Campus de São Carlos

"Modelagem de um processo fermentativo por rede perceptron multicamadas com atraso de tempo"

Autor:

Engº. Luis Fernando Manesco

Orientadora:

Profa. Dra. Vilma Alves de Oliveira

UNIVERSIDADE DE SÃO PAULO

**ESCOLA DE ENGENHARIA
DE SÃO CARLOS**

Modelagem de um Processo Fermentativo por Rede Perceptron Multicamadas com Atraso de Tempo

ENG. LUIS FERNANDO MANESCO

Dissertação apresentada à Escola de Engenharia de São Carlos, da Universidade de São Paulo, como parte dos requisitos para obtenção do Título de Mestre em Engenharia Eletrônica.

ORIENTADORA: Profa. Dra. Vilma Alves de Oliveira

DEDALUS - Acervo - EESC

31100035785

São Carlos
1996

Class.	1055 CESC
Cutt.	3261
Tombo	1049/97

Área: Engenharia Elétrica

α 746948

**Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca - EESC-USP**

M274m Manesco, Luis Fernando
Modelagem de um processo fermentativo por rede
perceptron muticamadas com atraso de tempo / Luis
Fernando Manesco. -- São Carlos, 1996.

Dissertação (Mestrado). -- Escola de Engenharia
de São Carlos-Universidade de São Paulo, 1996.

Área: Engenharia Elétrica

Orientador: Prof^a. Dr^a. Vilma Alves de Oliveira

1. Redes neurais artificiais. I. Título

FOLHA DE APROVAÇÃO

Dissertação defendida e aprovada em 09-8-1996
pela Comissão Julgadora:

Prof. Doutora VILMA ALVES DE OLIVEIRA - Orientadora
(Escola de Engenharia de São Carlos - Universidade de São Paulo)

Prof. Doutor ALUIZIO FAUSTO RIBEIRO ARAUJO
(Escola de Engenharia de São Carlos - Universidade de São Paulo)

Prof. Doutor ANTONIO MARIA F.L.J. BONOMI
(Instituto de Pesquisas Tecnológicas do Estado de S.Paulo - I.P.T.)

Coordenador da área "Engenharia Elétrica"
Prof. Dr. GERALDO ROBERTO M. DA COSTA

Presidente da Comissão de Pós-Graduação
Prof. Dr. JOSÉ CARLOS A. CINTRA

A meu pai, Renato, minha mãe, Lourdes,
minha irmã, Cintia Maria e a Jeferson e
Marilda Totti, aqueles que sempre me
apoiaram em todas as ocasiões.

AGRADECIMENTOS

Aos professores e amigos Vilma Alves de Oliveira, Edson Gesualdo, Aluizio F. R. Ribeiro e Jerson Barbosa de Vargas, pela orientação, pelo encaminhamento e, principalmente, por serem as maravilhosas pessoas que são.

Aos pesquisadores do IPT, em especial ao Prof. Bonomi, pelas explanações, sugestões, discussões e principalmente pela paciência.

A Carlos Alberto de Francisco e Carlos Alexandre Matioli Globo pela paciência e apoio.

A Adriano, Maria Inês e Antonio Blanco, pelo apoio e incentivo.

A Roseli Ribeiro, pela paciência, apoio e amizade.

À Coordenadoria de Aperfeiçoamento de Pessoal de Nível Superior - CAPES, pela bolsa de estudo concedida.

Ao pessoal do Laboratório de Controle e Eletrônica de Potência - LACEP, pelo companheirismo.

A todos os colegas, professores e funcionários do Departamento de Engenharia Elétrica da EESC/USP pela amizade.

Sumário

1. INTRODUÇÃO	1
2. MODELAGEM DE SISTEMAS DINÂMICOS	3
2.1. Modelos Matemáticos	3
2.1.1. Modelo Espaço de Estado.....	3
2.1.2. Modelo Entrada-Saída.....	4
2.2. Observabilidade e Controlabilidade	5
2.3. Relação entre Modelos Entrada-Saída e Espaço de Estado	5
2.4. Estimação de Estados	6
2.5. Identificação.....	6
2.6. Modelos de Processos Bioquímicos	6
3. DESCRIÇÃO DO PROCESSO FERMENTATIVO	9
3.1. Descrição do Problema.....	9
3.2. Variáveis do Processo	9
3.3. Diagrama do Processo.....	10
3.4. Modelo do Processo	11
4. REDES NEURAIS	15
4.1. Modelos de Neurônios	15
4.2. Padrão de Conectividade	17
4.3. Arquitetura	17
4.4. Perceptrons Multicamadas.....	19
4.5. Métodos de Treinamento	20

4.5.1. Algoritmo de Retro-Propagação Padrão (Backpropagation)	21
4.5.2. Algoritmo de Retro-Propagação com <i>momentum</i> e taxa de aprendizado adaptativa	23
4.5.3. Algoritmo de Levenberg-Marquadt.....	23
4.6. Redes Neurais e Identificação.....	24
 5. REDES NEURAIIS NA MODELAGEM DO PROCESSO FERMENTATIVO	27
5.1. Origem dos Ensaios	27
5.2. Intervalo de Tempo	28
5.3. Variável Não Disponível.....	29
5.4. Escolha dos Conjuntos de Dados	30
5.5. Normalização dos Dados.....	30
5.6. Modelo Discreto do Processo	31
5.7. Modelagem por Rede Perceptron	31
5.8. Treinamento	34
5.9. Redes Utilizadas	34
5.10. Figura de Mérito	36
5.11. Análise de Desempenho	36
 6. RESULTADOS.....	37
6.1. Treinamento	37
6.2. Predição	39
6.3. Análise do Atraso de Tempo	52
 7. CONCLUSÕES.....	55
7.1. Treinamento	55
7.2. Predição	55

LISTA DE FIGURAS

Figura 3.1 - Diagrama Esquemático do Processo.....	11
Figura 4.1 - Diagrama Esquemático de uma UBP	16
Figura 4.2 - Diagrama Esquemático de uma Rede Neural Artificial Recursiva.....	18
Figura 4.3 - Diagrama Esquemático de uma Rede de Elman.....	19
Figura 4.4 - Diagrama Esquemático de uma Rede Perceptron Multicamadas	20
Figura 4.5 - Diagrama de Blocos de um Algoritmo de Treinamento Supervisionado.	21
Figura 4.6 - Esquema de Identificação de Estados utilizando RNA.....	25
Figura 4.7 - Esquema de Identificação Entrada-Saída utilizando RNA.....	25
Figura 5.1 - Fluxo de Dados.....	29
Figura 5.2 - Diagrama do Treinamento de uma Rede Perceptron com Atraso de Tempo.	32
Figura 5.3 - RNA com $N=1$ e 8 unidades na camada oculta.....	35
Figura 6.1 - Evolução do Erro de Treinamento para BP padrão.....	38
Figura 6.2 - Evolução do Erro de Treinamento para BP c/ momentum.....	38
Figura 6.3 - Evolução do Erro de Treinamento para Levenberg-Marquadt.....	39
Figura 6.4 - Gráfico de Predição de X para $N=1$ 12 unidades logsig.....	41
Figura 6.5 - Gráfico de Predição de P para $N=1$ 12 unidades logsig.....	41
Figura 6.6 - Gráfico de Predição de S_1 para $N=1$ 12 unidades logsig.....	42
Figura 6.7 - Gráfico de Predição de S_2 para $N=1$ 12 unidades logsig.....	42
Figura 6.8 - Gráfico de Predição de X para $N=2$ 12 unidades tansig.....	43
Figura 6.9 - Gráfico de Predição de P para $N=2$ 12 unidades logsig.....	44
Figura 6.10 - Gráfico de Predição de S_1 para $N=2$ 12 unidades logsig.....	44
Figura 6.11 - Gráfico de Predição de S_2 para $N=2$ 12 unidades logsig.....	45
Figura 6.12 - Comparação de Predição para X ensaio 12.....	46
Figura 6.13 - Gráfico de Dispersão de X para $N=1$ 12 unidades tansig ensaio 12.....	47
Figura 6.14 - Gráfico de Dispersão de X para $N=2$ 12 unidades tansig ensaio 12.....	47
Figura 6.15 - Comparação de Predição para P ensaio 12.....	48
Figura 6.16 - Gráfico de Dispersão de P para $N=1$ 12 unidades tansig ensaio 12.....	48
Figura 6.17 - Gráfico de Dispersão de P para $N=2$ 12 unidades tansig ensaio 12.....	49
Figura 6.18 - Comparação de Predição para S_1 ensaio 12.....	49
Figura 6.19 - Gráfico de Dispersão de S_1 para $N=1$ 12 unidades tansig ensaio 12.....	50
Figura 6.20 - Gráfico de Dispersão de S_1 para $N=2$ 12 unidades tansig ensaio 12.....	50
Figura 6.21 - Comparação de Predição para S_2 ensaio 12.....	51
Figura 6.22 - Gráfico de Dispersão de S_2 para $N=1$ 12 unidades tansig ensaio 12.....	51
Figura 6.23 - Gráfico de Dispersão de S_2 para $N=2$ 12 unidades tansig ensaio 12.....	52

LISTA DE TABELAS

Tabela 2.1 - Classificação dos Modelos Bioquímicos	7
Tabela 3.1 - Características das Variáveis Envolvidas	10
Tabela 3.2 - Parâmetros Associados ao Modelo Fermentativo	13
Tabela 5.1 - Condições iniciais dos ensaios disponíveis	27
Tabela 5.2 - Ensaios presentes em cada conjunto	30
Tabela 5.3 - Arquiteturas de RNAs Estudadas	35
Tabela 6.1 - Comparação dos Algoritmos de Treinamento	37
Tabela 6.2 - Figuras de Mérito.	40
Tabela 6.3 - Comparação de EMQ para Ensaio 12	46
Tabela 6.4 - Matriz de Pesos da Camada Oculta para N=1 12 unidades logsig.....	53
Tabela 6.5 - Matriz de Pesos da Camada Oculta para N=2 12 unidades tansig	53

LISTA DE ABREVIATURAS E SIGLAS

ARMAX	Auto Regressivo Média Móvel Estendido (modelo)
CTE	Conjunto de Padrões de Testes
CTR	Conjunto de Padrões de Treinamento
HPLC	Cromatógrafo Líquido de Alta Performance
IPT	Instituto de Pesquisas Tecnológicas do Estado de São Paulo S.A.
MLP	Perceptrons Multicamadas (rede neural)
NARMAX	Não Linear Auto Regressivo Média Móvel Estendido (modelo)
NARX	Não Linear Auto Regressivo Estendido (modelo)
RNA	Rede Neural Artificial
TDMLP	Perceptron Multicamadas com Atraso de Tempo
UBP	Unidade Básica de Processamento

LISTA DE SÍMBOLOS

Parâmetros do modelo do processo fermentativo

A	parâmetro de produção de sorbose associado ao crescimento
B	parâmetro de produção de sorbose não associado ao crescimento
B_m	parâmetro de produção de sorbose máximo não associado ao crescimento
K_{bO}	parâmetro de saturação por oxigênio da formação de produto
K_{bS1}	parâmetro de saturação por sorbitol da formação de produto
K_{La}	coeficiente volumétrico de transferência de oxigênio
K_O	parâmetro de saturação por oxigênio do crescimento
K_P	parâmetro de inibição por sorbose do crescimento
K_{S1}	parâmetro de saturação por sorbitol do crescimento
K_{S2}	parâmetro de saturação por extrato de levedura do crescimento
N_a	taxa de transporte de oxigênio de gás para fase líquida
O	concentração de oxigênio dissolvido
O_g	concentração de oxigênio dissolvido no equilíbrio
P	concentração de sorbose
pH	índice de acidez do meio
r_{OP}	taxa de consumo de oxigênio para produzir sorbose
r_{OX}	taxa de consumo de oxigênio para produzir células
r_P	taxa de produção de sorbose
r_{S1P}	taxa de consumo de sorbitol para produzir sorbose
r_{S1X}	taxa de consumo de sorbitol para produzir células
r_{S2X}	taxa de consumo de extrato de levedura para produzir células
S_1	concentração de sorbitol
S_2	concentração de extrato de levedura
W	parâmetro de inibição por sorbitol do crescimento
X	concentração de células
Y_{PO}	fator estequiométrico para conversão de oxigênio em sorbose
Y_{PS1}	fator estequiométrico para conversão de sorbitol em sorbose
Y_{XO}	fator de produção para conversão de oxigênio em células
Y_{XS1}	fator de produção para conversão de sorbitol em células
Y_{XS2}	fator de produção para conversão de extrato de levedura em células
$\hat{\mu}$	taxa de crescimento específico máxima sem inibição
μ_P	taxa de produção específica

μ_X taxa de crescimento específico

Parâmetros da rede neural

B_{i1}	vetor de polarizações da camada oculta
B_{i2}	vetor de polarizações da camada de saída
b_n	polarização de uma unidade de processamento
E_p	função erro quadrático
ϵ	fator de folga de normalização
$f_j()$	função de ativação da j-ésima camada de processamento
i_j	j-ésima entrada de uma unidade de processamento
L	número de padrões contidos em um conjunto
o_n	saída de uma unidade de processamento
$Pe()$	vetor de entradas da rede neural
$T()$	vetor alvo de saídas desejadas da rede neural
$\hat{T}()$	vetor de saídas da rede neural
t_{jm}	saída desejada para a j-ésima unidade da camada de saída para o padrão m
W_f	matriz de pesos e polarizações da RNA
W_{i1}	matriz de pesos da camada oculta
W_{i2}	matriz de pesos da camada de saída
w_j	peso associado à j-ésima entrada de uma unidade de processamento
y_{jm}	saída real da j-ésima unidade da camada de saída para o padrão m

Parâmetros associados ao modelamento

A	matriz de parâmetros de um modelo espaço de estado
A_j	vetor de coeficientes do modelo ARMAX associado a y_p
B	matriz de parâmetros de um modelo espaço de estado
B_j	vetor de coeficientes do modelo ARMAX associado a $u(k-j)$
C	matriz de parâmetros de um modelo espaço de estado
D	matriz de parâmetros de um modelo espaço de estado
EMQ	Erro médio quadrático
$f()$	função arbitrária
$g()$	função arbitrária
k	tempo discreto (amostra)
lr	taxa de aprendizado

M	janela de saídas
N	janela de entradas
$\mathbf{n}_1()$	vetor de ruídos
$\mathbf{S}()$	Vetor de variáveis de estado
$\hat{\mathbf{S}}()$	Vetor de variáveis de estado estimadas
t	tempo
$\mathbf{U}()$	vetor de variáveis de entrada
$\mathbf{u}()$	vetor de entradas do processo
v	variável arbitrária
v_n	variável arbitrária normalizada
v_{in}	i -ésima variável normalizada
$\hat{v}_{in}$	estimativa normalizada da i -ésima variável
$\mathbf{x}()$	vetor de variáveis de estado do processo
$\mathbf{y}_p()$	vetor de saídas do processo

RESUMO

A utilização de Redes Neurais Artificiais para fins de identificação e controle de sistemas dinâmicos têm recebido atenção especial de muitos pesquisadores, principalmente no que se refere a sistemas não lineares. Neste trabalho é apresentado um estudo sobre a utilização de um tipo em particular de Rede Neural Artificial, uma Perceptron Multicamadas com Atraso de Tempo, na estimação de estados da etapa fermentativa do processo de Reichstein para produção de vitamina C. A aplicação de Redes Neurais Artificiais a este processo pode ser justificada pela existência de problemas associados à esta etapa, como variáveis de estado não mensuráveis e com incertezas de medida e não linearidade do processo fermentativo, além da dificuldade em se obter um modelo convencional que contemple todas as fases do processo. É estudado também a eficácia do algoritmo de Levenberg-Marquadt, na aceleração do treinamento da Rede Neural Artificial, além de uma comparação do desempenho de estimação de estados das Redes Neurais Artificiais estudadas com o filtro estendido de Kalman, baseado em um modelo não estruturado do processo fermentativo. A análise do desempenho das Redes Neurais Artificiais estudadas é avaliada em termos de uma figura de mérito baseada no erro médio quadrático sendo feitas considerações quanto ao tipo da função de ativação e o número de unidades da camada oculta. Os dados utilizados para treinamento e avaliação da Redes Neurais Artificiais foram obtidos de um conjunto de ensaios interpolados para o intervalo de amostragem desejado.

Palavras-chave: Redes Neurais Artificiais, Identificação de Sistemas Não Lineares, Perceptrons Multicamadas, algoritmo Levenberg-Marquadt, processo de Reichstein.

ABSTRACT

Identification and Control of dynamic systems using Artificial Neural Networks has been widely investigated by many researchers in the last few years, with special attention to the application of these in nonlinear systems. In this work, a study on the utilization of a particular type of Artificial Neural Networks, a Time Delay Multi Layer Perceptron, in the state estimation of the fermentative phase of the Reichstein process of the C vitamin production. The use of Artificial Neural Networks can be justified by the presence of problems, such as uncertain and unmeasurable state variables and process non-linearity, and by the fact that a conventional model that works on all phases of the fermentative processes is very difficult to obtain. The efficiency of the Levenberg Marquadt algorithm on the acceleration of the training process is also studied. Also, a comparison is performed between the studied Artificial Neural Networks and an extended Kalman filter based on a non-structured model for this fermentative process. The analysis of the Artificial Neural Networks is carried out using the mean square errors taking into consideration the activation function and the number of units presents in the hidden layer. A set of batch experimental runs, interpolated to the desired time interval, is used for training and validating the Artificial Neural Networks.

Keywords: Artificial Neural Networks, Non-linear Dynamic System Identification. Multi Layer Perceptrons, Levenberg-Marquadt training algorithm, Reichstein process.

1. INTRODUÇÃO

O Ácido Ascórbico (ou vitamina C) tem uma larga gama de aplicações, sendo a indústria farmacêutica e a indústria alimentícia (onde é utilizado como antioxidante para conservação) as principais consumidoras. Inicialmente, o ácido ascórbico era retirado de frutas cítricas, num processo de recuperação. Hoje em dia, a produção de vitamina C utiliza principalmente a síntese química, sendo que o processo mais comum (Reichstein) parte da glicose e envolve várias etapas, uma das quais fermentativa. Como todo processo industrial, também neste é necessário que se disponha de informações sobre todo o processo e também de meios de se controlá-lo, para que se possa ter controle sobre a produção e a qualidade. Isto é um requisito básico para a indústria em geral, para que se possa obter o melhor rendimento de um processo. No Brasil, a vitamina C não é fabricada, sendo que todas as necessidades da indústria são supridas via importação, mas existem no país grandes fontes da matéria prima do processo de Reichstein, a glicose. A etapa fermentativa tem uma forte característica não linear, o que dificulta a utilização dos métodos clássicos de identificação e modelagem, tornando-se difícil a obtenção de um modelo que possa fornecer todas as informações necessárias ao controle do processo.

Vários trabalhos têm sido publicados tendo como tema a utilização de Redes Neurais Artificiais (RNAs) em modelagem de processos não lineares, particularmente de processos fermentativos. Estes trabalhos têm demonstrado que as RNAs têm características interessantes para a modelagem destes processos, como tolerância a ruídos, capacidade de generalização e capacidade de aprendizado, no que diz respeito à dinâmica do processo ao qual ela está associada. Além disso, se retirarmos a tarefa de treinamento, que é realizada apenas uma vez para cada RNA, a implementação de uma RNA como um programa de computador é bem simples, já que a maioria das operações que ela realiza é de multiplicação e adição. A tarefa de treinamento é a mais demorada de todo o processo de utilização e tem sido alvo de vários trabalhos que visam melhorar o desempenho dos algoritmos de treinamento de RNAs.

Este trabalho tem como objetivo contribuir para a viabilização da produção de vitamina C (Ácido L-Ascórbico) no Brasil, fornecendo um meio de modelar e, por extensão, controlar a etapa fermentativa do processo de Reichstein e faz parte de um projeto desenvolvido no Agrupamento de Biotecnologia do Instituto de Pesquisas Tecnológicas do Estado de São Paulo S.A. (IPT). Visa estudar a utilização de uma rede neural artificial para a modelagem do processo fermentativo de transformação de D-Sorbitol em L-Sorbose por

meio do microorganismo *Gluconobacter Oxydans* processo este que tem vários problemas associados tais como: variáveis de estado não mensuráveis e tendo incertezas associadas ao processo de medida, tempo de amostragem demasiadamente longo para duas variáveis de estado devido ao processo de cromatografia utilizado para medida, além de ser um processo não linear e que apresenta um fenômeno associado à adaptação da colônia de microorganismos ao meio onde o processo ocorre e que ocorre por razões ainda não totalmente conhecidas, fenômeno esse conhecido como *fase lag*. Além disso, será feito um estudo comparativo entre três algoritmos de treinamento de RNA, com vistas a melhoria do desempenho de treinamento.

Nos Capítulos 2, 3, e 4 será dada uma introdução teórica à Modelagem de Sistemas Dinâmicos, com um destaque para modelamento de processos bioquímicos, ao Processo Bioquímico alvo deste estudo, destacando os problemas associados e o modelo fenomenológico obtido pelos pesquisadores do Agrupamento de Biotecnologia do Instituto de Pesquisas Tecnológicas, e às Redes Neurais Artificiais, com ênfase para as redes tipo Perceptron Multicamadas e para o modelamento de processos dinâmicos. No Capítulo 5 será proposta uma aplicação de Redes Neurais Artificiais à modelagem do processo bioquímico, com propostas para a resolução de alguns problemas que surgem quando da aplicação de Redes Neurais Artificiais para o processo dinâmico em questão, enquanto o Capítulo 6 apresentará os resultados desta aplicação, apresentando-se os resultados das Redes Neurais Artificiais propostas no Capítulo 5 e comparando-se o desempenho de duas redes com o de um filtro estendido de Kalman baseado no modelo fenomenológico do Instituto de Pesquisas Tecnológicas. No Capítulo 7 serão discutidos os resultados, apresentando-se as conclusões e possíveis extensões do estudo aqui realizado.

2. MODELAGEM DE SISTEMAS DINÂMICOS

Para melhorar o entendimento ou para fins de controle de um processo, deve-se tentar construir um modelo que descreva este processo, de modo a tornar possível a compreensão dos fenômenos internos ao processo ou a simulação do mesmo. Neste capítulo será dada uma introdução à modelagem matemática de sistemas dinâmicos, com uma especial atenção à modelagem de processos bioquímicos.

2.1. Modelos Matemáticos

Modelos matemáticos são descrições de processos de natureza qualquer por meio de equações matemáticas que expliquem as relações causa-consequência relacionadas a estes processos. Estes modelos matemáticos podem ser baseados apenas nas relações causa-consequência do processo a ser modelado (modelo entrada-saída ou caixa preta) ou nos fenômenos que ocorrem internamente a este processo (modelo espaço de estado). No caso do modelo entrada-saída, temos informações apenas sobre o que consideramos saídas do processo, enquanto que no modelo espaço de estado podemos ter também as informações relativas aos fenômenos e estados internos do processo.

2.1.1. Modelo Espaço de Estado

Neste tipo de modelo, o conceito mais importante é o de estado [1]: "o estado x é a mínima quantidade de informação requerida para descrever completamente o comportamento do sistema, quando as entradas para todos os instantes são conhecidas". O modelo espaço de estado é construído como um conjunto de equações que relacionam as entradas, os estados e as saídas. A forma particular deste conjunto depende do tipo escala de tempo (contínuo ou discreto). O modelo discreto mais geral [2], para sistemas não lineares, está descrito pelas equações (2.1) e (2.2).

$$x(k+1) = g[x(k), u(k)] \quad (2.1)$$

$$y_p(k) = h[x(k), u(k)] \quad (2.2)$$

onde $\mathbf{x}(k) \in \mathbb{R}^n$ representa os estados internos, $\mathbf{u}(k) \in \mathbb{R}^r$ representa as entradas e $\mathbf{y}_p(k) \in \mathbb{R}^m$ representa as saídas do sistema. As funções $g : \mathbb{R}^{n+r} \rightarrow \mathbb{R}^n$ e $h : \mathbb{R}^{n+r} \rightarrow \mathbb{R}^m$ fazem o mapeamento dos estados e entradas atuais para estados futuros e dos estados e entradas atuais para saídas atuais, respectivamente. Se o processo é discreto, linear e invariante no tempo, pode ser descrito pelas equações (2.3) e (2.4)

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \quad (2.3)$$

$$\mathbf{y}_p(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) \quad (2.4)$$

onde $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ e $\mathbf{D}$ são matrizes $n \times n$, $n \times r$, $m \times n$ e $m \times r$, respectivamente.

2.1.2. Modelo Entrada-Saída

O modelo entrada-saída é baseado na idéia de que as saídas em instantes futuros do processo podem ser descritas como sendo uma função das entradas e saídas em instantes anteriores, sem utilizar a noção de estados internos. Este tipo de modelagem é utilizado quando não se tem acesso aos fenômenos internos ao processo ou quando se deseja construir um modelo adaptativo do processo em um computador, caso em que o modelo é discreto. O tipo mais geral de modelo entrada-saída discreto é o modelo NARMAX (*Non-linear Auto Regressive Moving Average eXtended* - não linear auto-regressivo média móvel estendido), onde o processo é descrito como sendo uma função não linear de suas entradas $\mathbf{u}(\cdot)$ e suas saídas $\mathbf{y}(\cdot)$ em instantes anteriores de tempo e de um ruído $\mathbf{n}_1(\cdot)$, conforme mostra a equação (2.5).

$$\begin{aligned} \mathbf{y}_p(k+1) = f(\mathbf{y}_p(k), \mathbf{y}_p(k-1), \dots, \mathbf{y}_p(k-N+1), \mathbf{u}(k), \mathbf{u}(k-1), \dots \\ \dots \mathbf{u}(k-M+1), \mathbf{n}_1(k)) \end{aligned} \quad (2.5)$$

Este é o modelo mais geral, embora seja um modelo complexo, pois envolve a escolha de dois parâmetros de recorrência de tempo N e M , de uma função não linear $f(\cdot)$, e de um termo de ruído $\mathbf{n}_1(\cdot)$. Podemos particularizar este modelo para simplificá-lo. Se retirarmos o termo de ruídos $\mathbf{n}_1(\cdot)$, teremos o modelo NARMA (auto-regressivo media móvel não linear - *Non-linear Auto-Regressive Moving Average*). Se considerarmos o valor de M como sendo igual a 1 temos o modelo NARX (não linear auto regressivo estendido - *Non-linear Auto-Regressive eXtended*), descrito pela equação (2.6).

$$\mathbf{y}_p(k+1) = f(\mathbf{y}_p(k), \mathbf{y}_p(k-1), \dots, \mathbf{y}_p(k-N+1), \mathbf{u}(k), \mathbf{n}_1(k)) \quad (2.6)$$

Se considerarmos a função $f(.)$ como sendo linear, teremos o modelo ARMAX (auto regressivo média móvel estendido - *Auto-Regressive Moving Average eXtended*), que pode ser descrito pela equação (2.7)

$$y_p(k+1) = \sum_{j=0}^{N-1} (A_j * y_p(k-j)) + \sum_{j=0}^{M-1} (B_j * u(k-j)) + n_1(k) \quad (2.7)$$

onde A_j e B_j são coeficientes do modelo.

2.2. Observabilidade e Controlabilidade

O conceito de observabilidade é muito importante para a modelagem de processos dinâmicos. Este conceito diz respeito a possibilidade ou não de se determinar os estados de um sistema dinâmico a partir de suas saídas e entradas. Uma definição formal de observabilidade [1] pode ser: "Um sistema é dito observável se para quaisquer dois estados x_1 e x_2 existir uma seqüência de entradas de tamanho finito U tal que a seqüência de saídas Y do sistema partindo do estado x_1 com entradas U é diferente da seqüência de saídas Y do sistema partindo do estado x_2 com entradas U ". O fato de um sistema ser observável leva à possibilidade de se estimar os estados do sistema ou de se identificar o próprio sistema a partir da observação de suas entradas e saídas.

Já o conceito de controlabilidade diz respeito a capacidade de se influenciar os estados de um sistema dinâmico através de suas entradas. Uma definição formal de controlabilidade [1] pode ser: "Um sistema dinâmico é controlável se para quaisquer dois estados x_1 e x_2 existe uma seqüência finita de entradas que transfere o sistema do estado x_1 para o estado x_2 ". O conceito de controlabilidade, assim como o de observabilidade, é importante para modelos espaço de estado, pois se um sistema for não controlável, há pelo menos uma variável de estado que não pode ser influenciada via entradas do sistema, assim como se o sistema for não observável há pelo menos uma variável de estado que não pode ser estimada a partir das saídas do sistema.

2.3. Relação entre Modelos Entrada-Saída e Espaço de Estado

Se todas as variáveis de estado do modelo espaço de estado de um sistema forem mensuráveis e o sistema for controlável, o modelo espaço de estado pode ser substituído por um modelo entrada-saída e ainda assim termos informações sobre os estados do sistema. A exigência de que o sistema seja controlável vem do fato que as variáveis de estado precisam ser manipuladas via entradas e estados anteriores no modelo entrada-saída, enquanto que a exigência de que todas as variáveis de estado estejam

presentes na saída vem do fato de que os únicos dados que temos no modelo entrada-saída são as saídas do modelo, assim, se quisermos informações sobre os estados, devemos tê-los na saída do sistema. Assim, para estes tipos de sistemas, podemos utilizar um modelo entrada-saída e estimar os estados do sistema, os quais estarão na saída do modelo. Caso todas as variáveis de estado não estejam na saída do sistema, torna-se necessário utilizar-se um observador de estado em conjunto com o modelo entrada-saída para que se possa estimar o estado do sistema, desde que este seja observável.

2.4. Estimação de Estados

A estimação de estados de um sistema dinâmico observável é um problema que surge quando os fenômenos físicos que ocorrem dentro do sistema são conhecidos mas a quantificação destes fenômenos (ou seja, a medida de todos os estados) não é possível. Assim, surge a necessidade de se gerar uma estimativa para aqueles estados que não podem ser medidos, função esta de um observador de estados. A construção de um observador de estados é feita a partir de um modelo espaço de estados do sistema, pois necessitamos saber as relações entre as variáveis de saída e os estados do sistema.

2.5. Identificação

A identificação de um sistema dinâmico parte do problema em que não se conhece os processos internos do sistema e deseja-se um modelo cujo comportamento em relação a entradas e saídas possa aproximar o comportamento do sistema. Para isso, claramente é necessário que o sistema seja observável. Neste caso podemos utilizar um modelo entrada-saída ou um modelo espaço de estado baseado em relações empíricas que possam de alguma forma modelar o comportamento do sistema em relação a suas entradas e saídas.

2.6. Modelos de Processos Bioquímicos

Os modelos de processos bioquímicos pertencem a uma classe de modelos cuja obtenção pode apresentar problemas [3]. Isto ocorre devido ao fato de que nem sempre se conhece todos os fenômenos internos e também à existência, na maioria dos casos, de substratos limitantes ou inibitórios, que são substâncias necessárias para que o processo ocorra e cuja falta e/ou excesso pode interferir negativamente, retardando ou até mesmo paralisando o processo. Como existe uma colônia de microorganismos relacionada ao processo, o modelo inclui a taxa específica de crescimento microbiano μ_x como uma variável. Na presença de inibição por substrato [4], a taxa específica de crescimento microbiano pode ser descrita pela equação (2.8)

$$\frac{1}{X} \frac{dX}{dt} = \mu_x = \frac{\hat{\mu} * S}{K_s + S + K_i S^2} \quad (2.8)$$

onde X é a concentração de microorganismos, S é a concentração do substrato limitante, $\hat{\mu}$ é a taxa de crescimento máxima e K_s e K_i são constantes. A equação (2.8) aplica-se ao caso em que existe apenas um substrato limitante e que também é inibitório. Caso haja mais de um substrato limitante, esta equação inclui outros termos, caso do processo fermentativo descrito no Capítulo 3.

Os modelos de processos bioquímicos podem ser divididos em estruturados e não estruturados e em segregados e não segregados. A divisão em estruturado e não estruturado refere-se ao modo como o comportamento das células é visto na modelagem: como sendo uma caixa preta (não estruturado) ou como sendo resultado de um conjunto de fenômenos internos (estruturado). A divisão em segregado e não segregado refere-se à diferenciação ou não dos estados fisiológicos das células, respectivamente. A tabela (2.1) mostra a classificação dos modelos [3].

Tabela 2.1 - Classificação dos Modelos Bioquímicos

	Não Estruturado	Estruturado
Não Segregado	células são indistintas entre si célula é uma caixa preta com respeito a processos metabólicos (maioria dos modelos)	células são indistintas entre si célula tem processos metabólicos e fisiológicos distintos
Segregado	células podem estar em estados fisiológicos diferentes célula é uma caixa preta com respeito a processos metabólicos	células podem estar em estados fisiológicos diferentes célula tem processos metabólicos e fisiológicos distintos (situação real)

O modo como é feita a modelagem dos processos internos às células e a diferenciação das mesmas reflete o tipo de equipamento de medida utilizado e a precisão desejada do modelo. Os métodos de medida de concentração de células mais utilizados levam a resultados do tipo concentração (peso/volume) de células presentes, não fazendo distinção entre células vivas ou mortas. Neste caso, um modelo segregado torna-se inviável. Novos métodos de medida têm sido propostos [5] para se tentar modificar esta situação, mas falta confiabilidade e/ou precisão a estes métodos. Para o caso de modelos não estruturados, um dos problemas associados a este tipo de modelo é que o mesmo não contempla a chamada fase *lag* (*lag phase*) [6], uma fase de adaptação que ocorre quando da introdução do inóculo (quantidade inicial de microorganismos, vindos de um determinado

meio) no reator (cujo meio é diferente do meio de onde o inóculo veio). Durante a fase *lag*, o crescimento microbiano é virtualmente zero, pois a colônia precisa se adaptar ao meio do reator antes de começar o crescimento. Se o processo a ser modelado é contínuo, ou seja, uma vez iniciado o processo, alimenta-se de substrato(s) o reator e retira-se o meio para extração do produto, esta fase pode ser ignorada, pois ela ocorre em um (normalmente) pequeno intervalo de tempo no início do processo. Caso o processo seja do tipo batelada (*batch*), ou seja, todos os componentes são colocados no reator antes de iniciar o processo e só há extração do produto no final, a fase *lag* pode representar problemas, pois estará presente em todos os ciclos do processo, durante um intervalo de tempo que pode ser significativo.

3. DESCRIÇÃO DO PROCESSO FERMENTATIVO

Segundo estimativas, o consumo brasileiro de vitamina C é da ordem de 1200 toneladas anuais [7], sendo que esta quantidade é totalmente importada, a um custo de US\$ 10,00 por quilo. Inicialmente, o ácido ascórbico era extraído de frutas cítricas, mas hoje em dia utiliza-se a síntese química, sendo o processo de Reichstein o mais utilizado [8]. Este processo utiliza glicose como matéria-prima e envolve várias etapas, sendo uma delas transformação bioquímica. Esta etapa será o objeto de estudo deste trabalho.

3.1. *Descrição do Problema*

O problema consiste em realizar a predição para o conjunto de variáveis, definidas como sendo variáveis de estado para a construção do modelo do IPT, da transformação de D-Sorbitol em L-Sorbose em batelada (processo descontínuo ou em lote) por meio do microorganismo *Gluconobacter Oxydans* [9]. O processo é realizado em um biorreator fechado e instrumentado e é relativamente lento (dependendo da quantidade inicial de sorbitol, pode durar de 8 a 20 horas) e tem problemas de medidas das variáveis envolvidas. Algumas têm medida rápida, outras têm medida lenta e há uma variável que não pode ser medida.

3.2. *Variáveis do Processo*

As principais variáveis envolvidas no processo são: concentração de oxigênio, concentração de sorbitol, concentração de extrato de levedura, concentração de sorbose e concentração de microorganismos [9]. Algumas destas variáveis são consideradas substratos limitantes, ou seja, para uma dada concentração de microorganismos em um dado instante é necessária uma concentração mínima deste substrato para que a colônia não sofra um colapso. A variável extrato de levedura é um composto de várias substâncias e como não há informação de qual destas substâncias é importante para o processo, esta variável não pode ser medida, tendo-se como única informação a concentração inicial em cada ensaio. A tabela (3.1) resume as principais características das variáveis envolvidas. Existem ainda duas grandezas, temperatura e pH do meio de cultura, que não estão colocadas neste trabalho pois são mantidas constantes

durante todos os ensaios utilizados. A temperatura é mantida em torno de 30 °C e o pH, em 4,5.

Tabela 3.1 - Características das Variáveis Envolvidas

<i>Nome</i>	<i>Símbolo</i>	<i>Unidade</i>	<i>Intervalo de Medição</i>	<i>Substrato Limitante</i>
concentração de oxigênio	O	mg/l	contínuo	sim
concentração de sorbitol	S ₁	g/l	1 hora	sim
concentração de extrato de levedura	S ₂	g/l	não medida	sim
concentração de sorbose	P	g/l	1 hora	não
concentração de microorganismos	X	g/l	1 hora	não

As variáveis S₁ e P são medidas por cromatografia líquida, utilizando-se um cromatógrafo líquido de alto desempenho (*High Performance Liquid Chromatograph* - HPLC), enquanto que a variável O é medida continuamente por uma sonda galvânica instalada dentro do fermentador. A variável X é obtida pelo método de peso seco, onde uma amostra de volume definido do meio de cultura é centrifugada, é feita uma suspensão do resíduo sólido da centrifugação em água e esta suspensão é passada por uma membrana de filtro de peso determinado e que não retém água. A diferença entre o peso final da membrana mais o filtrado e o peso da membrana é a medida da variável X. Vale ressaltar que este método não dá informação sobre o número de células viáveis (vivas) presentes, mas sobre a quantidade total de células (vivas e mortas) presentes na amostra.

3.3. Diagrama do Processo

A figura (3.1) mostra o diagrama esquemático do processo, com os métodos de medida. Note-se que a variável O pode ser controlada por um controlador externo, que atua na aeração e na agitação do reator, para manter esta variável dentro de certos padrões preestabelecidos. O cromatógrafo e a sonda estão conectados ao sistema, ou seja, o método de medida é automatizado. No caso do cromatógrafo, a amostra pode ser tomada nos intervalos de tempo determinados, filtrada, diluída e enviada ao cromatógrafo automaticamente, sem interferência humana, mas nos ensaios utilizados, as amostras foram tomadas manualmente. No caso da variável X, o processo de tomada de amostras e medição é executado manualmente.

Figura 3.1 - Diagrama Esquemático do Processo

3.4. Modelo do Processo

Um modelo não estruturado e não segregado do processo fermentativo para processos em batelada e contínuos obtido pelos pesquisadores do Agrupamento de Biotecnologia do IPT [9], é descrito pelas equações (3.1) a (3.16), divididas em equações de balanço de massa, equações cinéticas e equação de transporte. As definições de cada um deste parâmetros se encontra na Lista de Símbolos.

Equações de balanço de massa:

$$\frac{dX}{dt} = r_X \quad (3.1)$$

$$\frac{dP}{dt} = r_P \quad (3.2)$$

$$\frac{dS_1}{dt} = -r_{S_1X} - r_{S_1P} \quad (3.3)$$

$$\frac{dS_2}{dt} = -r_{S_2X} \quad (3.4)$$

$$\frac{dO}{dt} = -r_{OX} - r_{OP} + N_a \quad (3.5)$$

Equações cinéticas:

$$r_X = \mu X \quad (3.6)$$

$$\mu_X = \frac{\hat{\mu} S_1}{K_{S_1} + S_1 + S_1^2/W} \frac{S_2}{K_{S_2} + S_2} \frac{O}{K_O + O} \exp(-K_P P) \quad (3.7)$$

$$r_P = \mu_P X \quad (3.8)$$

$$\mu_P = A\mu_X + B \quad (3.9)$$

$$B = \frac{B_m S_1}{K_{bS_1} + S_1} \frac{O}{K_{bO} + O} \quad (3.10)$$

Equações de produção:

$$r_{S_1X} = \frac{1}{Y_{XS_1}} r_X \quad (3.11)$$

$$r_{S_1P} = \frac{1}{Y_{PS_1}} r_P \quad (3.12)$$

$$r_{S_2X} = \frac{1}{Y_{XS_2}} r_X \quad (3.13)$$

$$r_{OX} = \frac{1}{Y_{XO}} r_X \quad (3.14)$$

$$r_o = \frac{1}{Y_o} r \quad (3.15)$$

Equação de Transporte:

$$N_a = K_{La}(O_g - O) \quad (3.16)$$

Este modelo tem 16 parâmetros, dos quais 13 são ajustados e outros 3 (Y_{PS1} , Y_{PO} e O_g) são constantes estequiométricas e de equilíbrio. O ajuste dos 13 parâmetros do modelo foi obtido por um problema de minimização, cuja função objetivo a ser minimizada é a soma dos erros quadráticos entre os dados reais e a saída do modelo. Foram obtidos dois conjuntos de parâmetros para o modelo, sendo que o primeiro conjunto (conjunto 1) fornece o melhor ajuste para os ensaios onde a concentração de sorbitol inicial é alta, enquanto que o segundo conjunto (conjunto 2) fornece o melhor ajuste para ensaios onde a concentração inicial de sorbitol é baixa. Em uma análise estatística (teste de hipóteses), verifica-se que o melhor ajuste é gerado pelo conjunto 1. Os valores destes parâmetros para os dois conjuntos estão listados na tabela (3.2).

Tabela 3.2 - Parâmetros Associados ao Modelo Fermentativo

Parâmetro	conjunto 1	conjunto 2	unidade
$\hat{\mu}$	0.627	0.649	h^{-1}
A	18.9	19.2	$g\ g^{-1}$
B_m	7.21	6.78	$g\ g^{-1}\ h^{-1}$
K_{bO}	1.84	1.74	$mg\ l^{-1}$
K_{bS1}	1.69	1.39	$g\ l^{-1}$
K_O	0.246	0.111	$mg\ l^{-1}$
K_P	0.00424	0.00675	$l\ g^{-1}$
K_{S1}	4.25	3.03	$g\ l^{-1}$
K_{S2}	0.737	0.519	$g\ l^{-1}$
O_g	7.50	7.5	$mg\ l^{-1}$
W	189.0	169.0	$g\ l^{-1}$
Y_{PO}	0.01125	0.01125	$g\ mg^{-1}$
Y_{PS1}	0.989	0.989	$g\ g^{-1}$
Y_{XO}	0.000661	0.000677	$g\ mg^{-1}$
Y_{XS1}	0.426	0.397	$g\ g^{-1}$
Y_{XS2}	0.773	0.607	$g\ g^{-1}$

Este modelo, por ser não estruturado e não segregado, não inclui o fenômeno conhecido como fase *lag* (*lag phase*). Assim, para os ensaios que apresentam fase *lag*, o modelo não representa bem a fase inicial do processo, correspondente à fase *lag*, passando a representar melhor o processo após o final da mesma, que tem duração variável, atribuída a fatores ainda não muito bem conhecidos, relacionados à história do inóculo, tais como seu tempo de conservação e modo de preparação.

4. REDES NEURAIS

Redes Neurais Artificiais (RNA) são uma tentativa de se imitar o funcionamento do cérebro utilizando-se unidades simples de processamento baseadas nos modelos disponíveis de neurônios [10] [11] [12]. Uma definição mais detalhada pode ser obtida de [13]: "Uma Rede Neural é um conjunto interconectado de elementos simples de processamento, *unidades* ou *nós*, cuja funcionalidade é fracamente baseada nos neurônios animais. A capacidade de processamento da rede está armazenada nas forças de conexão entre as unidades, ou *pesos*, obtidos por um processo de adaptação a partir de, ou *aprendidos* de, um conjunto de padrões de treinamento". A idéia principal é agrupar estas unidades de processamento, baseadas em um determinado modelo de neurônio, de modo a formar uma estrutura que possa realizar uma dada tarefa, tal como: reconhecimento de imagens [14] [15], síntese de fala [16], modelagem de sistemas dinâmicos [2] [17] e vários outros tipos de tarefas. Dependendo do modelo de neurônio utilizado e do padrão de conectividade das unidades de processamento, temos diferentes arquiteturas de redes neurais. Esta arquitetura passa então por um processo de treinamento apropriado à mesma, o qual visa adaptar a rede à tarefa a ser executada. Como principais características das RNAs, podemos citar:

- i) São tolerantes a presença de ruídos;
- ii) Realizam processamento paralelo;
- iii) São robustas quanto a falhas em uma unidade;
- iv) Têm capacidade de tratar com padrões não treinados, tendo assim capacidade de generalização.

4.1. Modelos de Neurônios

Existem disponíveis vários modelos de neurônios, que diferem entre si no tipo de entradas, nas funções de ativação e no tratamento da ativação (dinâmica ou estática). O modelo mais utilizado é o Neurônio de McCulloch-Pitts [12], que será referenciado neste trabalho como Unidade Básica de Processamento (UBP). Este modelo é composto por um conjunto de entradas, um conjunto de pesos associados a estas entradas (um peso para cada entrada), um somador, uma função ou regra de ativação e uma polarização (*bias*), conforme mostra a figura (4.1).

Figura 4.1 - Diagrama Esquemático de uma UBP

A cada instante de processamento o valor de cada entrada é multiplicado pelo peso associado a esta entrada e somado aos valores das outras entradas multiplicados pelos respectivos pesos. O valor obtido por esta soma é passado então por uma função de ativação, para a qual o valor da polarização tem a função de deslocar a função no eixo das abcissas, conforme mostra a equação (4.1).

$$o_n = f\left(\left(\sum_k i_j * w_j\right) - b_n\right) \quad (4.1)$$

A função de ativação pode ser linear, dada pela equação (4.2), limiar ou *threshold*, equação (4.3), ou sigmoidais (em forma de "S"), sendo que as funções mais utilizadas nesta classe são a tangente hiperbólica (valores de saída entre -1 e 1), dada pela equação (4.4) e a sigmoide exponencial (valores de saída entre 0 e 1), dada pela equação (4.5) ou ainda do tipo gaussiana, dada pela equação (4.6).

$$f(a) = a \quad (4.2)$$

$$f(a) = \begin{cases} 1 & \text{se } a \geq 0 \\ 0 & \text{se } a < 0 \end{cases} \quad (4.3)$$

$$f(a) = \tanh(a) \quad (4.4)$$

$$f(a) = \frac{1}{1 + \exp(-a)} \quad (4.5)$$

$$f(a) = \exp(-a^2) \quad (4.6)$$

Frequentemente, as funções do tipo sigmoide e gaussiana são utilizadas nas camadas ocultas. Nas unidades de saída o tipo de função depende da aplicação da RNA: caso a RNA seja utilizada para reconhecimento de padrões ou aplicações que requeiram saídas binárias,

podem ser utilizadas as funções tipo sigmoidais ou gaussianas, enquanto que para o caso de valores contínuos, podem ser utilizadas as funções lineares ou sigmoidais, neste caso conjugadas com uma normalização dos padrões de entrada e saída que utilize apenas a região linear destas funções.

4.2. Padrão de Conectividade

As RNAs podem ser organizadas em camadas de unidades de processamento e o modo como as unidades se conectam entre si na mesma camada e entre camadas diferentes é chamado de padrão de conectividade. Podemos ter as unidades se interconectando com outras na mesma camada ou não, ter conexões entre camadas indo apenas em direção à camada de saída ou bidirecionais.

4.3. Arquitetura

A arquitetura (também chamada de topologia) de uma RNA refere-se à configuração estrutural da rede, ou seja, ao número de camadas, ao número de unidades em cada camada, e ao padrão de conectividade. Alguns tipos de RNA têm arquitetura fixa quanto ao número de camadas ou ao padrão de conectividade, outras apresentam liberdade de escolha da arquitetura. Dependendo da arquitetura utilizada, podemos ter redes *feedforward*, que admitem conexões apenas em direção à camada de saída e apenas entre unidades de camadas diferentes (também chamadas de *one-shot*), ou recursivas, onde as conexões podem ser bidirecionais. Podemos ainda ter redes totalmente conectadas (*full connected*), onde todas as conexões permitidas existem ou redes parcialmente conectadas, onde algumas das conexões permitidas não existem (ou existem com peso associado fixado em zero). A figura (4.2) mostra um exemplo de rede recursiva, no caso uma rede tipo Memória Associativa Bidirecional (BAM) [18]. Neste tipo de arquitetura, existem duas camadas de processamento, chamadas normalmente de x e y , e todas as unidades de processamento estão conectadas bidirecionalmente entre camadas. A entrada pode ser aplicada à camada x ou a camada y , de acordo com o resultado que se deseja obter. Após um certo tempo de acomodação, as camadas x e y apresentarão padrões que são considerados saídas da RNA. Este tipo de arquitetura é utilizada principalmente para construção de sistemas que armazenam associações entre dois padrões.

Figura 4.2 - Diagrama Esquemático de uma Rede Neural Artificial Recursiva

As redes recursivas diferem das *one-shot* na utilização, pois nas últimas a saída é instantânea, enquanto que nas primeiras há um transiente, pois devido à existência de conexões indo e voltando entre as camadas, ela se torna um sistema dinâmico.

Além das redes *feedforward* e das recursivas, existem ainda as redes recorrentes. Nestas redes há laços internos ou externos de realimentação de estados ou saídas da rede. É o caso da Rede de Elman [19], que é basicamente uma rede *feedforward* de duas camadas de processamento onde as saídas das unidades da primeira camada oculta são atrasadas em um intervalo de tempo e aplicadas à camada de entrada. Este é um tipo de realimentação interna. Outro caso pode ser o de uma rede cuja saída é aplicada com atrasos à entrada, sendo esta rede um caso de realimentação externa. A figura (4.3) apresenta uma rede de Elman.

Figura 4.3 - Diagrama Esquemático de uma Rede de Elman

4.4. Perceptrons Multicamadas

Redes perceptron multicamadas (MLP - *Multi-Layer Perceptrons*) são RNAs que utilizam camadas de UBP, com padrão de conectividade direta ou *feedforward*, ou seja, as unidades não se conectam entre si na mesma camada e as conexões entre camadas são unidirecionais. Opcionalmente, pode-se ter conexões da camada de saída diretamente para a camada de entrada, formando um laço externo de realimentação, quando então a rede é dita recorrente. A primeira camada é dita camada de entrada e não realiza processamento algum, apenas propaga as entradas para a camada seguinte. A última camada é chamada camada de saída, realiza processamento e é onde se recolhe a saída da rede. As camadas que não são nem de entrada nem de saída são chamadas de camadas ocultas, por não serem acessíveis externamente a rede. A figura (4.4) apresenta o diagrama esquemático de uma rede perceptron multicamadas com uma camada oculta, n entradas e m saídas.

Figura 4.4 - Diagrama Esquemático de uma Rede Perceptron Multicamadas

4.5. Métodos de Treinamento

O treinamento de uma rede neural consiste em ajustar os pesos e as polarizações das unidades que realizam processamento de modo a fazer com que a rede desempenhe a tarefa desejada. O algoritmo de treinamento utilizado depende da arquitetura e do padrão de conectividade escolhidos e pode ser supervisionado (quando há uma correspondência entre entradas e saídas desejadas) e não supervisionado (quando a rede escolhe a saída para cada entrada). No treinamento supervisionado, o algoritmo de treinamento exerce a função de tutor, o qual informa a rede qual a saída desejada para cada padrão de entrada. O treinamento então tende a reforçar as conexões da rede para os casos corretos e penalizar para os casos errados. Para o caso de redes tipo MLP, o algoritmo mais utilizado é o Retro-Propagação de Erro (*Error Backpropagation*), um método supervisionado, ou alguma de suas variações. A figura (4.5) representa um esquema de treinamento supervisionado. O algoritmo atua sobre a rede alterando os valores dos pesos e das polarizações das unidades de modo a reduzir a diferença entre a saída desejada e a saída da rede.

Figura 4.5 - Diagrama de Blocos de um Algoritmo de Treinamento Supervisionado.

Para o caso de treinamento não supervisionado, a própria rede escolhe as saídas para cada padrão de entrada e o ajuste dos pesos é efetuado de modo a refletir as semelhanças e diferenças entre os padrões de entrada (padrões semelhantes tendem a ter saídas semelhantes).

Existe ainda o caso de treinamento por algoritmos genéticos, que pode ser supervisionado ou não, quando o algoritmo dá origem a novas redes por combinações de duas redes existentes ou por mudanças (mutações) em uma rede existente. A cada passo deste algoritmo é efetuada uma avaliação das redes resultantes. As redes que tiverem melhor desempenho segundo um critério preestabelecido continuam a sofrer o processo de combinação e mutação, as que não obedecerem o critério são eliminadas. Este algoritmo é baseado na teoria evolutiva, ou seja, o mais apto sobrevive enquanto que o mais fraco sucumbe.

4.5.1. Algoritmo de Retro-Propagação Padrão (Backpropagation)

Este algoritmo [20] [11] [21] [22] [23] supervisionado utiliza o método do Gradiente Descendente para minimizar a função erro quadrático, dada pela equação (4.7)

$$Ep = \frac{1}{2} \sum_{j=1}^L (t_{jm} - y_{jm})^2 \quad (4.7)$$

onde t_{jm} é a j -ésima saída desejada e y_{jm} é a j -ésima saída da rede, ambos para o padrão j e L é o número de padrões no conjunto de treinamento. O algoritmo consiste em inicializar os pesos e as polarizações aleatoriamente, operar a rede para padrões de entrada de dados (para se obter a saída da rede para estes padrões) e propagar os erros (diferenças entre

saídas da rede e saídas desejadas) da camada de saída para a camada de entrada, possibilitando-se assim calcular as atualizações dos pesos e da polarização (ΔW_i) de cada unidade pela equação (4.8),

$$\Delta W_i = -R_i^{-1} * \nabla E_p \quad (4.8)$$

que representa um algoritmo básico de busca para minimização. Como este algoritmo utiliza o método de Gradiente Descendente, o cálculo da direção da busca se resume ao cálculo do gradiente, então o parâmetro de direção R_i é igual à matriz identidade multiplicada pelo inverso da taxa de aprendizado (lr - *learning rate*), um fator que dá o tamanho do passo do algoritmo, conforme equação (4.9).

$$R_i = \frac{1}{lr} I \quad (4.9)$$

Este algoritmo impõe a condição de que as funções de ativação sejam contínuas e diferenciáveis, para que seja possível calcular o Jacobiano J do erro em relação aos pesos e polarizações, conforme equação (4.10).

$$\nabla E_p = \frac{\partial E_p}{\partial W_i} = J^T \quad (4.10)$$

A taxa de aprendizado é quem determina, em última análise, a velocidade de convergência do treinamento. Uma taxa de aprendizado muito baixa pode aumentar o tempo de treinamento ou fazer com que o treinamento pare em um mínimo local indesejável, enquanto que uma taxa de aprendizado muito alta pode fazer com que a rede não convirja. Deste modo, na escolha do valor deste parâmetro deve-se buscar um compromisso entre velocidade de convergência e convergência. Este algoritmo de treinamento é iterativo, ou seja requer múltiplas passagens de cálculo de atualização de pesos para se atingir o ponto de operação desejado. Muitas vezes, o número de iterações necessário é muito grande, o que torna este algoritmo lento. Algumas alternativas têm sido propostas para melhorar a velocidade de treinamento, como a introdução de taxa de aprendizado adaptativa, fator de momentum e utilização de métodos que utilizam mais de um método de busca, como o algoritmo Levenberg-Marquadt.

4.5.2. Algoritmo de Retro-Propagação com *momentum* e taxa de aprendizado adaptativa

Neste algoritmo [25], são combinados dois métodos para se aumentar a velocidade de convergência: a constante de *momentum* e a taxa de aprendizado adaptativa. A constante de momentum (mc) é um fator de ponderação entre a atualização de pesos do passo anterior e a atualização de pesos calculada pelo Gradiente Descendente, conforme mostra a equação (4.11)

$$\Delta \mathbf{W}(k) = mc * \Delta \mathbf{W}(k-1) + (1 - mc) \Delta \mathbf{W}_{GD} \quad (4.11)$$

onde $\Delta \mathbf{W}(k)$ é a atualização de pesos e polarizações no instante atual, $\Delta \mathbf{W}(k-1)$ é a atualização de pesos e polarizações anterior e $\Delta \mathbf{W}_{GD}$ é a atualização de pesos do instante atual calculada pelo método do Gradiente Descendente. Esta técnica faz com que o treinamento não seja capturado em pequenas depressões (mínimos locais) do hiperplano de erros, evitando que o treinamento pare nestes pontos. O método da taxa de aprendizado adaptativa consiste em se calcular os pesos a cada iteração e comparar com os pesos calculados na iteração anterior. Caso o erro seja maior que o erro anterior por um fator pré-definido, os pesos atuais são descartados, mantém-se os pesos anteriores e decrementa-se a taxa de aprendizado, caso contrário, os novos pesos são mantidos e a taxa de aprendizado é incrementada. A utilização destes dois métodos de aceleração leva a um aumento no número de operações matemáticas a cada iteração.

4.5.3. Algoritmo de Levenberg-Marquadt

Este algoritmo é uma variação do algoritmo de retro-propagação padrão, sendo que a diferença está no algoritmo de minimização do erro, que não mais utiliza o método do Gradiente Descendente, mas uma mistura deste método com o Quasi-Newton [24] [25]. Esta diferença está embutida no cálculo da direção de busca, mostrado na equação (4.12)

$$\mathbf{R}_i = \mu \mathbf{I} + \mathbf{J}^T \mathbf{J} \quad (4.12)$$

que é ponderada por um parâmetro μ , baseado no erro. Quando μ está grande, o algoritmo tende para o retro-propagação padrão e quando μ está próximo de zero o algoritmo tende para o Quasi-Newton. Assim, quando o erro está grande, domina o algoritmo de retro-propagação padrão com uma taxa de aprendizado (neste caso, μ) alta, enquanto que quando o erro se torna pequeno o algoritmo dominante é o Quasi-Newton. Este algoritmo requer mais memória e executa mais operações por iteração, mas o passo dado em cada iteração é maior, o que leva a uma aceleração do treinamento.

O cálculo de μ segue um padrão invertido em relação ao cálculo da taxa de aprendizado adaptativa: se o erro em uma iteração diminui em relação à anterior, μ é decrementado; se o erro aumenta, μ é incrementado. A maior desvantagem deste método é que ele exige um número maior de operações por iteração e requer mais memória que o retro-propagação padrão, mas em compensação, o número de iterações necessário para atingir o ponto de operação desejado é menor, o que acelera o processo de treinamento. Quanto aos requerimentos de memória, tipicamente a memória exigida pelo algoritmo Levenberg-Marquadt (mem_{LM}) se relaciona com a memória exigida pelo método retro-propagação padrão (mem_{BP}) de acordo com a equação (4.13), onde L é o número de unidades na camada de saída e Q é o número de padrões de treinamento.

$$mem_{LM} = mem_{BP} * L * Q \quad (4.13)$$

Assim, este algoritmo é adequado para redes com poucas unidades na camada de saída e conjuntos de treinamento não extensos.

4.6. Redes Neurais e Identificação

Para utilizar redes neurais na identificação de sistemas dinâmicos [26] [27], precisa-se escolher um modelo apropriado. Duas classes de modelos de identificação consideradas são a dos modelos espaço de estado e a dos modelos entrada-saída. Os modelos espaço de estado para identificação, mostrados esquematicamente na figura (4.6), requerem o uso de retro-propagação dinâmica.

Figura 4.6 - Esquema de Identificação de Estados utilizando RNA.

Entretanto, se um modelo entrada-saída, descrito por um modelo tipo NARMA, conforme equação (4.14), onde f é uma função arbitrária e M e N são os tamanhos das janelas de atraso das entradas e das saídas, respectivamente, e mostrado na figura (4.7), puder representar o comportamento do sistema, o procedimento de identificação fica computacionalmente mais atraente.

Figura 4.7 - Esquema de Identificação Entrada-Saída utilizando RNA.

$$y_p(k+1) = f[y_p(k), y_p(k-1), \dots, y_p(k-N+1), u(k), u(k-1), \dots, u(k-M+1)] \quad (4.14)$$

Os problemas associados à identificação de sistemas dinâmicos resumem-se em:

- i) gerar uma aproximação de $f(\cdot)$ para os dados disponíveis durante o processo de modelagem;
- ii) gerar uma interpolação de $f(\cdot)$ para os demais dados.

Estes dois problemas não podem ser resumidos em apenas um, pois uma boa aproximação de $f(\cdot)$ pode ser gerada, mas esta mesma aproximação pode não ser válida para interpolar $f(\cdot)$. Assim, torna-se necessária a utilização de algum tipo de modelo que aproxime os dados disponíveis e que tenha capacidade de interpolar (dentro de uma faixa de erro tolerável) os demais dados, ou seja, que tenha capacidade de generalização. Neste contexto a utilização de RNA, principalmente para sistemas não lineares, é uma alternativa que vem sendo utilizada. A base para a utilização de RNA em modelagem de sistemas dinâmicos vem do teorema de Stone-Weierstrass [28], de acordo com o qual as funções do tipo sigmoidais e gaussianas são candidatas para aproximação de um mapeamento contínuo arbitrário. O teorema Weierstrass diz que uma função contínua arbitrária pode ser uniformemente aproximada por uma seqüência de polinômios, dentro de uma desejada precisão. O teorema de Stone-Weierstrass estabelece um critério simples que determinadas funções devem satisfazer para aproximar uniformemente funções contínuas arbitrárias. Este critério é satisfeito pelas funções do tipo sigmoidais (tangentes hiperbólicas e sigmoidais exponenciais) e do tipo gaussianas, entre outras. Além disso, existe o teorema de Kolmogorov [29], que diz que podemos representar exatamente todas as funções contínuas como uma superposição de um número finito de funções contínuas de uma variável e somas. Estes dois teoremas indicam que redes *feedforward* compostas por múltiplas camadas de UBPs com funções de ativação sigmoidais ou gaussianas podem ser utilizadas em aproximação e interpolação de funções e modelagem de sistemas dinâmicos [30] [31] [32] [33]. Alguns trabalhos [1] [2] mostram que a realimentação de dados de um modelo tipo caixa-preta pode gerar problemas de instabilidade devido ao aparecimento de dinâmicas indesejáveis de mais alta ordem, que não existem no processo original, mas podem aparecer no modelo devido aos erros inerentes ao processo no qual se gera a aproximação.

5. REDES NEURAIS NA MODELAGEM DO PROCESSO FERMENTATIVO

Neste capítulo, será discutida a utilização de RNAs tipo Perceptron Multicamadas com Atraso de Tempo (TDMLP - *Time Delay MultiLayer Perceptron*) para realizar a predição de um passo de 15 minutos adiante no tempo das variáveis de estado do processo de bio-oxidação de D-Sorbitol em L-Sorbose por meio do microorganismo *Gluconobacter Oxydans*. Como já foi exposto no capítulo 3, este processo possui várias problemas associados (taxas de amostragens variadas e variáveis imensuráveis ou com incertezas de medidas), além de ser um processo não linear.

5.1. Origem dos Ensaios

Os ensaios foram efetuados em um fermentador instrumentado no IPT. Inicialmente, o intervalo de amostragem foi de uma hora, mas estes dados foram interpolados de modo a simular um intervalo de amostragem de 15 minutos. Alguns ensaios foram considerados perdidos por diversos motivos (contaminação, por exemplo), sendo utilizados neste trabalho apenas os considerados bons. Alguns ensaios têm concentração de oxigênio (O) variável, enquanto outros têm uma quebra na regularidade do intervalo de amostragem. A classe de ensaios com oxigênio variável não foi utilizada neste trabalho, enquanto que a classe de ensaios com quebra na regularidade do intervalo de amostragem não foi utilizada para treinamento, embora ambas estejam presentes no conjunto de ensaios fornecido pelo IPT. A tabela (5.1) resume as condições iniciais de cada ensaio, após o tratamento de interpolação.

Tabela 5.1 - Condições iniciais dos ensaios disponíveis

Ensaio	Células (X) ($g l^{-1}$)	Sorbitol (S1) ($g l^{-1}$)	Extrato de Levedura (S2) ($g l^{-1}$)	Oxigênio (O) ($mg l^{-1}$)	Observação
2	0.16	63.2	5	3.75	
6	0.14	50.75	1.25	3.75	
8	0.23	58.9	7.5	3.75	
10	0.09	10.2	5	3.75	
11	0.16	48	5	1.5	
12	0.13	47.25	5	6	

Ensaio	Células (X) (g l ⁻¹)	Sorbitol (S1) (g l ⁻¹)	Extrato de Levedura (S2) (g l ⁻¹)	Oxigênio (O) (mg l ⁻¹)	Observação
13	0.19	49.3	5	0.375	
14	0.12	203.5	5	3.75	quebra de intervalo
15	0.12	98.75	5	3.75	quebra de intervalo
16	0.14	48.75	5	7.5	O variável
17	0.14	46.7	5	7.5	O variável
18	0.24	49.6	5	7.5	O variável
19	0.16	46.6	5	7.5	O variável - quebra de intervalo
20	0.08	47	5	3.75	
21	0.10	46.6	5	6	
25	0.09	45.5	5	3.75	
34	0.22	89	10	3.75	
36	0.21	152.25	10	3.75	
37	0.21	185.5	10	3.75	
39	0.23	201	20	3.75	
43	0.11	49.5	5	3.75	
44	0.09	51.0	5	3.75	
46	0.15	51.75	5	3.75	
47	0.46	170.5	10	3.75	
48	0.38	312.0	10	3.75	quebra de intervalo
49	0.36	178.5	10	3.75	
50	0.36	174.0	15	3.75	
51	0.095	49.25	5	3.75	
53	0.09	49.5	2.5	3.75	

5.2. Intervalo de Tempo

Conforme descrito no Capítulo 3, não há uniformidade na taxa de amostragem das variáveis. Torna-se necessário então uniformizar os intervalos de amostragem para que a RNA possa operar com apenas um intervalo de amostragem. Este intervalo foi definido pelos pesquisadores do IPT como sendo de 15 minutos, sendo que os intervalos entre as amostras foram preenchidos com dados interpolados por eles. Este intervalo também é considerado satisfatório para as necessidades de processamento da RNA, portanto o intervalo de amostragem da RNA será de 15 minutos. Vale lembrar que esta interpolação contém apenas os dados de X, P, O e S₁. Para S₂, temos apenas a concentração inicial de cada ensaio.

5.3. Variável Não Disponível

Para o treinamento da rede neural há a necessidade dos valores de S_2 , os quais não são disponíveis pelo problema de medida exposto no Capítulo 3. Há a necessidade de se gerar os valores para esta variável, considerando-a como uma única substância. Isso foi feito por meio de um filtro estendido de Kalman baseado no modelo obtido pelos pesquisadores do IPT, com uma mudança: os valores de μ_x e de μ_p foram estimados por meio de interpolação polinomial da variável X utilizando-se as equações de μ_x e μ_p dadas pelo modelo fenomenológico, para que o filtro tivesse alguma informação sobre a fase *lag*, o que não seria possível utilizando-se o modelo original do IPT. Este filtro estendido de Kalman foi utilizado para gerar os valores de S_2 para todos os ensaios a cada intervalo de tempo interpolado (15 minutos) e como um método padrão para a comparação de desempenho de predição para as outras três variáveis. A figura (5.1) ilustra o processo de obtenção de S_2 e utilização e treinamento da RNA.

Figura 5.1 - Fluxo de Dados.

5.4. Escolha dos Conjuntos de Dados

Para escolha do conjunto de dados é necessário levar em conta o fato de que o conjunto utilizado para treinamento deve ser representativo do processo a ser modelado, ou seja, deve conter situações do processo que permitam à rede ter informações para lidar com padrões que não estejam no conjunto de treinamento, ou seja, que permitam à rede ter capacidade de generalização. Para tentar atingir este objetivo, as faixas de variação de cada variável do processo foram divididas em duas classes, alta e baixa, e os conjuntos de treinamento e de testes foram montados seguindo o critério de se tentar colocar o máximo de combinações das classes das variáveis, sem redundância. Seguindo este critério, dois conjuntos foram obtidos: um de treinamento (CTR) e um de testes (CTE). A tabela (5.2) mostra quais ensaios foram incluídos em cada conjunto. O conjunto de treinamento foi utilizado para o treinamento da RNA, enquanto o conjunto de testes serve para avaliação do desempenho da RNA.

Tabela 5.2 - Ensaios presentes em cada conjunto

<i>Treinamento (CTR)</i>	<i>Testes (CTE)</i>
2	25
36	20
10	37
21	46
49	47
51	8

Além destes ensaios, o ensaio número 12 foi selecionado como ensaio para comparação de desempenho das RNAs e do filtro de Kalman, devido a um comportamento anômalo ocorrido durante a estimação de S_2 pelo filtro de Kalman.

5.5. Normalização dos Dados

Devido ao fato de que as variáveis de estado e de entrada têm faixas de variação diferentes, torna-se necessário padronizar o intervalo de variação das mesmas, pois se esta medida não for tomada o algoritmo de treinamento tenderá a dar maior ênfase ao ajuste das variáveis com maior valor. Neste contexto, a equação (5.3) fornece um procedimento para normalização de cada uma das variáveis, enquanto que a equação (5.4) fornece o procedimento para desnormalização da mesma. Ambas as equações necessitam de parâmetros auxiliares, dados pelas equações (5.1) e (5.2).

$$a = \frac{1 - 2\varepsilon}{\max(v) - \min(v)} \quad (5.1)$$

$$b = \varepsilon - a * \min(v) \quad (5.2)$$

$$v_n(k) = a * v(k) + b \quad (5.3)$$

$$v(k) = \frac{v_n(k) - b}{a} \quad (5.4)$$

Nas equações acima, $v(\cdot)$ é a variável a ser normalizada, $v_n(\cdot)$ é a variável normalizada, ε é uma constante entre 0 e 1 e $\max(v)$ e $\min(v)$ representam os valores máximo e mínimos, respectivamente, desta variável em todo o conjunto de dados disponíveis.

Tanto o padrão de entrada quanto o padrão alvo de treinamento são normalizados, assim a saída da rede precisa ser desnormalizada para obtermos o valor real de cada variável quando a RNA for utilizada para predição. Este procedimento de normalização é apropriado para se utilizar função do tipo sigmoidal exponencial na camada de saída. O parâmetro ε , chamado de fator de folga, serve para evitar que os valores das variáveis na saída caiam nas regiões não lineares da função de ativação. Caso seja utilizada função de ativação linear na saída, este parâmetro pode ser zerado.

5.6. Modelo Discreto do Processo

O modelo do processo obtido pelo IPT, descrito pelas equações (3.1) a (3.16), é um modelo contínuo. Para modelagem por redes neurais, é necessário que o modelo seja discreto no tempo. Para isso, associamos à discretização do modelo obtido pelo IPT o modelo entrada-saída tipo não linear auto-regressivo estendido (NARMAX), descrito pela equação (5.5). Para simplificar, utilizaremos os mesmos valores para M e N .

$$\begin{aligned} y_p(k+1) = f[& y_p(k), y_p(k-1), \dots, y_p(k-N+1), u(k), u(k-1), \dots \\ & \dots, u(k-M+1), W_1(k)] \end{aligned} \quad (5.5)$$

5.7. Modelagem por Rede Perceptron

Para modelar o processo, escolhemos uma rede Perceptron com atraso de tempo, cujo diagrama esquemático de utilização está mostrado na figura (5.2). Neste diagrama, TDL representa uma linha de atraso com derivações (*Tapped Delay Line*),

utilizada para gerar os atrasos de tempo. Basicamente, este tipo de rede consiste de uma rede Perceptron Multicamadas cujas entradas são as entradas e saídas do sistema dinâmico no instante atual e em N instantes de tempo anteriores. A saída da RNA é a predição das variáveis de estado do sistema para um instante de tempo adiante.

Figura 5.2 - Diagrama do Treinamento de uma Rede Perceptron com Atraso de Tempo.

Para se poder utilizar este tipo de RNA em estimação de estados, é necessário que o sistema seja observável, pois precisamos ter todas as variáveis de estado disponíveis para o treinamento da RNA.

Para a modelagem por rede Perceptron, precisamos definir a representação das variáveis de entrada e de saída em relação à posição das unidades de saída. Para isso, escolhemos a representação vetorial para variáveis de entrada e de estado, descrita pelas equações (5.6) e (5.7). Há a necessidade da definição da representação pois a ordem das variáveis dentro dos respectivos vetores deve ser respeitada durante o treinamento e a utilização da rede.

$$\mathbf{S}(k) = \mathbf{y}_p(k) = [X(k), P(k), S_1(k), S_2(k)]^T \quad (5.6)$$

$$\mathbf{U}(k) = \mathbf{u}(k) = [O(k)] \quad (5.7)$$

Para o modelo matemático da rede Perceptron multicamadas, escolhemos a notação vetorial, pois esta pode ser implementada mais facilmente na linguagem de programação MATLAB [34] e na linguagem C. A notação utilizada impõe a condição de que a função de ativação das unidades de uma mesma camada seja única, ou seja, todas as unidades de uma mesma camada têm a mesma função de ativação. Assim, definimos os vetores de entrada $\mathbf{Pe}$ por meio da equação (5.8) e o vetor de saída $\mathbf{T}$ por meio da equação (5.9).

$$\mathbf{Pe}(k) = [S(k), U(k), S(k-1), U(k-1), \dots, S(k-N+1), U(k-N+1)]^T \quad (5.8)$$

$$\mathbf{T}(k) = [S(k+1)] \quad (5.9)$$

Definimos também um vetor $\hat{\mathbf{T}}$, que representa a saída da rede para o padrão $\mathbf{Pe}$, ou seja, representa a estimativa $\hat{\mathbf{S}}$ para o estado $\mathbf{S}$. A relação entre $\mathbf{Pe}$ e $\hat{\mathbf{T}}$ para uma rede com uma camada oculta é dada pela equação (5.10), o qual representa o modelo matemático da rede em notação vetorial.

$$\hat{\mathbf{T}}(k) = f_1(f_2(\mathbf{Pe}(k) * \mathbf{W}_{11} - \mathbf{B}_{11}) * \mathbf{W}_{12} - \mathbf{B}_{12}) \quad (5.10)$$

As funções f_1 e f_2 são definidas como sendo versões vetoriais de f_1 e f_2 para um vetor $\mathbf{a}$ de dimensão $n \times 1$, conforme a equação (5.11).

$$\mathbf{f}_j(\mathbf{a}) = \begin{bmatrix} f_j(a_1) \\ f_j(a_2) \\ \vdots \\ f_j(a_n) \end{bmatrix} \quad (5.11)$$

A RNA pode ser descrita como um modelo NARMAX, conforme a equação (5.12), onde $\mathbf{W}_r$ é a matriz de pesos e polarizações da RNA.

$$\hat{\mathbf{T}}(k) = \hat{\mathbf{S}}(k+1) = f[\mathbf{y}_p(k), \mathbf{y}_p(k-1), \dots, \mathbf{y}_p(k-N+1), \mathbf{u}(k), \mathbf{u}(k-1), \dots, \mathbf{u}(k-M+1), \mathbf{W}_r] \quad (5.12)$$

5.8. Treinamento

A função do treinamento será fazer com que a saída da rede $\hat{T}$ se aproxime da saída desejada T para cada padrão P_e . Como algoritmo de treinamento, será utilizado o algoritmo de Levenberg-Marquadt. Este algoritmo executa mais operações matemáticas por iteração e requer uma quantidade de memória maior do que a requerida pelo algoritmo de retro-propagação padrão, mas, em geral, o número de iterações requerida para o treinamento é muito menor, o que torna o treinamento mais rápido. Serão utilizadas as rotinas existentes no *Neural Networks Toolbox V2.0 for Matlab* [25], um grupo de expansão de funções do *software* Matlab, instalado em uma estação SUN Sparcstation LX. Como critérios de parada, adotamos um valor final de erro de treinamento, calculado pelo Matlab de acordo com a equação (5.13), igual a 10^{-5} e um número máximo de iterações igual a 50. Assim que um dos critérios é atingido, o treinamento é encerrado.

$$ET = \sum_{i=1}^4 \sum_{j=1}^L (v_{in}(j) - \hat{v}_{in}(j))^2 \quad (5.13)$$

A variável $v_{in}(j)$ representa o valor real normalizado de uma das quatro variáveis de estado para o padrão de entrada j . A variável de estado será X , P , S_1 e S_2 , conforme i varie de um a quatro, nesta ordem. A mesma regra se aplica para $\hat{v}_{in}(j)$, apenas com a diferença de que neste caso não é o valor real, e sim o estimado.

5.9. Redes Utilizadas

Neste trabalho, foram estudadas 12 RNAs. Todas elas são do tipo TDMLP, com uma camada oculta, 4 unidades lineares na camada de saída e entradas e saídas normalizadas. Metade delas tem a janela de atrasos fixada em um ($N=1$), ou seja, estão presentes na entrada da RNA apenas a entrada e o estado atuais, enquanto que a outra metade tem a janela de atrasos fixada em dois ($N=2$), ou seja, estão presentes na entrada da RNA a entrada e o estado no instante atual e no instante anterior. Para cada um destes grupos, existem dois subgrupos: um cuja função de ativação da camada oculta é tipo *tansig* (*tangente hiperbólica*) e outro cuja função de ativação é tipo *logsig* (*sigmoidal exponencial*). Para cada tipo rede nestes subgrupos, foram estudadas RNAs com 8, 12 e 16 unidades na camada oculta. A tabela (5.3) mostra as arquiteturas de RNAs estudadas.

Tabela 5.3 - Arquiteturas de RNAs Estudadas

<i>N</i>	Ativação	Unidades
1	tansig	8
1	tansig	12
1	tansig	16
1	logsig	8
1	logsig	12
1	logsig	16
2	tansig	8
2	tansig	12
2	tansig	16
2	logsig	8
2	logsig	12
2	logsig	16

A figura (5.3) mostra, a título de ilustração, uma das RNAs utilizadas, no caso uma RNA com $N=1$ e 8 unidades na camada oculta.

Figura 5.3 - RNA com $N=1$ e 8 unidades na camada oculta.

5.10. Figura de Mérito

Para efeito de análise de desempenho das RNAs, torna-se necessária a introdução de uma quantificação do desempenho, levando-se em conta algum parâmetro que seja de interesse. Para isto utiliza-se a figura de mérito de erro médio quadrático (EMQ), baseado no erro entre o valor predito normalizado da variável e o valor real normalizado desta para um determinado conjunto de padrões. A equação (5.14) mostra como o EMQ é calculado.

$$EMQ = \frac{\sum_{i=1}^4 \sum_{j=1}^L (v_{in}(j) - \hat{v}_{in}(j))^2}{L} \quad (5.14)$$

A variável $v_{in}(j)$ representa o valor real normalizado de uma das quatro variáveis de estado para o padrão de entrada j . A variável de estado será X , P , S_1 e S_2 , conforme i varie de um a quatro, nesta ordem. A mesma regra se aplica para $\hat{v}_{in}(j)$, apenas com a diferença de que neste caso não é o valor real, e sim o estimado. L representa o número de padrões existentes no conjunto de testes.

5.11. Análise de Desempenho

A análise do desempenho das RNAs estudadas terá como base a figura de mérito EMQ e utilizará os conjunto de treinamento (CTR) e de teste (CTE) para o cálculo do EMQ. Assim, teremos um EMQ de treinamento e um EMQ de teste. O EMQ calculado para o conjunto de treinamento servirá para avaliar a RNA como sistema aproximador, já que fará a comparação da resposta da RNA com a resposta real para padrões que foram apresentados à mesma durante a fase de treinamento, enquanto que o EMQ calculado para o conjunto de teste servirá para avaliar a RNA como sistema interpolador, neste caso fazendo a comparação da saída da RNA com a saída real para padrões desconhecidos para a RNA, avaliando-se deste modo a capacidade de generalização da RNA em questão. Vale ressaltar que o EMQ mais relevante é o EMQ de testes, pois durante a sua utilização a RNA terá de tratar com dados que, em sua grande maioria, não foram apresentado à rede durante o processo de treinamento. Além disso, para efeito de comparação, será verificado o desempenho de redes com bons resultados e do filtro estendido de Kalman utilizado para gerar os valores de S_2 no tempo para o ensaio 12.

6. RESULTADOS

Neste trabalho avaliamos a utilização de RNAs do tipo TDMLP na predição um passo adiante das variáveis de estado de um processo fermentativo e a eficácia do algoritmo de Levenberg-Marquadt no treinamento destas RNAs.

6.1. Treinamento

Para avaliação da eficácia do algoritmo de treinamento empregado, foi efetuado um teste comparativo entre os algoritmos de retro-propagação padrão (*BP padrão*), de retro-propagação com momentum e taxa de aprendizado variável (*BP c/ momentum*) e de Levenberg-Marquadt, o qual não seguiu o esquema de treinamento padrão descrito acima. O teste consistiu em verificar o tempo de processamento de treinamento para os três algoritmos, com todos eles partindo do mesmo ponto de inicialização da rede. O critério de parada do treinamento era atingir 2000 iterações ou o erro de treinamento (ET), dado pelo somatório dos erros para o conjunto de treinamento, ficar abaixo de 0.01, prevalecendo a condição que ocorrer primeiro. Os treinamentos foram repetidos tres vezes para cada método, para que eventuais diferenças na carga de processamento da máquina utilizada (devido a outros processos que poderiam estar rodando juntamente com os treinamentos) pudessem ser minimizadas utilizando-se a média dos tempos de treinamento. A tabela (6.1) e as figuras (6.1), (6.2) e (6.3) mostram os resultados deste teste para uma RNA com 12 unidades tipo tansig e N=1.

Tabela 6.1 - Comparação dos Algoritmos de Treinamento

<i>Algoritmo</i>	<i>BP padrão</i>	<i>BP c/ momentum</i>	<i>Levenberg-Marquadt</i>
Erro final (ET)	0.3501	0.2678	0.0041
Iterações	2000	2000	6
Tempo Total médio (min)	17.910	36.36	2.10
Tempo médio por Iteração(s)	0.5391	1.09	20.96

Os tempos médios de treinamento totais e por iteração são apenas indicativos, devido ao fato de que a máquina utilizada é multi-tarefas. Foi tomado o cuidado de se colocar os programas de treinamento para serem executados em condições semelhantes de carga de processamento da máquina, mas não é possível se assegurar que as cargas de

processamento eram rigorosamente as mesmas durante os tres treinamento, portanto os tempos médios de treinamento devem ser utilizados com cautela.

Figura 6.1 - Evolução do Erro de Treinamento para BP padrão.

Figura 6.2 - Evolução do Erro de Treinamento para BP c/ momentum.

Figura 6.3 - Evolução do Erro de Treinamento para Levenberg-Marquadt.

Nota-se que, apesar do tempo gasto por iteração ser muito maior no caso do algoritmo Levenberg-Marquadt, este foi o único algoritmo que atingiu o critério de parada por erro, além do que o número de iterações necessárias foi muito menor, levando à aceleração do treinamento.

6.2. Predição

As análises de desempenho das RNAs utilizadas baseiam-se na figura de mérito EMQ, apresentada no capítulo 5. A tabela 6.2 mostra o EMQ calculado para os conjuntos de treinamento e de teste para cada uma das RNAs estudadas, com 50 iterações de treinamento.

Tabela 6.2 - Figuras de Mérito.

<i>N</i>	<i>ativação</i>	<i>unidades</i> <i>na</i> <i>camada</i> <i>oculta</i>	<i>EMQ treinamento</i>	<i>EMQ teste</i>
1	logsig	8	17.95×10^{-6}	178.70×10^{-6}
1	logsig	12	8.80×10^{-6}	209.50×10^{-6}
1	logsig	16	10.74×10^{-6}	878.90×10^{-6}
1	tansig	8	22.15×10^{-6}	137.60×10^{-6}
1	tansig	12	8.93×10^{-6}	159.10×10^{-6}
1	tansig	16	15.75×10^{-6}	174.30×10^{-6}
2	logsig	8	13.91×10^{-6}	354.90×10^{-6}
2	logsig	12	10.39×10^{-6}	155.90×10^{-6}
2	logsig	16	9.42×10^{-6}	59.64×10^{-6}
2	tansig	8	11.09×10^{-6}	58.69×10^{-6}
2	tansig	12	8.68×10^{-6}	58.54×10^{-6}
2	tansig	16	8.97×10^{-6}	47.81×10^{-6}

Para ilustrar os resultados obtidos, as figuras (6.4) a (6.11) mostram gráficos de predição de duas RNAs com bom desempenho para cada uma das variáveis dos conjuntos de treinamento e teste. Para se definir bom desempenho, é necessário frisar novamente que o EMQ mais importante é o de teste, que verifica a capacidade de generalização da RNA, que durante sua utilização receberá padrões que, em sua grande maioria, não lhe foram apresentados durante a fase de treinamento. A primeira metade das amostras é composta pelos padrões do conjunto de treinamento (CTR), enquanto que a segunda metade vem do conjunto de testes (CTE).

Figura 6.4 - Gráfico de Predição de X para N=12 unidades logsig.

Figura 6.5 - Gráfico de Predição de P para N=12 unidades logsig.

Figura 6.6 - Gráfico de Predição de S_1 para $N=12$ unidades logsig.

Figura 6.7 - Gráfico de Predição de S_2 para $N=12$ unidades logsig.

Figura 6.8 - Gráfico de Predição de X para N=2 12 unidades tansig.

Figura 6.9 - Gráfico de Predição de P para N=2 12 unidades logsig.

Figura 6.10 - Gráfico de Predição de S_1 para N=2 12 unidades logsig.

Figura 6.11 - Gráfico de Predição de S_2 para N=2 12 unidades logsig.

A tabela (6.3) mostra o EMQ calculado para o ensaio 12 para as RNAs e o filtro de Kalman. Note-se que o EMQ para o filtro de Kalman não inclui a variável S_2 , pois os valores considerados reais são os gerados pelo filtro de Kalman e o erro para esta variável seria zero caso fosse calculado. As figuras (6.12), (6.15), (6.18) e (6.21) mostram os gráficos de predição destas mesmas RNAs e do filtro de Kalman utilizado para gerar a dinâmica de S_2 para um ensaio em particular (ensaio 12). Este ensaio foi escolhido devido ao fato de ocorrer um comportamento anômalo na predição de S_2 por meio do filtro de Kalman. Note-se que os valores considerados como reais para efeito de treinamento de S_2 provêm do filtro de Kalman, assim não existe a curva correspondente ao valor real desta variável na figura (6.21). Um tipo de gráfico interessante para avaliação do desempenho é o gráfico de dispersão. Neste tipo de gráfico o valor real da variável é colocado no eixo das abcissas, enquanto que o valor predito é colocado no eixo das ordenadas. Para uma predição ideal, os pontos devem se situar sobre a reta que forma um ângulo de 45° com o eixo das abcissas e passa pela origem. Quanto mais afastado desta reta, pior a predição. As figuras (6.13), (6.16), (6.19) e (6.22) apresentam os gráficos de dispersão para N=1 e as figuras (6.14), (6.17), (6.20) e (6.23), os gráficos de dispersão para N=2, ambos para o ensaio 12.

Tabela 6.3 - Comparação de EMQ para Ensaio 12

<i>Preditor</i>	<i>EMQ</i>
RNA N=1	17.95×10^{-6}
RNA N=2	21.87×10^{-6}
Filtro de Kalman	144.07×10^{-6}

Podemos ver por estes valores que as RNAs tiveram um EMQ muito menor que o filtro de Kalman, mesmo que a variável S2 não tenha sido incluída no calculo do EMQ do filtro de Kalman. Visualmente, as figuras seguintes apresentam os resultados para este ensaio.

Figura 6.12 - Comparação de Predição para X ensaio 12.

Figura 6.13 - Gráfico de Dispersão de X para N=12 unidades tansig ensaio 12.

Figura 6.14 - Gráfico de Dispersão de X para N=212 unidades tansig ensaio 12.

Figura 6.15 - Comparação de Predição para P ensaio 12.

Figura 6.16 - Gráfico de Dispersão de P para N=1 12 unidades tansig ensaio 12.

Figura 6.17 - Gráfico de Dispersão de P para N=2 12 unidades tansig ensaio 12.

Figura 6.18 - Comparação de Predição para S_1 ensaio 12.

Figura 6.19 - Gráfico de Dispersão de S_1 para N=12 unidades tansig ensaio 12.

Figura 6.20 - Gráfico de Dispersão de S_1 para N=212 unidades tansig ensaio 12.

Figura 6.21 - Comparação de Predição para S_2 ensaio 12.

Figura 6.22 - Gráfico de Dispersão de S_2 para N=1 12 unidades tansig ensaio 12.

Figura 6.23 - Gráfico de Dispersão de S_2 para $N=2$ 12 unidades tansig ensaio 12.

Nos gráficos acima podemos notar que ambas as rede apresentam um desvio menor do que o apresentado pelo filtro de Kalman. Além disso, no caso de S_2 , o comportamento teórico desta variável é monotônico decrescente, pois é um substrato que é consumido e não produzido. O valor gerado pelo filtro de Kalman (considerado como real) apresenta uma elevação entre as amostras 35 e 40, o que é um comportamento irreal. Ambas as redes minimizaram esta elevação, aproximando-se do comportamento teórico desta variável.

6.3. Análise do Atraso de Tempo

Como visto na seção 6.2, em geral as RNAs com $N=2$ apresentam resultados melhores que suas equivalentes com $N=1$. Este fato apenas não serve para justificar a utilização de $N=2$ como tamanho do atraso de tempo. Um outro método para analisar a influência do tamanho do atraso de tempo é verificar os pesos associados a cada unidade de processamento da camada oculta. Se tomarmos duas RNAs treinadas, com o mesmo número de unidades na camada oculta, mas uma com $N=1$ e outra com $N=2$, poderemos analisar as magnitudes destes pesos e verificar se o atraso de tempo influi ou não. O método segue o raciocínio de que para a matriz de pesos de uma RNA com $N=2$, teremos uma região que corresponde às variáveis de entrada no instante atual e outra que corresponde às variáveis de estado no instante anterior. Se a magnitude dos pesos da região das variáveis do instante anterior for muito menor que a magnitude dos pesos da região das variáveis do instante atual, significando pouca influência das variáveis do instante

Unidade	$X(t)$	$P(t)$	$S_1(t)$	$S_2(t)$	$O(t)$	$X(t-1)$	$P(t-1)$	$S_1(t-1)$	$S_2(t-1)$	$O(t-1)$
1	-0.347	-0.034	0.448	0.823	-0.408	-0.132	-0.497	0.073	-0.485	0.622
2	-0.273	-0.473	0.868	0.768	-0.395	0.120	0.402	-0.461	-0.123	0.171
3	0.695	-1.282	0.540	0.182	-0.626	-0.418	0.504	-0.480	-0.241	0.610
4	0.319	0.038	-0.195	0.277	-0.043	0.208	0.934	-0.376	-0.431	-0.752
5	0.441	0.469	-0.780	-0.779	0.858	0.354	-0.491	-0.197	-0.020	-0.274
6	-0.208	0.233	0.742	-0.149	0.732	-0.487	-1.183	0.215	0.604	0.297
7	0.057	0.303	-0.789	0.239	-0.560	0.078	0.285	0.610	-0.880	-0.439
8	0.937	0.124	0.104	-0.357	0.634	-0.342	0.284	-0.033	0.374	-0.975
9	-0.838	-0.061	0.890	-0.160	0.141	0.583	-0.035	0.496	0.243	-0.374
10	-1.199	-0.484	-0.399	0.278	0.334	0.428	0.403	-0.579	-0.014	-0.493
11	-0.249	0.556	0.452	-0.325	-0.601	-0.332	0.663	0.358	0.587	-0.223
12	0.310	0.355	0.018	-0.549	0.435	0.244	0.782	-0.230	-0.191	-1.224

A tabela (6.5) mostra que os pesos da área sombreada, correspondentes às variáveis de entrada do instante anterior, têm as mesmas ordens de magnitude que os pesos da área não sombreada, correspondentes às variáveis de entrada do instante atual. Portanto, para este caso, as variáveis de entrada do instante anterior têm influência comparável à das variáveis de entrada do estado atual, assim justifica-se o uso de $N=2$, e não de $N=1$, como tamanho do atraso de tempo.

em conta para a escolha da função de ativação, já que o EMQ poderia ser reduzido para qualquer uma das RNAs com a continuação do treinamento a partir do ponto de parada do treinamento inicial, mas a velocidade de convergência é maior no caso daquelas que apresentaram melhores resultados.

Em geral, as redes com 12 unidades na camada oculta apresentaram melhores resultados, mas possivelmente as redes com 16 unidades possam apresentar um comportamento ainda melhor caso seja efetuada uma continuação de treinamento a partir do ponto onde parou o treinamento original, tomando-se o devido cuidado para evitar o sobre-treinamento, que acarretaria perda da capacidade de generalização. O número de iterações fixado pode ser suficiente para ajustar os pesos e polarizações das RNAs com 8 e 12 unidades na camada oculta, mas não o das RNAs com 16 unidades, devido ao fato destas últimas terem muito mais pesos e polarizações que as anteriores, resultando assim num treinamento mais demorado.

As RNAs estudadas demonstraram ter aprendido o comportamento da fase *lag*, fato demonstrado pelo bom acompanhamento pelas RNAs das curvas de dados reais no início dos ensaios, quando a fase *lag* está presente. Isto se deve muito provavelmente ao fato da rede ter sido utilizada em um esquema de predição e ao fato de que a fase *lag* estava presente em todos os ensaios do conjunto de treinamento.

Finalmente, as redes neurais artificiais do tipo perceptron multicamadas com atraso de tempo estudadas neste trabalho demonstraram ser uma alternativa viável para a predição do processo de bio-oxidação de D-Sorbitol em L-Sorbose por meio do microorganismo *Gluconobacter Oxydans*.

Como extensões deste trabalho, podem ser sugeridos estudos quanto a utilização de RNAs do tipo TDMLP no controle do processo estudado, utilização das mesmas RNAs na estimação dos parâmetros de processo μ_p e μ_x , implementação em tempo real da RNA de melhor desempenho como estimador para o processo no IPT e utilização das RNAs também em processos de fermentação de alimentação contínua e não apenas processos em batelada. Se utilizadas em conjunto com o modelo do IPT, estas redes gerariam estimadores híbridos, os quais poderiam melhorar o modelo original do IPT sem que o estimador resultante fosse do tipo caixa preta, podendo assim apresentar informações sobre o processo que não foram obtidas pelas RNAs deste trabalho.

7. CONCLUSÕES

7.1. *Treinamento*

O algoritmo Levenbeg-Marquadt de treinamento de RNA mostrou-se eficaz para as RNAs utilizadas neste trabalho, superando com vantagem os algoritmos de Retro-Propagação padrão e Retro-Propagação com momentum e taxa de aprendizado variável, tanto na redução do número de iterações necessárias para se atingir o ponto de operação da RNA quanto na aceleração do treinamento em termos de tempo de processamento. O fato de que este algoritmo requer mais memória não se constituiu um problema, visto as RNAs utilizadas serem pequenas, assim o algoritmo Levenberg-Marquadt mostrou ser o método de treinamento mais indicado para esta aplicação.

Vale ressaltar que todos os treinamento efetuados com o algoritmo Levenberg-Marquadt pararam pelo critério de número máximo de iterações e não pelo erro fixado. Isto pode significar que a continuação do treinamento a partir do ponto em que o treinamento original parou ou o treinamento com um número máximo de iterações maior podem trazer melhorias no desempenho da rede. Contudo, esta continuação deve ser feita com critério, pois pode levar a um sobre-treinamento da RNA, com conseqüente diminuição da capacidade de generalização.

7.2. *Predição*

Para a predição do processo, as redes mostraram-se melhores que o filtro de Kalman, inclusive aproximando o comportamento teórico da variável S_2 para um ensaio que não foi utilizado para treinamento (ensaio 12) quando o filtro de Kalman gerou uma seqüência com comportamento diferente do teórico, mostrando assim ter capacidade de generalização.

Em geral, as redes com $N=2$ resultaram em uma predição melhor que as redes com $N=1$, fato esse devido muito provavelmente ao fato das redes com $N=2$ terem um histórico maior do processo (estados e entradas atuais e em um instante de tempo anterior) do que as redes com $N=1$ (apenas estado e entrada atuais), tendo este fato relevância, conforme mostrado na seção 6.3. Estas RNAs não acarretam um custo computacional muito maior que o das RNAs com $N=1$, portanto as RNAs com $N=2$ mostraram serem as mais indicadas para este tipo de aplicação.

Comparando as funções de ativação para RNAs equivalentes, as funções de ativação tipo tansig mostraram-se melhores que as logsig para o caso de RNAs com $N=2$, enquanto que para $N=1$ as melhores foram as logsig. Assim, este fato deve ser levado

BIBLIOGRAFIA

- [1] A. U. Levin, *Neural Networks in Dynamic Systems*, PhD. Thesis, Yale University, 1992.
- [2] A. U. Levin, K. S. Narendra, "Control on Nonlinear Dynamical Systems Using Neural Networks - Part II: Observability, Identification and Control", *IEEE Transactions on Neural Networks*, vol. 7, pp. 30-42, 1996.
- [3] P. S. Dhurjati, R. J. Leipold, "Biological Modelling", *Computer Control of Fermentation Processes*, D. R. Omstead (editor), CRC Press, pp. 207-220, 1990.
- [4] A. P. Fordyce, J. B. Rawlings, T. F. Edgar, "Control Estrategies for Fermentation Processes", *Computer Control of Fermentation Processes*, D. R. Omstead (editor), CRC Press, pp. 165-206, 1990.
- [5] J. A. Phillips, "'Novell' Sensors for the Monitoring of Fermentation Processes", *Computer Control of Fermentation Processes*, D. R. Omstead ed., CRC Press, pp. 15-72, 1990.
- [6] E. Kawashima, "Mathematical Model Taking Account the Lag Phase in Microbial Growth", *J. Ferment. Technol.*, vol. 51, pp. 41-46, 1973.
- [7] IPT, "Produção de Vitamina C: Levantamento de Dados para Conversão de Sorbitol em Sorbose por Via Fermentativa", Report 26, 072, São Paulo, 1988.
- [8] J. Boudrant, "Microbial Processes for Ascorbic Acid Biosynthesis: a Review", *Enzyme Microb. Technol.*, vol. 12, pp. 322-329, 1990.
- [9] A. Bonomi, E. F. P. Augusto, N. S. Barbosa, M. N. Mattos, L. R. Magossi, A. L. Santos, "Unstructured Model Proposal for the Microbial Oxidation of D-Sorbitol to L-Sorbose", *Journal of Biotechnology*, vol. 31, pp. 39-59, 1993.
- [10] J. J. Hopfield, D. W. Tank, "Computing with Neural Circuits, A Model", *Science*, vol. 233, pp. 625-633, 1986.

- [11] D. E. Rumelhart, J. McClelland and the PDP Research Group, *Parallel Distributed Processing*, Vols. 1 e 2, Cambridge MA: the M.I.T. Press, 1986.
- [12] W. S. McCulloch, W. H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.
- [13] K. Gurney, "Computer and Symbols versus Nets and Neurons", Notas de Aula, Brunel University, 1994.
- [14] D. E. Glover, "Optical Fourier/Electronic Neurocomputer Machine Vision Inspection System", *Proceedings of the Vision 88 Conference*, 1988.
- [15] J. A. Freeman, "Neural Networks for Machine Vision Applications: The Spacecraft Demonstration", *Exponent: Ford Aerospace Technical Journal*, pp.16-20, 1988.
- [16] T. J. Sejnowski, C. R. Rosenberg, "Parallel Networks that Learn to Pronounce English Text", *Complex Systems*, vol. 1, pp. 145-168, 1987.
- [17] K. S. Narendra, K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Transactions on Neural Networks*, vol.1, pp. 4-26, 1990.
- [18] B. Kosko, "Bidirectional Associative Memory", *IEEE Transactions on Systems, Man And Cybernetics*, vol. 1, pp. 49-60, 1988.
- [19] Elman, J. L. "Finding structures in time", *Cognitive Science*, vol. 14, pp. 179-211, 1990.
- [20] P. Werbos, *Beyond Regression: New Tool for Prediction and Analysis in The Behavioural Sciences*, PhD Thesis, Harvard University, Cambridge, MA., 1974.
- [21] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning Internal Representations by Error Propagating", *Parallel Data Processing*, vol. 1, chapter 8, Cambridge MA: the M.I.T. Press, pp. 318-362, 1986.
- [22] T. J. Sejnowski, "Neural Networks Learning Algorithms", *Neural Computers*, Springer-Verlag, pp. 291-300, 1988.
- [23] G. E. Hinton, "Connectionist Learning Procedures", *Artificial Intelligence*, vol. 40, pp. 107-118, 1989.

- [24] D. W. Marquadt, "Solution of Nonlinear Chemical Engineering Models", *Chemical Engineering Progress*, vol. 55, pp. 65-70, 1959.
- [25] H. Demuth, M. Beale, *Neural Network Toolbox User's Guide*, The MathWorks Inc., 1994.
- [26] K. S Narendra, K. Parthasarathy, "Identification and Control of Dynamical System using Neural Networks - A Survey", *Automatica*, vol. 28, pp. 1083-1112, 1992.
- [27] R. Zbikowski, K. J. Hunt, A. Dzielinski, R. Murray-Smith, P. J. Gawthrop, *A Review of Advances in Neural Adaptive Control Technologies*, 1994.
- [28] M. H. Stone, "The Generalized Weierstrass Approximation Theorem", *Mathematics Magazine*, vol. 21, pp. 167-184, 237-254, 1948.
- [29] A. N. Kolmogorov, "On the representation of Continuous Functions of Many Variables by Superpositions of Continuous Functions of One Variable and Addition", *Dokl. Akad. Nauk SSSR*, vol. 114, pp. 953-956, 1957, (Língua Original: Russo, Tradução para Língua Inglesa encontra-se no vol. 28 de American mathematical Society translations).
- [30] G. Cybenko, "Approximation by Superposition of a Sigmoidal Function", *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303-314, 1989.
- [31] K. Hornik, M. Stinchcombe, H. White, "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, vol.2, pp. 359-366, 1989.
- [32] V. V. Breusegem, J. Thibault, A. Chérui, "Adaptive Neural Models for On-Line Prediction in Fermentation", *The Canadian Journal of Chemical Engineering*, vol. 69, pp. 481-487, 1991.
- [33] T. Poggio, F. Girosi, "Networks for Approximation and Learning", *Proc. IEEE*, vol. 78, pp. 1481-1497, 1990.
- [34] The MathWorks, Inc., *Matlab User's Guide*, The MathWorks Inc. ,1994.