e os resultados alcançados em cenários de rastreabilidae e rejeição de distúrbios.

Na Seção 5.4 foi exposta uma avaliação geral dos controladores testados, mostrando as questões relacionadas às suas parametrizações e o comportamento específico de geração de suas ações de controle.

Capítulo 6

Conclusões

Na busca de soluções para controle de sistemas dinâmicos não-lineares ou processos onde há necessidades operacionais complexas, dinâmicas e com maior gama de funcionamento, foi explorada neste trabalho a utilização de neuro-controladores em sistema dinâmicos. A qual veio a culminar na proposta de um neuro-controlador que utiliza o método de otimização por eliminação de regiões e busca por seção áurea.

O neuro-controlador proposto possui uma particularidade incomum: incluir na função de custo, o resíduo. Objetivando assim, minimizar a influência do erro de modelagem, erro estacionário e aumentar a capacidade de rejeição de distúrbios. Além de possuir a capacidade de otimizar a ação de controle para que a mesma leve o processo a se comportar conforme desejado, em um menor tempo possível.

Desta forma, o neuro-controlador proposto apresenta uma característica de grande valia: se necessário, muda o valor da ação de controle de um patamar para outro bruscamente entre as iterações. Isto causa um aumento de desempenho quando aplicado em processo com alto grau de não linearidade, onde fica evidenciado tal necessidade de comportamento. No entanto, em processos onde a mudança de patamar da ação de controle deve ser suave, algumas restrições devem ser adicionadas, como por exemplo, a limitação da variação da ação de controle entre cada iteração do algoritmo do controlador.

Outra característica marcante do neuro-controlador proposto é a simplicidade para determinar seus parâmetros, comparado ao neuro-controlador com otimização por gradiente descendente. Pois, elimina a necessidade do λ (tamanho do passo), visto que seus parâmetros independem do tipo de processo a ser controlado. No entanto, o modelo do processo a ser extraido (RNA), assim como em todos os controladores baseado em modelo, tem a identificação

do sistema como o ponto chave.

Foi possível notar durante os testes, que o custo computacional dos algoritmos dos controladores são diferentes, sendo que o neuro-controlador proposto é o que possui o maior custo dentre os controladores testados. Isto devido a necessidade de simular o processo, por meio da RNA, a cada iteração do algoritmo de busca por seção áurea no bloco de otimização.

Apesar do método de otimização adotado no neuro-controlador proposto estar sujeito a mínimos locais, somente quando este ocorrer, o resultado da ação de controle será menos eficiente. No entanto, os resultados apresentados no Capítulo 5 confirmam o potencial do neuro-controlador proposto, tanto para os cenários de rastreabilidades quanto para o de rejeição de distúrbios.

6.1 Sugestões para trabalhos futuros

No decorrer deste trabalho alguns testes não puderam ser realizados, bem como outras idéias surgiram. Assim, deixa-se registrado algumas sugestões para trabalhos futuros:

- Realização de testes de rejeição de distúrbios na planta piloto de neutralização de pH com neuro-controlador em outras níveis de pH;
- Realização de testes na planta piloto de neutralização de pH com neuro-controlador com modelo de referência;
- Aplicação do neuro-controlador proposto em sistemas multi-variáveis;
- Adequação do neuro-controlador proposto para adaptação *on-line*;
- Utilização de outra representação matemática como modelo interno do neuro-controlador proposto;
- Utilização de outros métodos de otimização no neuro-controlador proposto.

Referências Bibliográficas

Agarwal, M. (1997). A systematic classification of neural-network based control. *IEEE Control Systems Magazine*, 17(2):75–93.

Aguirre, L. A. (2004). *Introdução à identificação de Sistemas: técnicas lineares e nãolineares aplicadas a sistemas reais*. Editora UFMG, Belo Horizonte - MG, 2º edição edição.

Alvarez, H., London, C., Sciascio, F., e Carelli, R. (2001). ph neutralization process as a benchmark for testing nonlinear controllers. *Ind. Eng. Chem. Res.*, 40(11):2467–2473.

Ansari, R. M. e Tadé, M. O. (2001). Non-linear model based process control: Applications in petroleum refining. *Automatica*, 37(10):1679–1681.

Assis, J. C. d. O. (2007). Controle preditivo baseado em redes neurais artificiais. Monografia, Centro Universitário do Leste de Minas Gerais, Coronel Fabriciano - MG.

Åström, K. J. e Hägglund, T. (2005). *Advanced PID Control*. ISA - The Instrumentation, Systems, and Automation Society, Research Triangle Park, NC 27709.

Bazaraa, M. S., Sherali, H. D., e Shetty, C. M. (1993). *Nonlinear Programming Theory and Algorithms*. John Wiley, New York, 2^a edição.

Boukabou, A. e Mansouri, N. (2005). Neural predictive control of unknown chaotic systems. *Nonlinear Analysis: Modelling and Control*, 10(2):95–106.

Braga, A. P., Ludermir, T. B., e Carvalho, A. C. (2000). *Redes Neurais: Teoria e Aplicações*. LTC - Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, RJ, 1^a edição.

Callender, A. e Stevenson, A. B. (1939). Automatic control of variable physical characteristics. U.S. patent 2,175,985. Filed February 17, 1936 in the United States. Filed February 13, 1935 in Great Britain. Issued October 10, 1939 in the United States.

Campos, R. C. C. (Maio de 2007). Projeto e construção de planta piloto de neutralização de ph e proposta de metodologia para incorporação de informações auxiliares na identificação narx racional. Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia - Centro Universitário do Leste de Minas Gerais, Coronel Fabriciano - MG.

Chauvin, Y. A. (1989). A backpropagation algorithm with optimal use of hidden units. Em Touretzky, D. S., editor, *Advances in neural information processing systems 1*, volume 1, pp. 519–526. Morgan Kaufmann Publishers Inc., San Mateo, CA.

Chen, S., Cowan, C. F. N., Billings, S. A., e Grant, P. M. (1990). Non-linear system identification using neural networks. *International Journal of Control*, 51:1191–1214.

Constantin, N. (2003). Adaptive neural predictive techniques for nonlinear control. *Studies in informatics and control*, 12(4):285–292.

Corliss, G. (1977). Which root does the bisection algorithm find? *SIAM Review*, 19(2):325–327.

Cybenko, G. (1989). Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

Dahlin, E. (1968). Designing and tuning digital controllers. *Instruments and Control Systems*, 41(6):77–83.

Daosud, W., Thitiyasook, P., Arpornwichanop, A., Kittisupakorn, P., e Hussain, M. A. (2005). Neural network inverse model-based controller for the control of a steel pickling process. *Computers & Chemical Engineering*, 29(10):2110–2119.

Dias, F. M. e Mota, A. A. (2000). Direct inverse control of a kiln. *Portuguese Conference on Automatic Control*, 4:336–341.

Doherty, S. K. (2000). *Control of pH in Chemical Processes using Artificial Neural Networks*. Tese de Doutorado, School of Engineering, Liverpool John Moores University, UK.

Doherty, S. K., Gomm, J. B., e Williams, D. (1995). Neural network identification and predictive control of an in-line ph process. *4th I.Chem.E. Conf. on Advances in Process Control*, 4:57–64.

Douratsos, I. e Gomm, J. B. (2007). Neural network based model reference adaptive control for processes with time delay. *International Journal of Information and Systems Sciences*, 3(1):161–179.

Edgar, T. F. e Himmelblau, D. M. (1988). *Optimization of Chemical Processes*. Mc Graw Hill Book Company, New York, NY, 1^a edição.

Fukuda, T. e Shibata, T. (1992). Theory and applications of neural networks for industrial control systems. *IEEE Transaction Industrial Electronics*, 39(6):472–489.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Netw.*, 2(3):183–192.

Ge, S., Hong, F., e Lee, T. H. (2004). Adaptive neural control of nonlinear time-delay systems with unknown virtual control coefficients. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(1):499– 516.

Hanson, S. e Pratt, L. Y. (1989). Comparing biases for minimal network construction with backpropagation. Em Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, volume 1, pp. 177–185. Morgan Kaufmann Publishers Inc., San Mateo, CA.

Hassibi, B., Stork, D. G., e Wolff, G. J. (1993). Optimal brain surgeon and general network pruning. Em *Proceedings of the 1993 IEEE International Conference on Neural Networks*, pp. 293–299.

Haykin, S. (1999). *Neural Networks. A Comprehensive Foundation*. Prentice Hall, New Jersey, 2^a edição.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234.

Hunt, K. J., Sbarbaro, D., Żbikowski, R., e Gawthrop, P. J. (1992). Neural networks for control systems: a survey. *Automatica*, 28(6):1083–1112.

Isidori, A. (2000). *Nonlinear Control Systems II*. Springer-Verlag, London, UK.

Iyoda, E. M. (2000). Inteligência computacional no projeto automático de redes neurais híbridas e redes neurofuzzy heterogêneas. Dissertação de Mestrado, Programa de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação - Universidade Estadual de Campinas.

Jones, D. M., Watton, J., e Brown, K. J. (2005). Comparison of hot rolled steel mechanical property prediction models using linear multiple regression, non-linear multiple regression and non-linear artifical neural networks. *Ironmaking and Steelmaking*, 32(5):435–442.

Kasparian, V. e Batur, C. (1998). Model reference based neural network adaptive controller. *ISA Transactions 37*, 37(1):21–39.

Khalil, H. K. (2002). *Nonlinear Systems*. Upper Saddle River, New Jersey, 3rd edição.

LeCun, Y., Denker, J. S., e Solla, S. A. (1990). Optimal brain damage. Em Touretzky, D. S., editor, *Advances in Neural Information Processing Systems II*, pp. 598–605, San Mateo, CA. Morgan Kauffman.

Lee, T. C., Yang, D. R., Lee, K. S., e Yoon, T. W. (2001). Indirect adaptive backstepping control of a ph neutralization process based on recursive prediction error method for combined state and parameter estimation. *Industrial & Engineering Chemistry Research*, 40(19):4102–4110.

Lehninger, A. L., Nelson, D. L., e Cox, M. M. (1995). *Princípios de Bioquímica*. Sarvier, São Paulo, SP, 2^a edição.

Liu, G. P., Kadirkamanathan, V., e Billings, S. A. (1999). Variable neural networks for adaptive control of nonlinear systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, 29(1):34–43.

Luenberger, D. G. (1984). *Linear and Nonlinear Programming*. Addison-Wesley, 2^a edição.

Marzzoco, A. e Torres, B. B. (2007). *Bioquímica Básica*. Guanabara Koogan S. A., Rio de Janeiro, RJ, 3^a edição.

Miller, W. T. I., Sutton, R. S., e Werbos, P. j., editors (1990). *Neural networks for control*. MIT Press, Cambridge, MA, USA.

Nahas, E. P., Henson, M., e Seborg, D. E. (1992). Nonlinear internal model control strategy for neural network models. *Computers Chem. Engng.*, 16:1039–1057.

Narendra, K. S. e Mukhopadhyay, S. (1994). Adaptive control of nonlinear multivariable systems using neural networks. *Neural Networks*, 7(5):737–752.

Ogata, K. (1997). *Modern control engineering (3rd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Patiño, H. D., Carelli, R., e Kuchen, B. R. (2002). Neural networks for advanced control of robot manipulators. *IEEE Transactions on Neural Networks*, 13(2):343–354.

Pozas, L. F. (2005). *Controladores Preditivos Não Lineares para Processos de Refino*. Tese de Doutorado, Programa de pós-graduação em engenharia elétrica - Universidade Federal de Santa Catarina, Florianópolis - SC.

Psaltis, D., Sideris, A., e Yamamura, A. A. (1988). A multilayered neural network controller. *IEEE Control Systems Magazine*, 8(2):17–21.

Qin, S. e Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764.

Reklaitis, G. V., Ravindran, A., e Ragsdell, K. M. (1983). *Engineering Optimization: Methods and Applications*. Jonh Wiley & Sons.

Riedmiller, M. (1994). Aspects of learning neural control. *IEEE International Conference on Systems, Man, and Cybernetics*, 2:1458–1463.

Riedmiller, M. (1999). Concept and facilities of a neural reinforcement learning control architecture for technical process control. *Neural Computing and Aplication Jornal*, 8(4):323–338.

Rivals, I. e Personnaz, L. (1996). Internal model control using neural networks. *Proceedings of the IEEE Intenational Symposium on Industrial Electronics*, 1:109–114.

Rivals, I. e Personnaz, L. (2000). Nonlinear internal model control using neural networks: Application to processes with delay and design issues. *IEEE Transactions on Neural Networks*, 11(1):80–90.

Rumelhart, D. E., Hinton, G. E., e Willians, R. J. (1986). Learning representations by back-propagation errors. *Nature*, 323:533–536.

Sanner, R. e Slotine, J.-J. (1995). Stable adaptive control of robot manipulators using neural networks. *Neural Computation*, 7(4):753–790.

Schnitman, L. e Fontes, A. (1999). The basic ideas of neural predictive control. Em *Proceedings of the 7th Mediterranean Conference on Control and Automation (MED99)*, pp. 497–510.

Schwarzenbach, J. e Gill, K. (1992). *System Modelling and Control*. Halsted Press, New York, NY, USA, 3rd edição.

Silva, L. N. C. (1998). Análise e síntese de estratégias de aprendizado para redes neurais artificiais. Dissertação de Mestrado, Programa de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação - Universidade Estadual de Campinas.

Syafiie, S., Tadeo, F., e Martinez, E. (2005). Model-free intelligent control using reinforcement learning and temporal abstraction-applied to ph control. Em 16^{th} *IFAC World Control Congress*, Prague, Czech.

Tan, Y. e Cauwenberghe, A. (1996). Nonlinear one-step-ahead control using neural networks: control strategy and stability design. *Automatica*, 32(12):1701–1706.

Te Braake, H. A. B., Van Can, E. J. L., Scherpen, J. M. A., e Verbruggen, H. B. (1998). Control of nonlinear chemical processes using neural models and feedback linearization. *Computers & chemical engineering*, 22(7-8):1113–1127.

Teixeira, R. A. (2001). *Treinamento de Redes Neurais Artificais Através de Otimização Multi-objetivo: Uma Abordagem para o Equilíbrio entre a Polarização e a Variância*. Tese de Doutorado, Programa de Pós-Graduação em Engenharia Elétrica - Universidade Federal de Minas Gerais.

Žilková, J., Timko, J., e Girovský, P. (2006). Nonlinear system control using neural networks. *Acta Polytechnica Hungarica*, 3(4):85–94.

Warwick, K., Zhu, Q. M., e Ma, Z. (2000). A hyperstable neural network for the modelling and control of nonlinear systems. *Sadhana*, 25(2):169–180.

Weigend, A. S., Rumelhart, D. E., e Hubermam, B. A. (1990a). Generalization by weight-elimination with application to forecasting. Em *NIPS-3: Proceedings of the 1990 conference on Advances in neural information processing systems 3*, pp. 875–882, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Weigend, A. S., Rumelhart, D. E., e Hubermam, B. A. (1990b). Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 3(1):193–209.

Widrow, B. e Hoff, M. E. (1960). Adaptive switching circuits. Em *IRE WESCON Convention Record*, pp. 96–104.

Wilson, H. S. e Wylupek, W. J. (1969). Design of ph control systems. *Measurement and Control*, 2:336–342.

Ying, S., Zengqiang, C., e Zhuzhi, Y. (2007). Neural network nonlinear predictive control based on tent-map chaos optimization. *Chin. J. Chem. Eng.*, 15(4):539–544.

Ylén, J. P. (2001). *Measuring, modelling and controlling the pH value and the dynamic chemical state*. Tese de doutorado, Helsinki University of Technology, Otakaari 1, FI-02150 Espoo, Finland.

Yoo, A., Lee, T. C., e Yang, D. R. (2004). Experimental simultaneous state and parameter identification of a ph neutralization process based on an extended kalman filter. *Korean Journal of Chemical Engineering*, 21(4):753–760.

Apêndice A

Algoritmos

Algoritmo 1 Busca por seção áurea

$a \Leftarrow$ Início da região de busca
$b \Leftarrow$ Fim da região de busca
$k_{max} \Leftarrow$ Número máximo de iterações
$tol \Leftarrow$ Tolerância
$c \Leftarrow \frac{3-\sqrt{5}}{2}$
$d \Leftarrow 1 - \frac{3-\sqrt{5}}{2}$
$u_1 \Leftarrow a + c(b-a)$
$u_2 \Leftarrow a + d(b-a)$
$J_{u_1} \Leftarrow J(u_1)$
$J_{u_2} \Leftarrow J(u_2)$
$k \Leftarrow 0$
Enquanto $(|b-a| > tol)$ e $(k < k_{max})$ **Faça**
  Se $J_{u_1} > J_{u_1}$ **Então**
    $a = u_1$
    $u_1 = u_2$
    $J_{u_1} = J_{u_2}$
    $u_2 = a + d(b-a)$
    $J_{u_2} = J(u_2)$
  Senão
    $b = u_2$
    $u_2 = u_1$
    $J_{u_2} = J_{u_1}$
    $u_1 = a + c(b-a)$
    $J_{u_1} = J(u_1)$
  Fim Se
  $k = k + 1$;
Fim Enquanto

Algoritmo 2 Neuro-controlador proposto

$a \Leftarrow$ Limite inferior da região de busca para $u(k)$
$b \Leftarrow$ Limite superior da região de busca para $u(k)$
$k_{max} \Leftarrow$ Número máximo de iterações para busca de $u(k)$
$tol \Leftarrow$ Tolerância (região final que contenha $u(k)$ ótimo)
$c = \frac{3-\sqrt{5}}{2}$
$d = 1 - \frac{3-\sqrt{5}}{2}$
$k = 0$
Enquanto 1 **Faça**
  $k = k + 1$
  $r(k+1) \Leftarrow$ Comportamento desejado (Referência)
  $y(k) \Leftarrow$ Estado do processo (leitura do valor atual)
  $\mathbf{x} = [y(k-1),y(k-2),...,y(k-(n+1)),u(k-1),u(k-2),...,u(k-(m+1))]$
  $\hat{y}(k) = \mathcal{F}(\mathbf{x})$
  $\xi(k) = y(k) - \hat{y}(k)$
  $u_1(k) = a + c(b-a)$
  $u_2(k) = a + d(b-a)$
  $\mathbf{x_1} = [y(k), y(k-1),..., y(k-n), u_1(k), u(k-1),..., u(k-m)]$
  $\mathbf{x_2} = [y(k), y(k-1),..., y(k-n), u_2(k), u(k-1),..., u(k-m)]$
  $\hat{y}_1(k+1) = \mathcal{F}(\mathbf{x_1})$
  $\hat{y}_2(k+1) = \mathcal{F}(\mathbf{x_2})$
  $J_1 =\mid r(k+1) - [\hat{y}_1(k+1) + \xi(k)] \mid$
  $J_2 =\mid r(k+1) - [\hat{y}_2(k+1) + \xi(k)] \mid$
  $i = 0$
  Enquanto $(|b-a| > tol)$ e $(i < k_{max})$ **Faça**
    Se $J_2 > J_1$ **Então**
      $a = u_1(k)$
      $u_1(k) = u_2(k)$
      $\hat{e}_{c1} = \hat{e}_{c2}$
      $u_2(k) = a + d(b-a)$
      $\mathbf{x_2} = [y(k), y(k-1),..., y(k-n), u_2(k), u(k-1),..., u(k-m)]$
      $\hat{y}_2(k+1) = \mathcal{F}(\mathbf{x_2})$
      $J_2 =\mid r(k+1) - [\hat{y}_2(k+1) + \xi(k)] \mid$
    Senão
      $b = u_2(k)$
      $u_2(k) = u_1(k)$
      $\hat{e}_{c2} = \hat{e}_{c1}$
      $u_1(k) = a + c(b-a)$
      $\mathbf{x_1} = [y(k), y(k-1),..., y(k-n), u_1(k), u(k-1),..., u(k-m)]$
      $\hat{y}_1(k+1) = \mathcal{F}(\mathbf{x_1})$
      $J_1 =\mid r(k+1) - [\hat{y}_1(k+1) + \xi(k)] \mid$
    Fim Se
    $i = i + 1$;
  Fim Enquanto
  $u(k) = a$
  Processo $\Leftarrow u(k)$
Fim Enquanto

Algoritmo 3 Neuro-controlador proposto com modelo de referência

$a \Leftarrow$ Limite inferior da região de busca para $u(k)$
$b \Leftarrow$ Limite superior da região de busca para $u(k)$
$k_{max} \Leftarrow$ Número máximo de iterações para busca de $u(k)$
$tol \Leftarrow$ Tolerância (região final que contenha $u(k)$ ótimo)
$c = \frac{3-\sqrt{5}}{2}$
$d = 1 - \frac{3-\sqrt{5}}{2}$
$k = 0$
Enquanto 1 **Faça**
  $k = k + 1$
  $r(k+1) \Leftarrow$ Comportamento desejado (Referência)
  $y(k) \Leftarrow$ Estado do processo (leitura do valor atual)
  $r_m(k+1) = f_m(r(k+1))$
  $\mathbf{x} = [y(k-1),y(k-2),...,y(k-(n+1)),u(k-1),u(k-2),...,u(k-(m+1))]$
  $\hat{y}(k) = \mathcal{F}(\mathbf{x})$
  $\xi(k) = y(k) - \hat{y}(k)$
  $u_1(k) = a + c(b-a)$
  $u_2(k) = a + d(b-a)$
  $\mathbf{x_1} = [y(k), y(k-1),..., y(k-n), u_1(k), u(k-1),..., u(k-m)]$
  $\mathbf{x_2} = [y(k), y(k-1),..., y(k-n), u_2(k), u(k-1),..., u(k-m)]$
  $\hat{y}_1(k+1) = \mathcal{F}(\mathbf{x_1})$
  $\hat{y}_2(k+1) = \mathcal{F}(\mathbf{x_2})$
  $J_1 =\mid r_m(k+1) - [\hat{y}_1(k+1) + \xi(k)] \mid$
  $J_2 =\mid r_m(k+1) - [\hat{y}_2(k+1) + \xi(k)] \mid$
  $i = 0$
  Enquanto $(|b-a| > tol)$ e $(i < k_{max})$ **Faça**
    Se $J_2 > J_1$ **Então**
      $a = u_1(k)$
      $u_1(k) = u_2(k)$
      $\hat{e}_{c1} = \hat{e}_{c2}$
      $u_2(k) = a + d(b-a)$
      $\mathbf{x_2} = [y(k), y(k-1),..., y(k-n), u_2(k), u(k-1),..., u(k-m)]$
      $\hat{y}_2(k+1) = \mathcal{F}(\mathbf{x_2})$
      $J_2 =\mid r_m(k+1) - [\hat{y}_2(k+1) + \xi(k)] \mid$
    Senão
      $b = u_2(k)$
      $u_2(k) = u_1(k)$
      $\hat{e}_{c2} = \hat{e}_{c1}$
      $u_1(k) = a + c(b-a)$
      $\mathbf{x_1} = [y(k), y(k-1),..., y(k-n), u_1(k), u(k-1),..., u(k-m)]$
      $\hat{y}_1(k+1) = \mathcal{F}(\mathbf{x_1})$
      $J_1 =\mid r_m(k+1) - [\hat{y}_1(k+1) + \xi(k)] \mid$
    Fim Se
    $i = i + 1$;
  Fim Enquanto
  $u(k) = a$
  Processo $\Leftarrow u(k)$
Fim Enquanto

Apêndice B

Plataforma de Testes

B.1 Introdução

Para a realização dos testes dos controladores, foi desenvolvido uma ferramenta chamada de plataforma de testes a fim de facilitar a realização dos mesmos. Sendo possível a alteração dos seguintes ítens:

- dos processos (planta);
- dos controladores;
- dos modelos de referência;
- da amplitude do sinal de referência;
- da alternância do sinal de referência;
- da inserção de ruídos;
- da inserção de distúrbios;
- dos parâmetros temporais da simulação;
- da criação e treinamento da RNA;
- do salvamento do conjunto de dados da simulação;
- da parametrização da comunicação com a planta de neutralização de pH.

A Figura B.1 apresenta a plataforma de testes desenvolvida em MatLab®.

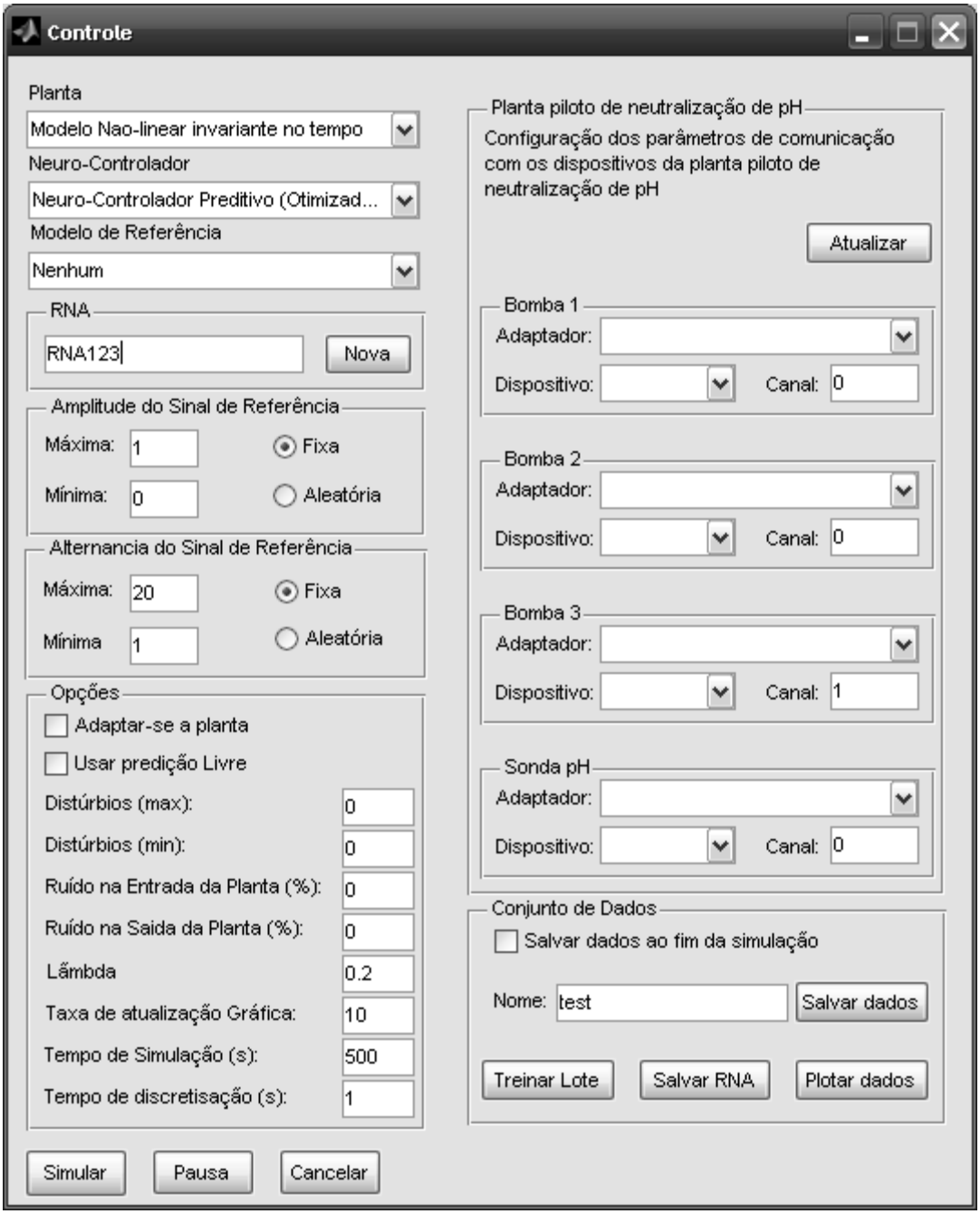

Figura B.1: Plataforma de teste.

B.2 Descrição dos elementos da plataforma de testes

B.2.1 Caixa de seleção: Planta

A caixa de seleção Planta, tem por finalidade selecionar o tipo do processo (planta) a ser controlado. Possui as seguintes opções:

- Modelo linear de primeira ordem:

$$y(k) = 0.7869u(k-1) + 0.6065y(k-1); \tag{B.1}$$

- Modelo linear de segunda ordem:

$$y(k) = u(k-1) + 0.5u(k-2) + 1.5y(k-1) - 0.7y(k-2); \tag{B.2}$$

- Modelo não-linear invariante no tempo:

$$\begin{aligned} y(k) \quad = \quad & \left[0{,}8 - 0{,}5e^{-y^2(k-1)}\right] y(k-1) - \left[0{,}3 + 0{,}9e^{-y^2(k-1)}\right] y(k-2) + \\ & u(k-1) + 0{,}2u(k-2) + 0{,}1u(k-1)u(k-2); \end{aligned} \tag{B.3}$$

- Planta piloto: estabelece conexão via porta USB com a planta piloto de neutralização de pH usando as configurações presentes na caixa **Planta piloto de neutralização de** pH.

B.2.2 Caixa de seleção: Neuro-Controlador

A caixa de seleção Neuro-Controlador, tem por finalidade selecionar o tipo de controlador a ser aplicado no processo. Possui as seguintes opções:

- Malha aberta;
- Neuro-controlador preditivo (otimização por gradiente descendente);
- Neuro-controlador proposto;
- Controlador PID.

B.2.3 Caixa de seleção: Modelo de referência

A caixa de seleção Modelo de referência, tem por finalidade selecionar o tipo de Modelo de referência a ser aplicado ao controlador. Possui as seguintes opções:

- Nenhum;
- Modelo de Primeira ordem lento:

$$r_m(k+1) = 0.2r(k+1) + 0.8r_m(k); \tag{B.4}$$

- Modelo de Primeira ordem rápido:

$$r_m(k+1) = 0.6r(k+1) + 0.4r_m(k); \tag{B.5}$$

B.2.4 Caixa: RNA

A caixa RNA, possui as informações referente a RNA. Nela contém uma caixa de texto com o nome da RNA usada no neuro-controlador e o botão **"Nova"**. Este botão abre uma interface para criação de uma nova RNA.

B.2.5 Caixa: Amplitude do sinal de referência

A caixa Amplitude do sinal de referência, contém as caixas de texto com os limites **Máximo** e **Mínimo** da amplitude do sinal de referência e as opções de alternância entre os limites, sendo:

- **fixa:** alteração do sinal de referência somente com valores do limite máximo e mínimo; e
- **aleatória:** alteração do sinal de referência com valores entre o valores do limite máximo e mínimo.

B.2.6 Caixa: Alternância do sinal de referência

A caixa Alternância do sinal de referência, contém as caixas de texto com os limites **Máximo** e **Mínimo** do tempo Alternância do sinal de referência e as opções de alternância entre os limites, sendo:

- **fixa:** tempo de alternância fixa a cada tempo presente na caixa de texto **Máxima**; e
- **aleatória:** tempo de alternância aleatória entre o tempo presente entre a caixa de valores do limite máximo e mínimo.

B.2.7 Caixa: Opções

A caixa Opções, permite selecionar as seguintes opções da simulação:

- **Adaptar-se a planta:** opção não disponível (futuro);

- **Usar predição livre:** habilita a predição livre para a RNA;
- **Distúrbios(máx):** caixa de texto que contém o valor máximo para inserção de distúrbios aleatórios entre este e o valor mínimo;
- **Distúrbios(mím):** caixa de texto que contém o valor mínimo para inserção de distúrbios aleatórios entre este e o valor máximo;
- **Ruído na entrada da planta (%):** caixa de texto que contém o valor percentual de ruído a ser aplicado na entrada da planta;
- **Ruído na saída da planta (%):** caixa de texto que contém o valor percentual de ruído a ser aplicado na saída da planta;
- **Lãmbda:** caixa de texto que contém o valor de λ para o neuro-controlador com otimizador com gradiente descendente;
- **Taxa de atualização gráfica:** caixa de texto que contém o intervalo de iterações em que o gráfico será atualizado;
- **Tempo de Simulação (s):** caixa de texto que contém o tempo de simulação do experimento em segundos;
- **Tempo de discretização (s):** caixa de texto que contém o tempo de discretização a ser aplicado no processo em segundos.

B.2.8 Caixa: Planta piloto de neutralização de pH

A caixa Planta piloto de neutralização de pH, possui as configurações para estabelecer a conexão via porta USB com a planta piloto de neutralização de pH, por meio da placa de aquisição de dados NI USB-6008. Conforme citado na seção 5.3, as bombas e a sonda de pH estão conectadas a placa de aquisição de dados. Assim cada um destes elementos é identificado pela plataforma de testes obedecendo o seguinte critério:

- **Adaptador:** Caixa de seleção que contém o nome do *driver* que estabelece comunicação com o dispositivo;
- **Dispositivo:** Caixa de seleção que contém o nome do dispositivo selecionado na caixa de seleção: **Adaptador**;
- **Canal:** Caixa de texto que contém o nome do canal do dispositivo selecionado na caixa de seleção **Dispositivo**;
- **Botão Atualizar:** Este botão atualiza a lista de *drivers* da caixa de seleção **Adaptadores** de todas as bombas e sonda de pH.

B.2.9 Caixa: Conjunto de dados

A caixa Conjunto de dados, possui as ferramentas para manipulação da massa de dados e RNA. A opção **Salvar dados ao fim da simulação** determina se os dados serão salvos automaticamente no final da simulação. A caixa de texto **Nome** define o nome da massa de dados a ser salva ou aberta para a simulação.

- **Botão Salvar dados:** Este botão realiza o salvamento da massa de dados a qualquer momento da simulação;
- **Botão Treinar Lote:** Este botão realiza o treinamento em lote da RNA localizada na caixa de texto **RNA** com a massa de dados presente na caixa **Nome** e com o número de época determinada na caixa de texto **Tempo de Simulação (s)** com uma taxa de atualização gráfica determinada na caixa de texto **Taxa de atualização gráfica**;
- **Botão Salvar RNA:** Este botão salva a RNA localizada na caixa de texto **RNA**;
- **Botão Plotar dados:** Este botão plota a massa de dados presente na caixa **Nome**.

B.2.10 Botão: Simular

O botão Simular, inicia a simulação com todas as parametrizações da plataforma de teste.

B.2.11 Botão: Pausa

O botão Pausa, efetua uma pausa na simulação.

B.2.12 Botão: Cancelar

O botão Cancelar, cancela a simulação.

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