UNIVERSIDADE ESTADUAL DE MARINGÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

PROGRAMA DE MESTRADO EM ENGENHARIA QUÍMICA

CONTROLE DE PROCESSOS COM REDES NEURAIS INVERSAS

Gilberto Clóvis Antonelli

Eng.º Químico, UEM, 1984

Orientador: Prof. Ivo Neitzel, D.Sc.

Dissertação de Mestrado submetida à Fundação Universidade Estadual de Maringá, como parte dos requisitos necessários à obtenção do Grau de Mestre em Engenharia Química, área de Desenvolvimento de Processos.

Maringá, PR – Brasil

1998

UNIVERSIDADE ESTADUAL DE MARINGÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

PROGRAMA DE MESTRADO EM ENGENHARIA QUÍMICA

Esta é a versão final da dissertação de Mestrado apresentada por Gilberto Clóvis Antonelli perante a Comissão Julgadora do Curso de Mestrado em Engenharia Química em 29 de Maio de 1998.

COMISSÃO JULGADORA

Prof. Ivo Nettzel, D.Sc.
Orientador

Prof. Ariovaldo Bolzan, D.Sc.
Membro

Prof. Mauro Antonio da Silva Sá Ravagnani, Dr.Eng.
Membro

à minha esposa, **Silvana Adriani Remundini**, minha *princesa*, companheira incansável, a qual devo muito do que sou hoje e um pouco do que pretendo ser amanhã.

Agradecimentos

Agradeço a todas as pessoas que direta ou indiretamente contribuíram para a elaboração deste trabalho, em especial;

a meus pais, *Antonio Antonelli e Izidia Silvestre,* que nunca mediram esforços para que eu seguisse minha carreira de engenheiro,

a meu irmão, *Umberto Carlos Antonelli*$_{(in\ memorium)}$, que me mostrou o caminho da engenharia que ele tanto amou, e com certeza, onde quer que esteja, me incentivou a fazer o melhor,

a meu orientador, *Prof. Ivo Neitzel*, que me ajudou grandemente em todos os momentos que necessitei, com paciência e boa vontade,

a minha amiga, *Adriana Rossoni Pedrozo do Nascimento,* que juntos montamos o módulo de testes e alteramos o *software* de controle utilizado neste trabalho,

aos meus amigos, *funcionários do DEQ e funcionários do PEQ,* que sempre foram prestativos quando precisei de ajuda,

e ao *CNPQ,* que me auxiliou financeiramente durante a elaboração deste trabalho.

CONTROLE DE PROCESSOS
COM REDES NEURAIS INVERSAS

AUTOR: GILBERTO CLÓVIS ANTONELLI

ORIENTADOR: PROF. DR. IVO NEITZEL

Dissertação de Mestrado; Programa de Pós-Graduação em Engenharia Química; Universidade Estadual de Maringá; Av. Colombo, 5790, BL E46 – 09; CEP: 87020-900 – Maringá – PR, Brasil, defendida em 29 de maio de 1998. 126 p.

Resumo

O técnica de controle de processos utilizando redes neurais artificiais teve seu grande impulso a partir da década de 80, quando surgiram várias propostas de utilização. Este trabalho visa fazer uma comparação entre essa técnica, considerada recente, com as técnicas denominadas convencionais, da década de 40. Para tanto, realizamos a comparação entre o desempenho de um controlador baseado em rede neural artificial não-linear e um controlador PID. Também comparamos dois tipos de treinamento envolvendo as redes neurais artificiais não-lineares, o *Steepest Descent* e o *Decomposição em Valores Singulares*.

Descrevemos neste trabalho os passos utilizados para essas comparações, iniciando com o desenvolvimento do *software* de controle, implantação dos algoritmos dos controladores utilizados e montagem de um módulo de testes.

São apresentados também detalhes do *software* de controle, dos tipos de treinamento e da caracterização do módulo de teste.

As vantagens, desvantagens e dificuldades de implantação, tanto no modo simulado quanto no módulo de testes, estão descritos neste trabalho destacando sempre qual o controlador que obteve melhor performance.

CONTROL OF PROCESSES
WITH NETS INVERSE NEURAIS

AUTHOR: GILBERTO CLÓVIS ANTONELLI

SUPERVISOR: PROF. DR. IVO NEITZEL

Dissertação de Mestrado; Programa de Pós-Graduação em Engenharia Química; Universidade Estadual de Maringá; Av. Colombo, 5790, BL E46 – 09; CEP: 87020-900 – Maringá – PR, Brasil, defendida em 29 de maio de 1998. 126 p.

Abstract

The use of artificial neural networks in control process had its great pulse starting from the decade of 80, when several use proposals appeared. This work seeks to do a comparison among that technique, considered recent, with the conventional denominated techniques of the decade of 40. For so much, we accomplished the comparison among the a controller's acting based on nonlinear artificial neural network and a controller PID. We also compared two training types involving the nonlineal artificial neural network, *Steepest Descent* and *Decomposition in Singular Values*.

We described in this work the steps used for those comparisons, beginning with the development of the control software, implementation of the used controllers' algorithms and assembly of a module of tests.

They are also presented details of the control software, of the training types and of the characterization of the test module.

The advantages, disadvantages and difficulties of implementation, so much in the simulate way as in the module of tests, are described in this work always highlighting which the controller that obtained better performance.

Índice

Figuras

Tabelas

Notação

alfabeto normal

a	constantes do processo de primeira ordem na forma discreta
c	constantes do controlador PID na forma discreta
C	capacitor
d	constantes do controlador PID na forma discreta
e	desvio da variável desejada em relação ao *set-point* do processo contínuo, erro
E^2	soma dos erros da variável controlada em relação ao *set-point* durante um período de tempo; erro quadrado
h_o	sinal de saída em integradores/diferenciadores eletrônicos
h_i	sinal de entrada em integradores/diferenciadores eletrônicos
k	instante em sistemas amostrados
K	constante de proporcionalidade em processo contínuo
Kc	constante do controlador PID em processo contínuo
K_P	constante do modo proporcional do controlador PID em processo contínuo
K_I	constante do modo integral do controlador PID em processo contínuo
K_D	constante do modo derivativo do controlador PID em processo contínuo
L	número de neurônios da camada intermediária da rede neural
M	número de neurônios da camada de saída da rede neural
M	valor da ação do controlador PID em processos contínuos
m	valor da ação do controlador PID em processos discretos
n	número de intervalos de amostragens
N	número de neurônios da camada de entrada da rede neural; total de conexões do elemento de processamento
o	valores de saída da rede neural
O	vetor de saída da rede neural
P	conjunto de pares de vetores de treino da rede neural; pares de vetores {X,Y}
R	resistência

t	tempo
T	tempo resultante do desvio de controladores eletrônicos PID da idealidade
T_s	tempo de amostragem do processo
t_m	tempo morto do processo contínuo
u	vetor de entrada de um processo contínuo, variável manipulada
w	valor do peso das conexões da rede neural; força das conexões
x	valor de entrada da rede neural
X	vetor de entrada da rede neural
y	vetor de saída de um sistema contínuo, variável controlada
Y	vetor dos valores corretos de saída da função de mapeamento

alfabeto grego

α	valor líquido de entrada do EP; valor de ativação; ativação
θ	peso das conexões *bias* da rede neural; força da conexão *bias*
τ	constante de tempo do processo contínuo
τ_i	constante do modo integral do controlador PID em processo contínuo; constante de integradores eletrônicos
τ_d	constante do modo derivativo do controlador PID em processo contínuo; constante de diferenciadores eletrônicos
ϕ	função de mapeamento do vetor X no vetor Y
γ	função de aproximação do vetor X no vetor Y
ω	frequência do sinal de entrada em integradores/diferenciadores eletrônicos

símbolos compostos

BPN	rede neural artificial *BackPropagation*
EP	elemento de processamento da rede neural; neurônio
PID	controlador Proporcional Integral Derivativo
RN	rede neural
ANN	rede neural artificial (*Artificial Neural Network*)
SP	valor do *set-point* do processo
RN	rede neural artificial

RN_{SD}	controlador RN com treinamento *Steepest Descent*
RN_{DVS}	controlador RN com treinamento Decomposição em Valores Singulares

outros símbolos

$Z\{\}$	transformada z
$\|\ \|$	valor absoluto
G(s)	função de transferência na transformada de Laplace
$f(s)$	transformada de Laplace da função $f(t)$
$\Delta v(k)$	diferença entre os valores de v no instante k e no instante anterior (k-1)
$f(i\omega)$	transformada de Fourier da função $f(t)$
H(z)	função de transferência na transformada z

1. Introdução

A tecnologia de controle baseada em redes neurais artificiais teve seu maior impulso na década de 80, por isso, podemos dizer que é uma técnica recente quando comparada com as técnicas denominadas convencionais surgidas na década de 40. Este trabalho tem por objetivo avaliar a implantação desta técnica no controle de um sistema experimental, bem como, comparar os resultados da mesma com os controladores convencionais da família PID.

Inicialmente elaboramos as alterações necessárias no *software* de controle em tempo real de processos, anteriormente desenvolvido pelo Dr. Ivo Neitzel. Essas alterações foram realizadas em conjunto com a Engª Adriana R. Pedrozo, a qual utilizou o mesmo *software* em seu trabalho de comparação de métodos de treinamento de redes neurais artificiais. As alterações envolveram primeiramente a mudança da linguagem de programação, pois o *software* original era escrito para o compilador FORTRAN MicroSoft[1] e foi rescrito para o compilador FTN77/486[2] da Universidade de Salford, já que o mesmo possui vantagens em relação ao FORTRAN MicroSoft (Capítulo 6). Realizamos outras modificações como a implantação do algoritmo de redes neurais artificias, pois o mesmo utilizava o algoritmo de controladores da família PID; mudanças em rotinas de leituras dos dados das placas de comunicações do computador com o módulo de teste, pois eram escritas em linguagem ASSEMBLER e incorporadas ao compilador FORTRAN. Já as novas rotinas foram escritas totalmente na linguagem FTN77/486.

A tarefa seguinte foi a construção do módulo experimental para testes utilizando equipamentos do Departamento de Engenharia Química (DEQ). O módulo foi construído com opções de aplicação das perturbações no fluxo de ar do aquecedor (Figura 4.1).

Neste trabalho incorporamos ao programa um sistema de controle com redes neurais inversas com dois tipos de treinamento para a rede neural, o *Steepest Descent* e o *Decomposição em Valores Singulares*.

Foram realizados vários testes de confiabilidade das mudanças incorporadas no *software* e no módulo de testes (Capítulo 7).

Utilizando o computador realizamos a simulação do processo em várias situações diferentes. Ajustamos o controlador baseado em redes neurais para o controle eficiente utilizando os dois tipos de treinamentos (Capítulo 9). Efetuamos a sintonia de um controlador

[1] FORTRAN Versão 3.34 - Compilador fortran que trabalha em processadores 386, ou superiores, em modo real
[2] FTN77/486 Versão 4.0 - Compilador fortran 77 que trabalha em processadores 486, ou superiores, em modo protegido

PID e observamos a sua performance em processo simulado. Com os resultados obtidos foram comparados os dois tipos de controladores e os dois tipos de treinamento da rede neural, procurando mostrar as principais diferenças no comportamento de cada situação.

Como parte final do trabalho verificamos o desempenho dos controladores no controle do módulo de testes, observando suas reações a situações com perturbações externas ao módulo. Mostramos também, utilizando o controlador baseado em redes neurais, a facilidade de mudança no valor do *set-point* do processo via *software* (Capítulo 10).

Com o intuito de mostrar as dificuldades existentes no controle de um processo real realizamos uma análise entre os resultados obtidos na simulação e os obtidos no controle do módulo de testes (Capítulo 11).

Através deste trabalho podemos ter uma idéia de como implantar e desenvolver um controle de processos utilizando controladores baseados em redes neurais, sua dificuldade de implantação e os cuidados que devemos ter para levar a bons resultados.

2. O Sistema de Controle Convencional - PID

Um sistema de controle autosintonizante é uma técnica na qual, o controlador em atuação é ajustado periodicamente com base em perturbações aplicadas ao processo (SANOFF E WELLSTEAD, 1985).

Considerando um controlador PID, podemos descrever esta técnica pelo diagrama:

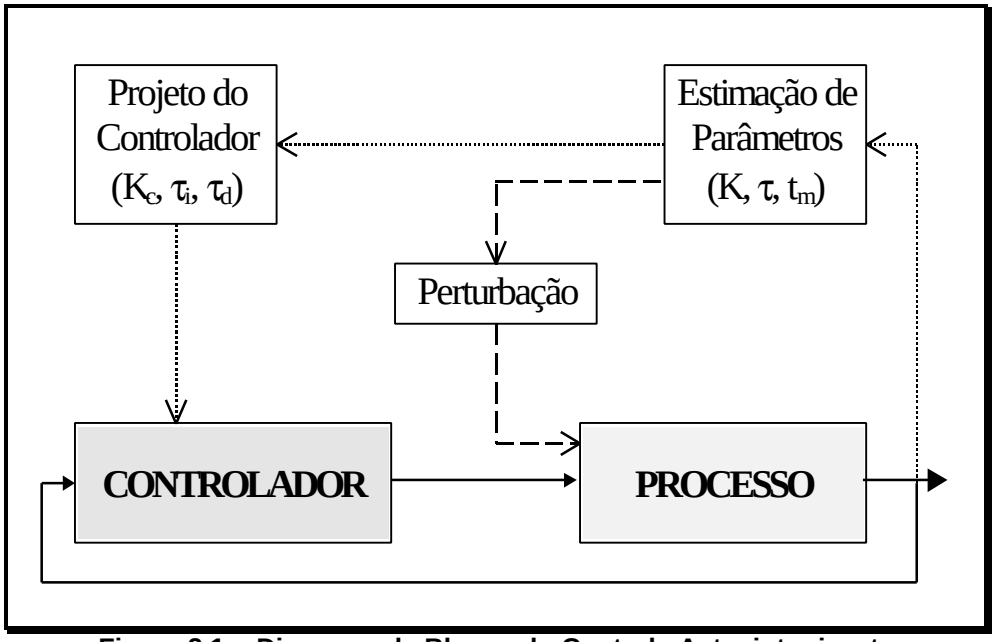


Figura 2.1 - Diagrama de Blocos do Controle Autosintonizante para Controladores PID

Como podemos observar este tipo de algoritmo é constituído de dois laços distintos, um superior e um inferior (GOODWIN E SIN, 1984). O laço inferior é o de controle convencional com realimentação (ZIEGLER E NICHOLS, 1942). O laço superior realiza uma estimação dos parâmetros para projeto do controlador, utilizando para isto as informações de resposta do processo a uma perturbação.

Deste modo a sintonia do controlador é avaliada periodicamente e atualizada no laço de controle, com uma frequência que depende das características do processo controlado, podendo chegar a uma vez por intervalo de amostragem.

O processo será modelado para um sistema contínuo de primeira ordem com tempo morto, descrito pela equação $t\,\dot{y}(t) + y(t) = Ku(t - t_m)$, onde;

y - variável controlada **u** - variável manipulada

K - constante de proporcionalidade do processo

t - constante de tempo do processo **t_m** - tempo morto

As constantes e o tempo morto serão determinadas a partir da resposta do processo a uma perturbação, sendo seus valores utilizados para projetar o controlador e determinar seus parâmetros (Figura 2.1).

Podemos observar que todo o conhecimento do processo é transferido para o controlador na forma de constantes.

Nos controladores PID utilizamos essas constantes para determinar as constantes do modo proporcional, integral e derivativo da equação do controlador, descrita como:

$$M(t) = \left[\mathrm{K}_{\mathrm{P}} e(t) + \mathrm{K}_{\mathrm{I}} \int_0^t e(t)dt + \mathrm{K}_{\mathrm{D}} \frac{de(t)}{dt} \right] \qquad (2A)$$

M - valor da ação do controlador

e - desvio da variável desejada em relação ao *set-point*, erro

$\mathbf{K_p}$ - constante do modo proporcional - corrige a variável manipulada com quantidade proporcional ao desvio da variável desejada em relação ao Set-point.

$\mathbf{K_I}$ - constante do modo integral - corrige a variável manipulada com velocidade proporcional ao desvio da variável desejada em relação ao Set-point.

$\mathbf{K_D}$ - constante do modo derivativo - corrige a variável manipulada de uma quantidade proporcional a velocidade de variação da variável desejada em relação ao Set-point.

A equação 2A também pode ser escrita da seguinte forma;

$$M(t) = \mathrm{K}_{\mathrm{C}} \left[e(t) + \frac{1}{t_i} \int_o^t e(t)\,dt + t_d \frac{de(t)}{dt} \right] \qquad (2B)$$

onde;

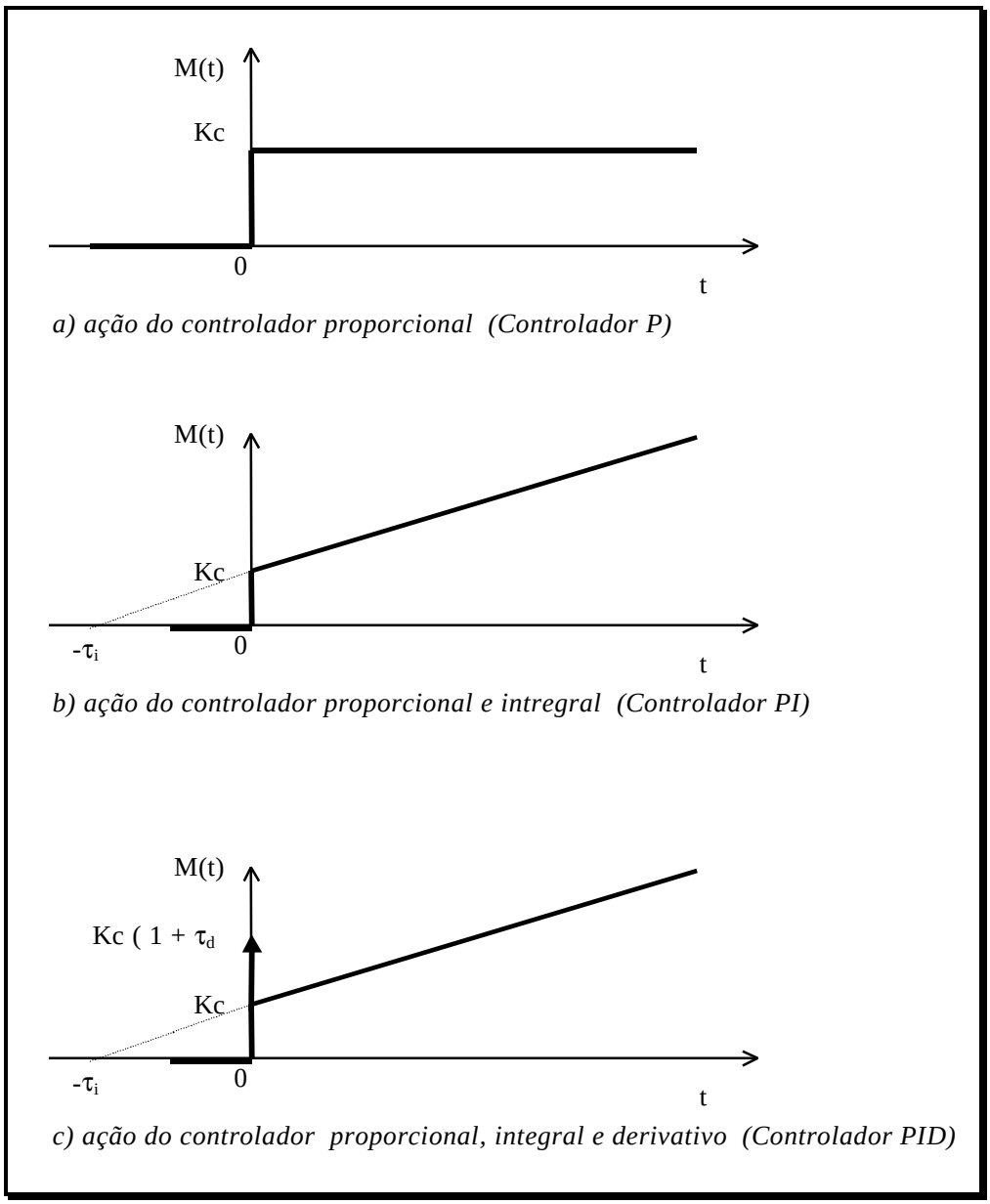
$$\mathrm{K_c} = \mathrm{K_p} \quad ; \quad t_i = \frac{\mathrm{K_p}}{\mathrm{K_I}} \quad \text{e} \quad t_d = \frac{\mathrm{K_D}}{\mathrm{K_P}}$$

2.1 Sintonia do Controlador PID

Neste trabalho ajustamos um controlador PID a um sistema de 1ª ordem com tempo morto. O controlador obedece a Equação 2A e possui a função de transferência:

$$G(s) = \frac{M(s)}{e(s)} = \left[\mathrm{K_p} + \frac{\mathrm{K_I}}{s} + \mathrm{K_D} s \right] = \mathrm{K_c} \left[1 + \frac{1}{t_i s} + t_d s \right] \qquad (2C)$$

A ação ideal de um controlador PID em termos dos modos proporcional, integral e derivativo, é representada graficamente na Figura 2.2



a) ação do controlador proporcional (Controlador P)

b) ação do controlador proporcional e intregral (Controlador PI)

c) ação do controlador proporcional, integral e derivativo (Controlador PID)

Figura 2.2 - Representação gráfica da ação de controladores da família PID

Na Figura 2.2a temos a ação do controlador constante desde o tempo inicial t=0. No caso dos controladores PI (Figura 2.2b) a ação inicial é produzida somente pela parte proporcional, porque a ação integral é nula no tempo t=0. Isto é facilmente observado na equação do controlador (Equação 2B), onde a integral (soma) dos desvios é zero nesse instante e cresce com o decorrer do tempo. Para os controladores PID (Figura 2.2c) temos no instante inicial o acréscimo de uma parcela decorrente da ação diferencial.

Observamos que na prática a implementação de controladores com modo integral e derivativo não ocorre de acordo com as Figuras 2.2b e 2.2c. Existe um desvio da idealidade devido à construção eletrônica.

2.1.1 Integrador eletrônico

Os integradores eletrônicos são construídos de acordo com o esquema da Figura 2.3, sendo representados pela transformada de Fourier:

$$\frac{h_o}{h_i}(iw) = \frac{1}{iw\,t_i + 1} \tag{2D}$$

onde;

h_o - sinal de saída

h_i - sinal de entrada

ω - frequência do sinal de entrada

τ_i - constante do integrador τ_i=RC

temos que para $\omega\tau_i >> 1$;

$$\frac{h_o}{h_i}(iw) \approx \frac{1}{iw\,t_i} \qquad \text{ou seja} \qquad h_o(t) \approx \frac{1}{t_i}\int_0^t h_i(t)dt \tag{2E}$$

Portanto o integrador eletrônico produz bons resultados para um sinal de entrada com espectro de frequência tal que $\omega\tau_i >> 1$. Aparentemente qualquer frequência ω pode ser englobada através da escolha de τ_i suficientemente grande. Contudo, grandes valores de τ_i retardam a ação de saída; podemos então reduzi-lo até o nível de ruído que o sistema permita.

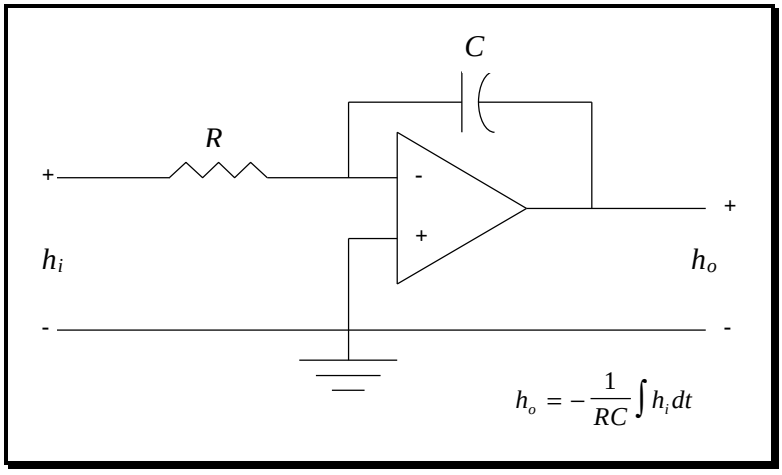


Figura 2.3 - Integrador eletrônico

2.1.2 Diferenciador eletrônico

Todos os filtros passa-alta (Figura 2.4) podem ser empregados como diferenciadores aproximados para um sinal de entrada, dentro de uma restrita faixa de frequência. São representados pela transformada de Fourier como:

$$\frac{h_o}{h_i}(iw) = \frac{iw\,t_d}{iw\,t_d + 1} \qquad (2F)$$

onde;

h_o - sinal de saída

h_i - sinal de entrada

ω - frequência do sinal de entrada

τ_d - constante derivativa do diferenciador, τ_d=RC

para $\omega\tau_d << 1$ temos ;

$$\frac{h_o}{h_i}(iw) \approx iw\,t_d \qquad \text{ou seja} \qquad h_o(t) \approx t_d\,\frac{dh_i(t)}{dt} \qquad (2G)$$

Notamos que para um dado valor de τ_d, a aproximação necessita de pequenos valores de ω. O valor de τ_d pode ser reduzido para estender a faixa de precisão do diferenciador à altas frequências. Contudo pequenos τ_d reduzem a sensibilidade; no entanto o nível de ruído não é limitante como no integrador aproximado.

Os diferenciadores produzem boas aproximações somente para frequências em que $\omega\tau_d << 1$, assim devemos estimar a maior frequência ω a ser utilizada e escolher τ_d apropriado.

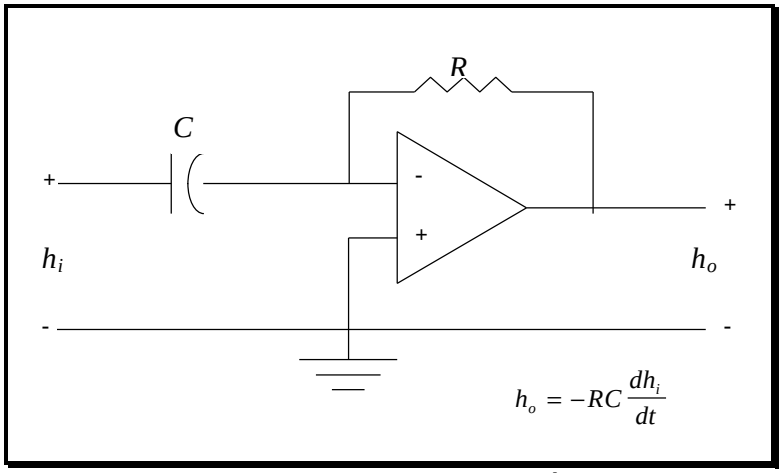


Figura 2.4 - Diferenciador eletrônico

2.1.3 Controlador PID eletrônico

A maior dificuldade no ajuste de um controlador PID eletrônico está no módulo diferenciador, pois os problemas do módulo integrador são contornáveis. Comparando-se as figuras 2.3c e figura 2.5, observamos o desvio da idealidade do módulo diferenciador. Utilizamos uma variável T para quantificar esse desvio. A inclusão dessa variável altera a função de transferência do modo diferencial, o que implica na transformação da função de transferência do controlador (Equação 2C) em;

$$G(s) = \frac{M(s)}{e(s)} = \left[\mathrm{K_p} + \frac{\mathrm{K_I}}{s} + \mathrm{K_D}\,\frac{\mathrm{T}s}{\mathrm{T}s+1} \right] = \mathrm{K_c}\left[1 + \frac{1}{t_i s} + \frac{t_d s}{\mathrm{T}s+1} \right] \quad (2H)$$

onde;

$$\mathrm{K_c} = \mathrm{K_p} \qquad ; \qquad t_i = \frac{\mathrm{K_p}}{\mathrm{K_I}} \qquad \text{e} \qquad t_d = \frac{\mathrm{K_D T}}{\mathrm{K_P}}$$

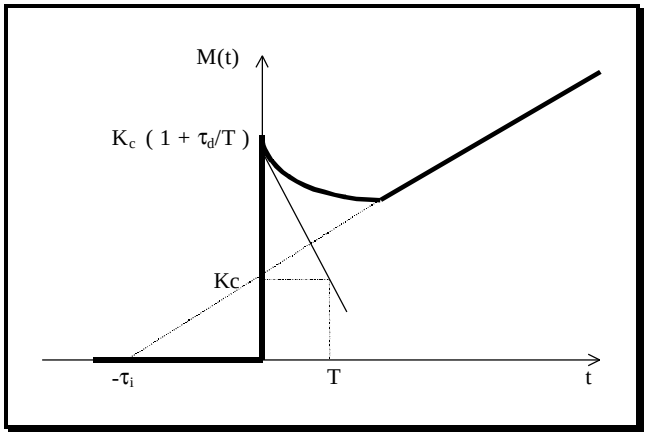


Figura 2.5 - Representação gráfica da ação de um controlador PID eletrônico

2.1.4 Controlador PID digital

Neste trabalho implementamos um controlador digital utilizando o algoritmo da velocidade na forma discreta (UNBEHAUEN, 1985). Para a obtenção do algoritmo utilizamos a função de transferência do controlador (Equação 2H) e determinamos a correspondente na transformada z. Segundo Unbehauen, a mudança da equação se faz pela substituição das equações a seguir na função de transferência;

$$s \approx \frac{z-1}{zT_s} \qquad \text{e} \qquad \frac{1}{s} \approx \frac{T_s}{2}\frac{z+1}{z-1}$$

onde T_s é o tempo de amostragem do processo.

Realizando a substituição temos:

$$H(z) = \frac{M(z)}{E(z)} = K_c\left[1 + \frac{T_s(z+1)}{2t_i(z-1)} + \frac{t_d}{zT_s}\frac{z-1}{1+T(z-1)/(zT_s)}\right] \qquad (2I)$$

através de manipulações matemáticas (Apêndice IV) obtemos a equação a seguir:

$$\Delta m(k) = d_0 e(k) + d_1 e(k-1) + d_2 e(k-2) - c_1 \Delta m(k-1) \qquad (2J)$$

onde ; $d_0 = \frac{1{,}2\,t}{K_p(T_m - T_s)}\left[\frac{1}{(1+{}^{T}\!/_{Ts})} + \frac{2Ts(T_m+T_s)}{(2T_m+T_s)^2} + \frac{T_m+T_s+2T}{1+{}^{T}\!/_{Ts}}\right]$

$$d_1 = \frac{1{,}2\,t}{K_p(T_m+T_s)}\left[\frac{T_s(T_m+T_s)}{(1+{}^{T}\!/_{Ts})(2T_m+T_s)^2} - \frac{T_m+T_s+2T}{T_s(1+{}^{T}\!/_{Ts})} - \frac{1}{1+{}^{T}\!/_{Ts}}\right]$$

$$d_2 = \frac{1{,}2\,t}{Kp(T_m+T_s)}\left[\frac{T_m+T_s+2T}{2T_s(1+{}^{v}\!/_{Ts})} - \frac{T(T_m+T_s)}{(1+{}^{T}\!/_{Ts})(2T_m+T_s)^2}\right]$$

$$c_1 = -\frac{T}{T_s+T}$$

Δm(k) = diferença entre a ação do controlador no instante (k) e no instante(k-1)

k = representa o instante atual

k-1 = representa um instante anterior

k-2 = representa dois instantes anteriores

T_s = tempo de amostragem do controlador

T = tempo resultante do desvio de controladores eletrônicos PID da idealidade. Na versão discreta T=0.

T_m = tempo morto do processo

Utilizando esse algoritmo implementamos no programa de controle os controladores P, PI e PID.

3. Rede Neural Artificial

A idéia de utilizar redes neurais artificiais como uma potencial estratégia de solução de problemas que requerem análises complexas de dados não é recente. A mais de 40 ou 50 anos, os cientistas tem procurado simular a estrutura real do cérebro humano e desenvolver um algoritmo equivalente ao processo de aprendizado humano. A principal motivação da pesquisa advém da capacidade de guardar sofisticados níveis de informações e processamentos que o cérebro possui. Entretanto, a estrutura do cérebro é extremamente complexa, com aproximadamente 10^{11} neurônios contendo de 10^{14} a 10^{15} sinapses (conexão entre os neurônios). Sabendo que a função de um único neurônio é bem definida, a arquitetura das ANNs[3] é baseada no conhecimento das funções primitivas do neurônio biológico. Entretanto, como não temos um modelo preciso das diversas funções do cérebro humano, as redes neurais artificiais procuram assimilar e utilizar uma confecção filosófica, mais modesta e em menor escala (WILLIS ET AL, 1990).

3.1 Definição e aplicação

As redes neurais são construídas com camadas altamente interconectadas de simples neurônios. Os neurônios atuam como um elemento de processamento (EP) não-linear dentro da rede. Uma propriedade atrativa das redes neurais artificiais é que elas são capazes de aproximar funções não-lineares.

A representação esquemática de um EP(elemento de processamento, ou neurônio) pode ser vista pela Figura 3.1. Cada elemento é numerado, de um até *i*-ésimo elemento recebendo várias entradas (x_j) e possuindo uma única saída (o_i). Cada conexão de entrada é associada a um peso ou força da conexão (w_{ij}) (FREEMAN E SKAPURA, 1992).

O valor líquido de entrada do elemento é uma somatória do produto das conexões e seus respectivos pesos. Assim, o valor líquido da entrada do *i*-ésimo elemento pode ser escrito como:

$$a_i = \sum_j^N w_{ij} x_j$$

onde *N* representa o total de conexões de entrada do elemento de processamento. Esse valor líquido (a_i) recebe o nome de *valor de ativação* , ou simplesmente *ativação*.

[3] Artificial Neural Networks - Redes neurais artificiais

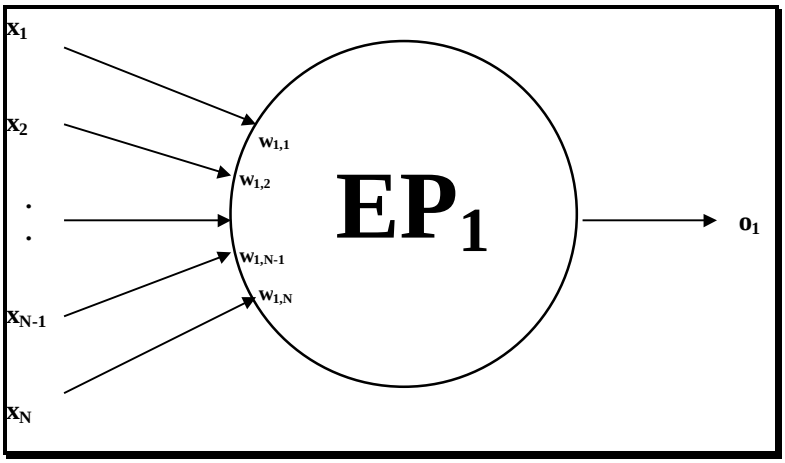


Figura 3.1 - Estrutura de um Elemento de Processamento

A saída do elemento de processamento é determinada a partir da ativação através da aplicação de funções lineares ou não-lineares, as chamadas *funções de ativação*. Podemos representar a saída dos elementos como:

$$o_i = f_i(a_i)$$

onde f_i representa a função de ativação do *i*-ésimo elemento. Serão analisados apenas dois tipos de funções de ativação. O primeiro, seria a função de ativação linear, isto é, o valor da função assume o mesmo valor da ativação:

$$f_i(a_i) = a_i$$

onde *i* representa o *i*-ésimo elemento de processamento. Essa função fornece às redes neurais artificiais características lineares. O segundo tipo, é a função de ativação sigmoidal, que apresenta características não-lineares. Matematicamente, a função sigmoidal pode ser expressa por:

$$f_i(a_i) = \frac{1}{(1 + e^{-a_i})}$$

A Figura 3.2 representa graficamente a resposta de uma função sigmoidal. Observamos que ela é limitada pelos valores extremos 0 e 1. Essa característica da função sigmoidal atua como um limitante do sinal de saída fornecendo valores com extremos bem definidos. É interessante notar que a não linearidade da função sigmoidal também é observada no comportamento dos neurônios humanos (WILLIS ET AL, 1990). Atuando sobre uma rede neural esta função fornece a ela a capacidade de representar relações não-lineares.

Uma rede neural é composta de vários elementos de processamento distribuídos em camadas. Cada EP envia seu sinal de saída a outros elementos. Assim, os elementos “ativam”

os outros de acordo com o seu peso ou força da conexão. Um esquema representativo pode ser visto pela Figura 3.3, onde visualizamos as conexões dos elementos de processamento em três camadas com vários elementos em cada uma.

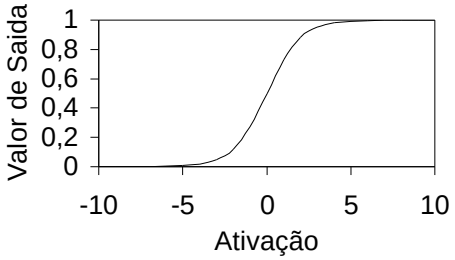


Figura 3.2 - Saída da Função Sigmoidal

As camadas intermediárias são também denominadas de *camadas escondidas*.

As redes neurais estão sendo utilizadas dentro da área de engenharia de processos, projetos e simulação, supervisão, controle e estimação, detecção de falhas, diagnósticos reais sobre o efeito de informações imprevistas e imprecisas, etc. Seus modelos são baseados em conhecimentos qualitativos (derivados de experiências), quantitativos (em termos de um modelo analítico do processo) ou uma mistura de ambos. Através desses modelos podemos obter soluções aceitáveis, existem no entanto, muitas situações que estão propensas a falhas devido às incertezas e as não-linearidades intrínsecas dos vários processos. Contudo, são para essas situações que as ANNs estão sendo projetadas, procurando formar uma base que forneça alternativas para a engenharia prática (DI MASSIMO ET AL,1992).

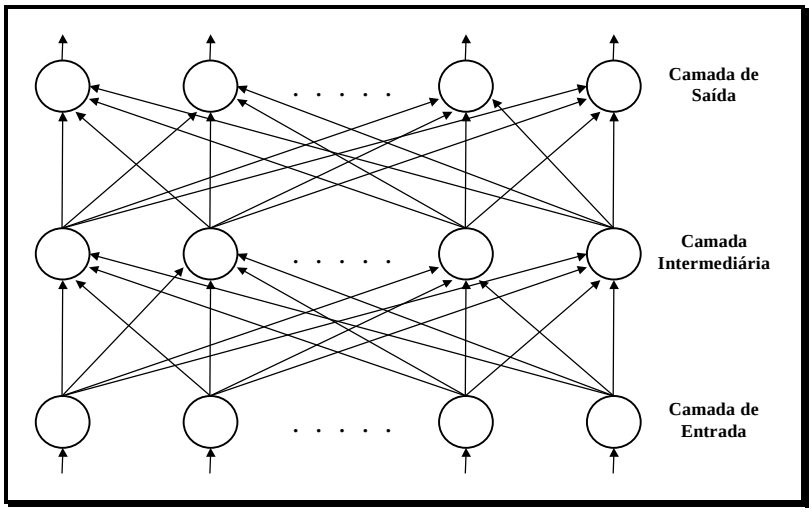


Figura 3.3 - Esquema Representativo das Conexões dos Neurônios de uma Rede Neural Artificial

3.2 *BackPropagation Network* (BPN)

A *BackPropagation Network* possui uma performance significativa no mapeamento de funções, destacando-se entre os diversos ANS[4] existentes. Esta rede, ilustrada genericamente na Figura 3.4, é projetada para trabalhar com multicamadas utilizando um modo particular de aprendizagem (FREEMAN E SKAPURA, 1992).

A operação da BPN para resolver problemas complexos, de um modo resumido, é iniciada com a rede *aprendendo* um conjunto de dados entrada-saída pré-definidos, utilizando um ciclo de aprendizagem. Após um conjunto de entrada ser aplicado como um estímulo na primeira camada da rede, este se propaga para as outras camadas até a camada de saída, gerando um resultado ou valores de saída. Estes resultados são comparados com os valores desejados e então o erro é computado para cada sinal de saída. Para a correção dos pesos o erro é transmitido *backward* (para trás, de volta) para cada nó das camadas intermediárias que contribuíram diretamente para a geração dos valores de saída. Entretanto, cada unidade intermediária recebe somente uma parte do erro total produzido, baseado na relativa

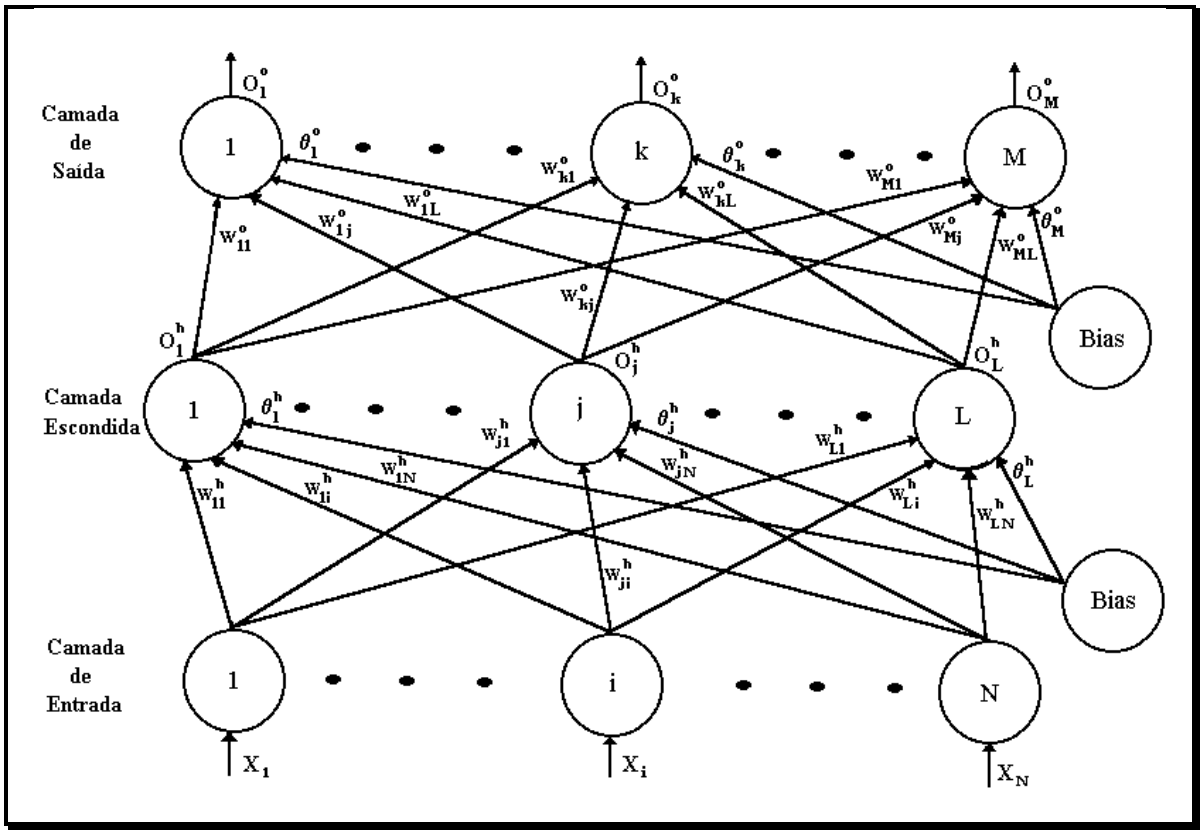


Figura 3.4 - Arquitetura das Três Camadas da BPN

[4] ANS - Artificial Neural System - Sistema Neural Artificial

contribuição que a unidade teve na construção do resultado na camada de saída. Este processo se repete, camada a camada, até que cada nó da rede tenha corrigido seus pesos.

As camadas da arquitetura BPN seguem a descrição geral das redes dada anteriormente. Existe no entanto, uma conexão fictícia, denominada *bias*, que possui valor unitário. Os pesos q_j^h e q_k^o , representam as forças dessas conexões e são tratados de maneira semelhante aos outros pesos.

3.2.1 Formulação Matemática

Apresentaremos as equações para uma BPN com três camadas por ser o modelo mais simples e porque este tipo foi implantado no *software* de controle.

Um vetor $X = (x_1, x_2, ..., x_N)$ aplicado na camada de entrada da rede e distribuído para a camada escondida, produz uma ativação em cada neurônio dessa camada descrita como:

$$a_j^h = \sum_{i=1}^{N} w_{ji}^h x_i + q_j^h \tag{3A}$$

onde $w_{j,i}^h$ é o peso da conexão da *i*-ésima unidade de entrada no neurônio *j* e q_j^h o peso da conexão *bias* desse neurônio. O sobrescrito *h* representa os valores referentes à camada escondida. O valor de saída dos neurônios da camada escondida é definido através da aplicação da função de ativação no neurônio, podendo ser linear ou não como visto anteriormente. Assumindo que a função de ativação da camada escondida é linear, ou seja, o valor de saída é igual à ativação temos:

$$o_j^h = f_j^h(a_j^h) = a_j^h \tag{3B}$$

Analogamente podemos deduzir que as correlações para os neurônios da camada de saída. A ativação dos neurônios é dada por:

$$a_k^o = \sum_{j=1}^{L} w_{kj}^o o_j^h + q_k^o \tag{3C}$$

onde *k* representa *o k-ésimo* neurônio de saída e o sobrescrito *o* representa valores da camada de saída. Aplicando-se uma função de ativação sigmoidal na camada de saída teremos:

$$o_k^o = f_k^o(a_k^o) = \frac{1}{1 + e^{-a_k^o}} \tag{3D}$$

Esse conjunto de equações nos fornece a saída da rede para um determinado valor de entrada.

A solução dessas equações necessita do conjunto de pesos das conexões, que envolve o treinamento da rede. Embora existam muitas técnicas, o método usado para a busca dos pesos ótimos neste trabalho é um processo iterativo que não depende de uma boa aproximação inicial.

3.3 Tipos de Treinamentos da Rede

O treinamento ou aprendizado da rede neural consiste em determinar os pesos ótimos para cada conexão dos neurônios, de modo que a rede possa fornecer valores de saída exatamente ou bem próximos aos valores esperados.

A disponibilidade dos dados para treino define o tipo de treinamento a ser utilizado, ou seja, treinamento *batelada*, *sequencial* ou *sequencial com janela*.

O *treinamento batelada* consiste em determinar os pesos ideais para um conjunto limitado de dados de treino. Esses pesos calculados permanecem inalterados durante a utilização da rede.

O *treinamento sequencial* atualiza os pesos a cada passo de utilização da rede. O treinamento é contínuo, sendo o conjunto de dados de treino ilimitado. Os pesos são atualizados toda vez que se utiliza a rede.

No *treinamento sequencial com janela* a rede é utilizada durante um certo período com atualização dos pesos de modo sequencial, após, interrompe-se a atualização mantendo-se os pesos constantes. Volta-se a corrigir os pesos, após algum tempo, quando a rede não consegue produzir valores satisfatórios. Isto pode ocorrer se a mesma estiver operando em condições diferentes às quais foi treinada.

O *software* de controle, utilizado neste trabalho, possui opções de implantação de todos os tipos de treinamentos, *batelada*, *sequencial* e *sequencial com janela*.

Neste trabalho adotamos o *treinamento sequencial* durante o método de treino *Steepest Descent*. Para o método de treino Decomposição em Valores Singulares utilizamos o treinamento *batelada* devido às características do método.

3.3.1 Treinamento *Steepest Descent*

O treinamento *Steepest Descent* é um algoritmo de cálculo baseado no gradiente de uma função erro. Como o gradiente informa a direção em que a função é crescente, esse método caminha em direção oposta ao gradiente (Figura 3.5), ou seja, para um ponto de

menor valor da função erro definida, assim, uma vez atingido o ponto de mínimo da função teremos os valores dos pesos que produzem os menores erros de saída.

Supondo que temos um conjunto $P = \{(X_1, Y_1), (X_2, Y_2), \ldots, (X_P, Y_P), \ldots, (X_R, Y_R)\}$ de pares de vetores, que são resultados do mapeamento de uma função $Y = f(X): X \in R^N, Y \in R^M$. Desejamos treinar a rede para que obtenha uma aproximação $O = g(X) \cong Y(X): X \in R^N, O \in R^M$.

O erro a ser minimizado pelo método *Steepest Descent* é definido como a soma do quadrado dos erros produzidos nas *M* saídas, ou seja:

$$E_p = \frac{1}{2} \sum_{k=1}^{M} d_{pk}^2$$

Sendo $d_{pk} = (y_{pk} - o_{pk}^o)$ o erro do *k*-ésimo elemento de saída da rede para o *p*-ésimo par de vetor de treinamento do conjunto *P*, onde y_{pk} é o valor correto de saída e o_{pk}^o o valor estimado pela rede.

O fator ½ na equação é definido por conveniência para o cálculo posterior da derivada da função.

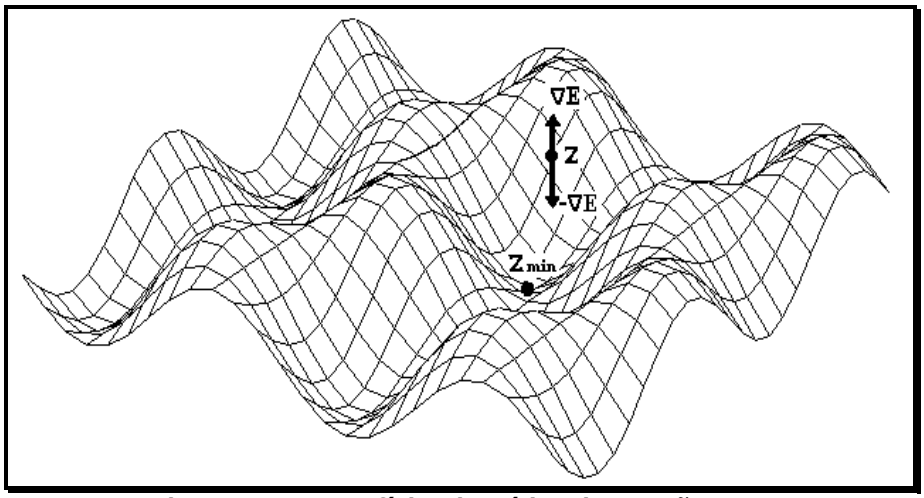


Figura 3.5 - Superfície Hipotética da Função Erro no Espaço dos Pesos

Como o método *Steepest Descent* utiliza o gradiente da função erro, devemos determinar esse gradiente em função dos pesos das camadas de saída e intermediária da rede, logo temos para a camada de saída;

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk}^o) \frac{\partial f_k^o}{\partial a_{pk}^o} \frac{\partial a_{pk}^o}{\partial w_{kj}^o}$$

e para uma função de ativação da camada de saída sigmoidal, temos:

$$\frac{\partial f_k^o(a_{pk}^o)}{\partial a_{pk}^o} = f_k^o(1-f_k^o) = o_{pk}^o(1-o_{pk}^o) \quad \text{e} \quad \frac{\partial a_{pk}^o}{\partial w_{kj}^o} = \left(\frac{\partial}{\partial w_{kj}^o}\sum_j^L w_{kj}^o o_{pj}^h + q_k^o\right) = o_{pj}^h$$

com as equações acima podemos definir a equação de correção dos pesos da camada de saída como sendo:

$$w_{kj}^o(k) = w_{kj}^o(k-1) - \eta\nabla E = w_{kj}^o(k-1) + \eta o_{pk}^o(1-o_{pk}^o)o_{pj}^h \qquad (3E)$$

onde o índice *k* representa o valor atual do peso calculado, *k-1* o valor anterior do peso e η, chamado de passo de aprendizagem, representa a razão com que o peso avança em direção do valor ótimo.

Podemos construir a equação de cálculo dos pesos da camada intermediária ou escondida de maneira análoga. Considerando a função de ativação linear e partindo da equação do erro, devemos determinar o gradiente da função em relação aos pesos da camada. A equação a seguir representa o gradiente da função erro em relação aos pesos da camada intermediária.

$$\frac{\partial E_p}{\partial w_{ji}^h} = -\sum_k^M (y_{pk} - o_{pk}^o)\frac{\partial o_{pk}^o}{\partial a_{pk}^o}\frac{\partial a_{pk}^o}{\partial o_{pj}^h}\frac{\partial o_{pj}^h}{\partial a_{pj}^h}\frac{\partial a_{pj}^h}{\partial w_{ji}^h} = x_{pi}\sum_k^M (y_{pk} - o_{pk}^o)w_{kj}^o$$

logo temos que;

$$w_{ji}^h(k) = w_{ji}^h(k-1) - \eta\nabla E = w_{ji}^h(k-1) + \eta x_{pi}\sum_k^M (y_{pk} - o_{pk}^o)w_{kj}^o \qquad (3F)$$

De posse dessas equações podemos determinar uma sequência de cálculo para o treinamento *Steepest Descent*.

Sequência de cálculo do treinamento *Steepest Descent* para a rede BPN:

1) Inicializar os pesos da rede com valores aleatórios.
2) Aplicar um vetor de entrada X_p nos neurônios da camada de entrada.
3) Utilizar a equação 3A e determinar a ativação da camada escondida.
4) Utilizar a equação 3B e determinar as saídas da camada escondida.
5) Determinar a ativação dos neurônios da camada de saída com a equação 3C.
6) Determinar os valores de saída da rede com a equação 3D.
7) Corrigir os pesos da camada de saída utilizando a equação 3E.
8) Corrigir os pesos da camada escondida utilizando a equação 3F.
9) Caso o valor do erro E_p seja aceitável, o processo é terminado senão voltar ao passo 2 e aplicar um novo vetor de entrada.

3.3.2 Treinamento Decomposição em Valores Singulares

Este tipo de treinamento visa acelerar o processo de treino utilizando funções polinomiais baseadas em funções de ativação não lineares e incorpora um mecanismo para a manipulação automática dos dados de entrada na forma mais apropriada para a rede.

Sabendo-se que a velocidade de treinamento é influenciada pela quantidade de neurônios existente na rede, utiliza-se uma técnica denominada *Análise do Componente Principal(ACP)* para avaliar a contribuição relativa das entradas sobre as variações de saída. A técnica ACP pode ser classificada como uma técnica de análise de dados multivariáveis, sendo um procedimento importante para a análise de conjuntos de dados complexos. Na realização de uma ACP, a análise de um grande número de variáveis pode ser reduzida à investigação de um pequeno conjunto transformado. Além disso, incorporando uma ACP na filosofia de treinamento da rede, é possível especificar o número mínimo de neurônios nas camadas intermediárias da rede.

Considerando um conjunto de *R* observações de *N* entradas da rede como uma matriz na forma $X = \{x_1, x_2, ..., x_j, ..., x_{N-1}, x_N\}$, onde o vetor x_j representa uma sequência das *R* observações da *j*-ésima variável. A matriz *X* pode então ser decomposta como:

$$X = U\Sigma^{1/2}V^T \tag{3G}$$

onde *U* é a matriz dos vetores característicos de XX^T com dimensão *RxR*, e *V* é a matriz de vetores característicos de X^TX com dimensão *NxN*. A matriz $\Sigma^{1/2}$ é diagonal, cujos elementos são as raízes quadradas positivas dos valores característicos $l_i (i = 1,...,R)$ de X^TX e são chamados *valores singulares*. As colunas *U* e *V* são denominadas *vetores singulares* de *X*, e apresentam as seguintes características:

$$V^TV = VV^T = I \qquad \text{e} \qquad U^TU = UU^T = I$$

A técnica de decomposição de matrizes descrita pela equação 3G é denominada Decomposição de Valores Singulares. Esta é uma das muitas técnicas para determinar os valores e vetores característicos de matrizes, cuja utilização baseia-se no fato de ser o procedimento numericamente mais robusto para realizar esta tarefa e também por ser aplicável a matrizes não quadradas, o que é útil na prática, uma vez que a quantidade de dados é geralmente maior que o número de variáveis em consideração(GOLUB E VAN LOAN, 1983)

Os componentes principais da matriz *X* são as colunas da matriz *P*, definida como:

$$P = XV = U\Sigma^{1/2} = \{p_1, p_2, ..., p_i, ..., p_{N-1}, p_N\} \tag{3H}$$

O *i*-ésimo componente principal definido pelo vetor p_i é uma soma ponderada de variáveis padronizadas, onde os pesos são definidos pelos elementos do *i*-ésimo vetor característico, v_i, isto é:

$$p_i = Xv_i = \sum_{j=1}^{N} v_{ij} x_i \qquad (i=1,...,N) \tag{3I}$$

A variância de cada componente principal é dada pelo correspondente valor característico,λ_i, da matriz de correlação de dados (MANLY,1986 ; RAWLINGS, 1988), ou seja;

$$\text{var}(p_i) = v_i^T X^T X v_i = l_i$$

Uma vez que a matriz correlação é simétrica, todos os seus valores característicos tem magnitude maior do que, ou igual a zero. Estes valores característicos podem ser organizados em ordem decrescente. Sendo assim, o primeiro componente principal, corresponde ao maior valor característico, explicará a maioria das variações nos dados fornecidos. O segundo componente principal explicará a causa do próximo nível de variação, e assim sucessivamente.

Pelo descarte dos componentes principais que não contribuem significativamente para a variação global dos dados, a dimensão do problema pode ser reduzida. Esta técnica pode ser utilizada como uma parte integrante do procedimento de treinamento de uma rede neural.

Uma BPN(item 3.2) com duas camadas geralmente é treinada utilizando uma rotina de minimização para obter o grau de aproximação necessário. Na metodologia aqui analisada, a determinação dos pesos nas duas camadas difere da determinação geralmente utilizada para a BPN. Os pesos das conexões entre a camada de entrada e a camada intermediária são determinados pelo resultado de uma ACP realizada nos dados de entrada do seguinte modo :

a) as variáveis escolhidas como entrada para a rede são agrupadas na matriz *X*.
b) a matriz *X* é transformada conforme a equação 3G
c) a matriz dos componentes principais *P* é obtida pela equação 3I
d) os pesos entre as camadas de entrada e intermediária são assumidos como os elementos do vetor característico

Os componentes principais resultantes são linearmente independentes, mesmo que as variáveis originais apresentem multicolinearidades. Desta forma a porcentagem de contribuição de cada componente principal em relação à variação global dos dados pode ser

determinada, e os componentes principais cuja contribuição é pequena, posteriormente, podem ser eliminados da análise. Esta eliminação de componentes principais de pequena importância, no presente contexto, representa a eliminação de neurônios da camada intermediária.

Sendo a função de ativação da camada intermediária uma função sigmoidal, podemos fazer uma expansão em série de Taylor, obtendo:

$$o_j^h = f_j^h(a_j^h) = \frac{1}{1+e^{-a_j^h}} = a_0 + a_1(a_j^h) + a_2(a_j^h)^2 + a_3(a_j^h)^3 + \ldots \qquad (3J)$$

Considerando que a saída da camada intermediária da rede possa ser definida como a equação 3J, podemos escrever a ativação da camada de saída como sendo:

$$a_k^o = \sum_{j=1}^{L} w_{kj}^o o_j^h = \sum_{j=1}^{L} w_{ij}^o \sum_{n=0}^{G} a_n (a_j^h)^n \qquad (3K)$$

onde G representa o grau da polinomial de aproximação da função sigmoidal. Entretanto a equação 3K pode ser rescrita como:

$$a_k^o = \sum_{j=1}^{L} \sum_{n=0}^{G} s_{nj} (a_j^h)^n \qquad (3L)$$

sendo s_{nj} os coeficientes da polinomial transformados que devem ser determinados, de forma a alcançar a exatidão desejada.

Para uma rede neural com um único valor de saída (k=1), podemos escrever para as R observações da matriz de entrada X, um vetor $Y = \left[y_1^1, y_1^2, \ldots, y_1^p, \ldots, y_1^{R-1}, y_1^R\right]^T$, onde y_1^p, representa a saída da p-ésima observação. Sendo a função de ativação da camada de saída sigmoidal, podemos escrever;

$$y_1^p = \frac{1}{1+e^{-(a_1^o)^p}} \therefore (a_1^o)^p = -\ln(1-\frac{1}{y_1^p}) = \ln\left(\frac{y_1^p}{y_1^p - 1}\right) \qquad (p=1,\ldots,R) \qquad (3M)$$

Durante o treinamento da rede conhecemos a matriz X e o vetor Y, e deste modo podemos determinar a ativação da camada de saída pela equação 3M.

Utilizando essas equações podemos construir uma sequência de cálculo para o treinamento Decomposição em Valores Singulares como sendo;

1) montar a matriz de observações X
2) determinar U, V, a partir de X
3) determinar a matriz P (equação 3H)
4) determinar os componentes principais p_i (equação 3I)

5) realizar a ACP eliminando os componentes principais que possuem pequena contribuição, isto é, determinar o número de neurônios da camada intermediária
6) determinar a ativação da camada intermediária (equação 3H)
7) determinar a ativação da camada de saída para as *R* observações (equação 3M)
8) determinar os coeficientes da polinomial transformados s_{nj} (equação 3L)

3.4 Considerações práticas

As seq*u*ências de cálculo mostradas nos treinamentos parecem muito simples de ser implantadas, mas na prática existem vários problemas que devem ser contornados. A seguir procuramos esclarecer essas dificuldades para cada tipo de treinamento utilizado e mostrando as soluções aplicadas neste trabalho.

3.4.1. Treinamento *Steepest Descent*

O primeiro item da sequência de cálculo do treinamento, onde temos que inicializar os pesos com valores aleatórios, induz a várias indagações. As grandezas desses valores? Podem ser valores negativos e positivos? Os valores podem ser todos iguais? Deve existir alguma relação entre eles? Perguntas como estas devem ser estudadas e resolvidas para cada caso que a rede se propõe a aprender. No aprendizado do processo estudado, valores iniciais de grandeza 1 ou superior, desestabilizaram o sistema de rede, gerando erros de *overflow*[5]. O mesmo aconteceu para valores iniciais -1 ou inferiores. Portanto, a inicialização dos pesos foi realizada com valores aleatórios entre 0 e 10^{-3} .

Outra dificuldade se refere às características do conjunto de dados a ser fornecido para o treinamento, qual a quantidade de dados para treino? Os dados devem ser seq*u*enciais?. A princípio não existe um quantidade definida de dados de treino, dependendo da natureza dos dados e das informações que a rede poderá absorver, haverá a necessidade de um conjunto maior de dados. Observa-se também que as características do sistema em treino influenciam na quantidade de dados necessários para o treinamento. Devido a natureza do aprendizado da rede, não há necessidade de que os dados de entrada estejam em sequência.

Quantos neurônios devem ser utilizados em cada camada? Esta pergunta é outro problema que deve ser considerado para cada sistema a ser estudado. O que se encontra geralmente é uma quantidade de neurônios diferente para cada trabalho publicado ou trabalho realizado. Alguns autores tentaram estabelecer um parâmetro preciso ou uma regra geral para

[5] Geração de um valor fora do range admissível pelo computador

determinação exata da quantidade de neurônios da rede. O que se faz geralmente é iniciar com uma certa quantidade (10 a 20 em cada camada) de neurônios e posteriormente otimiza-se a rede baseando-se nos resultados por ela oferecido. Sabemos que um número elevado de neurônios (nós) na rede proporciona um *esforço computacional muito grande*[6].

A utilização ou não da conexão fictícia *bias* é outra decisão a ser tomada. Dependendo do sistema estudado deve-se verificar os resultados com e sem a conexão para decidir sobre sua implantação. Não é uma decisão geral, e sim, uma decisão individual para cada sistema.

O passo de aprendizagem dos pesos η também é decidido por tentativas, iniciando-se com valores menores e aumentando-se até descobrir o melhor valor no sistema que estamos treinando. A maioria das redes trabalham com passo de aprendizado constante, mas nada impede que ele se ajuste durante o processo de aprendizado, diminuindo ou aumentando, conforme a necessidade.

Dependendo do passo de aprendizagem dos pesos ou do conjunto de dados de treinamento, podemos não encontrar os valores ótimos dos pesos, o ***mínimo global*** da função erro (ponto *Zmin*), como podemos observar pela Figura 3.6. Pode-se atingir um ***mínimo local*** (Z_1 ou Z_2), onde teremos valores mínimos do erro para uma determinada região de trabalho da rede.

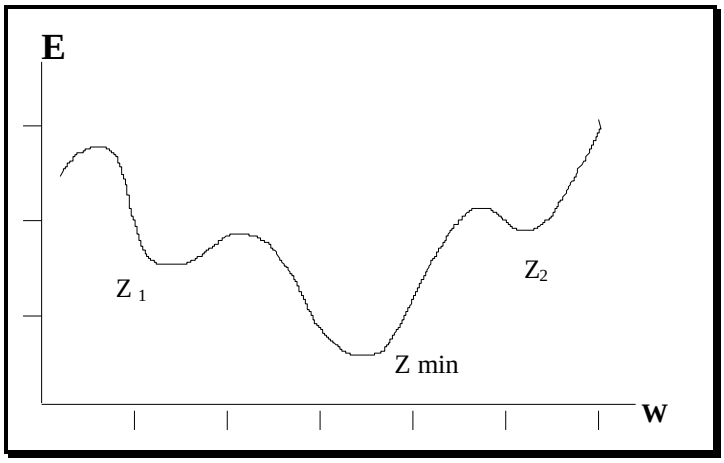


Figura 3.6 - Seção Transversal de uma Superfície Hipotética da Função Erro no Espaço dos Pesos

3.4.2. Treinamento Decomposição em Valores Singulares

A montagem da matriz de observações oferece a primeira dificuldade na sequência de cálculo desse treinamento porque devemos definir a quantidade de observações (R) e aquantidade de neurônios da camada de entrada (N). Cada função a ser aproximada por uma

[6] Aumento significativo na quantidade de operações lógicas e matemáticas realizadas pelo computador provocando um tempo maior na busca do resultado

rede neural deverá possuir valores diferentes de R e N, portanto, os dois dados devem ser determinados através de tentativas observando o aprendizado da rede para diversos valores, procurando assim, valores ótimos para esses parâmetros.

A decomposição da matriz *X*, segundo item da sequência de cálculo, pode ser realizada por qualquer rotina de decomposição. Neste trabalho utilizamos rotinas da biblioteca IMSL.

Determinar os componentes principais com valores representativos, realizar a ACP, oferece outra dificuldade, pois devemos definir o que é um valor representativo, qual a contribuição relativa mínima a ser utilizada. Neste trabalho eliminamos valores representativos com contribuição relativa inferior a 1%.

Finalmente os coeficientes da polinomial transformados s_{nj} (equação 3L), para serem determinados, devemos definir o grau da polinomial de aproximação(*G*) da função sigmoidal. Este é outro parâmetro estimado através de tentativas, pois deverão existir valores ótimos diferentes para cada caso estudado. Devemos observar que o acréscimo de 1 grau na polinomial representa *L* coeficientes a mais a serem determinados, onde *L* é o número de neurônios da camada intermediária, determinado na ACP.

4. Aquecedor de ar, o módulo de testes

O processo de aquecimento utilizado para testes e controle está representado pela Figura 4.1. Foi construído de modo a possibilitar aplicação de perturbações do tipo degrau na vazão de ar utilizando a válvula de *by-pass* ou abrindo e fechando a porta do aquecedor.

A operação do sistema é muito simples, o ar injetado pelo soprador é enviado ao aquecedor, dependendo da posição da válvula de *by-pass*, uma parte dessa vazão é desviada para a atmosfera, o transmissor de temperatura envia um sinal elétrico para o conversor analógico/digital que após transformá-lo envia-o para o computador. O computador envia um sinal ao conversor digital/analógico que atua sobre o controlador de tensão, que ajusta a tensão no conjunto de resistências, procurando assim, manter a temperatura de saída do ar no valor desejado.

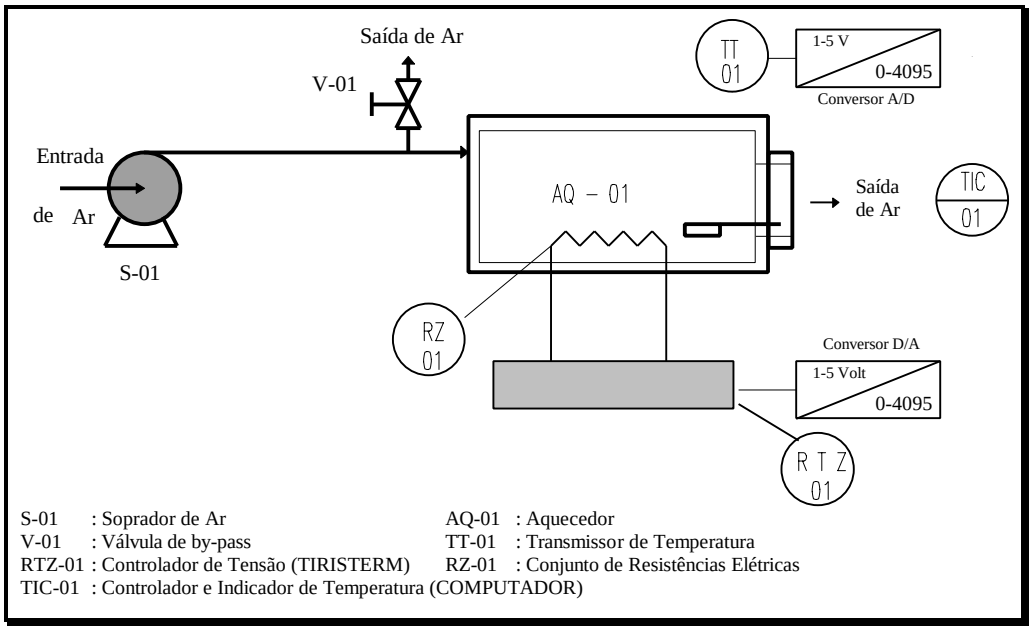


Figura 4.1 - Esquema do Módulo de Teste e Controle

4.1 Características do sistema

A caracterização do sistema foi realizada através da resposta do processo a perturbações do tipo degrau. Determinamos as constantes do processo para perturbações na tensão da resistência elétrica e observamos o seu comportamento sob a ação de uma perturbação externa; a variação da vazão de ar no aquecedor.

Como modelo consideramos um sistema contínuo de primeira ordem com tempo morto, descrito pela equação na forma;

$$t\,\overset{o}{y}(t) + y(t) = K_p u(t - t_m) \qquad 4A$$

onde *y* é a temperatura de saída do processo, variável controlada, e *u* a perturbação aplicada na tensão das resistências elétricas, variável manipulada.

A solução da equação diferencial(4A), para uma perturbação degrau aplicada no tempo *t=0* e de amplitude A, é descrita pela equação;

$$y(t) = \begin{cases} y(0) & t \le t_m \\ y(0) + AK_p(1 - e^{-(t-t_m)/t}) & t > t_m \end{cases} \qquad 4B$$

Considerando que o processo alcance o valor final com um tempo infinito (t_{final}=+∞), temos:

$$y(t_{final}) = y(0) + AK_p \;\therefore\; K_p = \frac{\left(y(t_{final}) - y(0)\right)}{A} \qquad 4C$$

Através da equação 4B podemos escrever;

$$\ln\left(1 - \frac{\left(y(t) - y(0)\right)}{AK_p}\right) = -\frac{1}{t}(t - t_m) \qquad t > t_m \qquad 4D$$

Construindo o gráfico *ln( )* x *t* para a equação 4D temos uma reta com a inclinação dada por -1/τ.

Com os dados dos testes, através da equação 4C determinamos K_p. Com a regressão linear dos dados fornecidos pela equação 4D, calculamos o valor da constante τ para as perturbações aplicadas na variável manipulada. O tempo morto foi determinado através da observação dos dados obtidos nos testes.

O testes iniciaram sempre com o processo estabilizado no *set-point* de 200 ºC, para então aplicarmos a perturbação degrau. O ajuste inicial no valor do *set-point* e a aplicação da perturbação, foram realizados de modo manual utilizando um voltímetro. Podemos observar, através das Figuras 4.2 e 4.3, uma oscilação na variável manipulada antes da aplicação da perturbação, devido ao ajuste manual.

A Figura 4.2 representa um degrau de +0,48V, com tempo de amostragem T_s=15s, aplicado após 50 minutos do início do teste, que levou a temperatura do processo, depois de 3 h 15 minutos, ao valor de 334 ºC. Obtivemos para τ o valor de 2400s, para K_p o valor de 1,273V/V e para t_m o valor de 180s.

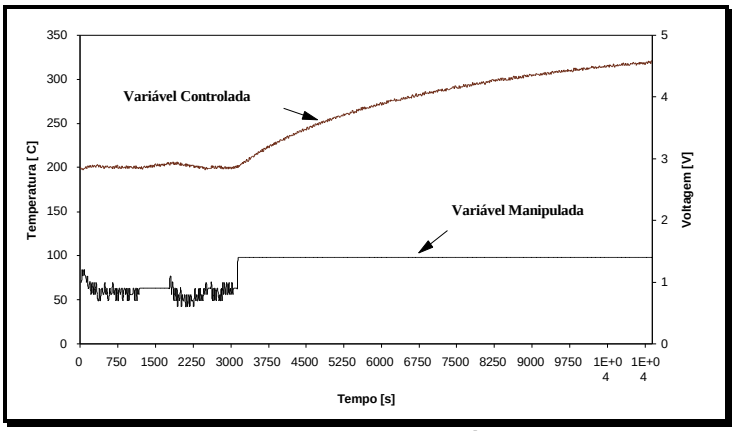


Figura 4.2 - Degrau +0,48V na Variável Manipulada do Módulo de Testes

A segunda perturbação, exibida pela Figura 4.3, corresponde a um degrau de -0,32V, com tempo de amostragem T_s=15s, aplicado após 45 minutos do início do teste. Depois de 2 h e 15 minutos a temperatura do processo baixou para 148,7 ºC. Neste caso obtivemos para τ o valor de 2087s, para K_p o valor de 0,902V/V e para t_m o valor de 135s.

Analisando os resultados observamos uma característica importante no processo. O tempo de resposta a um degrau positivo (aquecimento) é menor que o tempo de resposta a um degrau negativo (resfriamento). Esse fato está relacionado com as resistências de transferência de calor envolvidas nas duas situações.

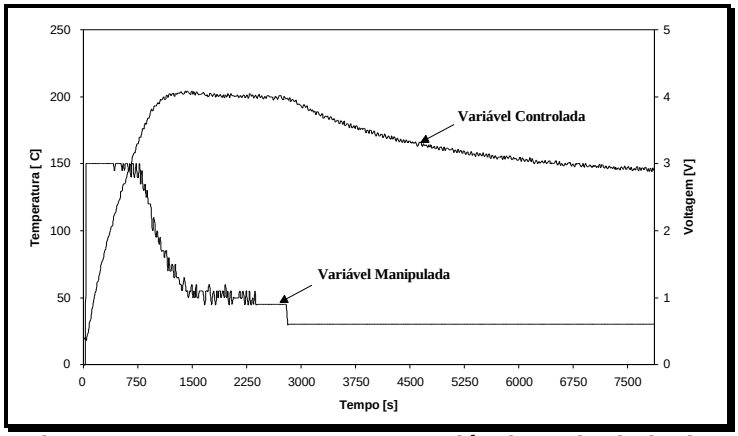


Figura 4.3 - Degrau -0,32V na Variável Manipulada do Módulo de Testes

Através da Figura 4.4, temos um corte da parede do aquecedor onde *Tr* representa a temperatura da resistência elétrica, *Tw* a temperatura da parede, e *Ta* a temperatura do ar no interior do aquecedor.

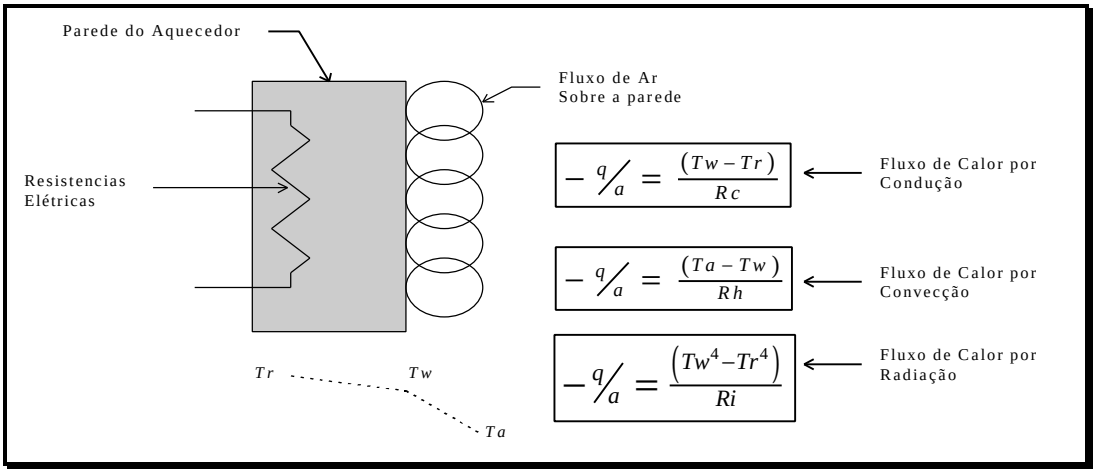


Figura 4.4 - Representação da Transferência de Calor no Interior do Aquecedor

Sabendo que a taxa de transferência de calor para o ar está diretamente relacionada com a temperatura da parede *Tw*, podemos considerar que a velocidade de variação da temperatura do ar de saída do aquecedor é proporcional a *Tw*, ou seja:

$$\frac{\partial(T_{saida})}{\partial t} \propto \frac{\partial Tw}{\partial t}$$

Verificamos que no processo de aquecimento ocorre a elevação da temperatura *Tw,* através da transferência de calor por *condução e radiação* das resistências elétricas para a parede. No resfriamento, a temperatura *Tw* diminui através da transferência de calor por *convecção* da parede para o ar.

Sendo as resistências por condução (*Rc)* e radiação (*Ri*) menores que a resistência por convecção (*Rh*), o fluxo de calor é maior no aquecimento, propiciando a *Tw* uma velocidade maior no aquecimento.

$$\left\| \frac{\partial Tw}{\partial t} \right\|_{aq} \rangle \left\| \frac{\partial Tw}{\partial t} \right\|_{resf}$$

portanto;

$$\left\| \frac{\partial T_{saida}}{\partial t} \right\|_{aq} \rangle \left\| \frac{\partial T_{saida}}{\partial t} \right\|_{resf}$$

o que implica no tempo de resposta ao aquecimento ser menor que o de resfriamento.

Outro teste de caracterização do sistema foi um degrau na vazão de ar de entrada do aquecedor (Figura 4.5), seguindo o mesmo padrão dos testes anteriores, após 50 minutos aplicamos um degrau na vazão de ar abrindo a porta de saída de ar do aquecedor, diminuindo assim a resistência à passagem do fluxo de ar e consequentemente aumentando a vazão. Observamos que a temperatura de saída do processo estabilizou a 125 ºC após 36 minutos.

Este teste foi realizado com a intenção de se comparar os dados obtidos, com os dados que posteriormente serão fornecidos com os controladores PID e RN atuando no processo.

Os testes apresentados foram realizados mais de uma vez procurando obter dados confiáveis. Esses dados estiveram bem próximos, de um teste para outro, apresentando uma boa repetibilidade.

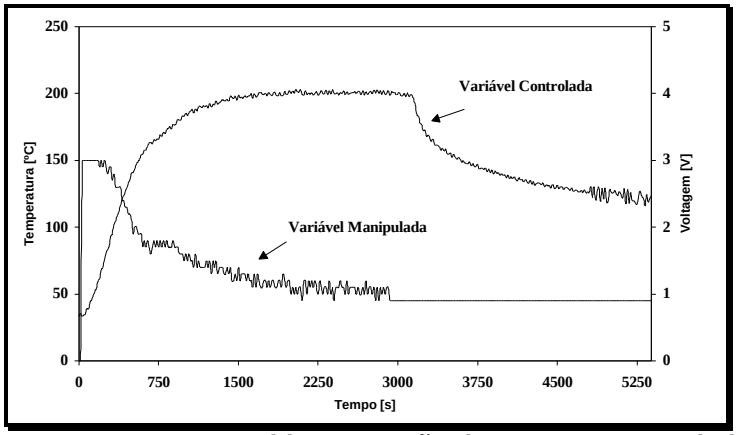


Figura 4.5 - Degrau Positivo na Vazão de Ar Sem Controlador no Módulo de Testes

4.2 Modelo para Simulação

Utilizando um sistema de 1ª ordem invariante no tempo, implantamos no *software* de controle um modelo para simulação do módulo de testes. O sistema de 1ª ordem utilizado obedece a equação:

$$t\,\overset{o}{y}(t) + y(t) = K_p u(t - t_m)$$

com a função de transferência:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_p e^{-t_m s}}{ts - 1}\,.$$

Acoplando um *Holder de ordem zero*[7] na variável de entrada conforme a Figura 4.6 temos que a função pulso de transferência para o sistema é descrita como:

$$H(z) = \frac{Y(z)}{U(z)} = \mathrm{Z}\{\mathrm{ZOH(s).G(s)}\} = \frac{(z-1)}{z}\mathrm{Z}\left\{\frac{G(s)}{s}\right\} = \frac{(z-1)}{z}\mathrm{Z}\left\{\frac{K_p e^{-t_m s}}{s(ts-1)}\right\}$$

logo;

[7] Transforma uma amostragem discreta de pontos em uma representação contínua

$$H(z) = \frac{(z-1)}{z} K_p \; Z\left\{ e^{-t_m s} \frac{1}{s(ts-1)} \right\}$$

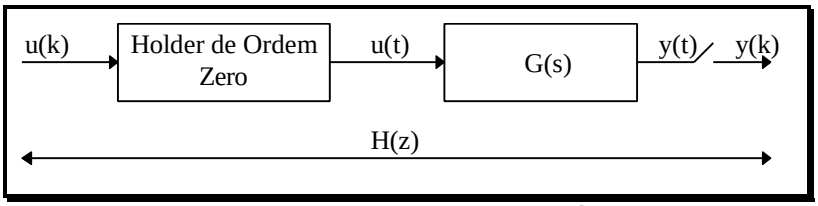


Figura 4.6 - Amostragem de Sistemas Contínuos no Tempo

definindo;

a função $F(s) = \dfrac{1}{s(\tau s + 1)}$

o tempo morto $t_m = n \,.T_s$ onde T_s = tempo de amostragem

n = número de intervalos de amostragem

e sabendo que $Z\{e^{-nTs} F(s)\} = z^{-n} F(z)$ e $Z\{ f(k-m)\} = z^{-m} F(z)$ temos;

$$H(z) = \frac{(z-1)}{z} K_P z^{-n} \frac{z(1-e^{-T_s/t})}{(z-1)(z-e^{-T_s/t})} = K_P z^{-n} \frac{1-e^{-T_s/t}}{z-e^{-T_s/t}} = \frac{a_1 z^{-n}}{z-a_2}$$

onde $a_1 = K_P(1-e^{-T_s/t})$ e $a_2 = e^{-T_s/t}$ então $zY(z) = a_1 z^{-n} U(z) + a_2 Y(z)$ que produz;

$$y(k+1) = a_1 u(k-n) + a_2 y(k) \quad \text{para} \quad y(k) = \begin{cases} 0 & k \le n \\ y(k) & k > n \end{cases} \qquad (4E)$$

esse modelo no domínio do tempo na forma recursiva foi implantado no *software* de controle, com os parâmetros calculados para o degrau positivo (Figura 4.2). Escolhemos o degrau positivo, porque a verificação da performance dos controladores no módulo de testes, seria semelhante ao teste da perturbação na vazão de ar (Figura 4.5), onde o controlador, necessita aplicar um degrau positivo na variável manipulada para manter o módulo no *set-point*.

5. O Controlador RN Não-linear

Existem várias maneiras de se fazer o controle de processos utilizando redes neurais artificiais. A diferença básica está no modo de treinamento da rede, e posteriormente como a rede gera o valor a ser implantado no processo.

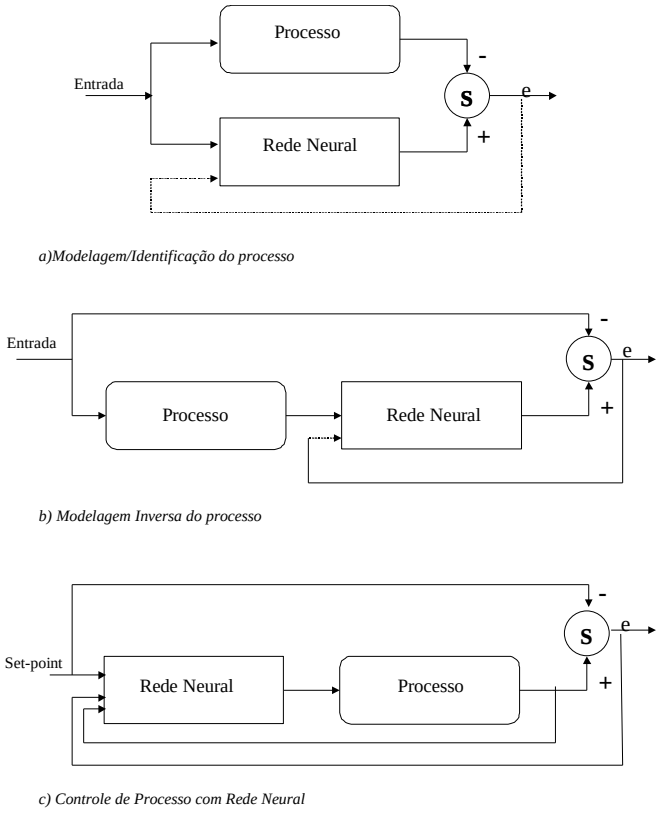


Figura 5.1 - Utilização de ANN em Controle de Processos

Na Figura 5.1a temos uma rede neural fazendo a identificação ou modelagem de um processo, notemos que a rede está procurando determinar o valor de saída do processo utilizando os mesmos valores de entrada. O erro produzido pela rede é determinado e enviado de volta ao sistema neural, para que sejam feitas as correções de seus pesos e, deste modo, para cada valor de entrada a rede corrige seus pesos e consequentemente minimiza o erro de saída.

Na modelagem inversa, representada pela Figura 5.1b, temos a rede neural sendo treinada para aprender o processo inverso. O treinamento é realizado, alimentando a rede com os valores de saída do processo de modo que a mesma produza os valores de entrada. Os pesos são corrigidos através do erro produzido como mostrado na figura. Este tipo de modelagem é interessante para uso no controle, porque estando a rede treinada e alimentando o valor do *set-point* ela deveria fornecer o valor a ser implantado no processo. Existe muito receio em se utilizar a modelagem inversa devido a geração de regiões de instabilidade de controle.

A representação de controle da Figura 5.1c nos mostra como a implantação do controle através de redes neurais pode ser realizada. Podemos observar que a rede produz o valor a ser implantado no processo utilizando o valor do *set-point* e informações de saída do processo. A

saída do processo comparada com o *set-point* produz o erro que é informado ao sistema neural para correção dos pesos.

Neste trabalho realizamos o treinamento da rede de acordo com o mostrado na Figura 5.1a. O controle foi implementado conforme o modelo da Figura 5.2, utilizando um algoritmo de cálculo do valor da variável manipulada baseado nas equações das redes neurais, de acordo com cada tipo de treinamento implantado (Capítulo 3). Com o valor do *set-point* determinamos o valor da variável manipulada *u* , a qual é enviada ao processo e à rede. Sabendo o valor da variável desejada *y* e o valor estimado pela rede *o* determinamos o erro *e*, utilizado para a correção dos pesos da rede.

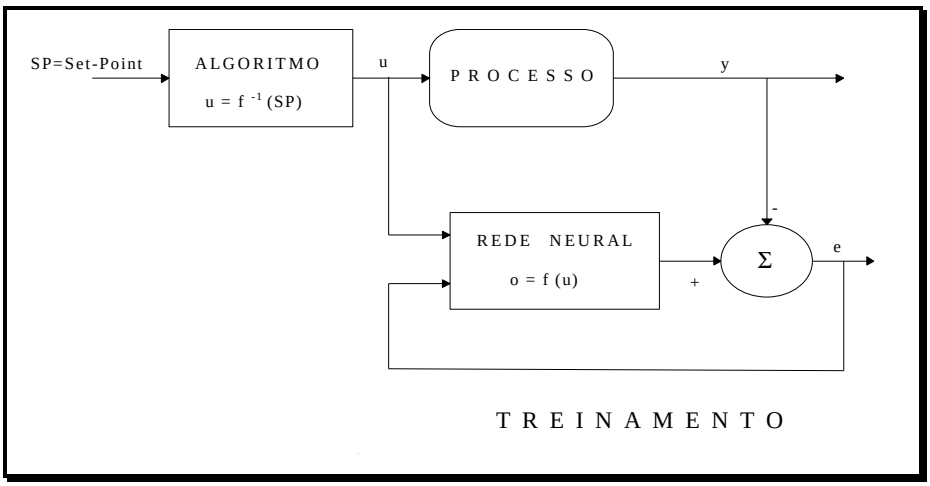


Figura 5.2 - Diagrama de Blocos do Controle RN Não-linear

5.1 Algoritmo de Cálculo da Variável Manipulada

Estando a RN treinada espera-se que uma vez fornecido um valor para a rede esta reproduza o resultado do processo, ou seja, a rede é capaz de prever a resposta do processo a um determinado valor de entrada. Como, o interesse é que a saída do processo se mantenha no *set-point*, o que precisamos descobrir é qual o valor de entrada na rede que produz na saída o *set-point*. Na verdade, o que estamos querendo descobrir é a função inversa da rede; têm-se o valor de saída e precisa-se do valor de entrada.

Para o treinamento *Steepest Descent*, utilizando as equações da rede, determinamos o valor de entrada da rede x_p que produz como saída o valor do *set-point*. Utilizamos uma RN com função de ativação linear na camada escondida e uma função de ativação sigmoidal na camada de saída, portanto, uma RN Não-linear.

Sabendo que:

$$o_k^o = \frac{1}{(1+e^{-a_k^o})} \therefore a_k^o = \ln\left(\frac{o_k^o}{1-o_k^o}\right) \tag{5A}$$

como;

$$a_k^o = \sum_j^L w_{kj}^o o_j^h + q_k^o = \sum_j^L w_{kj}^o \left(\sum_i^N w_{ji}^h x_i + q_j^h \right) + q_k^o$$

ou seja,

$$a_k^o = \sum_j^L \sum_i^N w_{ji}^h w_{kj}^o x_i + \sum_j^L w_{kj}^o q_j^h + q_k^o$$

para $i=p$ onde p representa o neurônio da camada de entrada, que recebe o valor da variável manipulada a ser implantado no processo, temos;

$$a_k^o = \sum_j^L w_{kj}^o w_{jp}^h x_p + \sum_j^L \sum_{\substack{i \\ i \# p}}^N w_{ji}^h w_{kj}^o x_i + \sum_j^L w_{kj}^o q_j^h + q_k^o$$

definindo

$$S_k = \sum_j^L \sum_{\substack{i \\ i \neq p}}^N w_{ji}^h w_{kj}^o x_i + \sum_j^L w_{kj}^o q_j^h + q_k^o \tag{5B}$$

podemos dizer que o valor de entrada da rede que produz na saída o valor do *set-point* é dado por:

$$x_p = \frac{a_k^o - S_k}{\sum_j^L w_{kj}^o w_{jp}^h} \tag{5C}$$

Utilizando o conjunto de equações 5A, 5B e 5C e considerando o_k^o = *set-point* determinamos o valor a ser implantado no processo $u=x_p$.

A equação 5A é descontínua para valores de o_k^o iguais a 0 e 1, porque representam os extremos da função sigmoidal (Figura 3.2). Logo, o valor do *set-point* do processo controlado, não pode estar situado nos extremos da faixa de trabalho. Esse problema é facilmente contornado por meio de um escalonamento apropriado.

No treinamento Decomposição em Valores Singulares utilizamos as equações definidas no item 3.3.2., para determinar o valor de entrada na rede que produz como saída o valor do *set-point.*

A equação 3H define a matriz *P* como sendo, *P=XV*, logo podemos dizer que:

$$P_{ij} = \sum_{k}^{N} x_{ik} v_{kj} \quad (i=1,...,R\ ;\ j=1,...,L)$$

para *i=q*, onde *q* representa a linha da matriz *P* que recebe a observação do instante atual, temos:

$$P_{qj} = \sum_{k}^{N} x_{qk} v_{kj} \qquad (j=1,...,L)$$

O elemento P_{qj} representa a ativação da camada escondida e portanto podemos dizer que $P_{qj} = a_j^h$. Utilizando essa definição, a equação 3L e considerando 1 neurônio na camada de saída, temos:

$$a_1^o = \sum_{j}^{L} \sum_{n=0}^{G} s_{nj} (a_j^h)^n = \sum_{j}^{L} \sum_{n=0}^{G} s_{nj} (\sum_{k}^{N} x_{qk} v_{kj})^n$$

para *k=p*, onde *p* representa o neurônio da camada de entrada que recebe o valor da variável manipulada a ser implantado no processo, definimos:

$$a_1^o - \sum_{j}^{L} \sum_{n=0}^{G} s_{nj} (x_{qp} v_{pq} + \sum_{\substack{k \\ k \# p}}^{N} x_{qk} v_{kj})^n = 0 \qquad (5D)$$

Para determinar o valor de x_{qp} que satisfaça a equação 5D necessitamos de um processo iterativo de cálculo. Neste trabalho foram testados vários métodos de cálculos, sendo que, o método do ajuste de uma função monotônica (VELOSO, 1985) apresentou melhores resultados (item 8.2). Este método possui as seguintes características:

- aproxima a função $f(x)$ através de $y = \dfrac{x - c_1}{c_2 x + c_3}$
- utiliza três pontos de partida (x_1,y_1),(x_2,y_2) e (x_3,y_3)
- determina o novo valor de *x* como sendo:

$$x_{novo} = \frac{x_1 y_2 y_3 (x_2 - x_3) - x_2 y_1 y_3 (x_1 - x_3) + x_3 y_1 y_2 (x_1 - x_2)}{y_2 y_3 (x_2 - x_3) - y_1 y_3 (x_1 - x_3) + y_1 y_2 (x_1 - x_2)} \qquad (5E)$$

Utilizando o conjunto de equações 5A,5D e 5E e considerando o_k^o o valor do *set-point*, determinamos o valor a ser implantado na rede $u=x_{qp}$.

5.1.1 Implantação do Valor da Variável Manipulada

Suspeitando que a implantação direta do valor estimado pelos algoritmos do item 5.1, poderia causar uma ação muito drástica do controlador, ou seja, haveria uma oscilação da variável manipulada muito grande e que poderia desestabilizar o controlador RN, procuramos

implantar de modo suave esse valor utilizando o esquema representado pela Figura 5.3, onde tem-se *n* como o número de intervalos de amostragens que o processo deverá demorar para atingir o *set-point*.

O valor de *n* para o processo estudado neste trabalho, foi determinado através de vários testes simulados observando o comportamento do controlador RN. Um valor *n*=50 foi encontrado como ótimo para o treinamento *Steepest Descent*, sendo que para o treinamento Decomposição em Valores Singulares o valor de n=5 apresentou melhores resultados. Utilizamos como aproximação inicial uma relação entre a constante de tempo e o intervalo de amostragem, $n = \frac{t}{Ts} = 25$, aumentando ou diminuindo gradativamente e observando o comportamento do controlador.

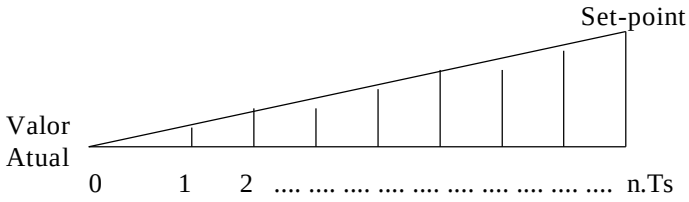


Figura 5.3 - Diagrama de Implantação do Valor Correto pela RN

Cada processo deverá possuir um valor de *n* diferente, não sendo necessário o conhecimento da constante de tempo para implementar o algoritmo.

5.2 Distribuição dos Dados na Entrada

No treinamento *Steepest Descent* os dados de entrada do controlador RN podem ser representados pelo vetor:

$$X = \left(u(k), u(k-1), u(k-2), \ldots, u(k-j-1), y(k), y(k-1), y(k-2), \ldots, y(k-(N-j-1)\right)$$

onde *u* representa a variável manipulada e *y* a variável controlada, o índice *N* representa a quantidade total de neurônios na camada de entrada, *j* a quantidade de neurônios de entrada que recebe valores da variável manipulada, sendo *N-j* a quantidade de neurônios que recebe o valor da variável controlada. O índice *k* representa o tempo de amostragem atual, *k-1* um tempo de amostragem anterior, *k-2* dois tempos de amostragens anteriores, etc.

A alimentação composta de valores anteriores das variáveis manipuladas e controladas, procura fornecer à rede possibilidades de aprender o tempo morto do processo.

Utilizando essa distribuição de dados de entrada consideramos, na Equação 5C, *p*=1, ou seja, o valor atual da variável manipulada é introduzido no primeiro neurônio da camada de entrada.

Para o treinamento Decomposição em Valores Singulares utilizamos uma distribuição dos dados de entrada idêntica à do treinamento *Steepest Descent*, sendo que, a matriz de observações ficou distribuída da seguinte forma:

$$X = \begin{Bmatrix} u^1(k) & u^1(k-1) & \cdots & u^1(k-j-1) & y^1(k) & \cdots & y^1(k-N+j+2) & y^1(k-N+j+1)) \\ u^2(k-1) & u^2(k-2) & \cdots & u^2(k-1-j-1) & y^2(k-1) & \cdots & y^2(k-N+j+1)) & y^2(k-N+j) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u^{R-1}(k-R+2) & u^{R-1}(k-R+1) & \cdots & u^{R-1}(k-R-j+1) & y^{R-1}(k-R+2) & \cdots & y^{R-1}(k-R-N+j+4) & y^{R-1}(k-R-N+j+3) \\ u^R(k-R+1) & u^R(k-R) & \cdots & u^R(k-R-j) & y^R(k-R+1) & \cdots & y^R(k-R-N+j+3) & y^R(k-R-N+j+2) \end{Bmatrix}$$

A primeira linha da matriz X representa a observação no instante atual, ou seja, a entrada da rede no instante atual. A segunda linha representa a entrada da rede um instante anterior, a terceira linha dois instantes anteriores e assim sucessivamente até *R-1* instantes anteriores.

A representação *k, j* e *N* obedecem o mesmo significado utilizado anteriormente para a formação da distribuição de entrada do treinamento *Steepest Descent* e o índice *R* representa o número da observações.

Através dessa distribuição de dados e com a equação 5D, consideramos p=q=1, ou seja, o valor atual da variável manipulada é introduzido no primeiro neurônio da camada de entrada e na primeira linha da matriz *X*.

6. O Software RTX

O programa RTX[8] é um sistema desenvolvido para a simulação e controle de processos. Utiliza a plataforma de um processador 80486 ou superior, operando sob o sistema MS-DOS[9] Versão 3.3 ou posterior, requer o mínimo de 4 MBytes de memória RAM, monitor padrão CGA ou superior (recomendável no mínimo VGA color), espaço em disco: 2 MBytes para o *software*, 360 KBytes para cada arquivo de dados na simulação e 120 KBytes para cada arquivo diário de dados no controle.

O sistema é composto por um módulo gerenciador, denominado Z4, que organiza as informações e realiza as chamadas das rotinas de execução contínua, através do teclado e pilha de execução. A lógica de execução do módulo gerenciador pode ser vista através da Figura 6.1.

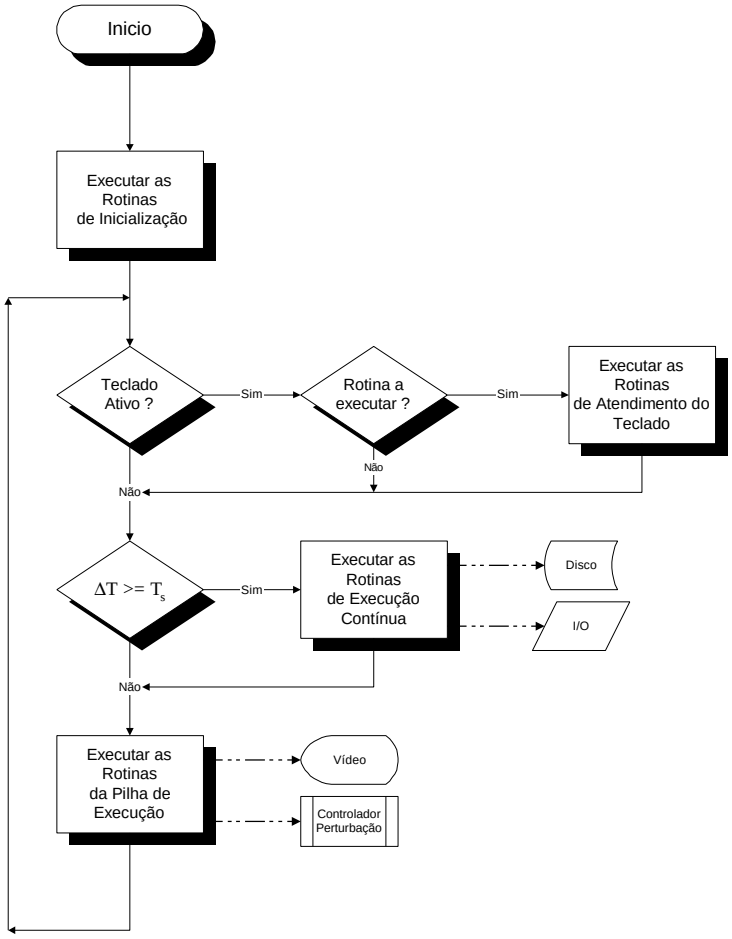


Figura 6.1 - Fluxograma da Lógica de Execução das Rotinas do Módulo Gerenciador Z4

O programa é um laço infinito, com dois conjuntos de tarefas, cuja execução é condicional. O *primeiro grupo* é constituído por tarefas que são executadas caso uma determinada tecla seja pressionada. O *segundo grupo* é constituído por tarefas rotineiras, cuja execução é disparada por um gatilho temporizado. Sendo o tempo de amostragem o gatilho para as rotinas de execução contínua e a frequência de execução para as rotinas da

[8] RTX - Real Time eXecution - Execução em tempo real, designação que se dá a um sistema que responde a um estímulo gerado externamente dentro de um finito e especifícado atraso de tempo.
[9] MS-DOS - MicroSoft Disk Operating System - Sistema operacional para microcomputadores padrão IBM desenvolvido pela MicroSoft Corporation

pilha de execução.

Dentre as rotinas de execução contínua temos as rotinas de armazenamento em disco e leituras das placas de comunicação.

As rotinas da atualização do vídeo, do controlador e das perturbações aplicadas ao processo, fazem parte das rotinas da pilha de execução.

Podemos observar que não existe um ponto de finalização no fluxograma, o sistema é encerrado através de uma rotina especial, acionada via teclado, que obriga-o a cancelar todos os procedimentos. As interligações e os nomes das diversas rotinas que compõem o sistema podem ser vistos através do Anexo I.

6.1 Reformulação Inicial

O programa RTX foi inicialmente desenvolvido para o compilador FORTRAN da MicroSoft Corporation Versão 3.34. As rotinas de comunicações com as placas analógicas/digitas eram escritas em linguagem ASSEMBLER e incorporadas ao *software*. As telas iniciais eram em modo texto, utilizando modo gráfico somente na tela de trabalho final. O algoritmo de controle implantado era para controladores da família PID.

O trabalho de reformulação começou com a mudança do sistema para o compilador FTN77/486 da Universidade de Salford. O mesmo possui maior velocidade de compilação e constrói sistemas com capacidade de endereçamento de toda a memória disponível, não limitados aos 640 Kbytes de memória convencional.

Optamos pelo compilador FTN77/486, no desenvolvimento do *software* RTX, porque ele apresenta várias facilidades e opções de trabalho(item 6.1.1).

A etapa seguinte, foi a construção de novas rotinas para as placas analógicas/digitais em linguagem FTN77/486, descritas no item 6.3.3, sem a necessidade da interface em ASSEMBLER.

Incorporamos ao *software* o algoritmo de controle das redes neurais artificiais, bem como, dois tipos de treinamento, *Steepest Descent* e Decomposição em Valores Singulares.

Com rotinas desenvolvidas para o tratamento de telas, o *software* passou a oferecer um novo visual, uma interface gráfica com o usuário.

6.1.1 Características do Compilador FTN77/486

As necessidades de *hardware* para a execução do compilador FTN77/486 são:

$\Rightarrow$ processador 80486 ou superior, com disco rígido.

⇒ mínimo 1MByte de memória RAM. Contudo, é recomendado uma memória estendida adicional, pois os programas não estão limitados aos 640 KBytes de memória convencional.

⇒ sistema operacional MS-DOS ou PC-DOS versão 3.3 ou superior.

As principais características do compilador são:

⇒ alta velocidade de compilação e linkedição. O FTN77/486 desenvolveu uma velocidade de compilação de 13000 linhas por minuto em um computador IBM PS/2 Model 80. O linkeditor, LINK77, é igualmente rápido.

⇒ os programas podem ser compilados em modo *check*, modo *otimização local*(padrão), ou modo *otimização global*.

⇒ diagnósticos em tempo de compilação. As mensagens de erros em tempo de compilação são escritas em Inglês, referenciando nomes, endereços, etc., como apropriado. Mensagens de erros em outras línguas, como Francês ou Alemão, podem ser construídas.

⇒ diagnósticos em tempo de execução. Checagens opcionais em tempo de execução estão disponíveis para tamanhos de *arrays*, aritmética *overflow*, consistência de argumentos de rotinas, variáveis indefinidas, etc.

⇒ linguagem ASSEMBLER in-line. O FTN77/486 suporta facilidades CODE/EDOC de instruções *assembler* para processadores 80486 em modo protegido 32-bit.

⇒ extensões de linguagem. Entre as extensões disponíveis estão as declarações DO/ENDDO e DO WHILE, ENCODE/DECODE e dados Hollerith. Compilação com códigos de pesquisa condicionais estão disponíveis utilizando-se CIF, CELSE e CENDIF. Declarações e tipos de variáveis usadas foram estendidas para INTEGER*1, INTEGER*2, INTEGER*4, REAL*4, REAL*8 (DOUBLE PRECISION), COMPLEX*8, COMPLEX*16, LOGICAL*1, LOGICAL*2 e LOGICAL*4.

6.2 Transferência de Informações Entre as Rotinas

A transferência de dados entre os diversos módulos é realizada através de comandos COMMON, inseridos nos arquivos com extensão INC. Estes arquivos são adicionados aos módulos através de instruções INCLUDE.

A concepção do sistema é idealizada de modo que qualquer rotina escreva ou leia na memória informações das diversas variáveis (Figura 6.2). A inicialização dessas variáveis é realizada por uma rotina de configuração, denominada CFGRED.

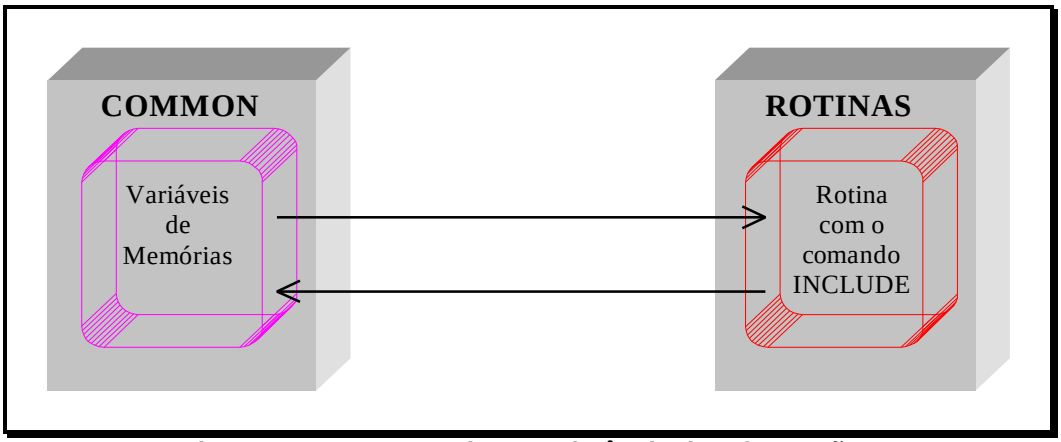


Figura 6.2 - Esquema de Transferência de Informações entre os Módulos do RTX

6.3 Rotinas de Inicializações

Essas rotinas visam realizar tarefas iniciais que não precisam ou não podem ser repetidas no *looping* do *software* RTX(ver Figura 6.1). Dentre essas tarefas podemos destacar, a atribuição de valores iniciais às variáveis das diversas rotinas do sistema e geração de arquivos de armazenamento de dados.

6.3.1 Leitura das Configurações Iniciais

A rotina CFGRED realiza a leitura dos dados iniciais informados através de um arquivo de configuração. O arquivo default é PADRAO.DAT, podendo ser informado outro nome durante a execução.

Após a leitura dos dados a rotina realiza a verificação de consistência dos mesmos, verificando se não foi informado algum dado que pode abortar o *software*, como por exemplo, o número de neurônios da camada de entrada da rede ser negativo ou nulo.

Através do *Anexo II* podemos verificar as variáveis necessárias para a configuração inicial do *software*. Essas variáveis são inicializadas utilizando um arquivo em formato

ASCII[10], que pode ser alterado por qualquer editor de textos. Assim, temos a flexibilidade para mudar as configurações do sistema antes da sua execução.

6.3.2 Inicialização do Sistema

Os dados e parâmetros utilizados pelo programa são inicializados pela rotina INICSIST. A rotina procura definir os parâmetros principais do *software*, como tempo de amostragem, número de registros do arquivo de armazenamento de dados e pesos iniciais da rede para o treinamento *Steepest Descent*.

Através de uma vetor denominado IAUTO, interno na rotina INICSIST, definimos as rotinas que fazem parte das *rotinas de execução continua*. Essas rotinas foram numeradas de 51 a 99 com o nome composto da forma "L"+(número da rotina). Na posição 1 do vetor IAUTO definimos a quantidade de rotinas de execução contínua a serem processadas, e nas demais posições o número dessas rotinas. Neste trabalho utilizamos a seguinte definição para IAUTO:

IAUTO(1) = 3 quantidade de rotinas de execução contínua
IAUTO(2) = 51 rotina L51: leitura das placas A/D e D/A
IAUTO(3) = 52 rotina L52: armazenamento de dados em disco
IAUTO(4) = 53 rotina L53: treinamento da RN

Para a geração do arquivo de armazenamento de dados a rotina INICSIST faz chamada a rotina DIARIO (Anexo I), que gera o nome do arquivo como sendo;

DTddmmaa.???, onde DT = identifica como arquivo de dados
dd = dia, mm = mês e aa = ano da amostragem
???= extensão que varia de 001 a 999

A rotina ABRDAT utiliza o nome definido pela DIARIO para a geração de um arquivo de armazenamento com a seguinte estrutura;

Registro 01 : Número de Registros | Hora Inicial | Tempo Amostragem
Registro 02 : Validade dos Canais [1 a 8=Entrada 9 a 16=Saida] [0=Inválido 1=Válido]
Registro 03 : Valor inferior do Range dos Canais [Unidade do Usuário]

[10] ASCII - American Standart Code International Interchange - consiste em uma tabela americana, onde cada caracter recebe um código padrão, de intercâmbio internacional.

Registro 04 : Valor superior do Range dos Canais [Unidade do Usuário]
Registro 05 : Valores de entrada[1 a 8] e Valores de Saída[9 a 16]
Registro 06 : Valores de entrada[1 a 8] e Valores de Saída[9 a 16]
. . . .
. . . .
. . . .
Registro MáxReg : Valores de entrada[1 a 8] e Valores de Saída[9 a 16]

onde;

- *Número de Registros* é a quantidade de registros gravados no arquivo
- *Hora Inicial* é a hora de início de gravação dos dados
- *Tempo de amostragem* é o intervalo de amostragem do processo
- *Validade dos canais* define quais os canais que possuem valores significativos
- *Valor inferior e superior do range* corresponde ao menor e maior valor possível dos dados dos canais gravados em unidade do usuário. Essa unidade corresponde ao sistema métrico ou inglês, pois o *software* internamente utiliza um escalonamento apropriado para esses valores
- *Valores de entrada e saída* representam os valores das variáveis a cada instante amostrado
- *MáxReg* é a quantidade máxima de registros a serem gravados no arquivo de dados. Esse valor é calculado através da razão entre o tempo máximo de gravação XRG(97) e o intervalo de amostragem IPAR(04), ambos definidos no arquivo de configuração PADRAO.DAT (Anexo II).

A inicialização dos pesos da rede é realizada através da rotina INICBPN, utilizando a função de geração de valores pseudo-aleatórios RANDOM().

6.3.3 Inicialização das Placas A/D e D/A

Os conversores de sinais analógico/digitais e digitais/analógicos, utilizados neste trabalho, são de fabricação da DATA TRANSLATION INCORPORATION, suas característica estão apresentadas na Tabela 6.1.

Modelo	Tipo	Canais	Entrada	Saída	Interface
DT2814	A/D	16	Range: 1 a 5 Volt	Range: 0 a 4095 Endereços: Base+0: 2 registros de 8-Bit Base+1: 1 registro de 8-Bit	Compatível IBM PC/XT/AT Endereço Base: 200H a 3FEH 220H(default) IRQs: 3, 4, 5, 6 ou 7 (Jumpers)
DT2815	D/A	8	Range: 0 a 4095 Endereços: Base+0: 2 registros de 8-Bit Base+1: 1 registro de 8-Bit	Range: 4 a 20 mA	Compatível IBM PC/XT/AT Endereço Base: 200H a 3FEH 224H(default) IRQs: não utiliza

Tabela 6.1 *- Características das Placas A/D e D/A*

Essas placas são inicializadas para uso, dentro do *software* RTX, através da rotina INICDA.

A inicialização da placa A/D é realizada escrevendo o valor 0 (zero) no *registro de controle*, localizado no endereço base (220H). O registro de controle é composto de 8-Bits descritos como:

BIT	7	6	5	4	3	2	1	0
FUNÇÃO	F2	F1	F0	ENB	C3	C2	C1	C0

Tabela 6.2 *- Registro de Controle da Placa A/D*

onde;

F2, F1 e F0 - especificam o fator decimal de divisão da frequência base de operação da placa, conforme a Tabela 6.3. A frequência base é definida através de *jumpers*.

Divisor			Frequência Selecionada
F2	F1	F0	
0	0	0	Frequência Base
0	0	1	Frequência Base/10
0	1	0	Frequência Base/100
0	1	1	Frequência Base/1k
1	0	0	Frequência Base/10k
1	0	1	Frequência Base/100k
1	1	0	Frequência Base/1M
1	1	1	Frequência Base/10M

Tabela 6.3 *- Fatores Decimais do Registro de Controle da Placa A/D*

ENB - especifica o uso do fator decimal de divisão de frequência no estado 1. No estado 0 desabilita o uso do fator decimal de divisão.

C3, C2, C1 e C0 - selecionam um dos 16 canais disponíveis na placa para a corrente conversão A/D. A configuração para seleção segue a Tabela 6.4.

Configuração dos Bits				**Canal**
C3	**C2**	**C1**	**C0**	**Selecionado**
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Tabela 6.4 *- Configuração dos Bits para Seleção de Canais da Placa A/D*

A inicialização da placa D/A é realizada através de duas operações:

- ***reset*** **da placa** - escrever qualquer valor no *registro de status*
- **inicialização de canais** - escrever no *registro de dados* o número de canais a serem utilizados pela placa

O *registro de status* da placa, localizado no endereço *base+1* (225H), possui a seguinte estrutura:

BIT	7	6	5	4	3	2	1	0
FUNÇÃO	X	S2	S1	S0	X	FO	IBF	X

Tabela 6.5 *- Estrutura do Registro de Status da Placa D/A*

onde;

X - bit não utilizado

S2 - estado 1 indica que ocorreu um erro durante o teste de RAM do *software* interno da placa. Estado 0 indica que não houve erros.

S1 - estado 1 indica que ocorreu um erro durante o teste de ROM do *software* interno da placa. Estado 0 indica que não houve erros.

S0 - estado 1 indica que a placa aguarda no *registro de dados* o valor superior de um byte. Estado 0 indica que aguarda o valor inferior de um byte.

FO - estado 1 indica que a placa está pronta para a inicialização de canais. Estado 0 indica que a placa não está preparada para a inicialização.

IBF - estado 1 indica que o *buffer* de entrada está cheio. Estado 0 indica *buffer* vazio.

O *reset* da placa é realizado através da verificação do bit 2 (FO) do *registro de status*. Estando FO no estado 0 a placa não pode ser inicializada. Escrevemos então, qualquer valor neste registro e verificamos novamente o estado de FO. Esta situação se repete até que o estado de FO tenha valor 1.

Após o *reset* da placa a primeira informação escrita no *registro de dados* é considerada como o número de canais a serem utilizados, sendo as informações seguintes consideradas dados para o processo.

O *registro de dados* esta localizado no endereço *base* (224H) e recebe a inicialização de canais através da estrutura:

BIT	7	6	5	4	3	2	1	0
FUNÇÃO	X	X	X	N1	N0	D2	D1	D0

Tabela 6.6 *- Estrutura do Registro de Dados da Placa D/A*

onde;

X - bit não utilizado

N1 e N0 - seleção de um dos quatros programas residentes na placa D/A

D2, D1 e D0 - especifica o número de canais a serem utilizados pela placa segundo a Tabela 6.7.

Estando as duas placas inicializadas, o *software* RTX pode receber e enviar informações para o processo.

Configuração dos Bits			Canais Disponíveis
D2	D1	D0	
0	0	0	0
0	0	1	0 a 1
0	1	0	0 a 2
0	1	1	0 a 3
1	0	0	0 a 4
1	0	1	0 a 5
1	1	0	0 a 6
1	1	1	0 a 7

Tabela 6.7 *- Configuração dos Bits para Definição do Número de Canais da Placa D/A*

6.3.4 Inicialização do Controlador PID

O controlador PID quando colocado em atividade, necessita das constantes do algoritmo da velocidade *d0, d1, d2 e c1* (Capítulo 2). Essas constantes são determinadas através da rotina TUNE_PID, utilizando os parâmetros do processo; constante de tempo, constante de proporcionalidade e tempo morto.

A rotina TUNE_PID é capaz de ajustar, para processos de 1ª ordem, controladores da família PID, dependendo do estado da variável IDATAR(28), ou seja:

IDATAR(28) = 0 ⟶ sem controlador
IDATAR(28) = 1 ⟶ Rede Neural
IDATAR(28) = 2 ⟶ PID
IDATAR(28) = 3 ⟶ PI
IDATAR(28) = 4 ⟶ P
IDATAR(28) = 5 ⟶ PD

Os valores das constantes calculadas, dos parâmetros utilizados e o tipo de controlador, são armazenados em disco pela rotina com o nome de PID.ASC, conforme podemos observar no *Anexo I.*

6.3.5 Condições Operacionais de Partida

O *software* RTX possui dois modos de partida definidos como *partida via RTX* e *pré-partida.* A escolha é realizada através das teclas F1 e F2 como mostrado na Figura 6.3.

Na partida via RTX, o *software* é iniciado com os valores dos canais e do *set-point* definidos no arquivo PADRAO.DAT (Anexo II).

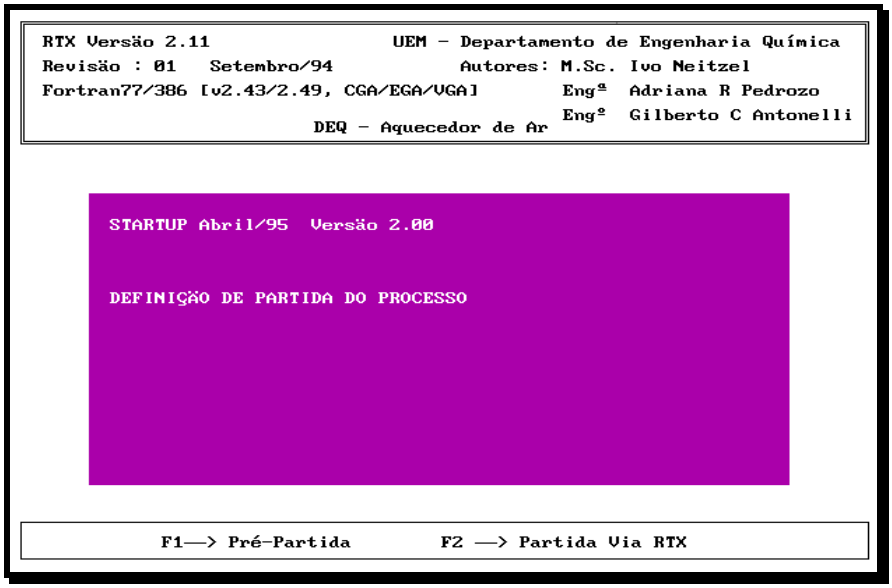


Figura 6.3 - Tela de Opções de Partida do Software RTX

A pré-partida foi construída com a finalidade de testar a comunicação entre o computador e o processo. Podemos alterar os valores iniciais dos canais de entrada e saída, e posteriormente, enviar ou receber essas informações. A Figura 6.4 mostra a tela de trabalho do *software* operando na pré-partida.

A rotina responsável pela escolha do tipo de partida e dos valores de partida é denominada STARTUP.

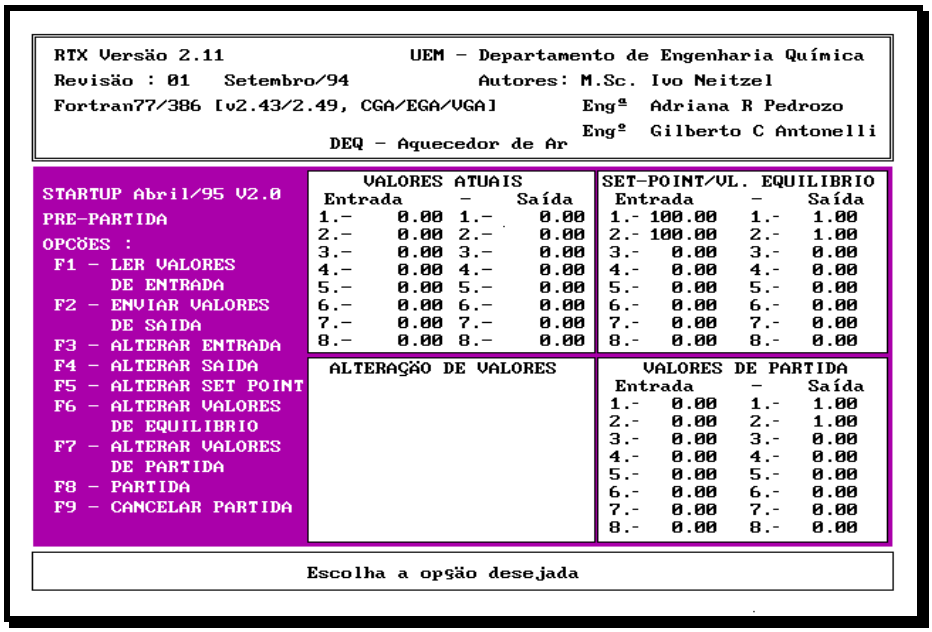


Figura 6.4 - Tela de Pré-Partida do Software RTX

6.4. Rotinas de Execução Através de Teclas Especiais

Através do uso de teclas especiais, no *software* RTX, podemos solicitar que seja executada uma determinada rotina. As teclas e suas respectivas rotinas são pré-definidas na montagem do *software*.

Os nomes das rotinas a serem executadas pelas teclas especiais são iguais às teclas, facilitando a localização, como por exemplo, as teclas CTRL+F10 fazem chamada à rotina CTRLF10.

As opções disponíveis, em termos de teclas especiais, são mostradas na tela de trabalho do *software* conforme a Figura 6.5.

6.4.1 As Tecla F1 a F10

As opções oferecidas através do conjunto de teclas F1 a F10 são:

- F1 - Ativa e desativa a possibilidade de utilização do teclado no *software*
- F2 - Encerra a execução do *software*, armazenando os dados finais em disco e devolvendo a tela para modo texto.
- F3 - Exibe informações atuais nos canais de entrada e saída do processo
- F4 - Ativa e desativa a atuação de um controlador (RN ou da família PID) no processo
- F5 - Aplica uma perturbação DEGRAU no processo utilizando a configuração definida no arquivo de dados DEGRAU.DAT (Anexo III)
- F6 - Aplica uma perturbação PULSO no processo utilizando a configuração definida no arquivo de dados PULSO.DAT (Anexo III)
- F7 - Aplica uma perturbação SENOIDE no processo utilizando a configuração definida no arquivo de dados SENO.DAT (Anexo III)
- F8 - Aplica uma perturbação PRBS no processo utilizando a configuração definida no arquivo de dados PRBS.DAT (Anexo III)
- F9 - Desativa qualquer perturbação aplicada no processo
- F10 - Ativa e desativa a atuação do controle do processo pela RN

6.4.2 A Tecla CTRL

A tecla CTRL acionada com as teclas F1 a F10 formam o segundo conjunto de opções do *software* RTX. Esse conjunto pode ser resumido como:

- CTRL + F1 - Grava a tela do sistema atual em arquivo formato PCX. Opção usada para documentar o sistema

- CTRL + F2 e F3 - Não utilizadas
- CTRL + F4 - Eleva o valor do *set-point* do processo a um valor acima do atual. Os valores são pré-definidos no arquivo de configuração PADRAO.DAT
- CTRL + F5 - Não utilizada
- CTRL + F6 - Ativa uma perturbação de SEQUÊNCIA DE PULSOS no processo
- CTRL + F7 e F8 - Não utilizadas
- CTRL + F9 - Ativa e desativa o treinamento contínuo da Rede Neural
- CTRL + F10 - Atualiza os pesos da Rede Neural com valores previamente gravados em um arquivo padrão. Utilizado somente no treino *Steepest Descent*

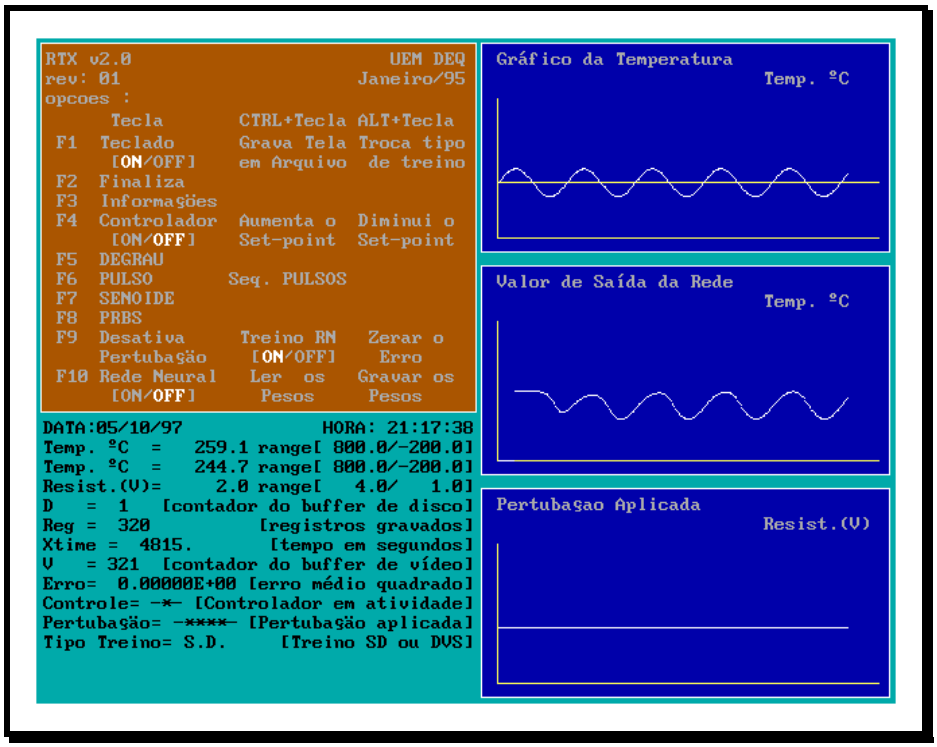


Figura 6.5 - Tela de Trabalho do Software RTX

6.4.3 A Tecla ALT

A tecla ALT aliada às teclas F1 a F10 formam o terceiro conjunto de opções, de rotinas de execução através do teclado, no *software* RTX. As opções disponíveis são:

- ALT + F1 - Troca o tipo de treino. Alterna entre *Steepest Descent* e Decomposição em Valores Singulares.
- ALT + F2 a F3 - Não utilizadas

- ALT + F4 - Diminui o valor do *set-point* do processo a um valor abaixo do atual. Os valores são pré-definidos no arquivo de configuração PADRAO.DAT
- ALT + F5 a F8 - Não utilizadas
- ALT + F9 - Zera a soma do erro médio quadrado
- ALT + F10 - Grava os pesos em uso atualmente pela rede em um arquivo padrão.

6.5 Rotinas de Execução Contínua

O grupo das rotinas de execução contínua é composto por todos os procedimentos que estão diretamente ligados ao tempo de amostragem, ou seja, devem ser executados após decorrido um intervalo de amostragem. Neste trabalho, fazem parte desse grupo as rotinas de entrada e saída de dados, armazenamento de dados em disco e treinamento da RN.

6.5.1 Entrada e Saída de Dados

A rotina L51 realiza o gerenciamento das entradas e saída de dados. Conforme o Anexo I, essa rotina faz chamada as rotinas AD e DA que efetuam, respectivamente, a leitura e o envio de dados através do canal solicitado por L51.

Sendo o range de trabalho das variáveis internas do *software* 0 a 5 (Volt) e das placas de conversão 0 a 4095, houve a necessidade de transformação, na recepção e no envio de valores.

A leitura de um único canal é realizada várias vezes, sendo utilizado o valor médio das leituras como valor de entrada. O objetivo é minimizar os erros produzidos pelas interferências externas (ruídos). A quantidade de leituras de um único canal é determinada pela variável IDATAR(32), definida no arquivo PADRAO.DAT (Anexo II).

Para realizarmos a leitura do valor de um canal na placa A/D, primeiramente é fornecido no *registro de controle* o número do canal a ser lido (Tabela 6.2). Após, é feita a leitura no *registro de dados* de dois registros de 8-Bits conforme a Tabela 6.8a e 6.8b.

BIT	7	6	5	4	3	2	1	0
FUNÇÃO	D11	D10	D09	D08	D07	D06	D05	D04

a) 1ª Leitura - Byte Superior

BIT	7	6	5	4	3	2	1	0
FUNÇÃO	D03	D02	D01	D00	0	0	0	0

b) 2ª Leitura - Byte Inferior

***Tabela 6.8** - Registro de Dados da Placa A/D*

onde;

D11 a D00 - representam o valor de entrada em formato binário (12-Bits)

Fazendo a conversão do valor lido, para formato decimal, temos o valor de entrada do canal especificado.

O envio de valores através da placa D/A é realizado mediante o preenchimento do *registro de dados*, no endereço *base* (224H), com dois valores de acordo com a Tabela 6.9a e 6.9b.

BIT	7	6	5	4	3	2	1	0
FUNÇÃO	D11	D10	D09	D08	D07	D06	D05	D04

a) 1º Valor Enviado - Byte Superior

BIT	7	6	5	4	3	2	1	0
FUNÇÃO	D03	D02	D01	D00	C2	C1	C0	X

b) 2º Valor Enviado - Byte Inferior

***Tabela 6.9** - Registro de Dados da Placa D/A*

onde;

X - bit não utilizado

D11 a D00 - representam o valor de saída em formato binário (12-Bits)

C2, C1 e C0 - número do canal de saída a ser utilizado (semelhante à Tabela 6.7)

O teste para verificar se a placa pode receber o valor de saída, superior ou inferior, é realizado através do bit 4 do *registro de status* (Tabela 6.5).

6.5.2 Armazenamento de Dados em Disco

Como visto no item 6.3.2, a rotina DIARIO cria um arquivo de dados, com uma estrutura que é preenchida pela rotina de execução contínua L52. A finalidade do arquivo já estar criado, é para agilizar o *software* no armazenamento dos dados, pois, o preenchimento é mais rápido do que a inclusão de registros.

A rotina L52 utiliza um *buffer* de memória, variáveis XVIN e XVOUT, para armazenamento temporário dos dados a serem gravados em disco. A gravação é realizada pela rotina GRAVADADOS quando o *buffer* está cheio, ou se o operador solicitou o encerramento do *software*. O tamanho do *buffer*, em termos de registros, é definida no arquivo PADRAO.DAT na variável IPAR(07).

Os arquivos de dados são gerados de acordo com a data da execução do programa, sendo criados automaticamente com a mudança de dia.

6.5.3 Treinamento da Rede Neural

Para treinamento da rede utilizamos a rotina L53, que realiza chamadas a diversas rotinas dependendo do tipo de treinamento. A rotina L53 representa a implantação no *software* das equações apresentadas no Capítulo 3.

Durante o treinamento *Steepest Descent*., a rotina L53 atualiza os dados da camada de entrada da rede, os pesos através da rotina BACKWARD e estima o valor de saída com a rotina FORWARD.

No treinamento Decomposição em Valores Singulares é chamada a rotina GERAR_AB que monta a matriz de observações, realiza a ACP e define as matrizes A e B, utilizadas pela rotina GERAR_WA para determinar os coeficientes da polinomial transformados. Finalmente a rotina GERAR_Y é utilizada para determinar o valor de saída da rede.

6.6 Rotinas da Pilha de Execução

A pilha de execução representa um conjunto de rotinas que são executadas durante um certo período ou com uma frequência definida. Por exemplo, uma rotina que aplica uma perturbação no processo deve ser executada durante um tempo pré-definido, sendo cancelada sua execução após esse tempo. Essas rotinas foram numeradas de 100 a 999 com o nome composto da forma "L" +(número da rotina).

Uma matriz de frequência denominada IFRQ, representa internamente no *software*, a pilha de execução (Tabela 6.10). Definida como uma matriz de 30 linhas e 33 colunas é possível programar 30 frequências diferentes com 30 rotinas cada uma.

	Contador CT	**Frequência FQ**	**Número de Rotinas válidas NR**	**1ª Rotina a Executar**	**2ª Rotina a Executar**	**. . .. .**	**30ª Rotina a Executar**
Colunas Linhas	1	2	3	4	5	. . . . .	30
1	CT_1	FQ_1	NR_1	$R_{1,1}$	$R_{1,2}$	. . . . .	$R_{1,30}$
2	CT_2	FQ_2	NR_2	$R_{2,1}$	$R_{2,2}$	. . . . .	$R_{2,30}$
. .	. .	. .	. .	. .	. .	. .	. .
30	CT_{30}	FQ_{30}	NR_{30}	$R_{30,1}$	$R_{30,2}$	. . . . .	$R_{30,30}$

Tabela 6.10 *- Pilha de Execução de Rotinas do Software RTX*

As NR_i rotinas de uma linha, são executadas com a frequência FQ_i, definida na segunda coluna. Primeiramente o contador CT_i, da primeira coluna, é inicializado com FQ_i, e a cada intervalo de amostragem ele é decrescido de uma unidade. Quando o valor desse contador é igual a zero, as rotinas válidas na linha começam a ser executadas.

As operações com a matriz de frequência são realizadas através das rotinas FRQPOE, que insere uma rotina na matriz, e FRQSAI, que retira uma rotina da matriz.

6.6.1 Atualização dos Dados em Vídeo

A tela base de trabalho do *software* RTX contendo os menus, os eixos dos gráficos , mensagens e contornos, é construída pela rotina L101 e armazenada em memória na variável TELA_BASE. Através de uma variável (IDATAR(40)) identificamos a última rotina que alterou o vídeo, verificando assim, a necessidade de restauração da tela base de trabalho através da variável de memória.

A tela de trabalho é dividida em três regiões distintas, *área de menus*, *área de dados e informações* e *área de gráficos* (Figura 6.5).

A *área de menus* informa as teclas de opções que o operador possui para "interferir" na execução do sistema. Para que o menu possa ser utilizado, primeiramente devemos ativar o teclado através da tecla F1.

A *área de dados e informações* é o local onde dados quantitativos são mostrados, são eles:

LINHA 1	- Data e Hora da execução do sistema
LINHA 2 a 4	- Valores atuais das variáveis do processo que estão sendo construídos os gráficos na área de gráficos bem como seu range
LINHA 5	- Indica o número de registros de dados armazenados no *buffer* que serão transferidos para disco. O quantidade é definida no arquivo PADRAO.DAT
LINHA 6	- Número de registros já gravados em disco
LINHA 7	- Tempo decorrido, em segundos, desde o início da execução do sistema
LINHA 8	- Quantidade de pontos plotados nos gráficos
LINHA 9	- Valor do erro médio quadrado da variável controlada em relação ao *set-point* durante a aplicação de uma perturbação
LINHA 10	- Controlador em atividade, as opções são PID e BPN
LINHA 11	- Perturbação aplicada DEGRAU, PULSO, SENOIDE, PRBS ou SEQÜÊNCIA DE PULSOS

A *área de gráficos* é a região da tela onde são construídos no máximo três gráficos de variáveis importantes do processo. A decisão de quais variáveis serão visualizadas em

gráficos deve ser tomada antes da inicialização do sistema, bem como, os títulos e rótulos dessas variáveis (lado superior esquerdo dos gráficos). Esses dados são fornecidos através do arquivo PADRAO.DAT

O valor do *set-point* ou os valores de equilíbrio das variáveis são representadas nos gráficos por uma linha horizontal.

Existem dois conceitos dentro do *software* quanto à apresentação de informações no vídeo, a *tela absoluta* e a *tela relativa*.

A tela absoluta representa a situação na qual nenhuma rotina pode escrever no vídeo. Sendo que, somente a rotina que provocou essa situação pode revertê-la.

A tela relativa é a situação na qual nenhuma rotina pode escrever em vídeo durante um certo período, ou seja, após esse tempo qualquer rotina pode manipular o vídeo. As rotinas que solicitam a tela relativa implementam na variável XDATAR(02) o tempo em que a tela deverá ficar inalterada. O teste de estado da tela é realizado por todas as rotinas que necessitam escrever em vídeo. Existe uma rotina lógica chamada TESTVID para essa finalidade. Essa rotina utiliza sinalizadores (flags) definidos como:

Tela Absoluta	KFLAG(41) = 1 ativada
	KFLAG(41) = 0 desativada
Tela Relativa	KFLAG(42) = 1 ativada
	KFLAG(42) = 2 desativada

6.6.2 Controlador PID

Utilizando os dados gerados pela rotina de inicialização TUNE_PID (item 6.3.4) a rotina de controle L108 determina o valor da variável manipulada a ser implantada no processo.

Na verdade, a rotina L108 é a implantação do algoritmo da velocidade descrito no Capítulo 2. Essa rotina se auto-retira da pilha de execução caso a variável IDATAR(19) não possua indicação de controladores da família PID em atuação, conforme descrito no item 6.3.4.

6.6.3 Controlador RN

O controle pela RN é realizado através da rotina L106, onde implantamos o algoritmo descrito no item 5.1, para os dois tipos de treinamentos. Esse algoritmo, com função de ativação sigmoidal na camada de saída, produz valores de saída dentro do range de 0 a 1 (ver

item 3.2). Como as variáveis internas no *software* trabalham com range de 0 a 5, essa rotina além de calcular o valor de saída da rede, realiza o escalonamento apropriado.

A rotina L106 se auto-retira da pilha de execução caso a variável IDATAR(11) não possua valor igual a 1. Essa variável indica se a rede está ativada ou não.

6.6.4 Perturbações aplicadas

A solicitação para a aplicação de uma perturbação no processo é realizada pelo operador através do teclado. A tecla pressionada chama a rotina que inclui na pilha de execução, o procedimento que introduz a perturbação.

Cada perturbação aplicada possui um arquivo de configuração em formato ASCII (Anexo III). Nestes arquivos estão definidos os parâmetros necessários para a aplicação da perturbação. A leitura destes arquivos é realizada pela rotina CNFPERT.

Todas as rotinas de aplicação de perturbações se auto-retiram da pilha de execução após decorrido o tempo de aplicação, ou se o operador cancelou a mesma.

O estado do sinalizador de perturbação, as rotinas utilizadas e o arquivo de configuração correspondente, podem ser verificados através da Tabela 6.11.

Perturbação	Sinalizador	Rotina de Aplicação	Arquivo de Configuração
DEGRAU	IDATAR(20)=1	L112	DEGRAU.DAT
PULSO	IDATAR(20)=2	L113	PULSO.DAT
SENÓIDE	IDATAR(20)=3	L104	SENO.DAT
P.R.B.S.	IDATAR(20)=4	L105	PRBS.DAT
SEQUÊNCIA DE PULSOS	IDATAR(20)=5	L103	PULSOC.DAT

Tabela 6.11 *- Rotinas e Arquivos de Configurações das Perturbações Aplicadas*

6.7 Rotinas de Apoio

As rotinas que não foram classificadas nos itens 6.3 a 6.6, fazem parte do grupo denominado *rotinas de apoio*. Essas rotinas realizam trabalhos especiais, como atualizar o gráfico das variáveis no vídeo, definir cores para o *software*, desenhar telas iniciais do sistema, etc.

A rotina definida como GRAFVID no Anexo I, na verdade é um conjunto de rotinas de apoio para manuseio da tela em modo gráfico. Essas rotinas estão agrupadas em um arquivo chamado GRAFVID.FOR. As rotinas que compõem esse arquivo são:

TIPOVIDEO	- define se o tipo de vídeo utilizado é CGA, EGA ou VGA. Faz a leitura do arquivo CORES.DAT , que define as cores a serem utilizadas no sistema
TESTVID	- testa o modo de vídeo, tela relativa ou tela absoluta
LINHA	- desenha uma linha no vídeo nas coordenadas e cores definidas
MSG	- escreve uma mensagem no rodapé das telas iniciais
ALERTA	- escreve mensagens no rodapé da área de dados da tela de trabalho
DISPSTR	- escreve uma STRING na tela nas coordenadas e cores definidas
RETANGULO	- desenha um retângulo na tela nas coordenadas definidas. O retângulo pode ser preenchido ou vazado
SAVESCREEN	- salva uma região da tela em uma variável de memória
RESTSCREEN	- restaura uma região da tela a partir de uma variável de memória
CLEARSCREEN	- limpa uma região da tela com a cor definida
SAVESCREENPCX	- salva a tela de trabalho do *software* em arquivo formato PCX

A função TESTVID é a primeira rotina a ser executada pelo *software*, deste modo, o arquivo das cores (Anexo III) é lido e as cores definidas são transferidas para a variável COR.

Para desenho dos gráficos no vídeo utiliza-se a rotina de apoio SPLOT. Essa rotina verifica através do sinalizador KFLAG(02) se deve ser construído todo o gráfico solicitado, ou se apenas deve-se fazer a inclusão do último valor lido ou enviado. A construção total do gráfico é necessária quando alguma rotina estampa em vídeo, informações que se sobrepõem aos gráficos.

7 . Testes de Confiabilidade

Procurando estabelecer a confiança no *software* RTX, nas conexões do módulo prático, no algoritmo da RN e no controlador PID ajustado, realizamos diversos testes. Cada teste , em particular, visou uma parte do sistema ou esquema de controle montado.

7.1 Software RTX

O *software* RTX foi modificado da versão original com reformulação de várias rotinas, como especificado no Capítulo 6. Procurando testar as mudanças, realizamos testes com sua tela de *pré-partida* (Figura 6.4), a qual oferece a possibilidade de enviar um valor para o processo e posteriormente ler o valor enviado. Deste modo, testamos as conversões de saída e de entrada de dados no *software*, bem como, a comunicação entre o computador e módulo de testes.

7.2 Sintonia do Controlador PID

Utilizando os resultados dos testes de caracterização, demonstrados no Capítulo 4, para um sistema de 1ª ordem, descrito pela Equação 4A, com τ =2400s , K_p =1,273V/V e t_m =180s ajustamos um controlador do tipo PID na forma da Equação 2J.

Com o *software* RTX em modo simulação e utilizando a Equação 4E para representar o processo, realizamos teste de ajustes, representado pela Figura 7.1, onde o controlador PID ajustado, procura manter a temperatura do processo no valor do *set-point* de 200 ºC com um tempo de amostragem T_s=15s. Inicialmente o controlador alterou a variável manipulada para um valor elevado de modo que a temperatura do processo aumentasse, como era de se esperar. Monitorando a variável manipulada o controlador mostrou-se eficiente, conduzindo o processo ao *set-point*.

Após o sistema permanecer em equilíbrio, aplicamos um degrau na variável manipula de +0,60V durante 7200s. Observamos uma atuação satisfatória do controlador conseguindo amortecer a perturbação aplicada ao sistema conduzindo-o ao seu *set-point*. Assim, consideramos ajustado o controlador PID e preparado para controlar o aquecedor de ar.

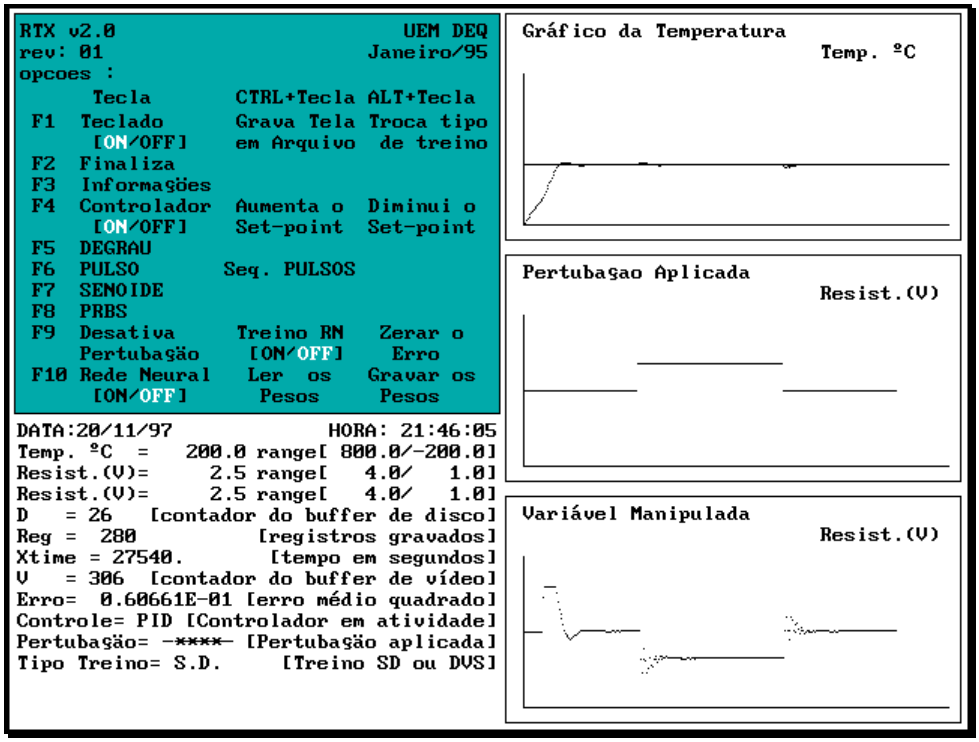


Figura 7.1 – Teste de Ajuste do Controlador PID

7.3 Algoritmo da RN

Para o teste de confiabilidade do algoritmo da rede neural e os dois tipos de treinamento implantados, utilizamos uma função senoidal de teste.

A variável IDATAR(01) define no *software* RTX a equação a ser utilizada para simulação, ou seja;

IDATAR(01) = 0 define a utilização do processo da Equação 4E

IDATAR(01) = 1 define a utilização da senóide de teste

Obtivemos para o treinamento *Steepest Descent* o resultado apresentado na Figura 7.2, onde observamos que a rede conseguiu aprender a função senoidal de modo satisfatório. Os parâmetros utilizados para esse teste foram os seguintes:

⇒ 2 (dois) neurônios na camada de entrada da rede

⇒ 10 (dez) neurônios na camada intermediária (escondida)

⇒ 1 (um) neurônio na camada de saída

⇒ passo de aprendizado η=0,001

⇒ amplitude da senóide 0,5 Volt

⇒ período da senóide 40π

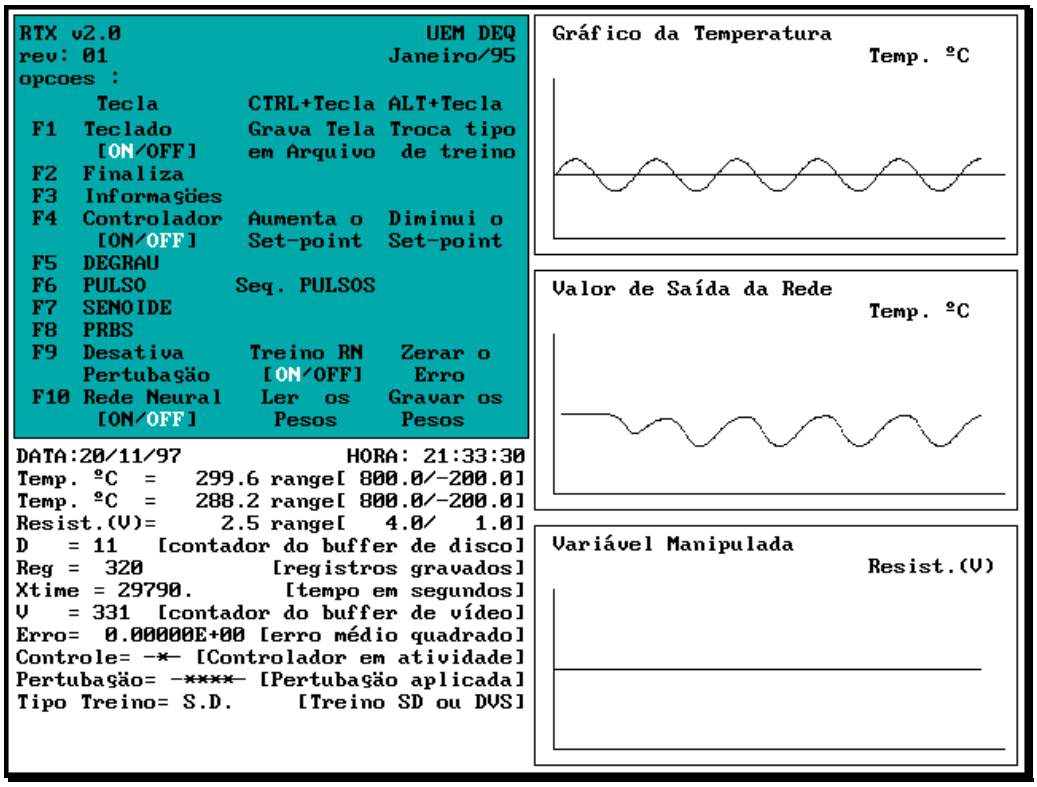


Figura 7.2 - Teste do Ajuste da RN para o Treinamento *Steepest Descent*

O treinamento Decomposição em Valores Singulares apresentou resultados semelhantes ao treinamento *Steepest Descent* como podemos observar na Figura 7.3. Existe um tempo inicial para que a rede produza o primeiro valor aproximado. Este tempo é devido ao fato que precisamos preencher a janela de observações para poder então estimar o primeiro valor.

Utilizamos os parâmetros listados a seguir para elaboração deste teste.

⇒ 2 (dois) neurônios na camada de entrada

⇒ 1 (um) neurônio na camada de saída

⇒ polinomial de grau 2 (dois) para aproximação da função sigmoidal

⇒ 50 (cinquenta) observações para treino

Os neurônios da camada intermediária são determinados através da ACP realizada a cada novo treinamento da rede. Como o treinamento neste caso foi contínuo, a cada valor

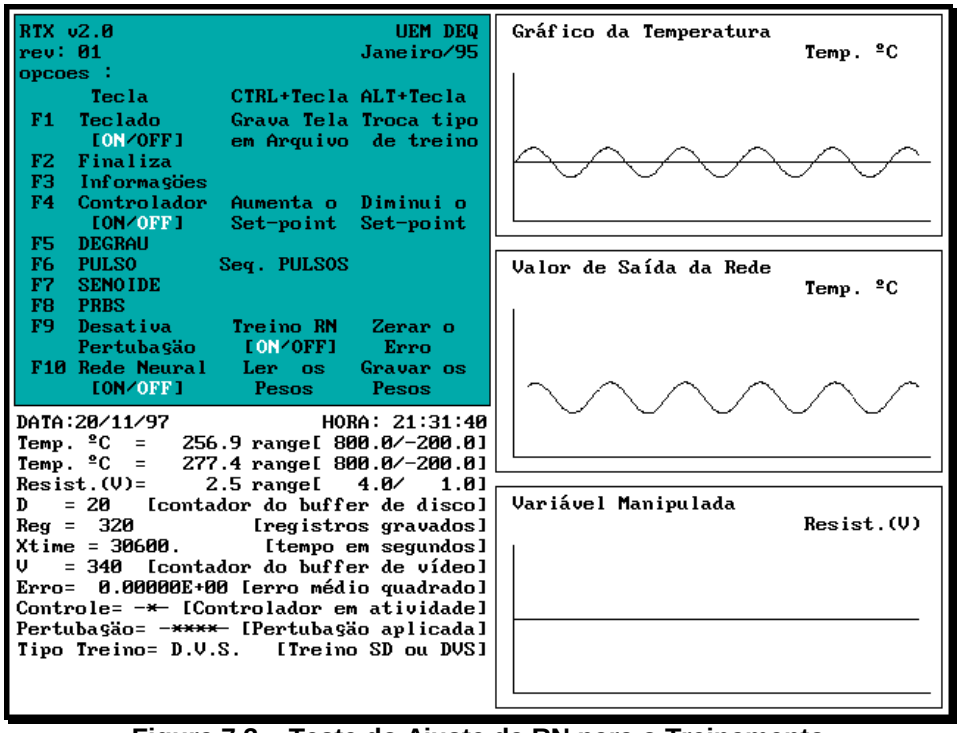


Figura 7.3 – Teste do Ajuste da RN para o Treinamento Decomposição em Valores Singulares

estimado pela rede o algoritmo determinou o melhor número de neurônios. Esta informação é interna do *software* não estando disponível ao usuário.

Com esses resultados consideramos os algoritmos de treinamentos da rede neural prontos para serem aplicados no modelo de controle a ser implantado no módulo de testes, ou seja, as equações implantadas no *software* RTX estavam corretas e confiáveis.

8. Treinamento da Rede Neural

Utilizando o *software* RTX no modo simulação, analisamos os parâmetros necessários para os dois tipos de treinamento implantados de modo que a RN aprendesse o processo simulado.

8.1 Ajustes de Parâmetros para o Treinamento *Steepest Descent*

Vários parâmetros da RN devem ser determinados para que a rede possa aprender o processo (ver item 3.4.1), assim, procuramos determinar esses valores no modo simulado para depois serem aplicados no módulo de testes.

8.1.1 Tempo de Amostragem

O tempo de amostragem utilizado para a caracterização do sistema nos testes (*Ts=15s*) se mostrou ineficiente quando utilizado para o treinamento *Steepest Descent*, conforme mostrado na Tabela 8.1. A RN não convergiu, porque tendo o processo uma constante de tempo τ = 2400s, os valores informados à rede a cada 15 segundos eram muito próximos entre si, gerando um vetor de entrada da rede quase constante. É importante observar que a rede para aprender o processo necessita de informações de várias situações diferentes, ou seja, pares de treinos entrada-saída independentes.

Procuramos através dos testes definir um tempo de amostragem de modo que a RN aprendesse mais rapidamente o processo, ou seja, reproduzisse o valor de saída com um erro inferior ou igual a 0,1%. A tabela a seguir mostra os resultados obtidos.

Tempo de Amostragem (*Ts*)	**Tempo para produzir erro £ 0,001**
15 segundos	não convergiu
30 segundos	12330 segundos
90 segundos	10350 segundos
120 segundos	11760 segundos

Tabela 8.1 *- Resultados da Simulação para o Tempo de Amostragem*

Após os testes passamos a utilizar o tempo de amostragem *Ts=90s*, já que o mesmo apresentou melhores.

8.1.2 Número de Neurônios da Camada de Entrada

Os testes mostrados na Figura 8.1a e 8.1b, seguiram o mesmo critério para o erro do item 8.1.1, utilizando *Ts=90s*. O tempo para que a rede reproduzisse um valor de saída com erro menor que 0,1%, foi denominado de *tempo de aprendizagem*.

Para que a rede possa produzir seu primeiro valor de saída, é necessário que sejam preenchidos todos os neurônios da camada de entrada. Sendo o treinamento realizado de modo sequêncial, existe um tempo, definido como *tempo inicial*, que a RN não produz saída alguma (afastamento do gráfico de saída da rede do eixo y). Observamos que quanto maior a quantidade de neurônios na camada de entrada, maior é o *tempo inicial*. O *tempo inicial* é uma parcela do *tempo de aprendizagem*.

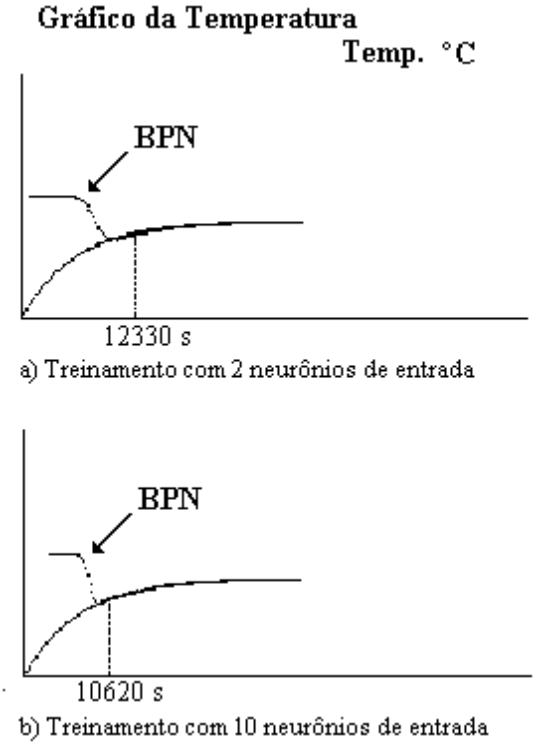


Figura 8.1 - Influência do Número de Neurônios na Camada de Entrada no Treinamento *Steepest Descent*

Realizamos testes para 2 e 10 neurônios na camada de entrada da rede, o tempo de aprendizagem para 2 neurônios foi de 12330s e para 10 neurônios 10620s, como mostrado na Figura 8.1. Houve um decréscimo de aproximadamente 30 minutos no tempo de aprendizagem, o que corresponde a um tempo 13,8% menor.

Analisando os resultados obtidos realizamos testes com 15 neurônios na camada de entrada da rede obtendo um tempo de aprendizagem de 4050s, diminuindo o tempo de aprendizagem em 67,2%.

Como um aumento significativo da quantidade de neurônios na camada de entrada produz um *esforço computacional muito grande*[11], optamos por utilizar 15 neurônios na camada de entrada, pois essa quantidade reduziu consideravelmente o tempo de aprendizagem.

Nesses testes mantivemos a distribuição dos dados de entrada (item 5.2) em 50% para a variável manipulada e 50% para a variável controlada.

[11] Aumento significativo na quantidade de operações lógicas e matemáticas realizada pelo computador provocando um tempo maior na busca do resultado

8.1.3 Distribuição dos Dados de Entrada

O conjunto de dados de entrada sempre obedeceu a distribuição do item 5.2. Procuramos através de testes determinar o valor ideal para o índice *j*, que representa, a quantidade de neurônios de entrada que recebe o valor da variável manipulada.

Utilizando 15 neurônios na camada de entrada, realizamos testes para valores de *j*=2, 5, 8 e 10. Os testes foram qualitativos, observando o comportamento da variável manipulada durante a ação do controlador RN, após a aplicação de uma perturbação degrau.

Para *j*=2 o controlador não conseguiu amortizar a perturbação completamente, houve uma grande oscilação da variável controlada. Isto ocorreu, porque a rede não conseguiu aprender a influência dos valores da variável manipulada no processo.

Utilizando *j*=5, 8 e 10, o controlador RN amortizou a perturbação aplicada, produzindo pequena oscilação da variável controlada. Observamos que a medida que *j* aumenta, ocorre um aumento das oscilações da variável manipulada em torno do seu *valor de equilíbrio*[12]. Para *j*=10 a variável manipulada não convergiu para um valor de equilíbrio, e permaneceu oscilando em torno de um outro valor.

Analisando os resultados adotamos como distribuição dos dados, na camada de entrada para o processo estudado, *j*=INTEIRO(N/3), ou seja, 1/3 dos dados de entrada da rede são informações da variável manipulada e 2/3 da variável controlada.

8.1.4 Número de Neurônios na Camada Intermediária

Para a definição da quantidade de neurônios na camada intermediária, realizamos testes com 5, 10 e 20 neurônios. Os testes foram qualitativos, observando o comportamento da variável manipulada durante a ação do controlador RN.

À medida que aumentamos a quantidade de neurônios na camada intermediária, a ação do controlador sobre a variável manipulada é amortecida, ou seja, não existe grande salto na variável manipulada. Acreditamos que essa característica é devido a um melhor aprendizado da rede do processo. Essa característica é importante do ponto de vista do elemento final de controle, pois o mesmo, teria um menor desgaste com o tempo. Por exemplo, tomemos como base uma válvula de controle, a mesma trabalharia sob ações suaves de abertura e fechamento, diminuindo os desgaste dos componentes internos com o tempo.

No processo em estudo utilizamos 50 neurônios na camada intermediária.

[12] Valor da variável manipulada que, estando o processo no *set-point*, o mantém estabilizado.

8.1.5 Número de Neurônios na Camada de Saída

O sistema em estudo neste trabalho possui uma única variável manipulada, a tensão nas resistências elétricas, necessitando apenas de uma única saída no controlador RN. Portanto a quantidade de neurônios na camada de saída (*M)* é igual a 1.

8.1.6 Passo de Aprendizagem

Os testes simulados para avaliar a influência do passo de aprendizagem(η) foram qualitativos, partindo-se de η=0,001 até η=0,100. Observou-se que para os valores dos extremos da faixa escolhida a rede não demonstrou estar aprendendo o processo.

Através da Figura 8.2 podemos visualizar o não aprendizado da rede do ponto de vista de um único peso de conexão. Como o tipo de treinamento é sequêncial, sendo o passo muito pequeno e estando no *ponto a*, a rede necessita de uma grande quantidade de passos para chegar em z. Isto implica em um tempo muito elevado. Já com o passo muito grande, a rede estando no *ponto b* ultrapassa o valor de mínimo da função atingindo o *ponto c* porque o salto provocado pelo passo é maior que a distância do *ponto b* ao ponto de mínimo *z*. Essa situação se repete com a função erro definida no espaço dos pesos das conexões.

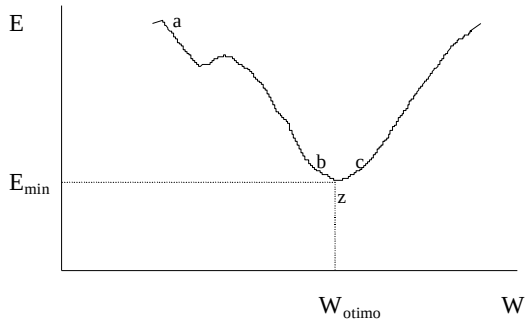


Figura 8.2 - Função Hipotética Erro x Peso W

Neste trabalho o valor de η=0,050 apresentou bons resultados e por isso foi adotado.

8.1.7 Incorporação de *Bias*

A incorporação de um *bias* na rede durante a simulação tornou o controlador RN instável, de modo que o mesmo, não conseguiu mais controlar o processo. Devido à esse fato, optamos por não utilizar a conexão *bias*.

8.2 Ajustes de Parâmetros para o Treinamento Decomposição em Valores Singulares

A determinação dos diversos parâmetros necessários para a implantação desse treinamento seguiu caminhos totalmente diferentes aos mostrados anteriormente no treinamento *Steepest Descent*. Vários parâmetros foram simplesmente definidos baseados em alguns critérios, sem efetuar testes de busca de valores ótimos, como é o caso do *tempo de*

amostragem e *incorporação de bias*. Outros porém, foram determinados como um grupo e não como um parâmetro isolado.

O critério de observação definido no item 8.1.1, determinação do tempo de amostragem para que a RN reproduzisse o valor de saída com um erro inferior ou igual a 0,1%, não pode ser utilizado devido às características desse método. Esse treinamento é do tipo *batelada* e o primeiro valor reproduzido pela rede após o treinamento possui um erro inferior ao estipulado. Por esse motivo, procuramos definir os parâmetros observando os resultados do *controle* realizado pela rede após o treinamento. Ajustamos os parâmetros de modo que o controle realizado pela RN convergisse para um valor de equilíbrio.

A maior dificuldade na determinação dos parâmetros para este tipo de treinamento está no fato de que alguns valores quando alterados sozinhos, não afetam consideravelmente os resultados do controle pela rede de modo que possamos definir um valor ótimo para ele. Na verdade a alteração de um conjunto de parâmetros faz com que o controle pela rede alterne entre melhores e piores resultados.

8.2.1 Tempo de Amostragem

Analisando os resultados obtidos para o treinamento *Steepest Descent* decidimos manter o tempo de amostragem *Ts=90s*, pois o mesmo mostrou-se eficiente no sentido de produzir informações do processo em várias situações diferentes.

8.2.2 Número de Neurônios na Camada de Saída

Devido às características do processo estudado como definido no item 8.1.5 a quantidade de neurônios na camada de saída é igual a 1.

8.2.3 Método Iterativo de Cálculo da Variável Manipulada

Para a determinação do valor a ser implantado no processo x_{qp}, da Equação 5D, necessitamos de um processo iterativo de cálculo. Vários métodos foram aplicados com o intuito de conseguir um método que convergisse na maioria das vezes que fosse utilizado. Para tanto, interrompemos a execução da rotina L106, do *software* RTX, responsável por determinar o valor de x_{qp}, e procuramos visualizar a cada acesso à rotina se esta havia convergido ou executado o limite máximo de 2000 iterações pré-estabelecidos. Os métodos utilizados estão apresentados na Tabela 8.2 em ordem de implantação e com suas respectivas características..

Ordem	Método	Características
01	Newton-Raphson	• utiliza um ponto inicial x_0 • aproxima da raiz com a derivada da função • novo valor $x_{n+1} = x_n + \frac{f(x_n)}{f'(x_n)}$
02	Substituições Sucessivas	• utiliza um ponto base *(x, f(x))* • aproxima *f(x)= f(x)-x=0* • novo valor *x = f(x)*
03	Falsa Secante	• utiliza um ponto como base *(c,f(c))* • novo valor $x_{n+1} = x_n + \frac{(c - x_n) f(x_n)}{f(c) - f(x_n)}$
04	Ajuste de uma Função Monotônica	• utiliza três pontos de partida $(x_1,y_1),(x_2,y_2)$ e (x_3,y_3) • aproxima $f(x) = y(x) = \frac{x - c_1}{c_2 x + c_3}$ • novo valor $x_{n+1} = \frac{x_1 y_2 y_3 (x_2 - x_3) - x_2 y_1 y_3 (x_1 - x_3) + x_3 y_1 y_2 (x_1 - x_2)}{y_2 y_3 (x_2 - x_3) - y_1 y_3 (x_1 - x_3) + y_1 y_2 (x_1 - x_2)}$

***Tabela 8.2** – Métodos Iterativos de Cálculos para Determinar a Variável Manipulada*

Para o método *Newton-Rapson* realizamos dois testes, o primeiro com o ponto inicial de partida sendo o valor de equilíbrio e o segundo com o ponto inicial sendo o último valor calculado. Em ambos os casos o resultado foi o mesmo, o método divergiu para a maioria das vezes em que foi aplicado, ultrapassando o limite máximo de iterações definido. O controle realizado pela RN não conseguiu levar o valor da variável manipulada para um valor de equilíbrio. Ocorreu uma oscilação em torno de um valor com uma amplitude equivalente ao range da variável manipulada.

A aplicação do método da *substituição sucessiva* requer a definição da função *f(x)*. Essa função foi obtida da Equação 5D considerando *n*=*G* e *j*=c, onde *c* é número do neurônio da camada escondida que possui a maior força de conexão (definido na ACP). Obtivemos a Equação 8A, sendo utilizada para a aplicação do método.

$$f(x) = \frac{1}{v_{pc}}\left[\left(\frac{a_1^{\,o} - S_1 - S_2}{s_{Gc}}\right)^{1/G} - \sum_{\substack{k \\ k\#p}}^{N} x_{qk} v_{kc}\right] \qquad (8A)$$

sendo;

$$S_1 = \sum_{\substack{j \\ j\#c}}^{N} \sum_{n=0}^{G} s_{nj}\left[\sum_{i}^{N} x_{qi} v_{ij}\right]^n \qquad \text{e} \qquad S_2 = \sum_{n=0}^{G-1} s_{nc}\left[\sum_{i}^{N} x_{qi} v_{ic}\right]^n$$

Realizamos novamente dois testes utilizando o valor de equilíbrio e o último valor calculado como ponto base. No primeiro caso, utilizando o valor de equilíbrio, o método apresentou resultados semelhantes ao Newton-Raphson, e o controle realizado pela RN oscilou em torno do valor de equilíbrio com uma amplitude igual ao valor do range da variável manipulada. Utilizando o último valor calculado o método divergiu sempre para valores na ordem de 10^{30}, chegando ao limite máximo de 2000 iterações, o que tornou impraticável o controle pela RN.

Os testes realizados com o método da *falsa secante* apresentaram melhores resultados do que os dois anteriores utilizando como ponto base o valor de equilíbrio. A maioria das vezes o método convergiu o que resultou na convergência do controle realizado pela RN ao valor de equilíbrio. No entanto, estando o processo próximo do *set-point* e portanto a variável manipulada próxima do valor de equilíbrio o método desestabilizava e começava a divergir, e consequentemente o controle realizado pela RN começava a oscilar com uma amplitude grande em torno do valor de equilíbrio.

O método do *ajuste de uma função monotônica* ofereceu excelentes resultados no controle realizado pela RN. O método convergiu em torno de 90% das vezes em que foi solicitado e geralmente com menos de 50 iterações. A divergência geralmente ocorria quando o valor da variável manipulada estava próxima dos limites inferiores e superiores do range. Definimos dois conjuntos de pontos de partida, o primeiro utilizou o último valor calculado da variável manipulada como x_2 (na Tabela 8.2) e usando uma variação em torno do ponto de 0.5V, ou seja, $x_1=x_2-0{,}5$ e $x_3=x_2+0{,}5$. Para o outro conjunto de pontos foi utilizando x_1=*limite superior do range da variável manipulada*, x_2=*valor de equilíbrio do canal da variável manipulada* e x_3=*limite inferior do range da variável manipulada* , sendo que este apresentou melhores resultados que o primeiro conjunto.

De posse dessas informações adotamos o ***método do ajuste de uma função monotônica*** para calcular o valor a ser implantado no processo.

8.2.4 Incorporação de *Bias*

O peso *bias* associado à camada escondida não pode ser especificado utilizando uma ACP (PEEL ET AL, 1992). Entretanto, tem-se mostrado que as redes sem *bias* podem tornar-se incapazes de aproximar certas funções (WRAY E GREEN, 1991). A idéia então, seria treinar a rede usando qualquer técnica padrão utilizando como valores iniciais os pesos determinados pela ACP, procedimento este, que aceleraria a velocidade de treinamento da rede. Neste trabalho não implantamos essa idéia pois nosso objetivo era a comparação entre os métodos de treinamento, e não a aceleração do treinamento, portanto, utilizamos o treinamento ACP sem a conexão de peso *bias*.

8.2.5 Parâmetros Determinados em Grupos

A busca do conjunto de parâmetros ideais para um bom controle pela RN no processo simulado, foi realizada alternando os valores de acordo com a Tabela 8.3, onde, iniciamos com os valores ótimos definidos no treinamento *Steepest Descent*.

Os testes seguiram sempre o seguinte padrão:

1. iniciamos o *software* RTX e esperamos que o processo atingisse o *set-point* de 300ºC
2. aplicamos uma perturbação do tipo PRBS para que o conjunto de dados de treino tivesse valores representativos suficientes para um bom treinamento
3. iniciamos o treinamento Decomposição em Valores Singulares (a duração desse treinamento é um intervalo de amostragem devido o mesmo ser do tipo batelada).
4. iniciamos o controle pela RN
5. determinamos o desempenho do controlador RN através da medida do erro quadrado definido como:

$$E^2 = \int_0^{\infty} e^2 dt \approx \sum_{i=1}^{N_e} e^2(i)\Delta T = \sum_{i=1}^{N_e} (y_i - SP)^2 T_s = T_s \sum_{i=1}^{N_e} (y_i - SP)^2 \qquad (8B)$$

onde; y = valor da variável controlada no instante i

SP = *set-point* do processo T_s= tempo de amostragem

N_e = número de intervalos de amostragem (definido como 13500s=150T_s)

Os resultados obtidos estão descritos nas Tabela 8.3 em ordem de aplicação.

n	**Camada de entrada**		**Janela de observações**	**Grau da polinomial**	E^2 **x** 10^{-5}
	nr	nr com *u*			
25	15	10	10	2	4,43322
25	15	10	30	2	0,61207
25	15	5	10	2	0.68566
25	15	8	10	2	0,44118
25	15	8	10	3	1,02798
25	15	10	10	3	0,11957
25	15	10	50	3	0,37010
20	15	10	20	3	0,73092
20	15	10	10	3	0,58379
25	**21**	**14**	**10**	**3**	**0,00912**
25	21	14	20	3	0,56930
25	21	14	10	4	0,64109
25	21	14	10	2	0.49849
25	24	16	10	3	0,62502
25	21	12	10	3	0,60535
20	21	14	10	3	0,57065
25	21	14	250	2	0,02462
25	15	10	250	2	0,02719
25	15	10	250	3	0,03166
25	9	6	250	3	0,90468
25	21	14	150	3	0,63167
25	21	14	250	3	0,04968
25	21	14	350	3	0,18621
25	1521	14	550	3	0,13045
25	21	14	750	3	0,16040
25	21	14	1000	3	0,41447

nr = número de neurônios *u*= variável manipulada

n =número de intervalos de amostragens para atingir o *set-point* (item 5.1.1)

Tabela 8.3 *– Resultados de Busca de Parâmetros para o Treinamento Decomposição em Valores Singulares*

O conjunto de parâmetros que produziu melhores resultados é o destacado na Tabela 8.3, a princípio, adotamos esse conjunto de parâmetros e procuramos alterar o conjunto de dados de treino para melhorar os resultados produzidos pelo controlador RN (item 8.3), uma vez que E^2 estava muito elevado, na ordem de 10^3.

8.3 Conjunto de Dados de Treino

Procurando determinar o melhor conjunto de dados de treino para a RN, realizamos testes gerando dados com as diversas perturbações oferecidas pelo *software* RTX.

Para o treinamento *Steepest Descent* realizamos os testes descritos a seguir, onde, primeiramente deixamos que a rede acompanhasse o processo até o *set-point*, esperamos que o processo estabilizasse e aplicamos a perturbação. Após o término da perturbação colocamos a RN para controlar o processo. Caso a rede estivesse bem treinada ela controlaria o processo mantendo-o no valor do *set-point*.

O primeiro teste, com uma perturbação do tipo DEGRAU, mostrou que após à aplicação da perturbação a rede neural não conseguiu controlar o processo, seus valores de saída divergiram como mostrado na Figura 8.3.

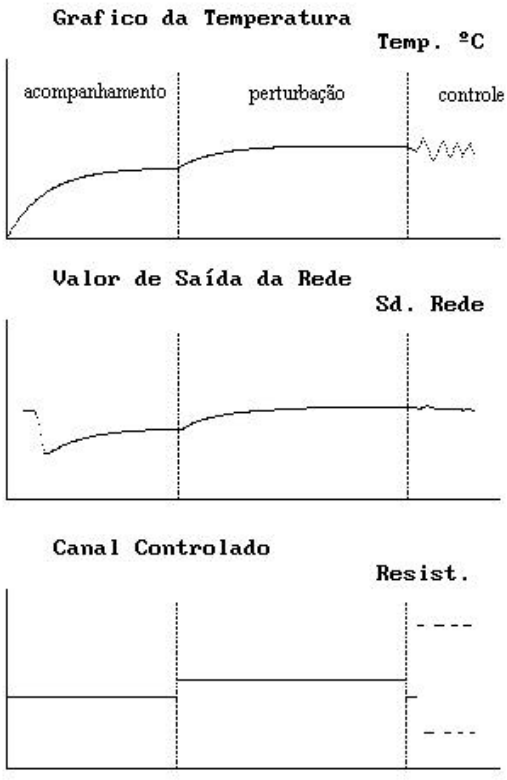


Figura 8.3 - Treinamento *Steepest Descent* com perturbação DEGRAU

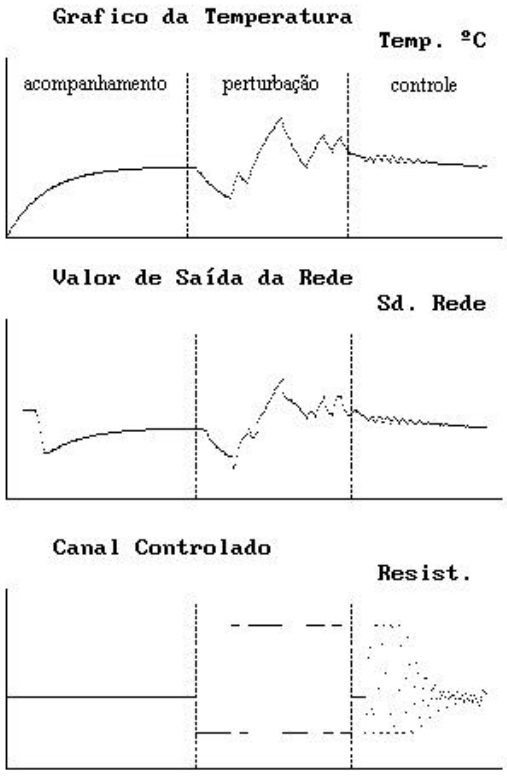


Figura 8.4 - Treinamento *Steepest Descent* com perturbação PRBS

A perturbação do tipo SENÓIDE produziu os mesmos resultados da perturbação degrau, fazendo com que a rede não conseguisse estabilizar o processo.

Com a perturbação do tipo PRBS a rede se mostrou bastante eficiente. Convergiu conseguindo controlar o processo (Figura 8.4).

O treinamento da RN com perturbação PRBS, devido às suas características, se mostrou melhor que o treinamento com outras perturbações. A perturbação PRBS conduz o processo a diferentes situações. A rede treinada com o processo em várias situações, aprende mais rapidamente o comportamento do mesmo, produzindo melhores resultados de saída.

Considerando os resultados obtidos, adotamos na simulação e nos testes do aquecedor de ar, a perturbação PRBS para a geração do conjunto de dados para o treinamento *Steepest Descent*.

No treinamento Decomposição em Valores Singulares inicialmente adotamos o conjunto de dados de treino produzidos por uma perturbação PRBS, considerando os resultados obtidos com o treinamento *Steepest Descent*. Como os resultados não foram satisfatórios, adotamos a perturbação DEGRAU e aplicamos a mesma sequência do item 8.2.5, obtendo os seguintes resultados em ordem de aplicação.

n	**Camada de entrada**		**Janela de observações**	**Grau da polinomial**	E^2 **x** 10^{-4}
	nr	nr com *u*			
25	21	14	1000	3	0,09891
25	21	14	500	3	0,13302
25	21	14	250	3	0,21310
n = número de intervalos de amostragens que o processo deverá demorar para atingir o *set-point* (item 5.1.1) nr = número de neurônios *u* = variável manipulada					

Tabela 8.4 *– Resultados de Busca do Conjunto de Dados de Treino*

Os resultados dos testes mostrados na Tabela 8.4 também produziram valores elevados do E^2, na ordem de 10^3. Na busca por melhores resultados alteramos a sequência de condução dos testes descrita no item 8.2.5, fazendo a inversão do passo 2 com o passo 3, ou seja, primeiro ativamos o treinamento e depois aplicamos a perturbação. Neste caso, a rede foi treinada com o conjunto de dados iniciais, ou seja, a condução inicial do processo até o *set-point* sem a aplicação de uma perturbação.

Para uma janela de observação de $125T_s$ e mantendo os mesmos valores de *n*, *nr*, *u* e grau da polinomial da Tabela 8.4, obtivemos um $E^2=0{,}4\text{x}10^{-3}$, muito inferior aos erros dos testes anteriores. O conjunto de dados iniciais foi suficiente para um bom treinamento, sendo dispensado a aplicação de uma perturbação.

Em face dos resultados obtidos, adotamos como conjunto de dados de treino para o treinamento do tipo Decomposição em Valores Singulares, os valores iniciais de condução do processo ao *set-point*.

A grande vantagem de utilizar-se o conjunto inicial de dados está na aplicação prática, pois nem todos os processo podem ser perturbados para obtermos um conjunto de dados com características necessárias a um bom treinamento.

9. Simulação do Aquecedor de Ar

Depois de realizados os testes de confiabilidade do controlador PID (Item 7.2) e treinamento da rede RN (Capítulo 8) realizamos a simulação no processo de 1ª ordem descrito pela Equação 4E com o intuito de comparar os resultados com testes práticos (Capítulo 4).

Utilizando um tempo de amostragem T_s=90s e o *set-point* de 300ºC, esperamos inicialmente que o controlador levasse o processo ao equilíbrio, para então, aplicarmos a perturbação desejada. No momento em que a perturbação é aplicada, o *software* começa a armazenar os desvios da variável controlada em relação ao *set-point*, ou seja, determina o erro quadrado E^2 definido pela Equação 8B.

Procuramos trabalhar com um número de intervalos (*Ne*) suficientemente grande de modo que o desvio em relação ao *set-point* no final fosse desprezível.

Os valores utilizados que caracterizam cada uma das perturbações aplicada estão descritos na Tabela 9.1. As unidades de tempo são mostradas em segundos e em número de intervalos de amostragem nT_s.

Perturbação	**Tempo de Duração**	**Altura do Salto**	**Tempo de Soma dos Erros**	**Período**	**Duração do Pulso Elementar**	**Intervalo de Aplicação**
DEGRAU	18000s 200 T_s	50% 1,5 V	36000s 400 T_s			
PULSO	90s 1 T_s	50% 1,5 V	21600s 240 T_s			
SEQ. DE PULSOS	7200s 80 T_s	30% 0,9 V	21600s 240 T_s		90s 1 T_s	360s 4 T_s
SENÓIDE	10800s 120 T_s	25% 0,75 V	21600s 240 T_s	1800s 20 T_s		
PRBS	10800 120 T_s	50% 1,5 V	36000s 400 T_s		540s 6 T_s	

Tabela 9.1 *- Caracterização das Perturbações Aplicadas no Processo Simulado*

9.1 Performance do Controlador PID

O desempenho do controlador PID para as diversas perturbações aplicadas está resumido na Tabela 9.2. O erro normalizado (E^2_n), informado na tabela, foi obtido tomando como base o menor E^2 produzido pelo controlador.

Perturbação Aplicada	$E^2 x10^{-2}$	E^2_n
DEGRAU	0,0163	1,0
PULSO	0,0227	1,4
SEQÜÊNCIA DE PULSOS	0,0651	4,0
SENÓIDE	0,4723	29,0
PRBS	6,5044	399,0

Tabela 9.2 *- Desempenho para Diversas Perturbações do Controlador PID*

Como era esperado a performance do controlador PID foi melhor no controle de uma perturbação do tipo degrau, pois o mesmo foi ajustado com os parâmetros de processo extraídos de um teste com esse tipo de perturbação. Na Figura 9.1 podemos acompanhar o comportamento do controlador com o processo simulado recebendo uma perturbação do tipo degrau.

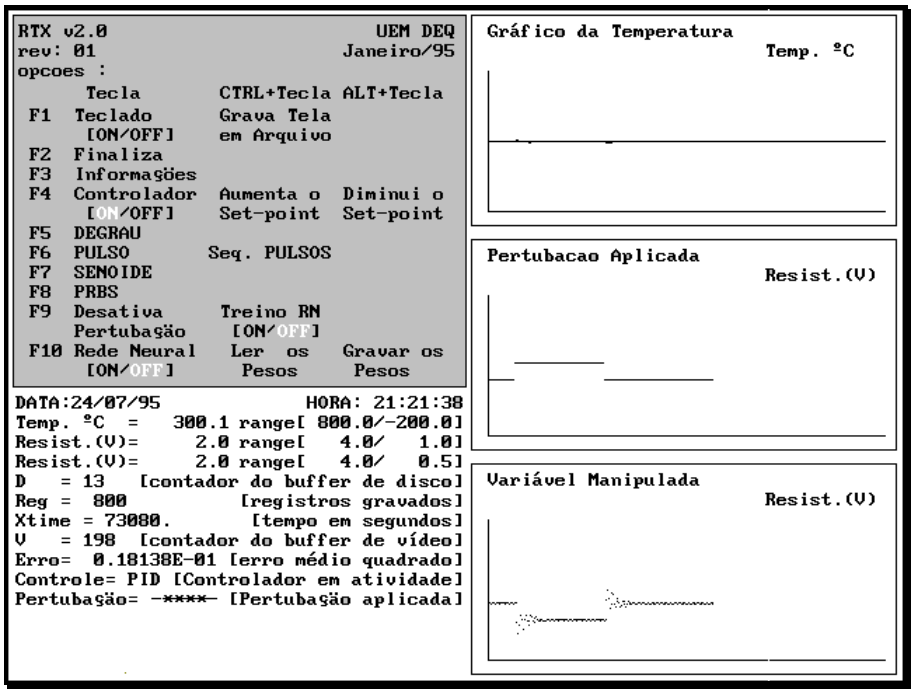


Figura 9.1 - Perturbação DEGRAU com Controlador PID

Para perturbações do tipo pulso e seq*u*ência de pulsos, o controlador PID se mostrou eficiente, mantendo o processo no *set-point*. Através das Figuras 9.2 e 9.3 podemos observar seu comportamento no controle dessas perturbações.

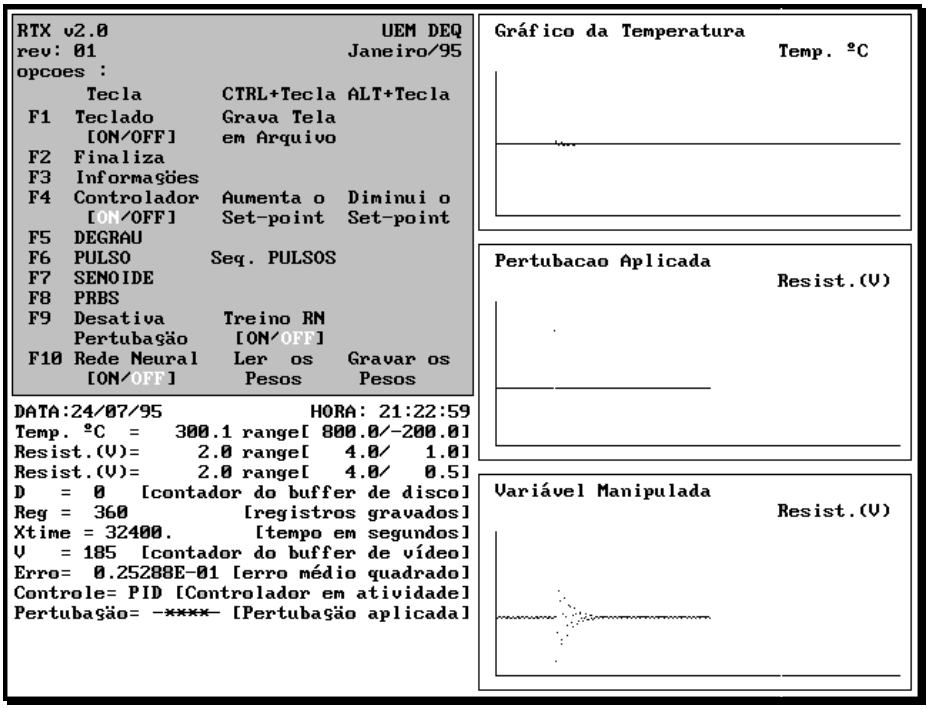


Figura 9.2 - Perturbação PULSO com Controlador PID

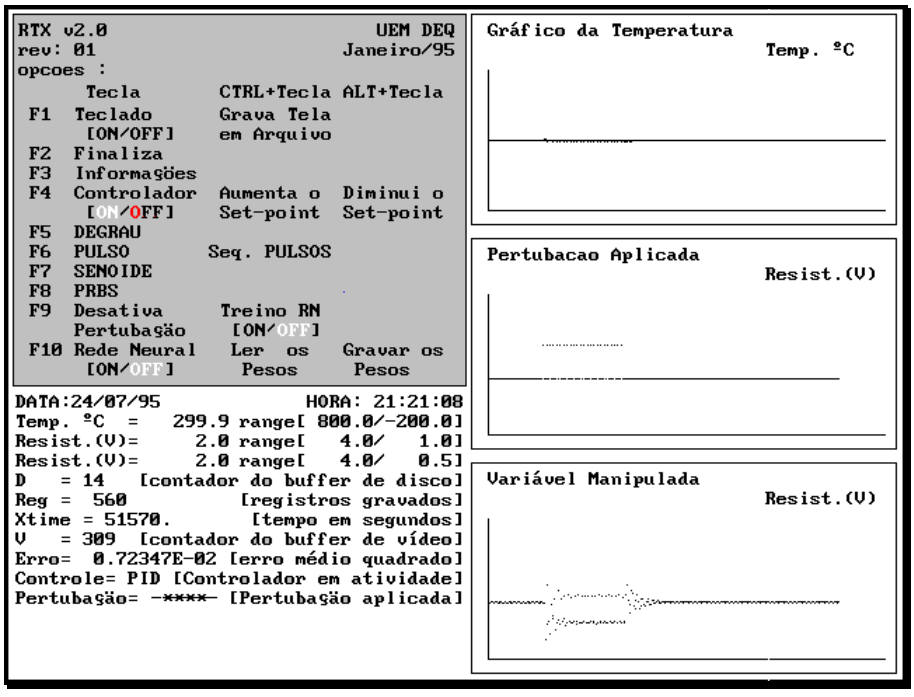


Figura 9.3 - Perturbação SEQÜÊNCIA DE PULSOS com Controlador PID

Com a perturbação senóide, Figura 9.4, observamos que o processo tendeu a acompanhar a perturbação, demonstrando que o PID não estava conseguindo controlar a mesma

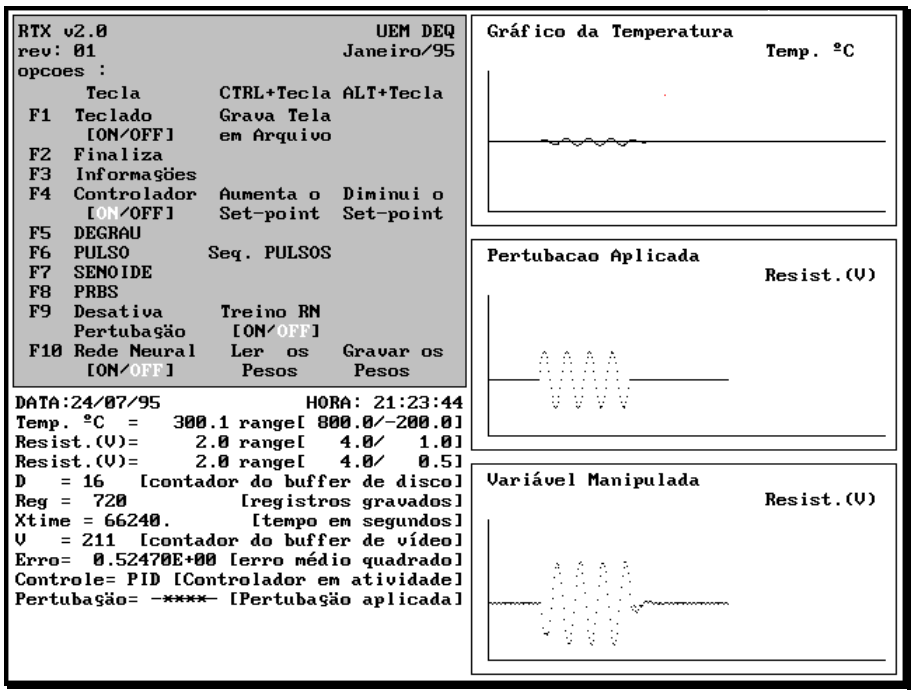


Figura 9.4 - Perturbação SENÓIDE com Controlador PID

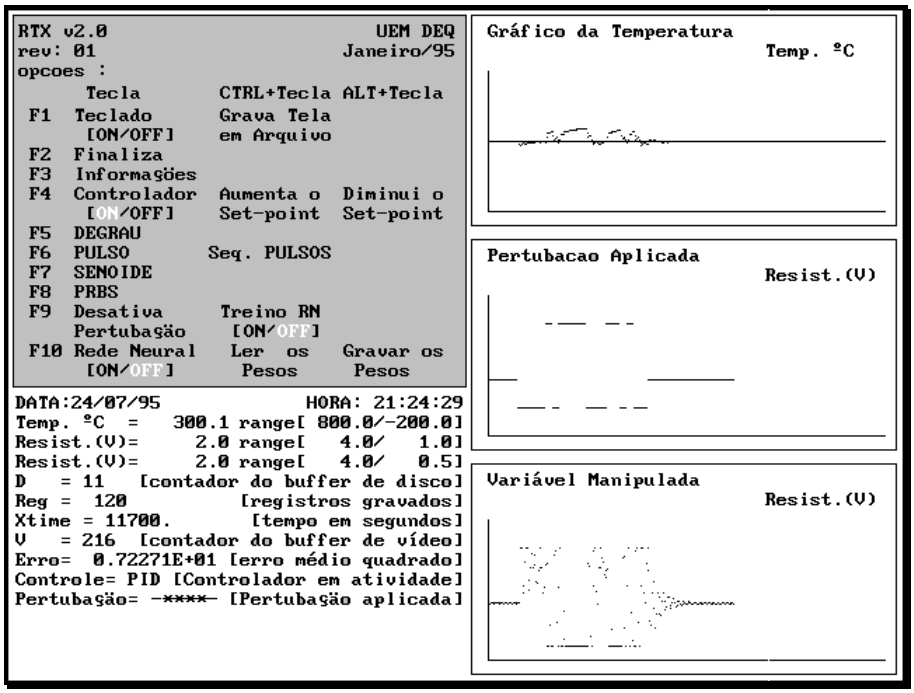


Figura 9.5 - Perturbação PRBS com Controlador PID

O Controlador não conseguiu bom desempenho também frente a uma perturbação do tipo PRBS. Obtivemos o maior desvio da variável controlada em relação ao *set-point*. O E^2 produzido foi 399 vezes superior ao menor erro, o da perturbação degrau.

9.2 Performance do Controlador RN com Treinamento *Steepest Descent*

A Tabela 9.2 mostra os valores dos erros obtidos através do controle simulado com o uso da RN para o treinamento do tipo *Steepest Descent*.

Perturbação Aplicada	E^2x10^{-2}	E^2_n
DEGRAU	0,0075	1,0
SEQÜÊNCIA DE PULSOS	0,0106	1,4
SENÓIDE	0,0214	2,9
PULSO	0,0516	6,9
PRBS	2,2767	303,6

Tabela 9.3 *- Desempenho para Diversas Perturbações do Controlador RN e Treinamento Steepest Descent*

Observamos um bom desempenho do controlador RN, demonstrado pela Figura 9.6, para uma perturbação do tipo degrau, conseguindo amortecer a perturbação e produzindo o

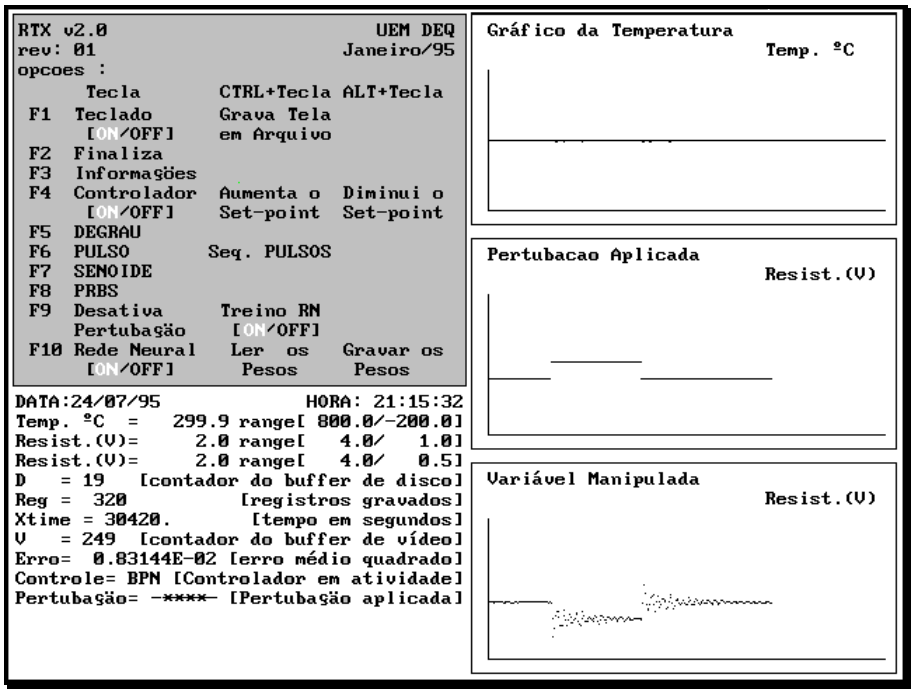


Figura 9.6 - Perturbação DEGRAU com Controlador RN e Treinamento *Steepest Descent*

menor E^2 em relação às outras perturbações.

Com as perturbações sequência de pulsos, senóide e pulso, a rede mostrou desempenho semelhante, mantendo E^2 na mesma potência, conforme podemos visualizar nas Figuras 9.7, 9.8 e 9.9.

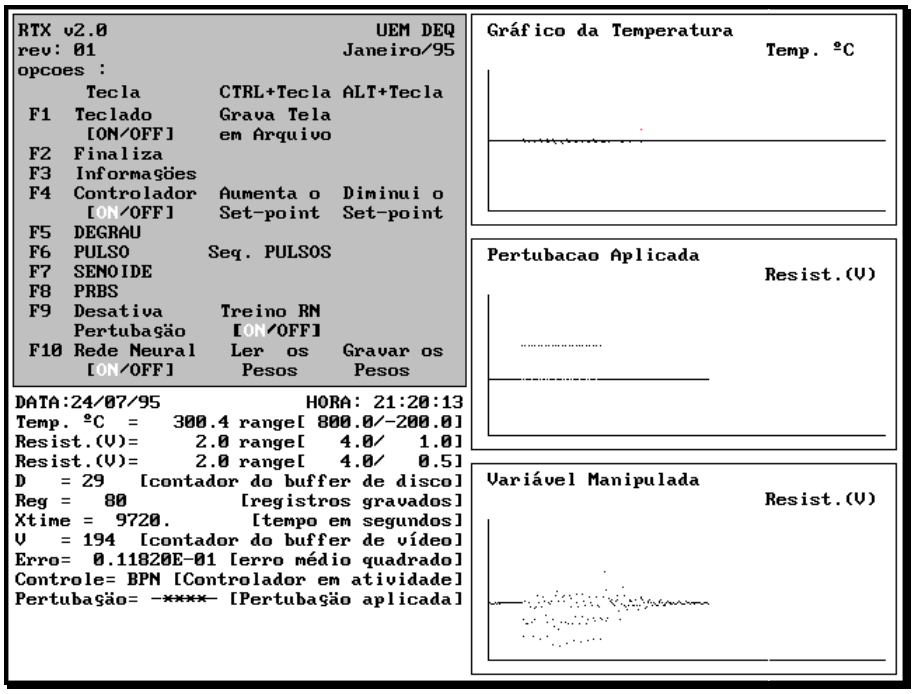


Figura 9.7 - Perturbação SEQÜÊNCIA DE PULSOS com Controlador RN e Treinamento *Steepest Descent*

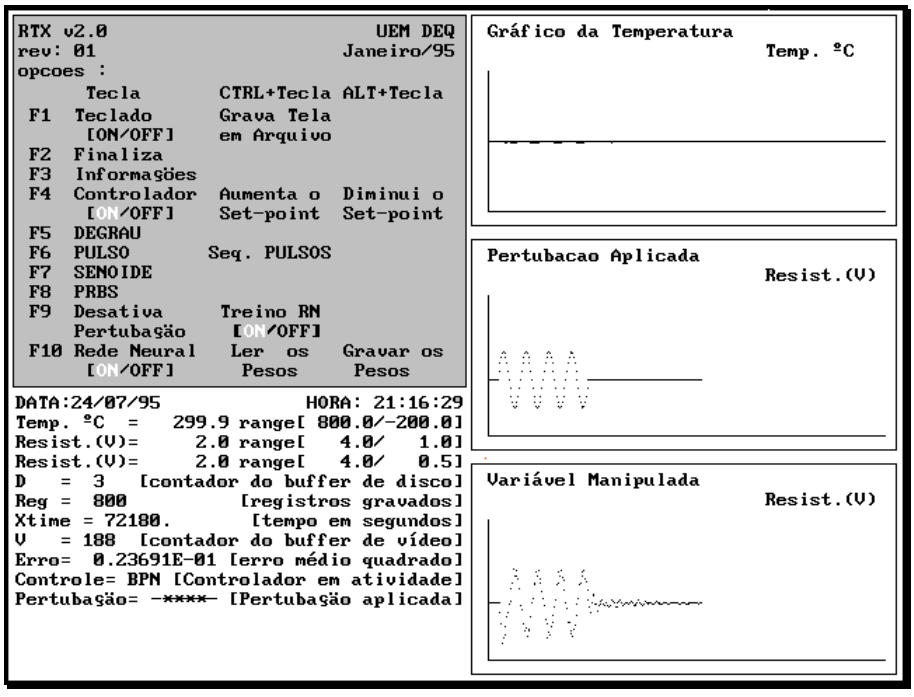


Figura 9.8 - Perturbação SENÓIDE com Controlador RN e Treinamento *Steepest Descent*

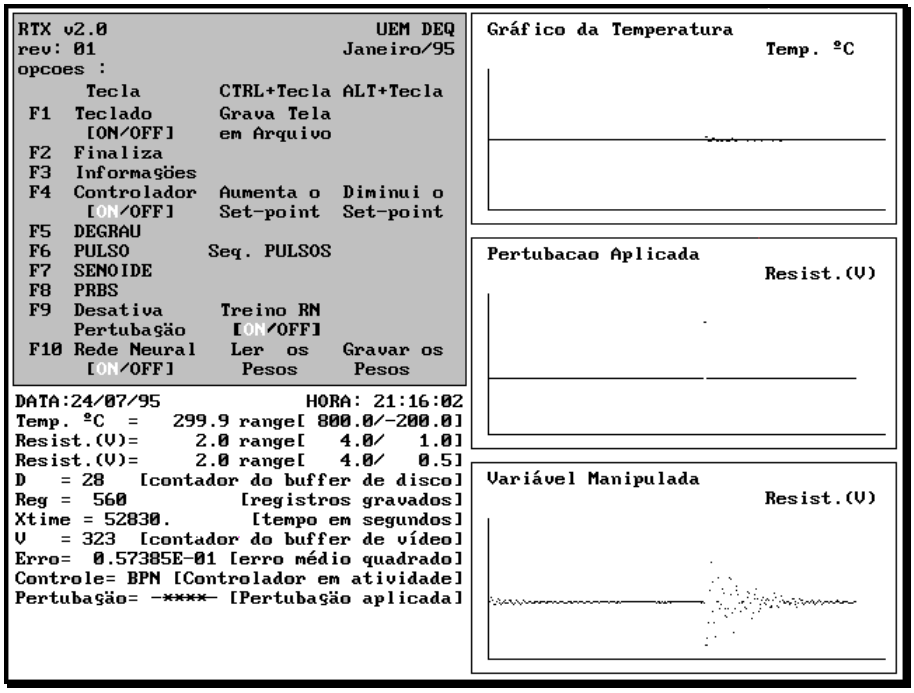


Figura 9.9 - Perturbação PULSO com Controlador RN e Treinamento *Steepest Descent*

Comparando com as outras perturbações, a do tipo PRBS, mostrou-se a mais difícil de ser amortecida pelo controlador RN, provocando um E^2 de grandeza 303 vezes superior ao menor erro, o da perturbação degrau. A Figura 9.10 mostra o comportamento da RN no controle dessa perturbação.

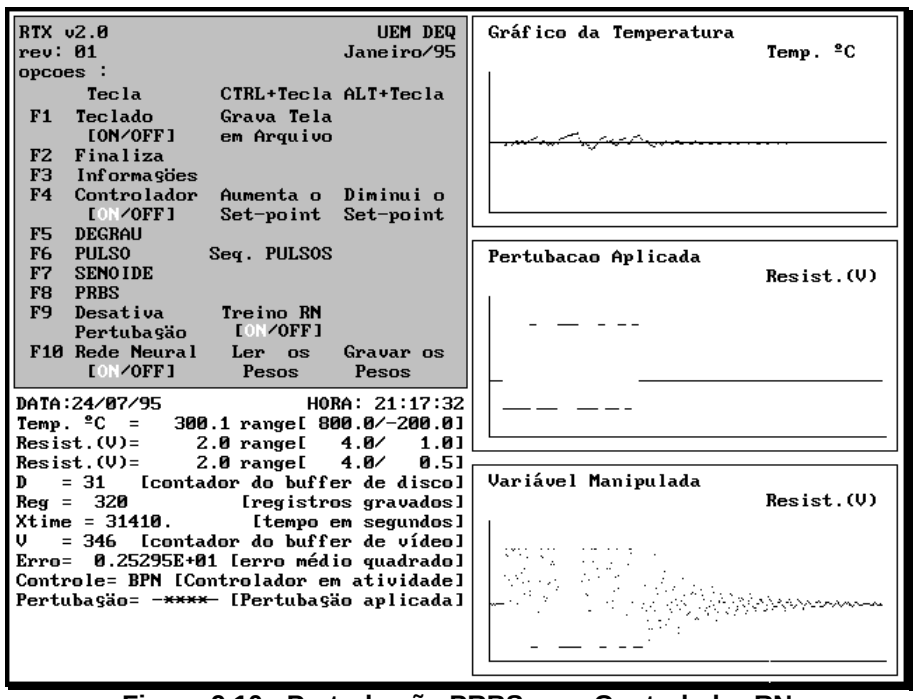


Figura 9.10 - Perturbação PRBS com Controlador RN e Treinamento *Steepest Descent*

9.3 Performance do Controlador RN com Treinamento Decomposição em Valores Singulares

Através dos resultados expressos na Tabela 9.3 podemos verificar o comportamento da RN com o treinamento de Decomposição de Valores Singulares para as diversas perturbações aplicadas.

Perturbação Aplicada	E^2x10^{-2}	E^2_n
SEQÜÊNCIA DE PULSOS	0,0152	1,0
PULSO	0,0386	2,5
DEGRAU	0,0536	3,5
SENÓIDE	0,1407	9,2
PRBS	0,5164	33,9

***Tabela 9.4** - Desempenho para Diversas Perturbações do Controlador RN e Treinamento Decomposição em Valores Singulares*

O melhor desempenho obtido pela RN com treinamento Decomposição em Valores Singulares foi para uma perturbação do tipo sequência de pulsos, onde a perturbação foi bem amortecida como podemos observar na Figura 9.11, entretanto, para as perturbações do tipo pulso e degrau o desempenho foi semelhante resultando em E^2 de mesma potência, podendo ser visualizado nas Figuras 9.12 e 9.13.

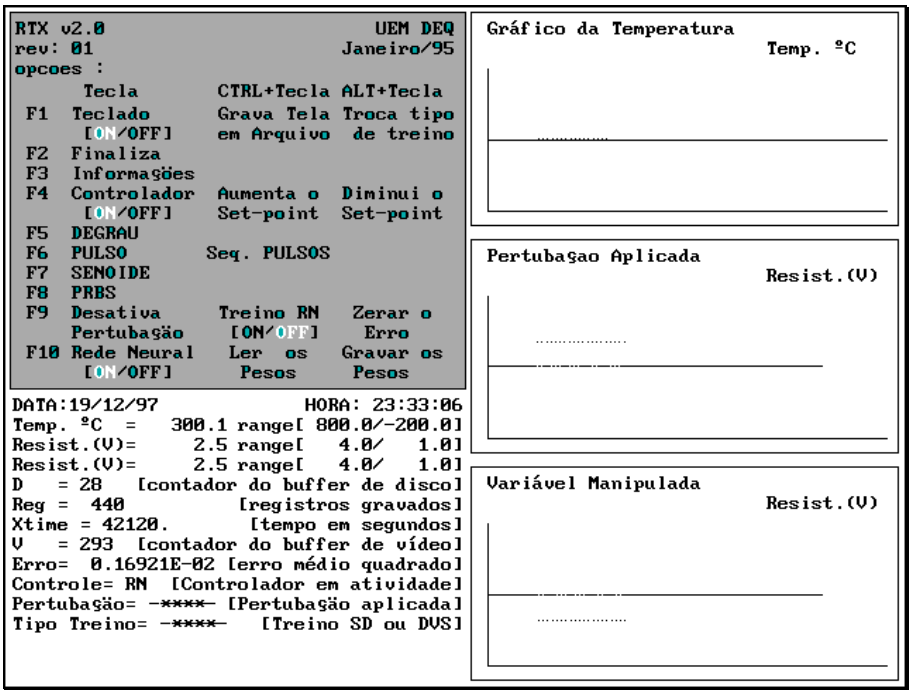


Figura 9.11 - Perturbação SEQUÊNCIA DE PULSOS com Controlador RN e Treinamento Decomposição em Valores Singulares

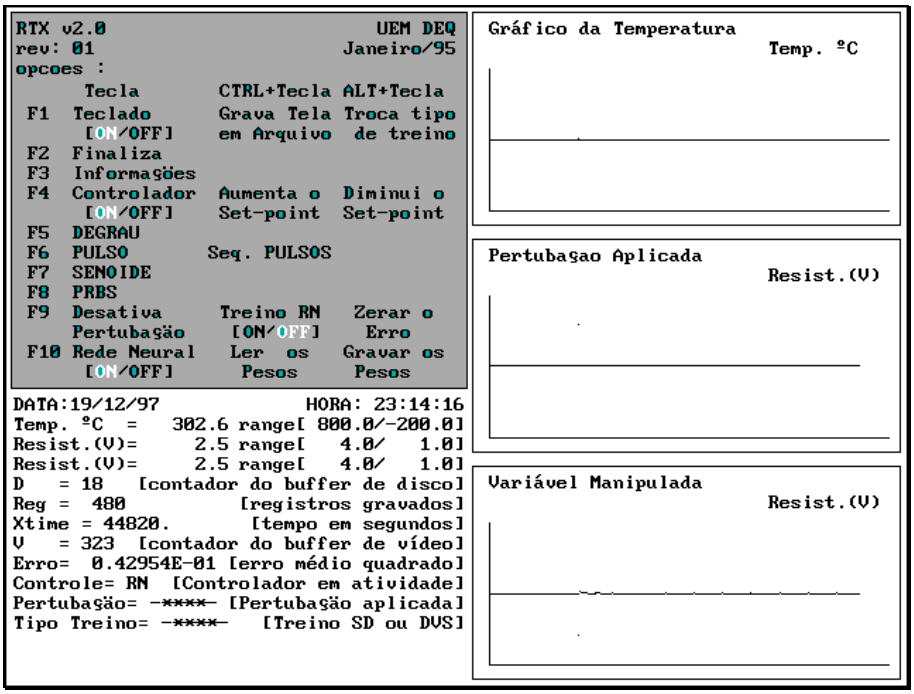


Figura 9.12 - Perturbação PULSO com Controlador RN e Treinamento Decomposição em Valores Singulares

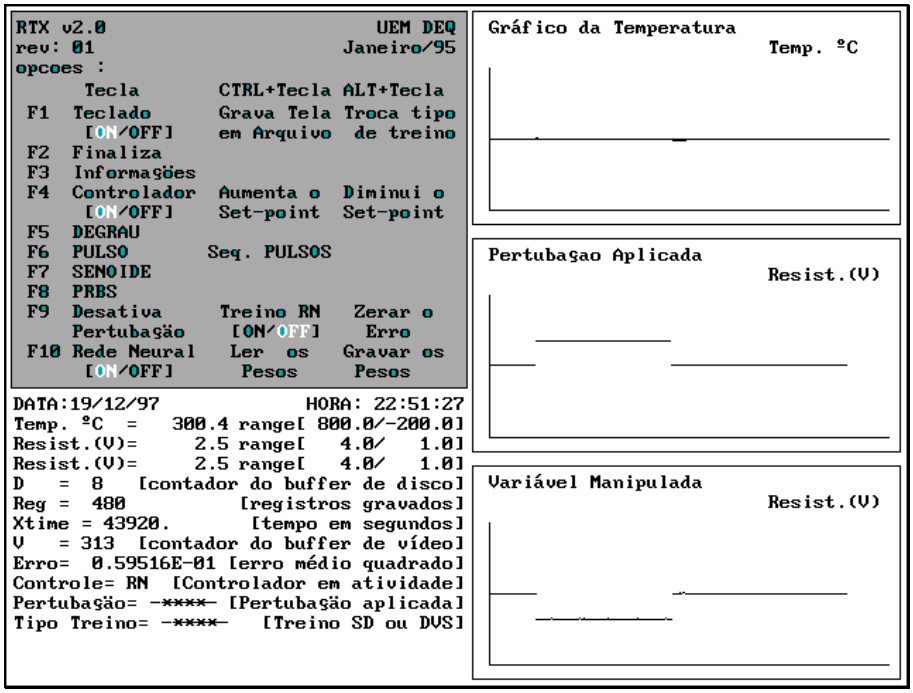


Figura 9.13 - Perturbação DEGRAU com Controlador RN e Treinamento Decomposição em Valores Singulares

O controlador não produziu bons resultados para uma perturbação do tipo senóide como podemos observar na Figura 9.14, ocorreu uma pequena oscilação da variável controlada produzindo E^2 bem superior ao esperado, considerando os resultados obtidos com o treinamento *Steepest Descent* (item 9.2).

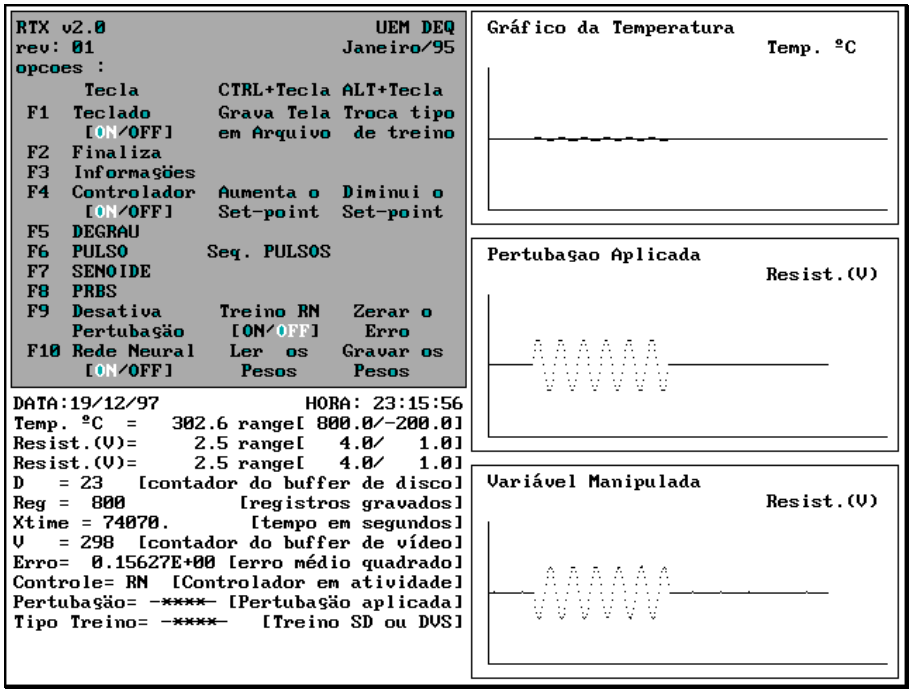


Figura 9.14 – Perturbação SENÓIDE com Controlador RN e Treinamento Decomposição em Valores Singulares

Como era esperado a perturbação PRBS produziu o maior valor de E^2, sendo a mais difícil de ser amortecida (Figura 9.15). Entretanto, o E^2 produzido foi aproximadamente 5 vezes menor que o E^2 produzido com a mesma perturbação e treinamento *Steepest Descent*, demonstrando uma melhora no treinamento e controle da rede.

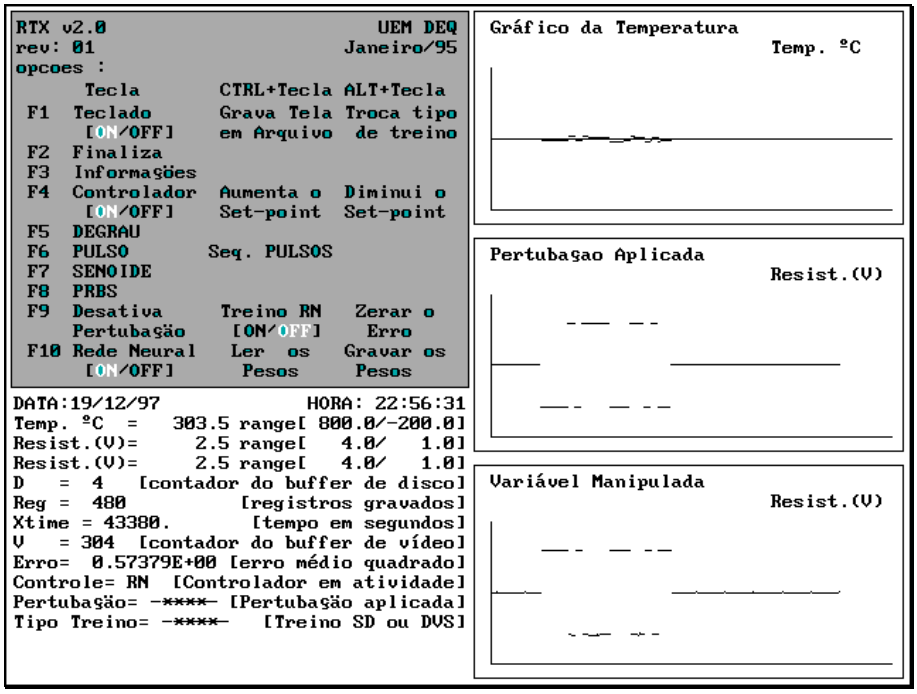


Figura 9.15 - Perturbação PRBS com Controlador RN e Treinamento Decomposição em Valores Singulares

9.4 PID x RN

Uma comparação de desempenho entre os dois controladores pode ser efetuada com base nos E^2 apresentados nos itens anteriores.

De um modo geral, o controlador RN mostrou-se mais eficiente do que o controlador PID para as diversas perturbações aplicadas no sistema independente do tipo de treinamento utilizado. Em quase todas as perturbações aplicadas o E^2 produzido pelo controlador RN foi menor, com exceção da perturbação pulso para ambos os treinamentos e para a perturbação degrau no treinamento Decomposição em Valores Singulares.

Perturbação Aplicada	$E^2 x10^{-2}$			**Razão PID/RN**	
	PID	**RN_{SD}**	**RN_{DVS}**	**PID/RN_{SD}**	**PID/RN_{DVS}**
PULSO	0,0227	0,0516	0,0386	0,44	0,59
DEGRAU	0,0163	0,0075	0,0536	2,17	0,30
PRBS	6,5044	2,2766	0,5164	2,86	12,60
SEQÜÊNCIA DE PULSOS	0,0651	0,0106	0,0152	6,14	4,28
SENÓIDE	0,4723	0,0214	0,1407	22,07	3,36

PID = controlador proporcional integral derivativo

RN_{SD} = controlador RN com treinamento *Steepest Descent*

RN_{DVS} = controlador RN com treinamento Decomposição em Valores Singulares

Tabela 9.5 *- Comparação de Desempenho dos Controladores PID e RN*

Determinando a razão erro PID/RN, definida na Tabela 9.5, procuramos mostrar o quanto maior foi o erro produzido pelo controlador PID, em relação ao erro produzido pelo controlador RN.

Alguns casos chamaram a atenção, o primeiro está no fato que para uma perturbação do tipo pulso o E^2 do controlador RN é aproximadamente 2 vezes maior do que o E^2 do controlador PID para ambos os tipos de treinamento, isto é explicado devido à característica da perturbação, pois, se verificarmos na Figura 9.16, onde temos a aplicação da perturbação pulso sem a ação de controladores, observaremos que $E^2=0,00293x10^2$, ou seja, menor do que o erro produzido com a ação dos controladores. Portanto, para uma perturbação do tipo pulso, quanto menor for a ação do controlador, no sentido de amenizar a perturbação, menor será o

erro produzido na variável controlada, isto é, se o controlador não tomasse nenhuma atitude o erro produzido seria menor. Outro fato interessante está na maior razão PID/RN apresentada. O controlador RN com treinamento *Steepest Descent*, mostrou melhor desempenho no controle de uma perturbação do tipo senóide, em relação ao controlador PID, produzindo um E^2 com valor 22 vezes menor.

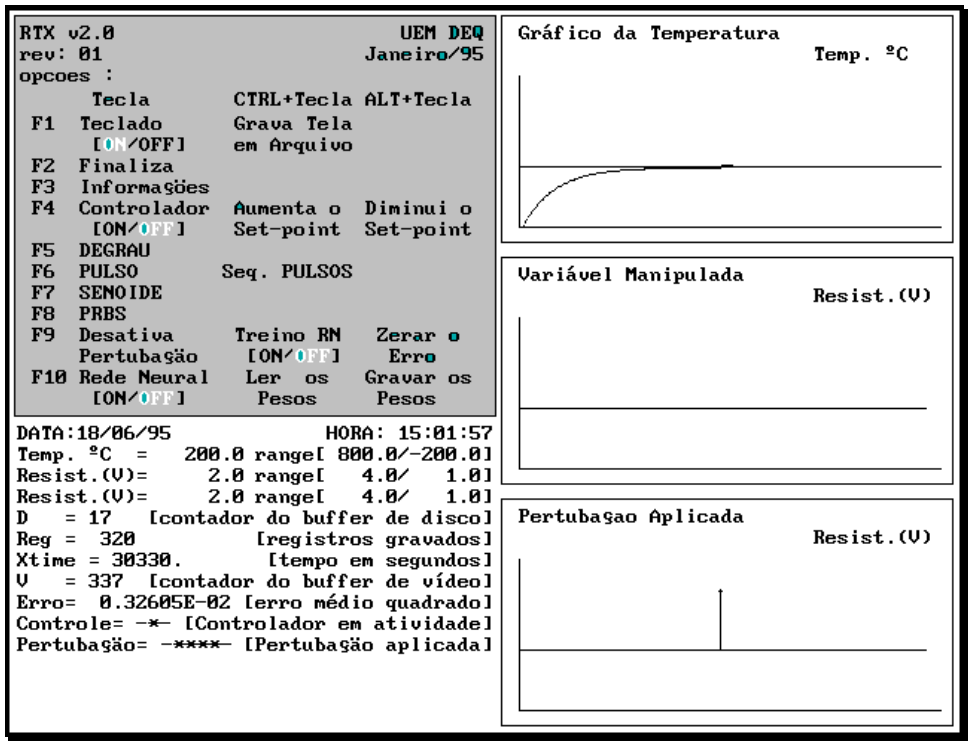


Figura 9.16 – Resposta do Sistema Simulado a uma Perturbação Pulso

É importante observar o baixo E^2 produzido pelo controlador RN com treinamento Decomposição em Valores Singulares para uma perturbação do tipo PRBS, sendo 12 vezes menor que o E^2 produzido pelo PID.

Para uma melhor visualização dos resultados temos o gráfico da Figura 9.17, construído a partir da Tabela 9.5, onde mostra como o controlador RN ofereceu melhores desempenhos do que o controlador PID no processo simulado.

9.5 RN com Treinamento *Steepest Descent* x RN com Treinamento Decomposição em Valores Singulares

Para uma melhor análise comparativa dos resultados dos dois tipos de treinamento implementados no *software* RTX, definimos a razão RN_{SD}/RN_{DVS} entre os E^2 produzidos nos testes. Os valores estão expressos na Tabela 9.6.

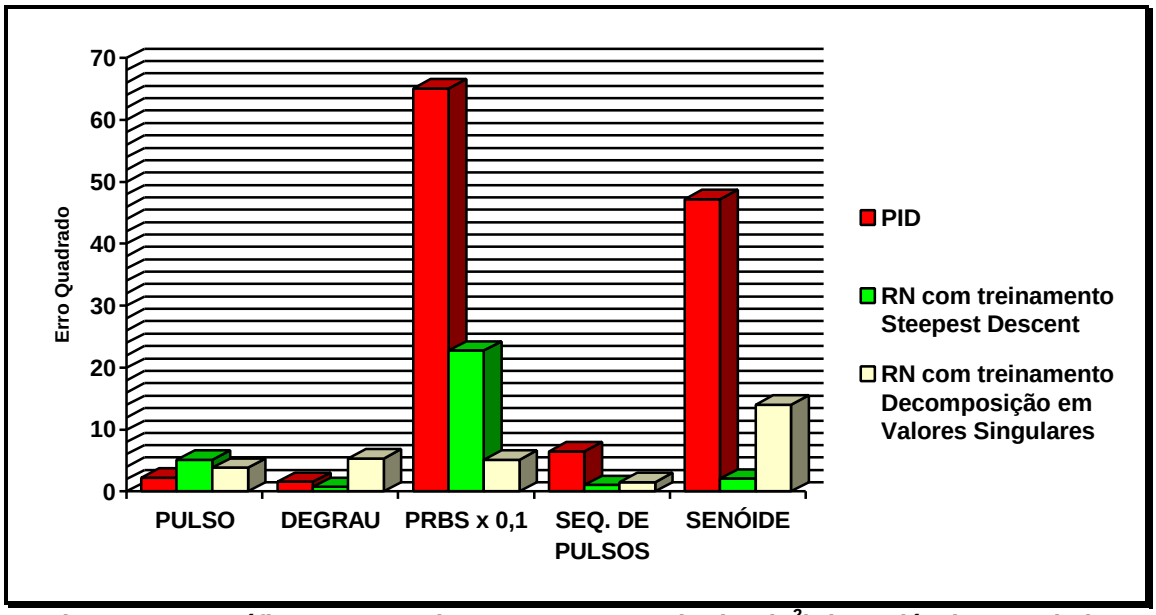


Figura 9.17 - Gráfico Comparativo Dos Erros Quadrados (E^2) da Variável Controlada

Apesar dos resultados com o treinamento Decomposição em Valores Singulares terem produzidos em três casos, erros relativos superiores ao treinamento *Steepest Descent*, observamos que a grande vantagem do método reside no fato de que não há necessidade de aplicação de uma perturbação para se efetuar o treinamento. Esse fator é importante, pois na prática muitos processos não poderão ser excitados de forma a produzirem pontos com valores significativos para treinamento, como os requeridos pelo treino *Steepest Descent*.

Perturbação Aplicada	E^2x10^{-2}		**Razão**
	RN_{SD}	**RN_{DVS}**	**RN_{SD}/RN_{DVS}**
PULSO	0,0516	0,0386	1,34
DEGRAU	0,0075	0,0536	0,14
PRBS	2,2766	0,5164	4,41
SEQÜÊNCIA DE PULSOS	0,0106	0,0152	0,70
SENÓIDE	0,0214	0,1407	0,15
RN_{SD} = controlador RN com treinamento *Steepest Descent* RN_{DVS} = controlador RN com treinamento Decomposição em Valores Singulares			

Tabela 9.6 *- Comparação de Desempenho dos Treinamentos Steepest Descent e Decomposição em Valores Singulares*

10. Testes no Aquecedor de Ar

Nos testes do aquecedor de ar utilizamos alternadamente os dois controladores ajustados, PID e RN, determinando assim, a performance de cada um no controle de perturbações externas, ou seja, no controle da variação da vazão de ar no aquecedor.

Para o controlador RN realizamos testes para os dois tipos de treinamentos implantados na simulação do processo, o *Steepest Descent* e Decomposição em Valores Singulares.

Para uma análise quantitativa, definimos o erro relativo quadrado da variável controlada como sendo:

$$E^2_{relativo} = \int_0^{\infty} e^2 dt \approx \sum_{i=1}^{N_e} e^2(i)\Delta T = \sum_{i=1}^{N_e} \left(\frac{y_i - SP}{SP}\right)^2 T_s = \frac{T_s}{SP^2} \sum_{i=1}^{N_e} (y_i - SP)^2 \qquad (10A)$$

onde;

Ne = número de intervalos de amostragem de soma dos erros

y_j = valor da variável controlada no intervalo j

SP = valor do *set-point*.

T_s = tempo de amostragem

O valor de *Ne* para os testes foi definido, suficientemente grande, de forma que os erros relativos finais fossem desprezíveis.

10.1 Performance do Controlador PID

O teste de performance do controlador PID está representado na Figura 10.1, onde, definimos o tempo de amostragem *Ts*=15s, o *set-point* em 200ºC e iniciamos a partida no processo com o controlador em atividade. Nos primeiro instantes, como era esperado, o controlador alterou a variável manipulada para um valor elevado, de modo que a temperatura do processo aumentasse. Monitorando a variável manipulada o controlador mostrou-se eficiente no ajuste do *set-point*, o que foi conseguido em aproximadamente 40 minutos, tempo inferior ao que se conseguiu manualmente nos testes de caracterização (Capítulo 4). Após o sistema permanecer em equilíbrio, aplicamos o degrau na vazão de ar abrindo a porta do aquecedor, diminuindo assim, a resistência à passagem do fluxo de ar e consequentemente,

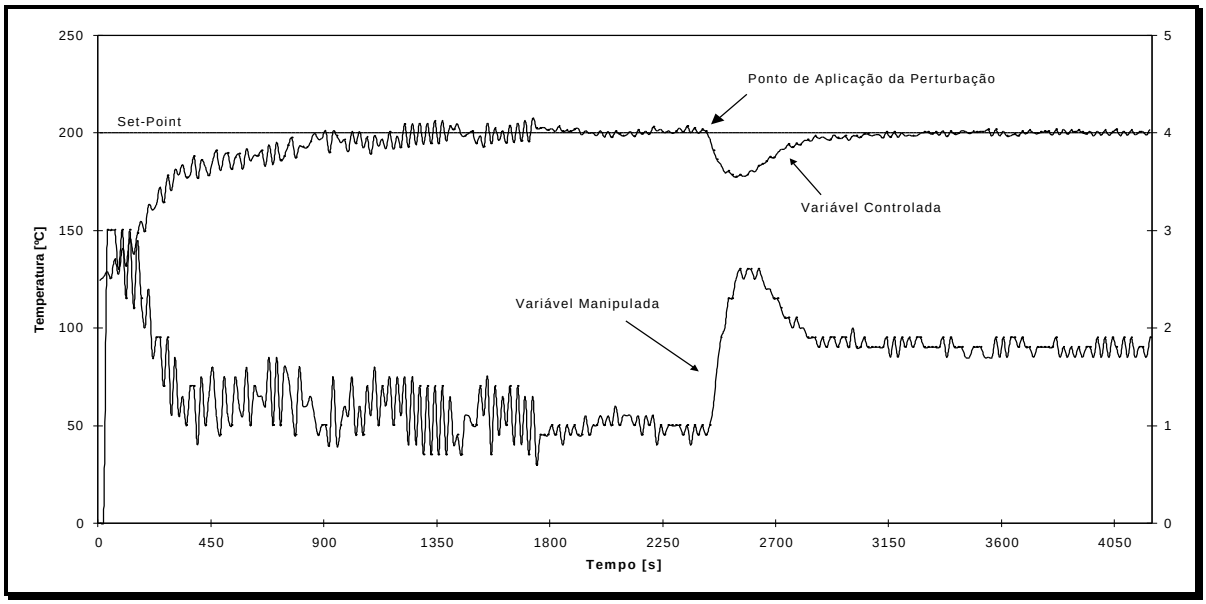


Figura 10.1 – Testes no Aquecedor de Ar com Controlador PID

aumentando a vazão. O que observamos, foi uma atuação satisfatória do controlador, conseguindo amortecer a perturbação aplicada ao sistema conduzindo-o ao seu *set-point*, produzindo $E^2_{relativo} = 3{,}5420$.

10.2 Performance do Controlador RN com Treinamento *Steepest Descent*

O treinamento *Steepest Descent* da rede, para o controle do módulo de testes, foi realizado em duas etapas. Na primeira etapa, treinamos a rede em *modo simulado*, com o processo de 1ª ordem definido no Capítulo 8, utilizando uma perturbação PRBS. A segunda etapa, consistiu em ajustar os pesos da rede com o processo real (módulo de testes), sendo que, os pesos foram inicializados com os valores calculados na primeira etapa. Não utilizamos a perturbação PRBS diretamente no módulo de testes, procurando uma alternativa para processos que não podem sofrer esse tipo de perturbação, por exemplo, controle de pH em sistemas bioquímicos. O treinamento na segunda etapa ficou resumido às oscilações provocadas pelo controle nas primeiras 7 horas.

Podemos observar a performance do controlador RN com treinamento *Steepest Descent* através da Figura 10.2, onde utilizamos um tempo de amostragem *Ts*=90s, e o *set-point* de 350ºC. O aumento inicial da temperatura, atingindo 519 ºC, revela que o conjunto inicial de pesos não era o ideal, obrigando a rede a corrigi-los, comprovando que a suposição de sistema de 1ª ordem igual ao simulado não estava totalmente correta.

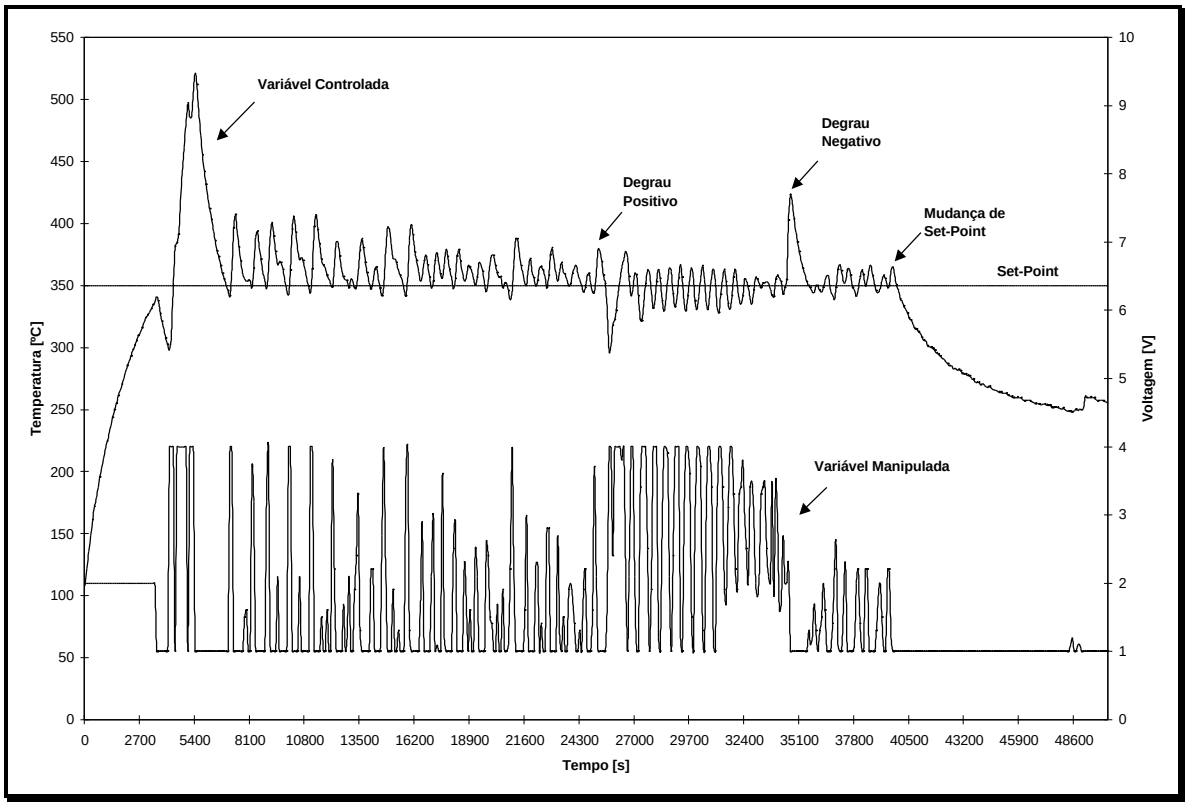


Figura 10.2 - Testes no Aquecedor de Ar com Controlador RN e Treinamento *Steepest Descent*

10.2.1 Perturbação DEGRAU na Vazão de Ar

Após 7 horas de teste, consideramos que o processo tenderia ao *set-point* e aplicamos uma perturbação do tipo degrau na vazão de ar, abrindo a porta de saída de ar do aquecedor. Este instante é descrito na Figura 10.2 como *degrau positivo*.

Observamos que o controlador RN conseguiu amortecer a perturbação, conduzindo o processo ao *set-point* e gerando $E^2_{relativo}$ = 6,2041. O tempo total do teste foi de 2 h e 30 minutos.

O *degrau negativo* mostrado na Figura 10.2 representa o instante em que fechamos a porta de saída de ar do módulo, aumentando a resistência à passagem do fluxo de ar e consequentemente diminuindo a vazão. Observamos que o controlador RN conseguiu amortecer essa perturbação mais rapidamente do que a anterior, no entanto, obtivemos $E^2_{relativo}$ = 7,7483. Um valor mais elevado do que o do primeiro teste. Isto ocorreu devido às características do módulo de testes, descritas no Capítulo 4, onde existe diferença ente o aquecimento e o resfriamento.

10.2.2 Alteração do *SET-POINT* do processo

O teste de alteração do *set-point* do processo teve como objetivo verificar a facilidade da alteração via *software*, bem como o comportamento do controlador RN, pois o mesmo havia sido treinado para operar em condições diferentes.

Optamos por uma alteração no *set-point* para um valor menor, ou seja 250 ºC, já que o controlador RN se mostrou menos eficiente no resfriamento, do ponto de vista dos testes degrau. Os resultados obtidos, representados na Figura 10.2, foram satisfatórios, demonstrando que o controlador, apesar de treinado em níveis diferentes, conseguiu assimilar a mudança de *set-point*, levando o processo à sua nova condição em um tempo de 2 horas e 57 minutos para uma variação de 100ºC no *set-point.*

10.3 Performance do Controlador RN com Treinamento Decomposição em Valores Singulares

Nos testes do aquecedor de ar procuramos treinar a rede de maneira similar ao treinamento realizado na simulação. Iniciamos o teste com um valor de 2,0 Volt na variável manipulada e mantivemos esse valor até que a variável controlada chegasse ao valor de 300ºC, o que ocorreu em aproximadamente 3 horas e 30 minutos, em seguida, realizamos o treinamento da rede com esse conjunto de dados no instante indicado na Figura 10.3 como *ponto do 1º treinamento*. Imediatamente após realizado o treinamento ativamos o controle do processo através da rede neural. Utilizamos nos testes o tempo de amostragem T_s=90s, e o valor do *set-point* de 300^{o}C.

Inicialmente esperamos que o controlador mantivesse o processo em torno do *set-point*. Após esse período consideramos o processo treinado e iniciamos os testes. Como o resultado do primeiro teste com a perturbação degrau produziu um erro relativamente alto, após 6 horas e 25 minutos de início do teste realizamos um novo treinamento na rede para a continuação dos demais testes. Na Figura 10.3 esse instante está indicado como *ponto do 2º treinamento*.

10.3.1 Perturbação DEGRAU na Vazão de Ar

Com o processo considerado estável, após 3 horas e 36 minutos, aplicamos a perturbação na vazão de ar abrindo a porta do aquecedor, sendo esse instante indicado na Figura 10.3 como *degrau positivo*.

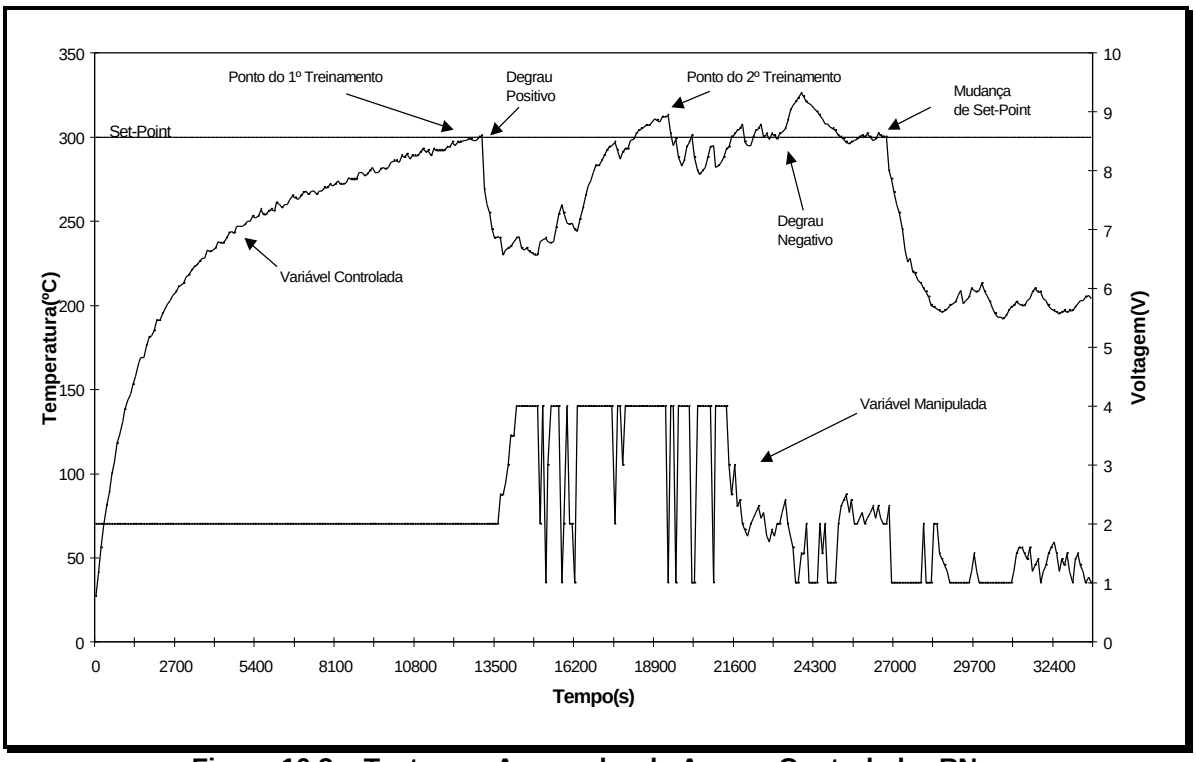


Figura 10.3 – Testes no Aquecedor de Ar com Controlador RN e Treinamento Decomposição em Valores Singulares

O controlador RN procurou estabilizar o processo em seu *set-point*, como podemos observar na Figura 10.3. O tempo necessário para conduzir novamente o processo ao seu *set-point* foi de 1 hora e 48 minutos, produzindo $E^2_{relativo} = 48{,}5770$.

O comportamento do controlador durante a aplicação do *degrau negativo*, conforme mostrado na Figura 10.3, foi satisfatório, pois, o controlador conseguiu amenizar a perturbação em 57 minutos e produziu $E^2_{relativo} = 1{,}4447$, bem abaixo do erro produzido com o degrau positivo. Esse fato demonstra que o primeiro treinamento não foi suficiente para que a rede aprendesse o processo de modo a realizar um controle eficiente.

10.3.2 Alteração do *SET-POINT* do Processo

O teste de alteração do *set-point* foi conduzido de maneira similar ao realizado com treinamento *Steepest Descent* (item 10.2.2), como o objetivo é verificar a facilidade de mudança de ponto operacional do processo alteramos o *set-point* para 200ºC, esse instante está indicado na Figura 10.3 como *mudança de set-point*. O tempo necessário para o processo se estabilizar no novo *set-point* foi de 3 horas e 6 minutos para uma variação de 100ºC.

10.4 PID x RN

No módulo de testes o controlador PID mostrou um melhor desempenho em relação ao controlador RN, no controle de um degrau positivo na vazão de ar. Isto pode ser observado através da razão PID/RN (Tabela 10.1) para cada tipo de treinamento da rede.

A diferença entre o PID e o controlador RN com treinamento *Steepest Descent* pode ser resultado de pouco tempo de utilização da rede em treinamento, já que a mesma, treinada de modo sequencial, produz melhores resultados com o decorrer do tempo.

O erro produzido pelo controlador RN com treinamento Decomposição em Valores Singulares foi aproximadamente 4 vezes superior ao erro produzido pelo controlador PID, demonstrando que o treinamento inicial da rede não foi suficiente para que a mesma aprendesse o processo.

Perturbação Aplicada	$E^2_{relativo}x10^{-2}$			**Razão PID/RN**	
	PID	**RN_{SD}**	**RN_{DVS}**	**PID/RN_{SD}**	**PID/RN_{DVS}**
Aumento da vazão de ar	0,0354	0,0620	0,4858	0,57	0,07

PID = controlador proporcional integral derivativo

RN_{SD} = controlador RN com treinamento *Steepest Descent*

RN_{DVS} = controlador RN com treinamento Decomposição em Valores Singulares

Tabela 10.1 *- Comparação de Desempenho dos Controladores PID e RN no Módulo de Teste*

10.5 RN com Treinamento *Steepest Descent* x RN com treinamento Decomposição em Valores Singulares

A comparação do desempenho dos dois métodos de treinamento empregados nos testes, pode ser realizada através de E^2 expresso na Tabela 10.2. Devemos lembrar no entanto, que para o treinamento Decomposição em Valores Singulares houve duas etapas de treinamento, a primeira antes da aplicação da perturbação *degrau positivo na vazão de ar* e a

Perturbação Aplicada	$E^2_{relativo}x10^{-2}$		**Razão**
	RN_{SD}	**RN_{DVS}**	**RN_{SD}/RN_{DVS}**
Degrau positivo na vazão de ar	0,0620	0,4858	0,13
Degrau negativo na vazão de ar	0,0775	0,0145	5,34

RN_{SD} = controlador RN com treinamento *Steepest Descent*

RN_{DVS} = controlador RN com treinamento Decomposição em Valores Singulares

Tabela 10.2 *- Comparação de Desempenho dos Treinamentos Steepest Descent e Decomposição em Valores Singulares no Módulo de Teste*

segunda antes da aplicação da perturbação *degrau negativo na vazão de ar*.

Definimos a razão RN_{SD}/RN_{DVS} para uma melhor comparação do desempenho dos métodos de treinamento.

O controlador RN com treinamento *Steepest Descent* mostrou melhor resultado no controle da perturbação degrau positivo, enquanto o controlador RN com treinamento Decomposição em Valores Singulares foi melhor no controle da perturbação degrau negativo.

O fato do controlador RN com treinamento Decomposição em Valores Singulares apresentar melhores resultados somente após a aplicação da segunda etapa de treinamento, demonstra que a primeira etapa de treinamento não foi suficiente para um bom ajuste dos pesos de modo a oferecer um bom controle.

11. Teste x Simulação

A comparação quantitativa entre o teste e a simulação do aquecedor de ar, pode ser efetuada utilizando os dados apresentados nas Tabelas 9.5 e 10.1, entretanto, podemos comparar somente os valores obtidos pela perturbação DEGRAU, porque as demais perturbações não foram aplicadas no módulo de teste.

Como os resultados na simulação estão na forma de E^2, definido pela Equação 8B, os mesmos foram transformados em $E^2_{relativo}$ para efeito de comparação.

Utilizando a Equação 10A podemos escrever;

$$E^2_{relativo} = \frac{T_s}{SP^2}\sum_{j}^{N_e}(y_j - SP)^2 = \frac{E^2}{SP^2} \qquad (11A)$$

onde *SP* é o *set-point* do processo. A Tabela 11.1 expressa os $E^2_{relativo}$ obtidos na simulação e nos testes para uma perturbação DEGRAU.

Controlador	$E^2_{relativo} x10^{-1}$		**Razão Teste/Simulação**
	Teste	Simulação	
PID	0,3542	0,0164	21,60
RN_{SD}	0,6204	0,0077	80,57
RN_{DVS}	4,8577	0,0545	89,13
PID = controlador proporcional integral derivativo RN_{SD} = controlador RN com treinamento *Steepest Descent* RN_{DVS} = controlador RN com treinamento Decomposição em Valores Singulares			

Tabela 11.1 *- Comparação entre Teste e Simulação para os Controladores PID e RN*

A relação Teste/Simulação apresentada na Tabela 11.1 demonstra que os controladores, PID e RN, tiveram comportamento semelhante quanto ao desempenho entre teste e simulação. Na simulação, utilizando o processo de 1ª ordem, os controladores apresentaram resultados muito superiores aos produzidos no teste.

Os resultados obtidos já eram esperados, uma vez que o módulo possuía características diferentes relativas ao aquecimento e resfriamento (item 4.1), de difícil implantação em processo simulado.

Em ambos os casos, teste e simulação, com o treinamento *Steepest Descent*, onde a rede é treinada sequêncialmente, observamos que a medida que o tempo avança os resultados fornecidos melhoram, demonstrando que a rede realmente vai "conhecendo" o processo.

12. Conclusões e Observações

Os resultados obtidos foram melhores do que o esperado, acreditávamos no potencial das redes neurais como controlador, mas não tínhamos dados quantitativos de desempenho para compará-las com controladores PID, com este trabalho realizamos tais comparações.

Através da simulação demonstramos que uma RN, ajustada e treinada, pode oferecer melhores resultados que um controlador PID. Acreditamos, na grande possibilidade das redes também oferecerem melhores performances nos processos reais, devemos no entanto, olhar atentamente o treinamento.

A dificuldade da implantação do controle por redes neurais reside no conjunto de parâmetros a serem ajustados, como existe uma certa dependência entre eles é difícil estabelecer um conjunto ótimo para um bom desempenho. Os valores referenciados neste trabalho são válidos somente para o problema estudado, não havendo uma relação de comparação para outros processos.

Sabendo da alta performance dos computadores existentes hoje em dia, a limitação da quantidade de neurônios nas camadas da rede (item 8.1.2), devido ao esforço computacional, talvez fique sem sentido. Observamos que durante um intervalo de amostragem o computador é capaz de realizar o processamento de um número muito maior de neurônios, o que poderia melhorar o desempenho da rede.

Na continuidade deste trabalho poderíamos utilizar uma união dos dois tipos de treinamento implantados. Através do treinamento Decomposição em Valores Singulares determinaríamos os pesos iniciais para o treinamento *Steepest Descent*, com essa proposta estaríamos certamente acelerando o treinamento *Steepest Descent* e evitando a excitação inicial do processo para coletar pontos para esse treinamento. A grande dificuldade de implantação dessa idéia está na determinação dos pesos da camada escondida, pois teríamos que resolver um sistema de equação para cada peso a ser determinado.

Outras opções de continuidade poderiam ser; a implantação de um treinamento com passo de aprendizagem (η) variável, utilizar outro modelo de rede (várias camadas intermediárias) e realizar testes em outros processos.

13. Bibliografia

1. **ASTRÖM, K. J. ; B. W. WITTENMARK**: *Computer Controlled Systems: Theory and Design*, Printice-Hall, 1990.

2. **BHAT, N.; T. J. McAVOY:** *Determining Model Structure for Neural Models by Network Stripping*, Computers Chem. Eng. Vol 16, Nr. 4, pgs 271-281,1992.

3. **BHAT, N.; T. J. McAVOY:** *Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems*, Computers Chem. Eng. Vol 14, Nr. 4/5, pgs 573-583, 1990.

4. **COUGHANOWR, D. R.; L. B. KOPPEL**: *Process Systems: Analysis and Control*, McGraw-Hill, 1965.

5. **COULSON, J. M.; J. F. RICHARDSON; D. G. PEACOCK**: *Chemical Engineering: Volume Three: Chemical Reactor Design, Biochemical Reactor Engineering including Computational Techniques and Control*, Pergamon Press, 1979.

6. **DAHHOU B.; M. LAKRORI; I. QUEINNEC; E. FERRET; A. CHÉRUY**: *Control of a Continuous Fermentation Process*, Journal Proc. Cont. Vol 2, Nr. 2, pgs 103-111, 1992.

7. **DAYAL, B. S.; F. A. TAYLOR; J. F. MACGREGOR**: *The Design of Experiments, Training and Implementation of Nonlinear Controllers Based on Neural Networks*, The Canadian J. Chem. Eng. 72, pgs 1066-1074, 1994.

8. **Di MASSIMO, C.; G. A. MONTAGUE; M. J. WILLIS; M. T. THAM; A. J. MORRIS**: *Towards Improved Penicillim Fermentation Via Artificial Neural Networks*, Comp. Chem Eng. Vol. 16, Nr. 4, pgs 283-291,1992.

9. **GOLUB, G. H. ; VAN LOAN, C. F.**: *Matrix Computations***,** North Oxford Pup. Co., 1983

10. **GOMIDE, F. A.; M. L. de Andrade Neto**; *Introdução à Automação Industrial Informatizada*, Editorial Kapeluz, 1987.

11. **GOODWIN, G. C.; K. S. SIN:** *Adaptive Filtering Prediction and Control*, Prentice-Hall, Englewood Cliffs N. J., 1984.

12. **FREEMAN, J. A.; D. M. SKAPURA**: *Neural Networks: Algorithms, Applications and Programming Tecniques*, Addison-Wesley Publishing Company, 1992.

13. **KERN, D. Q.**; *Processos de Transmissão de Calor*, Editora Guanabara Dois S.A., Rio de Janeiro, 1982.

14. **MANLY, B. F. J**.: *Multivariate Statistical Methods: a Primer*, Chapman and Hall, Londres, 1986

15. **MOHLER, R. R**.: *Nonlinear Systems: Dynamics and Control*, Printece-Hall, Englewood Cliffs N. J., 1991.

16. **NEITZEL, I.;***Um Controlador DMC Autosintonizante, Programa de Tese de Doutorado no Programa de Engenharia Química da COPPE/UFRJ*, 1995.

17. **PEEL, C.; MARK J. W. e MING T. T.;** *A Fast Procedure for The Training of Neural Networks*, Journal of Process Control, Vol. 2, Nr 4, pgs 205-211, 1992

18. **PONTON, J. W.:** *Neural Networks: Some Questions and Answers*, Journal of Process Control, Vol 2, Nr. 3, pgs 163-165, 1992.

19. **PSICHOGIOS, D. C.; L. H. UNGAR:** *Direct and Indirect Model Based Control Using Artificial Neural Networks*, Ind. Chem. Res. Vol 30, pgs 2564-2573,1991.

20. **RAMSEIER, M.; P. AGRAWAL; D. A. MELLICHAMP:** *Non-Linear Adaptive Control of Fermentation Processes Utilizing a Priori Modelling Knowledge*, J. Proc. Cont. Vol 2, Nr. 3, pgs 129-138, 1992.

21. **RAWLINGS, J. O.**: *Applied Regression Analysis: A Research Tool*, Wadsworth and Brooks, Califórnia, 1988

22. **SANOFF, S. P. e P. E. WELLSTEAD**; *Expert Identification and Control*, IFAC Proceeding Series, N^{o} 07, Identification and System Parameter Estimation, Editora H. A. Barker + P. C. Young, York, U.K., 1985

23. **UNBEHAUEN, H**.; *Regelemgs Technik*, F. Vieweg & Sohn, Vol I e II, 1985.

24. **UNGAR, L. H.; B. A. POWELL; S. N. KAMENS;** *Adaptive Networks for Fault Diagnosis and Process Control*, Comp. Chem. Eng. Vol 14, Nr . 4/5, pgs 561-572, 1990.

25. **VELOSO, E.**; *Curso de Métodos Numéricos*, PETROBRÁS – Petróleo Brasileiro S/A, CENPES - DIVEN, 1985.

26. **WHITE, D. A.; D. A. SOFGE:** *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold, 1992.

27. **WILLIS, M. J.; C. D. Di MASSIMO; G. A. MONTAGUE; M. T. THAN; A. J. MORRIS**: *Artificial Neural Networks in Process Engineering*, IEE Proceeding-D Vol. 138, Nr. 3, pgs 256-266, 1990

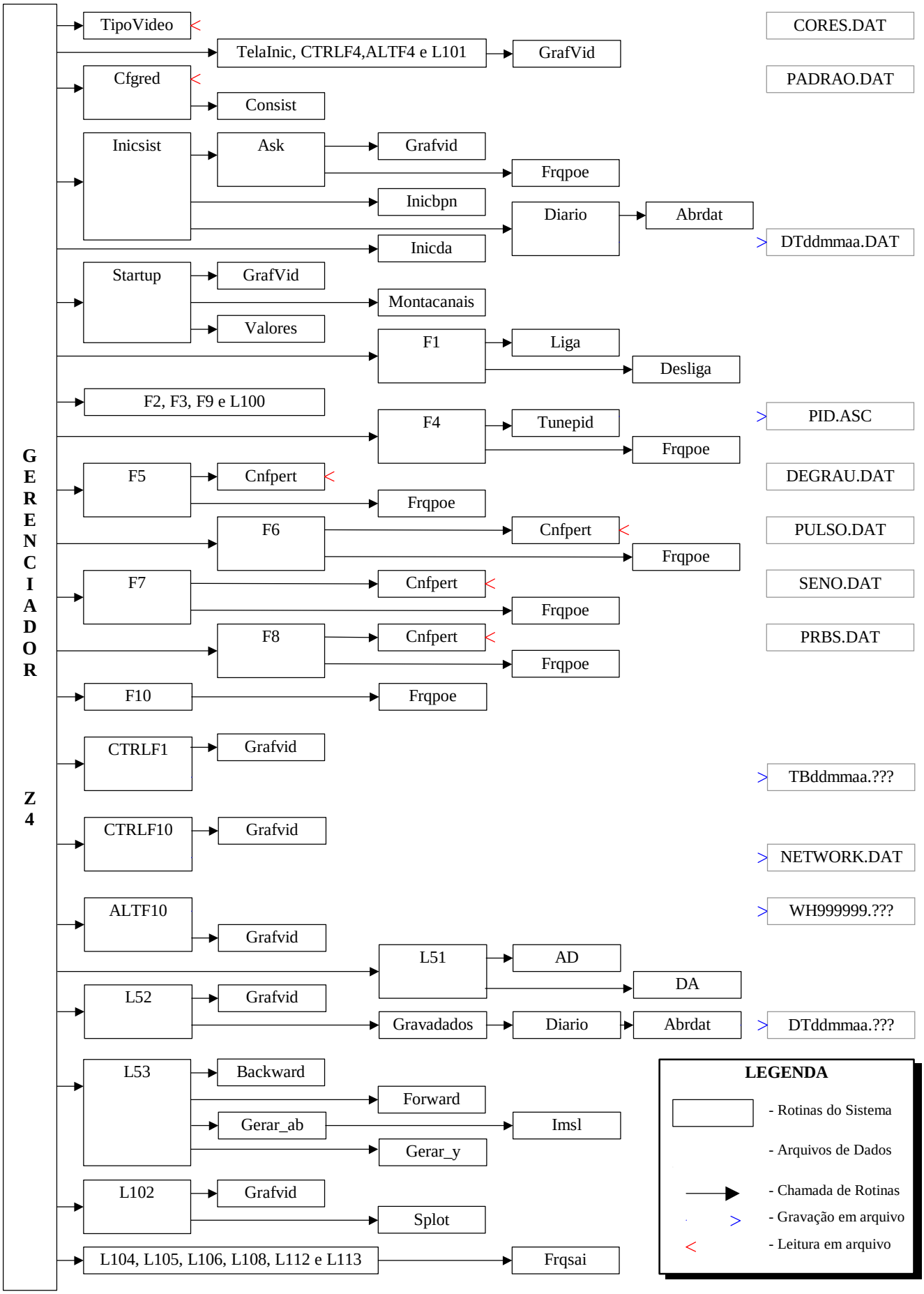
28. **WRAY, J.; GREEN, G. G. C.**: *Analysis of networks that have learnt control problems*, IEE Int. Conf. CONTROL '91, Edinburgh, Scotland, 1991

29. **YOUNG, S. J**.: *Real Time Languages: Design and Development*, Ellis Horwood Limited, 1982

30. **ZIEGLER, J. G.; N. B. NICHOLS**: *Optimum Settings for Automatic Controllers*,Trans. ASME, 64, 1942.


ANEXOS

Anexo I - Interligação das Rotinas do Software RTX



Anexo II - Arquivo de Configuração Geral do Software RTX

```
configuracao 94    ARQUIVO : PADRAO.DAT
versao=3.0 : 05 Out 94
XRG
quant=100
01  ..valor maximo range canal 1 [unid. usuario].. Ent.....    800.0000000
02  ..valor minimo range canal 1 [unid. usuario].. Ent.....   -200.0000000
03  ..valor maximo range canal 2 [unid. usuario].. Ent.....    800.0000000
04  ..valor minimo range canal 2 [unid. usuario].. Ent.....   -200.0000000
05  ..valor maximo range canal 3 [unid. usuario].. Ent.....      0.0000000
06  ..valor minimo range canal 3 [unid. usuario].. Ent.....      0.0000000
07  ..valor maximo range canal 4 [unid. usuario].. Ent.....      0.0000000
08  ..valor minimo range canal 4 [unid. usuario].. Ent.....      0.0000000
09  ..valor maximo range canal 5 [unid. usuario]. .Ent.....      0.0000000
10  ..valor minimo range canal 5 [unid. usuario]. .Ent.....      0.0000000
11  ..valor maximo range canal 6 [unid. usuario].. Ent.....      0.0000000
12  ..valor minimo range canal 6 [unid. usuario].. Ent.....      0.0000000
13  ..valor maximo range canal 7 [unid. usuario].. Ent.....      0.0000000
14  ..valor minimo range canal 7 [unid. usuario].. Ent.....      0.0000000
15  ..valor maximo range canal 8 [unid. usuario].. Ent.....      0.0000000
16  ..valor minimo range canal 8 [unid. usuario].. Ent.....      0.0000000
17  ..valor maximo range canal 1 [em volt] ..... Saida....      4.0000000
18  ..valor minimo range canal 1 [em volt] ..... Saida....      1.0000000
19  ..valor m ximo range canal 2 [em volt] ..... Saida....      4.0000000
20  ..valor minimo range canal 2 [em volt] ..... Saida....      1.0000000
21                ..... ..... ..... .....       .0000000
22                ..... ..... ..... .....       .0000000
23                ..... ..... ..... .....       .0000000
24           8 E ..... ..... ..... .....       .0000000
25    ..     1 E ..... ..... .. ..... .....       .0000000
26    ..     2 E ..... ..... ..... .....       .0000000
27    ..     3 E ..... ..... ..... .....       .0000000
28    ..     4 E ..... ..... ..... .....       .0000000
29    ..     1 E ..... ..... ..... .....       .0000000
30    ..     2 E ..... ..... ..... .....       .0000000
31    ..     3 E ..... ..... ..... .....       .0000000
32    ..     4 E ..... ..... ..... .....       .0000000
33    ..     1 S ..... ..... ..... .....       .0000000
34    ..     2 S ..... ..... ..... .....       .0000000
35    ..     3 S ..... ..... ..... .....       .0000000
36    ..     4 S ..... ..... ..... .....       .0000000
37    ..     1 E ..... ..... ..... .....       .0000000
38           2 E ..... ..... ..... .....       .0000000
39           3 E ..... ..... ..... .....       .0000000
40           4 E ..... ..... ..... .....       .0000000
41    ..     1 E ..... ..... ..... .....       .0000000
42           2 E ..... ..... ..... .....       .0000000
43           3 E ..... ..... ..... .....       .0000000
44           4 E ..... ..... ..... .....       .0000000
45               ..... ..... ..... .....       .0000000
46               ..... ..... ..... .....       .0000000
47               ..... ..... ..... .....       .0000000
48               ..... ..... ..... .....       .0000000
49   ..      1 S ..... ..... ..... .....       .0000000
50   ..      2 S ..... ..... ..... .....       .0000000
51   ..      3 S ..... ..... ..... .....       .0000000
52   ..      4 S ..... ..... ..... .....       .0000000
53               ..... ..... ..... .....       .0000000
54               ..... ..... ..... .....       .0000000
55               ..... ..... ..... .....       .0000000
56               ..... ..... ..... .....       .0000000
57               ..... ..... ..... .....       .0000000
```

```
58                ..... ..... ..... .....       .0000000
59                ..... ..... ..... .....       .0000000
60                ..... ..... ..... .....       .0000000
61                ..... ..... ..... .....       .0000000
62                ..... ..... ..... .....       .0000000
63                ..... ..... ..... .....       .0000000
64                ..... ..... ..... .....       .0000000
65   ..   Alarme de alta  n¡vel 1 do 1§ canal [0 a 5].....      2.0000000
66   ..      "      "     n¡vel 2 ...".. ....".....  .....      4.0000000
67   ..   Alarme de baixa n¡vel 1 ....". .....".... .....      2.0000000
68   ..      "      "     n¡vel 2.....". .....".... .....      0.5000000
69   ..   Alarme de alta  n¡vel 1 do 2§ canal [0 a 5].....      0.0000000
70   ..      "      "     n¡vel 2 ....". .....".... .....      0.0000000
71   ..   Alarme de baixa n¡vel 1.....". .....".... .....      0.0000000
72   ..      "      "     n¡vel 2.....". .....".... .....      0.0000000
73   ..   ..... ..... ..... ..... .....      0.0000000
74   ..   ..... ..... ..... ..... .....      0.0000000
75   ..   ..... ..... ..... ..... .....      0.0000000
76   ..   ..... ..... ..... ..... .....      0.0000000
77   ..   ..... ..... ..... ..... .....      0.0000000
78   ..   ..... ..... ..... ..... .....      0.0000000
79   ..   ..... ..... ..... ..... .....      0.0000000
80   ..   ..... ..... ..... ..... .....      0.0000000
81   ..   ..... ..... ..... ..... .....      0.0000000
82   ..   ..... ..... ..... ..... .....      0.0000000
83   ..   ..... ..... ..... ..... .....      0.0000000
84   ..   ..... ..... ..... ..... .....      0.0000000
85   ..   ..... ..... ..... ..... .....      0.0000000
86   ..   ..... ..... ..... ..... .....      0.0000000
87   ..   ..... ..... ..... ..... .....      0.0000000
88   ..   ..... ..... ..... ..... .....      0.0000000
89   ..   ..... ..... ..... ..... .....      0.0000000
90   ..   ..... ..... ..... ..... .....      0.0000000
91   ..   ..... ..... ..... ..... .....      0.0000000
92   ..   ..... ..... ..... ..... .....      0.0000000
93   ..   ..... ..... ..... ..... .....      0.0000000
94   ..   ..... ..... ..... ..... .....      0.0000000
95   ..   ..... ..... ..... ..... .....      0.0000000
96   ..   ..... ..... ..... ..... .....      0.0000000
97  valor max. de tempo 24*60*60 seg [rtx chama diario].....  86400.0000000
98                ..... ..... ..... .....       .0000000
99                ..... ..... ..... .....       .0000000
100               ..... ..... ..... .....       .0000000
IDIG
quant=010
01                ..... ..... ..... .....    0
02                ..... ..... ..... .....    0
03                ..... ..... ..... .....    0
04                ..... ..... ..... .....    0
05                ..... ..... ..... .....    0
06                ..... ..... ..... .....    0
07                ..... ..... ..... .....    0
08                ..... ..... ..... .....    0
09                ..... ..... ..... .....    0
10                ..... ..... ..... .....    0
KFLAG
quant=050
01 Indicador de atualiza‡Æo dos pesos da rede[1=At 0=NÆo At]    1
02 Indica se atualiza os pontos dos gr ficos[1=Sim 0=NÆo]...    1
03 Indica se existem pontos para atualizar rede [DVS].......    0
04                ..... ..... ..... .....    0
05                ..... ..... ..... .....    0
06                ..... ..... ..... .....    0
07 Alarme de Alta N¡vel 1..... [0 = normal  1 = violado ]...    0
08 Alarme de Alta N¡vel 2..... [..... ..".... .....]...    0
```

```
09 Alarme de Baixa N¡vel 1.... [........ ..".... .....]...    0
10 Alarme de Baixa N¡vel 2.... [........ ..".... .....]...    0
11 Situacao do processo [Startup]....... ........ ........    0
12                 ............ ........ ........ ........    0
13                 ............ ........ ........ ........    0
14                 ............ ........ ........ ........    0
15                 ............ ........ ........ ........    0
16                 ............ ........ ........ ........    0
17                 ............ ........ ........ ........    0
18                 ............ ........ ........ ........    0
19                 ............ ........ ........ ........    0
20                 ............ ........ ........ ........    0
21                 ............ ........ ........ ........    0
22                 ............ ........ ........ ........    0
23                 ............ ........ ........ ........    0
24                 ............ ........ ........ ........    0
25 Indica que e hora de executar IAUTO [0=n„o most. 1=most.]    0
26                 ............ ........ ........ ........    0
27                 ............ ........ ........ ........    0
28                 ............ ........ ........ ........    0
29                 ............ ........ ........ ........    0
30                 ............ ........ ........ ........    0
31                 ............ ........ ........ ........    0
32                 ............ ........ ........ ........    0
33                 ............ ........ ........ ........    0
34                 ............ ........ ........ ........    0
35                 ............ ........ ........ ........    0
36                 ............ ........ ........ ........    0
37                 ............ ........ ........ ........    0
38                 ............ ........ ........ ........    0
39                 ............ ........ ........ ........    0
40 Inibe video    [0=desinibido  1=inibido]....... ........    0
41 Define tela absoluta  [0=desativado  1=ativado] ........    0
42 Define tela relativa  [0=desativado  1=ativado] ........    0
43 Define tela padrÆo    [0=nÆo gravada 1=gravada] ........    0
44                 ............ ........ ........ ........    0
45                 ............ ........ ........ ........    0
46                 ............ ........ ........ ........    0
47                 ............ ........ ........ ........    0
48                 ............ ........ ........ ........    0
49                 ............ ........ ........ ........    0
50                 ............ ........ ........ ........    0
IPAR
quant=050
01 Numero maximo de linhas de IFREQ  [30]......... .........   30
02 Numero maximo de colunas de IFREQ  [30]........ .........   30
03 Numero maximo de pontos da matriz vd(4,num max) [0 a 350]  350
04 Intervalo de amostragem em segundos  [Ts]...... .........   90
05 Numero maximo de canais de entrada   [1 a 8]... .........    2
06 Numero maximo de canais de saida     [1 a 8]..  .........    2
07 Numero de registro no buffer de armaz. [0 a 40] .........   40
08 Numero de set-point dispon¡veis [1 a 5][ver xdatar(41)]..    5
09                 ............ ........ ........ ........    0
10 Numero de neuronios da camada de entrada [1 a 50]........   24
11 Numero de neuronios da camada escondida  [1 a 50]........   10
12 Numero de neuronios de entrada da variavel manipulada....   14
13                 ............ ........ ........ ........    0
14                 ............ ........ ........ ........    0
15 Numero m x. iteracoes algoritmo D.V.S.[Variavel manip.].. 2000
16                 ............ ........ ........ ........    0
17                 ............ ........ ........ ........    0
18                 ............ ........ ........ ........    0
19                 ............ ........ ........ ........    0
20                 ............ ........ ........ ........    0
21                 ............ ........ ........ ........    0
```

```
22                ..... ..... ..... .....    0
23                ..... ..... ..... .....    0
24                ..... ..... ..... .....    0
25                ..... ..... ..... .....    0
26                ..... ..... ..... .....    0
27                ..... ..... ..... .....    0
28                ..... ..... ..... .....    0
29                ..... ..... ..... .....    0
30                ..... ..... ..... .....    0
31                ..... ..... ..... .....    0
32                ..... ..... ..... .....    0
33                ..... ..... ..... .....    0
34                ..... ..... ..... .....    0
35                ..... ..... ..... .....    0
36                ..... ..... ..... .....    0
37                ..... ..... ..... .....    0
38                ..... ..... ..... .....    0
39                ..... ..... ..... .....    0
40                ..... ..... ..... .....    0
41                ..... ..... ..... .....    0
42                ..... ..... ..... .....    0
43 Numero de variaveis a serem plotadas em video [1 a 3]....    3
44 Canal a ser plotado como 1@ variavel  [1 a 16]. .....    1
45 Canal a ser plotado como 2@ variavel  [1 a 16]. .....   10
46 Canal a ser plotado como 3@ variavel  [1 a 16]. .....    9
47 Numero de pontos do grafico  [1 a 350]..... .....  350
48                ..... ..... ..... .....    0
49                ..... ..... ..... .....    0
50 Tempo de espera para simula‡„o (segundos)..... .....    0
XDATAR
quant=100
01 Tempo de exibicao de tela relativa... ..[seg].. .....    900.0000000
02 Hora de parar exibicao da tela relativa.[seg].. .....       .0000000
03 Tempo inicial dos dados gravados em disco [Usado L52]....      0.0000000
04 Erro medio quadrado [ E(set-point - xio)^2 ]... .....       .0000000
05 ..................................................  .....       .0000000
06 Tempo de duracao da soma dos erros .. ..... .....  21600.0000000
07 Passo para correcao dos pesos da rede [NETA]... .....      0.0010000
08 Fator de inicializacao dos pesos RANDOM()/F [INICBPN]....   -100.0000000
09 Valor inicial do canal de sa¡da [usado p/ L106] .....       .0000000
10 Constante de tempo do processo em teste [TAU seg]........   2400.9370000
11 Ganho proporcional  .......... Kp1 [usado p/ teste]......      1.2739580
12 Variavel da equacao usada para teste [senoide].... ......      0.0000000
13 Tempo morto do processo em teste...[Seg]....... .....    180.0000000
14                ..... ..... ..... .....       .0000000
15 Valor m¡nimo da funcao F(Xqp) p/ iteracoes D.V.S. [L106].       0.000001
16                ..... ..... ..... .....       .0000000
17 Valor inferior para o tempo em graficos.[seg].se int i*4.       .0000000
18                ..... ..... ..... .....       .0000000
19                ..... ..... ..... .....       .0000000
20 Tempo por ponto grafico.... ..... ..... .....      90.0000000
21                ..... ..... ..... .....       .0000000
22 Valores temporarios usado pelas pertubacoes.... .....       .0000000
23 Valores tempor rios usado pela [BPN-Erro m,dio] .....       .0000000
24 Valores temporarios usado pelas pertubacoes.... .....       .0000000
25 Valores temporarios usado pelas pertubacoes.... .....       .0000000
26 Valores temporarios usado pelas pertubacoes.... .....       .0000000
27                ..... ..... ..... .....       .0000000
28                ..... ..... ..... .....       .0000000
29                ..... ..... ..... .....       .0000000
30                ..... ..... ..... .....       .0000000
31                ..... ..... ..... .....       .0000000
32                ..... ..... ..... .....       .0000000
33                ..... ..... ..... .....       .0000000
34                ..... ..... ..... .....       .0000000
```

```
35                ..... ..... ..... .....       .0000000
36                ..... ..... ..... .....       .0000000
37                ..... ..... ..... .....       .0000000
38                ..... ..... ..... .....       .0000000
39                ..... ..... ..... .....       .0000000
40 Valor da vari vel de sa¡da [usado p/ pertub]... .....       .0000000
41 Set point do canal controlado [n¡vel   I]..... .....    100.0000000
42 Set point do canal controlado [n¡vel  II]..... .....    200.0000000
43 Set point do canal controlado [n¡vel III]..... .....    300.0000000
44 Set point do canal controlado [n¡vel  IV]..... .....    400.0000000
45 Set point do canal controlado [n¡vel   V]..... .....    500.0000000
46                ..... ..... ..... .....       .0000000
47                ..... ..... ..... .....       .0000000
48                ..... ..... ..... .....       .0000000
49                ..... ..... ..... .....       .0000000
50 Set point do canal controlado[Usado p/ programa 0 a 5]...       .0000000
51 Set point do canal 1 [Entrada]..... ..... .....    300.0000000
52 Set point do canal 2 [Entrada]..... ..... .....    300.0000000
53 Set point do canal 3 [Entrada]..... ..... .....       .0000000
54 Set point do canal 4 [Entrada]..... ..... .....       .0000000
55 Set point do canal 5 [Entrada]..... ..... .....       .0000000
56 Set point do canal 6 [Entrada]..... ..... .....       .0000000
57 Set point do canal 7 [Entrada]..... ..... .....       .0000000
58 Set point do canal 8 [Entrada]..... ..... .....       .0000000
59 Valor de equilibrio do canal 1 [Saida]..... .....      2.5000000
60 Valor de equilibrio do canal 2 [Saida]..... .....      2.5000000
61 Valor de equilibrio do canal 3 [Saida]..... .....       .0000000
62 Valor de equilibrio do canal 4 [Saida]..... .....       .0000000
63 Valor de equilibrio do canal 5 [Saida]..... .....       .0000000
64 Valor de equilibrio do canal 6 [Saida]..... .....       .0000000
65 Valor de equilibrio do canal 7 [Saida]..... .....       .0000000
66 Valor de equilibrio do canal 8 [Saida]..... .....       .0000000
67                ..... ..... ..... .....       .0000000
68                ..... ..... ..... .....       .0000000
69 Valor inicial do canal 1 [Entrada] [0 a 5]..... .....       .0000000
70 Valor inicial do canal 2 [Entrada] [0 a 5]..... .....       .0000000
71 Valor inicial do canal 3 [Entrada] [0 a 5]..... .....       .0000000
72 Valor inicial do canal 4 [Entrada] [0 a 5]..... .....       .0000000
73 Valor inicial do canal 5 [Entrada] [0 a 5]..... .....       .0000000
74 Valor inicial do canal 6 [Entrada] [0 a 5]..... .....       .0000000
75 Valor inicial do canal 7 [Entrada] [0 a 5]..... .....       .0000000
76 Valor inicial do canal 8 [Entrada] [0 a 5]..... .....       .0000000
77 Valor inicial do canal 1 [Saida] [0 a 5]..... .....      2.5000000
78 Valor inicial do canal 2 [Saida] [0 a 5]..... .....      2.5250505
79 Valor inicial do canal 3 [Saida] [0 a 5]..... .....       .0000000
80 Valor inicial do canal 4 [Saida] [0 a 5]..... .....       .0000000
81 Valor inicial do canal 5 [Saida] [0 a 5]..... .....       .0000000
82 Valor inicial do canal 6 [Saida] [0 a 5]..... .....       .0000000
83 Valor inicial do canal 7 [Saida] [0 a 5]..... .....       .0000000
84 Valor inicial do canal 8 [Saida] [0 a 5]..... .....       .0000000
85 Tau [seg.] usado se Kflag(12) = 1.. ..... .....    2400.9370000
86 Kp [volt/volt] usado se Kflag(12) = 1 ..... .....      1.2739580
87 Tm [seg.] usado se Kflag(12) = 1..... ..... .....     180.0000000
88   Kc  [ constante do controlador ]... ..... .....       .0000000
89   1/Ti [ 1/Ki - tempo integral ]..... ..... .....       .0000000
90   Td [ Kd - tempo derivativo ]..... ..... .....       .0000000
91   d0 [ q0 ]    ..... ..... ..... .....       .0000000
92   d1 [ q1 ]    ..... ..... ..... .....       .0000000
93   d2 [ q2 ]    ..... ..... ..... .....       .0000000
94   c1 [ constante da formula do control. PID ].. .....       .0000000
95   Tv [ altura do salto da resp. simul. do degrau ].....       .0000000
96   e(k) [ erro instante atual ]..... ..... .....       .0000000
97   e(k-1) [ erro instante anterior (-1) ]..... .....       .0000000
98   e(k-2) [ erro instante anterior (-2) ]..... .....       .0000000
99   d u(k-1) [ perturbacao instante anterior ]... .....       .0000000
```

```
100  d u(k) [ perturbacao instante atual ]........ .........       .0000000
IDATAR
quant=050
01 Define eq. de teste [0=processo 1¦ ord 1=seno ] .........    0
02 Tempo de dura‡ao da perturba‡ao...... [seg,nTs] .........    0
03 Pontos de controle do processo pela BPN [Usado por L106].    0
04 Fator de controle [Ft * Ts para atingir SP] usado L106...   25
05 Set-point ativo [usado por CTRF4 e ALTF4][ver xdatar(41)]    3
06 Numero de observacoes p/ matriz de dados....... .........  125
07 Ordem do polinomio de ACP. ......... ...[>1]. ..........    3
08 Parametro para calculo dos autovalores......... .........   11
09 Canal de entrada a ser controlado.... ......... .........    1
10 Canal de saida a ser controlado...... ......... .........    1
11 Tamanho da janela de observacao disponiveia p/ treino DVS 1500
12 Numero de pontos para o grafico de video.[Val inicial=0].    0
13 Tipo de pertubacao [usado por CNFPERT]......... .........    0
14 Valores tempor rios para pertuba‡ao.. ......... .........    0
15 Valores tempor rios para pertuba‡ao.. ......... .........    0
16 Valores tempor rios para pertuba‡ao.. ......... .........    0
17 Valores tempor rios para pertuba‡ao.. ......... .........    0
18 Valores tempor rios para pertuba‡ao.. ......... .........    0
19 Define controlador em atividade [0=s/ contr. 1=c/contr.]     0
20 Define pert. em atividade [0=s/ pert. 0<>c/pert.].........    0
21 Define Rede Backpropagation em atividade [1=ON 0=OFF]....    0
22 Define Bias na camada escondida da rede..[1=ON 0=OFF]....    0
23 Define Bias na camada de sa¡da da rede...[1=ON 0=OFF]....    0
24                ............ ......... ......... .........    0
25                ............ ......... ......... .........    0
26                ............ ......... ......... .........    0
27 Define tipo de treino [1=STEEP DESCENT 2=DVS].. .........    2
28 Define controlador[1=BPN 2=PID 3=PI 4=P 5=PD 0=Nao def.].    2
29 Tempo de amostragem do PID   :  numero * ipar(4)[Ts].....    1
30 Endereco base da placa de entrada [Hexadecimal] .........  220
31 Endereco base da placa de saida   [Hexadecimal] .........  224
32 Numero de leituras da placa A/D...... ......... .........   10
33 Contador de nTs da soma dos erros.... ......... .........    0
34 Contador de nTs de aplicacao de pertubacao..... .........    0
35 Numero de Ts a ser aplicado a pertubacao....... .........    0
36 Contador de aplicacao da rotina l103. ......... .........    0
37 Numero de Ts a ser utilizado na soma dos erros .........    0
38 Contador de nTs passados (usado por DVS)....... .........    0
39                ............ ......... ......... .........    0
40 Armazenamento de Tela gr fica [usado em TelaInic.for]....    0
41 .......  . . . . . . . . . . . . . . . . . ..... .........    0
42 N£mero de neur“nios na camada escondida [D.V.S.] ........    0
43                ............ ......... ......... .........    0
44                ............ ......... ......... .........    0
45                ............ ......... ......... .........    0
46 N£mero do canal de sa¡da com pertuba‡„o........ .........    2
47                ............ ......... ......... .........    0
48 Contador do buffer de armaz. em disco .[L51 Val. inic.00]    0
49 Indicador de rotina que utilizou a tela ....... .........    0
50                ............ ......... ......... .........    0
VARIAVEL    [Tamanho m ximo 10 caracteres]
quant=016
01 Temperatura T1 -canal de entrada ......... Temp. §C
02 Sa¡da da rede..-canal de entrada ......... Temp. §C
03 ............ ..-canal de entrada .........
04 ............ .........".......... .........
05 ............ .........".......... .........
06 ............ .........".......... .........
07 ............ .........".......... .........
08 ............ .........".......... .........
09 Resistencia R...canal de saida.. ......... Resist.(V)
10 ............ ......... ......... ......... Resist.(V)
```

```
11 ............ .........".......... .........
12 ............ .........".......... .........
13 ............ .........".......... .........
14 ............ .........".......... .........
15 ............ .........".......... .........
16 ............ .........".......... .........
TITULO    [Tamanho m ximo 35 caracteres]
quant=016
01 Temperatura T1 -canal de E Gr fico da Temperatura
02 Rede em caso de Teste.de E Valor de Sa¡da da Rede
03 ............ .........de E
04 ............ ........."...
05 ............ ........."...
06 ............ ........."...
07 ............ ........."...
08 ............ ........."...
09 Resistencia. ...canal de s Vari vel Manipulada
10 ............ .........".... Pertuba‡ao Aplicada
11 ............ ........."...
12 ............ ........."...
13 ............ ........."...
14 ............ ........."...
15 ............ ........."...
16 ............ ........."...
```

Anexo III - Arquivos de Configurações do Software RTX

Arquivo CORES.DAT[0=Preta 1=Azul 2=Verde 3=Cyan 4=Verm 5=Magen 6=Pardo
quant=12 [7=Bran 8=Verd Esc {9,10,11,12,13}=Azul Clar. 14=Amar 15=Bran Int.
01 Cor de fundo da tela de trabalho 15
02 Cor de fundo das janelas de aviso 01
03 Cor das molduras 00
04 Cor de escrita destacada..... 15
05 Cor do ponto grafico..... 00
06 Cor dos eixos dos graficos..... 00
07 Cor dos Titulos dos graficos..... 00
08 Cor da area do Menu de Op‡oes das teclas..... 03
09 Cor da linha do Set-Point 00
10 Cor da escrita na area de mensagens na tela principal.... 00
11 Cor da area dos graficos..... 15
12 Cor da escrita do Menu de Opcoes..... 00

configuração 94 - ARQUIVO | PULSO.DAT
versao=2.0 :07 Jul 95
IDATAR
quant= 07
01 ..idatar(02) - nulo..... 0
02 ..idatar(13) - tipo da perturbacao...[=2]..... 2
03 ..idatar(14) - altura do salto... [% do span]..... 50
04 ..idatar(15) - duracao do pulso... [seg,>Ts]..... 90
05 ..idatar(16) - nulo..... 0
06 ..idatar(17) - nulo..... 0
07 ..idatar(18) - nulo..... 0

configuracao 94 - ARQUIVO | DEGRAU.DAT
versao= 2.0 : 07 Jul 95 .
IDATAR .
quant= 07
01 ..idatar(02) - tempo de duracao da perturbacao...[seg]..18000
02 ..idatar(13) - tipo da perturbacao...[=1]..... 1
03 ..idatar(14) - altura do salto...[% do span]..... 50
04 ..idatar(15) - nulo..... 1
05 ..idatar(16) - nulo..... 1
06 ..idatar(17) - nulo..... 1
07 ..idatar(18) - nulo..... 1

configuracao 94 - ARQUIVO | PRBS.DAT
versao= 2.0 : 07 Jul 95 .
IDATAR .
quant= 07
01 ..idatar(02) - tempo de duracao da perturbacao..[seg]...10800
02 ..idatar(13) - tipo da perturbacao...[=4]..... 4
03 ..idatar(14) - altura do salto...[% do span]..... 50
04 ..idatar(15) - duracao do pulso elementar...[seg,nTs]... 540
05 ..idatar(16) - comprimento do registro gerador..... 8
06 ..idatar(17) - 1o. registro de feedback..... 1
07 ..idatar(18) - 2o. registro de feedback..... 1

configuracao 94 - ARQUIVO | PULSOC.DAT

```
versao=2.0 :07 Jul 95                                          .
IDATAR                                                         .
quant= 07
01  ..idatar(02) - tempo de aplicacao da pertubacao.[seg]... 7200
02  ..idatar(13) - tipo da perturbacao...[=5]..............    5
03  ..idatar(14) - altura do salto... [% do span]..........   30
04  ..idatar(15) - duracao do pulso... [seg,>Ts]............   90
05  ..idatar(16) - intervalos de aplicacao dos pulsos [seg].  360
06  ..idatar(17) - flag de treino da rede [1=cont. 0=desc]..    1
07  ..idatar(18) - contador da aplicacao da pertubacao......    0
```

configuracao 94 - ARQUIVO | SENO.DAT

```
versao= 2.0 : 07 Jul 95                                        .
IDATAR                                                         .
quant= 07
01  ..idatar(02) - tempo de duracao da perturbacao..[seg]...10800
02  ..idatar(13) - tipo da perturbacao...[=3]..............    3
03  ..idatar(14) - amplitude da senoide...[% do span].....   25
04  ..idatar(15) - periodo da senoide...[seg].............. 1800
05  ..idatar(16) - nulo.....................................    0
06  ..idatar(17) - nulo.....................................    0
07  ..idatar(18) - nulo.....................................    0
```

Anexo IV - Equação do controlador PID no modo discreto

Partindo-se da equação 2I temos que:

$$H(z) = \frac{M(z)}{E(z)} = K_c\left[1 + \frac{T_s(z+1)}{2t_i(z-1)} + \frac{t_d}{zT_s}\frac{z-1}{1+T(z-1)/(zT_s)}\right] \text{ ou seja:}$$

$$H(z) = \frac{M(z)}{E(z)} = K_c\left[1 + \frac{T_s}{2t_i}\frac{z+1}{z-1} + \frac{t_d}{T_s}\frac{z-1}{z(1+T/T_s) - T/T_s}\right]$$

$$H(z) = \frac{M(z)}{E(z)} = K_c\left[1 + \frac{T_s}{2t_i}\frac{z+1}{z-1} + \frac{t_d/T_s}{1+T/T_s}\frac{z-1}{z - \frac{T/T_s}{1+T/T_s}}\right]$$

definindo $c_1 = -\dfrac{T/T_s}{1+T/T_s} = -\dfrac{T}{T+T_s}$ temos;

$$H(z) = \frac{K_c}{1+T/T_s}\left[(1+T/T_s) + \frac{T_s(1+T/T_s)}{2t_i}\frac{z+1}{z-1} + \frac{t_d}{T_s}\frac{z-1}{z+c_1}\right]$$

multiplicando-se as frações por $\dfrac{z}{z}$ obtemos;

$$H(z) = \frac{K_c}{1+T/T_s}\left[(1+T/T_s) + \frac{(T_s/2t_i)(1+T/T_s)(1+z^{-1})}{(1-z^{-1})} + \frac{(t_d/T_s)(1-z^{-1})}{(1+c_1z^{-1})}\right]$$

$$H(z) = \frac{K_C}{1+T/T_S}\left[\frac{(1+T/T_S)(1-z^{-1})(1+c_1z^{-1}) + (T_S/2t_I)(1+T/T_S)(1+z^{-1})(1+c_1z^{-1}) + (t_d/T_S)(1-z^{-1})(1-z^{-1})}{(1-z^{-1})(1+c_1z^{-1})}\right]$$

definindo $d_o^{'} = (1+T/T_s) + \dfrac{T_s+T}{2t_i} + \dfrac{t_d}{T_s} = 1 + \dfrac{T_s+T}{2t_i} + \dfrac{T+t_d}{T_s}$,

$$d_1^{'} = -(1+T/T_s) + c_1(1+T/T_s) + \frac{T_s+T}{2t_i} + c_1\frac{T_s+T}{2t_i} - \frac{2t_d}{T_s} = -1 + \frac{T_s}{2t_i} - \frac{2(T+t_d)}{T_s} \quad \text{e}$$

$d_2^{'} = -c_1(1+{}^{T}\!/_{T_s}) + c_1\dfrac{T_s+T}{2t_i} + \dfrac{t_d}{T_s} = \dfrac{T+t_d}{T_s} - \dfrac{T_s}{2t_i}$ e aplicando, temos;

$$H(z) = \frac{M(z)}{E(z)} = \frac{K_c}{1+{}^{T}\!/_{T_s}}\left[\frac{d_0^{'} + d_1^{'}z^{-1} + d_2^{'}z^{-2}}{(1-z^{-1})(1+c_1z^{-1})}\right] = \frac{d_0 + d_1z^{-1} + d_2z^{-2}}{(1-z^{-1})(1+c_1z^{-1})}$$

sendo definidos:

$$d_0 = \frac{K_c}{1+{}^{T}\!/_{T_s}}\left[1 + \frac{T_s+T}{2t_i} + \frac{T+t_d}{T_s}\right],$$

$$d_1 = \frac{K_c}{1+{}^{T}\!/_{T_s}}\left[-1 + \frac{T_s}{2t_i} - \frac{2(T+t_d)}{T_s}\right] \quad \text{e} \quad d_2 = \frac{K_c}{1+{}^{T}\!/_{T_s}}\left[\frac{T+t_d}{T_s} - \frac{T_s}{2t_i}\right]$$

Sabendo que $Z\{f(k-m)\} = z^{-m}F(z)$ tem-se o algoritmo na forma discreta é dado por:

$$m(k) = d_0e(k) + d_1e(k-1) + d_2e(k-2) + (1-c_1)m(k-1) + c_1m(k-2)$$

Considerando as variáveis desvios:

$$\Delta m(k) = m(k) - m(k-1) \quad \text{e} \quad \Delta m(k-1) = m(k-1) - m(k-2)$$

temos que;

$$\Delta m(k) = d_0e(k) + d_1e(k-1) + d_2e(k-2) - c_1\Delta m(k-1)$$

portanto, a equação acima é utilizada para a implantação de um controlador PID no modo discreto.