UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

TRABALHO DE CONCLUSÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Estratégia baseada em Redes Neurais Artificiais para a seleção on-line de controlador

Autor: Aline Thaís Käfer

Orientador: Pedro Rafael Bolognese Fernandes

Porto Alegre, 17 de novembro de 2014

Sumário

Agradecimentos

Agradeço de coração a todos os professores que fizeram parte deste trabalho, direta e indiretamente; a todos os meus colegas e amigos pelo companheirismo, partilha de aprendizado e companhia; e por fim a minha família e namorado, pela motivação e suporte necessários durante toda a caminhada. Um agradecimento especial ao professor Pedro Fernandes, por me dar sempre a direção certa a seguir durante o desenvolvimento deste trabalho.

Resumo

Neste trabalho foi implementado um método para seleção em tempo real do controlador mais adequado, dentro de um conjunto de controladores pré-definidos, para o controle de um processo sujeito a distúrbios não medidos e variações de comportamento em função do ponto de operação. A estrutura responsável pela seleção do controlador é uma Rede Neural Artificial (RNA) que é treinada de modo a selecionar o controlador de melhor desempenho com base em dados de simulação ou de um período de operação simulado. Com isto, pretende-se que a estratégia proposta seja simples e robusta para o controle de processos nos quais um controlador com sintonia fixa não apresenta um desempenho adequado. Objetivando verificar o funcionamento da proposta, foi considerado o modelo de um tanque esférico, um sistema não linear, sujeito a uma perturbação não medida na entrada. O ajuste da rede foi feito a partir de simulações do modelo do tanque em malha fechada, considerando-se diferentes controladores de nível do tipo PID sintonizados para diferentes pontos de operação. Uma nota, consistindo da combinação de diversos critérios de desempenho, foi atribuída para o desempenho de cada sistema, e empregada para treinar a rede. Depois de treinada, a rede tem como entradas o valor atual da saída e o valor de referência e, como saída, o melhor controlador para o cenário de operação. Diversos parâmetros podem ser variados em cada etapa, possibilitando a escolha da melhor configuração para que o controle final seja eficiente. A simulação da operação do sistema, consistindo do conjunto de controladores e do bloco de seleção baseado na rede neural, apresentou bons resultados tanto para a função servo quanto para a função regulatória.

PALAVRAS-CHAVE: sistema não-linear, controle, PID, seleção, redes neurais.

Abstract

At this paper, it was implemented a method of real-time selection of the most adequate controller, inside a set of pre-defined controllers, to control a process in which there are unmeasured disturbances and behavior variations dependent of point of operation. The structure responsible to select the controller is an Artificial Neural Network (ANN), trained for the selection of the controller whose performance is the best, based in data acquired from simulation or during the operation of the process. It is intended the proposed strategy to be simple and robust, to control processes in which a fix tuning does not have an adequate performance. A spherical tank was used to simulate the method behavior, whose model is nonlinear and whose outflow is a perturbation of the system. The ANN were trained with data acquired from closed loop simulations at the tank's model, considering different controllers from a set of controllers tuned at different operation points. From the data acquired, grades were created to represent their performance at each point, considering various performance criteria. After adjusted, the network entries are the tank's current level and the target level and, as output, the better controller to the operation scenario. Plenty of parameters can be changed to achieve an efficient control action. The simulations with the operating system, consisting in the set of controllers and the selection block based in Neural Network, showed good results to slave function as to regulatory function.

Key-words: nonlinear system, control, PID, selection, neural networks.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Ki – constante integral do controlador PID
Kp – constante proporcional do controlador PID
Kd – constante derivativa do controlador PID
u_k – entradas de uma RNA
y_j– saídas de uma RNA
a_{ij} – pesos da saída j de uma RNA, no neurônio i
A – matriz contendo os pesos das saídas de uma RNA
b_{ik} - pesos daentrada k de uma RNA, no neurônio i
B – matriz contendo os pesos das entradas de uma RNA
w_i – peso do *bias* de uma rede RNA, no neurônio i
w – vetor contendo os valores de *bias*de todos os neurônios de uma RNA
v_i – somatório das entradas e saídas ponderadas, e do *bias* de uma RNA, no neurônio i
x_i – saídas da fase de dinâmica linear de uma RNA, no neurônio i
H(s) – dinâmica linear em uma RNA
g(x(t)) – função de transferência de uma RNA
α e μ- taxa de aprendizagem de uma RNA
M - massa no tanque esférico, em kg
F_{out} - vazão de saída do tanque esférico, em cm^3/s
F_{in} - vazão de entrada do tanque esférico, em cm^3/s
C_D - coeficiente de descarga da válvula
A_0 - área inicial da abertura da válvula, em cm^2
ρ - densidade do líquido, em kg/cm^3
D o diâmetro da esfera, em cm
g - aceleração da gravidade, em cm/s^2
h - nível no tanque, em cm
μ - escalar utilizado no método de*Levenberg-Marquardt*
e_{RP} – erro em regime permanente
t_r - instante em que a curva cruza o valor de referência
t_d–tempo de subida (50% da amplitude)
t_s- tempo de acomodação
M_P – sobrepasso absoluto
M% - sobrepasso porcentual
t_u – tempo de subida (dos 10% aos 90% da amplitude)

Lista de Abreviaturas e Siglas

PID – *Proportional, IntegrativeandDerivative* (controller) – Controlador Proporcional, Integral e Derivativo

RNA – Redes Neurais Artificiais (ANN – *Artificial Neural Networks*)

FCOR – *FilteringandCorrelationAlgorithm*

1 Introdução

O controle de sistemas não lineares é uma área de conhecimento que tem estimulado várias pesquisas, com a criação e o desenvolvimento de diferentes métodos. Alguns destes têm como estratégia empregar uma variável interna que represente as variações do processo, se adaptando a cada instante ao sistema, tais como os conhecidos *GainScheduling*e *Controle Adaptativo*. Uma família de métodos do tipo *GainScheduling*propõe o uso de um ou mais controladores lineares ao mesmo tempo ou agindo em pontos diferentes, como forma de manter a simplicidade inerente a este tipo de controladores. Dentre estas estratégias, pode-se citar o método dos *Múltiplos Modelos Lineares*, no qual se combinam as saídas de controladores lineares ajustados para cada ponto de operação do sistema, e também o método da seleção de controladores, que avalia em cada ponto qual o melhor controlador a ser utilizado, dentro de certo número de controladores candidatos. Há ainda a aplicação de técnicas oriundas do campo da informática para a resolução do problema de classificação e aprendizagem das nãolinearidades do processo, como os *Controladores Fuzzy*, baseados em regras decididas por um especialista,e o *Controle por Redes Neurais Artificiais (RNAs).*Os métodos de seleção de controlador apresentamalgumas vantagens: a seleção é feita durante a operação, considerando seus valores atuais e os controladores candidatos podem ser, por exemplo, PIDs, controladores muito utilizados na indústria e de fácil sintonia.

Existem vários métodos para selecionar o controlador ideal. De modo geral, estes métodos atribuem ao sistema composto por processo e controlador um critério de desempenho, e a cada instante determinam qual controlador melhor atende tal objetivo. Dentre eles, pode-se citar o método H_∞ (H-infinito), no qual o objetivo é selecionar o controlador mais robusto dentre um conjunto de controladores. Utilizar uma rede neural para a seleção não se distancia muito desta prática, com a diferença de que este método é baseado em dados.

Pretende-se com este método que, a partir de dados da planta, uma rede neural artificial seja treinada de forma a adquirir a capacidade de selecionar o melhor controlador, entre os candidatos, para cada ponto de operação. A intensão é controlar um processo não linear, cujo comportamento varia em função do ponto de operação, com um melhor desempenho global do que cada um dos controladores candidatos.

Conforme já comentado, é proposto um método para controle de plantas não lineares baseado na seleção on-line de controladores. Três etapas constituem a proposta: a primeira consiste em se obter dados do processo em malha fechada para verificar o desempenho dos controladores candidatos em diversas condições operacionais. O segundo passo é avaliar o desempenho de forma quantitativa, por meio de um atributo (nota), e utilizar estes atributos para o treinamento da rede. Por fim, depois de treinada, a rede será capaz de, a partir das entradas que determinam o ponto de operação, atribuir uma nota a cada instante, para cada um dos controladores. O controlador de maior nota é selecionado para atuar no processo naquele instante.

1.1 Objetivos do presente trabalho

O principal objetivo deste trabalho é testar o emprego de Redes Neurais Artificiais como selecionador de controladores para um processo não linear. Como segundo passo, pretende-se fazer diversos testes para verificar o melhor conjunto de parâmetros de

aquisição de dados e procedimento de treinamento da rede, de forma que seja possível avaliar o desempenho do método e propor configurações aceitáveis para aplicações futuras.

1.2 Estrutura do Trabalho

O trabalho está organizado da seguinte maneira:

O Capítulo 2apresenta uma revisão bibliográfica sobre RNAs (Redes Neurais Artificiais), sobre controladores de processos não-lineares e seleção de controladores e sobre critérios de desempenho em sistemas de controle.

O Capítulo 3 descreve a metodologia e o estudo de caso, o qual consiste no controle de um tanque esférico.

No Capítulo 4são apresentadas simulações feitas com o método proposto no ambiente Simulink/Matlab®, variando parâmetros de treinamento e de aplicação do selecionador.

Finalmente, no Capítulo 5são apresentadas as avaliações dos resultados, as conclusões do trabalho e as propostas de trabalhos futuros.

2 Revisão Bibliográfica

Nesta seção são apresentadas revisões sobre os principais temas tratados neste trabalho: Redes Neurais Artificiais (RNA), a seleção de controladores e a utilização de RNA no controle, bem como quais os principais trabalhos apresentados em relação a estes temas.

2.1 Redes Neurais Artificiais

Uma rede neural artificial procura imitar o comportamento de uma rede de neurônios. Conforme Hajek (2005), os primeiros registros da "neurocomputação" datam da década de 1940, quando a estrutura formal de um neurônio artificial foi descrita pela primeira vez. Na década de 1950, foi criado o "Perceptron", neurônio capaz de representar funções lineares. Depois de um certo desinteresse na área, em 1980 Hopfield publicou artigos que reestimularam as pesquisas, e várias aplicações foram sendo descobertas. Atualmente, a *Inteligência Artificial* é um ramo de vastas aplicações e ainda em fase inicial de desenvolvimento. As RNAs, em suas diferentes configurações e usando diferentes métodos, tem diversas capacidades distintas, como memória, mapeamento, classificação, representação, etc.

Do mesmo modo que um neurônio biológico, um neurônio artificial possui sinapses (conexões com o exterior) e um núcleo (interpretação das informações). Conforme se verifica naFigura2.1, um neurônio artificial é composto pelas seguintes estruturas básicas: somador ponderado das entradas e das saídas, dinâmica linear e função não-linear e estática. O peso w_j é chamado de *Bias*, pois como é multiplicador de uma entrada constante, define o deslocamento linear do neurônio.

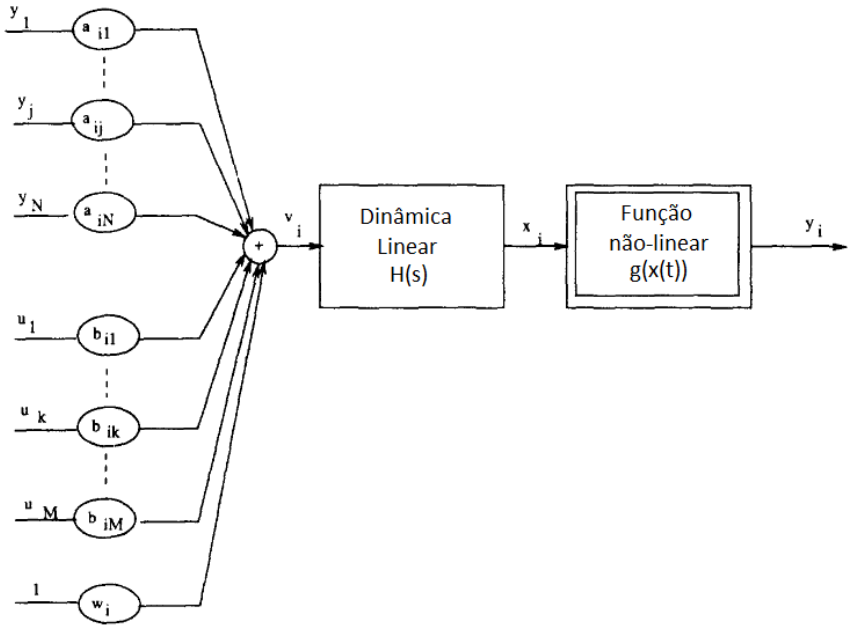


Figura2.1– Estrutura de um neurônio artificial. (adaptada de Hunt, 1992).

Desta estrutura básica surgem todas as outras variando os componentes básicos. Escolhas comuns para H(s) são:

$$H(s) = 1, \qquad H(s) = \frac{1}{s}, \qquad H(s) = \frac{1}{1+sT}, \qquad H(s) = e^{-sT} \qquad (2.1)$$

Já a função g(x) deve ser uma função nãolinear que permita ao neurônio a representação dos valores que ele deve aprender. Por exemplo, se forem valores binários, a função deve ter na imagem os valores zero e 1. Para representar uma função contínua positiva, a imagem deve estar contida entre zero e 1 (as entradas de g(x) sempre são normalizadas para valores entre zero e 1). As funções mais usadas são descritas na Tabela2.1:

Tabela2.1–Lista das funções de ativação mais usadas em redes neurais artificiais.

Treshold	$g(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$ ou $\begin{cases} 1, & x > 0 \\ -1, & x \leq 0 \end{cases}$
Sigmóide	$g(x) = \dfrac{1}{1 + e^{-ax}}$
Tangente hiperbólica	$g(x) = \tanh(x)$
Gaussiana	$g(x) = e^{-x^2/a^2}$

As redes que possuem H(s) = 1 são as que não apresentam dinâmica. Assim, a função de transferência entre V(s) e X(s) é igual a um, ou seja:

$$x_i(t) = v_i(t) = \sum_{j=1}^{N} a_{ij} y_j(t) + \sum_{k=1}^{M} b_{ik} u_k(t) + w_i \qquad (2.2)$$

Assim, o valor de x(t) será a soma ponderada dos valores da entrada, dos valores da saída edo valor de w_i, o *Bias.* O valor da saída será a função não linear dos valores de x(t). Considerando a soma ponderada como uma multiplicação de uma matriz de pesos por um vetor de variáveis, pode-se escrever a equação (2.2)conforme consta na equação (2.3), onde as matrizes A e B são chamadas de matrizes de pesos sinápticos, **w** representa o vetor de *Bias*, u(t) as entradas, e y(t) as saídas.

$$\begin{aligned} x(t) &= Ay(t) + Bu(t) + w \\ y(t) &= g(x(t)) \end{aligned} \qquad (2.3)$$

De acordo comHunt et al.(1992),os neurônios em si não têm muitas aplicações, mas suas conexões permitem que se tenham diferentes capacidades computacionais muito poderosas.Essas conexões são chamadas de sinapses. O padrão de uma RNA é possuir uma camada de entrada, um número qualquer de camadas ocultas, e uma camada de saída. O número de neurônios em cada camada depende da aplicação.

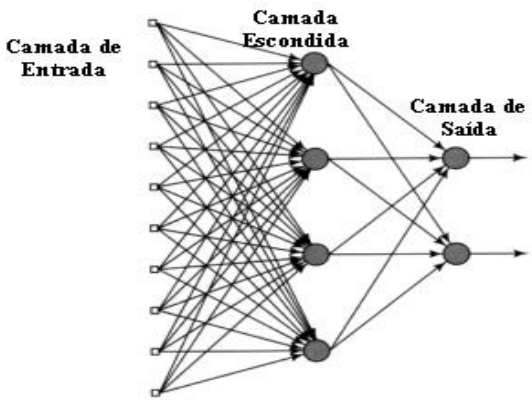


Figura 2.2 – Configuração de RNAs.

Uma rede como a da Figura 2.2 é chamada de rede *Feedforward*, pois cada camada possui conexões somente com a camada anterior, ou seja, $A = 0$. Caso $A \neq 0$, a rede se chama recorrente.

2.1.1 Treinamento de uma RNA

Os pesos sinápticos são determinados durante a fase de treinamento da rede, o qual pode ser do tipo supervisionado, quando se deseja vincular saídas conhecidas para cada combinação de entradas, ou não supervisionado, caso somente sejam apresentadas entradas para a rede.

A fase de treinamento pode empregar diferentes algoritmos. Basicamente, nestes algoritmos, os pesos são atualizados iterativamente,proporcionalmente ao fator α, chamado de taxa de aprendizagem, e a um certo erro de referência. Este erro de referência pode ser, dependendo da estrutura escolhida, a diferença entre os valores das saídas desejadas e as obtidas com os pesos atuais, ou a diferença entre um peso b_i e um valor de entrada u_i.

O treinamento de uma rede segue geralmente um padrão composto por três etapas: de aprendizado, validação e teste. Cada uma destas etapas utiliza uma parcela diferente dos dados, para garantir que a rede realmente os caracterize. A configuração padrão da ferramenta do Matlab® utiliza 70% dos dados para o treinamento, 15% para a validação e 15% para o teste. A distribuição dos dados para cada uma das três etapas pode seguir uma distribuição randômica (padrão) ou ordenada. Também nesta etapa, pode haver uma redistribuição dos dados em busca de uma melhor resposta.

Um treinamento em que a saída é conhecida e o erro entre a saída desejada e a obtida em cada iteração é utilizado para atualizar os pesos é comumente chamado de *backpropagation*, significando que o erro se propaga para trás. No caso em que o erro utilizado para a atualização é o erro da entrada, o treinamento é do tipo *forwardpropagation.*

Os algoritmos de treinamento mais comuns estão presentes na ferramenta que o software Matlab® oferece, e são descritos na Tabela 2.2. Alguns destes métodos serão aplicados no trabalho para o treinamento da RNA.

Tabela 2.2 – Funções de treinamento de redes neurais.

train()	Treinar rede neural
trainlm()	*Levenberg-Marquardtbackpropagation*
trainbr()	*Bayesianregulationbackpropagation*
trainscg()	*Scaledconjugategradientbackpropagation*
trainrp()	*Resilientbackpropagation*
traincgb()	*Conjugate Gradient with Powell/Beale Restarts*
trainbfg()	*BFGS Quasi-Newton*
trainoss()	*OneStepSecant*
traingdx()	*Variable Learning Rate Backpropagation*

Geralmente a diferença entre os tipos de treinamento é o método de otimização utilizado para fazer a atualização dos pesos e do *Bias.* Na função *trainlm(),* por exemplo, é

utilizado o método de otimização de *Levenberg-Marquardt,* sendo uma das mais comumente usadas.

Esta função utiliza a soma de quadrados como função de performance, aproximando a matriz Hessiana como $\boldsymbol{H} = \boldsymbol{J}^T\boldsymbol{J}$e o gradiente como $g = \boldsymbol{J}^T\boldsymbol{e}$, onde$\boldsymbol{J}$ é a matriz jacobiana que contem as primeiras derivadas dos erros da rede com relação aos pesos e ao *Bias,* e $\boldsymbol{e}$ é um vetor de erros da rede. A função de atualização de pesos é então:

$$\boldsymbol{a}_{k+1} = \boldsymbol{a}_k - (\boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I})^{-1}\boldsymbol{J}^T\boldsymbol{e} \qquad (2.4)$$

A peculiaridade do método é o parâmetro μ. Quando este escalar é nulo, a equação acima é igual à do método de Newton, usado na função *trainbfg(),* usando a matriz Hessiana aproximada. Quando o escalar é muito grande, a equação passa a ter gradiente decrescente com um passo pequeno. Como a equação de Newton só é precisa com erros pequenos, o valor de μ é reduzido a cada iteração bem sucedida, acelerando sua convergência. Por isto este é chamado de método de otimização não linear.

Já a função de treinamento *Resilientbackpropagation* atualiza os pesos e *Biases* somente em função do sinal do erro. É usada quando, por uma série de efeitos, o passo do treinamento permanece pequeno, ainda que os pesos estejam longe de seus valores ótimos. Assim, cada método tem propriedades matemáticas diferentes e é aplicável a uma determinada situação.

Para determinar o número de iterações a serem feitas em um treinamento, ainda é possível determinar os critérios de parada e de análise do treinamento, conforme se confere naTabela 2.3. São critérios de parada para o treinamento os parâmetros de 1 a 6. De 7 a 9 são valores de incremento, decremento e valor inicial da taxa de aprendizagem. 10 e 11 são parâmetros auxiliares para gerar o código relativo à rede e para mostrar a interface gráfica de treinamento.

Tabela 2.3 – Parâmetros de treinamento de uma rede neural no Matlab®.

1. net.trainParam.epochs	Número limite de iterações no treinamento
2. net.trainParam.goal	Objetivo de performance
3. net.trainParam.max_fail	Máximo de falhas na validação
4. net.trainParam.min_grad	Gradiente de performance mínimo
5. net.trainParam.time	Tempo máximo de treinamento
6. net.trainParam.mu_max	Taxa de aprendizado máxima
7. net.trainParam.mu	Taxa de crescimento inicial
8. net.trainParam.mu_dec	Fator de decréscimo de taxa de crescimento
9. net.trainParam.mu_inc	Fator de incremento do taxa de crescimento
10. net.trainParam.showCommandLine	Gera uma script de saída
11. net.trainParam.showWindow	Mostra GUI de treinamento

2.1.2 Configurações de RNAs

Como há várias possibilidades de conexões, de funções de transferência e de pesos sinápticos, existem diversas configurações diferentes de redes neurais. Além de apresentar as modalidades de não dinâmica (H(s)=1) ou dinâmica (demais funções H(s)),

uma rede pode ser supervisionada, quando os valores de saída desejados são conhecidos, ou não supervisionada, quando desconhecidos. Em *Artificial Neural Networks: a ReviewofTrainning Tools(Baptista and Morgado-Dias, 2013)*, pode-se encontrar uma extensa lista de denominações representando 52 configurações de redes e 49 diferentes algoritmos, utilizando variações do modelo mostrado pela Figura2.1.

Alguns exemplos de configurações bastante utilizadas são as seguintes:

- *Perceptron*

O *perceptron* é uma das configurações mais antigas, em que não há dinâmica (H(s)=1), não há realimentação (A=0) e a função de ativação é a função do tipo *Treshold.* Um *perceptron* pode ser usado sozinho, com a capacidade de representar funções linearmente separáveis, como o *AND* lógico, por exemplo, ou como *perceptron* de múltiplas camadas, possuindo a capacidade de representar funções não linearmente separáveis. Nesse caso, o treinamento é supervisionado, sendo os pesos de cada iteração atualizados proporcionalmente ao erro da saída (*backpropagation*).

- ADALINE (Adaptive Linear Neuron):

Estrutura que surgiu simultaneamente ao Perceptron e de configuração muito parecida, com a exceção de ter como g(x) uma função linear.

- Mapas auto-organizáveis (Redes de Kohonen):

Se trata de uma rede com aprendizado não supervisionado, usando um algoritmo de aprendizado competitivo. Os padrões de entrada são apresentados para a rede, e os pesos são atualizados de forma a compensar um neurônio vencedor. No fim do treinamento, a rede representa o padrão de entrada, por isso o nome da rede.

- Rede de Hopfield:

Aprendizado não supervisionado, não existindo camada de saída. A atualização dos pesos é feita em relação à diferença entre os pesos e os valores da entrada. Assim, uma rede treinada, assim que submetida a uma certa saída, irá tender aos valores de entrada com que foi treinada. A rede também é chamada de memória associativa.

2.2 Redes Neurais Artificiais eControle por seleção de controlador

O controle por seleção de controlador apresenta algumas boas características: ativar o melhor controlador em relação ao ponto de operação,utilizar estruturas simples e de fácil sintonia e o controlar eficientemente processos de natureza variada.Assim, existe uma grande e variada literatura sobre o tema, seguindo uma característica básica: envolve otimização de algum critério. Pode ser otimizaçãodo erro, ou o tempo de resposta, ou o sobrepasso, ou de outras medidas de desempenho, até mesmo várias delas ao mesmo tempo.

Esta otimização também pode ser feita de diferentes maneiras, utilizando diferentes funções. Os controladores candidatos podem ser de diferentes naturezas, lineares ou não, e os métodos podem utilizar a ação de vários controladores ao mesmo tempo, como soma ponderada, ou de só um deles em cada ponto.

Em Fu e Chai(2007) é proposto um método de controle que varia a ação do controle entre um controladoradaptativo linear e um controlador neural não linear. Este é um

exemplo de uma rede neural utilizada como controlador. Usando duas estruturas de controle diferentes, a performance do sistema é otimizada.

A seleção de controladores é comum na área de processos também. Em Chatzidoukas et al.(2003)foi proposto um método de controle para um reator de polimerização que opera sujeito a diferentes frações dos reagentes, baseado em seleção. Segundo o autor, o desempenho da planta seria melhorado com a sintonia do controlador em paralelo com sua seleção.

Behroozsarand e Shafiei(2011)propuseram uma resolução ao problema de otimização em função de variados critérios de desempenho, utilizando um algoritmo genético. Segundo os autores, muitos problemas de controle envolvem a otimização simultânea de múltiplos critérios de desempenho, o que dá origem a um conjunto de soluções ótimas, conhecidas como soluções ótimas de Pareto. Em seu trabalho, Behroozsarand e Shafiei aplicam um selecionador neural para escolher o parâmetro que será otimizado, usando o resultado para sintonizar um controlador PID que será aplicado no controle de uma coluna de destilação.

Há métodos também que utilizam a otimização para fazer a seleção do controlador *off-line*. *Yin et al.(2013)* propoem um critério de seleção que avalia tanto a performance quanto o custo computacional da aplicação das estruturas de controle.

Os algoritmos genéticos têm lugar na robótica igualmente. Um exemplo disto é o que se trata no artigo de Hultmann Ayala e Dos Santos Coelho(2012). O artigo apresenta o desenvolvimento e a sintonia de dois controladores PID a partir do método NSGA – II (Non-dominated Sorting Genetic Algorithm – Algoritmo Genético de Seleção não-Dominada) para o controle de um manipulador robótico.

Como visto nos artigos citados acima, a seleção de controladores é uma prática altamente aplicada na teoria de controle. Do mesmo modo, a flexibilidade dos algoritmos neurais faz com que sejam frequentemente aplicados no controle, seja como controlador, como selecionador, como modelo do processo, etc.

2.3 Critérios de desempenho de sistemas de controle

O controle clássico trata como principais critérios de desempenho quatro medidas básicas: sobrepasso, tempo de acomodação, tempo de subida e erro em regime permanente. A definição de cada critério pode variar, mas cada critério avalia certa característica do sistema:

- Sobrepasso (*overshoot)*: o sobrepasso é a diferença entre o ponto máximo atingido pela resposta do sistema a um degrau na entrada em malha fechada e o valor alvo (*setpoint*). Pode ser calculado em termos percentuais, dividindo esta diferença pela amplitude do degrau aplicado, ou seja, no valor alvo, ou em termos absolutos, sem a divisão;

- Tempo de acomodação: é o tempo que a resposta do sistema toma para atingir o valor de equilíbrio. Como a taxa de convergência ao valor de equilíbrio diminui conforme se aproxima deste, a acomodação exata é extremamente demorada. Assim, para calcular o tempo de acomodação é normalmente tolerado um erro relativo em torno do valor alvo (Bazanella e da Silva Jr., 2005).

- Tempo de subida: é uma medida que serve para avaliar a velocidade de resposta da curva. Pode ser o intervalo que a curva leva para atingir metade do valor alvo, ou o tempo que a resposta do sistema leva dos 10% do seu valor alvo até os 90%, etc.

- Erro em regime permanente: o e_{rp} é a diferença entre o valor em que a resposta do sistema acomoda e o valor alvo. Quando um controlador integral é adicionado, o erro em regime permanente tende a zero se a referência for constante.

A Figura 2.3 mostra três dos critérios de desempenho descritos acima, em um caso de resposta de um sistema subamortecido. Na figura, *Mp*é o sobrepasso absoluto,t_s é o tempo de acomodação, t_d é o tempo de subida. Em casos de sistemassub-amortecidos, em que a curva sempre cruza o valor alvo, o tempo t_r em que isto acontece pode ser utilizado como tempo de subida da mesma maneira.

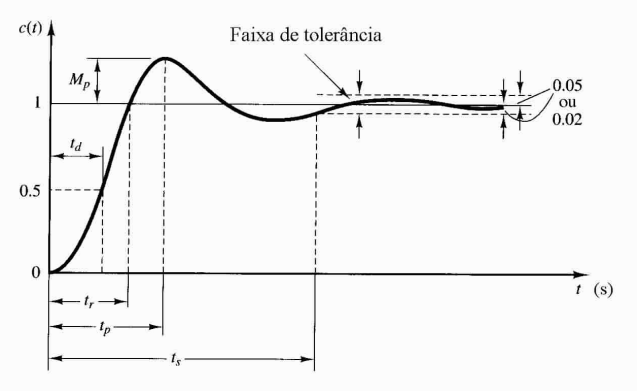


Figura 2.3 – Demonstração dos critérios de desempenho (retirada de Ogata,1997).

Além destes critérios mais básicos, existem ainda outros critérios que podem ser aplicados, tais como os critérios estatísticos, que consideram a variabilidade da malha de controle ao longo do tempo. Alguns destes critérios são descritos por (Kempf, 2003). Entre eles, pode-se citar a regra FCOR (filtragem e correlação), que compara a variância da malha de controle com um padrão de desempenho ideal (*controlador de variância mínima*) para calcular um índice de desempenho absoluto.

3 Metodologia

São três os principais passos para a implementação do método de seleção de controladores: aquisição de dados para o treinamento da rede, o treinamento em si e a aplicação da rede no sistema em malha fechada. Neste capítulo, serão detalhados o método e as etapas que o compõe.

3.1 Descrição

Como já comentado, este método consiste em utilizar uma RNA como selecionador de controladores para um processo não linear. Assim, tal rede deve ser treinada de forma a adquirir a capacidade de seleção e os dados para este treinamento devem ser obtidos a partir da planta que se deseja controlar, podendo ser interpretados e adquiridos de diversas maneiras.

Após a aquisição dos dados, deve-se compor um conjunto de entradas e saídas para o treinamento da RNA. Neste método, cada conjunto de entradas corresponde a n saídas que consistem das notas dos n controladores. Estas notas são calculadas utilizando os critérios de desempenho adotados, ponderados e somados. A intensão é que a nota mais alta corresponda a um controlador de melhor desempenho.

O treinamento da rede é feito segundo a configuração de ajuste de curvas, ou seja, a rede irá representar a curva da relação entre as entradas e as saídas.

Depois de treinada, a rede pode ser aplicada ao controle do processo. A planta oferece os dados de entrada, a rede calcula as notas dos controladores e o de maior nota é selecionado para controlar o processo naquele dado instante. A cada instante as notas são recalculadas e outro controlador pode assumir o controle. Assim, para que não haja transições bruscas, cada vez que um controlador diferente atuar ele deve ter suas condições iniciais correspondentes às condições atuais do processo.

A vantagem deste método é a possibilidade de obter um acréscimo de desempenho utilizando controladores de fácil sintonia e que já estejam atuantes no processo. Mas inevitavelmente, por ser um método numérico (considerando o treinamento da RNA), incumbe em maior custo computacional em relação a outros métodos de controle para a obtenção de um bom selecionador.

A estabilidade do método depende da ordem do sistema em que é aplicado, e também das configurações que se utilize, devendo ser analisada separadamente para cada aplicação.

3.1.1 Obtenção dos dados para o treinamento

Os dados necessários para o treinamento são os desempenhos de cada um dos controladores candidatos em variados pontos de operação e as entradas da planta que resultaram nestes desempenhos. Estes dados podem ser adquiridos por meio de simulação matemática em um modelo da planta, mas também, caso o custo matemático e/ou computacional de aquisição do modelo seja muito alto, é possível utilizar dados adquiridos empiricamente na planta que se deseja controlar.

O desempenho dos controladores pode ser avaliado a partir dos critérios que se considerem importantes: tempo de acomodação, tempo de subida, sobrepasso, etc. Também é possível atribuir pesos diferentes aos critérios.

3.1.2 Treinamento da rede neural

O software Simulink/Matlab® oferece diversas ferramentas de redes neurais, possibilitando uma boa integração entre suas interfaces gráficas, os códigos escritos (*Scripts*) e os esquemas gráficos (*Simulink)*. Assim, é necessário somente que se insiram na ferramenta escolhida os parâmetros desejados e ela retorna uma rede treinadacom esses parâmetros. Uma maior interferência durante o processo também é possível.

Em primeiro lugar, como se deseja que a rede calcule os pesos sinápticos tendo como referência saídas determinadas, por definição, o treinamento deverá ser supervisionado, ou seja, devem-se apresentar para o treinamento as saídas desejadas. As saídas desejadas são as notas de cada controlador em cada simulação, obtidas pelo passo anterior.

Em segundo lugar, é importante destacar que a ferramenta faz um tratamento dos sinais antes e depois do treinamento. No primeiro tratamento, todas as variáveis são normalizadas entre -1 e 1, ou entre zero e 1, para variáveis estritamente positivas. Isto é necessário porque se as variáveis forem de magnitude muito grande, os pesos sinápticos seriam de dimensão muito pequena, estando muito vulneráveis a erros numéricos. O tratamento posterior faz o retorno das variáveis às dimensões originais.

O objetivo da rede neural é o de aproximar uma função contínua, de múltiplas variáveis. O número de variáveis de entrada depende de que sinais do processo se deseja considerar. Por ter a função de modelar uma relação entre entrada e saída, náo é necessária atuação dinâmica, sendo então do tipo estática (H(s) = 1).

Como as saídas são contínuas, a função de transferência deve também ser contínua. Assim, deve ser uma das três últimas apresentadas na Tabela2.1. Podem-se fazer diferentes simulações para determinar qual a melhor escolha. A função padrão utilizada na*toolbox*do Matlab®é a *sigmoide.*

Por fim, são determinadas as funções de treinamento e de desempenho da rede. A função com melhor taxa de convergência para casos de representação de relações entrada-saídaé a *trainlm()*, sendo a função utilizada caso os dados não exijam alguma outra. A função de desempenho padrão é a de erro médio quadrático.

3.1.3 Aplicação da RNA na malha de controle

Depois de treinada, a RNA pode ser utilizada na malha de controle como selecionador de controlador. A função do bloco que atua como controlador é, sucintamente, fazer com que atue no processo somente o controlador de maior nota. No entanto, para que isto seja possível, é necessário que se atenda a alguns pré-requisitos:

- a saída da RNA deve ser interpretada, determinando qual controlador será ativo e quais inativos;
- os controladores de menor nota devem ter seus sinais desconsiderados;
- ao ser ativado, o controlador deve ter condições iniciais que não causem oscilações no sistema;

- deve haver um mecanismo que garanta que qualquer controlador que esteja ativo possa efetivamente controlar o processo.
- no caso de dois controladores de nota muito próxima, o selecionador pode trocar de controlador demasiadamente.

O dois primeiros pré-requisitos são atendidos por meio da criação de uma função do tipo *S-function*, chamada de *interpreta*no diagrama. Esta função tem como entrada o sinal de saída da RNA e o valor do nível no tanque. Ela avalia qual controlador tem a maior nota, multiplica o seu sinal de controle por 1 e dos outros por zero. Este mesmo sinal, que é utilizado para selecionar um dentre os sinais de controle, reinicia todos os controladores assim que é selecionado um controlador diferente do atuante.

Como condições iniciais dos controladores se podem usar duas opções: ou uma realimentação do sinal de controle atuante, ou o sinal de controle correspondente ao ponto atual do processo. No primeiro caso, tem-se um sinal de controle sempre contínuo, pois o controlador que será ativado começará a atuar no ponto em que o outro parou. No segundo caso, tem-se uma saída contínua, pois o sinal de controle inicial é equivalente ao ponto em que o processo se encontra.

O terceiro pré-requisito é atendido por um somatório dos sinais. Como os controladores não atuantes têm suas ações de controle multiplicadas por zero, o resultado desta soma sempre será igual ao sinal do controlador atuante.

O último pré-requisito pode ser atendido aplicando uma margem de troca. Se a diferença entre as notas for menor que certo limite, a troca não é feita, mantendo controlador atual atuante. Esta margem é aplicada na função *interpreta.*

O esquema de simulação pode ser visto naFigura 3.1.

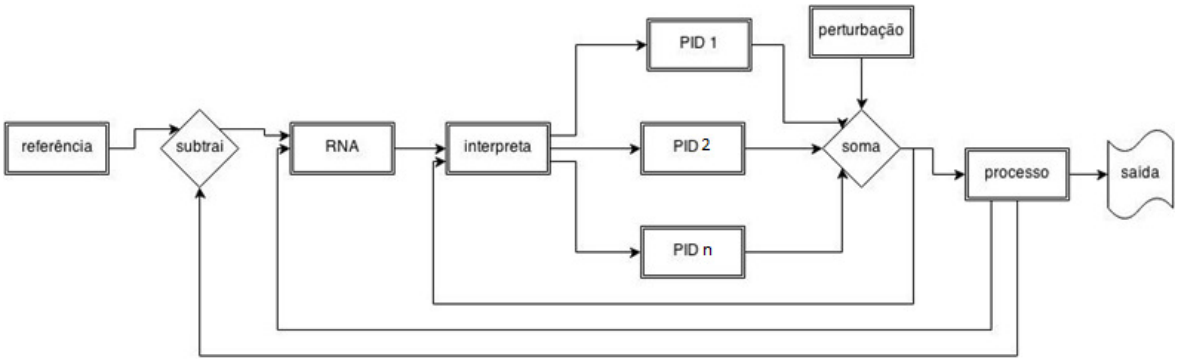


Figura 3.1 – Esquema de simulação com o selecionador neural.

3.2 Estudo de caso: tanque esférico

Um tanque esférico pode ser representado por um sistema não linear devido à relação entre o volume e o nível. Além disso, a vazão de saída tem uma relação com a raiz quadrada do nível, considerando que controlada por uma válvula.Assim, as equações que regem o sistema são mostradas da equação (3.1) à (3.3):

$$\frac{dM}{dt} = \rho f_{in} - \rho f_{out} \,, \qquad (3.1)$$

$$f_{out} = C_D A_o \sqrt{2gh} = k\sqrt{h}\,, \tag{3.2}$$

$$M = \rho \frac{\pi}{3} h^2 \left(\frac{3}{2} D - h\right), \tag{3.3}$$

onde M é a massa no tanque, f_{out} é a vazão de saída, f_{in} a vazão de entrada, C_D o coeficiente de descarga, A_0 a área inicial de descarga da válvula, ρ a densidade do líquido, D o diâmetro da esfera, g a aceleração da gravidade e h o nível no tanque. A Figura 3.2demonstra o esquema do tanque esférico.

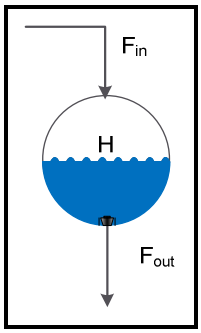


Figura 3.2 – Representação do tanque esférico.

A equações do tanque esférico são simuladas em ambiente Simulink® por meio de uma função *S-function*, descrita no anexo B. A linearização do tanque para um ponto de operação genérico (γ) bem como a composição da função de transferência do tanque são apresentadas no apêndice A.

3.2.1 Projeto dos controladores

Foram escolhidos três pontos de operação para projetar três diferentes controladores. Esses pontos são γ = 3,75cm,γ= 11,25 cm e γ = 18,75 cm. O diâmetro D do tanque vale D=22,5cm, a constante k, resultado da multiplicação da área inicial da válvula por sua constante de descarga e pela raiz quadrada da constante gravitacional, vale k= 17,6980$cm^{5/2}$/s. Esta constante pode ser modificada durante o processo para simular uma abertura ou fechamento de válvula, usada como uma perturbação na saída do processo. Para as simulações, o valor utilizado será 70% do valor inicial de k, 12,4 $cm^{5/2}$/s. A função de transferência, substituindo as constantes e os três pontos de equilíbrio, são as seguintes:

$$\frac{\Delta H(s)}{\Delta F_{in}(s)} = \frac{{}^{1}/_{\pi\gamma(D-\gamma)}}{\left(s + {}^{k}/_{2\gamma\sqrt{\gamma}\pi(D-\gamma)}\right)} \tag{3.4}$$

$$\left.\frac{\Delta H(s)}{\Delta F_{in}(s)}\right|_{\gamma=3,75} = \frac{0,0045}{s + 0,0145} \tag{3.5}$$

$$\left.\frac{\Delta H(s)}{\Delta F_{in}(s)}\right|_{\gamma=11,25} = \frac{0,0025}{s + 0,0046} \tag{3.6}$$

$$\left.\frac{\Delta H(s)}{\Delta F_{in}(s)}\right|_{\gamma=18,75} = \frac{0,0045}{s + 0,0065} \tag{3.7}$$

As constantes de tempo nos três casos são respectivamente 68,96 s, 217,4 s e 153,84 s, o que resulta em tempos de acomodação de aproximadamente 275,8 s, 869,6 s e 615,36 s, para saltos unitários Como se pode notar pelos valores, se trata de um processo de convergência relativamente lenta, já que a menor constante de tempo passa de quatro minutos.

Teoricamente, um controlador proporcional já é satisfatório para o controle de um processo de primeira ordem. No entanto, o controlador proporcional não anula o erro em regime permanente e, se for requerida uma grande precisão em regime, a ação de controle pode ser muito grande, saturando a variável de controle. Outra desvantagem do controle puramente proporcional é que pode levar à instabilidade quando há dinâmicas desconsideradas no modelo. Pode-se então aplicar um controlador PI (Proporcional e Integral). O fator integral leva o erro em regime permanente a zero, o que permite diminuir o fator proporcional e obter-se um melhor desempenho. O controle derivativo não é aconselhado para processos de primeira ordem, já que o controle PI já é satisfatório.

O controlador PI projetado é descrito pela Equação (3.8):

$$C(s) = k_p + \frac{k_i}{s} = k_p \frac{(s + k_i/k_p)}{s} \qquad (3.8)$$

$$C(s)G(s) = \frac{a}{s+b} \frac{k_p(s + k_i/k_p)}{s} \qquad (3.9)$$

O controlador possui um zero em k_i/k_pe um polo em zero. O projeto dos valores de k_i e k_p pode ser feito pelo método do lugar das raízes. A escolha mais simples para o valor do zero do controlador seria iguala-lo ao valor do polo do processo ($b = k_i/k_p$), fazendo com que o sistema em malha fechada tivesse um comportamento de primeira ordem.

No entanto, se o valor do zero do controlador for muito próximo do polo do processo, mas não igual, a saída do processo pode ter oscilações indesejadas, dependendo do valor de k_p escolhido. A Figura 3.3 mostra esse fenômeno, havendo pontos em que a parte imaginária é diferente de zero, causando oscilações.Não é possível calcular precisamente os valores de ganho que previnem este comportamento, pois variam em função da distância entre o polo do sistema real e o zero do controlador.

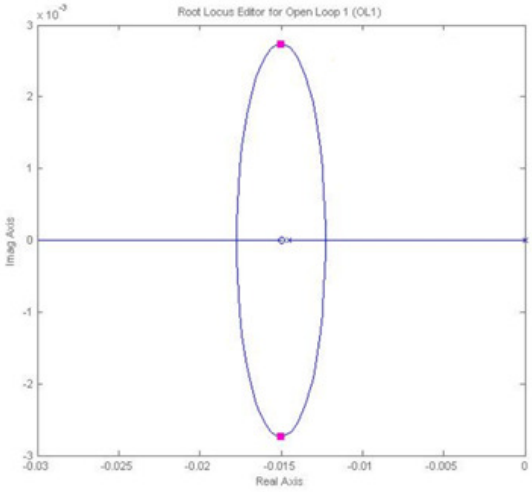


Figura 3.3 – Lugar das raízes para polo e zero próximos, mas diferentes.

Pode-se, no entanto, determinar um valor para o zero do controlador e calcular as constantes considerando um sistema de segunda ordem. A escolha de um zero menor que o polo do controlador não se mostrou vantajosa, pois o polo dominante da planta em malha fechada é menor que o valor do zero, resultando em tempos de acomodação maiores do que os do próprio processo.

Por outro lado, se o módulo do zero for muito maior que o do polo, seria necessário muito esforço de controle para estabilizar o processo. Assim, é feita a escolha do zero para que seja possível tanto evitar oscilações quanto manter o esforço de controle em níveis aceitáveis.

Para o primeiro ponto de operaçãoconsiderou-se um tempo de acomodação alvo de 50s, ou seja, mais de 4 vezes mais rápido que o processo em malha aberta.Para tanto, o zero do controlador foi escolhido como 0,02. Para o segundo ponto, buscou-se ter um tempo de acomodação em torno de 200 s, e para o terceiro em torno de 150 s. Seguindo o mesmo método, chegou-se aos valores daTabela 3.1, que serão usados na aquisição de dados para o treinamento.

Tabela 3.1 - Valores de k_i ek_p para os pontos de operação escolhidos.

Linearização	k_p	k_i
$\gamma = 3{,}75$	10,3	0,206
$\gamma = 11{,}25$	5,28	0,043
$\gamma = 18{,}75$	4,17	0,018

3.2.2 Aquisição de dados para o treinamento

A rede neural que desempenhará o papel de selecionador deverá ser treinada com os valores dasentradas e das saídasequivalentes. Assim, o primeiro passo é adquirir estes dados para treinamento.

A aquisição será feita em simulação do tanque em malha fechada com um controlador PI, o qual terá seus parâmetros (k_i e k_p) modificados a cada etapa. A estrutura da simulação pode ser vista naFigura 3.4:

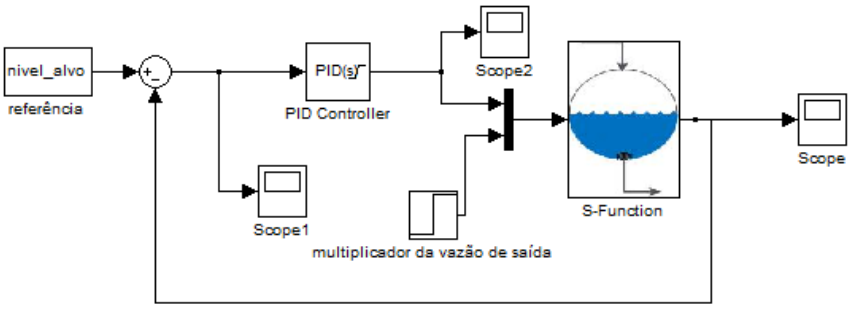


Figura 3.4 – Estrutura de simulação para aquisição dos dados.

O tanque, modelado pela função S-function, possui três parâmetros de entrada. Dois deles podem ser vistos na estrutura de simulação:a vazão de entrada, funcionando como ação de controle, e o *step*do coeficiente de vazão da saída. Este último tem efeito sobre a variação da vazão de saída do tanque, que será considerada como uma perturbação no processo. O terceiro sinal, dado como parâmetro do tanque, é o seu nível inicial.

A simulação é feita por um *Script* do Matlab®, variando parâmetros a cada etapa: o nível alvo e o nível incial do tanque, que podem assumir valores entre zero e 22,5 cm, e os parâmetros do controlador (k_i e k_p), entre os candidatos da Tabela 3.1. Em cada simulação em que o nível inicial do tanque for diferente de zero, é necessário fornecer ao controlador uma condição inicial correspondente ao nível inicial do tanque. Estas condições iniciais foram calculadas em estado estacionário no tanque.

Estes parâmetros foram escolhidos para as simulações pois, por tratar-se de um processo não linear, seu comportamento varia em função do ponto de operação. São variados os parâmetros do controlador de modo que cada simulação utilize um dos três controladores candidatos.

Para que o controlador só apresentasse valores de vazão na entrada do tanque entre os valores aceitáveis (zero a 66,7 cm³/s), foi adicionada uma saturação ao bloco PID.

Para que seja possível calcular o critério de variância da resposta dos controladores, é necessário realizar uma segunda simulação, aplicando perturbações e ruído branco ao sistema, conforme apresentado na Figura 3.5.

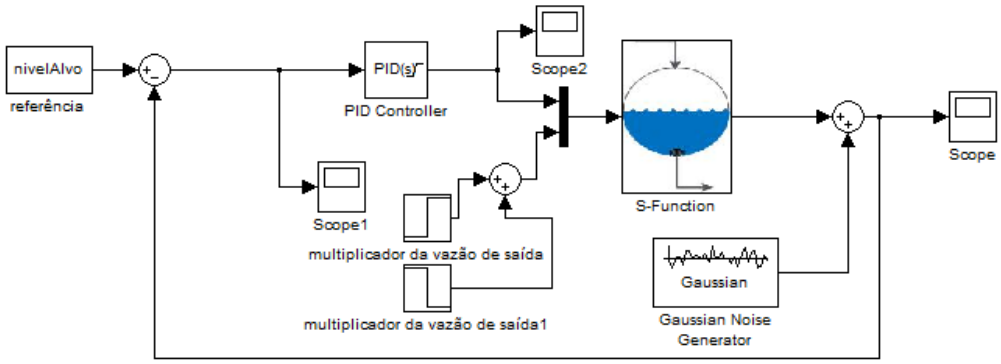


Figura 3.5 – Simulação em malha fechada com perturbações e ruído branco.

3.2.3 Avaliação dos dados adquiridos

Como resultado para cada simulação há um conjunto dados de entrada (relativos ao nível inicial e ao nível alvo) e de saída (notas dos controladores), de forma que possam ser utilizados para o treinamento da rede neural.

As notas dos controladores são calculadas em função dos critérios de desempenho escolhidos: sobrepasso, tempo de acomodação, tempo de subida e variância. O sobrepasso é calculado como a diferença entre o ponto máximo atingido pela curva dividido pela amplitude do salto aplicado, conforme a Equação (3.10).

$$M\% = \frac{max - ref}{ref - xs} * 100\% \qquad (3.10)$$

Na equação M% é o sobrepasso porcentual, *max*o ponto máximo atingido pela nível, *ref* o nível alvo e *xs*o nível inicial.

O tempo de acomodação (t_s) é o tempo que a curva leva para estabilizar dentro de uma margem de 2% da amplitude do salto. Já o tempo de subida, é o tempo que a curva leva dos 10% aos 90% desta amplitude.

$$t_s = t' sse\ |nível(t') - ref| \leq 0{,}02 * |ref - xs|\ \forall\ t > t' \qquad (3.11)$$

$$t_u = t_{90\%} - t_{10\%} \qquad (3.12)$$

A variância do sistema é calculada através do método implementado por Kempf, 2003, na função *fcor()* descrita no apêndice A.

Os valores dos critérios de desempenho são armazenados em matrizes n por 3, em que n é o número de simulações feitas e 3 o número de controladores usados.

De posse de todos os critérios de desempenho, é necessário compor uma nota que leve todos ou parte deles em consideração. Para tanto, é necessário que tenham a mesma dimensão. A estratégia adotada é dividir cada uma das matrizes de desempenho por seu valor máximo, para que independentemente da unidade original, estivessem mapeadas entre zero e 1. Assim, foram determinadas as seguintes notas(equações 3.13 a 3.15):

$$notaTempo = 2 - (subidaNorm + acomNorm) \qquad (3.13)$$

$$notaVar = 2 - (sobrepassoNorm + varNorm) \qquad (3.14)$$

$$notaTotal = notaVar + notaTempo \qquad (3.15)$$

Nas equações (3.13) a (3.15), *subidaNorm, acomNorm, sobrepassoNorm*e *varNorm* são, respectivamente os vetores de tempo de subida, tempo de acomodação, sobrepasso e variância normalizados. Como cada um dos critérios é inversamente proporcional ao desempenho do sistema, seus sinais foram invertidos para que uma maior nota representasse efetivamente um melhor desempenho. Foi feita a separação das notas em *notaTempo* e *notaVar* para que nas simulações fosse possível utilizar estas de forma independente.

3.2.4 Treinamento da rede para o estudo de caso

Para o caso do tanque esférico, são utilizados os parâmetros padrão de treinamento da rede neural, exceto pelo número de neurônios da camada oculta, que foi aumentado visando melhora nos resultados. Cada treinamento pode ser avaliado pelos seus valores de erro médio quadrático e de gradiente. Quanto menor a dimensão destes valores, melhor a rede representa a relação entrada-saída.

3.2.5 Estrutura de simulação

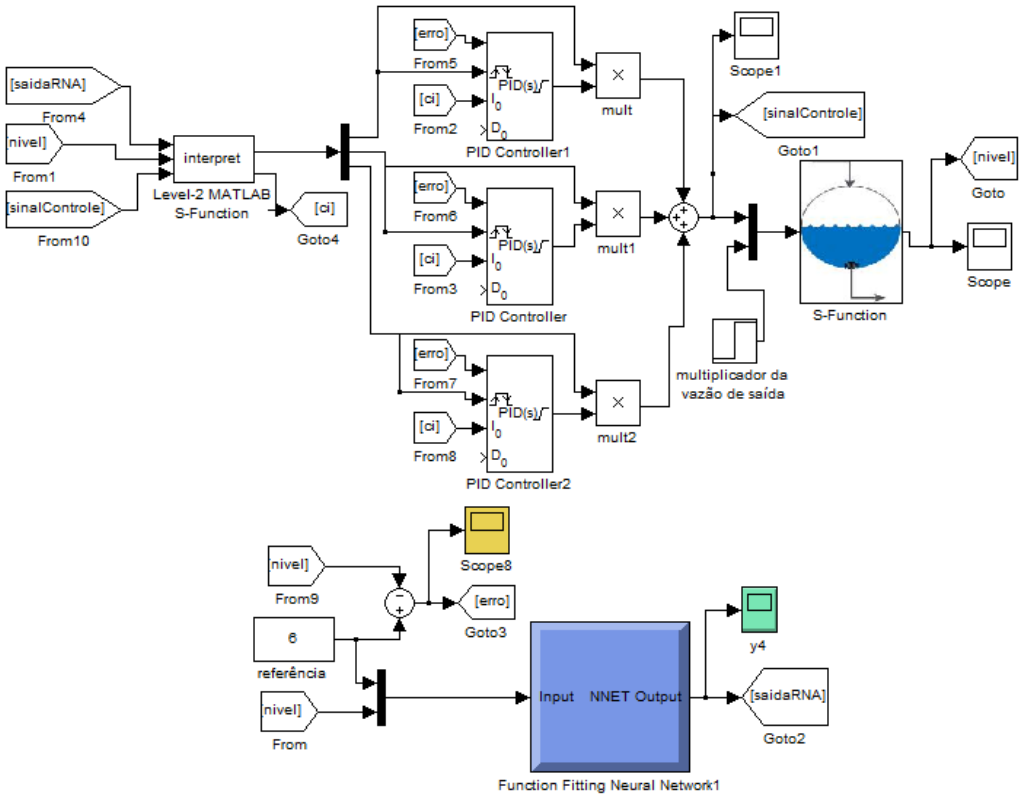


Figura 3.6 –Simulação do processo com o selecionador de controlador em malha fechada.

A função “interpret” analisa as saídas da rede neural e o valor do nível atual, gerando os sinais de seleção de controladores e o sinal de condição inicial para os integradores dos blocos PID. A seleção é efetivada de maneira que os sinais de controle dos blocos inativos são multiplicados por zero, e do bloco ativo por 1. Cada vez que se alterna entre um bloco inativo e um ativo, todos os blocos são “resetados”, ou seja, recomeça sua operação a partir dos pontos iniciais.

Os blocos PID possuem as seguintes entradas: erro entre valor de nível alvo e valor atual, sinal de *reset* e condição inicial para o integrador. A quarta entrada apresentada é a condição inicial do filtro interno, que é considerada nula quando não atribuído um valor a ela.Os sinais dos blocos de PID são somados, e o sinal resultante é o sinal efetivamente aplicado para controlar o processo. Como somente um controlador terá saída diferente de zero, somente um atua de cada vez.

A fim de encontrar a melhor maneira de fazer a escolha e a alternância entre os controladores, são realizadas diversas simulações, variando alguns parâmetros. As condições iniciais dos blocos de PID podem ser calculadas a partir do nível do tanque ou a partir do sinal de controle. As entradas da rede são duas: nível atual do tanque e nível alvo. Este último pode ser substituído pela amplitude do salto (nível alvo – nível atual).

4 Resultados

Foram feitas várias simulações de forma que se encontrasse a melhor maneira de aplicar as redes neurais como selecionador.As primeiras simulações são feitas como experimentos, a fim de variar os parâmetros certos para encontrar esta melhor maneira.

4.1 Resultados dos treinamentos das RNA

O software apresenta uma série de informações sobre o treinamento. É possível verificar qual foi o fator de parada utilizado, qual o valor das demais variáveis de treinamento, verificar o gráfico da performance, da variação do erro, entre outros.

A Figura 4.1 mostra dados de um treinamento com 2 variáveis de entrada, 3 de saída, uma camada escondida com 20 neurônios e o restante dos parâmetros de simulação em valor padrão. São os resultados do treinamento da primeira rede neural testada neste trabalho. As especificações desta rede podem ser vistas naFigura 4.1.

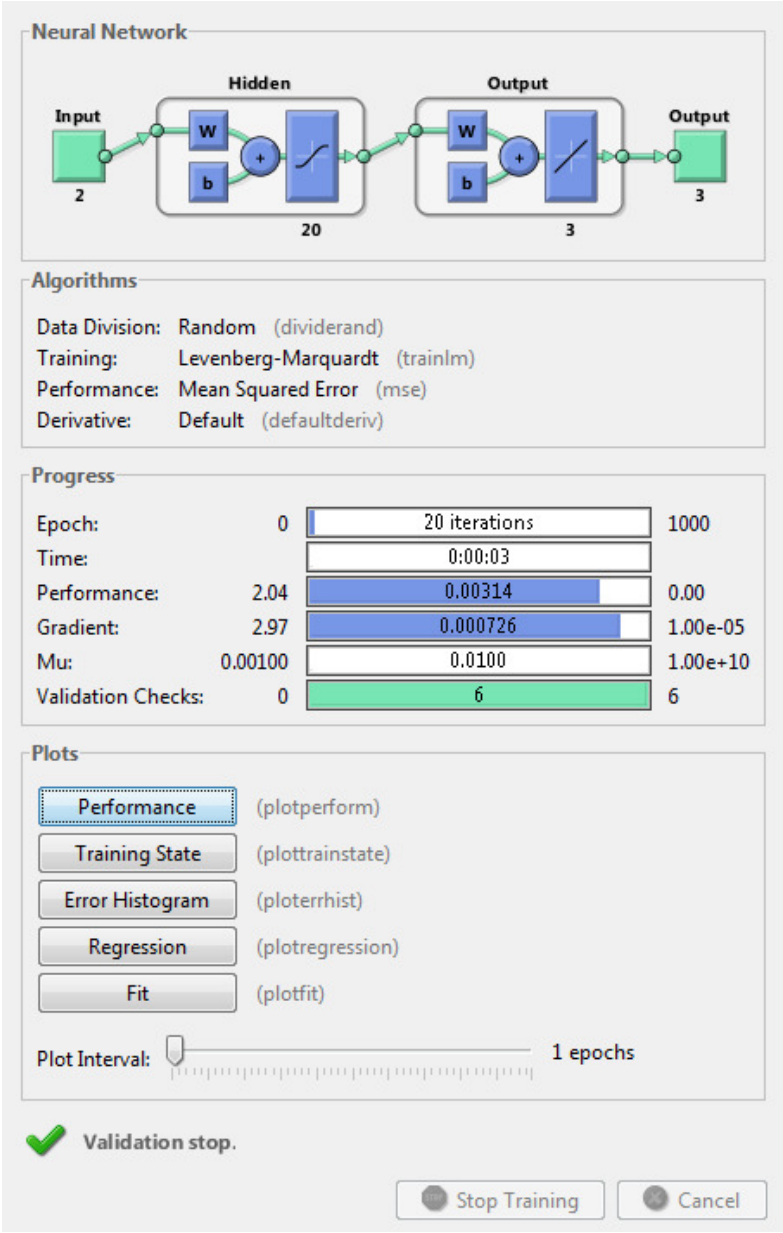


Figura 4.1 – Interface gráfica que descreve as características de treinamento.

É possível verificar que o treinamento representado na Figura 4.1 atingiu o critério de parada do número de validações. Os outros dados do treinamento mostrados são, respectivamente: 20 iterações, 3 s de duração, erro médio quadrático de 0,00314, gradiente de 0,000726, taxa de aprendizagem de 0,001.

Por apresentar erro da ordem de 10^{-3} e gradiente da ordem de 10^{-4}, o treinamento é considerado de sucesso, ou seja, há uma boa representação entre as saídas dadas e as calculadas pela rede.

4.2 Variações na etapa de treinamento

Há diversos parâmetros da rede que podem ser modificados buscando um melhor resultado no treinamento. No entanto, não são tão significantes para o resultado do controle quanto o cálculo das notas, por exemplo. Somente podem melhorar o quanto as saídas da rede representam as saídas dadas para o treinamento.

Desse modo, serão variados parâmetros em busca do menor erro e melhor desempenho na fase de treinamento, tendo como base a terceira configuração de simulação, detalhada no Capítulo 4.3.

4.2.1 Variação do número de neurônios

Fazendo testes iniciais, pode-se perceber que um número de neurônios entre 10 (padrão) e 20 apresentava um desempenho na ordem de 10^{-1}. Não existe uma regra sobre qual a quantidade de neurônios apresentará o menor erro, já que o treinamento depende muito dos dados do processo e de outros parâmetros. No entanto, variando o número de neurônios da camada oculta entre 20 e 30 foi possível chegar a um erro na ordem de 10^{-3}.

Como se pode verificar na Tabela 4.1, o menor erro ocorre com 24 neurônios, assim como um dos menores gradientes. Também é o treinamento com o maior número de iterações, o que resulta em um maior custo computacional. No entanto, como este não é o foco do trabalho, este número de neurônios será utilizado para os demais experimentos.

Tabela 4.1 –Experimentos variando o número de neurônios da camada oculta.

neurônios	iterações	desempenho	gradiente	taxa de aprendizagem
10	19	0,0257	0,062	0,01
15	14	0,0151	0,0127	0,01
20	9	0,0109	0,0071	0,01
21	10	0,0224	0,0197	0,1
22	14	0,00959	0,00815	0,01
23	12	0,0131	0,00708	0,01
24	29	0,00349	0,00926	0,001
25	14	0,00847	0,0377	0,01
26	10	0,00994	0,00421	0,01
27	12	0,00461	0,0106	0,001
28	16	0,00438	0,0483	0,001

4.2.2 Variação da função de treinamento

Pode-se fazer testes com outras funções de treinamento que a ferramenta oferece. Cada função possui diferentes critérios de parada. Utilizando a função *trainbr()* (Bayesianregulationbackpropagation), o treinamento teve duração de 36 s, enquanto usando a função padrão *trainlm()*, o treinamento durava no máximo 1 s. Comparando os valores de gradiente e desempenho (Figura 4.2) com a função *trainlm()*, o treinamento não se mostra satisfatório.

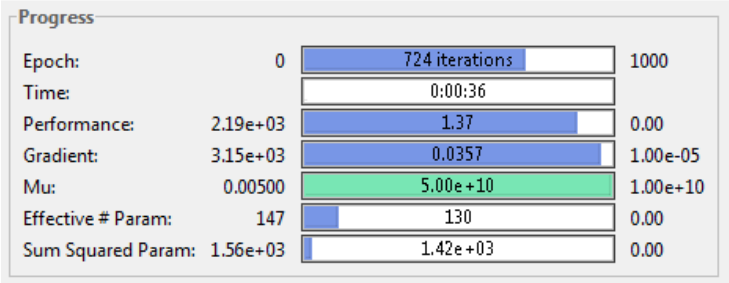


Figura 4.2 – Parâmetros do treinamento com *trainbr*.

A Figura 4.3 mostra o treinamento com a função *trainscg* (Scaledconjugategradientbackpropagation). Também obteve gradiente e performance abaixo do alcançado pela função padrão.

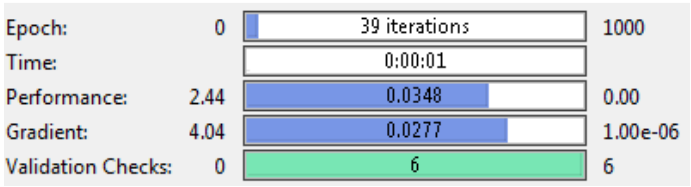


Figura 4.3 – Parâmetros do treinamento com *trainscg.*

Ao exemplo da função *trainscg*e *trainbr* outras funções foram testadas, mas não apresentaram um resultado tão satisfatório quanto a função *trainlm*. Por este motivo esta é a função de treinamento padrão para uma rede do tipo *Fitting*, ou seja, de adequação.

4.2.3 Demais variações possíveis

Do mesmo modo que para a função de treinamento, há vários parâmetros que já estão configurados por padrão como os melhores possíveis para a adequação de curvas. Uma opção, no entanto, é aumentar o número de validações, já que foi impreterivelmente o critério de parada para os treinamentos.

No entanto, a partir de um certo número de iterações, tanto o desempenho quanto o gradiente não reduzem mais, fazendo com que o número de validações sempre seja um critério de parada, exceto se o limite for tão grande que seja atingido o máximo de 1000 iterações. Isto pode ser comprovado no gráfico da Figura 4.4, cujo treinamento foi feito com o número limite de validações em 200.

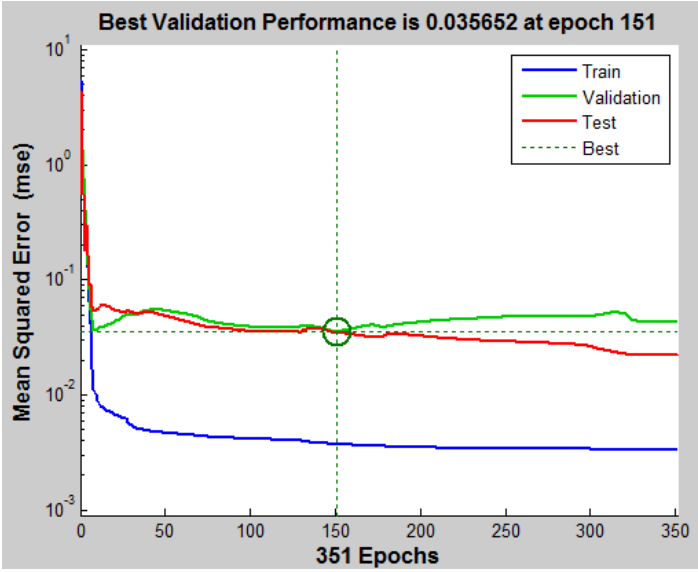


Figura 4.4 – Erro quadrático médio para um treinamento com 200 validações.

4.3 Alteração dos parâmetros em malha fechada

Realizando diversas simulações foi possível constatar que há três parâmetros que podem ser alterados de forma a melhorar os resultados da malha de controle: as entradas da rede neural, o cálculo das notas e as condições iniciais dos controladores. Foram testadas duas possibilidades para cada parâmetro, resultando em oito diferentes simulações.

Para as entradas da RNA, inicialmente foram utilizadas as variáveis do nível de referência e do nível inicial do tanque. Posteriormente, testou-se também utilizar a amplitude do salto (nível alvo – nível inicial), obtendo-se resultados positivos.

Para as saídas se pode utilizar as notas de todos os critérios ou somente dos critérios relacionados ao tempo. Já como condições iniciais dos controladores se pode utilizar o sinal atuante no instante passado ou as condições calculadas para o nível do tanque.

Cada uma das opções apresenta vantagens e desvantagens, sendo que a escolha entre uma delas dependerá das características que se considerarem importantes na resposta do sistema.

4.3.1 Primeira simulação

A primeira simulação é feita utilizando nível inicial e alvo como entradas da RNA, notas de todos os critérios de desempenho como saídas, e condições iniciais dos controladorescalculadas com o modelo do processo. O nível de saída com esta estrutura e com referência em 7,5 cm pode ser visto na Figura 4.5, e com referência em 20 cm na Figura 4.6, assim como as respostas do sistema sob a ação dos controladores correspondentes.

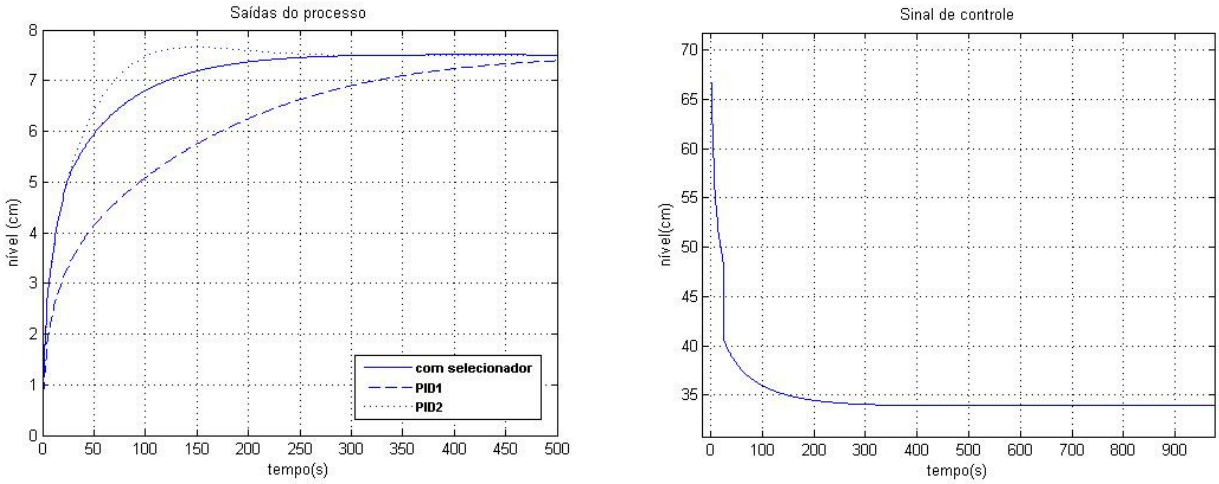


Figura 4.5– Saída dos sistemas e sinal de controle com referência em 7,5 cm.

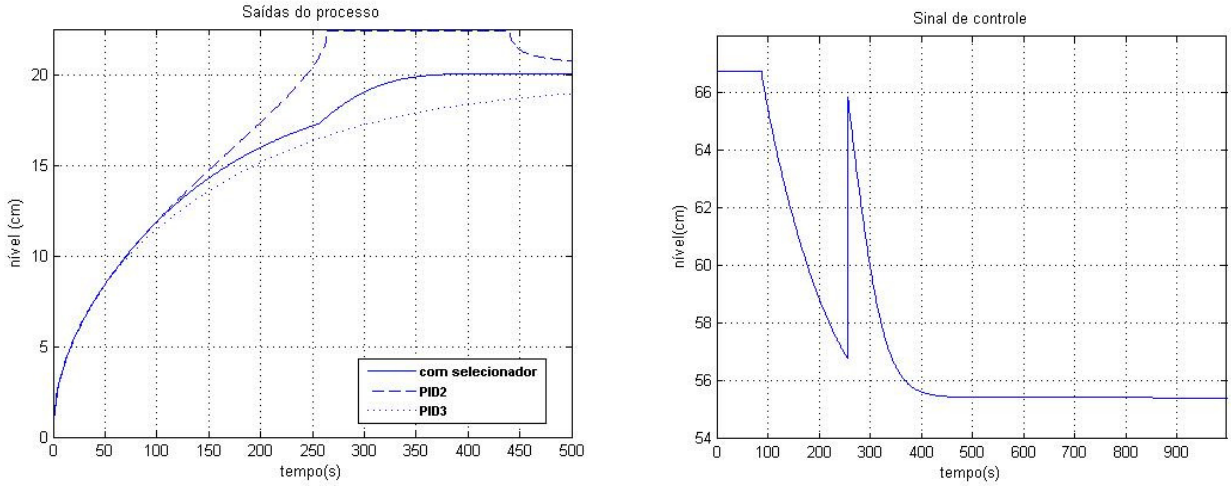


Figura 4.6– Saída dos sistemas e sinal de controle com referência em 20 cm.

Pode-se perceber que em ambos os pontos obteve-se melhora no desempenho em comparação ao desempenho dos controladores PID 1 e PID 2. Para a primeira referência houve uma troca entre os controladores 1 e 2, e para a segunda, uma troca entre o 2 e o 3. Os sinais de controle tiveram uma mudança abrupta por que com as condições iniciais aplicadas se garante a continuidade da saída, e não do sinal de controle. Os gráficos das saídas da RNA podem ser vistos no apêndice B.

4.3.2 Segunda simulação

Utilizar as condições iniciais calculadas mostrou bons resultados, mas como há possibilidade perturbação no processo, seria necessário calcular as condições iniciais para cada valor possível de perturbação, o que resulta em alto custo computacional. Assim, é usada nesta simulação outra opção de condições iniciais, que é o próprio sinal de controle atuante no processo, em um instante passado.

Dessa forma se garante a continuidade do sinal de controle, ainda que sob perturbação. No entanto, se perde em desempenho em comparação à primeira simulação, principalmente para um sinal de referência baixo. A saída para as referências de 7,5 cm e 20 cm podem ser vistas nas Figuras4.7 eFigura 4.8, respectivamente.Os sinais de saída das RNAs pode ser visto no apêndice B.

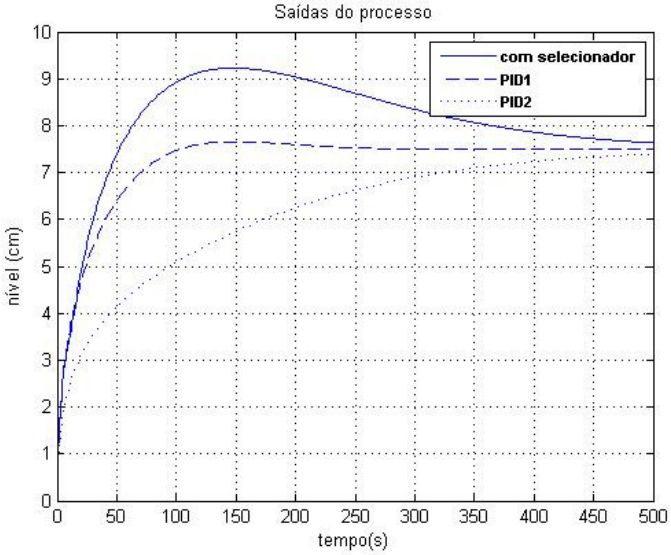


Figura 4.7 – Saída dos sistemas com referência em 7,5 cm.

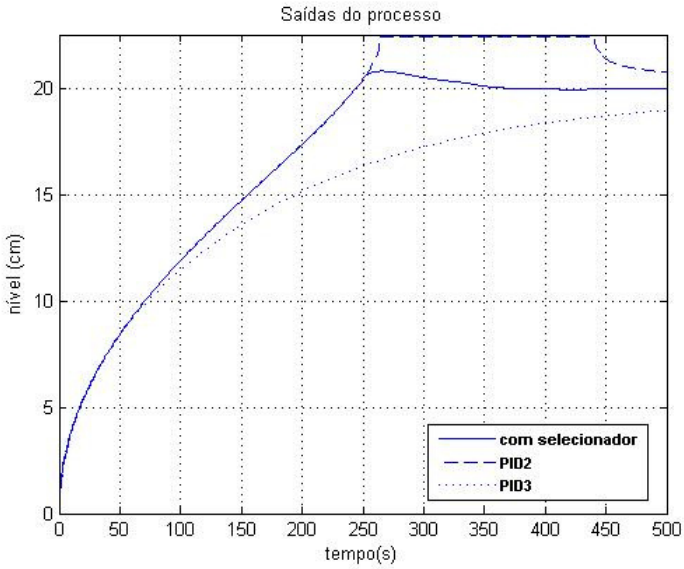


Figura 4.8 – Saída dos sistemas com referência em 20 cm.

4.3.3 Terceira e quarta simulações

Se pode perceber com as primeiras simulações que há somente uma troca entre controladores. Isto acontece porque uma das entradas da rede, o nível alvo, é sempre constante. Para melhorar a dinâmica da rede, foi testada uma rede treinada com a amplitude do salto em vez do nível alvo. Então, são feitas simulações com as condições iniciais calculadas e com as notas de todos os critérios, alterando a entrada da RNA.

Dessa forma, com a referência em 7,5, praticamente somente o segundo controlador atua, ou seja, não há ganho em desempenho. Já, com a referência em 20 cm, há cerca de 4 trocas, e o último controlador atuante é o número 1, ou seja, o mais rápido. Assim, o sistema acomoda-se em cerca de 260 s, sem sobressaltos, fornecendo o melhor resultado para esta referência comparado às primeiras simulações. A Figura 4.9 mostra a resposta dos sistemas com referência em 20 cm, e a Figura 4.10as saídas da RNA para esta referência.

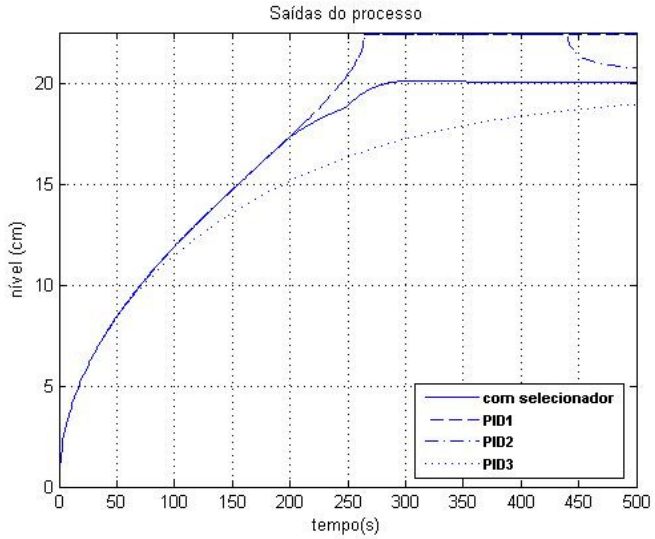


Figura 4.9 - Saída dos sistemas com referência em 20 cm.

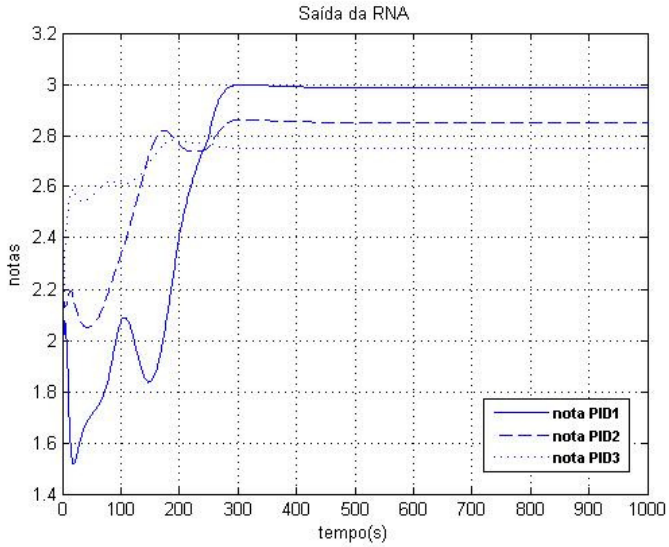


Figura 4.10 – Saída da RNA, mostrando a alternância entre controladores.

Na quarta simulação, as condições iniciais são obtidas do sinal de controle atuante. Da mesma maneira que para a segunda simulação, o desempenho foi semelhante mas ligeiramente inferior. Os gráficos correspondentes constam no Apêndice B.

4.3.4 Simulações de 5 a 8

Uma opção alternativa à utilização de todas as notas é utilizar somente as notas com relação ao desempenho temporal do sistema. Em sistemas de nível, certo sobrepasso pode ser tolerado em função de uma resposta mais rápida. Assim, foram feitos treinamentos da rede utilizando as notas do tempo de acomodação e do tempo de subida, para todas as combinações de parâmetros anteriores. As figuras relativas a estas simulações se encontram no Apêndice B.

4.3.5 Perturbação no sistema

A perturbação é crítica no sistema no caso em que o nível alvo é alto, por que pode fazer o liquido armazenado transbordar. Assim, foi feita uma simulação com perturbação na simulação de número 3, a configuração com menor tempo de acomodação para referência de 20 cm.

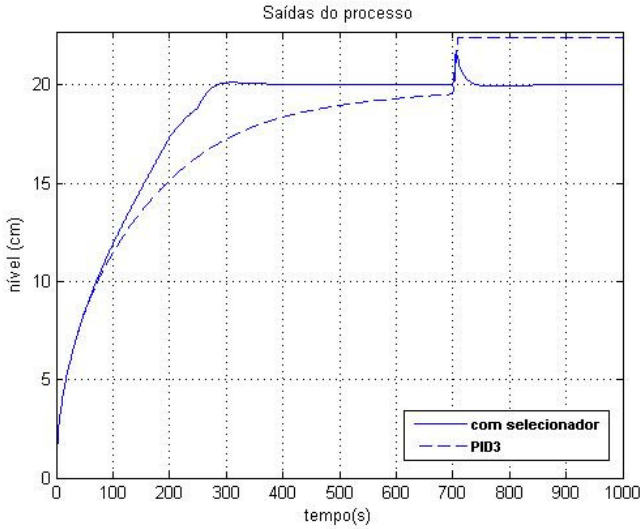


Figura 4.11 – Resposta à perturbação para a configuração da simulação 3, referência em 20 cm.

Por meio daFigura 4.11, verifica-se que, com o controlador número 3, o nível satura na presença de perturbação. Já, com a utilização do selecionador neural, a perturbação causa um pico em 21,1 cm, voltando ao nível de referência em menos de 50 s.

5 Conclusões e Trabalhos Futuros

5.1 Conclusões

As simulações apresentadas no Capítulo 4 mostram que o método proposto é válido. A maior parte dos resultados apresentou melhora no desempenho.

Ainda assim, cada opção de parâmetro mostrou vantagens e desvantagens. No caso das condições iniciais dos blocos PID, quando usados os valores obtidos no modelo do tanque, os resultados foram melhores, tanto em sobrepasso quanto em tempo de acomodação. Mas, a aquisição destes dados requer uma etapa a mais de simulação para cada valor possível de perturbação, já que os pontos de operação mudam em função da vazão de saída. Já, utilizar como condição inicial ação de controle atuante garante a continuidade do controle sem a necessidade de simulações extras, mas causa sobrepassos e reduz o tempo de acomodação.

Do mesmo modo, utilizar a amplitude do salto aplicado como entrada da RNA se mostrou eficiente no caso de ser necessária uma boa resposta em pontos distantes do inicial. Mas, em pontos de nível baixo, não há ganho de desempenho. Utilizando o nível de referência como entrada, o desempenho em pontos distantes do inicial não fornece resultados equivalentes, mas há melhora de desempenho em todos os pontos de operação.

Por fim, a utilização de notas diferentes também influencia a saída da malha fechada. Utilizando somente as notas temporais, é possível acelerar as respostas, mas dependendo da amplitude do sobrepasso, este pode levar ao transbordo do tanque ou à saturação do controlador, o que pode não seraceitável.

Desse modo, a configuração usada para a aplicação deste método depende de que características se considerem importantes, e que custo computacional é possível aceitar para melhorar o desempenho do sistema.

5.1 Trabalhos futuros

São possíveis diversas combinações, desde a parte da aquisição dos dados para o treinamento até a utilização destes dados para a simulação. Podem ser usados outros critérios de desempenho além dos usados neste trabalho, assim como se pode dar pesos diferentes a eles, conforme a necessidade. Podem-se utilizar todos para o treinamento, ou escolher entre os que forem melhores para o processo. Também se pode utilizar outros que não sejam controladores PID como candidatos, como controladores neurais, por exemplo, e mais de três controladores.

Assim, para trabalhos futuros, é possível explorar as diversas possibilidades citadas. Pode-se tentar aplicar este método em outros processos também, já que um tanque esférico é um processo relativamente simples, SISO, com somente uma perturbação.

6 Referências

- OGATA, K. **Engenharia de controle moderno**, Prentice-Hall do Brasil, Rio de Janeiro, 1997.

- HAJEK, Milan. **Neural Networks**, University of KwaZulu, Natal, 2005.

- KEMPF, Ariel de Oliveira. **Avaliação de Desempenho de Malhas de Controle.** 2003. Tese (Mestrado em Engenhadria) – Departamento de Engenharia Química, Universidade Federal do Rio Grande do Sul, Rio Grande do Sul.

- BAZANELLA, Alexandre Sanfelice; DA SILVA JR., João Manoel.2005)**Sistemas de Controle, princípios e métodos de projeto**. 1ª ed. Porto Alegre: Editora da UFRGS, 205. 299p.

-BAPTISTA, Darío; MORGADO-DIAS, Fernando. **A survey of artificial neural network training tools**. 2013. Neural Computing and Applications, Londres, v. 23, n.1, p. 609-615, fev./jun. 2013.

-BEHROOZSARAND, Alireza;SHAFIEI, Sirous. **Optimal control of distillation column using Non-Dominated Sorting Genetic Algorithm-II**. 2011. Journal of Loss Prevention in the Process Industries, v. 24, p. 25-33, mar./ago. 2010.

-CHATZIDOUKAS, Christos et al. **Optimal grade transition and selection of closed-loop controllers in a gas-phase olefin polymerization fluidized bed reactor**. 2003. Chemical Engineering Science, v. 58, p. 3643-3658, jan./jul. 2003.

-FU, Yue; CHAI, Tianyou. **Nonlinear multivariable adaptive control using multiple models and neural networks**.2007. Automatica, v. 43, p. 1101-1110, mai. 2005/ dez. 2006.

-AYALA, Helon Vicente Hultmann; COELHO, Leandro dos Santos. **Tuning of PID controller based on a multiobjective genetic algorithm applied to a robotic manipulator**. 2012. Expert Systems with Applications, v. 39, p. 8968-8974, fev./jul. 2012.

-HUNT, K. J. et al.. **Neural networks for control systems - A survey**. 1992. Automatica, v. 28,p. 1083-1112, jul./set. 1992.

-LEE, Jong Min; LEE, Jay H..**Value function-based approach to the scheduling of multiple controllers**.2008. Journal of Process Control, v. 18, p. 533-542, mai./out. 2007.

-LIN, Jeen; LIAN, Ruey-Jing. **Hybrid fuzzy-logic and neural-network controller for MIMO systems**. 2009. Mechatronics,v. 19, p 972-986, jan./jul. 2009.

-YIN, Xiaohong et al..**Control structure selection for vapor compression refrigeration cycle**. 2013. In Control structure selection for vapor compression refrigeration cycle, Communications in Computer and Information Science, v. 355, p 477-485, fev.set. 2013.

Apêndices

Apêndice A: Modelo linearizado

A.1-Linearização do modelo

A linearização do modelo do processo é uma etapa importante para configurar os controladores candidatos. Cada um será projetado para um ponto de operação do tanque esférico.

Uma prática básica de linearização é aproximar a função em uma série de Taylor truncada no segundo termo. O primeiro passo para a linearização é determinar que função será linearizada. Para isto, é necessário unir as equações que modelam a dinâmica do tanque. A equação completa resulta no seguinte é descrita na Equação (A.1).

$$\pi \frac{dh}{dt}(Dh - h^2) + k\sqrt{h} = f_{in} \qquad (A.1)$$

Nesta equação há dois termos não lineares. O primeiro termo é a multiplicação da derivada de uma variável por um polinômio da mesma, e o segundo contém uma raíz quadrada da variável. Para linearizar o primeiro termo são necessárias algumas atribuições:

$$F(h) = \frac{dh}{dt}(Dh - h^2) = g(h) * (Dh - h^2) \qquad (A.2)$$

$$g(h) = \frac{dh}{dt} \qquad (A.3)$$

Conforme as Equações(A.2) e (A.3), o termo que será linearizado é uma função de duas variáveis, g e h. Para tanto, é necessário utilizar a série de Taylor de duas variáveis para a linearização. Para esta aplicação, a série de Taylor é truncada no termo que apresenta a primeira derivada, já que, para pequenas variações, a magnitude dos próximos termos seria insignificante.

$$F(g,h) = F(g_\gamma, h_\gamma) + F_g(g_\gamma, h_\gamma)(g - g_\gamma) + F_h(g_\gamma, h_\gamma)(h - h_\gamma) \qquad (A.4)$$

$$F_g(g_\gamma, h_\gamma) = \left.\frac{\partial F}{\partial g}\right|_{h=h_\gamma, g=g_\gamma} \quad F_f(g_\gamma, h_\gamma) = \left.\frac{\partial F}{\partial h}\right|_{h=h_\gamma, g=g_\gamma} \qquad (A.5)$$

Ondeg_γ e h_γ são os valores das funções g(h) e h no ponto de equilíbrio h = γ. Para o ponto de equilíbrio, todas as derivadas são nulas, assim

$$g_\gamma = g(\gamma) = \left.\frac{dh}{dt}\right|_{h=\gamma} = 0 \;;\; h_\gamma = \gamma \qquad (A.6)$$

Assim, substituindo os termos da Equação(A.6), a função F(g,h) resulta no seguinte:

$$F(g,h) \cong \frac{dh}{dt}(D\gamma - \gamma^2) \qquad (A.7)$$

O segundo termo é linearizado pelo mesmo processo, mas utilizando a série de Taylor para uma função de uma só variável.

$$Z(h) = Z(h_\gamma) + \left.\frac{dZ}{dh}\right|_{h=\gamma} (h - h_\gamma) \quad (A.8)$$

$$\left.\frac{dZ}{dh}\right|_{h=\gamma} = \left.\frac{d}{dh} k\sqrt{h}\right|_{h=\gamma} = \frac{1}{2\sqrt{\gamma}} \quad (A.9)$$

$$Z(h) = k\sqrt{\gamma} + k\frac{1}{2\sqrt{\gamma}}(h - \gamma) \quad (A.10)$$

Unindo os dois termos linearizados, obtem-se a equação linearizada em torno do ponto γ do tanque esférico:

$$f_{in} = \frac{dh}{dt}\pi(D\gamma - \gamma^2) + k\sqrt{\gamma} + k\frac{1}{2\sqrt{\gamma}}(h - \gamma) \quad (A.11)$$

A.2- Função de transferência

Para que seja possível projetar os controladores que utilizados no método é necessário obter uma função de transferência da equação linearizada. Para tanto será necessário uma mudança de variável. Em vez de utilizar as variáveis serão utilizadas suas variações. Em primeiro lugar, a mudança da variável de entrada:

$$\Delta f_{in} = f_{in}(t) - f_{in}(\gamma) \;\therefore\; f_{in}(t) = \Delta f_{in} + f_{in}(\gamma) \quad (A.12)$$

Substituindo h = γ na equação linearizada, obtém-se $F_{in}(\gamma) = k\sqrt{\gamma}$. Em seguida, a variável de saída $\Delta h = h - \gamma$, fazendo ambas as mudanças de variável na equação linearizada, e considerando que

$$\frac{d\Delta h}{dt} = \frac{d}{dt}(h - \gamma) = \frac{dh}{dt} - \frac{d\gamma}{dt} = \frac{dh}{dt} \quad (A.13)$$

Obtém-se a equação a seguir.

$$\Delta f_{in} + f_{in}(\gamma) = \frac{d\Delta h}{dt}\pi(D\gamma - \gamma^2) + k\sqrt{\gamma} + k\frac{1}{2\sqrt{\gamma}}\Delta h \quad (A.14)$$

Como $f_{in}(\gamma) = k\sqrt{\gamma}$, a relação entre a variação da entrada e a variação da saída é a seguinte:

$$\Delta f_{in} = \frac{d\Delta h}{dt}\pi(D\gamma - \gamma^2) + k\frac{1}{2\sqrt{\gamma}}\Delta h \quad (A.15)$$

Assim, depois de uma transformada de Laplace, a função de transferência do tanque é a seguinte:

$$\frac{\Delta H(s)}{\Delta F_{in}(s)} = \frac{{}^{1}/_{\pi\gamma(D-\gamma)}}{\left(s + {}^{k}/_{2\gamma\sqrt{\gamma}\pi(D-\gamma)}\right)} \qquad (A.16)$$

Como pode-se perceber, o valor em regime e o polo desta função de transferência dependem do ponto em que foi feita a linearização. Já o valor relativo à válvula (k), influencia somente na posição do polo.

Apêndice B: gráficosdas simulações

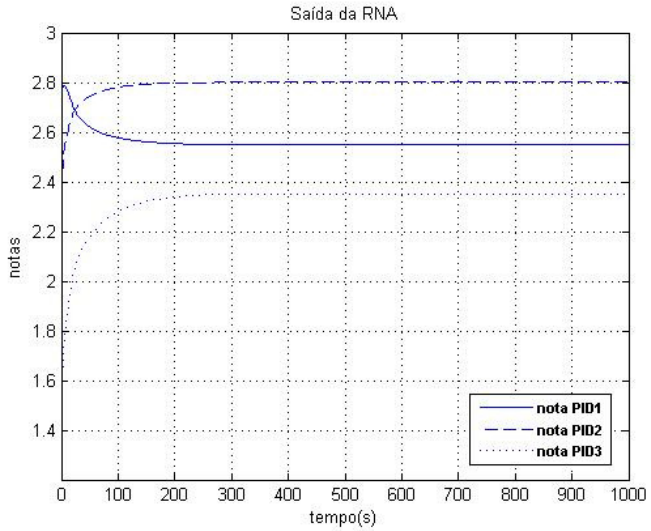


Figura B.1– Saída da RNA para a simulação 1 e referência em 7,5 cm.

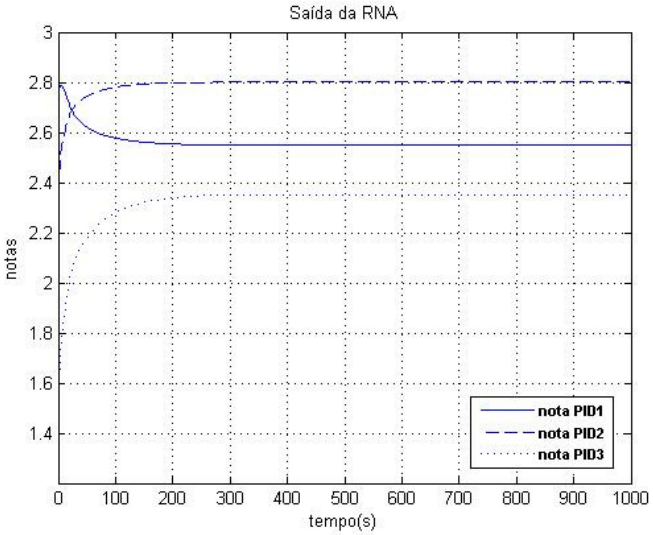


Figura B.2– Saída da RNA para a simulação 1 e referência em 20 cm.

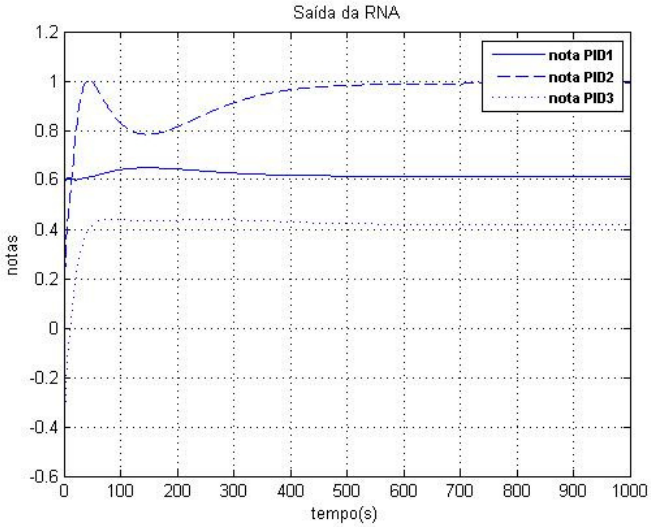


Figura B.3– Saída da RNA para a simulação 2 e referência em 7,5 cm.

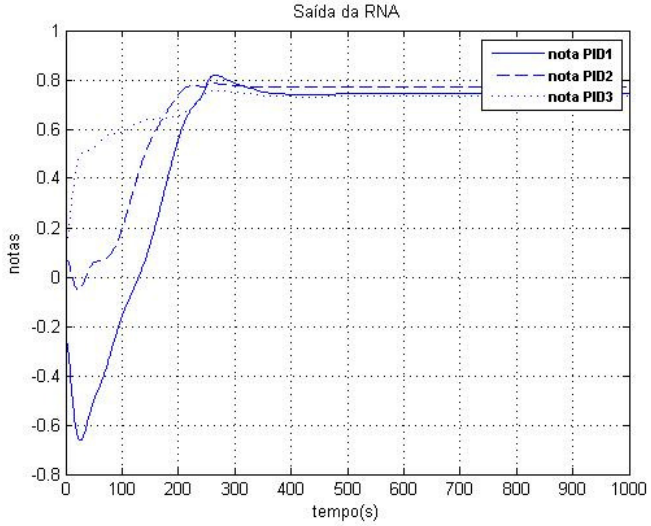


Figura B.4– Saída da RNA para a simulação 2 e referência em 20 cm.

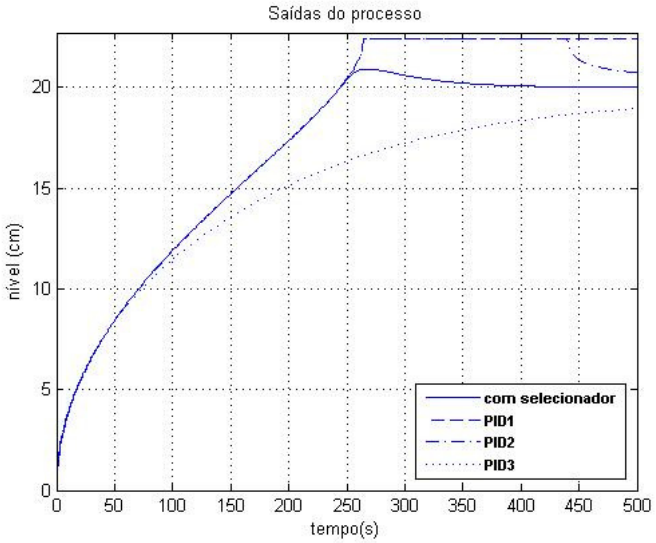


Figura B.5 - Saída do processo para a simulação 4 e referência em 20 cm.

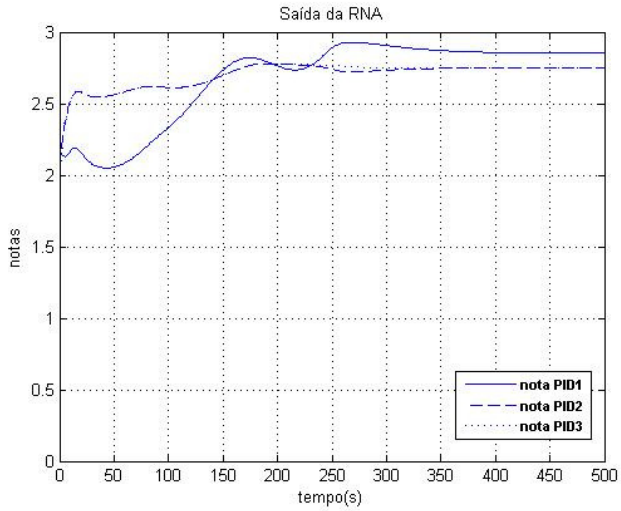


Figura B.6 -Saída da RNA para a simulação 4 e referência em 20 cm.

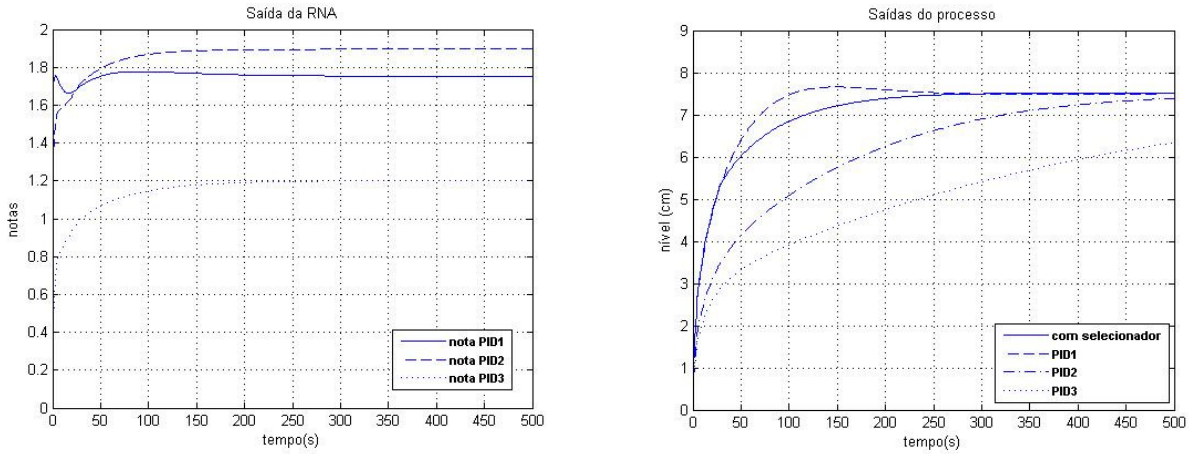


Figura B.7 – saídas da RNA e do processo para a simulação 5, referência em 7,5 cm.

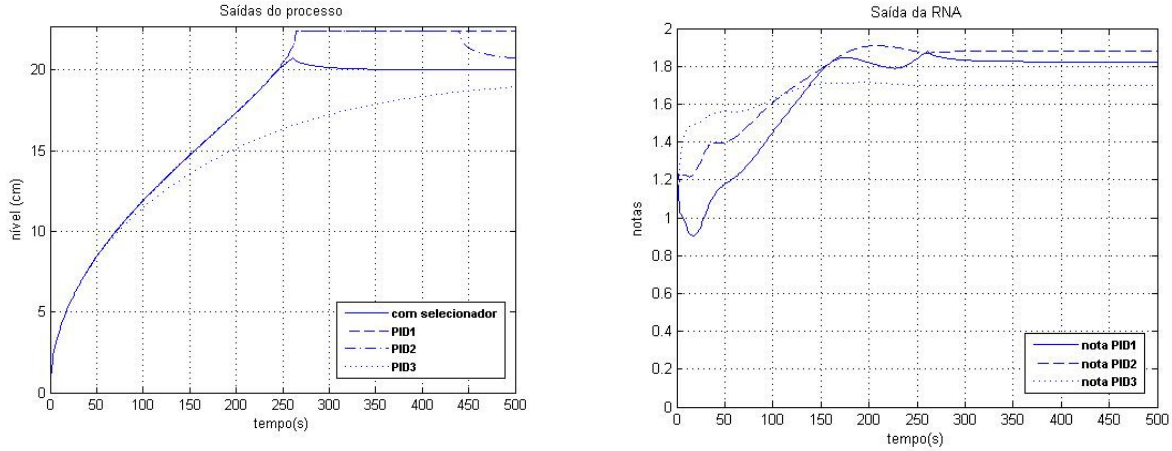


Figura B.8 -saídas da RNA e do processo para a simulação 5, referência em 20 cm.

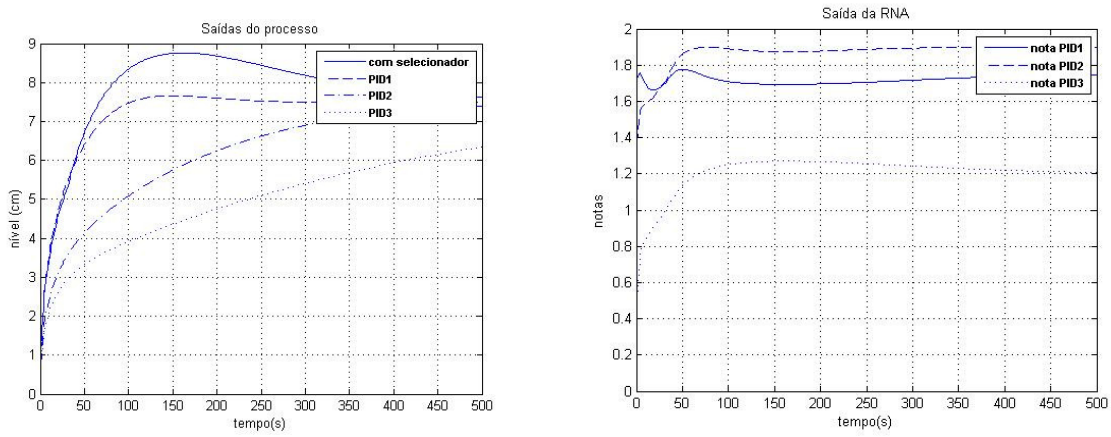


Figura B.9 - saídas da RNA e do processo para a simulação 6, referência em 7.5 cm.

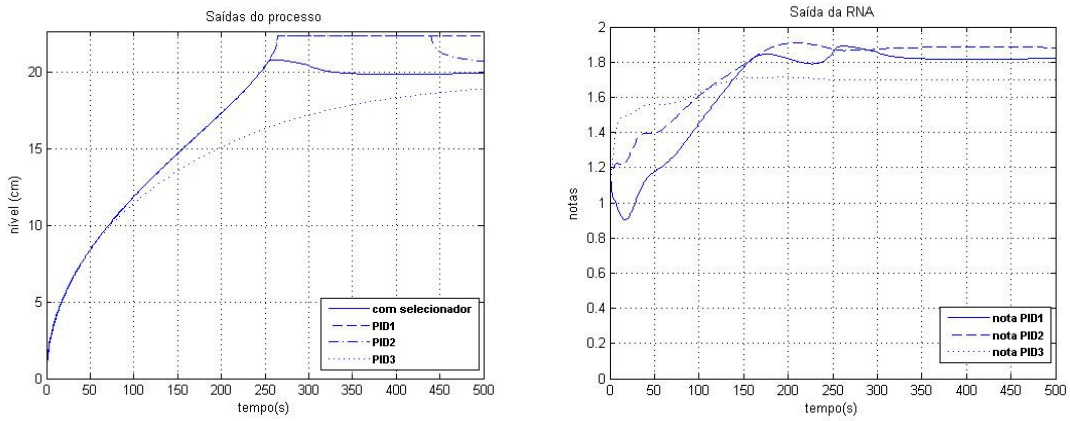


Figura B.10 - saídas da RNA e do processo para a simulação 6, referência em 20 cm.

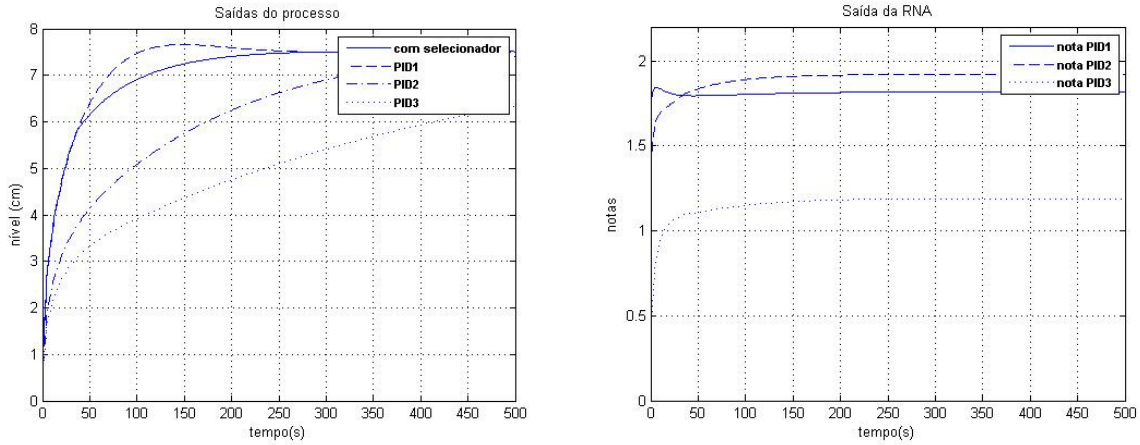


Figura B.11 - saídas da RNA e do processo para a simulação 7, referência em 7.5 cm.

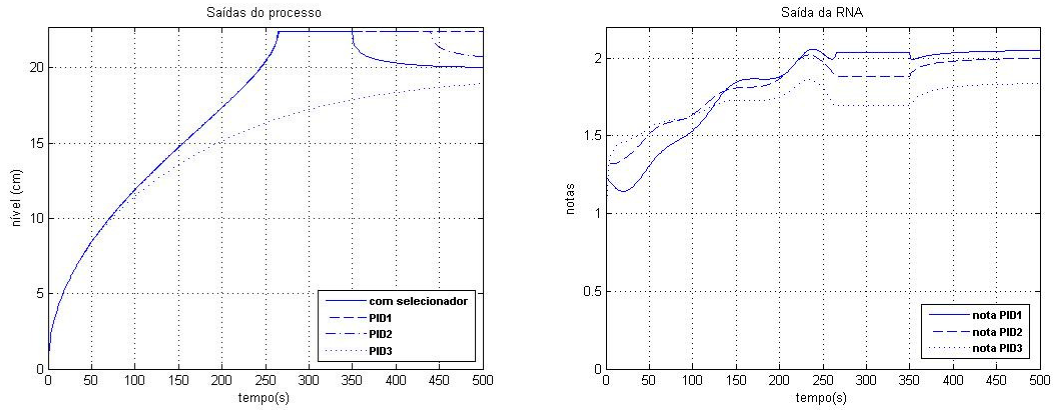


Figura B.12 - saídas da RNA e do processo para a simulação 7, referência em 20 cm.

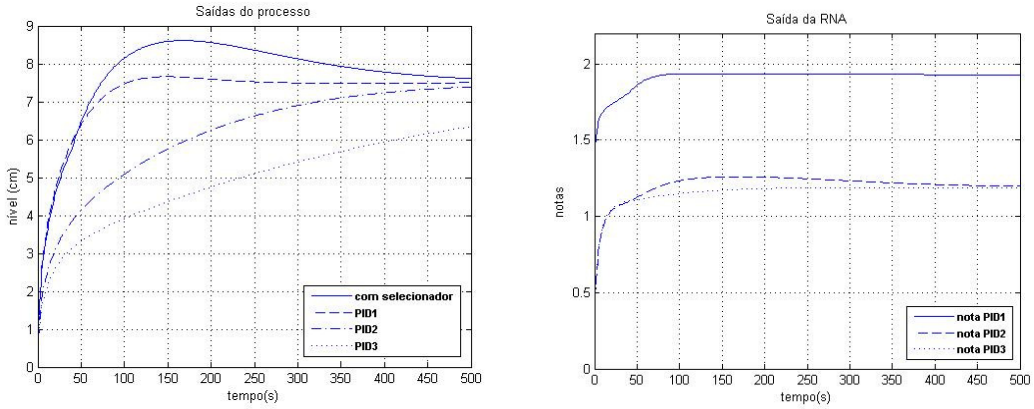


Figura B.13 - saídas da RNA e do processo para a simulação 8, referência em 7,5 cm.

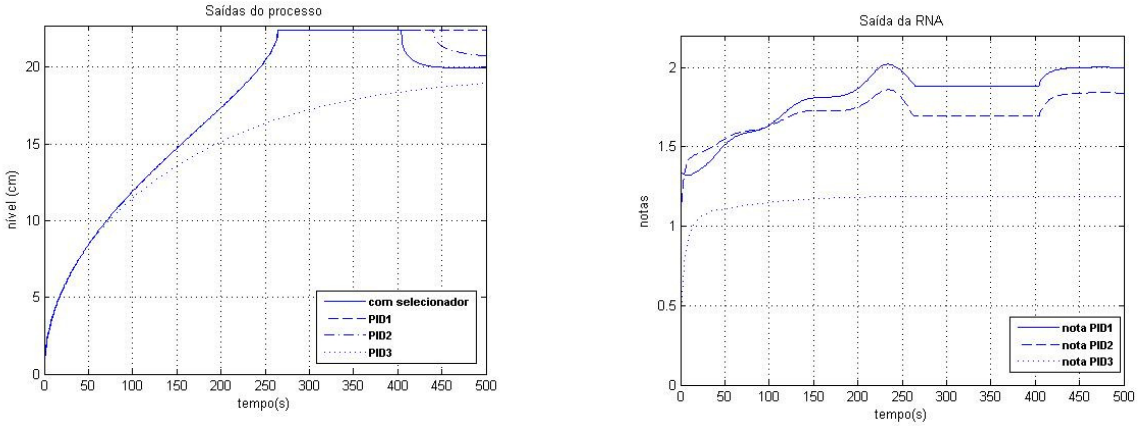


Figura B.14 - saídas da RNA e do processo para a simulação 8, referência em 20 cm.

Apêndice C: Códigos em Matlab®

C.1 - *Código para aquisição de dados para o treinamento da rede*

```
%%Script que irá adquirir todos os dados para treinamento da rede neural.
clc;
clearall;

alvo = [0.1,2,4,6,8,10,12,14,16,18,20,22];
inicial = [0.1,2,4,6,8,10,12,14,16,18,20,22];
```

```
controle = [10.3 5.28 4.17
         0.206 0.043 0.018
0 0 0];
%%os valores das condiçoes iniciais dos PIDs (calculados)
CIPID =[ 3.9400 17.5200 24.7800 30.3500 35.0500 39.1900 42.9500 46.3700
49.6000 52.5800 55.4200 58.1200];

%inicialização as matrizes
ides = zeros(length(alvo)*length(inicial) -12,length(controle));
tempoSubida = ides;
pico = ides;
overshoot = ides;
tempoAcomodacao = ides;
amplitude = ides;
entradasRNA = zeros(length(alvo)*length(inicial)-12,2);
entradasRNA2 = entradasRNA;
count=0;

%%
for x=1:length(alvo)
nivelAlvo = alvo(x);
for y = 1:length(inicial)
xs = inicial(y);
ifxs~=nivelAlvo

count=count+1
entradasRNA2(count,1) = nivelAlvo - xs;
entradasRNA2(count,2) = xs;
entradasRNA(count,1) = nivelAlvo;
entradasRNA(count,2) = xs;
for z = 1:length(controle)
ci = CIPID(y);
kp = controle(1,z);
ki = controle(2,z);
%kd = controle(3,z);
sim('tanque_malha_fechada');
sim('tanque_ruido_perturb');
ides(count,z) = fcor(OutputVar.signals.values-nivelAlvo, 20, 2);

%medidas de desempenho do sistema
%overshoot em porcentagem
ifnivelAlvo>xs
            [~,ind]=min(abs(Output.signals.values-0.1*abs(nivelAlvo-xs)-
xs));
            tempo10pc=Output.time(ind);
            [~,ind]=min(abs(Output.signals.values-0.9*abs(nivelAlvo-xs)-
xs));
            tempo90pc= Output.time(ind);
pico(count,z) = max(Output.signals.values);
ifpico(count,z)-nivelAlvo>0
overshoot(count, z) = (pico(count,z)-nivelAlvo)*100/(nivelAlvo-xs);
else
overshoot(count, z) = 0;
end
else
            [~,ind]=min(abs(Output.signals.values-0.1*abs(nivelAlvo-xs)-
nivelAlvo));
            tempo10pc=Output.time(ind);
            [~,ind]=min(abs(Output.signals.values-0.9*abs(nivelAlvo-xs)-
nivelAlvo));
            tempo90pc= Output.time(ind);
pico(count,z) = min(Output.signals.values);
ifpico(count,z)-nivelAlvo<0
overshoot(count, z) = (nivelAlvo-pico(count,z))*100/abs(nivelAlvo-xs);
```

```
else
overshoot(count, z) = 0;
end

end
erro = abs(nivelAlvo-Output.signals.values);
%calcular o tempo de acomodação, considerando 98% do regime

      tempo=1;
achou=0;

forind = 1:length(Output.time)
iferro(ind)>0.02*nivelAlvo+0.01
tempo = ind;
end
end
amplitude(count,z) = abs(nivelAlvo-xs);
tempoSubida(count,z) = abs(tempo90pc-tempo10pc);
tempoAcomodacao(count,z) = Output.time(tempo);

end
end
end
end

% cálculo das notas dos controladores

notaTempo=tempoAcomodacao/max(max(tempoAcomodacao))+tempoSubida/max(max(t
empoSubida));
notaDes= overshoot/max(max(overshoot))+ides/max(max(ides));
notas = 2-notaDes-notaTempo;
```

C.2 - Código para treinamento da rede neural

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by NFTOOL
% Created Tue Aug 12 15:46:48 BRT 2014
%
% This script assumes these variables are defined:
%
%   inputs - input data.
%   outputs - target data.

inputs = entradasRNA2';
targets = notas4';

% Create a Fitting Network
hiddenLayerSize = 24;
net = fitnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
```

```
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% For help on training function 'trainlm' type: help trainlm
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainlm';  %Levenberg-Marquardt

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
'plotregression', 'plotfit'};


% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs);

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs);
valPerformance = perform(net,valTargets,outputs);
testPerformance = perform(net,testTargets,outputs);

% View the Network
view(net)
gensim(net);
% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotfit(net,inputs,targets)
%figure, plotregression(targets,outputs)
%figure, ploterrhist(errors)
```

C.3 - Código para cálculo das condições iniciais

```
% funcao para calcular que vazão que dará a condição inicial dos PIDs
clc;
clearall;
%descarga do tanque em 0,6

nivelInicial = [0.1,2,4,6,8,10,12,14,16,18,20,22];
armazena = zeros(length(nivelInicial), 2);
CIPID = zeros(length(nivelInicial),1);
cont=0;
mult=0.7;

for x= 1:667
vazao=x/10;
sim('Simulacao_Tanque');
nivelOut = vazaoSaida.signals.values(1399);
```

```
armazena(x,1) = nivelOut;
armazena(x,2)= vazao;
cont=cont+1
end
```

C.4- Código para a função de interpretação das saídas da RNA

```
functioninterpret(block)
% Interpreta os dados da RNA e retorna os sinais de saída desejados
setup(block);
%endfunction

function setup(block)
%% Register number of input port and output port
block.NumInputPorts  = 3;
block.NumOutputPorts = 2;

block.NumDialogPrms  = 0;

  %% Setup functional port properties
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

block.InputPort(1).DatatypeID    = 0;
block.InputPort(1).Complexity    = 'Real';
block.InputPort(1).Dimensions    = -1;
block.InputPort(1).SamplingMode  = 'Sample';
block.InputPort(1).DirectFeedthrough = 1;

block.InputPort(2).DatatypeID    = 0;
block.InputPort(2).Complexity    = 'Real';
block.InputPort(2).Dimensions    = 1;
block.InputPort(2).SamplingMode  = 'Sample';
block.InputPort(2).DirectFeedthrough = 1;

block.InputPort(3).DatatypeID    = 0;
block.InputPort(3).Complexity    = 'Real';
block.InputPort(3).Dimensions    = -1;
block.InputPort(3).SamplingMode  = 'Sample';
block.InputPort(3).DirectFeedthrough = 1;

block.InputPort(2).DatatypeID    = 0;
block.InputPort(2).Complexity    = 'Real';
block.InputPort(2).Dimensions    = 1;
block.InputPort(2).SamplingMode  = 'Sample';
block.InputPort(2).DirectFeedthrough = 1;

block.OutputPort(1).DatatypeID   = 0;
block.OutputPort(1).Complexity   = 'Real';
block.OutputPort(1).Dimensions   = -1;
block.OutputPort(1).SamplingMode = 'Sample';

block.OutputPort(2).DatatypeID   = 0;
block.OutputPort(2).Complexity   = 'Real';
block.OutputPort(2).Dimensions   = 1;
block.OutputPort(2).SamplingMode = 'Sample';

  %% Set the block simStateCompliance to default (i.e., same as a built-
in block)
block.SimStateCompliance = 'DefaultSimState';
```

```
%%  Register sample times
block.SampleTimes = [0 0];

  %% Block runs on TLC in accelerator mode.
block.SetAccelRunOnTLC(true);

  %% Reg methods

block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);
block.RegBlockMethod('Outputs',                 @Outputs);
block.RegBlockMethod('Start', @Start);
block.RegBlockMethod('Terminate',               @Terminate);
block.RegBlockMethod('Update', @Update);%

%endfunction

functionDoPostPropSetup(block)
block.NumDworks = 1;

block.Dwork(1).Name            = 'x1';
block.Dwork(1).Dimensions      = 3;
block.Dwork(1).DatatypeID      = 0;      % double
block.Dwork(1).Complexity      = 'Real'; % real
block.Dwork(1).UsedAsDiscState = true;

function Start(block)

block.Dwork(1).Data = [1 0 0];

function Outputs(block)

x1 = block.Dwork(1).Data;
u1 = block.InputPort(1).Data;
u2 = block.InputPort(2).Data;
u3 = block.InputPort(3).Data;
u4 = block.InputPort(4).Data;

if u4 = 0.7
 [~, ind] = min(abs(ci(:,1)-u2));
 y2 = ci(ind,2);
else
 [~, ind] = min(abs(ci2(:,1)-u2));
y2 = ci2(ind,2);

%
x2 = x1;
[max1, index_max] = max(x2);
x2(index_max) = [];
[max2, ~] = max(x2);

ifabs(max1-max2)>0.0001 %limiar de mudança de controlador
y = [0 0 0];
y(index_max)=1;
else
    y = block.InputPort(3).Data;
end

block.OutputPort(1).Data = y;
block.OutputPort(2).Data = y2;
%endfunction
```

```
function Update(block)
block.Dwork(1).Data = block.InputPort(1).Data;
%%
functionTerminate(block)
%endfunction
```

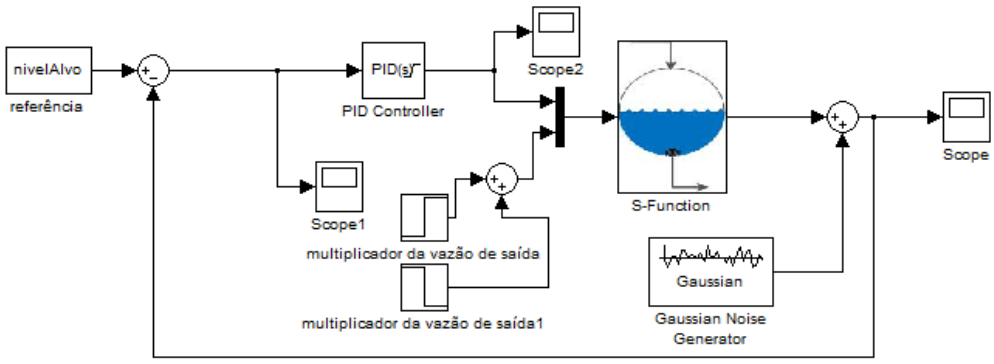
C.5 - Código do modelo do tanque esférico

```
function [sys,x0,str,ts] = TQfunc(t,x,u,flag,xs)
%  TQfunc - s-function do Sistema de 1TQ - Modelo Fenomenológico
%
%Inputs
%u(1)    - Vazão de Alimentação   F     - [cm³/s]
%
%States
%x(1)    - Nível do Tanque      H        - [cm]
%
%Outputs
%y(1)    - Nível do Tanque      H        - [cm]

% Criado por P.R.B.F @GIMSCOP/DEQUI 26-06-2012
% Modificado por Aline Käfer 28-10-2014
switchflag,
case 0,
   [sys,x0,str,ts] = mdlInitializeSizes(xs);
case 1,
sys = mdlDerivatives(t,x,u);
case 3,
sys = mdlOutputs(t,x,u);
case { 2, 4, 9 },
sys = [];
otherwise
error(['Unhandled flag = ',num2str(flag)]);
end
%=============================================================================
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-
function.
%=============================================================================
function [sys,x0,str,ts] = mdlInitializeSizes(xs)

sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

% Intitial Conditions
sys = simsizes(sizes);
x0  = xs;
str = [];
ts  = [0 0];
%=============================================================================
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
function sys = mdlDerivatives(t,x,u)
% Estados dos Sistema =>  H; [cm]
% Parâmetros do Modelo
% Coeficientes de Descarga (19/01/11)
% fact é um fator de ajuste (empírico)
fact = u(2);
```

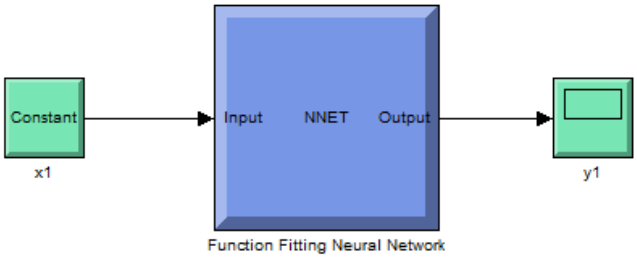
```
CD(1) = fact*17.6980;                  % Coeficiente de Descarga =>
CD[(cm^(5/2))/s]

% Variáveis de Entrada - Vazões
F = zeros(1,1);
F(1) = u(1);                              % Vazão de Entrada F => [cm³/s]
D(1) = 22.5;                              %diâmetro do tanque D => [cm]

% Equações Diferenciais
if x(1) > D(1)-1e-1
   x(1)= D(1)-1e-1;           % Limite de Nível Máximo
else
if x(1)  < 1e-1
   x(1)=  1e-1;           % Limite de Nível Mínimo
end
end
%if x(1) > 1e-3 && x(1) <= D(1)-1e-3
sys =  (F(1) - CD(1)*sqrt(x(1)))/(pi*x(1)*(D(1) - x(1)));
%=============================================================================
% mdlOutputs
% Return the block outputs.
%=============================================================================
function sys = mdlOutputs(t,x,u)
if x(1) > 22.5- 1e-1
sys  = 22.5- 1e-1;           % Limite de Nível Máximo
else
if x(1)  <  1e-1
sys  =  1e-1;           % Limite de Nível Mínimo
else
sys =  x(1);
end
end
```

Apêndice D: Esquemas de simulação em Simulink®

D.1 - Esquema de simulação com perturbação e ruído branco



D.2 - Resultado da função gensim()



Anexos

Anexo A: Código da função fcor()

```
function [perfind, sigmv, F] = fcor(yt, m, d)
% [IDES, MV, F] = FCOR(YT, M, D)
% Implementaçao do algoritmo FCOR (FilteringandCorrelationAnalysis)
% para sistemas SISO.
%
% Estimação do ruído através do "pre-whitening" do sinal de saída yt
% usando AR;
% O ruído é o resíduo da modelagem da série temporal:r = YT - Yest
%
% M: é a ordem do modelo a ser estimado para Yest.
% D: é o tempo morto do sistema.
% IDES: é o índice de performance da malha segundo Desborough e Harris
(1992).
% MV: é a mínima variança (variança do controlador de variança mínima).
% F: é o polinômio de ordem N resultante da modelagem de YT.

% Ariel Kempf 2002
% Modificado por Pedro Fernandes 01-10-2014 - Adaptação para Matlab 2009

dim = size(yt);
if dim(1) == 1;
yt = yt';
end

thyest = ar(yt, m, 'ls');
%thyest = ar(yt, m, 'yw');
%thyest = ar(yt, m);
%thyest = arx(yt, m);
%thyest = armax(yt, [m m]);

yest = predict(yt, thyest, 1);
yest = yest{1};
r = yt - yest;

%den = sqrt(mean((yt-mean(yt)).^2) * mean((r-mean(r)).^2));
%den = std(yt,1)*std(r,1);
den = sqrt(mean((yt).^2) * mean((r).^2));

yv = yt(m+d : end);

for k=1:d
ind = m + d - k + 1;

%acov = cov(yv, r(ind:end-k+1), 1);
%rho(k) = acov(2) / den;
%rr = r(ind:end-k+1);
%yvv = yv(end-length(rr)+1 : end);

%rho(k) = mean((yv - mean(yv)).*(r(ind:end-k+1) - mean(r(ind:end-k+1))))
/ den;
rho(k) = mean((yv).*(r(ind:end-k+1))) / den;

%cor = corrcoef(yv, r(ind:end-k+1));
%rho(k) = cor(2);
end
%F = alfa;
%F = th2poly(thyest);
%F = F(2:end)/F(2);
```

```
%[la, lo, F] = th2poly(thyest);

sigmv = sum(rho.^2) * var(yt, 1);
F = fliplr(rho) * var(yt, 1);
F = F/F(1);
perfind = 1 - sum(rho.^2);
%perfind = sum(rho.^2);
```

Anexo B: Código da função de simulação do tanque esférico

```
function [sys,x0,str,ts] = TQfunc(t,x,u,flag,xs)

%  TQfunc - s-function do Sistema de 1TQ - Modelo Fenomenológico
%
%Inputs
%u(1)    - Vazão de Alimentação   F     - [L/min]
%
%States
%x(1)    - Nível do Tanque      H        - [cm]
%
%Outputs
%y(1)    - Nível do Tanque      H        - [cm]

%% Rotina %%

% Criado por:
% P.R.B.F @GIMSCOP/DEQUI
% 26-06-2012
% Modificado por: Aline Kafer, 12-10-2014


switchflag,

%%%%%%%%%%%%%%%%%%
% Initialization %
%%%%%%%%%%%%%%%%%%

case 0,
   [sys,x0,str,ts] = mdlInitializeSizes(xs);

%%%%%%%%%%%%%%%%
% Derivatives %
%%%%%%%%%%%%%%%%

case 1,
sys = mdlDerivatives(t,x,u);

%%%%%%%%%%%
% Outputs %
%%%%%%%%%%%

case 3,
sys = mdlOutputs(t,x,u);

%%%%%%%%%%%%%%%%%%
% Unhandled flags %
%%%%%%%%%%%%%%%%%%

case { 2, 4, 9 },
sys = [];
```

```
%%%%%%%%%%%%%%%%%%%%
% Unexpected flags %
%%%%%%%%%%%%%%%%%%%%

otherwise
error(['Unhandled flag = ',num2str(flag)]);

end

%%

%
%=============================================================================
=====
% mdlInitializeSizes
% Return the sizes, initial conditions, and sample times for the S-
function.
%=============================================================================
=====
%

function [sys,x0,str,ts] = mdlInitializeSizes(xs)

sizes = simsizes;
sizes.NumContStates  = 1;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 0;
sizes.NumSampleTimes = 1;

% Intitial Conditions

sys = simsizes(sizes);
x0  = xs;
str = [];
ts  = [0 0];

% end mdlInitializeSizes

%
%=============================================================================
=====
% mdlDerivatives
% Return the derivatives for the continuous states.
%=============================================================================
=====
%

function sys = mdlDerivatives(t,x,u)

% Estados dos Sistema =>  H; [cm]

% Parâmetros do Modelo
% Coeficientes de Descarga (19/01/11)
% fact é um fator de ajuste (empírico)
fact = u(2);
CD(1) = fact*17.6980;                % Coeficiente de Descarga =>
CD[(cm^(5/2))/s]

% Diâmetro dos Tanques Esféricos
D = 22.5*ones(1,1);             % Diâmetro do Tanque  => [cm]
```

```
% Estados
h = x;

% Variáveis de Entrada
% Vazões
F = zeros(1,1);
F(1) = u(1);                                % Vazão de Entrada F => [L/min]

B(1) = F(1)*(1000/60);          % Bomba do Tanque => [(cm^3)/s]

% Modelo
% Equações Diferenciais

if h(1) > 1e-3 | h(1) < D(1) - 1e-3
sys(1) =  (B(1) - CD(1)*sqrt(h(1)))/pi/h(1)/(D(1) - h(1));
else
sys(1) = 0;
end

%if x(1) >= D(1)-.1;
%x(1)  = D(1)-.1;              % Limite de Nível Máximo
%else
%   if x(1)   <= 0;
%      x(1)=  0;          % Limite de Nível Mínimo
%   end
%end

%=============================================================================
=====
% mdlOutputs
% Return the block outputs.
%=============================================================================
=====
%

function sys = mdlOutputs(t,x,u)
D = 22.5*ones(1,1);
% Output
if x(1)>=D(1)-.1;
x(1)>=D(1)-.1;
end
if x(1) > 0
sys(1) = x(1);
else
sys(1) = 0;
end


% endmdlOutputs
```