Identificação de Um Sistema Dinâmico Assimétrico com Protocolo *Foundation Fieldbus* Aplicando Rede Neural Artificial Feedforward com Sinal PRBS

Larissa Moura Andrade, Filipe Miguel Hylário

Resumo — Alguns sistemas não lineares são complexos, de difícil identificação e se opõem a métodos matemáticos convencionais. Assim sendo, não é possível atingir o modelo ideal do sistema. Foi então que, buscando uma melhor solução para tal problema, apresenta-se como alternativa esperançosa a área de Inteligência Artificial (IA). Dentre as linhas de pesquisa dessa área, pode-se dar ênfase nas Redes Neurais Artificias. As suposições feitas em identificação utilizando redes neurais são mais flexíveis que aquelas feitas para identificar sistemas utilizando métodos matemáticos convencionais, logo que a rede neural não exige nenhum conhecimento prévio dos relacionamentos funcionais entre a entrada e a saída do sistema. A proposta deste trabalho é implementar uma rede *Perceptron* Multicamadas com arquitetura *feedfoward* e algoritmo de aprendizagem *backpropagation* modificado por *Levenberg-Marquardt* utilizando sinal *PRBS* através do *software MATLAB* para identificar o comportamento da Planta Didática *SMA*R com Protocolo *Foundation Fieldbus*. A comunicação entre a planta e o *MATLAB* será feita utilizando a interface de comunicação *OPC*.

Palavras Chaves — Modelagem e Identificação de sistemas, Redes Neurais Artificiais, Planta Didática *SMAR* e Padrão de Comunicação *OPC*.

I. *INTRODUÇÃO*

Identificação de sistemas nada mais é que a determinação de um modelo matemático do sistema representando suas características principais, utilizando para tanto observações de dados de entrada e saída [1].

Alguns sistemas não lineares são complexos e de difícil identificação e resistem a métodos matemáticos convencionais. Assim sendo, não é possível atingir o modelo ideal do sistema e então o projetista é obrigado a aproximar o sistema real, consequentemente, o modelo adquirido não é totalmente fiel a realidade do sistema. Foi então, que buscando uma melhor solução para o problema, apresentou-se uma alternativa promissora a área de Inteligência Artificial (IA), que é constituída por ferramentas não convencionais capazes de identificar e controlar sistemas complexos e que envolvem não linearidades. Dentre as linhas de pesquisa da IA, pode- se dar ênfase nas Redes Neurais Artificias, as quais são inspiradas no neurônio biológico [5].

[1]Larissa Moura Andrade é mestranda em Engenharia Elétrica - Sistemas de Energia pela Universidade Federal de Juiz de Fora (UFJF). Pós Graduada com Especialização em Engenharia de Segurança do Trabalho pela Faculdade Redentor (2015). Possui graduação em Engenharia de Controle e Automação pelo Centro Federal de Educação Tecnológica de Minas Gerais (2014) e curso técnico em Segurança do Trabalho pelo Instituto Federal Fluminense (2013). (E-mail: larissa.andrade@engenharia.ufjf.br).
Filipe Miguel Hylário é mestrando em Engenharia Elétrica - Sistemas de Energia pela Universidade Federal de Juiz de Fora (UFJF). Possui graduação em Engenharia Elétrica pela Univerisidade Federal de Juiz de Fora (UFJF) (2013). (E-mail: filipe.miguel@engenharia.ufjf.br).

As suposições feitas em identificação utilizando redes neurais são menos ríspidas que aquelas feitas para identificar sistemas utilizando métodos matemáticos convencionais. Para a identificação clássica, o projetista deve detalhar as entradas e saídas do sistema, enquanto para identificação utilizando redes neurais, é imprescindível apenas especificar a topologia da rede, logo que já é o suficiente para descrever o mapeamento de entrada-saída. O benefício de utilizar tal técnica é que a rede neural não exige nenhum conhecimento prévio dos relacionamentos funcionais entre a entrada e a saída do sistema, pelo contrário, a ferramenta simplesmente origina um mapeamento da entrada-saída a partir dos dados do treinamento da rede [8].

II. Redes Neurais Artificiais

Redes neurais artificiais são sistemas computacionais que imitam as funções do sistema nervoso biológico, usando um grande número de elementos básicos interconectados, chamados neurônios artificias. Da interação destes elementos relativamente simples emerge um comportamento global coerente, e que pode ser adaptado a diversos contextos de aplicação [8].

III. Redes aplicadas à apresentação de Modelos

A utilização das redes neurais faz com que a dinâmica de sistemas complexos seja modelada e um controle exato possa ser atingido devido ao treinamento, mesmo sem ter informação a priori sobre os parâmetros do sistema [12, 13].

As redes neurais são empregadas em identificação de sistemas dinâmicos complexos devido à capacidade de aproximar qualquer mapeamento contínuo via aprendizagem e de generalização. Também são adequadas ao processamento paralelo e exibem tolerâncias a falhas[9].

IV. Treinamento de uma rede Perceptron Multicamadas

O modelo de aprendizado utilizado em redes *Perceptron* Multicamadas é a Regra Delta e a Regra Delta Generalizada ou *backpropagation*. Conforme Braga [2] é possível treinar eficientemente redes com várias camadas escondidas, utilizando o algoritmo *backpropagation*. Os pesos sinápticos

são ajustados (aplicando o método do gradiente) de acordo o erro encontrado entre a saída da rede e o resultado desejado.

V. Planta Didática com Protocolo Fundation Fieldbus

A Planta Didática da *SMAR* com protocolo Foundation Fieldbus, com o auxílio de instrumentos e ferramentas de configurações iguais aos das indústrias, tem como finalidade simular diversos processos reais, os quais são comumente encontrados nas fábricas, porém, em pequenas escalas e dentro de um laboratório.

Dentre os vários instrumentos que a planta possui, os que foram utilizados para a implementação deste trabalho foram:

– 1 tanque de processo (tanque de aquecimento);
– 1 tanque de armazenamento de água;
– 1 bomba hidráulica (circulação da água na planta);
– 2 válvulas de controle (controle de fluxo de água na planta);
– 1 posicionador de válvula de controle FY-31 (abertura/fechamento da válvula);
– 1 transmissor de pressão diferencial Foundation Fieldbus LD302D (mede o nível do tanque de aquecimento utilizando um microprocessador) juntamente com o medidor FIT-31 (vazão de água na entrada do tanque) e o medidor LIT-31 (nível do tanque de água quente);
– 1 rotâmetro de água (valor instantâneo da vazão de água no circuito);

A variável de processo nível utilizada, apresenta não linearidade com característica de dinâmica assimétrica, o que se caracteriza por apresentar crescimento rápido no início e lento próximo à região de estado estacionário. Isto se dá devido ao fato da pressão aumentar sobre a tubulação de alimentação do tanque de nível [14].

Na Figura 1, pode ver a Planta Didática da *SMAR* com protocolo *Foundation Fieldbus* com vista frontal.

Figura 1 - Planta Didática *SMAR*

VI. Padrão de Comunicação OPC

O *OPC* ("*OLE* for *Process Control*", onde significado de *OLE* é "*Object Linking and Embedding*") é uma interface de comunicação que possui o objetivo de simplificar o acesso aos instrumentos de campo e diminuir os problemas de drivers de equipamentos industriais de diferentes fabricantes. Este padrão de comunicação tem como principais funções: procurar e mostrar os servidores *OPC* presentes, procurar e mostrar as *TAG* de dados disponíveis nos servidores encontrados, gerenciar grupo de dados, ler/escrever dados e, por fim, diagnosticar erros. O *OPC* é responsável por estipular as normas para desenvolver sistemas com interfaces padronizadas para comunicação dos dispositivos de campo, como: *CLP*, sensores, transmissores, balanças, dentre outros com sistemas de monitoração, supervisão e gerenciamento.

Para conseguir a comunicação, faz-se necessário um cliente e um servidor *OPC*. O cliente *OPC* é uma aplicação de software que precisa dos dados de processo, enquanto o servidor *OPC* é o driver que faz a conexão com os hardwares de campo e envia a informação aos clientes. Neste trabalho, o cliente *OPC* será o *MATLAB* e todos os comandos da planta serão dados através desse software, eliminando assim o uso do *CLP*. Portanto, o *MATLAB* que irá inspecionar e escrever valores nos dispositivos da planta através do servidor *OPC*. Cada dispositivo é configurado dentro do servidor *OPC* através de um TAG da SMAR que será instalado no mesmo computador que o cliente *OPC*, ou seja, o mesmo computador irá funcionar como cliente e servidor *OPC* ao mesmo tempo.

VII. Metodologia: Treinamento, Validação e Teste

O problema geral de identificação de sistemas não lineares pode ser observado na Figura 2. Os parâmetros do modelo de identificação foram estimados conforme o modelo foi alterado ao longo do tempo, de modo que a diferença entre a saída da planta e a saída do modelo fosse minimizada. A ideia foi que o processo de identificação produzisse um modelo de um sistema sem o conhecimento prévio da dinâmica do sistema, modelagem caixa preta.

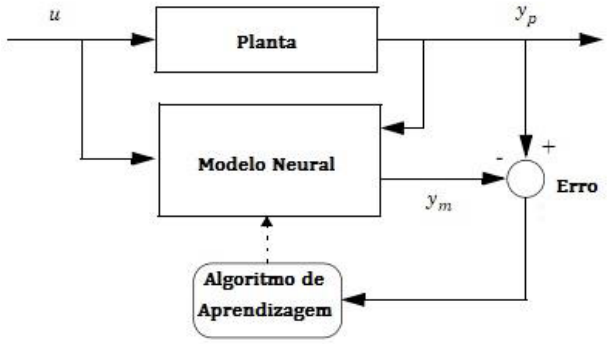

Figura 2 - Modelo de identificação do sistema

A primeira etapa na identificação do sistema foi a experimentação com a planta para desenvolver por utilizar este sinal no treinamento. Este conjunto de dados foi utilizado para treinar a rede. Foi aplicado um sinal *PRBS* (*Pseudo Random Binary Sequence*) à planta, fazendo com que toda gama de espectro de frequência da planta fosse identificado. Dessa forma, a válvula recebeu aberturas e fechamentos de maneira

aleatória. Optou-se por utilizar este sinal no treinamento e validação da rede, segundo NGUYEN [11], a identificação do sistema será mais robusta do que se tivéssemos aplicado um degrau, por exemplo. Aplicando o sinal *PRBS* será abrangido todo o funcionamento da planta, evitando, assim, que se fizessem diversos testes de curva de reação.

O passo seguinte é a definição dos parâmetros da rede. O treinamento utilizado foi o supervisionado juntamente com o algoritmo de aprendizagem *backpropagation*, modificado por *Levenberg-Marquardt*, usado para treinar a rede. Com o treinamento da rede neural foi possível classificar o comportamento da planta, observando a saída da mesma, a qual deve ter como base os erros que puderam ocorrer na resposta do sistema em comparação com uma resposta desejada. O algoritmo *backpropagation* ajustou os pesos da rede e na prática esse algoritmo tende a convergir muito lentamente, fazendo com que tivesse um grande esforço do computador. Para contornar, tal situação, técnicas foram introduzidas ao algoritmo para diminuir esse tempo de convergência, bem como o esforço computacional. Dentre as técnicas mais utilizadas, destacou-se o algoritmo de *Levenberg-Marquardt* [6]. Utilizou-se a rede *Perce*ptron Multicamadas com arquitetura *feedfoward*. Sendo assim, fez-se necessário escolher a topologia interna da rede.

Através da resposta da planta, foi possível que o modelo neural fizesse a identificação do sistema, sendo que para a rede aprender, teve que considerar o nível do tanque em certo instante (y), bem como no instante anterior (y-1), logo que para saber qual o nível do tanque em um determinado momento, foi necessário saber o quanto de água já havia dentro dele no instante anterior. Portanto, faz-se necessário esses valores como os sinais de entrada da rede. A abertura do posicionador da válvula de controle também foi uma entrada da rede, logo que o nível do tanque depende diretamente da abertura da mesma, ou seja, a rede neural teve 3 sinais de entradas. A escolha do número de neurônios na camada de saída do modelo foi com base no número de saídas do sistema. Como a rede deve identificar somente a resposta da planta, ela tem 1 neurônio na saída.

Quanto ao número de camadas escondidas, optou-se por utilizar 2. Segundo Curry e Morgan [3], para problemas envolvendo sistemas variantes no tempo como este, recomenda-se a utilização de 2 ou mais camadas intermediárias, pois sabe-se que redes *PMC* com uma única camada escondida são, normalmente menos propensas a estacionar em mínimos locais devido sua estrutura mais compacta reduzi a complexibilidade geométrica da função que mapeia o erro quadrático médio.

Segundo Silva et al. [15] para este tipo de modelo, as funções de ativação mais apropriadas são a logística ou tangente hiperbólica, a fim de evitar a saturação dos neurônios. Devido ao resultado da saída da função tangente hiperbólica sempre assumir valores entre -1 e 1, optou-se por utilizar a função logística em todas as camadas, visto que o nível da água nunca assumirá valores negativos. Utilizando a função logística teremos sempre como resultado da saída valores reais entre 0 e 1, dessa forma, faz-se necessário normalizar os valores da entrada da rede dentro desse intervalo. Como o valor mínimo das entradas é 0 (válvula totalmente fechada) e o valor máximo é 100 (válvula totalmente aberta), basta dividir todos os valores por 100.

A utilização de redes *PMC* para identificar sistemas dinâmicos podem ter duas configurações diferentes, sendo a rede *PMC* com entradas atrasadas no tempo e a rede *PMC* com saídas recorrentes às entradas. Como o modelo utilizado foi não linear e variante no tempo, pois o nível do tanque varia conforme a abertura ou fechamento da válvula de controle, a rede utilizada foi a *PMC* com saídas recorrentes às entradas, as quais possibilitam a recuperação de respostas passadas a partir da realimentação de sinais produzidos em instantes anteriores.

Esse tipo de topologia possui memórias apresentando a capacidade de memorizar saídas passadas a fim de gerar a resposta atual ou futura, além de ser uma ferramenta bem flexível para aplicações abrangendo identificação de sistemas [15]. Devido a este tipo de configuração, os quais somente os resultados gerados pelos neurônios de saída da rede neural foram realimentados às suas entradas, essa rede também foi chamada de rede de Elman ou rede recorrente simples [4].

A. Treinamento e Validação da Rede

A seguir, os passos que foram utilizados para o treinamento e validação da Rede Neural Artificial [11]:

Passo 1. A primeira necessidade foi obter os pares de entrada-saída de dados que puderam ser usados para treinar uma rede neural. Neste modelo, foi simulado o comportamento do sistema não linear e desenvolvido os dados de formação adequada.

Passo 2. Gerar um vetor de entrada u.

Passo 3. Simular a resposta do sistema não linear utilizando o vetor de entrada gerado no Passo 2 e criar os pares de entrada e saída de dados de treinamento. Os dados de formação de entrada e saída foram escalados, neste caso, por um fator de 100, após vários testes. Acima ou abaixo da escala de ampliação pode afetar significativamente as propriedades de convergência da rede, logo que esse valor influencia diretamente no aprendizado da rede.

Passo 4. Definir-se os vetores de entrada e saída de dados para treinamento da rede neural. Vetor de entrada = *plantin* e vetor de saída = *plantout.*

Passo 5. Como primeira tentativa de obter uma rede neural adequada, foi escolhida a topologia da rede, isto é, o número de camadas e o número de neurônios em cada camada. A escolha de um grande número de neurônios aumenta os cálculos e, portanto, afeta o tempo de convergência. Outra decisão importante foi a escolha da função de ativação apropriada que caracteriza o comportamento de entrada-saída. O algoritmo "*trainlm*", o qual será utilizado, é um treinamento da rede que atualiza os valores de peso de acordo com *Levenberg-Marquardt*.

Passo 6. Inicializar a estrutura de rede neural escolhida.

Passo 7. Definir o número de épocas de treinamento e a tolerância de erro desejada. O número de épocas deverá ser aumentado se a tolerância de convergência não for cumprida dentro do número especificado. Por outro lado, pode ser que se tenha que aumentar a tolerância para a obtenção de convergência. Como tal, é necessária alguma experimentação no dimensionamento de dados de treinamento para obter um conjunto adequado.

Passo 8. Treinar a rede neural para os dados de entrada e saída. Como a inicialização dos pesos foi feita de forma aleatória, então se obtem vários resultados de treino para uma mesma base de dados, com mesma arquitetura e parâmetros. Portanto, foi necessário realizar vários treinos na rede até que se alcance uma rede que consiga reconhecer padrões, generalizar e classificar novas entradas dentro destes padrões treinados.

Passo 9. Simular o comportamento da planta e o da rede neural treinada. Comparar os resultados obtidos para se analisar a capacidade de generalização da rede.

Passo 10. Se os resultados forem satisfatórios, então é feita a identificação do sistema não linear e pode usar a rede neural treinada para fins de controle. Se, contudo, os resultados não forem satisfatórios, certa quantidade de experimentação seria necessária na escolha do número apropriado de neurônios em cada uma das camadas intermediárias e/ou o aumento o número de camadas escondidas, a escolha de um fator de escala apropriado ou um algoritmo de treinamento diferente.

B. Testes Finais da Rede

Após se fazer o treinamento e a validação da rede, a etapa seguinte foi a realização de teste com uma nova entrada aplicada à rede com o intuito de certificar que a rede aprendeu o comportamento da planta. Ao invés do sinal *PRBS*, foi aplicado um sinal degrau na planta e posteriormente na rede. Dessa forma, a análise do teste foi que a rede conseguiu aprender o novo comportamento do sistema, ou seja, a mesma obteve a capacidade de generalização do funcionamento da planta.

Passo 11. Gerar um vetor de dados de entrada diferente do vetor de entrada utilizado para o treinamento e validação da rede. Simular o comportamento da planta e o da rede neural treinada.

Passo 12. Se os resultados forem satisfatórios, então foi feita a identificação do sistema não linear e pode usar a rede neural treinada para fins de controle. Se, contudo, os resultados não são forem satisfatórios, certa quantidade de experimentação seria necessária para selecionar o número apropriado de neurônios nas camadas escondidas, a escolha do fator de escala apropriado e algoritmo de treinamento diferente. Será então preciso realizar mais experimentos, a fim de obter uma melhor ideia de como esses parâmetros afetam no desempenho da rede neural e nas características do treinamento.

VIII. *Resultados*

Primeiramente, aplicou-se um sinal *PRBS* no modelo neural para treiná-lo e validá-lo. Posteriormente, para a realização do teste, aplicou-se uma entrada degrau na rede para certificar-se de que a mesma conseguiu generalizar o funcionamento da Planta Didática *SMAR* com Protocolo *Foundation Fieldbus.*

A. Aquisição da Curva de Reação da Planta

Para realizar a identificação da planta didática, foi desenvolvida uma aplicação utilizando o software *MATLAB/SIMULINK* que, através do padrão de comunicação *OPC*, se comunica com a planta. Neste caso, o cliente *OPC* é o software *MATLAB/SIMULINK*, responsável por ler e escrever valores na saída e da entrada periférica do servidor *OPC* (*TAG List* da *SMAR).*

O *MATLAB*® contém uma toolbox chamada *OPC* a qual que faz com que o servidor *OPC* se comunique com a planta. O *Simulink* já possui os blocos da interface *OPC*, sendo somente necessário fazer a configuração desses blocos para utilizá-lo na comunicação. Utilizaremos 3 blocos da *OPC toolbox*:

– *OPC Configuration*: é através desse bloco, que se faz a configuração do *MATLAB* como cliente *OPC*. Para isto, deve-se realizar a comunicação entre o servidor previamente instalado. O servido da *SMAR*, é o "*Smar.DfiOleServer*".

– *OPC Read*: este bloco faz a leitura da variável de entrada, podendo ser sensores de nível, de temperatura, de vazão, etc.

– *OPC Write*: este bloco é utilizado para escrever um valor em um dispositivo de saída, podendo ser uma válvula de controle, uma bomba, um contator, etc.

Devido à utilização destes blocos, é possível que o *software* se comunique com a planta didática e adquira a resposta ao sinal *PRBS* para realizar a identificação do sistema.

Foi aplicado um sinal *PRBS* sobre a válvula que faz parte da malha de nível da planta didática *Foundation Fieldbus*. Este sinal, que varia de 0 a 1, foi multiplicado por 100, devido à abertura da válvula variar entre 0% e 100%. A Figura 3 mostra o sinal aplicado.

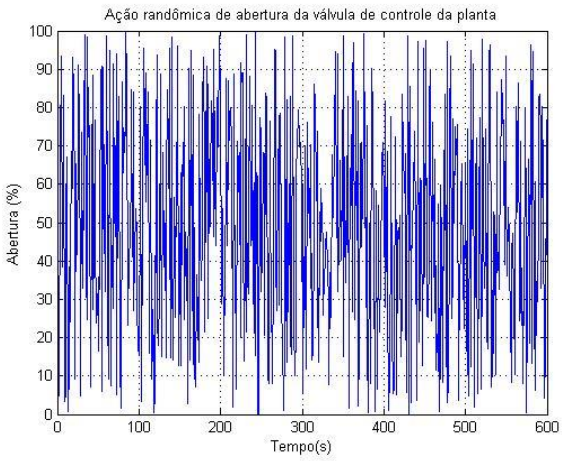

Figura 3 - Sinal *PRBS* aplicado

Para aplicar o sinal *PRBS* utilizou-se blocos no *Simulink* com as *TAGs* já configuradas (Figura 4).

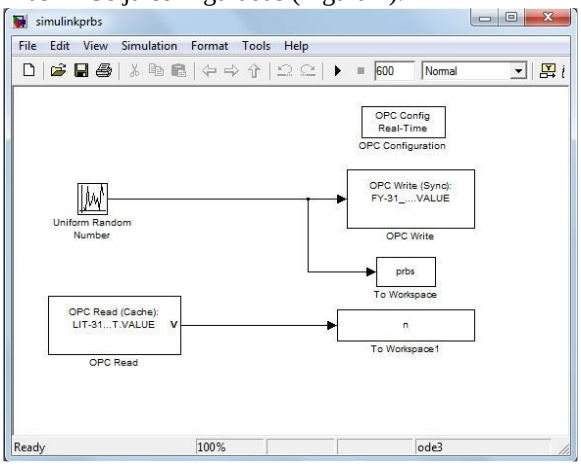

Figura 4 - Diagrama de blocos do *Simulink* para aplicação do sinal *PRBS* e leitura do sinal de nível.

Ao aplicar o sinal *PRBS* na válvula, pode-se observar na Figura 5, a resposta da planta no tocante ao sensor de nível.

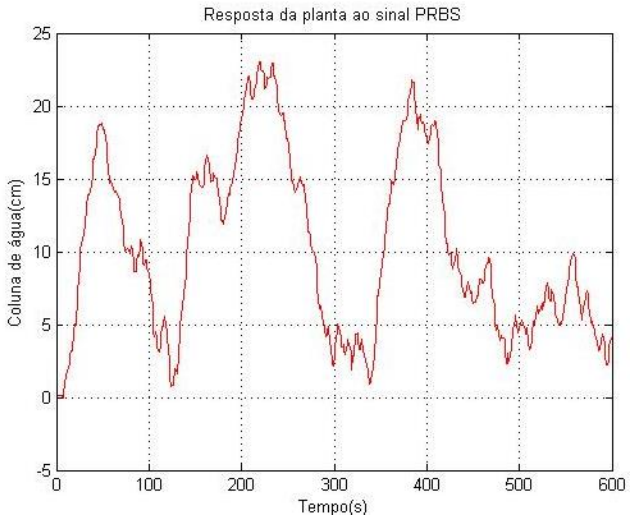

Figura 5 - Resposta da planta de nível com abertura variável.

Estes sinais são utilizados para o treinamento da rede segundo metodologia proposta por Nguyen et al [11].

B. Escolha da Topologia do Modelo Neural

Foram feitos testes com diversas topologias diferentes. As que mais se aproximaram dos valores esperados, e que possuíam uma menor quantidade de neurônios, são as três topologias mostradas a seguir. À vista disso, definiu-se para o experimento:

- Número de épocas de treinamento = 500;
- Tolerância de erro desejada = 0.0000005;
- Taxa de aprendizado = 0.01.

Topologia 1

- Camada de entrada → 3 sinais de entrada;
- 1ª Camada neural escondida → 10 neurônios;
- 2ª Camada neural escondida → 1 neurônio;
- Camada neural de saída → 1 neurônio.

Topologia 2

- Camada de entrada → 3 sinais de entrada;
- 1ª Camada neural escondida → 10 neurônios;
- 2ª Camada neural escondida → 3 neurônios;
- Camada neural de saída → 1 neurônio.

Topologia 3

- Camada de entrada → 3 sinais de entrada;
- 1ª Camada neural escondida → 10 neurônios;
- 2ª Camada neural escondida → 5 neurônios;
- Camada neural de saída → 1 neurônio.

Observou-se que, acima de dez neurônios na primeira camada escondida, o erro é irrelevante de uma topologia quando comparada a outra. Porém, com valores menores que dez, o resultado não é satisfatório, visto que que a rede perde a capacidade de aprendizado, ocorrendo o que chamamos de *underfitting*. Portanto, optou-se, por escolher dez neurônios nesta camada.

Na segunda camada escondida, após diversos testes, foi constatado que com poucos neurônios a rede conseguia se comportar conforme o desejado, já com uma quantidade maior de neurônios, ela perdia a capacidade de generalização, o que chamamos de *overfitting*, visto que com mais neurônios, o modelo neural considerava sinais indesejados. Portanto, foi feita a escolha de topologias com um baixo número de neurônios nesta camada, como foi visto nas Topologias 1, 2 e 3.

Na Tabela 1, podem-se ver os valores dos Erros Quadráticos Médios (EQM) das três topologias.

Tabela 1 – Relação dos Erros Quadráticos Médios das 3 topologias.

	EQM
Topologia 1	$7{,}74 \times 10^{-6}$
Topologia 2	$7{,}64 \times 10^{-6}$
Topologia 3	$7{,}51 \times 10^{-6}$

Os valores mostrados na Tabela 1 apresentam pouca diferença uns dos outros. Portanto, poderíamos optar por qualquer uma das três topologias. Decidiu-se, porém, escolher a Topologia 3, não só pelo menor Erro Quadrático Médio em relação as demais, mas, também, porque quanto maior o número de neurônios, maior será a capacidade de mapeamento não linear da rede.

C. Resultados Encontrados

Foi realizado o treinamento da rede com o auxílio do *MATLAB*, o qual pode ser visto na Figura 6.

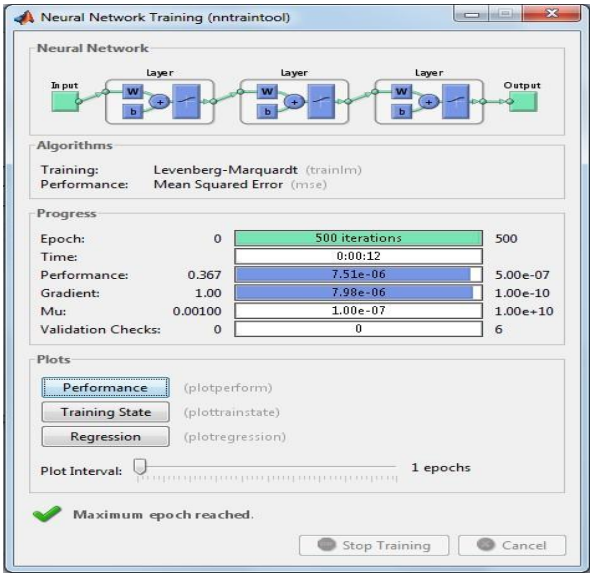

Figura 6 – Resultado do treinamento da rede neural.

Para demonstrar que o modelo neural conseguiu aprender o comportamento da planta, plotou-se um gráfico (Figura 7) com a resposta real juntamente com a resposta da rede.

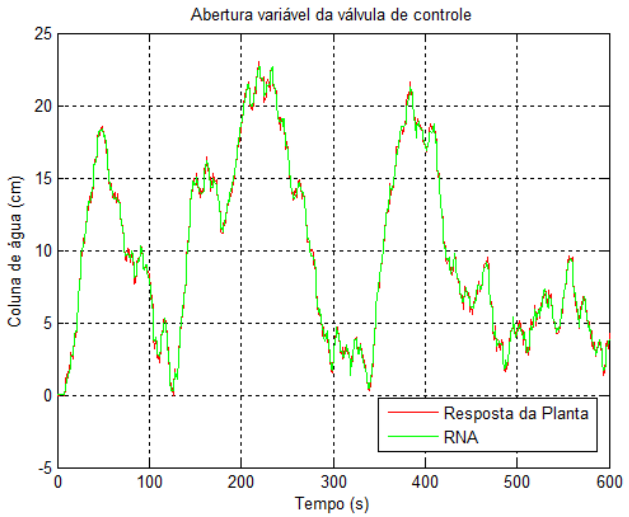

Figura 7 – Resposta da planta x Resposta da *RNA* com sinal *PRBS* aplicado.

Nota-se que a rede conseguiu aprender o comportamento da planta didática, aproximando-se de forma factível ao modelo real. Através da Figura 7, a resposta da *RNA* está praticamente sobreposta a resposta da planta. Visto que o treinamento e a validação da rede foram realizados com sucesso, foi feito um teste com o modelo neural, para certificar-se de que esse alcançou a capacidade de generalização do funcionamento da planta.

Desta forma, retirou-se o sinal *PRBS* que estava sendo aplicado na válvula de controle da planta didática *Foundation Fieldbus* e introduziu-se uma entrada degrau unitário, ou seja, 100% de abertura da válvula. A Figura 8 mostra o diagrama de blocos utilizado para abrir a válvula em 100% e na Figura 9, pode-se ver o comportamento do sistema quando aplicado um degrau unitário na válvula de nível.

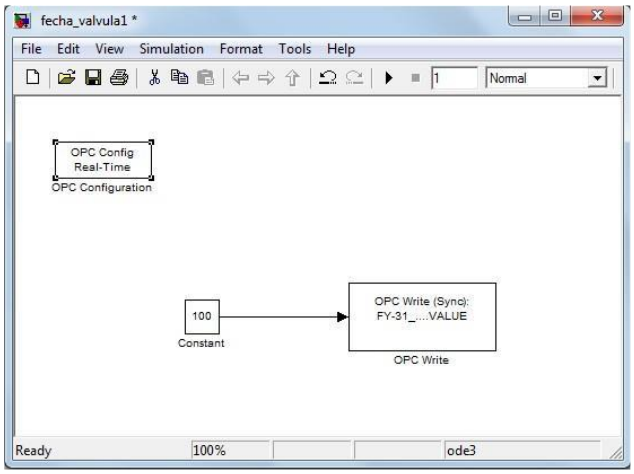

Figura 8 – Diagrama de blocos do Simulink para escrever na saída do posicionador da válvula de controle.

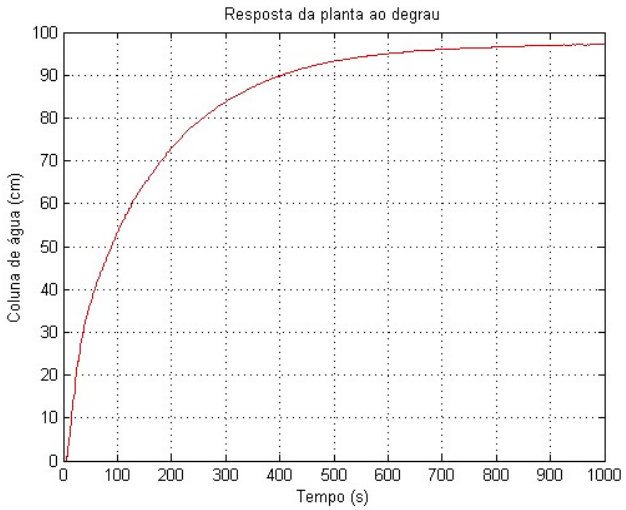

Figura 9 – Resposta da planta de nível com abertura da válvula em 100%.

Observou-se que, mesmo com a válvula de controle 100% aberta, o nível não alcança o topo do tanque, com o seu máximo chegando a 96 cm, visto que a saída contínua de água neste tanque não permite que o mesmo fique cheio por completo. Então, aplicou-se um degrau unitário na rede para demonstrar que o modelo neural conseguiu generalizar o comportamento da planta. A Figura 10 apresenta a resposta real juntamente com a resposta da rede.

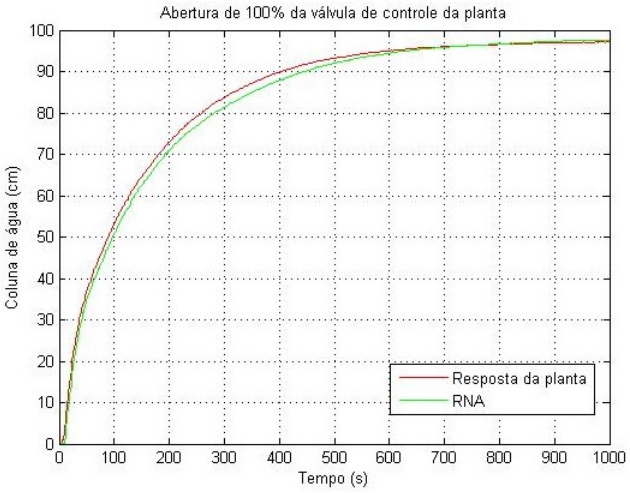

Figura 10 – Resposta da planta x Resposta da *RNA* com sinal degrau unitário aplicado.

Como pode ser visto, ocorreu um erro entre a saída da planta e a resposta da rede no regime transitório, porém o modelo obteve um resultado satisfatório, aproximando-se bastante da saída real. Observou-se também que quando o sistema entra em regime estacionário, o modelo neural atingiu erro quase nulo em relação à curva de reação.

IX. CONCLUSÕES

Percebeu-se com a realização deste trabalho que a seleção dos parâmetros da rede é uma etapa importante para que o treinamento e validação obtenham sucesso A escolha da topologia foi realizada de maneira empírica através de diversos experimentos até que se alcançasse o melhor resultado entre aqueles que foram testados.

Uma *RNA* se mostra de grande utilidade e bastante flexível, visto que o mapeamento de sistemas dinâmicos é feito de forma bem simples e viável. Neste caso, a identificação do sistema foi realizada utilizando uma rede *Perceptron* Multicamadas com arquitetura *feedforward* com saídas recorrentes às entradas, treinamento supervisionado e algoritmo de aprendizagem backpropagation modificado por *Levenberg-Marquardt*.

Foi possível verificar que a rede conseguiu aprender a dinâmica da planta, logo que foi aplicado um sinal *PRBS* no modelo neural, e este, por sua vez, se aproximou do processo real. Posteriormente, foi aplicada uma nova entrada no modelo neural, a fim de certificar se a rede conseguiu generalizar o sistema. Além disto, foi possível aplicar técnicas de identificação de sistemas em um processo real dentro de um laboratório com o auxílio da interface *OPC*, que fez a conexão entre a planta didática e o software *MATLAB*. Os resultados obtidos foram satisfatórios, ou seja, o objetivo deste trabalho foi alcançado

REFERÊNCIAS

[1] AGUIRRE, L. A., "Introdução à Identificação de Sistemas: Técnicas Lineares e Não Lineares Aplicadas a Sistemas Reais", Editora da UFMG, 1ª ed., Belo Horizonte, MG, 2000.

[2] BRAGA, A. P.,"Fundamentos de Redes Neurais Artificiais", Rio de Janeiro, 1998.

[3] CURRY,B., MORGAN, P. H, "Model Selection in Neural Networks: Some Difficulties", European Journal of Operational Research, vol. 170, 2006.

[4] ELMAN, J. L.,"Finding Structure in Time". Cognitive Science, vol. 14, 1990.

[5] FOLGER,T. A., "Fuzzy Sets, Uncertainty, and Information", State University of New York, Binghamton, Pretince Hall International, 1988.

[6] HAGAN, M. T., MENHAJ, M.B, "Training feedforward networks with the Marquart algorithm", IEEE Transactions on Neural Networks, vol. 5, nº 6, pp. 989-993, 1994.

[7] ISERMANN, R.,"Practical Aspects of Identification", Automática, vol. 16, 1980.

[8] LIMA, Clodoaldo Aparecido de Moraes, "Emprego de teoria de agentes no desenvolvimento de dispositivos neurocomputacionais híbridos e aplicação ao controle e identificação de sistemas dinâmicos", Tese de Mestrado, FEEC-UNICAMP, Campinas, SP, 2000.

[9] LJUNG, L. and SJÖBERG, J., "A system identification perspective on neural nets", Technical Report LiTH–ISY–I–1373, Division of Automatic Control Systems, Linköping, University, Linköping, Sweden, 1992.

[10] LJUNG, L., "Development of System Identification", 13th IFAC World Congress, San Francisco, USA, 1996.

[11] NGUYEN, Hung T., PRASAD Nadipuram R., WALKER Carol L., WALKER, Elbert A., "A First Course in FUZZY and NEURAL CONTROL," CHAPMAN & HALL/CRC, 2003.

[12] RONCO, E., GAWTHOP, P. J., "Modular neural networks: a state of the art. Technical Report CSC-95026", Centre for System and Control, Faculty of Mechanical Engineering, University of Glasgow, UK, 1995.

[13] RONCO, E., GOLLEE, H., GAWTHROP, P. J., "Modular neural network and self decomposition. Technical Report CSC-96012" Centre for System and Control, University. Of Glasgow, UK, 1996.

[14] SANTOS, Murillo Ferreira dos. Ambiente, "GUIDE-MATLAB para controle de um processo dinâmico assimétrico", Trabalho de Conclusão de Curso, CEFET-MG, Leopoldina, MG, 2011.

[15] SILVA, I. N., SPATTI, D. H., FLAUZINO, R. A.,"Redes Neurais Artificiais para Engenharia e Ciências Aplicadas", Editora ArtLib, 2010.

[16] VARGAS, José Alfredo Ruiz, "Identificação de Sistemas Dinâmicos via Redes Neurais Artificiais", Tese de Mestrado, ITA, São José dos Campos, SP, 1997.