UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Controle Vetorial de Velocidade de um Motor de Indução Trifásico com Estimação Neural de Fluxo

Francisco Canindé Holanda de Queiroz

Natal - RN, março de 2008

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Controle Vetorial de Velocidade de um Motor de Indução Trifásico com Estimação Neural de Fluxo

Francisco Canindé Holanda de Queiroz

Orientador: **Andres Ortiz Salazar**
Co-orientador: **André Laurindo Maitelli**

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área Automação e Sistemas), como parte dos requisitos necessários para a obtenção do título de Mestre.

Natal - RN, março de 2008

Controle Vetorial de Velocidade de um Motor de Indução Trifásico com Estimação Neural de Fluxo

Francisco Canindé Holanda de Queiroz

Dissertação aprovada em 17 de março de 2008 pela banca examinadora composta pelos seguintes membros:

Prof. **Andres Ortiz Salazar**, D. Sc. (Orientador) ..... (UFRN)

Prof. **André Laurindo Maitelli**, D. Sc. (Co-orientador) ..... (UFRN)

Prof. **Ricardo Ferreira Pinheiro**, D. Sc. ..... (UFRN)

Profa. **Jossana Maria de Souza Ferreira**, D. Sc. ..... (CEFET/BA)

Natal - RN, março de 2008

Agradecimentos

A Deus, por tudo.

À minha esposa e filhos, pela compreensão.

À Escola Técnica Federal de Palmas e à Universidade Federal do Rio Grande do Norte.

Ao meu Orientador, Prof. Dr. Andrés Ortiz Salazar, e Co-orientador, Prof. Dr. André Laurindo Maitelli.

Ao Prof. Dr. José Álvaro de Paiva do CEFET-RN, pela grande contribuição, principalmente na parte de programação e implementação.

Aos meus colegas André Macedo Santana, pelas valiosas informações, aos bolsistas Leonardo Fava Souza e Rafael Ferreira Lira e ao Técnico de Laboratório Jefferson Doolan Fernandes, pela colaboração.

À CAPES, pelo apoio financeiro, e ao PPgEE.

Enfim, a todos àqueles que de forma direta ou indireta contribuíram na elaboração e conclusão deste trabalho.

Aos meus pais, Raimundo e Maria, e meus irmãos.

À minha esposa Mônica e meus filhos Eduardo e Letícia.

Sumário

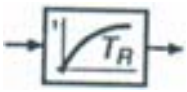
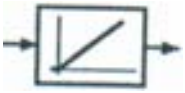
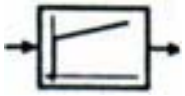
Lista de Figuras

Lista de Símbolos

P_{nom}	Potência Nominal.
ω_{nom}	Velocidade Nominal.
V_{nom}	Tensão Nominal.
I_{nom}	Corrente Nominal.
R_S	Resistência de estator por fase.
R_R	Resistência de rotor por fase.
L_{dS}	Indutância de dispersão do estator por fase.
L_{dR}	Indutância de dispersão do rotor por fase.
L_m	Indutância de magnetização.
L_S	Indutância do estator por fase.
L_R	Indutância do rotor por fase.
T_R	Constante de tempo do rotor.
n_p	Número de pares de pólos.
J	Momento de inércia do rotor.
σ	Fator de dispersão.
k	Índice de neurônios ou de instantes de amostragem.
D	Fator de carga.
t	Tempo em segundos.
j	Operador complexo.
h	Intervalo de amostragem.
$\underline{i}_S$	Vetor de corrente de estator.
i_{S1}, i_{S2} e i_{S3}	Valores instantâneos das correntes trifásicas do estator.

i_{Sa} e i_{Sb}	Valores instantâneos das correntes bifásicas em coordenadas do estator.
i_{Sd} e i_{Sq}	Valores instantâneos das correntes de campo e de torque em coordenadas de campo do rotor.
i_{mR} / $\hat{i}_{mR}$	Valores instantâneos das correntes de magnetização real/estimada.
i_{S1ref}, i_{S2ref} e i_{S3ref}	Valores de referência das correntes trifásicas de estator.
i_{Saref} e i_{Sbref}	Valores de referência das correntes bifásicas em coordenadas de estator.
i_{Sdref} e i_{Sqref}	Valores de referência das correntes de campo e de torque em coordenadas de campo do rotor.
$i_{m\,Ref}$	Valor de referência da corrente de magnetização.
m_M	Valor instantâneo do torque elétrico.
m_L	Valor instantâneo do torque de carga.
m_{Mref}	Valor de referência do torque elétrico.
ω_{ref}	Velocidade mecânica de referência em rpm.
ω_{mec}	Valor instantâneo da velocidade mecânica em rpm.
ω_{mR}	Velocidade angular do fluxo do rotor.
$velorads$	Valor instantâneo da velocidade mecânica em rad/s.
$\rho / \hat{\rho}$	Posição angular real/estimada do fluxo do rotor.
ε	Posição angular instantânea do eixo do rotor.
δ	Posição angular instantânea do vetor de corrente de estator.
γ	Deslocamento angular de 120º.
$\frac{d}{dt}$	Operador de derivação de uma função ou variável.
$\cos\rho$	Função cosseno do ângulo ρ.
$sen\rho$	Função seno de um ângulo ρ.
$\arctan\rho$	Função arcotangente de um ângulo ρ.
$e^{(x)}$	Função exponencial de uma variável x genérica.
$sig(x)$	Função sigmóide de uma variável x genérica.
p	Número de entradas das redes implementadas.
$x_1, x_2, x_3, ..., x_p$	Valores instantâneos das p entradas das redes implementadas.

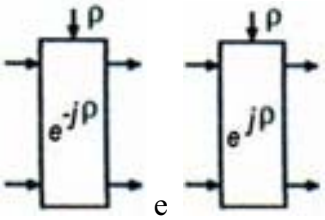
w_{kp}	Valores dos pesos sinápticos da $p-$ésima entrada do $k-$ésimo neurônio.
b_k	Valor do peso *bias* do $k-$ésimo neurônio.
$\sum$	Operador de Somatório na junção aditiva do $k-$ésimo neurônio.
v_k	Potencial de ativação do $k-$ésimo neurônio.
u_k	Saída do combinador linear do $k-$ésimo neurônio.
$\varphi(.)$	Função de ativação das saídas dos neurônios.
y_k	Valor da saída do $k-$ésimo neurônio.
$\hat{y}$	Valor instantâneo de uma saída estimada genérica.
erro	Erro instantâneo entre duas variáveis.
$P(k)$	Ação de controle proporcional.
$I(k)$	Ação de controle integrativa.
K_P	Valor do ganho proporcional.
K_I	Valor do ganho integrativo.
$u(k)$	Ação de controle genérica.
$I_a,\ I_b\ e\ I_c$	Correntes trifásicas de referência por fase.
$I_1,\ I_2,\ I_3,\ I_4,\ I_5\ e\ I_6$	Correntes individuais de cada bobina do estator.
$\bar{i}_a, \bar{i}_b$ e $\bar{i}_c$	Média das correntes de fase.
Q_i	Representação geral de números inteiros de i bits.
i	Número de bits.
N	Número real a ser transformado para o formato Q_i.
N_i	Número inteiro transformado para o formato Q_i.
$d\hat{i}_{mR}$	Derivada da corrente de magnetização após a discretização.
$d\hat{\rho}$	Velocidade angular estimada do fluxo após a discretização.
$d\omega_{mec}$	Velocidade angular estimada do fluxo após a discretização.
K_r	Constante paramétrica para a corrente de magnetização.
K_i	Constante paramétrica para a velocidade de escorregamento.
K_w	Constante paramétrica para o incremento angular do vetor de fluxo do rotor.

K_M	Constante paramétrica para o torque elétrico.
K_j	Constante paramétrica para a velocidade mecânica.
$x_{norm}(k)$	Valor normalizado de variável x genérica.
$y_{desnorm}(k)$	Valor desnormalizado de variável y genérica.
$x_{\max}$	Valor máximo de uma variável x genérica.
$x_{\min}$	Valor mínimo de uma variável x genérica.
>>	Deslocamento de bits à direita.
<<	Deslocamento de bits à esquerda.
	Bloco de controlador em atraso.
	Bloco de integração numérica.
	Bloco de cálculo de senos e cossenos.
	Bloco de controlador PI.
	Ponto de entrada das redes implementadas.
	Neurônio.
	Vetores de entrada e saída, conexões entre blocos ou conexão entre neurônios.
	Ponto de conexão.
	Bloco soma de variáveis.
	Ponto de soma de duas variáveis.
	Bloco multiplicador de uma variável por n.
	Ponto de divisão de duas variáveis.
	Bloco de multiplicação de variáveis.

Ponto de multiplicação de duas variáveis.

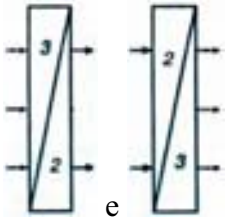
Conversor de sinal Analógico para Digital.

Conversor de sinal Digital para Analógico.



Blocos das transformações de Park direta e inversa.

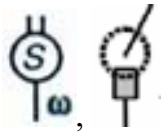
e



Bloco de transformação de variáveis trifásicas para variáveis bifásicas e bloco de transformação de variáveis bifásicas para variáveis trifásicas.

e

Sensor de Corrente.



Sensor de velocidade.

,

Resumo

Este trabalho descreve o estudo e a implementação de um controle vetorial de velocidade para um motor de indução trifásico de 1.1 kW / 4 pólos utilizando estimação neural de fluxo do rotor. O controle vetorial de velocidade opera em conjunto com o controle das correntes nos enrolamentos de cada fase do estator. A estimação neural de fluxo aplicada ao controle vetorial de velocidade tem como objetivo compensar a dependência dos estimadores convencionais em relação às variações nos parâmetros da máquina devido a aumentos de temperatura ou saturação magnética do rotor. O sistema de controle implementado possibilita uma comparação direta dos respectivos desempenhos de velocidade sob orientação do estimador neural em relação ao estimador convencional de fluxo. Todo o controle do sistema é realizado por um programa desenvolvido em linguagem padrão ANSI C. Os principais recursos do DSP utilizados pelo sistema são, respectivamente, os canais de conversão A/D, as saídas PWM e as interfaces paralela e serial RS-232, as quais são responsáveis, respectivamente, pela programação do DSP e a captura de dados através de um sistema de supervisão.

Palavras-chave: Controle vetorial, máquina de indução, estimação neural, DSP.

Abstract

This work describes the study and the implementation of the speed control for a three-phase induction motor of 1,1 kW and 4 poles using the neural rotor flux estimation. The vector speed control operates together with the winding currents controller of the stator phasis. The neural flux estimation applied to the vector speed controls has the objective of compensating the parameter dependences of the conventional estimators in relation to the parameter machine's variations due to the temperature increases or due to the rotor magnetic saturation. The implemented control system allows a direct comparison between the respective responses of the speed controls to the machine oriented by the neural rotor flux estimator in relation to the conventional flux estimator. All the system control is executed by a program developed in the ANSI C language. The main DSP recources used by the system are, respectively, the Analog/Digital channels converters, the PWM outputs and the parallel and RS-232 serial interfaces, which are responsible, respectively, by the DSP programming and the data capture through the supervisory system.

Keywords: Vector control, induction machine, neural estimation, DSP.

Capítulo 1
Introdução

1.1 Introdução

No passado, os motores de corrente contínua eram usados extensivamente em áreas onde era necessário que operassem com velocidades variáveis. Entretanto, algumas de suas desvantagens, como a existência do comutador e das escovas, implicam em uma manutenção periódica. A superação desses problemas veio com o uso dos motores de indução, por eles apresentarem estrutura mais simples e manutenção mais fácil e econômica, além de serem mais robustos.

Todavia, a variação uniforme de velocidade nesse tipo de máquina é bem mais complexa pelas dificuldades de modelagem (devido às variações na constante de tempo T_R do rotor, por exemplo) e definição da estratégia de controle adequada quanto aos equipamentos adicionais necessários. Porém, o progresso na eletrônica de potência e na microeletrônica vem possibilitando a implementação de soluções eficientes e econômicas. Várias estratégias de controle e acionamento das máquinas de indução vem sendo desenvolvidas no sentido de melhorar seu desempenho e aumentar sua gama de aplicações (Leonhard, 2001).

Comumente, o controle de velocidade desse tipo de máquina tem sido realizado utilizando a estratégia de controle tipo escalar através da variação proporcional tensão/freqüência. Porém, por se tratar de máquinas com características não-lineares, esse tipo de controle não apresenta desempenho satisfatório quando submetido a variações bruscas de carga, principalmente durante o regime transitório.

Como solução às limitações impostas pelo controle escalar surgiram as técnicas vetoriais de controle de velocidade. Tais técnicas aproximam o modelo trifásico da máquina de indução do modelo de uma máquina de corrente contínua (Leonhard, 2001). Esta transformação facilita o controle de velocidade, pois faz o desacoplamento do vetor de fluxo em relação ao de torque, permitindo o controle independente de ambos (Barbi, 1985). Através das técnicas de controle vetorial é possível obter bons desempenhos de velocidade e de torque

tanto durante o regime transitório quanto durante o regime permanente, mesmo sob variações de carga.

Entretanto, para que se possa obter este alto desempenho, é necessário que se tenha um conhecimento preciso da magnitude e da posição do campo girante, pois é através dele que se obtém o desacoplamento entre o controle de fluxo e o de torque. Isto pode ser conseguido com a utilização de sensores de fluxo ou através de algoritmos que os estimem.

O uso de sensores não é muito adequado nos motores de indução, pois seria necessário fazer modificações na estrutura da máquina para que eles pudessem ser adaptados. Além disso, os sensores não apresentam medições confiáveis para operação em freqüências de rotação muito baixas. Quando se tem o fluxo estimado o controle vetorial se torna mais confiável, pois a estimação se dá através de um algoritmo que envolve o modelo matemático da máquina, cujas variáveis recebem valores de medições feitas nos terminais da máquina.

Dentre as técnicas modernas de controle e estimação de sistemas que foram desenvolvidas nas últimas décadas, a estimação de modelos através das Redes Neurais Artificiais (RNAs) se apresenta como uma solução viável para plantas não-lineares e variantes no tempo (Narendra, 1990).

As redes neurais têm como objetivo principal a aprendizagem do comportamento de um sistema para diversas condições de funcionamento, fazendo com que variáveis estimadas ou controladas do mesmo se adaptem a possíveis variações e incertezas paramétricas.

1.2 Objetivo

O propósito maior deste trabalho é a implementação de um estimador do vetor de fluxo do rotor de um motor de indução trifásico com base nas técnicas de Redes Neurais Artificiais. As redes neurais são do tipo *feedfoward* multicamadas, aplicadas à estimação da corrente de magnetização e da velocidade angular do fluxo, e seu treinamento é feito *off-line* baseado em dados reais obtidos de ensaios realizados com o motor submetido a dois perfis de variações de referência de velocidade. As variáveis estimadas supracitadas estão relacionadas à magnitude e à posição do campo girante, respectivamente.

O estimador neural faz parte do sistema de controle vetorial de velocidade para um motor de indução implementado neste trabalho, sendo este sistema baseado em outro implementado por Paiva (2007), cujo controle é feito digitalmente por um dispositivo microcontrolado do tipo DSP (*Digital Signal Processor*) que oferece baixos tempos de processamento e alta capacidade de executar instruções dentro de um laço de controle.

O uso do DSP é justificado pela necessidade de se amostrar e controlar variáveis como as correntes trifásicas da máquina em freqüências elevadas e ainda efetuar operações aritméticas e lógicas. O DSP utilizado é um microcontrolador de 32 bits operando a uma freqüência de 150 MHz, o qual possibilita a comunicação com um computador PC através das interfaces paralela e serial RS-232, utilizadas para a programação e comunicação com o sistema supervisório (Spectrum, 2003), respectivamente.

O protótipo utilizado é um motor de indução trifásico de 1.1 kW / 4 pólos. O programa de controle vetorial de velocidade foi desenvolvido em linguagem padrão ANSI C através de um ambiente próprio no qual estão implementados, também, os controladores do tipo PI (Proporcional-Integrativo) para o controle das correntes do estator.

Com isso, espera-se contribuir com a pesquisa sobre controle de motores elétricos de indução sem mancais desenvolvida na Base de Pesquisa em Controle e Acionamento de Sistemas (BPCAS) da UFRN.

1.3 Sumário

A seguir, apresenta-se uma breve descrição dos capítulos que compõem este trabalho com seus respectivos conteúdos.

No Capítulo 2 o controle vetorial é introduzido. São apresentados o modelo vetorial da máquina de indução e o esquema de controle vetorial genérico de um sistema de controle vetorial de velocidade. Além disso, são mostrados os resultados de dois ensaios realizados com o motor utilizando o estimador convencional que servem de referência e motivação para a implementação do estimador neural.

Os aspectos gerais sobre Redes Neurais Artificiais e sua aplicação no desenvolvimento do estimador neural do vetor de fluxo do rotor são abordados no Capítulo 3.

O Capítulo 4 trata da implementação do sistema. Neste capítulo se apresentam os procedimentos realizados no sentido de alcançar o objetivo proposto.

Os resultados são apresentados no Capítulo 5. Uma análise comparativa do comportamento do sistema operando com cada um dos estimadores é realizada, assim como são relatadas as características e limitações práticas impostas pelo sistema.

No Capítulo 6, das conclusões e trabalhos futuros, são descritos os principais aspectos teóricos e práticos observados durante o desenvolvimento do sistema. Estes aspectos serão os pontos norteadores para o desenvolvimento e otimização do sistema em outros trabalhos na área de controle e acionamento das máquinas de indução.

Capítulo 2
Controle Vetorial de Velocidade para Máquinas de Indução

2.1 Introdução

Ao contrário do controle escalar do tipo V/F (tensão/freqüência), as técnicas vetoriais de controle para máquinas de indução apresentam alto desempenho tanto em regime permanente quanto durante os transitórios. Este rendimento é conseguido através do controle independente do torque e do fluxo (Barbi, 1985).

O controle vetorial, também conhecido como controle por orientação de campo, foi proposto por (Blaschke, 1972), e objetivamente consiste em controlar as correntes do estator da máquina, representadas por um vetor.

Esse tipo de controle se baseia em projeções que transformam um sistema trifásico em um sistema de duas coordenadas (d e q), implicando numa transformação do modelo da máquina de indução em um modelo similar ao de uma máquina de corrente contínua, de onde decorre o desacoplamento entre o controlador de torque e o de fluxo, tornando o controle bem mais simples e eficaz tanto para as altas quanto para as baixas rotações.

A principal exigência dos controladores vetoriais é o conhecimento do valor exato da magnitude e da posição do campo girante. Este fator gera a necessidade do uso de sensores de fluxo colocados no interior da máquina, o que pode ser inviável em determinados sistemas pela dificuldade de acesso e/ou pelo alto custo destes sensores.

Uma forma de contornar essas limitações é utilizar estimadores de fluxo baseados no modelo vetorial da máquina. Para implementar um estimador de fluxo, as entradas devem ser escolhidas de forma que elas estejam relacionadas a um referencial específico, sendo esta escolha dependente das condições de projeto e do grau de complexidade que é exigido pelo sistema. As entradas podem ser as correntes e as tensões do estator ou ainda a velocidade mecânica, que pode ser disponibilizada por um sensor acoplado ao eixo principal da máquina, e o referencial de fluxo pode ser definido entre os referenciais do fluxo do estator, do rotor ou ainda do entreferro, cada um deles com suas vantagens e desvantagens.

Os modelos representados em coordenadas de campo do estator e do entreferro são precisos e permitem a estimação tanto de fluxo quanto de velocidade mecânica através da leitura das correntes e tensões dos enrolamentos da máquina. Porém, estes modelos exigem um esforço computacional bem maior devido ao elevado número de equações neles presentes.

O modelo cujo referencial é o vetor de fluxo do rotor é representado por um número reduzido de equações sem perdas significativas de exatidão. A estimação utilizando-se deste referencial necessita apenas das correntes de fase do estator e da velocidade mecânica (Santisteban, 2001).

Analisando estas opções, é possível concluir que o referencial no fluxo do rotor tem preferência quando se deseja minimizar o esforço computacional e simplificar a implementação prática de um sistema.

Por essas razões, neste trabalho o referencial utilizado no controle vetorial é orientado pelo fluxo do rotor.

2.2 Modelo Vetorial da Máquina de Indução

A máquina de indução tem suas correntes representadas em regime permanente pelo diagrama vetorial ilustrado na Fig. 2.1 (Leonhard, 2001).

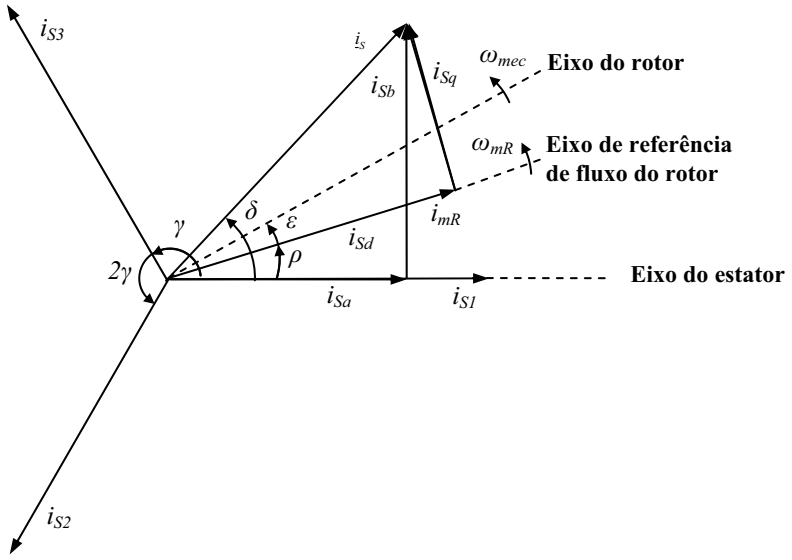


***Fig. 2.1**. Relações angulares entre os vetores de correntes.*

A partir deste diagrama e da consideração do sistema ser balanceado ($i_{S1} + i_{S2} + i_{S3} = 0$), pode-se descrever inicialmente a seguinte relação:

$$\underline{i}_S = i_{S1} + i_{S2}.e^{j.\gamma} + i_{S3}.e^{j.2.\gamma} = i_{Sa} + j.i_{Sb} \tag{2.1}$$

em que $\underline{i}_S$ é o vetor de corrente gerado pelas correntes trifásicas i_{S1}, i_{S2} e i_{S3} presentes no estator da máquina e i_{Sa} e i_{Sb} são as correntes correspondentes a um sistema bifásico equivalente no referencial estacionário.

Realizadas algumas substituições, com base na Eq. 2.1, obtêm-se as Eqs. 2.2 e 2.3 seguintes:

$$i_{Sa} = \frac{3}{2}.i_{S1} \tag{2.2}$$

$$i_{Sb} = \frac{\sqrt{3}}{2}.[i_{S2} - i_{S3}] \tag{2.3}$$

Representando essas correntes em coordenadas de campo através da posição angular ρ do campo girante, têm-se as Eqs. 2.4 e 2.5 seguintes:

$$i_{Sd} = i_{Sa}.\cos\rho + i_{Sb}.\sin\rho \tag{2.4}$$

$$i_{Sq} = i_{Sb}.\cos\rho - i_{Sa}.\sin\rho \tag{2.5}$$

Representadas no sistema $d-q$, essas correntes são, respectivamente, as correntes de campo i_{Sd} e de quadratura i_{Sq}, que são aplicadas ao modelo da máquina para o estudo e estimação de estados da mesma.

A estrutura do modelo vetorial da máquina de indução com referência no fluxo do rotor está mostrada no diagrama de blocos da Fig. 2.2 (Leonhard, 2001).

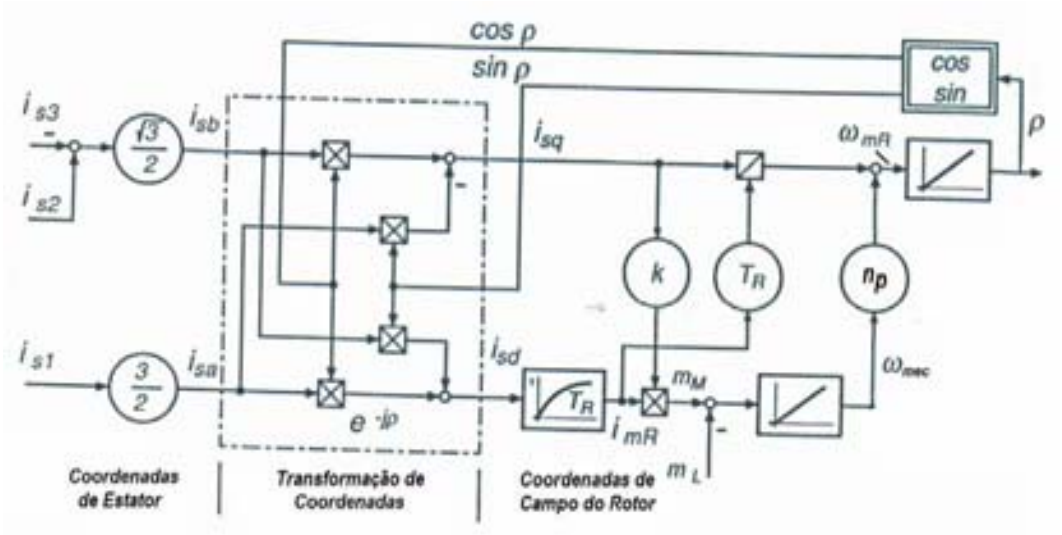


***Fig. 2.2**. Modelo da máquina de indução em coordenadas de campo do rotor.*

Com base nesse diagrama, as seguintes equações são obtidas:

$$m_M = k.i_{mR}.i_{Sq} \tag{2.6}$$

$$k = \frac{2}{3}.(1-\sigma).L_S \tag{2.7}$$

$$\frac{di_{mR}}{dt} = \frac{1}{T_R}.i_{Sd} - \frac{1}{T_R}.i_{mR} \tag{2.8}$$

$$\frac{d\rho}{dt} = n_p.\omega_{mec} + \frac{1}{T_R}.\frac{i_{Sq}}{i_{mR}} \tag{2.9}$$

em que m_M é o torque elétrico aplicado à máquina, k é uma constante relacionada à indutância própria do estator, L_S, e ao fator de dispersão, σ, i_{mR} é a corrente de magnetização que está diretamente relacionada à magnitude do campo girante, T_R é a constante de tempo do rotor, ω_{mec} é a velocidade mecânica do rotor, n_p é o número de pares de pólos da máquina e ρ é a posição do campo girante em coordenadas de fluxo do rotor.

Fazendo uma comparação, a Eq. 2.6 lembra a equação de torque elétrico de uma máquina de corrente contínua, onde i_{mR} corresponde ao fluxo principal e i_{Sq} (corrente de torque) à corrente de armadura (Leonhard, 2001). Conforme mostra a Fig. 2.1, a corrente de magnetização i_{mR} é controlada pela corrente de campo i_{Sd} do vetor de corrente do estator, a qual pode ser comparada com a tensão de campo, por estarem consideravelmente ligadas magneticamente. Portanto, i_{Sd} e i_{Sq} são duas quantidades de entrada independentes.

Para o estudo do comportamento mecânico do modelo da máquina são utilizadas as Eqs. 2.10 e 2.11. Nestas equações o atrito viscoso foi desprezado.

$$m_L = D.m_M \tag{2.10}$$

$$\frac{d\omega_{mec}}{dt} = \frac{1}{J}.[m_M - m_L] \tag{2.11}$$

em que m_L é o torque de carga, D é o fator de carga e J é o momento de inércia do rotor.

O estimador convencional de fluxo se baseia no modelo vetorial apresentado na Fig. 2.2 e é utilizado em sistemas de controle vetorial de velocidade para máquinas de indução.

2.3 Controle Vetorial de Velocidade

O objetivo básico dos controladores vetoriais é a obtenção de respostas rápidas de velocidade com torque elevado para as máquinas de indução durante o regime transitório e também no regime permanente.

Genericamente, os sistemas de controle vetorial de velocidade para as máquinas de indução seguem o esquema apresentado no diagrama da Fig. 2.3 (Leonhard, 2001).

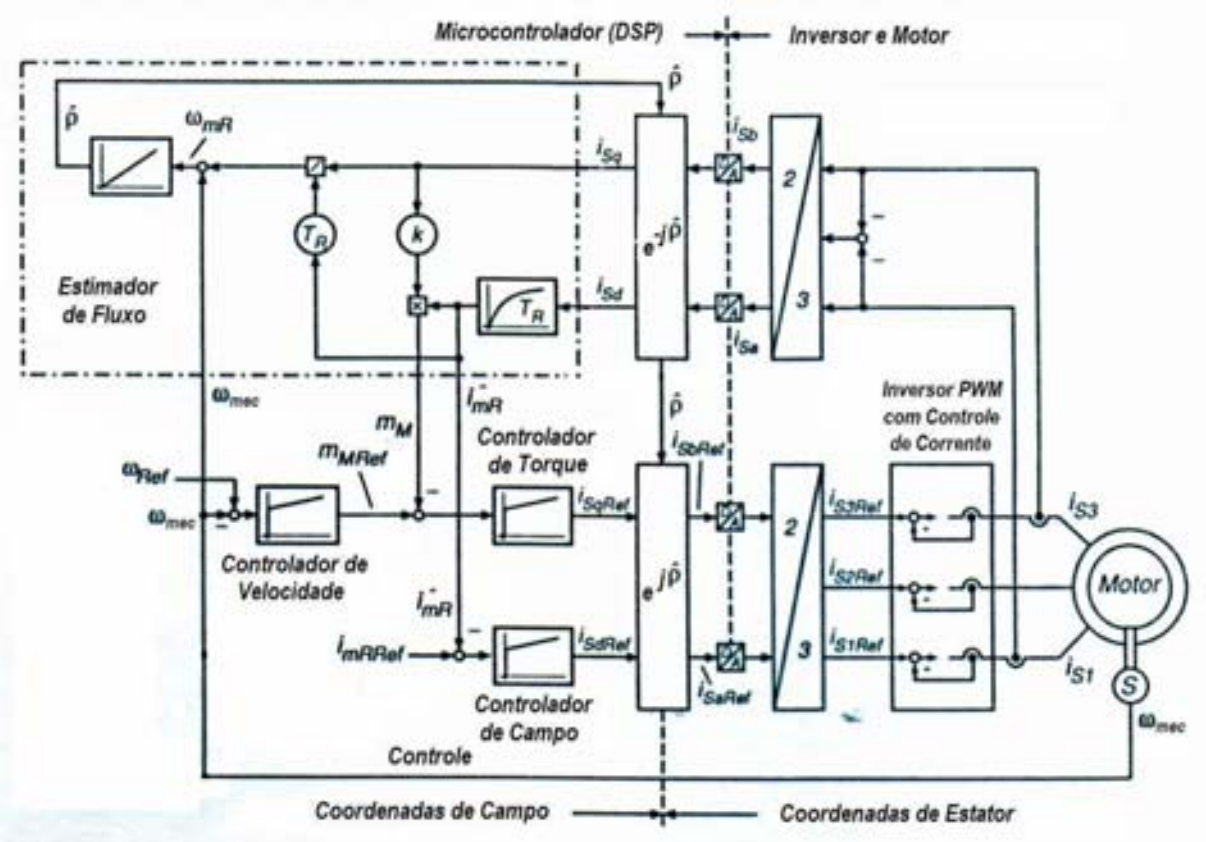


***Fig. 2.3**. Esquema genérico de um sistema de controle vetorial de velocidade orientado pelo fluxo do rotor alimentado por correntes impostas.*

De acordo com a Fig. 2.3, o controle vetorial de velocidade é composto basicamente por três controladores do tipo PI, sendo eles: um controlador para o erro de velocidade cuja saída é o torque elétrico de referência, m_{MRef}. Em série com esse controlador, tem-se o controlador de torque, o qual é responsável pela geração da referência de corrente de torque, i_{SqRef}. E o terceiro controlador executa o controle da corrente de magnetização, i_{mR}, o qual é responsável pela geração da referência de corrente de campo, i_{SdRef}.

As saídas de todos os controladores devem possuir limitadores do sinal de controle como forma de obedecer aos valores possíveis de serem aplicados à máquina.

As correntes i_{SdRef} e i_{SqRef} passam por uma transformação rotacional gerando as correntes de um sistema bifásico, i_{SaRef} e i_{SbRef}. Essas correntes bifásicas passam por uma outra transformação para se obter as correntes trifásicas de referência, que são aplicadas à máquina:

$$i_{S1Ref} = \frac{2}{3}.i_{SaRef} \tag{2.12}$$

$$i_{S2Ref} = -\frac{1}{3}.i_{SaRef} + \frac{\sqrt{3}}{3}.i_{SbRef} \tag{2.13}$$

$$i_{S3Ref} = -\frac{1}{3}.i_{SaRef} - \frac{\sqrt{3}}{3}.i_{SbRef} \tag{2.14}$$

As transformações angulares são realizadas com base no ângulo ρ gerado pelo estimador de fluxo. O modelo do estimador de fluxo disponibiliza também o torque elétrico m_M e a corrente de magnetização i_{mR} atuais. O estimador de fluxo tem como entradas as correntes i_{Sd} e i_{Sq} e a velocidade mecânica, ω_{mec}.

Toda a comunicação do microcontrolador com os circuitos de interface é realizada através dos conversores A/D e D/A disponíveis na maioria dos dispositivos microcontrolados dedicados. Em microcontroladores do tipo DSP (*Digital Signal Processor*) as funções de controle de corrente e geração de sinais PWM podem ser implementadas internamente ao próprio dispositivo.

Os sistemas de controle vetorial de velocidade para máquinas de indução podem ser implementados através de modelos alimentados por tensões ou correntes impostas.

Quando se utiliza o modelo alimentado por correntes impostas, as equações de malha do estator são eliminadas, fazendo com que a implementação digital seja mais simples. No entanto, gera-se uma dependência considerável em relação à constante de tempo do rotor, T_R. Caso os parâmetros do rotor variem ou não sejam conhecidos com exatidão, esta dependência produz erros consideráveis na estimação do fluxo.

Para a alimentação por tensão, tem-se uma estimação mais exata dos estados internos da máquina. Porém, sua implementação digital é mais complexa e demanda mais tempo de processamento.

Apesar das limitações, o sistema de controle deste trabalho utiliza alimentação por correntes impostas e os parâmetros do rotor do estimador convencional de fluxo advêm dos dados obtidos nos ensaios feitos com o motor de indução apresentados na seção 2.4.

Pode-se implementar ainda neste sistema o enfraquecimento de campo para a geração de i_{mRRef}, porém esta operação demanda mais processamento do microcontrolador. Neste trabalho é utilizada uma referência constante de corrente de magnetização.

O modelo e sistema de controle vetorial de velocidade do motor de indução desenvolvido neste trabalho se baseiam no modelo e sistema mostrados nas Figs. 2.2 e 2.3, respectivamente, e são implementados em DSP. Porém, no que tange ao sistema, apenas os controladores de velocidade e da corrente de magnetização são implementados.

2.4 Ensaios Práticos com o Modelo Vetorial

São apresentados a seguir os resultados de dois ensaios realizados, cada um com suas variações de referência de velocidade, cujo objetivo é mostrar características tanto de não-linearidades do motor de indução quanto de desempenho do sistema de controle vetorial de velocidade em malha aberta, além de servirem de referência e motivação para a implementação de um estimador neural de fluxo.

Os gráficos seguintes mostram o comportamento das variáveis do modelo vetorial do motor de indução com referência no fluxo do rotor o qual é submetido a variações de referência de velocidade. Estas variáveis são as velocidades mecânica ω_{mec} e angular do fluxo do rotor $\frac{d\rho}{dt}$, as correntes de campo i_{Sd} e de magnetização i_{mR} e a corrente de torque i_{Sq}.

O motor de indução utilizado nos ensaios teve seus parâmetros determinados através dos métodos convencionais de medição (Simone, 2000), listados no apêndice A.

2.4.1 Ensaio 1: Referências de velocidade variando em degraus

Nas Figs. 2.4, 2.5 e 2.6 estão ilustrados os gráficos dos comportamentos das variáveis do motor de indução submetido a referências de velocidade variando em degraus sem a aplicação de carga.

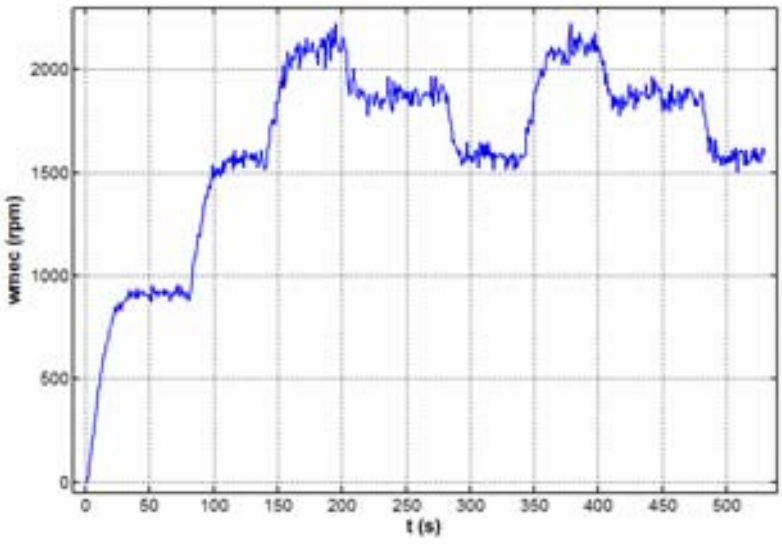


Fig. 2.4. *Resposta de velocidade mecânica com referências variando em degraus.*

Como se pode observar na Fig. 2.4, tem-se uma rápida resposta de velocidade devido à ausência de carga.

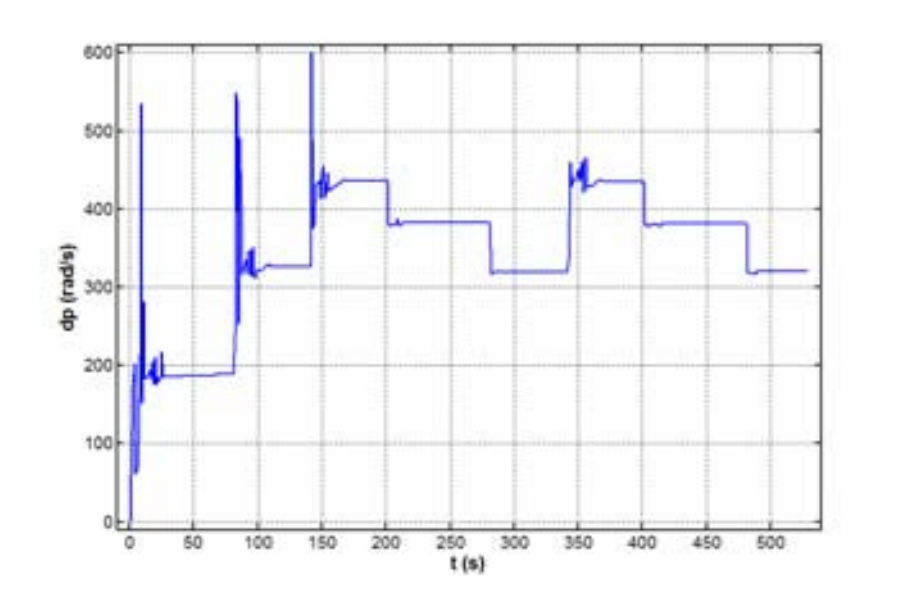


Fig. 2.5. *Comportamento da velocidade angular do fluxo magnético do rotor.*

Na Fig. 2.5 se observam distúrbios antes que a velocidade angular do fluxo do rotor se estabilize, mostrando claramente características de não-linearidades advindas por exemplo da variação de T_R com a temperatura ou saturação do fluxo magnético do rotor.

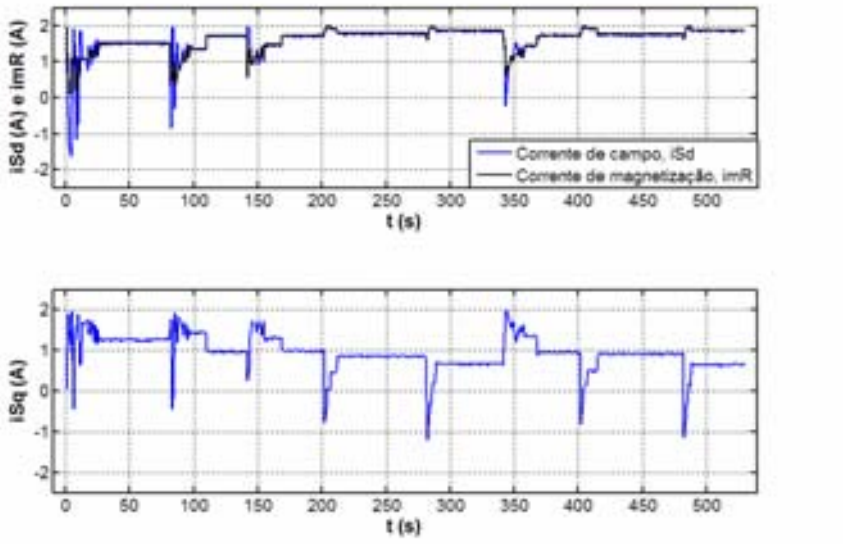


Fig. 2.6. *Correntes de campo e de magnetização e corrente de torque.*

E, pelo que se vê na Fig. 2.6, as correntes i_{Sd} e i_{mR} e i_{Sq} tendem para valores contínuos e constantes, sendo que a corrente de campo i_{Sd} tende para a amplitude máxima das

correntes de fase e a corrente de torque i_{Sq} tenderia para zero devido a sua relação com o torque elétrico, o que não ocorre devido a inércia existente.

2.4.2 Ensaio 2: Referências de velocidade variando em rampas

Da mesma forma que no Ensaio 1, as Figs. 2.7, 2.8 e 2.9 ilustram, respectivamente, o comportamento de cada uma das mesmas variáveis, só que agora com o motor de indução submetido a referências de velocidade variando em rampas e também não se aplicou carga.

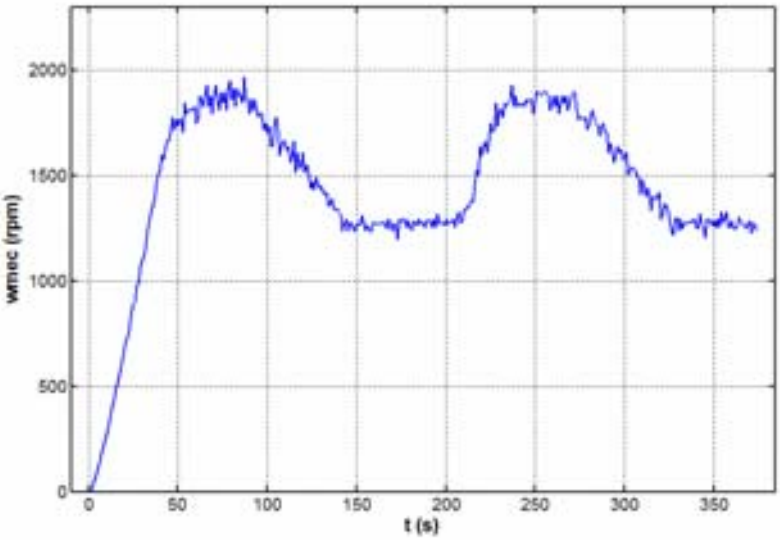


Fig. 2.7. *Resposta de velocidade mecânica com referências variando em rampas.*

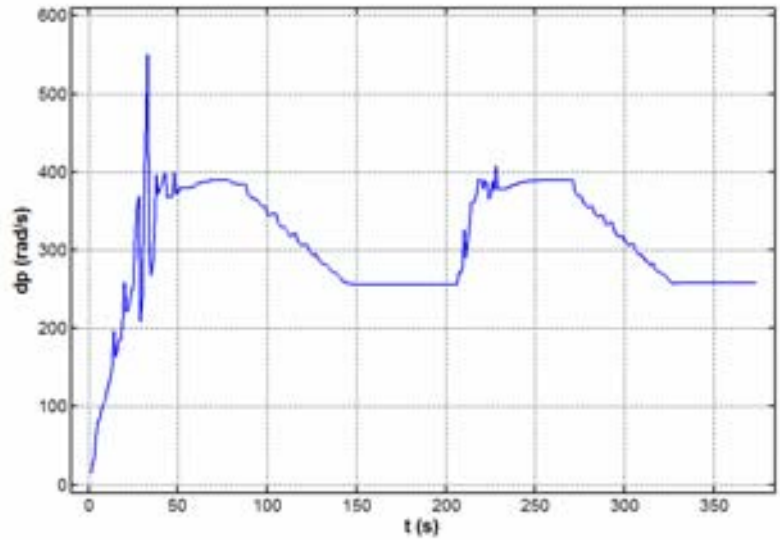


Fig. 2.8. *Comportamento da velocidade angular do fluxo magnético do rotor.*

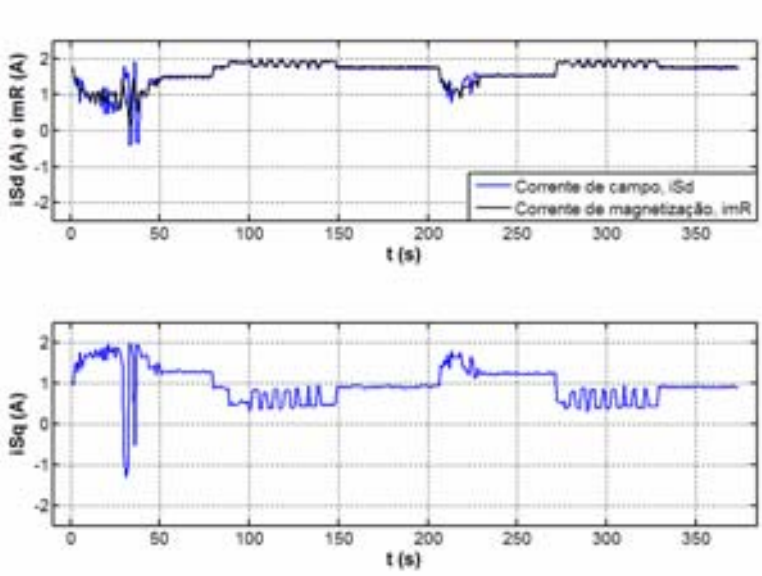


Fig. 2.9. *Correntes de campo e de magnetização e corrente de torque.*

2.5 Conclusões

Este capítulo tratou da estratégia de controle vetorial de velocidade para máquinas de indução. Em termos de desempenho, se comparado ao controle escalar, tem-se que a técnica vetorial apresenta alto desempenho tanto em regime permanente quanto durante os transitórios das máquinas de indução.

O estudo apresentou o modelo vetorial da máquina de indução orientado pelo fluxo do rotor que tem um estimador convencional de fluxo nele baseado. Por esse modelo ser similar ao modelo de uma máquina de corrente contínua torna seu controle bem mais simples e eficaz tanto para as altas quanto para as baixas rotações devido ao desacoplamento entre o controlador de torque e o de fluxo.

Foi também mostrado o esquema genérico de um sistema de controle vetorial de velocidade orientado pelo fluxo do rotor no qual se baseia a implementação prática do sistema de controle deste trabalho, a menos do controlador de torque.

E para mostrar características de não-linearidades do motor de indução utilizado assim como o desempenho do sistema de controle em malha aberta e sob diferentes condições de funcionamento foram apresentados resultados de ensaios práticos utilizando o modelo vetorial do motor que servem de referência e motivação para a implementação do estimador neural de fluxo.

Capítulo 3

Redes Neurais Artificiais na Estimação Neural de Fluxo

3.1 Introdução

A área de controle de sistemas vem sofrendo profundas modificações na abordagem e manipulação de suas grandezas físicas e matemáticas. Estas modificações afetam também a aplicação de controladores clássicos projetados para modelos lineares com parâmetros constantes e ajustados para pontos de operações específicos.

Como a maioria dos sistemas reais é não-linear, esses controladores apresentam limitações de desempenho em regiões distantes do ponto de operação de projeto.

Para solucionar os problemas de não-linearidades em sistemas de difícil modelagem, algumas técnicas modernas de estimação de modelos não-lineares vêm sendo desenvolvidas e têm apresentado bons resultados quando aliadas aos controladores clássicos ou mesmo aos controladores modernos, como por exemplo, os sistemas de controle robustos e controladores adaptativos (Vaz, 1996; Furtunato, 2001).

Dentre as técnicas de estimação de plantas não-lineares em evidência, destacam-se os estimadores baseados nas Redes Neurais Artificiais (RNAs). Este destaque se deve a sua ampla faixa de aplicação e pela capacidade de adaptação e aprendizado do comportamento das plantas a serem estimadas.

Uma das áreas de pesquisa mais promissoras para a implementação das redes neurais artificiais é a área de controle e de estimação dos parâmetros e estados das máquinas elétricas, como as máquinas de indução (Chen, 1994; Almeida, 1999).

As máquinas de indução possuem não-linearidades em seus modelos e também sofrem variações nos valores de seus parâmetros devido a fatores internos como a saturação magnética do rotor e externos como as variações de temperatura. Desta forma, a aplicação de controladores clássicos ajustados através de modelos lineares ou linearizados fica limitada a pontos de operações específicos.

Para contornar estes problemas, os estimadores baseados em RNAs podem ser treinados para vários pontos de operação, de modo a contemplar e compensar as variações nos

parâmetros da máquina e também manter o desempenho desejado para os controladores do sistema, sejam eles clássicos ou modernos.

A proposta deste trabalho é utilizar a estimação neural de fluxo rotórico da máquina de indução em substituição a estimação convencional baseada no modelo inverso com parâmetros associados aos dados obtidos nos ensaios realizados no Capítulo 2.

Este capítulo apresenta os fundamentos das RNAs e sua aplicação na estimação dos principais estados da máquina elétrica de indução.

Com o objetivo de justificar a eficiência das redes neurais, ao final deste capítulo são apresentados os resultados dos mesmos ensaios realizados no Capítulo 2 utilizando o estimador neural de fluxo do rotor implementado onde são comprovadas melhorias quando se comparam estes com aqueles que utiliza o estimador convencional de fluxo.

3.2 Fundamentos Teóricos das Redes Neurais Artificiais

O estudo das Redes Neurais Artificiais (RNAs) fornece um enfoque alternativo a ser aplicado em problemas onde os enfoques numérico e simbólico não são muito adequados. As Redes neurais artificiais são apenas inspiradas no nosso conhecimento atual sobre sistemas nervosos biológicos da natureza, e não buscam ser realísticas em todos os detalhes, isto é, o modelamento de sistemas nervosos biológicos não é o ponto principal de interesse (Nascimento Jr., 2004).

O cérebro humano contém aproximadamente 10^{11} células nervosas elementares chamadas de neurônios. Cada um desses neurônios está conectado a cerca de 10^3 a 10^4 outros neurônios. Portanto, estima-se que o cérebro humano teria 10^{14} a 10^{15} interconexões.

Estas características fornecem uma imensa capacidade de processamento computacional e de memória.

Desta forma, uma RNA é uma máquina projetada para tentar reproduzir a maneira pela qual o cérebro realiza uma tarefa particular ou função de interesse.

Normalmente, as RNAs são implementadas com componentes eletrônicos ou simuladas em computadores digitais. Dentre as muitas definições sobre as RNAs, cita-se Haykin (2001):

> *A rede neural é um processador maciçamente paralelamente distribuído, constituída de unidades de processamento simples, que têm a propensão natural para armazenar o conhecimento experimental e torná-lo disponível para o uso. Ela se assemelha ao cérebro em dois aspectos:*

i) O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem;

ii) Forças de conexão entre os neurônios artificiais, conhecidas como pesos sinápticos, são usadas para armazenar o conhecimento adquirido."

3.2.1 O Neurônio Artificial

O neurônio artificial é a unidade de processamento de informação de uma rede neural e pode ser representado por um modelo linear ou não-linear.

Os modelos não-lineares têm a capacidade de representar com maior exatidão os sistemas físicos, que em sua grande maioria, possuem características não-lineares. No entanto, sua implementação prática requer elevada carga computacional em relação aos neurônios com funções lineares.

Por esta razão, neste trabalho, optou-se por utilizar neurônios com funções lineares baseados no modelo de McCulloch-Pitts (1943), cuja estrutura está mostrada na Fig. 3.1.

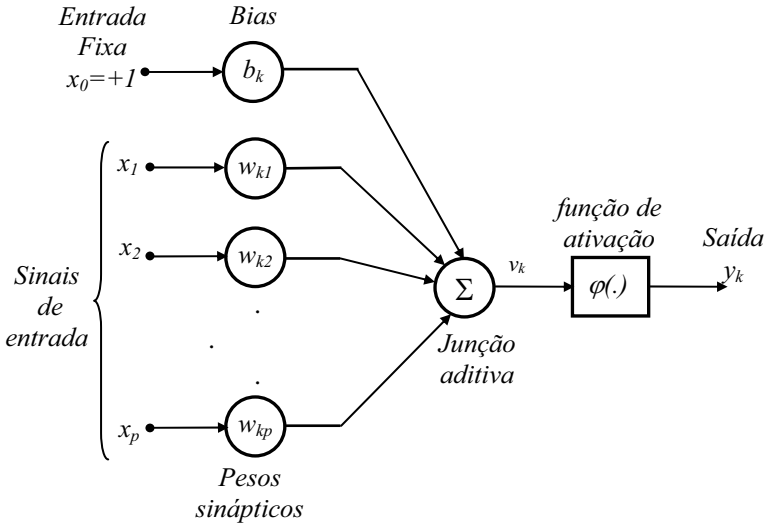


Fig. 3.1. *Modelo de neurônio Artificial de McCulloch-Pitts.*

O neurônio artificial mostrado é constituído por um conjunto de conexões sinápticas (pesos), um somador, uma função de ativação ou função restritiva, sinais de entrada e sinal de saída.

Uma das principais características deste modelo é a inclusão no somador do efeito gerado pelo *bias*, cuja função é aumentar ou diminuir a entrada líquida da função de ativação $\varphi(.)$.

Em termos matemáticos, a operação do *k*-ésimo neurônio de uma rede neural é expressa por:

$$u_k = \sum_{j=1}^{p} w_{kj}.x_j \tag{3.1}$$

$$v_k = u_k + b_k \tag{3.2}$$

e

$$y_k = \varphi(v_k) \tag{3.3}$$

em que p é o número de entradas, $x_1, x_2, ..., x_p$ são as p entradas, $w_{k1}, w_{k2}, ..., w_{kp}$ são os pesos sinápticos do neurônio k, x_0 é a entrada de polarização, b_k é o peso *bias*, u_k é a saída do combinador linear das entradas $x_1, x_2, ..., x_p$, v_k é o potencial de ativação, $\varphi(.)$ é a função de ativação e y_k é a saída do $k-$ésimo neurônio.

3.2.2 Funções de Ativação

Em geral, o neurônio artificial contém uma função de ativação não-linear que busca representar o comportamento não-linear do neurônio biológico. O uso de funções de ativação não-lineares nos neurônios possibilita a identificação de modelos não-lineares. Porém, a implementação digital deste tipo de função gera uma carga computacional considerável, principalmente quando o processador opera com aritmética de ponto fixo. Portanto, a escolha da função de ativação depende de diversos fatores como: o tamanho da rede a ser implementada, a complexidade do sistema de controle no qual a rede será inserida, o erro de estimação permitido e a capacidade de processamento do dispositivo digital que executará o algoritmo de estimação (Narendra, 1990).

O *k*-ésimo valor de ativação de um modelo é dado por:

$$f(v_k) = f\left(\sum_{j=1}^{p} w_{kj}.x_j + b_k\right) \tag{3.4}$$

Dentre os vários tipos de funções de ativação utilizados nas redes neurais artificiais, os mais comuns são:

- ***Função de limiar*** $$\varphi(v) = \begin{cases} 1 & \text{se } v \geq 0 \\ 0 & \text{se } v < 0 \end{cases} \tag{3.5}$$

- ***Sigmóide*** $$\varphi(v) = sig(v) = \frac{1}{1 + e^{(-v)}} \tag{3.6}$$

- ***Tangente hiperbólica*** $$\varphi(v) = \frac{\left(e^{(v)} - e^{(-v)}\right)}{\left(e^{(v)} + e^{(-v)}\right)} \tag{3.7}$$

- ***Arcotangente*** $$\varphi(v) = \frac{2}{\pi} arctan(v) \quad (3.8)$$

- ***Linear*** $$\varphi(v) = v \quad (3.9)$$

Em sistemas não-lineares como por exemplo, as máquinas elétricas de indução, existe uma tendência natural da escolha de funções de ativação não-lineares para representar com maior aproximação os estados internos dos mesmos. No entanto, levando em consideração os fatores citados anteriormente, a implementação digital de estimadores com este tipo de função pode se tornar difícil ou mesmo inviável, em caso de uso de dispositivos dedicados para implementação. Por essa razão, neste trabalho, optou-se por utilizar funções de ativação lineares em todos os neurônios que compõem as redes do estimador neural de fluxo do rotor.

3.2.3 Arquiteturas das Redes Neurais Artificiais

A maneira pela qual os neurônios estão interconectados define a arquitetura da Rede Neural. Em geral, existem três classes de arquiteturas de redes fundamentalmente diferentes: redes alimentadas diretas com camadas únicas, redes alimentadas diretas de múltiplas camadas e as redes recorrentes (Haykin, 2001).

A arquitetura das redes neurais com aplicações significativas na área de controle e acionamento de máquinas elétricas são as redes neurais alimentadas adiante com múltiplas camadas, ou redes do tipo *Feedfoward.*

Esta rede possui uma ou mais camadas ocultas, cujos nós computacionais são chamados de neurônios ocultos, como mostra a Fig. 3.2.

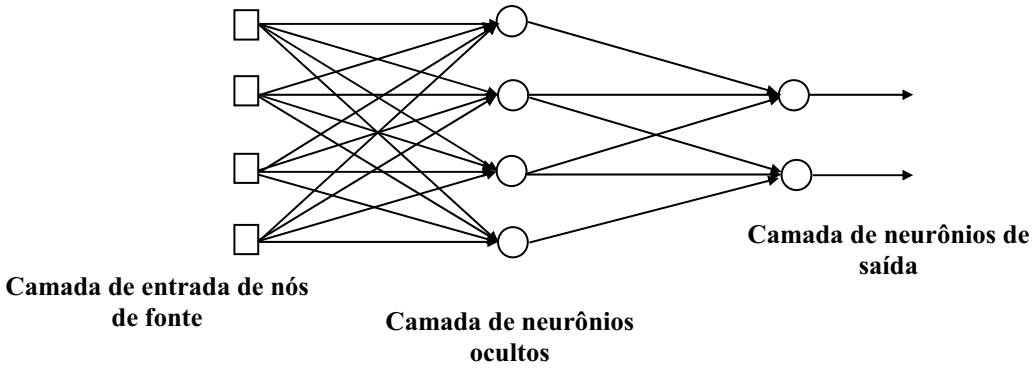


Fig. 3.2. *Arquitetura de uma Rede Alimentada Adiante com Múltiplas Camadas ou redes tipo Feedfoward.*

A função dos neurônios ocultos é intervir entre a entrada externa e a saída da rede de maneira útil. Porém, estes elementos tornam o processamento das informações e o algoritmo de aprendizagem mais complexo.

Desta forma, a escolha da topologia da rede influencia diretamente na exatidão obtida pelo algoritmo de treinamento, pois, ao utilizar-se um número pequeno de neurônios ou mesmo um número reduzido de camadas escondidas, acelera-se o processo de treinamento. No entanto, a saída da rede fica limitada a pontos de operação muito restritos.

Caso se opte por utilizar um número elevado de neurônios ou de camadas escondidas, geralmente o aprendizado é mais eficiente e, conseqüentemente, generaliza-se melhor a saída da rede para diferentes pontos de operação. Todavia, a carga computacional para se executar em tempo real o cálculo da rede é bem maior.

Por essa razão, ao se iniciar o processo de treinamento (mesmo que no modo *off-line*) deve-se testar diferentes topologias de modo a obter uma solução que contemple as condições de operação dos valores para a saída sem comprometer o desempenho do dispositivo de controle.

3.2.4 Tipos de Redes Neurais Artificiais

Com o avanço das pesquisas na área das redes neurais artificiais, diversos tipos de redes foram desenvolvidos e adaptados às mais diversas aplicações (Narendra, 1996). Como conseqüência, foram geradas redes que possuem estruturas híbridas, com e sem realimentação, diferenciando-se na complexidade de seus projetos e algoritmos de aprendizado.

Dentre os tipos mais conhecidos estão as redes do tipo PERCEPTRON, ADALINE e MADALINE, Rede WAVELET Polinomial e Rede de FOURIER e rede de Funções SAMPLE. Cada um destes tipos de redes possui aplicações diversas como, por exemplo, aproximação de funções, reconstituição e tratamento de sinais e principalmente a estimação de estados de sistemas físicos.

Dentre estes tipos de redes, as redes PERCEPTRON de múltiplas camadas têm sido aplicadas com sucesso para resolver problemas complexos, através do treinamento de forma supervisionada. Para as aplicações em estimação de estados das máquinas elétricas em geral, elas vêm se difundindo através de diversos trabalhos na área de controle e acionamento de máquinas.

As redes PERCEPTRON são do tipo totalmente conectado, isto significa que um neurônio em qualquer camada da rede está conectado a todos os nós da camada anterior, como mostra a Fig. 3.3.

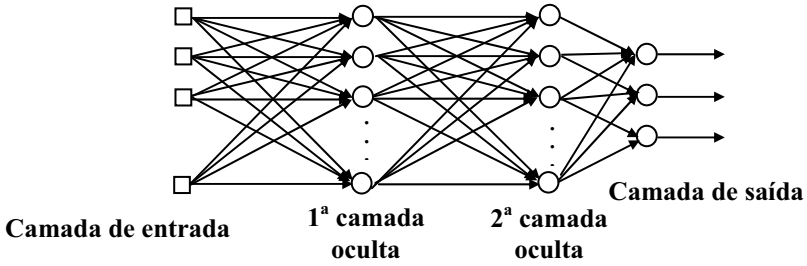


Fig. 3.3. *Estrutura de uma rede Perceptron de múltiplas camadas.*

Devido ao número de aplicações bem sucedidas deste tipo de rede na estimação de estados das máquinas elétricas em geral, optou-se neste trabalho por utilizar este tipo de rede para a estimação neural do fluxo do rotor para o controle vetorial de velocidade do motor de indução.

3.2.5 Aprendizagem e Treinamento de uma Rede Neural Artificial

Aprender é o ato que produz um comportamento diferente a um estímulo externo devido a excitações recebidas no passado e é, de uma certa forma, sinônimo de aquisição de conhecimento.

Em inteligência artificial é comum se falar de aprendizado pela máquina e aprender pode ser considerado como atributo fundamental de um comportamento inteligente. As RNAs possuem a capacidade de aprenderem por exemplos, e fazerem interpolações do que aprenderam. Existem diversas maneiras de aprendizado, dentre elas podemos citar:

- Memorização;
- Por ser contado;
- Por exemplos;
- Por analogia;
- Por exploração e descoberta.

As RNAs aprendem principalmente por uma mistura dos três últimos.

Existem basicamente dois tipos de aprendizado para as redes neurais, são eles:

i) *Aprendizado Supervisionado* – Existe a figura do professor que indicará se o comportamento do aprendizado está bom ou ruim, e este deve aplicar as correções necessárias para o bom desempenho da rede.

ii) *Aprendizado fracamente supervisionado* – É quando para se fazerem modificações nos valores das conexões sinápticas não se usam informações sobre se a resposta da rede foi correta ou não.

Após a definição do modo de aprendizado da rede, inicia-se então o processo de

treinamento cujo principal objetivo é a obtenção de um conjunto de pesos, de modo que o erro médio entre entrada e saída seja mínimo.

Para a realização desta tarefa, existem diferentes algoritmos. Dentre os mais conhecidos estão a regra Delta (Haykin, 2001) para redes de camadas únicas e o algoritmo *back-propagantion*, que pode ser considerado como uma generalização da regra Delta para redes de múltiplas camadas. O algoritmo *back-propagation* consiste basicamente de dois passos através das camadas da rede: um passo para frente e outro para trás. No passo para frente, os pesos da rede são fixos e o conjunto das entradas se propaga através dos diferentes nós até atingir a saída da rede. O sinal de saída é comparado com o sinal desejado, gerando um erro. Este erro é processado através de uma regra de correção de erros, cujo efeito se propaga para trás corrigindo todos os pesos da rede até a camada de entrada. Os processos de propagação e retropropagação continuam até que seja atingido o erro mínimo desejado entre a saída real da rede e a saída utilizada como padrão (Haykin, 2001).

Este algoritmo pode ser utilizado tanto no treinamento *off-line* como no treinamento *on-line*. Porém seu uso no treinamento *on-line* gera uma grande demanda computacional para o dispositivo digital de controle. Esta demanda pode inviabilizar seu treinamento, pois além do algoritmo da rede, este dispositivo deve executar o controle global do sistema. Neste trabalho, o treinamento foi feito *off-line* a partir dos dados obtidos de ensaios realizados com o motor de indução, apresentados no Capítulo 2.

3.3 Estimação Neural aplicada ao Controle de Sistemas

Um sistema de controle é dito ser neural quando usa de alguma forma uma rede neural como parte componente.

As pesquisas na área dos Neurocontroladores têm obtido avanços significativos, principalmente nos casos em que o modelo do processo a ser controlado oferece grande dificuldade de ser obtido e se deseja usar a capacidade de aprendizado das RNAs para obter a solução do problema.

Os sistemas neurocontrolados já estão presentes em uma ampla faixa de aplicações como as refinarias de petróleo, plantas químicas, siderúrgicas, eletrodomésticos, controle de robôs, etc.

Uma das grandes barreiras para o uso das redes neurais em sistemas de controle em tempo real é a elevada carga computacional dos algoritmos de aprendizagem, fazendo com que se opte por uma aprendizagem do tipo *off-line*, podendo levar a rede a não se adaptar às mudanças de parâmetros da planta.

Uma das aplicações mais difundidas dos neurocontroladores é a identificação de modelos de plantas. Geralmente, estas plantas apresentam não-linearidades que tornam a modelagem pelos métodos tradicionais inviável. Um exemplo de sistema de identificação de modelos está mostrado na Fig. 3.4.

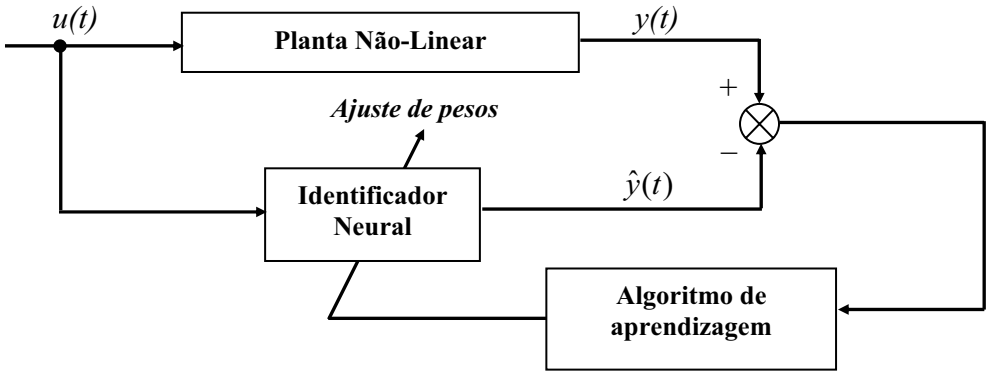


Fig. 3.4. *Exemplo de sistema com identificador neural para planta não-linear.*

Neste sistema, a ação do controle é aplicada simultaneamente às entradas da planta e do identificador neural (Rede Neural). Então, a saída da planta $y(t)$ é comparada à saída do identificador $\hat{y}(t)$, e o erro gerado $erro(t)$ é aplicado ao algoritmo de aprendizagem para o ajuste dos pesos da rede, até que o erro entre planta e modelo neural seja nulo. Ao terminar a aprendizagem, considera-se que a dinâmica do modelo da planta foi identificada.

Utilizando estes conceitos para a estimação de modelos, encontramos aplicações importantes para os observadores neurais e as técnicas vetoriais de controle para máquinas elétricas de indução. Dentre as principais variáveis estimadas estão o fluxo magnético, o torque elétrico e também a velocidade mecânica, conforme está descrito nas aplicações a seguir.

Uma das primeiras aplicações das redes neurais para máquinas elétricas está apresentada em Simões (1995). Neste trabalho é proposta uma rede do tipo *feedfoward* para a estimação dos sinais de realimentação de fluxo, torque e vetores de seno e cosseno do ângulo de referência, para a aplicação em um sistema de controle vetorial direto (DVC) para máquinas de indução. Neste trabalho, os resultados de estimação foram apresentados através de simulações computacionais e sua principal contribuição se deve à proposta de operação do estimador neural implementado em um computador PC em conjunto com um programa de controle desenvolvido para DSP, que nesta época já se apresentava como um dispositivo bastante versátil e eficiente para a implementação de sistemas digitais de controle.

Outra aplicação importante das redes neurais é apresentada em Maia (1997). Neste trabalho, foi proposta uma rede neural para a orientação pelo campo e controle de torque da

máquina de indução. Nesta aplicação, o controle da rede é derivado de uma técnica conhecida como "*gain sheduling technique"*, a qual executa a média de um modelo da máquina de indução variante no tempo para diversas condições de operação. Os resultados de simulação mostraram que a estratégia neural de controle permite uma boa interpolação com o modelo utilizado para o treinamento, gera respostas rápidas de torque e mantém a orientação de fluxo com erro de ordem zero nulo.

Dando segmento às aplicações dos estimadores neurais, em Almeida (1999) apresenta-se a estimação neural do fluxo do rotor para controle orientado pelo campo de uma máquina de indução.

Recente implementação de estimador neural de fluxo do rotor em DSP é apresentada em Lisboa (2007). Nesse caso, a eficiência é comprovada pelas comparações geradas em tempo real entre os estimadores neurais e os estimadores convencionais baseados no modelo inverso com parâmetros fixos. Estas comparações forneceram resultados importantes em relação à robustez e eficiência no consumo de energia do sistema para a operação com o estimador neural de fluxo. Em Paiva (2007), o comportamento da rede para variações (em termos percentuais) na constante de tempo do rotor, T_R, é estudado através de simulações computacionais, sendo as informações obtidas depois aplicadas ao estimador neural de fluxo do sistema de controle de velocidade respectivo.

O estimador neural desenvolvido neste trabalho foi treinado para aprender de modo eficiente a dinâmica do modelo inverso da planta a partir dos dados obtidos de ensaios feitos com a máquina. Para permitir uma adaptação eficiente às diferentes condições de operação, durante o processo de treinamento, foram aplicadas variações nas referências de velocidade. Os resultados apresentados comprovam a hipótese de robustez e versatilidade das redes neurais aplicadas à estimação de fluxo do rotor.

Enfim, os resultados estudados nos trabalhos citados para as máquinas de indução motivaram e direcionaram a implementação do sistema de controle vetorial de velocidade da máquina de indução. Esta implementação teve o objetivo de aliar as vantagens promovidas pelo uso das técnicas vetoriais de controle e a capacidade de aprendizado das redes neurais sobre plantas não-lineares como a máquina de indução.

O sistema de controle vetorial com estimação neural de fluxo para um motor de indução utiliza quase todos os elementos presentes no modelo clássico de controle vetorial orientado pelo fluxo do rotor proposto em Leonhard (2001). A diferença está no tipo de estimador utilizado e na inexistência do controlador de torque, como mostra o diagrama de

blocos da Fig. 3.5. Observando esta mesma figura, vê-se que está sendo feito o cálculo do consumo de energia a partir da integração do produto entre a velocidade mecânica ω_{mec} e o torque elétrico m_M aplicado à máquina, que também é calculado.

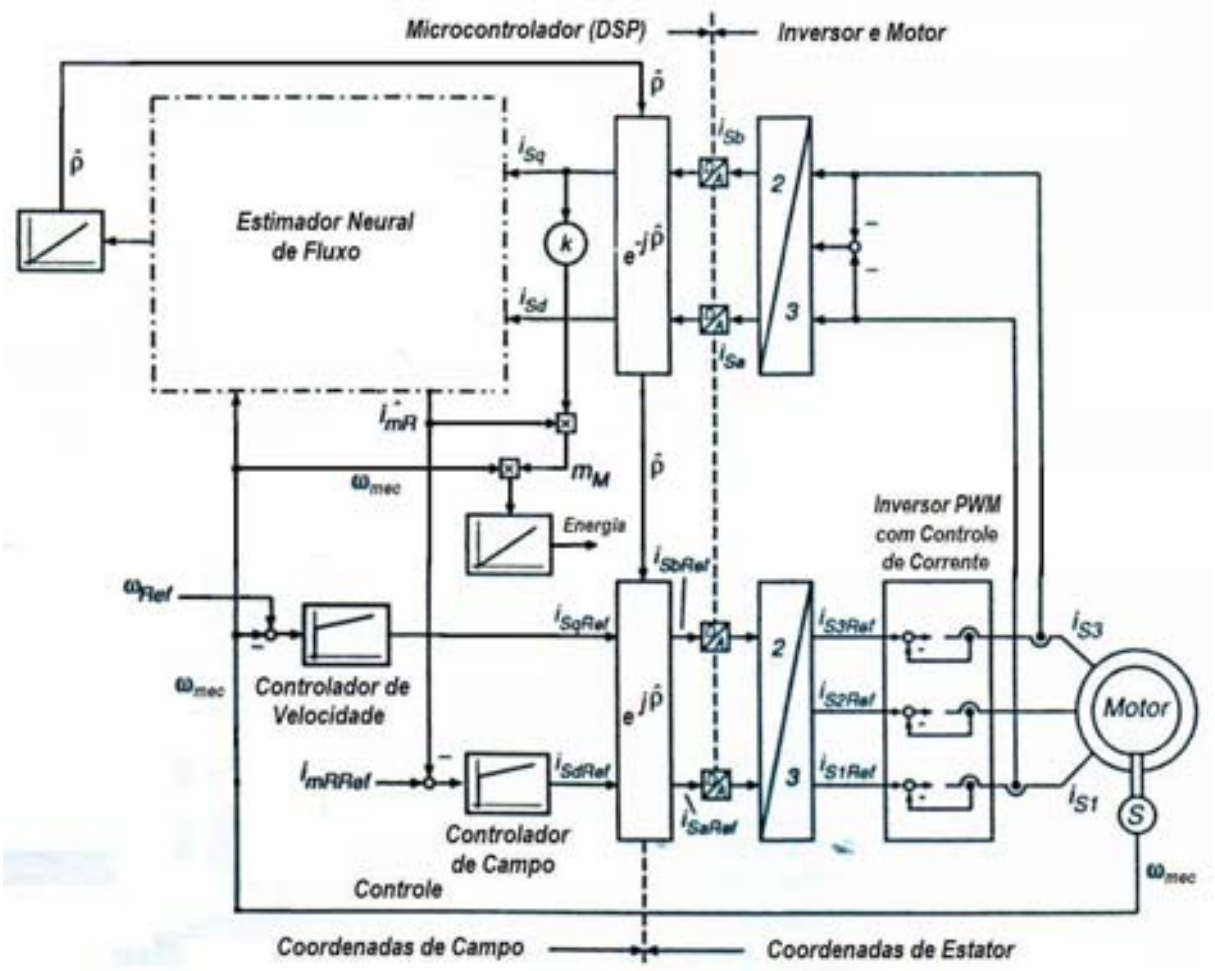


Fig. 3.5. *Diagrama de blocos de um controlador vetorial de velocidade com estimação neural de fluxo.*

A partir desse modelo, idealizou-se o sistema de controle vetorial de velocidade. Todos os detalhes da implementação em DSP do sistema completo de controle estão descritos no Capítulo 4.

3.4 Implementação do Estimador Neural de Fluxo

Para simplificar a implementação digital e minimizar a carga computacional do sistema, o estimador neural foi implementado através de duas redes neurais de múltiplas camadas do tipo *feedfoward* com funções de ativação lineares.

O aprendizado foi do tipo supervisionado através do modelo da máquina com parâmetros nominais e o processo de treinamento foi realizado com os dados reais oriundos dos ensaios apresentados no Capítulo 2.

As redes neurais implementadas executam a estimação independente da velocidade angular do fluxo $\frac{d\rho}{dt}$ e da corrente de magnetização i_{mR}.

3.4.1 Treinamento do Estimador Neural de Fluxo do Rotor

O treinamento das redes neurais foi realizado a partir de dados obtidos utilizando o modelo convencional apresentado no Capítulo 2 com alguns parâmetros nominais do motor de indução apresentados no apêndice A.

O processo de treinamento consistiu em aplicar variações na referência de velocidade, de maneira a permitir que a rede se adaptasse a diversas condições de funcionamento. Como ferramenta de treinamento, utilizou-se a toolbox *neural network* do MATLAB®.

As redes neurais implementadas são redes de três camadas, sendo: uma camada de entrada, uma camada escondida e uma camada de saída. Todos os neurônios das redes implementadas executam funções lineares.

A primeira rede implementada é responsável pela estimação da velocidade angular do fluxo do rotor, $\frac{d\hat{\rho}}{dt}$, a qual é integrada para a obtenção da posição angular do vetor de fluxo do rotor, $\hat{\rho}$. Suas entradas são as correntes i_{Sd} e i_{Sq} e a velocidade mecânica *velorads*, em rad/s, atuais e anteriores. Sua estrutura está mostrada na Fig. 3.6.

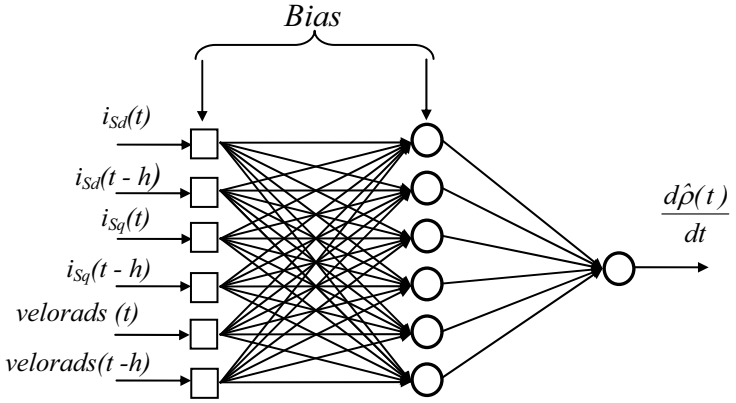


Fig. 3.6. *Diagrama da rede neural para estimação de* $\frac{d\rho}{dt}$.

A segunda rede implementada é responsável pela estimação da corrente de magnetização, $\hat{i}_{mR}$, cujas entradas são i_{Sd} atual e anterior e $\hat{i}_{mR}$ anterior, como mostra a Fig. 3.7.

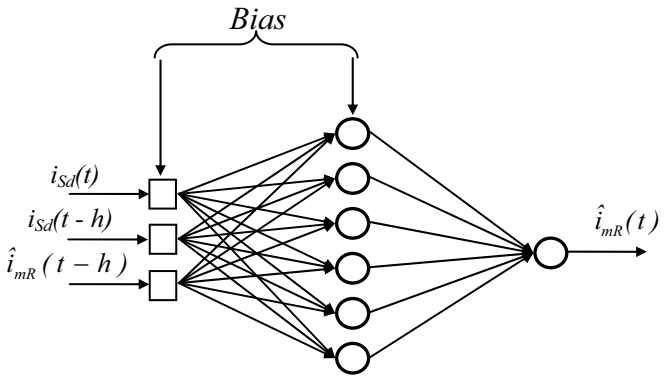


Fig. 3.7. *Diagrama da rede neural para estimação de* i_{mR}.

Conforme colocado, para o treinamento das redes projetadas foram utilizados os dados oriundos dos ensaios apresentados no capítulo 2.

Após o processo de treinamento, as curvas de erro convergiram para as metas desejadas como mostram as Figs. 3.8 e 3.9. Estas metas de erros foram de (10^{-3}) e (10^{-8}) para a redes de estimação de $\frac{d\rho}{dt}$ e de i_{mR}, respectivamente. A escolha destas metas foi feita empiricamente através de vários ensaios.

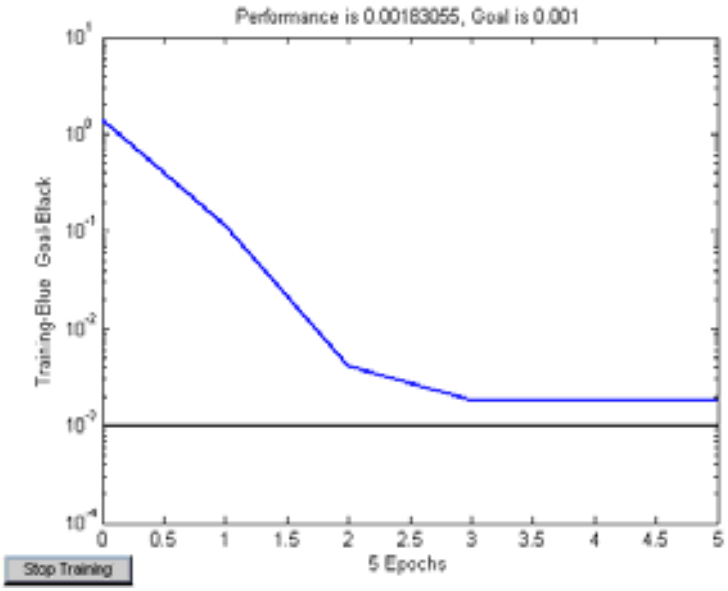


Fig. 3.8. *Curva de erro de treinamento para o estimador de* $\frac{d\rho}{dt}$.

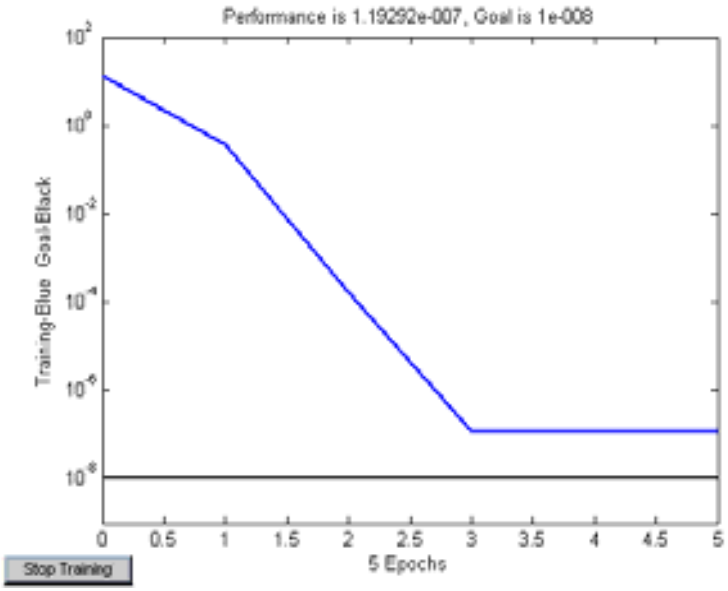


Fig. 3.9. *Curva de erro de treinamento para o estimador de* i_{mR}.

A eficiência das redes neurais implementadas pode ser comprovada nos resultados obtidos mostrados nas Figs. 3.10, 3.11 e 3.12 para o Ensaio 1 do Capítulo 2.

As variáveis de saída do modelo convencional estão representadas pelas linhas em azul ou preta e as variáveis de saída das redes neurais estão representadas em vermelho ou verde.

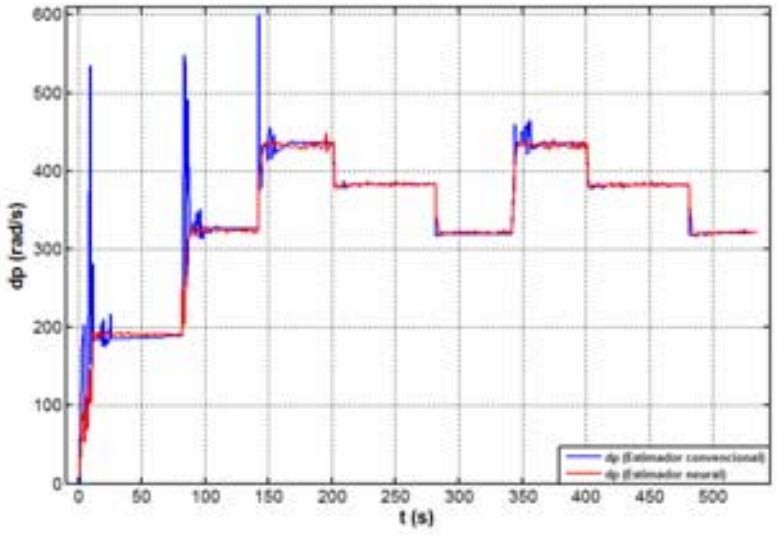


Fig. 3.10. *Velocidade angular estimada* $\frac{d\hat{\rho}}{dt}$ *do modelo com estimador convencional e com estimador neural.*

Na Fig. 3.10, observa-se que a velocidade angular do fluxo $\frac{d\hat{\rho}}{dt}$ gerada pelo estimador neural além de seguir a referência desejada apresenta melhoria em relação aos distúrbios gerados pelo estimador convencional.

Os respectivos comportamentos das velocidades angulares estimadas pelos estimadores convencional e neural se refletem diretamente na geração das correntes i_{Sd}, $\hat{i}_{mR}$ e i_{Sq} apresentadas nos gráficos da Fig. 3.11. Nestes gráficos, vê-se que nos transitórios as correntes geradas pelo estimador neural atingem valores mais elevados e sua chegada ao valor de regime se dá de forma mais suave e sem os chaveamentos que existem com o estimador convencional. Em se tratando de valores, as correntes de campo e de magnetização resultantes da utilização do estimador neural são maiores que as do estimador convencional. Já para a corrente de torque, fazendo a mesma comparação, esses valores são menores.

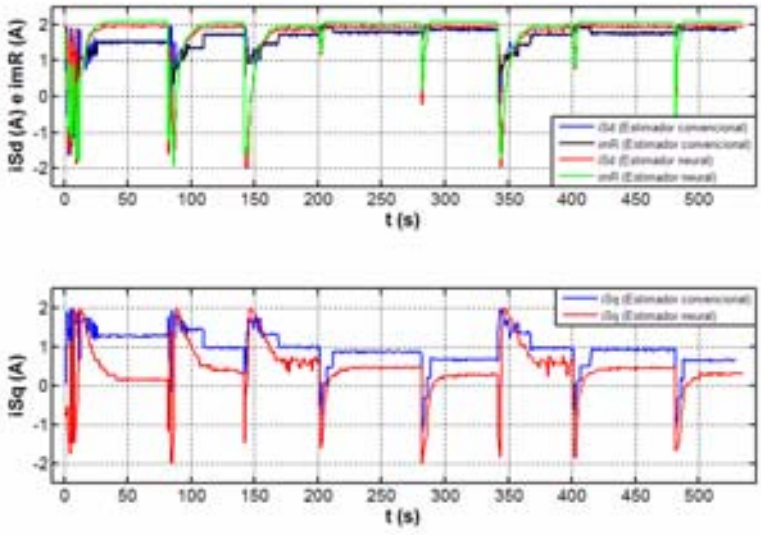


Fig. 3.11. *Correntes* i_{Sd}, $\hat{i}_{mR}$ e i_{Sq} *do modelo com estimador convencional e com estimador neural.*

O comportamento da velocidade gerada pelo modelo orientado pelos estimadores convencional e neural está mostrado na Fig. 3.12. Neste gráfico, a velocidade resultante da utilização do estimador neural praticamente segue a velocidade obtida com o uso do estimador convencional.

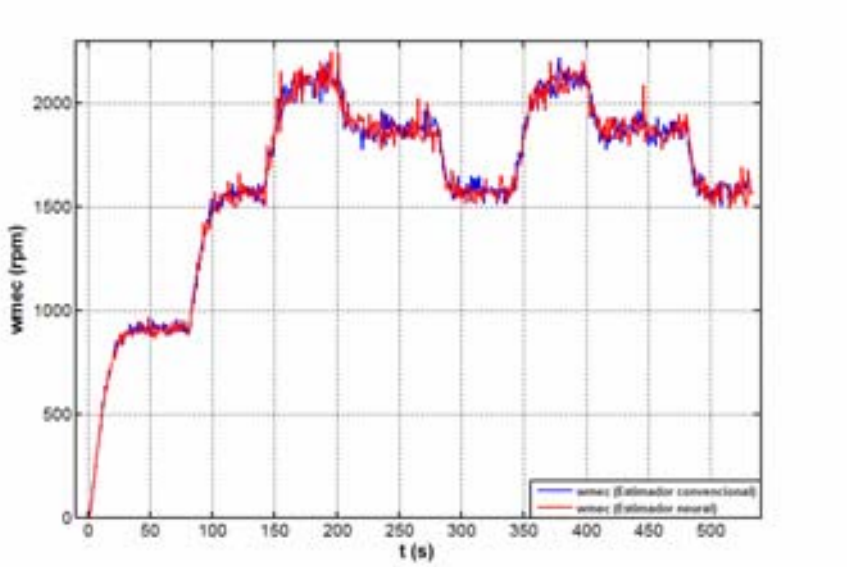


Fig. 3.12. *Velocidade mecânica do modelo com estimador convencional e com estimador neural.*

Para o Ensaio 2, os resultados são os apresentados nas Figs. 3.13, 3.14 e 3.15.

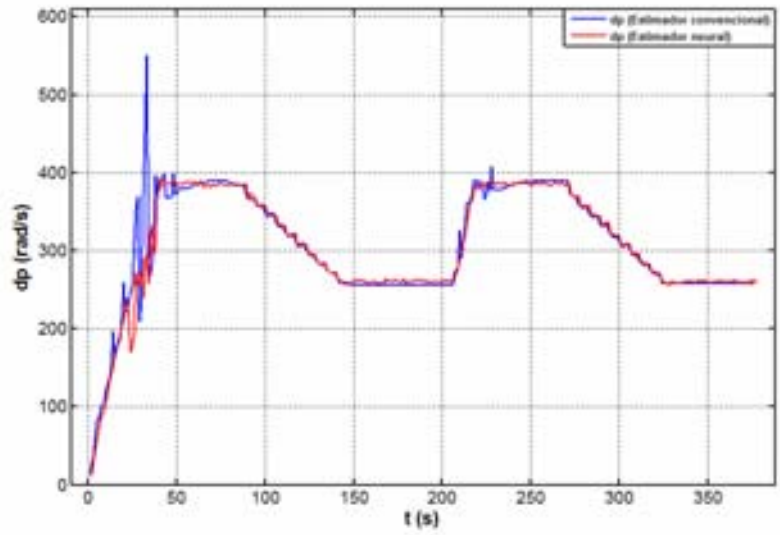


Fig. 3.13. *Velocidade angular estimada* $\frac{d\hat{\rho}}{dt}$ *do modelo com estimador convencional e com estimador neural.*

Da mesma forma que no Ensaio 1 apresentado no Capítulo 2, a velocidade angular do fluxo segue a referência e apresenta melhorias quanto aos distúrbios gerados com o uso do estimador convencional.

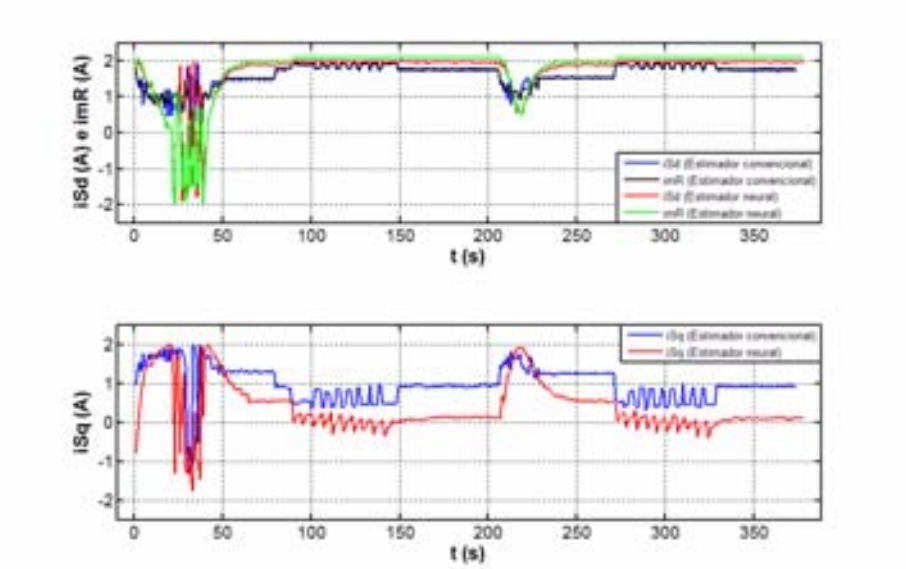


Fig. 3.14. *Correntes* i_{Sd}, $\hat{i}_{mR}$ e i_{Sq} *do modelo com estimador convencional e com estimador neural.*

Similarmente ao mostrado na Fig. 3.11, as correntes apresentadas na Fig. 3.14 praticamente possuem as mesmas características.

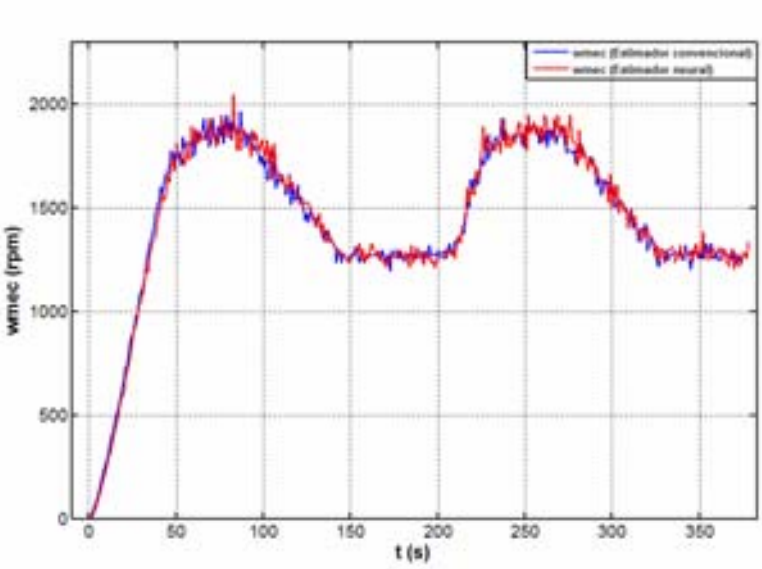


Fig. 3.15. *Velocidade mecânica do modelo com estimador convencional e com estimador neural.*

Igualmente ao que se vê na Fig. 3.12, na Fig. 3.15 observamos que a velocidade resultante da utilização do estimador neural praticamente coincide com aquela produzida pelo estimador convencional.

Após a comprovação da eficácia na etapa de treinamento das redes propostas, fez-se a conversão dos pesos e equações para a aritmética de ponto fixo para possibilitar a implementação em DSP do algoritmo de estimação neural do fluxo do rotor e do controle

vetorial de velocidade para o motor de indução. As equações que executam os cálculos das saídas das redes propostas estão descritas no Capítulo 4 e os pesos obtidos após o treinamento estão apresentados no apêndice B.

3.5 Conclusões

De acordo com as aplicações citadas e resultados de ensaios apresentados, foi possível constatar a grande potencialidade das redes neurais artificiais aplicadas na estimação de fluxo no controle das máquinas de indução.

Esta potencialidade motivou a implementação do sistema de controle vetorial de velocidade com estimação neural de fluxo do rotor para as máquinas de indução em substituição aos estimadores convencionais baseados no modelo inverso.

Enfim, a implementação da estimação neural de fluxo do rotor aplicado ao sistema de controle vetorial de velocidade para a máquina de indução teve o objetivo principal de aliar o que há de mais moderno e eficiente nas áreas de controle de máquinas elétricas e inteligência artificial.

Capítulo 4
Implementação do Controle Vetorial de Velocidade

4.1 Introdução

O sistema de controle de velocidade implementado neste trabalho alia o uso das eficientes técnicas de controle vetorial à estimação neural de fluxo do rotor.

Todo o controle do sistema está implementado em um DSP (*Digital Signal Processor*), que afora as desvantagens que ele possua, suas características de alto desempenho e capacidade de processamento tem sedimentado sua aplicação nas mais diversas áreas de pesquisas como controle de máquinas, telecomunicações e processamento de imagens (Stronach, 1998).

Apesar do foco deste trabalho ser a estimação neural de fluxo aplicada ao controle vetorial de velocidade de um motor de indução, a versatilidade do hardware implementado permite sua comparação com outros tipos de estimadores como, por exemplo, os baseados no modelo inverso da máquina. A opção de estimar a posição e a magnitude do fluxo do rotor se justifica pela minimização dos custos de implementação do sistema e também por que o posicionamento de sensores de fluxo no interior da máquina apresenta uma dificuldade considerável.

A implementação digital do sistema de controle buscou uma comparação direta entre os respectivos desempenhos da máquina sob orientação do estimador convencional de fluxo (baseado no modelo com parâmetros associados a dados de ensaios) e sob orientação do estimador neural de fluxo.

Nas seções seguintes faz-se uma descrição detalhada de todo o sistema de controle proposto. Esta descrição trata do projeto e execução de cada elemento do sistema e suas respectivas funções no desempenho global do mesmo.

4.2 Descrição do Sistema de Controle

O sistema de controle do motor de indução divide-se em duas partes constituídas respectivamente pelos estágios de controle vetorial da velocidade e controle das correntes em

suas bobinas. O controlador de velocidade é responsável pela geração das correntes de referências para o controle das correntes das bobinas do estator. Após a ação do controle das correntes de cada bobina, geram-se as saídas PWM para os dois inversores trifásicos, cujas saídas são aplicadas aos enrolamentos da máquina.

O estimador neural de fluxo do rotor implementado utiliza como entradas as correntes $d-q$ do estator e a velocidade mecânica rotacional disponibilizada por um sensor óptico.

O diagrama geral do sistema implementado está mostrado na Fig. 4.1.

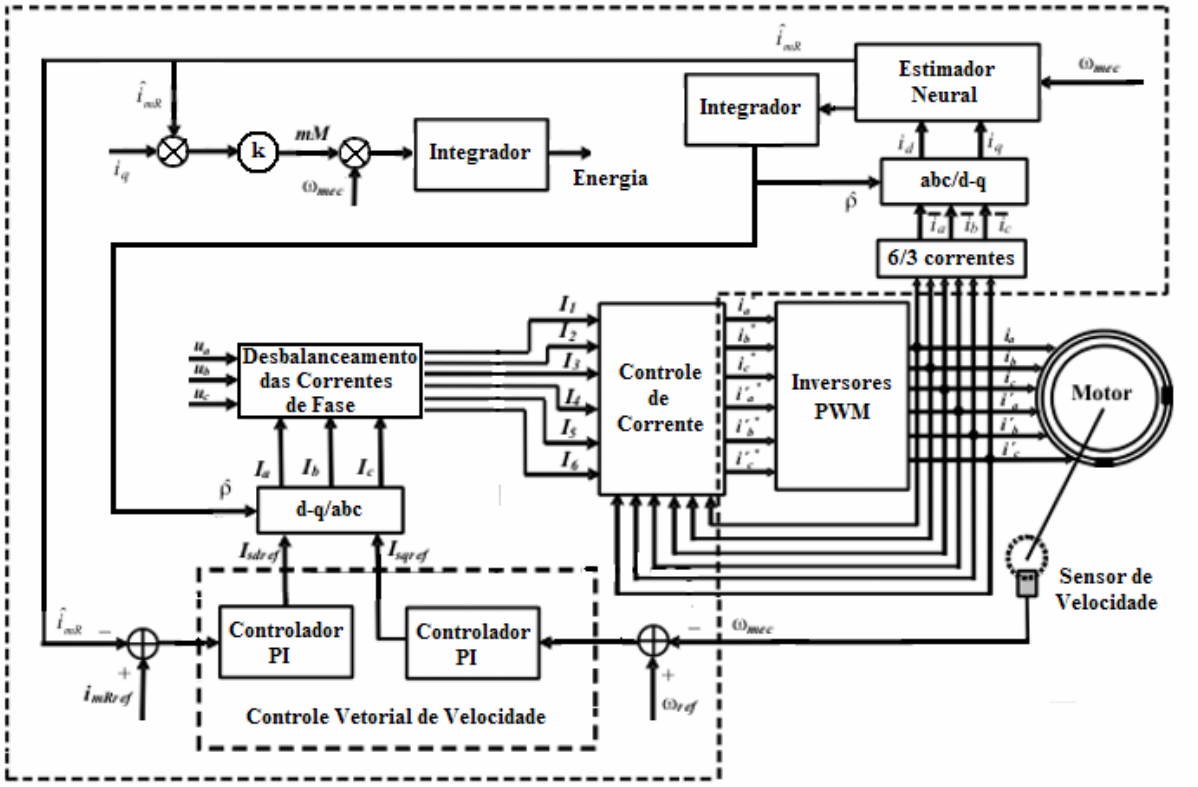


***Fig. 4.1**. Diagrama geral do sistema de controle implementado.*

Os sinais de velocidade rotacional, torque elétrico $m_M(k)$, correntes $i_{Sd}(k)$, $\hat{i}_{mR}(k)$ e $i_{Sq}(k)$, são monitorados por um sistema de supervisão implementado no ambiente do software LabView®, o qual disponibiliza saídas gráficas e numéricas dos sinais da velocidade mecânica $velorads(k)$, das correntes $i_{Sd}(k)$, $\hat{i}_{mR}(k)$ e $i_{Sq}(k)$, e do torque elétrico $m_M(k)$.

O DSP utilizado é um processador de 32 bits operando a 150 MHz e possui diversos recursos com canais Analógicos/Digitais, saídas PWM e digitais, além de portas seriais e paralelas para a comunicação com computadores PC (Spectrum, 2003). Toda a programação do sistema foi realizada na linguagem C com aritmética de ponto fixo.

Nas subseções seguintes estão descritos detalhes de implementação e funcionamento de cada bloco relacionado com as malhas de controle.

4.2.1 Estágio de Controle de Velocidade

O estágio de controle vetorial de velocidade é composto por dois controladores do tipo PI (Propocional-Integral) digitais. Um deles é responsável pelo controle específico de velocidade. Sua ação tem como saída, a corrente de referência $i_{Sqref}(k)$. O segundo controlador executa o controle da corrente de magnetização e é responsável pela geração da corrente de campo $i_{Sdref}(k)$.

Para a execução eficiente do controle do motor, a sintonia do controlador de velocidade deve ser executada independentemente do controlador de corrente de magnetização. As equações gerais dos controladores do tipo PI digitais implementados em todo o sistema são as seguintes:

- Cálculo do erro

$$erro(k) = I(k) - y(k) \tag{4.1}$$

- Ação Proporcional

$$P(k) = K_P.erro(k) \tag{4.2}$$

- Ação Integral

$$I(k) = I(k-1) + K_I.erro(k) \tag{4.3}$$

- Ação de controle para os Controladores PI

$$u(k) = P(k) + I(k) \tag{4.5}$$

Em que $y(k)$ é a variável de controle e $u(k)$ é a ação de controle aplicada no intervalo de tempo da $k-$ésima amostra e K_P e K_I são, respectivamente, os ganhos proporcional e integrativo dos controladores do sistema.

Para a execução do controle de velocidade utiliza-se a velocidade lida por um sensor óptico acoplado ao eixo da máquina. Durante o seu funcionamento geram-se pulsos quadrados que são enviados diretamente para o canal digital de captura do DSP. A partir da contagem dos pulsos capturados, calcula-se a velocidade mecânica em rad/s.

Para facilitar o ajuste dos controladores que compõem o estágio de controle de velocidade foi implementado um filtro digital para tratar o ruído inerente à saída do sensor óptico confeccionado. A aplicação deste filtro também melhorou as respostas dos estimadores de fluxo estudados.

O protótipo utilizado em verdade é uma máquina de indução trifásica sem mancais com bobinado dividido, conforme se vê em Paiva (2007), sendo uma de suas principais características a semelhança com uma máquina de indução trifásica convencional.

Para o contexto deste trabalho ela foi então preparada para funcionar como uma máquina de indução trifásica convencional, e dessa forma o desbalanceamento das correntes de fase que se vê na Fig. 4.1 deixa de existir. Assim sendo, as ações de controle $u_a(k)$, $u_b(k)$ e $u_c(k)$ são feitas iguais a zero e portanto:

$$\begin{bmatrix} I_1(k) = \\ I_2(k) = \\ I_3(k) = \\ I_4(k) = \\ I_5(k) = \\ I_6(k) = \end{bmatrix} = \begin{bmatrix} I_a(k) \\ I_b(k) \\ I_c(k) \\ I_a(k) \\ I_b(k) \\ I_c(k) \end{bmatrix} \tag{4.6}$$

Estas correntes de referência são aplicadas aos controladores individuais das correntes de cada meia bobina das fases do estator.

4.2.2. Estágio de Controle de Corrente

Para o controle de corrente foram implementados seis controladores digitais idênticos do tipo PI (Proporcional-Integrativo), sendo um para cada bobina do estator. Estes controladores utilizam as correntes de referência geradas pelo controlador de velocidade as quais são comparadas com as correntes lidas diretamente dos sensores de correntes, e os respectivos erros são processados digitalmente pelo DSP.

As saídas do DSP são sinais do tipo PWM (*Pulse With Modulation*) modulados pelos sinais oriundos dos controladores de corrente. Por fim, os sinais PWM são aplicados aos inversores trifásicos que alimentam individualmente as bobinas do estator.

4.2.3. Estimadores de Fluxo do Rotor

O sistema permite uma comparação entre diversos tipos de estimadores. Neste trabalho foram implementados e estudados desempenhos dos estimadores convencional e neural de fluxo do rotor aplicados ao controle vetorial de velocidade. A discretização e implementação destes estimadores estão descritas nas seções seguintes.

4.2.3.1. Estimador Convencional de Fluxo

O estimador convencional de fluxo foi desenvolvido a partir do modelo do motor de indução por correntes impostas em coordenadas $d-q$ com referência no fluxo do rotor, cujo diagrama de blocos está mostrado na Fig. 2.3. A sua implementação digital em aritmética de ponto fixo necessita da transformação numérica dos parâmetros e estados da máquina para números no formato Q_i, o qual facilita divisões por deslocamentos de bits. Esta representação foi realizada da seguinte forma:

$$N_i = N * 2^i \tag{4.7}$$

em que i é o numero de bits, N é o número a ser transformado e N_i é o número no formato Q_i.

A escolha do número de bits levou em consideração a precisão desejada para cada grandeza discretizada.

A partir do modelo da máquina apresentado no Capítulo 2 (Fig. 2.2), chegou-se às seguintes equações discretizadas:

- *Equações elétricas:*

$$d_{imR}(k) = (i_{Sd}(k) - \hat{i}_{mR}(k)).K_r \tag{4.8}$$

$$\hat{i}_{mR}(k) = \hat{i}_{mR}(k-1) + d_{imR}(k) >> 5 \tag{4.9}$$

$$d\hat{\rho}(k) = np.velorads(k) + \frac{i_{Sq}(k).K_i}{\hat{i}_{mR}(k)} \tag{4.10}$$

$$\hat{\rho}(k) = \hat{\rho}(k-1) + K_w.d\hat{\rho}(k) >> 8 \tag{4.11}$$

$$m_M(k) = \left(K_M.\hat{i}_{mR}(k).i_{Sq}(k)\right) >> 10 \tag{4.12}$$

As constantes presentes nestas equações foram definidas da seguinte forma:

$$K_r = \frac{h}{T_R}.2^5 \approx 1 \tag{4.13}$$

$$K_M = k.I_{nom}^2.2^{10} \approx 280 \tag{4.14}$$

$$K_i = \frac{1}{T_R} \approx 70 \tag{4.15}$$

$$K_w = \left[h.2^8.\left(\frac{2^{10}}{2.\pi}\right)\right] \approx 21 \tag{4.16}$$

A corrente nominal da máquina I_{nom} é de 1,0 A e serviu de base para todos os valores inteiros da discretização do sistema. Os demais parâmetros foram definidos no Capítulo 2.

- *Equações mecânicas:*

$$m_L(k) = D.m_M(k) \tag{4.17}$$

$$d\omega_{mec}(k) = (m_M(k) - m_L(k)).K_j \tag{4.18}$$

$$\omega_{mec}(k) = n_p.\omega_{mec}(k-1) + d\omega_{mec}(k) >> 8 \tag{4.19}$$

$$velorads(k) = \omega_{mec}(k) >> 8 \tag{4.20}$$

$$K_j = \frac{12.h}{J} \approx 1 \tag{4.21}$$

em que D é o fator de carga, K_j é a constante paramétrica e $velorads(k)$ é a velocidade da máquina em rad/s. Como a velocidade mecânica, em rads/s, foi disponibilizada diretamente do sensor óptico, as equações mecânicas serviram apenas para ajustes *off-line* dos ganhos do controlador. Assim, estas equações não estão implementadas no programa de controle do DSP.

4.2.3.2. Estimador Neural de Fluxo

O estimador neural de fluxo implementado é composto por duas redes multicamadas "*feedforward"* e utiliza neurônios que implementam funções lineares. Conforme discutido no Capítulo 3, a escolha deste tipo de rede teve como objetivo a minimização da carga computacional do DSP sem que houvesse perdas significativas de desempenho.

A implementação em DSP do estimador neural de fluxo do rotor desenvolvido no Capítulo 3 foi realizada no formato de representação numérica Q_{12}. Este formato proporcionou perdas mínimas para os valores dos pesos das redes de $d\hat{\rho}(k)$ e $\hat{i}_{mR}(k)$.

Para uniformizar a faixa dos valores das entradas das redes implementadas com os valores dos pesos, foi implementada uma rotina de normalização das mesmas em valores inteiros compreendidos entre –4096 a 4096. Esta normalização é executada através da seguinte expressão:

$$x_{norm}(k) = \frac{((x_i(k) - x_{min}) * 2 * 4096)}{(x_{max} - x_{min})} - 4096 \tag{4.22}$$

em que $x_{norm}(k)$ é o valor normalizado da entrada, $x_i(k)$ é o valor atual da entrada i e x_{max} e $x_{\min}$ são, respectivamente, os valores máximos e mínimos de cada uma das entradas.

Após a execução do cálculo das saídas de cada rede, faz-se a desnormalização das mesmas através da expressão:

$$y_{desnorm}(k) = ((y(k) + 4096) >> 1) * (y_{max} - y_{min})) >> 12 + y_{min} \tag{4.23}$$

em que $y_{desnorm}(k)$ é o valor desnormalizado da saída atual $y(k)$ e y_{max} e y_{min} são, respectivamente, os valores máximos e mínimos de cada uma das saídas das redes.

A topologia das redes implementadas é composta por uma camada de entrada, uma camada escondida com seis neurônios e uma camada de saída com um neurônio. O vetor de entrada da rede que fornece $d\hat{\rho}(k)$ é definido como:

$$x_{d\rho}(k) = \begin{bmatrix} i_{Sd}(k) \\ i_{Sd}(k-1) \\ i_{Sq}(k) \\ i_{Sq}(k-1) \\ velorads(k) \\ velorads(k-1) \end{bmatrix} \tag{4.24}$$

A saída da camada escondida é dada por:

$$y_1(k) = b_1^T + (w_i^T * x_{d\rho}(k)) >> 12 \tag{4.25}$$

em que b_1^T é o vetor transposto de *bias* e w_i^T é a matriz transposta de pesos da camada escondida. Tomando a saída $y_1(k)$ como entrada da camada de saída $y_2(k)$ tem-se:

$$y_2(k) = b_2^T + (w_1^T * y_1(k)) >> 12 \tag{4.26}$$

em que b_2^T é o vetor transposto de *bias* e w_1^T é a matriz transposta de pesos da camada da saída. Fazendo $d\hat{\rho}(k) = y_2(k)$, integra-se o ângulo estimado através da mesma expressão (Eq. 4.11) utilizada para o estimador convencional de fluxo.

A rede que estima a corrente de magnetização $\hat{i}_{mR}(k)$ tem estrutura similar à anterior, porém o seu vetor de entradas é dado por:

$$x_{imR}(k) = \begin{bmatrix} i_{Sd}(k) \\ i_{Sd}(k-1) \\ i_{mR}(k-1) \end{bmatrix} \tag{4.27}$$

As demais expressões das camadas internas são idênticas às Eqs. 4.25 e 4.26, porém, as matrizes de pesos w_i e w_1 e os vetores das *bias* b_1 e b_2 são específicos para esta rede.

4.3. Descrição do Programa de Controle

O programa de controle do sistema foi desenvolvido na linguagem C dentro de um ambiente próprio do DSP. É possível dividi-lo em três blocos de operações bem definidas, conforme Fig. 4.2.

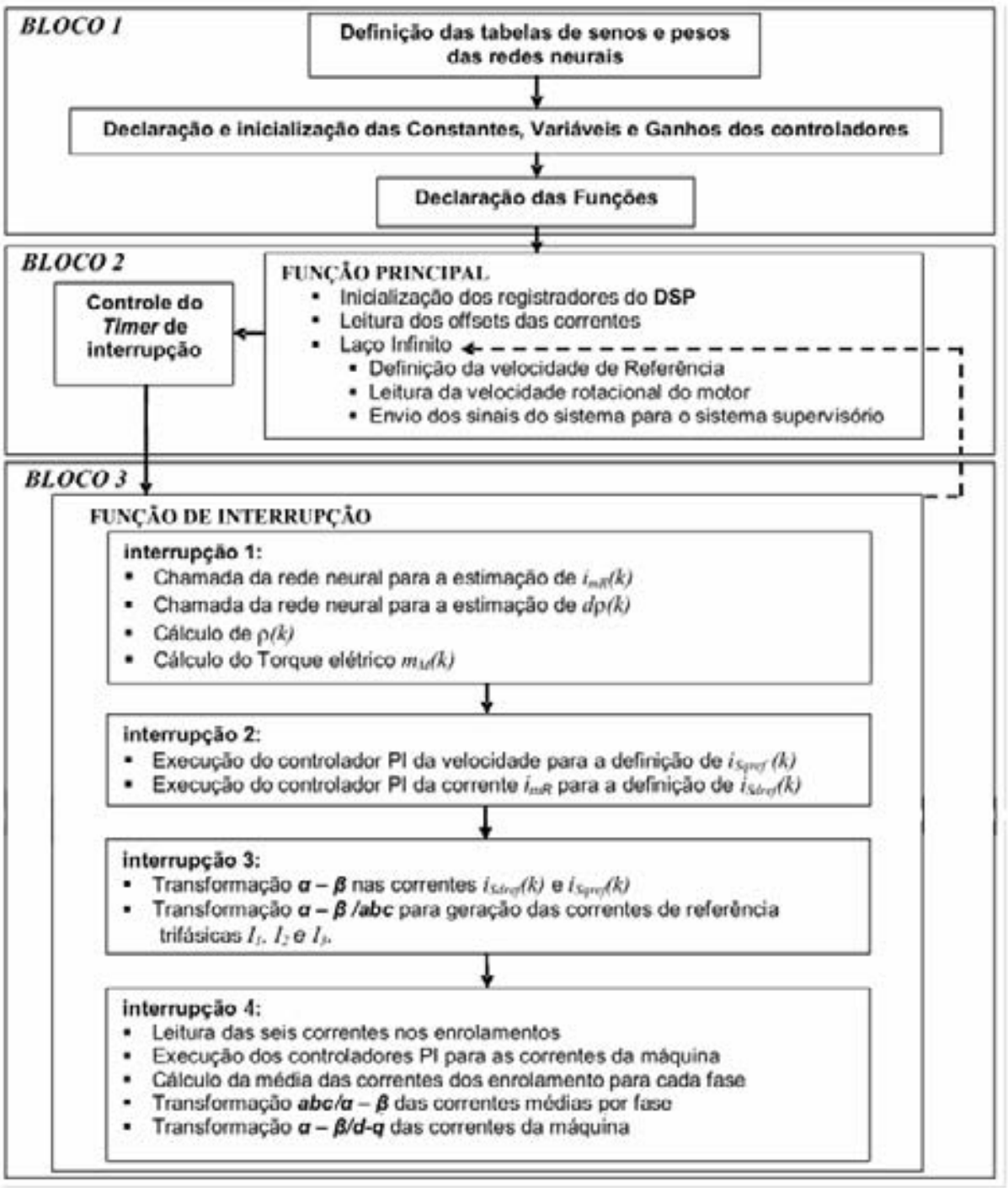


Fig. 4.2. *Fluxograma do programa de controle.*

O primeiro bloco executa as definições e declarações dos seguintes elementos do sistema: vetor de senos, matrizes de pesos das redes, constantes, ganhos dos controladores, variáveis globais e funções auxiliares do programa.

O segundo bloco está definido na função principal do programa. Neste bloco estão as operações executadas uma única vez como: inicialização de registradores e variáveis e leitura dos *offsets* de corrente. Executa-se ainda neste bloco um laço infinito que: lê o estado da chave que habilita os sinais PWM, faz a leitura da velocidade da máquina, em rad/s, e realiza a comunicação serial com o PC junto ao sistema de supervisão.

O terceiro bloco está alocado dentro da função de interrupção e executa todas as funções de controle do sistema.

Para a realização do controle global do sistema, utilizam-se os canais Analógicos/Digitais para leituras dos sinais de correntes oriundas dos sensores de efeito *Hall*. Para a leitura da velocidade rotacional, utiliza-se um canal digital de captura, o qual recebe pulsos gerados pelo sensor óptico.

Para a execução de todos os passos de controle do sistema foi escolhida uma freqüência de interrupção de 10 kHz, que coincide com a freqüência de chaveamento dos canais PWM. Essa escolha tomou como base o número de instruções possíveis de serem realizadas dentro do intervalo dessas interrupções (100 μs). Como o número de instruções necessárias para o controle geral do sistema é extenso, subdividiu-se essas instruções em blocos alocados em 4 interrupções subseqüentes. Cada um destes blocos é responsável pela execução de tarefas específicas. As tarefas executadas são basicamente os cálculos das equações de estimação e controle, leitura de sinais do sistema e envio dos sinais de referência de correntes para os canais PWM, cujos sinais controlam os inversores de freqüência. Desta forma, o laço de controle geral do sistema tem um passo de integração h de 500 μs, o qual foi utilizado para a definição das constantes definidas nas Eqs. 4.13, 4.16 e 4.21. Uma listagem das principais rotinas desenvolvidas para o programa de controle está disponibilizada no apêndice C.

4.4. Alimentação do Sistema

A fonte de potência que alimenta a máquina está dividida em duas partes distintas. A primeira parte é um retificador trifásico de onda completa, responsável pela carga do barramento de capacitores que alimenta o inversor de potência. A outra parte é composta pelo inversor de potência que alimenta individualmente cada meia-bobina da máquina.

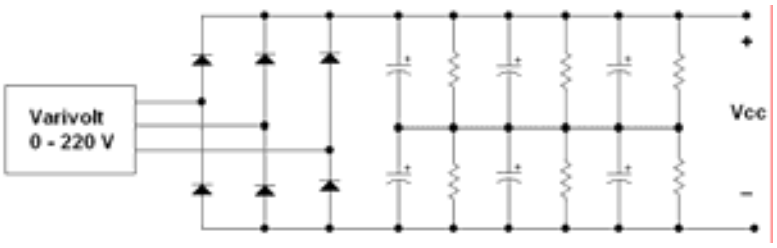


***Fig. 4.3**. Retificador Trifásico de Potência.*

Conforme Fig. 4.3, o retificador trifásico de potência tem como fonte primária um varivolt trifásico de 0 - 220 V.

O sistema conta ainda com outras duas fontes de corrente contínua simétricas reguladas de ±5V e ±15V responsáveis pela alimentação dos seguintes elementos do sistema:

- DSP;
- Circuito *driver* de disparo dos IGBTs;
- Sensores de correntes;
- Sensor de velocidade óptico.

Cada um destes elementos opera com tensões CC diferentes. Por esta razão, foram colocados reguladores de tensão específicos para cada componente do sistema (Castro, 2004).

A fonte de corrente alternada que alimenta a máquina é composta por dois inversores trifásicos montados com IGBTs, como mostra a Fig. 4.4.

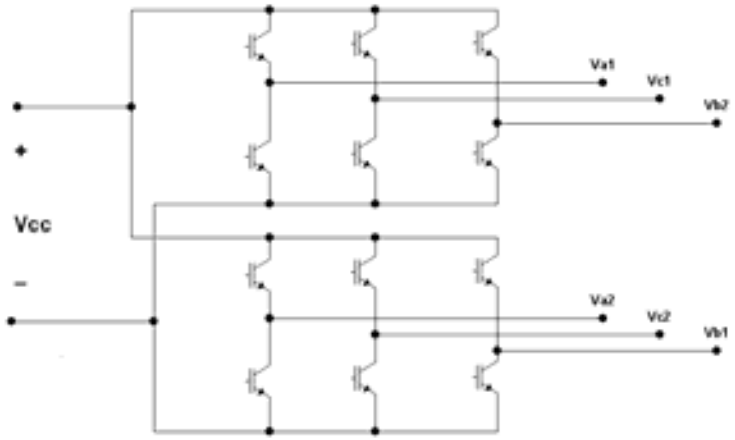


***Fig. 4.4**. Inversor de potência duplo trifásico.*

A segurança do sistema conta com um disjuntor trifásico, três fusíveis, lâmpadas indicadoras de operação internas e externas.

O painel de acionamento do sistema é composto por uma botoeira NA-NF que aciona um contactor trifásico, um voltímetro, um amperímetro, um botão de emergência e uma chave responsável por habilitar os pulsos PWM para os IGBTs.

4.5. Conclusões

De acordo com o conteúdo apresentado neste capítulo, alguns pontos importantes são enumerados depois da implementação do sistema de controle.

O sistema implementado permite a integração entre os controles de velocidade e de correntes, é bastante versátil, permitindo tanto a análise de diferentes estimadores de fluxo quanto o teste de diferentes tipos de controladores sem a necessidade de modificações no *hardware* desenvolvido. Além disso, o *hardware* e o *software* desenvolvidos utilizam os dispositivos e as técnicas mais modernas na área de controle das máquinas elétricas que são o DSP e o controle vetorial de fluxo aliado às técnicas de estimação neural de sistemas.

Pelas razões citadas, conclui-se que os principais objetivos de implementação propostos para o sistema de controle para a máquina de indução foram atingidos.

Capítulo 5
Resultados

5.1 Introdução

A implementação do sistema de controle vetorial de velocidade para um motor de indução foi realizada de forma que permitisse uma comparação direta entre os desempenhos dos estimadores de fluxo convencional e neural.

Depois da implementação prática do sistema de controle em DSP, foram realizados ensaios para ajustes e refinamento das respostas dos controladores de velocidade mecânica e corrente de cada bobina do motor de indução.

Todos os ensaios foram executados sob condições iguais para os dois estimadores em estudo, condições essas tais que:

- Não se aplicou carga ao sistema;
- Todos os controladores foram sintonizados empiricamente de modo a oferecer respostas rápidas e suaves de velocidade para o sistema;
- Os ganhos individuais de cada um dos controladores implementados foram mantidos.

A Execução dos ensaios foi dividida em dois perfis de variações de referência de velocidade. Estes perfis utilizaram variações de referência de velocidade em degraus e em rampas, ambos com valores constantes e variáveis. Para os perfis variáveis, foram implementadas variações ascendentes e descendentes.

5.2 Resultados Experimentais do Sistema em Malha Fechada

As subseções seguintes apresentam e analisam as respostas das principais variáveis de saída do sistema de controle, em função do tempo em segundos. Todas as variáveis são capturadas pelo sistema de supervisão desenvolvido no ambiente LabView®, a menos da energia consumida pelo sistema, que é calculada tomando-se a integração do produto da velocidade mecânica com o torque aplicado à máquina. No caso, o torque também é calculado

através do produto da constante k (Eq. 2.7) com as correntes de magnetização estimada e de quadratura mencionadas a seguir.

As variáveis capturadas em função do tempo são, respectivamente: a velocidade mecânica $\omega_{mec}(k)$, a corrente de campo $i_{Sd}(k)$, em conjunto com as correntes de magnetização estimada $\hat{i}_{mR}(k)$ e de quadratura $i_{Sq}(k)$, e a velocidade angular estimada $d\hat{\rho}(k)$.

5.2.1 Perfil 1: Variações de referência de velocidade em degraus

Neste perfil foram aplicadas diferentes referências de velocidade em degraus com o intuito de analisar o comportamento da máquina no regime transitório e em regime permanente para este tipo de situação.

A primeira variável apresentada na Fig. 5.1 é a velocidade mecânica $\omega_{mec}(k)$ em função do tempo.

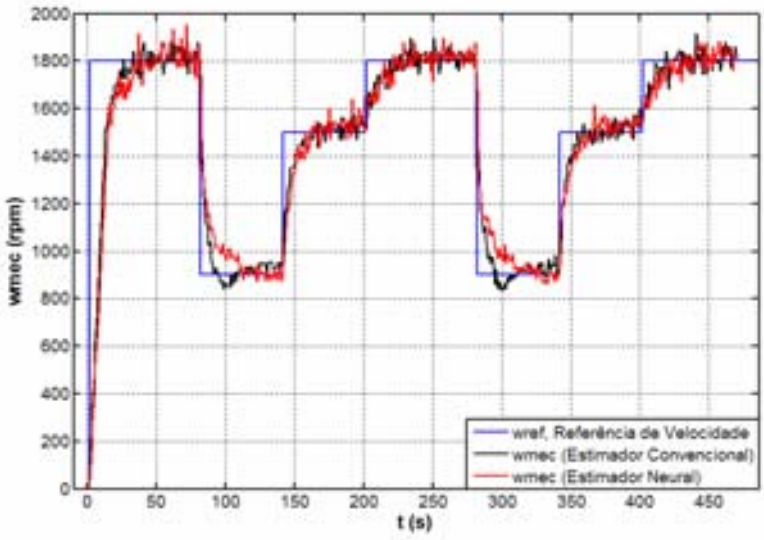


Fig. 5.1. *Velocidade mecânica em função do tempo para referências de velocidade em degraus.*

Observando o gráfico, vê-se que o comportamento da velocidade mecânica $\omega_{mec}(k)$ da máquina é ligeiramente mais rápido nos transitórios das baixas para as altas velocidades para a operação do sistema com o estimador convencional. Porém, nos transitórios das altas para as baixas velocidades, operando com o estimador neural esse comportamento é mais suave, ao passo que operando com o estimador convencional existe um pequeno *overshoot*. Em regime os dois estimadores se comportam de forma similar.

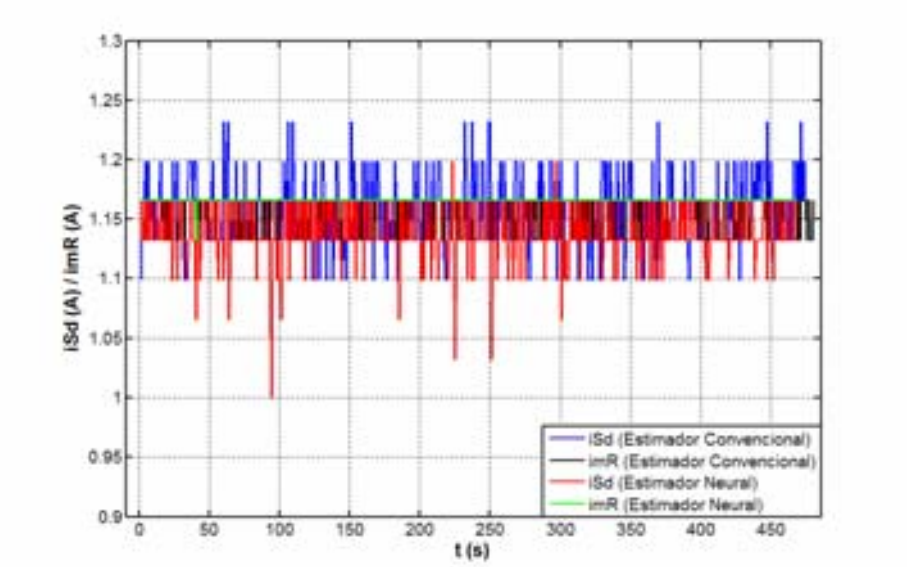


Fig. 5.2. *Correntes de campo e de magnetização.*

No gráfico da Fig. 5.2 vê-se que as correntes de campo $i_{Sd}(k)$ e de magnetização estimada $\hat{i}_{mR}(k)$ oscilam em torno da corrente de referência. Nota-se que essas oscilações são mais acentuadas para o sistema operando com o estimador convencional. Os valores destas correntes de referência de campo $i_{Sd}(k)$ e de magnetização estimada $\hat{i}_{mR}(k)$, responsáveis pela magnitude do fluxo magnético do rotor, foram tomados em função da limitação de torque da máquina, obrigando o sistema a operar com baixos valores de correntes de magnetização.

Na Fig. 5.3 seguinte são apresentadas as velocidades angulares estimadas $d\hat{\rho}(k)$.

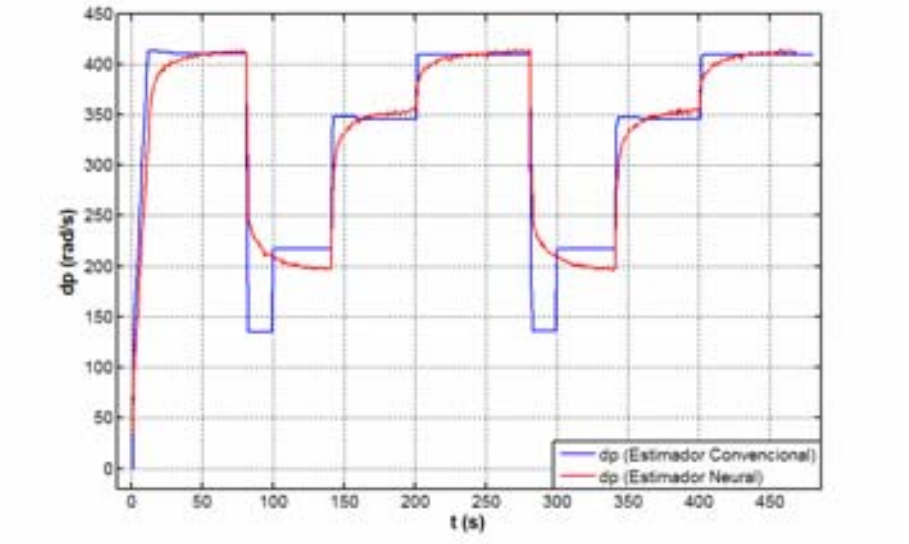


Fig. 5.3. *Velocidade angular do fluxo do rotor.*

Analisando o gráfico, observa-se que para a operação do sistema com o estimador convencional existem chaveamentos nos transitórios das altas para as baixas velocidades. Todavia, para a operação com o estimador neural, isso se dá de forma suave das altas para as baixas velocidades e vice-versa.

Concluindo a análise do comportamento do sistema para o perfil em questão, o cálculo de energia consumida durante sua operação para cada um dos estimadores foi realizado e está mostrado graficamente na Fig. 5.4.

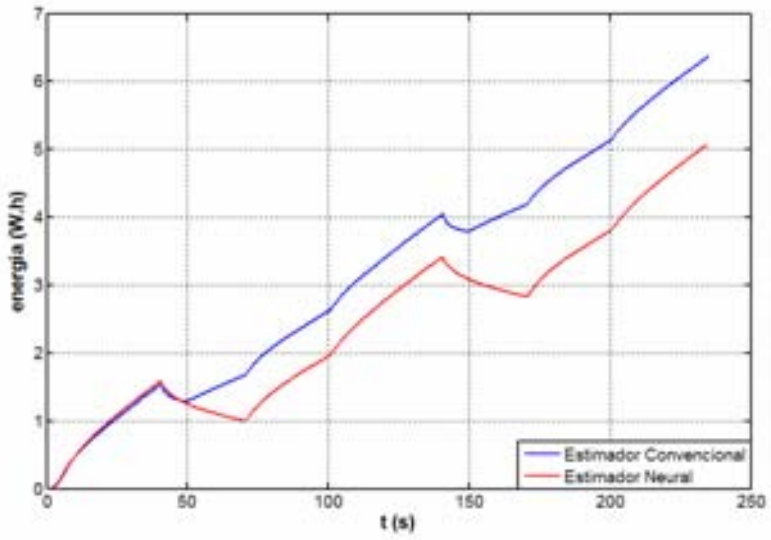


Fig. 5.4. *Energia consumida pelo sistema.*

De acordo com o gráfico, observa-se que quando o sistema opera com o estimador neural o consumo é menor comparado à operação com o estimador convencional.

5.2.2 Perfil 2: Variações de referência de velocidade em rampas

Para este segundo perfil, foram aplicadas diferentes referências de velocidade agora em rampas com o mesmo objetivo de se analisar o comportamento da máquina nos regimes transitório e permanente.

A velocidade mecânica está apresentada na Fig. 5.5. Conforme o gráfico, nas variações ascendentes e descendentes o comportamento de velocidade da máquina operando com o estimador convencional é ligeiramente mais rápido, sendo que antes que a velocidade se estabilize nas baixas velocidades existe um pequeno *overshoot*. Operando com o estimador neural, a resposta do sistema é mais suave nas variações descendentes. Em regime os dois estimadores se comportam de forma similar.

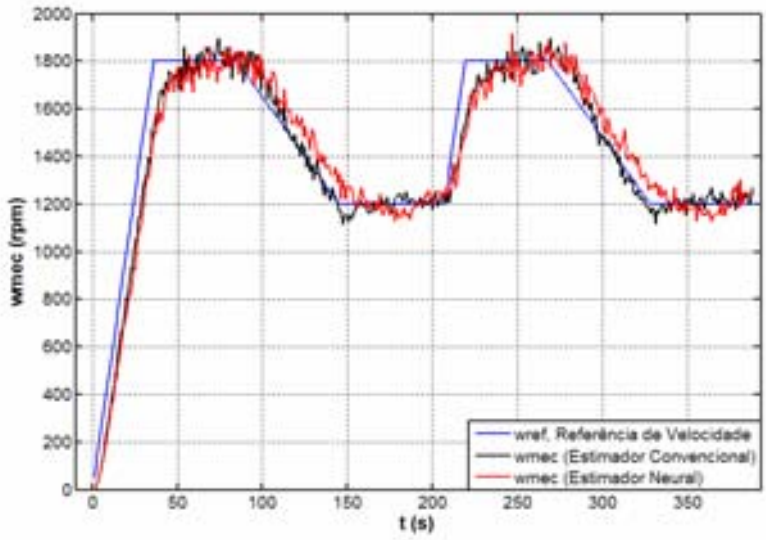


Fig. 5.5. *Velocidade mecânica em função do tempo para referências de velocidade em rampas.*

Da mesma forma que o apresentado no gráfico da Fig. 5.2, observa-se na Fig. 5.6 que as correntes de campo $i_{Sd}(k)$ e de magnetização estimada $\hat{i}_{mR}(k)$ também oscilam significativamente em torno da corrente de referência.

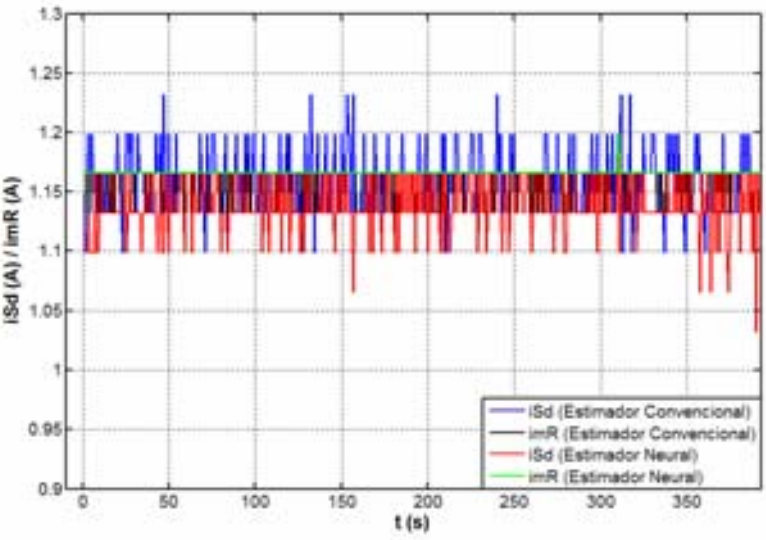


Fig. 5.6. *Correntes de campo e de magnetização.*

Na Fig. 5.7 estão mostrados a saída principal dos estimadores convencional e neural, as velocidades angulares estimadas $d\hat{\rho}(k)$.

Analisando o gráfico, observa-se que para o perfil em questão, operando com o estimador convencional, a velocidade angular chaveia em determinados pontos ao passar do tempo. Porém, operando com o estimador neural, isso ocorre, mas se dá de forma mais suave.

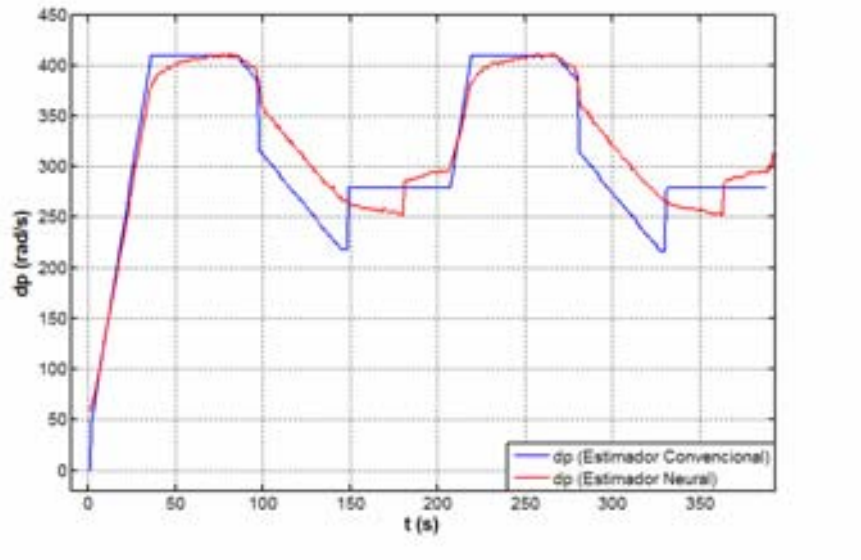


Fig. 5.7. *Velocidade angular de fluxo do rotor.*

Finalmente, concluindo a análise do comportamento do sistema para o perfil em análise, calculou-se a energia consumida durante sua operação para cada um dos estimadores e mostrou-se graficamente na Fig. 5.8.

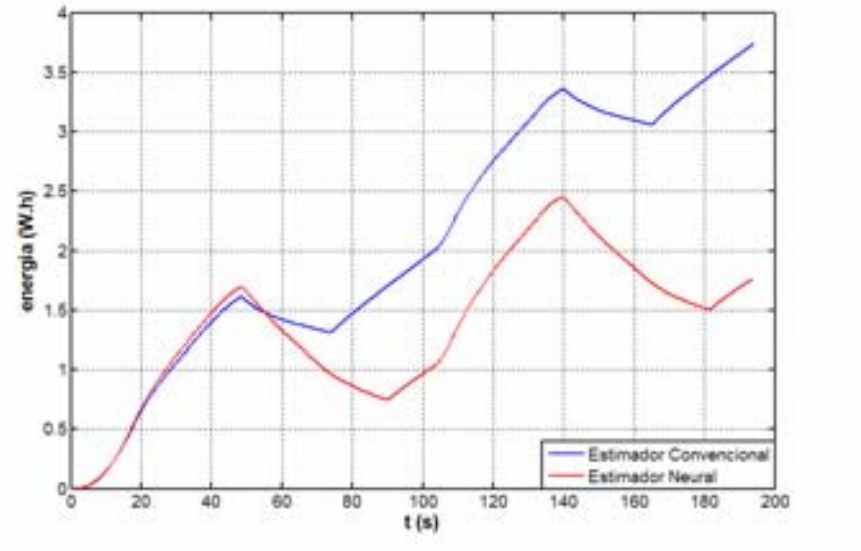


Fig. 5.8. *Energia consumida pelo sistema.*

Da mesma forma, observa-se no gráfico que quando o sistema opera com o estimador neural o consumo também é menor comparado à operação com o estimador convencional.

5.3 Limitações impostas pelo Sistema de Controle

Ao longo do desenvolvimento do sistema de controle foram observadas diversas limitações impostas por elementos do sistema e pela máquina utilizada, obtida de outra que possui características específicas (máquina de indução sem mancais). Estas limitações não permitiram a captura de resultados mais refinados. As principais limitações detectadas foram as seguintes:

- Devido à existência de dois conjuntos estator/rotor com peso elevado e os controles de velocidade mecânica terem sido implementados em apenas um destes conjuntos, obrigou-se todo o sistema de controle a conviver com uma elevada inércia do rotor. Esta limitação elevou consideravelmente os tempos de estabilização de velocidade mecânica da máquina.
- O sensor de velocidade utilizado foi construído de forma artesanal com algumas imperfeições no alinhamento junto ao eixo do rotor. Estas imperfeições geraram sinais com ruídos significativos, obrigando o uso de filtros digitais para tratar as variáveis dependentes da velocidade.

Enfim, para a obtenção de resultados experimentais significativos, foram necessários ajustes computacionais para se chegar a soluções que proporcionassem respostas de velocidades mecânicas satisfatórias sob orientação de ambos tipos de estimadores estudados.

5.4 Conclusões

Os resultados apresentados mostram que, apesar das limitações de ordem prática impostas por elementos do sistema, o estimador neural proporcionou respostas de velocidade semelhantes às respostas proporcionadas pelo estimador convencional. Além disso, verifica-se que ao utilizar o estimador neural o consumo de energia é menor.

Esta comparação proporcionou conclusões importantes que reforçam a importância dos estimadores neurais no controle de plantas não-lineares, como as máquinas de indução.

Pelas razões descritas acima, o sistema de controle vetorial de velocidade do motor de indução utilizando estimação neural de fluxo se firma como mais uma aplicação bem sucedida das redes neurais artificiais aplicadas às máquinas elétricas.

Capítulo 6
Conclusões e Trabalhos Futuros

6.1 Conclusões Gerais

Fez parte do objetivo principal deste trabalho implementar e estudar comparativamente o desempenho dos estimadores convencional e neural de fluxo aliado à técnica de controle vetorial de velocidade em quadratura para um motor de indução.

A escolha das estratégias de estimação e controle baseou-se nas suas diversas aplicações bem sucedidas para a máquina de indução.

Ao finalizar a implementação prática de todo o sistema de controle em DSP e a captura de importantes resultados, foi possível chegar às seguintes conclusões gerais:

- O controle de velocidade é limitado a pequenas e suaves variações de referência de velocidade;
- Por se tratar de uma máquina com características não-lineares, as redes neurais artificiais implementadas foram treinadas para diversas condições, contemplando as não-linearidades e proporcionando a melhoria do desempenho do sistema;
- Ao se utilizar o estimador neural de fluxo no sistema de controle há uma diminuição no consumo de energia.

Enfim, apesar de todas as limitações práticas impostas pelos elementos do sistema e pelas características do motor utilizado, os resultados experimentais obtidos em tempo real contribuíram significativamente para reafirmar a importância das técnicas vetoriais de controle e dos estimadores neurais aplicados ao controle e acionamento das máquinas elétricas.

6.2 Perspectivas e Trabalhos Futuros

As pesquisas com as máquinas de indução vêm recebendo contribuições importantes nas áreas de modelagem e controle para a viabilização do seu uso em plantas que apresentem dificuldades e/ou alto custo.

Diante deste cenário, faz-se necessário desenvolver novos trabalhos com este tipo de máquina com o objetivo de otimizar os resultados obtidos ou mesmo de compará-los com outras estratégias de controle e estimação de plantas não-lineares.

Por essas razões, são propostas a seguir algumas idéias de modificação dos elementos do sistema e também de estudos com outras técnicas de controle e estimação de plantas não-lineares.

As principais propostas para trabalhos futuros são:

- A aplicação dos Filtros de Kalman para a estimação do modelo matemático deste tipo de máquina, permitindo uma comparação direta dos futuros resultados com os obtidos neste trabalho.
- A implementação de um sistema de controle vetorial de velocidade com referência no fluxo do estator ou entreferro que também possibilite uma comparação direta de desempenho com este trabalho.
- A implementação de um sistema de controle com técnicas do tipo *Sensorless* para eliminação de sensores e conseqüente minimização de custos.

Enfim, ao apresentar estas propostas, busca-se atingir algumas das diversas possibilidades de pesquisa com a máquina de indução, mantendo-a em sintonia com as estratégias mais eficientes e modernas na área de controle e acionamento de máquinas elétricas.

Diante de todo o conteúdo exposto neste documento, conclui-se que os principais objetivos almejados neste trabalho foram atingidos de forma consistente e objetiva.

Referências Bibliográficas

(Almeida, 1999) Almeida, P. R., Stephan, M., Branco, P. J. C., Suemitsu, W. I., *"Rotor Flux Angle Estimation Using Neural Networks"*, In: Electrimacs'99, 1999, Lisboa. Electrimacs'99, Vol. 1. pp. I-157-I-162, 1999.

(Barbi, 1985) Barbi, I., "*Teoria Fundamental de Motor de Indução*", Editora da UFSC, Florianópolis, 1985.

(Blaschke, 1972) Blaschke, F., "*The principle of field orientation as applied to new transvector closed-loop system for rotating field machines*", *Siemens Review* 39(5).

(Castro, 2004) Castro, F. E. F., *"Motor de Indução Trifásico sem Mancais com Bobinado Dividido: Otimização do Sistema de Posicionamento Radial"*, Dissertação de mestrado, PPgEE/UFRN, Natal-RN, 2004.

(Chen, 1994) Chen, T. C., Liaw, C. Y., *"Design of a Neural Fuzzy Controller For Induction Motor Speed Control"*, 20th International conference on Industrial Electronics, Control and Instrumetation, 1994. IECON ´94, Vol. 1, pp. 611-616, September 5-9, 1994.

(Ferreira, 2002) Ferreira, J. M. S., *"Proposta de Máquinas de Indução Trifásica sem Mancal com Bobinado Dividido"*, Dissertação de mestrado, PPgEE/UFRN, Natal-RN, 2002.

(Ferreira, 2007) Ferreira, J. M. S., *"Modelagem de Máquina de Indução Trifásica sem Mancais com Bobinado Dividido"*, Tese de doutorado, PPgEE/UFRN, Natal-RN, 2007.

(Furtunato, 2001) Furtunato, A. F. A., Araújo, A. D., Salazar, A. O., *"Implementação de um Controlador de Velocidade Usando modos Deslizantes Suaves Para um Motor de Indução Trifásico"*, Revista Controle & Automação, Vol. 12, Nº 2, Maio-Agosto, 2001.

(Garcia, 1990) Garcia, G, O, *"Controle de Velocidade de Motor de Indução Usando Técnicas Campo Orientado Indireto e Escorregamento Controlado"*, Tese de Doutorado, COPPE/UFRJ, Rio de Janeiro, 1990.

(Gomes, 2007) Gomes, R. G., *"Motor mancal com controle implementado em DSP"*, Dissertação de mestrado, COPPE/UFRN, 2007.

(Haykin, 2001) Haykin, S., *"Rede Neurais: Princípios e Prática"*, Tradução Paulo Martins Engel. - 2ª Edição, Bookman, Porto Alegre-RS, 2001.

(Jacobina, 1984) Jacobina, C. B., "*Acionamentos de Máquinas Elétricas de Alto Desempenho*", Minicurso do XIV Congresso de Brasileiro de Automática, Setembro, 2002.

(Leonhard, 1998) Leonhard, W., *"Field-orientation for controlling ac machines – principle and application"*. 3rd International Conference on Power Electronics and Variable-Speed Drives, pp. 277-282, July, 1988.

(Leonhard, 2001) Leonhard, W., *"Control of electrical drives",* Springer-Verlag, Third Edition, Berlin Heidelberg New York, Germany, 2001.

(Lisboa, 2007) Lisboa, S. D., Paiva, J. A., Queiroz, F. C. H., Salazar, A. O., Maitelli, A. L., *"Neural Flux Estimation Applied To The Vector Speed Control For Induction Machines"*, 9º Congresso Brasileiro de Eletrônica de Potência", 30 de Setembro à 04 de Outubro, Blumenau-SC, 2007.

(Maia, 1997) Maia, C. A., Resende, P. Silvino, J. L., *"A Neural Vector Control for Induction Machine"*, ISIE - IEEE International Symposium

on Industrial Electronics, pp.1265-1269, 1997.

(Narendra, 1990) Narendra, K. S., Parthasarathy, K., *"Identification and control of dynamical systems using neural networks"*, IEEE Transactions on Neural Networks, Vol. 1, pp. 2-27, March, 1990.

(Narendra, 1992) Narendra, K, S, Mukhopadhyay, S, *"Intelligent Control Using Neural Networks"*, IEEE Control Systems Magazine, Vol. 12, Nº 2, pp.11-18, April, 1992.

(Narendra, 1996) Narendra, K, S, *"Neural Networks for Control: Theory and Pratice"*, Proceedings of the IEEE, Vol. 84, Nº 10, pp. 1385-1406, October, 1996.

(Nascimento, Jr., 2004) Nascimento Jr., Cairo L., Yoneyama, T., "*Inteligência Artificial em Controle e Automação"*, Edgard Blücher: FAPESP, São Paulo, 2004.

(Paiva, 2006) Paiva, J. A., Ferreira, J. M. S., Salazar, A. O. Maitelli, A. L., *"Vectorial Speed Control using a Flux Estimator for Bearingless Machines with Divided Winding"*, The 10th International Symposium on Magnetic Bearing, Martigny, Switzerland, Aug. 21-23rd, 2006.

(Paiva, 2007) Paiva, J. A., "*Controle Vetorial de Velocidade de uma Máquina de de Indução sem Mancais Trifásica com Bobinado Dividido utilizando Estimação Neural de Fluxo"*, Tese de Doutorado, PPgEE/UFRN, Natal-RN, Dezembro, 2007.

(Santisteban, 2001) Santisteban, J. A., Stephan, R. M., *"Vector Control Methods for Induction Machines: An Overview"*, IEEE Transactions on Education, Vol. 44, Nº 2, pp. 170-175, May, 2001.

(Simões, 1995) Simões, M. G., Bose, B. K., *"Neural Network Based Estimation of Feedback Signals for Vector Controlled Induction Motor Drive"*, IEEE Transactions on Industry Applications, Vol. 31, Nº 3, pp. 620 – 629, May/June, 1995.

(Simone, 2000) Simone, G. A. *"Máquinas de Indução Trifásicas: Teoria e*

Exercícios", Editora Érica, 2000.

(Spectrum, 2003) Spectrum Digital, *"EzDsp LF2812 Technical Reference"*, DSP developem systems, September, 2003.

(Stronach,1998) Stronach, A. F., Vas, P., *"Design, DSP implementation, and performance of artificial-inteligence-based speed estimators for electromachanical drives"*, IEE proceedings on control Theory Applications, Vol, 145, Nº 02, pp. 197-203, March, 1998.

(Texas, 1998) Texas Instruments, *"Implementation of a Speed Field Orientated Control of Three Phase AC Induction Motor using TMS320F240"*, Application Notes for Industrial Applications, Nº BPRA076, March, 1998.

(Texas, 2001) Texas Instruments, *"Motor Speed Measurement Considerations when using TMS320C24x DSPs"*, Application Report, Nº SPRA771, August, 2001.

(Vas, 1996) Vas, P., Stronach, A. F., *"Adaptive Fuzzy-Neural DSP Control of High-Performance Drives"*, Power Electronics and Variables Speed Drives, Conference Publication Nº 429, IEE, 23-25, September, 1996.

Apêndice A
Parâmetros da Máquina de Indução

Os parâmetros da máquina de indução utilizada nos ensaios estão listados na Tab. 1 e foram utilizados no modelo vetorial apresentado no Capítulo 2, servindo de base para todo o projeto e desenvolvimento do sistema.

Tab. A.1. *Parâmetros da Máquina de Indução.*

Parâmetro	***Descrição***	***Valor***
P_{nom}	Potência Nominal	1,1 kW
ω_{nom}	Velocidade Nominal	1.800 rpm
V_{nom}	Tensão nominal	145 V
I_{nom}	Corrente Nominal	1 A
R_S	Resistência de Estator por fase	4,5853 Ω
R_R	Resistência de Rotor por fase	32,0894 Ω
L_{dS}	Indutância de dispersão do estator por fase	41,6 mH
L_{dR}	Indutância de dispersão do rotor por fase	41,6 mH
L_m	Indutância de magnetização	278,6 mH
L_S	Indutância do estator por fase	459,6 mH
L_R	Indutância do Rotor por fase	459,6 mH
n_p	Número de Par de Pólos	2
J	Momento de Inércia	$6{,}06.10^{-3}$ Kg.m^2
σ	Fator de Dispersão	0,1

Os parâmetros físicos da máquina foram obtidos experimentalmente através de ensaios de curto-circuito e circuito aberto (Simone, 2000; Ferreira, 2007).

As indutâncias próprias de estator e rotor foram calculadas a partir das respectivas indutâncias de dispersão e de magnetização através das Eqs. A.1 e A.2 (Santisteban, 2001):

$$L_S = L_{dS} + \frac{3}{2} L_m \tag{A.1}$$

$$L_R = L_{dR} + \frac{3}{2} L_m \tag{A.2}$$

Apêndice B

Pesos, Equações e Rotinas das RNAs do Estimador Neural

Este apêndice apresenta todos os dados, equações e rotinas desenvolvidas para a execução da estimação neural de fluxo do rotor.

B.1. Pesos e equações para a implementação das redes que compõem o Estimador Neural de Fluxo do Rotor.

Após o treinamento no ambiente Matlab® das redes neurais que executam a estimação de $d\hat{\rho}(k)$ e $\hat{i}_{mR}(k)$, foram obtidos os pesos listados a seguir:

- **Pesos da rede 1 que executa a estimação de** $d\hat{\rho}(k)$:

$$w_i = \begin{bmatrix} -0.62585 & -0.76378 & 0.34807 & 0.82384 & 0.49042 & 0.27349 \\ -0.71184 & 0.13846 & -0.74821 & -0.85334 & -0.35142 & 0.26833 \\ 1.2811 & -0.74768 & -0.78785 & 0.37799 & 0.21666 & -0.51263 \\ -0.73185 & 0.239 & -0.9592 & 0.15989 & 0.23935 & 0.30611 \\ -0.89785 & 1.2559 & -0.44406 & 0.39101 & -0.26579 & -0.78652 \\ 0.60168 & 0.84415 & 0.61065 & -0.26645 & 0.47027 & -0.031131 \end{bmatrix};$$

$$w_1 = \begin{bmatrix} -0.11254 & -0.23789 & 0.96384 & -0.44256 & -1.2962 & -0.4687 \end{bmatrix};$$

$$b_1 = \begin{bmatrix} 0.1867 \\ 0.42063 \\ 0.23734 \\ 0.72359 \\ -0.84977 \\ -0.27316 \end{bmatrix};$$

$$b_2 = \begin{bmatrix} -1.1115 \end{bmatrix}.$$

- **Pesos da rede 2 que implementa o cálculo de $\hat{i}_{mR}(k)$:**

$$w_{i\_i_{mR}} = \begin{bmatrix} -0.78112 & -0.42278 & 0.77907 \\ -0.18729 & -0.41936 & -0.22892 \\ -0.17426 & 0.50061 & 0.417 \\ -0.74353 & 0.40483 & 1.0391 \\ 0.44994 & 0.87017 & -0.43725 \\ 0.72603 & -0.25667 & 0.26152 \end{bmatrix};$$

$$w_{1\_i_{mR}} = \begin{bmatrix} 0.17709 & -0.14533 & -0.17965 & 0.66304 & 0.053866 & 0.79337 \end{bmatrix};$$

$$b_{1\_i_{mR}} = \begin{bmatrix} 0.027488 \\ 0.36543 \\ 0.034605 \\ 0.12817 \\ 0.01476 \\ -0.56814 \end{bmatrix};$$

$$b_{2\_i_{mR}} = \begin{bmatrix} 0.41526 \end{bmatrix}.$$

Após a determinação destes pesos foram aplicadas transformações em cada uma das matrizes e vetores para colocá-los no formato Q_{12} da seguinte forma: $w_{Q_{12}} = w * 2^{12}$, em que $w_{Q_{12}}$ é a matriz ou vetor no formato Q_{12} e w é a matriz ou o vetor no formato original.

Para normalizar todas as entradas das redes implementadas, foram aplicadas transformações dadas pela equação:

$$x_n = \frac{2.(x - x_{min})}{x_{max} - x_{min}} - 1 \qquad \text{(B.1)}$$

em que x_n é o valor normalizado da entrada x, e $x_{\max}$ e $x_{\min}$ são, respectivamente, os valores máximos e mínimos da entrada x.

Para desnormalizar as saídas das redes implementadas, foram aplicadas transformações dadas pela equação:

$$y = \frac{1}{2}(y_n + 1).(y_{\max} - y_{\min}) + y_{\min} \qquad \text{(B.2)}$$

em que y é o valor desnormalizado da saída y_n, e $y_{\max}$ e $y_{\min}$ são, respectivamente, os valores máximos e mínimos da saída y.

Na seção seguinte, está apresentada a listagem completa das rotinas e funções para o cálculo dos estimadores neurais implementados.

B.2. Listagem das rotinas e funções para o cálculo das Redes Neurais implementadas

```
/* Definição da topologia das Redes implementadas */

/* Topologia da rede 1 */
#define num_entradas_1 6
#define num_saidas_1   1
#define camada_1_1     6

/* Topologia da rede 2 */
#define num_entradas_2 3
#define num_saidas_2   1
#define camada_1_2     6

/* Função de normalização das entradas da rede */
int normaliza(long int x, int min, int max){
    long int saida;
       saida = (((x-min)<<1)*4096) / (max-min)  - 4096;
    return saida;
   }

/* Função de desnormalização das saidas da rede */
int desnormaliza(long int x, int min, int max){
   int saida;
       saida = ((((x+4096)>>1)*(max-min))>>12) + min;
   return saida;
   }

/* Função de implementação da rede 1 */
long int rna1 (long int x[]){

/* Declaração das variáveis e pesos da rede 1 no formato Q12 */
long int y1[camada_1_1] = {0, 0, 0, 0, 0, 0};
long int y2[num_saidas_1] = {0};

long int wi[num_entradas_1][camada_1_1] =
{{-2563,       -2916,        5247,       -2998,       -3678,        2464},
 {-3128,         567,       -3062,         979,        5144,        3458},
  {1426,       -3065,       -3227,       -3929,       -1819,        2501},
  {3374,       -3495,        1548,         655,        1602,       -1091},
  {2009,       -1439,         887,         980,       -1089,        1926},
  {1120,        1099,       -2100,        1254,       -3222,        -128}};

long int w1[camada_1_1][num_saidas_1] =
{{-461},      {-975},      {3948},     {-1813},     {-5309},      {-1920}};

long int b1[camada_1_1] = {765, 1723, 972, 2964, -3481, -1119};

long int b2[num_saidas_1] = {-4553};

int i,j;

for (i = 0; i < camada_1_1; i++)
 {
 for (j = 0; j < num_entradas_1; j++)
  {
    y1[i] = y1[i]+((x[j]*wi[j][i])>>12);
  }
    y1[i]=y1[i] + b1[i];
 }

for (i = 0; i < num_saidas_1 ; i++)
 {
 for (j = 0; j < camada_1_1; j++)
  {
    y2[i] = y2[i]+((y1[j] * w1[j][i])>>12);
  }
    y2[i] =y2[i] + b2[i];
```

```
 }
 return (y2[0]);
 }

/* Função de implementação da rede 2 */
long int rna2(long int x[]){

/* Declaração das variáveis e pesos da rede w no formato Q12 */
long int y1[camada_1_2] = {0,0,0,0,0,0};
long int y2[camada_1_2] = {0};

long int wi_imR[num_entradas_2][camada_1_2]=
{{-3199,        -767,        -714,        -3045,        1843,        2974},
 {-1732,       -1718,        2050,        1658,         3564,       -1051},
  {3191,        -938,        1708,        4256,        -1791,        1071}};

long int w1_imR[camada_1_1][num_saidas_1]=
{{725},       {-595},      {-736},      {2716},       {221},       {3250}};

long int b1_imR[camada_1_2] = {113, 1497, 142, 525, 60, -2327};

long int b2_imR[num_saidas_1] = {1701};

int i,j;

for (i = 0; i < camada_1_2; i++)
 {
 for (j = 0; j < num_entradas_2; j++)
  {
    y1[i] = y1[i]+((x[j]*wi_imR[j][i])>>12);
  }
    y1[i]=y1[i] + b1_imR[i];
 }

for (i = 0; i < num_saidas_2 ; i++)
 {
 for (j = 0; j < camada_1_2; j++)
  {
    y2[i] = y2[i]+((y1[j] * w1_imR[j][i])>>12);
  }
    y2[i] =y2[i] + b2_imR[i];
  }
  return (y2[0]);
  }
```

Apêndice C

Listagem do Programa de Controle

Apresenta-se a seguir a versão final do programa de controle do sistema implementado.

```
/*************************************************************************
Programa para o Controle de Velocidade da Máquina de Indução utilizando
Estimação Neural de Fluxo.
Nome: contveloRNA.c
Versão: 2.0
07/02/2008: Última atulização do controle de Velocidade;
*************************************************************************/

#include "senlc.h"
#include "DSP281x_Device.h"     // DSP281x Headerfile Include File
#include "DSP281x_Examples.h"   // DSP281x Examples Include File

// Prototype statements for functions found within this file.
interrupt void timer_isr (void);
void inicializa(void);
void ComSerial(void);
void desabilita_PWM(void);
void habilita_PWM(void);
void init_SCI(void);

// ADC start parameters
#define ADC_MODCLK 0x3    // HSPCLK=SYSCLKOUT/2*ADC_MODCLK2 = 150/(2*3) = 25MHz
#define ADC_CKPS   0x1    // ADC module clock = HSPCLK/2*ADC_CKPS =
                          // = 25MHz/(1*2) = 12.5MHz
#define ADC_SHCLK  0xf    // S/H width in ADC module periods

/********* Parâmetros da máquina no formato Qi ***************************/
#define KM 280
#define K  70
#define Kr 1
#define Ki 70
#define Kj 1
#define Kw 21

#define I2 4
#define IDMAX 80
#define IQMAX 100

/******************** Parâmetros do Controlador ****************************/
#define  KPW 11
#define  KIW 1

#define  KPIMR 2
#define  KIIMR 2

#define MIW 20
#define MIIMR 35

/************** Parâmetros dos filtros ******************/
#define  Aw  15
#define  Bw  1

#define  Adp  15
#define  Bdp  1
```

```
/************** Parâmetros dos filtros digitais ******************/
#define  Aw  15
#define  Bw  1

#define  Adp  15
#define  Bdp  1

/******************** Controle de corrente **********************/
#define GPC 145     /* 220 / 8 ganho proporcional de corrente; tensão CA RMS */
                    /* Obs.: colocar 145 Vac no varivolt!!!! */
#define GIC 30      /* 30 / 8 */
#define MPI 312     /* Limite da ação integrativa (Max. PWM Integrativo) */
#define CORRENTE_NOMINAL   60       /* max. = 400; min. = 40; nom. = 45 */

#define PWM_MAX 1250 // ciclo de trabalho igual a 100% */
#define PWM_50  625  // ciclo de trabalho igual a 50%  */

/* Deslocamentos de ciclo de trabalho para os braços inversores */
#define DES0 625
#define DES1 625
#define DES2 625
#define DES3 625
#define DES4 625
#define DES5 625

/****** Máximos mínimos da RNA 1 - Treinamento 22/01 ***************/
#define MIN_ISD     -53
#define MIN_ISDANT  -54
#define MIN_ISQ     -48
#define MIN_ISQANT  -45
#define MIN_VEL     0
#define MIN_VELOANT 0

#define MAX_ISD     60
#define MAX_ISDANT  60
#define MAX_ISQ     59
#define MAX_ISQANT  60
#define MAX_VEL     225
#define MAX_VELOANT 222

#define MIN_DP      -50
#define MAX_DP      695

/****** Máximos mínimos da RNA 2- Treinamento 22/01 ***************/
#define MIN_IMR     -1268
#define MAX_IMR     2100

#define MIN_IMRANT  -1268
#define MAX_IMRANT  2100

#define CHAVE (GpioDataRegs.GPEDAT.bit.GPIOE0)

#define TX_Pronto (SciaRegs.SCICTL2.bit.TXRDY)  /* se 1 indica pronto para enviar
                                                   novo caractere */
#define RX_Pronto (SciaRegs.SCIRXST.bit.RXRDY) /* se 1 indica caractere recebido e
                                                  pronto para ser lido */

/******* Parâmetros do sensor de velocidade *******/
#define per_tmr2    40946   /* período do Timer 2 */
#define per_tmr3    640     /* período do Timer 3 */
#define n           60      /* número de pulsos por rotação do disco */
#define CT          40946   /* máximo valor de delta */
#define krads       20715   /* fator de conversão para cálculo da velocida de em
                               rad/s */
#define krpm        195312 /* fator de conversão para cálculo do número de rpm */

/********* Comunicacao Serial *********/
#define BRR         243    /* determina a taxa de dados (Baud Rate) */
#define NUM_DADOS   9      /* número máximo de dados a serem transmitidos  */
char fase_com = 1;         /* indica a fase atual da comunicação serial */
int dado[NUM_DADOS];       /* armazena um instantâneo das variáveis a serem
                           transmitidas */
char indice_dado = 6;      /* 1 índice para o dado atual */
char indice_byte = 1;     /* índice para o byte (MSB ou LSB) atual de
                          determinado   dado */
```

```
char caractere=0;

/************************ Variáveis do controle de velocidade *********************/
long int wrefrpm=0,isd=0,isq=0,isaref=0,isbref=0,incp=0,isa=0,isb=0;
long int imRn=0,imR=1,dpmed=0,dimR=0,velorads=0,vradsmed=0,dwmref=0;
long int isdref=0,isqref=0,wmref=0,dp=0;
long int mM=0,intw=0,intimR=0;
long int mMref=0,erromM=0,intmM=0;
int dalfa=0,is1=0,is2=0,is3=0,limdpmax=0,limdpmin=0;
int mMint=0,z=0,npassos=0,m=1;
int errow=0,flag=0,isqi=0;
int erroimR=0;
int p=0,senl,cossl,imRef=5;
int i=0,j=0,np=2;
long int contint=0;
long int entrada[6]={0,0,0,0,0,0},temp1,entrada1[3]={0,0,0},imRnant=0,temp2=0;
long int veloradsant=0,isdant=0,isqant=0;

/******* Variáveis de controle de corrente **********/
int ganho = CORRENTE_NOMINAL, offset[6], ref[6], erro[6], ui[6], controle[6], k = 1,
cont = 0;
int corrente0=0, corrente1=0, corrente2=0, corrente3=0, corrente4=0, corrente5=0;

/************************ Variáveis do controle de velocidade *********************/
long int wrefrpm=0,isd=0,isq=0,isaref=0,isbref=0,wmec=0,incp=0,isa=0,isb=0;
long int  imRn=1,imR=1,dpmed=0,velorads=0, isdant=0,isqant=0,imRnant=0;
long int isdref=0,isqref=0,wmref=0,dp=0;
long int dimR=0,dwmec=0,mM=0,dwmref=50;
long int vradsmed=0,veloradsant=0;
int dalfa=0,is1=0,is2=0,is3=0;
int mMint=0,mL=0,z=0,npassos=0,m=1;
int flag=0;
int p=0,senl,cossl,senla,cossla;
int alfs,i=0,j=0,np=2;
unsigned int pos=0,fref=60;
long int contint=0;

/******* Variáveis de controle de corrente **********/
int ganho = CORRENTE_NOMINAL, offset[6], ref[6], erro[6], ui[6], controle[6],
Ia = 0, Ib = 0, Ic = 0, k = 1;
int corrente0=0, corrente1=0, corrente2=0, corrente3=0, corrente4=0, corrente5=0;

/******* Variáveis para medição de velocidade *******/
unsigned int c1=0, c2=0;   /* c1 e c2 armazenam valores do TXCNT quando um pulso
                           é recebido */
unsigned int delta=CT;     /* delta: contém a diferença c2-c1 */

/*######################## Fim das declarações ######################## */
/* Início das funções */
/**************** Configuração da Comunicação Serial ****************/
void init_SCI(void){
/* configura modulo SCI (comunicacao serial) */
       SciaRegs.SCICCR.all = 0x0007;     /* 1 bit de parada, 8 bits de dados,
                                         sem paridade, idle-mode */
       SciaRegs.SCICTL1.all = 0x0003;    /* inicializa flags e Máquina de estado
                                         da SCI, habilita tx e rx */
       SciaRegs.SCICTL2.all = 0x00C0;    /* seta flags rx e tx, desabilita
                                         interrupções rxrdy/brkdt e txrdy */
       SciaRegs.SCIHBAUD = BRR >> 8; /* define a taxa de transmissao (parte MSB) */
       SciaRegs.SCILBAUD = BRR & 0x00FF; /* define a taxa de transmissao
                                         (parte LSB) */
       SciaRegs.SCIPRI.all = 0x0010; /* prioridade das interrupções rx/tx,
                                    comportamento sob emulation suspend */
       SciaRegs.SCICTL1.bit.SWRESET = 1; /* habilita SCI*/
}

void desabilita_PWM (void){
       EvaRegs.ACTRA.all = 0x0FFF;
       EvbRegs.ACTRB.all = 0x0FFF;
}

void habilita_PWM (void){
       EvaRegs.ACTRA.all = 0x0666;
       EvbRegs.ACTRB.all = 0x0666;
}
```

```
/* ComSerial: Controla e envia pacote de dados pela serial. */
void ComSerial (void){
/* calcula velocidade do motor */
if (EvaRegs.CAPFIFOA.bit.CAP1FIFO == 2){ /* testa se CAP1FIFO possui 2 entradas */
       EvaRegs.CAPCONA.all = 0xA000; /* desabilita novas capturas
                                    (CAPCONA[7,6]-->CAP1EDGE=00) */
       c1 = EvaRegs.CAP1FIFO; /* retira o primeiro valor da pilha */
       c2 = EvaRegs.CAP1FIFO; /* (aguarda e) retira o segundo valor da
                                 pilha */
       EvaRegs.CAPCONA.bit.CAP1EDGE = 1; /* habilita novas capturas
                                          (CAPCONA[7,6]-->CAP1EDGE=01) */
       if (c2 <= c1)/* caso 2 */
              delta = CT + c2 - c1;
       else         /* caso 1 */
              delta = c2 - c1;
              velorads= krads/delta;
}

// envia pacote de informações pela serial, se requisitado
       switch (fase_com){
       case 1: {
// realiza o controle de fluxo da comunicação serial
              if (RX_Pronto){            // novo caractere recebido?
                     caractere = SciaRegs.SCIRXBUF.bit.RXDT;
              // lê o buffer de recepção
              if (caractere == 'T') // se recebeu´'T' então vá para a fase 2
                     fase_com = 2;
                     }
              // senão permaneça apenas ouvindo
       break;
       }

       case 2: {
// Envio das variáveis para o treinamento da rede
              dado[0] = wrefrpm;
              dado[1] = delta;
              dado[2] = mMint;
              dado[3] = isd;
              dado[4] = imR;
              dado[5] = isqi;
              dado[6] = dpmed;
              dado[7] = 0;
              dado[8] = 0;

              fase_com = 3;                           // vai para a fase 3
       break;
              }

       case 3: {
// envia um conjunto de dados (pacote), um dado de cada vez,
até a transferência completa.
       if (indice_byte == 1)      // envia byte menos significativo (LSB) do dado
              caractere = 0x00FF & dado[indice_dado];
       else
       if (indice_byte == 2) // envia byte mais significativo
                             (MSB) do dado
              caractere = dado[indice_dado] >> 8;
       //envia
       if (TX_Pronto){
              SciaRegs.SCITXBUF = caractere;    // transmite o caractere
       if (indice_byte == 1) // transmitiu somente o LSB ?
              indice_byte = 2; // na próxima, transmita o MSB
       else
       if (indice_byte == 2){ // já transmitiu o LSB e MSB ?
              indice_dado++; // aponta para o próximo dado a ser transmitido
              indice_byte = 1; // começando pelo LSB
       if (indice_dado > NUM_DADOS-1){   // todo o pacote já foi transmitido
              indice_dado = 0;
              fase_com = 1;              // vá para a fase 1
                                  }
                           }
                     }      // if (TX_Pronto)
       break;
                     }  // case 3
       default:
                           {
```

```
        fase_com = 1;
        break;
                    }
        } // switch (fase_com)
}

void inicializa(void){
        isd=0;isq=0;isa=0;isb=0;isaref=0;isbref=0;dp=0;mM=0;
        isdref=0;isqref=60;is1=0;is2=0;is3=0;
        wmref=0;
        imR=1;dalfa=0;
        incp=0;p=0;flag=0,dwmref=50;
        mMint=16384,z=0;
        errow=0;npassos=0;
        erroimR=0;
        senl=0;cossl=0;imRef=0;
        i=0;j=0;
        contint=0;
        npassos=0,intw=0;intimR=0;
        wrefrpm=0;senl=0;cossl=0;velorads=0;
        dalfa = ((wrefrpm * np * 578) >> 16);
        dado[2]=16384;
    }

/*************************************************************************/
/*                              ROTINA PRINCIPAL                         */
/*************************************************************************/
void main(void) {
        int i, j, tmp;
        //char caractere=0;
        // Step 1. Initialize System Control:
        // PLL, WatchDog, enable Peripheral Clocks
        // This example function is found in the DSP281x_SysCtrl.c file.

        InitSysCtrl();
        // Specific clock setting for this example:
        EALLOW;
        SysCtrlRegs.HISPCP.all = ADC_MODCLK; // HSPCLK = SYSCLKOUT/ADC_MODCLK
        EDIS;
        // Step 2. Initalize GPIO:
        // This example function is found in the DSP281x_Gpio.c file and
        // illustrates how to set the GPIO to it's default state.

        InitGpio();  // Skipped for this example
        // Step 3. Clear all interrupts and initialize PIE vector table:
        // Disable CPU interrupts
        DINT;
        // Initialize PIE control registers to their default state.
        // The default state is all PIE interrupts disabled and flags
        // are cleared.
        // This function is found in the DSP281x_PieCtrl.c file.

        InitPieCtrl();
        // Disable CPU interrupts and clear all CPU interrupt flags:
        IER = 0x0000;
        IFR = 0x0000;
        // Initialize the PIE vector table with pointers to the shell Interrupt
        // Service Routines (ISR).
        // This will populate the entire table, even if the interrupt
        // is not used in this example.  This is useful for debug purposes.
        // The shell ISR routines are found in DSP281x_DefaultIsr.c.
        // This function is found in DSP281x_PieVect.c.

        InitPieVectTable();
        // Interrupts that are used in this example are re-mapped to
        // ISR functions found within this file.
        EALLOW;  // This is needed to write to EALLOW protected registers
        PieVectTable.T1UFINT = &timer_isr;
        EDIS;   // This is needed to disable write to EALLOW protected registers
        /* configura registradores de status do sistema */
        //*SCSR1 = 0x00FD;
        //*SCSR2 = (*SCSR2 | 0x000B) & 0x000F;
        EALLOW;
        GpioMuxRegs.GPAMUX.all = 0x013F;
        GpioMuxRegs.GPADIR.all = 0x007F;
        GpioMuxRegs.GPBMUX.all = 0x007F;
```

```
GpioMuxRegs.GPBDIR.all = 0x013F;
GpioMuxRegs.GPDMUX.all = 0x0000;
GpioMuxRegs.GPDDIR.all = 0x0010;
GpioMuxRegs.GPEMUX.all = 0x0000;
GpioMuxRegs.GPEDIR.all = 0x0000;
GpioMuxRegs.GPFMUX.all = 0x0030;
GpioMuxRegs.GPFDIR.all = 0x0010;
GpioMuxRegs.GPGMUX.all = 0x0000;
GpioMuxRegs.GPGDIR.all = 0x0000;
EDIS;

InitAdc();
/* configura o módulo ADC para trabalhar sem usar interrupções */
//AdcRegs.ADCTRL1.all = 0x6F90;
//AdcRegs.ADCTRL2.all = 0x4000;
// Specific ADC setup for this example:
AdcRegs.ADCTRL1.bit.ACQ_PS = ADC_SHCLK;
AdcRegs.ADCTRL3.bit.ADCCLKPS = ADC_CKPS;
AdcRegs.ADCTRL1.bit.SEQ_CASC = 1;
AdcRegs.ADCTRL1.bit.CONT_RUN = 1;

/* ----------------- Gerenciadores de eventos -------------------- */
/* Configuração do Timer 2 - Utilizado para medir velocidade */
EvaRegs.T2CNT = 0x0000; /* limpa contador do timer 2 */
EvaRegs.T2PR = per_tmr2; /* define período de contagem */
EvaRegs.T2CON.all = 0x1740; /* inicia contagem */
// prescaler 111 = 128 para medir velocidade 4.77Hz

/* Configuração Timer 1 */
EvaRegs.GPTCONA.all = 0x0000; /* registradores de controle dos timers */
EvbRegs.GPTCONB.all = 0x0041; // ??????? REVISAR
EvaRegs.T1CNT = 0x0000; /* limpa contador do timer 1 */
EvaRegs.T1PR = PWM_MAX; /* set the PWM carrier period */
EvaRegs.T1CON.all = 0x0840;
// prescaler 000 = 0 para pwm 10kHz

/* Configuração Timer 3 */
EvbRegs.T3CNT = 0x0000; /* limpa contador do timer 3 */
EvbRegs.T3PR = PWM_MAX; //per_tmr3;
EvbRegs.T3CON.all = 0x0840;//0x0842;
// prescaler 000 = 0 para pwm 19.531kHz

/* Configuração dos PWM's" */
EvaRegs.DBTCONA.all = 0x0000; /* desabilita deadband units */
EvbRegs.DBTCONB.all = 0x0000;

EvaRegs.ACTRA.all = 0x0999; /* configura cada par de PWM para que */
EvbRegs.ACTRB.all = 0x0999; /* um seja ativo-alto e outro ativo-baixo */

EvaRegs.COMCONA.all = 0x8200; /* habilita operação das compare units, */
EvbRegs.COMCONB.all = 0x8200; // REVISAR

/* Configura interrupções do EVA */
EvaRegs.EVAIFRA.all = 0xFFFF;    /* clear all EVA group A interrupts */
EvaRegs.EVAIFRB.all = 0xFFFF;    /* clear all EVA group B interrupts */
EvaRegs.EVAIFRC.all = 0xFFFF;    /* clear all EVA group C interrupts */
EvaRegs.EVAIMRA.all = 0x0200;    /* enable desired EVA group A interrupts
                                 0200*/
EvaRegs.EVAIMRB.all = 0x0000;    /* enable desired EVA group B interrupts */
EvaRegs.EVAIMRC.all = 0x0000;    /* enable desired EVA group C interrupts */

/* Configura interrupções do EVB */
EvbRegs.EVBIFRA.all = 0xFFFF;    /* clear all EVB group A interrupts */
EvbRegs.EVBIFRB.all = 0xFFFF;    /* clear all EVB group B interrupts */
EvbRegs.EVBIFRC.all = 0xFFFF;    /* clear all EVB group C interrupts */
EvbRegs.EVBIMRA.all = 0x0000;    /* enable desired EVB group A interrupts */
EvbRegs.EVBIMRB.all = 0x0000;    /* enable desired EVB group B interrupts */
EvbRegs.EVBIMRC.all = 0x0000;    /* enable desired EVB group C interrupts */

/* Configuração da unidade de captura */
EvaRegs.CAPFIFOA.all = 0x0000;  /* 0000 0000 0000 0000 */
EvaRegs.CAPCONA.all = 0x2040;  /* 0010 0000 0100 0000 CAPCONA[15]=0,
                               limpa registradores de captura */
EvaRegs.CAPCONA.all = 0xA040;  /* 1010 0000 0100 0000 CAPCONA[15]=1,
                              não faz nada */
```

```
        /* Inicializa memória dos integradores */
        ui[0] = 0;
        ui[1] = 0;
        ui[2] = 0;
        ui[3] = 0;
        ui[4] = 0;
        ui[5] = 0;

        /* Inicializa o módulo de comunicação serial */
        init_SCI();

        /* Inicializa todas as variáveis */
        inicializa();

// Step 5. User specific code, enable interrupts:

// Enable PIE group 2 interrupt 6 for T1UFINT
        PieCtrlRegs.PIEIER2.all = M_INT6;

// Enable CPU INT2 for T1UFINT
        IER |= (M_INT2);

// Enable global Interrupts and higher priority real-time debug events:
        EINT;   // Enable Global interrupt INTM
        ERTM;   // Enable Global realtime interrupt DBGM

// laço principal infinito
        while (1){
        desabilita_PWM ();
// Calcula referência trifásica inicial
        while (!CHAVE){     // fica preso aqui enquanto a chave não for ligada
                ComSerial(); // A comunicação serial permanece ativa.
                contint=0;
        }
        IER = 0x0000;   // mascara (desabilita) todas as interrupções
// Lê os "offset's" dos sensores de corrente
        AdcRegs.ADCMAXCONV.all = 0x0000;        // Lê um canal por vez
        for (i = 0; i < 6; i++){
                offset[i] = 0;
                AdcRegs.ADCCHSELSEQ1.all = i; // seleciona o canal a ser capturado
                j = 0;
                while (j < 32){
                        AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;
                        AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1;// gatilha o início da
                                                         conversão
                        while (AdcRegs.ADCST.bit.INT_SEQ1== 0); // espera fim da
                                                                conversão
                        AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
// o resultado sempre é colocado em RESULT0
                        tmp = (AdcRegs.ADCRESULT0 >> 6) - DES_ADC;
                        if ((tmp < 525) && (tmp > 505)){
                                offset[i] += tmp;
                                j++;
                        }
                        for (tmp = 0; tmp < 10; tmp++);
// reseta o sequenciador da conversão
                                AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;
                        for (tmp = 0; tmp < 10; tmp++);
                        }
                                offset[i] = offset[i] >> 5;
                }

// Prepara para a aquisição das posições
        AdcRegs.ADCMAXCONV.all = 0x0001;    // Lê dois canais por vez
        AdcRegs.ADCCHSELSEQ1.all = 0x0076; // seleciona os canais a serem capturados
        AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;  // reseta o sequenciador da conversão
        k = 3;

        IER = 0x003F;       // habilita interrupções
        habilita_PWM ();

        while(CHAVE){

/********* Variação em Degraus **************/

        if ((contint <= 10) && (flag == 0)){
```

```
                wrefrpm = 1800;
                }
                if((contint >= 400000) && (contint <= 700050)){
                        wrefrpm = 900;
                        flag = 1;
                }
                if((contint >= 700000) && (contint <= 1000020)){
                        wrefrpm = 1500;
                }
                if((contint > 1000020)){
                        contint = 0;
                        wrefrpm = 1800;
                }

/********* Variação em Rampas Ascendentes e descendentes em 10 rpm *******/
/*
        switch(m){
        case 1:
                if((npassos <= 35) && (wrefrpm <= 1800)){
                        dwmref = 50;
                }
                if((npassos > 35) && (npassos < 85)){
                        dwmref = 0;
                }
                if((npassos >= 85) && (npassos <= 144)){
                        flag = 1;
                        dwmref = -10;
                }
                if((npassos > 144) && (npassos <= 205) && (flag == 1)){
                        dwmref = 0;
                }
                if(npassos > 205){
                        flag = 0;
                        npassos = 23;
                }
                        m = 2;
        break;

        case 2:
        if((contint > 5000) && (contint < 5040)){
                npassos = npassos + 1;
                wrefrpm = wrefrpm + dwmref;
                contint = 0;
                m = 1;
                }
        break;
        }

//*************************************************************************/
ComSerial(); // A comunicação serial permanece ativa.
         }
        } // while (1)

} // void main()
/************** INTERRUPT SERVICE ROUTINES **************/
/* Interrupção gerada pelo timer 1 */
interrupt void timer_isr (void){
        int i, temp;
        BIT_ON;
        contint++;
        switch (k){

        case 1:
        k++;

/********* Filtragem da velocidade do sensor ******/
         vradsmed = ((Aw*vradsmed + Bw*velorads)) >> 4;

/***********Cálculo das redes implementadas***********/
/* Cálculo da rede 2 - Cálculo da corrente de magnetização */
        entrada1[0] = normaliza(isd, MIN_ISD, MAX_ISD);
        entrada1[1] = normaliza(isdant, MIN_ISDANT, MAX_ISDANT);
        entrada1[2] = normaliza(imRnant, MIN_IMR, MAX_IMR);
        temp2 = rna2(entrada1);
        imRn = desnormaliza(temp2, MIN_IMR, MAX_IMR);
         imR = (imRn) >> 5;
```

```
/* Cálculo da rede 1 - Cálculo da velocidade angular do fluxo */
       entrada[0] = normaliza(isd, MIN_ISD, MAX_ISD);
       entrada[1] = normaliza(isdant, MIN_ISDANT, MAX_ISDANT);
       entrada[2] = normaliza(isq, MIN_ISQ, MAX_ISQ);
       entrada[3] = normaliza(isqant, MIN_ISQANT, MAX_ISQANT);
       entrada[4] = normaliza(vradsmed, MIN_VEL, MAX_VEL);
       entrada[5] = normaliza(veloradsant, MIN_VELOANT, MAX_VELOANT);
       temp1 = rna1(entrada);
       dp = desnormaliza(temp1, MIN_DP, MAX_DP);
       dp = (dp >= 600)? 1000 : dp;  /*Limitação da velocidade angular */
       dp = (dp <= -600)? -1000 : dp;

/* Filtragem da velocidade angular do fluxo */
       dpmed = (Adp*dpmed + Bdp*dp) >> 4;
       incp = ((Kw*dpmed) >> 8);   /* Cálculo do incremento angular */
       p = p + incp;

/* Cálculo do torque eletrico */
       mM = (KM*imR*isq) >> 11;
       mMint = mM + 16384;  /* Torque com Offset para envio para o Labview/

/********** Determinação do seno e cosseno do ângulo estimado ******************/
       p = (p >= 1024)? p - 1024 : p ;
       p = (p <= -1024)? p + 1024 : p ;
       z = (p >= 0)? p : -p;
       if(z <= 255){
              i = z;
              senl = senlc[i];
              cossl = senlc[255 - i];
              }
                     else{
                            if(z <= 511){
                                   i = 511 - z;
                                   senl = senlc[i];
                                   cossl = -senlc[255 - i];
                            }
                            else{
                                   if(z <= 767){
                                          i = (z - 512);
                                          senl = -senlc[i];
                                          cossl = -senlc[255 - i];
                                   }
                                   else{
                                          if(z <= 1023){
                                                 i = 1023 - z;
                                                 senl = -senlc[i];
                                                 cossl =  senlc[255 - i];
                                          }
                                   }
                            }
                     }
       break;

       case 2:
       k++;

/********* Filtragem da velocidade do sensor******/
       veloradsant = velorads;
       vradsmed = (Aw*vradsmed + Bw*velorads) >> 4;

/*********Tranformação de RPM para rad/s da velocidade de Referência**********/
       wmref = ((wrefrpm * 107) >> 10);

/***************Algoritmo de Controle de Velocidade *************************/
/* Controlador de Velocidade */
       errow = (wmref - vradsmed);
       intw = intw  +  ((KIW*errow)/10);
       intw = (intw>= MIW) ?  MIW : intw;
       intw = (intw < -MIW) ? -MIW : intw;
       isqref = ((KPW*errow)/10) + intw;
       isqref = (isqref>IQMAX)? IQMAX: isqref;
       isqref = (isqref<-IQMAX)? -IQMAX: isqref;

/* calculo do erro de fluxo */
       imRef = 35;
```

```
        erroimR = imRef - imR;
        intimR = intimR  +  ((KIIMR*erroimR));
        intimR = (intimR>= MIIMR) ? MIIMR: intimR;
        intimR = (intimR< -MIIMR) ?-MIIMR: intimR;
        isdref = ((KIIMR*erroimR)) + intimR;
        isdref = (isdref>IDMAX)?IDMAX: isdref;
        isdref = (isdref<-IDMAX)?-IDMAX: isdref;

/************ Fim do Algoritmo de Controle***************/
        break;

        case 3:
        k++;

/* Transformação inversa de PARK das Correntes isaref e isbref ************/
        isaref = (isdref * cossl - isqref * senl) >> 10;
        isbref = (isqref * cossl + isdref * senl) >> 10;

// Transformação para o modelo d-q
        Ia = ((86 * isaref) >> 7);
        Ib = ((-43 * isaref + 74 * isbref) >> 7);
        Ic = ((-43 * isaref - 74 * isbref) >> 7);

        case 4:
        k++;

/* Cálculo das referências de corrente em cada Bobina do estator*/
/* Calcula os sinais de referência */
        ref[0] = Ia + offset[0];  /* cmpr1 */
        ref[1] = Ib + offset[1];  /* cmpr2 */
        ref[2] = Ic + offset[2];  /* cmpr3 */
        ref[3] = Ia - offset[3];  /* cmpr4 */
        ref[4] = Ib - offset[4];  /* cmpr5 */
        ref[5] = Ic - offset[5];  /* cmpr6 */

/* Preparação para a aquisição das correntes */
        AdcRegs.ADCMAXCONV.all = 0x0005;        /* volta a ler seis canais por vez */
        AdcRegs.ADCCHSELSEQ1.all = 0x3210;      /* seleciona os canais
                                                   a serem capturados */
        AdcRegs.ADCCHSELSEQ2.all = 0x0054;
        AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;       /* reseta seqüenciador da conversão */
        break;

        case 5:
        k = 1;

/* Início das leituras das correntes de fase*/
        AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 1;       /* gatilha o início da conversão */
        while (AdcRegs.ADCST.bit.INT_SEQ1== 0); /* espera fim da conversão */
        AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
        corrente0 = (AdcRegs.ADCRESULT0 >> 6) ;
        corrente1 = (AdcRegs.ADCRESULT1 >> 6) ;
        corrente2 = (AdcRegs.ADCRESULT2 >> 6) ;
        corrente3 = (AdcRegs.ADCRESULT3 >> 6) ;
        corrente4 = (AdcRegs.ADCRESULT4 >> 6) ;
        corrente5 = (AdcRegs.ADCRESULT5 >> 6) ;

/* Algoritmo de controle de corrente: PI digital */
        erro[0] = ref[0] - corrente0;
        ui[0] += ((GIC * erro[0]) >> 3);
        if (ui[0] > MPI)
              ui[0] = MPI;
        else
        if (ui[0] < -MPI)
              ui[0] = -MPI;
              controle[0] = ((GPC * erro[0]) >> 3) + ui[0] + DES0;
              EvaRegs.CMPR1 = (controle[0] < 0) ? 0 : controle[0];

        erro[1] = ref[1] - corrente1;
        ui[1] += ((GIC * erro[1]) >> 3);
        if (ui[1] < -MPI)
              ui[1] = -MPI;
        else
        if (ui[1] > MPI)
              ui[1] = MPI;
              controle[1] = ((GPC * erro[1]) >> 3) + ui[1] + DES1;
```

```
            EvaRegs.CMPR2 = (controle[1] < 0) ? 0 : controle[1];

      erro[2] = ref[2] - corrente2;
      ui[2] += ((GIC * erro[2]) >> 3);
      if (ui[2] > MPI)
            ui[2] = MPI;
      else
      if (ui[2] < -MPI)
            ui[2] = -MPI;
            controle[2] = ((GPC * erro[2]) >> 3) + ui[2] + DES2;
            EvaRegs.CMPR3 = (controle[2] < 0) ? 0 : controle[2];

      erro[3] = ref[3] - corrente3;
      ui[3] += ((GIC * erro[3]) >> 3);
      if (ui[3] < -MPI)
            ui[3] = -MPI;
      else
      if (ui[3] > MPI)
            ui[3] = MPI;
            controle[3] = ((GPC * erro[3]) >> 3) + ui[3] + DES3;
            EvbRegs.CMPR4 = (controle[3] < 0) ? 0 : controle[3];

      erro[4] = ref[4] - corrente4;
      ui[4] += ((GIC * erro[4]) >> 3);
      if (ui[4] > MPI)
            ui[4] = MPI;
      else
      if (ui[4] < -MPI)
            ui[4] = -MPI;
            controle[4] = ((GPC * erro[4]) >> 3) + ui[4] + DES4;
            EvbRegs.CMPR5 = (controle[4] < 0) ? 0 : controle[4];

      erro[5] = ref[5] - corrente5;
      ui[5] += ((GIC * erro[5]) >> 3);
      if (ui[5] < -MPI)
            ui[5] = -MPI;
      else
      if (ui[5] > MPI)
            ui[5] = MPI;
            controle[5] = ((GPC * erro[5]) >> 3) + ui[5] + DES5;
            EvbRegs.CMPR6 = (controle[5] < 0) ? 0 : controle[5];

/******* Atribuição dos valores anteriores das entradas da rede **********/
      imRnant = imRn;
      isdant = isd;
      isqant = isq;
      veloradsant = vradsmed;

/* Cálculo da média das correntes de saída utilizando as correntes lidas ******/
      is1 = (corrente0 + corrente3)>>1 - (offset[0] + offset[3]) >> 1;
      is2 = (corrente1 + corrente4)>>1 - (offset[1] + offset[4]) >> 1;
      is3 = (corrente2 + corrente5)>>1 - (offset[2] + offset[5]) >> 1;

/* Transformação inversa de PARK das Correntes isaref e isbref ************/
/* Transformação para o modelo alfa-beta */
      isa = 3*is1 >> 1;
      isb = 111 * (is2 - is3) >> 7;

/* Transformação para o modelo d-q */
      isd = (isa * cossl + isb * senl) >> 10;
       isq = (isb * cossl - isa * senl) >> 10;
       isqi = isq + 101;   /* corrente isq com Offset para envio dos valores
                           para o Labview/
      break;

      default:
      k = 1;
           }

      EvaRegs.EVAIFRA.bit.T1UFINT = 1; /* limpa bit correspondente à interrupção
                                          do timer */

// Acknowledge interrupt to receive more interrupts from PIE group 2
      PieCtrlRegs.PIEACK.all = PIEACK_GROUP2;
}
```