SOFTWARE PARA IMPLEMENTAÇÃO DE REDE NEURAL ARTIFICIAL - ESTUDO DE CASO: ESTIMAÇÃO DA VELOCIDADE MECÂNICA DO MOTOR DE INDUÇÃO

Cláudio Ferreira Carneiro*, Carlos Henrique Silva de Vasconcelos*

* *Centro Federal de Educação Tecnológica de Minas Gerais*
Campus - Leopoldina
Leopoldina, Minas Gerais, Brasil

Emails: claudiofcarneiro@hotmail.com, vasconcelos@cefetmg.br

Abstract— Induction motors are used in most variable speed drives, being most of these drive systems based on the direct measurement of the velocity; generating problems such as increased implementation cost and reduced robustness of the system. An alternative for these problems is in the use of neural estimators for the rotor speed. This paper presents the result of the development of a free and open-source software for training artificial neural networks, RNAs, with a friendly interface. One of the software characteristics is the possibility to export as a C fuction, in a file, the trained neural network. For it's validation, the software will be used in the development of a neural estimator for a three phase induction motor rotor speed. This study is based on the evaluation of two different neural models built with the software, subjjected to the same training conditions and different datasets.

Keywords— Artificial Neural Network, Induction Machines, Speed Estimation.

Resumo— Os motores de indução são utilizados em grande parte dos acionamentos de velocidade variável, sendo a maioria desses sistemas de acionamento baseados na medição direta da velocidade; gerando problemas como o aumento do custo de implementação e redução da robustez do sistema. Uma alternativa para tais problemas é a utilização de estimadores neurais para a velocidade do motor. Este trabalho apresenta o resultado do desenvolvimento de um *software* livre e de código aberto voltado ao treinamento de redes neurais artificiais, RNAs, por meio de uma interface amigável. Uma de suas características é a opção de exportar como uma função na linguagem C, em arquivo, a RNA treinada. Para sua validação, o *software* será utilizado no desenvolvimento de um estimador de velocidade mecânica do motor de indução trifásico, MIT. Este estudo baseia-se na avaliação de dois modelos neurais construídos com o *software*, submetidos às mesmas condições de treinamento mas com bases de dados distintas.

Palavras-chave— Rede Neural Artificial, Máquina de Indução, Estimação de velocidade.

1 Introdução

Este artigo visa contribuir com o processo de implementação de redes neurais artificiais (RNAs), apresentando um *software* livre e de código fonte aberto, cujo objetivo é simplificar o processo de desenvolvido, treinamento, validação e uso de RNAs. O *software* suporta o desenvolvimento de redes neurais de arquitetura do tipo *Multilayer Perceptron* (MLP), utilizada principalmente em aproximações de funções e problemas de classificação. O processo de treinamento ocorre de forma supervisionada *online* e com o uso do algoritmo *Backpropagation*. A partir de uma interface intuitiva o usuário escolhe a topologia da RNA, executa o treinamento e exporta o código fonte em C, de forma rápida e simples. O *software* possui ferramentas para o tratamento do banco de dados, como a normalização dos *datasets*. É oferecida ao usuário a opção de salvar o estado atual de treinamento, possibilitando possíveis alterações e paradas prematuras.

Com o uso do *software* em questão, um controle *sensorless* de velocidade do motor de indução trifásico (MIT) foi desenvolvido. Possibilitando de tal forma, validar o *software*, sua aplicabilidade em situações reais e sua usabilidade. Informações relativas ao uso de sistemas neurais no controle do MIT também foram obtidas com esse trabalho.

A grande utilização dos motores de indução trifásicos no setor industrial, comercial e residencial deve-se à sua simplicidade de construção e robustez. Com o advento dos inversores de frequência, as restrições de utilização em acionamentos de velocidade variável foram superadas (Bose, 2002). Além disso, a evolução dos processadores digitais de sinais (DSP) e o estabelecimento das técnicas de controle da velocidade proporcionam ao MIT um desempenho dinâmico superior aos acionamentos em corrente contínua (Leonhard, 2001).

As técnicas de controle do MIT exigem métodos eficientes e robustos para a determinação dos parâmetros da máquina, entre eles a velocidade mecânica do eixo. A obtenção da velocidade mecânica pode ser realizada através da medição direta ou através de técnicas de estimação. A medição direta da velocidade pode ser realizada por *encoders*, *resolvers* ou tacogeradores. A utilização desses dispositivos geram um aumento no custo do sistema de acionamento e ocasionam uma redução da robustez mecânica do sistema, visto que novos componentes e pontos de falhas serão adicionados (Nguyen et al., 2017). Além disso, esses sistemas são susceptíveis a interferências e falhas mecânicas (Vas, 1998).

As técnicas de controle de velocidade *sensorless*, ou seja, que não necessitam da medição direta da velocidade, podem ser realizadas com o

uso de estimadores em malha aberta através do monitoramento das correntes e tensões do estator, muitas vezes provenientes do modelo matemático do MIT. Observadores de Estado, que envolvem resoluções de equações diferenciais, sistemas de referência com modelos adaptativos e estimadores baseados em sistemas inteligentes (dos Santos et al., 2013) são alguns exemplos.

Técnicas de estimação que se originam do modelo matemático da máquina são altamente dependentes dde seus parâmetros físicos, podendo provocar uma imprecisão na estimação da velocidade (Abedi et al., 2013). Em contrapartida os estimadores baseados em sistemas inteligentes são independentes do conhecimento prévio dos parâmetros da máquina. Entre as técnicas inteligentes, podemos destacar principalmente as redes neurais artificiais (RNAs) (Niasar and Khoei, 2015). Nos últimos anos, vários métodos de estimação da velocidade mecânica utilizando RNAs, vêm sendo investigados e apresentando resultados promissores; como observado nos trabalhos de Niasar and Khoei (2015), dos Santos et al. (2013) e Pyne et al. (2014).

Aplicações de redes neurais artificias em sistemas de controle do motor de indução trifásico vêm ganhando espaço devido à sua capacidade de extrair padrões de sistemas não lineares. A partir de um conjunto de dados confiáveis é possível treinar a rede para reconhecer os padrões desejados. A obtenção do banco de dados pode ser realizada a partir da aquisição das variáveis primárias da máquina como tensões, correntes e velocidade. Para o processo de treinamento da rede, utiliza-se ferramentas computacionais para projetar, treinar e validar a rede.

2 Redes Neurais Artificiais

O princípio de funcionamento da RNA é baseado no modelo biológico do neurônio humano (figura 1), onde este pode ser visto como uma unidade de processamento que compõe o sistema nervoso. A associação desses neurônios entre si é denominada rede neural. A partir da excitação das conexões de entrada do neurônio, denominados de dendritos, esses sinais são processados pelo núcleo. Após o processamento um sinal é emitido através do axônio excitando outros neurônios por meio de sinapses (Lent, 2001).

2.1 Neurônio Artificial

O neurônio biológico mostrado na figura 1 pode ser modelado considerando um conjunto de entradas (dendritos) nas quais sinais externos, provenientes de outros neurônios ou não, são ponderados individualmente por um peso W_{ij}. Onde estes são somados e aplicados a uma função de ativação.

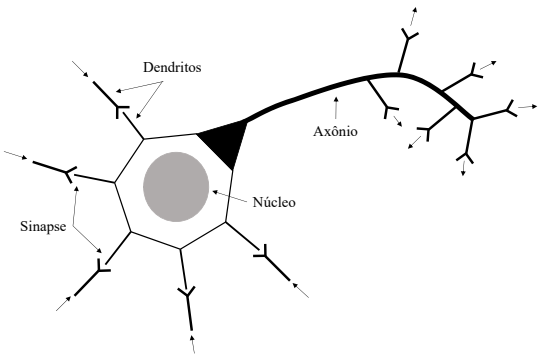

Figura 1: Neurônio biológico.

O modelo matemático do neurônio artificial mostrado na figura 1 é dado por:

$$Y_j(t) = bias + f(\sum_{i=1}^{I}(W_{ij} * X_i(t))) \qquad (1)$$

onde Y_j é o j-ésimo valor de saída do neurônio, $bias$ é o limiar associado a cada neurônio, $f()$ é a função de ativação, W_{ij} o peso sináptico associado a i-ésima entrada e ao j-ésimo neurônio e X_i é a i-ésima entrada do neurônio. A figura 2.1 apresenta uma representação do neurônio artificial citado.

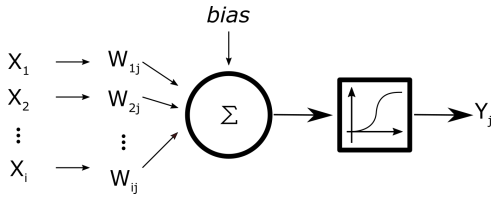

Figura 2: Neurônio artificial.

A escolha da função de ativação depende do contexto em que a RNA será aplicada. Dentre as funções de ativação típicas estão: Sigmoides (logística $f(x) = \frac{1}{(1+e^{-x})}$, tangente hiperbólica $f(x) = tanh(x)$), lineares e gaussianas; sendo que a utilização de funções de ativação não lineares possibilita a modelagem de sistemas não lineares (LeCun et al., 1998).

2.2 Arquitetura da RNA

A arquitetura de uma RNA é definida pela disposição de seus neurônios e pelo direcionamento das conexões sinápticas. Já a topologia define a estrutura da RNA, quantidade de neurônios por camadas e quais as funções de ativação utilizadas. Neste artigo será utilizada uma rede com arquitetura *Multilayer Perceptron* (MLP) não recorrente na qual o fluxo de informações entre os neurônios unidirecional; tal escolha de deve ao fato de que será construído um aproximador de funções, o estimador de velocidade. A figura 3 mostra a representação de uma rede MLP de topologia 3-3-1,

duas camadas internas (*hidden layers*) com funções de ativação sigmoidais e uma camada externa (*output layer*) com função de ativação linear.

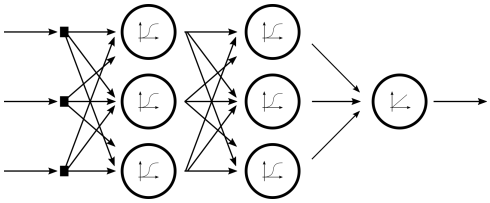

Figura 3: RNA de arquitetura MLP com funções de ativação sigmoidais nas camadas internas e linear na camada externa.

Em redes MLP os neurônios de uma mesma camada não comunicam entre si, recebem apenas informações da camada imediatamente anterior e passam informações para a camada imediatamente posterior (da Silva et al., 2010).

3 *Software* Desenvolvido

O *software* foi desenvolvido em Java, utilizando a plataforma JavaFX que oferece suporte à utilização de CSS (*Cascading Style Sheets*) em seus componentes e uma interface gráfica de boa qualidade. Sua estrutura utiliza o padrão MVC (*Model View Controller*) que facilita futuras alterações e manutenções no código fonte.

A interface gráfica foi construída utilizando a ferramenta RAD (*Rapid Application Development*) *Scene Builder*, que gera os arquivos de extensão .fxml utilizado na plataforma JavaFX. Janelas simples foram construídas diretamente no código fonte. O padrão de desenvolvimento observador-observável, ou *observer pattern*, foi utilizado no *background* da interface gráfica do aplicativo. Neste padrão, os objetos interessados nas mudanças, *observers*, registram-se no objeto observável, *subject*, e passam a ser notificados quando mudanças no *subject*, ou *observable* como é referido na API do Java, ocorrem.

O *software* oferece uma interface limpa e amigável, sendo o processo de implementação de RNAs dividido em seis etapas: Configuração, carregamento de dados, pré-processamento, treinamento, validação e exportação da rede. O fluxograma do *software* é mostrado na figura 4.

A figura 5 mostra a tela principal do *software*. Nesta janela o usuário acessa o menu de configurações, dados de treino, dados de teste, exportação da rede e os comandos para o treinamento. Além disso, o usuário pode visualizar o processo de treinamento e o comportamento do erro da RNA através de uma interface gráfica.

3.1 Configuração da Rede

Na janela de configurações, figura 6, são definidos parâmetros como: topologia da rede, número

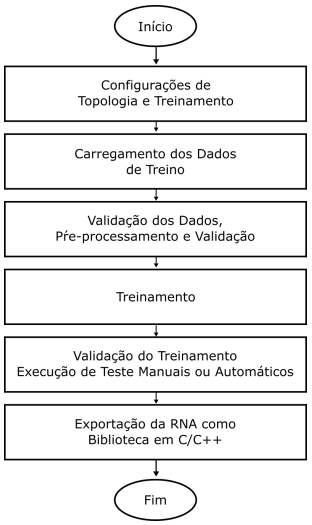

Figura 4: Fluxograma do *software*

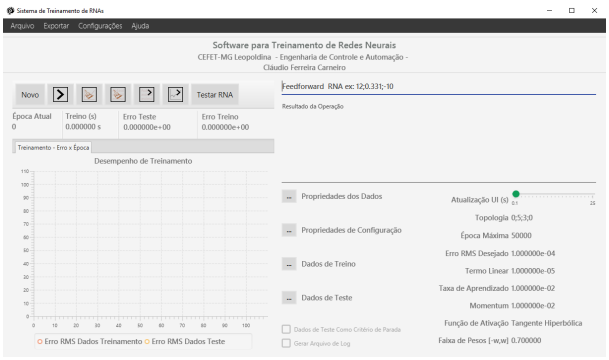

Figura 5: Tela principal

de épocas máxima, erro médio desejado, taxa de aprendizagem, valor do termo de *momentum*, termo linear, pesos iniciais e o tipo de função de ativação utilizado nas camadas internas e externas. Além disso, é possível determinar o intervalo em que os dados de treinamento serão normalizados e se arquivos de *log* serão gerados no treinamento.

3.2 Carregamento e Pré-Processamento dos Dados

A inserção dos *datasets* (conjuntos de dados) de treinamento e validação da RNA ocorre na tela principal. O *software* aceita três formatos de arquivos: '.csv' (*Comma-separated values*), '.out' ou '.txt'. Nesta janela são informadas quais colunas dos *datasets* são referentes aos dados de entrada e de saída.

As ações de pré-processamento de dados são realizadas conforme configurações do usuário. O *software* executa automaticamente a normalização dos *datasets*, facilitando o processamento de informações com diferentes ordens de grandeza, e realiza o embaralhamento das amostras contidas em tais conjuntos.

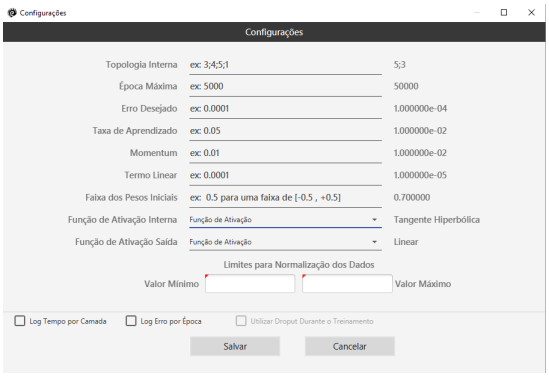

Figura 6: Tela de Configurações

3.3 Treinamento e Validação

Após a realização das etapas anteriores, inicia-se o processo de treinamento da rede. O usuário pode optar por utilizar o conjunto de testes já no momento do treinamento. Isso permite associar critérios de parada, como o erro médio, ao conjunto de teste. Sendo o erro médio gerado a partir de todas as saídas da RNA, e este exibido no gráfico "Erro Médio x Época". Este gráfico é mostrado na figura 7.

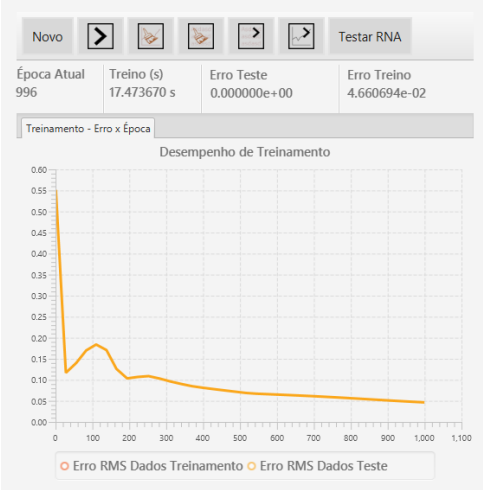

Figura 7: Gráfico: "Erro Médio x Época"

A partir do momento em que o processo de treinamento atingir um dos critérios de parada, a tela de treinamento disponibiliza uma ferramenta para a validação da RNA (Testar RNA). Utilizando o conjunto de teste previamente carregado, o *software* exibe um gráfico de dispersão do treinamento juntamente com as informações relevantes. É possível gerar um novo subconjunto de dados utilizando uma faixa de erro médio para testar a rede. A janela de teste é mostrada na figura 8.

3.4 Exportação da RNA

Após o treinamento e validação da RNA, esta pode ser exportada em forma de uma função em linguagem *C*. Esta função pode ser utilizada em aplicativos de terceiros compatíveis com esta linguagem. O comando para a exportação da rede se encontra no canto superior da tela principal (figura 5) no menu "Exportar". A figura 9 mostra a janela de exportação da rede.

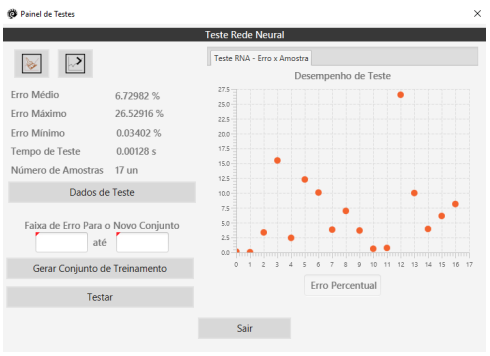

Figura 8: Janela de Testes

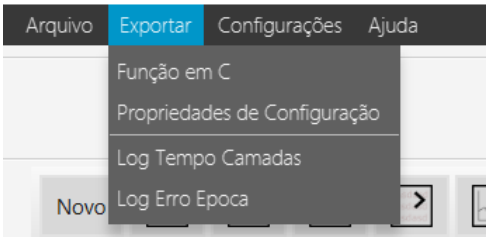

Figura 9: Janela de exportação

4 Modelo da Máquina de Indução e do Sistema de Controle

O primeiro passo para a implementação do estimador neural da velocidade mecânica da máquina, consiste em levantar um conjunto de dados que relacionam as entradas e saídas do sistema. Estes dados devem assegurar que a estrutura neural seja exposta a padrões que representam o comportamento do sistema analisado. De posse do banco de dados inicia-se o processo de treinamento que consiste em ajustar os pesos sinápticos da RNA. Para gerar o banco de dados de treinamento foi simulado uma máquina de indução acionada através da técnica de controle vetorial indireto orientado segundo o fluxo do rotor.

4.1 Modelo da Máquina de Indução

Considerando a densidade de fluxo magnético senoidal, enrolamentos trifásicos simétricos e desconsiderando a saturação magnética e os efeitos da temperatura, o modelo matemático da máquina de indução pode ser considerado linear. O modelo da máquina de indução num referencial genérico (sobrescrito "g") pode ser representado por:

$$\overline{u}_s^g = R_s \bar{i}_s^g + \frac{d\overline{\psi}_s^g}{dt} + j\omega^g \overline{\psi}_s^g \quad (2)$$

$$\overline{u}_r^g = R_r \bar{i}_r^g + \frac{d\overline{\psi}_r^g}{dt} + j(\omega^g - \omega_r)\overline{\psi}_r^g \quad (3)$$

$$\overline{\psi}_s^g = L_s \bar{i}_s^g + M \bar{i}_r^g \quad (4)$$

$$\overline{\psi}_r^g = M \bar{i}_s^g + L_r \bar{i}_r^g. \quad (5)$$

A equação mecânica da máquina pode ser representada por:

$$T(t) - T_L(t) = J\frac{d\omega_m}{dt} + B\omega_m. \quad (6)$$

O conjugado eletromagnético é dado por:

$$m = -\frac{3}{2}\frac{P}{2} M \left(\bar{i}_s^g \times \bar{i}_r^g\right) \quad (7)$$

em que os subscritos s e r representam o estator e o rotor respectivamente, u a tensão, i a corrente, ψ o fluxo, P o número de polos, ω_m a velocidade mecânica, T e T_L o conjugado eletromagnético e o de carga, B o coeficiente de atrito, J o momento de inércia e R, L e M a resistência, indutância própria e a indutância mútua.

4.2 Controle Vetorial Indireto

O controle vetorial indireto é definido a partir das equações da máquina que relacionam o vetor do fluxo do rotor e as correntes de estator. Esta relação, no referencial do fluxo do rotor, é dada por:

$$\psi_{rd}^r = \psi_r \quad e \quad \psi_{rq}^r = 0 \quad (8)$$

$$\frac{M}{\tau_r} i_{sd}^r = \frac{1}{\tau_r}\psi_r + \frac{d\psi_r}{dt} \quad (9)$$

$$0 = \omega sl - \frac{1}{\tau_r}\frac{i_{sq}r}{i_{sd}r} \quad (10)$$

$$T = P\frac{M}{L_r} i_{sq} r \psi_r \quad (11)$$

onde $\omega sl = \omega_e - \omega_r$ é o escorregamento, ω_e é a velocidade síncrona e ω_r é a velocidade elétrica do rotor.

5 Estudo de Caso: Estimação da Velocidade

As simulações foram realizadas utilizando sistema composto, basicamente, de uma máquina de indução classe B de 5 HP/460 V/6.4 A/1750 RPM, uma carga linear ($T_L = 1,47 + 0,1\omega_m$) e um inversor de frequência trifásico a IGBT. Os sinais de excitação do estator são sintetizados através de SVPWM (modulação por largura de pulso vetorial) com frequência de chaveamento de 8 kHz e 800 Vdc no elo CC. Utilizou-se o controle vetorial indireto orientado segundo o fluxo do rotor para o controle da velocidade. A figura 10 mostra o diagrama em blocos do sistema.

O estimador proposto neste trabalho tem como entrada as variáveis de tensão do estator de eixo direto (u_{sd}) e as correntes de eixo direto e em quadratura do estator (i_{sd} e i_{sq}). A arquitetura e os parâmetros da RNA são mostrados na tabela 1.

Tabela 1: **RNA: arquitetura e parâmetros**

Arquitetura	Perceptron multicamadas
Treinamento	Supervisionado on-line
Algoritmo de treinamento	*Backpropagation*
Topologia	4-5-1
Entradas	3 (u_{sd}, i_{sd} e i_{sq})
Função ativação camada escondida	Tangente hiperbólica
Função ativação camada de saída	Linear
Taxa de aprendizagem	0,025
Épocas máxima	20.000
Erro	0,00001

Neste trabalho são geradas duas RNAs, onde a primeira (caso 01) utiliza um banco de dados que considera apenas a partida e o regime permanente da máquina. Já a segunda RNA (caso 02) após a máquina chegar em regime permanente é introduzida uma redução de velocidade de 60% no valor da referência. Para gerar os bancos de dados em questão, a referência de velocidade do controle vetorial foi ajustada para: 10, 50, 90, 110, 160, 182, -40 e -120 rad/s. A figura 11 mostra a curva de velocidade de 110 rad/s para o caso 01 e a de 90 rad/s do caso 02.

5.1 Caso 01

Na primeira simulação a máquina é acionada a 40 rad/s e durante o intervalo entre 1 e 2 segundos a referência de velocidade e ajustada para 130 rad/s. Nesta simulação a RNA é apenas utilizada para estimar a velocidade (sem realimentação do valor estimado no controle). A figura 12 mostra a estimação da velocidade, observa-se que em regime permanente a RNA estimou a velocidade com precisão. Nota-se também que durante a partida a velocidade estimada acompanha a velocidade mecânica. O mesmo não ocorre durante as variações de velocidade, principalmente em reduções de velocidade.

A figura 13 mostra o comportamento da máquina sendo acionado pelo controle vetorial utilizando a realimentação do valor de velocidade estimada pela RNA. Observa-se que a velocidade estimada pela RNA rastreia a velocidade mecânica e apenas na redução de velocidade o estimador diverge do valor real.

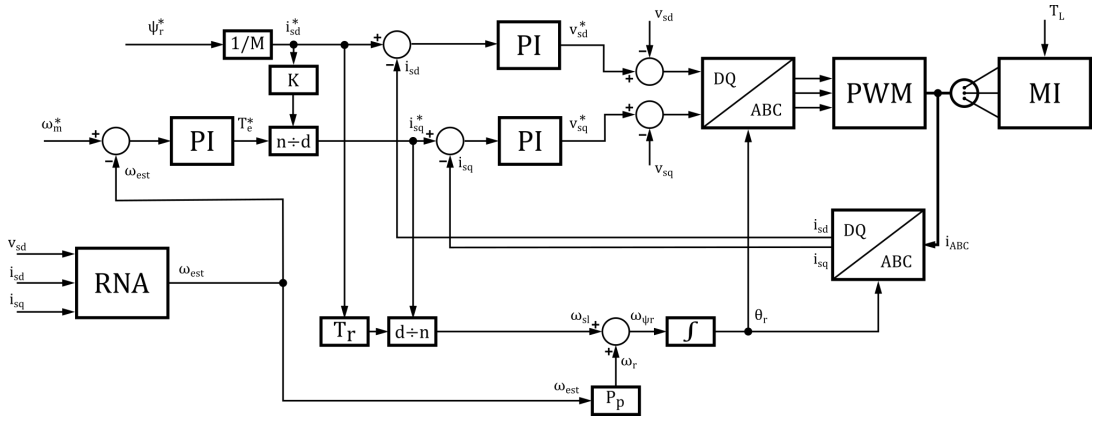

Figura 10: Estudo de caso: diagrama em blocos do sistema de acionamento.

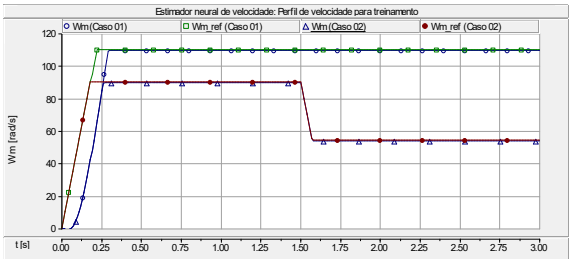

Figura 11: RNA: exemplo da curva do treinamento do caso 01 (110 rad/s) e caso 02 (90 rad/s).

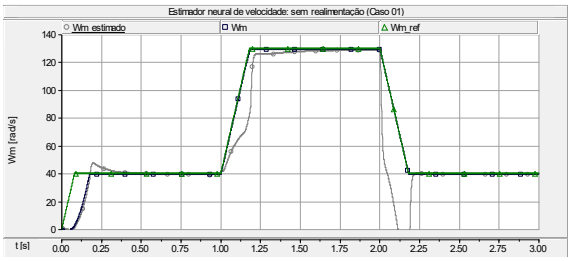

Figura 12: Caso 01: variação de velocidade (sem realimentação).

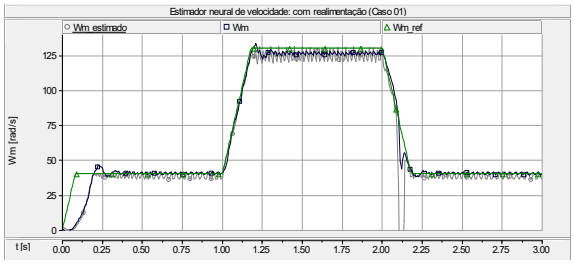

Figura 13: Caso 01: variação de velocidade (com realimentação).

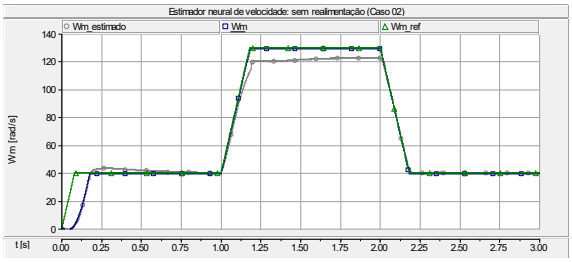

Figura 14: Caso 02: variação de velocidade (sem realimentação).

5.2 Caso 02

Para esta simulação a máquina é acionada nas mesmas condições anteriores do caso 01. A figura 14 mostra a estimação da velocidade; observa-se que o valor estimado acompanha a velocidade mecânica, divergindo apenas em regime permanente para a velocidade de 130 rad/s. Nota-se também que durante as acelerações e na redução de velocidade a RNA reproduz a velocidade mecânica.

A figura 15 mostra o comportamento da máquina sendo acionada pelo sistema de controle com realimentação da velocidade pela RNA. Observa-se que a velocidade estimada pela RNA rastreia velocidade mecânica. Neste caso durante a desaceleração da máquina o valor estimado da RNA não diverge da velocidade mecânica. A inclusão de desacelerações no processo de treinamento, quando comparado com o caso 01, apresenta uma melhora na estimação da RNA.

Para comprovar o comportamento da RNA perante variações de velocidade, a máquina é acionada a 40 rad/s e no intervalo de 1 a 2 segundos a referência de velocidade é ajustada para -40 rad/s, introduzindo uma reversão no sentido de rotação da máquina. A figura 16 mostra a estimação da velocidade durante a reversão de velocidade, sem a realimentação do controle pela RNA. Observa-se que o valor estimado da RNA acompanha a velocidade mecânica da máquina durante as reversões de velocidade. Na figura 17, a máquina é acionada a -40 rad/s e no intervalo de 1 a 2 segundos a referência de velocidade é ajustada para 60 rad/s utilizando a realimentação do valor de velocidade da RNA. Observa-se que a RNA consegue rastrear a velocidade independente do sentido de giro e durante as reversões de velocidade.

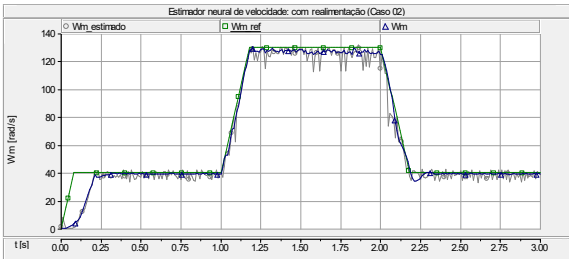

Figura 15: Caso 02: variação de velocidade (com realimentação).

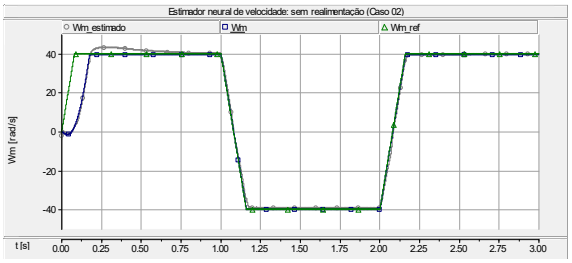

Figura 16: Caso 02: reversão de velocidade (sem realimentação).

6 Conclusões

Este trabalho apresentou um estudo de caso envolvendo RNA para a estimação de velocidade da máquina de indução. Além disso, foi apresentado o *software* desenvolvido para treinamento das RNAs com arquitetura MLP.

O *software* desenvolvido provou ser uma ferramenta útil para o treinamento de redes neurais de arquitetura MLP. Entre suas características pôde-se destacar sua a interface, usabilidade e a ferramenta de exportação da RNA em linguagem C.

Comparando os dois casos analisados, ambos treinados sobre as mesmas condições, observou-se que com a inclusão da desaceleração nos dados de treinamento no caso 02 proporcionou uma melhora no rastreamento da velocidade quando comparado com o caso 01. Observou-se que com a realimentação da velocidade estimada no sistema de controle, introduziu-se perturbações na velocidade da máquina, podendo causar instabilidade no sistema de controle.

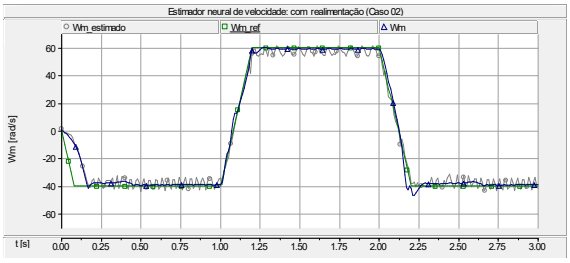

Figura 17: Caso 02: partida em sentido oposto e reversão de velocidade (com realimentação).

Agradecimentos

Os autores agradecem o apoio dado pelo CEFET-MG e pela FAPEMIG para o desenvolvimento deste trabalho.

Referências

Abedi, S., Buyamin, S., Tousizadeh, M. and Rahim, N. A. (2013). Sensorless speed estimation of induction motor based on feed-forward neural network algorithm, *2013 IEEE Conference on Clean Energy and Technology (CEAT)*, pp. 71–75.

Bose, B. K. (2002). *Modern Power Eletronics*, Prentice Hall.

da Silva, I. N., Spatti, D. H. and Flauzino, R. A. (2010). *Redes Neurais Artificiais para Engenharia e Ciências Aplicadas*, ArtLiber.

dos Santos, T. H., Goedtel, A., da Silva, S. A. O., Graciola, C. L. and Junior, P. B. (2013). Uma abordagem neural aplicada no controle escalar do motor de indução trifásico, *Simpósio Brasileiro de Automação Inteligente* .

LeCun, Y., Bottou, L., Orr, G. B. and Muller, K. R. (1998). *Neural Networks: Tricks of The Trade*, Springer Berlin Heidelberg.

Lent, R. (2001). *Cem Bilhões de Neurônios*, Atheneu.

Leonhard, W. (2001). *Control of Eletrical Drives*, Springer-verlag.

Nguyen, S. T., Pham, P. H., Pham, T. V., Ha, H. X., Nguyen, C. T. and Do, P. C. (2017). A sensorless three-phase induction motor drive using indirect field oriented control and artificial neural network, *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1454–1459.

Niasar, A. H. and Khoei, H. R. (2015). Sensorless direct power control of induction motor drive using artificial neural network, *Hindawi Publishing Corporation* .

Pyne, M., Chatterjee, A. and Dasgupta, S. (2014). Speed estimation of three phase induction motor using artificial neural network, *International Journal of Energy and Power Engineering* .

Vas, P. (1998). *Sensorless Vector and Direct Torque Control*, Oxford University Press.