UNIVERSIDADE FEDERAL DO PARANÁ

ELINA SANDRA RAMOS DE LIRA E SILVA

INVESTIGAÇÃO DO COMPORTAMENTO DINÂMICO E AVALIAÇÃO DE ESTRATÉGIAS DE IDENTIFICAÇÃO, CONTROLE E OTIMIZAÇÃO DE UM REATOR FCC

CURITIBA

2006

ELINA SANDRA RAMOS DE LIRA E SILVA

INVESTIGAÇÃO DO COMPORTAMENTO DINÂMICO E AVALIAÇÃO DE ESTRATÉGIAS DE IDENTIFICAÇÃO, CONTROLE E OTIMIZAÇÃO DE UM REATOR FCC

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Engenharia Química.

Orientador: Prof. Dr. Luiz Augusto da Cruz Meleiro

CURITIBA

2006

Aos meus pais, Eliezer e Joseina, por seu amor, seu apoio total e incondicional, seu indispensável suporte físico e emocional.

Agradecimentos

A Deus, meu melhor amigo, por todas as oportunidades e bênçãos da minha vida.

Ao Prof. Dr. Luiz Augusto da Cruz Meleiro pela orientação, incentivo, apoio e dedicação em todas as etapas do desenvolvimento deste projeto, por sua confiança em mim, por tornar agradável um trabalho, por vezes cansativo, por seu sempiterno otimismo e bom humor, por dar sentido à palavra Mestre!

Aos Professores, Dr. Everton Fernando Zanoelo e Dr. Luiz Fernando de Lima Luz Júnior, por toda a ajuda, apoio, sugestões.

À Universidade Federal do Paraná e ao Programa de Pós-Graduação em Engenharia, PIPE, pela oportunidade de realizar este trabalho.

À minha família, com seu carinho e apoio sempre presentes, por suportarem minha eventuais indisposições de ânimo com a naturalidade dos íntimos. Amo vocês!

Aos amigos de pós-graduação, pelo apoio mútuo e pela amizade. A presença de vocês nestes dois anos de trabalho foi fundamental para mim.

À Bibiana, especialmente, pela disposição em discutir as dúvidas que surgiram no desenrolar deste último ano, por suas sugestões sempre pertinentes, por sua imprescindível ajuda na parte final deste trabalho.

A todos aqueles que embora não tenham sido mencionados contribuíram direta ou indiretamente para a realização deste trabalho.

“In questions of science, the authority of a thousand is not worth the humble reasoning of a single individual”

Galilei, Galileo (1564-1642)

Resumo

O objetivo desta dissertação é apresentar um estudo detalhado sobre a identificação de um modelo neural para um reator de craqueamento catalítico em leito fluidizado (FCCU) que possa ser implementado tanto na estrutura de um controlador preditivo quanto na otimização do processo usando métodos heurísticos estocásticos. Estratégias de controle baseadas em modelos não lineares podem prover significativa melhoria para processos não lineares quando comparadas com o desempenho dos controladores lineares. Redes neurais artificiais são usadas em uma grande variedade de aplicações da indústria química por causa de sua capacidade de representar as relações de entrada e saída de um sistema. Há na literatura vários modelos matemáticos para a FCCU, a maioria com abordagem fenomenológica e com descrição simplificada do processo. Baseado em um conjunto de dados gerado por um modelo mais elaborado, desenvolvido por Moro e Odloack (1995), validado com dados reais da unidade industrial da refinaria Alberto Pasqualini (REFAP) da PETROBRÁS, identificou-se a dinâmica do processo com um modelo neural mais simples e facilmente implementável em algoritmos de controle e otimização do que os modelos fenomenológicos rigorosos. Várias arquiteturas de rede foram testadas e são apresentados estudos sobre a influência da estrutura da rede na qualidade da resposta do modelo. Os resultados mostraram que o modelo neural obtido descreveu adequadamente o comportamento dinâmico do processo. Os resultados da otimização, obtidos através do algoritmo genético e do enxame de partículas, foram comparados entre si e com os resultados fornecidos pelo modelo fenomenológico do processo, apresentando ótima convergência. O

modelo neural foi implementado na estrutura de um controlador preditivo do processo apresentando resultados bastante satisfatórios.

Palavras-Chave: FCC, Identificação de Processos, Redes neurais, Controle Preditivo, Métodos Heurísticos de Otimização, Algoritmo Genético, Enxame de Partículas.

Abstract

The main objective of this work is to present a detailed study about the identification of a FCC unity using a neural model that can be used in the design of a predictive controller as well as in the process optimization using heuristic methods. Control strategies based on nonlinear models can potentially provide significant improvement in the operation of nonlinear process when compared to control algorithms based on linear methods. Neural networks are used for a wide variety of chemical process due to their ability to learn the systems' features from input-output process data. There are many mathematical models for the FCCU in the literature, most of them with phenomenological approach and very simplified process description. Based on more elaborated model, developed by Moro and Odloak (1995), validated with real data of Petrobras' Alberto Pasqualini Refinery (REFAP), the process dynamic has been identified with a neural model suitable to be implemented in control and optimization algorithms. Many architectures have been tested, and studies of their influence on the model accuracy are presented. The results show that the neural model gives a good description of the process dynamic behavior. Moreover, a comparison between the results provided by the genetic algorithm, particle swarm optimization, and the phenomenological model evidences a good agreement. The neural model was also used in the design of a model based predictive controller providing satisfactory results in closed-loop simulations.

Keywords: FCC, Identification, Neural Networks, Predictive Control, Process Optimization, Genetic Algorithm, Particle Swarm .

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

ARX	Modelo auto-regressivo com entradas externas (*AutoRegressive with eXogenous inputs*)
EQM	Erro Quadrático Médio
FCC	Craqueamento catalítico em leito fluidizado (*fluid catalytic cracking*)
AG	Algoritmo Genético
MIMO	Múltiplas entradas e múltiplas saídas (*multiple input*, *multiple output*)
MLP	Perceptron de múltiplas camadas (*Multi-layer perceptron*)
MPC	Controle preditivo baseado em modelo (*Model Predictive Control*)
MISO	Múltiplas entradas e uma saída (*multiple input*, *single output*)
RNA	Rede Neural Artificial
SISO	Uma entrada e uma saída (*single input*, *single output*)
SQP	Programação Quadrática Sucessiva (*Successivel Quadratic Programming*)

Lista de Símbolos

$A(\cdot)$	Matriz de polinômios
A_{LCV}	Abertura da válvula de controle de nível do reator
A_O	Área da seção transversal da tubulação a montante da válvula (m^2)
A_{ra}	Área da seção transversal do reator (m^2)
A_{rg1}	Área da seção transversal do primeiro estágio do regenerador (m^2)
A_s	Severidade da reação de craqueamento
A_{sc}	Severidade estimada da reação de craqueamento
A_V	Área da válvula de catalisador regenerado (depende da abertura) (m^2)
b_k	*Bias* do neurônio k
$B(\cdot)$	Matriz de polinômios
C_{arb1}	Taxa mássica de coque queimado no 1º estágio do regenerador (kgmol min^{-1})
C_{cat}	Concentração de coque no catalisador (% mássica)
CO_{d1}	Concentração de CO na fase diluída do 1º estágio do regenerador (% molar)
C_{O2}	Concentração de oxigênio na fase diluída do 1º estágio do regenerador (% molar)
C_{rc1}	Concentração de coque no catalisador no 1º estágio do regenerador (% molar)
C_{rc2}	Concentração de coque no catalisador no 2º estágio do regenerador (% molar)
C_{sc}	Concentração de coque no catalisador usado (% molar)
C/O	Razão mássica entre catalisador e gasóleo
CO_2/CO	Razão molar entre CO_2 e CO no leito do regenerador
CV_{LCV}	Coeficiente de fluxo da válvula de controle de nível do reator
D_{tf}	Densidade da alimentação (ton m^{-3})
F	Filtro de um modelo genérico
$F(\cdot)$	Mapeamento entrada-saída da planta
$\hat{F}(\cdot)$	Mapeamento não linear entrada-saída feito pelo modelo
F^{μ}	Função *fitness*

F_{at1} Razão mássica entre oxigênio consumido e coque queimado no 1º estágio

F_g Corrente total de gás produzido no regenerador (ton min^{-1})

F_{g1} Vazão de gás do 1º estágio do regenerador (kg min^{-1})

F_{gm1} Vazão de gás da fase densa do 1º estágio do regenerador (kgmol min^{-1})

F_{go} Vazão total de gás que deixa o regenerador para o refervedor (kg min^{-1})

F_{12} Fração da vazão de ar que entra no segundo estágio do reator

G Filtro de um modelo genérico

ΔH_{c1} Calor de combustão do coque no 1º estágio do regenerador (kcal $kgmol^{-1}$)

ΔH_{cr} Calor da reação de craqueamento (kcal kg^{-1})

ΔH_{fv} Calor latente de vaporização da alimentação de gasóleo (kcal kg^{-1})

h_1 Altura do leito fluidizado no 1º estágio do regenerador (m)

h_{ra} Altura do leito fluidizado do reator (m)

h_{sp} Altura do *stand pipe*

h_w Altura da vertedouro que separa o 1º do 2º estágio do regenerador (m)

H_{ra} Quantidade de catalisador no reator e *riser* (ton)

H_{rg1} Quantidade de catalisador no 1º estágio do regenerador (ton)

K_w Constante de fluxo da seção (ton min^{-1} $m^{-0,5}$)

m Quantidade de valores passados da entrada do modelo neural/linear (regressores)

M_{rg} Peso molecular médio da corrente gasosa no regenerador

n Quantidade de valores passados da saída do modelo neural/linear

n_y Máximo atraso nos regressores de entrada

n_w Máximo atraso nos regressores de saída

N Número de instantes de amostragem

O_{d1} Concentração de oxigênio na fase diluída do 1º estágio do regenerador (% molar)

O_{fg1} Concentração de oxigênio na fase densa do 1º estágio do regenerador (% molar)

p^{μ} Probabilidade de seleção do indivíduo

ΔP_{TCV} Pressão diferencial na válvula de catalisador regenerado (kgf cm^{-2})

ΔP_{LCV} Pressão diferencial na válvula de controle de nível do reator (kgf cm^{-2})

P_{ra} Pressão do reator (kgf cm^{-2})

P_{rg}	Pressão do regenerador (kgf cm^{-2})
R	Função de correlação do sinal de entrada
R	Constante dos gases ideais (kcal K^{-1} $kmol^{-1}$) (capítulo 5)
R_{a1}	Vazão de ar no 1º estágio do regenerador (ton h^{-1})
R_{cb1}	Taxa de combustão do coque no 1º estágio do regenerador (% mássica min^{-1})
R_{cf}	Taxa de formação de coque (ton min^{-1})
R_{co1}	Taxa de combustão de CO na fase diluída do regenerador (kgmol m^{-3} s^{-1})
R_{ma1}	Vazão de ar no 1º estágio (não incluindo o ar que vai para o 2º estágio) (kgmol min^{-1})
R_{oc}	Taxa da reação de craqueamento (ton min^{-1})
R_{rc}	Vazão de catalisador para o *riser* (ton min^{-1})
R_{rc1}	Vazão de catalisador do 1º para o 2º estágio (ton min^{-1})
R_{sc}	Vazão de catalisador usado (ton min^{-1})
R_{tf}	Vazão total de alimentação do *riser* (ton min^{-1})
S_a	Calor específico do ar e da corrente gasosa (kcal kg^{-1} $ºC^{-1}$)
S_c	Calor específico do catalisador (kcal kg^{-1} $ºC^{-1}$)
S_r	Calor específico do gasóleo (kcal kg^{-1} $ºC^{-1}$)
t	Tempo (min)
T_a	Temperatura do ar (ºC)
T_{d1}	Temperatura da fase diluída do 1º estágio do regenerador (ºC)
T_{fp}	Temperatura da entrada do *riser* (ºC)
T_{ra}	Temperatura do leito do reator (ºC)
T_{rg}	Temperatura média da corrente gasosa no regenerador (ºC)
T_{rg1}	Temperatura da fase densa do 1º estágio do regenerador (ºC)
T_{rg2}	Temperatura da fase densa do 2º estágio do regenerador (ºC)
T_{rx}	Temperatura do *riser* (ºC)
$u(t)$	Vetor de entradas do modelo matemático
V_1	Volume da fase densa do 1º estágio do regenerador (m^3)
V_{d1}	Volume da fase diluída do 1º estágio do regenerador (m^3)
V_{rg}	Volume livre total do regenerador (m^3)
$WHSV$	*Weight hourly space velocity* (ton de alimentação h^{-1} (ton de catalisador)$^{-1}$)

w^{μ}	*Fitness* biológico
w_{ij}	Peso aferido à ligação do neurônio *i* com o neurônio *j*
x_1, x_2, x_m	Sinais de entrada
$y(t)$	Vetor de saídas do modelo matemático
y_k	Sinal de saída do neurônio *k*
$\hat{y}_k$	Sinal de saída estimado do neurônio *k*

Símbolos Gregos

α	Probabilidade [0,1]
β	Controlador do grau de seleção do algoritmo genético
γ^m	Taxa de mutação
γ	Densidade do leito fluidizado (kgf m^3)
ϕ	Densidade espectral
φ	Função de ativação do neurônio
Λ	Matriz de covariâncias do ruído
$\vec{S}^{\mu}$	Vetor genético
S_i^{μ}	Indivíduo do vetor genético
ν	Potencial de ativação do neurônio
Π	População de soluções
θ	Vetor de parâmetros do modelo
ρ_1	Densidade molar da corrente gasosa da fase densa do 1º estágio do regenerador (kmol m^{-3})
ρ_{d1}	Densidade molar da corrente gasosa da fase diluída do 1º estágio do regenerador (kmol m^{-3})
$\sigma_e^{\ 2}$	Variância do ruído
χ	Razão entre o número de átomos de hidrogênio e carbono no coque

χ_i	Variável binária aleatória
ξ	Vetor de resíduos
ψ	Vetor de regressão
Σ	Somatório

Capítulo 1 - Introdução

1.1 MOTIVAÇÃO PARA O TRABALHO

Os modelos dinâmicos de processos são ferramentas básicas para diversos propósitos tais como simulação, controle, predição de estados, identificação de sistemas e otimização.

Uma das principais técnicas de modelagem matemática é a abordagem fenomenológica (que leva em conta as equações de balanço de massa e energia, leis da Física, Economia, entre outras). Os modelos assim gerados são extremamente úteis para apresentar o comportamento dinâmico do processo, podendo ser tão rigorosos quanto se queira. Embora estes modelos sejam preferíveis com relação aos modelos “caixa-preta” (baseados apenas nos dados de entrada e saída do processo, não necessitando de um conhecimento prévio sobre este), sua obtenção pode ser tarefa bastante custosa. Dada a grande disponibilidade de dados operacionais de alguns processos industriais, os modelos caixa-preta (RNA, ARX, etc.) apresentam-se como uma alternativa interessante.

Os modelos dinâmicos lineares, devido à sua versatilidade e simplicidade, têm sido muito utilizados, no entanto não são adequados para descrever processos com a multiplicidade de estados estacionários e as não linearidades características de grande parte dos processos da Engenharia Química.

O projeto de sistemas de controle para operações de processos com alta eficiência e desempenho deve garantir que o controlador seja capaz de atender às

especificações de produção sem violar as restrições (cada vez maiores devido à crescente exigência de plantas com alta eficiência).

A necessidade do desenvolvimento de algoritmos de controle avançado que sejam capazes de operar os processos com elevados níveis de desempenho, conciliado a condições de segurança, tem sido amplamente discutida na literatura. A estratégia preditiva é uma das técnicas propostas e uma das mais aplicadas em processos da indústria química.

Para tanto, é necessário desenvolver modelos cujas respostas apresentem o menor desvio possível com relação ao processo real, tempo de processamento pequeno e que sejam facilmente implementáveis nos controladores. Por apresentarem estas características, os modelos neurais são uma alternativa bastante viável para este uso.

Com o objetivo de propor uma estratégia de controle avançado para o processo (o que poderia resultar na diminuição de custos ou aumento de lucro da unidade), optou-se pela identificação da FCCU através de um modelo neural com dimensão bastante reduzida (número reduzido de parâmetros). A opção das redes neurais como modelo não linear deve-se ao fato deste modelo apresentar características altamente desejáveis para implementação no controle preditivo do processo.

Com o objetivo de testar métodos heurísticos estocásticos de otimização, propôs-se uma estratégia de otimização do processo utilizando uma associação entre algoritmos genéticos (AG) e de exame de partículas (PSO) e o modelo neural do processo. Os resultados obtidos na otimização da FCCU com os métodos heurísticos estocásticos foram comparados entre si e com as respostas do modelo fenomenológico, apresentando ótima convergência.

Por ser o processo de craqueamento catalítico muito lucrativo e envolver grande volumes de produto, qualquer melhoria nas condições operacionais resulta em grande retorno econômico. Isto motivou vários estudos no sentido de otimizar as condições de operação deste processo. Métodos heurísticos de otimização vêm sendo largamente empregados neste sentido. Os Algoritmos Genéticos (AG), que utilizam um método de busca inspirado na seleção natural (conceito emprestado da Genética), despontam como uma das estratégias de otimização mais promissoras. Do mesmo modo, a otimização por enxame de partículas (PSO) também se apresenta como uma estratégia bastante promissora.

1.2 OBJETIVOS DO TRABALHO

O principal objetivo deste trabalho é propor metodologias que possam melhorar o comportamento em malha fechada e a otimização da unidade de craqueamento catalítico em leito fluidizado. Para tanto, identificou-se um modelo neural do processo (o mais parcimonioso possível) capaz de predizer adequadamente o comportamento dinâmico da unidade de FCC. Este modelo foi posteriormente implementado nas estratégias de controle preditivo e de otimização da FCCU através de sua associação com o algoritmos genético e o enxame de partículas. Ambos os algoritmos apresentaram-se como alternativas promissoras para otimização do processo.

1.3 ESTRUTURA DA DISSERTAÇÃO

O Capítulo 2 apresenta uma breve introdução sobre modelagem matemática. São discutidos alguns tipos de modelos e é apresentada uma estrutura geral para modelos matemáticos.

O Capítulo 3 aborda alguns tópicos em identificação de processos. Um procedimento básico para a identificação de processos por redes neurais artificiais é apresentado e discutido.

No Capítulo 4 apresenta-se uma breve introdução sobre controle preditivo, incluindo-se o Controle Preditivo baseado em Modelo (MPC), com destaque para modelos não-lineares.

No Capítulo 5 é apresentada uma visão geral do processo de craqueamento catalítico, as principais variáveis de interesse e as equações do modelo dinâmico.

O Capítulo 6 contém uma breve discussão a respeito da otimização de processos e de métodos heurísticos estocásticos de otimização, com destaque para os algoritmos genéticos e o enxame de partículas.

Os resultados e a discussão são apresentados no Capítulo 7 e as considerações finais e algumas perspectivas futuras no Capítulo 8.

Capítulo 2 – Modelos Matemáticos

Este capítulo tem a finalidade de apresentar uma breve introdução sobre modelagem matemática. Apresenta-se a classificação dos modelos matemáticos, assim como sua estrutura geral.

2.1 INTRODUÇÃO

Um modelo é uma descrição matemática de um processo real qualquer. Pode-se agrupar as técnicas de modelagem em duas grandes categorias (SÖDERSTRÖM e STOICA, 1989; AGUIRRE, 2000). A primeira é uma aproximação analítica onde leis básicas da Física (como as leis de Newton e equações de balanço), Economia, entre outras, são usadas para descrever o comportamento de um fenômeno ou processo. É a abordagem físico-matemática ou, ainda, fenomenológica ou determinística. A identificação de sistemas é, por sua vez, uma aproximação experimental. Ela leva à construção de modelos matemáticos de um sistema dinâmico baseada em dados de entrada e saída medidos. A obtenção dos parâmetros de um modelo determinístico, por exemplo, pode ser chamada também de identificação.

Dependendo do nível de conhecimento inicial do sistema, o problema de modelagem pode ser abordado de maneiras diferentes. O termo *modelagem caixa-branca* é utilizada quando o sistema é modelado totalmente a partir de leis físico-químicas (modelagem fenomenológica). Os modelos fenomenológicos são, em tese, potencialmente úteis para a adequada investigação do comportamento de sistemas reais nas mais variadas condições de operação. Se a identificação é baseada em

dados amostrados, assumindo muito pouco ou nenhum conhecimento sobre o sistema, o processo de identificação é chamado *modelagem caixa-preta*. Uma abordagem intermediária, a *modelagem caixa-cinza*, não exige um profundo conhecimento prévio sobre o sistema mas permite que esta informação, caso esteja disponível, seja incorporada ao modelo (AGUIRRE *et al*.,1998 *apud* MELEIRO, 2002).

O grande apelo da modelagem caixa-preta é devido, principalmente, à dificuldade na obtenção de um tratamento matemático genérico para sistemas não-lineares, considerando que é praticamente impossível representar adequadamente todo tipo de conhecimento em sistemas não-lineares. Um outro problema é que, mesmo que algum conhecimento sobre o sistema esteja disponível, relacioná-lo corretamente a uma descrição dinâmica do sistema contínua no tempo (isto é, em termos de equações diferenciais) pode ser uma tarefa extremamente difícil. Portanto, a escolha da modelagem caixa-preta não é incomum, mesmo quando certo nível de conhecimento sobre o sistema está disponível. Contudo, um bom entendimento sobre as leis físicas que regem o comportamento do sistema é sempre útil e facilita bastante o processo de identificação. Tal conhecimento pode estar relacionado à ordem do sistema, à dinâmica principal do sistema (rápida ou lenta), à freqüência de amostragem adequada, às características de estabilidade, à faixa de operação em que se pretende operar o sistema, tempos de atraso, grau de não linearidade, entre outros. (MELEIRO, 2002)

2.2 CLASSIFICAÇAO DOS MODELOS

Modelos matemáticos de sistemas dinâmicos podem ser classificados de várias maneiras. Eles podem descrever como o efeito de um sinal de entrada influenciará o

comportamento do sistemas nos instantes subseqüentes (SÖDERSTRÖM e STOICA, 1989).

Segue-se algumas formas de classificação de modelos dinâmicos:

- Modelos monovariáveis – modelos multivariáveis. Nos modelos SISO (*Single input, single output*) há influência de uma saída e uma entrada. Quando mais variáveis estão envolvidas tem-se um modelo multivariável MISO (*multi-input, single output*) ou MIMO (*multi-input, multi-output*). (SÖDERSTRÖM e STOICA, 1989).
- Modelos lineares – modelos não lineares. Em um modelo linear a saída está linearmente relacionada com a entrada do processo. Sua principal característica é o princípio da superposição. Caso contrário, o modelo será não linear. (SÖDERSTRÖM e STOICA, 1989).
- Modelos que variam no tempo – modelos que não variam no tempo. Para modelos que variam com o tempo é necessário um método especial de identificação. Em alguns onde os parâmetros variam com o tempo usa-se identificação dos parâmetros em tempo real. (SÖDERSTRÖM e STOICA, 1989).
- Modelos de tempo discreto – modelos de tempo contínuo. Um modelo de tempo discreto descreve a relação entre entradas e saídas em pontos discretos. (SÖDERSTRÖM e STOICA, 1989).
- Modelos determinísticos – modelos estocásticos. Um modelo determinístico possibilita o cálculo exato da saída logo que o sinal de entrada é conhecido. Em contrapartida, um modelo estocástico contém termos aleatórios que tornam um cálculo exato impossível. (SÖDERSTRÖM e STOICA, 1989).

2.3. ESTRUTURA GERAL DE UM MODELO

Segundo Söderström e Stoica (1989), a forma geral de um modelo pode ser descrita do seguinte modo:

$$y(t) = G(q^{-1};\theta)u(t) + H(q^{-1};\theta)e(t)$$
$$Ee(t)e^{T}(s) = \Lambda(\theta)\delta_{t,s} \qquad (2.1)$$

Na equação 2.1, $y(t)$ é a saída no tempo t e $u(t)$ é a entrada, $e(t)$ é uma seqüência de variáveis independentes distribuídas aleatoriamente com média zero. Tal seqüência é chamada ruído branco. $G(q^{-1};\theta)$ e $H(q^{-1};\theta)$ são filtros. O argumento q^{-1} denota o operador de atraso, ou seja: $q^{-1}u(t) = u(t-1)$. O modelo 2.1 pode ser melhor visualizado na figura 2.1.

Os filtros $G(q^{-1};\theta)$ e $H(q^{-1};\theta)$ bem como a matriz de covariâncias do ruído, $\Lambda(\theta,)$ são funções do vetor θ (que assumimos ser um vetor de dimensão $n\theta$). Este vetor é um parâmetro que identifica o modelo (SÖDERSTRÖM e STOICA, 1989).

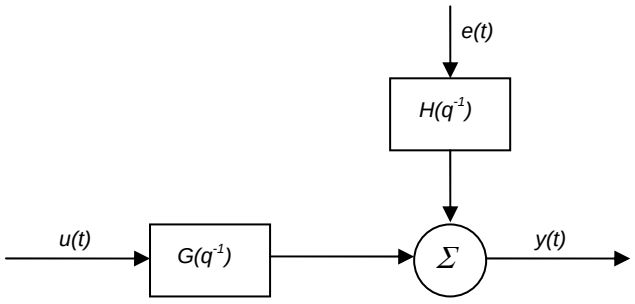

Figura 2.1 Diagrama de blocos de um modelo genérico
Fonte: SÖDERSTRÖM e STOICA, 1989, pg 148

2.4 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

Neste capítulo foi apresentada uma introdução geral e simplificada à modelagem matemática, uma breve classificação de modelos dinâmicos e uma estrutura geral de um modelo matemático.

Capítulo 3 – Identificação de Processo

Este capítulo aborda alguns itens em identificação de sistemas. A seção 3.2 descreve o procedimento básico do processo de identificação. O paradigma neural e os tópicos específicos como neurônios, arquitetura de rede, regressores e treinamento são apresentados na seção 3.3. A estimação de modelos neurais é discutida na seção 3.4.

3.1 INTRODUÇÃO

A identificação de sistemas se propõe a obter um modelo matemático que explique, pelo menos em parte e de forma aproximada, a relação de causa e efeito presente entre os dados de entrada e saída disponíveis do sistema ou processo. Os parâmetros deste modelo são ajustados até que suas saídas coincidam, considerando um erro arbitrariamente estipulado, com as saídas medidas do processo. Assim sendo, um dos primeiros problemas em identificação de processos é a obtenção de dados de entrada e saída adequados, em quantidade e amplitude de condições de operação, para o processo de identificação. Outro problema que se apresenta é a escolha da representação matemática a ser usada e, decidido o modelo, que estrutura usar. No caso da escolha de uma rede neural entende-se por estrutura do modelo a determinação do número de neurônios, regressores das variáveis, etc. O último problema enfrentado é a validação do modelo: obtidos alguns modelos, é necessário verificar se eles representam satisfatoriamente o sistema original e se algum é melhor do que os demais, o que também pode ser chamado de discriminação dos modelos (FROMENT e BISCHOFF,1990).

A escolha da estrutura do modelo depende do conhecimento prévio que se tem do processo a ser identificado. Em aplicações reais é impossível obter uma estrutura capaz de descrever o sistema exatamente (SJÖBERG,1995). Uma suposição comum nesta abordagem é que o sistema é linear, o que raramente é verdadeiro em aplicações reais, mas com freqüência é uma boa aproximação. Isto se deve ao fato de que a teoria de sistemas lineares é bem desenvolvida e há muitos resultados que podem ser aplicados para obter modelos não-lineares, embora a complexidade aumente. Esta escolha de estrutura é uma das razões pelas quais a identificação de sistemas é muito mais complicada para modelos não-lineares considerando que há muito mais alternativas: a não linearidade de uma função pode ser causada por vários fatores; em contrapartida pode trazer maior flexibilidade ao modelo (mais parâmetros ajustáveis) (SJÖBERG,1995).

3.2 PROCEDIMENTO BÁSICO

Vários autores (SÖDERSTRÖM e STOICA, 1989; AGUIRRE, 2000; NØRGAARD *et al*., 2000 *apud* MELEIRO, 2002) destacam algumas etapas básicas necessárias para identificar sistemas dinâmicos não lineares. De uma maneira geral, estas etapas podem ser resumidas da seguinte forma:

1. Realização de experimentos para extração de dados
2. Seleção da estrutura do modelo
3. Estimação do modelo
4. Validação do modelo

Contudo, deve-se ter em mente que conhecimentos sobre a dinâmica do sistema e/ou informações sobre o uso que se pretende dar ao modelo podem influenciar os estágios do procedimento. (MELEIRO, 2002)

3.2.1 Realização de experimentos

Este é o primeiro estágio do processo de identificação de modelos e o principal objetivo é projetar e realizar um conjunto de experimentos que descreva como o sistema responde quando submetido a diferentes entradas (ações de controle ou perturbações), assegurando que toda a faixa de operação seja contemplada. A importância desta etapa está diretamente associada à precisão do modelo que será obtido, uma vez que o conjunto de dados de entrada-saída gerado no experimento será utilizado no processo de treinamento das redes neurais (de modo geral, isto se aplica a toda modelagem caixa-preta) (MELEIRO, 2002).

A idéia básica é variar a(s) entrada(s) do sistema, $\underline{\boldsymbol{u}}$, de modo a cobrir toda a sua faixa de operação e observar o impacto dessas entradas sobre a(s) sua(s) saída(s), $\underline{\boldsymbol{y}}$. O efeito das perturbações pode influenciar fortemente o comportamento do sistema e, tratando-se de perturbações mensuráveis, o seu registro é de fundamental importância na etapa de estimação do modelo do processo. (MELEIRO,2002).

O projeto de experimentos, no entanto, é mais complexo do que simplesmente gerar dados de entrada-saída para o treinamento das redes neurais. Tópicos de grande importância neste contexto são: i) Teste para decidir se a abordagem de identificação não linear é relevante; ii) Projetos de sinais de entrada que levem a um conjunto de dados com informações adequadas sobre a dinâmica do sistema; e iii)

Técnicas de preparação dos dados para o treinamento das redes neurais. (MELEIRO, 2002).

Nesta dissertação os dados foram obtidos de um simulador do processo, baseado no modelo de Moro e Odloak ,1995.

3.2.1.1 A “Maldição da Dimensionalidade”

Uma vez que os sistemas não lineares não apresentam as propriedades de superposição e homogeneidade, existe a necessidade de excitar o sistema adequadamente através de um sinal de entrada rico o suficiente em freqüências e amplitudes. Isto significa que o sistema deve ser excitado com todas as combinações de freqüências e amplitudes possíveis dentro de sua faixa operacional. A geração de uma base de dados como esta implica, invariavelmente, em um número maior de dados necessários para identificar um sistema não linear quando comparado a sistemas lineares. Como conseqüência o número de parâmetros e o conjunto de dados necessários para identificar um sistema não linear com uma dada precisão cresce exponencialmente com a dimensão do espaço de entradas do mapeamento a ser aproximado. Este é um problema conhecido como a “Maldição da Dimensionalidade” (HAYKIN, 2001). Este problema configura um ponto fraco da abordagem caixa-preta não linear, embora alguns resultados envolvendo estruturas hierárquicas tenham demonstrado soluções viáveis para este tipo de problema (MELEIRO, 2002).

3.2.1.2 Projeto do sinal de entrada

A seleção de uma seqüência adequada de sinais de entrada está intimamente relacionada com a correta verificação da faixa de operação do processo. É importante aplicar uma seqüência de sinais de entrada que realmente excitem o sistema de modo que toda a sua faixa operacional seja coberta, garantindo, assim, um conjunto de dados bem amplo, evitando ao máximo a coleta de informações redundantes.

Duas estratégias capazes de gerar sinais que atendam à demanda existente são apresentadas a seguir;

I) Uma seqüência de sinais de entrada com mudança de nível de ativação (amplitude) variando em intervalos (freqüência) constantes pode ser gerada da seguinte forma:

Seja *e(t)* um sinal correspondente a um ruído com variância σ_e^2. O sinal de entrada definido de acordo com a equação (3.1) mudará para um novo nível de ativação a cada novo instante de amostragem *N*.

$$u(t) = e\left(\text{int}\left[\frac{t-1}{N} \right] + 1 \right), \quad t = 1,2,... \tag{3.1}$$

onde *int* significa a parte inteira da divisão. A função de correlação do sinal de entrada é dada por:

$$R_u(\tau) = \frac{N-\tau}{N}\sigma_e^2 \tag{3.2}$$

correspondendo à seguinte densidade espectral

$$\phi(\omega) = \frac{\sigma_e^2}{2\pi N}\frac{1-\cos(N\omega)}{1-\cos(\omega)} \tag{3.3}$$

II) Uma extensão da metodologia descrita no item I, pode ser obtida introduzindo-se uma variável aleatória adicional que determine quando o sinal de entrada será alterado:

$$u(t) = \begin{cases} u(t-1) & com\ probabilidade\ \alpha \\ e(t) & com\ probabilidade\ (1-\alpha) \end{cases} \tag{3.4}$$

Este sinal tem a seguinte função de covariância

$$R_u(\tau) = \alpha^{\tau}\sigma_e^2 \tag{3.5}$$

correspondendo à densidade espectral

$$\phi(\omega) = \frac{\sigma_e^2}{2\pi}\ \frac{1-\alpha}{1+\alpha^2-2\alpha\cos(\omega)} \tag{3.6}$$

Um exemplo de sinal gerado desta forma é mostrado na figura a seguir

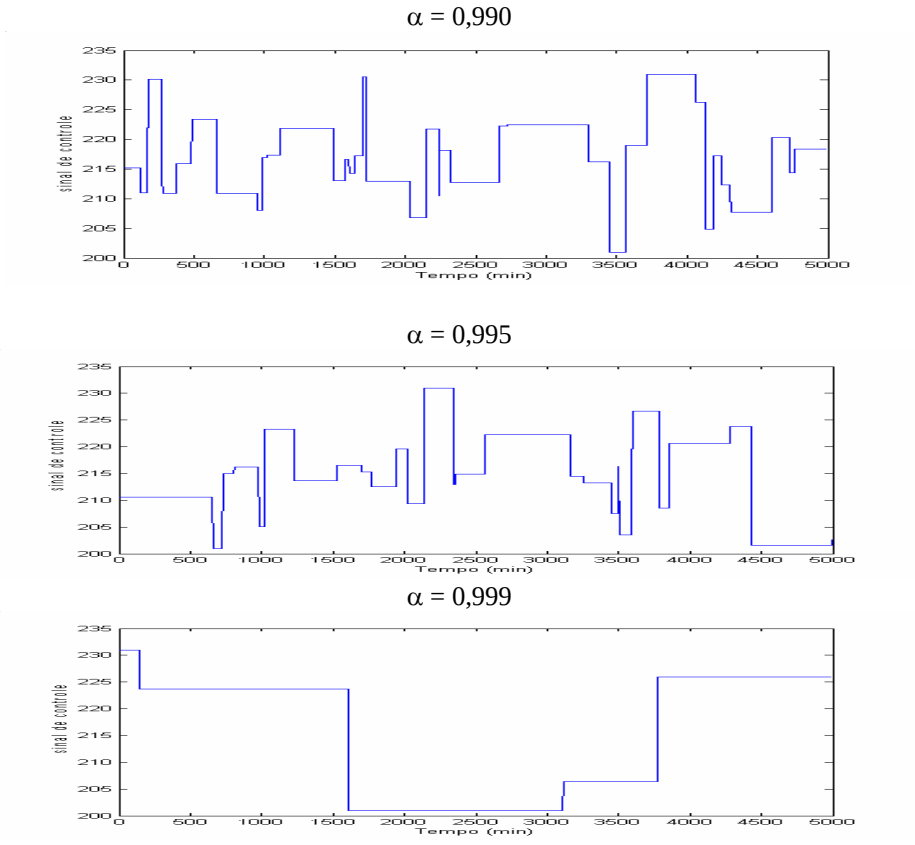

Figura 3.1 Sinal de entrada com nível de ativação variável

3.3 REDES NEURAIS ARTIFICIAIS

Entre os benefícios atribuídos ao uso de redes neurais artificiais na modelagem de processos estão a sua habilidade de ajustar-se dinamicamente às mudanças no ambiente, a possibilidade de generalização a partir de exemplos específicos e a habilidade de reconhecer invariâncias de dados complexos de espaços multidimensionais (VENKATASUBRAMANIAN e CHAN, 1989).

As redes neurais artificiais têm sido largamente utilizadas para identificação e controle de sistemas dinâmicos não-lineares. Uma das principais razões para este sucesso é a capacidade de aproximação universal dos modelos neurais , isto é, tais modelos são capazes de aproximar com precisão arbitrária qualquer mapeamento contínuo definido em um domínio compacto.

3.3.1 Inteligência Artificial

A Inteligência Artificial é o ramo da ciência que estuda o conjunto de paradigmas que pretendem justificar como um comportamento inteligente pode emergir de implementações artificiais em computadores. Uma das características dos sistemas inteligentes é a capacidade de aprender, de se adaptar a um ambiente desconhecido ou a uma situação nova (BAUCHSPIESS, 2004).

De acordo com a disponibilidade de dados e/ou teoria, diferentes métodos são indicados para cada situação particular. A figura 3.2 apresenta os principais métodos:

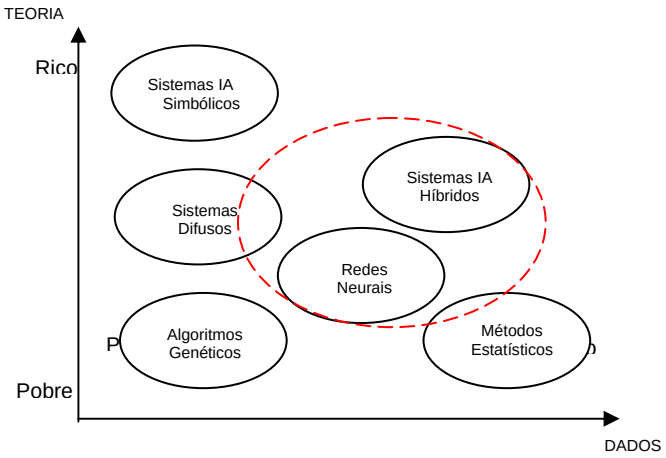

Figura 3.2 Métodos de engenharia do conhecimento
Fonte: BAUCHSPIESS, 2004

Quando existem apenas exemplos (amostras representativas) de um dado processo, sem regras que expliquem a sua geração, então os métodos estatísticos permitem obter os melhores resultados. No extremo oposto do gráfico estão os métodos de Inteligência Artificial Simbólicos, como, por exemplo, os sistemas especialistas. Neste caso, o importante são as regras e o processo de inferência que permite resolver um certo problema. A teoria – lógica clássica baseada em axiomas – é muito bem estabelecida e prescinde de exemplos para a implementação do sistema especialista (BAUCHSPIESS, 2004).

Os sistemas baseados em Redes Neurais Artificiais (RNA) ocupam uma região intermediária em relação ao gráfico Teoria x Dados. Estes sistemas exploram razoavelmente bem as amostras do processo. De fato, uma das grandes vantagens desta abordagem é a possibilidade de treinamento das RNAs a partir dos dados. Não são necessárias regras ou uma teoria que descreva o processo, as RNAs simplesmente “aprendem” (BAUCHSPIESS, 2004), ou seja, criam uma relação de entrada e saída dos dados.

3.3.2 Revisão Bibliográfica

Considerando a importância deste tópico no desenvolvimento do trabalho, apresenta-se a seguir uma breve revisão bibliográfica, de forma a apresentar uma visão geral da evolução da aplicação de Redes Neurais Artificiais (RNA) na indústria

As idéias iniciais dos estudos de redes neurais surgiram com os trabalhos de MacCulloch e Pitts (1943), em que sugeriram a construção de uma máquina baseada ou inspirada no cérebro humano. Hebb (1949) traduziu matematicamente a sinapse dos neurônios biológicos. Minski (1951) construiu o primeiro neurocomputador com capacidade de aprendizado, ou seja, que ajustava automaticamente os pesos entre as sinapses. Em 1956 surgiram os dois paradigmas da Inteligência Artificial: a simbólica e a conexionista. A simbólica procura simular o comportamento inteligente humano desconsiderando os mecanismos responsáveis por ele. A conexionista acredita que, construindo-se um sistema que simule a estrutura do cérebro, este sistema apresentará inteligência, ou seja, será capaz de aprender, assimilar, errar e aprender com seus erros. Rosemblat (1957) concebeu o perceptron, uma rede neural com duas camadas, usada no reconhecimento de caracteres. Widrow (1962) desenvolveu um processador para redes neurais (HAYKIN, 2001)

A partir da década de 80, publicações de grande contribuição científica no desenvolvimento das redes neurais foram retomadas.

Venkatasubramanian e Chan (1989) apresentam uma metodologia utilizando Redes Neurais para detecção de falhas de processo.

Henrique *et al*. (2000) propuseram uma metodologia para determinar a estrutura de modelo em redes neurais *feedforward*.

Aldrich e Slater (2001) apresentaram uma identificação de uma extração líquido-líquido utilizando redes neurais com sucesso. Outros artigos apresentando outras iniciativas de identificação de processos se seguiram a estes.

Algumas propriedades vantajosas das redes neurais como mapeamento não linear e capacidade de aprendizagem as tem tornado uma ferramenta útil para solução de muitos problemas em Engenharia Química (NAGY *et al*. 2000). Por esta causa, tem surgido vários trabalhos de aplicação de Redes Neurais em identificação de processos e posterior aplicação em controladores preditivos.

Sendo o processo de craqueamento catalítico um dos mais complexos e lucrativos da indústria petroquímica moderna, é natural o surgimento de pesquisas que proponham alguma melhoria no processo. Assim, novas soluções de implementação de redes neurais na identificação e no controle do processo tem surgido (NAGY *et al.*, 2000; SANTOS *et al.*, 2000; ALARADI e ROHANI, 2002; BOLLAS *et al.*, 2003; VIEIRA *et al.*, 2005).

3.3.3 Redes Neurais Artificiais

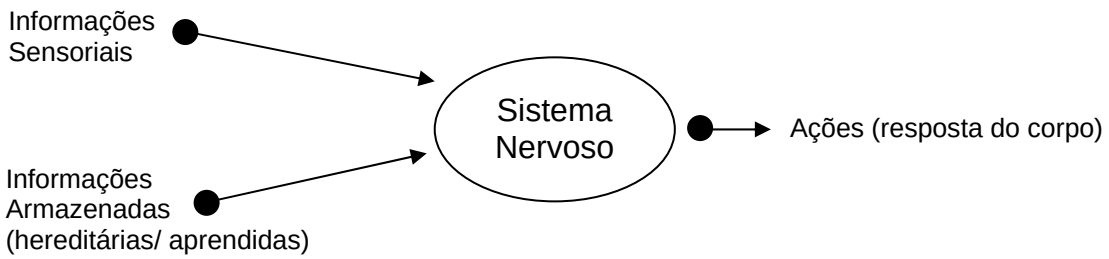

Figura 3.3 Funcionamento do sistema nervoso

Fonte: Bauchspiess, 2004

O sistema nervoso humano obtém informações do meio ambiente através de sensores que são combinadas com informações armazenadas para produzir as ações do corpo. Apenas uma pequena parte das informações obtidas é relevante para o funcionamento do corpo (BAUCHSPIESS, 2004).

As redes neurais artificiais são inspiradas pela arquitetura do sistema nervoso biológico, que consiste de um grande número de células nervosas ou neurônios relativamente simples funcionando paralelamente para facilitar decisões rápidas. Uma rede neural artificial consiste de um grande número de elementos computacionais primitivos arranjados em uma estrutura paralela. Estes elementos são conectados por sinapses artificiais, caracterizadas por uma matriz de pesos ou valores numéricos, que podem ser ajustados por um processo de aprendizagem. As redes neurais artificiais têm sido usadas com muito sucesso em controle de processos, modelagem, simulação e identificação de sistemas. (ALDRICH e SLATER, 2001)

3.3.3.1 O neurônio matemático

Essencialmente, uma rede neural é uma rede de elementos primitivos de processamento (também chamados nós computacionais ou neurônios), como mostrados na figura 3.4.

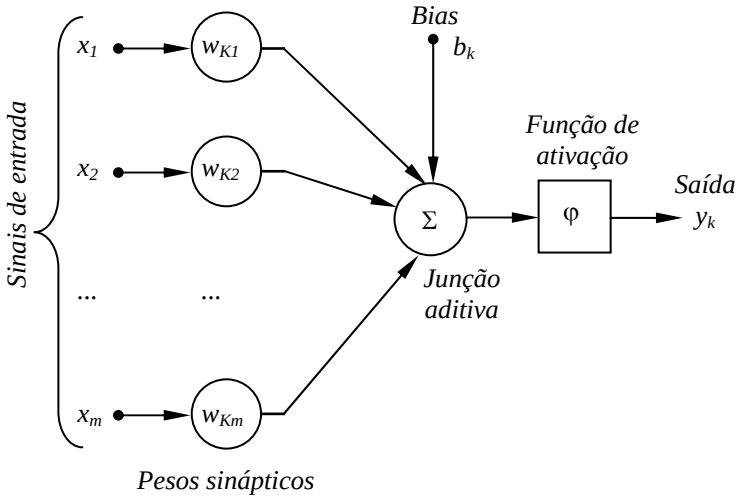

Figura 3.4 Modelo de neurônio

Fonte: HAYKIN, 2001, p.36

No neurônio biológico, a informação é recebida pelo neurônio através dos dendritos, que são terminações nervosas. A informação é então processada pelo corpo celular e a resposta é transmitida para outra célula nervosa através do axônio. O neurônio matemático é uma representação deste sistema. (HAYKIN, 2003)

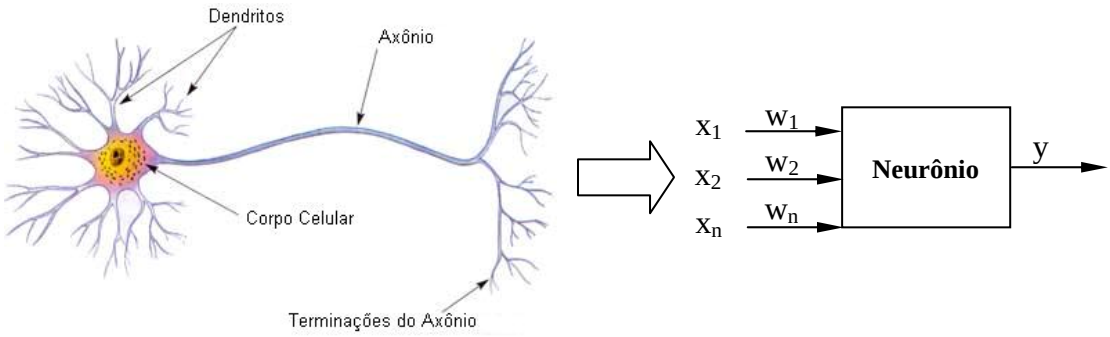

Figura 3.5 Analogia entre os neurônios biológico e matemático

Fonte: GONZAGA; 2003, p. 53.

Pode-se identificar três elementos básicos no modelo da Figura 3.4:

Primeiro, um conjunto de sinapses caracterizadas por um peso próprio. Um sinal x_j na entrada da sinapse j conectada ao neurônio k é multiplicado pelo peso sináptico w_{kj}. Segundo, um somador, que soma os sinais de entrada, ponderados

pelos respectivos pesos dos neurônios (combinador linear). E, terceiro, uma função de ativação, que restringe a amplitude da saída de um neurônio.

O modelo apresentado na figura tem também um *bias*, aplicado externamente (b_k), que tem o efeito de aumentar ou diminuir a entrada líquida da função de ativação (HAYKIN, 2001).

Matematicamente, pode-se descrever um neurônio k através das seguintes equações:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.7}$$

$$y_k = \varphi(u_k + b_k) \tag{3.8}$$

Onde $x_1, x_2, ..., x_m$ são os sinais de entrada; $w_{k1}, w_{k2}, ..., w_{km}$ são os pesos sinápticos do neurônio k; u_k é a sida do combinador linear devido aos sinais de entrada; b_k é o bias; φ (•) é a função de ativação e y_k é o sinal de saída do neurônio.

A forma da função de transferência (ou ativação), que define o sinal de saída do neurônio, varia bastante, pode ser linear, degrau ou sigmoidal, entre outras.

A função limiar, definida no intervalo de normalização unitário [0,1] (HAYKYN, 2001):

$$\varphi(v) = \begin{cases} 1 \ se \ v \geq 0 \\ 0 \ se \ v < 0 \end{cases} \tag{3.9}$$

onde v é o potencial de ativação do neurônio k, isto é, a soma ponderada de todas as entradas sinápticas e do *bias*:

$$v_K = \sum_{j=1}^{m} w_{kj} x_j + b_k \tag{3.10}$$

A saída do neurônio k que emprega a função de limiar é expressa como:

$$y_k = \begin{cases} 1 \ se \ v_k \geq 0 \\ 0 \ se \ v_k < 0 \end{cases} \tag{3.11}$$

A função de ativação linear por partes assume que o fator de ampliação dentro da região linear de operação é a unidade. Esta forma de função de ativação pode ser vista como uma aproximação de um amplificador não-linear. A função de ativação linear por partes definida para o intervalo de normalização unitário [0,1] (HAYKYN, 2001):

$$\varphi(v) = \begin{cases} 1, & v \geq \frac{1}{2} \\ v, & \frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{3.12}$$

A função sigmoidal tem sido usada freqüentemente com bastante sucesso (SANTOS *et al*., 2000):

$$\varphi(v) = \frac{1}{1+e^{-\alpha}} \tag{3.13}$$

onde α é o parâmetro de inclinação da função sigmóide. No limite, quando o parâmetro de inclinação se aproxima de infinito, a função sigmóide se reduz à função de limiar (HAYKYN, 2001).

Um outro exemplo de função sigmóide é a tangente hiperbólica, definida para intervalos de normalização [-1,1] (HAYKYN, 2001):

$$\varphi(v) = \tanh(v) \tag{3.14}$$

3.3.3.2 Arquitetura da rede

O modo como os neurônios são conectados e sua estrutura determinam a arquitetura da rede (VIEIRA *et al*., 2005). Esta arquitetura está intimamente ligada ao

algoritmo de aprendizagem utilizado para treinar a rede. Em geral, pode-se identificar três classes de arquiteturas de rede fundamentalmente diferentes: (HAYKIN, 2001)

1) *Feedforward* de camada única. Nesta rede uma camada de entrada se projeta sobre uma camada de saída (o termo “camada única” refere-se à camada de saída) (Figura 3.7).

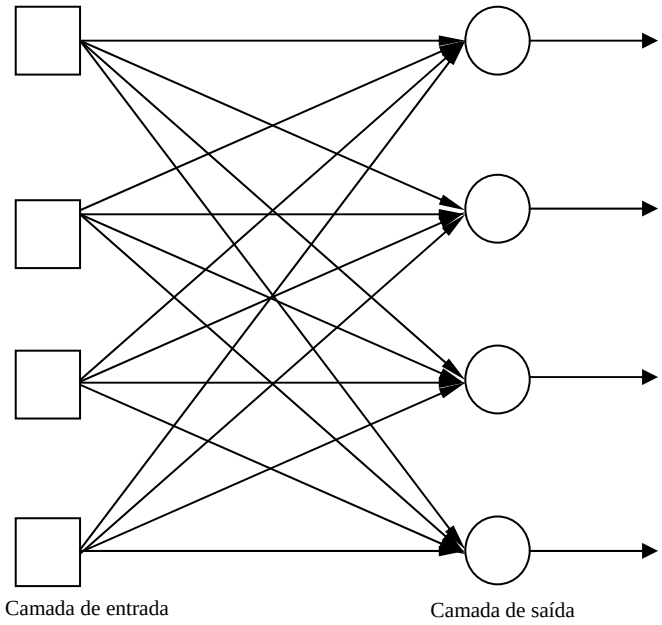

Figura 3.6 Rede feedforward de camada única
Fonte: HAYKIN, 2001, p.47

2) *Feedforward* de múltiplas camadas. Distingue-se da anterior pela presença de uma ou mais camadas ocultas. A estrutura de uma rede *feedforward* básica é mostrada na Figura 3.7. A rede tem uma camada de entrada, uma de saída e uma ou mais camadas escondidas. Os neurônios são conectados por sinapses artificiais (cada uma delas associada a um peso ou valor numérico). A rede é treinada (ou seja, os pesos são adaptados) baseando-se em

exemplos do processo. Nesta estrutura, todos os neurônios de uma camada se conectam com todos os neurônios da camada seguinte, não havendo conexão com neurônios da mesma camada ou camadas anteriores. Tais modelos apresentam a importante característica de serem aproximadores universais. As redes neurais utilizadas com mais freqüência são as do tipo Perceptron Multicamadas (MLP) com estrutura *feedforward* (HAYKIN, 2001).

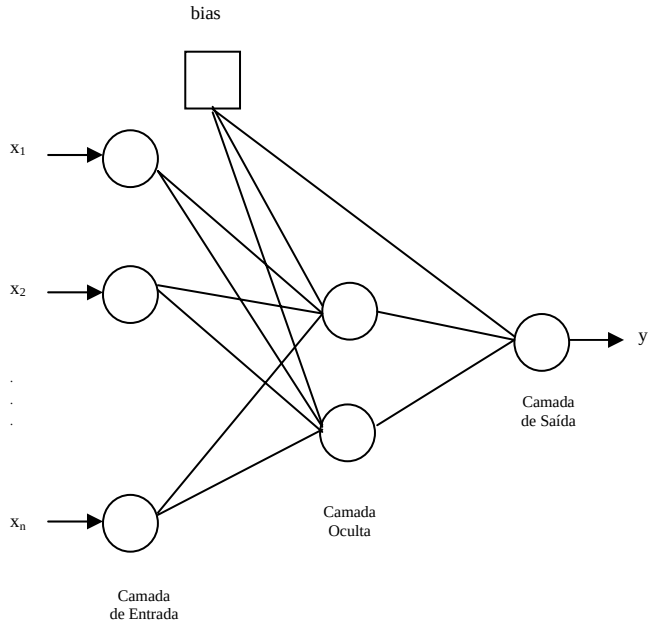

Figura 3.7 Estrutura genérica de uma rede feedforward com uma camada oculta
Fonte: ALDRICH e SLATER, 2001 pg 7

3) Recorrente. Se distingue de uma rede feedforward por ter, pelo menos, um laço de realimentação. A presença de laços de realimentação tem um impacto profundo na capacidade de aprendizagem da rede e de seu desempenho. Figura 3.8.

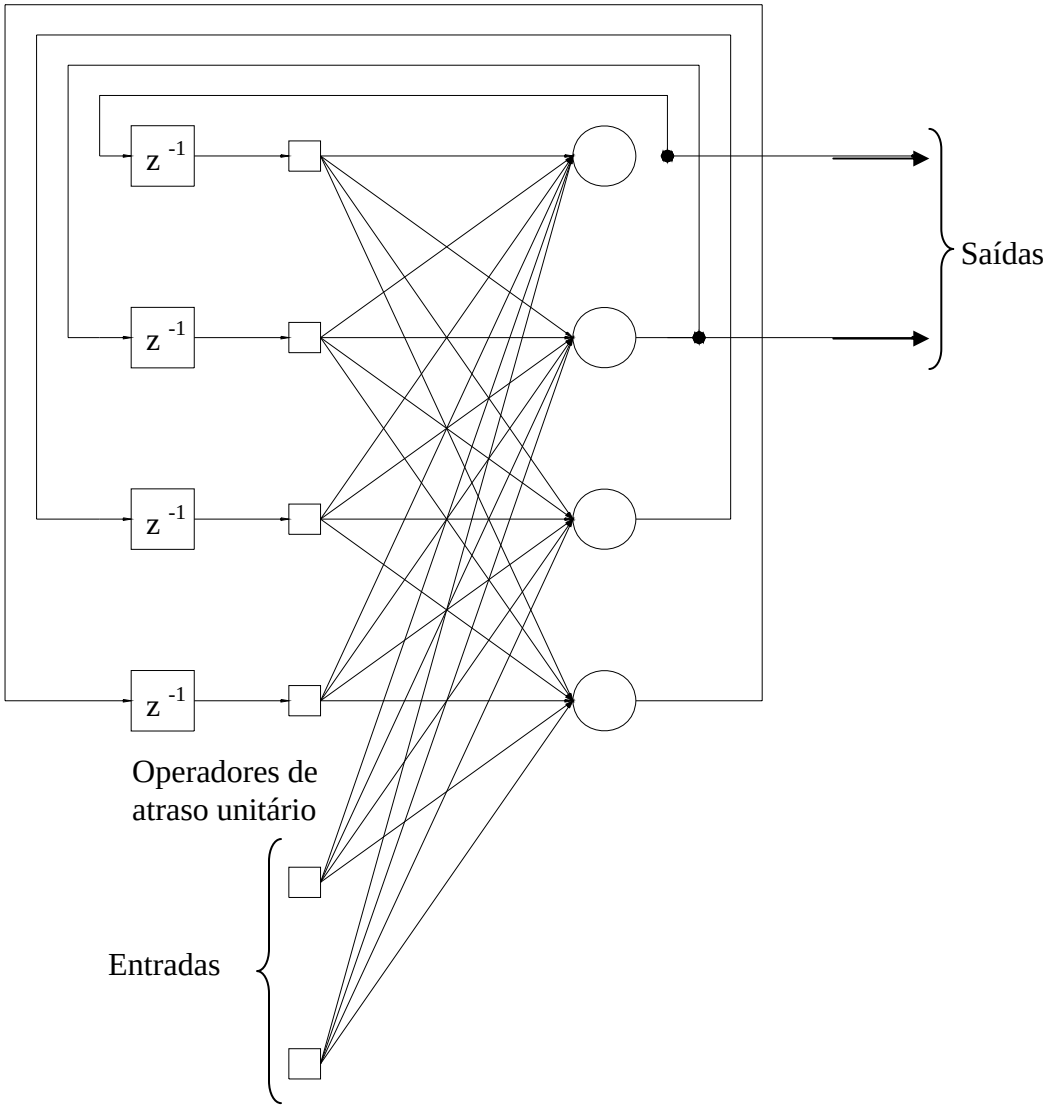

Figura 3.8 Rede recorrente com neurônios ocultos

Fonte: HAYKYN, 2001, p.49

Usualmente as camadas são classificadas em três grupos:

1) **Camada de Entrada**: onde os padrões são apresentados à rede;
2) **Camadas Intermediárias ou Escondidas**: onde é feita a maior parte do processamento, através das conexões ponderadas. Podem ser consideradas como extratoras de características;
3) **Camada de Saída**: onde o resultado final é concluído e apresentado.

O estabelecimento de uma estrutura de rede neural deve considerar tanto o número de amostras utilizadas na fase de treinamento, como a precisão desejada nas previsões a serem feitas fora do conjunto utilizado para treinamento. O problema de se definir arquiteturas de redes neurais tem sido investigado por vários autores e as causas de imprecisão que devem ser consideradas para aproximação de sistemas não lineares são as seguintes:

a) O número e a qualidade das amostras utilizadas para o treinamento pode ser insuficiente para captar a estrutura real da função ou comportamento do sistema a serem aproximados;
b) A rede poderá estar sub-dimensionada, não possuindo um número suficiente de neurônios na camada intermediária e, portanto, a tolerância desejada não poderá ser atingida.

O item *a* pode ser considerado como um projeto de rede e, dessa forma, torna-se necessário um certo conhecimento do mapeamento a ser ensinado à rede. A segunda fonte de imprecisão pode ser evitada pela escolha correta da arquitetura da rede neural. Para isso, são estabelecidos limites para o número de elementos das camadas intermediárias.

Como conseqüência da saída da rede neural ser descrita por uma função não linear, seus parâmetros devem ser determinados por algoritmos específicos para estimação não linear. Existem muitos tipos de algoritmos específicos para estimação de parâmetros de redes neurais, dentre os quais se destaca o método do gradiente como algoritmo utilizado com maior freqüência (embora não seja o mais eficiente) (MELEIRO, 2002).

3.3.3.2.1 Os Regressores

Na identificação de sistemas lineares, a determinação da estrutura do modelo consiste, basicamente, na determinação da ordem do modelo. Já no caso de sistemas não lineares, o problema torna-se mais complexo e, considerando que a identificação do processo será feita com redes neurais artificiais, a seleção da estrutura do modelo consiste em escolher não somente as entradas da rede mas também um conjunto adequado de regressores, além da própria arquitetura da rede neural. Por regressores entende-se o conjunto de termos passados relevantes, associados às variáveis de entrada e saída do processo, que serão utilizados como entradas do modelo a ser estimado. A necessidade de escolher um valor adequado para a ordem de um sistema pode ser entendida verificando-se que, se a ordem utilizada for muito menor do que a ordem efetiva do sistema real, o modelo não possuirá a complexidade estrutural necessária para reproduzir a dinâmica do sistema. Por outro lado, se a ordem do modelo for muito maior que a necessária, a estimação dos parâmetros provavelmente será mal condicionada (apresentará problemas numéricos). Para o caso de modelos baseados em redes neurais artificiais, a estratégia utilizada com mais freqüência é selecionar os regressores baseado nos conceitos advindos da teoria de identificação de sistemas lineares e então determinar a melhor arquitetura possível na RNA tomando os regressores como entradas da rede neural. NØRGAARD *et al.* (2000) destacam as seguintes vantagens nesta abordagem:

- É uma extensão natural da (bem estabelecida) abordagem linear.
- A arquitetura interna da rede pode ser expandida para possibilitar mapeamentos não lineares mais complexos.

- Nível de complexidade não muito alto associado à determinação da estrutura do modelo.
- Adequado para o projeto de controladores.

A estrutura de um modelo não linear pode ser obtida genericamente por

$$y(t) = g[\boldsymbol{\varphi}(t,\boldsymbol{\theta}),\boldsymbol{\theta}] + e(t) \tag{3.15}$$

onde $\boldsymbol{\varphi}(t,\boldsymbol{\theta})$ é o **Vetor de Regressão** (ou regressores) do modelo, que contém as entradas e saídas passadas; $\boldsymbol{\theta}$ é o vetor de parâmetros ajustáveis da rede neural (também conhecidos como pesos), g é a função que realiza o mapeamento não linear (aqui representada por uma rede neural com estrutura *feedforward*) e $e(t)$ é o ruído (MELEIRO, 2002).

Alternativamente, o modelo pode ser escrito na forma de preditor (um passo adiante): (MELEIRO, 2002).

$$\hat{y}(t|\boldsymbol{\theta}) = g[\boldsymbol{\varphi}(t,\boldsymbol{\theta}),\boldsymbol{\theta}] \tag{3.16}$$

3.3.3.2.2 Predição Um Passo Adiante e Simulação Recursiva

A fim de esclarecer os conceitos de predição um passo adiante e simulação recursiva, considere-se o seguinte modelo cujo conjunto de regressores é composto de várias tomadas até o instante $(k-1)$:

$$y(k) = \boldsymbol{\psi}^T(k-1)\hat{\boldsymbol{\theta}} + \boldsymbol{\xi}(k) \tag{3.17}$$

Onde $\boldsymbol{\xi}$ é o vetor de resíduos definido como :

$$\boldsymbol{\xi} = y - \boldsymbol{\psi}\hat{\boldsymbol{\theta}} \tag{3.18}$$

A predição um passo adiante é obtida montando-se o vetor de regressão $\boldsymbol{\psi}$ com as observações (dados reais) obtidas do conjunto de dados até o instante $(k-1)$ e então calculando a predição da variável de interesse no instante k (um passo adiante):

$$\hat{y}(k) = \boldsymbol{\psi}^T(k-1)\hat{\boldsymbol{\theta}} \tag{3.19}$$

Para obter a predição no instante $(k+1)$ tomam-se as observações do conjunto de dados (ou da própria planta) até o instante k, ou seja, $\hat{y}(k+1) = \boldsymbol{\psi}^T(k)\hat{\boldsymbol{\theta}}$, e assim sucessivamente. (MELEIRO, 2002)

Em simulações deste tipo, a predição um passo adiante gerada pelo modelo, $\hat{y}(k)$, não é utilizada na próxima predição, $\hat{y}(k+1)$. O erro de predição, $y(k) - \hat{y}(k)$, é definido no instante k que, por sua vez, é utilizado pela maioria dos algoritmos de estimação de parâmetros como argumento da função de custo (minimização do quadrado dos resíduos). Considerando tais algoritmos, para um determinado conjunto de regressores, os erros de predição de um passo serão minimizados. Desse modo, predições deste tipo não são um bom teste para validar modelos dinâmicos, dado que mesmo modelos ruins costumam apresentar bons resultados em simulações um passo adiante (AGUIRRE, 2000; MELEIRO, 2002).

Pode-se visualizar uma rede simples, de predição um passo adiante, na Figura 3.9.

Considere-se agora o caso em que o modelo a ser simulado tem o seguinte vetor de regressores:

$$\boldsymbol{\psi}^T(k-1) = [y(k-1)\ y(k-2)\ u(k-1)\ u(k-2)] \tag{3.20}$$

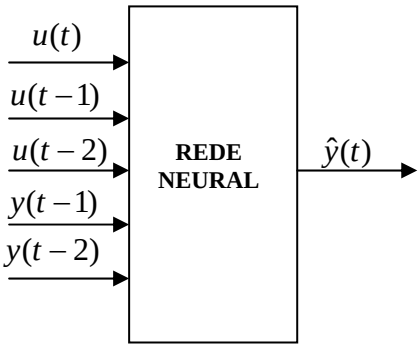

Figura 3.9 Ilustração esquemática de rede neural com predição um passo adiante

Para iniciar a simulação é preciso iniciar o modelo com valores medidos (reais). Sejam $y(1)$ e $y(2)$ os dois primeiros valores da saída do conjunto de dados e, similarmente $u(1)$ e $u(2)$ correspondem aos seus respectivos valores de entrada. Tem-se então que

$$\hat{y}(3) = [y(2)\ y(1)\ u(2)\ u(1)]\hat{\boldsymbol{\theta}} \tag{3.21}$$

As predições seguintes são feitas utilizando os valores preditos pelo próprio modelo em um procedimento recursivo:

$$\begin{aligned} \hat{y}(4) &= [\hat{y}(3)\ y(2)\ u(3)\ u(2)]\hat{\boldsymbol{\theta}} \\ \hat{y}(5) &= [\hat{y}(4)\ \hat{y}(3)\ u(4)\ u(3)]\hat{\boldsymbol{\theta}} \end{aligned} \tag{3.22}$$

Este procedimento, ao contrário da predição um passo adiante, fornece uma boa medida da capacidade do modelo de representar o comportamento dinâmico do sistema. A Figura 3.10 ilustra a simulação recursiva.

Existe ainda um preditor intermediário, k passos adiante, que utiliza o modelo como preditor recursivo apenas durante k intervalos de amostragem, depois dos quais o modelo é reiniciado com dados medidos.

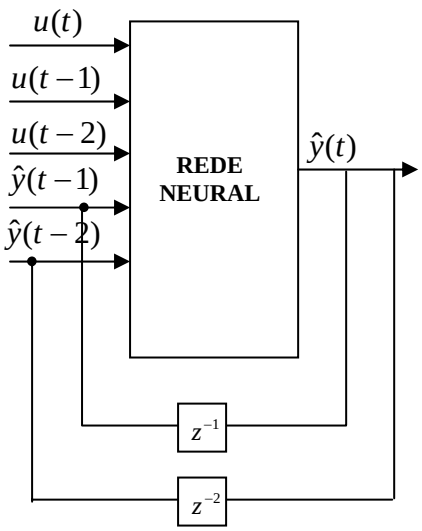

Figura 3.10 Ilustração esquemática de rede neural com simulação recursiva

3.3.3.3 Treinamento da Rede

A determinação do número de amostras e da forma pela qual o mapeamento deve ser realizado é o passo mais importante para que se atinja o sucesso no processo de treinamento da rede neural. Mapeamentos bastante complexos podem ser aproximados por uma rede neural mas, para que se obtenha uma generalização precisa, é necessário que se consiga captar a estrutura do sistema não linear pelas conexões da rede.

A maioria dos modelos de redes neurais possui alguma *Regra de Treinamento*, onde os pesos de suas conexões são ajustados de acordo com os padrões apresentados. Em outras palavras, as redes têm a capacidade de "aprender" através de exemplos. Esta habilidade de aprender a partir de seu ambiente e melhorar seu desempenho através da aprendizagem é uma das

principais propriedades das redes neurais. Uma rede neural aprende acerca do seu ambiente através de um processo interativo de ajustes aplicados a seus pesos sinápticos e níveis de bias.

Denomina-se *algoritmo de aprendizado* um conjunto de regras bem definidas para a solução de um problema de aprendizado. Existem muitos tipos de algoritmos de aprendizado específicos para determinados modelos de redes neurais, estes algoritmos diferem entre si principalmente pelo modo como os pesos são modificados.

Outro fator importante é a maneira pela qual uma rede neural se relaciona com o ambiente. Nesse contexto existem os seguintes paradigmas de aprendizado:

1) *Aprendizado Supervisionado*, quando é utilizado um agente externo que indica à rede a resposta desejada para o padrão de entrada;
2) *Aprendizado Não Supervisionado* (auto-organização), quando não existe uma agente externo indicando a resposta desejada para os padrões de entrada;
3) *Reforço*, quando um crítico externo avalia a resposta fornecida pela rede.

O treinamento supervisionado de redes neurais artificiais com várias camadas envolve uma etapa crucial – a determinação dos pesos das conexões sinápticas – que pode ser vista como um problema de otimização não linear irrestrita, onde uma função do erro global é minimizada através do ajuste dos parâmetros (pesos) da rede neural.

Denomina-se *ciclo* uma apresentação de todos os *N* pares (entrada e saída) do conjunto de treinamento no processo de aprendizado. A correção dos pesos em um ciclo pode ser executada de dois modos:

1) *Modo Padrão*: A correção dos pesos acontece a cada apresentação à rede de um exemplo do conjunto de treinamento. Cada correção de pesos baseia -se somente no erro do exemplo apresentado naquela iteração. Assim, em cada ciclo ocorrem *N* correções.
2) *Modo Batelada*: Apenas uma correção é feita por ciclo. Todos os exemplos do conjunto de treinamento são apresentados à rede, seu erro médio é calculado e a partir deste erro fazem-se as correções dos pesos.

3.3.3.3.1 As redes MLP e o algoritmo backpropagation

O modelo de RNA mais utilizado atualmente são as redes MLP (multi-layer percetron) treinadas com o algoritmo *backpropagation* (retropropagação). Este algoritmo de treinamento foi desenvolvido Rummelhart, Hinton e Williams, em 1986, e mostrou ser possível treinar eficientemente redes com camadas intermediárias (GONZAGA, 2003).

Nas redes MLP, cada camada tem uma função específica. A camada de saída recebe os estímulos da camada intermediária e *constrói o padrão que será a resposta*. As camadas intermediárias funcionam como *extratoras de características*, seus pesos são uma codificação de características apresentadas nos padrões de entrada e permitem que a rede crie sua própria representação, mais rica e complexa, do problema. Se existirem as conexões certas entre as unidades de entrada e um conjunto suficientemente grande de unidades intermediárias (neurônios), pode-se sempre encontrar a representação que irá produzir o mapeamento correto da entrada para a saída através das unidades intermediárias. Também foi provado que uma rede neural do tipo MLP com apenas uma camada

intermediária, contendo um número suficiente de neurônios, é capaz de aproximar qualquer função contínua e limitada (MELEIRO, 2002).

Durante o treinamento com o algoritmo *backpropagation*, a rede opera em uma seqüência de dois passos. No primeiro passo (Figura 3.11), um padrão é apresentado à camada de entrada da rede. A atividade resultante flui através da rede, camada por camada, até que a resposta seja produzida pela camada de saída. Durante a propagação todos os pesos sinápticos são fixos (HAYKIN, 2001). No segundo passo, a saída obtida é comparada à saída desejada para esse padrão particular. Se o valor da saída calculada não estiver correto, o erro é calculado. Este erro é então propagado a partir da camada de saída até a camada de entrada (daí o nome “backpropagation”), e os pesos das conexões dos neurônios da camada interna vão sendo modificados conforme o erro é retropropagado. Os pesos sinápticos são ajustados para fazer com que a resposta da rede se mova para mais perto da resposta desejada, em sentido estatístico (HAYKIN, 2001).

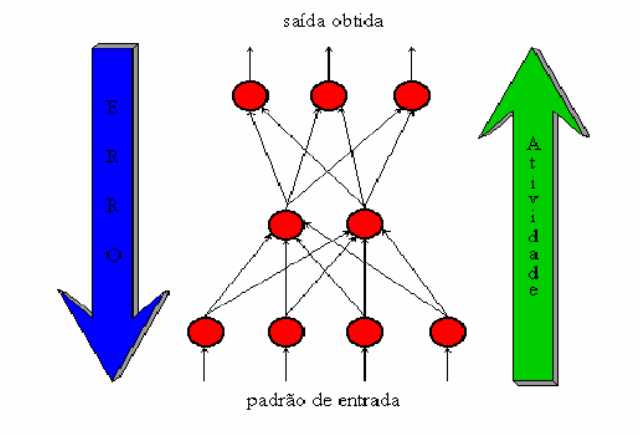

Figura 3.11 Esquema de treinamento backpropagation

Fonte: GONZAGA, 2003 pg. 64

3.3.3.4 Descrição do Procedimento e dos Critérios

O procedimento de modelagem neural envolve cinco etapas básicas: a coleta e a classificação de dados; a configuração da estrutura da rede; o treinamento da rede neural e a análise da confiabilidade do modelo e o uso da rede.

A primeira etapa envolve a coleta e o tratamento de dados referentes às variáveis diretamente envolvidas no processo. O objetivo é obter um histórico representativo de todas as condições operacionais passíveis de ocorrer numa planta. A seguir os dados devem ser separados em conjunto de treinamento e conjunto de teste. Esta última tarefa requer uma análise cuidadosa, para minimizar ambigüidades e erros nos dados (NØRGAARD *et al*., 2000).

Os dados coletados devem ser significativos e cobrir amplamente o domínio do problema; não cobrir apenas as operações normais e rotineiras, mas também exceções e as condições limites do domínio do processo. Um ponto de destacada importância está relacionado à confiabilidade das medidas que serão apresentadas à RNA. Uma rede treinada com dados oriundos de medidas incertas produzirá respostas de má-qualidade, uma vez que a rede aprenderá os erros (NØRGAARD *et al*., 2000).

Os dados, depois de separados, nos conjuntos de treinamento e de teste (utilizados para verificar o desempenho da rede sob condições reais e que não pertencem ao conjunto de treinamento) são, geralmente, colocados em ordem aleatória para a prevenção de tendências associadas à ordem de apresentação dos dados. Além disso, é necessário pré-processar estes dados através de normalizações, escalonamentos e/ou conversões de formato para torná-los mais apropriados à sua utilização na rede. Dados com ordem de grandeza discrepante podem causar problemas de convergência para a rede (NØRGAARD *et al*., 2000).

A terceira etapa é a definição da configuração da rede, que pode ser dividido em três passos (NØRGAARD *et al*., 2000):

1) Seleção do paradigma neural apropriado à aplicação. Embora este trabalho tenha apresentado apenas o modelo MLP (modelo selecionado para a execução deste trabalho), existem inúmeros outros modelos;
2) Determinação da topologia da rede: número de camadas e número de neurônios em cada camada;
3) Determinação do algoritmo de treinamento e funções de ativação. Este é um dos aspectos de maior impacto no desempenho do sistema resultante.

Existem metodologias para a condução destas tarefas. Normalmente estas tarefas são feitas de forma empírica, uma fez que a definição da configuração da rede neural é ainda considerada uma arte. Arquitetar a rede pode demandar bastante tempo, requerer experiência dos projetistas sendo interessante um conhecimento profundo do processo a ser modelado pela rede neural (NØRGAARD *et al*., 2000).

A quarta etapa do processo de modelagem é o treinamento da rede. Nesta etapa, os pesos das conexões são ajustados de acordo com os critérios do algoritmo de treinamento escolhido (NØRGAARD *et al*., 2000).

A etapa de teste da rede tem início após o treinamento da rede. A validação do modelo é dividida em duas partes: i) validação da rede com os dados do histórico do processo previamente separado do arquivo de dados inicial; ii) testes com dados atuais do processo (NØRGAARD *et al*., 2000).

Durante a fase de teste da rede, o conjunto de teste é utilizado para determinar o desempenho da rede com dados que não foram previamente utilizados.

O desempenho da rede medido nesta fase é uma boa indicação de seu desempenho real. Devem, ainda, serem procedidos testes, como análise de comportamento da rede utilizando entradas relacionadas a condições especiais de operação e análise dos pesos atuais da rede. Se existirem valores muito pequenos, as conexões associadas podem ser consideradas insignificantes e serem descartadas (poda da rede). Em contrapartida, valores substancialmente maiores que os outros poderiam indicar que houve sobreajuste (*over-training*) da rede (NØRGAARD *et al*., 2000).

Redes Neurais Artificiais de processamento numérico, arquitetura em camadas e fluxo de informação com e sem realimentação, produzem estruturas de processamento de sinais com grande poder de adaptação e capacidade de representação não linear, além disso, são capazes de tratar dados com ruídos, típicos de situações industriais. A presença de realimentação cria a possibilidade de armazenar informações na forma de representações internas, além de introduzir dinâmica no processo. Sempre que métodos tradicionais, derivados da teoria de sistemas lineares, não apresentarem bom desempenho no tratamento de problemas envolvendo, por exemplo, aproximação de funções multivariáveis ou identificação e controle de sistemas dinâmicos, as RNA do tipo descrito acima despontam como alternativas bastante competitivas. Tais características devem-se principalmente à sua capacidade de representação de comportamentos não lineares arbitrários.

Redes neurais de várias camadas têm sido utilizadas em diversas aplicações tais como o reconhecimento de padrões e identificação de sistemas não lineares através da aproximação funcional. Teoricamente, a aplicação de redes neurais para aproximação funcional pode ser analisada utilizando-se o teorema de Wierstrass (VON ZUBEN, 1996). Basicamente, este teorema afirma que qualquer função contínua

de valores reais, definida em um intervalo limitado, pode ser aproximada por um polinômio. Se a função de ativação de cada elemento da rede for uma função contínua de valor real, ela também poderá ser aproximada por um polinômio e, conseqüentemente, a relação funcional de entrada-saída da rede poderá também ser aproximada por um polinômio. Assim sendo, será sempre possível definir uma rede neural de múltiplas camadas para atuar como aproximação de um sistema não linear específico.

3.4 ESTIMAÇÃO DE MODELOS NEURAIS

O processo de seleção de um modelo a partir de uma estrutura previamente determinada é conhecido na literatura estatística como estimação (MELEIRO, 2002). Ao se tratar de redes neurais, no entanto, é tratado como treinamento ou aprendizado (NØRGAARD *et al*., 2000).

Escolhido um conjunto de modelos candidatos, escolhe-se um modelo particular deste conjunto. Esta escolha é feita, normalmente, baseada em algum critério de desempenho, que deve estar relacionado com a futura aplicação do modelo. A estratégia mais freqüente é escolher o modelo que exibe o melhor desempenho em predições um passo adiante em termos de menor erro quadrático médio entre as saídas observadas e as predições do modelo. Todavia, este critério pode não ser adequado para aplicações em estratégias de controle avançado, onde o desempenho desejado é não apenas a melhor predição um passo adiante, mas também predições com a maior precisão possível em um horizonte relativamente grande (predições k passos adiante). A necessidade de modelos que

forneçam bons resultados em simulações recursivas está intimamente relacionada com a filosofia dos controladores preditivos (MELEIRO, 2002).

Segue-se algumas das abordagens mais utilizadas para o treinamento de redes neurais.

3.4.1 Modelagem Direta

Seja uma planta dinâmica discreta, casual, invariante no tempo cuja entrada, $u(\bullet)$ é uma função do tempo uniformemente limitada e cuja saída é representada por $y_p(\bullet)$. A planta é considerada estável, com parametrização conhecida mas com valores dos parâmetros desconhecidos. O objetivo da identificação é construir um modelo neural adequado, de modo que, quando submetido à mesma entrada u da planta, forneça uma saída y_m segundo um determinado critério. Este processo de identificação está representado graficamente na Figura 3.12 (MELEIRO, 2002).

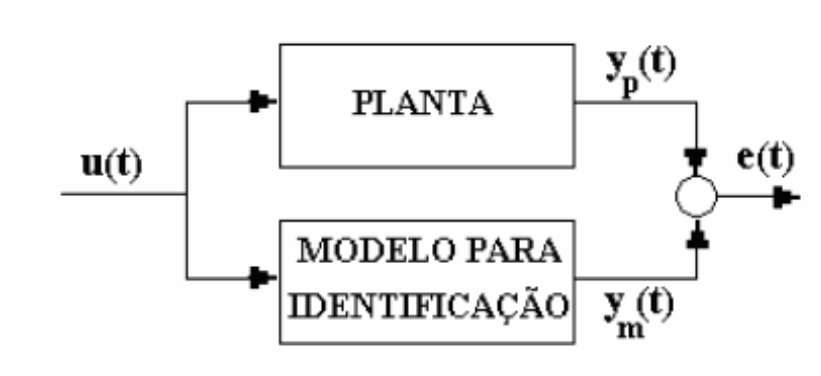

Figura 3.12 Identificação de uma planta usando RNA

Fonte: MELEIRO, 2002, pg. 96

O treinamento de uma rede neural pode representar a dinâmica de uma planta e é conhecido como modelagem direta. O modelo neural é colocado em paralelo com a planta e os erros entre as saídas do sistema e as da rede neural (os erros de predição) são utilizados como sinal para o treinamento da rede (MELEIRO, 2002).

Considere-se uma planta descrita pela seguinte equação:

$$y_p(t+1) = F[y_p(t),...,y_p(t-n+1);u(t),...,u(t-m+1)] \tag{3.23}$$

A saída da planta y_p no tempo $(t+1)$ depende dos n valores passados da própria saída e dos m valores passados da entrada u. Neste ponto, somente a parte dinâmica da resposta da planta será considerada; o modelo não representa explicitamente as possíveis perturbações agindo sobre a planta (MELEIRO, 2002)

A abordagem mais comum para a modelagem de sistemas é escolher a estrutura de entrada-saída da rede neural idêntica à do sistema que se deseja modelar:

$$y_m(t+1) = \hat{F}[y_p(t),...,y_p(t-n+1);u(t),...,u(t-m+1)] \tag{3.24}$$

sendo y_m a saída da rede neural, $\hat{F}(\bullet)$ representando o mapeamento não linear entrada-saída realizado pela rede neural para aproximar o mapeamento $F(\bullet)$ da planta. Note-se que as entradas da rede neural incluem os valores passados da saída da planta e não os valores passados da saída da rede (a rede não possui realimentação – *feedback*) (MELEIRO, 2002).

Considerando-se que após um tempo adequado de treinamento a rede neural é capaz de fornecer uma boa representação da planta, então a própria saída da rede neural e seus valores atrasados podem ser realimentados e, neste caso, a rede neural pode ser utilizada independente da planta (MELEIRO, 2002). Tal rede é descrita por:

$$y_m(t+1) = \hat{F}[y_p(t),...,y_m(t-n+1);u(t),...,u(t-m+1)] \tag{3.25}$$

Esta estrutura (3.24) também pode ser usada desde o início, ou seja, durante todo o processo de aprendizagem (NARENDRA E PARTHASARATHY, 1990 *apud* MELEIRO, 2002).

3.4.2 Modelagem Inversa

Nesta abordagem, o modelo do sistema dinâmico fornece uma entrada para que a saída desejada seja obtida. Tais modelos desempenham um papel fundamental em estruturas neurais com propósito de controle. Conceitualmente a abordagem mais simples é a modelagem direta-inversa mostrada na Figura 3.13, onde um sinal de treinamento sintético (a entrada da planta) é introduzido no sistema. A saída da planta é utilizada como entrada para a rede neural e a saída desta é então comparada com o sinal de treinamento (a entrada do sistema). O erro gerado é utilizado para ajustar os parâmetros da rede neural através de um algoritmo de treinamento.

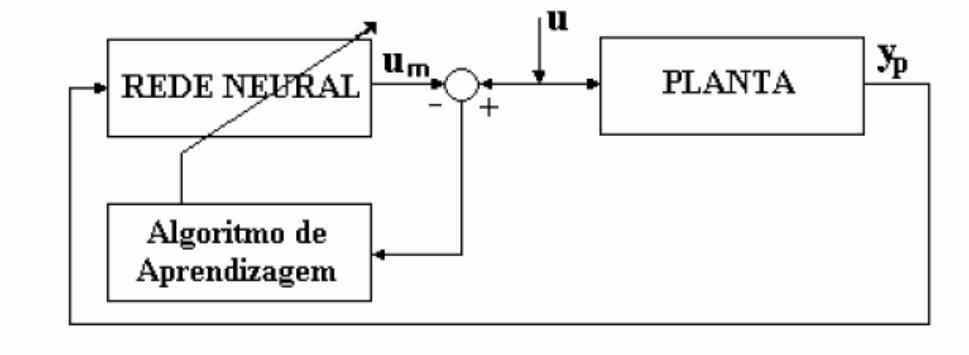

Figura 3.13 Modelagem neural direta-inversa

Fonte: MELEIRO, 2002, pg. 100

Uma segunda abordagem é conhecida como aprendizado inverso especializado (NØRGARD *et a*l., 2000 *apud* MELEIRO, 2002), cuja estrutura está representada na (Figura 3.14). Nesta abordagem o modelo neural inverso precede o sistema e recebe como entrada um sinal de treinamento que cobre o espaço operacional desejado de saída do sistema controlado (isto é, corresponde ao sinal de referência do sistema). Esta estrutura de aprendizado também conta com um modelo neural direto do sistema colocado em paralelo com a planta. Neste caso, o sinal de erro para o algoritmo de treinamento é a diferença entre o sinal de treinamento e a saída do sistema (se o sistema for ruidoso, o erro também pode ser definido como sendo a diferença entre o sinal de treinamento e a saída do modelo neural direto). O erro pode ser retropropagado através do modelo neural direto e então através do modelo neural inverso. Durante este procedimento, somente os pesos do modelo neural inverso são ajustados.

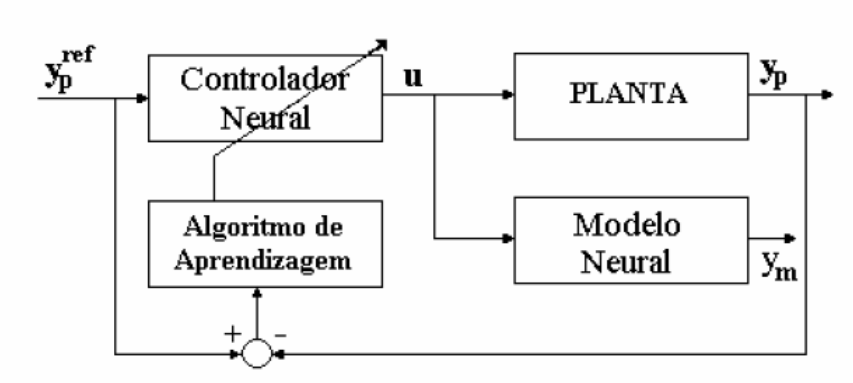

Figura 3.14 Modelagem inversa especializada

Fonte: MELEIRO, 2002 pg. 100

Analisando a estrutura entrada-saída da modelagem neural inversa, verifica-se que a determinação da inversa F^{-1} que leva à obtenção de $u(t)$ a partir da equação (3.22), requer o conhecimento do valor futuro $y_p(t+1)$. Para superar este problema, pode-se substituir este valor por $y_p^{ref}(t+1)$ que se admite estar disponível no tempo t. Esta consideração é bastante razoável uma vez que y_p^{ref} está relacionado com o sinal de referência que é, normalmente, conhecido de previamente. Assim, o mapeamento não linear entrada-saída da modelagem neural é dado por (MELEIRO, 2002):

$$u(t) = F^{-1}[y_p(t),...,y_p(t-n+1);\ y_p^{ref}(t+1);\ u(t-1),...,u(t-m+1)] \tag{3.26}$$

3.5 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

Neste capítulo, foram apresentados alguns tópicos em identificação de sistemas, dando ênfase especial na utilização de redes neurais para este procedimento. As principais etapas da identificação de processos foram brevemente comentadas aqui. Apresentou-se uma breve revisão bibliográfica sobre o uso de redes neurais. Em especial, as redes chamadas MLP foram mais exploradas, devido ao fato de terem sido utilizadas na identificação do processo em estudo neste trabalho. É provável que a vantagem mais interessante do uso de redes neurais encontrada neste trabalho seja a relativa facilidade de identificação do modelo (ajuste dos parâmetros) quando ocorrem mudanças nas condições de operação causadas, por exemplo, pela mudança no tipo de carga (ocorrência bastante comum no país).

Capítulo 4 - Controle Preditivo Baseado em Modelo

Este capítulo apresenta uma visão geral sobre o desenvolvimento e as aplicações do Controle Preditivo baseado em Modelo (MPC) na indústria química. A fundamentação teórica do MPC é apresentada na seção 4.2 e a seção 4.3 discute a aplicação de modelos neurais no projeto de controladores preditivos.

4.1 INTRODUÇÃO

Segundo Henson e Seborg (1997, *apud* MELEIRO 2002), o controle de sistemas não lineares vem recebendo grande atenção, tanto da indústria como da universidade, desde o início da década de 90. Este interesse pelo projeto de controladores não lineares deve-se a vários fatores e, segundo estes autores, o principal é o desempenho inadequado apresentado pelos controladores lineares quando utilizados em sistemas fortemente não lineares ou mesmo a sistemas moderadamente não lineares que operam em uma faixa ampla de condições.

Um dos recentes avanços na teoria de controle tem sido a inclusão de modelos preditivos no projeto de sistema de controle regulatório. Este tipo de controle pode ser encontrado na literatura sob vários nomes: Controle por Matriz Dinâmica (*Dynamic Matrix Control* – DMC), Controle por Modelo Interno (*Internal Model Control* – IMC), Controle por Modelo Algorítmico (*Model Algorithmic Control* - MAC) e, propriamente, Controle Preditivo Baseado em Modelo (*Model Predictive Control* – MPC) (RAMIREZ, 1994). O MPC possui muitas vantagens sobre o PID convecional porque é baseado em técnicas de otimização e modelos dinâmicos do processo. Tem a habilidade de lidar com restrições de processo, *time-delays*, respostas

inversas e acoplamentos. Além disso, não é necessário que se encontre a melhor estrutura de controle para que se consiga a melhor performance de controle (HAN *et al*., 2002)

O conceito básico da estratégia de controle preditivo é que um modelo dinâmico do processo e as medidas de interesse são utilizados para predizer o comportamento futuro do processo. Foi originalmente desenvolvido para prover o controle de processos em refinarias de petróleo e é, hoje, aplicado nas mais diferentes áreas. (QIN e BADGWEL, 1996)

As ações de controle são obtidas minimizando uma função do erro entre a resposta predita do processo e a trajetória desejada (*set-point*). A estratégia de controle MPC foi desenvolvida para tratar de controle multivariáveis complexos onde existam interações significativas entre as entradas (variáveis manipuladas) e as saídas (variáveis controladas). Uma grande vantagem da estratégia MPC é a sua capacidade de incorporar restrições nas variáveis de processo diretamente no processo do controlador (QIN e BADGWEL, 1996; SEBORG, 1999).

4.2 A ESTRATÉGIA DO CONTROLE PREDITIVO

A figura 4.1 ilustra o princípio geral do controle preditivo. No instante de amostragem k, o algoritmo de controle recebe informações sobre o estado atual do sistema e, baseado nesta informação e no modelo do processo, calcula as ações de controle futuras para que a saída do processo siga as referências. A trajetória de entradas futuras é calculada otimizando algum critério de desempenho e, no algoritmo preditivo, somente a primeira ação de controle é implementada até a

próxima amostragem, quando todo o processo é repetido baseado nas novas informações medidas.

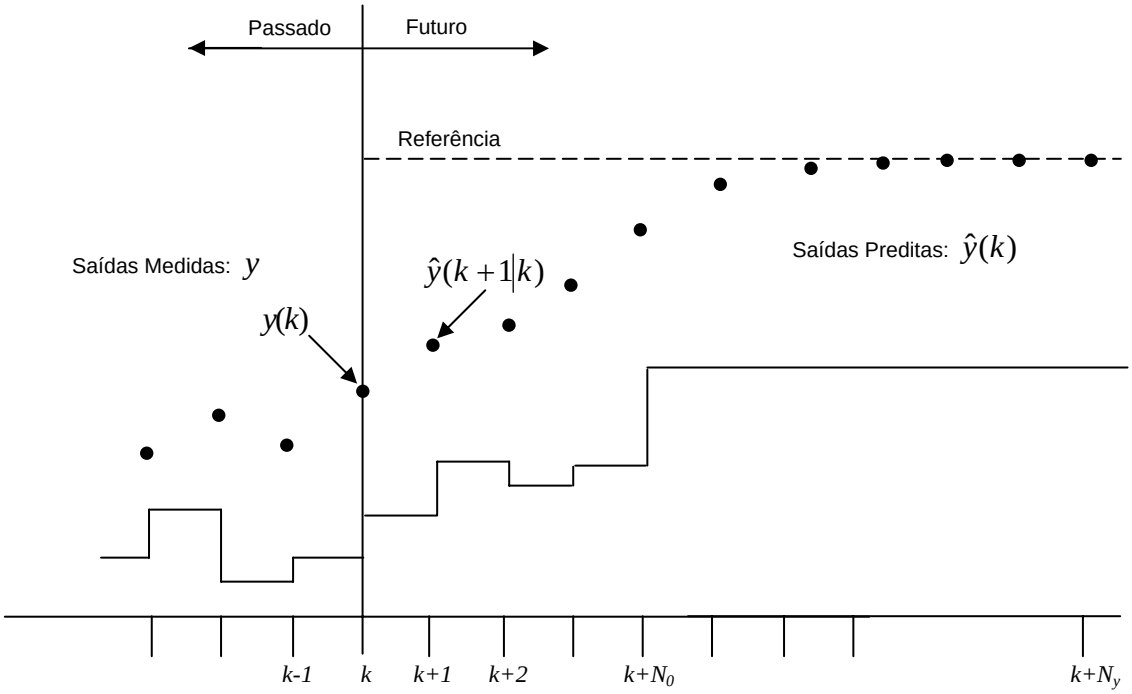

Figura 4.1 Representação esquemática dos elementos básicos do controle preditivo

Fonte: MELEIRO, 2002, pg. 122

Embora a figura 4.1 represente a essência do controle preditivo, esta abordagem pode ser melhor compreendida separando-a em duas partes. As figuras 4.2 e 4.3 representam o modo como os controladores preditivos atuam sobre um sistema SISO. As escalas de tempo localizadas no fundo destas figuras representam a escala de tempo absoluto, enquanto as escalas de tempo relativas ao instante de amostragem, k, denotam o presente (SOETERBOEK, 1992 *apud* MELEIRO, 2002). Desse modo, as figuras 4.2 e 4.3 estão dispostas em ordem cronológica.

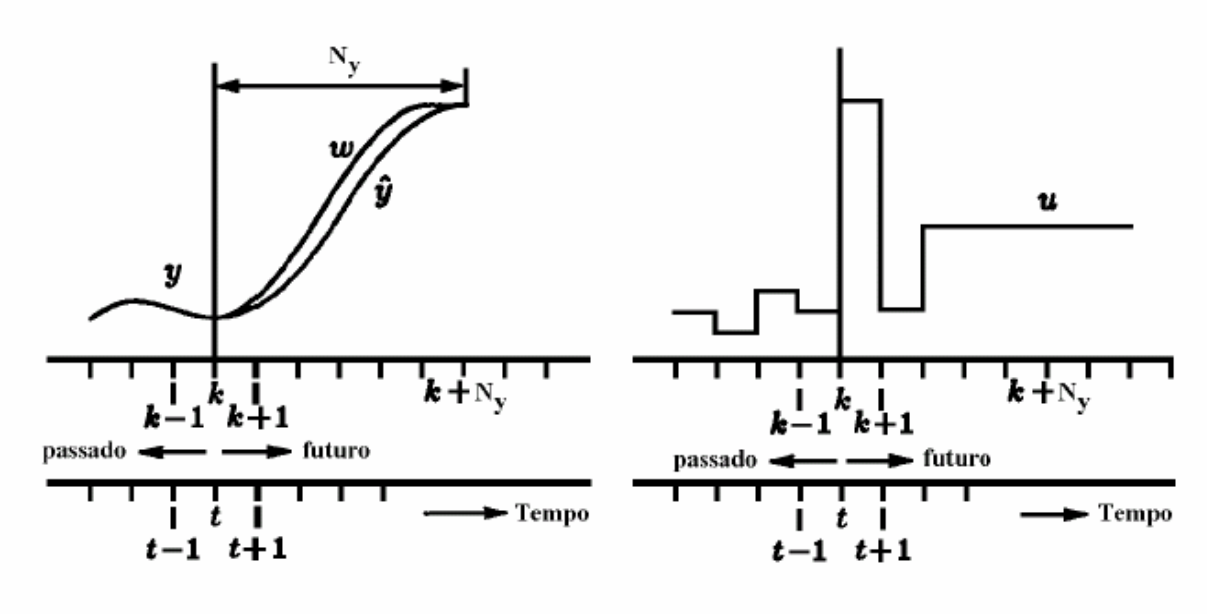

Figura 4.2 Representação do princípio do horizonte móvel: Situação no tempo *t*

Fonte: Meleiro, 2002, pg. 125

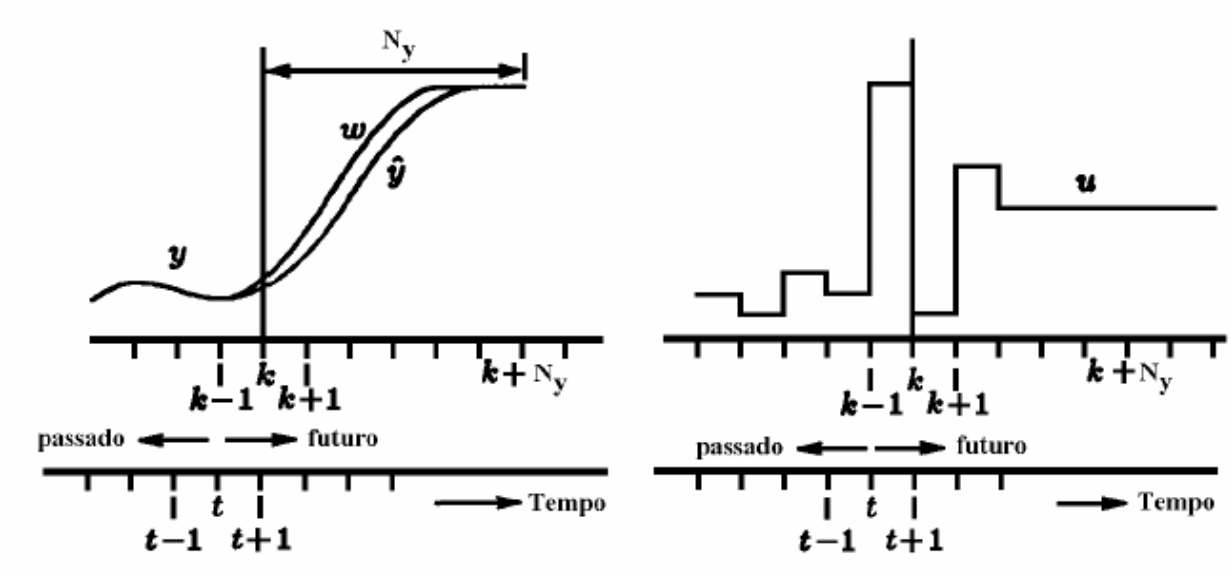

Figura 4.3 Representação do princípio do horizonte móvel: Situação no tempo *t*+1

Fonte: Meleiro, 2002, pg. 125

As variáveis *u(k), y(k) e w(k)* são definidas como sendo a saída do controlador (ação de controle), saída do processo e saída desejada do processo (*set-point*) no instante de amostragem, k, respectivamente. Os seguintes vetores podem então ser definidos:

$$\boldsymbol{u} = [u(k), \ldots, u(k + N_y - 1)]^T$$

$$\hat{\boldsymbol{y}} = [\hat{y}(k), \ldots, \hat{y}(k + N_y)]^T$$

$$\hat{\boldsymbol{w}} = [w(k), \ldots, w(k + N_y)]^T$$

onde N_y é o horizonte de predição e o símbolo ^ de nota uma variável estimada.

Considere inicialmente a
Figura 4.2, onde o tempo de amostragem atual, *k*, corresponde ao tempo absoluto, t. O controlador preditivo calcula a seqüência de ações de controle futura, $\boldsymbol{u}$, para que a saída predita do processo, $\hat{\boldsymbol{y}}$, se aproxime da saída desejada $\hat{\boldsymbol{w}}$. Contudo, no lugar de se utilizar toda a seqüência de ações de controle $\boldsymbol{u}$, calculada pelo algoritmo de otimização para controlar o processo nos próximos N_y instantes de amostragem, somente o primeiro elemento desta seqüência, $u(k)$, é utilizado para controlar o processo. No próximo instante de amostragem (Figura 4.3), todo o procedimento é repetido utilizando as últimas informações medidas do processo. Esta estratégia é conhecida como “horizonte móvel” (*receding horizon*). Assumindo que não há perturbações nem erros de modelagem, a saída do processo $\hat{y}(k+1)$, predita no tempo t é exatamente igual á saída do processo, $\hat{y}(k)$, medida no tempo (t+1). Outra seqüência de ações de controle é então calculada de modo que a saída predita do processo se aproxime da saída de referência e assim sucessivamente (SOETERBOEK, 1992 *apud* MELEIRO, 2002).

A principal razão para a utilização do horizonte móvel é a possibilidade que esta estratégia oferece para a correção de erros de modelagem e/ou devido a

perturbações futuras. Assim, se a saída predita do processo no tempo t, $\hat{y}(k+1)$, não for igual à saída medida do processo no tempo (t+1), $\hat{y}(k)$, esta diferença será corrigida através de um mecanismo *feedback*. Além disso, no tempo (t+1), as próximas predições serão feitas a partir da saída medida do processo e não a partir da saída predita no instante anterior. O resultado da aplicação da abordagem do horizonte móvel é que o horizonte sobre o qual a saída do processo é predita é deslocado em um instante de amostragem na direção do futuro, a cada instante de amostragem (SOETERBOEK, 1992 *apud* MELEIRO, 2002).

Uma maneira de se determinar a qualidade da predição fornecida pelo modelo do processo, ou seja, o quanto a saída predita se aproxima da saída desejada, é através de um critério que seja função de $\hat{\boldsymbol{y}}$, $\boldsymbol{w}$ e $\boldsymbol{u}$. Uma função geralmente utilizada é do tipo:

$$J = \sum_{i=1}^{N_y} (\hat{y}(k+i) - w(k+i))^2 \qquad (4.1)$$

e a seqüência ótima de ações de controle $\underline{u}_{opt}$, sobre o horizonte de predição é obtida através da minimização de J em relação a $\underline{u}$:

$$\boldsymbol{u}_{opt} = \arg\min_{\boldsymbol{u}} J \qquad (4.2)$$

Assim, a seqüência $\boldsymbol{u}_{opt}$ é ótima em relação à função de custo que é minimizada e, conseqüentemente, o erro de predição é minimizado (SOETERBOEK, 1992 *apud* MELEIRO, 2002).

O cálculo da seqüência das ações de controle é um problema de otimização ou, mais especificamente, de minimização, cuja solução geralmente requer um procedimento iterativo. Contudo, quando o critério é quadrático, o modelo é linear e não existem restrições, a solução é analítica.

4.3 CONTROLE PREDITIVO NÃO LINEAR BASEADO EM MODELO NEURAL

4.3.1 Introdução

O sucesso da tecnologia MPC como um paradigma de controle de processo se deve a três importantes fatores. O primeiro e mais importante é a incorporação de um modelo explícito do processo no controle. Isto permite que o controlador leve em conta todos os aspectos significantes da dinâmica do processo, diretamente. O segundo fator que o algoritmo MPC considera é o comportamento da planta sobre um horizonte futuro no tempo, ou seja, que os efeitos de perturbações *feedforward* e *feedback* podem ser antecipados e removidos, permitindo ao controlador conduzir a planta com menos “folga” por uma trajetória futura desejada. Finalmente, o MPC considera as entradas do processo, estado e restrições de saída diretamente no cálculo do controle. Isto significa que violações nas restrições são menos prováveis, resultando num controle mais justo. O que mais claramente distingue o MPC de outros paradigmas de controle de processo é a inclusão de restrições. (TYAGUNOV, 2004)

Embora processos industriais sejam inerentemente não lineares, a maioria das aplicações do MPC até hoje é baseada em modelos dinâmicos lineares, sendo os mais comuns os modelos de resposta a degrau e impulso, gerando os controladores lineares conhecidos como DMC (TYAGUNOV, 2004). Há diversas razões para tanto. Modelos empíricos lineares podem ser identificados de modo direto dos dados do processo. Além disso, a maioria das aplicações até hoje têm sido feitas em refinarias de petróleo, onde o objetivo é manter o processo no estado estacionário (problema regulatório), no lugar de movê-lo rapidamente de um ponto

operacional para outro (problema servo). Um modelo de identificação linear é suficientemente exato nas proximidades de um determinado ponto de operação para tais aplicações. Com o uso de um modelo linear e uma função objetivo quadrática, o algoritmo MPC toma forma de um problema de programação quadrática convexa, para o qual encontram-se facilmente algoritmos que o resolvam. Isto é importante porque a solução do algoritmo deve convergir em não mais que dez segundos para ser aproveitável em aplicações industriais. Por estas razões, em muitos casos, um modelo linear provê a maioria das vantagens possíveis com a tecnologia MPC (TYAGUNOV, 2004).

Entretanto, há casos onde os efeitos não lineares são significativos o bastante para justificar o uso da tecnologia NMPC. Estes incluem pelo menos, duas grandes categorias de aplicações (TYAGUNOV, 2004):

- Problemas de controle regulatório onde os processos são altamente não lineares e sujeitos a freqüentes perturbações.
- Problemas de controle servo onde os pontos operacionais mudam freqüentemente.

Muitos processos são não lineares com vários graus de severidade. Apesar de que, em muitas situações o processo pode ser operado nas proximidades do estado estacionário e, por isso, uma representação linear pode ser adequada, há muitas situações onde isto não ocorre. Há processos para os quais há uma não linearidade tão severa (mesmo nas proximidades do estado estacionário) e é tão necessária a estabilidade da malha fechada, que um modelo linear não é suficiente. Há também processos que experimentam transições contínuas (*start-ups*, *shutdowns*, entre outros) e demoram bastante para alcançar a região operacional no

estado estacionário ou nunca a alcançam, como é o caso de processos em batelada, onde toda a operação se dá no estado transiente.

Para estes processos a lei do controle linear não é muito efetiva, sendo então necessários controles não lineares para aumentar a performance ou simplesmente para estabilizar a operação. Não há nada nos conceitos básicos do MPC que seja contra o uso de modelos não lineares. Por isso a extensão das idéias do MPC para os processos não lineares é direta, pelo menos, conceitualmente. Entretanto este não é um problema trivial e há várias questões abertas, como (TYAGUNOV, 2004):

- Avaliação de modelos lineares devido à carência de técnicas de identificação para processos não lineares.
- A complexidade computacional para solução de MPC de processos não lineares.
- A falta de estabilidade e robustez no caso de sistemas não lineares.

Alguns destes problemas são parcialmente solucionados e o MPC, com o uso de modelos não lineares, tem se tornado um campo de intensa pesquisa.

O desenvolvimento de modelos não lineares empíricos adequados pode ser bastante difícil e não há uma forma claramente mais adequada para representar genericamente processos não lineares. Parte do sucesso do MPC padrão é devido à facilidade com que respostas a impulso ou degrau ou outras funções de transferência de ordens mais baixas podem ser obtidas. A construção de modelos não lineares é muito mais difícil tanto de correlações de dados de entrada e saída quanto pelo uso das leis de conservação de massa e energia. O maior obstáculo matemático para uma teoria completa dos processos não lineares é a falta de um princípio de superposição para sistemas não lineares (TYAGUNOV, 2004). Por esta causa, a determinação de modelos advindos dos dados de entrada e saída do processo torna-se uma questão difícil. A quantidade de testes em planta requerida

para identificar uma planta não linear é muito maior que para um processo linear. O principal obstáculo na abordagem NMPC é a necessidade de resolver um problema de otimização dinâmica não linear em cada instante de amostragem (MELEIRO, 2002).

Existem basicamente três diferentes classes de NMPCs, que podem ser agrupadas como segue:

1) Extensões Não Lineares do Controle por Matriz Dinâmica – DMC e do Controle Quadrático por Matriz Dinâmica – QDMC. Existem várias contribuições nesta área, cujas abordagens diferem umas das outras basicamente em função do modelo ou do método de otimização utilizado.
2) Algoritmos NMPC com Critérios de Otimização Baseados no Método de Newton. Nas abordagens mais representativas desta classe de algoritmos a idéia principal é linearizar o modelo não linear em torno de uma trajetória de referência determinada pela seqüência de entradas calculada no instante da amostragem anterior (MELEIRO, 2002).
3) Algoritmos NMPC com Critérios de Otimização Baseados em Técnicas de Programação Não Linear: Esta abordagem contém técnicas de programação não linear com modelos não lineares para a resolução do problema de otimização. As principais diferenças são os métodos como as equações dos modelos são resolvidas e o método de otimização utilizado. Os algoritmos SQP (*Successive Quadratic Programming*) são os mais utilizados para resolver este tipo de problema de programação não linear e, embora estes algoritmos exijam mais esforço computacional quando comparado às outras técnicas não lineares descritas anteriormente, o modelo não linear pode ser utilizado diretamente no

algoritmo de otimização, sem que haja necessidade de linearização (QIN e BADGWEL, 1988 *apud* MELEIRO, 2002).

Os tipos de modelos entrada-saída não lineares estudados com mais freqüência para aplicações em estratégias de controle preditivo não linear são:

- Redes Neurais Artificiais;
- Modelos Nebulosos;
- Modelos de Volterra;
- Modelos polinomiais do tipo NARX.

Embora existam diferenças nas apresentações dos modelos não lineares, todos os NMPCs desta classe são muito similares em relação à formulação e à solução do problema de otimização (MELEIRO, 2002).

O controlador proposto neste trabalho foi projetado baseado na abordagem NMPC correspondente ao item 3, considerando-se o modelo neural obtido. A solução do problema de otimização foi obtida através do algoritmo SQP (Edgard e Hilmmelblau, 1988). A estrutura básica deste tipo de controlador está representada na Figura 4.4.

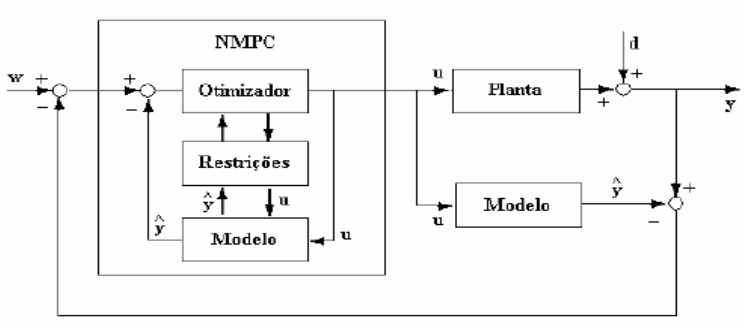

Figura 4.4 Estrutura do controlador preditivo não linear utilizado neste trabalho

4.3.2 Controle baseado em modelo neural

Em um controlador preditivo, os valores preditos das variáveis controladas são obtidos do modelo do processo. No controle preditivo baseado em modelo não linear (NMPC) as predições do comportamento futuro do sistema são usualmente obtidas pela integração do modelo analítico do processo, descrito por equações algébricas, lineares e não lineares, e diferenciais. No entanto, esta aproximação tem duas principais desvantagens comparadas com os métodos LMPC (*Linear Model Predictive Control*) (NAGY *et al*., 2000):

a) requer a elaboração de um complexo e preciso modelo analítico do processo o que, para a maioria dos processos químicos, é tarefa bastante árdua.

b) a resolução do problema de otimização por integração do modelo analítico em larga escala, pode demandar muito tempo e um grande esforço computacional.

Estas problemas podem ser evitados usando RNAs como o modelo não linear usado no controle (MELEIRO, 2002).

4.4 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

O Capítulo 4 apresentou uma visão geral do controle preditivo baseado em modelo, mostrando a estratégia utilizada neste tipo de controle. A ênfase do capítulo recaiu na utilização de modelos neurais no projeto de controladores preditivos de processo a fim de evitar alguns problemas que surgem neste procedimento, como a resolução do problema de otimização por integração do modelo analítico, por exemplo.

Capítulo 5 - O Processo de Craqueamento Catalítico em Leito Fluidizado

Apresenta-se neste capítulo a descrição do processo de craqueamento catalítico e o equacionamento matemático que deu origem ao modelo fenomenológico. Descreve-se aqui o modelo apresentado por Moro e Odloak (1995) referente a um reator Kellog, o mais comumente usado no país. O simulador rigoroso do processo, desenvolvido a partir do modelo fenomenológico, foi utilizado para gerar os dados de entrada e saída do processo, que serviram de base para a identificação por modelo neural apresentada nesta dissertação.

5.1 INTRODUÇÃO

O processo de craqueamento catalítico em leito fluidizado (FCC) é utilizado na produção de gasolina e gás liquefeito de petróleo (GLP) através da conversão de cortes pesados, provenientes da destilação do petróleo, em frações mais leves.

O craqueamento no conversor FCC é o maior processo catalítico do mundo pois envolve uma grande quantidade de material processado, grande dimensão dos equipamentos envolvidos e grande quantidade de catalisador. A unidade de craqueamento catalítico (FCCU) produz alguns dos produtos de maior valor agregado nas refinarias: a gasolina e o GLP.

O reator FCC é um equipamento de operação bastante complexa, pois envolve um processo explosivo, multi-variável, altamente não linear, com fortes interações entre as variáveis de processo e apresenta restrições operacionais nas variáveis de entrada e saída. Além disso, há um alto grau de incerteza na cinética da reação de craqueamento, na desativação do catalisador pela deposição de coque no *riser* e na queima deste no regenerador (ALARADI e ROHANI, 2002).

Pelo fato do processo envolver a produção de produtos estratégicos para o país e devido aos altos lucros gerados pela unidade FCC, a implementação de estratégias adequadas de identificação e controle deste processo é plenamente justificada.

Muitos autores têm demandado esforços substanciais para modelar o comportamento de uma unidade de FCC. Uma revisão detalhada pode ser encontrada no artigo de Arbel *et al.* (ARBEl *et al. apud* RAMIREZ *et al.*, 2004).

5.2 VISÃO GERAL DO PROCESSO

A unidade FCC considerada nesta dissertação está representada esquematicamente na Figura 5.1. Uma das seções mais importantes do sistema FCC é o *riser* (RR na Figura 5.1) onde ocorre a reação de craqueamento. A vazão de catalisador para o *riser* é controlada pela abertura (c_{TCV}) de uma válvula localizada na base do *riser* (TCV na Figura 5.1). A vazão (R_{tf}) e a temperatura (T_{fp}) da corrente de alimentação (gasóleo) do sistema também são manipuladas para o controle da operação do sistema reator–regenerador do FCC. A temperatura do *riser* (T_{rx}) deve ser controlada para garantir uma conversão adequada da reação de craqueamento. A conversão também pode ser representada pela severidade da

reação. Os produtos de reação estão em fase gasosa e são separados do catalisador em um vaso de separação (RX na Figura 5.1). Os produtos seguem então para uma torre de fracionamento (MF) e o catalisador vai para a terceira seção do sistema, chamada regenerador (RG - Figura 5.1), onde o catalisador é regenerado. O coque depositado na superfície do catalisador é queimado na presença de ar, cuja vazão (R_{ai}) pode ser manipulada. O regenerador tem dois estágios onde as temperaturas (T_{rg1},T_{rg2}) são muito altas. Assim, para evitar danos metalúrgicos no regenerador elas precisam ser mantidas dentro de uma faixa específica para o material usado no regenerador.

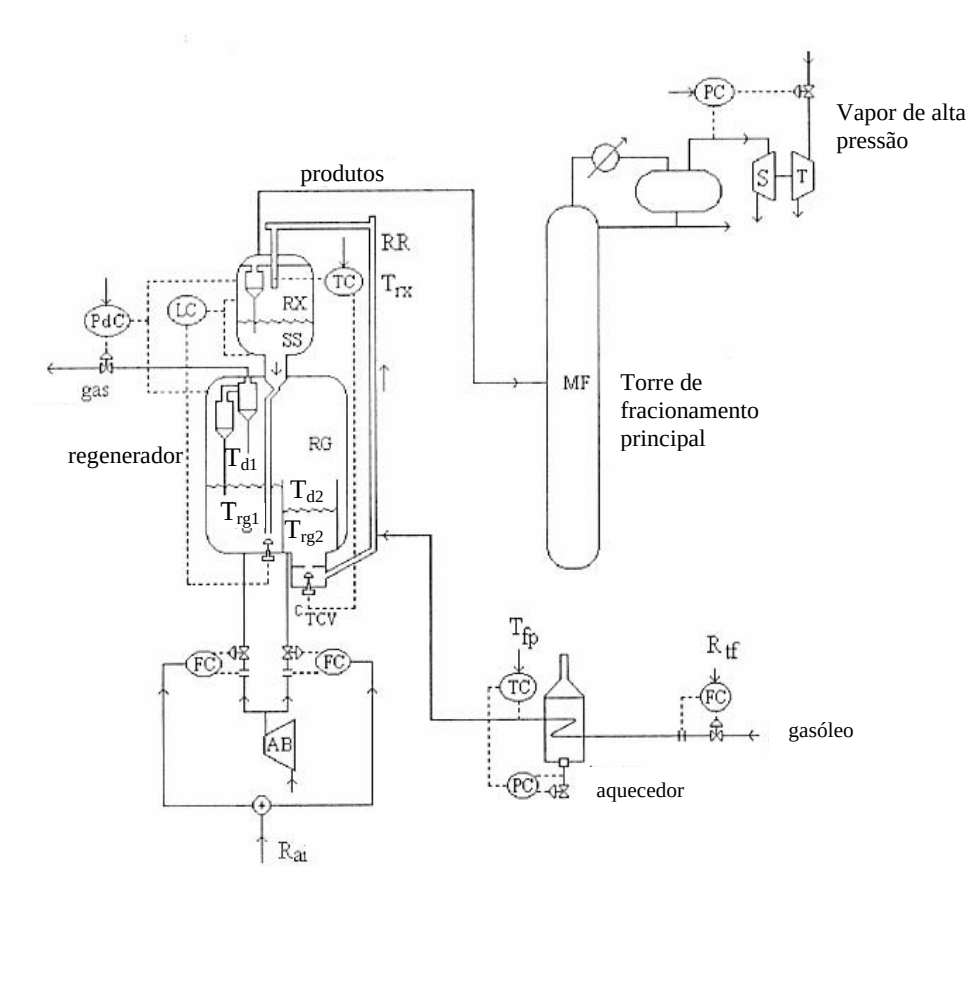

Figura 5.1 Representação esquemática de uma unidade FCC típica
Fonte: Moro e Odloak, 1995

O regenerador é ainda dividido em cinco partes:

- 1º estágio da fase densa: o catalisador que vem do vaso de separação é depositado neste compartimento, onde sofre a primeira combustão.
- 2º estágio da fase densa: do 1º estágio o catalisador trasborda para o 2° estágio, onde sofre uma segunda queima de coque para então retornar ao *riser*.
- 1º estágio da fase diluída: localiza-se acima da fase densa do 1º estágio e é constituída pelos gases de combustão provenientes da queima do coque desta fase.
- 2º estágio da fase diluída: localiza-se acima da fase densa do 2º estágio e é constituída pelos gases de combustão provenientes da queima do coque desta fase.
- Fase diluída geral: é formada pela mistura dos gases das outras fases, dando origem a uma região mais diluída cujos gases são levados para uma fornalha através de ciclones.

5.3 VARIÁVEIS OPERACIONAIS DO FCC

Conforme apresentado por Moro e Odloak (1995), as principais variáveis do processo são classificadas da seguinte forma:

5.3.1 Variáveis de entrada (manipuladas)

- **Temperatura da carga** (T_{fp}) – ajustada através de um forno de aquecimento. Fundamental no controle do balanço térmico da unidade.
- **Vazão de circulação do catalisador** (c_{TCV}) – utilizada para controlar a temperatura na saída do *riser*.

- **Vazão da carga** (R_{tf}) – procura-se mantê-la no valor máximo possível visando o aumento de lucratividade da unidade de FCC.
- **Vazão de ar de queima** (R_{ai})– seu ajuste é fundamental para na manutenção da qualidade de regeneração do catalisador e, portanto, da estabilidade operacional.
- **Qualidade da carga** – quando a carga da unidade é formada por uma mistura de correntes de diferentes características, pode-se variar as proporções de cada uma para ajustar a qualidade final desejada.
- **Atividade do catalisador** – uma maior atividade aumenta a conversão e o rendimento de produtos nobres (gasolina e GLP).

5.3.2 Variáveis de saída (monitoradas e controladas)

- **Temperatura da fase densa do 1° estágio do regenerador** (T_{rg1})
- **Temperatura da fase densa do 2° estágio do regenerador** (T_{rg2})
- **Severidade** ($Severidade$) – Variável associada ao rendimento da reação.
- **Temperatura de saída do *riser*** (T_{rx}) – principal variável controlada da unidade FCC, mantida pelo ajuste da vazão do catalisador regenerado para o *riser*.
- **Temperatura do regenerador** – é a variável que ajusta o balanço térmico no conversor FCC.
- **Conversão** – todas as variáveis influenciam a conversão e o perfil dos produtos obtidos na unidade.
- **Relação catalisador-óleo** – variável dependente através da qual pode-se inferir o grau catalítico das reações.

- **Pressão diferencial entre o vaso separador e o regenerador** – garante a circulação do catalisador no sentido requerido.
- **Nível do separador** – utilizado para manter uma pressão maior a montante da LCV.
- **Nível do regenerador** – mantido através de reposição e descarte de catalisador.
- **Inventário do catalisador** – acompanhado indiretamente através do nível do regenerador.

5.3.3 Equações do modelo dinâmico

5.3.3.1 Balanço de coque no *riser* e no reator

$$H_{ra}\frac{dC_{cat}}{dt} = -R_{rc}C_{cat} + 100R_{cf} \tag{5.1}$$

onde R_{rc} é a vazão de catalisador regenerado para o *riser*. Esta vazão pode ser relacionada com a abertura e a pressão diferencial na válvula TCV pela expressão:

$$R_{rc} = \frac{125.5\Delta P_{TCV}}{\sqrt{\left(\frac{1}{60A}\right)^2 - \left(\frac{1}{60A_O}\right)^2}} \tag{5.2}$$

onde R_{cf} é a taxa de formação de coque.

$$R_{cf} = \frac{4.2P_{ra}\left(C/O\right)^{0.65}}{C_{cat}C_{rc2}^{0.06}}WHSV^{-0.35} \times \frac{R_{tf}D_{tf}}{1.44\cdot 10^{-5}}\exp\left[\frac{-1500}{R\left(T_{rx}+273\right)}\right] \tag{5.3}$$

O coque total no catalisador desativado (que deixa o *riser*) é a soma do coque catalítico e o coque residual no catalisador regenerado. Assim:

$$H_{ra}\frac{dC_{sc}}{dt} = -R_{rc}\left(C_{rc2} - C_{sc}\right) + 100R_{cf} \tag{5.4}$$

onde C_{sc} é a quantidade de coque no catalisador que deixa o reator e entra no 1º estágio do regenerador.

5.3.3.2 Balanço de energia no *riser*

Considerando o tempo de residência do catalisador e hidrocarbonetos no *riser*, é aceitável assumir que o sistema de reação é quase estacionário:

$$S_c R_{rc}\left[T_{rg2} - T_{rx}\right] + S_f D_{tf} R_{tf}\left[T_{fp} - T_{rx}\right] - \Delta H_{fv} D_{tf} R_{tf} - 1440\Delta H_{cr} R_{oc} = 0 \tag{5.5}$$

R_{oc} é a taxa da reação de craqueamento, dada por:

$$R_{oc} = \frac{0.844\cdot 10^{-3} A_s}{\left[1 + A_s\right]} R_{tf} D_{tf} \tag{5.6}$$

onde A_s é a severidade de reação definida pela expressão:

$$A_s = \frac{4040}{C_{cat} C_{rc2}^{0.06}} P_{ra}\left[C/O\right]^{0.65} WHSV^{-0,35} \times \exp\left[-\frac{15000}{R\left(T_{rx} + 273\right)}\right] \tag{5.7}$$

5.3.3.3 Inventário de catalisador no reator

O *hold-up* do catalisador no reator é função das vazões de catalisador regenerado e desativado:

$$\frac{dH_{ra}}{dt} = R_{rc} - R_{sc} \tag{5.8}$$

A vazão de catalisador desativado R_{sc} depende da pressão diferencial na válvula do reator. Esta pressão pode ser calculada por:

$$\Delta P_{LCV} = P_{ra} + \gamma h_{ra} + \gamma h_{sp} + P_{rg} \tag{5.9}$$

A altura de catalisador no reator é dada por:

$$h_{ra} = \frac{H_{ra}}{\gamma A_{ra}} \tag{5.10}$$

A temperatura do leito do reator é calculada pela equação:

$$H_{ra}\frac{dT_{ra}}{dt} = R_{rc}T_{rx} - R_{sc}T_{ra} \tag{5.11}$$

5.3.3.4 Balanço de massa do catalisador na fase densa do 1º estágio do regenerador

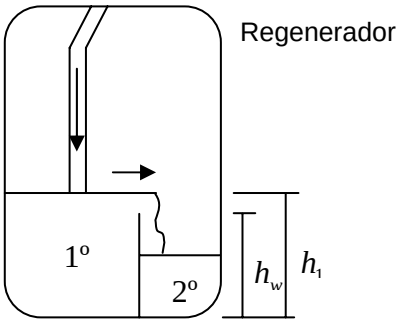

Figura 5.2 Vazão de catalisador no regenerador
Fonte: Moro, 1992 pg.24

A

Figura 5.2 mostra esquematicamente o sentido de escoamento de catalisador no regenerador. O catalisador desativado vem do reator através de uma válvula de

controle de nível. O catalisador regenerado vai do primeiro para o segundo estágio dependendo da altura do leito fluidizado sobre vertedouro que separa os dois estágios.

O balanço de massa pode, então, ser dado por:

$$\frac{dH_{rg1}}{dt} = R_{sc} - K_W \left[h_1 - h_W\right]^{0.5} \tag{5.12}$$

onde h_1 é a altura do leito no primeiro estágio, dada por:

$$h_1 = \frac{dH_{rg1}}{\gamma A_{rg1}} \tag{5.13}$$

e a vazão de catalisador para o 2º estágio é dada por:

$$R_{c1} = K_W \left[h_1 - h_W\right]^{0.5} \tag{5.14}$$

5.3.3.5 Balanço de energia na fase densa do 1º estágio do regenerador

Assume-se que o leito do regenerador comporta-se como uma mistura perfeita, com concentração e temperatura homogêneas. Assume-se também que parte do ar injetado no 1º estágio é carregada para o 2º estágio pelo catalisador. Dessa forma, explica-se a combustão residual que se processa no 2º estágio quando não há injeção de ar neste. O balanço energético fica então:

$$H_{rg1}S_c\frac{dT_{rg1}}{dt} = S_cR_{sc}T_{ra} - S_cR_{rc1}T_{rg1} + S_a\frac{R_{a1}}{60}\left(T_a - F_{12}T_{rg1}\right) - 0.001F_{g1}S_aT_{rg1} - 0.012\Delta H_{c1}C_{arb1} \tag{5.15}$$

onde F_{12} é a fração da vazão de ar que entra no 2º estágio. O calor de combustão ΔH_{c1} é considerado como uma função da relação CO_2/CO, calculado por:

$$\Delta H_{c1} = \frac{\left[7831 + \frac{5416}{CO_2/CO}\right]}{\left[1 + \frac{1}{CO_2/CO}\right]} \tag{5.16}$$

Esta equação assume que as reações de combustão são:

$$C + O_2 \rightarrow CO_2 \tag{5.17}$$

$$C + \frac{1}{2}O_2 \rightarrow CO \tag{5.18}$$

A presença de hidrogênio e enxofre no coque é negligenciada e, para fins de simulação, este controle não é relevante. A relação CO_2/CO é função da temperatura:

$$CO_2/CO = 600\exp\left(-\frac{6240}{T_{rg1}+273}\right) \tag{5.19}$$

A taxa (mássica) de coque queimado (C_{arb1}) é dada por:

$$C_{arb1} = \frac{10}{12}R_{cb1}H_{rg1} \tag{5.20}$$

onde R_{cb1} é a taxa de combustão do coque, dada por:

$$R_{cb1} = 1.5\cdot 10^8 C_{rc1}O_{fg1}P_{rg}\exp\left[\frac{18900}{T_{rg1}+273}\right] \tag{5.21}$$

5.3.3.6 Balanço do coque no 1º estágio do regenerador

O balanço do coque no 1º estágio do regenerador é dado por:

$$\frac{dC_{rc1}}{dt} = \left[\frac{R_{sc}C_{sc} - R_{rc1}C_{rc1}}{}H_{rg1}\right] - R_{cb1} \tag{5.22}$$

5.3.3.7 Balanço de oxigênio na fase densa do 1º estágio do regenerador

$$\frac{V_1\rho_1}{100}\frac{dO_{fg1}}{dt} = 0.21R_{ma1} - \frac{F_{gm1}O_{fg1}}{100} - \frac{10}{12}R_{cb1}H_{rg1}F_{at1} \tag{5.23}$$

onde R_{ma1} é a vazão de ar no primeiro estágio, não inclusa a entrada de ar pelo catalisador para o segundo estágio. F_{at1} é a razão mássica entre o oxigênio consumido e o coque queimado dada por:

$$F_{at1} = \frac{\left(CO_2/CO + 2\right)}{2\left(CO_2/CO + 1\right)} + \frac{\chi}{4} \tag{5.24}$$

e χ é a razão entre o número de átomos de hidrogênio e carbono no coque.

As equações para o 1º estágio podem ser repetidas (com diferenças mínimas para o 2º estágio da fase densa).

Assume-se que, na fase diluída, tem-se apenas a combustão de CO:

$$2CO + O_2 \rightarrow 2CO_2 \tag{5.25}$$

com taxa de reação dada por:

$$R_{CO1} = 1.5\exp\left(-\frac{15000}{\left(T_{d1} + 273\right)}\right)\left[O_{d1}\right]^{0.5}\left[CO_{d1}\right]P_{rg}^{1.5} \tag{5.26}$$

Com esta hipótese, o balanço de massa de oxigênio na fase diluída torna-se:

$$\frac{V_{d1}\rho_{d1}}{100}\frac{d\left[O_{d1}\right]}{dt} = F_{g1}\frac{\left\{O_{fg1} - \left[O_{d1}\right]\right\}}{100} - 30R_{CO1}V_{d1} \tag{5.27}$$

e o balanço de energia nesta fase é simplificado para:

$$V_{d1}\rho_{d1}S_a\frac{dT_{d1}}{dt} = F_{g1}S_a\left(T_{rg1} - T_{d1}\right) + 4.058\cdot10^6 R_{CO1}V_{d1} \tag{5.28}$$

Assim como na fase densa, as equações para o 2º estágio da fase diluída e para a fase diluída geral podem ser escritas analogamente às equações para o 1º estágio.

A pressão do regenerador é calculada como:

$$\frac{V_{rg} M_{rg}}{R\left(T_{rg} + 273\right)} \frac{dP_{rg}}{dt} = F_g - F_{go} \tag{5.29}$$

onde F_{go} é a vazão de gás que deixa o regenerador e é usada na caldeira de recuperação.

5.4 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

O simulador que gerou os dados para identificação do processo foi baseado no modelo elaborado por Moro e Odloak (1995) cujas equações são apresentadas neste capítulo.

Capítulo 6 - Otimização do Processo de Craqueamento Catalítico Usando Métodos Heurísticos Estocásticos

Na seção 6.1 apresenta-se um resumo a respeito dos métodos de otimização. Uma breve descrição dos algoritmos genéticos é apresentada na seção 6.2, incluindo os operadores de evolução (seleção, mutação e recombinação) e alguns métodos de seleção. Na seção 6.2.3 apresenta-se um exemplo de algoritmo genético simplificado e outro adaptado à otimização do FCC Na seção 6.3 o mesmo procedimento é adotado para o método PSO e na seção 6.4 discute-se o problema de otimização na unidade FCC.

6.1 MÉTODOS DE OTIMIZAÇÃO

A otimização matemática é uma área da ciência computacional que procura responder a pergunta “o que é melhor?” para problemas em que a qualidade de uma resposta pode ser medida por um número. Estes problemas aparecem em quase todas as áreas do conhecimento, inclusive na Engenharia. A otimização de um processo, por exemplo, pode levar à determinação dos valores das variáveis de processo que apresentam o melhor desempenho, segundo um critério pré-determinado (EDGARD e HIMMELBLAU, 1989).

Problemas típicos em engenharia química têm muitas e, em alguns casos, infinitas soluções. A otimização consiste em selecionar a melhor solução dentre um conjunto de soluções através de métodos quantitativos eficientes (EDGARD e HIMMELBLAU, 1989).

Um método de otimização é chamado determinístico se for possível prever todos os seus passos conhecendo seu ponto de partida. Em outras palavras, um método determinístico sempre leva à mesma resposta se partir do mesmo ponto inicial. Em oposição a estes métodos, existem os chamados métodos estocásticos ou aleatórios, onde o caráter aleatório de vários processos é simulado. Nestes métodos, várias escolhas são feitas com base em números aleatórios, sorteados no momento da execução do código. Como a cada execução os números aleatórios são diferentes, um método aleatório não executará a mesma seqüência de operações em duas execuções sucessivas. Partindo de um mesmo ponto inicial, cada execução seguirá seu próprio caminho e, possivelmente, levará a uma resposta final diferente.

Uma abordagem freqüente em otimização é a abordagem heurística. O termo heurístico significa que, com base na experiência ou julgamento, chega-se a uma solução razoável de um problema, o que não garante a solução matematicamente ótima (SILVER, 2004) mas permite, em geral, uma boa redução de custo e tempo (BAUCHPIESS, 2004). Espera-se que a técnica heurística funcione muitas vezes, mas não sempre.

Os métodos heurísticos de otimização podem ser determinísticos ou estocásticos, dependendo do emprego ou não de números sorteados aleatoriamente para executar seu algoritmo.

6.2 ALGORITMOS GENÉTICOS

Algoritmos genéticos (AG) têm sido objeto de considerável interesse nos últimos anos porque provêm uma busca robusta para solução de vários problemas

difíceis. Esses algoritmos fornecem uma solução heurística para um problema qualquer proposto. O incentivo ao uso destes métodos é, principalmente, devido ao fato de métodos determinísticos, como o SQP (*Successive Quadratic Programming*), falhem para processos altamente não lineares.

Os AG são uma técnica de busca global de otimização de uma função inspirada em idéias advindas da genética e do processo de seleção natural. Conceitos de computação evolutiva têm sido empregados em uma variedade de disciplinas. A idéia básica, surgida nos anos 50, é aplicar o processo de evolução natural como um paradigma de solução de problemas (MICHALEWICZ e FOGEL, 2000 apud VON ZUBEN, 2004).

Os conceitos básicos do AG são:

a) Cromossomo – representando um indivíduo ou solução possível
b) População
c) Cruzamento
d) Mutação
e) Critério de avaliação
f) Seleção

Basicamente, um AG trabalha com uma população inicial que pode, por exemplo, corresponder a valores numéricos de uma variável particular. O tamanho desta população pode variar e é, geralmente, relativa ao problema considerado (VON ZUBEN, 2004). A idéia é, então, buscar a solução ótima para um problema partindo-se de uma população inicial de candidatos à solução. A partir do cruzamento destes indivíduos, chega-se a uma nova geração. A seleção dos que estiverem mais próximos da solução ótima, isto é, os mais aptos, permite levar a uma nova geração

e assim por diante. Eventualmente, alguns cromossomos podem ser alterados, representando uma mutação.

O AG manipula populações de cromossomos, que são representações da solução do problema. Os cromossomos do algoritmo são abstrações do DNA biológico e cada posição particular no cromossomo é denominado gene. Os genes são os responsáveis pelas características físicas e psicológicas de seu portador. Analogamente, o cromossomo matemático é um vetor que contém valores para as variáveis independentes do problema, e cada gene é, portanto, uma variável independente (MCCALL, 2005).

Para uma maior compreensão do algoritmo genético, apresenta-se a seguir um modelo simples que contém as principais características da técnica, envolvendo populações em diferentes contextos. Naturalmente, muitas variações deste exemplo podem ser feitas.

Considere-se a população $\Pi(t)$, formada por uma geração t e a próxima, (t+1). A população tem P indivíduos. Os indivíduos são caracterizados por um vetor de variáveis genéticas que podem ser descritas descritas por:

$$\vec{S}^{\mu} = (S_1^{\mu}, S_2^{\mu}, \cdots, S_N^{\mu}) \qquad (6.1)$$

Os sobrescritos em grego indicam diferentes indivíduos (cromossomos) da população, μ = 1, ... , P e os subscritos em latim indicam a posição no vetor, i =1, ..., N . Neste protótipo, S_i^{μ} podem ser +1 ou –1.

6.2.1 Operadores de evolução

A população é mudada por um conjunto de operadores de evolução. Entre eles há a seleção, mutação e cruzamento (também denominado recombinação).

6.2.1.1 Seleção

A seleção no AG procura garantir que os cromossomos que possuem as melhores soluções possuam as maiores chances de serem selecionados mais de uma vez, assegurando assim que elas se mantenham na nova geração. O método tradicional de seleção é a roleta, que confere a cada cromossomo um peso relativo à qualidade da solução que ele contém. Ou seja, o melhor cromossomo possui o maior peso e ocupa, desta forma, a maior fatia da roleta, aumentando sua chance de ser escolhido para ser recombinado e ser mantido na próxima geração (McCall, 2005).

Existem ainda outros métodos de seleção, como o método do torneio, onde dois cromossomos são selecionados da população e aquele que possui o melhor grau de ajuste é selecionado para a recombinação e a seleção. Mais informações sobre métodos de seleção de indivíduos no algoritmo genético podem ser encontradas em McCall (2005) e Michalewicz e Fogel (2000).

6.2.1.1.1 Seleção por fitness

O potencial reprodutivo depende do *fitness* dos membros da população. Na Biologia, *fitness* tem um significado técnico: um *fitness* individual é proporcional ao número de sua descendência. Pode-se chamar de *fitness* biológico. Para fins de

modelagem é prático usar *fitness* em um sentido mais restritivo, como uma medida de potencial de reprodução.

Neste modelo simplificado, o *fitness* μ de um indivíduo depende apenas do vetor genético e pode ser expresso por:

$$F^{\mu} = F(\vec{S}^{\mu}) \tag{6.2}$$

A função F é conhecida como a função *fitness*. No modelo que se apresenta, a função *fitness* é dada por:

$$F(\vec{S}^{\mu}) = \sum_{i-1}^{N} S_i^{\mu} \tag{6.3}$$

Assim, na etapa de seleção os membros mais qualificados da população (com relação ao *fitness*) se reproduzem preferencialmente em relação aos menos qualificados. No modelo apresentado, um membro da população pode ser escolhido com a probabilidade:

$$p^{\mu} = \frac{w^{\mu}}{\sum_{\nu=1}^{P} w^{\nu}}, \qquad w^{\mu} = e^{\beta F^{\mu}} \tag{6.4}$$

onde β controla o grau de seleção. Neste modelo simplificado, o *fitness* biológico é dado por w^{μ}.

6.2.1.1.2 Seleção proporcional ao fitness

O AG clássico utiliza a seleção proporcional ao fitness, geralmente implementado com o algoritmo da roleta (*roulette wheel*) que atribui a cada indivíduo de uma população uma probabilidade de passar para a outra proporcional ao seu *fitness* normalizado (*fitness* medido em relação à somatória do *fitness* de todos os indivíduos da população). Esta seleção permite a perda do melhor indivíduo da população. (VON ZUBEN, 2004).

6.2.1.1.3 Seleção por Torneio

É um dos mais refinados processos de seleção, que é feita em função do número de vitórias de cada indivíduo em *N* competições contra *q* oponentes aleatórios da população. Vence uma competição aquele que apresentar o maior fitness comparado ao(s) de seu(s) oponente(s) (VON ZUBEN, 2004).

6.2.1.1.4 Seleções bi-classista e elitista

Na seleção bi-classista, são escolhidos os *b%* melhores indivíduos e os *w%* piores indivíduos da população. O restante, *(100-(b+w))%,* é selecionado aleatoriamente, com ou sem reposição. O elitismo consiste em um caso particular da seleção bi-classista na qual um dos melhores indivíduos da população é sempre mantido e nenhum dos piores é selecionado (VON ZUBEN, 2004).

6.2.1.2 Mutação

O operador de mutação altera os vetores mudando algumas das variáveis S_i^{μ}. No modelo apresentado, cada variável S_i^{μ} pode sofrer mutação com a probabilidade:

$$S_i^{\mu} \rightarrow S_i^{\mu'} = \begin{cases} S_i^{\mu} & com\ probabilidade\ 1-\gamma^m \\ -S_i^{\mu} & com\ probabilidade\ \gamma^m \end{cases} \tag{6.5}$$

onde γ^m é a taxa de mutação. A figura (6.1) ilustra este operador.

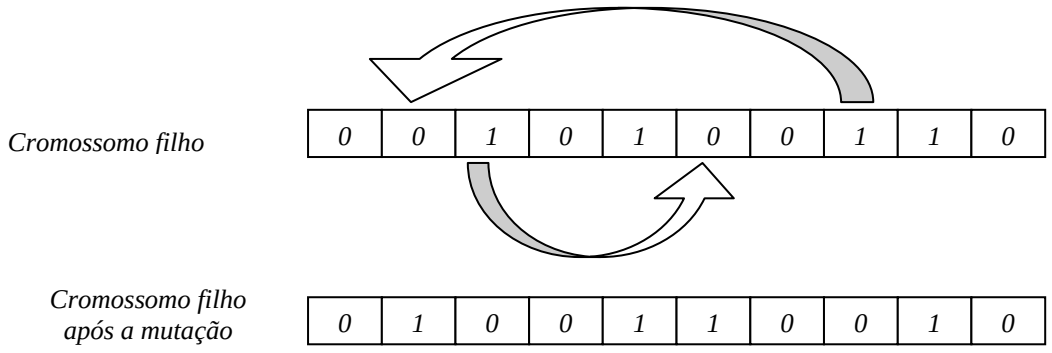

Figura 6.1 Operador genético de mutação

6.2.1.3 Recombinação sexual ou cruzamento.

A recombinação é o processo no qual os indivíduos selecionados são recombinados para formar os membros da próxima população. A idéia é simular a mistura do material genético que ocorre na reprodução de organismos. Há duas maneiras de proceder a recombinação, que são os operadores genéticos de

cruzamento e de mutação. Esses operadores não possuem comportamento determinístico, existindo, portanto, um fator estatístico que interfere tanto no cruzamento como na mutação dos cromossomos (MCCALL, 2005).

Os cromossomos são combinados e formam novos cromossomos. Isto é usualmente feito em pares; tomando-se dois cromossomos “pais” e criando um cromossomo “filho”. No algoritmo apresentado são feitos pares de cromossomos e um filho é criado escolhendo-se a variável na posição i para o filho, de um de seus pais (um ou outro). Chamando os pais de $\vec{S}^{\mu}$ e $\vec{S}^{\nu}$ e seu filho de $\vec{S}^{\kappa}$, tem-se:

$$S_i^{\kappa} = \chi_i S_i^{\mu} + (1-\chi_i) S_i^{\nu} \tag{6.6}$$

Onde χ_i é uma variável binária aleatória. Além disso, pode-se criar um “filho complementar”, S_i^{λ}, gerado pelas variáves que foram deixadas de fora da primeira combinação, isto é:

$$S_i^{\lambda} = (1-\chi_i) S_i^{\mu} + \chi_i S_i^{\nu} \tag{6.7}$$

Isto assegura que nenhum alelo (“pedaço”) é perdido através do cruzamento.

Este operador está ilustrado na figura a seguir:

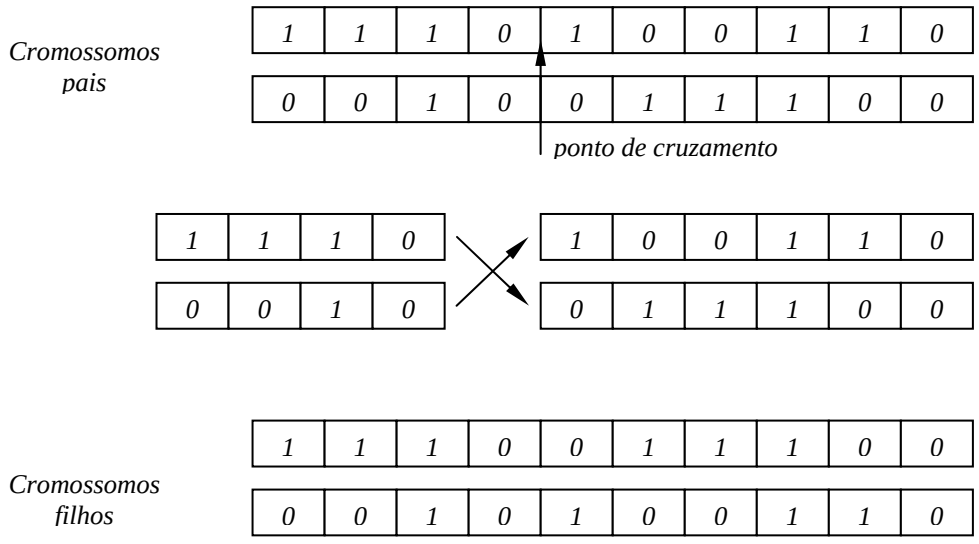

Figura 6.2 Operador genético de cruzamento
Fonte: MCCALL, 2005.

6.2.2 Um algoritmo evolutivo

Há diversos modos de combinar os operadores de evolução. Pode-se, por exemplo, assumir que a população evolui de uma geração para outra.

Começando com a população:

$$\Pi(t) = (\vec{S}_1(t), \vec{S}_2(t), \cdots, \vec{S}_N(t)) \tag{6.8}$$

no tempo t pode haver uma seleção para obter uma nova geração $\Pi^s(t)$ Pode-se assumir que esta nova população tem o mesmo tamanho da inicial. O operador de mutação é aplicado em cada indivíduo desta população e cria-se uma terceira população $\Pi^{sm}(t)$. Finalmente os membros desta última são postos lado a lado e recombinados para gerar uma nova população $\Pi^{smc}(t)$, que serve como próxima

geração $\Pi(t+1)$. Sucintamente, pode-se descrever este processo da seguinte forma:

$$\Pi\ (t) \ \rightarrow\ \Pi^{s}(t) \ \rightarrow\ \Pi^{sm}(t) \ \rightarrow\ \Pi^{smc}(t) \ =\ \Pi(t+1) \qquad (6.9)$$

onde os sobrescritos s, m e c denominam as populações depois da seleção, mutação e crossover, respectivamente. Os processos de seleção e recombinação são então repetidos e uma outra população é criada. O processo iterativo (evolutivo) continua até que o critério de parada seja satisfeito. Este critério pode ser um número de gerações fixo, a observação do valor da função objetivo, ou uma solução que satisfaça as restrições do problema (McCALL, 2005).

O Algoritmo Genético deve começar com uma população inicial $\Pi(0)$. Para este modelo simplificado, assume-se que a população inicial é um vetor escolhido aleatoriamente de ± 1's. Pode-se resumir este algoritmo genético da seguinte forma:

1. Gerar a população inicial.
2. Quantificar o *fitness* de cada indivíduo da população.
3. Selecionar preferencialmente os membros mais qualificados.
4. Aplicar o operador de mutação em cada membro da população.
5. Recombinar os membros em pares.
6. Retornar ao passo 2 até que alguma condição de parada seja satisfeita.

Uma das maneiras utilizadas na escolha do esquema de evolução da população envolve a quantidade de cromossomos que não sofrerá quaisquer alterações quando passarem de uma geração para outra. É possível criar uma população completamente nova, na qual todos os cromossomos da geração anterior passem pelos operadores de recombinação ou uma população que cria um único

cromossomo novo a cada geração. O esquema evolutivo usado com mais freqüência é a substituição por elitismo. Neste esquema toda a população é renovada sendo que o melhor indivíduo da geração anterior é preservado na nova geração para garantir que uma boa solução não seja perdida (MCCALL, 2005).

6.3 OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

A otimização por enxame de partículas (PSO – *Particle Swarm Optimization*) é um método heurístico estocástico de busca, relativamente recente, usado para encontrar a solução ótima (ou uma próxima a ela) para problemas numéricos e qualitativos. Este método tem apresentado soluções efetivas e rápidas quando aplicada a diversos conjuntos de problemas de otimização (POMEROY, 2003). Seu mecanismo é inspirado pelo comportamento cooperativo das populações biológicas como, por exemplo, pássaros e abelhas.

O PSO é similar ao algoritmo genético (AG) por ser um método heurístico de busca baseado em uma população de soluções, ou seja, ambos se movem de um conjunto de pontos (população) a outro conjunto em uma única interação, com uma provável melhora do conjunto, usando uma combinação de regras determinísticas e probabilísticas (HASSAN *et al*, 2004). O PSO é altamente dependente dos processos estocásticos assim como os algoritmos genéticos (KENNEDY e EBERHART, 1995). É importante enfatizar que neste algoritmo, de forma distinta do algoritmo genético, o indivíduo (a partícula) é mantido íntegro durante todo o processo, *sobrevivendo* sem envelhecer até o final do procedimento, a única modificação sofrida pelo indivíduo é sua localização no espaço (VON ZUBEN, 2004). Os indivíduos na população retém a memória de boas soluções conhecidas enquanto continua a procura por melhores

soluções, diferentemente do AG, onde o conhecimento é destruído entre gerações (COCKSHOTT e HARTMAN, 2001).

Este mesmo comportamento cooperativo e de troca de informações é encontrado entre várias espécies animais: em revoadas de pássaros, em cardumes de peixes, em enxames de abelhas, etc. Cientistas de diferentes áreas tentaram modelar estes comportamentos e simulá-los. Reynolds (1987), em particular, discute a riqueza do movimento de revoadas de pássaros, de manadas de animais terrestres ou de cardumes de peixes procurando estabelecer regras básicas do comportamento individual que justifiquem o comportamento do grupo. A conclusão mais pertinente do artigo se refere ao fato do movimento simulado da revoada de pássaros ser o resultado de comportamentos relativamente simples do movimento de cada um dos indivíduos (pássaros).

O primeiro trabalho sobre o algoritmo de otimização natural denominado *ENXAME DE PARTÍCULAS* (*Particle swarm*) é o de Eberhart e Kennedy (1995), que, propuseram um algoritmo de otimização não determinístico bastante eficiente, robusto e de simples implementação computacional.

Kennedy e Eberhart (1995) tiveram particular interesse nos modelos desenvolvidos pelo biólogo Frank Heppner sobre revoada de pássaros. Devido a regras simples, cada pássaro adapta sua direção e velocidade para, essencialmente, ficar no meio dos pássaros vizinhos. Quando um deles sai da revoada para pousar em um lugar apropriado, os pássaros vizinhos farão o mesmo até que toda a revoada pouse. Encontrar um lugar para pousar é análogo a encontrar uma solução num conjunto de possíveis soluções (um espaço de soluções). O modo pelo qual um pássaro que encontra um lugar para pousar leva seus vizinhos consigo, aumentando as possibilidades de que eles também o

encontrem, leva ao estudo do comportamento social dos indivíduos (POMEROY, 2003). Assim, o PSO é definido num contexto social em oposição ao contexto biológico (COCKSHOTT e HARTMAN, 2001).

Um importante aspecto deste ponto de vista do “comportamento social” é que, primeiramente, o indivíduo aprende com o sucesso de seus vizinhos. Compara-se a si mesmo com os demais e imita-se o comportamento dos indivíduos que obtiveram sucesso. É necessário, desta forma, um balanço entre a exploração (procura própria por uma boa solução) e o aproveitamento (tirar vantagem do sucesso de outros). Idealmente quer-se indivíduos que sejam preferencialmente individualistas mas também é necessário que saiba onde foram encontradas boas soluções pelos demais para aprender com eles (POMEROY, 2003).

A tradução matemática do algoritmo de Eberhart & Kennedy (1995) é apresentada a seguir, entendendo-se que o termo partícula se refere a cada um dos indivíduos do grupo (termo equivalente a indivíduo no algoritmo genético) e o termo enxame se refere ao grupo de indivíduos.

A idéia fundamental do algoritmo é o estabelecimento, em cada passo ou iteração, do movimento de cada uma das partículas do grupo composto por ***n*** partículas, sendo o valor de ***n*** escolhido pelo usuário. Este movimento é norteado pela *lembrança* da *melhor* posição (melhor valor da função objetivo) no espaço que a partícula já encontrou em seu movimento e no *conhecimento* da *melhor* posição já encontrada por todo o grupo. A utilização da melhor posição individual da partícula pode ser classificada como uma espécie de autoconfiança e a informação da melhor posição do grupo pode ser classificada como um comportamento gregário do indivíduo. Para assegurar a existência de uma certa *personalidade* em cada indivíduo, dota-se cada indivíduo do grupo de um comportamento distinto e aleatório

em que a ponderação destas duas informações é *sorteada* ao longo do processo, assumindo assim um valor diferente para cada uma das partículas e variando este valor ao longo do processo iterativo (procurando traduzir uma certa mudança humor do indivíduo ao longo do tempo). Assim, considerando o movimento da partícula em um plano ou em um espaço bi-dimensional e sua localização neste espaço caracterizada por suas coordenadas (VON ZUBEN, 2004).

6.3.1 Algoritmo Básico

O algoritmo PSO básico consiste de três passos: i) geração de posições e velocidades de partículas, ii) atualização da velocidade e, finalmente, iii) atualização da posição. Uma partícula refere-se ao ponto no espaço que muda sua posição de um movimento (iteração) a outro baseado nas atualizações de velocidade. Primeiramente, as posições, x_k^i, e velocidades , v_k^i, do enxame inicial de partículas são gerados aleatoriamente usando-se os limites máximos e mínimos dos valores das variáveis, $x_{\max}$ e $x_{\min}$, como expressado nas equações 6.10 e 6.11. As posições e velocidades são dadas em formato de vetor com sobrescritos e subscritos denotando a *i-ésima* partícula no tempo k. Nas equações 6.10 e 6.11, *rand* é uma variável aleatória de distribuição uniforme que pode tomar qualquer valor entre 0 e 1. Este processo de inicialização leva o enxame de partículas a ser aleatoriamente distribuído no espaço (HASSAN *et al*, 2004).

$$x_0^i = x_{\min} + rand(x_{\max} - x_{\min}) \tag{6.10}$$

$$v_0^i = \frac{x_{\min} + rand(x_{\max} - x_{\min})}{\Delta t} = \frac{posição}{tempo} \tag{6.11}$$

O segundo passo é atualizar as velocidades de todas as partículas no tempo ($k+1$) usando os objetivos das partículas ou valores de *fitness* que são funções das posições atuais das partículas no espaço no tempo k. O valor da função *fitness* da partícula determina que partícula tem o melhor valor global no enxame atual, p_k^g, e também determina a melhor posição de cada partícula através do tempo, p^i, isto é, na atual e em todas as iterações anteriores. A fórmula da atualização da velocidade usa estas duas informações para cada partícula no enxame com o efeito do movimento atual, v_k^i, prover a direção de busca, v_{k+1}^i, para a próxima iteração. A fórmula de atualização da velocidade inclui alguns parâmetros aleatórios, representados pelas variáveis uniformemente distribuídas, *rand*, para assegurar uma cobertura de todo o espaço de busca e evitar que se fique preso em um ótimo local. Os três valores que afetam a nova direção de busca, seu atual movimento, a memória própria da partícula e a influência do enxame, são incorporados através de uma soma, como mostrado na equação 6.12, com três fatores de ponderação, denominados fator de inércia, w, um fator de individualidade, c_1, e um fator de identidade de grupo, c_2.

$$v_{k+1}^i = \underbrace{w v_k^i}_{\substack{movimento \\ atual}} + \underbrace{c_1 rand \frac{(p^i - x_k^i)}{\Delta t}}_{\substack{influência\ da\ memória \\ da\ partícula}} + \underbrace{c_2 rand \frac{(p_k^g - x_k^i)}{\Delta t}}_{influência\ do\ enxame} \tag{6.12}$$

A atualização da posição é o último passo em cada iteração. A posição de cada partícula é atualizada usando seu vetor velocidade como mostrado na Equação 6.13.

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i}\Delta t \tag{6.13}$$

As três etapas de atualização da velocidade, atualização da posição e cálculo de *fitness* são repetidas até que o critério de convergência desejado seja alcançado.

No algoritmo de PSO clássico, as variáveis podem assumir qualquer valor, mesmo fora de seus limites de restrição. Este fenômeno pode levar à divergência. Para evitar este problema, quando as variáveis violarem seus limites mínimos ou máximos elas são artificialmente trazidas novamente para dentro dos limites.

Além de aplicar penalidades às partículas que violarem os limites propostos é recomendado que o vetor velocidade desta partícula seja igualado a zero na fórmula da atualização da velocidade, como se segue:

$$v_{k+1}^{i} = c_1\, rand\, \frac{(p^{i} - x_{k}^{i})}{\Delta t} + c_2\, rand\, \frac{(p_{k}^{g} - x_{k}^{i})}{\Delta t} \tag{6.14}$$

6.4 Otimização da FCCU

Muitos estudos de otimização têm sido levados a efeito com relação à FCCU. A maioria deles usa algum tipo de função custo como objetivo. Algumas das variáveis de decisão comumente usadas nestes estudos são a temperatura do reator (T_{rx}), as temperaturas do regenerador (T_{rg1} e T_{rg2}), a taxa de circulação do catalisador, que pode ser relacionada com a abertura da válvula de catalisador

regenerado (c_{TCV}) e a vazão de ar no regenerador (R_{ai}). Estes estudos usam uma única função objetivo. Nos últimos anos, estudos mais detalhados de otimização usando múltiplas funções objetivo e restrições tem sido reportados na literatura usando uma variedade de algoritmos matemáticos (KASAT *et al*., 2002).

Diversas funções objetivo podem ser consideradas de modo a maximizar a lucratividade da unidade e satisfazer as restrições do processo. Na FCCU, um aumento na produção de gasolina, GLP (gás liquefeito de petróleo) ou diesel sempre leva a um aumento no lucro. No entanto, um aumento na produção de gasolina leva, geralmente, ao aumento na produção de GLP de modo que o uso da produção de gasolina como função objetivo é suficiente (KASAT *et al*., 2002).

O aumento da produção de gasolina, entretanto, tem efeito adverso na formação de coque, aumentando-a. O coque diminui a atividade do catalisador e precisa ser queimado no regenerador, requerendo uma quantidade maior de alimentação de ar. A combustão do coque resulta na formação de CO e CO_2. O monóxido de carbono deve ser convertido em dióxido, na fase diluída, antes que os gases entrem nos ciclones (Figura 5.1). Esta segunda combustão de CO pode produzir temperaturas muito altas. Há, usualmente, duas opções: uma é proceder à combustão total no regenerador de modo que o CO emitido na corrente gasosa seja bem baixa. A outra é fazer apenas uma combustão parcial, o que aumentaria as emissões de CO. A corrente gasosa é então levada a um conversor onde o CO é convertido a CO_2 antes de ser finalmente emitida para a atmosfera. O modo que utiliza a combustão completa de CO no regenerador requer grandes quantidades de alimentação de ar o que, automaticamente, aumenta o custo de operação. Adicionalmente, a combustão completa leva a uma quantidade muito maior de calor produzido, o que pode criar problemas. Há também a necessidade de combustão da

máxima quantidade de coque no regenerador para manter a atividade do catalisado no *riser* em níveis altos (KASAT *et al*., 2002).

Desta forma, o mesmo desafio do acoplamento de variáveis enfrentado na identificação e no controle da FCCU, repete-se na otimização do processo de craqueamento catalítico, dificultando a obtenção de uma solução ótima através de métodos tradicionais de otimização.

6.5 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

O capítulo 6 apresenta dois métodos heurísticos estocásticos de otimização, relativamente recentes, AG e PSO, ambos métodos evolucionários de busca global.

Seus algoritmos básicos foram apresentados aqui. Estes foram posteriormente adaptados para a resolução de um problema proposto de otimização da unidade FCC. É importante destacar que não foram encontradas, até o momento, referências do uso do algoritmo PSO na otimização da unidade FCC.

O problema de otimização da unidade FCC também é apresentado em linhas gerais.

Capítulo 7 – Resultados e discussão

Apresentam-se neste capítulo os resultados obtidos com o estudo na busca da melhor arquitetura de rede possível para um modelo neural que possa ser posteriormente implementado em uma estratégia de otimização da FCCU e/ou no projeto de um controlador preditivo do processo.

Um estudo da influência de cada variável de entrada no comportamento dinâmico do processo é apresentado na seção 7.2. A seção 7.3 descreve a metodologia de geração de dados para a identificação adotada neste trabalho. Para fins de identificação do processo foi testado inicialmente um modelo linear ARX MIMO (*autoregressive with exogenous inputs, multi-input, multi-output)* e os resultados estão apresentados na seção 7.4. A identificação do processo com redes neurais MLP MISO (*multi layer perceptron*, *multi-input, single output*) e MIMO é apresentada nas seções 7.5.1 e 7.5.2, respectivamente. Estudou-se também a influência da arquitetura da rede (número de regressores e neurônios na camada escondida) na qualidade da resposta apresentada pelo modelo neural. O objetivo foi identificar um modelo que apresentasse o melhor desempenho com o menor número de parâmetros possível.

Os resultados da otimização do processo utilizando os métodos heurísticos estocásticos AG e PSO, devidamente modificados para a implementação da melhor rede neural obtida assim como para considerar as restrições do processo, são apresentados nas seções 7.6 e 7.7.

Esta mesma rede neural foi utilizada como modelo não linear da FCCU no projeto de um controlador MPC do processo. Os resultados obtidos nos problemas de controle servo e regulatório são apresentados na seção 7.8.

7.1 INTRODUÇÃO

O projeto e a implementação do sistema de controle devem resolver questões desafiadoras. O caráter multivariável de um processo que apresenta fortes interações, o comportamento não linear que leva a um controle não linear e a demanda para operar a unidade na presença de restrições materiais e de operação são as principais delas. Além disso, o sistema de controle deve enfrentar constantes de tempo tanto pequenas quanto grandes devido a mudanças nas condições de operações e, freqüentemente, na presença usual de perturbações não medidas. Como conseqüência, o MPC mostra-se um bom candidato para implementação de controle avançado numa unidade de FCC (CRISTEA *et al*., 2003).

As principais variáveis de processo relacionadas ao problemas de otimização e controle da FCCU são: a temperatura do riser (T_{rx}); a temperaturas das fases densas do regenerador nos 1º e 2º estágios (T_{rg1} e T_{rg2}, respectivamente); a temperatura e vazão da alimentação de gasóleo (T_{fp} e R_{tf}), a vazão de ar para o regenerador (R_{ai}) e a abertura da válvula de catalisador regenerado (c_{TCV}). Estas variáveis são importantes para definir outras importantes variáveis como a severidade da reação de craqueamento ($Severity$) (ZANIN, 2001).

A faixa de operação das variáveis de interesse e os valores do estado estacionário de referência são apresentadas na Tabela 7.1 a seguir (ZANIN, 2001):

Tabela 7.1 Faixa de operação e estado estacionário das variáveis de interesse

		Faixa Operacional		Estado Estacionário
		max	min	
Variáveis Controladas/ Monitoradas	Trg1 (ºC)	710	640	670,15
	Trg2 (ºC)	710	640	700,89
	Severity	92	70	77,489
	Trx (ºC)	545	520	542,20
Variáveis Manipuladas	Rai (kg/m3)	231	201	221
	Ctcv (%)	92	42	82
	Rtf(kg/m3)	9840	5000	9700
	Tfp(ºC)	245	215	235

7.2 ESTUDO DA DINÂMICA DO PROCESSO

A temperatura da fase densa no primeiro estágio do regenerador (T_{rg1}) tem comportamento semelhante ao da temperatura da fase densa no segundo estágio (T_{rg2}), considerando que a vazão de ar que alimenta o primeiro e segundo estágios do regenerador é a mesma. Devido à acentuada interação entre o regenerador e o reator (*riser*), a severidade das reações de craqueamento (*Severity*) é fortemente influenciada pela temperatura da reação (T_{rx}), desta forma as respostas dinâmicas destas duas variáveis são similares (ZANIN, 2001). Este comportamento é ilustrado na Figura 7.1. Por este motivo, no controle do processo, são utilizadas como variáveis controladas apenas uma de cada par, sendo a outra apenas monitorada.

A tabela 7.2 apresenta as respostas das variáveis controladas do processo a degraus de $\pm$4,5% aplicados a cada uma das variáveis manipuladas separadamente, mantendo-se as demais nos valores do estado estacionário de referência (descritos na tabela 7.1).

Tabela 7.2 Estudo da dinâmica do processo

			Trg1	Trg2	Severidade	Trx
		Valor inicial	670,14	700,89	77,49	542,20
Rai	4,50%	**novo valor**	678,76	710,14	78,36	547,14
		variação %	1,29	1,32	1,12	0,91
	-4,50%	**novo valor**	661,69	691,48	76,61	537,29
		variação %	1,26	1,34	1,14	0,90
Rtf	4,50%	**novo valor**	662,32	693,20	75,11	533,53
		variação %	1,17	1,10	3,07	1,60
	-4,50%	**novo valor**	678,78	709,31	79,86	551,72
		variação %	1,29	1,20	3,06	1,76
Tfp	4,50%	**novo valor**	675,48	706,16	78,48	547,91
		variação %	0,80	0,75	1,29	1,05
	-4,50%	**novo valor**	664,92	695,72	76,46	536,58
		variação %	0,78	0,74	1,32	1,04
Ctcv	4,50%	**novo valor**	669,19	699,60	78,39	545,21
		variação %	0,14	0,18	1,16	0,56
	-4,50%	**novo valor**	670,98	702,01	76,46	538,76
		variação %	0,12	0,16	1,33	0,63

Com o objetivo de ilustrar o comportamento do processo quando submetido a degraus nas variáveis manipuladas, apresentam-se a seguir as respostas dinâmicas das variáveis controladas.

Os resultados obtidos na figura 7.1 foram obtidos com degraus de ±4,5% no valor de operação de referência da vazão de alimentação de gasóleo (Rtf). O valor do degrau foi estabelecido de modo que nenhuma variável ultrapassasse os limites de operação. É importante destacar que, para que a planta opere de forma segura, existem 3 controladores PI que controlam 3 variáveis críticas: 1) a quantidade (*hold-up*) de catalisador no reator; 2) A diferença de pressão entre o regenerador e o reator; 3) a pressão de sucção do compressor no topo da torre de fracionamento (Figura 5.1). Deste modo, as figuras a seguir são respostas típicas de processos operando sob ação de controle PI.

,

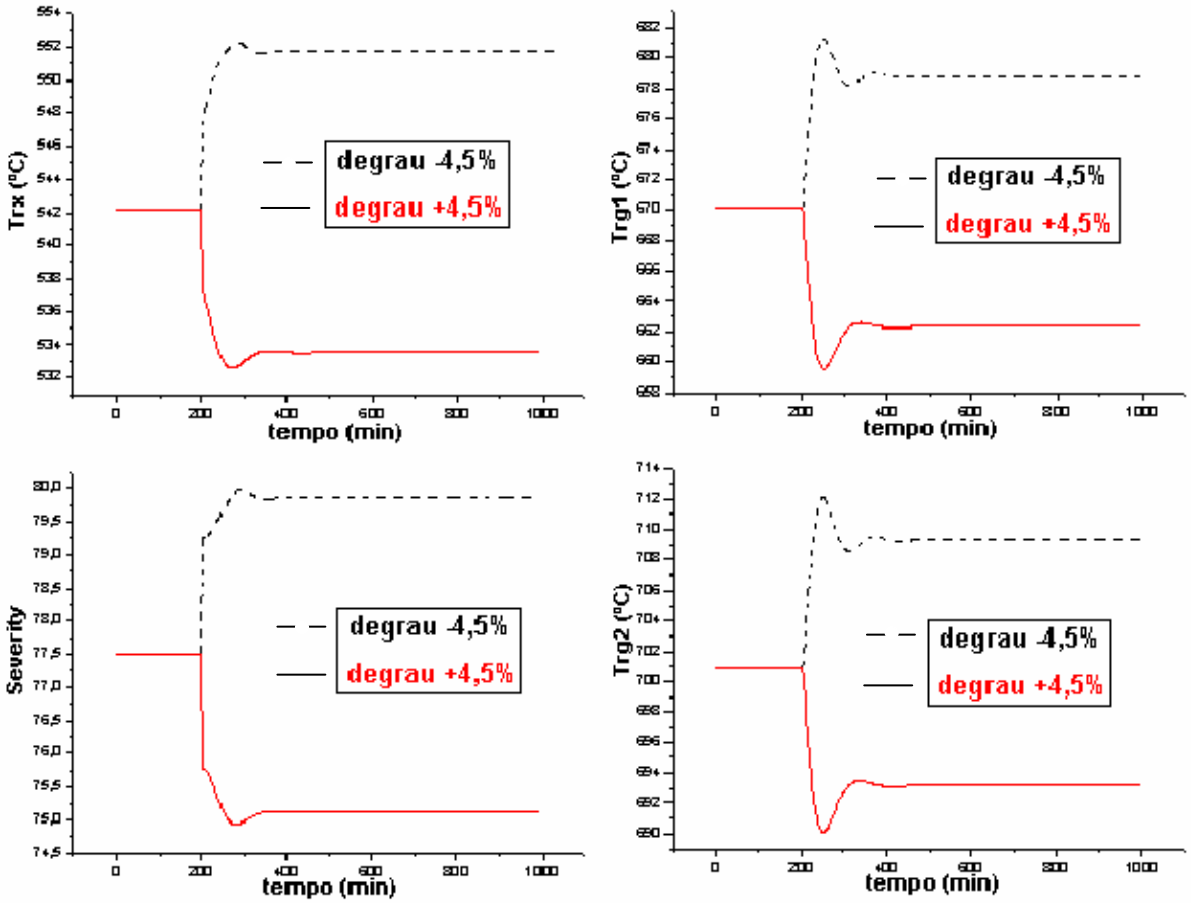

Figura 7.1 Respostas a degrau de ±4,5% em R_{tf}

Os resultados ilustrados na figura 7.2 foram obtidos com degraus de mesma intensidade na abertura da válvula de catalisador regenerado (c_{TCV}). Da mesma forma as figuras 7.3 e 7.4 equivalem a degraus idênticos na vazão de ar do regenerador (R_{ai}) e na temperatura da alimentação (T_{fp}).

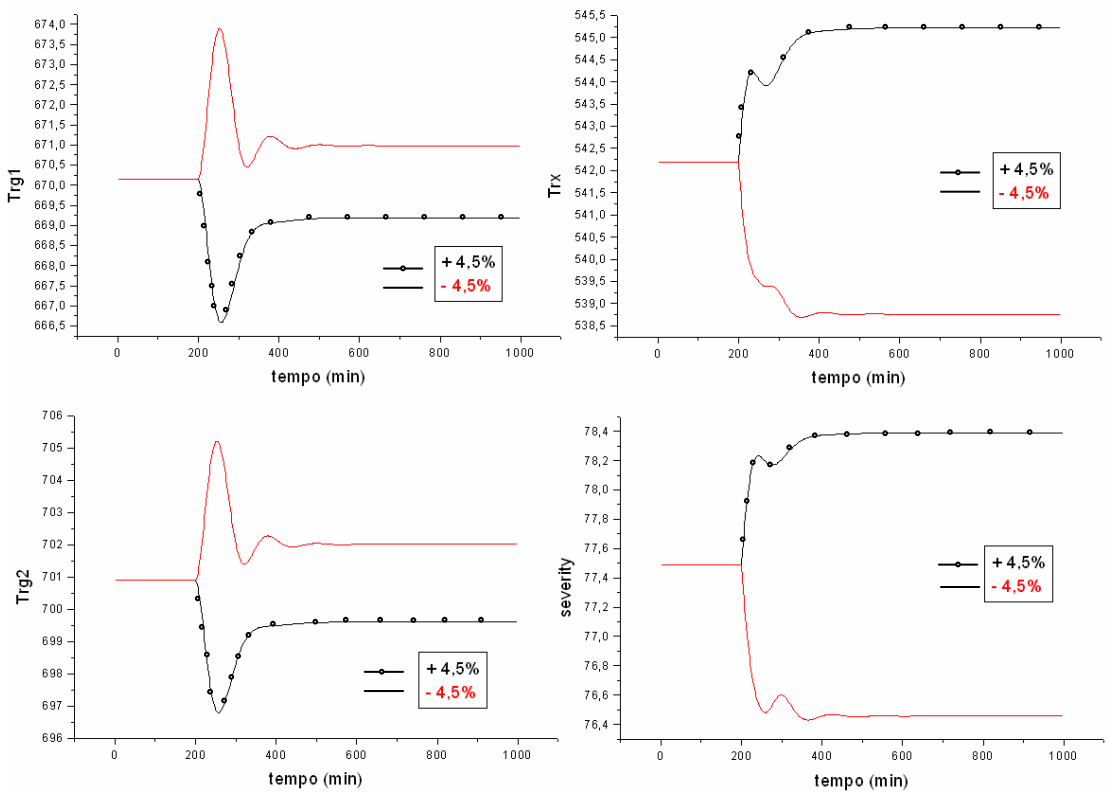

Figura 7.2 Respostas a degrau de ± 4,5% em c_{TCV}

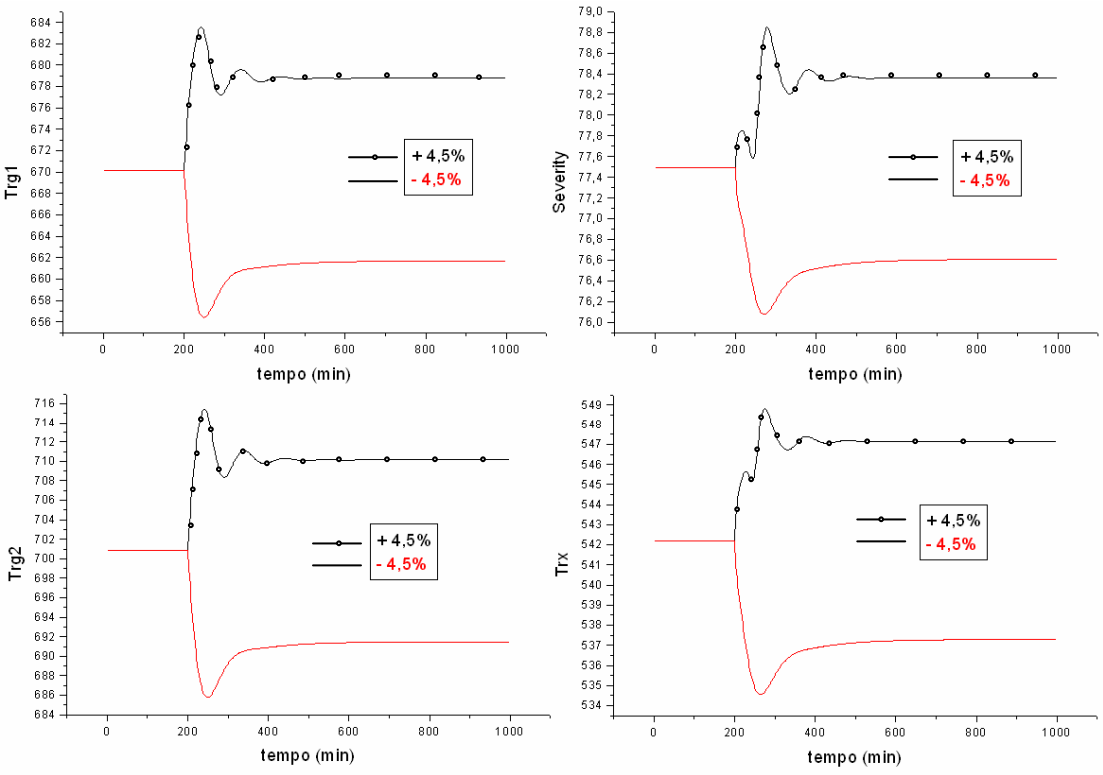

Figura 7.3 Respostas a degrau de ± 4,5% em R_{ai}

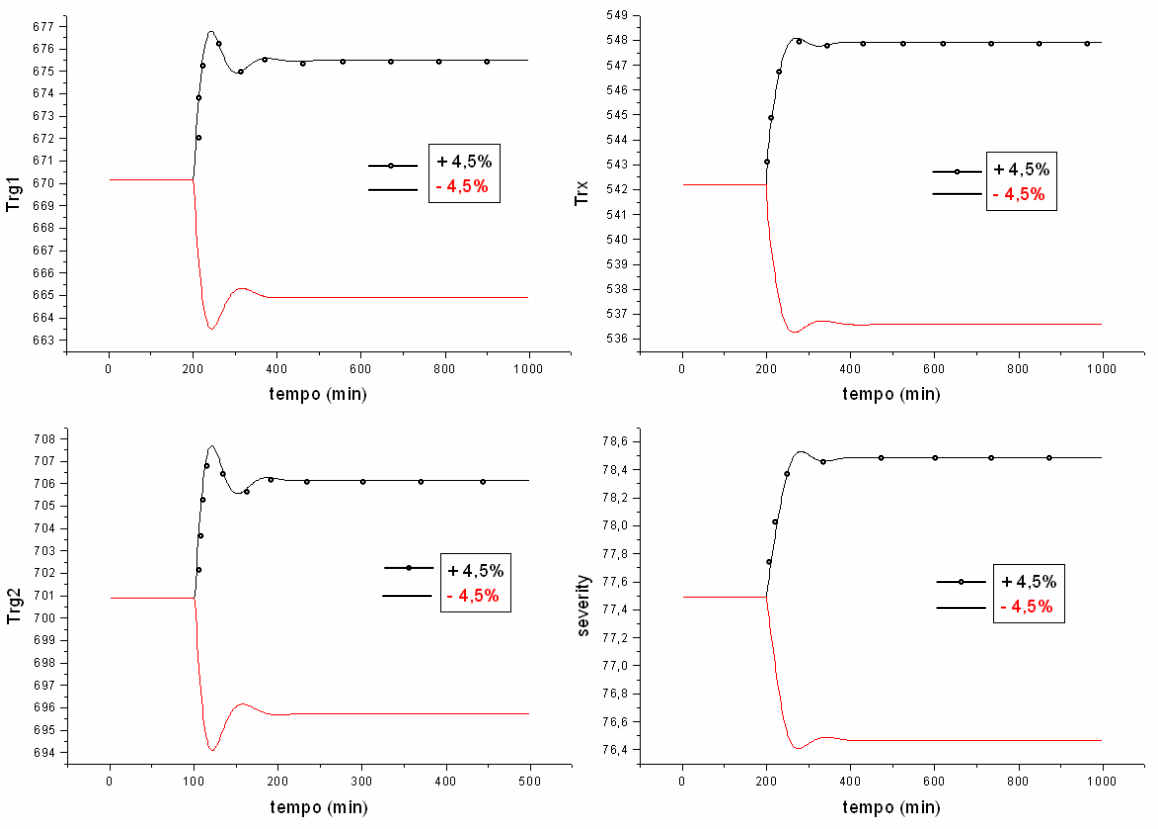

Figura 7.4 Respostas a degrau de ±4,5% em T_{fp}

Sobre os resultados na tabela 7.2, pode-se acrescentar:

a) **Vazão de ar para o regenerador (R_{ai})**. Na operação do regenerador em combustão parcial, a adição de ar no regenerador aumenta a temperatura do mesmo (T_{rg1}), pois ocorre liberação adicional de energia pela maior conversão do CO em CO_2. A vazão de catalisador regenerado não é alterada, pois a abertura da válvula está fixa. Assim, o catalisador mais quente fornece maior quantidade de energia no *riser*, elevando a temperatura de saída do mesmo (T_{rx}) (ZANIN, 2001).

b) **Abertura da válvula de catalisador regenerado (c_{TCV})**. Mantida constante a carga da unidade, a circulação de maior quantidade de catalisador aumenta a relação catalisador/óleo, introduzindo maior quantidade de energia no riser e elevando sua temperatura (T_{rx}). Como uma maior quantidade de coque é formada e a vazão total de ar é mantida fixa, as reações de carbono a CO competem com as

de CO a CO_2, reduzindo, assim, a relação CO_2/CO nos gases de combustão e, e, conseqüência, a temperatura do regenerador (T_{rg1}) (ZANIN, 2001).

c) **Vazão da carga (R_{ff})**. Um degrau positivo na vazão da carga reduz a relação catalisador/óleo e poder-se-ia supor que as respostas das variáveis controladas seriam opostas ao degrau positivo na abertura da válvula de catalisador regenerado. Porém a temperatura do regenerador (T_{rg1}) não segue este comportamento, pois o efeito da redução do coque gerado é sobreposto pelo da redução da temperatura do catalisador desativado (ZANIN, 2001).

d) **Temperatura da carga (T_{fp})**. Um degrau positivo na temperatura da carga, mantidas as demais manipuladas constantes, introduz mais energia no sistema, incrementando todas as temperaturas do mesmo (ZANIN, 2001).

7.3 GERAÇÃO DE DADOS PARA IDENTIFICAÇÃO DO PROCESSO

Um simulador do processo de craqueamento catalítico baseado no modelo de Moro e Odloak (1995) e Gouvêa (1997) foi utilizado para gerar um conjunto de dados contendo os sinais de entrada e saída referente a aproximadamente 167 horas, algo em torno de uma semana de operação. Embora com o simulador se tenha a vantagem de poder gerar arquivos de dados para identificação tão grandes quanto se queira, optou-se por utilizar dados referentes a apenas uma semana. Escolheu-se esta quantidade de dados por se entender que se trata de um número facilmente obtido em uma FCCU típica. Deve-se destacar que uma quantidade maior de dados, provavelmente, melhoraria o desempenho do modelo neural. As variáveis

de entrada foram perturbadas de acordo com o processo descrito no item II da seção 3.2.1.2, gerando sinais, já normalizados, ilustrados nas figuras a seguir.

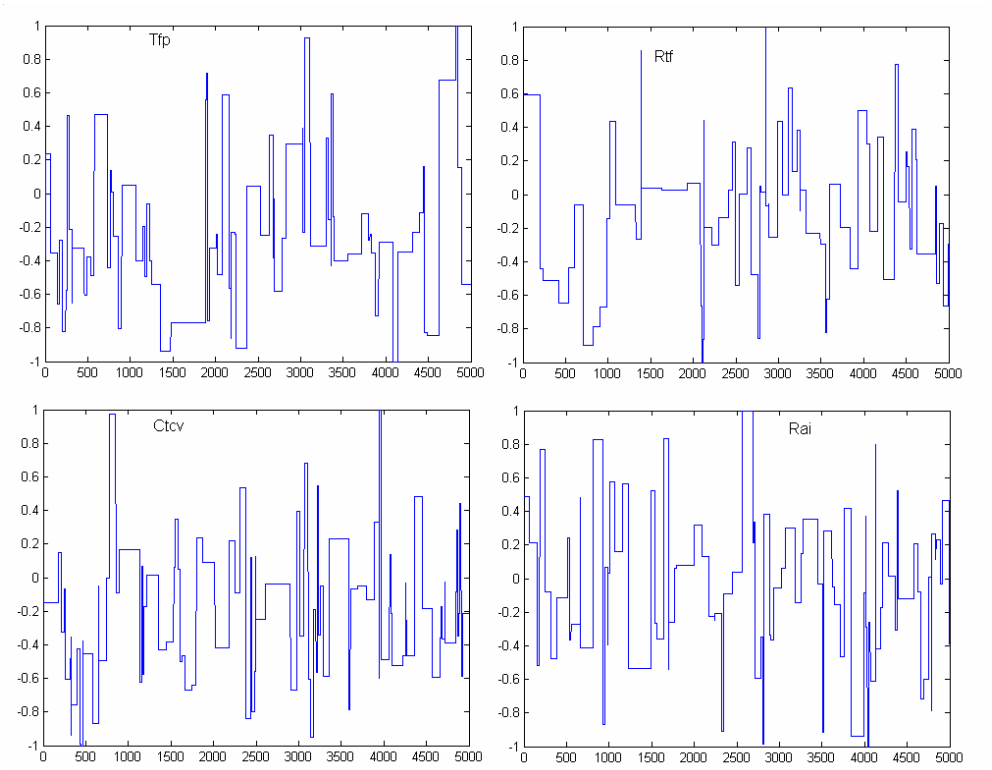

Figura 7.5 Dados de entrada

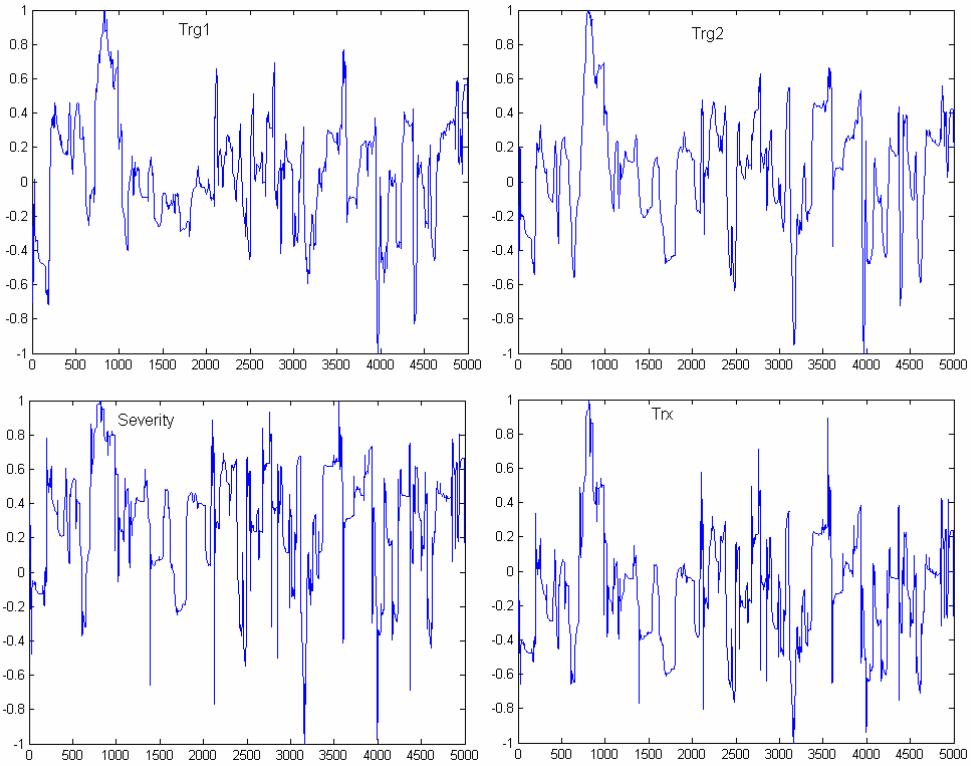

Figura 7.6 Dados de saída

7.4 IDENTIFICAÇÃO UTILIZANDO MODELOS LINEARES

Prática comum em identificação de sistemas, a identificação de um processo com um modelo linear, além de mais simples, pode ajudar na determinação do grau de não linearidade do sistema, considerando que muitos processos podem ser bem descritos por um modelo linear. Da comparação entre os resultados obtidos com modelos lineares e os obtidos com modelos não lineares mais elaborados poder-se-ia concluir que ambos representam bem a dinâmica do processo e, nesse caso, o modelo linear deveria ser escolhido por ser mais simples.

Inicialmente, empregou-se para representar a dinâmica do processo o modelo linear ARX MIMO, representado pela seguinte expressão (AGUIRRE, 2000):

$$y(k) = -A_1 y(k-1) - ... - A_{n_y} y(k-n_y) + B_1 u(k-1) + ... + B_{n_u} u(k-n_u) + \nu(k) \qquad (7.1)$$

onde $A_i \in \Re^{n_s \times n_s}$, $B_i \in \Re^{n_s \times n_e}$, n_u é o máximo atraso entre os regressores de entrada e n_y é o máximo atraso entre os regressores de saída, n_e é o número de entradas e n_s é o número de saídas. Os vetores de saída, entrada e ruído branco, $\boldsymbol{y}$, $\boldsymbol{u}$ e $\boldsymbol{v}$, respectivamente, são definidos da seguinte forma:

$$\begin{aligned} \boldsymbol{y}(k) &= [y_1(k)\ y_2(k) \ldots y_{n_s}(k)]^T \\ \boldsymbol{u}(k) &= [u_1(k)\ u_2(k) \ldots u_{n_e}(k)]^T \\ \boldsymbol{v}(k) &= [\nu_1(k)\ \nu_2(k) \ldots \nu_{n_s}(k)]^T \end{aligned} \qquad (7.2)$$

Pode-se definir as seguintes matrizes de polinômios

$$\begin{aligned} \boldsymbol{A}(q^{-1}) &= I + \boldsymbol{A}_1 q^{-1} + \cdots + \boldsymbol{A}_{n_y} q^{-n_y}; \\ \boldsymbol{B}(q^{-1}) &= \boldsymbol{B}_1 q^{-1} + \cdots + \boldsymbol{B}_{n_u} q^{-n_u} \end{aligned} \tag{7.3}$$

resultando na expressão:

$$A(q^{-1})y(k) = B(q^{-1})u(k) + \nu(k) \tag{7.4}$$

ou, alternativamente

$$\boldsymbol{y}(k) = \Theta^T \Psi(k-1) + \nu(k) \tag{7.5}$$

onde

$$\begin{aligned} \Theta &= [-A_1 \; -A_2 \; \ldots \; -A_{n_y} \; B_1 \ldots B_{n_u}]^T \\ \Psi(k-1) &= [y^T(k-1) \; \ldots \; y^T(k-n_y) \; u^T(k-1) \; \ldots \; u^T(k-n_u)]^T \end{aligned} \tag{7.6}$$

O processo de identificação consiste em uma etapa de ajuste dos pesos do modelo (treinamento) e outra etapa de teste do modelo identificado (validação). De acordo com a eq. (7.1), o modelo ARX para este processo é representado por:

$$\boldsymbol{y}(k) = -\boldsymbol{A_1} y(k-1) + \boldsymbol{B_1} u(k-1) + \boldsymbol{B_2} u(k-2) + \boldsymbol{B_3} u(k-3) \tag{7.7}$$

onde:

$$\boldsymbol{y} = [T_{rg1} \; T_{rg2} \; Severity \; T_{rx}] \quad ; \quad \boldsymbol{u} = [R_{ai} \; C_{TCV} \; R_{tf} \; T_{fp}]$$

E as matrizes $\boldsymbol{A_1}$, $\boldsymbol{B_1}$, $\boldsymbol{B_2}$ e $\boldsymbol{B_3}$ contêm os parâmetros identificados do modelo.

Apresenta-se a seguir o desempenho do modelo ARX MIMO frente aos dados de validação. Os dados de treinamento e validação são os obtidos do modelo fenomenológico de Moro e Odloack (1995). O sistema *stiff* de equações diferenciais do modelo foi resolvido através da sub-rotina SDRIV2 (KAHANER *et al.*, 1988).

O modelo linear conta com 3 regressores para as variáveis de entrada e 1 para as variáveis de saída. Escolheu-se apresentar esta arquitetura por ser semelhante à da melhor rede não-linear encontrada, para fins de comparação.

A tabela 7.3, a seguir, apresenta os erros quadráticos médios do modelo ARX em relação ao modelo fenomenológico para cada variável controlada considerada.

Tabela 7.3 EQM do modelo ARX MIMO

	EQM	
VARIÁVEIS CONTROLADAS	1 passo adiante	simulação recursiva
T_{rg1}	0,0138	0,1105
T_{rg2}	0,0212	0,2192
Severity	0,0573	0,0242
T_{rx}	0,0488	0,1215

As figuras 7.7 a 7.10 ilustram os resultados apresentados na tabela 7.3.

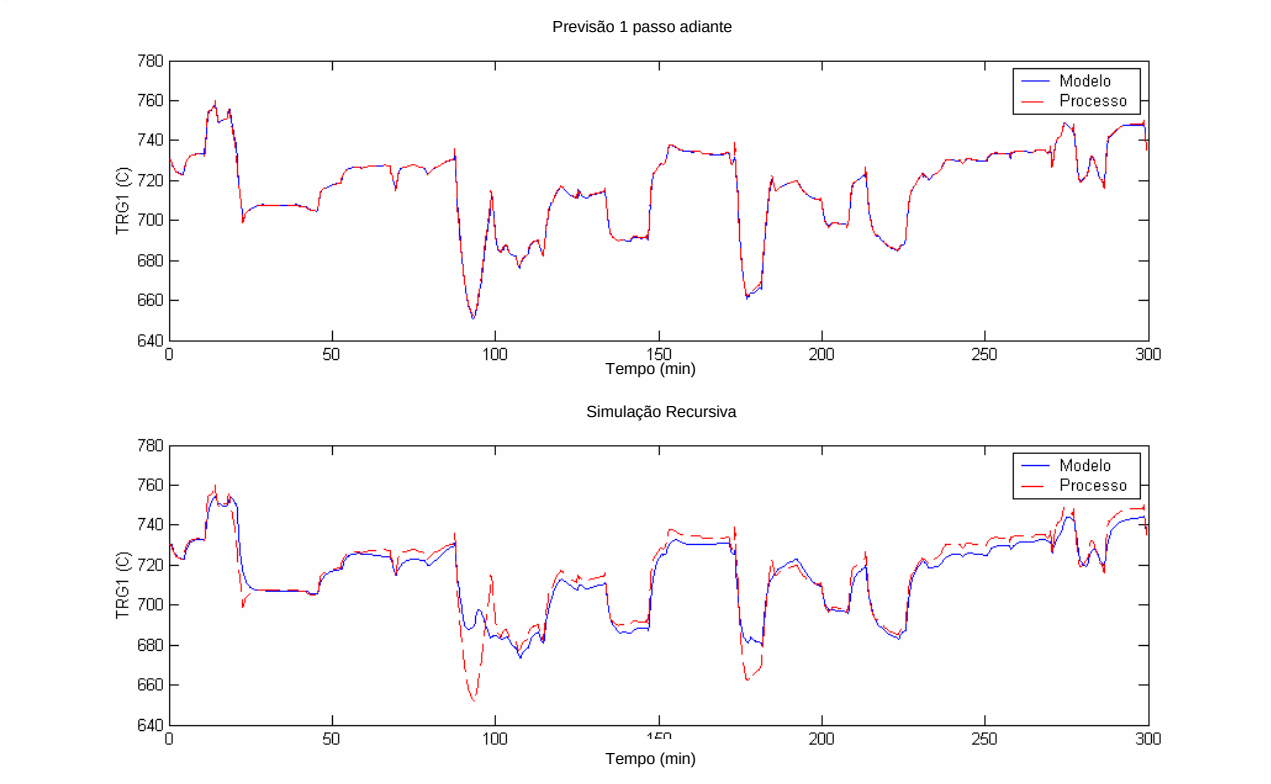

Figura 7.7 Desempenho modelo ARX para a variável T_{rg1}

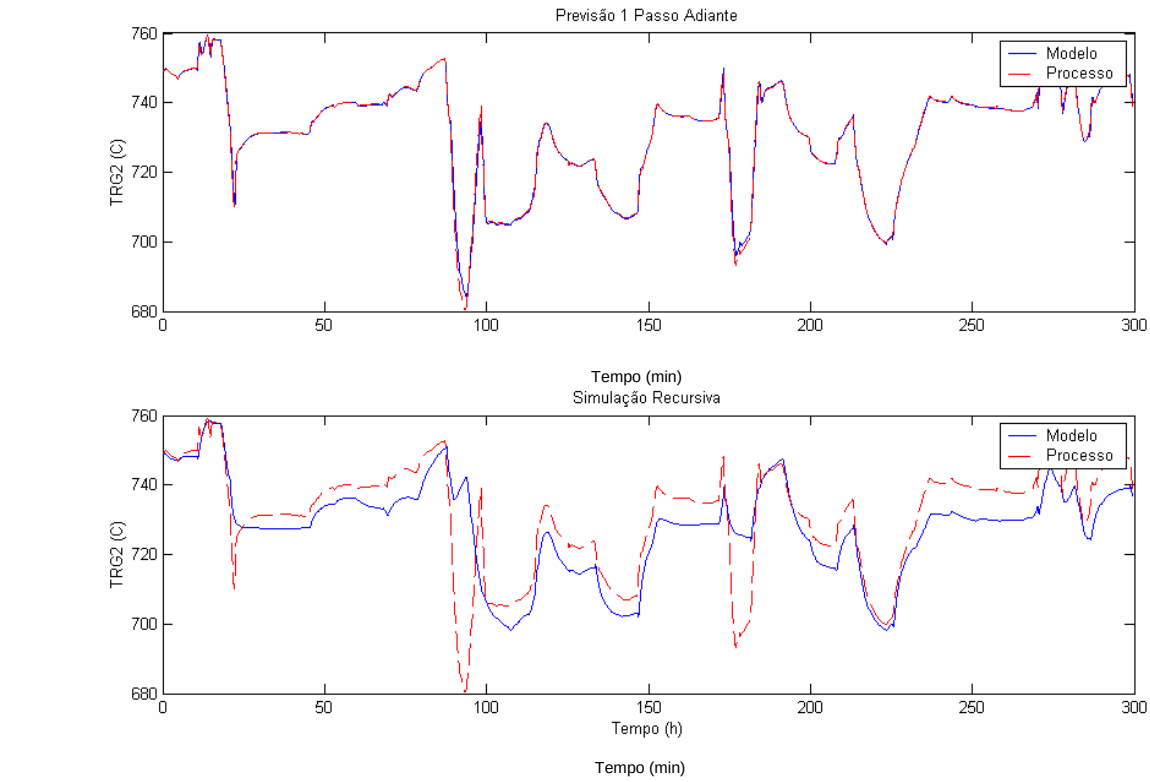

Figura 7.8 Desempenho modelo ARX para a variável T_{rg2}

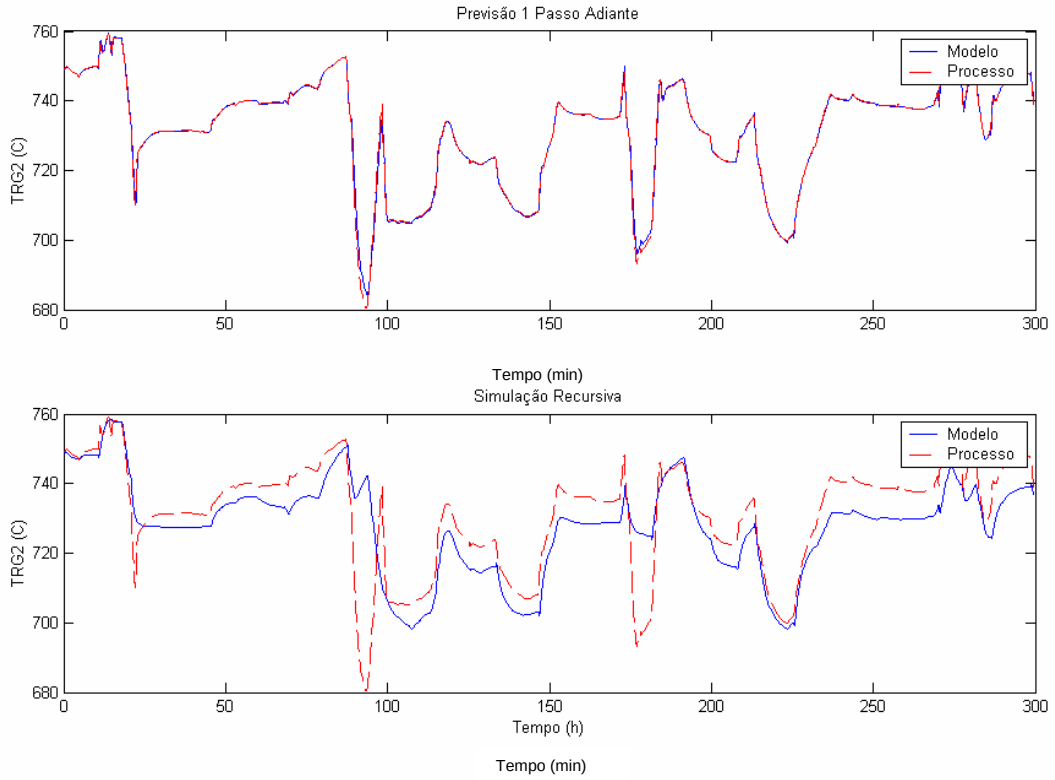

Figura 7.9 Desempenho modelo ARX para a variável *Severidade*

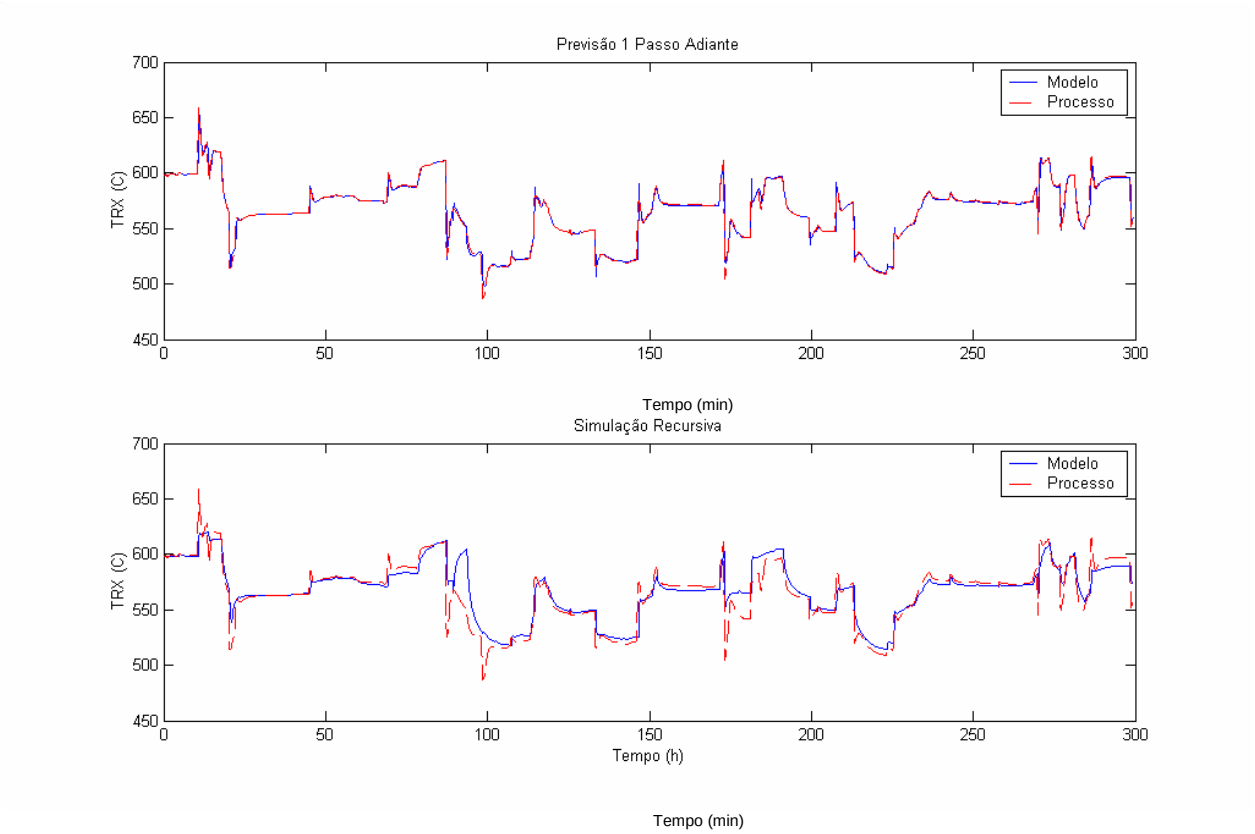

Figura 7.10 Desempenho modelo ARX para a variável T_{rx}

Embora o processo seja não linear, deve-se notar o comportamento quase perfeitamente simétrico das variáveis de saída quando foram aplicados degraus positivos e negativos nas variáveis de entrada (Figuras 7.1 a 7.4), comportamento característico de processos lineares. Esta é uma explicação provável para justificar o desempenho razoável do modelo linear ARX MIMO na simulação recursiva do processo.

Como o objetivo da obtenção do modelo é sua posterior implementação em um controlador preditivo do processo e em uma estratégia de otimização da FCCU, o erro do modelo ARX MIMO é maior do que o considerado satisfatório para tais propósitos.

7.5 IDENTIFICAÇÃO UTILIZANDO MODELOS NEURAIS MLP

Apresenta-se o resultado da modelagem do processo de craqueamento catalítico em leito fluidizado fornecido pelo modelo não linear representado por uma rede neural *feedforward* do tipo Perceptron Multicamadas (MLP).

7.5.1 MLP MISO

Foram inicialmente testadas redes MISO (4 entradas e uma saída) para cada variável controlada devido à maior simplicidade de um modelo MISO quando comparado com um modelo MIMO.

Avaliou-se a influência do número de iterações e da arquitetura das redes (número de regressores e de neurônios na camada oculta) na resposta do modelo, de modo que, variando um parâmetro, os demais foram mantidos constantes. Isto foi feito para cada uma das saídas consideradas.

As tabelas 7.4 e 7.5 apresentam os resultados encontrados para a variável *Severidade* da reação.

Tabela 7.4 Avaliação da influência do número regressores

	EQM	
nº regressores nas entradas	1 passo adiante	simulação recursiva
1	0,02	0,37
2	0,02	0,52
3	0,02	0,31

Tabela 7.5 Avaliação da influência do número neurônios

	EQM	
nº neurônios na camada oculta	1 passo adiante	simulação recursiva
2	0,02	0,55
5	0,02	0,40
10	0,02	0,43
15	0,02	0,60

As figuras 7.11 a 7.14 ilustram o desempenho do modelo MLP MISO para cada saída considerada, para uma rede com 15 neurônios, 1 regressor nas entradas e na saída.

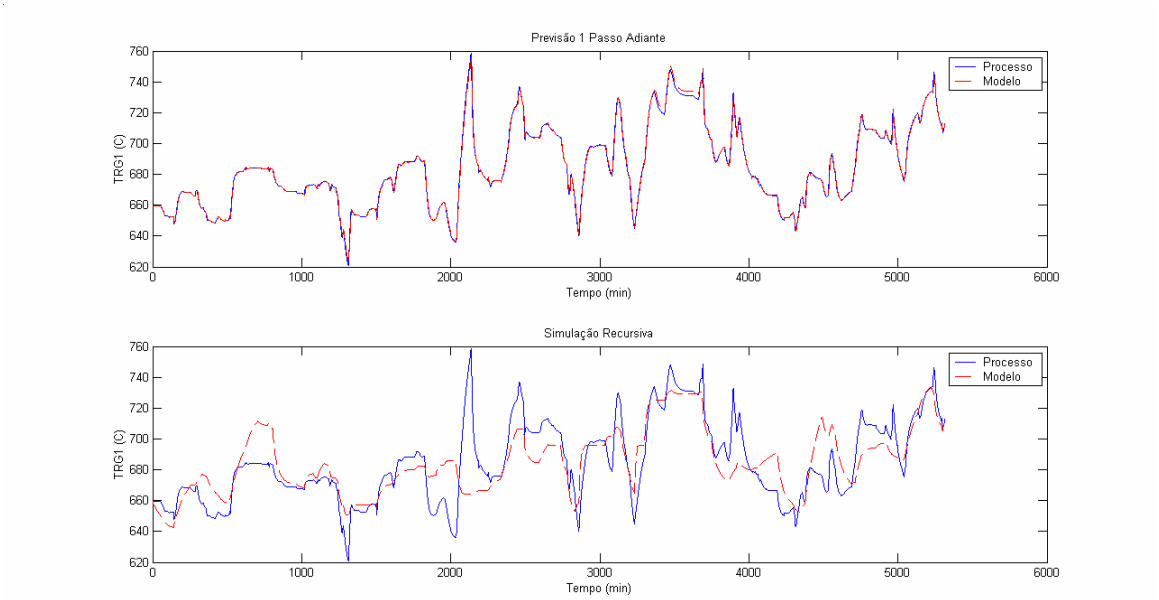

Figura 7.11 Desempenho do modelo MLP MISO (saída considerada: T_{rg1})

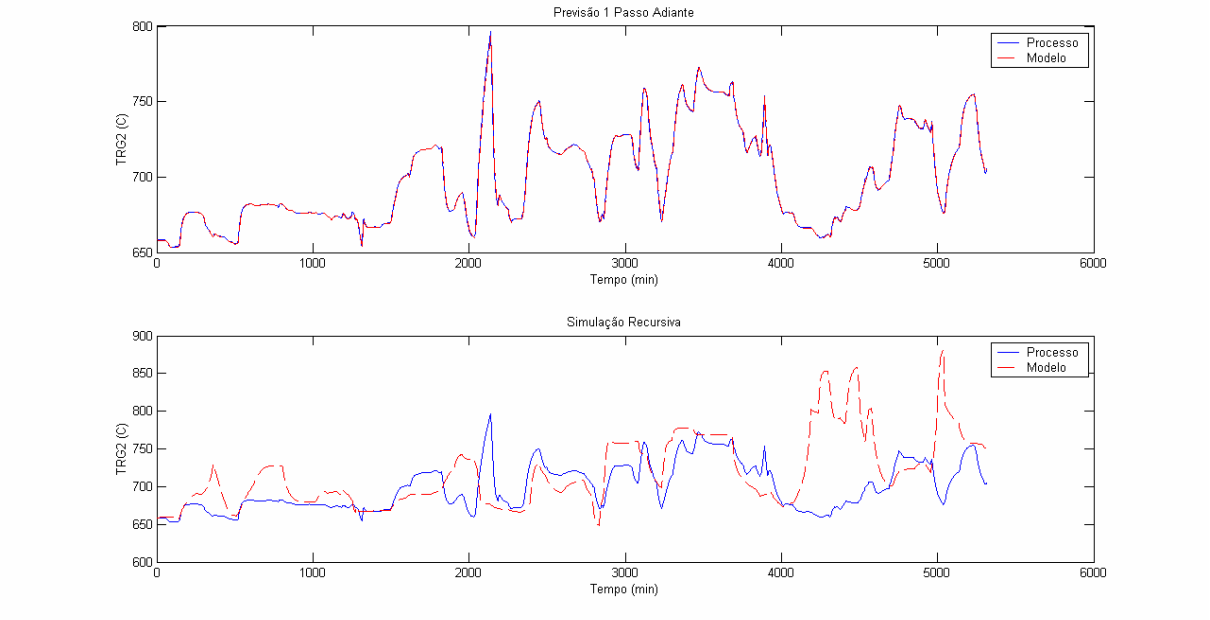

Figura 7.12 Desempenho do modelo MLP MISO (saída considerada: T_{rg2})

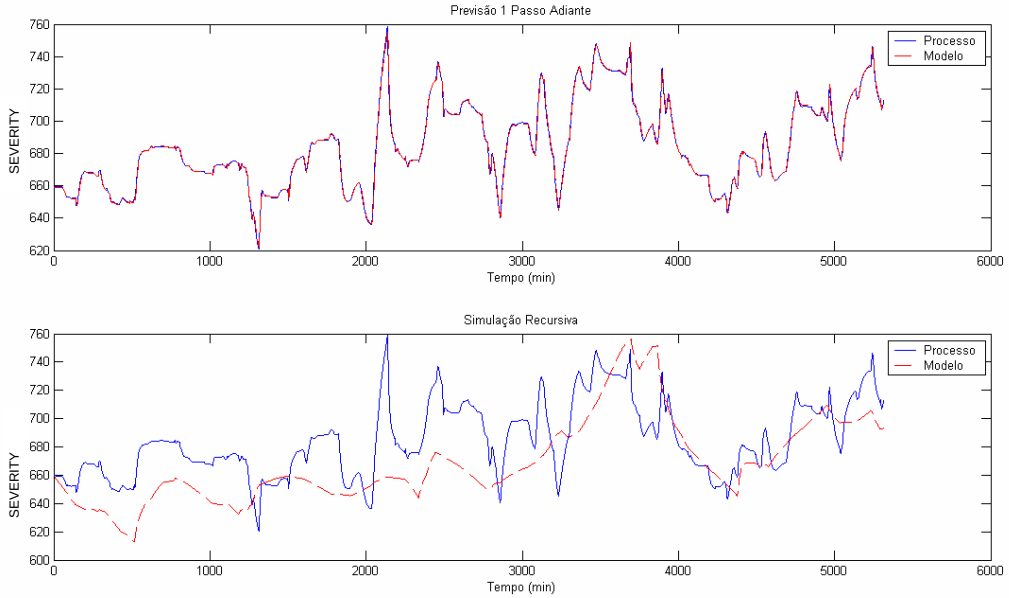

Figura 7.13 Desempenho do modelo MLP MISO (saída considerada: *Severidade*)

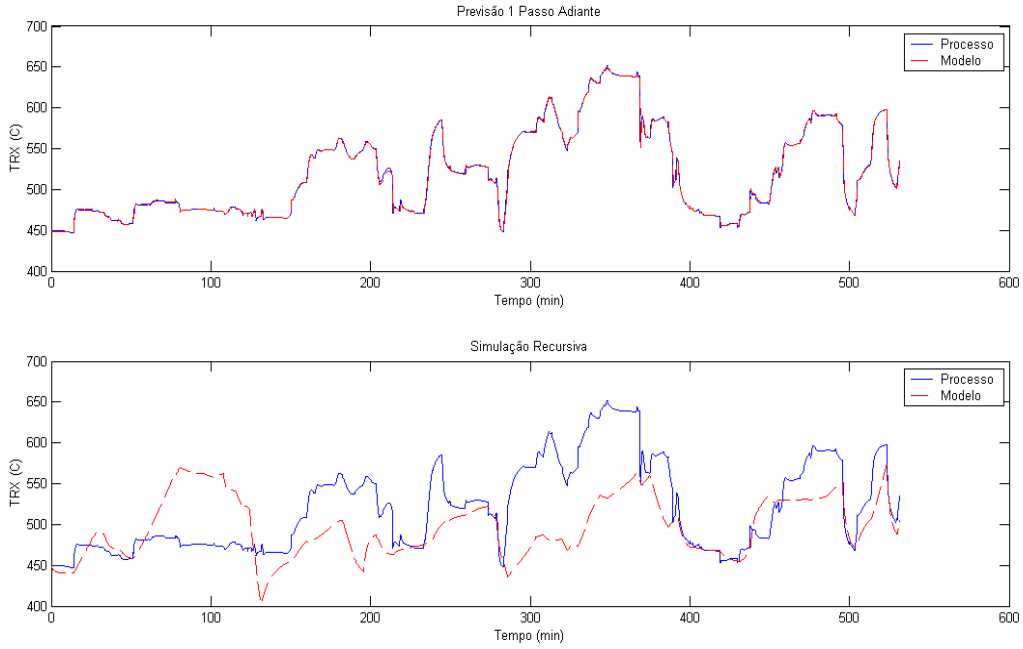

Figura 7.14 Desempenho do modelo MLP MISO (saída considerada: T_{rx})

Torna-se difícil avaliar a influência da arquitetura da rede na qualidade da resposta porque a relação não é linear. O resultado final depende de vários fatores aleatórios, como os valores iniciais de pesos usados no treinamento da rede. Não há como usar os mesmos pesos iniciais porque para cada arquitetura de rede há uma matriz de pesos específica. Os valores encontrados para as demais variáveis foram similares.

Os valores encontrados para as redes MIMO usando os mesmos números de regressores nas entradas e saídas do modelo são significativamente melhores. Isto isto sugere que, pelo fato da interação entre as variáveis do processo ser muito forte, não há condições de se obter um bom resultado desconsiderando esta interação, ou seja, usando redes MISO.

7.5.2 MLP MIMO

Os modelos neurais apresentaram melhor desempenho quando a representação MIMO foi utilizada. A determinação do número de regressores foi feita por tentativa e erro, variando-se o número de regressores para cada variável de 1 a 4, tanto nas entradas quanto nas saídas. O melhor resultado foi apresentado na Figura 7.15. Assim, a rede neural selecionada apresenta estrutura geral mostrada na Figura 7.15, com 16 entradas e 4 saídas.

As redes testadas utilizam funções de ativação do tipo tangente hiperbólica na camada escondida, l_e, e linear na camada de saída, l_s. O modelo neural MLP utilizado foi treinado com um algoritmo de otimização que utiliza o método do Gradiente Conjugado Escalonado Modificado – GCEM (DE CASTRO e VON ZUBEN,

1998a *apud* MELEIRO, 2002), que permite o cálculo exato da informação de 2ª ordem com custo computacional substancialmente reduzido. O número de neurônios da camada escondida, n_{l_e}, foi determinado através da comparação dos resultados obtidos para o conjunto de dados de validação quando diferentes valores de n_{l_e} foram testados.

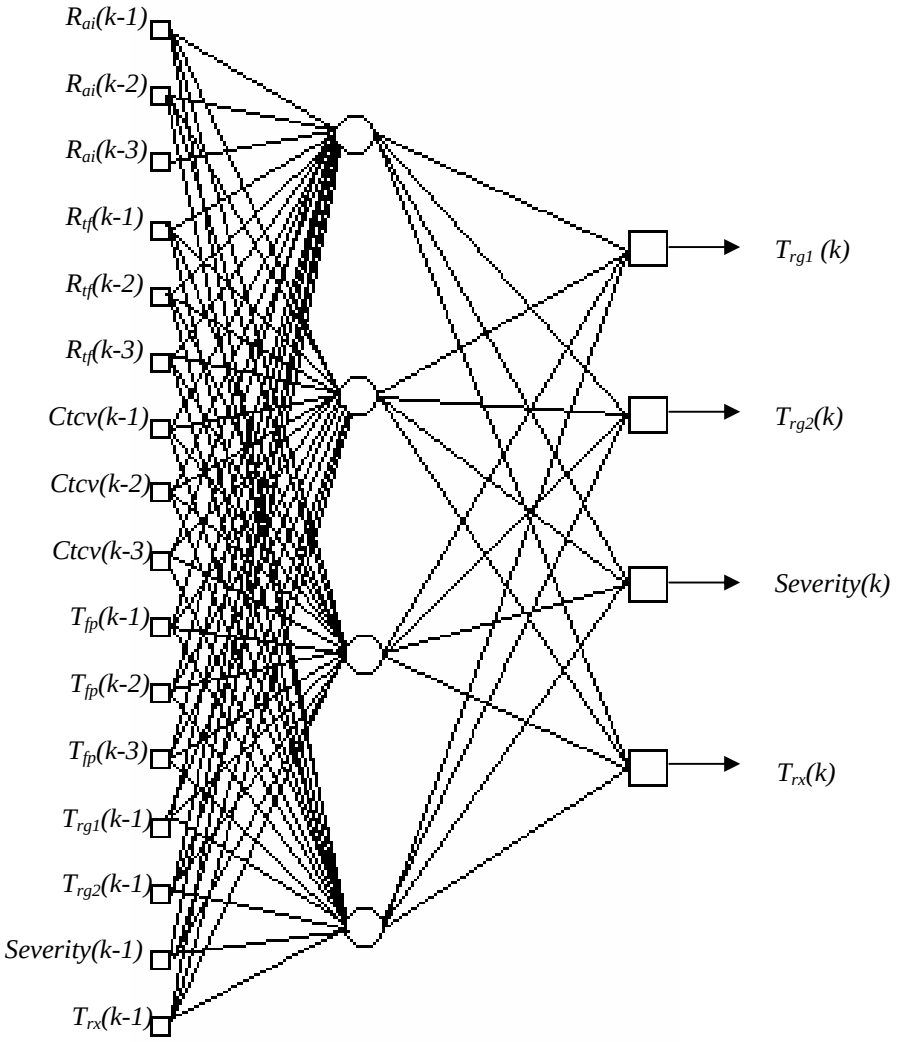

Figura 7.15 Melhor estrutura de rede MLP MIMO encontrada

A avaliação do desempenho dos modelos foi feita considerando os erros finais de aproximação (EQM – erro quadrático médio), como também através da análise visual de simulações para previsão um passo adiante e recursiva. A tabela

7.6 mostra os EQM para vários valores de n_{l_e}. As redes foram treinadas , no mínimo, três vezes cada, com um regressor nas saídas e três nas entradas do modelo e o melhor resultado obtido está apresentado nas figuras 7.17 a 7.21. Os valores assinalados correspondem às saídas da melhor rede encontrada. Os resultados diferem entre si devido caráter aleatório do modelo (razão pela qual as redes foram exaustivamente testadas).

Tabela 7.6 Valores de EQM para rede MLP MIMO com 1 regressor nas saídas e 3 regressores nas entradas do modelo

NÚMERO DE NEURÔNIOS NA CAMADA OCULTA (n_{l_e})	T_{rg1}			T_{rg2}			*Severity*			T_{rx}		
	EQM (1passo adiante)											
1	0,1613	0,1639	0,1613	0,0980	0,0992	0,1017	0,1348	0,1326	0,1345	0,1111	0,1094	0,1089
2	0,0315	0,0313	0,0353	0,1038	0,1056	0,0967	0,0766	0,0766	0,0823	0,0570	0,0570	0,0627
3	0,0161	0,0131	0,0151	0,0347	0,0318	0,397	0,0589	0,0581	0,0591	0,0560	0,0555	0,0556
4	0,0138	0,0114	0,0138	0,0209	0,0209	0,0234	0,0531	0,0539	0,0528	0,0452	0,0452	0,0447
5	0,0136	0,0150	0,0146	0,0245	0,0204	0,0233	0,0553	0,0537	0,0557	0,0441	0,0435	0,0452
6	0,0127	0,0135	0,0147	0,0214	0,0283	0,0261	0,0502	0,0535	0,0556	0,0443	0,0459	0,0457
7	0,0126	0,0135	0,0116	0,0197	0,0268	0,0229	0,0486	0,0560	0,0541	0,0392	0,0448	0,0455
8	0,0134	0,0129	0,0133	0,0211	0,0218	0,0249	0,0511	0,0551	0,0546	0,0425	0,0463	0,0469
10	0,0220	0,0150	0,0131	0,0199	0,0239	0,0221	0,0282	0,0547	0,0546	0,0207	0,0466	0,465
15	0,0139	0,0114	0,0148	0,0152	0,0158	0,0219	0,0554	0,0562	0,0552	0,0464	0,0453	0,0459
	EQM (simulação recursiva)											
1	0,1903	0,1942	0,1845	0,1614	0,1625	0,1597	0,1339	0,1342	0,1369	0,1198	0,1186	0,1230
2	0,0926	0,0928	0,1147	0,1588	0,1581	0,1757	0,1298	0,1212	0,1077	0,0872	0,0828	0,1047
3	0,1217	0,1156	0,1161	0,2151	0,2062	0,2053	0,1208	0,1168	0,1167	0,1053	0,0985	0,0981
4	0,0765	0,1084	0,0774	0,1344	0,1849	0,1364	0,0940	0,1049	0,0881	0,0740	0,0895	0,0733
5	0,1808	0,1363	0,1335	0,3183	0,2365	0,2440	0,3163	0,1724	0,1667	0,2799	0,1437	0,1409
6	0,1552	0,1207	0,1048	0,3125	0,2275	0,1963	0,3992	0,2116	0,1028	0,3432	0,1347	0,0965
7	0,2120	0,1084	0,0909	0,4345	0,2056	0,1681	0,4835	0,1248	0,1170	0,3731	0,1147	0,0805
8	0,1097	0,0991	0,1163	0,1955	0,1502	0,1952	0,1432	0,1404	0,1460	0,1231	0,1147	0,1333
10	0,3704	0,1032	0,0957	0,3447	0,1479	0,0138	0,4173	0,2750	0,0138	0,3546	0,1077	0,1134
15	0,0951	0,0784	0,1010	0,1263	0,1031	0,1453	0,1261	0,1495	0,1239	0,1038	0,1235	0,1047

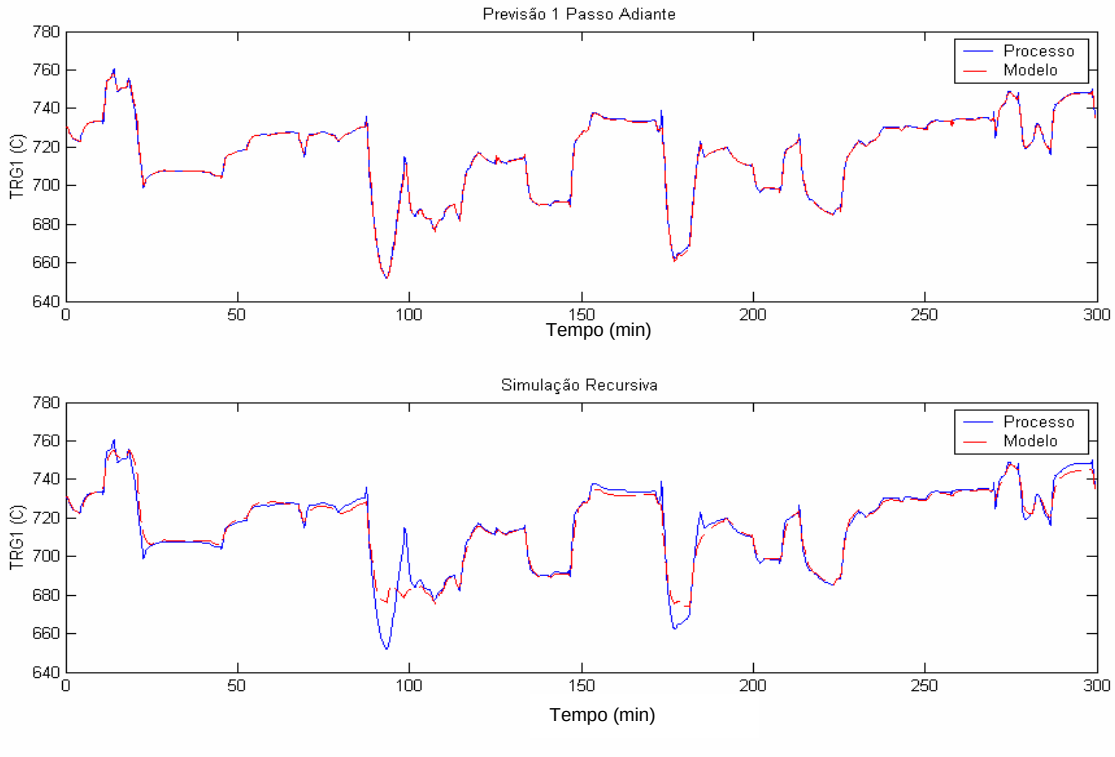

Figura 7.16 Desempenho modelo MLP MIMO para a variável T_{rg1}

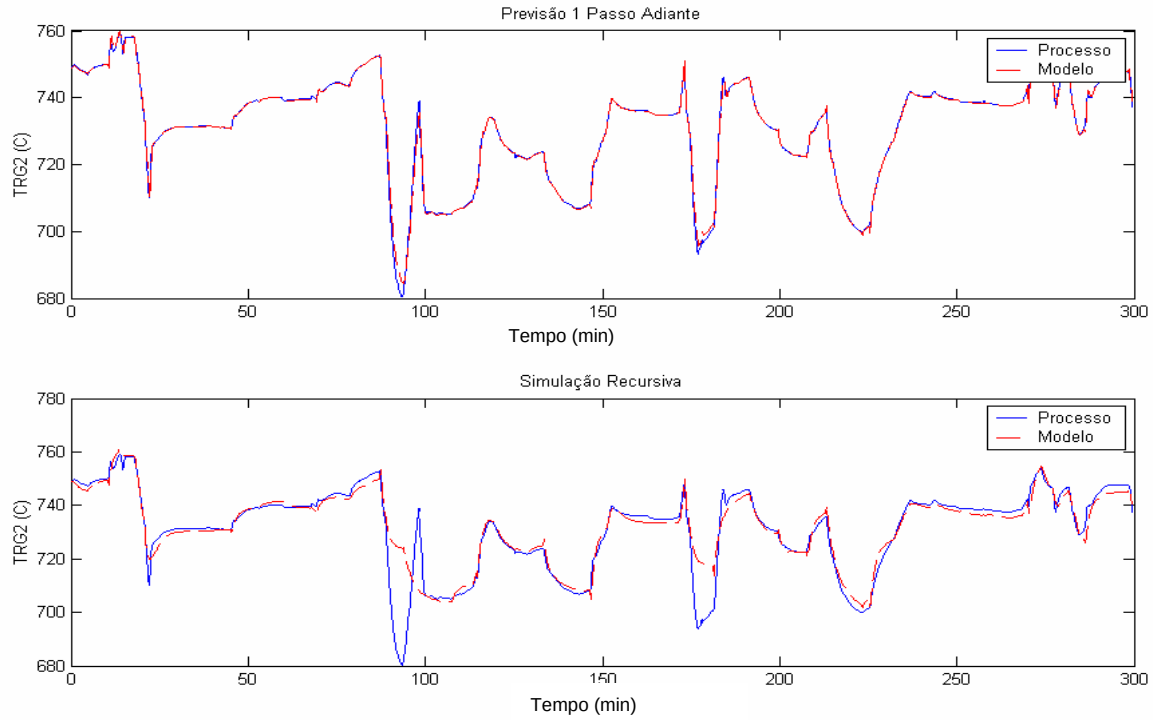

Figura 7.17 Desempenho modelo MLP MIMO para a variável T_{rg2}

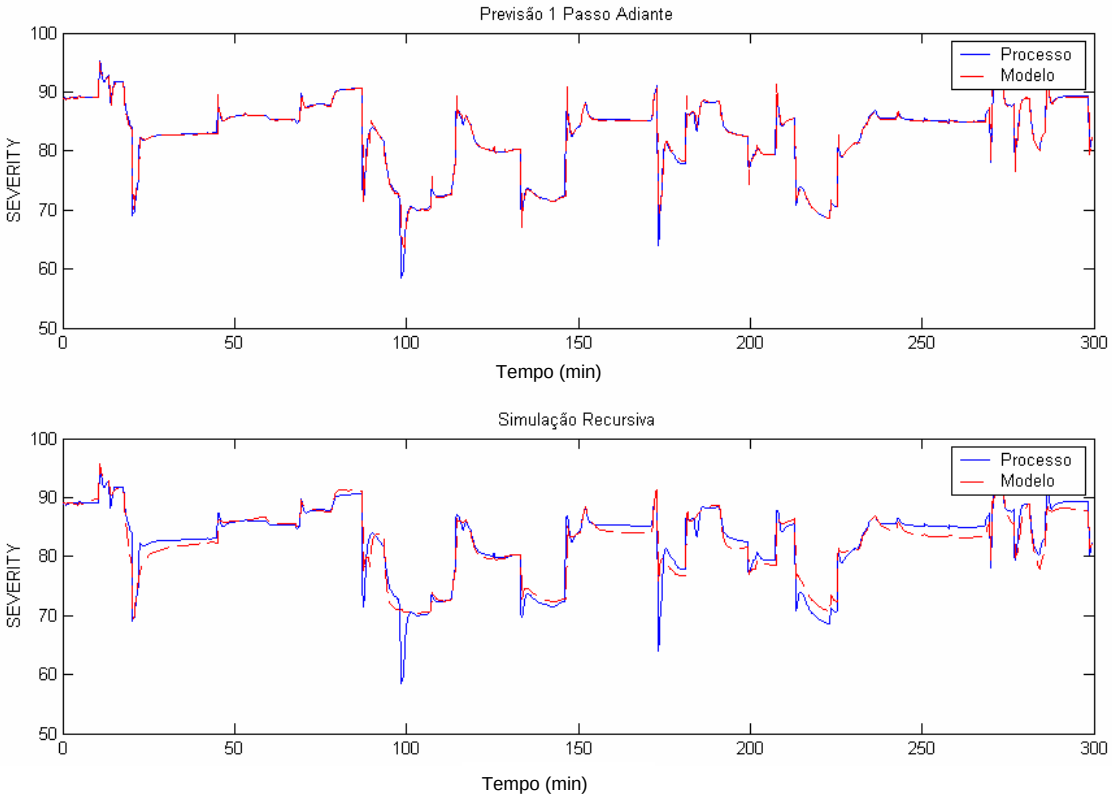

Figura 7.18 Desempenho modelo MLP MIMO para a variável *Severidade*

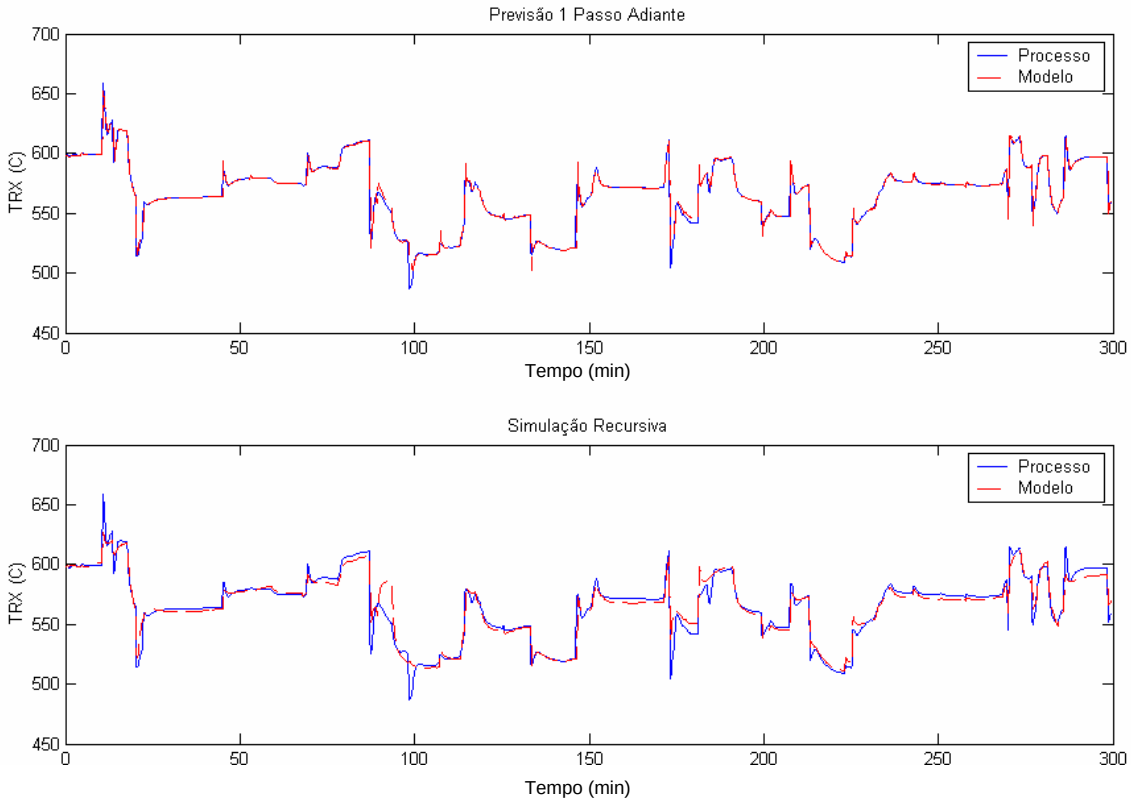

Figura 7.19 Desempenho modelo MLP MIMO para a variável T_{rx}

Nota-se que o desempenho do modelo MLP MIMO é melhor que o ARX MIMO pelos motivos já comentados. Foram testadas várias configurações com diferente número de regressores nas entradas. Mas, como visto anteriormente (Tabela 7.2) o processo responde de modo similar a estímulos em todas as variáveis de entrada. Ficou claro, diante dos resultados obtidos durante a pesquisa, que as melhores respostas eram obtidas com igual número de regressores para as variáveis de entrada.

Quanto ao número de regressores nas saídas, os resultados foram sensivelmente piores quando este número foi aumentado, por isso utilizou-se apenas um regressor nas variáveis de saída. Este procedimento minimiza a realimentação de erros de modelagem. Como ilustração, segue-se as figuras 7.20 a 7.23, referentes às respostas de uma rede com 2 regressores nas variáveis de saída.

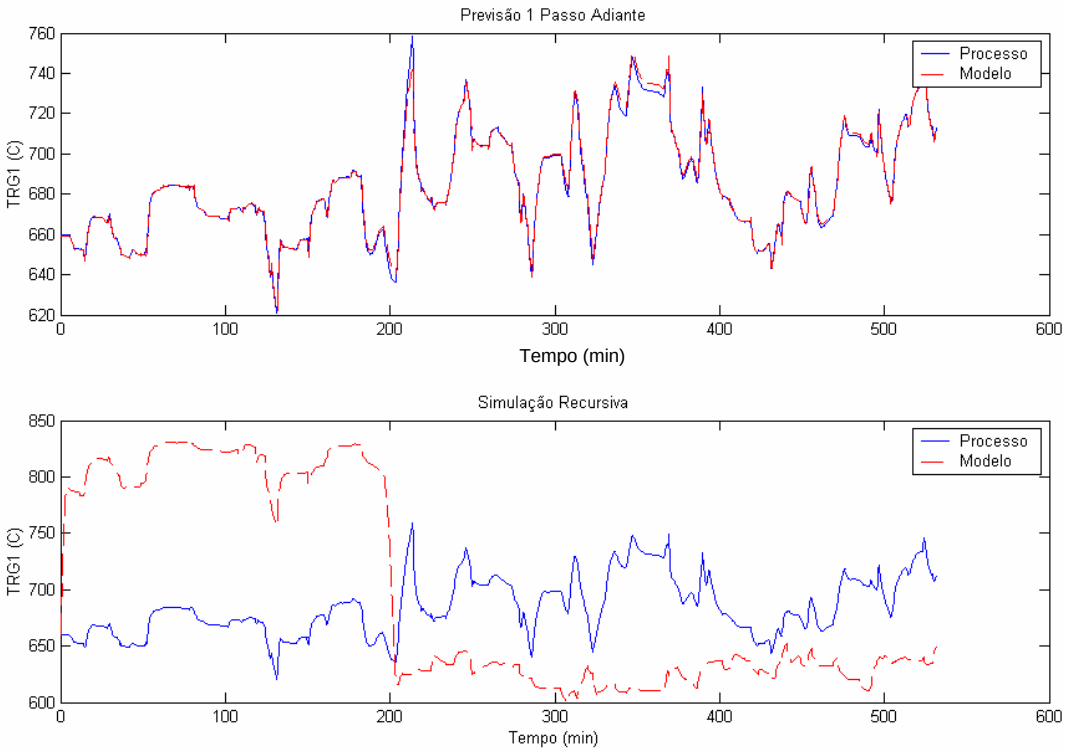

Figura 7.20 Desempenho modelo MLP MIMO para a variável T_{rg1} com 2 regressores nas variáveis de saída

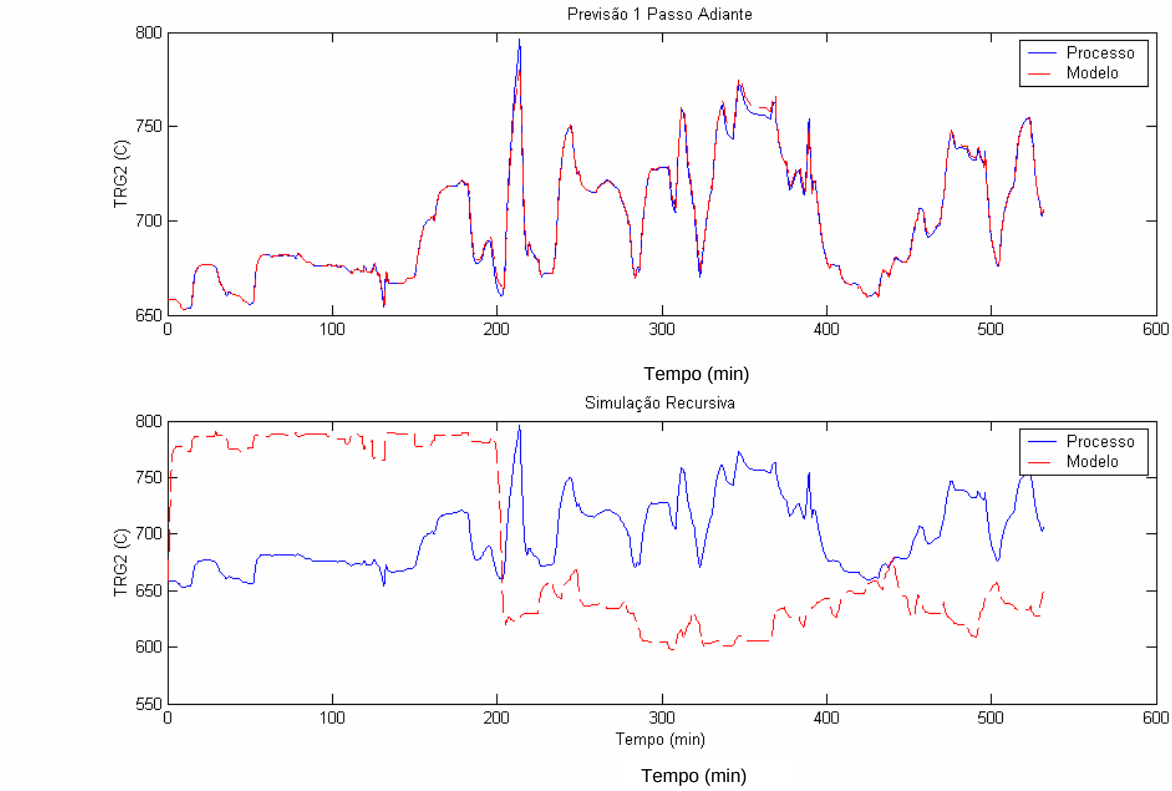

Figura 7.21 Desempenho modelo MLP MIMO para a variável T_{rg2} com 2 regressores nas variáveis de saída

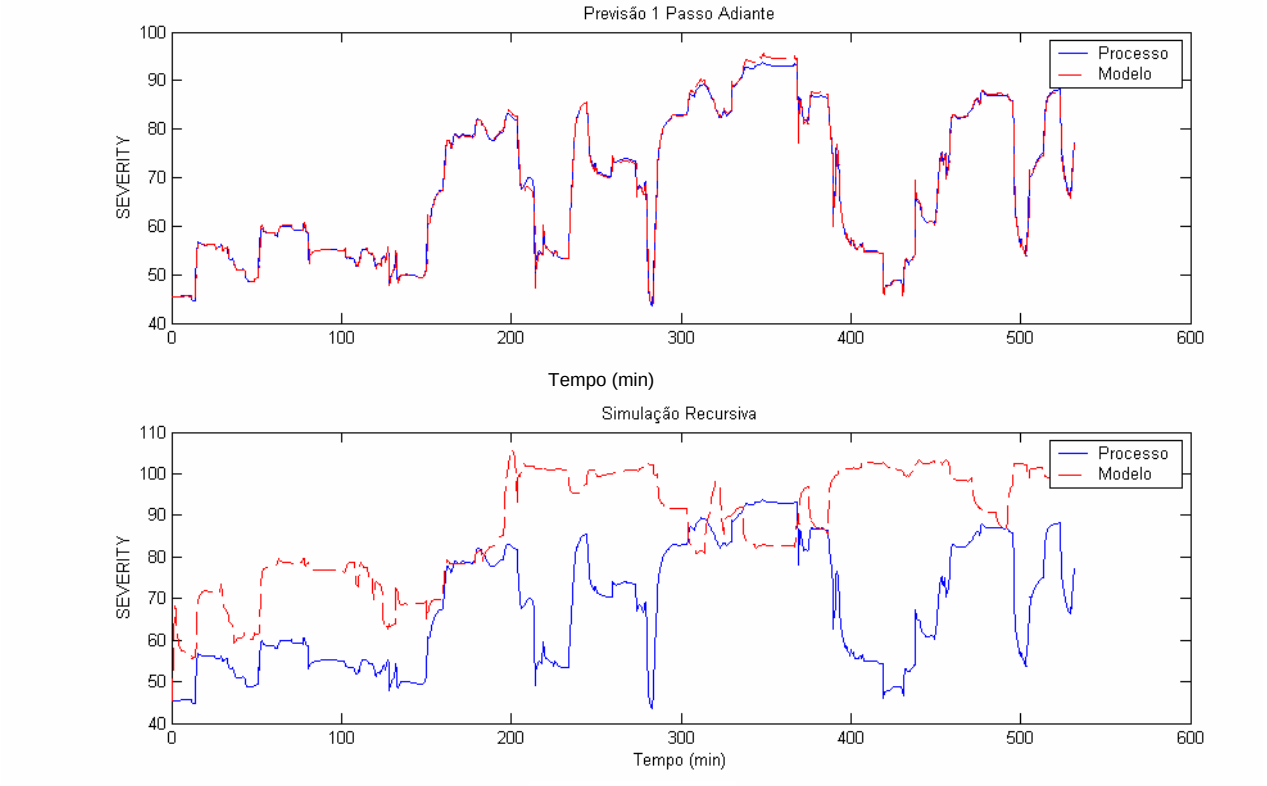

Figura 7.22 Desempenho modelo MLP MIMO para a variável *Severidade* com 2 regressores nas variáveis de saída

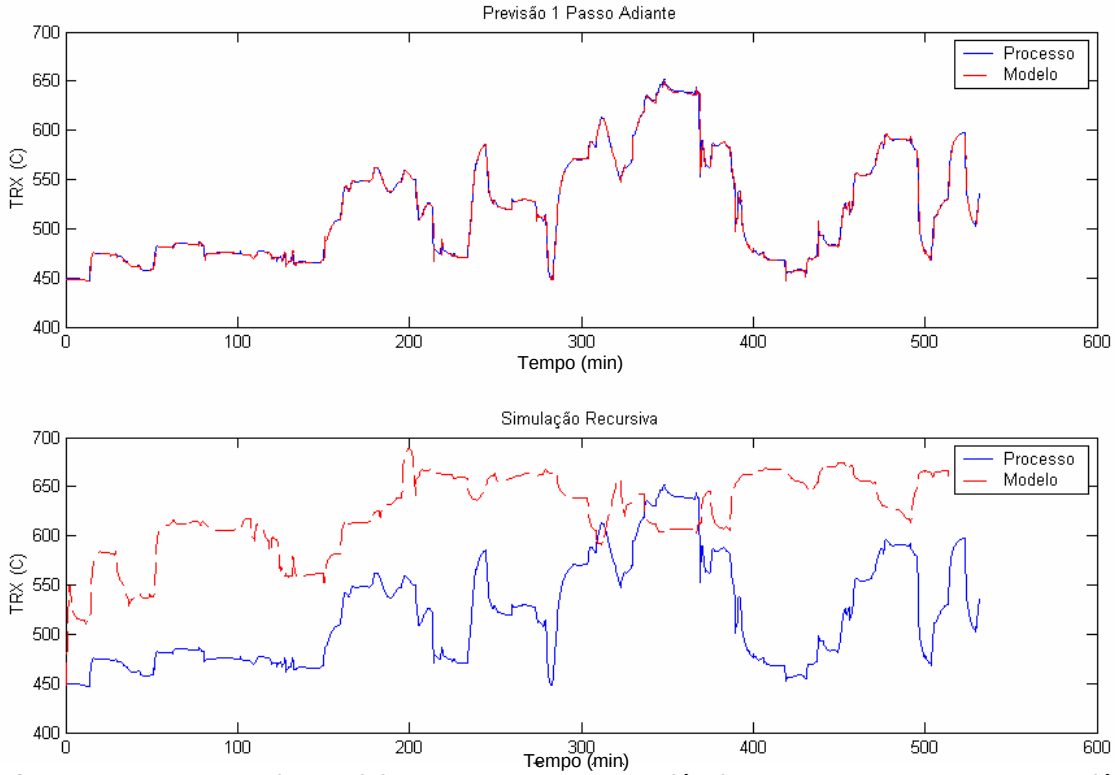

Figura 7.23 Desempenho modelo MLP MIMO para a variável T_{rx} com 2 regressores nas variáveis de saída

O erro na série recursiva dá-se, principalmente, por causa da realimentação do erro de predição na entrada do modelo. A tabela 7.7, a seguir, compara os erros entre redes com 1 e 2 regressores nas saídas, com o mesmo número de neurônios e com 1 regressor nas entradas.

Tabela 7.7 Comparação entre redes com 1 e 2 regressores nas saídas

		EQM			
REGRESSORES NAS SAÍDAS		T_{rg1}	T_{rg2}	*Severity*	T_{rx}
1	1 passo adiante	**0,02**	**0,02**	**0,03**	**0,02**
	Simulação recursiva	**0,62**	**0,65**	**0,61**	**0,83**
2	1 passo adiante	**0,02**	**0,02**	**0,03**	**0,02**
	Simulação recursiva	**1,18**	**0,96**	**0,76**	**0,98**

A fim de aumentar o desempenho da melhor rede encontrada (4 neurônios na camada oculta, 3 regressores nas entradas e 1 nas saídas do modelo), o horizonte de previsão foi diminuído. Este assunto foi discutido na seção 3.2.3.2.2. O erro é

significativamente menor quanto menor for o horizonte de predição. No entanto, o uso de horizontes muito pequenos não é interessante para posterior implementação em controladores preditivos. Na tabela 7.8 são apresentados valores de EQM para vários horizontes.

Tabela 7.8 Avaliação de EQM para diferentes horizontes

k (simulação k passos adiante)	**EQM**			
	T_{rg1}	T_{rg2}	*Severity*	T_{rx}
1	0,01	0,02	0,05	0,04
5	0,02	0,06	0,07	0,06
10	0,04	0,08	0,08	0,07
20	0,06	0,11	0,08	0,07
50	0,07	0,12	0,09	0,07
100	0,08	0,14	0,09	0,07
∞	0,08	0,13	0,09	0,07

Apresentam-se a seguir as figuras que ilustram o desempenho do modelo com simulação 10 passos adiante.

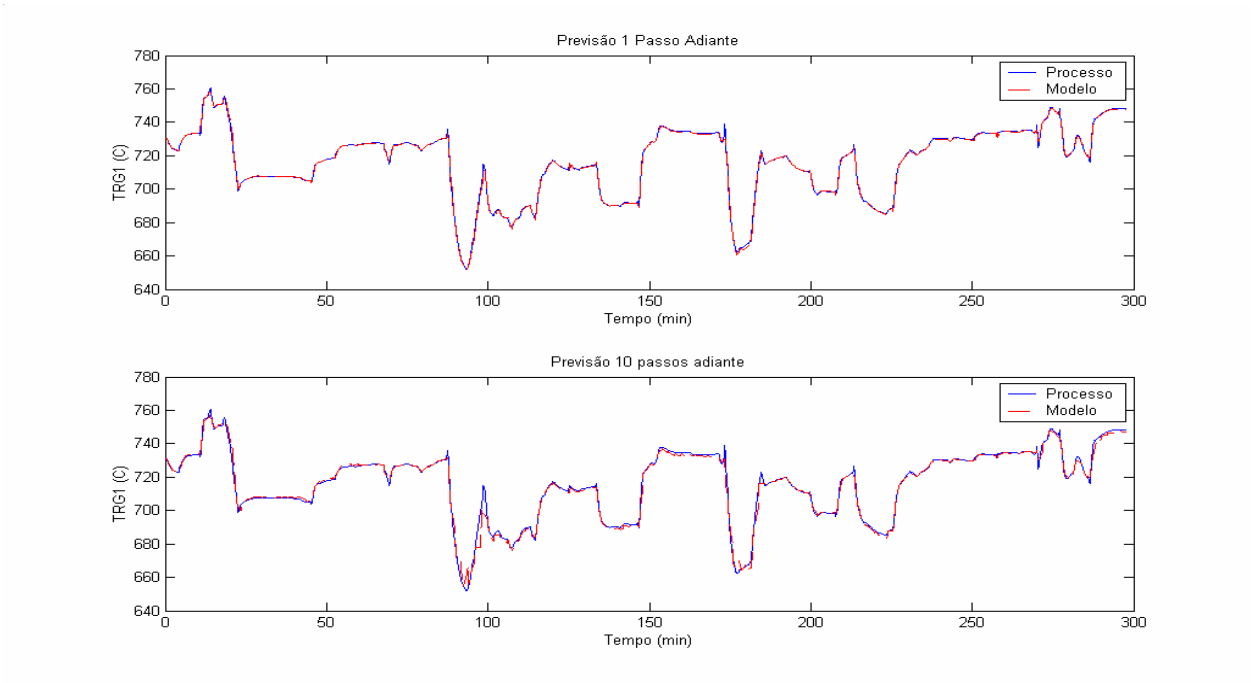

Figura7.24 Desempenho modelo MLP MIMO para a variável T_{rg1} para 1 e 10 passos adiante

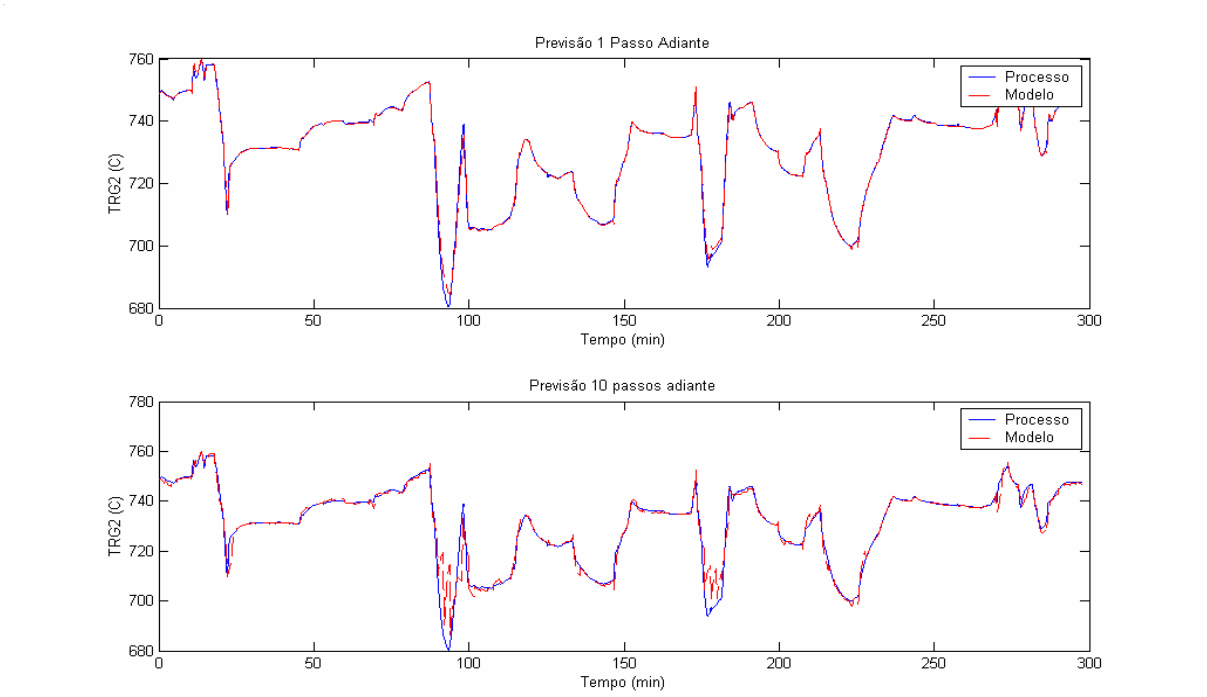

Figura 7.25 Desempenho modelo MLP MIMO para a variável T_{rg2} para 1 e 10 passos adiante

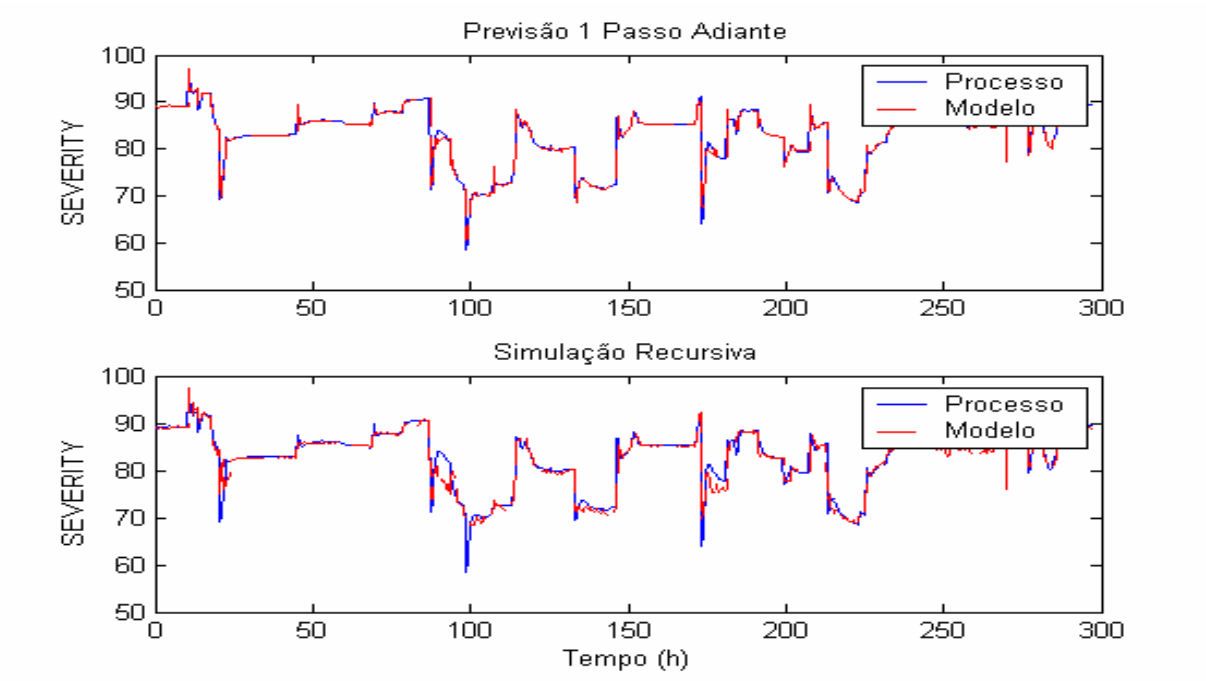

Figura 7.26 Desempenho modelo MLP MIMO para a variável *Severidade* para 1 e 10 passos adiante

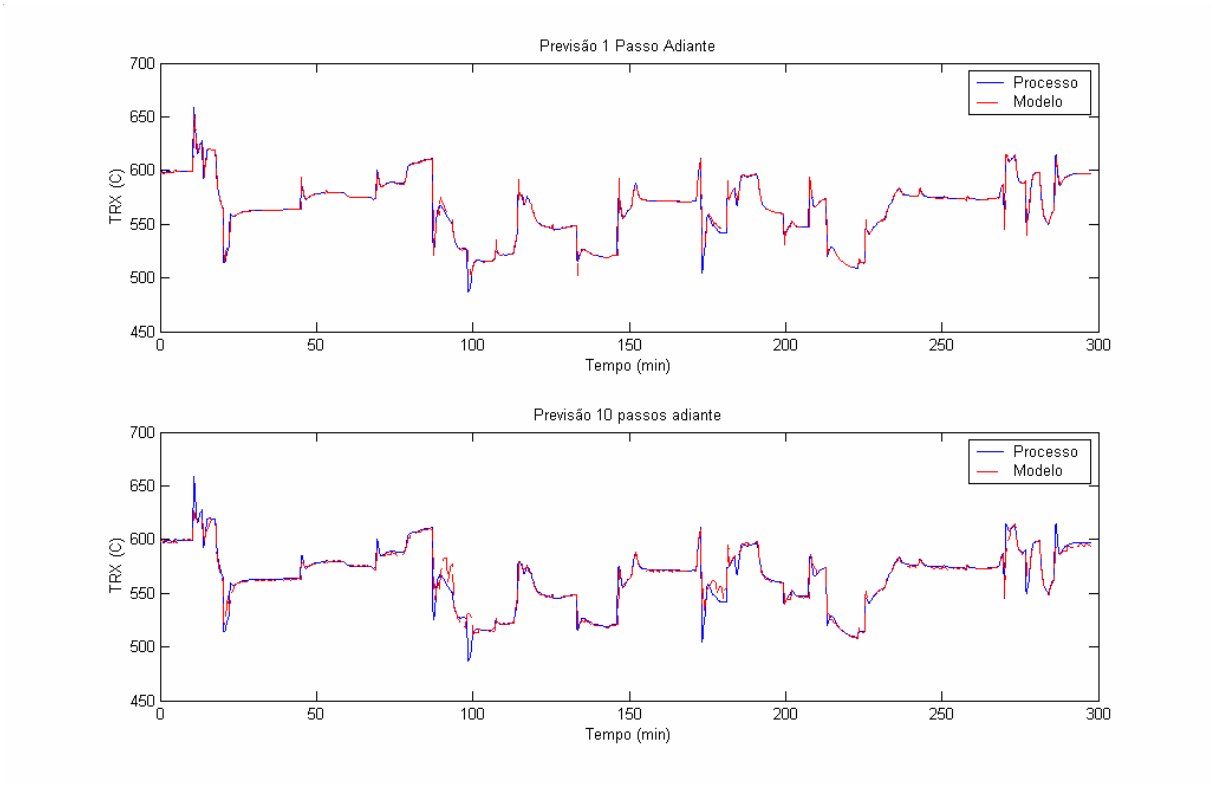

Figura 7.27 Desempenho modelo MLP MIMO para a variável T_{rx} para 1 e 10 passos adiante

7.6 OTIMIZAÇÃO DO PROCESSO USANDO ALGORITMOS GENÉTICOS

É bem estabelecido na literatura que o algoritmo genético e suas adaptações são bastante robustos e fornecem uma resposta razoável para vários problemas de otimização. Muitos estudos demonstram que vários problemas reais são melhor resolvidos usando AG em comparação com outros métodos como a programação quadrática (SQP –*Sequential Quadratic Programming*) (KASAT *et al*., 2002).

Neste trabalho, o AG foi adaptado de modo a determinar o conjunto de variáveis manipuladas adequado para levar a FCCU (variáveis controladas) do estado estacionário de referência a um outro ponto de operação previamente determinado. Para tanto, o modelo neural da FCCU foi utilizado pelo AG para

simular a mudança de ponto de operação do processo, a partir do estado estacionário de referência, quando submetido às mudanças nas variáveis manipuladas determinadas pelo algoritmo de otimização.

Apresenta-se a seguir uma descrição resumida dos parâmetros do AG adaptado: cada indivíduo do AG representa um conjunto de variáveis manipuladas [R_{ai}, c_{TCV}, R_{tf}, T_{fp}] gerado pelo algoritmo. O melhor indivíduo, selecionado pelos operadores genéticos do algoritmo, será a solução do problema. Esta solução, por sua vez, corresponde aos valores das variáveis manipuladas que levarão a FCCU a operar na faixa de operação previamente especificada pelo operador do processo.

No estudo apresentado a seguir, optou-se por fixar o valor da temperatura de reação no *riser* em T_{rx_SET} = 543,5 ºC. Esta variável foi escolhida por ter bastante relevância no processo de craqueamento, influenciando diretamente a conversão da reação. As demais variáveis foram mantidas dentro de sua faixa de operação (Tabela 7.1).

População é o conjunto de indivíduos, cada um representando, neste caso, um conjunto de variáveis manipuladas cujos valores iniciais foram gerados aleatoriamente pelo AG. A diversidade de cada população é garantida por um mecanismo que mantém a população com um número mínimo de indivíduos. Caso a população fique menor que este mínimo estabelecido, devido aos critérios de avaliação, o algoritmo cria novos indivíduos aleatoriamente para que o número mínimo seja atingido.

O erro admitido, que também servirá como critério de desempenho, é dado por:

$$erro = \frac{\left| T_{rx\_SET} - T_{rx\_calculado} \right|}{T_{rx\_SET}} \leq 2\%$$

ALGORITMO GENÉTICO ADAPTADO

1. Inicializar a população aleatoriamente.
2. Simular o processo com o modelo neural para cada uma das condições operacionais fornecida pelos indivíduos (conjunto de variáveis manipuladas) e obter as saídas do processo, ou seja, o conjunto de variáveis controladas.
   2.1. Verificar se há saídas cujos valores estão fora dos limites operacionais estabelecido.
   2.2. Eliminar os indivíduos que produziram saídas fora dos limites operacionais.
3. Incluir novos indivíduos, gerados aleatoriamente, até que o tamanho mínimo da população seja atingido.
4. Avaliar a população através do critério de desempenho e armazenar o melhor indivíduo.
5. Iniciar os Operadores Genéticos:
   5.1. Selecionar os indivíduos pais para produzir a nova geração.
      5.1.1. SELEÇÃO dos pais por torneio.
      5.1.2. PERMUTAÇÃO para obter os indivíduos filhos.
      5.1.3. MUTAÇÃO nos indivíduos filhos.
      5.1.4. Criação da nova geração com os indivíduos filhos.
   5.2. Utilizar o conjunto de variáveis manipuladas fornecido pelos indivíduos da nova geração para calcular as saídas do processo com auxílio do modelo neural do processo.
      5.2.1. Verificar se há saídas cujos valores estão fora dos limites operacionais estabelecido.
      5.2.2. Eliminar os indivíduos que produziram saídas fora dos limites operacionais.

5.3. Incluir novos indivíduos, gerados aleatoriamente, até que o tamanho mínimo da população seja atingido.

5.4. Escolher o melhor indivíduo por ELITISMO, considerando a participação do melhor indivíduo da geração anterior.

6. Repetir o passo 5 por 10 gerações.

7. Repetir os passos 1 ao 6 enquanto *erro* > 2%.

Apresentam-se a seguir os resultados encontrados pelo algoritmo genético modificado, com uma população inicial de 30 indivíduos, com um mínimo de 20 indivíduos, acoplado com o modelo neural do processo, usando os operadores de seleção por torneio, combinação cruzada e mutação, além de um critério elitista, como apresentado nos parágrafos anteriores.

Estabeleceu-se como critério de parada o valor do erro entre o valor da temperatura na região do *riser* (T_{rx}) e o valor calculado pela rede neural. O algoritmo retorna os valores das entradas (variáveis manipuladas) que conduzem ao valor determinado para T_{rx}. O conjunto de variáveis manipuladas geradas pelo algoritmo genético foram, então, testadas no modelo fenomenológico do processo para verificar a qualidade da solução fornecida pelo AG. O erro, estabelecido como critério de parada do algoritmo, foi $\leq$ 2%. Os resultados estão na tabela 7.9. A tabela 7.10 apresenta vários resultados da otimização utilizando AG, referentes a vários valores de T_{rx} apresentados como objetivo da otimização.

Tabela 7.9 Resultados da otimização utilizando Algoritmos Genéticos

		Algoritmo Genético	Modelo Fenomenológico	Erro relativo (%)
variáveis controladas (saídas)	T_{rg1} (ºC)	680,10	676,40	0,55
	T_{rg2} (ºC)	714,49	705,80	1,23
	Severidade	74,49	75,90	1,86
	T_{rx} (ºC)	533,68	533,99	0,06
Variáveis manipuladas (entradas)	R_{ai} (Ton/h)	201,00	201,00	
	c_{TCV} (%)	63,70	63,70	
	R_{tf} (Ton/h)	8487,50	8487,50	
	T_{fp} (ºC)	226,90	226,90	

Tabela 7.10 Resultados do AG otimizando uma variável ($T_{rx} = 543{,}5ºC$)

		Teste1			Teste2			Teste3		
		AG	Real	Erro(%)	AG	Real	Erro(%)	AG	Real	Erro(%)
variáveis controladas (saídas)	T_{rg1}	663,78	657,36	0,98	721,98	723,14	0,16	744,09	743,67	0,06
	T_{rg2}	711,92	687,67	3,53	742,42	743,99	0,21	751,22	750,39	0,11
	Sev	75,44	75,12	0,43	86,62	86,30	0,36	89,31	89,25	0,06
	T_{rx}	543,98	530,73	2,50	588,25	584,18	0,70	602,95	599,62	0,55
Variáveis manipuladas (entradas)	R_{ai}	216,07	216,07		226,22	226,22		231,00	231,00	
	c_{TCV}	0,82	0,82		0,74	0,74		0,70	0,70	
	R_{tf}	9840,00	9840,00		7584,46	7584,46		6477,25	6477,25	
	T_{fp}	223,25	223,25		231,36	231,36		221,33	221,33	
		Teste4			Teste5			Teste6		
		AG	Real	Erro(%)	AG	Real	Erro(%)	AG	Real	Erro(%)
variáveis controladas (saídas)	T_{rg1}	724,64	727,23	0,36	715,63	717,04	0,20	683,19	679,89	0,49
	T_{rg2}	741,31	742,82	0,20	740,92	741,85	0,13	724,23	710,02	2,00
	Sev	86,82	86,69	0,14	86,91	87,22	0,36	81,70	81,72	0,03
	T_{rx}	581,65	583,01	0,23	589,42	587,80	0,27	571,98	557,27	2,64
Variáveis manipuladas (entradas)	R_{ai}	212,45	212,45		210,31	210,31		221,15	221,15	
	c_{TCV}	0,65	0,65		0,74	0,74		0,86	0,86	
	R_{tf}	6797,41	6797,41		7263,46	7263,46		8869,26	8869,26	
	T_{fp}	225,00	225,00		228,31	228,31		221,82	221,82	

7.7 OTIMIZAÇÃO DO PROCESSO USANDO ENXAME DE PARTÍCULAS

No algoritmo PSO clássico as variáveis podem assumir qualquer valor, como apresentado na seção 6.3. Hassan e colaboradores (2004) recomendam uma modificação no algoritmo, sugerindo que, quando as variáveis violarem seus limites máximos ou mínimos elas sejam artificialmente trazidas de volta para dentro dos limites. Neste trabalho, entretanto, optou-se por descartá-las. Impôs-se restrições às variáveis (os próprios limites de operação) e qualquer resposta que violasse estes limites eram descartadas de imediato. Além disso, a rede neural identificada (item 7.5.2) foi implementada no código, de modo similar ao AG, para fornecer a resposta do processo, dado o conjunto de entradas geradas pelo algoritmo PSO. Cada partícula corresponde a um conjunto de variáveis manipuladas cujos valores iniciais foram gerados pelo algoritmo, similar ao indivíduo do algoritmo genético.

Tomou-se como primeiro objetivo da otimização a temperatura do *riser*, T_{rx}= 543,5ºC (tabela 7.11). Ou seja, o algoritmo deveria fornecer como resposta o conjunto de variáveis manipuladas do processo para obter o valor desejado de T_{rx}. Os algoritmos de otimização fornecem apenas as variáveis manipuladas. As variáveis controladas são fornecidas pelo modelo neural.

Os parâmetros do algoritmo PSO foram (Eq. 6.12 e 6.14): $c_1 = 0{,}9$, $c_2 = 1{,}1$, $n = 100$ e $m = 1$, onde c_1 é o fator de individualidade, c_2 é o fator de identidade de grupo, n é o número de partículas iniciais e m é o número de iterações.

O algoritmo PSO clássico foi adaptado para o problema apresentado como segue:

ALGORITMO PSO ADAPTADO

1. Inicializar o conjunto de partículas aleatoriamente, com velocidades iguais a zero.
2. Simular o processo com o modelo neural para cada uma das condições operacionais fornecida pelas partículas (conjunto de variáveis manipuladas) e obter as saídas do processo, ou seja, o conjunto de variáveis controladas.
   2.1. Verificar se há saídas cujos valores estão fora dos limites operacionais estabelecido.
   2.2. Eliminar as partículas que produziram saídas fora dos limites operacionais.
3. Avaliar o enxame através do critério de desempenho e armazenar a melhor partícula.
4. Atualizar as partículas (posição e velocidade).
   4.1. Utilizar o conjunto de variáveis manipuladas fornecido pelas partículas atualizadas para calcular as saídas do processo com auxílio do modelo neural do processo.
      4.1.1. Verificar se há saídas cujos valores estão fora dos limites operacionais estabelecido.
      4.1.2. Eliminar as partículas que produziram saídas fora dos limites operacionais.
5. Avaliar novamente o enxame através do critério de desempenho (nesta avaliação a melhor partícula anterior é considerada) e armazenar a melhor partícula.
6. Repetir os passos 4 e 5 enquanto *erro* > 2%.

Um segundo estudo foi feito, mantendo-se a temperatura T_{rx}= 543,5ºC e diminuindo-se os limites (faixa de operação) da variável T_{rg1} (temperatura da fase densa do 1º estágio do regenerador). Os resultados encontrados estão

apresentados nas tabela 7.11 e 7.12. Os erros apresentados são a diferença percentual entre os valores dados pelo algoritmo e os valores dados pelo simulador.

Tabela 7.11 Resultados do PSO otimizando uma variável ($T_{rx} = 543{,}5^{o}C$)

		teste1			teste2			Teste3		
		PSO	**Real**	**Erro(%)**	**PSO**	**Real**	**Erro(%)**	**PSO**	**Real**	**Erro(%)**
variáveis controladas (saídas)	T_{rg1}	696,45	701,24	0,68	666,39	670,87	0,67	711,13	716,04	0,69
	T_{rg2}	721,13	724,80	0,51	696,21	714,92	2,62	722,31	726,60	0,59
	Sev	77,02	78,35	1,69	74,78	76,46	2,20	77,54	78,62	1,38
	T_{rx}	543,51	547,03	0,64	543,49	537,91	1,04	543,51	547,10	0,66
Variáveis manipuladas (entradas)	R_{ai}	228,80	228,80		207,30	207,30		224,06	224,06	
	c_{TCV}	0,67	0,67		0,76	0,76		0,58	0,58	
	R_{tf}	8684,20	8684,20		9484,00	9484,00		7845,20	7845,20	
	T_{fp}	232,60	232,60		242,27	242,27		243,08	243,08	

Tabela 7.12 Resultados do PSO otimizando uma variável ($T_{rx} = 543{,}5^{o}C$) e diminuindo-se a faixa devariação da variável T_{rg1} ($673 \leq T_{rg1} \leq 675$)

		teste1			teste2			Teste3		
		PSO	**Real**	**Erro(%)**	**PSO**	**Real**	**Erro(%)**	**PSO**	**Real**	**Erro(%)**
variáveis controladas (saídas)	T_{rg1}	673,80	668,62	0,77	674,07	670,70	0,50	674,49	673,99	0,07
	T_{rg2}	714,91	699,23	2,24	715,81	700,98	2,11	715,67	705,16	1,49
	Sev	76,15	76,85	0,91	74,95	76,76	2,35	74,92	76,56	2,14
	T_{rx}	543,50	536,37	1,33	543,46	539,53	0,73	543,50	540,21	0,61
Variáveis manipuladas (entradas)	R_{ai}	215,82	215,82		211,20	211,20		222,81	222,81	
	c_{TCV}	0,76	0,76		0,75	0,75		0,78	0,78	
	R_{tf}	9101,80	9101,80		9416,68	9416,68		9748,47	9748,47	
	T_{fp}	215,68	215,68		240,57	240,57		238,90	238,90	

Com o intuito de comparar os métodos de otimização, apresenta-se a seguir (tabela 7.13) os resultados obtidos com o Algoritmo Genético para o mesmo estudo realizado com o PSO.

Tabela 7.13 Resultados do AG otimizando uma variável ($T_{rx} = 543{,}5^{o}C$) e diminuindo-se a faixa de variação da variável T_{rg1} ($673 \leq T_{rg1} \leq 675$)

		teste1			teste2			Teste3		
		GA	**Real**	**Erro(%)**	**GA**	**Real**	**Erro(%)**	**GA**	**Real**	**Erro(%)**
variáveis controladas (saídas)	T_{rg1}	673,32	666,68	0,99	673,23	671,62	0,24	674,00	669,01	0,74
	T_{rg2}	714,51	696,71	2,55	715,18	702,96	1,74	715,43	699,03	2,35
	Sev	79,56	76,64	3,81	75,57	76,53	1,26	75,17	76,75	2,06
	T_{rx}	541,39	535,03	1,19	544,61	538,59	1,12	542,53	537,90	0,86
Variáveis manipuladas (entradas)	R_{ai}	207,22	207,22		225,92	225,92		207,45	207,45	
	c_{TCV}	0,73	0,73		0,79	0,79		0,74	0,74	
	R_{tf}	8910,90	8910,90		9713,80	9713,80		9165,10	9165,10	
	T_{fp}	219,12	219,12		228,36	228,36		233,95	233,95	

Neste estudo foram feitos dois testes a fim de comparar os dois métodos de otimização. O primeiro teste relaciona-se à efetividade (busca do ótimo global) dos algoritmos e o segundo relaciona-se à eficiência (custo computacional) dos algoritmos. Efetividade é definida como a habilidade do algoritmo de repetidamente encontrar a conhecida solução ótima global ou apresentar soluções suficientemente próximas quando o algoritmo é solicitado de diferentes pontos aleatórios do espaço. Em outras palavras, efetividade é definida como uma alta probabilidade de encontrar-se uma solução de alta qualidade (HASSAN *et al*., 2004). Aqui, qualidade da solução é medida pela aproximação com uma solução conhecida, dada pelo simulador do processo:

$$Q_{sol} = \left| \frac{solução - solução\ conhecida}{solução\ conhecida} \right| \% \qquad (7.8)$$

A qualidade da solução (Q_{sol}) nada mais é que o erro percentual entre a solução encontrada pelo algoritmo e a solução do modelo fenomenológico.

Embora este método não garanta que a resposta encontrada seja a ótima, permite verificar que as respostas dos algoritmos não divergiram das respostas reais do processo (Tabelas 7.12 e 7.13). Quando se confronta as respostas dos dois algoritmos (PSO e AG) percebe-se que não diferem muito entre si, o que sugere, com uma probabilidade razoável, serem estas as respostas ótimas para o problema proposto.

O segundo teste diz respeito à eficiência do algoritmo, ou seja, ao custo computacional. Para tanto, o tempo de processamento foi computado, usando-se um único processador, e levando-se em conta o mesmo critério de parada (erro em relação ao objetivo de otimização menor que 2%). O tempo de processamento do PSO ficou em torno de 6 horas e o do AG entre 20 e 23 horas.

Deve-se levar em conta que a rede neural é solicitada várias vezes nos dois algoritmos e o tempo de simulação estipulado para o modelo neural é bastante grande. Este tempo garante que a rede atinja, de fato, o estado estacionário do processo para cada conjunto de variáveis fornecido pelos algoritmos de otimização.

7.8 CONTROLE PREDITIVO BASEADO EM MODELO

Dependendo da necessidade, os modelos de processo e econômicos podem ser simples ou bastante elaborados. Para questões de controle preditivo baseado em modelo, um modelo preciso da planta é desejável. Ultimamente tem sido bastante disseminado o uso de técnicas de inteligência artificial, como as redes neurais artificiais, para modelagem de processos químicos e posterior implementação do modelo em controladores preditivos.

Considere um processo multi-variável com V_c variáveis controladas e variáveis V_m manipuladas. O objetivo de um controlador preditivo do tipo MPC é calcular um conjunto de ações de controle futuras que minimize a seguinte função objetivo:

$$J = \sum_{j=1}^{V_c} \delta_j \sum_{i=N_1}^{N_y} (\hat{y}^c_{j,k+i} - w_{j,k+i})^2 + \sum_{j=1}^{V_m} \lambda_j \sum_{i=1}^{N_u} (\Delta u_{j,k+i-1})^2 \qquad (7.9)$$

onde δ e λ são fatores de ponderação (parâmetros de ajuste). Os fatores λ penalizam as ações de controle e os fatores δ adicionam um grau de liberdade no ajuste das ações relativas a cada variável controlada. Quanto menor o desvio do estado estacionário permitido para uma dada variável, maior será o fator δ desta respectiva variável (GEORGASKI e KALRA, 1994). Δu são os incrementos nas

variáveis manipuladas, N_1 é o horizonte inicial, N_y é o horizonte de predição, N_u é o horizonte de controle, w são as trajetórias de referência (*set-points*) e $\hat{y}^c$ são as previsões do modelo corrigidas. Estas predições são obtidas recursivamente até um total de N_y predições futuras para cada variável controlada, V_c, e corrigidas da seguinte forma:

$$\hat{y}^c_{k+i} = \hat{y}_{k+i} + d_k \quad (i = 1,\ldots,N_y) \tag{7.10}$$

Na equação 7.9 , $\hat{y}_{k+i}$ são as predições feitas pelo modelo para as saídas controladas do processo no instante de amostragem futuro, $k+i$, e d_k são os termos de correção dados por:

$$d_k = y_k - \hat{y}_k \tag{7.11}$$

onde y_k são as saídas medidas do processo no instante de amostragem presente e $\hat{y}_k$ são as respectivas predições do modelo (calculadas no instante de amostragem anterior).

O algoritmo de otimização calcula os incrementos, Δu, para as variáveis manipuladas de modo a minimizar a função objetivo (7.9). As ações de controle futuras são, então, derivadas a partir dos incrementos ótimos de controle:

$$u_{k+i} = u_{k+i-1} + \Delta u_{k+i} \quad (i = 0,\ldots,N_u - 1) \tag{7.12}$$

É importante observar que somente as primeiras N_u ações de controle são otimizadas, enquanto que as demais são mantidas constantes, ou seja:

$$u_{k+i} = u_{k+N_u-1} \quad (i = N_u, \ldots, N_y - 1) \tag{7.13}$$

O problema de otimização também está sujeito às seguintes restrições:

$$y_{\min} \leq \hat{y}_{k+i} \leq y_{\max} \quad (i = 1, \ldots, N_y) \tag{7.14}$$

$$u_{\min} \leq u_{k+i-1} \leq u_{\max} \quad (i = 1, \ldots, N_u) \tag{7.15}$$

$$\left| u_{k+i-1} - u_{k+i-2} \right| \leq \Delta u_{\max} \quad (i = 1, \ldots, N_u) \tag{7.16}$$

A estratégia de horizonte móvel (*recending horizon*) é adotada (SOETERBOEK, 1992), onde só a primeira ação de controle (para cada variável manipulada), u_k, é implementada e o problema de otimização é resolvido novamente em cada instante de amostragem. O problema de otimização (com restrições) é resolvido através da técnica da Programação Quadrática Sucessiva – SQP (EDGARD e HIMMELBLAU, 1989)

Para o caso específico da unidade de FCC estudada, as restrições operacionais impostas às variáveis manipuladas, $u = \left\lfloor R_{ai} \;\; R_{tf} \;\; T_{fp} \;\; C_{TCV} \right\rfloor$, e controladas $y = \left\lfloor T_{rg1} \;\; T_{rg2} \;\; Severidade \;\; T_{rx} \right\rfloor$ são dadas por:

- Restrições nas variáveis controladas

$$\begin{aligned} &540 \leq T_{rg1}(k+i) \leq 770 && (i = 1, \ldots, N_y) \\ &540 \leq T_{rg2}(k+i) \leq 770 && (i = 1, \ldots, N_y) \\ &50 \leq Severidade\,(k+i) \leq 100 && (i = 1, \ldots, N_y) \\ &400 \leq T_{rx}(k+i) \leq 700 && (i = 1, \ldots, N_y) \end{aligned} \tag{7.17}$$

- Restrições nas variáveis manipuladas

$$\begin{aligned} &201 \le R_{ai}(k+i) \le 231 && (i=1,\ldots,N_u) \\ &0{,}45 \le C_{TCV}(k+i) \le 0{,}82 && (i=1,\ldots,N_u) \\ &50 \le R_{tf}(k+i) \le 100 && (i=1,\ldots,N_u) \\ &400 \le T_{fp}(k+i) \le 700 && (i=1,\ldots,N_u) \end{aligned} \tag{7.18}$$

- Restrições nas ações de controle

$$\begin{aligned} &\left|R_{ai}(k+i) - R_{ai}(k+i-1)\right| \le 1{,}5 && (i=1,\ldots,N_u) \\ &\left|C_{TCV}(k+i) - C_{TCV}(k+i-1)\right| \le 0{,}015 && (i=1,\ldots,N_u) \\ &\left|R_{tf}(k+i) - R_{tf}(k+i-1)\right| \le 50 && (i=1,\ldots,N_u) \\ &\left|T_{fp}(k+i) - T_{fp}(k+i-1)\right| \le 2{,}2 && (i=1,\ldots,N_u) \end{aligned} \tag{7.19}$$

Os resultados do controle da unidade FCC são apresentados a seguir utilizando um controlador MPC-MLP. A estratégia foi testada em problemas servo e regulador. Em cada uma das simulações, será imposta uma trajetória que deverá ser seguida por uma das variáveis controladas (problema servo) enquanto as demais deverão permanecer o mais próximo possível de seus estados estacionários de referência (problema regulador).

Segue-se o desempenho do controlador preditivo não linear baseado no modelo neural MLP identificado na seção 7.5.2. Os parâmetros do controlador foram sintonizados com os seguintes valores: $N_1 = 1$, $N_y = 30$, $N_u = 2$, $\lambda = \begin{bmatrix} 0{,}01 & 0{,}01 & 0{,}0001 & 0{,}001 \end{bmatrix}^T$ e $\delta = \begin{bmatrix} 10^{-5} & 10^{-5} & 10^{-3} & 10^{-1} \end{bmatrix}^T$.

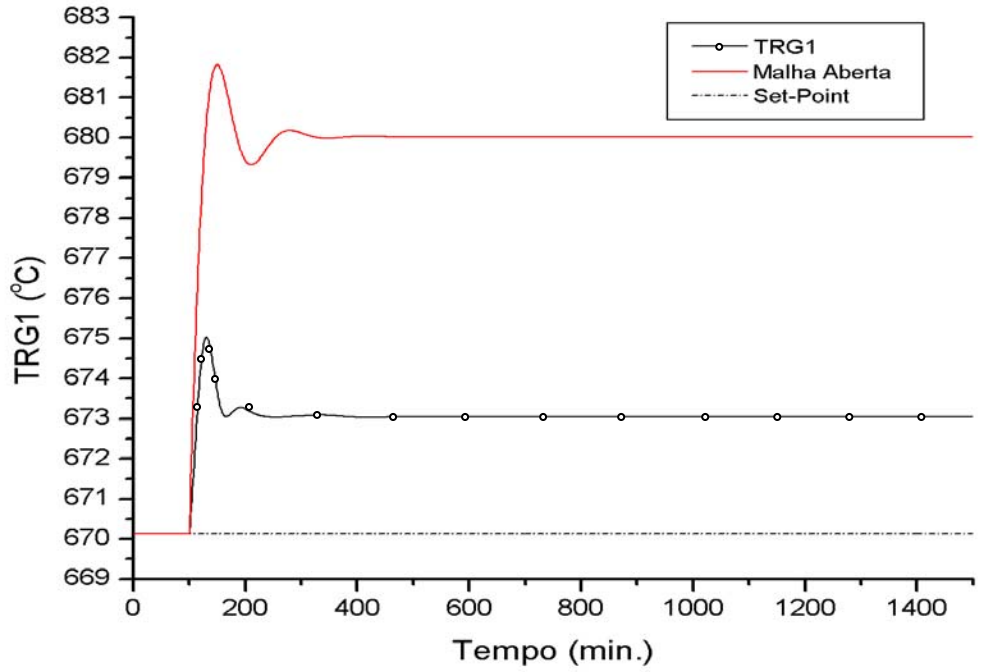

Figura 7.28 Problema regulador – variável T_{rg1}

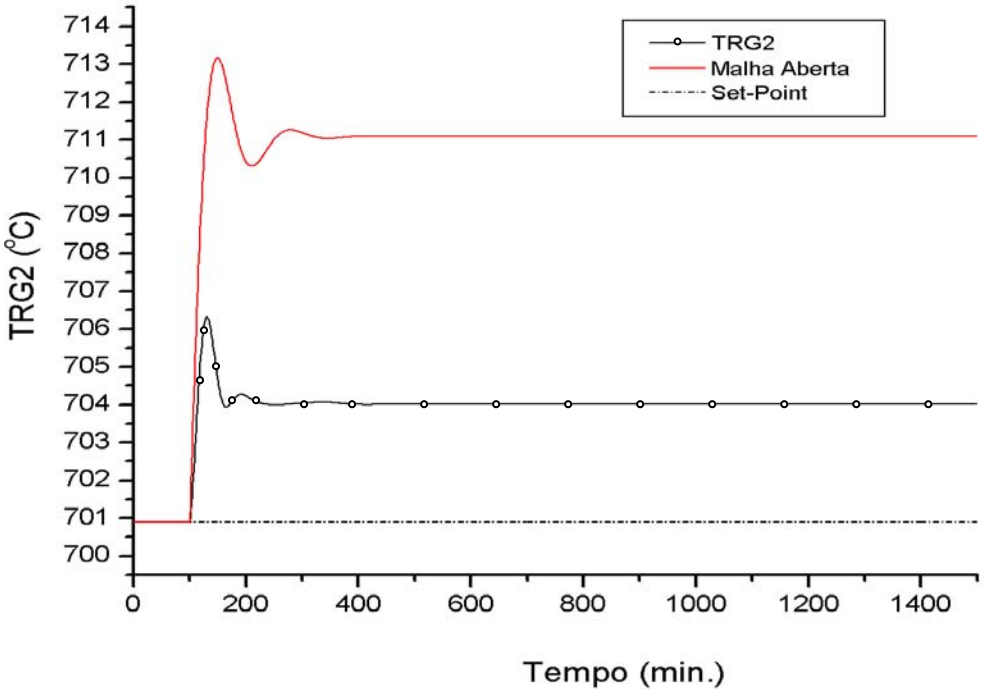

Figura 7.29 Problema regulador – variável T_{rg2}

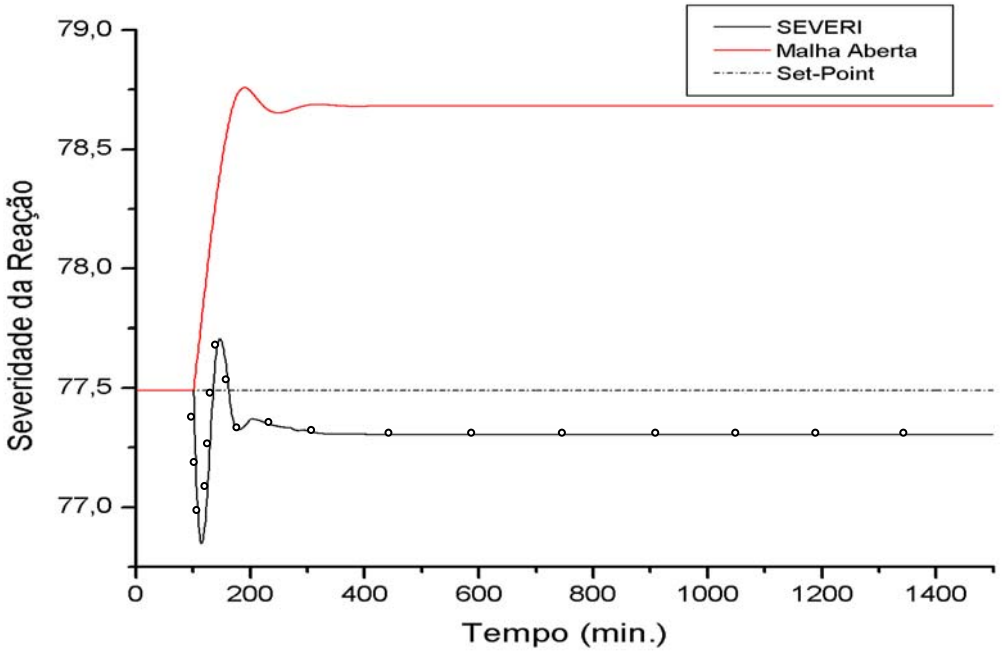

Figura 7.30 Problema regulador – variável *Severidade*

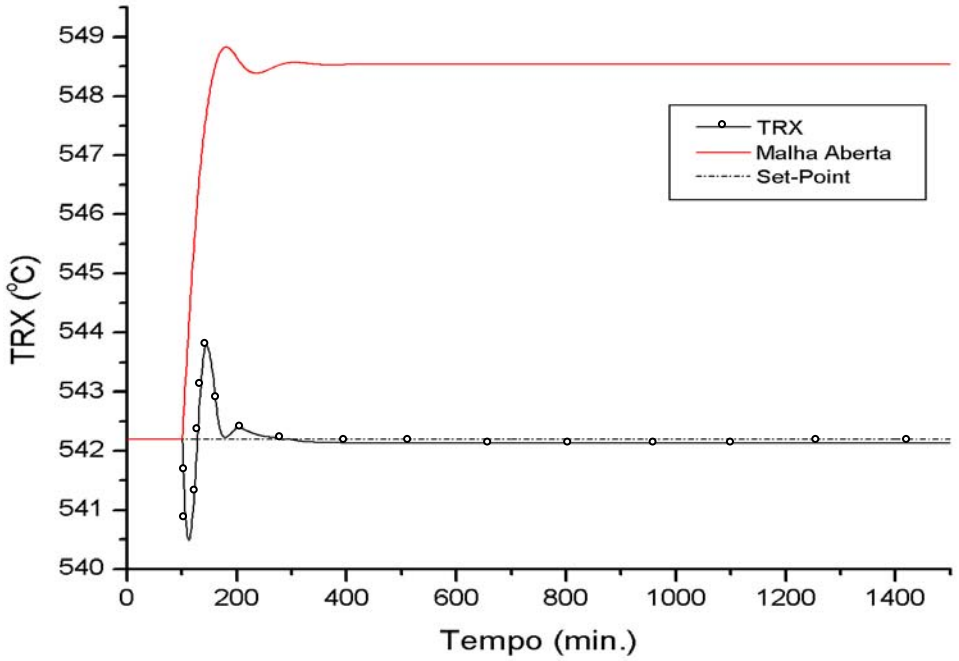

Figura 7.31 Problema regulador – variável T_{rx}

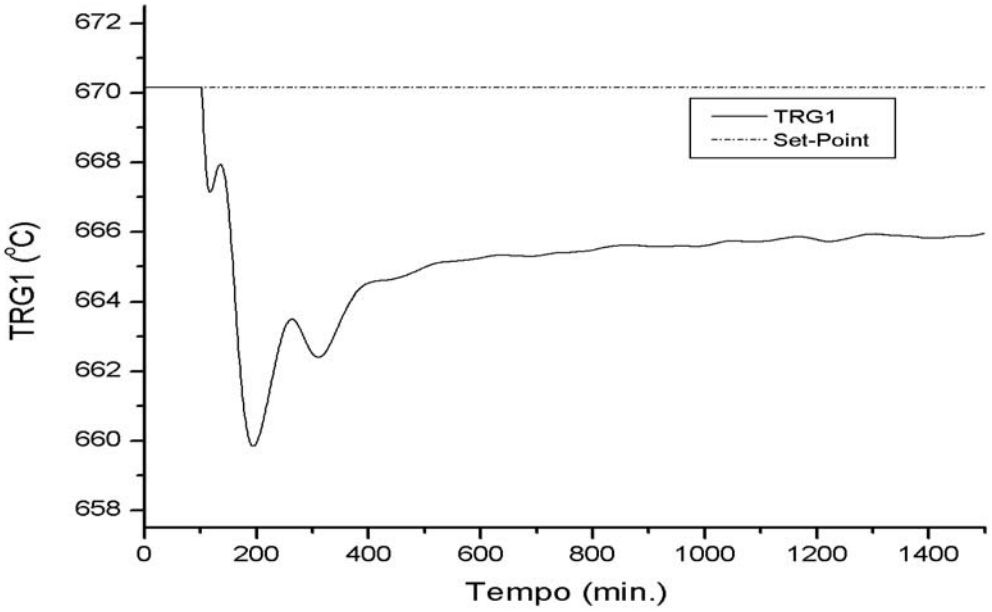

Figura 7.32 Problema servo – variável T_{rg1}

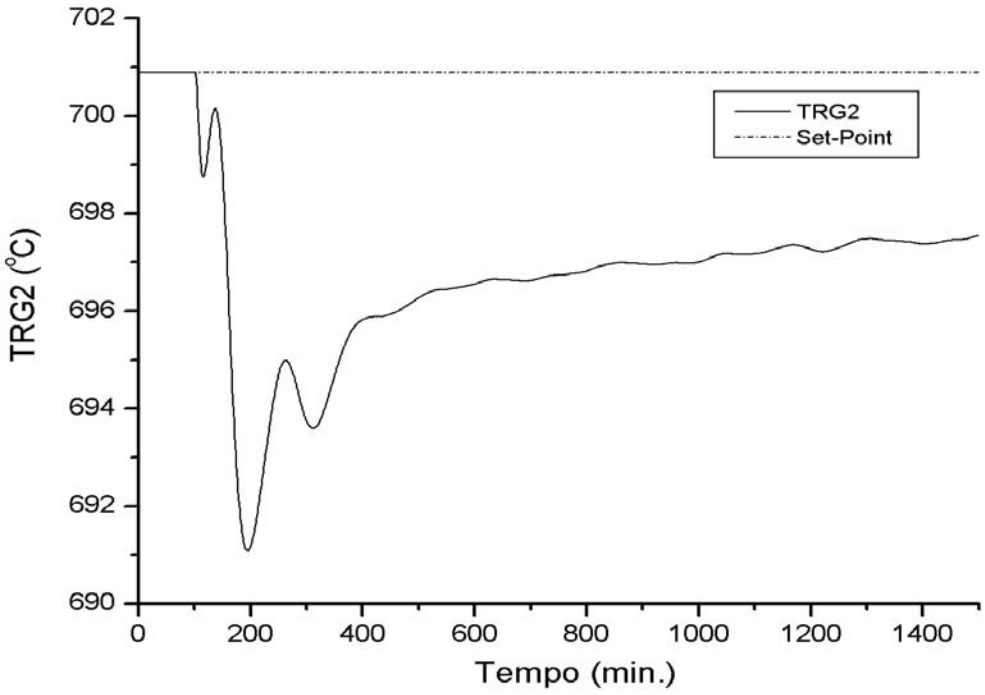

Figura 7.33 Problema servo – variável T_{rg2}

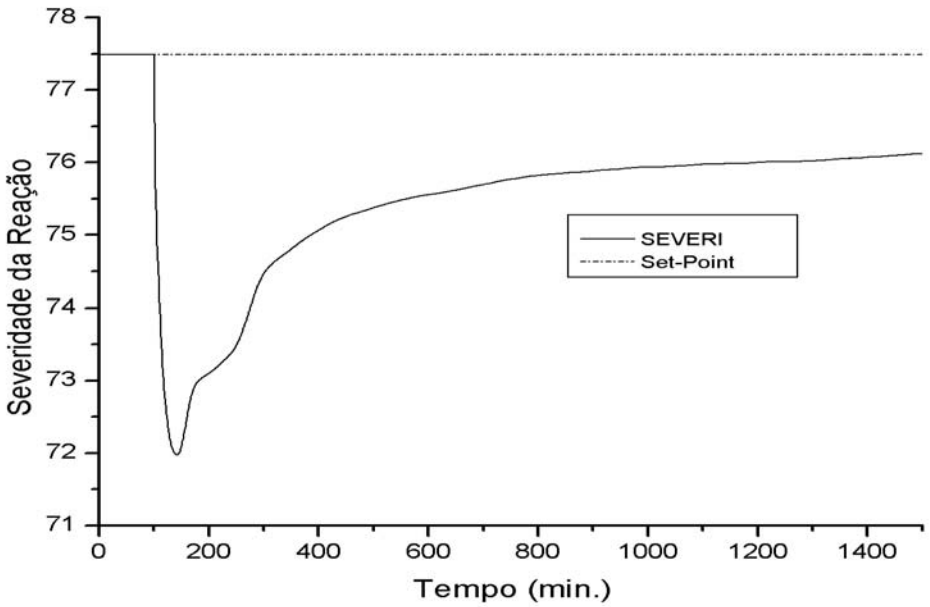

Figura 7.34 Problema servo – variável *Severidade*

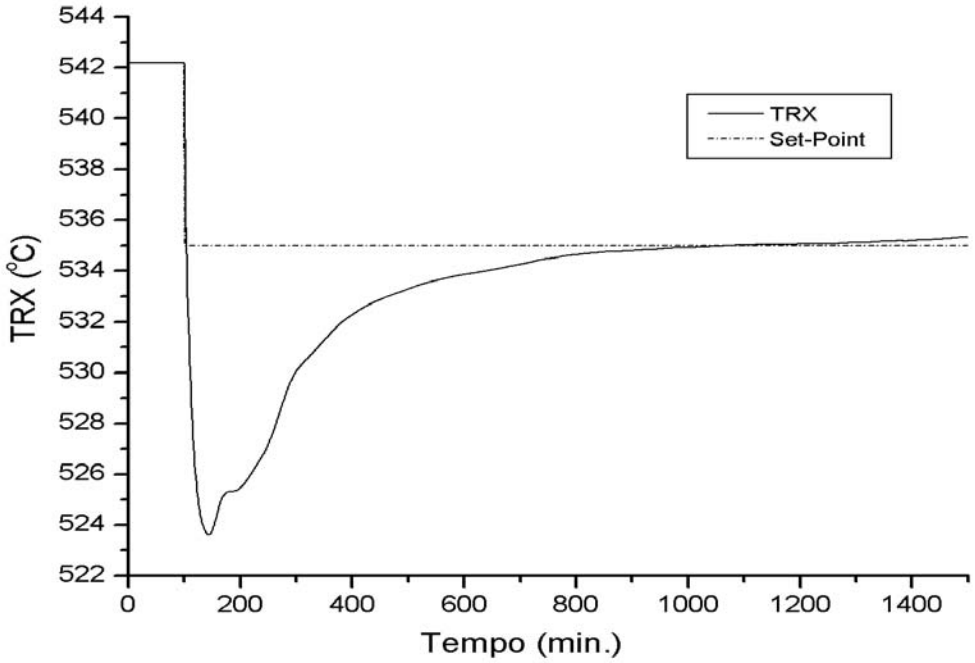

Figura 7.35 Problema servo – variável T_{rx}

7.9 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

Neste capítulo foram apresentados os resultados e as discussões geradas no decorrer do estudo.

Procedeu-se a um estudo da dinâmica do processo aplicando-se degraus nas variáveis manipuladas e acompanhando-se a resposta das variáveis controladas a estes degraus.

O processo de craqueamento catalítico foi identificado primeiramente por um modelo linear, com resultado na simulação recursiva não satisfatório para o uso que se pretendia posteriormente. Foram também testadas redes neurais MISO com resultados igualmente insatisfatórios, devido ao forte acoplamento entre as variáveis.

Finalmente foi testada uma rede MLP MIMO que apresentou ótimos resultados para todas as variáveis controladas na simulação recursiva.

Utilizou-se dois métodos heurísticos estocásticos de otimização para o processo, PSO e AG, adaptando-os a um problema proposto. Os resultados obtidos foram bastante satisfatórios.

O modelo neural obtido foi implementado no projeto de um controlador preditivo do processo. Os resultados obtidos também são apresentados aqui. Este controlador foi capaz de rejeitar perturbações impostas ao processo e seguir as trajetórias propostas.

Capítulo 8 Considerações Finais

8.1 Conclusões

Um dos objetivos da dissertação cujo resumo se apresenta aqui foi identificar o processo de craqueamento catalítico utilizando um modelo neural com a finalidade de implementá-lo em uma estratégia de otimização de um controlador preditivo do sistema. A identificação com modelo neural foi escolhida por ser relativamente simples e por terem as redes neurais grande poder de adaptação e capacidade de representação não linear, além disso, são capazes de tratar dados com ruídos, típicos de situações industriais. É provável que a vantagem mais interessante do uso de redes neurais encontrada neste trabalho seja a relativa facilidade de identificação de um novo modelo (ajuste dos parâmetros) quando ocorrem mudanças nas condições de operação causadas, por exemplo, pela mudança no tipo de carga (ocorrência bastante comum no país).

Além disso, o modelo neural foi escolhido por ser não linear e, desse modo, apresentar maior capacidade para descrever o processo de craqueamento catalítico, sabidamente não linear. Com o intuito de avaliar a qualidade do modelo neural foi identificado um modelo linear (ARX MIMO) e as respostas foram comparadas. Observou-se que o resultado do modelo ARX não foi satisfatório para o uso pretendido.

O trabalho foi desenvolvido com o objetivo de se obter um modelo de dimensões reduzidas e, para tanto, um estudo da influência dos parâmetros da rede foi conduzido.

A estratégia de identificação não linear proposta neste trabalho apresentou bons resultados, sendo o modelo identificado implementado em dois algoritmos de otimização e no projeto de um controlador preditivo do processo.

O outro objetivo foi otimizar o processo utilizando os métodos evolutivos AG e PSO. De acordo com os resultados apresentados, houve uma convergência do algoritmo genético em relação ao modelo fenomenológico e um erro relativo da otimização de 1,8% (isto porque o critério de parada estabelecido no algoritmo foi de 2%), o que permite concluir que esta é uma alternativa promissora para otimização do processo. Da mesma forma o PSO obteve ótimos resultados, com um esforço computacional significativamente menor. É importante destacar que não foram encontradas, até o momento, referências do uso do algoritmo PSO na otimização da unidade FCC.

Dentro das possibilidades operacionais da planta, isto é, considerando a dinâmica complexa do processo, a reação que se dá em 2 a 5 segundos, o caráter multivariável, não linear e com forte acoplamento das variáveis, o desempenho do controlador MPC-MLP proposto neste trabalho mostrou-se satisfatório. O controlador foi capaz de rejeitar a perturbação imposta ao processo e também foi eficiente na tarefa de seguir a trajetória estabelecida para uma das variáveis (mantendo-se todas as variáveis dentro da faixa operacional).

8.2 PERSPECTIVAS FUTURAS

Segue-se algumas sugestões para trabalhos futuros:

- Identificação do processo através de outros modelos não lineares menos sujeitos à realimentação do erro (séries de Volterra, por exemplo).

- Desenvolvimento de outros modelos heurísticos de otimização computacionalmente mais rápidos bem como métodos híbridos (com métodos determinísticos) para refinar a busca.
- Utilizar algoritmos neurais que determinem automaticamente a arquitetura da rede (métodos construtivos e métodos de poda).
- Identificar a rede neural com dados reais de operação da unidade FCC.
- Desenvolvimento de métodos para determinação dos parâmetros de ajuste do controlador.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUIRRE, Luis Antônio. **Introdução à identificação de sistemas:** Técnicas lineares e não lineares aplicadas a sistemas reais. Belo Horizonte: Editora UFMG, 2000.

ALARADI, A.A.; ROHANI, S. Identification and Control of a riser-type FCC unit using neural networks. **Computer and Chemical Engineering**, v. 26, p.401-421, 2002.

ALDRICH, C.; SLATER, M. J. Simulation of Liquid-Liquid Extraction Data with Artificial Neural Networks. In: MUJTABA, I. M.; HUSSAIN, M. A. (Ed.). **Application of Neural Networks and other Learning.** London: Imperial College Press, 2001. p. 03-22.

BAUCHPIESS, Adolfo. **Introdução aos Sistemas Inteligentes**: Aplicações em Engenharia de Redes Neurais Artificiais, Lógica Fuzzy e Sistemas Neuro-Fuzzy. Apostila. Universidade de Brasília, Departamento de Engenharia Elétrica. Brasília: UnB, 2004.

BOLLAS, G.M.; PAPADOKONSTADAKIS, S.; MICHALOPOULOS, J.; ARAMPATZIS, A.A.; VASALOS,I.A.; LYGEROS,A. Using Hybrid Neural Networks in Scaling Up an FCC model from Pilot to an Industrial Unit. **Chemical Engineering and Process**, v. 42, p.697-713, 2003.

COCKSHOTT, A.R.; HARTMAN, B.E. **Improving the fermentation medium for Echinocandin B production part II: Particle Swarm Optimization**. Process Biochemistry v.36 p. 661-669, 2001.

CRISTEA, M.V.; AGACHI, S. P.; MARINOIV, V. Simulation and model predictive control of a fluid catalytic cracking unit. **Chemical Engineering and Process**, v. 42, p.67-91, 2003.

EDGAR, T.F.; HIMMELBLAU,D.M. **Optimization of Chemical Processes.** Singapura: McGraw-Hill Book Co, 1989.

FROMENT, G.F.; BISCHOFF, K.B. Chemical Reactor Analysis and Design. 2nd ed. New York: John Wiley & Sons, 1990.

GEORGASKI, C; KALRA, L. Effect of Process Nonlinearity on the performance of Linear Model Predictive Controllers for the Environmentally Safe Operation of a Fluid Catalytic Cracking Unit. **Industrial & Engineering Chemistry Research**, v.33, p. 3063-3069, 1994.

GONZAGA, João Carlos B. **Integração de processos em tempo real para monitoramento e controle:** aplicação para planta de PET. 2003.112 f. Dissertação (Mestrado) - Departamento de Engenharia Química, Universidade Estadual de Campinas, Campinas, 2003.

GOUVÊA, Míriam Tvrzská. **Uso de um Algoritmo SQP na Otimização de Processos Químicos Contínuos em Tempo Real**. 1997. 278 f. Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, São Paulo, 1997.

HAN, I. S.; RIGGS, J. B.; CHUNG, C.B.. Multivariable control of a fluidized catalytic cracking process under full and partial combustion modes. **Journal of Chemical Engineering of Japan**, v. 35, p.830-839, 2002.

HASSAN, R.; COHANIM, B. WECK, O. VENTER, G. **A Comparison of Particle Swarm Optimization and the Genetic Algorithm**. American Institute of Aeronautics and Astronautics, 2004.

HAYKIN, Simon. **Redes Neurais:** Princípios e prática. 2. ed. Porto Alegre: Bookman, 2001.

HENRIQUE, Humberto Molinar. **Uma contribuição ao estudo de redes neuronais aplicadas ao controle de processos.** 1999. 254 f. Tese (Doutorado) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1999.

HENRIQUE, H. M.; LIMA, E. L.; SEBORG, D.E. Model Structure determination in neural networks models. **Chemical Engineering Science**, v. 55, p.5457-5469, 2000.

KAHANER, D.; MOLER, C.; NASH, S. **Numerical Methods and Software**. Prentice Hall: 1988.

KASAT, R.B.; KUNZRU,D.; SARAF, D.N.; GUPTA, S.K. Multiobjective Optiization of Industrial FCC Units Using Elitist Nondominated Sorting Genetic Algotithm. **Ind. Eng. Chem. Res.**, v. 41, p.4765-4776, 2002.

KENNEDY, J. & EBERHART, R. Particle **Swarm Optimization**. Proceedings IEEE International Conference on Neural Networks. p.1942-1948. Perth:1995.

McCALL, John. Genetic algorithms for modeling and optimization. **Journal of Computational and Applied Mathematics**, v. 184, p. 205 – 222, 2005.

MELEIRO, Luiz Augusto da Cruz. **Projeto e Aplicação de Controladores baseados em Modelos Lineares, Neurais e Nebulosos.** 2002. 330 f. Tese (Doutorado) - Faculdade de Engenharia Química, Universidade Estadual de Campinas, Campinas, 2002.

MICHALEWICZ, Z. E FOGEL, D.B. **How to solve it**: Modern Heuristics. Spring-Verlag, 2000.

MORO, Lincoln Fernando Lautenschlager. **Desenvolvimento de um controlador preditivo multivariável para um conversor industrial de craqueamento catalítico.** 1992. 77 f. Dissertação (Mestrado) - Universidade de São Paulo, São Paulo, 1992.

MORO, L. F. L.; ODLOAK, D. Constrained multivariable control of a fluid catalytic cracking. **Journal of Process Control**, v. 5, n. 1, p.29-39, dez. 1995.

NAGY, Z.; AGACHI, S.; BODIZS, L.. Adaptive Neural Network Model Based Nonlinear Predictive Control of a Fluid Catalytic Cracking Unit. In: EUROPEAN SYMPOSIUM ON COMPUTER AIDED PROCESS ENGINEERING, 2000, Florence. **Computer-Aided Chemical Engineering .** Florence: 10, 2000. p. 235 -240.

NØRGAARD, M. RAVN,O; POULSEN,N.K. e HANSEN,L.K. **Neural Networks for Modelling an Control of Dynamic Systems.** Springer-Verlag, 2000.

POMEROY, Paul. **An Introduction to Particle Swarm Optimization**. http://www.adaptiveview.com/articles/ipsoprnt.html, 2003.

QIN, S. J.; BADGWELL, T. A. **An Overview of Industrial Model Predictive Control Technology.** Disponível em: <www.che.utexas.edu/~qin/cpcv/cpcv14.html>. Acesso em: 29 mar. 1996.

RAMIREZ, Fred. **Process and Control Identification.** San Diego: Academic Press, 1994.

REYNOLDS, C.W. **Flocks, herds and schools** *:* a distributed behavioral model. Computer Graphics, 21 (4): 25-34, 1987.

SANTOS, Marlova Gonçalves. **Modelo dinâmico para o conversor de uma unidade de FCC UOP STACKED.** 2000. 138 f. Dissertação (Mestrado) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 1999.
SANTOS, V. M. L.; CARVALHO, F. R.; SOUZA JUNIOR, M. B. de. Predictive Control based on neural networks: an application to a fluid catalytic cracking industrial unit. **Brazilian Journal of Chemical Engineering**, , v. 17, p.401-421, 2000.

SEQUEIRA, Sebastián Eloy. **Real Time Evolution (RTE) for on-line optimization of continuous and semi-continuous chemical process.** 2003. Tese (Doutorado) – Universitat Politècnica de Catalunya, Barcelona, 2003.

SILVER, E.A. An overview of heuristic solution methods. **Journal of the Operational Research Society**, v. 55, n.9, p. 936 – 956, 2004.

SJÖBERG, Jonas. **Non-Linear System Identification with Neural Networks.** 1995. 223 f. Tese (Doutorado) - Departamento de Engenharia Elétrica, Universidade de Linköping, Linköping, Suécia, 1995.

SÖDERSTRÖM, T.; STOICA, P.. **System Identification.** London: Prentice Hall, 1989.

TYAGUNOV, Andrey A.. **High-Performance Model Predictive Control for Process Industry.** 2004. 179 f. Tese (Ph. D.) - Technische Univesiteit Eindhoven, Eindhoven, 2004.

VENKATASUBRAMANIAN, V.; CHAN, K.. A Neural Networks Methodology for Fault Diagnosis. **AIChe Journal**, v. 35, p.1993-2002, dez. 1989.

VIEIRA, W. G. et al. Identification and Predictive Control of a FCC Unit Using a MIMO neural model. **Chemical Engineering and Process**, v. 44, p.855-868, 2005.

VON ZUBEN, Fernando J. **Computação evolutiva***: uma abordagem pragmática*. Technical Report. Campinas: Unicamp, 2004.

YANG, S. H. et al. Soft sensor based predictive control of industrial fluid catalytic cracking processes. **IChE Journal**, v. 76, p.499-508, 1998.

ZANIN, Atônio Carlos. **Implementação Industrial de um Otimizador em Tempo Real.** 2001. 183 f. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2001.