FERRAMENTA COMPUTACIONAL PARA PROJETOS DE REDES NEURAIS ARTIFICIAIS UTILIZANDO LINGUAGEM GRÁFICA

Hugo da Silva Bernardes Gonçalves – hugo.bernardes@gmail.com
IFSP – Instituto Federal de Ciência e Tecnologia de São Paulo
Rua Pedro Vicente, 625
01109-010 – São Paulo - SP
Alexandre Brincalepe Campo – brincalepe@gmail.com
IFSP – Instituto Federal de Ciência e Tecnologia de São Paulo
Rua Pedro Vicente, 625
01109-010 – São Paulo - SP
Paulo Roberto Barbosa – pebarbosa@gmail.com
IFSP – Instituto Federal de Ciência e Tecnologia de São Paulo
Rua Pedro Vicente, 625
01109-010 – São Paulo - SP

Resumo: *Este trabalho apresenta um conjunto de ferramentas computacionais desenvolvidas especialmente para projetos de Redes Neurais Artificiais (RNA's) com algoritmos de retropropagação. Estas ferramentas são construídas através de uma plataforma baseada em linguagem gráfica (LabVIEW®), propondo uma alternativa de fácil interpretação para a construção de RNA's. Também é apresentada uma interface intuitiva para construção de modelos de RNA's do tipo Perceptron de múltiplas camadas com algoritmo de retropropagação. É possível, desta forma, extrair os resultados ou o modelo após o treinamento da rede. A engenharia de software aplicada ao desenvolvimento da ferramenta permite a apresentação das principais configurações para o treinamento das RNA's com algortimo de retropropagação. Os modelos gerados são testados para verificação das ferramentas. O acesso prático a essas ferramentas possibilita maior produtividade e entendimento dos principais conceitos de RNA's pelo usuário.*

Palavras-chave: *Linguagem Gráfica, Redes Neurais Artificiais, Engenharia de Software*

1. INTRODUÇÃO

As redes neurais artificiais (RNA's) são utilizadas em diversas aplicações comerciais e industriais como, por exemplo, em modelagem e controle de processos, reconhecimento de caracteres, reconhecimento de imagens, classificação de dados, controle inteligente. É possível observar, também, como as aplicações de RNA's nas

áreas de engenharias biológica e agrícola cresceram no período de 2003 a 2008 (HUANG, 2009). Há uma grande variedade de RNA's tais como: as redes de múltiplas camadas do tipo feed-forward, as redes com funções de bases radiais (BRF), os mapas auto-organizáveis de Kohonen e as redes recorrentes. Cada uma destas redes possui o seu algoritmo de treinamento como o de retropropagação, para o Perceptron de múltiplas camadas e a clusterização de dados para as BRF's e as redes de Kohonen (HUANG, 2009).

O presente documento apresenta uma alternativa para construção de modelos de Redes Neurais Artificiais (RNA's) implementada em liguagem gráfica LabVIEW®. A ferramenta computacional foi desenvolvida com o objetivo de fornecer a capacidade de configuração de modelos de RNA's, permitindo o aproveitamento dos resultados em sistemas de acordo com a finalidade do usuário que irá utilizá-la. Após a implementação, o sistema é testado e os resultados obtidos serão aplicados em um sistema existente para validação das fórmulas empregadas.

1.1. Ferramentas existentes

Existem diversas ferramentas computacionais que disponibilizam uma interface para configuração e treinamento de RNA's. Entretanto cada ferramenta destas possui características diferentes e com parâmetros de entrada específicos. Em diversos casos estas ferramentas não oferecem o produto do treinamento para que este seja utilizado em outros sistemas de forma simples e direta. Nestes casos ainda existe a necessidade de adquirir licenças destas ferramentas além de ser necessário familiarizar-se com a linguagem visual específica em cada uma delas.

Todo este processo é, muitas vezes, trabalhoso, sendo que o objetivo principal do usuário é utilizar uma RNA's para uma análise ou aplicação de forma rápida e eficiente. A seguir é apresentado um modelo de ferramenta que pode ser adquirida mediante o uso de plataformas pré-existentes, neste caso o MATLAB®, e a aquisição dos respectivos módulos ou toolbox para utilização de Redes Neurais Artificiais. Na Figura 1 está apresentada uma parte da interface deste módulo que permite também a utilização de comandos em formato de texto. Nesta ferramenta, caso o usuário queira acrescentar algum recurso como a importação de um arquivo, terá que digitar comandos em formato de texto.

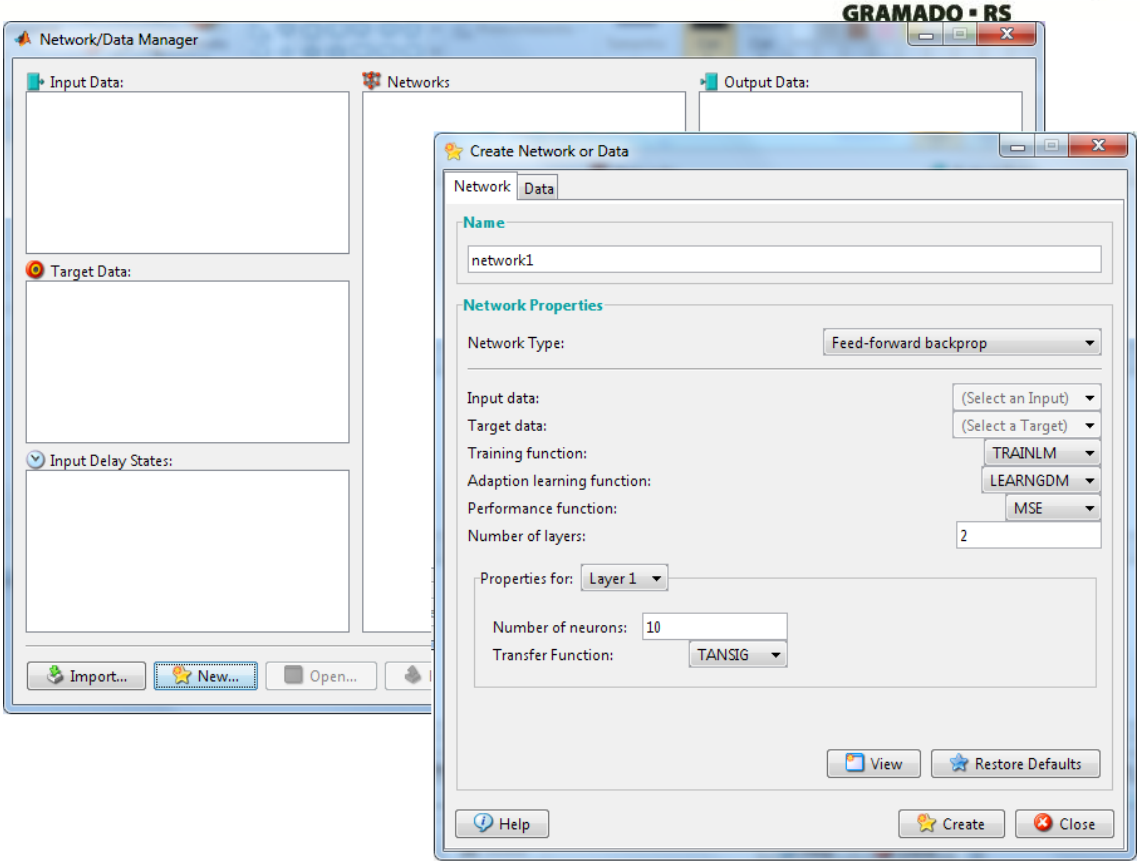

Figura 1 - Visual das telas do toolbox para o MATLAB®
Fonte: print screen do toolkit do MATLAB® executado em sistema operacional Windows

Alternativas de utilização das ferramentas do MATLAB® em sistemas de identificação e controle podem ser criadas como visto em Canete *et* al. (2008). Neste trabalho os autores também se utilizam dos recursos de interconexão do MATLAB® com a ferramenta Simulink® para a construção dos modelos em blocos. A seguir na Figura 2, uma parte da interface produzida neste trabalho.

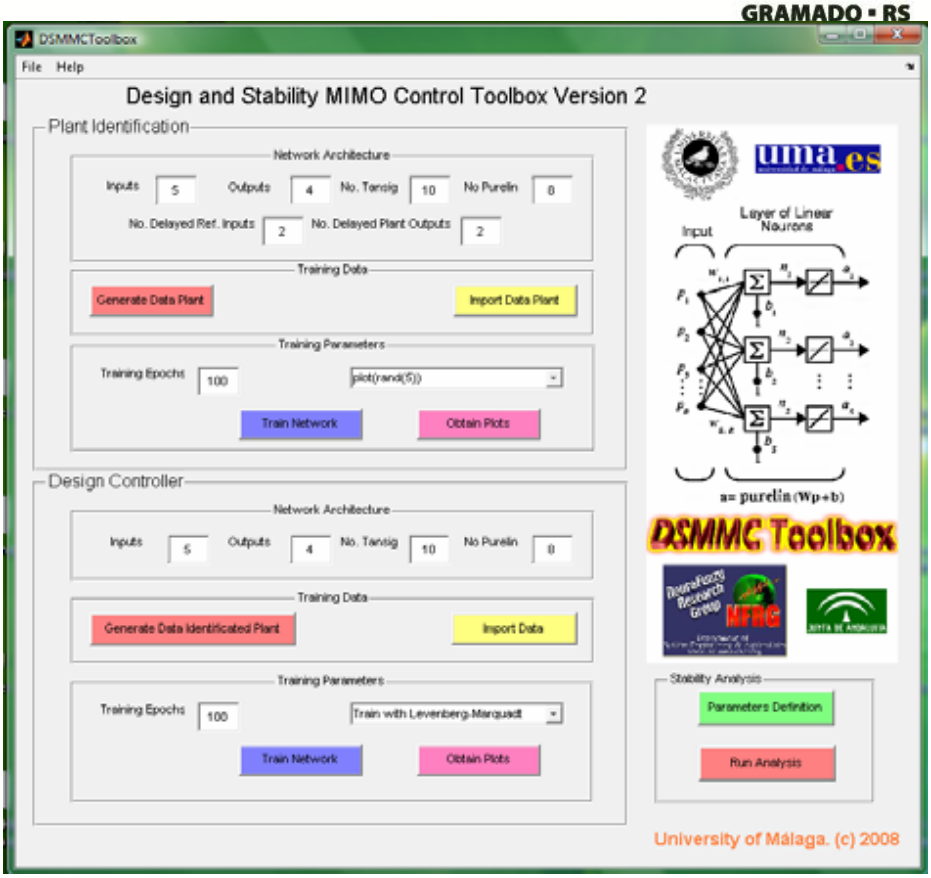

Figura 2 - Software de RNA's utilizando o Simulink e MATLAB
Fonte: CANETE *et* al., 2008

Em Campos *et* al. (2010) é possível verificar a construção de um software para treinamento de redes feed-forward com algoritmo de retropropagação. A ferramenta é desenvolvida em linguagem de programação JAVA utilizando o paradigma de programação orientada a objeto difundido na década de 90 que pressupõe uma organização de software em termos de coleção de objetos discretos incorporando estrutura e comportamento próprios. A seguir na Figura 3, é apresentada uma das telas da ferramenta desenvolvida em linguagem de programação JAVA (CAMPOS, 2010) sendo executado em sistema operacional Windows :

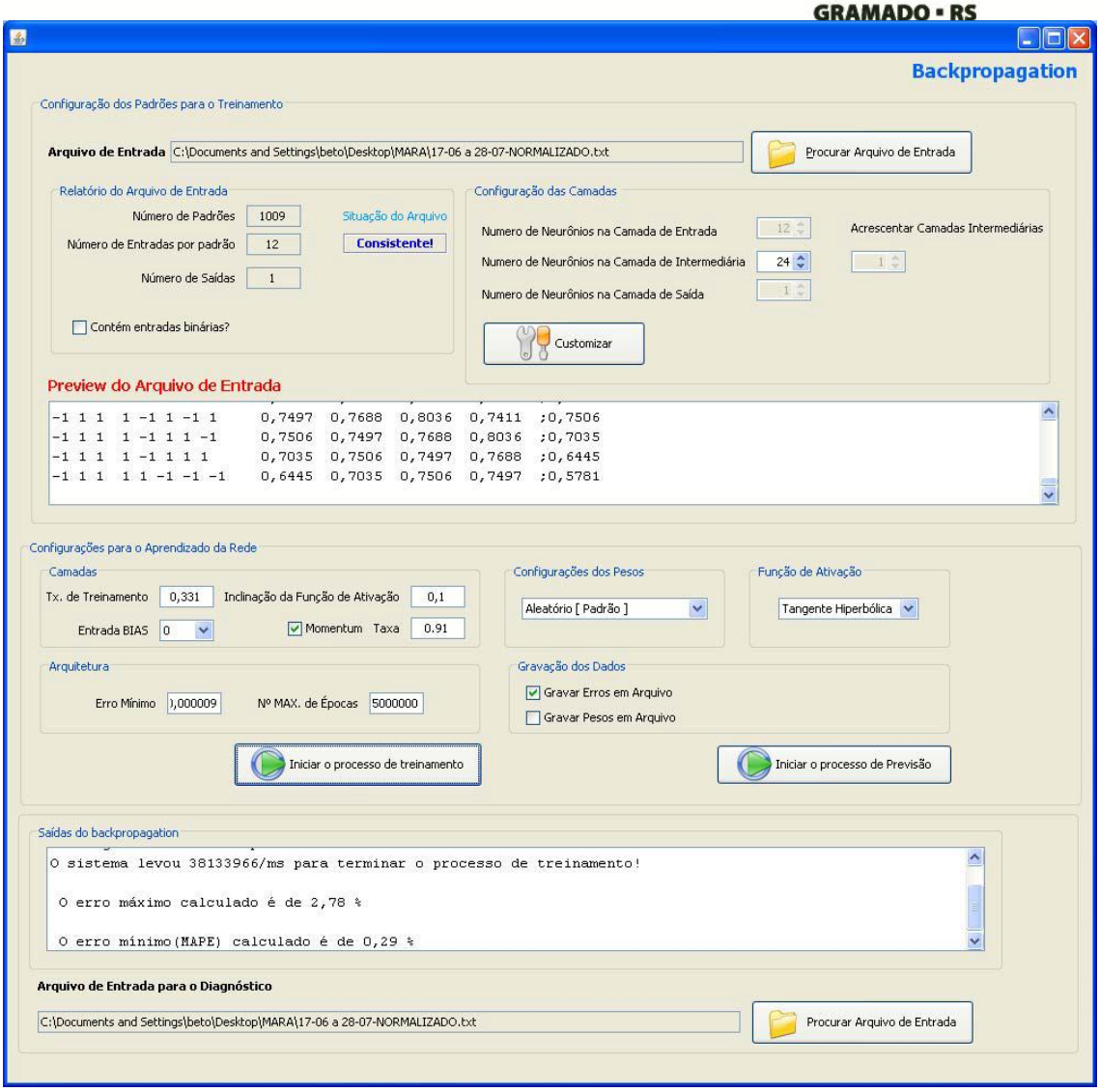

Figura 3 - Software para treinamento de RNA's desenvolvido em linguagem JAVA
Fonte: CAMPOS, 2010

Em outro trabalho, como é o caso do programa Multiple Back-Propagation (LOPES e RIBEIRO, 2001), também foi desenvolvido uma ferramenta para configurações de Redes Neurais Artificiais que executa em sistema operacional Windows. A imagem do software está na Figura 4:

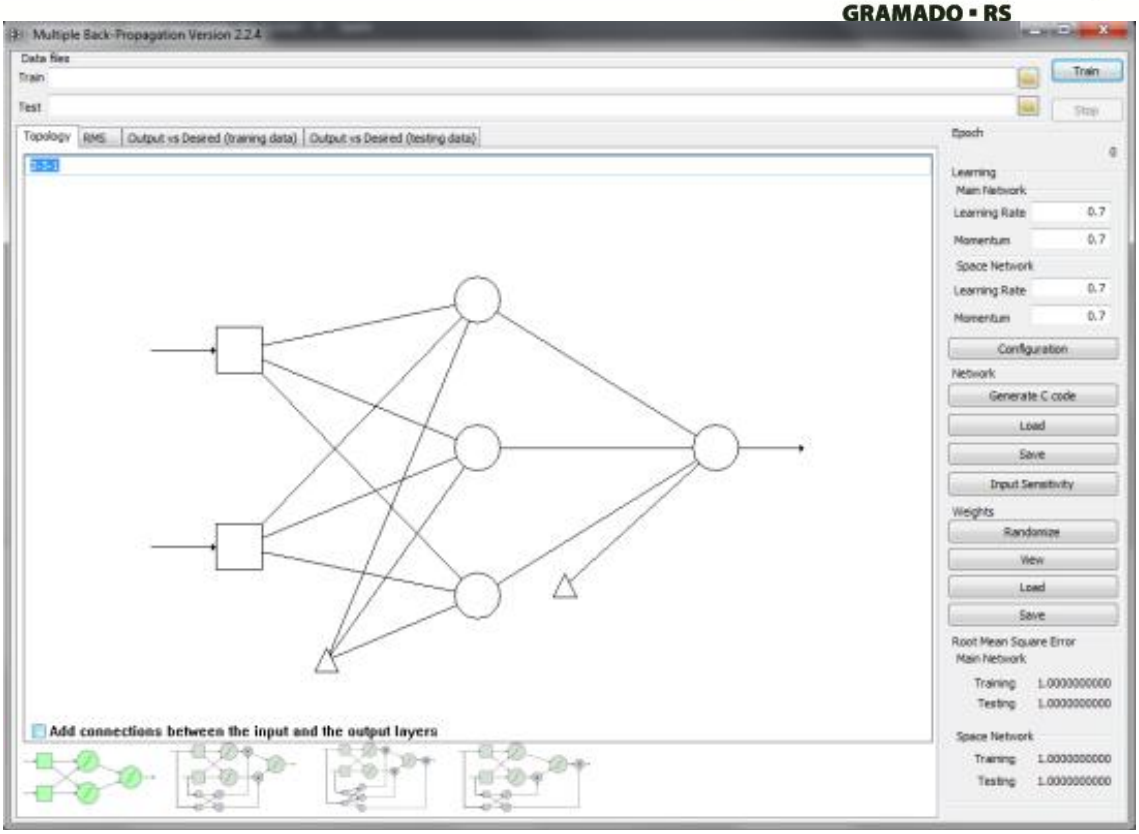

Figura 4 - Ferramenta para treinamento de modelos de redes neurais artificiais
Fonte: print screen do software MBP executado em sistema operacional Windows

A ferramenta desenvolvida por Lopes e Ribeiro (2001) permite algumas configurações mas para utilizá-la é necessário construir um arquivo com os dados de entrada e saída desejados seguindo um padrão pré-determinado.

No trabalho de Kowalska e Kaminski (2011), é apresentada a implementação de uma rede neural de múltiplas camadas para um estimador de velocidade em um sistema de driver duas massas. O algoritmo é implementado em um chip FPGA utilizando a linguagem LabVIEW®. O modelo com 2 camadas intermediárias, é desenvolvido especificamente para a aplicação que teve de levar em consideração questões como o espaço que o algoritmo ocupa no chip FPGA, já que este possui memória e tamanho de variáveis determinados. Segundo Kowalska e Kaminski (2011), o uso do código em LabVIEW® como linguagem de alto nível para desenvolver RNA's oferece vantagens como a rápida prototipação, rápida verificação, e a possibilidade de implementação em FPGA. A seguir na Figura 5 é apresentado o diagrama com a representação da RNA e sua interação no chip FPGA.

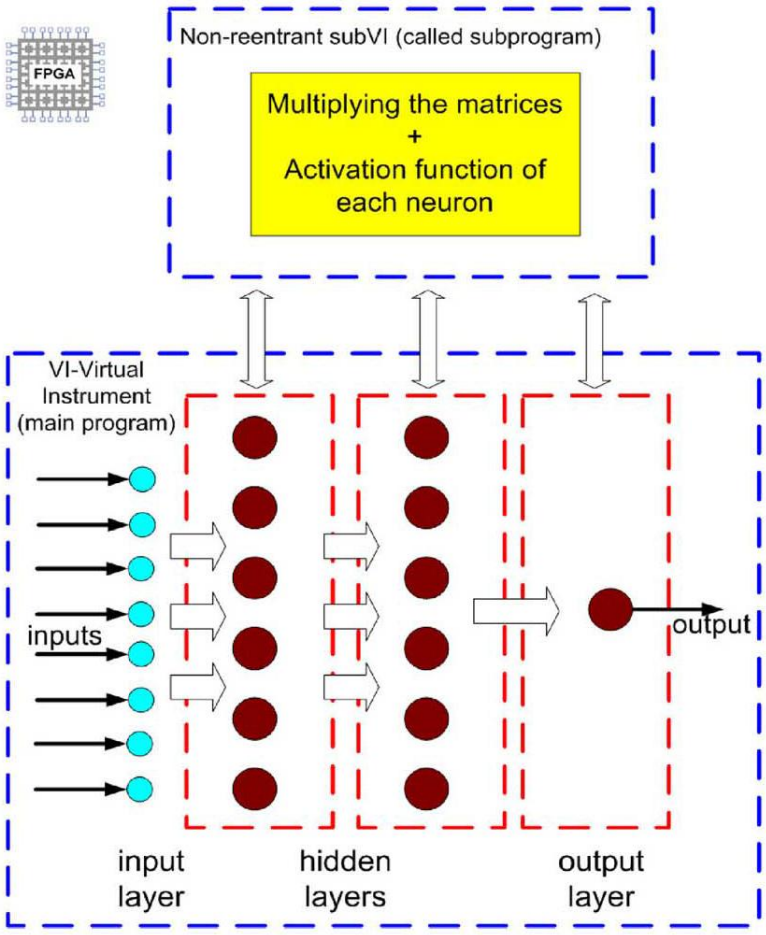

Figura 5 - Diagrama da implementação da RNA com duas camadas intermediárias
Fonte: KOWALSKA e KAMINSKI, 2011

Nos trabalhos de Ponce e Gutierrez (2010), é possível encontrar um conjunto de algoritmos de RNA's especialmente desenvolvido para aplicações em LabVIEW®. Entretanto neste trabalho não há uma interface padronizada para inserção dos dados de treinamento em um algoritmo como o Perceptron de múltiplas camadas, por exemplo. A seguir na Figura 6, a imagem de um código para algoritmos genéticos utilizando estas ferramentas:

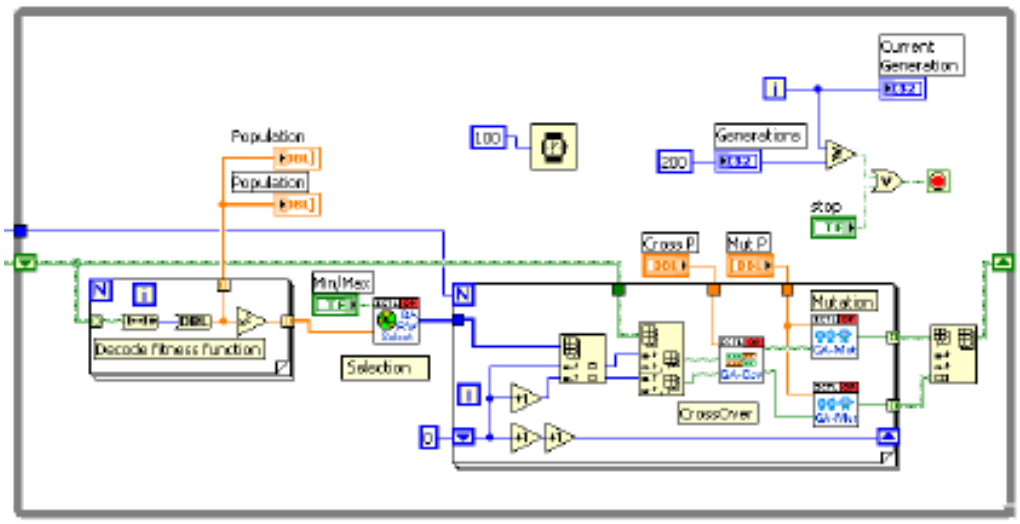

Figura 6 - Código em LabVIEW para implementação de RNA's
Fonte: PONCE e GUTIERREZ, 2010

2. OBJETIVOS

Observou-se que, apesar de existirem alternativas para implementações de RNA's não se tem uma interface intuitiva e padronizada para o treinamento das redes sendo possível extrair os modelos para utilizá-los rapidamente em outros sistemas. As ferramentas existentes possuem características muito específicas que exigem muito conhecimento e estudo para operá-las.

Optou-se por desenvolver uma ferramenta com maior capacidade para integração em sistemas de reconhecimento de padrão e sistemas de controle, por exemplo, utilizando a linguagem gráfica LabVIEW. Neste contexto, foi desenvolvida uma alternativa para utilização das ferramentas e também dos conceitos de RNAs utilizando redes do tipo Perceptron de Múltiplas Camadas com algoritmo de retropropagação. Foram identificadas as principais características de modelos de redes neurais artificiais e com isso determinados os parâmetros para as configurações dos modelos. A interface deve ser intuitiva e, portanto, a ferramenta deve ser baseada nas principais referências utilizadas para o estudo de RNA's com o algoritmo de retropropagação. Ao final do desenvolvimento a ferramenta foi disponibilizada em ambiente Web, no qual o usuário pode acessá-la através da internet.

3. REDES NEURAIS ARTIFICIAIS

Uma rede neural artificial é uma estrutura que processa informação de forma paralela e distribuída e que consiste de unidades computacionais (as quais podem possuir uma memória local e podem executar operações locais) interconectadas por canais unidirecionais chamados de conexões. Cada unidade computacional possui uma única conexão de saída, que pode ser dividida em quantas conexões laterais se fizer necessário, sendo que cada uma destas conexões transporta o mesmo sinal, o sinal de saída da unidade computacional. O processamento executado por cada unidade computacional pode ser definido arbitrariamente, com a restrição de que ele deve ser

completamente local, isto é, deve depender somente dos valores atuais dos sinais de entrada que chegam até a unidade computacional via as conexões e dos valores armazenados na memória local da unidade computacional.(HETCH-NIELSON, 1990)

3.1. A unidade computacional: neurônio artificial

Um neurônio artificial é uma unidade computacional que será a base para a formação das redes neurais artificiais. Na Figura 7 são identificadas as partes do modelo neuronal:

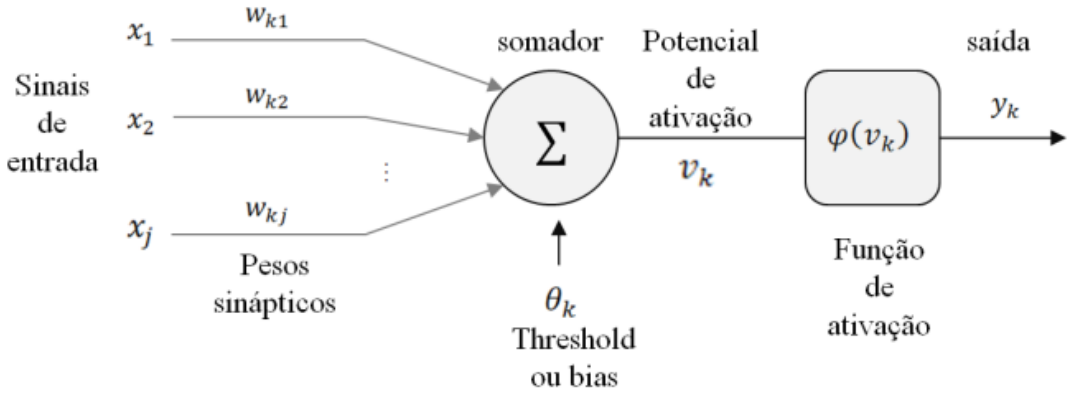

Figura 7 - Neurônio computacional
Fonte: elaborado pelo autor

O modelo apresentado na Figura 7 possui três partes:

a) Os elos de conexão ou as sinapses, onde é aplicado ao sinal de entrada x_j, e um peso ou força representado por w_{kj}.

b) Um somador que soma os sinais de entrada após eles serem ponderados pelos respectivos pesos.

c) Uma função de ativação, onde é aplicada uma função que restringe a saída do neurônio.

Este modelo inclui também um bias θ_k que tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se ele é positivo ou negativo.

A saída y_k do neurônio é dada conforme a equação (1) (HAYKIN, 2001):

$$y_k = \varphi\left(\theta_k + \sum_{i=1}^{p} w_{ki}x_i\right) \quad (1)$$

Sendo $\varphi_j(.)$ correspondente à função de ativação sobre a saída.

As funções de ativação podem ser do tipo Linear, Tangente hiperbólica ou Sigmóide.

3.2. Aprendizado da rede neural artificial

O aprendizado da rede ocorre através do seu treinamento onde serão ajustados os seus pesos. Considerando o aprendizado com supervisão forte (NASCIMENTO e

YONEYAMA, 2008) e utilizando o modelo perceptron de múltiplas camadas com algoritmo de retropropagação para seu treinamento, tem-se o ajuste dos pesos definido por:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) \quad (2)$$

sendo (HAYKIN, 2001):,
$\Delta w_{ji}(n) =$ variação do peso que conecta
o neurônio i ao neurônio j na iteração n;
$\eta =$ a taxa de aprendizagem;
$y_i(n) =$ sinal de entrada do neurônio j na iteração n representado pela equação (3):

$$y_i(n) = \varphi_j\left(v_j(n)\right) \quad (3)$$

E $\delta_j(n)$ será conforme a equação (4) para neurônios de camadas intermediárias:

$$\delta_j(n) = \varphi_j'\left(v_j(n)\right)\sum_k \delta_k(n)_k w_{kj}(n) \quad (4)$$

E conforme a equação (5) para neurônios na camada de saída:

$$\delta_j(n) = e_j(n)\varphi_j'\left(v_j(n)\right) \quad (5)$$

onde,

$$e_j(n) = d_j(n) - y_j(n)$$

Correspondendo ao erro entre o valor desejado e o sinal de saída na iteração n;

$$v_j(n) = \left(\theta_j + \sum_{i=1}^{p} w_{ji}x_i\right)$$

$\varphi_j'(.) =$ derivada da função de ativação.

Ainda é possível utilizar a uma constante de momento α (HUANG, 2009), conforme a equação (6):

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_j(n) \quad (6)$$

4. CONSTRUÇÃO DO ALGORITMO

Seguindo principais conceitos sobre RNA's (HAYKIN, 2001), (NASCIMENTO e YONEYAMA, 2008), (HUANG, 2009), tem-se as seguintes etapas para construção do algoritmo;

4.1. A fase de propagação para frente

O cálculo a partir da entrada e propagado até a geração da saída possui os seguintes passos:

a) Multiplicação das entradas pelos pesos sinápticos nas conexões iniciais;
b) Somatório dos resultados das multiplicações;
c) Função de ativação do resultado;
d) Multiplicação pelos pesos sinápticos da camada adiante (se houver), somatório e função de ativação;
e) Repetição destes cálculos até a última camada.
f) Geração do sinal de saída.

4.2. Propagação para trás

Cálculo que parte da saída obtida até a primeira camada com os seguintes passos:

a) Cálculo do erro representado pela diferença entre o valor desejado e a saída;
b) Multiplicação do erro pelo peso no sentido em direção à penúltima camada;;
c) Somatório destes novos valores nos neurônios da camada anterior (caso os valores não sejam originados da camada de saída) produzindo novos gradientes;
d) multiplicação destes novos erros pelos pesos em direção à camada anterior e assim por diante até os neurônios de entrada.

4.3. Ajuste dos pesos sinápticos

Em seguida serão calculados os novos pesos da rede:

a) Calculam-se os novos pesos das conexões de entrada somando-se o peso da iteração inicial ao produto entre a taxa de aprendizagem, o gradiente da conexão correspondente e a derivada da função de ativação dos neurônios da primeira camada em relação ao potencial de ativação;
b) Repete-se esta operação ajustando-se todos os pesos das conexões adiante.

Estes passos constituem uma iteração ou época. São realizadas quantas épocas forem necessárias até que se diminua o $\varepsilon_{\mathrm{med}}$, apresentado na equação (7), a um valor mínimo desejado (HAYKIN, 2001).

$$\mathcal{E}_{\mathrm{med}} = \frac{1}{\mathrm{N}} \sum_{\mathrm{n}=1}^{\mathrm{N}} \mathcal{E}(\mathrm{n}) \tag{7}$$

com $\mathcal{E}(\mathrm{n})$ conforme a equação 8:

$$\mathcal{E}(\mathrm{n}) = \frac{1}{2} \sum_{\mathrm{j} \in \mathrm{C}} \mathrm{e}_{\mathrm{j}}^{2}(\mathrm{n}) \tag{8}$$

onde C representa todos os neurônios de saída da rede.

5. IMPLEMENTAÇÃO EM LINGUAGEM GRÁFICA

Foi escolhida para a implementação a linguagem gráfica LabVIEW®. Esta linguagem se destaca por possuir uma fácil visualização do código como um todo. Possui recursos que facilitam identificar o fluxo dos dados nos algoritmos e corrigir possíveis erros durante o desenvolvimento. É uma linguagem de programação com código basicamente representado por blocos e linhas, constituindo um paradigma de código diferente dos habituais estruturados em texto.

5.1. Funções de ativação em linguagem gráfica

Na Figura 8 são apresentadas as funções de ativação implementadas em LabVIEW®:

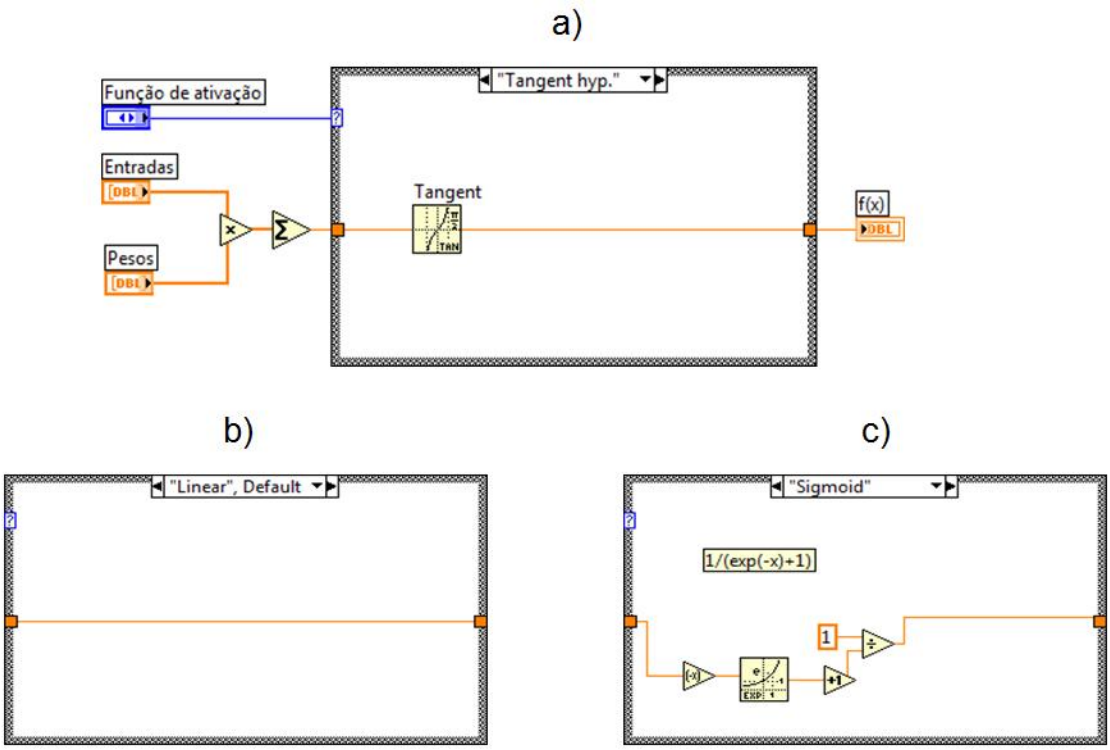

Figura 8 - Código com as funções de ativação: a) função tengente hiperbólica; b) função linear; c) função sigmoidal
Fonte: elaborado pelo autor

Na Figura 9 são apresentadas as derivadas das funções de ativação.

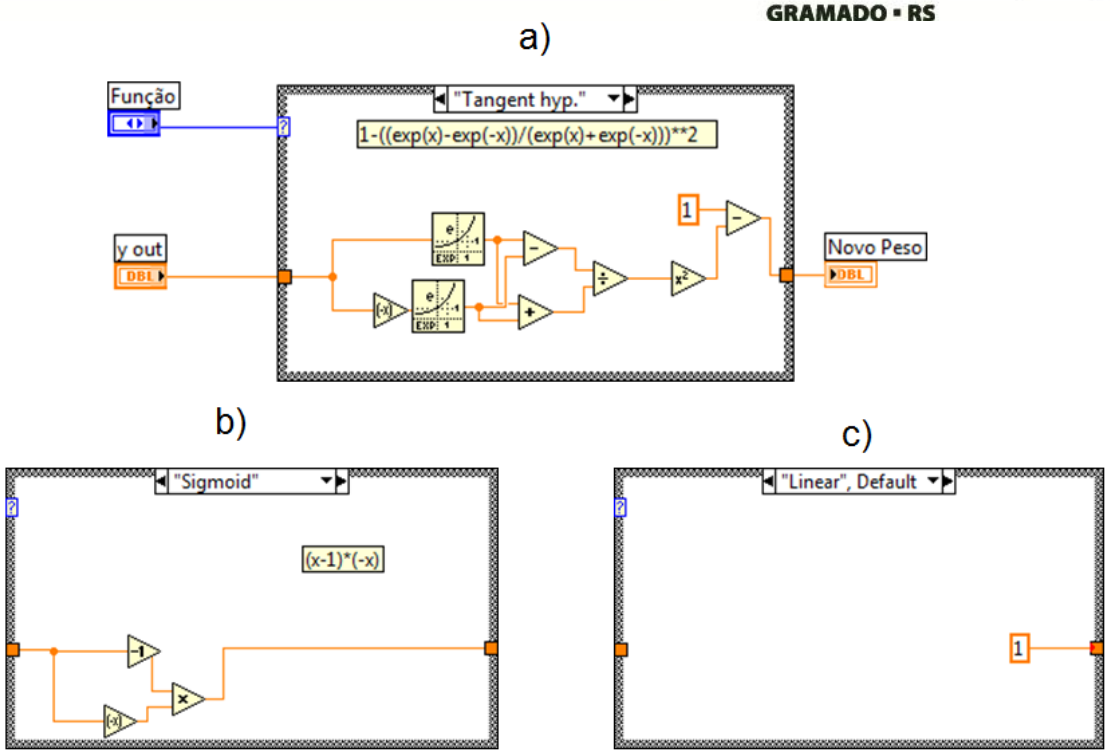

Figura 9 - Derivada da função de ativação: a) Derivada da função tangente hiperbólica; b) Derivada da função sigmoidal; c) Derivada da função linear
Fonte: elaborado pelo autor

Foram definidos os seguintes parâmetros a serem inseridos pelo usuário:

a) Os pares de entrada e saídas desejadas;
b) O número de camadas intermediárias;
c) O número de neurônios em cada camada;
d) A função de ativação em cada camada;
e) O algoritmo de treinamento (neste caso utilizaremos apenas a retropropagação);
f) A taxa de aprendizagem;
g) O número de iterações (épocas) a serem executadas ou o erro quadrado médio mínimo a ser atingido durante o treinamento.

O objetivo do design da interface depende do contexto -- quem são os usuários, o que eles pretendem fazer, e quais são seus objetivos (COOPER, REINMAN e CRONIN, 2007).

A interface com o usuário é apresentada a seguir na Figura 10:

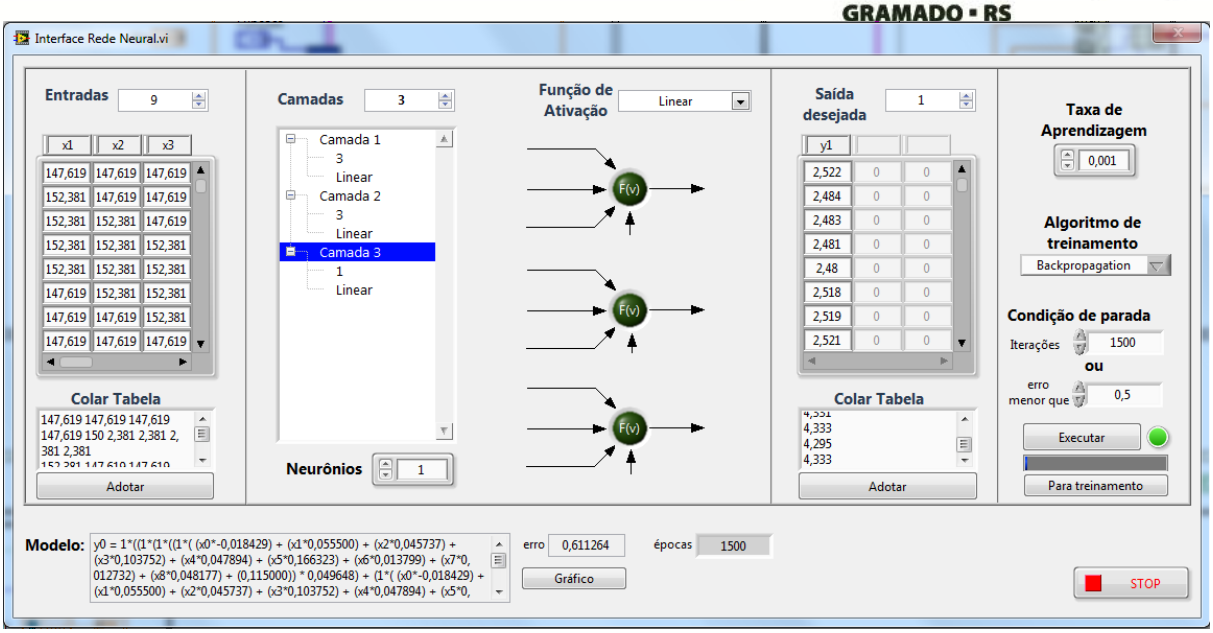

Figura 10 - Interface desenvolvida em LabVIEW®

6. TESTE DA FERRAMENTA

Para testar a funcionalidade da interface de treinamento utilizou-se um sistema de controle de velocidade em malha fechada representado pelo seguinte diagrama da Figura 11:

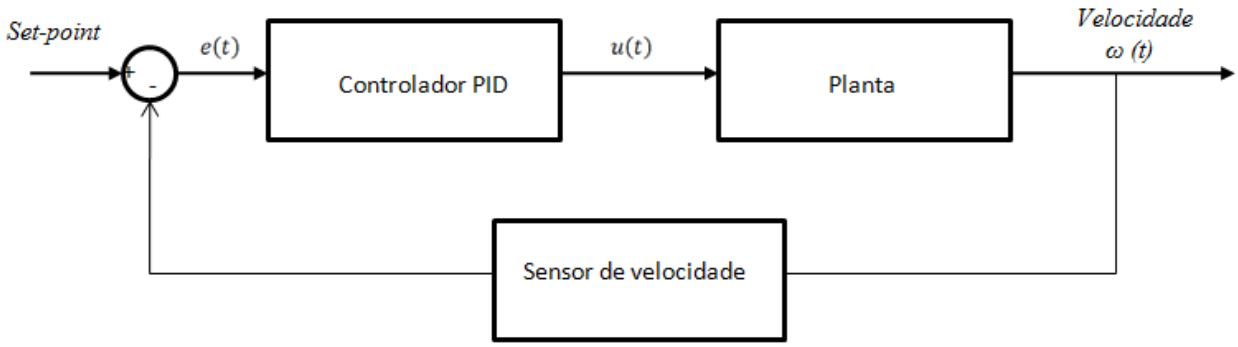

Figura 11 - Diagrama da malha de controle

O sistema foi montado utilizando um hardware USB para aquisição e geração dos sinais. Assim, obteve-se o controle de velocidade de uma ventoinha utilizando-se de um foto-transistor para a leitura da velocidade. Na Figura 12 é apresentada uma imagem do sistema físico:

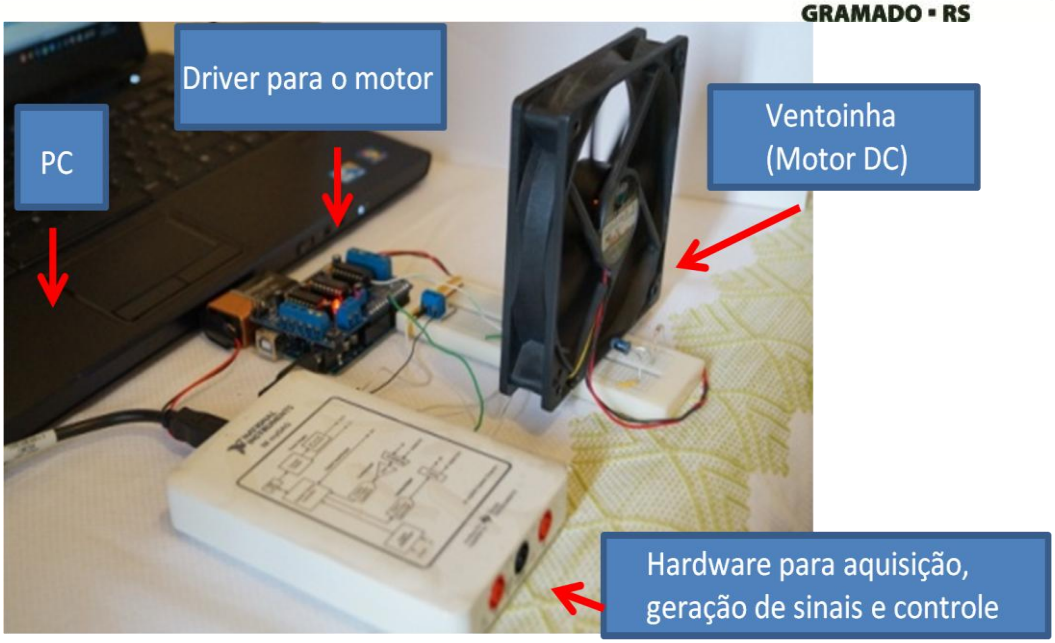

Figura 12 - Sistema para o controle de velocidade da ventoinha

A rede neural artificial foi treinada a partir dos dados de entrada e saída do controlador existente conforme representado pela figura 13:

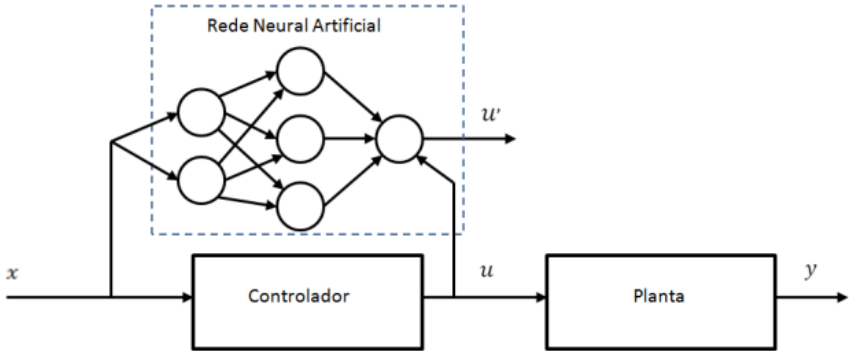

Figura 13 – Diagrama com a representação dos dados de entrada para a RNA a partir do controle existente no sistema

Após o treinamento da rede, substituiu-se o controle existente pelo modelo aprendido, conforme demonstrado na Figura 14:

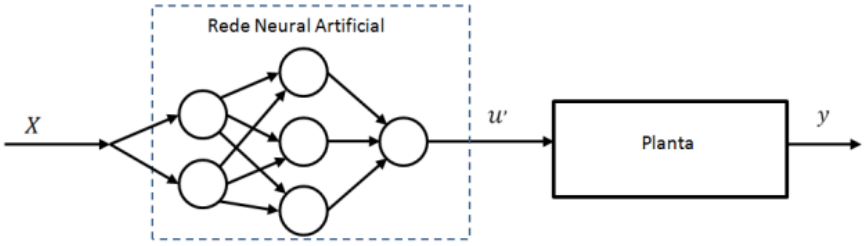

Figura 14 - Diagrama com o controle realizado pelo modelo neural

6.1. Parâmetros da rede

Para o treinamento foi utilizada a seguinte rede neural representada na Figura 15:

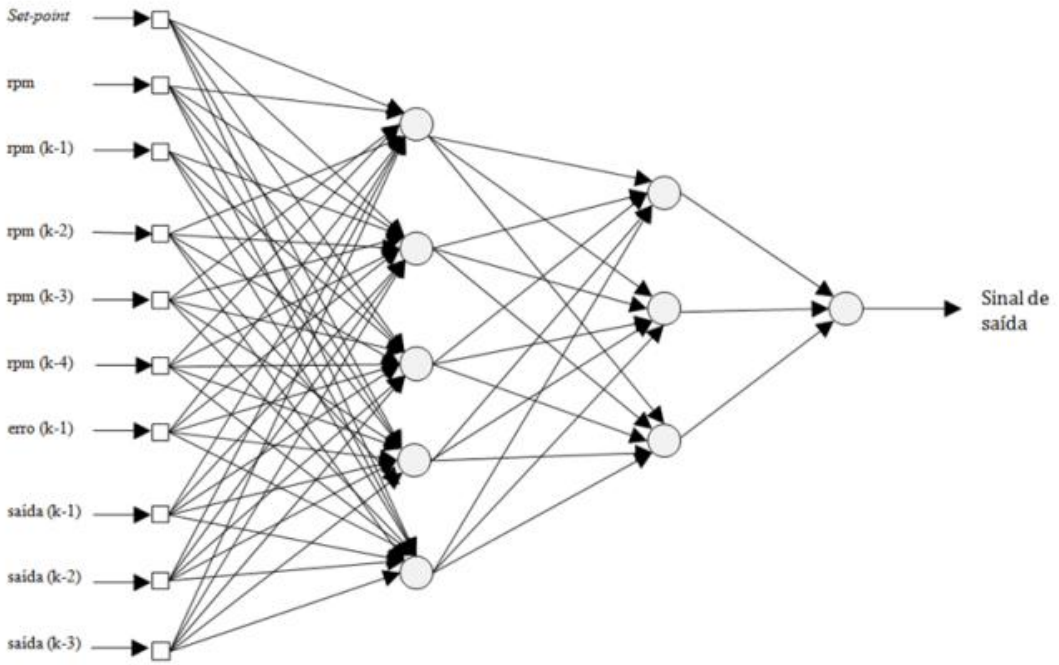

Figura 15 - Modelo de rede neural artificial utilizado para o treinamento

onde o número de camadas é igual a 3 com 5, 3 e 1 neurônio em cada uma respectivamente, as funções de ativação são todas lineares, a taxa de aprendizagem e o termo de momento é igual a 0,001. Definiu-se o máximo de 3500 iterações e erro menor que 0,4. Os valores escolhidos como entrada da rede a ser treinada foram (adaptado de CUI e SHIN, 1993):

Set- point;
rpm;
rpm(k-1);
rpm(k-2);
rpm(k-3);
rpm(k-4);
erro(k-1);
saída(k-1);
saída(k-2;
saída(k-3).

6.2. Resultados

A seguir nas Figuras 16 e 17 são apresentados os resultados do controle pré-existente (PID) e do controle após o treinamento e a substituição pelo modelo neural,

onde a amplitude do eixo y é dada em rotações por minuto (rpm) e o eixo x é o tempo decorrido em segundos. A linha vermelha corresponde ao set-point e a linha azul à resposta do sistema:

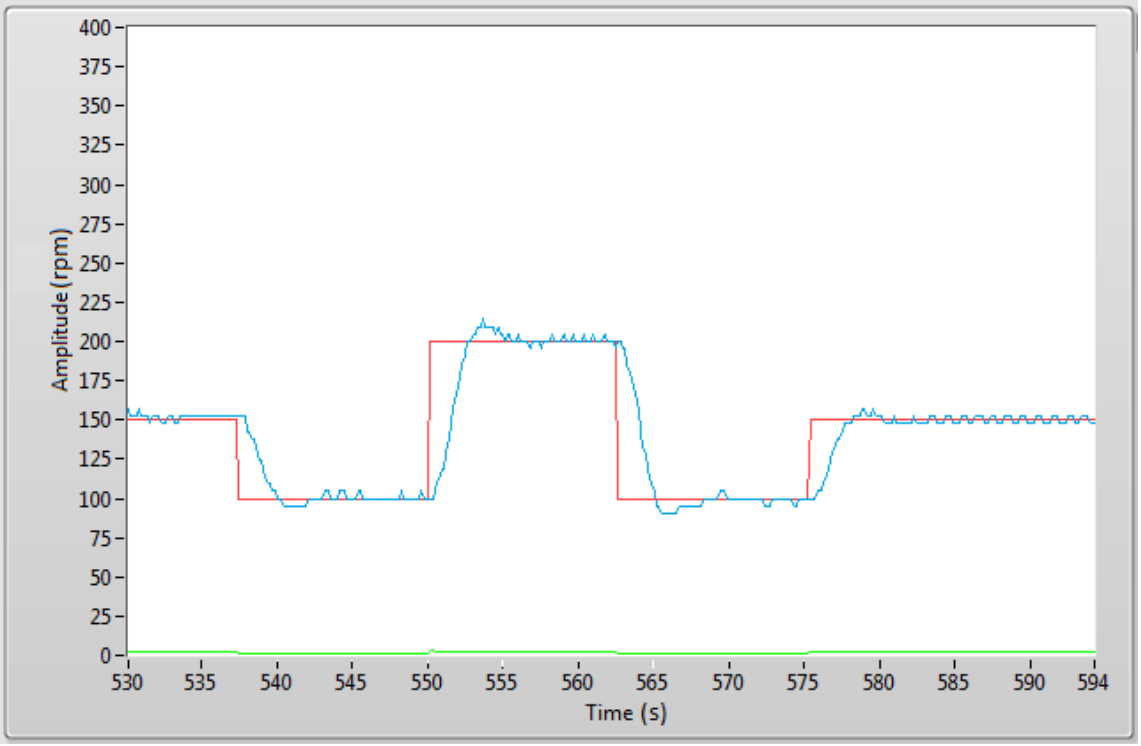

Figura 16 - Gráfico com o comportamento da velocidade com o controlador pré-existente

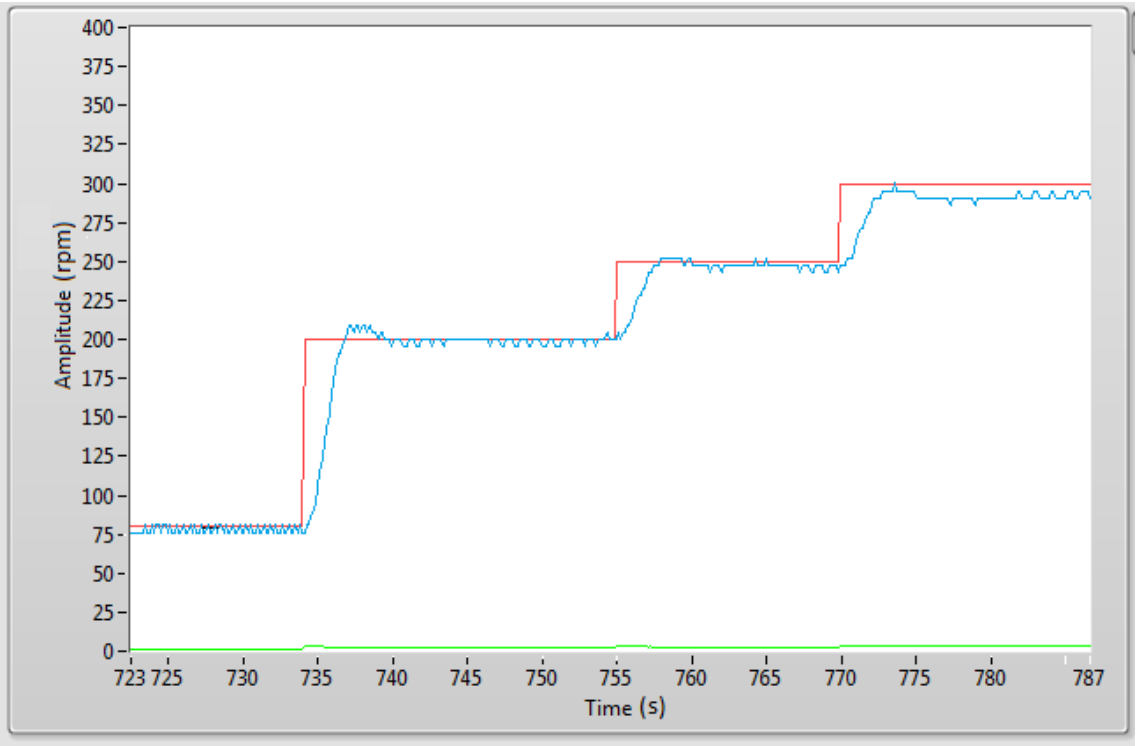

Figura 17 - Gráfico com o comportamento da velocidade após a inserção do modelo neural

Nota-se que o desempenho do controlador PID foi reproduzido pelo sistema construído pela ferramenta proposta.

Nos dois casos a resposta teve comportamento sub-amortecido com sobressinal equivalente. No entanto, os dados utilizados para treinamento não haviam contemplado o comportamento do sistema inicial na região dos 300 rpm. Por isso observa-se uma diferença no erro estacionário da Figura 17 próximo a este valor ao utilizar o modelo neural para controle do sistema.

7. CONCLUSÕES

Foi possível observar neste trabalho a utilização de uma ferramenta para configuração e treinamento de RNA's através de uma interface intuitiva desenvolvida em linguagem gráfica. Através dos principais parâmetros segundo a bibliografia estudada, foi possível configurar uma rede do tipo *perceptron* de múltiplas camadas com algoritmo de retropropagação. Também puderam ser testados os resultados do treinamento em um sistema de controle, substituindo o controle pré-existente por um modelo neural. Foi observada a importância da utilização do termo de momento que permitiu que o erro médio atingisse o desempenho desejado durante o treinamento.

A interface destacou-se por possuir apenas uma tela, apresentar os principais parâmetros e ainda permitir que o modelo final fosse diretamente extraído para a sua utilização em qualquer outro sistema.

Um modelo de interface padronizado demonstrou-se importante para o desenvolvimento de novas redes que podem ser utilizadas em diversas aplicações. Assim este trabalho propõe a utilização desta ferramenta para o treinamento inicial nas aplicações que necessitem da criação de uma rede, seja para identificação de modelos como para o controle de sistemas.

REFERÊNCIAS BIBLIOGRÁFICAS

CAMPOS, J. R. et al. **Implementação de Redes Nurais Artificiais Utilizando a linguagem de Programação JAVA**. Serra Negra: 9th Brazilian Conference on Dynamics, Control and their Applications, 2010.

CAMPOS, J. R. **Desenvolvimento de um sistema dinâmico para predição de cargas elétricas por redes neurais através do paradigma de programação orientada a objeto sob a linguagem JAVA**. Ilha Solteira: UNESP, 2010.

CANETE, J. F.; PEREZ, S. G.; SAZ, O. P. **Software Tools for System Identification and Control using Neural Networks in Process Engineering**. [S.l.]: World Academy of Science, Engineering and Technology, v. 47, 2008.

COOPER, A.; REINMAN, R.; CRONIN, D. **About Face 3:** The Essential of Interaction Design. Indianapolis: Wiley Publishing, 2007.

CUI, X.; SHIN, K. G. **Direct Control and Coordination using Neural Networks**. 3. ed. [S.l.]: IEEE Trans. Syst., Man, and Cybern., v. 23, 1993.

HAYKIN, S. **Redes Neurais:** princípios e práticas. Tradução de Paulo Martins Engel. 2ª Edição. ed. Porto Alegre: Bookman, 2001.

HETCH-NIELSON, R. **Neurocomputing**. [S.l.]: Addison-Wesley, 1990.

HUANG, Y. **Advances in Artificial Neural Networks - Methodoloical Development and Application**. Open Access - Algorithms, 2009. Disponível em: <www.mdpi.com/journal/algorithms>. Acesso em: Julho 2012.

KOWALSKA, T. O.; KAMINSKI, M. FPGA **Implementation of the Multilayer Neural Network for the Speed Estimation of the Two-Mass Drive System**. IEEE Transaction on Industrial Informatics, 2011.

LOPES, N.; RIBEIRO, B. **Hybrid learning multi neural architecture**. [S.l.]: INNS-IEEE International Joint Conference on Neural Networks (IJCNN 2001), v. 4, 2001.

LOPES, N.; RIBEIRO, B. **Stochastic GPU-based multithread implementation of multiple back-propagation**. [S.l.]: Second International Conference on Agents and Artificial Intelligence (ICAART 2010), 2010.

LOPES, N.; RIBEIRO, B. **An evaluation of multiple feed-forward networks on GPUs**. [S.l.]: International Journal of Neural Systems (IJNS), 2011.

NASCIMENTO, J. C. L.; YONEYAMA, T. **Inteligência Artificial em Controle e Automação**. São Paulo: Blucher, 2008.

PONCE, P.; GUTIERREZ, A. M. **LabVIEW for Intelligent Control Research and Education**. IEEE, 2010.

COMPUTATIONAL TOOL FOR RNA'S PROJECTS DEVELOPMENT USING GRAPHICAL PROGRAMING LANGUAGE

Abstract: *This paper presents a computational tool developed specifically to Artificial Neural Network (ANN) projects with back-propagation algorithm. This tools are built by a graphical based platform (LabVIEW), proposing an easy interpretation way to build ANN's. Also it is presented an intuitive way to build ANN using Multi Layer Perceptron and back-propagation. It's possible extract the results or model after the network training. The software engineering applied on this tools development allows to present the essential configurations for ANN's training that uses back-propagation algorithm. The generated models are tested for checking tools and evaluate its ease of use. These tools allow more understanding of the RNA's knowledge base algorithms.*

Key-words: *Graphical Programming Language, Artificial Neural Network, Software Engineering*