UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
ENGENHARIA ELÉTRICA

Implementação de Controle Vetorial de Motor de Indução Trifásico por Imposição de Correntes

Ivan Lucas Arantes

Dissertação de Mestrado

Itajubá-MG
2007

Implementação de Controle Vetorial de Motor de Indução Trifásico por Imposição de Correntes

Dissertação apresentada ao
Programa de Pós-Graduação em
Engenharia Elétrica da UNIFEI
como requisito parcial para obtenção
do Título de Mestre em Engenharia Elétrica

Área de Concentração
Automação e Sistemas Elétricos Industriais

Orientador: Prof. Dr. Carlos Alberto Murari Pinheiro
Co-Orientador: Prof. Dr. Ângelo José Junqueira Rezek

Itajubá-MG
2 de Março de 2007

DEDICATÓRIA

Dedico esse trabalho a minha família e minha noiva

AGRADECIMENTOS

Agradeço a Deus por me dar forças, capacidade e paciência na busca de meus ideais

Aos professores orientadores Carlos Alberto Murari Pinheiro e Ângelo José Junqueira Rezek pela confiança, boa vontade e tempo dedicados

Aos meus pais Gilmar e Lúcia, meus irmãos Rafael e Leonardo, e minha noiva Cristiane, pelo apoio e compreensão

Aos colegas pela amizade e apoio técnico

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E NOMENCLATURAS

u_{S1}, u_{S2}, u_{S3} tensões de alimentação trifásicas;

R_S , R_R resistências do estator e rotor por fase;

ψ_S , ψ_R fluxos enlaçados do estator e rotor;

i_S e i_R correntes do estator e rotor;

ε - ângulo que representa a posição relativa do eixo do rotor em relação ao estator, medido entre os eixos magnéticos das fases 1 do estator e do rotor;

L_0 - indutância mútua estator-rotor por fase;

L_S, L_R - indutâncias por fase do estator e rotor.

i_S - vetor corrente do estator;

i_R - vetor corrente do rotor;

P - número de pares de pólos;

J - momento de inércia do rotor e carga;

w - velocidade angular do rotor;

m_d - torque eletromagnético no eixo do motor;

m_l - torque de carga;

Im(x) - operador que indica a parte imaginária do número complexo X;

$(X)^*$ - operador complexo conjugado do número X.

$T_R = L_R / R_R$ constante de tempo do rotor;

w - velocidade angular;

ρ - argumento de i_{mR} ;

i_{mR} - vetor corrente de magnetização;

i_{sd} - corrente de campo;

i_{sq} - corrente de torque;

wmr- velocidade angular do vetor corrente de magnetização;

ws- velocidade síncrona;

Vvr- Ganho do regulador de velocidade

Vrt-	Ganho do regulador de torque
Vrf-	Ganho do regulador de fluxo
Trv-	Constante de tempo do regulador de velocidade
Trt-	Constante de tempo do regulador de torque
Trf-	Constante de tempo do regulador de fluxo
Tm-	Constante de tempo mecânica
Tr -	Constante de tempo elétrica do rotor
k1-	Valor base da velocidade síncrona
k -	Constante do torque elétrico
Te-	Constante de tempo do bloco de atraso

RESUMO

Até algumas décadas atrás, os sistemas de controle para motores de indução trifásicos (inerentemente mais robustos que outros tipos de máquinas elétricas), eram mais custosos se comparado com sistemas de controle de motores de corrente contínua. Com o advento da teoria de controle vetorial e do desenvolvimento tecnológico dos semicondutores esse contexto se alterou, praticamente igualando as características de acionamento de motores de indução aos de corrente contínua. Este trabalho irá apresentar a implementação de um sistema de controle vetorial por imposição de correntes para motor de indução trifásico. O sistema foi desenvolvido utilizando um microcomputador, um sistema de aquisição de dados, uma ponte retificadora de potência a diodos e uma ponte inversora com transistores IGBT´s e respectivos drivers para gatilhamento. Será apresentada uma introdução teórica sobre controle vetorial, onde será mostrada uma malha de controle proposta por Leonhard (1986) na qual a estratégia deste trabalho foi baseada. Depois serão apresentados o software e hardware desenvolvidos que, por meio de técnicas de histerese de corrente, irão impor a corrente de acionamento do sistema. Finalmente serão mostrados os resultados reais obtidos com o controle de velocidade do motor na presença de variações de cargas e também com inversão do sentido de rotação do sistema.

ABSTRACT

Until a few decades, control systems for induction engines (totally more robust than the other kind of engines) were more costly when compared with the control of motors of continuous current. With the advent of theory of vector control and the technological development of semiconductors this context has altered itself, nearly matching the characteristics of the induction and continous current motors. This work will present the implementation of a system for the vector control for the imposition of currents control for the trifase induction. The system was developed through a microcomputer, a system of data acquisition, a rectifying bridge with diodes and an inverter bridge with IGBTs with drivers for pivoting. A theoretical introduction will be presented about vector control, where a control loop will be shown that has been proposed for Leonard (1986) in which the control was based. Following this the presentation of the software and hardware developed through the hystherisis of current, will impose the current to the motor. Finally the results obtained will be seen, showing the speed control of the motor with load variations, including with the inversion of rotation of the same.

Capítulo 1 – Introdução

Este trabalho propõe a implementação de um sistema de controle vetorial de um motor de indução trifásico por imposição de correntes, utilizando-se de técnicas de controle digital.

O próximo capítulo contém um breve histórico sobre controle vetorial. Haverá uma descrição mais detalhada do trabalho realizado nos capítulos que se seguirão.

A implementação do controle vetorial será dividida em duas partes, uma de software e outra de hardware.

Programas confeccionados em Delphi serão utilizados para interface com o usuário na escolha dos parâmetros do motor a ser controlado, na obtenção dos gráficos resultantes do controle e, principalmente, para implementar a malha de controle em tempo real.

A malha de controle terá como entrada a informação da referência de velocidade desejada para o motor, e como saída as correntes trifásicas de referência do estator, a serem impostas com o objetivo de se atingir uma velocidade real igual a de referência. Como sinais de realimentação têm-se as correntes reais do estator e a velocidade do rotor do motor (Figura 1.1). Para a implementação do controle em tempo real o programa ficará em laço fechado fazendo a aquisição das informações das correntes e velocidade, calculando os sinais de comando para gatilhamento dos IGBTs objetivando a imposição das correntes. Os sinais de modulação (PWM) para os drivers são calculados por método de histerese de corrente.

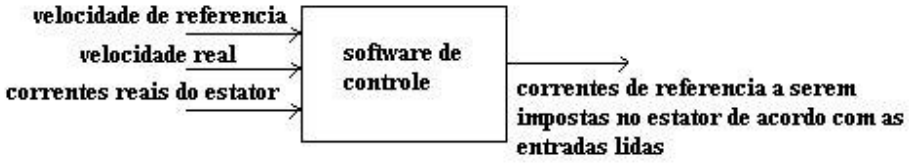


Figura 1.1 – Diagrama das informações do sistema de controle.

O hardware proposto na implementação da parte experimental deste trabalho será composto essencialmente por uma ponte retificadora de diodos, uma ponte inversora de IGBT´s e drivers de potência que recebem os pulsos do microcomputador e acionam o gates dos IGBT´s. Detalhes do sistema de controle serão apresentados em outros capítulos desta dissertação.

O capítulo 2 traz uma resenha bibliográfica básica dos assuntos pesquisados no desenvolvimento desta dissertação. O capítulo 3 mostra a modelagem utilizada neste trabalho. No capítulo 4 será apresentado uma implementação computacional para simulação do modelo matemático mostrado no capítulo 3. No capítulo 5 tem-se o detalhamento da parte experimental da dissertação. O capítulo 6 traz os resultados obtidos. E finalmente no capítulo 7 tem-se a conclusão e propostas para trabalhos futuros.

Capítulo 2 – Resenha Bibliográfica

O motor de indução ou de corrente alternada (CA) é o tipo de motor elétrico mais utilizado e difundido em aplicações industriais. Sua principal vantagem é a sua robustez e simplicidade construtiva, o que possibilita um custo baixo de fabricação e manutenção em relação a outras máquinas, como as de corrente contínua (CC), por exemplo.

Comparando tipos de motores com relação aos aspectos de controle, os motores CC são conhecidos pela facilidade de se controlar os mesmos. As máquinas de indução por apresentarem dependências mais complexas nas suas variáveis eram mais difíceis de serem controladas. Apesar se serem mais caros os motores DC eram mais utilizados em sistemas onde havia a necessidade de sistemas de controle precisos.

Ao longo de vários anos um grande número de pesquisas foram realizadas com o intuito de substituir os motores CC pelos CA. Os resultados desses esforços resultaram diversos trabalhos de cunho teórico que levaram a obtenção de modelos matemáticos elaborados que constituem a base para os sistemas de controle atuais. Dentre as técnicas inovadoras, pode-se dar ênfase a teoria de controle vetorial ou controle por campo orientado, no qual o presente trabalho está alicerçado e que será comentado nos parágrafos seguintes. Aliado aos desenvolvimentos dos modelos matemáticos, teve-se em paralelo um grande crescimento na área de eletrônica de potência, com semicondutores capazes de fazer chaveamentos mais rápidos e com capacidade de correntes maiores. E na área de microeletrônica houve a fabricação de microprocessadores com velocidades de processamento suficientes para a realização dos cálculos matemáticos necessários para realizar estas novas estratégias de controle. Todos estes fatores culminaram então nos sistemas inversores atuais, que viabilizam o uso dos motores CA em vários tipos de aplicações.

A técnica de controle vetorial iniciou-se em 1972 com Blaschke. O controle vetorial aplicado a motores de indução é baseado no princípio de controle de máquina CC de excitação independente. Neste tipo de controle o campo (estator) é um enrolamento separado do enrolamento da armadura (rotor). Assim, as

correntes de armadura e de campo podem ser controladas de forma independente, e apesar de variações de carga no sistema pode-se manter a velocidade constante. Já em um motor de indução as correntes do enrolamento de estator geram o fluxo e o torque resultante, logo é difícil controlar o fluxo e o torque separadamente até o advento do controle vetorial que resolveu este problema.

No controle vetorial as correntes alternadas do estator, que são variáveis temporais referenciadas aos eixos d-q do rotor (separando-as em duas componentes, Isd corrente de fluxo, e Isq corrente de torque). Nesta referência as correntes são estáticas, assim os controladores operam em modo contínuo. Como as correntes do motor são manipuladas na referência d-q do rotor, significa que as correntes medidas devem ser matematicamente transformadas da referência trifásica estática dos enrolamentos do estator, para os dois eixos d-q rotativos antes de serem processadas pelo controlador. De modo análogo, as correntes devem ser matematicamente transformadas do eixo d-q do rotor para a referência trifásica do estator, antes de serem utilizadas nos processamentos das saídas PWM do driver de acionamento do motor controlado.

Pode-se dizer que a implementação prática propriamente dita de controle vetorial iniciou-se no final da década de 70 com desenvolvimento dos microprocessadores e transistores de potência velozes (Leonhard et al, 1980).

Vários métodos de implementação de controle vetorial tem sido desenvolvidos, mas estas técnicas podem ser classificadas em dois grupos básicos: controle direto e controle indireto.

O controle direto foi proposto por Hasse (1969) e requer uma alta resolução de sensores de posição do rotor, como encoder ou resolver para determinar a posição do fluxo rotórico. O controle indireto originalmente sugerido por Blaschke (1972), estima o fluxo rotórico, ou fluxo enlaçado pelo rotor por meio de um parâmetro conhecido como vetor corrente de magnetização.

Este trabalho irá tratar do controle por campo orientado indireto, baseado em um dos tipos de controladores propostos pelo artigo de Leonhard (1986), mais precisamente no controlador através de imposição de correntes. Este controlador

tende manter o fluxo do motor constante, controlando o mesmo por meio da variação das correntes de armadura.

Existem trabalhos que além de variar as correntes, promovem a variação do fluxo também (Canudas e Ramirez, 1997).

Capítulo 3 – Modelagem de Sistemas de Controle Vetorial

3.1 – Modelo Matemático de Motores de Indução Trifásicos

De acordo com o modelo proposto por Leonhard (1986), as equações que regem a dinâmica de um motor de indução são as mostradas a seguir. A figura 3.1 apresenta o modelo de um motor de indução típico.

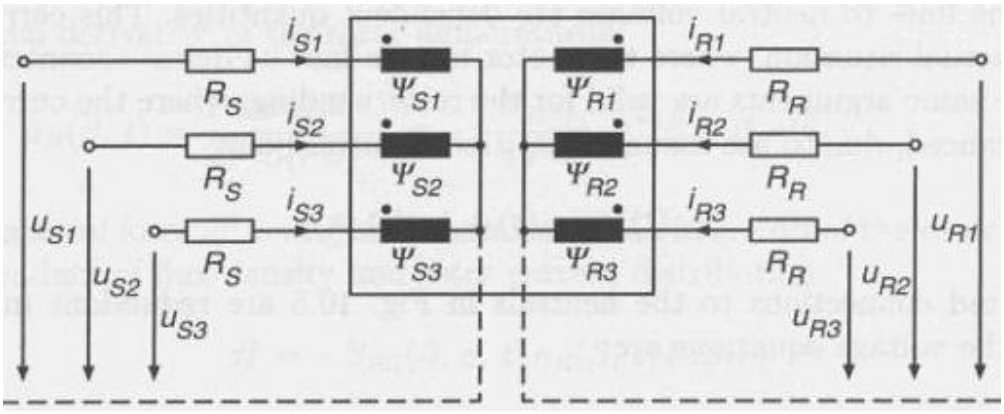


Figura 3.1 – Modelo básico de motores de indução.

As variáveis u_{S1}, u_{S2}, u_{S3} são as tensões de alimentação trifásicas, R_S e R_R são as resistências do estator e rotor por fase, ψ_S e ψ_R são os fluxos enlaçados do estator e rotor, i_S e i_R são as correntes do estator e rotor, respectivamente.

O equacionamento da figura anterior é mostrado em (1) para o circuito do estator e em (2) para o circuito do rotor.

$$R_S.i_{S1} + \frac{d\psi_{S1}}{dt} = u_{S1}(t)$$

$$R_S.i_{S2} + \frac{d\psi_{S2}}{dt} = u_{S2}(t)$$

$$R_S.i_{S3} + \frac{d\psi_{S3}}{dt} = u_{S3}(t) \qquad (1)$$

$$R_R.i_{R1} + \frac{d\psi_{R1}}{dt} = u_{R1}(t)$$

$$R_R.i_{R2} + \frac{d\psi_{R2}}{dt} = u_{R2}(t)$$

$$R_R.i_{R3} + \frac{d\psi_{R3}}{dt} = u_{R3}(t) \qquad (2)$$

Escrevendo as equações anteriores na forma vetorial vem (3) e (4).

$$u_S(t) = R_s.i_S + \frac{d\psi_S}{dt} \qquad (3)$$

$$u_R(t) = R_R.i_R + \frac{d\psi_R}{dt} \qquad (4)$$

Os fluxos enlaçados pelo rotor e estator podem ser escritos como segue.

$$\psi_S(t) = L_S.i_S(t) + L_0.i_R(t).e^{j\varepsilon} \qquad (5)$$

$$\psi_R(t) = L_R.i_R(t) + L_0.i_S(t).e^{-j\varepsilon} \qquad (6)$$

Onde:

ε - ângulo que representa a posição relativa do eixo do rotor em relação ao estator, medido entre os eixos magnéticos das fases 1 do estator e do rotor;

L_0 - indutância mútua estator-rotor por fase;

L_S, L_R - indutâncias por fase do estator e rotor.

Estas indutâncias do estator e rotor relacionam-se com a indutância mútua conforme as equações (7) e (8).

$$L_S = (1+\sigma_S).L_0 \qquad (7)$$

$$L_R = (1+\sigma_R).L_0 \qquad (8)$$

Baseando-se nas equações anteriores e fazendo-se algumas considerações, chega-se ao modelo matemático completo de um motor de indução representado pelas equações abaixo.

$$u_S = R_S.i_S + L_S.\frac{d(i_S)}{dt} + L_0.\frac{d(i_R.e^{j\varepsilon})}{dt} \quad (9)$$

$$R_R.i_R + L_R.\frac{d(i_R)}{dt} + L_0.\frac{d(i_S e^{-j\varepsilon})}{dt} = u_R = 0 \quad (10)$$

u_R=0, pois é considerado que os terminais do rotor foram curto-circuitados (rotor em gaiola).

$$\frac{1}{P}.J.\frac{d(w)}{dt} = m_d(t) - m_l \quad (11)$$

$$m_d = \frac{2}{3}.P.L_0.\mathrm{Im}[i_S.(i_R e^{j\varepsilon})^*] \quad (12)$$

$$\frac{1}{P}.w = \frac{d(\varepsilon)}{dt} \quad (13)$$

Onde:

u_S - vetor tensão de alimentação do estator;

i_S - vetor corrente do estator;

i_R - vetor corrente do rotor;

P - número de pares de pólos;

J - momento de inércia do rotor e carga;

w - velocidade angular do rotor;

m_d - torque eletromagnético no eixo do motor;

m_l - torque de carga;

Im(x) - operador que indica a parte imaginária do número complexo X;

$(X)^*$ - operador complexo conjugado do número X.

3.2 – Controle Vetorial por Campo Orientado Indireto

Motores de indução com gaiola de esquilo têm grandes vantagens por sua construção mecânica robusta e do baixo custo de produção. São alternativas interessantes para aplicações diversas. Adotando o controle com imposição de correntes e levando em conta que em um rotor em gaiola $u_R = 0$, substituindo a equação (8) na (10) vem:

$$R_R.i_R + L_0.\frac{d((1+\sigma_R)i_R + i_S e^{-j\varepsilon})}{dt} = 0. \qquad (14)$$

Sendo as correntes mensuráveis, um vetor corrente de magnetização (i_{mR}) é então definido:

$$i_{mR}(t) = i_{mR}e^{j\rho} = i_S + (1+\sigma_R).i_R e^{j\varepsilon}. \qquad (15)$$

O mesmo é usado para eliminação da corrente do rotor, então na equação (14), obtendo-se:

$$T_R.\frac{d(i_{mR})}{dt} + (1 + jwT_R)i_{mR} = i_S. \qquad (16)$$

Onde:

$T_R = L_R / R_R$ é uma constante de tempo do rotor;

w - velocidade angular;

ρ - argumento de i_{mR}.

Escrevendo a corrente i_{mR} na forma polar, a equação (16) pode ser dividida em uma parte real e outra imaginária. A parte real é chamada de corrente de campo (i_{sd}) e a imaginária de corrente de torque (i_{sq}).

$$i_{sd} = T_R.\frac{d(i_{mR})}{dt} + i_{mR} \qquad (17)$$

$$i_{sq} = T_R.i_{mR}.\left(\frac{d\rho}{dt} - w\right) \qquad (18)$$

$$\frac{d\rho}{dt} = w_{mR} \qquad (19)$$

Estas correntes i_{sd} e i_{sq} constituem as parcelas reais e imaginárias das correntes do estator, mas com referência no rotor (i_{mR}).

O equacionamento apresenta uma transformação de coordenadas, onde a referência é determinada pelo rotor, sendo conhecida como controle em coordenadas de campo. Nestas coordenadas a referência não é fixa, mas move-se com o rotor. Em estado permanente as correntes i_{sd} e i_{sq} são constantes.

É possível reescrever a equação do torque elétrico aplicando a transformação anterior:

$$m_d(t) = K.i_{mR}.i_{sq}. \tag{20}$$

Onde:

$$K = \frac{2}{3}\left(\frac{L_0}{1+\sigma_R}\right). \tag{21}$$

As relações angulares entre as grandezas analisadas podem ser observadas na Figura 3.2.

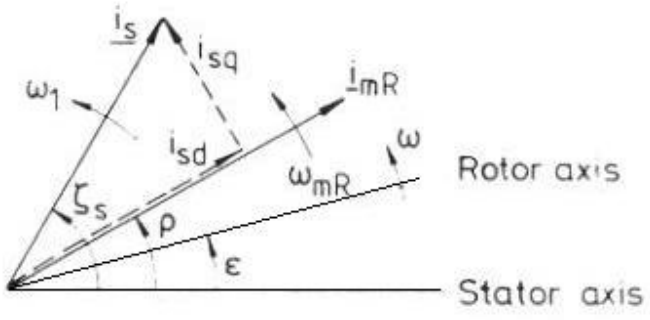


Figura 3.2 – Vetores e relações angulares de um motor de indução.

As equações (17), (18), (19), (20) e (21) estão ilustradas na próxima figura, mostrando a transição do estator para as coordenadas de campo. Pode-se observar que a corrente i_{mR} é obtida através das correntes i_{sd} e i_{sq}.

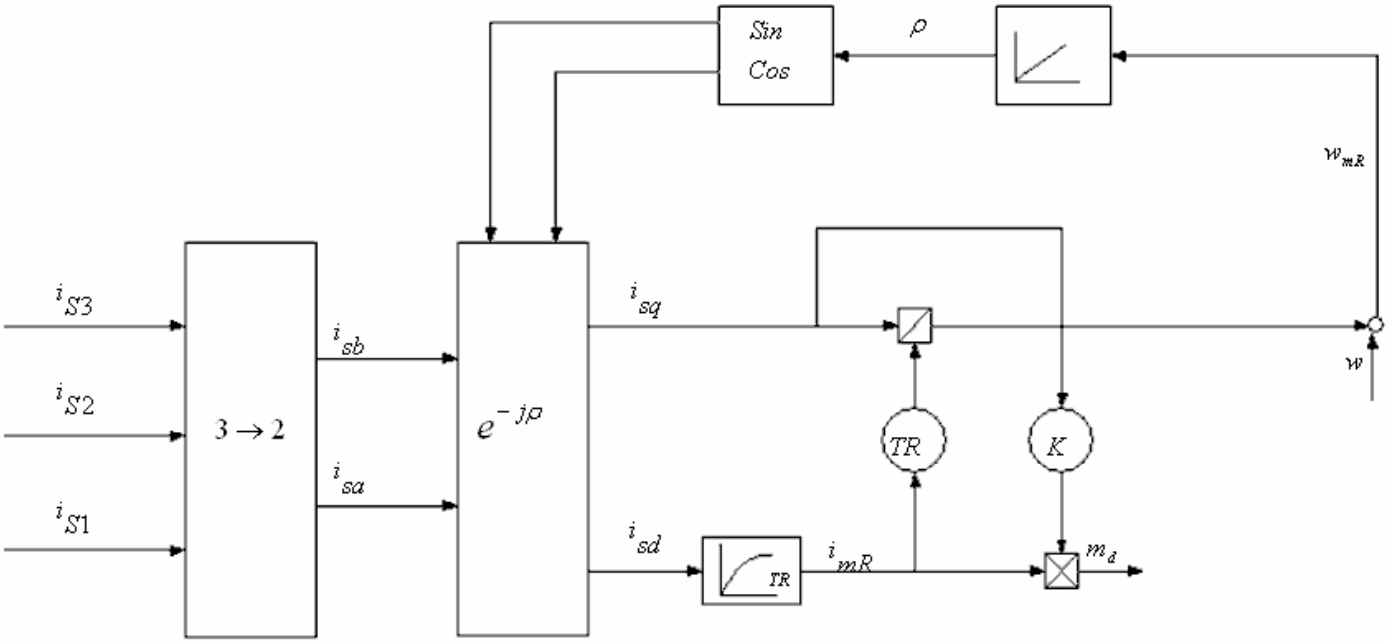


Figura 3.3 – Transição do estator para as coordenadas de campo.

3.3 – Malha de Controle

Baseando-se na malha de controle proposto por Leonhard (1986), e tendo em vista o equacionamento apresentado anteriormente, pode-se implementar o controle vetorial de um motor de indução trifásico, assumindo imposição de correntes. Esta estrutura de controle é mostrada na Figura 3.4 e pode ser dividida em duas partes. A parte física composta por motor, conversores, transdutores de velocidade, de corrente, etc., e a parte do sistema que implementa a malha reguladora.

As informações de leitura dos sensores de velocidade e corrente são digitalizadas e processadas pela malha de controle, de modo que o motor atinja a velocidade de referência (w_{ref}) estabelecida.

O esquema de controle apresentado corresponde a técnica de controle por campo orientado direto. Neste controle o vetor i_{mR} é obtido na parte superior da malha de controle da figura em questão.

Para uma melhor compreensão será apresentada uma descrição resumida sobre os blocos ou funções utilizadas.

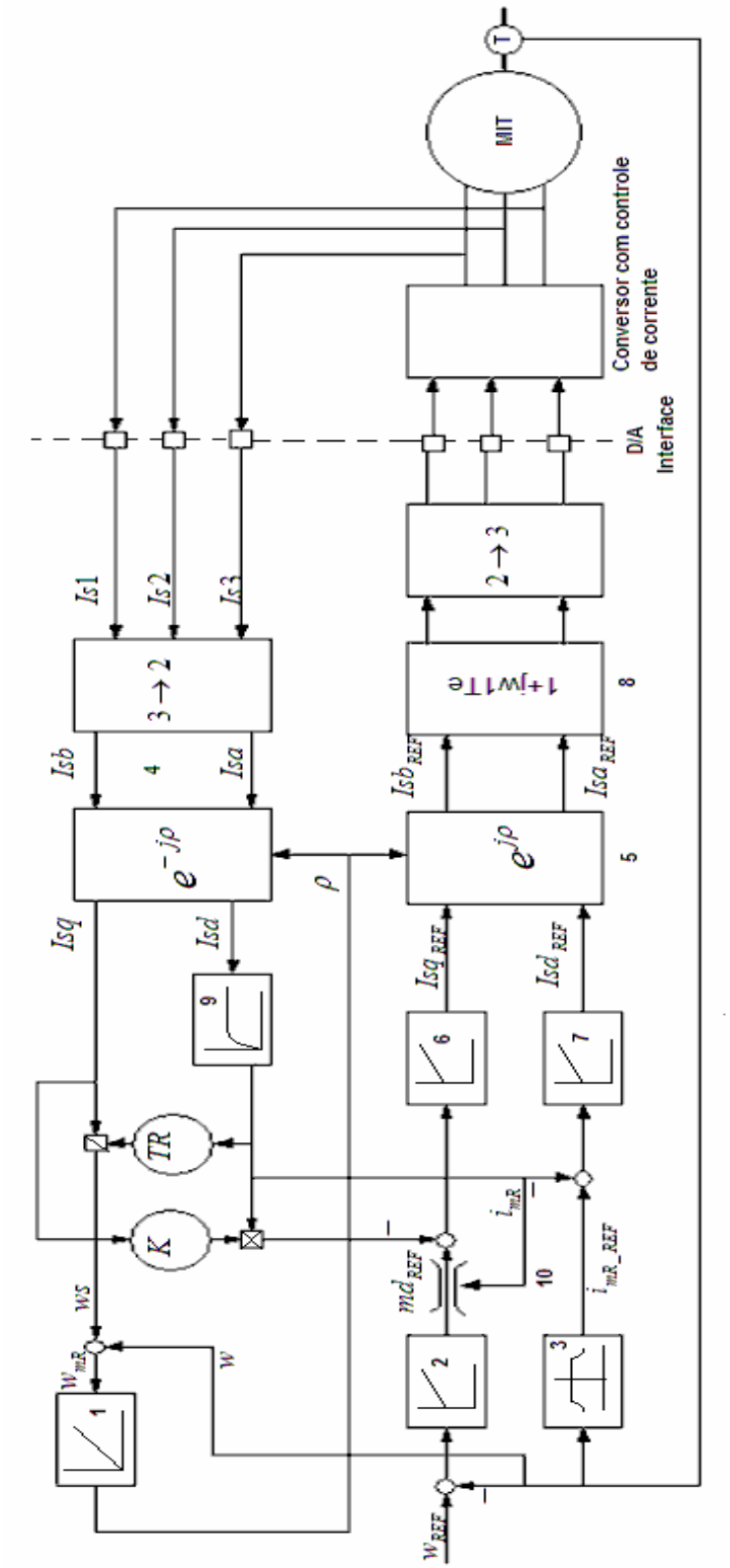


Figura 3.4 – Estrutura da malha de controle vetorial por imposição de correntes.

1- Bloco integrador simples: Realiza a integração da velocidade do campo magnético girante com a obtenção do ângulo do vetor i_{mR}.

2- Regulador de velocidade: Atua no controle de velocidade. Os parâmetros são a constante de tempo T_{RV} e o ganho V_{RV}.

3- Enfraquecimento de campo: Promove o enfraquecimento do campo para que o motor de indução trifásico opere acima de velocidade nominal.

4- Bloco modulador $e^{-j\rho}$ e $3 \rightarrow 2$: Blocos responsáveis para realizar a transformação das correntes do estator para as coordenadas de campo.

5- Bloco demodulador $e^{j\rho}$ e $2 \rightarrow 3$: Processo inverso ao do bloco 4.

6- Regulador de torque: Controla o torque gerado. Os parâmetros são a constante de tempo T_{RT} e o ganho V_{RT}.

7- Regulador de fluxo: Regula o fluxo enlaçado com o rotor. Os parâmetros são as constantes de tempo T_{RF} e o ganho V_{RF}.

8- Bloco de atraso: Compensador de atraso, os parâmetros são a constante de tempo T_e e a freqüência angular das correntes do estator w_1.

9- Sistema de primeira ordem: Bloco que representa uma função de primeira ordem.

10- Limitador: Limita a função de saída dentro de um dado intervalo de valores.

Capítulo 4 – Simulação do Sistema de Controle

Com o objetivo de testar a estratégia de controle a ser implementada, desenvolveu-se um programa de simulação baseado no trabalho de Santos (2004). Os resultados a serem simulados indicarão a validade da modelagem apresentada no capítulo anterior e a eficiência da estratégia de controle a ser adotada. A seguir serão mostradas as partes principais do programa de simulação desenvolvido. O código fonte do programa é mostrado no Anexo 3.

Para melhor entendimento do programa, será explicado com mais detalhes como foram implementadas as principais funções do mesmo. Como referência será utilizada a Figura 4.1 onde há o bloco com a denominação 3→ 2. Esse bloco é responsável em converter os valores das correntes do estator para a referência no rotor. O conceito desta transformação pode ser vista com mais detalhes em Leonhard (1996). As expressões (22) e (23) indicam as conversões. O próximo bloco realiza a transformação bifásica para um sistema a correntes constantes (d-q) conhecida como transformada de Park representada pelas equações (24) e (25). As transformações inversas são dadas por (26), (27), (28), (29) e (30).

$$i_{sa} = \frac{3}{2} * i_{s1} \tag{22}$$

$$i_{sb} = \frac{\sqrt{3}}{2} * (i_{s2} - i_{s3}) \tag{23}$$

$$i_{sd} = i_{sa} * \cos\rho + i_{sb} * sen\rho \tag{24}$$

$$i_{sq} = i_{sb} * \cos\rho - i_{sa} * sen\rho \tag{25}$$

$$i_{sa} = i_{sd} * \cos\rho - i_{sq} * sen\rho \tag{26}$$

$$i_{sb} = i_{sq} * \cos\rho + i_{sd} * sen\rho \tag{27}$$

$$i_{S1} = \frac{2}{3} * i_{Sa} \tag{28}$$

$$i_{S2} = \frac{1}{\sqrt{3}} * i_{Sb} - \frac{1}{3} * i_{Sa} \tag{29}$$

$$i_{S3} = -\frac{1}{3} * i_{Sa} - \frac{1}{\sqrt{3}} * i_{Sb} \tag{30}$$

As malhas de controle utilizam controladores PI (Proporcional+Integral) cujo modelo básico está representado em (31), sendo *K*1 e *K*2 parâmetros e *S* representa o operador de Laplace. Para a simulação é necessário acrescentar o cálculo da velocidade do motor e de eventuais variações da carga. Informações adicionais sobre métodos de implementações numéricas de controladores PI podem ser encontradas em Phillips e Nagle (1995).

$$\frac{Saída}{Entrada} = K1 * \frac{1 + S * K2}{S * K2} \tag{31}$$

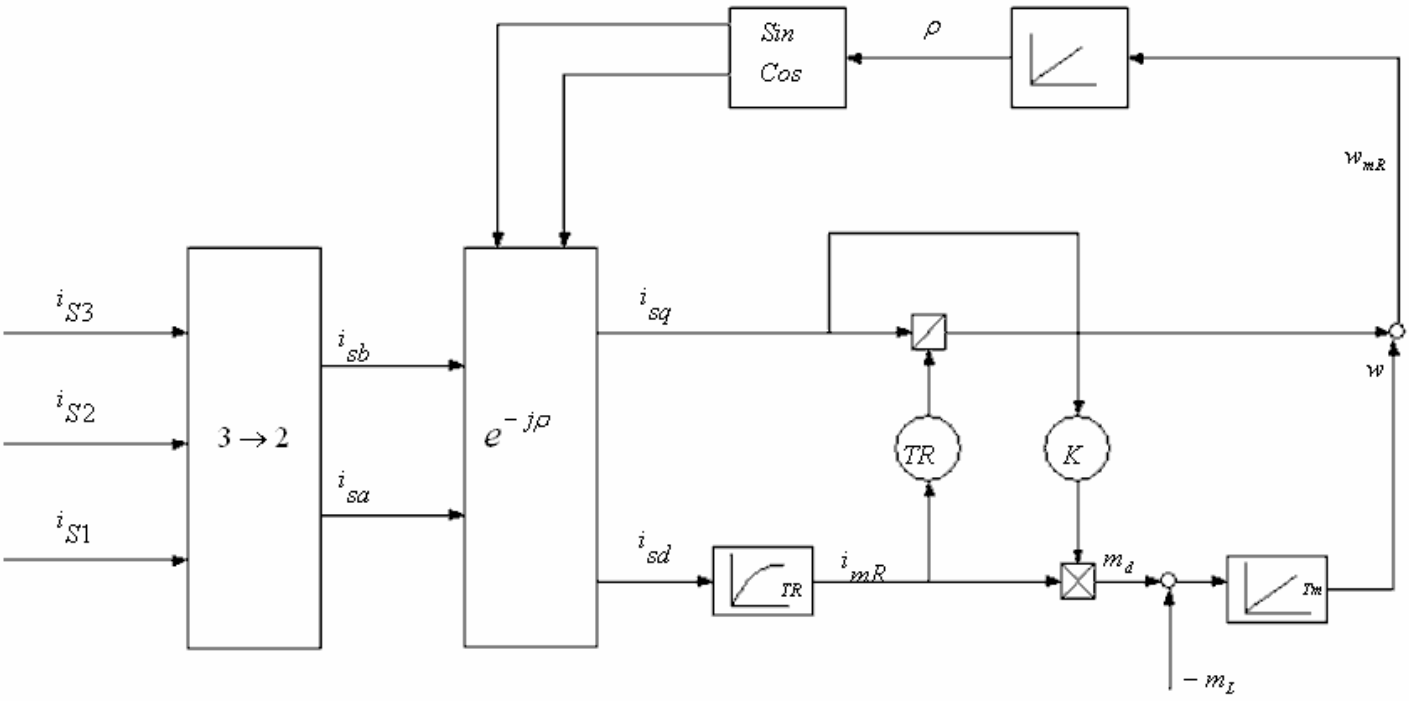


Figura 4.1 – Representação de conversão de coordenadas e malha de controle.

O programa tem como objetivo simular a malha de controle de velocidade de um motor de indução de acordo com uma velocidade de referência desejada. É simulado o comportamento do motor durante aproximadamente trinta e dois segundos, tempo suficiente para o sistema entrar em regime permanente. O usuário tem a opção de mudar a velocidade de referência, inclusive mudando o sentido de rotação, mudar os parâmetros do motor, e escolher o tipo de carga.

O aplicativo está divido em duas partes. Na primeira o usuário entra com os dados manualmente (através da opção "Nova Simulação") ou através de um arquivo ("Carregar Parâmetros Salvos") e depois executa a simulação. Na segunda parte o usuário pode analisar os gráficos de alguma simulação já realizada ("Carregar Simulação"). A tela principal do programa está ilustrada na Figura 4.2. Colocando o mouse em cima de qualquer caixa de texto aparecerá uma descrição sobre o respectivo parâmetro. Algumas medidas são dadas em p.u., por exemplo, w_{ref} e w_1. O programa inicia com valores padrões para os parâmetros a serem digitados, cujos valores são mostrados na Tabela 4.1.

Tabela 4.1 – Parâmetros padrões utilizados no programa de simulação.

Parâmetro	**Valor**	**Parâmetro**	**Valor**
T_{desc}	15	T_{RF}	0.02
T_{desa}	0	T_{RT}	0.01
W_{ref}	1	T_{RV}	0.9
V_{RF}	0.8	T_e	0.15
V_{RT}	0.15	T_R	0.16
V_{RV}	7	T_M	2.5
K	1	W_1	1
K_1	377		

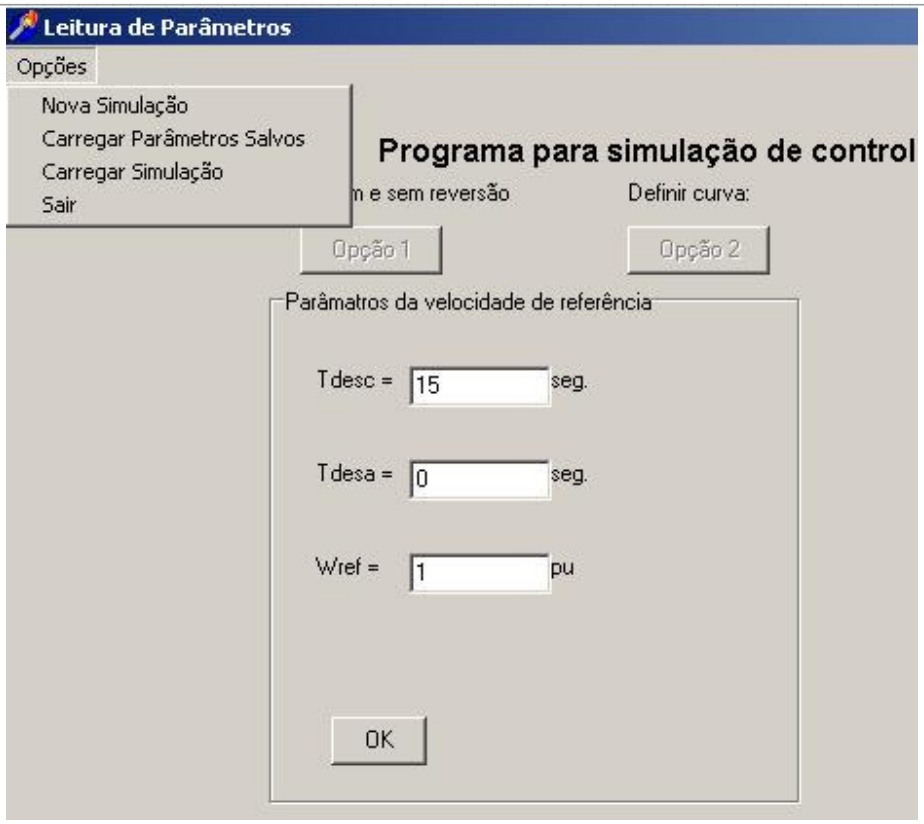


Figura 4.2 – Tela inicial do programa de simulação.

O programa de simulação apresenta as seguintes opções:

a) Opções no controle de velocidade:

- Partida com velocidade nominal e reversão.

Ao escolher esta opção (Figura 4.3) o usuário poderá simular a operação nos dois sentidos de rotação com velocidade de referência igual a W_{ref}, onde T_{desc} representa o instante que o motor inverte o sentido de rotação. $Tdesa$ é o tempo de desaceleração para que o motor parta da velocidade nominal no sentido horário e atinja a velocidade nominal no sentido inverso. Deve-se considerar que este tempo depende das características físicas do motor.

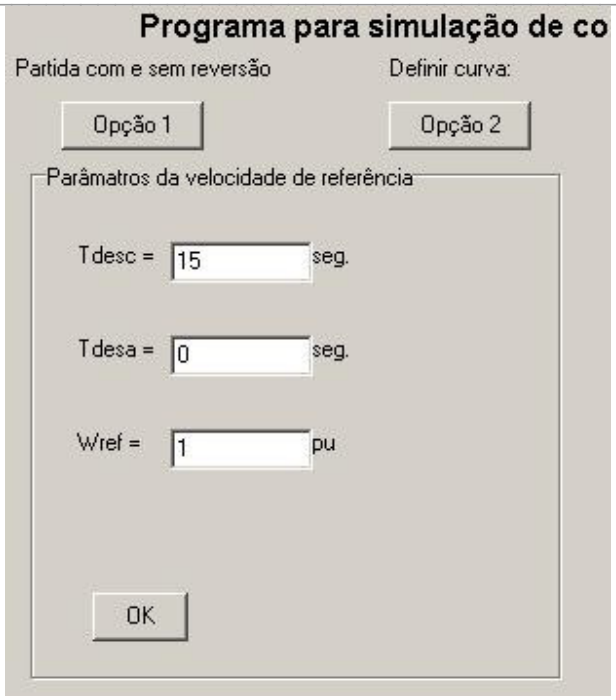


Figura 4.3 - Opção para o controle da velocidade.

- Escolher mudar a velocidade de referência cinco vezes durante a simulação.

Escolhendo esta opção é possível definir cinco velocidades de referência diferentes durante toda a simulação. O intervalo para cada uma é igual e está descrito na Figura 4.4.

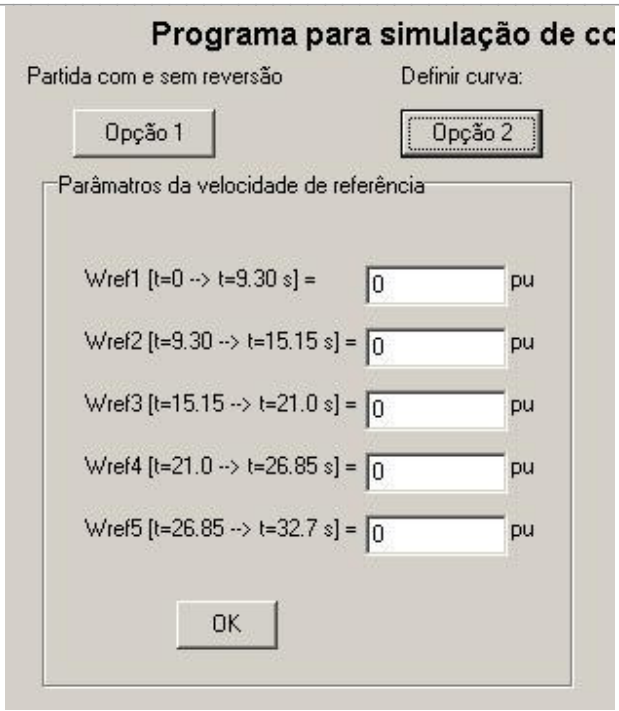


Figura 4.4 - Escolha de cinco velocidades de simulação.

b) Entrada dos parâmetros de regulação da malha de controle (Figura 4.5):

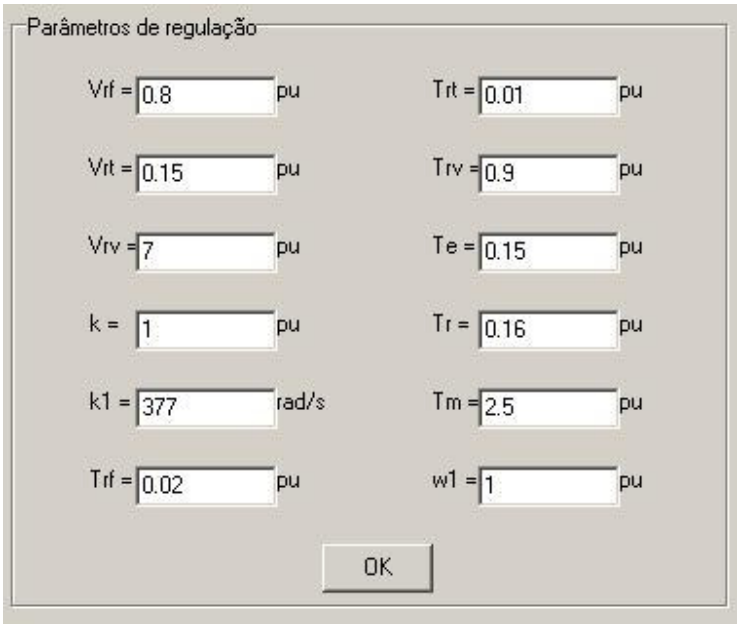


Figura 4.5 - Parâmetros de regulação.

Os parâmetros de regulação são divididos da seguinte maneira descrita abaixo.

- Ganho do regulador de velocidade (Vvr)
- Ganho do regulador de torque (Vrt)
- Ganho do regulador de fluxo (Vrf)
- Constante de tempo do regulador de velocidade (Trv)
- Constante de tempo do regulador de torque (Trt)
- Constante de tempo do regulador de fluxo (Trf)

- Parâmetros do MIT
  - Constante de tempo mecânica (Tm)
  - Constante de tempo elétrica do rotor (Tr)

- Outros parâmetros
  - Velocidade síncrona (w1)
  - Valor base da velocidade síncrona (k1)
  - Constante do torque elétrico (k)
  - Constante de tempo do bloco de atraso (Te)

c) Escolha do tipo de carga e diretório para salvar a simulação com as opções:

- Degrau: Simula uma mudança repentina no torque da carga.
- Carga com atrito viscoso: Carga = constante x velocidade angular.
- Carga de ventiladores: Carga = constante x velocidade angular^2.

A Figura 4.6 ilustra a entrada de dados desta opção.

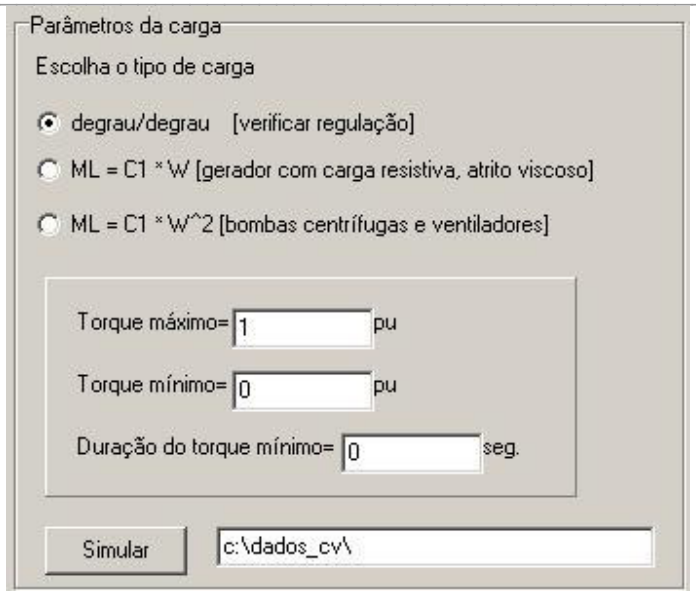


Figura 4.6 – Opção de com escolha do tipo de carga.

d) Visualização dos resultados:

Após a simulação realizada é possível visualizar os resultados através de gráficos. Acionando o mouse em cada opção obtém-se o gráfico respectivo. A seguir serão vistos dois exemplos de simulação.

Exemplo 1 - O motor é acionado com carga constante e com inversão no sentido de rotação no tempo de 15 segundos. Os parâmetros estão indicados abaixo.
- Velocidade: Opção 1 com Tdesc= 15 [s]; Tdesa = 0 [s]; Wref = 1 [pu].
- Carga: Degrau; Torque máximo= 1 [pu]; Torque mínimo = 0 [pu]; Duração do torque mínimo = 0 [s].

As Figuras 4.7 a 4.13 ilustram os resultados da simulação deste exemplo.

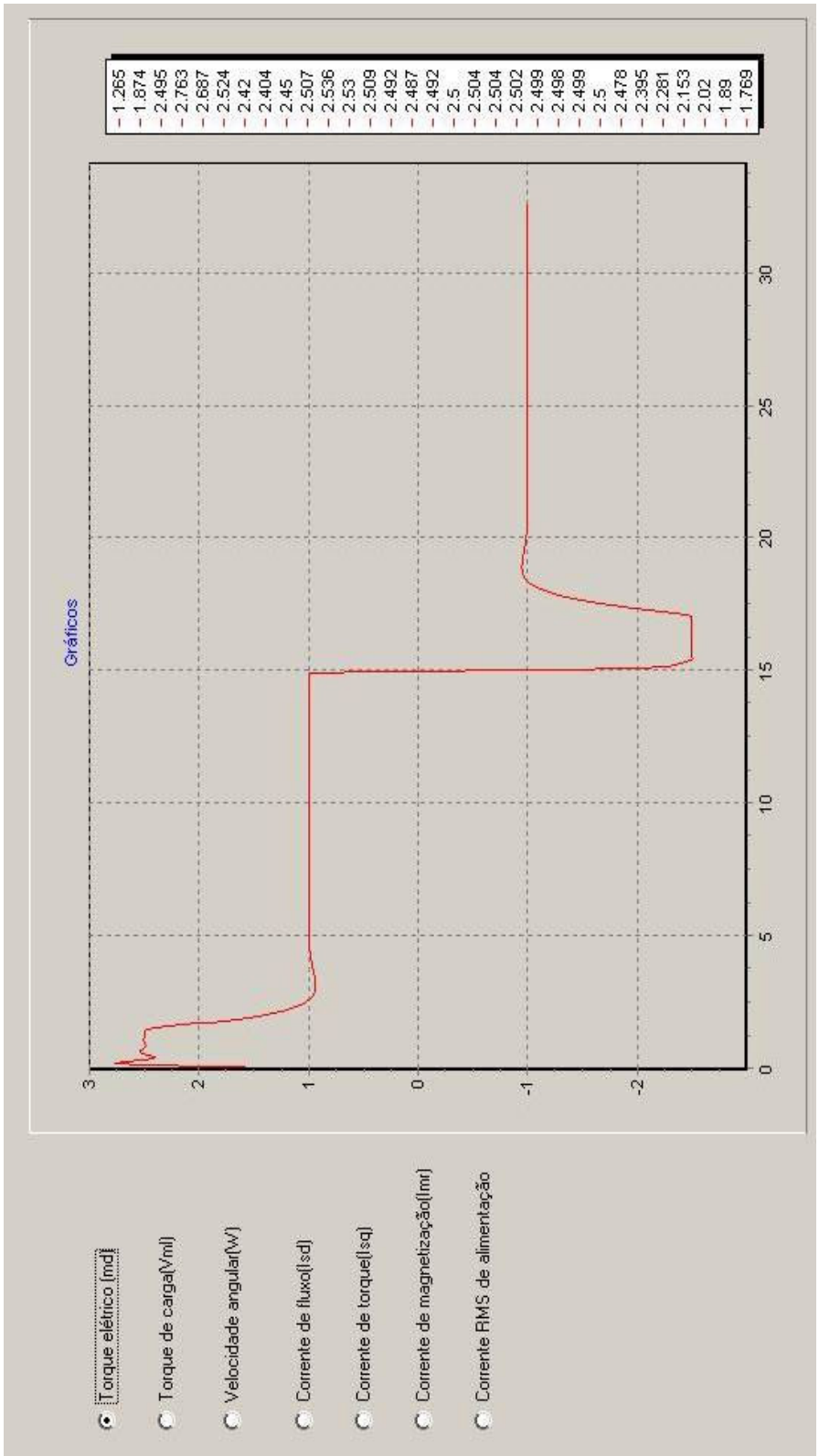


Figura 4.7 - Torque elétrico (Exemplo 1).

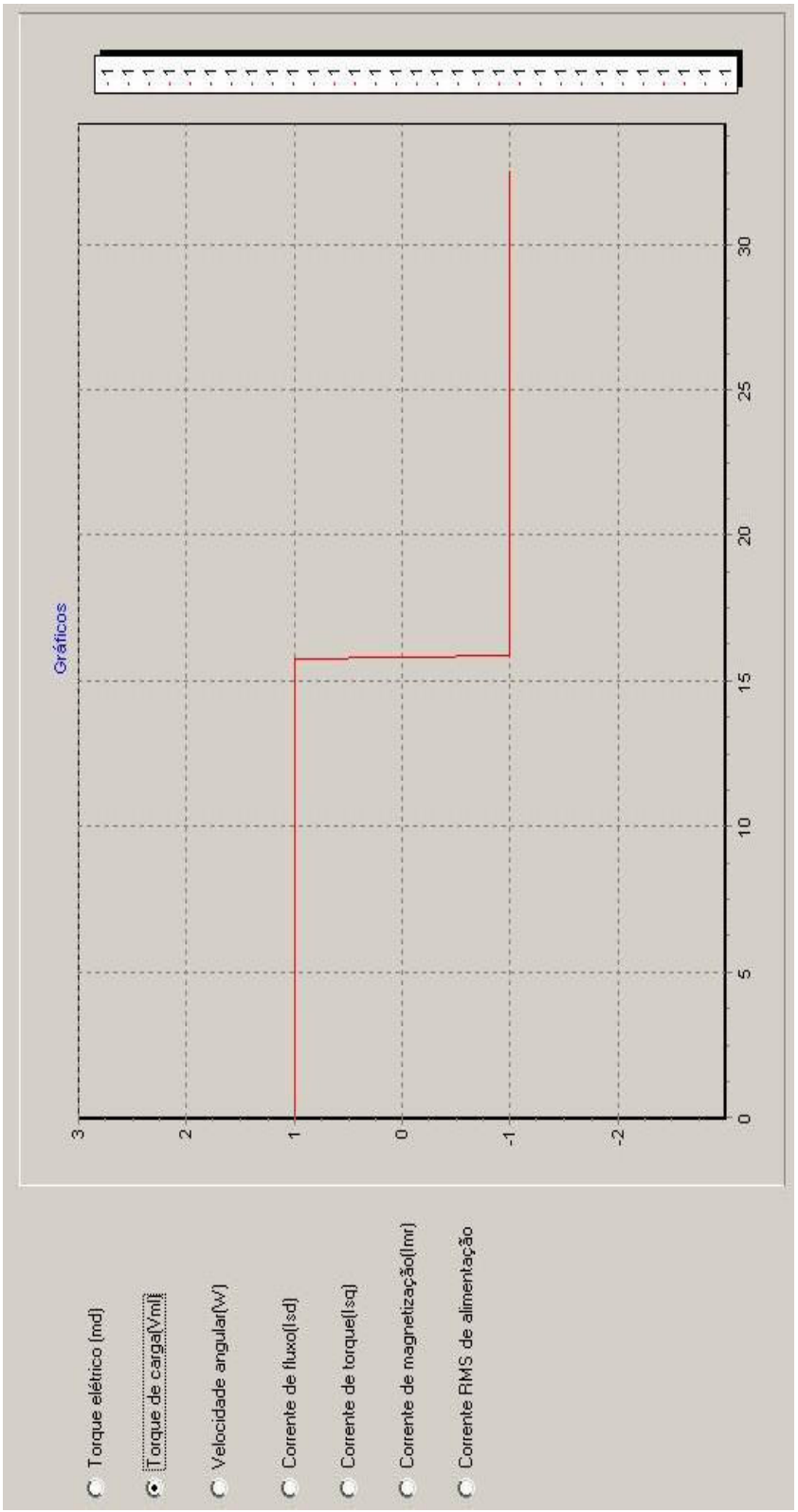


Figura 4.8 - Torque da carga (Exemplo1).

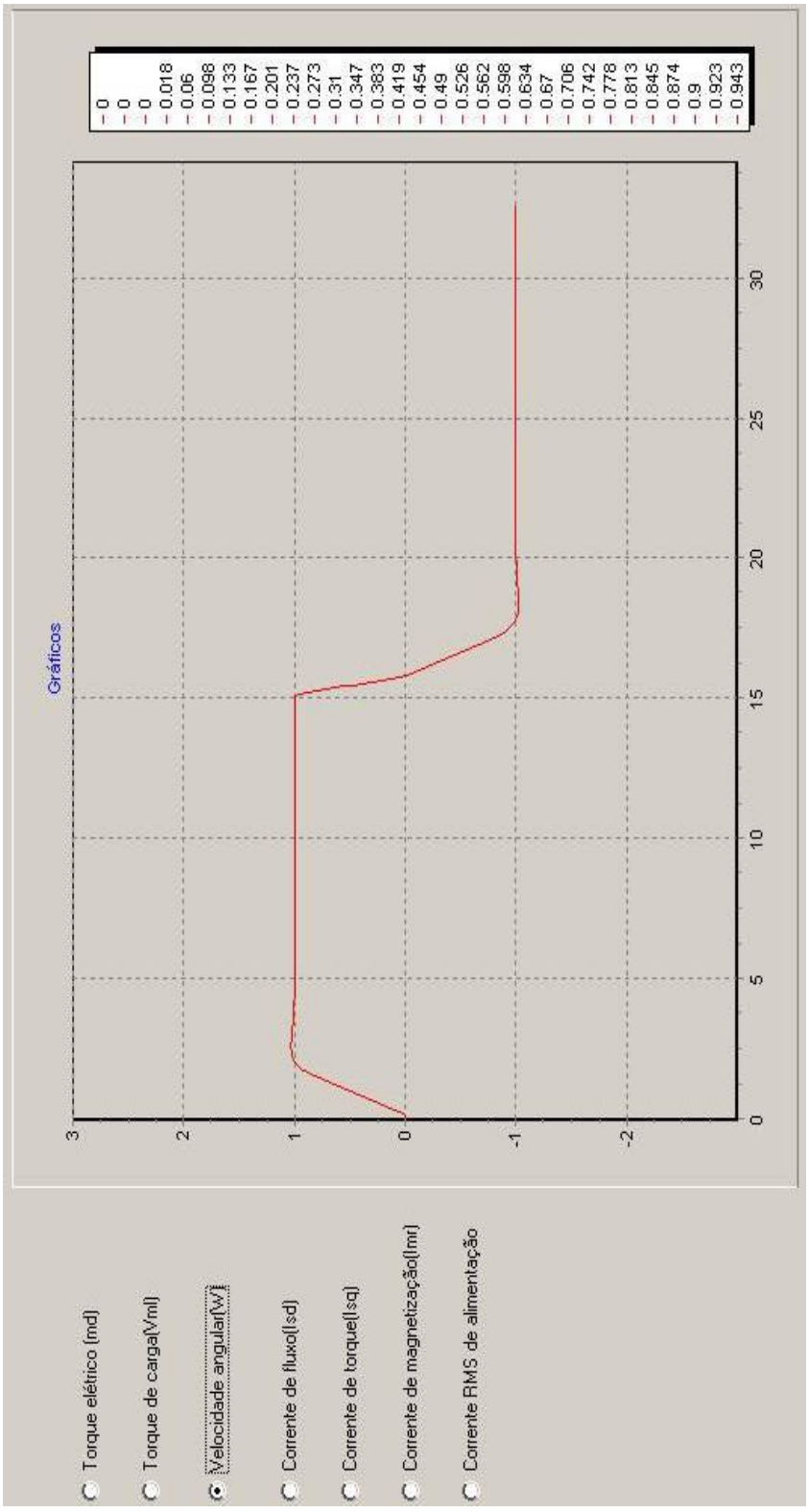


Figura 4.9 – Velocidade angular (Exemplo1).

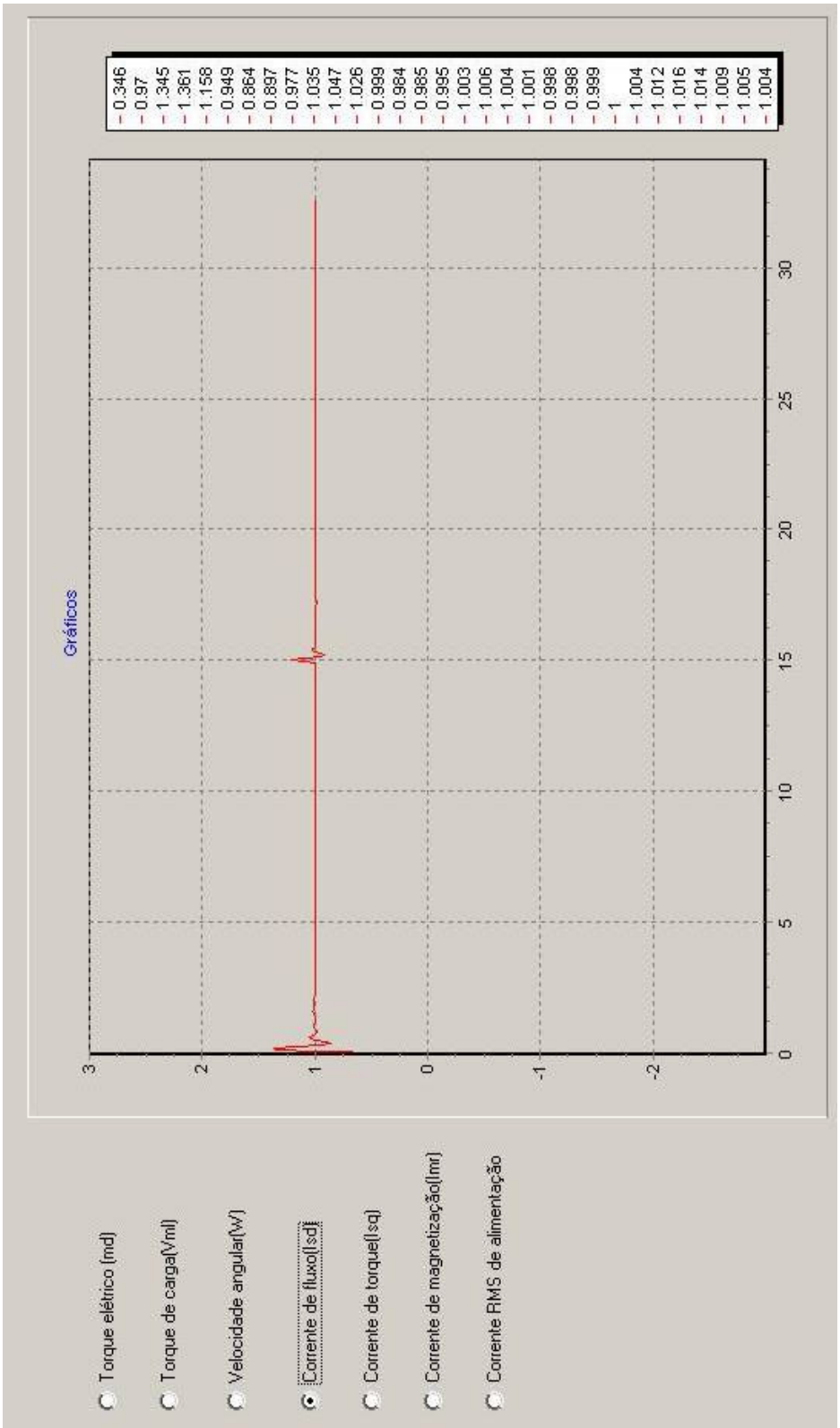


Figura 4.10 - Corrente de fluxo (Exemplo 1).

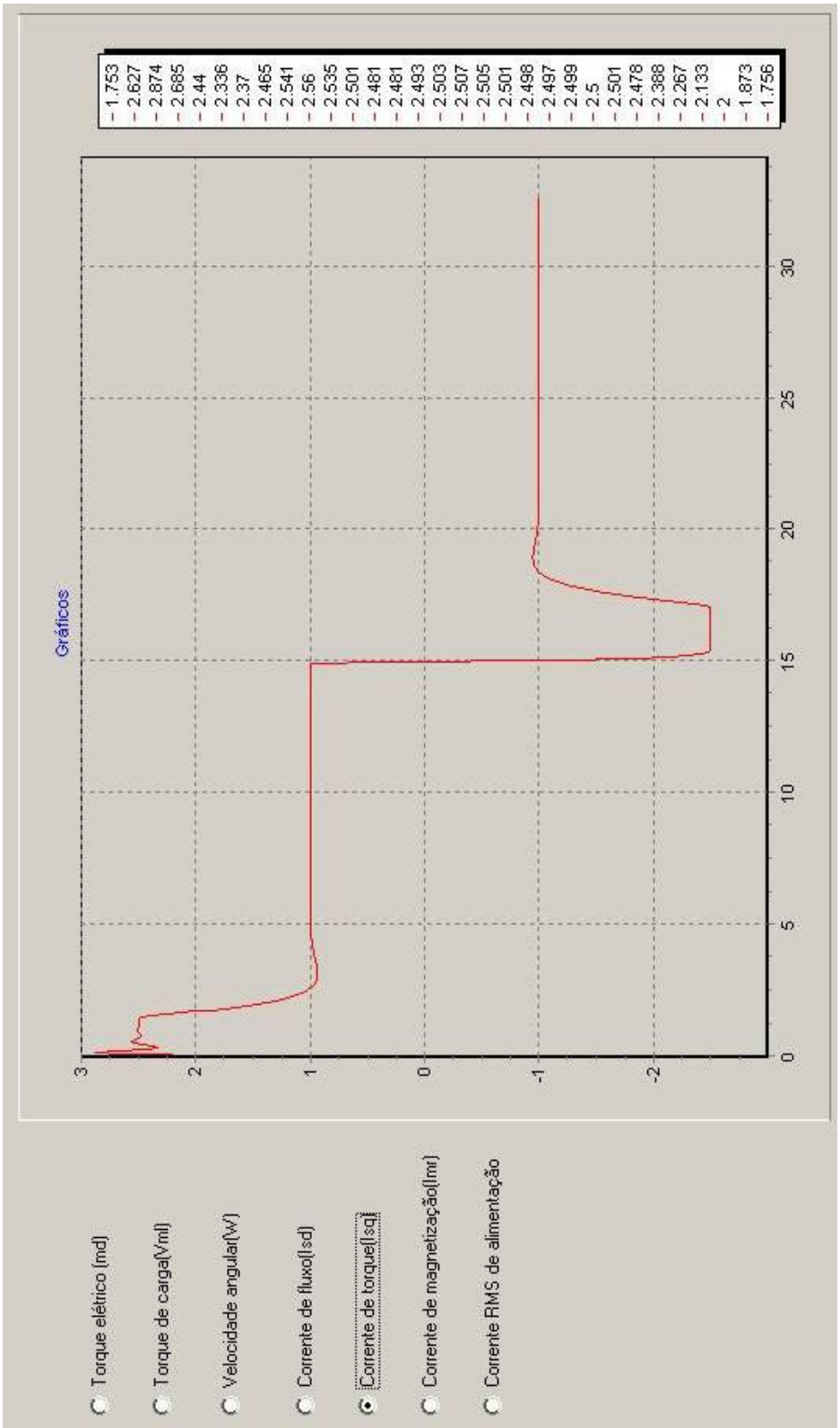


Figura 4.11 - Corrente proporcional ao torque (Exemplo 1).

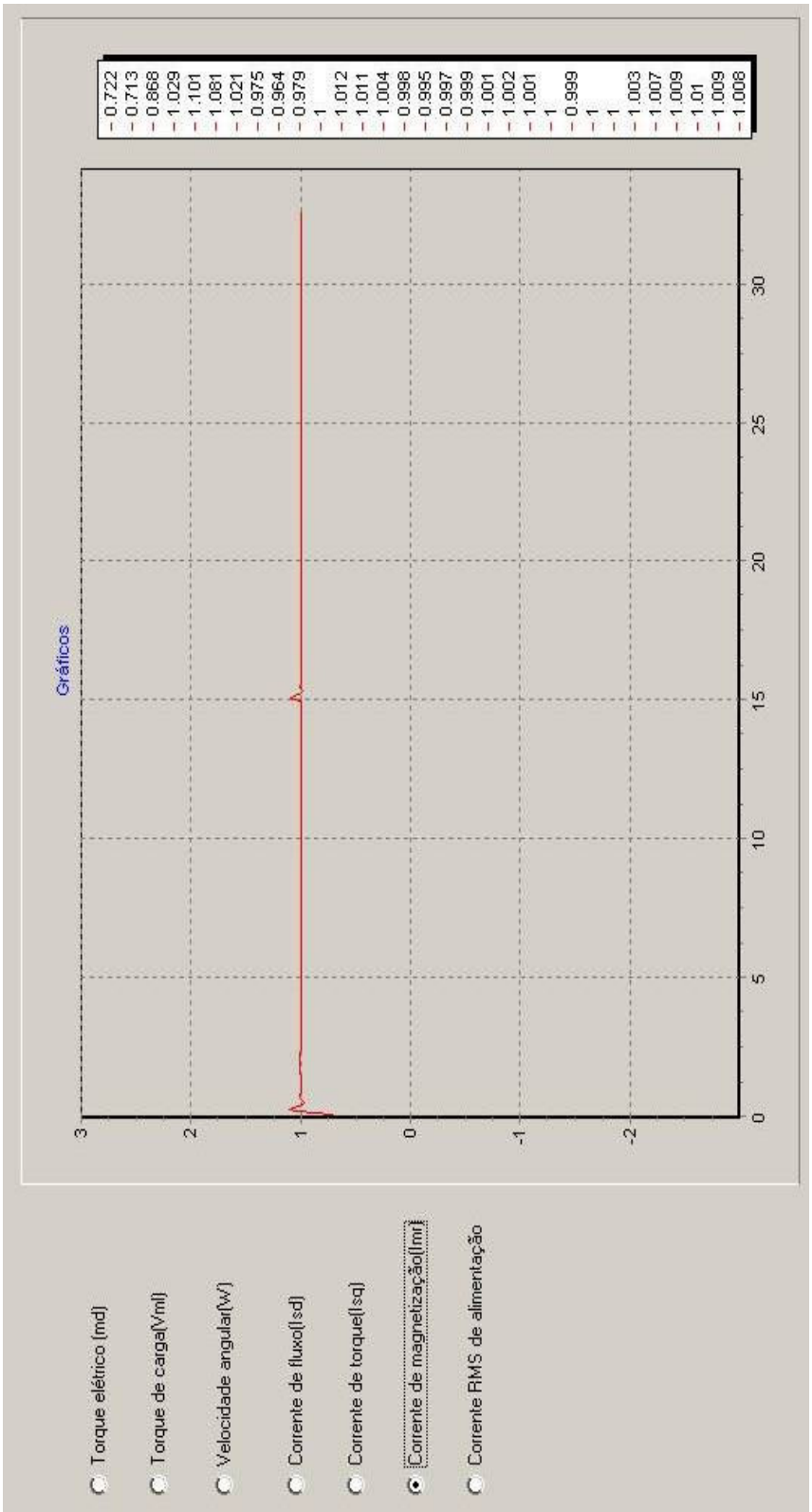


Figura 4.12 – Corrente de magnetização (Exemplo 1).

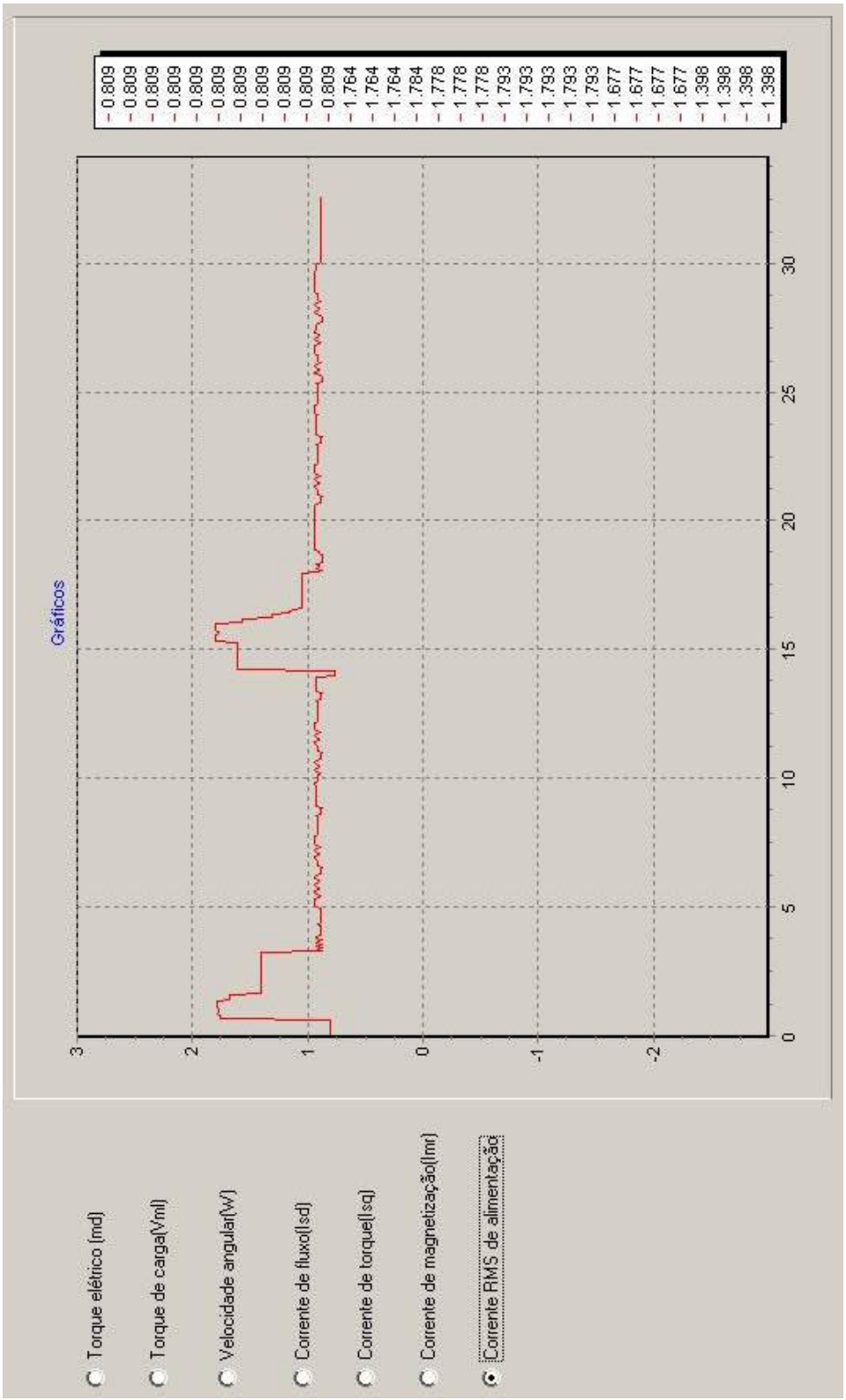


Figura 4.13 – Valor da corrente de alimentação (Exemplo 1).

Exemplo 2 - Motor com variação de velocidade e de carga com os seguintes parâmetros indicados abaixo.

- Velocidade: Opção 2; Wref1 = 0 [pu]; Wref2 = 0.5 [pu]; Wref3 = 1 [pu]; Wref4 = 1 [pu]; Wref5=1 [pu].
- Parâmetros de carga: Degrau; Torque máximo = 1 [pu]; Torque mínimo = 0 [pu]; Duração do torque mínimo = 28 [s].

As Figuras 4.14 a 4.20 ilustram os resultados da simulação deste exemplo.

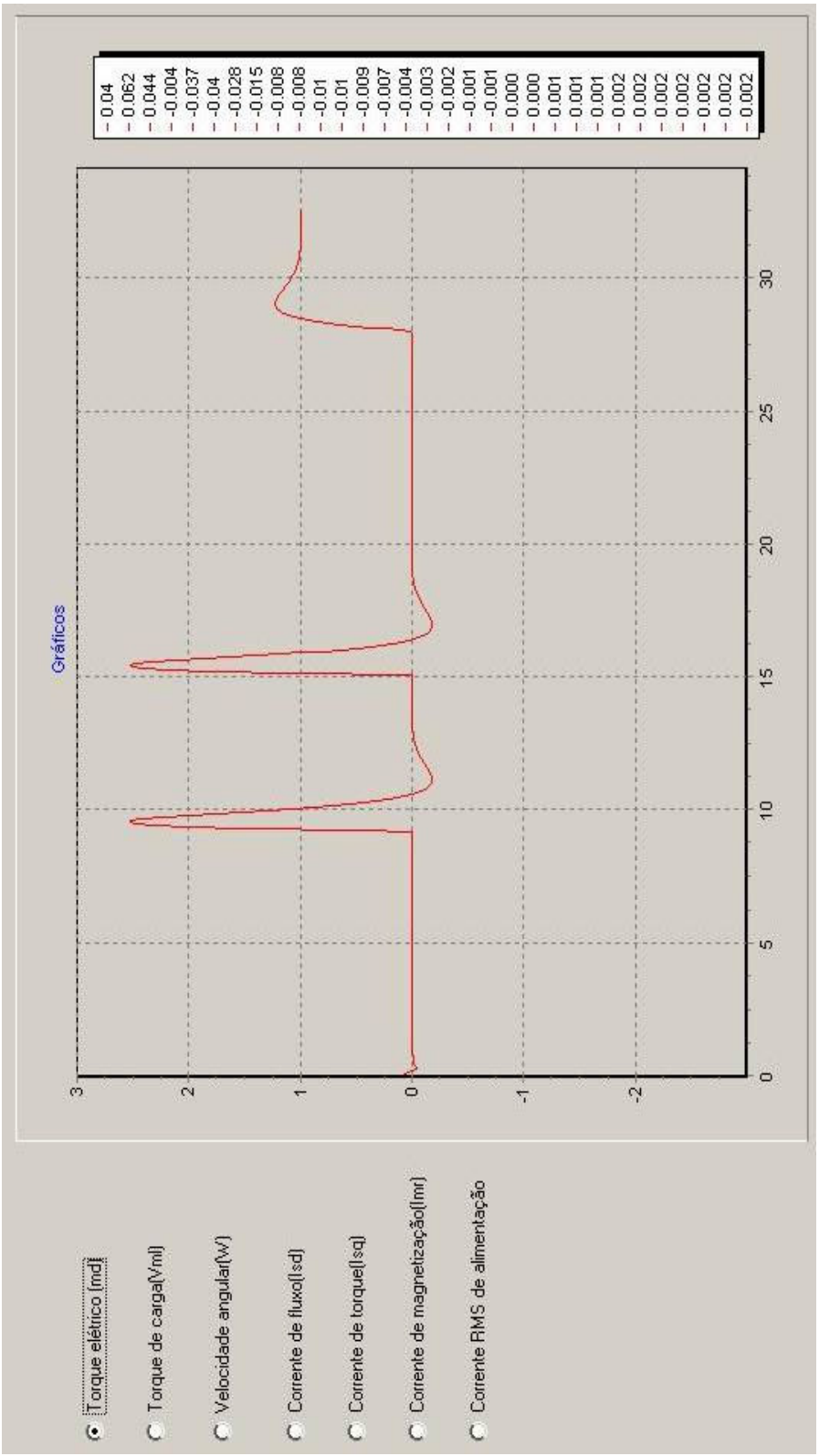


Figura 4.14 – Torque elétrico (Exemplo 2).

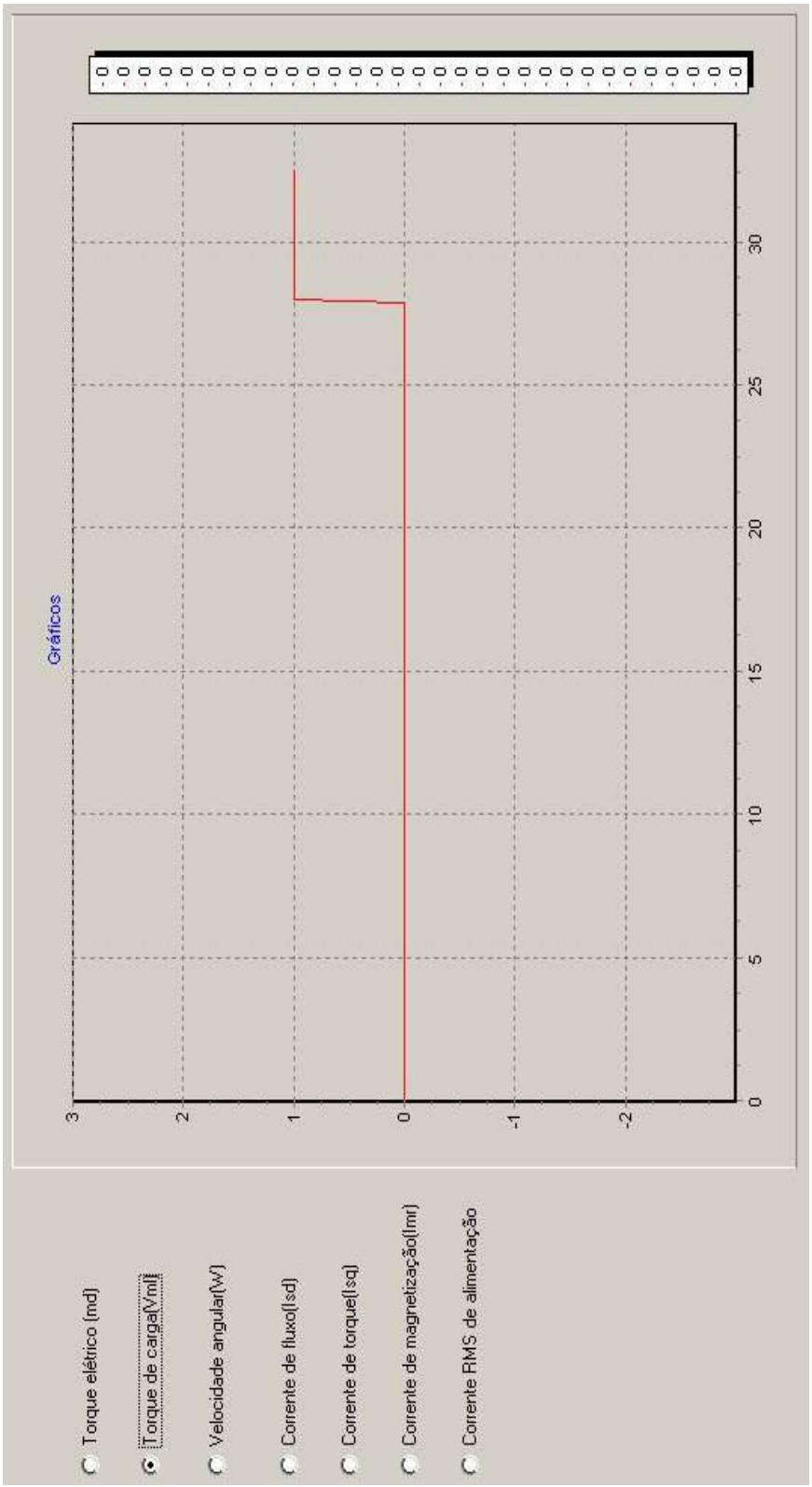


Figura 4.15 – Torque de carga (Exemplo 2).

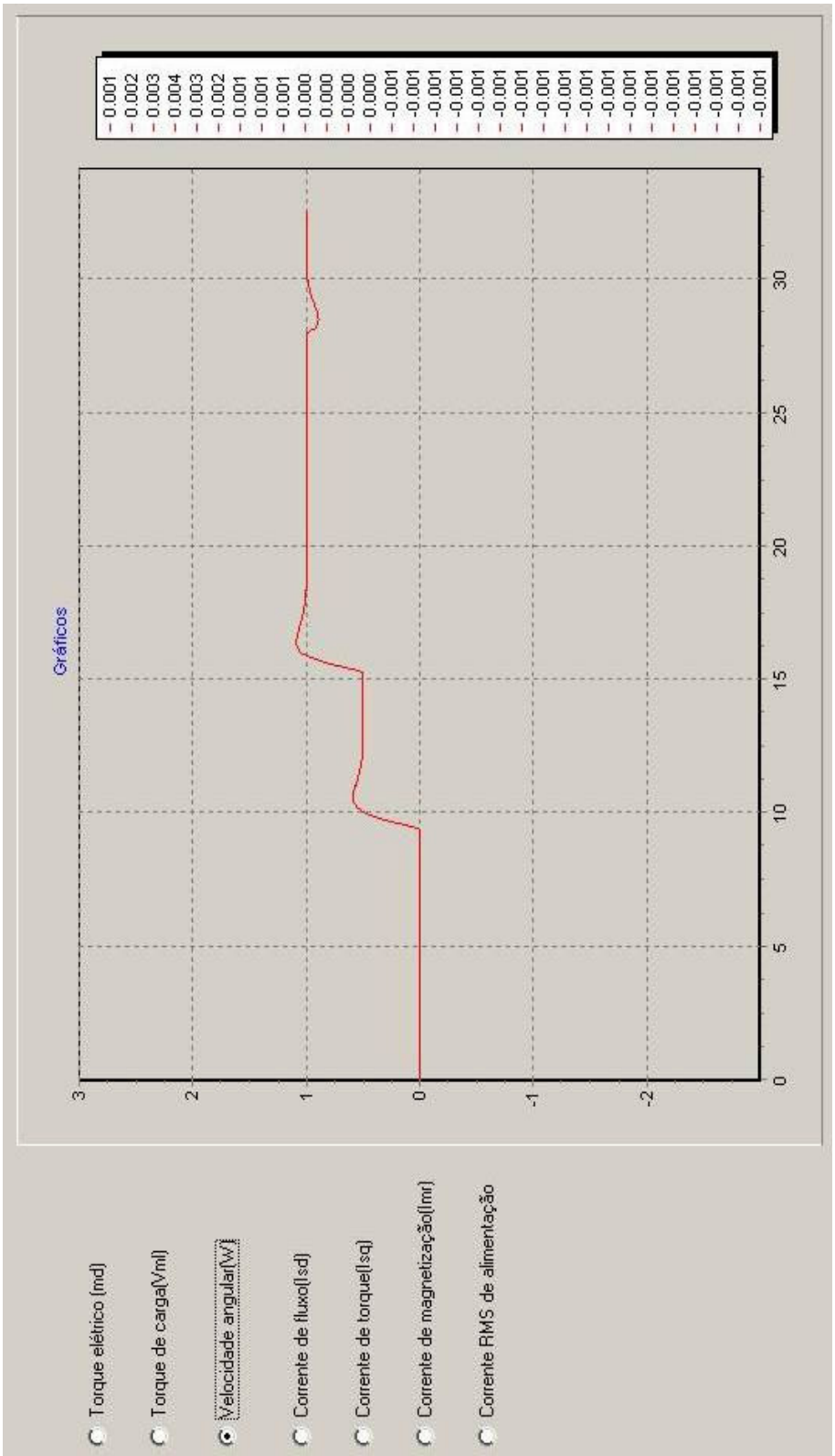


Figura 4.16 – Velocidade angular (Exemplo 2).

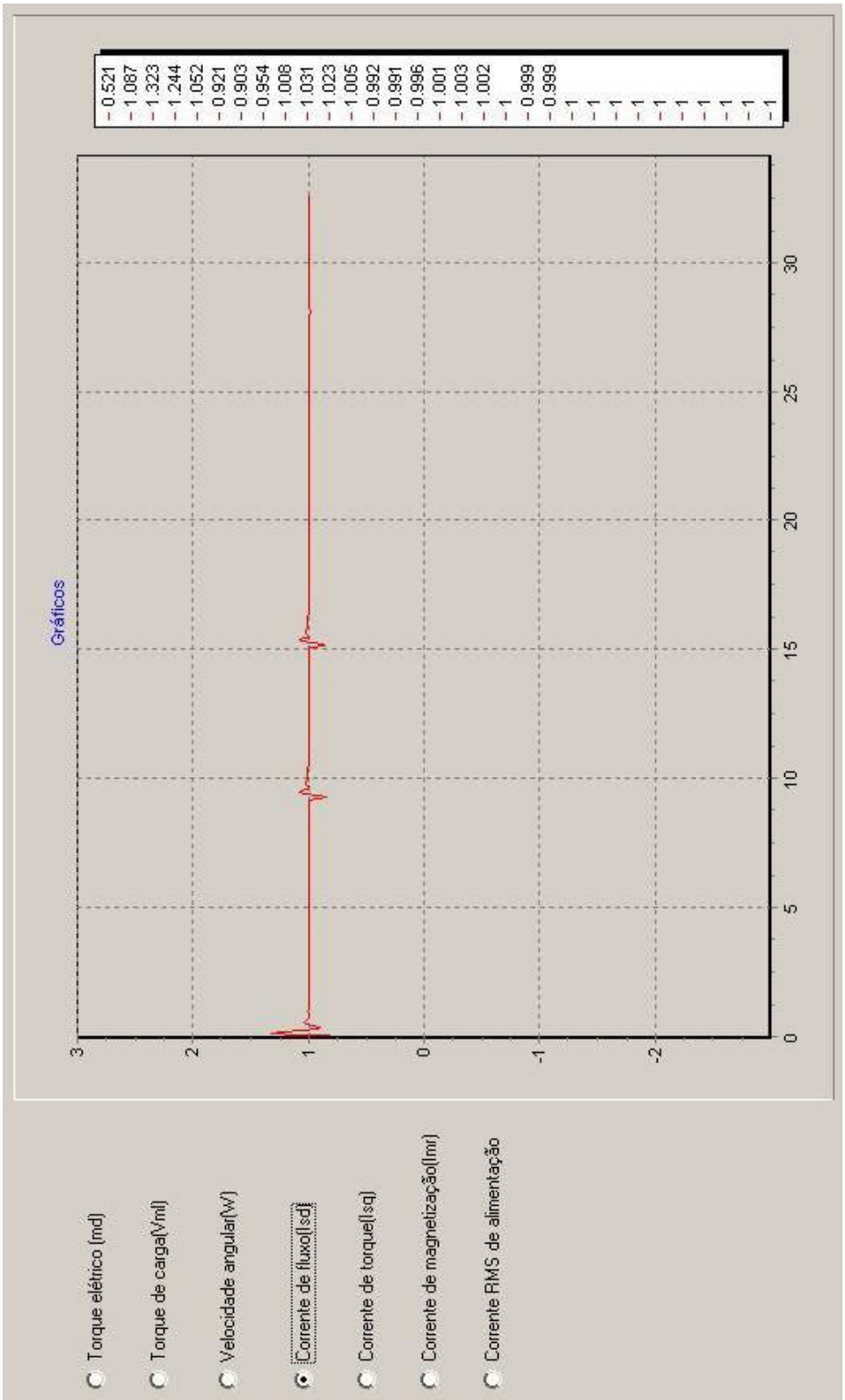


Figura 4.17 – Corrente de fluxo (Exemplo 2).

Figura 4.18 – Corrente proporcional ao torque (Exemplo 2).

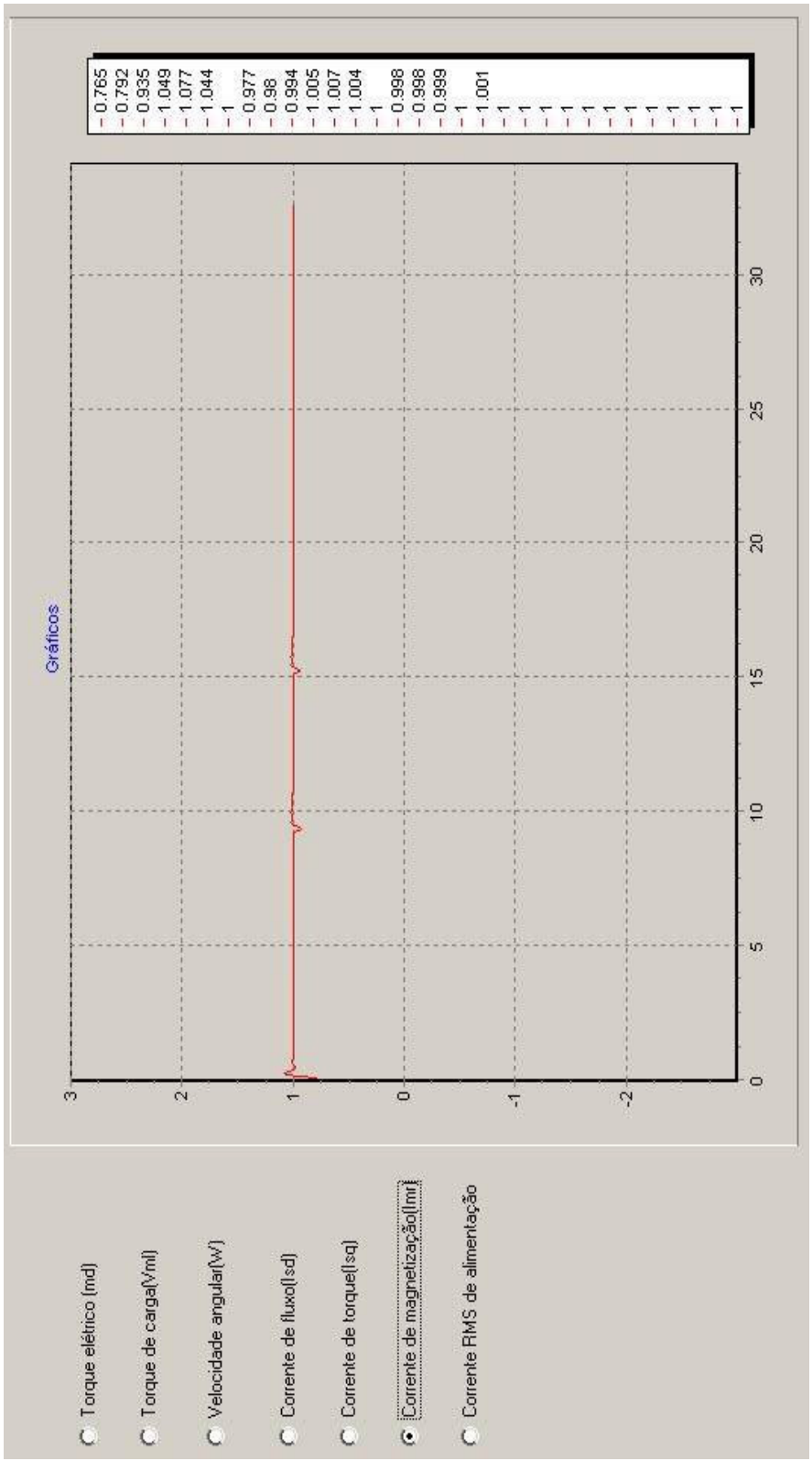


Figura 4.19 – Corrente de magnetização (Exemplo 2).

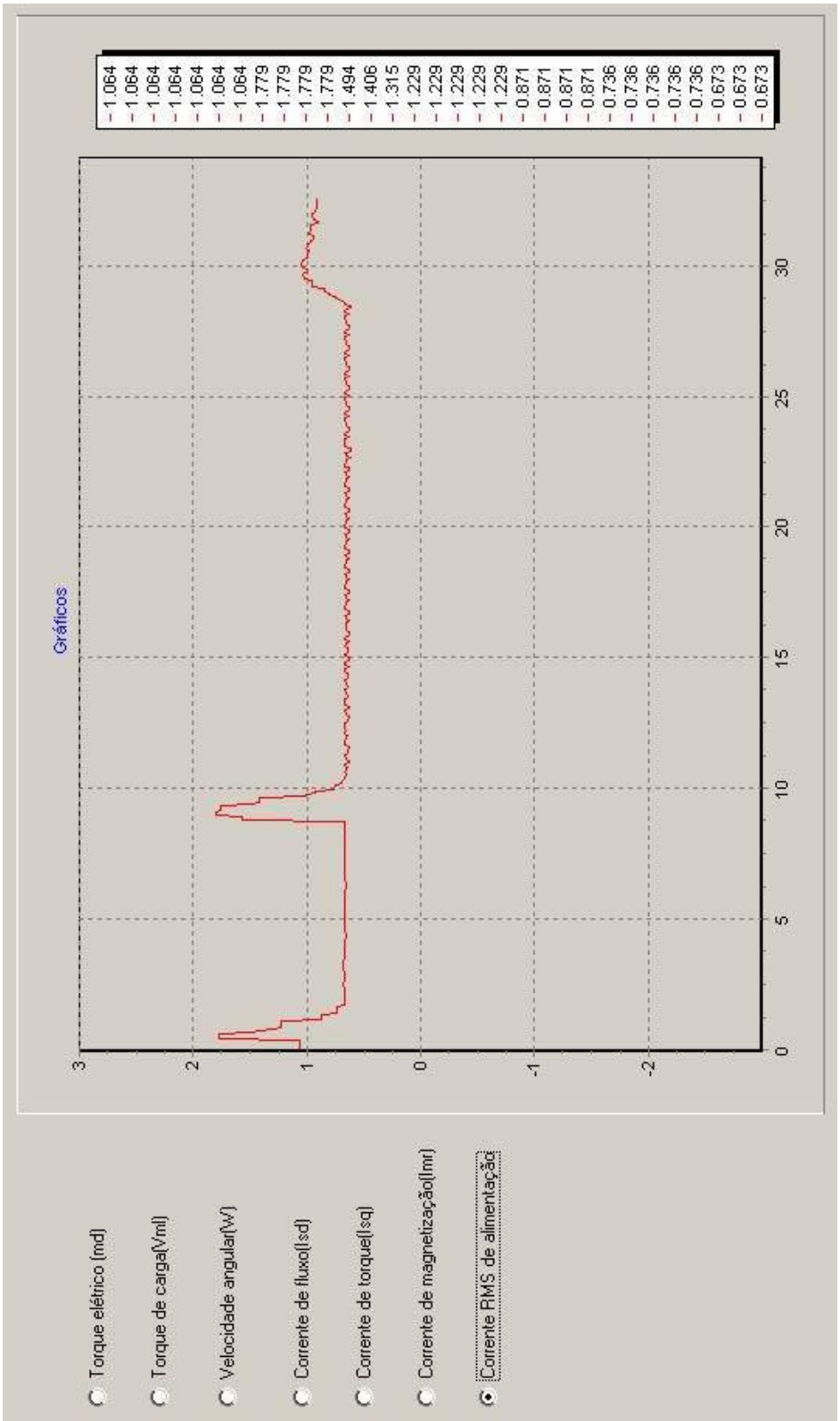


Figura 4.20 – Corrente de alimentação (Exemplo 2).

Capítulo 5 - Implementação Prática do Controle Vetorial

Depois da verificação dos resultados da estratégia de controle através de simulações, foi implementada uma bancada experimental para a realização de ensaios práticos. A implementação foi realizada em duas etapas, uma com o desenvolvimento do hardware da bancada, e outra com o desenvolvimento de um software para a realização do controle vetorial em tempo real baseado no programa de simulação.

5.1 – Estrutura do Hardware

Este item irá mostrar com detalhes o hardware desenvolvido para a bancada experimental. A estrutura do hardware está esquematizada na Figura 5.1 e na Figura 5.2 tem-se a foto da bancada montada. A seguir será feita uma descrição das partes constituintes da mesma.

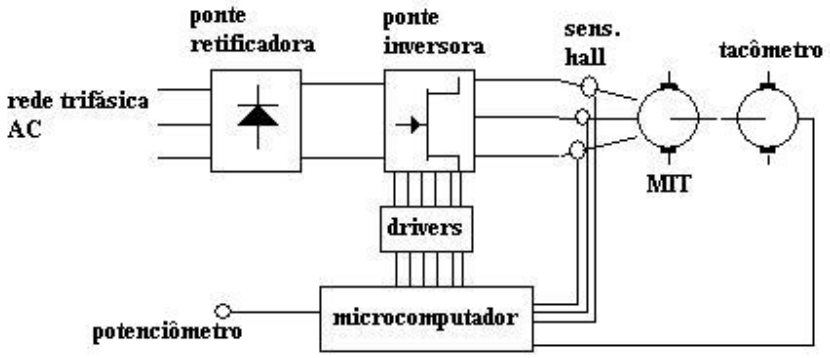


Figura 5.1 – Estrutura do hardware da bancada experimental.

Figura 5.2 – Visão geral da bancada montada.

5.1.1 – Alimentação de Corrente Contínua

Circuito constituído de uma ponte trifásica composta por seis diodos de potência, fusível de proteção, indutor e capacitor de filtro, que praticamente constituem o link DC do sistema (Figura 5.3). A foto da Figura 5.4 ilustra a montagem efetuada.

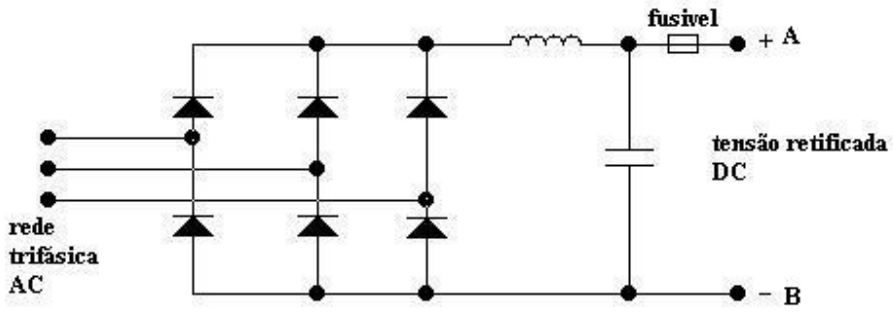


Figura 5.3 – Alimentação de corrente contínua.

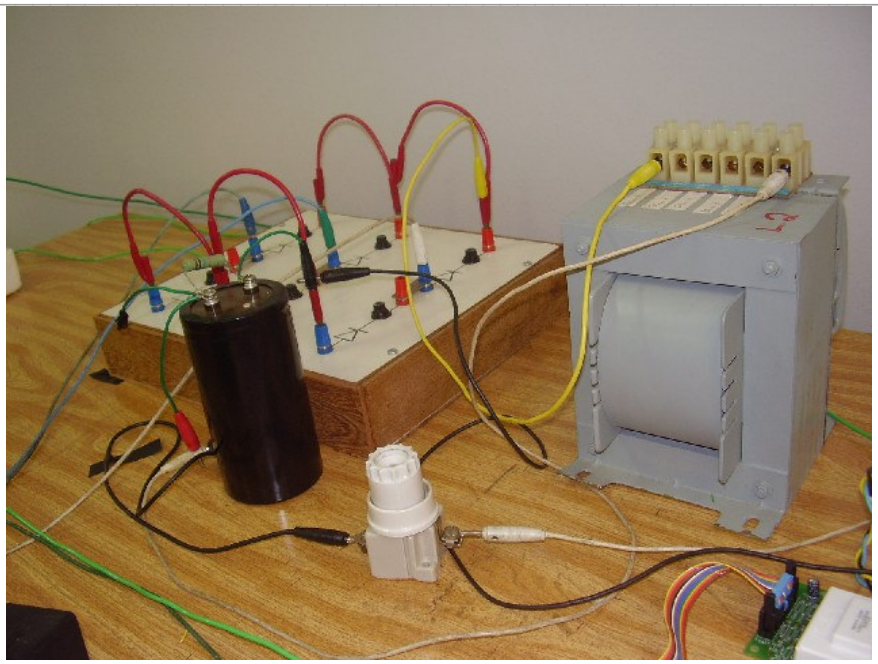


Figura 5.4 – Foto da alimentação de corrente contínua.

5.1.2 - Ponte Inversora

Constituída por seis transistores do tipo IGBT (Insulated Gate Bipolar Transistor) e diodos de livre circulação associados (Figura 5.5), com montagem estrutural ilustrada na Figura 5.6.

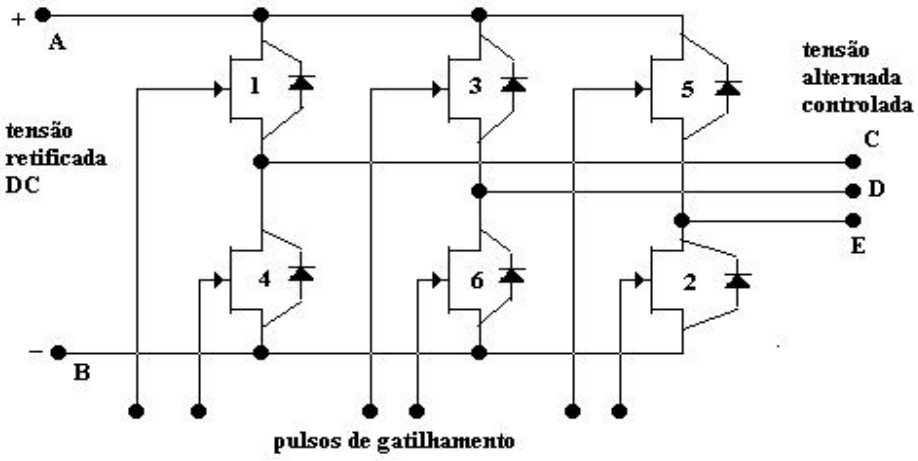


Figura 5.5 - Ponte inversora.

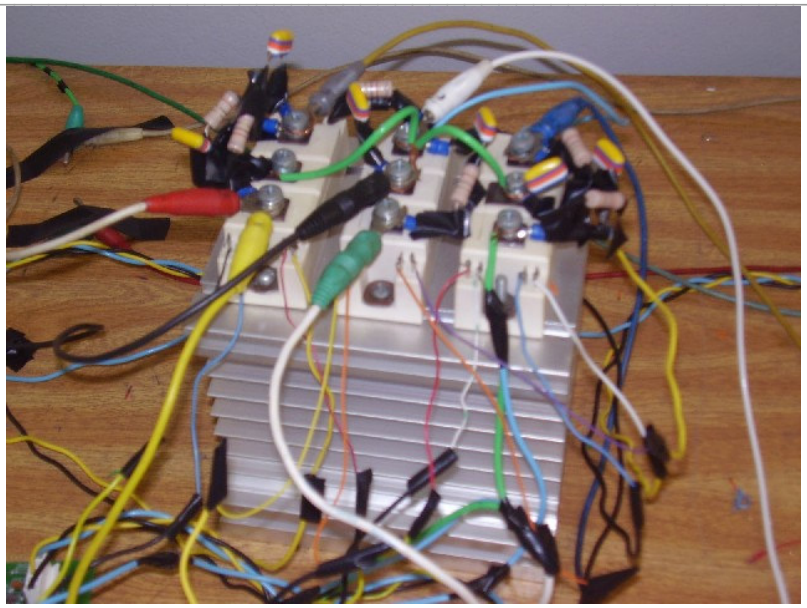


Figura 5.6 – Estrutura da ponte inversora.

5.1.3 - Drivers para gatilhamento dos IGBTs

Neste trabalho foram utilizados drivers dedicados para gatilhamento de IGBTs. No Anexo 1 encontra-se as folhas de dados dos mesmos. Cada circuito de driver é capaz de gatilhar um par de IGBTs da ponte inversora, utilizando-se no total três drivers (Figura 5.7) na montagem da bancada. O funcionamento do driver é simples, pares de pulsos de controle são aplicados em pinos dos conectores de entrada e eles são repassados aos terminais dos gates e dos emissores dos IGBTs. O circuito do driver possui proteção no caso de ser aplicado dois pulsos ao mesmo tempo em nível alto nas entradas de comando de dois IGBTs na mesma fase da ponte. Os pulsos são bloqueados e é gerada uma sinalização de erro. Outra proteção existe quando a tensão de alimentação do sistema está reduzida, os IGBT's são bloqueados e é gerada uma sinalização. O driver também possui uma configuração para inserir tempos de retardo entre o início de um pulso de ativação de um IGBT e o bloqueio do outro da mesma fase. No modelo de driver utilizado estes pinos de configuração não estão acessíveis ao usuário.

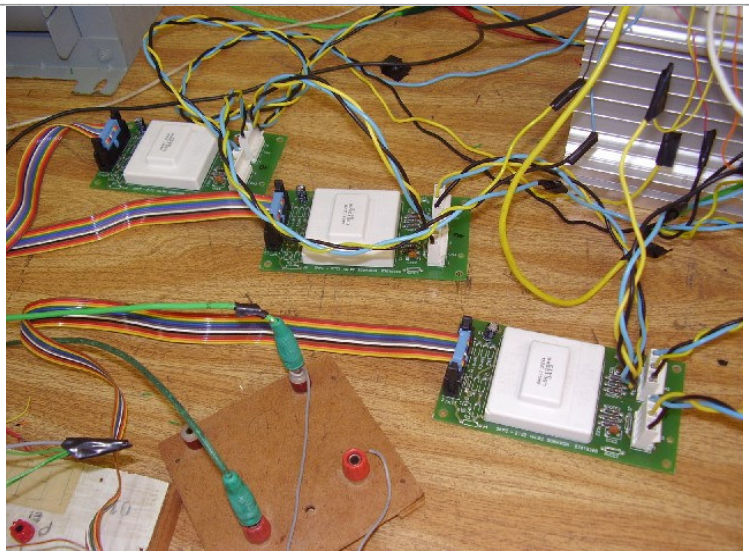


Figura 5.7 – Foto dos drivers utilizados.

Como pode ser visto na figura 5.7 os drivers já vem montados em uma placa de circuito impresso, cuja configuração é mostrada abaixo:

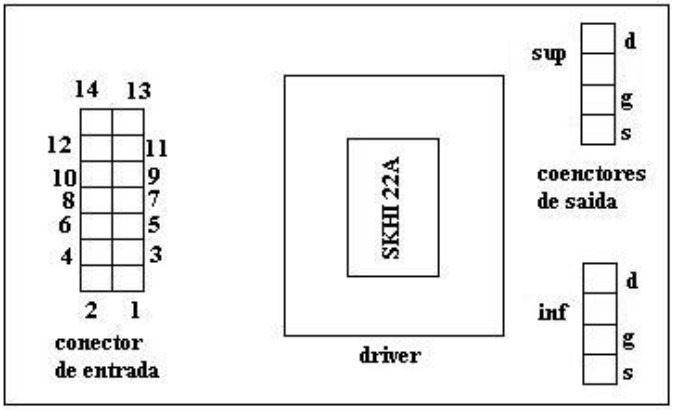


Figura 5.8 – Esquemático da placa do driver.

Onde: d = dreno; g = gate; s = source (sendo ligados nos respectivos IGBT's). As funções de cada pino do conector da placa estão descritadas abaixo (NF significa nenhuma função).

1 – NF.	5 – NF.	9 – Vs (15Vcc).	13 – NF.
2 – Vin inf.	6 – NF.	10 – GND.	14 – NF.
3 – Erro.	7 – NF.	11 – GND.	
4 – Vin sup.	8 – Vs (15Vcc).	12 – NF.	

O pino 3 (Erro) fica em nível baixo se uma situação de erro ocorrer, ou em nível alto caso o driver esteja funcionamento corretamente.

5.1.4 - Interface com Microcomputador

Como interface entre os drivers de gatilhamento dos IGBTs do hardware da bancada e o microcomputador utilizado para implementar a lógica de controle, foi empregado um circuito para adequar os níveis das tensões de comando utilizadas.

Este circuito foi adicionado entre os pinos de saída do periférico do computador e as entradas de comando dos drivers de gatilhamento. Os drivers operam com níveis lógicos de 0 e 15 [V] para o gatilhamento dos IGBT's. As saídas do periférico trabalham com 0 e 5 [V]. Foram utilizados transistores de sinais (BC 458) e resistores de baixa potência (1/8 [W]; 4,7 [KΩ]) conforme ilustrado na Figura 5.8. A foto da Figura 5.9 ilustra a montagem realizada.

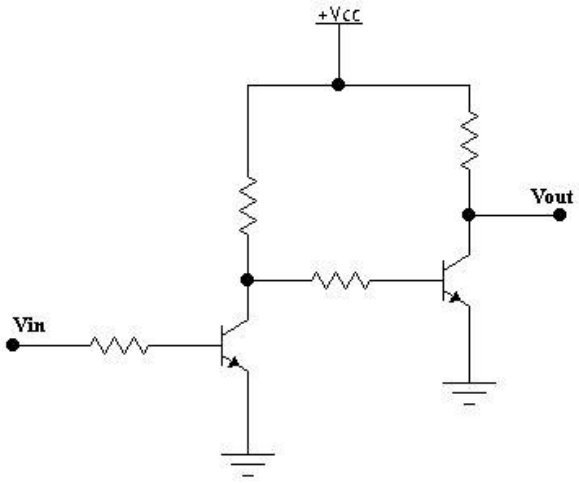


Figura 5.9 – Circuito de interface.

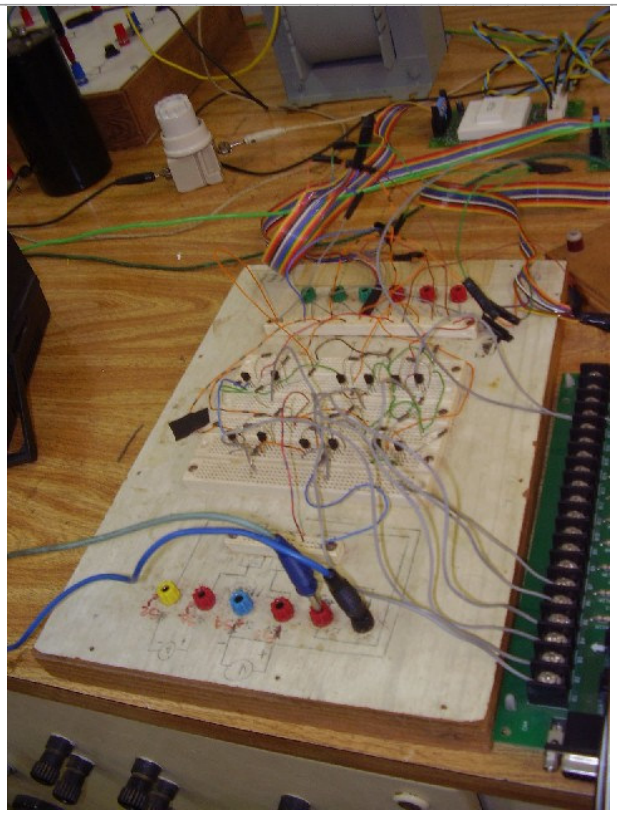


Figura 5.10 – Montagem do circuito de interface.

5.1.5 – Máquinas Elétricas

Na montagem da bancada foi utilizado um conjunto de máquinas acopladas entre si através dos seus eixos (Figura 5.10). Um motor de indução, uma máquina de corrente contínua e uma máquina síncrona. A primeira será o motor a ser controlado. A segunda será utilizada como taco-gerador para fornecer a informação de rotação do sistema. E a terceira máquina funcionará como gerador alimentando um conjunto de cargas elétricas variáveis, usadas para simular variações de carga no eixo do MIT. O motor de indução possui as seguintes características seguintes (dados de placa). Foi utilizada a configuração 220 V e 8,8 A.

Tipo EA4.5A
V – Lig 760 440 380 220
KW 2.25
FP 0,82
RPM 1700
Rotor 77 λ
V – A 20.2

A – 2.6 4.4 5.1 8.8
Fases 3
HZ 60
Isol B
Rend 0.82
Norma ABNT

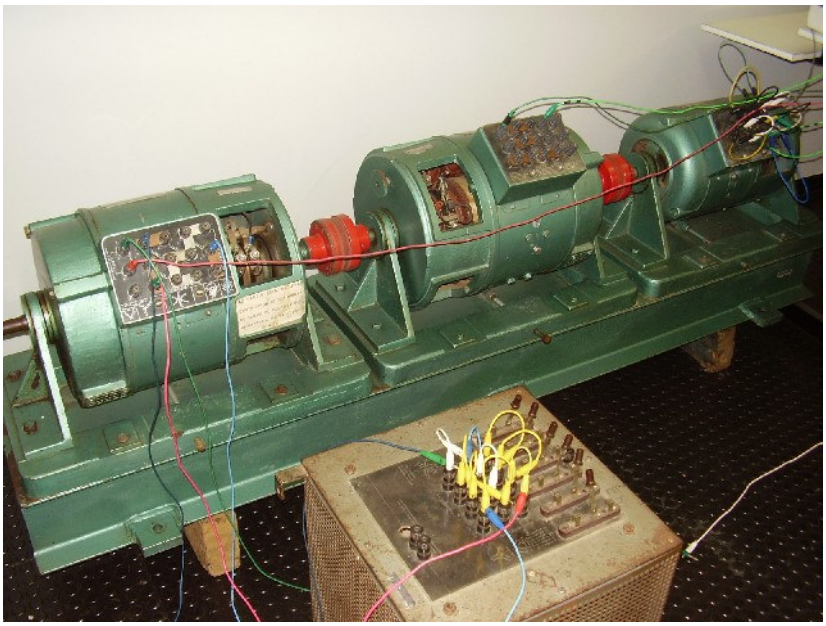


Figura 5.11 – Conjunto de máquinas utilizadas.

Na saída dos terminais de armadura da máquina de corrente contínua foi utilizado um divisor de tensão resistivo para fornecer a informação da velocidade do sistema. A Tabela 5.1 mostra algumas medidas realizadas, indicando alguns valores de velocidade do MIT, com valores de tensões correspondentes a de um tacômetro de medição e da informação do divisor resistivo. Nota-se que as relações de velocidade são proporcionais e bem lineares. A Figura 5.11 ilustra a relação entre os valores de velocidade do motor e de tensões do tacômetro.

Tabela 5.1 – Medições de corrente e informações de velocidade.

Velocidade do MIT [rpm]	Saída direta do tacômetro [V]	Saída após divisor resistivo [V]
1691	5,37	2,71
1701	5,42	2,73
1711	5,44	2,74
1720	5,47	2,76
1727	5,50	2,77
1737	5,53	2,79

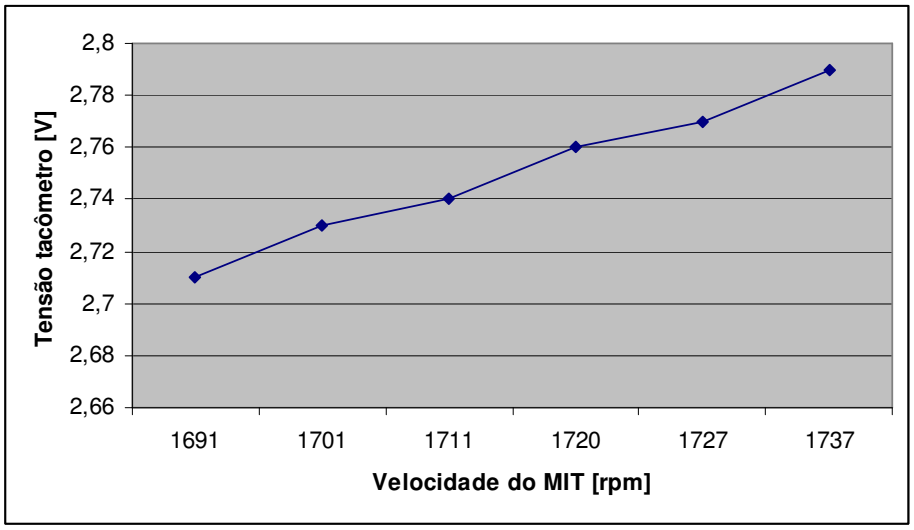


Figura 5.12 – Relações entre velocidade do MIT e tensão do tacômetro.

5.1.6 - Sensores Hall

Para a leitura das correntes das fases do MIT foram utilizados três sensores de efeito Hall. Foram realizados ensaios para verificar a relação entre os valores eficazes das correntes de fase e as tensões correspondentes dos transdutores. As

fases do motor foram alimentadas com correntes senoidais e foram medidas as tensões de saída correspondentes dos sensores. A Tabela 5.2 mostra alguns valores medidos e a Figura 5.12 ilustra o gráfico resultante, onde nota-se que a relação é razoavelmente linear. A foto da Figura 5.13 ilustra os sensores usados.

Tabela 5.2 – Relação entre corrente e sinal de saída dos sensores Hall.

Valores RMS da fase [A]	Valores RMS de saída do sensor [V]
8,8	1,79
8,0	1,63
7,5	1,53
7,0	1,41
6,5	1,33
6,0	1,22

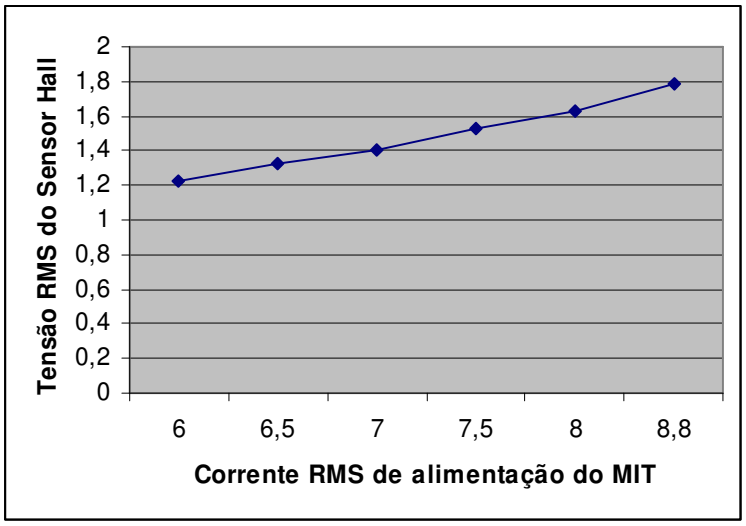


Figura 5.13 – Gráfico de relação de saída da sonda Hall.

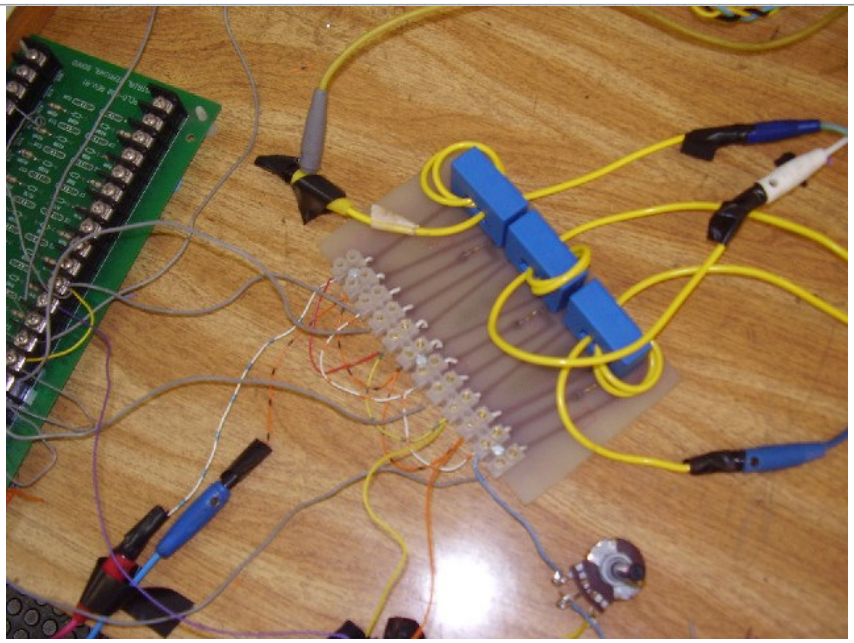


Figura 5.14 – Foto dos sensores de corrente utilizados.

5.1.7 – Placa de Aquisição de Dados

Foi utilizado uma placa de aquisição de dados para realizar a interface entre o hardware da bancada e o microcomputador empregado para implementar a lógica de controle. No Anexo 2 encontra-se as folhas de dados da placa utilizada. São utilizadas cinco entradas analógicas da mesma para ler informações do sistema. Três delas adquirem as informações das correntes de fase do motor de indução. As outras duas entradas analógicas são utilizadas para adquirir as informações da velocidade do motor e da referência de entrada da velocidade desejada (dada por meio de um potenciômetro alimentado por uma fonte de tensão estabilizada).

5.2 – Implementação do Software de Controle em Tempo Real

Semelhante ao software de simulação apresentado no capítulo anterior, o programa de controle em tempo real é divido em duas partes: uma interface com o usuário para entrada de dados, parâmetros de regulação, etc. e visualização

gráfica de resultados; um módulo de controle que implementa a estratégia de acionamento em tempo real, onde pode-se variar a velocidade nominal do sistema utilizando um sinal de referência. Na interface com o usuário inicialmente têm-se as opções de iniciar um controle ou traçar os gráficos de um ensaio já realizado anteriormente. A Figura 5.14 ilustra a tela de entrada de dados desta interface. Os parâmetros de regulação foram inicialmente baseados nos valores do programa de simulação. Porém, para o correto funcionamento do programa alguns deles tiveram que ser modificados devido a incertezas dos dados de placa do motor. Na simulação deve-se entrar com o algumas opções relacionadas com a carga. No programa de controle real estas opções não são necessárias, pois a carga real é aplicada no próprio eixo do motor controlado.

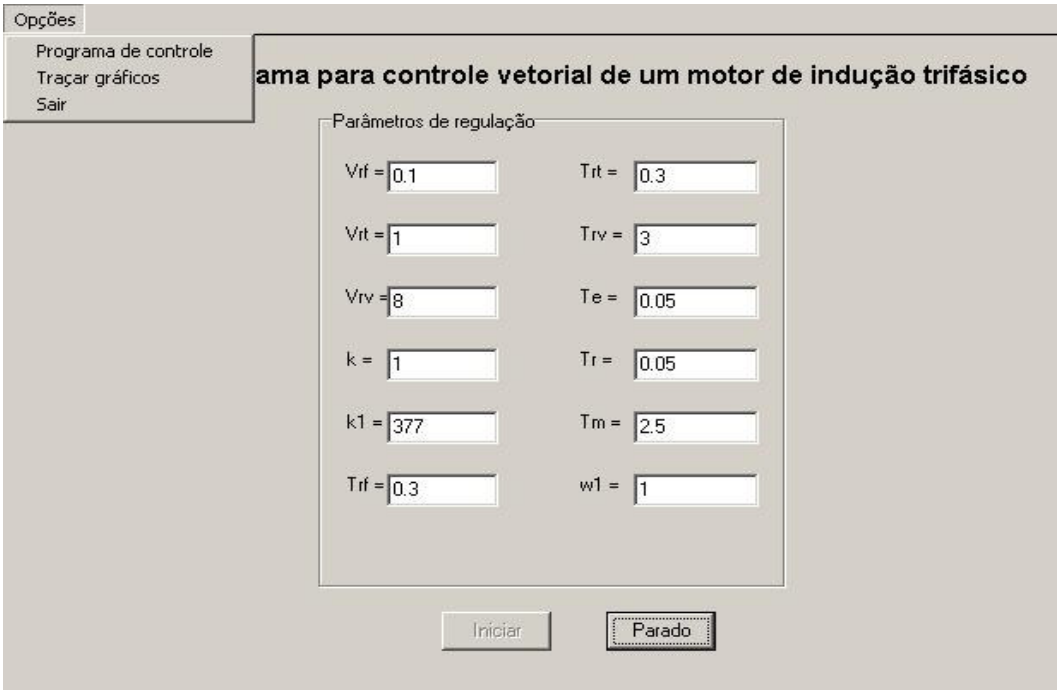


Figura 5.15 – Interface de entrada de dados.

Deve-se digitar um caminho e nome de arquivo para armazenar os dados de entrada ou recuperá-los posteriormente (figura 5.15). Ao acionar o botão “Iniciar”, o controle é iniciado e alguns dados adquiridos do sistema são armazenados nos primeiros vinte e cinco segundos do ensaio. Após este tempo os

dados não são mais registrados, mas o controle continua sendo executado normalmente até ser acionado o botão “Parado”, finalizando o controle e parando o acionamento do motor.

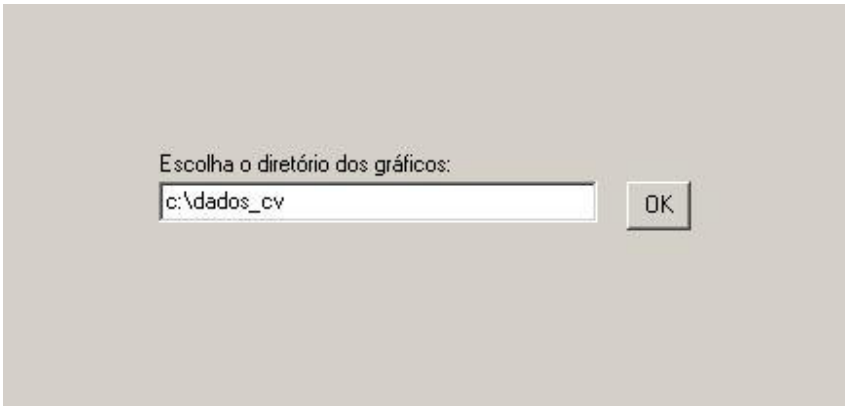
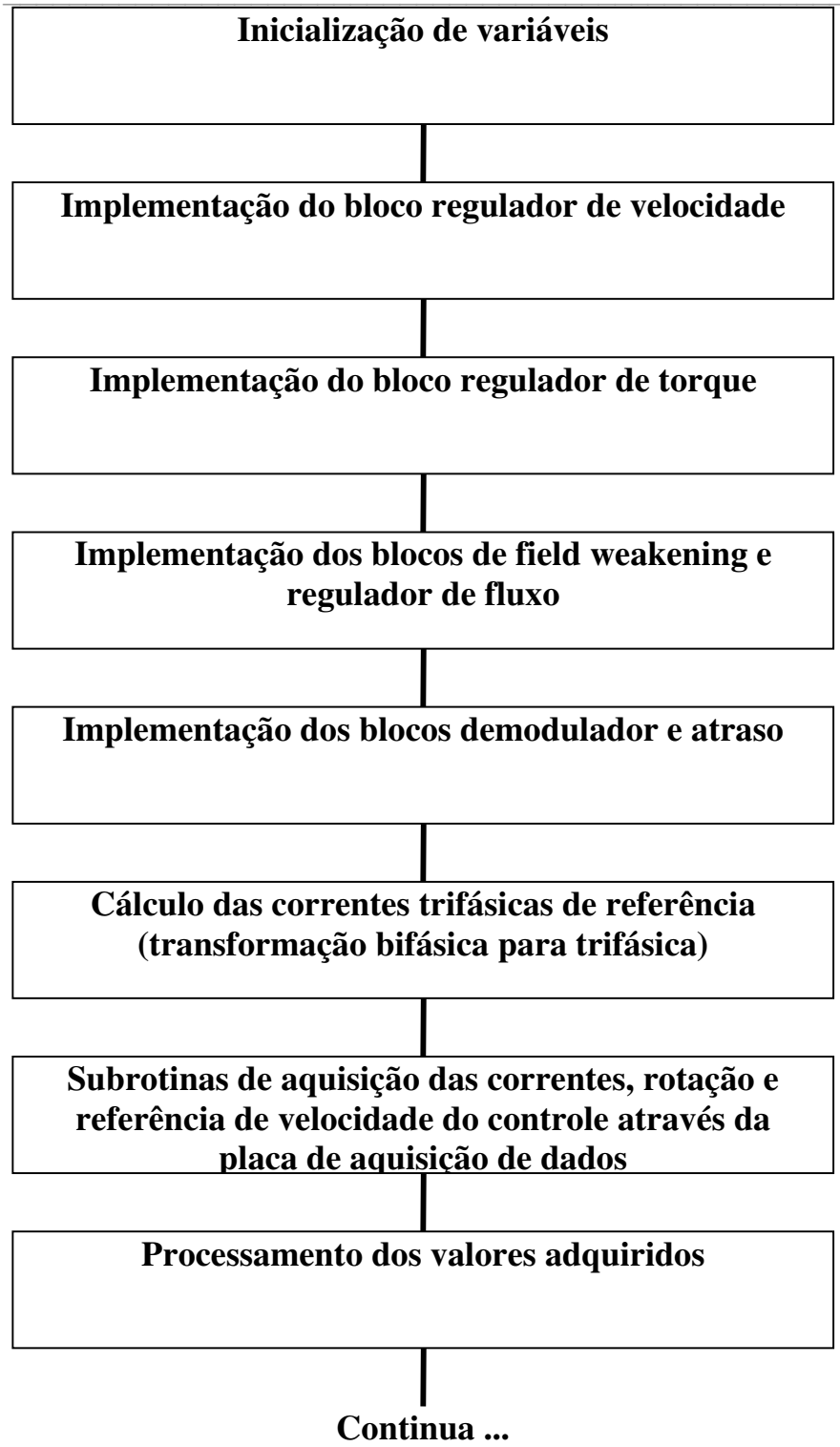


Figura 5.16 – Escolha do caminho para salvar ou ler arquivos de dados.

5.3 – Descrição do algoritmo de controle real

Para melhor entendimento deste algoritmo será apresentado um fluxograma funcional (Figura 5.16) referente à implementação real da malha de controle. O programa de controle real é baseado no software de simulação descrito no capítulo 3. As diferenças existentes são que no aplicativo de tempo real os valores das correntes, rotação e referência de velocidade são adquiridos pela placa de aquisição de dados. O código fonte do programa está listado no Anexo 4.


Inicialização de variáveis
Implementação do bloco regulador de velocidade
Implementação do bloco regulador de torque
Implementação dos blocos de field weakening e regulador de fluxo
Implementação dos blocos demodulador e atraso
Cálculo das correntes trifásicas de referência (transformação bifásica para trifásica)
Subrotinas de aquisição das correntes, rotação e referência de velocidade do controle através da placa de aquisição de dados
Processamento dos valores adquiridos
Continua ...

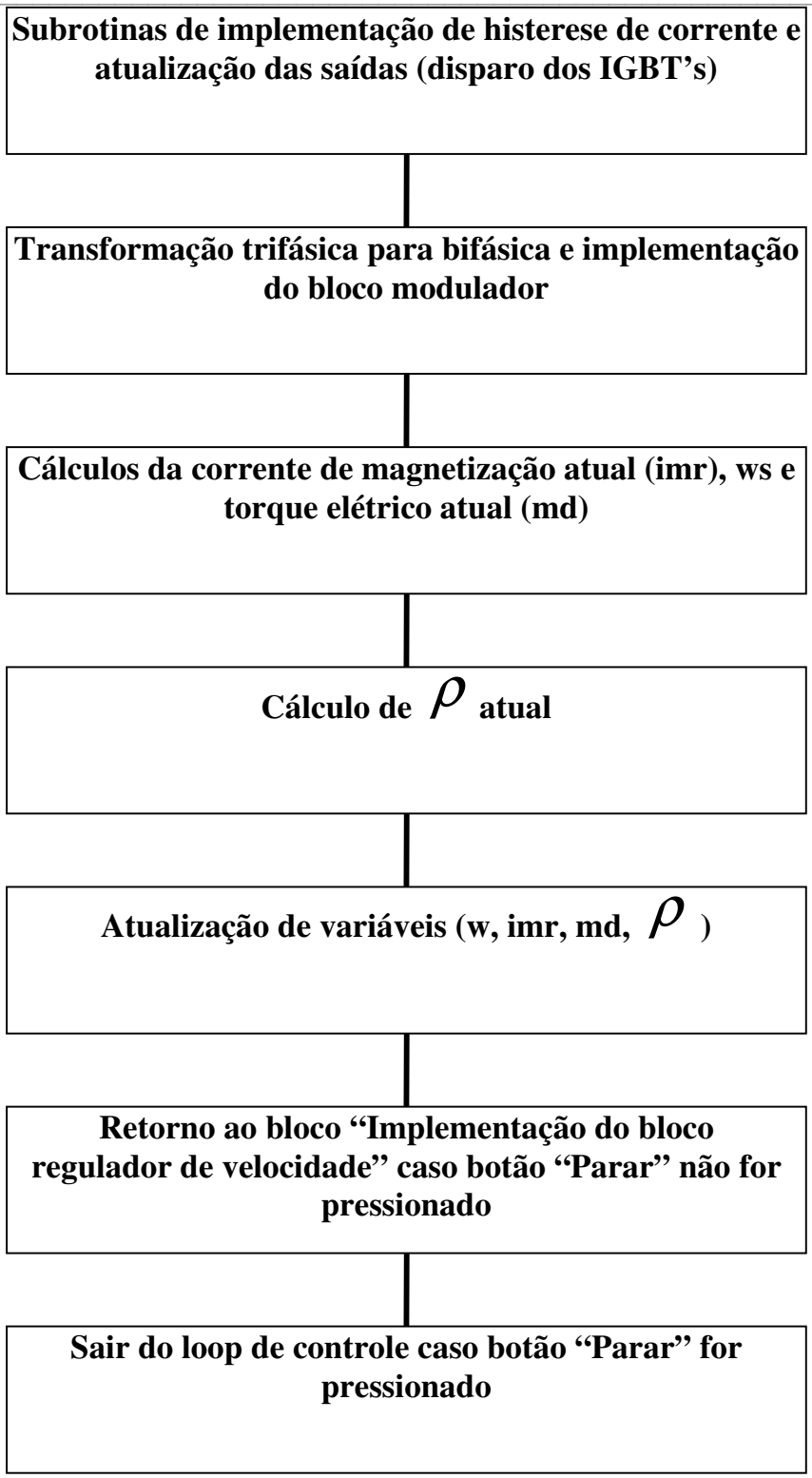


Figura 5.17 – Fluxograma do programa de controle real.

Neste trabalho optou-se por implementar uma histerese simples por software no programa aplicativo, visando a imposição de correntes da estratégia de controle adotada. Em várias implementações práticas utiliza-se hardwares específicos para realizar histereses mais complexas.

A histerese é realiza utilizando comparações entre um sinal de referência determinado e valores reais das correntes lidas. A cada comparação obtém-se uma diferença. Se esta diferença atingir valores limites atua-se no chaveamento dos IGBTs do inversor da malha de controle.

A lógica de histerese pode ser resumida da seguinte maneira:

- Se a corrente real é menor que 99% da corrente de referencia, então liga-se o IGBT superior e desliga-se o inferior da fase correspondente.

- Se a corrente real é maior que 101% da corrente de referência, então desliga-se o IGBT superior e liga-se o inferior da fase em questão.

A visualização do método descrito acima será mostrado a seguir no gráfico que foi feito no início da partida do MIT.

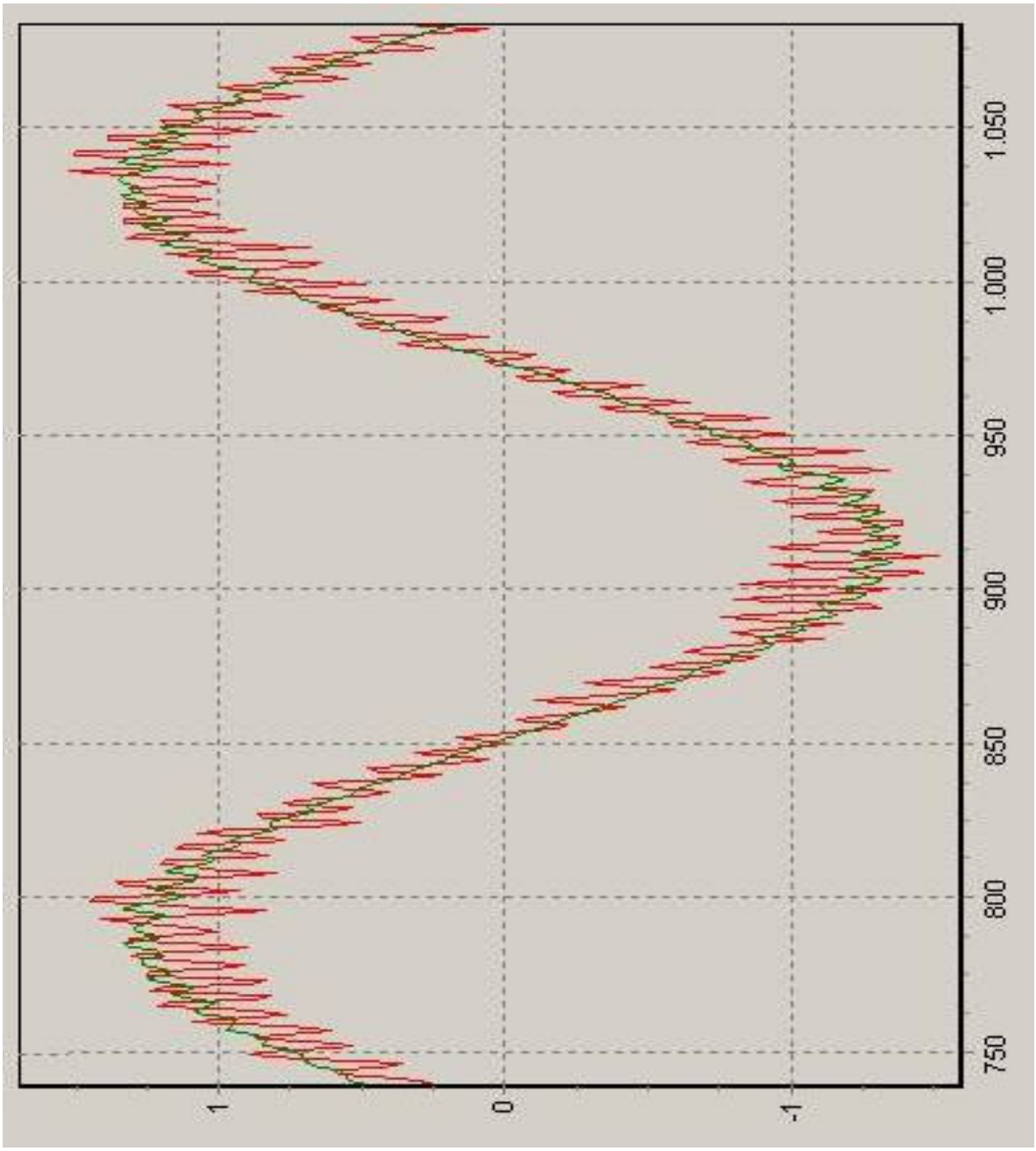


Figura 5.18 – Ilustração de histerese simples na imposição de corrente.

No próximo capítulo serão mostrados resultados reais obtidos por meio de ensaios práticos realizados na bancada de teste montada. O conceito da imposição de correntes e o controle de velocidade resultante serão ilustrados através de gráficos obtidos de dados lidos em tempo real no sistema.

6 – Resultados Obtidos

Foram realizadas inúmeras experiências no sistema proposto com o intuito de analisar aspectos diversos da malha de controle resultante. Por exemplo, partida a vazio e com carga, variações de carga, reversão de velocidade, etc. Em cada ensaio foram obtidos gráficos que serão mostrados e comentados neste capítulo.

6.1 – Testes

Inicialmente foram realizadas algumas medidas no sistema para servir como base de referência dos ensaios a serem efetuados, e objetivando verificar a resolução e controlabilidade da malha de controle montada. Nestes testes, mudava-se a velocidade de referência, e para cada mudança media-se a velocidade do motor com um tacômetro. As medidas estão indicadas nas tabelas a seguir.

Tabela 6.1 – Motor a vazio e com rotação no sentido horário.

Velocidade de referência [p.u.]	Referência de velocidade [V]	Velocidade de referência [rpm]	Velocidade real do motor [rpm]
1	4.78	1700	1668
0.75	3.58	1275	1242
0.5	2.388	850	835
0.25	1.198	425	415
0.1	0.477	170	170

Tabela 6.2 – Motor com carga nominal e rotação no sentido horário.

Velocidade de referência [p.u.]	Referência de velocidade [V]	Velocidade de referência [rpm]	Velocidade real do motor [rpm]
1	4.78	1700	1665
0.75	3.58	1275	1239
0.5	2.388	850	836
0.25	1.198	425	418
0.1	0.477	170	170

Tabela 6.3 – Motor a vazio e com rotação no sentido anti-horário.

Velocidade de referência [p.u.]	Referência de velocidade [V]	Velocidade de referência [rpm]	Velocidade real do motor [rpm]
1	- 4.98	1700	1642
0.75	- 3.73	1275	1230
0.5	- 2.492	850	817
0.25	- 1.241	425	408
0.1	- 0.498	170	164

Tabela 6.4 – Motor com carga nominal e rotação no sentido anti-horário.

Velocidade de referência [p.u.]	Referência de velocidade [V]	Velocidade de referência [rpm]	Velocidade real do motor [rpm]
1	- 4.98	1700	1646
0.75	- 3.73	1275	1228
0.5	- 2.492	850	814
0.25	- 1.241	425	407
0.1	- 0.498	170	162

Notar que na primeira coluna de cada tabela estão as velocidades de referência em p.u. a serem impostas e na segunda coluna têm-se tensões do

potenciômetro de referência de rotação. As terceiras e quartas colunas contêm, respectivamente, as velocidades que o motor deveria atingir para cada tensão de referência e as velocidades reais do motor. Analisando as tabelas é fácil notar que o sistema apresenta boa resolução e controlabitidade. O erro médio de velocidade é de aproximadamente 1,7%.

6.2 – Ensaios

As figuras que serão mostradas contêm gráficos de ensaios práticos realizados na bancada experimental. A finalidade é verificar características de resposta dinâmica da malha de controle implementada. A cada ensaio realizado o software de controle produz quatro conjuntos de gráficos, cujo horizonte de tempo é de vinte e cinco segundos no registro de algumas informações da malha de controle. As informações dos gráficos estão em p.u. e são basicamente as seguintes:

1 – A rotação real do motor e a referência de velocidade de entrada da malha.
2 – As correntes trifásicas reais das fases do motor.
3 – A estimativa das correntes responsáveis pelo fluxo (isd) e pelo torque (isq).
4 – A estimativa da corrente de magnetização (imr).

6.2.1 – Partida a vazio no sentido horário

A Figura 6.1 mostra resultados de um ensaio real realizado no sistema com referência de rotação nominal (sentido horário) e motor partindo a vazio. A referência de velocidade (wref) da malha de controle é aproximadamente um degrau unitário (poderia ser também uma rampa que definiria uma determinada taxa de crescimento de velocidade do motor). O motor atinge rotação nominal após um período de tempo do início do controle. Para melhor visualização, alguns gráficos (nas Figuras 6.2 e 6.3) serão mostrados separadamente devidamente ampliados com recursos de “zoom” nas telas geradas.

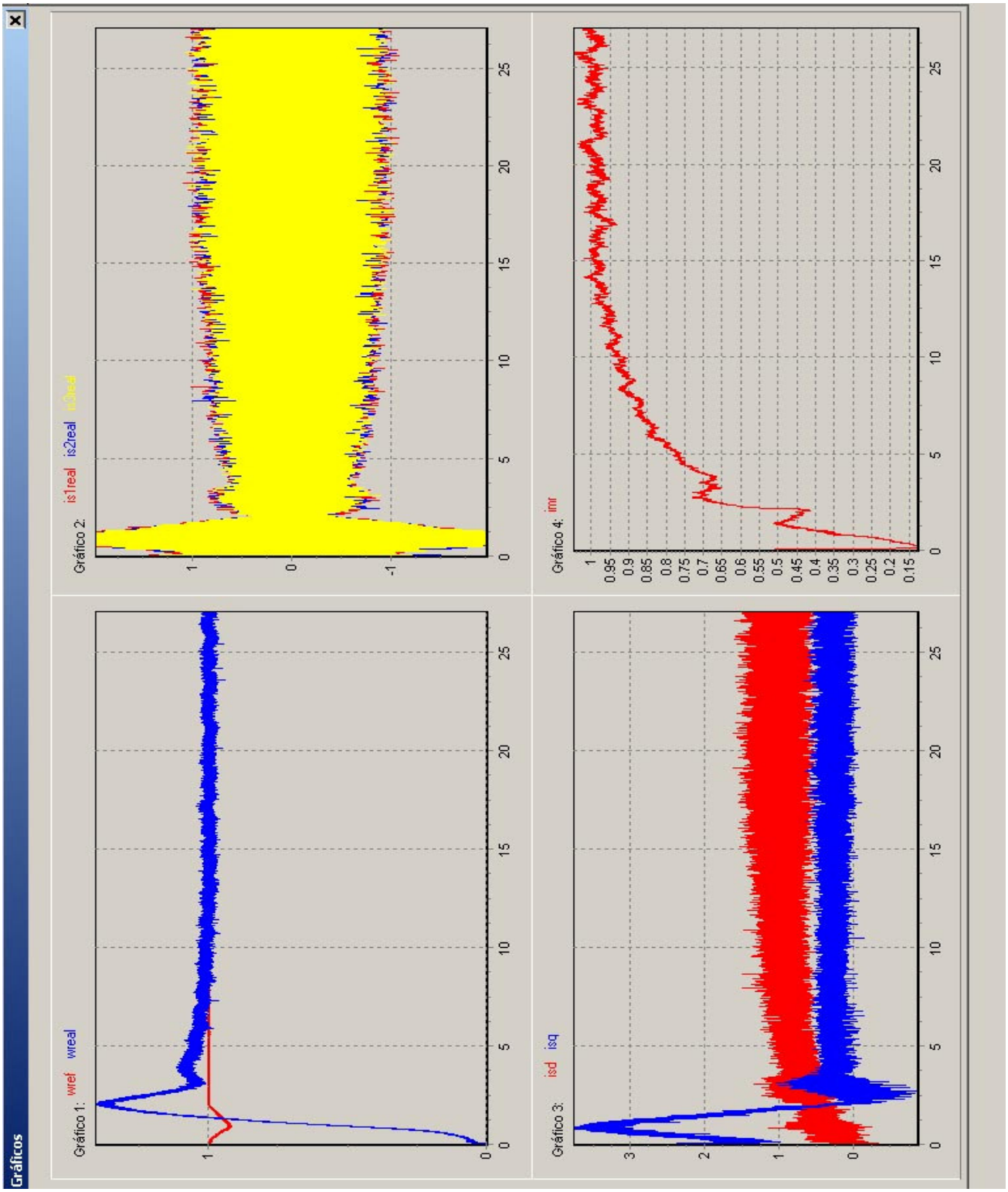


Figura 6.1 – Motor a vazio e referência de rotação nominal (sentido horário).

No gráfico têm-se a velocidade real (azul) e a de referencia (vermelho) lidas através da placa de aquisição de dados. Pode-se observar um pequeno "afundamento" na tensão de referência. A explicação para este efeito é atribuída a indução de ruídos na partida do motor, é que na montagem os cabos de potência que alimentam o motor estão próximos dos cabos do potenciômetro de referência, o que pode ter provocado interferência durante a partida, já que neste momento a corrente no motor é maior. Como o efeito foi transitório não houve maiores problemas, mas em montagens prática estes cabos deveriam estar perpendiculares e/ou deveria-se usar cabo blindado no sinal de referência, ou ainda a inclusão de um filtro no canal de entrada do sinal de referência objetivando a minimização de interferências da parte de potência do sistema.

Para melhor visualização, alguns gráficos (nas Figuras 6.2 e 6.3) serão mostrados separadamente devidamente ampliados com recursos de "zoom" nas telas geradas.

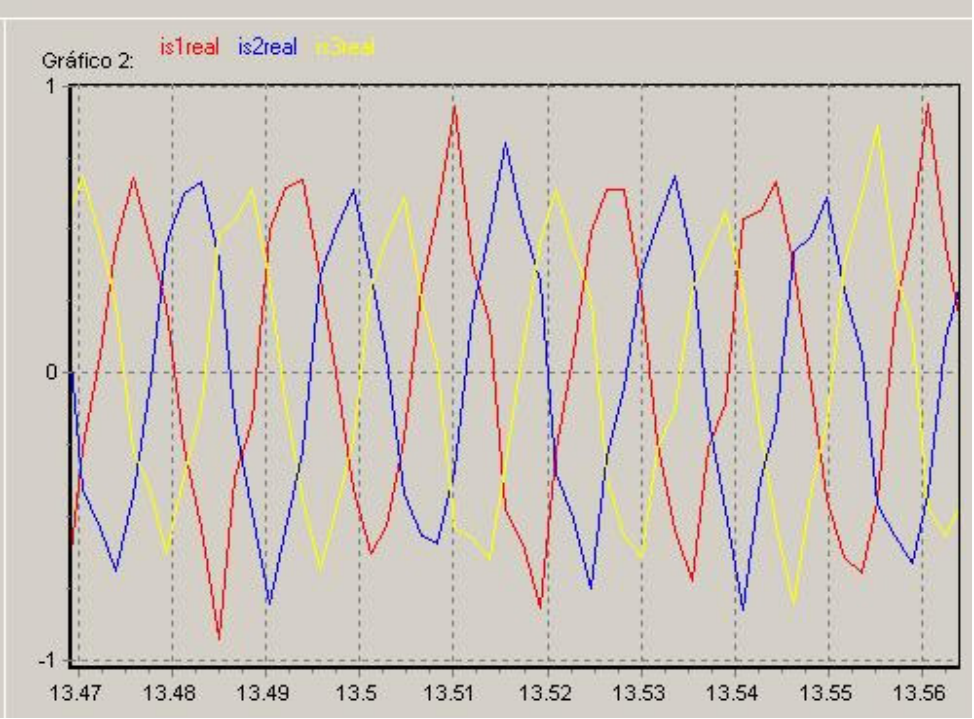


Figura 6.2 – Correntes de fase reais do ensaio.

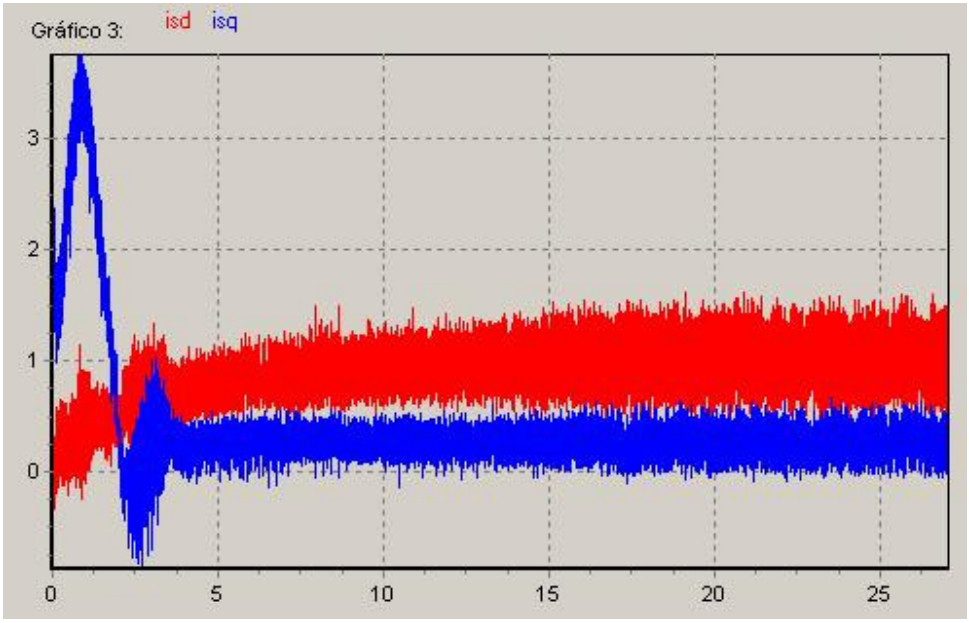


Figura 6.3 – Estimação das correntes isd e isq.

O tempo de partida do motor está um pouco alto, assim como a regulação da corrente isd não está adequada. Ajustes posteriores nestes parâmetros irão melhorar sensivelmente as características de regulação do sistema.

6.2.2 – Partida com carga no sentido horário

A Figura 6.4 contém dados de um ensaio real realizado na malha de controle com referência de rotação nominal (sentido horário) e motor partindo sob carga. Observando os gráficos nota-se que a máxima sobre-elevação da rotação real do motor é menor que a do ensaio anterior (com motor a vazio), devido a presença de carga agora no eixo da máquina. O tempo de acomodação da velocidade se manteve.

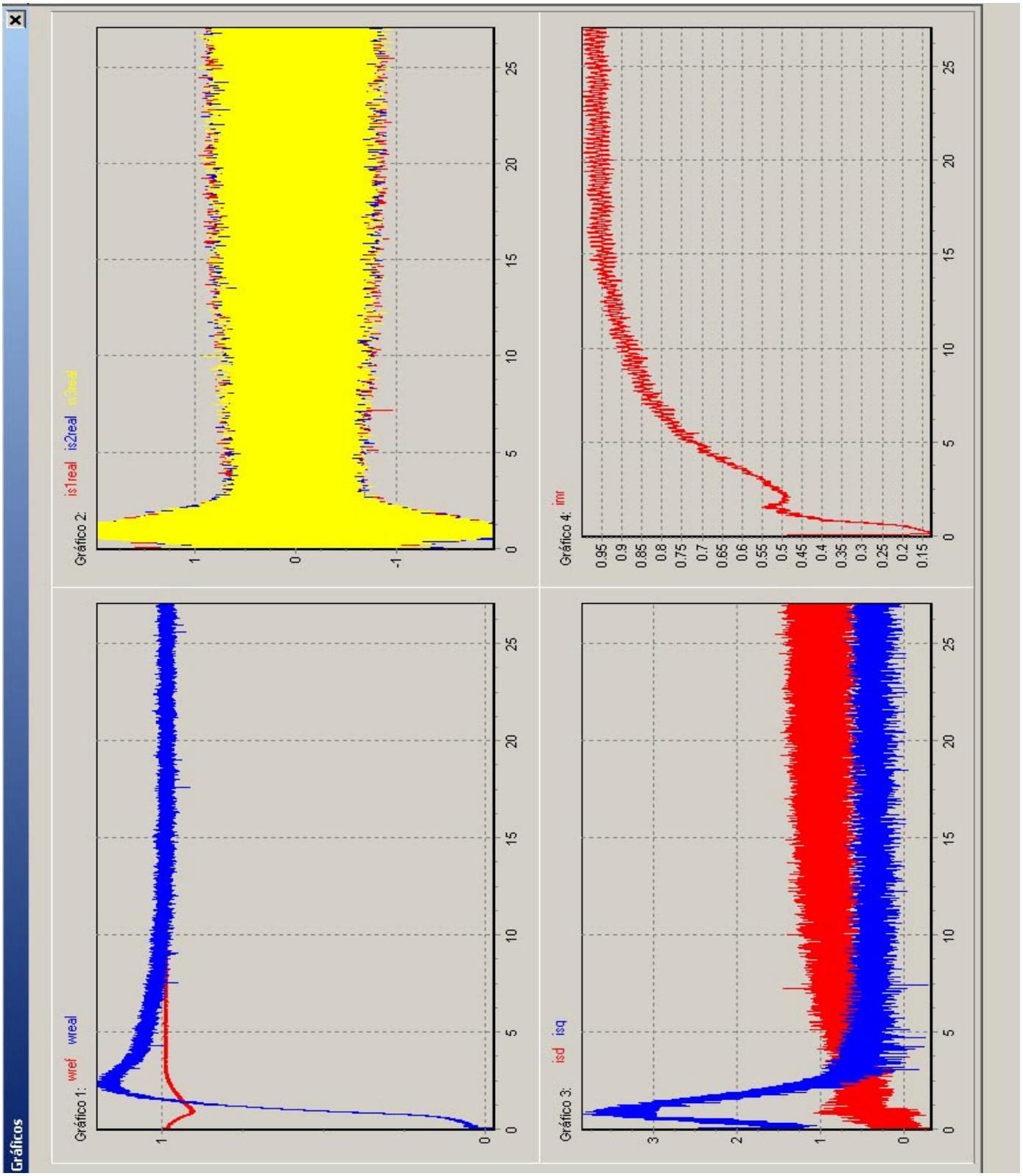


Figura 6.4 – Motor a plena carga e referência de rotação nominal (sentido horário).

6.2.3 – Variação de velocidade a vazio no sentido horário

A Figura 6.5 ilustra informações reais de um ensaio prático realizado na malha de controle. A referência de rotação (sentido horário) inicialmente está em torno de 0,25 [p.u.]. Após sete segundos da partida ela é aumentada para 1 [p.u.], algum tempo depois reduzida para 0,5 [p.u.], e em torno de dezesseis segundos é reduzida para 0,375 [p.u.]. Observa-se que a resposta dinâmica do sistema tende a se manter para diferentes valores de referências, mostrando a controlabilidade da malha de controle implementada. Neste ensaio o motor está sem carga no seu eixo.

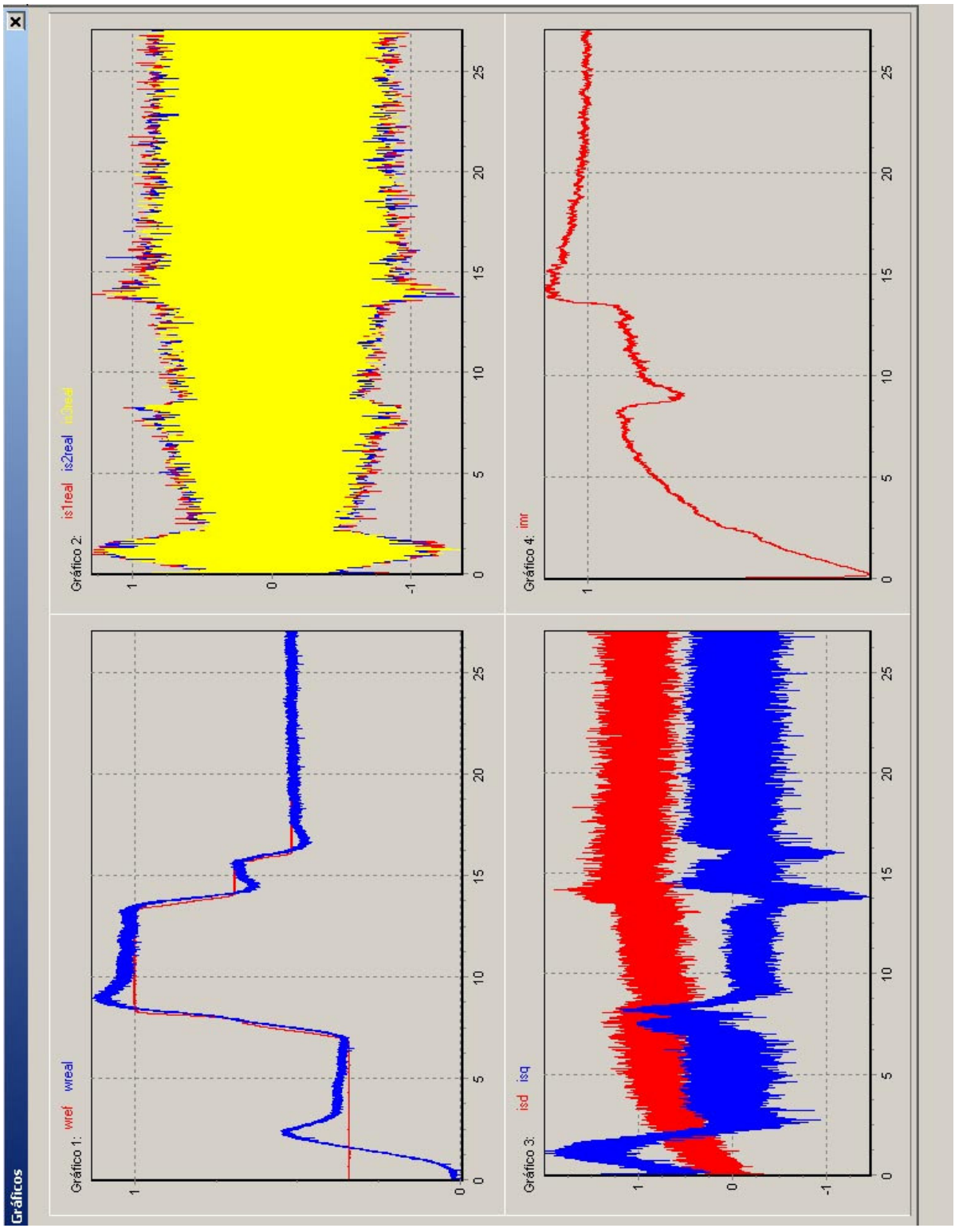


Figura 6.5 – Resposta da malha de controle a variações de referência de rotação.

6.2.4 – Variação de carga no sentido horário

A Figura 6.6 traz dados reais de um ensaio prático realizado no sistema. A referência de rotação (sentido horário) é mantida em 1 [p.u.] após a partida do motor sob carga nominal. Em aproximadamente sete segundos após a partida a carga é retirada, e em torno de dezesseis segundos a carga é aplicada novamente. Observando a resposta temporal da informação da velocidade real do motor, nota-se que houve a regulação restaurando a rotação do sistema em valores bem próximos da referência de velocidade desejada.

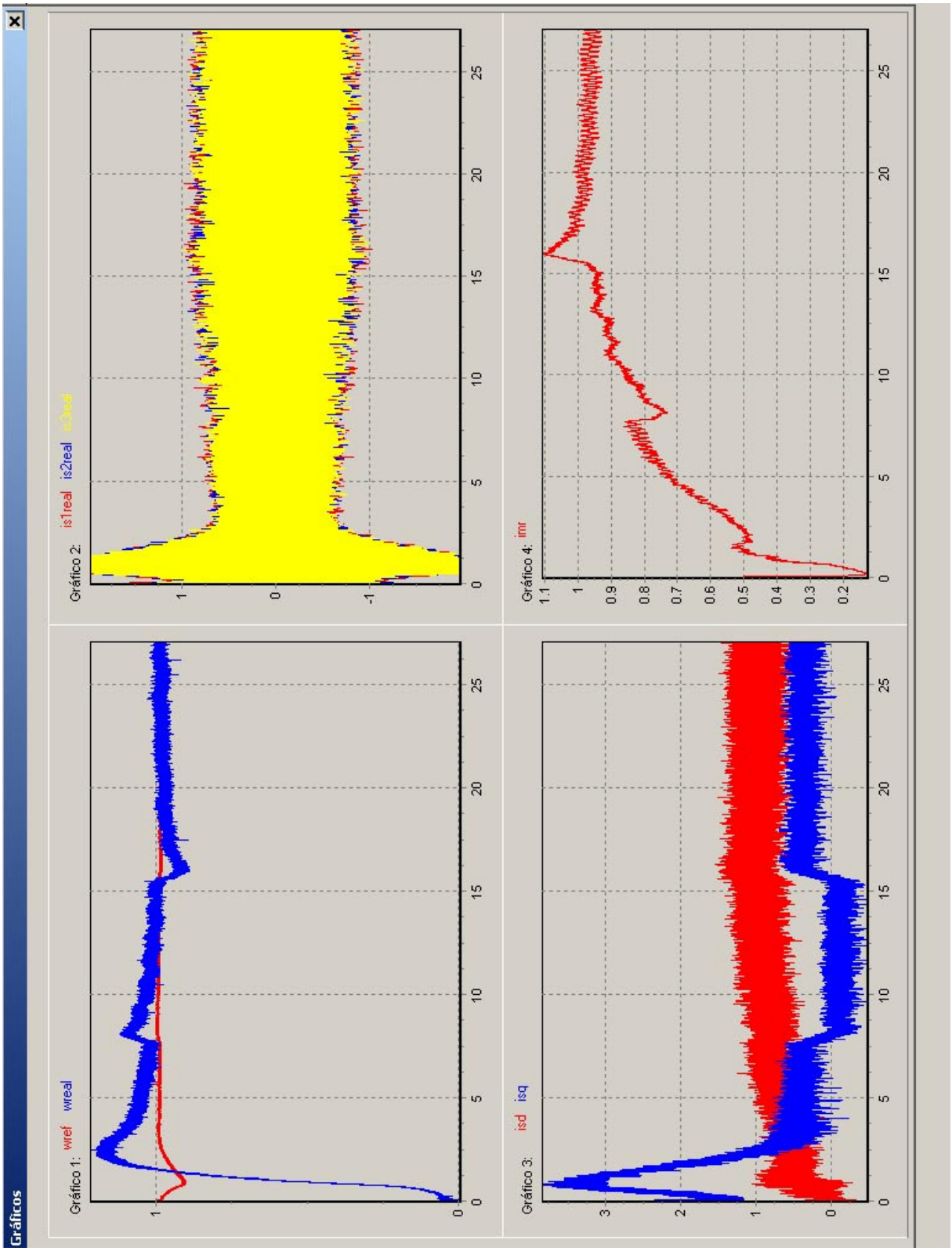


Figura 6.6 – Resposta do sistema a variações de carga no motor.

6.2.5 – Reversão no sentido de rotação a vazio

A Figura 6.7 mostra dados de um ensaio real realizado no sistema. O motor parte a vazio com referência de rotação no sentido horário em valor nominal. Aproximadamente após onze segundos a referência de rotação é invertida (sentido anti-horário) para -1 [p.u.], e em vinte e três segundos volta para 1 [p.u.] novamente (sentido horário de rotação). Nota-se que as respostas dinâmicas estão iguais aos padrões de comportamento já verificados nos ensaios anteriores. A Figura 6.8 ilustra o comportamento das correntes reais das fases do motor e a Figura 6.9 mostra um zoom no gráfico obtido. Pode-se notar que nos instantes de tempo em que ocorreram as inversões nos sentidos de rotação do motor, as correntes de fase são invertidas (assim como a corrente isq indicada na Figura 6.10). A Figura 6.11 mostra a estimação da corrente imr do motor.

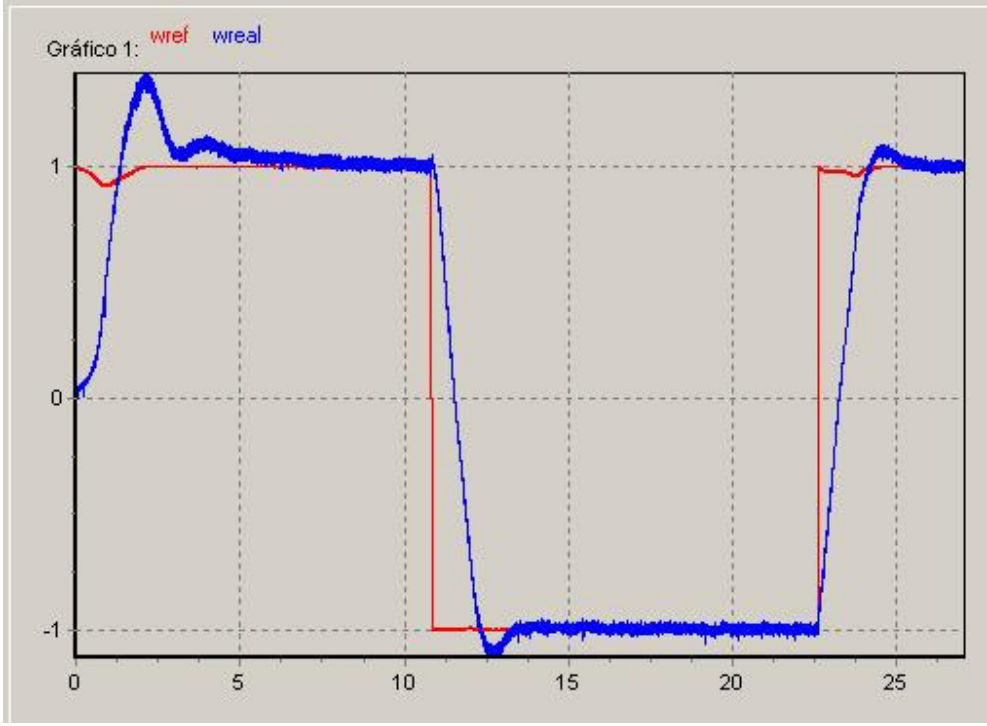


Figura 6.7 – Ensaio de reversão no sentido de rotação do motor.

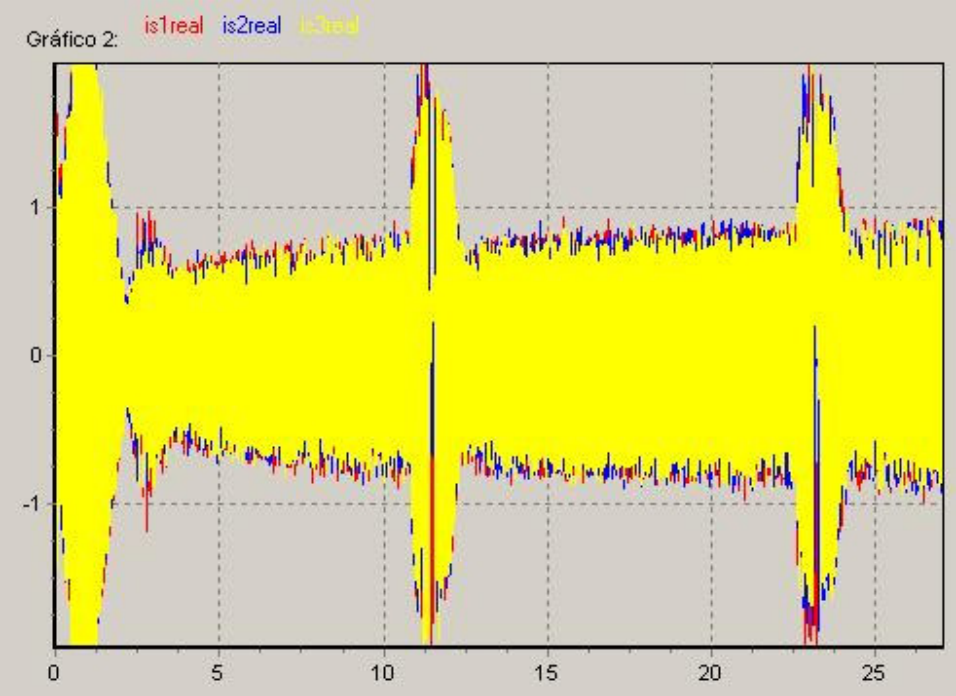


Figura 6.8 – Registro das correntes de fase do motor.

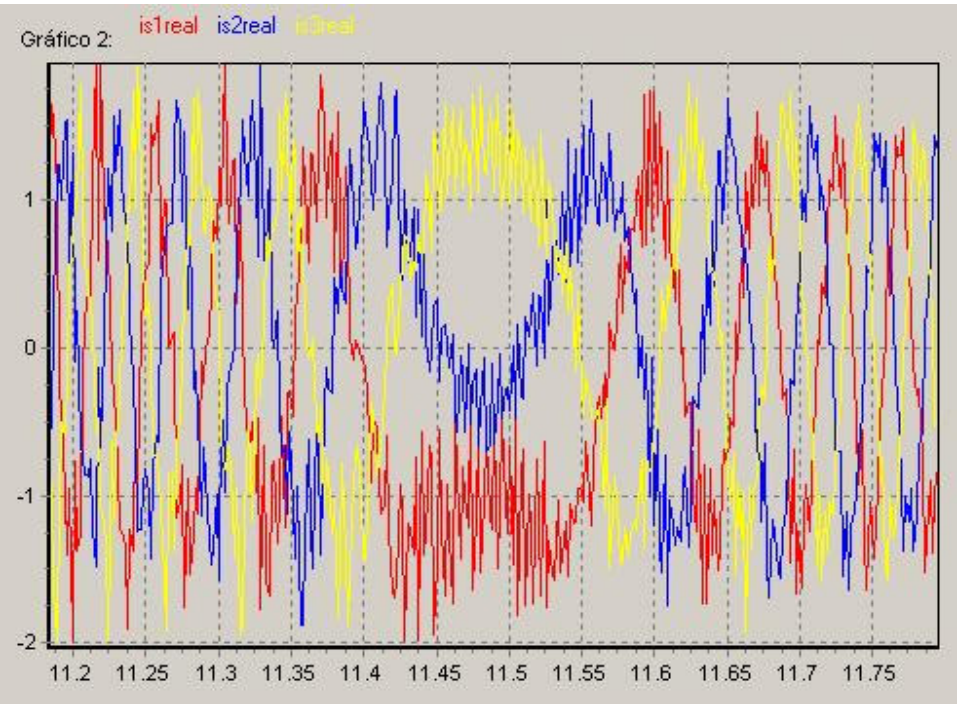


Figura 6.9 – Zoom no gráfico das correntes da Figura 6.8.

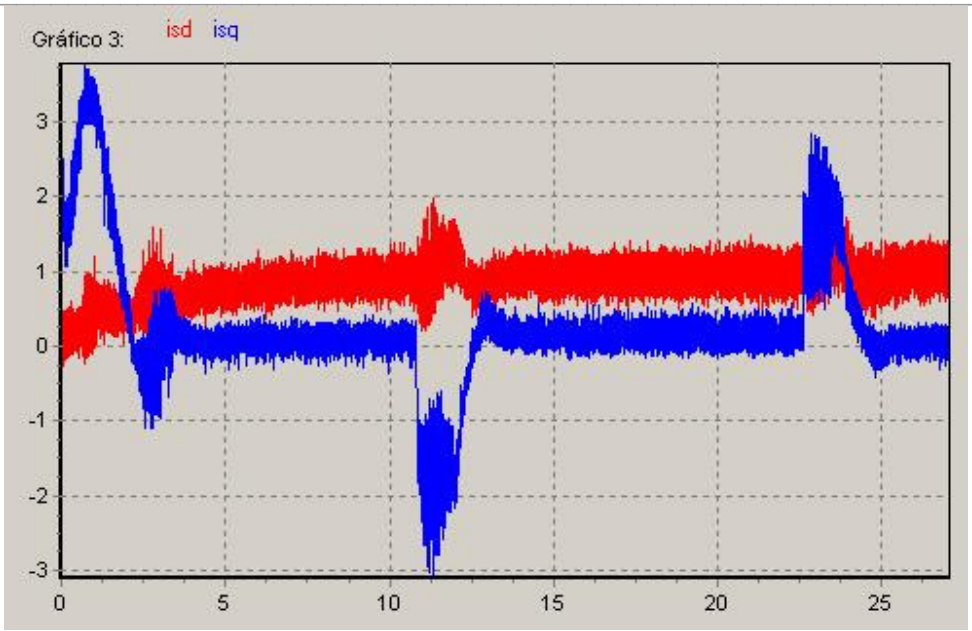


Figura 6.10 – Estimação das correntes isd e isq.

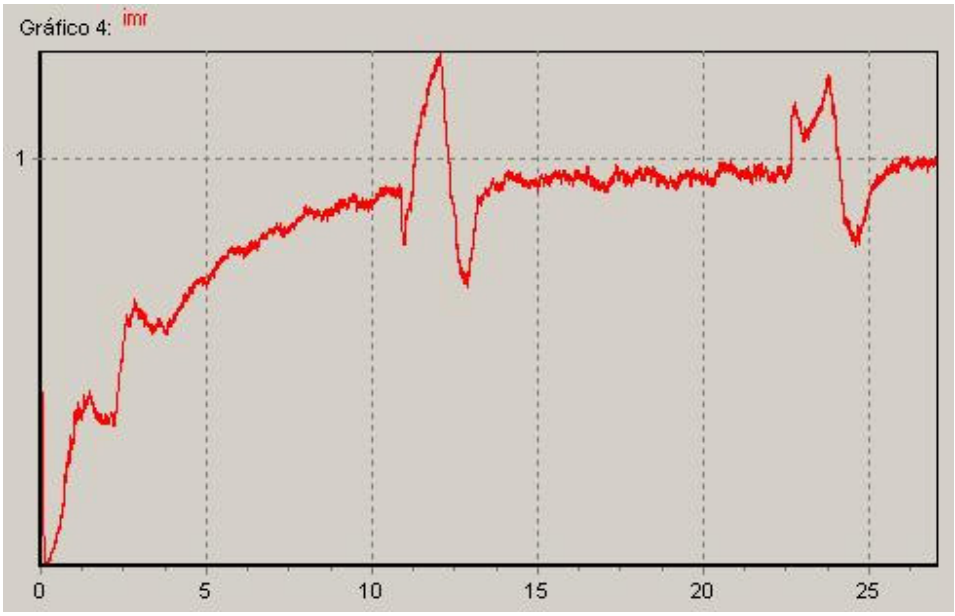


Figura 6.11 – Estimação da corrente imr.

Como já citado anteriormente, a malha de controle em questão está ativa todo tempo, e não foi utilizada nenhuma técnica para frenagem do sistema antes da reversão, a própria malha gerou a contra corrente para a inversão de rotação. Como a potência é baixa, este ensaio não causou problemas, mas caso o motor fosse de grande porte seria necessário limitar a corrente durante a reversão de velocidade. A implementação de rampas de aceleração e desaceleração no sinal de referência de velocidade também seria adequado.

6.2.6 – Reversão no sentido de rotação com carga

A Figura 6.12 traz o resultado de um ensaio no sistema de controle com o motor partindo com carga plena e realizando-se variações de sentido de rotação no mesmo para valores nominais de referência de velocidade. Observa-se que o comportamento do sistema tende a ser parecido com os ensaios anteriores, confirmando a regulação da malha de controle implementada.

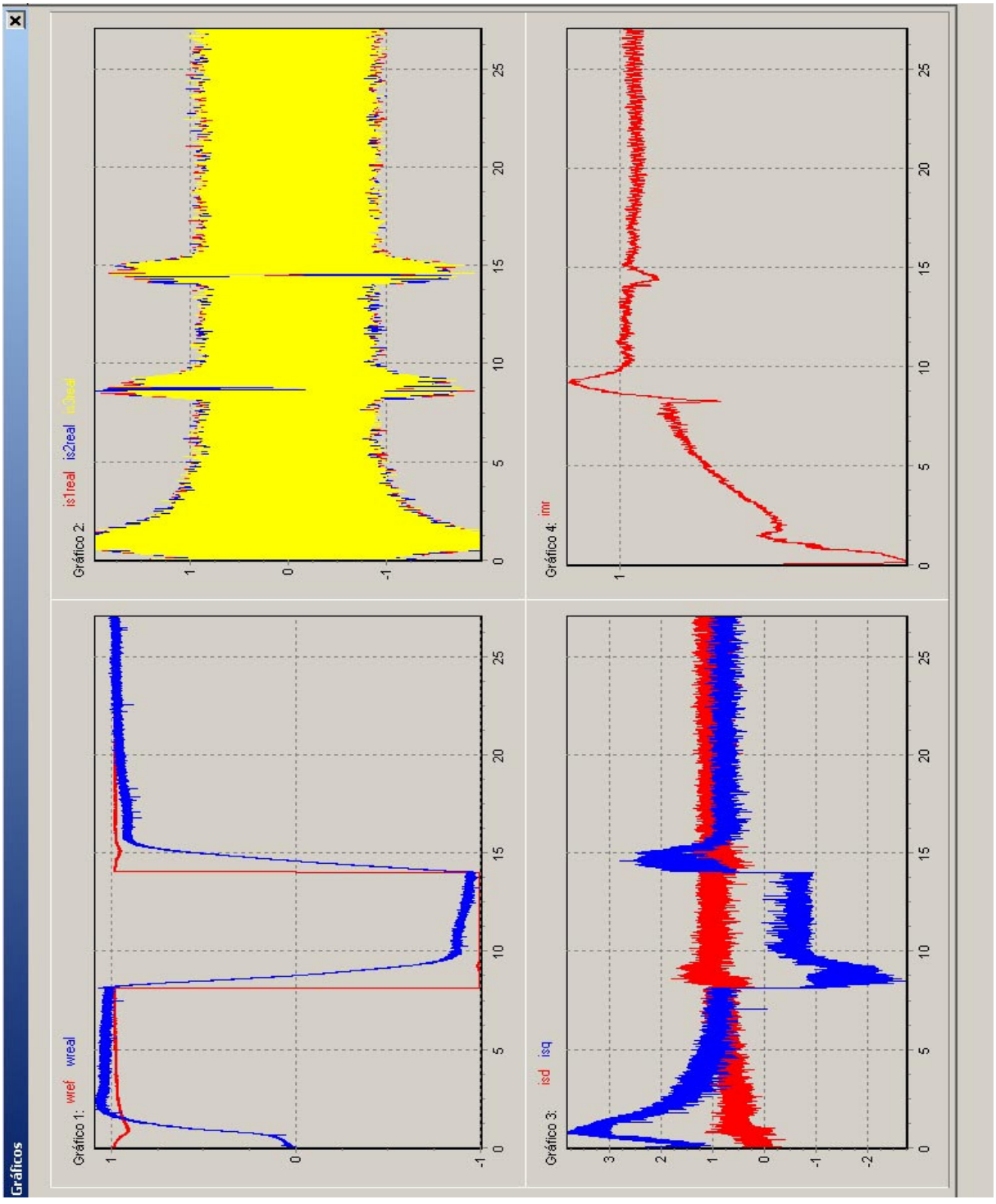


Figura 6.12 – Reversão no sentido de rotação com carga nominal.

6.2.7 – Variação de velocidade com carga no sentido horário

A Figura 6.13 mostra o resultado de um ensaio no sistema com o motor partindo com carga plena e realizando-se variações na referência de rotação (no sentido horário) na malha de controle.

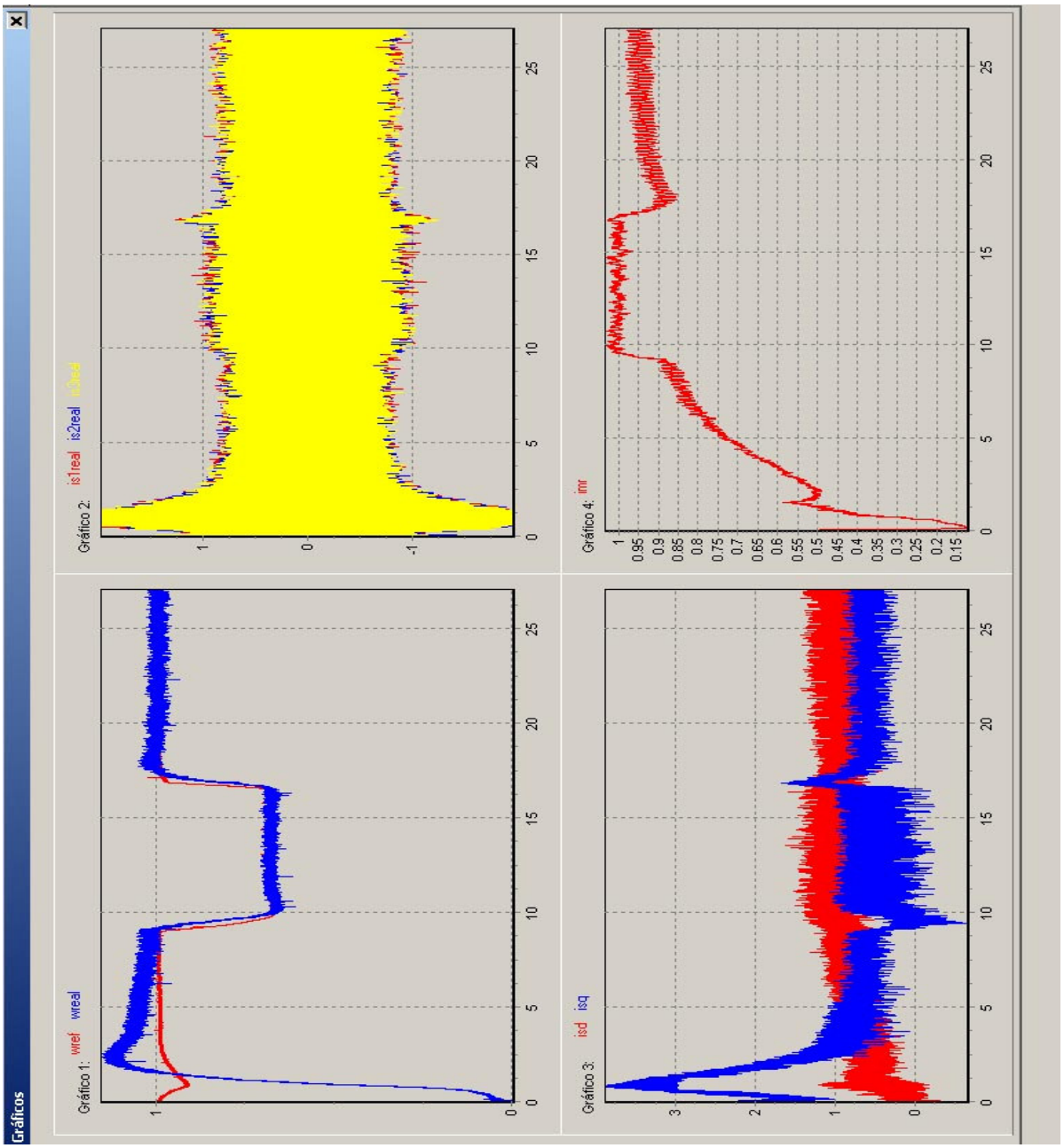


Figura 6.13 – Motor com carga e variação de velocidade (sentido horário).

6.2.8 – Ensaios no sentido anti-horário com variação de carga e velocidade

As Figuras 6.14, 6.15 e 6.16 mostram resultados de ensaios no sistema com o motor partindo a vazio, variações na referência de velocidade na malha de controle e variações na carga, agora no sentido anti-horário de rotação. Verifica-se que o sistema de controle funcionou adequadamente em diversas condições operacionais.

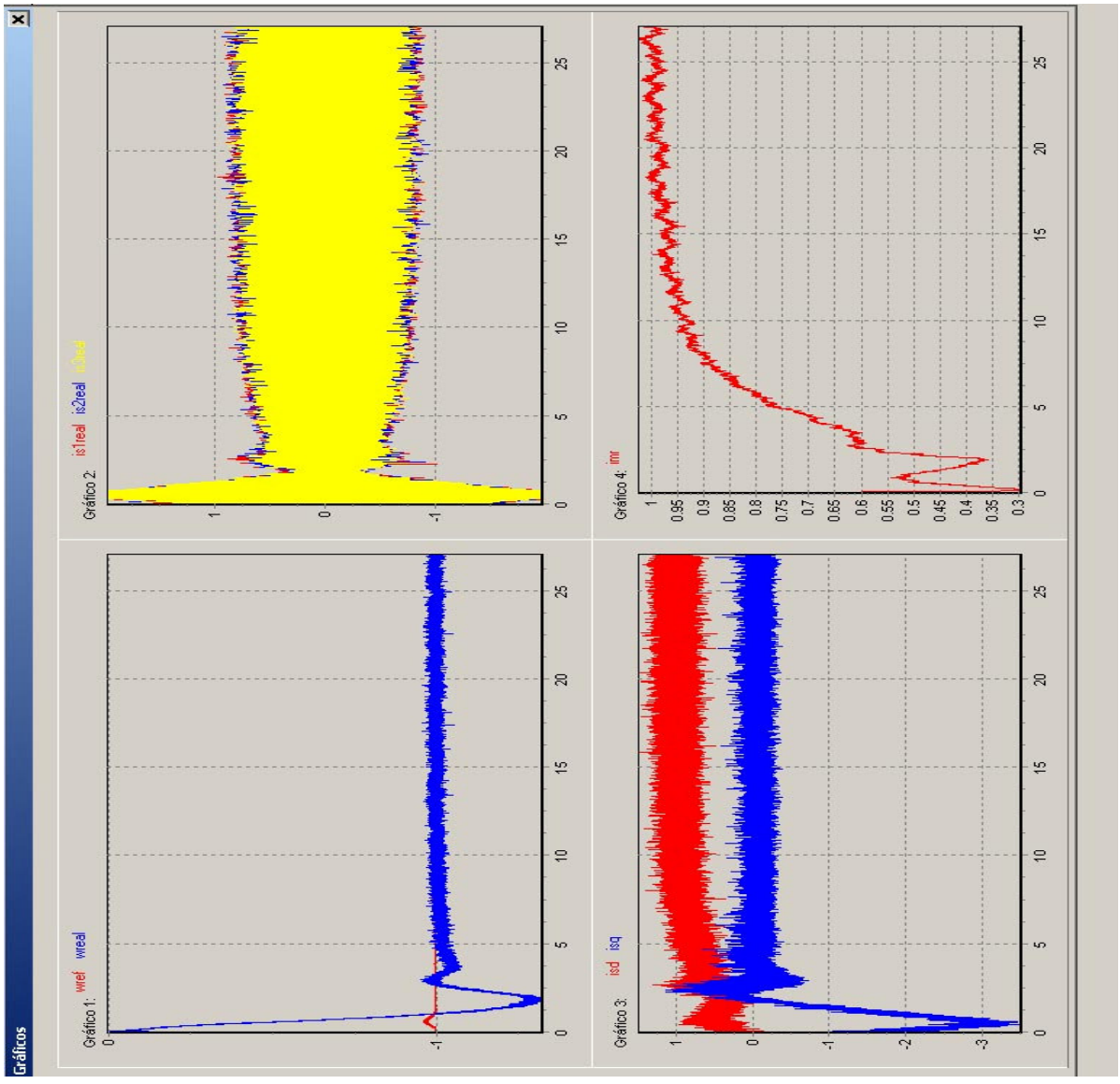


Figura 6.14 - Partida a vazio e sentido de rotação anti-horário.

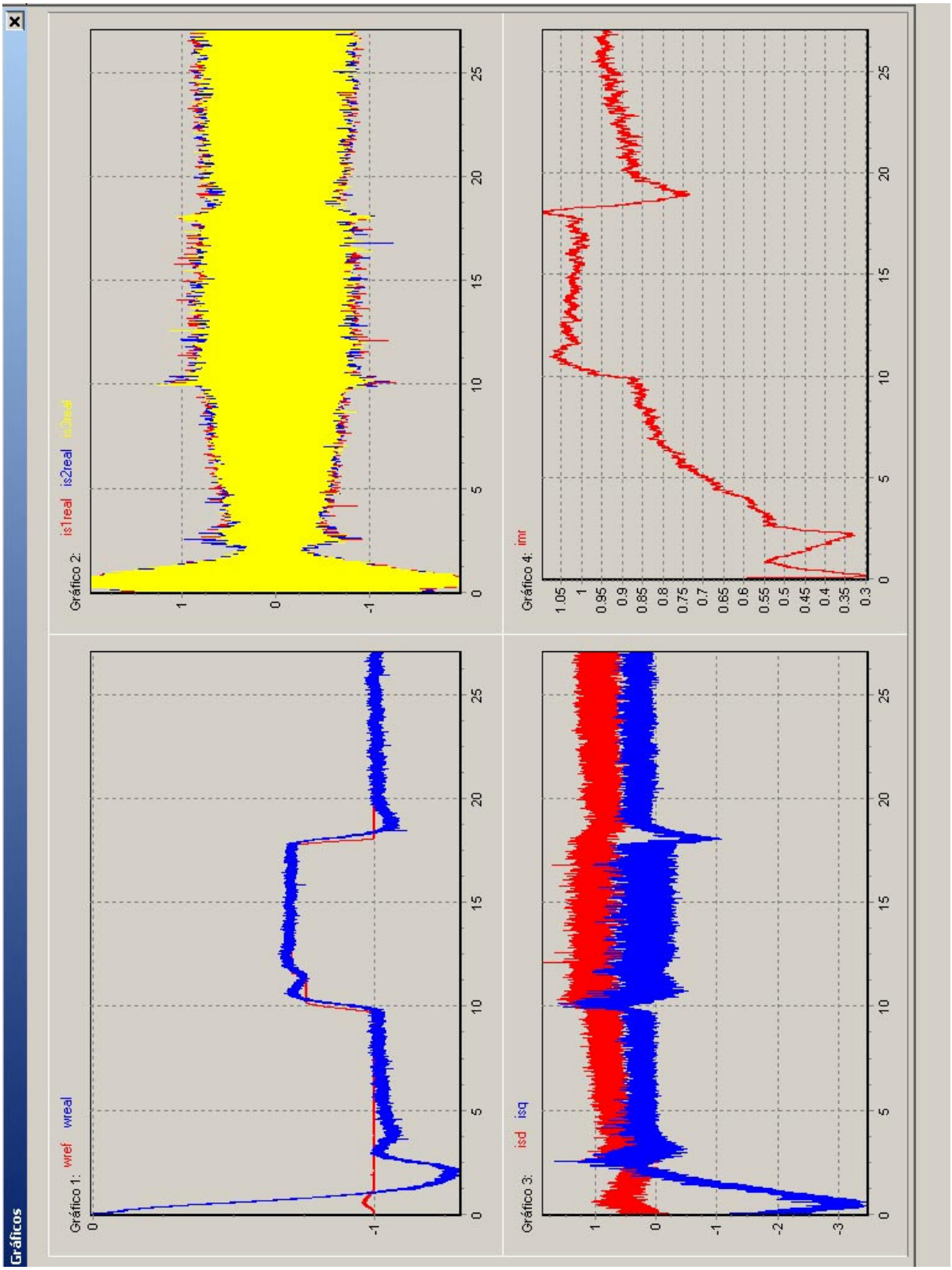


Figura 6.15 - Motor a vazio e variação de referência de rotação (anti-horário).

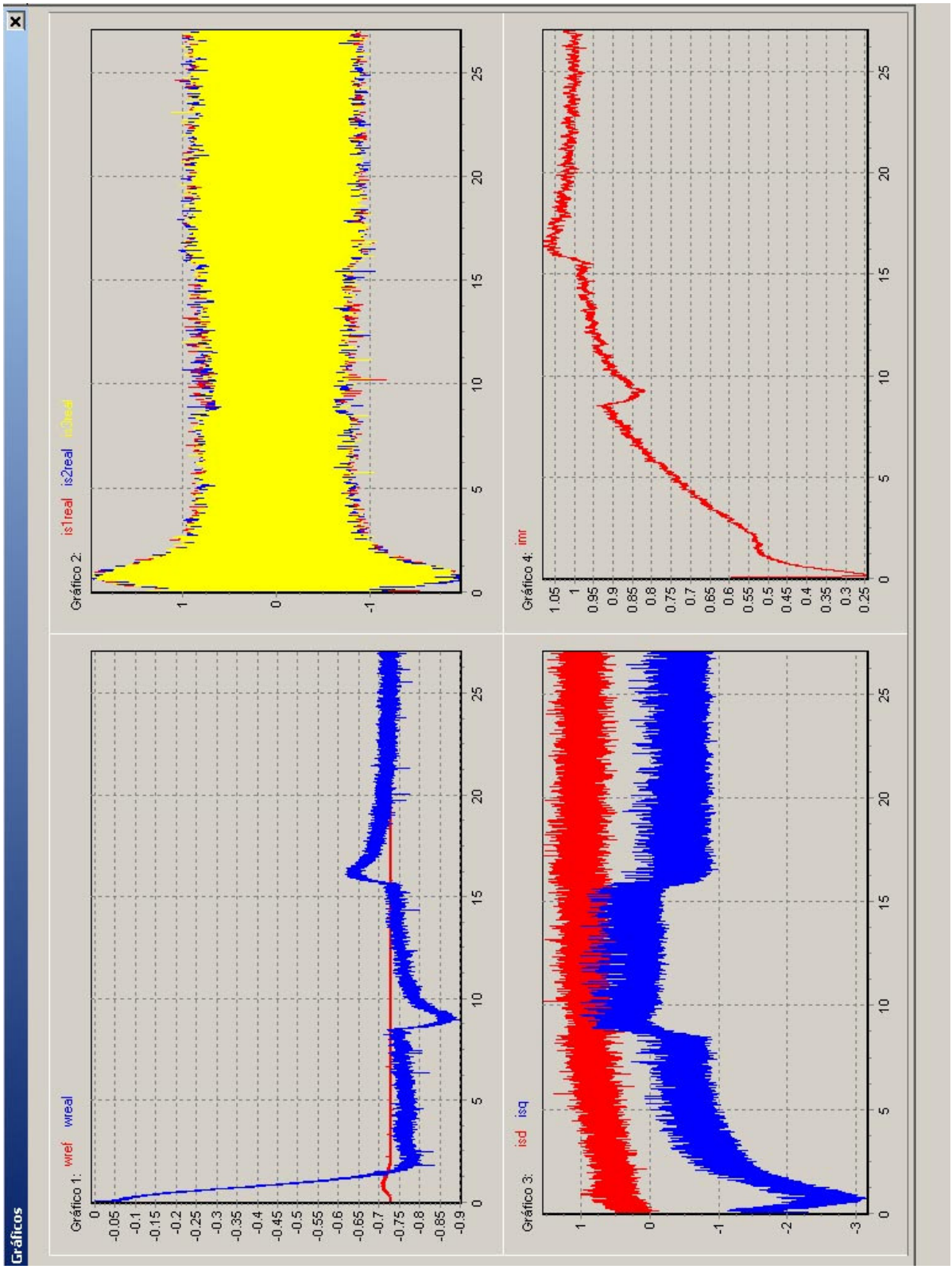


Figura 6.16 – Motor sob variação de carga e rotação no sentido anti-horário.

6.3 – Ensaios registrados por meio de um osciloscópio digital

Com a finalidade de verificar os resultados obtidos pela interface gráfica do programa de controle desenvolvido e obter outras informações adicionais, foi utilizado um osciloscópio digital para registrar algumas informações da malha de controle implementada. As pontas de prova do osciloscópio foram ligadas nos mesmos transdutores de corrente (sondas Hall) e de velocidade (taco-gerador) da bancada de ensaio.

O sinal registrado no canal denotado por {1} é relativo à informação de corrente de fase da máquina, e o sinal do canal denominado {2} está relacionado com a rotação do motor. As escalas são de 2 [V/div.] e de 2,5 [s/div.], respectivamente.

6.3.1 – Partida a vazio no sentido horário

A Figura 6.17 contém o registro do osciloscópio para um ensaio com referência de entrada correspondente a aproximadamente 1 [p.u.] e com o motor sem carga.

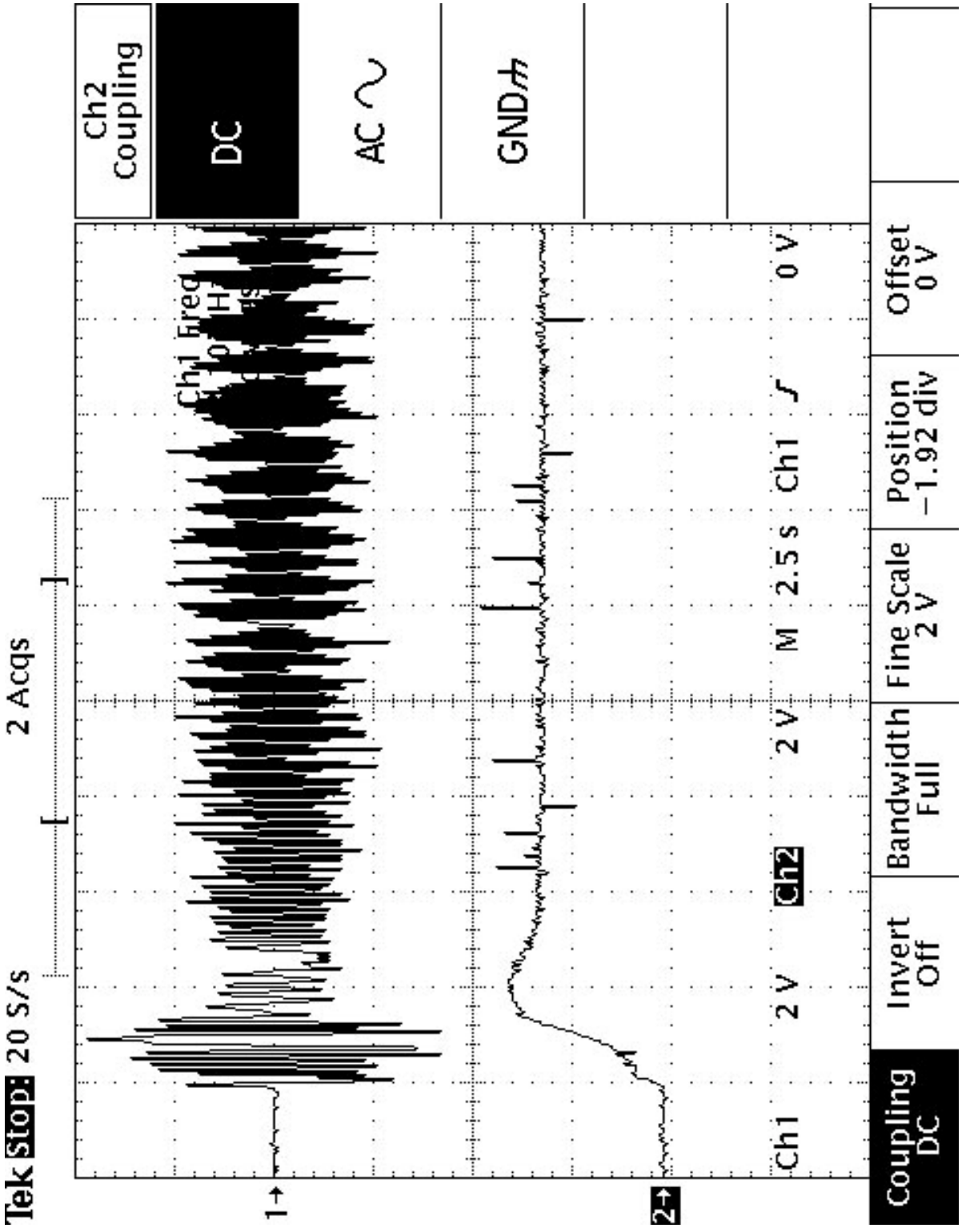


Figura 6.17 – Motor sem carga e referência de rotação no sentido horário.

6.3.2 – Variação de velocidade no sentido horário

A Figura 6.18 mostra um ensaio com o motor sem carga e com rotação (sentido horário) inicialmente em torno de 1 [p.u.]. Após 2,5 [s] a referência da rotação é reduzida a zero, e depois de 7,5 [s] o *set-point* retorna a 1 [p.u.].

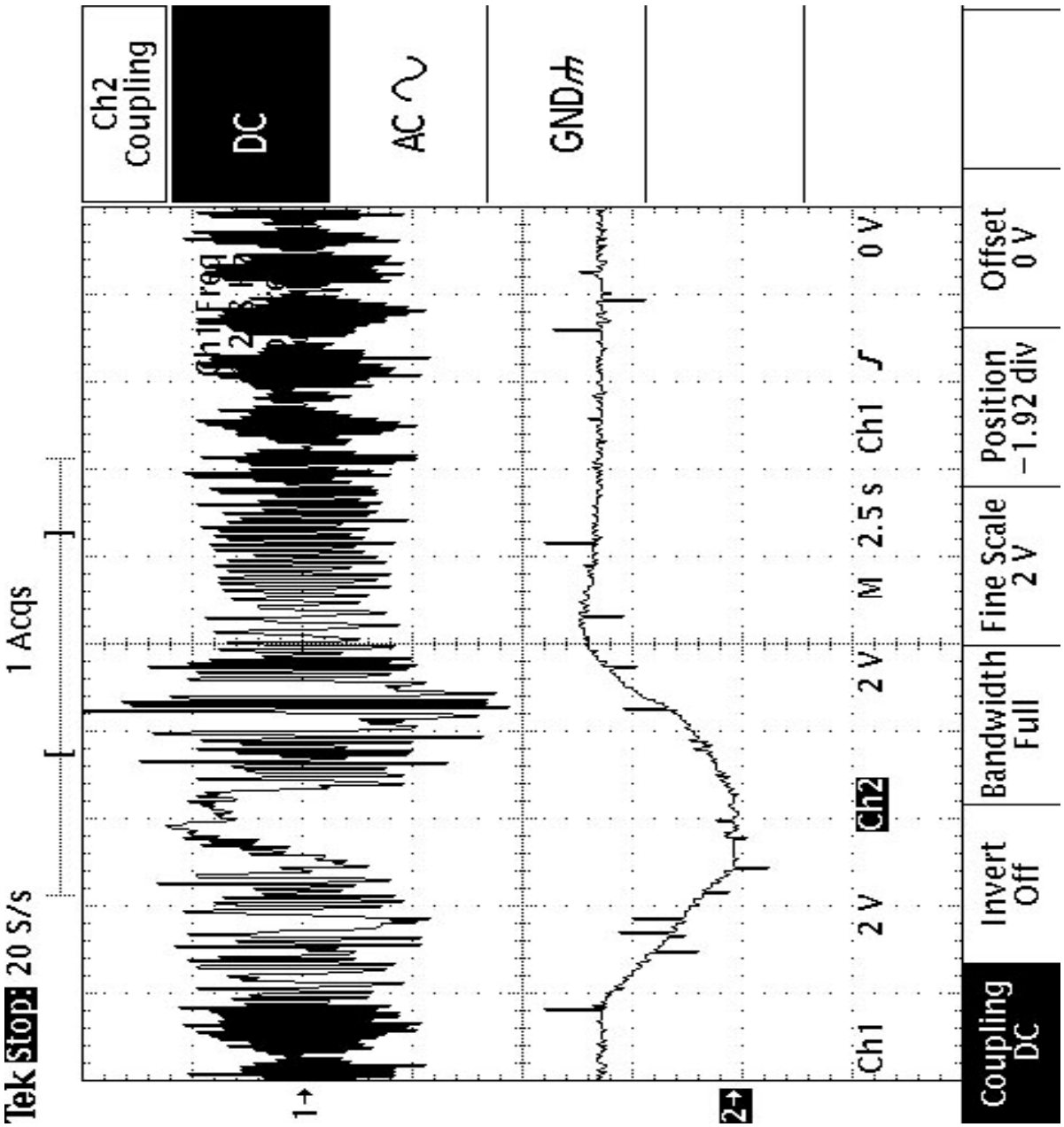


Figura 6.18 – Motor sem carga e variação de referência de rotação.

6.3.3 – Partida com carga no sentido anti-horário

A Figura 6.19 ilustra o registro de um ensaio com o motor partindo com carga e com referência de rotação no sentido anti-horário.

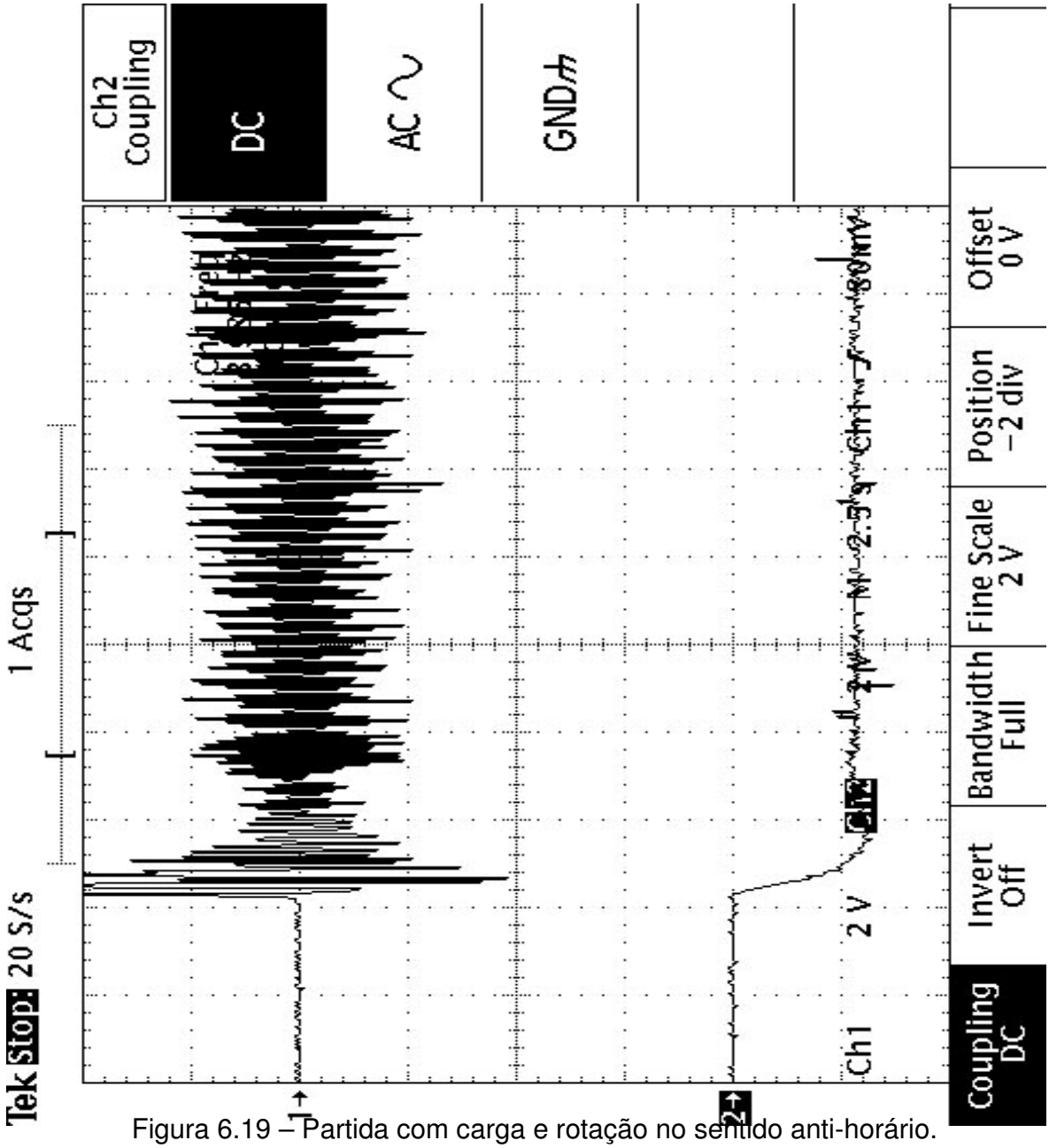


Figura 6.19 – Partida com carga e rotação no sentido anti-horário.

6.3.4 – Variação de carga no sentido horário

A Figura 6.20 mostra o registro de um ensaio com o motor acionado inicialmente sem carga e com velocidade em regime permanente. Em torno de 3,75 [s] é aplicada carga na máquina e depois em aproximadamente 18,75 [s] a carga é retirada. Observa-se nos gráficos que quando a carga é aplicada a corrente aumenta e a velocidade tende a reduzir, mas logo a malha de controle corrige a velocidade e a corrente é reduzida. Quando a carga é retirada a corrente diminui e a velocidade tende a aumentar, mas a atuação da malha de controle corrige novamente as condições do sistema.

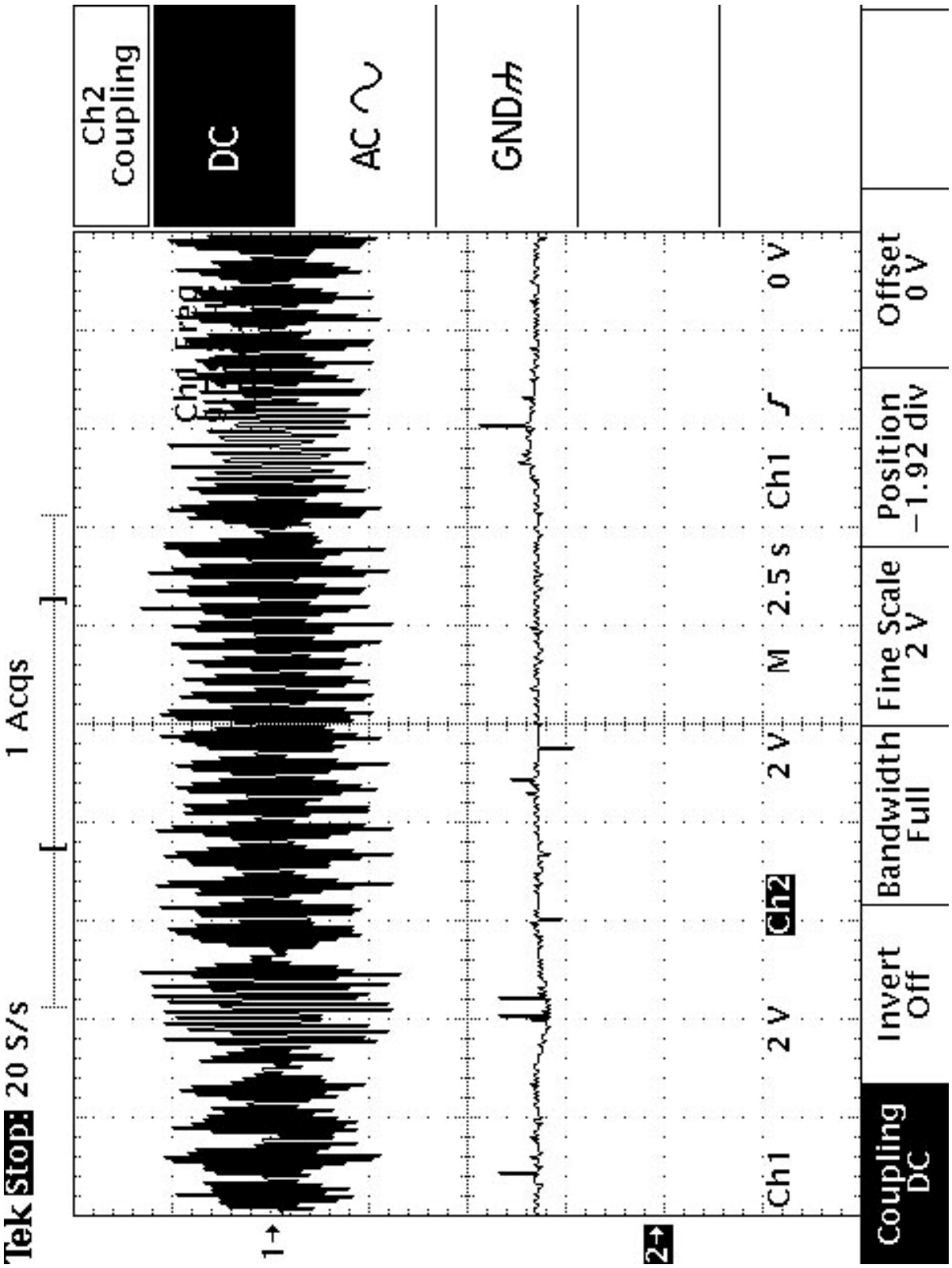


Figura 6.20 – Ensaio com variação de carga.

6.3.5 – Reversão no sentido de rotação

Na Figura 6.21 tem-se o registro de um ensaio com inversão de rotação do sistema, com modificação da referência de rotação do sentido horário para o anti-horário.

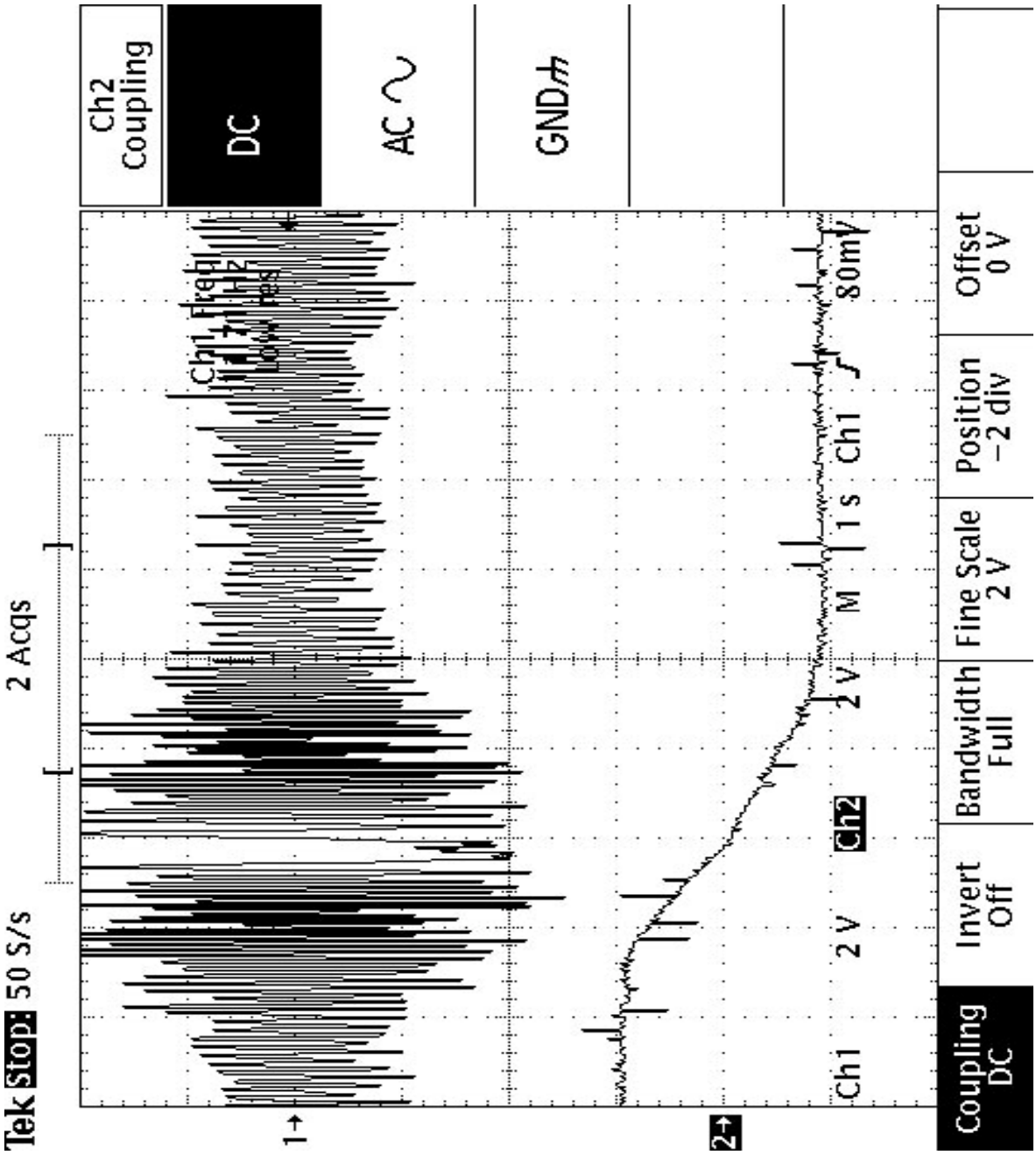


Figura 6.21 – Ensaio com inversão do sentido de rotação no sistema.

6.3.6 – Tensões fase-fase

Finalmente as Figuras 6.22 e 6.23 trazem o registro de tensões do sistema obtidas nas fases do inversor. Os gráficos são relativos a duas condições operacionais distintas, como pode se verificar nas larguras de pulso das formas de onda correspondentes. Para estes ensaios só foi utilizado o canal 1, com as escalas de 2[V/div] e 5[ms/div].

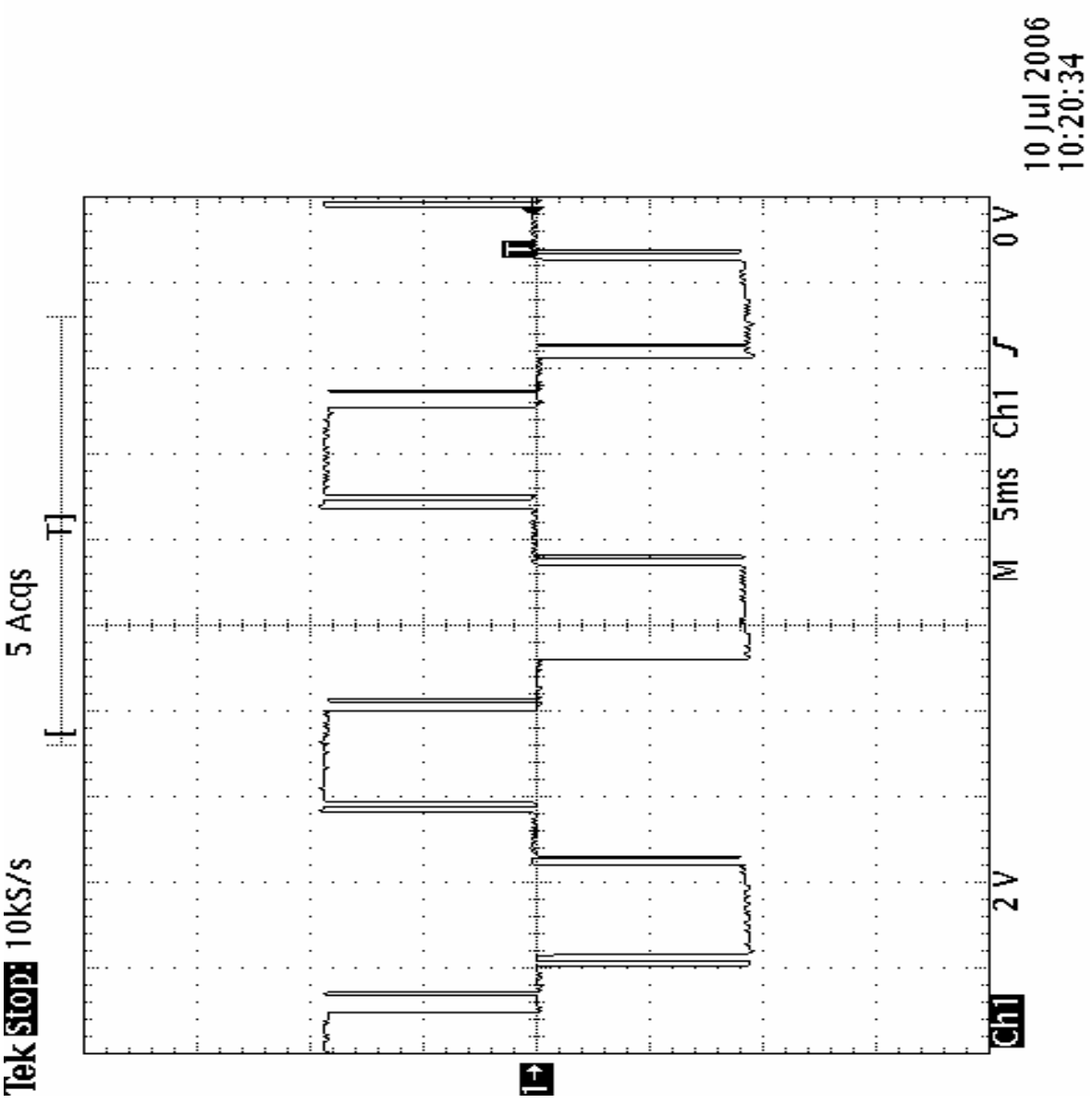


Figura 6.22 – Tensão fase-fase para uma determinada condição operacional.

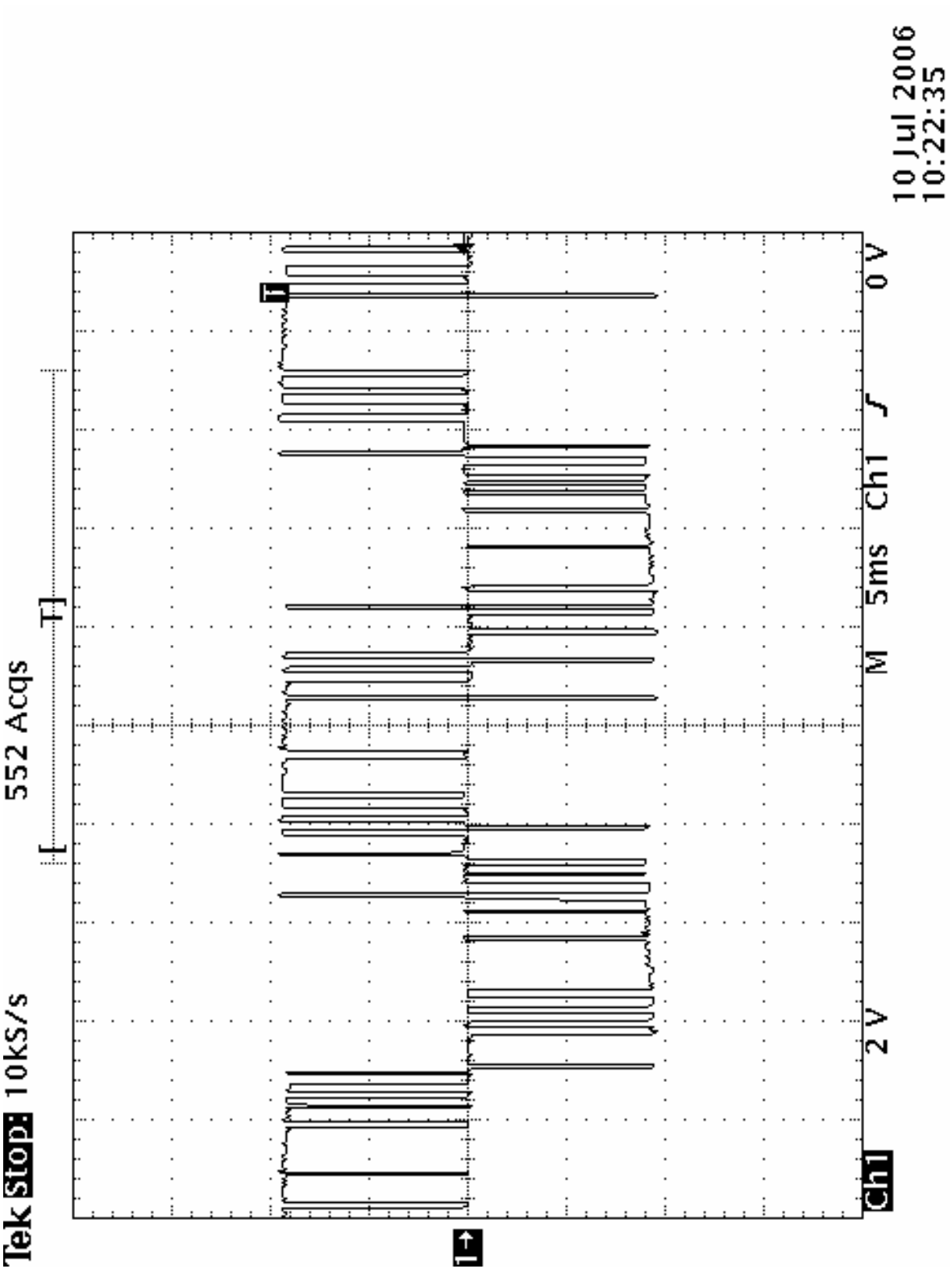


Figura 6.23 – Tensão fase-fase para outra condição operacional.

6.4 - Comparações entre os resultados das simulações e do controle real

Existem diferenças nos resultados obtidos pela simulação e o controle real, como pode ser visto no capítulo 4 e 6. Elas se devem a vários fatores, entre eles:

- Primeiramente como era natural de se esperar, na simulação os parâmetros são teóricos. A velocidade é obtida por meio de estimativas realizadas através de cálculo numérico. Além disso, o sistema simulado não está sujeita a ruídos de um sistema real. As correntes estimadas nas simulações são as próprias correntes de referência calculadas pelo programa. Deste modo é como se o inversor atuasse de tal maneira instantânea e perfeitamente igual a referência, o que na prática é impossível de ser alcançado.

- Não foi utilizado nenhum procedimento de identificação no sistema montado para obter os parâmetros do processo, assim os ganhos dos controladores PI do sistema real foram escolhidos empiricamente e diferem do sistema simulado. Na simulação as constantes de tempo são menores e os ganhos dos reguladores são maiores, quando comparados com o controle real, ocasionando uma demora maior na regulação das correntes Isd e Imr nos resultados reais. Também não foi utilizado nenhum procedimento de inicialização do sistema de controle real, como esperar a estabilização do mesmo e depois fazer com que a malha de controle assuma regulação operacional do sistema.

- Na simulação o tempo de varredura das integrações numéricas do modelo do sistema não possui restrições e pode ser tão pequeno quanto se queira, pois a resposta de velocidade e correntes são ideais. Já no sistema real o tempo é um fator muito importante, qualquer atraso ocasionado pela placa de aquisição de dados, pelo sistema operacional do microcomputador ou pelos cálculos do algoritmo de controle vetorial provoca uma demora na atualização dos IGBT´s e conseqüente atraso e erros na resposta. Outro efeito que pode provocar atrasos nas correntes impostas operando em freqüências mais altas (para rotações mais

elevadas) é a implementação da histerese por software. Uma solução seria realizar a histerese por hardware, assim a atualização não seria apenas a cada loop da malha de controle, mas sim a todo tempo, já que seu processamento seria separado.

Apesar das limitações apresentadas no sistema implementado, isto não invalida a proposta como uma bancada de experimentação e demonstração didática de um sistema de controle vetorial por imposição de correntes. O ajuste adequado dos ganhos dos controladores PI pode melhorar bem as respostas dinâmicas obtidas.

7 – Conclusões finais e trabalhos futuros

O objetivo deste trabalho foi apresentar a implementação prática de um sistema de controle de motor de indução trifásico por meio de imposição de correntes, utilizando-se da teoria de controle vetorial indireto.

O trabalho resultou na montagem de uma bancada experimental composta por um hardware de interface e um software que em conjunto foram responsáveis pela implementação prática do sistema de controle proposto.

Analisando os dados obtidos mediante vários ensaios experimentais conclui-se os resultados são promissores. O sistema apresentou a capacidade de controlar o motor utilizado frente a várias alterações de carga em diferentes valores operacionais de rotação. Tanto em rotações nominais quanto em baixas rotações a controlabilidade do sistema foi satisfatória. Assim, pode-se afirmar que o objetivo deste trabalho foi alcançado, inclusive com superação de algumas expectativas iniciais como se a utilização de um microcomputador mais simples seria possível atingir a velocidade (freqüência) nominal do motor utilizado. Assim como a implemetação da histerese simples de corrente por software.

Como proposta para continuação deste trabalho pode-se citar os seguintes itens:

- Utilização de um microcontrolador ou DSP comercial para implementação da malha de controle no lugar no microcomputador visando a obtenção de um sistema mais rápido e dedicado.

- Realizar ensaios com diversos modelos de motores de indução comerciais e com cargas diversas.

- Os parâmetros do motor utilizados neste trabalho foram obtidos de dados de placa, mas como trabalho futuro propõem-se implementar algum algoritmo de identificação ou estimação de parâmetros como sugerido em Garces (1980) e Rowan (1991).

- Realizar um sistema que atue também no controle do fluxo, além do controle das correntes impostas.

- Usar o sistema desenvolvido para acionamento de máquinas síncronas no lugar de máquinas assíncronas.

- Implementação da histerese de corrente por hardware ao invés de software.

Referências Bibliográficas

Blaschke, F. The Principle of Field Orientation as Applied to the New Transvector Closed-Loop Control System for Rotating-Field Machines. Adjustable Speed AC Drive Systems, IEEE Press, Bimal K. Bose, 1972.

Hasse, K. – Zur Dynamik drehzahlgeregelter Antriebe mit stromrichtergespeisten AsynchronkurzschluBläufermaschien. Diss. TH. Darmstadt. 1969.

CANUDAS, C. and RAMIREZ, J. "Optimal Torque Control for Current-Fed Induction Motors". IEEE Transactions on Automatic Control, April 1997.

El-Sayed, I. F. “A Powerful and Efficient Hysteresis PWM Controlled Inverter”. EPE Journal, vol 4, nº 4, December 1994.

GARCÉS, L. J. "Parameter Adaption for the Speed-Controlled Static AC Drive with a Squirrel Cage Induction Motor". IEEE Trans. Ind. Appl., vol. 16, no. 2, pp. 173-178, 1980.

Leonhard, W; Gabriel, R; Nordby, C. J. “Field-Oriented Control of a Standard AC Motor Using Microprocessors”. IEE Transactions on Industry Applications, vol. 16, Nº2, March/April 1980.

Leonhard, W. Microcomputer Control of High Dynamic Performance ac-Drives: A Survey. Automática, vol. 22, n1, 1986.

Leonhard, W. Control of Electrical Drives, Springer-Verlag, 2nd Edition, 1996.

Santos, J. R. – Relatório Final: Controle Vetorial de MIT Assumindo Imposição de Correntes,Trabalho de Diploma, Universidade Federal de Itajubá, 2004.

Phillips, C. L, Nagle, H. T. Digital Control System. Prentice Hall, 1995.

ROWAN, T. M. et al. “A Simple "On-Line Adaption for Indirect Field Orientation of an Induction Machine". IEEE Trans. Ind. Appl., vol. 27, no. 4, pp. 720-727, 1991.

Referências Consultadas

Ajay Tripathi, Paresh C. Sen. “Comparative Analysis of Fixed and Sinusoidal Band Hysteresis Current Controllers for Voltage Source Inverters”. IEEE Trans. Ind. Elet., vol. 39, nº1, february 1992.

G. Heinemann. “Comparison of Several Control Schemes for AC Induction Motors”. In Proc EPE’89, 1989, pp. 843-848.

G. O. Garcia, J. C. Mendes Luís, R. M. Stephan, E. H Watanabe. “An Efficient Controller for na Adjustable Speed Induction Motor Drive”. IEEE Trans. Ind. Elet., vol. 41, nº5, october 1994.

G. O. Garcia, R. M. Stephan, E. H Watanabe. “Comparing the Indirect Field-Oriented Control with a Scalar Method”. IEEE Trans. Ind. Elet., vol. 41, nº2, april 1994.

R. Novotony, Thomas A. Lipo. “Introduction to Field Orientation e High Performance AC Drives”. IEEE Ind. Applic. Soc. Ann. Meeting Tutorial Course, 1986.

Robert D. Lorenz, Thomas A. Lipo, Donald W. Novotny. “Motion Control with Induction Motors”. Proceedings of the IEEE, vol. 82, nº8 , august 1994.

Ying-Hu Tzou, Hsiang-Jui Wu. “Multimicroprocessor-Based Robust Control of an AC Induction Servo Motor”. IEEE Trans. Ind. Applic. vol. 26, no. 3, may/june, 1990

Anexo 1

Folha de dados do driver utilizado (SKHI22A) e a placa padrão do fabricante no qual ele é montado.

SKHI 21A (R) ...

SEMIDRIVER™

Hybrid Dual MOSFET Driver

SKHI 21A (R)

Preliminary Data

Features

- drives MOSFETs with $V_{DS(on)}$ < 10 V
- is compatible to old SKHI 21
- CMOS compatible inputs
- Short circuit potection by V_{CE} monitoring and switch off
- Drive interlock top / bottom
- Isolation by transformers
- Supply undervotage protection (13 V)
- Error latch / output

Typical Applications

- Driver for MOSFET modules in bridge circuits in choppers, inverter drives, UPS and welding inverters

1) see fig. 6

2) At R_{CE} = 18 kΩ, C_{CE} = 330 pF

Absolute Maximum Ratings

Symbol	Conditions	Values	Units
V_S	Supply voltage prim.	18	V
V_{iH}	Input signal volt. (High)	V_S + 0,3	V
$Iout_{PEAK}$	Output peak current	8	A
$Iout_{AVmax}$	Output average current	40	mA
f_{max}	max. switching frequency	50	kHz
V_{CE}	Collector emitter voltage sense across the IGBT	1200	V
dv/dt	Rate of rise and fall of voltage secondary to primary side	50	kV/µs
V_{isolIO}	Isolation test voltage input - output (2 sec. AC)	2500	Vac
V_{isol12}	Isolation test voltage output 1 - output 2 (2 sec. AC)	1500	V
R_{Gonmin}	Minimum rating for R_{Gon}	3	Ω
$R_{Goffmin}$	Minimum rating for R_{Goff}	3	Ω
$Q_{out/pulse}$	Max. rating for output charge per pulse	4[1)]	µC
T_{op}	Operating temperature	- 40 ... + 85	°C
T_{stg}	Storage temperature	- 40 ... + 85	°C

Characteristics T_a = 25 °C, unless otherwise specified

Symbol	Conditions	min.	typ.	max.	Units
V_S	Supply voltage primary side	14,4	15	15,6	V
I_{SO}	Supply current primary side (no load)		80		mA
	Supply current primary side (max.)			290	mA
V_i	Input signal voltage on/off		15 / 0		V
V_{iT+}	Input threshold voltage (High)	10,9	11,7	12,5	V
V_{iT-}	Input threshold voltage (Low)	4,7	5,5	6,5	V
R_{in}	Input resistance		10		kΩ
$V_{G(on)}$	Turn on gate voltage output		+ 15		V
$V_{G(off)}$	Turn off gate voltage output		0		V
R_{GE}	Internal gate-emitter resistance		22		kΩ
f_{ASIC}	Asic system switching frequency		8		MHz
$t_{d(on)IO}$	Input-output turn-on propagation time	0,85	1	1,15	µs
$t_{d(off)IO}$	Input-output turn-off propagation time	0,85	1	1,15	µs
$t_{d(err)}$	Error input-output propagation time		0,6		µs
$t_{pERRRESET}$	Error reset time		9		µs
t_{TD}	Top-Bot Interlock Dead Time	3,3		4,3	µs
V_{CEsat}	Reference voltage for V_{CE}-monitoring		5[2)]	10	V
C_{ps}	Coupling capacitance primary secondary		12		pF
MTBF	Mean Time Between Failure T_a = 40°C		2,0		10^6 h
w	weight		45		g

This technical information specifies semiconductor devices but promises no characteristics. No warranty or guarantee expressed or implied is made regarding delivery, performance or suitability.

SKHI 22 A / B (R) ...

SEMIDRIVER™

Hybrid Dual IGBT Driver

SKHI 22 A / B (R)

Preliminary Data

Features

- Double driver for halfbridge IGBT modules
- SKHI 22A is compatible to old SKHI 22
- SKHI 22B has additional functionality
- CMOS compatible inputs
- Short circuit potection by V_{CE} monitoring and switch off
- Drive interlock top / bottom
- Isolation by transformers
- Supply undervotage protection (13 V)
- Error latch / output

Typical Applications

- Driver for IGBT modules in bridge circuits in choppers, inverter drives, UPS and welding inverters

1) see fig. 6

2) At R_{CE} = 18 kΩ, C_{CE} = 330 pF

Absolute Maximum Ratings

Symbol	Conditions	Values	Units
V_S	Supply voltage prim.	18	V
V_{iH}	Input signal volt. (High) SKHI 22A	V_S + 0,3	V
	SKHI 22B	5 + 0,3	V
$Iout_{PEAK}$	Output peak current	8	A
$Iout_{AVmax}$	Output average current	40	mA
f_{max}	max. switching frequency	50	kHz
V_{CE}	Collector emitter voltage sense across the IGBT	1200	V
dv/dt	Rate of rise and fall of voltage secondary to primary side	50	kV/µs
V_{isolIO}	Isolation test voltage input - output (2 sec. AC)	2500	Vac
V_{isol12}	Isolation test voltage output 1 - output 2 (2 sec. AC)	1500	V
R_{Gonmin}	Minimum rating for R_{Gon}	3	Ω
$R_{Goffmin}$	Minimum rating for R_{Goff}	3	Ω
$Q_{out/pulse}$	Max. rating for output charge per pulse	4[1)]	µC
T_{op}	Operating temperature	- 40 ... + 85	°C
T_{stg}	Storage temperature	- 40 ... + 85	°C

Characteristics T_a = 25 °C, unless otherwise specified

Symbol	Conditions	min.	typ.	max.	Units
V_S	Supply voltage primary side	14,4	15	15,6	V
I_{SO}	Supply current primary side (no load)		80		mA
	Supply current primary side (max.)			290	mA
V_i	Input signal voltage SKHI 22A on/off		15 / 0		V
	SKHI 22B on/off		5 / 0		V
V_{iT+}	Input threshold voltage (High) SKHI 22A	10,9	11,7	12,5	V
	SKHI 22B	3,5	3,7	3,9	V
V_{iT-}	Input threshold voltage (Low) SKHI 22A	4,7	5,5	6,5	V
	SKHI 22B	1,5	1,75	2,0	V
R_{in}	Input resistance SKHI 22A		10		kΩ
	SKHI 22B		3,3		kΩ
$V_{G(on)}$	Turn on gate voltage output		+ 15		V
$V_{G(off)}$	Turn off gate voltage output		- 7		V
R_{GE}	Internal gate-emitter resistance		22		kΩ
f_{ASIC}	Asic system switching frequency		8		MHz
$t_{d(on)IO}$	Input-output turn-on propagation time	0,85	1	1,15	µs
$t_{d(off)IO}$	Input-output turn-off propagation time	0,85	1	1,15	µs
$t_{d(err)}$	Error input-output propagation time		0,6		µs
$t_{pERRRESET}$	Error reset time		9		µs
t_{TD}	Top-Bot Interlock Dead Time SKHI 22A	3,3		4,3	µs
	SKHI 22B	no interlock		4,3	µs
V_{CEsat}	Reference voltage for V_{CE}-monitoring		5[2)]	10	V
C_{ps}	Coupling capacitance primary secondary		12		pF
MTBF	Mean Time Between Failure T_a = 40°C		2,0		10^6 h
w	weight		45		g

This technical information specifies semiconductor devices but promises no characteristics. No warranty or guarantee expressed or implied is made regarding delivery, performance or suitability.

SKHI 22 A / B H4 (R) ...

SEMIDRIVER™

Hybrid Dual IGBT Driver

SKHI 22 A / B H4 (R)

Preliminary Data

Features

- Double driver for halfbridge IGBT modules
- SKHI 22A H4 is compatible to old SKHI 22 H4
- SKHI 22B H4 has additional functionality
- CMOS compatible inputs
- Short circuit protection by V_{CE} monitoring and switch off
- Drive interlock top / bottom
- Isolation by transformers
- Supply under voltage protection (13V)
- Error latch / output

Typical Applications

- Driver for IGBT modules in bridge circuits in choppers, inverter drives, UPS and welding inverters
- DC bus voltage up to 1200 V

1) see fig. 6

2) At R_{CE} = 36 kΩ, C_{CE} = 470 pF, R_{VCE} = 1 kΩ

Absolute Maximum Ratings

Symbol	Conditions	Values	Units
V_S	Supply voltage prim.	18	V
V_{iH}	Input signal volt. (High) SKHI 22A H4	V_S + 0,3	V
	SKHI 22B H4	5 + 0,3	V
$Iout_{PEAK}$	Output peak current	8	A
$Iout_{AVmax}$	Output average current	40	mA
f_{max}	max. switching frequency	50	kHz
V_{CE}	Collector emitter voltage sense across the IGBT	1700	V
dv/dt	Rate of rise and fall of voltage secondary to primary side	50	kV/µs
V_{isolIO}	Isolation test voltage input - output (2 sec. AC)	4000	Vac
V_{isol12}	Isolation test voltage output 1 - output 2 (2 sec. AC)	1500	V
R_{Gonmin}	Minimum rating for R_{Gon}	3	Ω
$R_{Goffmin}$	Minimum rating for R_{Goff}	3	Ω
$Q_{out/pulse}$	Max. rating for output charge per pulse	4[1)]	µC
T_{op}	Operating temperature	- 40 ... + 85	°C
T_{stg}	Storage temperature	- 40 ... + 85	°C

Characteristics T_a = 25 °C, unless otherwise specified

Symbol	Conditions	min.	typ.	max.	Units
V_S	Supply voltage primary side	14,4	15	15,6	V
I_{SO}	Supply current primary side (no load)		80		mA
	Supply current primary side (max.)			290	mA
V_i	Input signal voltage SKHI 22A H4 on/off		15 / 0		V
	SKHI 22B H4 on/off		5 / 0		V
V_{iT+}	Input threshold volt. (High) SKHI 22A H4	10,9	11,7	12,5	V
	SKHI 22B H4	3,5	3,7	3,9	V
V_{iT-}	Input threshold volt. (Low) SKHI 22A H4	4,7	5,5	6,5	V
	SKHI 22B H4	1,5	1,75	2,0	V
R_{in}	Input resistance SKHI 22A H4		10		kΩ
	SKHI 22B H4		3,3		kΩ
$V_{G(on)}$	Turn on gate voltage output		+ 15		V
$V_{G(off)}$	Turn off gate voltage output		- 7		V
R_{GE}	Internal gate-emitter resistance		22		kΩ
f_{ASIC}	Asic system switching frequency		8		MHz
$t_{d(on)IO}$	Input-output turn-on propagation time	0,85	1	1,15	µs
$t_{d(off)IO}$	Input-output turn-off propagation time	0,85	1	1,15	µs
$t_{d(err)}$	Error input-output propagation time		0,6		µs
$t_{pERRRESET}$	Error reset time		9		µs
t_{TD}	Top-Bot Interl. Dead Time SKHI 22A H4	3,3		4,3	µs
	SKHI 22B H4	no interlock		4,3	µs
V_{CEsat}	Reference voltage for V_{CE}-monitoring		5[2)]	10	V
C_{ps}	Coupling capacitance primary secondary		12		pF
MTBF	Mean Time Between Failure T_a = 40°C		2,0		10^6 h
w	weight		45		g

This technical information specifies semiconductor devices but promises no characteristics. No warranty or guarantee expressed or implied is made regarding delivery, performance or suitability.

SKHI 22 A / B H4 (R) ...

External Components

Component	Function	Recommended Value
R_{CE}	Reference voltage for V_{CE}-monitoring $V_{CEstat}(V) = \frac{10 \cdot R_{CE}(k\Omega)}{10 + R_{CE}(k\Omega)} - 1{,}4$ (1) with R_{VCE} = 1kΩ (1700V IGBT): $V_{CEstat}(V) = \frac{10 \cdot R_{CE}(k\Omega)}{10 + R_{CE}(k\Omega)} - 1{,}8$ (1.1)	10kΩ < R_{CE} < 100kΩ 18kΩ for SKM XX 123 (1200V) 36kΩ for SKM XX 173 (1700V)
C_{CE}	Inhibit time for V_{CE} - monitoring $t_{min} = \tau_{CE} \cdot \ln\left[\frac{15 - V_{CEstat}(V)}{10 - V_{CEstat}(V)}\right]$ (2) $\tau_{CE}(\mu s) = C_{CE}(nF) \cdot \frac{10 \cdot R_{CE}(k\Omega)}{10 + R_{CE}(k\Omega)}$ (3)	C_{CE} < 2,7nF 0,33nF for SKM XX 123 (1200V) 0,47nF for SKM XX 173 (1700V) 0,5µs < t_{min} < 10µs
R_{VCE}	Collector series resistance for 1700V IGBT-operation	1kΩ / 0,4W
R_{ERROR}	Pull-up resistance at error output $\frac{U_{Pull-Up}}{R_{ERROR}} < 15mA$	1kΩ < R_{ERROR} < 10kΩ
R_{GON}	Turn-on speed of the IGBT [4]	R_{GON} > 3Ω
R_{GOFF}	Turn-off speed of the IGBT [5]	R_{GOFF} > 3Ω

[4] Higher resistance reduces free-wheeling diode peak recovery current, increases IGBT turn-on time.
[5] Higher resistance reduces turn-off peak voltage, increases turn-off time and turn-off power dissipation

PIN array

Fig. 2 shows the pin arrays. The input side (primary side) comprises 10 inputs (SKHI 22A / 21A 8 inputs), forming the interface to the control circuit (see fig.1).

The output side (secondary side) of the hybrid driver shows two symmetrical groups of pins with 4 outputs, each forming the interface to the power module. All pins are designed for a grid of 2,54 mm.

Primary side PIN array

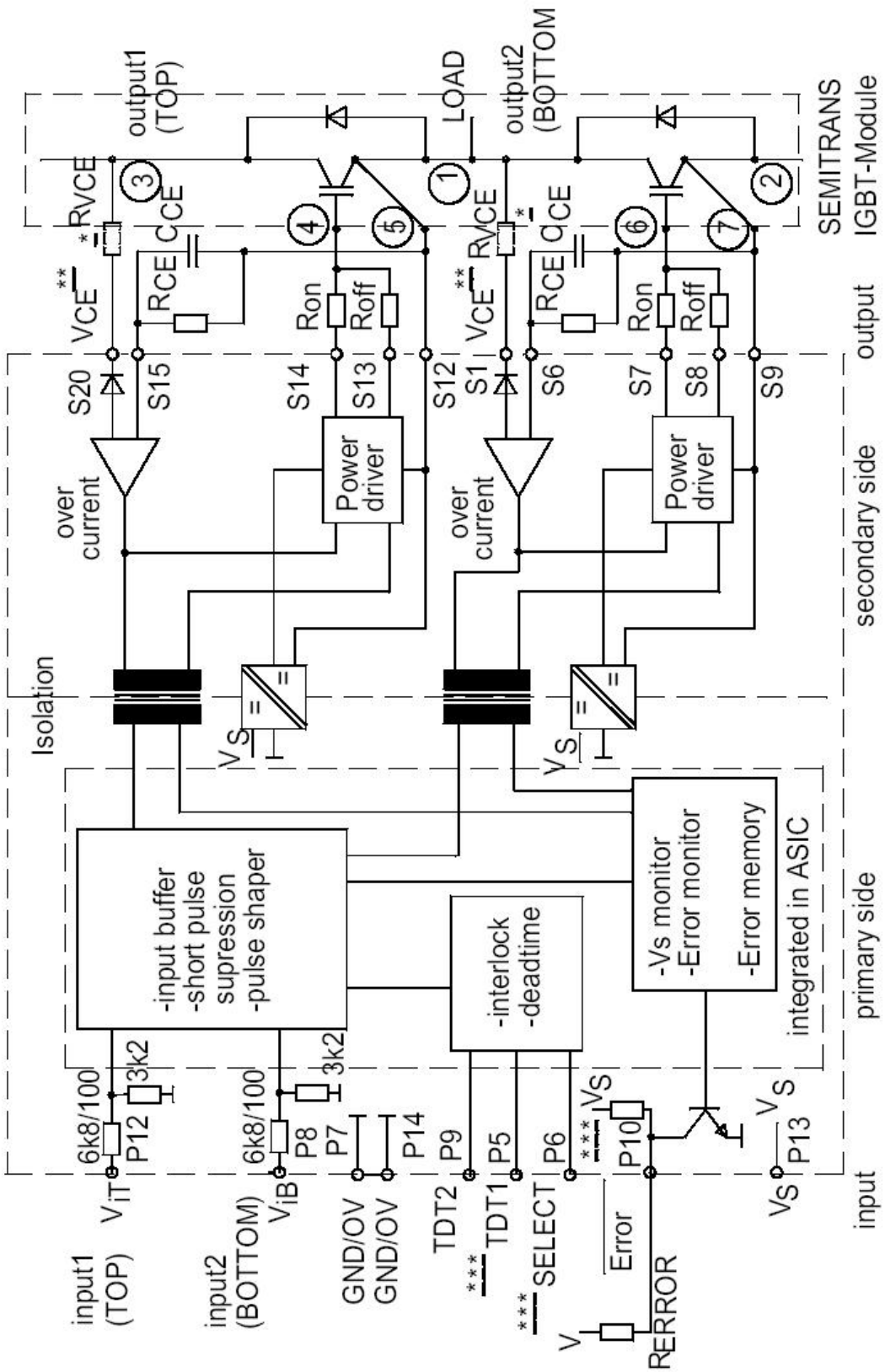
PIN No.	Designation	Explanation
P14	GND / 0V	related earth connection for input signals
P13	V_S	+ 15V ± 4% voltage supply
P12	V_{IN1}	switching signal input 1 (TOP switch) positive 5V logic (for SKHI22A /21A, 15V logic)
P11	free	not wired
P10	/ERROR	error output, low = error; open collector output; max 30V / 15mA (for SKHI22A /21A, internal 10kΩ pull-up resistor versus V_S)
P9	TDT2	signal input for digital adjustment of interlocking time; **SKHI22B: to be switched by bridge to GND (see fig. 3)** **SKHI22A /21A: to be switched by bridge to V_S**
P8	V_{IN2}	switching signal input 2 (BOTTOM switch); positive 5V logic (for SKHI22A /21A, 15V logic)
P7	GND / 0V	related earth connection for input signals
P6	SELECT	signal input for neutralizing locking function; to be switched by bridge to GND
P5	TDT1	signal input for digital adjustment of locking time; to be switched by bridge to GND

ATTENTION: Inputs P6 and P5 are not existing for SKHI 22A/ 21A. The contactor tracks of the digital input signals P5/ P6/ P9 must not be longer than 20 mm to avoid interferences, if no bridges are connected.

Secondary side PIN array

PIN No.	Designation	Explanation
S20	V_{CE1}	collector output IGBT 1 (TOP switch)
S15	C_{CE1}	reference voltage adjustment with R_{CE} and C_{CE}
S14	G_{ON1}	gate 1 R_{ON} output
S13	G_{OFF1}	gate 1 R_{OFF} output
S12	E1	emitter output IGBT 1 (TOP switch)
S1	V_{CE2}	collector output IGBT 2 (BOTTOM switch)
S6	C_{CE2}	reference voltage adjustment with R_{CE} and C_{CE}
S7	G_{ON2}	gate 2 R_{ON} output
S8	G_{OFF2}	gate 2 R_{OFF} output
S9	E2	emitter output IGBT 2 (BOTTOM switch)

ATTENTION: The connector leads to the power module should be as short as possible.



* When SKHI22B is driving 1700V IGBTs, a 1kΩ / 0,4W R_{VCE}-resistor must be connected in series to the V_{CE} input.
** The VCE-terminal is to be connected to the IGBT collector C. If the V_{CE}-monitoring is not used, connect S1 to S9 or S20 to S12 respectively.
*** Terminals P5 and P6 are not existing for SKHI22A/21A; internal pull-up resistor exists in SKHI22A/21A only.
1-7 Connections to SEMITRANS GB-module

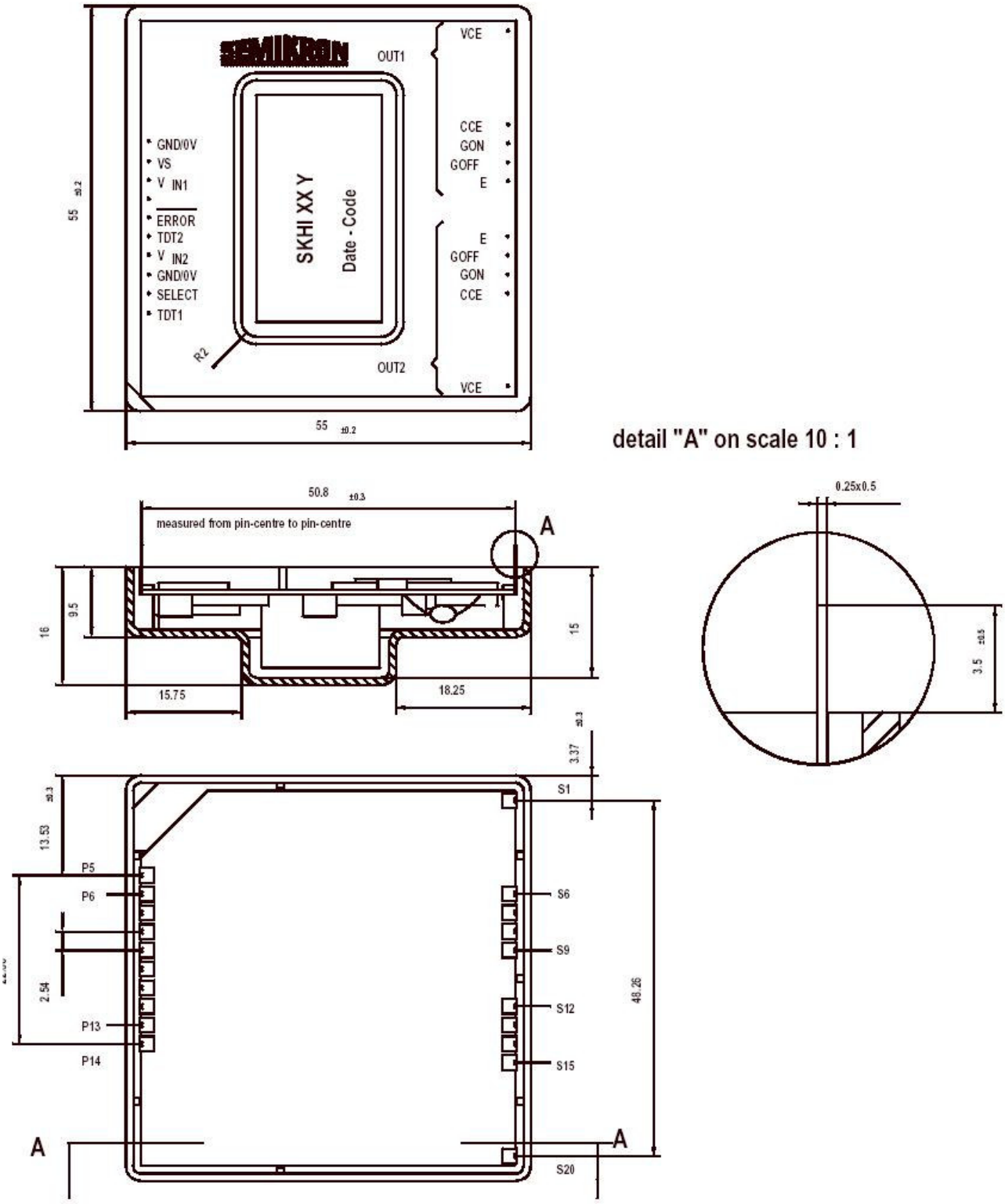


Fig. 2 Dimension drawing and PIN array (P5 and P6 are not existing for SKHI22A/21A)

SEMIDRIVER™

SKHI 22A / 22B und SKHI 21A

Hybrid dual drivers

The driver generation SKHI 22A/B and SKHI 21A will replace the hybrid drivers SKHI 21/22 and is suitable for all available low and medium power range IGBT and MOSFETs.

The SKHI 22A (SKHI 21A) is a form-, fit- and mostly function-compatible replacement to its predecessor, the SKHI 22 (SKHI 21).

The SKHI 22B is recommended for any new development. It has two additional signal pins on the primary side with which further functions may be utilized.

The SKHI 22A and SKHI 22B are available with standard isolation (isolation testing voltage 2500 VAC, 2sec.) as well as with an increased isolation voltage (type "H4") (isolation testing voltage 4000 VAC, 2sec.). The SKHI 21A is only offered with standard isolation features.

Differences SKHI 22-22A (SKHI 21-21A)

Compared to the old SKHI 22/21 the new driver SKHI 22A / 21A is absolutely compatible with regards to pins and mostly with regards to functions. It may be equivalently used in existing PCBs.

The following points have to be considered when exchanging the drivers:

- **Leave out the two resistors RTD for interlocking dead time adjustment at pin 11 and pin 9.**
- **The interlocking time of the driver stages in halfbridge applications is adjusted to 3,25 µs. It may be increased up to 4,25 µs by applying a 15 V (VS) supply voltage at Pin 9 (TDT2) (wire bridge)**
- **The error reset time is typically 9µs.**
- **The input resistance is 10 kΩ.**

As far as the SKHI 22A is concerned, the negative gate voltage required for turn-off of the IGBT is no longer -15V, but -7V.

General description

The new driver generation SKHI 22A/B, SKHI 21A consists of a hybrid component which may directly be mounted to the PCB.

All devices necessary for driving, voltage supply, error monitoring and potential separation are integrated in the driver. In order to adapt the driver to the used power module, only very few additional wiring may be necessary.

The forward voltage of the IGBT is detected by an integrated short-circuit protection, which will turn off the module when a certain threshold is exceeded.

In case of short-circuit or too low supply voltage the integrated error memory is set and an error signal is generated.

The driver is connected to a controlled + 15 V-supply voltage. The input signal level is 0/15 V for the SKHI 22A/ 21A and 0/5 V for the SKHI 22B.

In the following explanations the whole driver family will be designated as SKHI 22B. If a special type is referred to, the concerned driver version will explicitly be named.

Technical explanations[1]

Description of the circuit block diagram and the functions of the driver

The block diagram (fig.1) shows the inputs of the driver (primary side) on the left side and the outputs (secondary side) on the right.

The following functions are allocated to the primary side:

Input-Schmitt-trigger, CMOS compatible, positive logic (input high = IGBT on)

Interlock circuit and deadtime generation of the IGBT

If one IGBT is turned on, the other IGBT of a halfbridge cannot be switched. Additionally, a digitally adjustable interlocking time is generated by the driver (see fig. 3), which has to be longer than the turn-off delay time of the IGBT. This is to avoid that one IGBT is turned on before the other one is not completely discharged. This protec-tion-function may be neutralized by switching the select input (pin6) (see fig. 3). fig. 3 documents possible interlock-times. "High" value can be achieved with no connection and connection to 5 V as well.

P6 ; SELECT	P5 ; TDT1	P9 ; TDT2	interlock time t_{TD}/µs
open / 5V	GND	GND	1,3
open / 5V	GND	open / 5V	2,3
open / 5V	open / 5V	GND	3,3
open / 5V	open / 5V	open / 5V	4,3
GND	X	X	no interlock

Fig. 3 SKHI 22B - Selection of interlock-times: „High"-level can be achieved by no connection or connecting to 5 V

Short pulse suppression

The integrated short pulse suppression avoids very short switching pulses at the power semiconductor caused by high-frequency interference pulses at the driver input signals. Switching pulses shorter than 500 ns are suppressed and not transmitted to the IGBT.

Power supply monitoring (V_S)

A controlled 15 V-supply voltage is applied to the driver. If it falls below 13 V, an error is monitored and the error output signal switches to low level.

1.The following descriptions apply to the use of the hybrid driver for IGBTs as well as for power MOSFETs. For the reason of shortness, only IGBTs will be mentioned in the following. The designations "collector" and "emitter" will refer to IGBTs, whereas for the MOSFETs "drain" and "source" are to be read instead.

Error monitoring and error memory

The error memory is set in case of under-voltage or short-circuit of the IGBTs. In case of short-circuit, an error signal is transmitted by the V_{CE}-input via the pulse transformers to the error memory. The error memory will lock all switching pulses to the IGBTs and trigger the error output (P10) of the driver. The error output consists of an open collector transistor, which directs the signal to earth in case of error. SEMIKRON recommends the user to provide for a pull-up resistor directly connected to the error evaluation board and to adapt the error level to the desired signal voltage this way. The open collector transistor may be connected to max. 30 V / 15 mA. If several SKHI 22Bs are used in one device, the error terminals may also be paralleled.

ATTENTION: Only the SKHI 22A / 21A is equipped with an internal pull-up resistor of 10 kΩ versus V_S. The SKHI 22B does not contain an internal pull-up resistor.

The error memory may only be reset, if no error is pending and both cycle signal inputs are set to low for > 9 µs at the same time.

Pulse transformer set

The transformer set consists of two pulse transformers one is used bidirectional for turn-on and turn-off signals of the IGBT and the error feedback between primary and secondary side, the other one for the DC/DC-converter. The DC/DC-converter serves as potential-separation and power supply for the two secondary sides of the driver. The isolation voltage for the "H4"-type is 4000 V_{AC} and 2500 V_{AC} for all other types.

The secondary side consists of two symmetrical driver switches integrating the following components:

Supply voltage

The voltage supply consists of a rectifier, a capacitor, a voltage controller for - 7 V and + 15 V and a + 10 V reference voltage.

Gate driver

The output transistors of the power drivers are MOSFETs. The sources of the MOSFETs are separately connected to external terminals in order to provide setting of the turn-on and turn-off speed by the external resistors R_{ON} and R_{OFF}. Do not connect the terminals S7 with S8 and S13 with S14, respectively. The IGBT is turned on by the driver at + 15 V by R_{ON} and turned off at - 7 V by R_{OFF}. R_{ON} and R_{OFF} may not chosen below 3 Ω. In order to ensure locking of the IGBT even when the driver supply voltage is turned off, a 22 kΩ-resistor versus the emitter output (E) has been integrated at output G_{OFF}.

V_{CE}-monitoring

The V_{CE}-monitoring controls the collector-emitter voltage V_{CE} of the IGBT during its on-state. V_{CE} is internally limited to 10 V. If the reference voltage V_{CEref} is exceeded, the IGBT will be switched off and an error is indicated. The reference voltage V_{CEref} may dynamically be adapted to the IGBTs switching behaviour. Immediately after turn-on of the IGBT, a higher value is effective than in the steady state. This value will, however, be reset, when the IGBT is turned off. V_{CEstat} is the steady-state value of V_{CEref} and is adjusted to the required maximum value for each IGBT by an external resistor R_{CE} to be connected between the terminals C_{CE} (S6/S15) and E (S9/S12). It may not exceed 10 V. The time constant for the delay of V_{CEref} may be increased by an external capacitor C_{CE}, which is connected in parallel to R_{CE}. It controls the time tmin which passes after turn-on of the IGBT before the V_{CE}-monitoring is activated. This makes possible any adaptation to the switching behavior of any of the IGBTs. After t_{min} has passed, the V_{CE}-monitoring will be triggered as soon as $V_{CE} > V_{CEref}$ and will turn off the IGBT.

External components and possible adjustments of the hybrid driver

Fig. 1 shows the required external components for adjustment and adaptation to the power module.

VCE - monitoring adjustment

The external components R_{CE} and C_{CE} are applied for adjusting the steady-state threshold and the short-circuit monitoring dynamic. R_{CE} and C_{CE} are connected in parallel to the terminals C_{CE} (S15/ S6) and E (S12/ S9) .

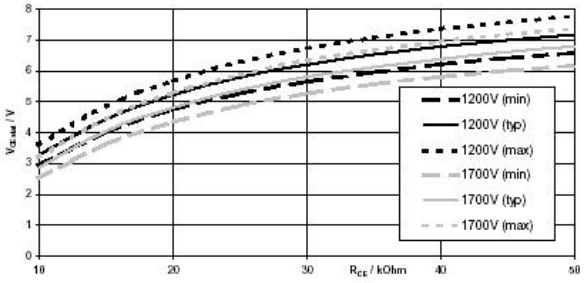


Fig. 4 V_{CEstat} in dependence of R_{CE} (T_{amb} = 25°C)

Dimensioning of R_{CE} and C_{CE} can be done in three steps:

1. Calculate the maximum forward voltage from the datasheet of the used IGBT and determine V_{CEstat}
2. Calculate approximate value of R_{CE} according to equation (1) or (1.1) from V_{CEstat} or determine R_{CE} by using fig.4.
3. Determine t_{min} and calculate C_{CE} according to equations (2) and (3).

Typical values are

for 1200 V IGBT: V_{CEstat} = 5 V; t_{min} = 1,45 µs,
R_{CE} = 18 kΩ, C_{CE} = 330 pF

for 1700 V IGBT: V_{CEstat} = 6 V; t_{min} = 3 µs,
R_{CE} = 36 kΩ, C_{CE} = 470 pF

Adaptation to 1700 V IGBT

When using 1700 V IGBTs it is necessary to connect a 1 kΩ / 0,4 W adaptation resistor between the V_{CE}-terminal (S20/ S1) and the respective collector.

Adaptation to error signal level

An open collector transistor is used as error terminal, which, in case of error, leads the signal to earth. The signal has to be adapted to the evaluation circuit's voltage level by means of an externally connected pull-up resistor. The maximum load applied to the transistor shall be 30 V / 15 mA.

As for the SKHI 22A / 21A a 10 kΩ pull-up resistor versus V_S (P13) has already been integrated in the driver.

IGBT switching speed adjustment

The IGBT switching speed may be adjusted by the resistors R_{ON} and R_{OFF}. By increasing R_{ON} the turn-on speed will decrease. The reverse peak current of the free-wheeling diode will diminish. SEMIKRON recommends to adjust R_{ON} to a level that will keep the turn-on delay time $t_{d(on)}$ of the IGBT < 1 µs.

By increasing R_{OFF} the turn-off speed of the IGBT will decrease. The inductive peak overvoltage during turn-off will diminish.

The minimum gate resistor value for R_{OFF} and R_{ON} is 3 Ω. Typical values for R_{ON} and R_{OFF} recommended by SEMIKRON are given in fig. 5

SK-IGBT-Modul	R_{Gon} Ω	R_{Goff} Ω	C_{CE} pF	R_{CE} kΩ	R_{VCE} kΩ
SKM 50GB123D	22	22	330	18	0
SKM 75GB123D	22	22	330	18	0
SKM 100GB123D	15	15	330	18	0
SKM 145GB123D	12	12	330	18	0
SKM 150GB123D	12	12	330	18	0
SKM 200GB123D	10	10	330	18	0
SKM 300GB123D	8,2	8,2	330	18	0
SKM 400GA123D	6,8	6,8	330	18	0
SKM 75GB173D	15	15	470	36	1
SKM 100GB173D	12	12	470	36	1
SKM 150GB173D	10	10	470	36	1
SKM 200GB173D	8,2	8,2	470	36	1

Fig. 5 Typical values for external components

Interlocking time adjustment

Fig. 3 shows the possible interlocking times between output1 and output2. Interlocking times are adjusted by connecting the terminals TDT1 (P5), TDT2 (P9) and SELECT (P6) either to earth/ GND (P7 and P14) according to the required function or by leaving them open.

A typical interlocking time value is 3,25 µs (P9 = GND; P5 and P6 open). For SKHI 22A / 21A the terminals TDT1 (P5) and SELECT (P6) are not existing. The interlocking time has been fixed to 3,25 µs and may only be increased to 4,25 µs by connecting TDT2 (P9) to V_S (P13).

ATTENTION: If the terminals TDT1 (P5), TDT2 (P9) and SELECT (P6) are not connected, eventually connected track on PC-board may not be longer than 20 mm in order to avoid interferences.

SEMIKRON recommends to start-up operation using the values recommended by SEMIKRON and to optimize the values gradually according to the IGBT switching behaviour and overvoltage peaks within the specific circuitry.

Driver performance and application limits

The drivers are designed for application with halfbridges and single modules with a maximum gate charge Q_{GE} < 4 µC (see fig. 6).

The charge necessary to switch the IGBT is mainly depending on the IGBT's chip size, the DC-link voltage and the gate voltage.

This correlation is also shown in the corresponding module datasheet curves.

It should, however, be considered that the SKHI 22B is turned on at + 15 V and turned off at - 7 V. Therefore, the gate voltage will change by 22 V during every switching procedure.

Unfortunately, most datasheets do not indicate negative gate voltages. In order to determine the required charge, the upper leg of the charge curve may be prolonged to + 22 V for determination of approximate charge per switch.

The medium output current of the driver is determined by the switching frequency and the gate charge. For the SKHI 22B the maximum medium output current is $Iout_{AVmax}$ < ± 40 mA.

The maximum switching frequency f_{MAX} may be calculated with the following formula, the maximum value however being 50 kHz due to switching reasons:

$$f_{MAX}(kHz) = \frac{4 \cdot 10^4}{Q_{GE}(nC)}$$

Fig. 6 shows the maximum rating for the output charge per pulse for different gate resistors.

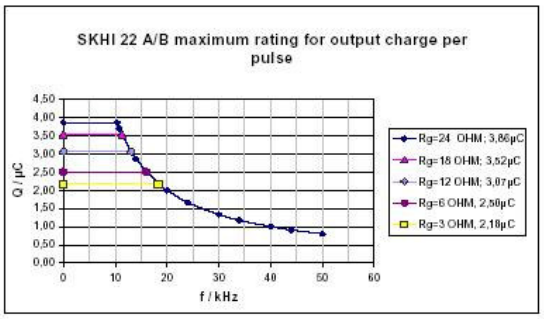


Fig. 6 Maximum rating for output charge per pulse

Further application notes

The CMOS-inputs of the hybrid driver are extremely sensitive to over-voltage. Voltages higher than V_S + 0,3 V or below – 0,3 V may destroy these inputs. Therefore, control signal over-voltages exceeding the above values have to be avoided.

Please provide for static discharge protection during handling. As long as the hybrid driver is not completely assembled, the input terminals have to be short-circuited.

Persons working with CMOS-devices have to wear a grounded bracelet. Any synthetic floor coverings must not be statically chargeable. Even during transportation the input terminals have to be short-circuited using, for example, conductive rubber. Worktables have to be grounded. The same safety requirements apply to MOSFET- and IGBT-modules!

The connecting leads between hybrid driver and the power module should be as short as possible, the driver leads should be twisted.

Any parasitic inductances within the DC-link have to be minimized. Over-voltages may be absorbed by C- or RCD-snubbers between the main terminals for PLUS and MINUS of the power module.

When first operating a newly developed circuit, SEMIKRON recommends to apply low collector voltage and load current in the beginning and to increase these values gradually, observing the turn-off behaviour of the free-wheeling diode and the turn-off voltage spikes generated accross the IGBT. An oscillographic control will be necessary. In addition to that the case temperature of the module has to be monitored. When the circuit works correctly under rated operation conditions, short-circuit testing may be done, starting again with low collector voltage.

It is important to feed any errors back to the control circuit and to switch off the device immediately in such events. Repeated turn-on of the IGBT into a short circuit with a high frequency may destroy the device.

Mechanical fixing on PCB:

In applications with mechanical vibrations (vehicles) do not use a ty-rap for fixing the driver, but - after soldering and testing - apply special glue. Recommended types: CIBA GEIGY XP 5090 + 5091; PACTAN 5011; WACKER A33 (ivory) or N199 (transparent), applied around the case edge (forms a concave mould). The housing may not be pressed on the PCB; do not twist the PCB with the driver soldered on, otherwise the internal ceramics may crack. The driver is not suitable for big PCBs.

Proven, within the scope of the product qualification, was the use of the driver with the printed circuit board SKPC 2006 (L x B x H = 97,0 x 67,5 x 1,5 mm). During the test, the driver was stuck with glue on the printed circuit board. Based on this information the technical conclusion arises, that in an application with big printed circuit boards, this board must be supported and reinforced in the area of the driver.

If a PCB is directly plugged to IGBT modules, the PCB has to be fixed to the heat sink by thread bolts.

The temperature of the solder must not exceed 265℃ , and solder time must not exceed 4 seconds. The ambient temperature must not exceed the specified maximum storage temperature of the driver.
The driver is not suited for hot air reflow or infrared reflow soldering processes.

All electrical and mechanical parameters should be validated by user´s technical experts for each application.

For further details please contact SEMIKRON.

SEMIKRON
SEMICONDUTORES
SEMIKRON Semicondutores Ltda
Av. Inocêncio Seráfico, 6300
06366-900 - Carapicuíba - SP
Brasil
Tel.: (0xx11) 7286.1055
Fax: (0xx11) 7286.3567

SKPC - 2122

Placa de circuito impresso para SKHI 22, SKHI 22H4 e SKHI 21

Diagrama de interligação:

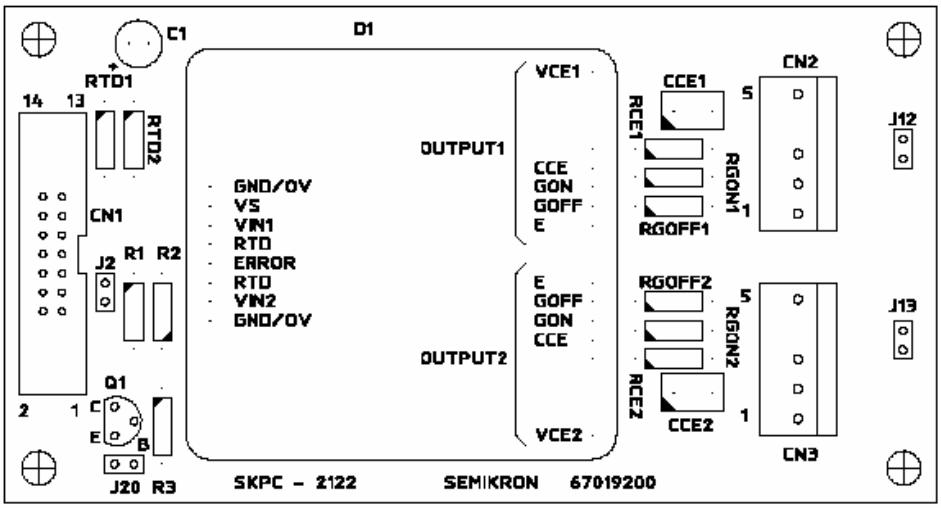


Fig. 1 Desenho geral da placa

CN1 - Conector de entrada (para cabo-plano)

Pino	Função
1	Blindagem (ver J20)
2	Entrada IGBT inferior (0 / 15V)
3	Erro (ativo em zero - ver J2)
4	Entrada IGBT superior (0 / 15V)
8	Vs (+15V)
9	Vs (+15V)
10	Terra (0V)
11	Terra (0V)
outros	Não usados

CN2 / CN3 - Conectores de saída para os IGBTs

Pino	IGBT simples	IGBTs em paralelo
1	Emissor	Emissor
2	Não usado	Gate OFF
3	Gate	Gate ON
4	Não usado	Não usado
5	Terminal de coletor	Terminal de coletor

Jumpers - Configuração da placa

- **J2** - Polaridade do sinal de erro - para inverter o sinal de erro é necessário abrir J2 e montar o circuito formado por R1, R2, R3 e Q1. Este jumper está normalmente fechado (erro ativo em zero).
- **J12** - Interligação entre os pinos 2 e 3 de CN2 - usado para permitir o uso de IGBT único (J12 fechado) ou em paralelo (J12 aberto). O padrão é fechado.
- **J13** - Interligação entre os pinos 2 e 3 de CN3 - usado para permitir o uso de IGBT único (J13 fechado) ou em paralelo (J13 aberto). O padrão é fechado.
- **J20** - Permite a interligação do pino 1 de CN1 ao 0V da placa, para ser usado como blindagem. O padrão é aberto (não conectado ao terra).

Observações:

- Os valores dos componentes devem ser determinados de acordo com a aplicação, favor consultar a folha de dados do driver ou contactar a Semikron.
- Para controlar IGBTs em paralelo é necessário o uso de componentes extras.
- O cabo-plano de entrada e os fios de saída devem ser curtos, para evitar a captação de ruído pelo driver ou pelos IGBTs. Se o cabo de entrada ultrapassar 50cm, então deverá ser blindado. Cabos acima de 1m de comprimento não são recomendados.

Anexo 2

Folha de dados da placa de aquisição de dados

CHAPTER 1. INTRODUCTION

The PCL-711B PC-MultiLab Card is an easy-to-use and cost/effective IBM PC/XT/AT compatible multifunction data acquisition card. This card's specifications and user-friendly software driver make it a popular solution for a wide range of industrial and laboratory applications. Such applications include: data acquisition, process control, automatic testing, and factory automation.

1.1. Features

- 12-bit resolution A/D conversion
- Accepts 8 single-ended analog inputs
- Programmable analog input ranges: ±5V, ±2.5V, ±1.25, ±0.625V, ±0.3125V
- Support software trigger, programmable pacer trigger, and external trigger
- Programmable IRQ level for A/D data transfer
- One 12-bit multiplying D/A output channel, with the output range of 0 to +5V or 0 to +10V
- On-board 16-bit digital input and digital output
- Versatile language drivers including BASIC, PASCAL, C and C++

1.2. Specifications

1.2.1. Analog Input (A/D Converter)

Channels: 8 single-ended inputs.

Resolution: 12 bits, successive approximation.

Input range: ±5V, ±2.5V, ±1.25, ±0.625V, and ±0.3125V, software programmable.

Converter:	AD574 or equivalent.
Conversion time*:	25μs max.
Accuracy:	0.015% of reading ± 1 LSB.
Nonlinearity:	±1 bit.
Amplification gains:	x1, x2, x4, x8, and x16, software programmable
Trigger mode:	By software, pacer and external trigger.
Data transfer:	By software or interrupt.
Overvoltage:	Continuous ±30V max.
IRQ level:	IRQ2 to IRQ7.

1.2.2. Analog Output (D/A Converter)

Channels:	One channel.
Resolution:	12 bits
Output range:	0 to +5V or 0 to +10V.
Settling time:	30μs.
Reference voltage:	Internal -5V and -10V (± 0.05V).
Converter:	PM7548GP or equivalent.
Nonlinearity:	±½ LSB.
Output capacity:	±5mA max.

Note:

1. The hardware of PCL-711B/711S features 25 KS/s sampling rate during data acquisition process and the data transfer rate from PCL-711B/711S board to host PC depends on the computer hardware architecture and software environment. The rates may vary due to programming language, code efficiency, CPU utilization and so on.
2. For example, when you are using interrupt data transfer mode of PCL-711S to acquire data in Windows operating system, the acquired data would be overlapped when the sampling rate is faster than the period that the OS could handle the interrupt service routine requested by PCL-711S.

1.2.3. Digital Input

Channels: 16 bits, TTL compatible

Input voltage: Low - 0.8V max.
High - 2.0V min.

Input load: Low - 0.4mA max. @0.5V
High - 0.05mA max. @2.7

1.2.4. Digital Output

Channel: 16 bits, TTL compatible

Output voltage: Low (sink): 8mA @0.5V max.
High (source): 0.4mA @2.4V min.

1.2.5. General Specifications

Power consumption:
+5V: 100mA, typical; 500mA max.
+12V: 40mA, typical; 100mA max.
-12V: 20mA, typical; 50mA max.

I/O connector:
- One 20-pin connector for A/D and D/A
- One 20-pin connector for digital input
- One 20-pin connector for digital output

I/O ports: requires 16 consecutive I/O ports per card

Operating temperature: 0 to 50°C (32 to 122°F).

Storage temperature: -20 to 65°C (-4 to 149°F).

Weight: 127 gm (4.49 oz.).

CHAPTER 2. INSTALLATION

2.1. Initial Inspection

The PCL-711B was thoroughly inspected before being shipped to you. Before installing the card into your PC, make sure that everything has been included with the package. You should also inspect the card for any defects or damages that may have occurred during shipment. If you find anything missing, defective or damaged, contact your PC-LabCard dealer immediately.

Here is a list of the materials included with your PCL-711B package:

- One PCL-711B PC-MultiLab Card
- One User's Manual
- One utility diskette which includes the card's software driver

In the PCL-711S package, two additional accessories are also included:

- One PCLD-7115 Wiring Terminal Board
- One 1 meter cable

2.2. Switch and Jumper Settings

The PCL-711B has been designed with ease-of-use in mind. On board the card you will notice that there is only one DIP switch (SW1), and only one set of jumper pins (JP1). These are used to set the PCL-711B's base address, and to select its D/A output voltage range. The following sub-sections go into this in more detail.

2.2.1. I/O Address Selection

Most peripheral devices and interface cards are controlled via your PC's I/O ports. These devices and cards should be placed in an appropriate I/O space so that there will be no conflicts between them and the PCL-711B. Keep in mind that the PCL-711B uses 16 consecutive address locations in your PC's I/O space. Appendix A provides an I/O port address map for your reference. This will assist you in locating an appropriate address for your peripheral devices and interface cards.

I/O port base addresses are selected from the 6-position DIP switch, SW1, on-board the PCL-711B. Valid addresses are from 000 to 3F0 (hexadecimal). The factory default address setting is 220. From time to time, you may find that you will have to use some of these spaces for other devices. If this is the case, then you can change the address according to the information given in the following table.

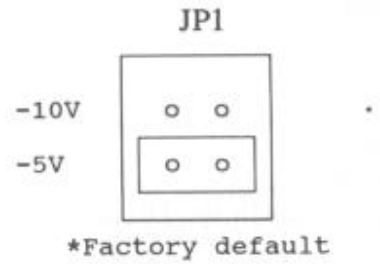
I/O ADDRESS RANGE	SWITCH POSITION (SW1)					
	1	2	3	4	5	6
(HEXADECIMAL)	A9	A8	A7	A6	A5	A4
000 - 00F	0	0	0	0	0	0
100 - 10F	0	1	0	0	0	0
.						
.						
200 - 20F	1	0	0	0	0	0
210 - 21F	1	0	0	0	0	1
220 - 22F *	1	0	0	0	1	0
.						
300 - 30F	1	1	0	0	0	0
.						
3F0 - 3FF	1	1	1	1	1	1

NOTE: 0 = ON, 1 = OFF
A4 through A9 correspond to your PC's address lines.
* Factory default

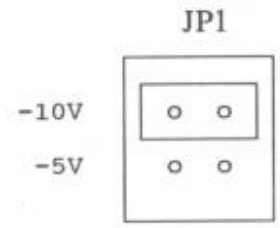
2.2.2. D/A Range Selection

PCL-711B's D/A output range depends on the reference voltage you select at the jumper, JP1. The reference voltages can be assigned as either -5V or -10V, which gives you an D/A output range of 0 to +5V or 0 to +10V, respectively.

When an D/A output range of 0 to +5V is required, set the jumper to -5V (see the illustration below).



If your application requires an D/A output range of 0 to +10V, then set the jumper to -10V (refer to the illustration below).



NOTE: When -10V reference is selected, the maximum D/A output voltage should be +10V. However, it may be less than +10V because the +12V voltage source supplied by your PC may be lower than +11.5V which is required by the D/A circuit.

2.3. Connector Pin Assignment

The PCL-711B is equipped with three 20-pin connectors. Two connectors are located at CN3 and CN4. These are used for digital input (CN4), and digital output (CN3). The third connector is located at CN1, and is used for analog I/O.

Each of these connectors can be connected with the same type of ribbon cables. They can also be connected to a D37 connector using the PCLK-1050 industrial wiring kit.

An illustration of each of these connectors are given in the following illustrations.

Key:

A/D	=	Analog input
AGND	=	Analog ground
D/A	=	Analog output
D/O	=	Digital output
D/I	=	Digital input
DGND	=	Digital ground
VREF	=	Voltage reference
STROBE	=	External signal to latch the D/I data

Analog I/O (CN1)

A/D 0	1	2	AGND
A/D 1	3	4	AGND
A/D 2	5	6	AGND
A/D 3	7	8	AGND
A/D 4	9	10	AGND
A/D 5	11	12	AGND
A/D 6	13	14	AGND
A/D 7	15	16	AGND
D/A	17	18	AGND
AGND	19	20	AGND

Digital Output (CN3)

D/O 0	1	2	D/O 1
D/O 2	3	4	D/O 3
D/O 4	5	6	D/O 5
D/O 6	7	8	D/O 7
D/O 8	9	10	D/O 9
D/O 10	11	12	D/O 11
D/O 12	13	14	D/O 13
D/O 14	15	16	D/O 15
DGND	17	18	DGND
+5V	19	20	+12V

Digital Input (CN4)

D/I 0	1	2	D/I 1
D/I 2	3	4	D/I 3
D/I 4	5	6	D/I 5
D/I 6	7	8	D/I 7
D/I 8	9	10	D/I 9
D/I 10	11	12	D/I 11
D/I 12	13	14	D/I 13
D/I 14	15	16	D/I 15
DGND	17	18	DGND
+5V	19	20	STROBE

CHAPTER 3. CONTROLLING THE PCL-711B

This chapter has been written for those of you who wish to write their own software driver instead of using the PCL-711B's. Here, you will find detailed information about the PCL-711B's register formats and control procedures.

3.1. I/O Port Address Map

The following table shows you which base I/O addresses are used by the PCL-711B. Refer to this map from time to time in order to become familiar with each of the card's register formats and their purpose. 16 consecutive registers corresponding to their I/O addresses are used to control the PCL-711B's various functions. The following table has been provided in this chapter as a preface which outlines these addresses relative to their location and control (read or write) assignments.

LOCATION	READ	WRITE
BASE+0	Counter 0	Counter 0
BASE+1	Counter 1	Counter 1
BASE+2	Counter 2	Counter 2
BASE+3	N/A	Counter Control
BASE+4	A/D low byte	D/A low byte
BASE+5	A/D high byte	D/A high byte
BASE+6	D/I low byte	N/A
BASE+7	D/I high byte	N/A
BASE+8	N/A	Clear interrupt status
BASE+9	N/A	Gain control
BASE+10	N/A	Multiplexer scan control
BASE+11	N/A	Mode and interrupt control
BASE+12	N/A	Software A/D trigger
BASE+13	N/A	D/O low byte
BASE+14	N/A	D/O high byte
BASE+15	N/A	N/A

The sections that follow provide further information about each register's data format according to its specific operation.

3.2. A/D Conversion

3.2.1. A/D Data Registers

The PCL-711B uses the data registers located at I/O ports BASE+4 and BASE+5 to store the converted A/D data. The low byte data is stored at BASE+4, and the high byte data is stored at BASE+5.

BASE+4 A/D Low Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0

BASE+5 A/D High Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	DRDY	AD11	AD10	AD9	AD8

Where:

AD0 through AD11: Represent the PCL-711B's A/D data bits. AD0 is the Least Significant Bit (LSB), and AD11 is the Most Significant Bit (MSB).

DRDY: Data ready bit. When A/D conversion is in progress, this bit remains as 1. It becomes 0 when the A/D conversion is completed. It will become to 1 after reading the low byte A/D data from BASE+4.

3.2.2. Gain Control Register

BASE+9 is used to set the PCL-711B's amplification gain for A/D conversion. The PCL-711B provides five different gains: x1, x2, x4, x8, and x16.

The following tables outline BASE+9's register format and corresponding gain settings:

BASE+9 Gain Control Register (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	G2	G1	G0

G2	G1	G0	GAIN
0	0	0	**x1**
0	0	1	**x2**
0	1	0	**x4**
0	1	1	**x8**
1	0	0	**x16**

3.2.3. Multiplexer Scan Register

The PCL-711B can multiplex up to 8 channels of analog input. Users have to set this register, located at BASE+10, to select to the desired channel , which is going to be measured, before performing any A/D conversion. The register format is as below:

BASE+10 Multiplexer Scan Control (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	C2	C1	C0

C2	C1	C0	CH.
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

3.2.4. Mode and Interrupt Control Register

The PCL-711B's A/D conversion can be triggered in one of the following three ways:

- **By software**
  Writing any data to BASE+12 will generate a trigger pulse to the PCL-711B's on-board A/D converter.

- **By the on-board clock (pacer)**
  The PCL-711B is equipped with a Intel 8253, a programmable interval timer/ counter, to generate precise clock output (pacer). The pacer clock rate of the PCL-711B is between 0.5MHz and 35 minutes per pulse. (See section 3.7 for details on programming the Intel 8253 timer/counter.)

- **By external pulse**
  The PCL-711B allows users to use external signal, from D/I 0 (Pin 1 of the connector CN4), as the A/D trigger pulse. The A/D conversion will be triggered at the rising edge of the external signal.

Also, the PCL-711B provides two ways to transfer the converted A/D data to certain variables:

- **By software control (foreground)**
  The software control data transfer utilizes advantage of the foreground polling concept. After the A/D converter has been triggered, the application program should keep checking the DRDY bit of I/O port BASE+5 until the DRDY bit is detected as 0. Then the program should read data from BASE+4 and BASE+5 to get the whole converted data.

- **By interrupt (background)**
  The PCL-711B also provides background data transfer support, if it is programmed to be in the interrupt data transfer mode. If the PCL-711B is in interrupt data transfer mode, it will generate an interrupt to your PC after each A/D conversion is completed. The corresponding ISR (interrupt service routine) should handle everything to transfer the converted data to memory variables in your program.

I/O port BASE+11 is used to set the PCL-711B's operation mode and the IRQ level. The register format is as following:

BASE+11 Mode and Interrupt Control Register (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	I3	I1	I0	-	S2	S1	S0

Where:

S0 to S2: mode selection

S2	S1	S0	Operation Mode
0	0	0	S/W trigger with S/W data transfer
0	0	1	
0	1	0	External trigger * with S/W data transfer
0	1	1	External trigger * with INT data transfer
1	0	0	Pacer trigger with S/W data transfer
1	0	1	Reserved
1	1	0	Pacer trigger with INT data transfer
1	1	1	Reserved

Note: External trigger signal go through DI0 of CN4

I0 to I2: IRQ level selection

I2	I1	I0	Interrupt Level
0	0	0	IRQ2
0	0	1	N/A
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

3.2.5. Interrupt Status Register

If the PCL-711B is in interrupt data transfer mode, a hardware status flag will be set after each A/D conversion. Users have to clear the status flag, by writing any data to BASE+8, to let the PCL-711B accept next interrupt.

BASE+8 Clear Interrupt Status (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	-

3.2.6. Software Trigger Register

Writing any data to BASE+12 will generate a trigger pulse to the PCL-711B 's A/D converter.

BASE+12 Software A/D Trigger (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	-	-	-	-

3.3. D/A Conversion

The PCL-711B provides one D/A output channel. The low byte and the high byte D/A data are set via BASE+4 and BASE+5 respectively.

Since the PCL-711B uses so-called double-buffer D/A output technology to avoid output glitch, the low byte data should be written first and the high byte data second, that is write BASE+4 first and BASE+5 second. The D/A output will not change until BASE+5 is updated.

BASE+4 D/A Low Byte Data (Write)

D7	D6	D5	D4	D3	D2	D1	D0
DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0

BASE+5 D/A High Byte Data (Write)

D7	D6	D5	D4	D3	D2	D1	D0
-	-	-	-	DA11	DA10	DA9	DA8

where:

DA0 through DA11: the D/A output data. DA0 represents the D/A' LSB data, while DA11 represents the D/A' MSB data.

3.4. Digital Input and Output

3.4.1. Digital Input Registers

The PCL-711B provides 16 bits of digital input. The registers are located at BASE+6 and BASE+7.

BASE+6 D/I Low Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0

BASE+7 D/I High Byte Data (Read)

D7	D6	D5	D4	D3	D2	D1	D0
DI15	DI14	DI13	DI12	DI11	DI10	DI9	DI8

3.4.2. Digital Output Registers

The PCL-711B provides 16 bits of digital output. The registers are located at BASE+13 and BASE+14.

BASE+13 D/O Low Byte Data (Write)

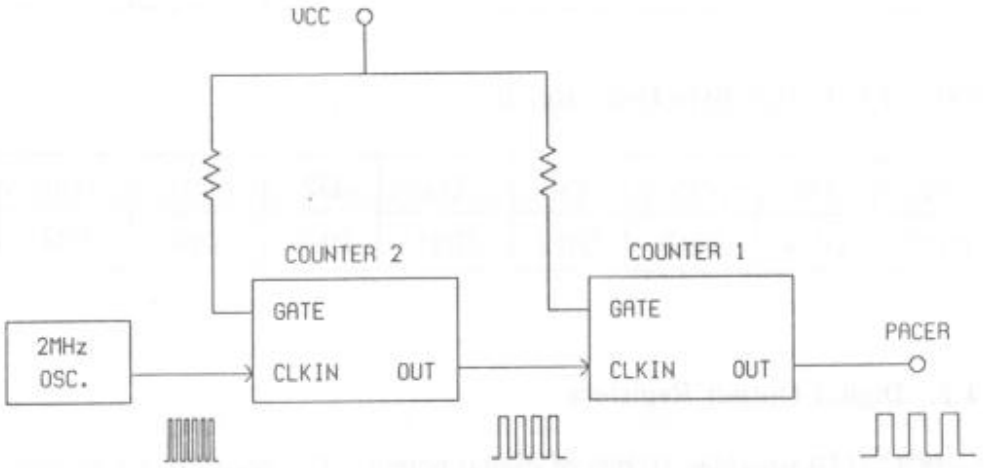
D7	D6	D5	D4	D3	D2	D1	D0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

BASE+14 D/O High Byte Data (Write)

D7	D6	D5	D4	D3	D2	D1	D0
DO15	DO14	DO13	DO12	DO11	DO10	DO9	DO8

3.5. Pacer Programming

The PCL-711B uses an Intel 8253, a 16-bit programmable counter/timer, to generate pacer clock. Each Intel 8253 provides three independent counter/timer channels, Counter 0, Counter 1 and Counter 2. The PCL-711B cascades Counter 1 and Counter 2 as a 32-bit frequency divider to support wide range of pacer clock rate, as shown below:



Intel 8253 has six operation modes, from Mode 0 through Mode 5. To generate pacer clock, both Counter 1 and Counter 2 should be programmed as Mode 2 (rate generator mode).

Four I/O ports, from BASE+0 through BASE+3, are used to program the on-board Intel 8253:

BASE+0: Counter 0 (Read/Write)
BASE+1: Counter 1 (Read/Write)
BASE+2: Counter 2 (Read/Write)
BASE+3: Counter Control (Write only)

Please refer to the following steps to set desired pacer clock rate:

Step 1: Write '74H' to BASE+3 to make Counter 1 work at Mode 2.

Step 2: Write an appropriate data (16-bit data, ranging from 2 to 65535) to BASE+1 to set Counter 1's divisor constant C1. Since C1 is a 16-bit data, you have to first write the low byte of C1 to BASE+1, then write the high byte of C1 to BASE+1.

Step 3: Write 'B4H' to BASE+3 to make Counter 2 work at Mode 2.

Step 4: Write an appropriate data (16-bit data, ranging from 2 to 65535) to BASE+1 to set Counter 1's divisor constant C2. Since C2 is a 16-bit data, you have to first write the low byte of C2 to BASE+2, then write the high byte of C2 to BASE+2.

The pacer rate is determined by the following formula:

Pacer rate = (2 MHz)/(C1*C2)

In the following example (written in BASIC), C1 is set as 40 and C2 is set as 10, thus the pacer rate will be 5 KHz (5KHz=2MHz/(40*10)).

```
.
.
.
500  OUT (BASE+3,&H74)      ' Set Counter 1 as Mode 2
510  OUT (BASE+1,40)        ' write low byte of C1
520  OUT (BASE+1,0)         ' write high byte of C1
530  OUT (BASE+3,&HB4)      ' Set Counter 2 as Mode 2
540  OUT (BASE+2,10)        ' write low byte of C2
550  OUT (BASE+2,0)         ' write high byte of C2
.
.
```

NOTE 1: Counter 0 is reserved to future developement.

NOTE 2: For more detailed information about Intel 8253's register formats, please refer to Appendix B.

Anexo 3

A seguir é apresentado o código fonte do programa de simulação. As frases entre parênteses e aspas não fazem parte deste código, são apenas declarações explicativas para entendimento do mesmo.

a) Código fonte da tela (form) principal:

```
unit principal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls, ToolWin, Menus, ExtDlgs, Grids, Outline,
  DirOutln, FileCtrl;
("Declaração de variáveis e métodos")
type
  TForm1 = class(TForm)
    label_titulo: TLabel;
    botao_opcao1: TButton;
    botao_opcao2: TButton;
    label_opcao1: TLabel;
    label_opcao2: TLabel;
    grupo_opcao1: TGroupBox;
    grupo_opcao2: TGroupBox;
    texto_tdesc: TEdit;
    label_tdesc: TLabel;
    label_tdesa: TLabel;
    texto_tdesa: TEdit;
    botao_opcao1_ok: TButton;
```

label_wref1: TLabel;
texto_wref1: TEdit;
label_wref2: TLabel;
texto_wref2: TEdit;
texto_wref3: TEdit;
label_wref3: TLabel;
texto_wref4: TEdit;
texto_wref5: TEdit;
label_wref4: TLabel;
label_wref5: TLabel;
botao_opcao2_ok: TButton;
texto_wref: TEdit;
label_wref: TLabel;
grupo_dados: TGroupBox;
texto_Vrf: TEdit;
texto_Vrt: TEdit;
texto_Vrv: TEdit;
texto_k: TEdit;
texto_k1: TEdit;
texto_Trf: TEdit;
texto_Trt: TEdit;
texto_Trv: TEdit;
texto_Te: TEdit;
texto_Tr: TEdit;
texto_Tm: TEdit;
texto_w1: TEdit;
label_Vrf: TLabel;
label_Vrt: TLabel;
label_Vrv: TLabel;
label_k: TLabel;
label_k1: TLabel;

```
label_Trf: TLabel;
label_Trt: TLabel;
label_Trv: TLabel;
label_Te: TLabel;
label_Tr: TLabel;
label_Tm: TLabel;
label_w1: TLabel;
texto_salvdado: TEdit;
botao_salvdado: TButton;
grupo_carga: TGroupBox;
radio_carga1: TRadioButton;
radio_carga2: TRadioButton;
radio_carga3: TRadioButton;
label_carga: TLabel;
botao_simular: TButton;
painel_carga1: TPanel;
texto_torque_max: TEdit;
label_torque_max: TLabel;
label_torque_min: TLabel;
texto_torque_min: TEdit;
label_duracao_torque: TLabel;
texto_duracao_torque: TEdit;
painel_carga2: TPanel;
texto_c1: TEdit;
label_c1: TLabel;
painel_carga3: TPanel;
label_c2: TLabel;
texto_c2: TEdit;
MainMenu1: TMainMenu;
menu_1: TMenuItem;
menu_2: TMenuItem;
```

```
    menu_3: TMenuItem;
    menu_4: TMenuItem;
    menu_5: TMenuItem;
    procedure botao_salvdadoClick(Sender: TObject);
    procedure botao_opcao1_okClick(Sender: TObject);
    procedure botao_opcao2_okClick(Sender: TObject);
    procedure botao_simularClick(Sender: TObject);
    procedure radio_carga1Click(Sender: TObject);
    procedure radio_carga2Click(Sender: TObject);
    procedure radio_carga3Click(Sender: TObject);
    procedure botao_opcao1Click(Sender: TObject);
    procedure botao_opcao2Click(Sender: TObject);
    procedure menu_3Click(Sender: TObject);
    procedure menu_4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure menu_5Click(Sender: TObject);
    procedure menu_2Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    function f_ml(n: integer): double;
  end;
const
max: double = 16350;
pi: double =3.141593;
var
  Form1: TForm1;
  //max: real;
```

```
  j_guarda,i,n,e,x,escolha,j,vez,cont,mudou: integer;
  aux: string;
  e9,wref,a,maior_is2ref,ie9d,ml,Vrv,dt,Trv,s9,ie9a,mdref,soma: double;
  w,p,vml,is2ref,isd,md,isq,imr,ief,ordenada,sd: array [0..16350] of double;
  ie7d,Vrt,Trt,e7,k0,k2,k3,isqref,ie7a,imrref,e8,ie8d,Vrf,Tdesc,Tdesc1: double;
  Tdesc2,Tdesc3,Tdesa,Trf,isdref,ie8a,isaref,isbref,isarefa,sinal,w1:double;
  k1,Te,ma,f_wref,limite,is1ref,isbrefa,is3ref,isa,isb,m,entrada:double;
  k,ws,Tr,wmr,ie3d,a3,ie3a,degrau,Wref0,C1,C2,min_ml,max_ml,Tsub: double;
  Wref1,Wref2,Wref3,Wref4, Tm: double;
  arqX : textFile;
implementation

uses abrir_arquivo;

{$R *.DFM}

procedure TForm1.botao_salvdadoClick(Sender: TObject);
begin
grupo_dados.enabled:=false;
grupo_carga.enabled:=true;
if(wref=10.000000) then wref:=1.000000;
AssignFile(arqX,texto_salvdado.text+'fonte2.txt');
Rewrite(arqX);
if(e=2) then wref:=10.000000;

Trt:=strtofloat(texto_trt.text);
Trv:=strtofloat(texto_trv.text);
Te:=strtofloat(texto_te.text);
Tr:=strtofloat(texto_tr.text);
Tm:=strtofloat(texto_tm.text);
W1:=strtofloat(texto_w1.text);
```

```
Vrf:=strtofloat(texto_vrf.text);
Vrt:=strtofloat(texto_vrt.text);
Vrv:=strtofloat(texto_vrv.text);
K:=strtofloat(texto_k.text);
K1:=strtofloat(texto_k1.text);
Trf:=strtofloat(texto_trf.text);

WriteLn(arqX,Wref);
WriteLn(arqX,Vrf);
WriteLn(arqX,Vrt);
WriteLn(arqX,Vrv);
WriteLn(arqX,K);
WriteLn(arqX,K1);
WriteLn(arqX,Trf);
WriteLn(arqX,Trt);
WriteLn(arqX,Trv);
WriteLn(arqX,Te);
WriteLn(arqX,Tr);
WriteLn(arqX,Tm);
WriteLn(arqX,W1);
CloseFile(arqX);

end;

(“Procedimento para leitura de parâmetros”)
procedure TForm1.botao_opcao1_okClick(Sender: TObject);
var
conv :string;
begin
Tdesc:=strtofloat(texto_tdesc.text);
Tdesa:=strtofloat(texto_tdesa.text);
```

```
Wref:=strtofloat(texto_wref.text);
//grupo_opcao1.color:=cl3DLight;
grupo_opcao1.enabled:=false;
botao_opcao1.enabled:=false;
botao_opcao2.enabled:=false;
grupo_dados.enabled:=true;
end;

(“Procedimento para leitura de parâmetros: figura 7”)
procedure TForm1.botao_opcao2_okClick(Sender: TObject);
begin
Wref0:=strtofloat(texto_wref1.text);
Wref1:=strtofloat(texto_wref2.text);
Wref2:=strtofloat(texto_wref3.text);
Wref3:=strtofloat(texto_wref4.text);
Wref4:=strtofloat(texto_wref5.text);
grupo_opcao2.enabled:=false;
grupo_dados.enabled:=true;
botao_opcao1.enabled:=false;
botao_opcao2.enabled:=false;
end;

(“Procedimento que faz os cálculos da simulação”)
procedure TForm1.botao_simularClick(Sender: TObject);
var
conv:string;
begin
vez:=0;
if radio_carga1.checked= true then n:=1;
if radio_carga2.checked= true then n:=2;
if radio_carga3.checked= true then n:=3;
```

```
min_ml:=strtofloat(texto_torque_min.text);
max_ml:=strtofloat(texto_torque_max.text);
Tsub:=strtofloat(texto_duracao_torque.text);
C1:=strtofloat(texto_c1.text);
C2:=strtofloat(texto_c2.text);
(“Inicialização das variáveis”)
i:=0;
dt:=0.002;
w[i]:=0.0;
sinal:=1;
is2ref[i]:=0.0;
mudou:=0;
ie9a:=0.0;
md[i]:=0.0;
ie7a:=0.0;
imr[i]:=1.0;
ie8a:=0.0;
p[i]:=pi/3.0;
ie3a:=1.0;
soma:=0.0;
ief[i]:=0.0;
ief[1]:=0.0;
escolha:=1;
a:=0;
ordenada[0]:=0;
maior_is2ref:=0;

(“Loop principal que irá fazer os cálculos dos parâmetros de controle do motor durante os 32 segundos de simulação”)
while (i<max-1)do
begin
```

```
    i:=i+1;
//  referência de velocidade, partida com/sem reversão
```
(“Iguala a velocidade de referência ao valor wref estabelecido pela usuário dentro dos limites de tempo pré-determinados”)

```
if(e=1)then
   begin

      if(i*dt<=Tdesc) then f_wref:= wref;
      if((i*dt<Tdesc+Tdesa/2)and(i*dt>Tdesc))  then    f_wref:=  (-2*wref/(2*(Tdesc-
1)+Tdesa))*i*dt + ((wref*2*Tdesc+Tdesa)/(2*(Tdesc-1)+Tdesa));
      if(i*dt>=Tdesc+Tdesa) then   f_wref:=-wref;
   end;

//  referência de velocidade, definir curva

   if(e=2)then
   begin
      if(i*dt<9.30)then    f_wref:= Wref0;
      if((i*dt>=9.30)and(i*dt<15.15))then f_wref:= Wref1;
      if((i*dt>=15.15)and(i*dt<21.00))then      f_wref:= Wref2;
      if((i*dt>=21.00)and(i*dt<26.85))then       f_wref:= Wref3;
      if((i*dt>=26.85)and(i*dt<32.7))then       f_wref:= Wref4;

   end;
```
(“Implementa o bloco regulador de velocidade, (e) representa o valor da entrada e (s) o da saída instantâneos – bloco 2”)
```
  e9:=(f_wref - w[i-1]);                  //regulador de velocidade
  ie9d:=dt*e9*(Vrv/Trv) + ie9a;
```

```
  s9:=e9*Vrv+ie9d;
    if((s9*imr[i-1]<=-3.0)or(s9*imr[i-1]>=3.0))then  ie9d:=ie9a;  // congela integrador
- limitador de OV
  ie9a:=ie9d;

("Limita o valor de saída do bloco regulador de velocidade dentro de certos valores
– bloco 10")
  if((s9*imr[i-1]>-2.5)and(s9*imr[i-1]<2.5))then  mdref:=s9*imr[i-1];  // bloco limitador
  if(s9*imr[i-1]<=-2.5)then  mdref:=-2.5;
  if(s9*imr[i-1]>=2.5)then  mdref:=2.5;

("Implementa o bloco regulador de torque, onde a saída é o valor da corrente isq
que deverá ser imposta ao motor – bloco 6")
  e7:=mdref-md[i-1];                    // regulador de torque
  ie7d:=dt*(Vrt/Trt)*e7+ie7a;
  isqref:=e7*Vrt+ie7d;
  ie7a:=ie7d;

("Bloco de enfraquecimento do campo – bloco 3")
  imrref:=1.0;              // field weakening (simplificação)

("Implementa o bloco regulador de fluxo, a saída é o valor da corrente isq imposta
ao motor – bloco 7")
  e8:=(imrref-imr[i-1]);
  ie8d:=dt*e8*(Vrf/Trf)+ie8a;           // regulador de fluxo
  isdref:=e8*Vrf+ie8d;
  ie8a:=ie8d;


("Implementa a transformação d-q para ab – bloco 5 demodulador")
//bloco demodulador
```

```
  isaref:=isdref*cos(p[i-1]) - isqref*sin(p[i-1]);
  isbref:=isqref*cos(p[i-1]) + isdref*sin(p[i-1]);

("Bloco de atraso – bloco 8")
  isarefa:=isaref-w1*Te*isbref;                    // bloco de atraso
  isbrefa:=isbref+w1*Te*isaref;

("Realiza a transformação das correntes bifásicas ab para as correntes trifásicas
do estator -  bloco 5  2 -> 3")
  is1ref:=(2.0/3.0)*isarefa;
  is2ref[i]:=(1.0/sqrt(3.0))*isbrefa-(1.0/3.0)*isarefa;
  is3ref:=(-1.0/3.0)*isarefa-(1.0/sqrt(3.0))*isbrefa;

 ("Realiza a leitura das correntes. No caso desta simulação os valores de saída
dos cálculos anteriores são utilizados como correntes lidas. A seguir é
implementado o bloco que faz a transformação das correntes trifásicas do estator
para as correntes bifásicas ab - bloco 4  3 -> 2")
      isa:=(3.0/2.0)*is1ref;              //leitura de correntes
  isb:=(sqrt(3.0)/2.0)*(is2ref[i]-is3ref);

("Realiza a transformação ab para d-q - Bloco 4 modulador")
  isd[i]:=isa*cos(p[i-1]) + isb*sin(p[i-1]);
  isq[i]:=isb*cos(p[i-1]) - isa*sin(p[i-1]);

("Implementa o bloco para o cálculo de imr – bloco 9")
  a3:=(isd[i]-imr[i-1])*(1.0/Tr);
  ie3d:=dt*a3+ie3a;
  ie3a:=ie3d;
  imr[i]:=ie3a;
("Calcula o torque elétrico – bloco de multiplicação")
md[i]:=k*isq[i]*imr[i];
```

```
  if ((imr[i]=0.0)and(vez=0))then
  begin
      w[i]:=0.0;
      p[i]:=p[0];
      md[i]:=md[0];
      ie9a:=0.0;
      vez:=1;
  end
  else
  begin

      ws:=(isq[i]/(Tr*imr[i]))/k1;
   (“Como é uma simulação então calcula-se a velocidade w ao invés de se fazer a medição”)
      vml[i-1]:=f_ml(n);
      ma:=md[i]-vml[i-1];
      w[i]:=w[i-1]+(dt/Tm)*ma;
//trecho que evita  pequeno aumento de velocidade no instante da reversão
      if(((md[i]-md[i-1])>0.008)or((md[i]-md[i-1])<-0.008))then  w[i]:=w[i-1];
  (“Cálculo da fase da corrente de magnetização – Bloco 1”)
      wmr:=k1*(ws+w[i]);
      p[i]:=p[i-1] + dt*wmr;
      if(p[i]>2*pi)then  p[i]:=p[i]-2*pi;
      if(p[i]<-2*pi)then  p[i]:=p[i]+2*pi;

  end;
end;
// valor eficaz da corrente de alimentação

i:=0;
```

```
j:=0;
x:=0; // contador lógico
(“Depois de terminada a simulação os códigos a seguir irão gravar os parâmetros simulados em arquivos para posterior plotagem”)
while(i<max-1)do
begin
    while(is2ref[i]>=0)do
    begin              // aprox 1638 picos
        if(maior_is2ref<is2ref[i])then maior_is2ref:=is2ref[i];
        i:=i+1;
    end;
    a:=a+maior_is2ref;
    maior_is2ref:=0;
    x:=x+1;
    if(x=2)then
    begin
          ief[j]:=a;
        j:=j+1;
        a:=0;
        maior_is2ref:=0;
        x:=0;
    end;
    j_guarda:=j;
    i:=i+1;
    end;
cont:=7;
while(cont>0)do
begin
    if(cont=7)then
    begin
        i:=0;
```

```
    for j:=0 to 544do
    begin
        ordenada[j]:=imr[i];
        i:=i+30;
    end;
    if(j=545)then
    begin
        AssignFile(arqX,(texto_salvdado.text + 'da_imr.txt'));
        Rewrite(arqX);
        for i:=0 to 544 do WriteLn(arqX,ordenada[i]);
        CloseFile(arqX);
    end;
end;
if(cont=6)then
begin
    i:=0;
    for j:=0 to 544do
    begin
        ordenada[j]:=w[i];
        i:=i+30;
    end;
    if(j=545)then
    begin
        AssignFile(arqX,texto_salvdado.text+'da_w.txt');
        Rewrite(arqX);
        for i:=0 to 544 do WriteLn(arqX,ordenada[i]);
        CloseFile(arqX);
     end;
end;
if(cont=5)then
begin
```

```
    i:=0;
    for j:=0 to 544do
    begin
        ordenada[j]:=md[i];
        i:=i+30;
    end;
    if(j=545)then
    begin
        AssignFile(arqX,texto_salvdado.text+'da_md.txt');
        Rewrite(arqX);
        for i:=0 to 544 do WriteLn(arqX,ordenada[i]);
        CloseFile(arqX);
    end;
end;
if(cont=4)then
begin
    i:=0;
    for j:=0 to 544do
    begin
        ordenada[j]:=vml[i];
        i:=i+30;
    end;
    if(j=545)then
    begin
        AssignFile(arqX,texto_salvdado.text+'da_vml.txt');
        Rewrite(arqX);
        for i:=0 to 544 do WriteLn(arqX,ordenada[i]);
        CloseFile(arqX);
    end;
end;
if(cont=3)then
```

```
begin
    i:=0;
    for j:=0 to 544do
    begin
        ordenada[j]:=ief[i];
        if(ordenada[j]=0) then ordenada[j]:=ordenada[j-1];
        conv:=floattostr(i+int(j_guarda/500));
        i:=strtoint(conv);
    end;
    if(j=545)then
    begin
        AssignFile(arqX,texto_salvdado.text+'da_ief.txt');
        Rewrite(arqX);
        for i:=0 to 544 do WriteLn(arqX,ordenada[i]);
        CloseFile(arqX);
    end;
end;
if(cont=2)then
begin
    i:=0;
    for j:=0 to 544do
    begin
        ordenada[j]:=isd[i];
        i:=i+30;
    end;
    if(j=545)then
    begin
        AssignFile(arqX,texto_salvdado.text+'da_isd.txt');
        Rewrite(arqX);
        for i:=0 to 544 do WriteLn(arqX,ordenada[i]);
        CloseFile(arqX);
```

```
        end;
    end;
    if(cont=1)then
    begin
        i:=0;
        for j:=0 to 544do
        begin
            ordenada[j]:=isq[i];
            i:=i+30;
        end;
        if(j=545)then
        begin
            AssignFile(arqX,texto_salvdado.text+'da_isq.txt');
            Rewrite(arqX);
            for i:=0 to 544 do WriteLn(arqX,ordenada[i]);
            CloseFile(arqX);
        end;
    end;
    cont:=cont-1;
end;


end;
```

(“Procedimentos para o correto funcionamento do programa durante a escolha das opções da carga”)

```
procedure TForm1.radio_carga1Click(Sender: TObject);
begin
painel_carga1.visible:=true;
painel_carga2.visible:=false;
painel_carga3.visible:=false;
```

```
end;

procedure TForm1.radio_carga2Click(Sender: TObject);
begin
painel_carga1.visible:=false;
painel_carga2.visible:=true;
painel_carga3.visible:=false;
end;

procedure TForm1.radio_carga3Click(Sender: TObject);
begin
painel_carga1.visible:=false;
painel_carga2.visible:=false;
painel_carga3.visible:=true;
end;

("Função para a análise da carga de acordo com a opção escolhida pelo usuário")
function TForm1.f_ml(n: integer): double;
begin

if n=1 then
begin
    j:=1;
    escolha:=0;
    if((i*dt)<=Tsub) then ml:=min_ml
    else
    ml:=max_ml;
    if(w[i-1]<0)then sinal:=-1  // o torque vira junto com a velocidade?
    else
    sinal:=1;
```

```
    ml:=ml*sinal;  // torque de carga
    result:=ml;
end;

if n=2 then
begin
    j:=2;
    escolha:=0;
    ml:=C1*w[i-1];
    result:= ml;
end;

if n=3 then
begin
    j:=3;
    if(w[i-1]<0)then sinal:=-1;
    ml:=sinal*C2*w[i-1]*w[i-1];
    result:=ml;
end;
result:=ml;

end;
(“Continuação de procedimentos para o funcionamento do programa”)
procedure TForm1.botao_opcao1Click(Sender: TObject);
var
ij:integer;
begin
botao_opcao1.Font.Color:=clRed;
e:=1;
grupo_opcao1.enabled:=true;
grupo_opcao1.visible:=true;
```

```
grupo_opcao2.enabled:=false;
grupo_opcao2.visible:=false;
{for ij:=88 to 300 do
begin
grupo_opcao1.top:=ij;
grupo_opcao1.left:=ij+28;
end;}
end;

procedure TForm1.botao_opcao2Click(Sender: TObject);
begin
e:=2;
grupo_opcao1.enabled:=false;
grupo_opcao1.visible:=false;
grupo_opcao2.enabled:=true;
grupo_opcao2.visible:=true;
end;



procedure TForm1.menu_3Click(Sender: TObject);
begin
abrir:=1;
form_abrir.visible:=true;
end;

procedure TForm1.menu_4Click(Sender: TObject);
begin
abrir:=2;
form_abrir.visible:=true;
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
decimalseparator:='.';
end;

procedure TForm1.menu_5Click(Sender: TObject);
begin
close;
end;

procedure TForm1.menu_2Click(Sender: TObject);
begin
botao_opcao1.enabled:=true;
botao_opcao2.enabled:=true;
end;

end.
```

a) Código fonte da tela (form) onde são mostrados os gráficos resultantes:

Os códigos do form a seguir irão fazer as leituras dos parâmetros gravados nos arquivos e depois irão plotá-los de acordo com a escolha do usuário

```
unit graficos;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, TeEngine, Series, ExtCtrls, TeeProcs, Chart;
```

```
type
  Tform_graficos = class(TForm)
    chart_graficos: TChart;
    radio_torqueel: TRadioButton;
    radio_torquecarga: TRadioButton;
    radio_w: TRadioButton;
    radio_correntefluxo: TRadioButton;
    radio_correntetorque: TRadioButton;
    radio_magnetizacao: TRadioButton;
    radio_correntealimentacao: TRadioButton;
    Series1: TFastLineSeries;
    procedure radio_torqueelClick(Sender: TObject);
    procedure radio_torquecargaClick(Sender: TObject);
    procedure radio_wClick(Sender: TObject);
    procedure radio_correntefluxoClick(Sender: TObject);
    procedure radio_correntetorqueClick(Sender: TObject);
    procedure radio_magnetizacaoClick(Sender: TObject);
    procedure radio_correntealimentacaoClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  form_graficos: Tform_graficos;
  arqX : textFile;
  X: array [0..16350] of double;
implementation
```

```
uses abrir_arquivo;

{$R *.DFM}

procedure Tform_graficos.radio_torqueelClick(Sender: TObject);
var
i: integer;
n_amostras: integer;
aux: string;
begin
chart_graficos.Series[0].clear;
i:=0;
AssignFile(arqX,form_abrir.simulacao+'da_md.txt');
Reset(arqX);
ReadLn(arqX,aux);
repeat
     ReadLn(arqX,aux);
     X[i]:=strtofloat(aux);
     i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i-1;

for i:=0 to n_amostras do
begin
    chart_graficos.Series[0].AddXY ((i*30*0.002),X[i],'',clTeeColor);
end;
end;
procedure Tform_graficos.radio_torquecargaClick(Sender: TObject);
var
i: integer;
```

```
n_amostras: integer;
aux: string;
begin
chart_graficos.Series[0].clear;
i:=0;
AssignFile(arqX,form_abrir.simulacao+'da_vml.txt');
Reset(arqX);
ReadLn(arqX,aux);
repeat
     ReadLn(arqX,aux);
     X[i]:=strtofloat(aux);
     i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i-1;

for i:=0 to n_amostras do
begin
    chart_graficos.Series[0].AddXY ((i*30*0.002),X[i],'',clTeeColor);
end;
end;

procedure Tform_graficos.radio_wClick(Sender: TObject);

var
i: integer;
n_amostras: integer;
aux: string;
begin
chart_graficos.Series[0].clear;
i:=0;
```

```
AssignFile(arqX,form_abrir.simulacao+'da_w.txt');
Reset(arqX);
ReadLn(arqX,aux);
repeat
     ReadLn(arqX,aux);
     X[i]:=strtofloat(aux);
     i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i-1;

for i:=0 to n_amostras do
begin
    chart_graficos.Series[0].AddXY ((i*30*0.002),X[i],'',clTeeColor);
end;
end;

procedure Tform_graficos.radio_correntefluxoClick(Sender: TObject);

var
i: integer;
n_amostras: integer;
aux: string;
begin
chart_graficos.Series[0].clear;
i:=0;
AssignFile(arqX,form_abrir.simulacao+'da_isd.txt');
Reset(arqX);
ReadLn(arqX,aux);
repeat
     ReadLn(arqX,aux);
```

```
    X[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i-1;

for i:=0 to n_amostras do
begin
   chart_graficos.Series[0].AddXY ((i*30*0.002),X[i],'',clTeeColor);
end;
end;

procedure Tform_graficos.radio_correntetorqueClick(Sender: TObject);

var
i: integer;
n_amostras: integer;
aux: string;
begin
chart_graficos.Series[0].clear;
i:=0;
AssignFile(arqX,form_abrir.simulacao+'da_isq.txt');
Reset(arqX);
ReadLn(arqX,aux);
repeat
    ReadLn(arqX,aux);
    X[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i-1;
```

```
for i:=0 to n_amostras do
begin
    chart_graficos.Series[0].AddXY ((i*30*0.002),X[i],'',clTeeColor);
end;
end;

procedure Tform_graficos.radio_magnetizacaoClick(Sender: TObject);

var
i: integer;
n_amostras: integer;
aux: string;
begin
chart_graficos.Series[0].clear;
i:=0;
AssignFile(arqX,form_abrir.simulacao+'da_imr.txt');
Reset(arqX);
ReadLn(arqX,aux);
repeat
     ReadLn(arqX,aux);
     X[i]:=strtofloat(aux);
     i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i-1;

for i:=0 to n_amostras do
begin
    chart_graficos.Series[0].AddXY ((i*30*0.002),X[i],'',clTeeColor);
end;
```

```
end;

procedure Tform_graficos.radio_correntealimentacaoClick(Sender: TObject);

var
i: integer;
n_amostras: integer;
aux: string;
begin
chart_graficos.Series[0].clear;
i:=0;
AssignFile(arqX,form_abrir.simulacao+'da_ief.txt');
Reset(arqX);
ReadLn(arqX,aux);
repeat
     ReadLn(arqX,aux);
     X[i]:=strtofloat(aux);
     i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i-1;

for i:=0 to n_amostras do
begin
    chart_graficos.Series[0].AddXY ((i*30*0.002),X[i],'',clTeeColor);
end;
end;

procedure Tform_graficos.FormCreate(Sender: TObject);
begin
```

```
end;

end.
```

c)Código fonte do form para escolha do diretório onde a simulação está armazenada:

```
unit abrir_arquivo;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  Tform_abrir = class(TForm)
    botao_abridado: TButton;
    texto_abridado: TEdit;
    label_caminho: TLabel;
    procedure botao_abridadoClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    simulacao: string;
    { Public declarations }
  end;

var
```

```
  form_abrir: Tform_abrir;
  abrir: integer;
  arqX : textFile;
implementation

uses principal, graficos;

{$R *.DFM}

procedure Tform_abrir.botao_abridadoClick(Sender: TObject);
var
aux: string;
begin
if abrir =1 then
begin
    AssignFile(arqX,texto_abridado.text+'fonte2.txt');
    Reset(arqX);
    ReadLn(arqX,aux);
    form1.texto_wref.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_vrf.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_vrt.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_vrv.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_k.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_k1.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_trf.text:=aux;
```

```
    ReadLn(arqX,aux);
    form1.texto_trt.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_trv.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_te.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_tr.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_tm.text:=aux;
    ReadLn(arqX,aux);
    form1.texto_w1.text:=aux;
    CloseFile(arqX);
    form_abrir.visible:=false
end;
if abrir=2 then
begin

    simulacao:=texto_abridado.text;
    form_abrir.visible:=false;
    form_graficos.visible:=true;
end;

    end;

procedure Tform_abrir.FormCreate(Sender: TObject);
begin

end;

end.
```

Anexo 4

A seguir é apresentado o código fonte do programa de controle em tempo real. As frases entre parênteses e aspas não fazem parte deste código, são apenas declarações explicativas para entendimento do mesmo.

a)Código fonte do form principal

```
unit principal;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls, ToolWin, Menus, ExtDlgs, Grids, Outline,FileCtrl;
("Declaração dos objetos e procedimentos")
type
  Tprincipal_form = class(TForm)
    label_titulo: TLabel;
    grupo_dados: TGroupBox;
    texto_Vrf: TEdit;
    texto_Vrt: TEdit;
    texto_Vrv: TEdit;
    texto_k: TEdit;
    texto_k1: TEdit;
    texto_Trf: TEdit;
    texto_Trt: TEdit;
    texto_Trv: TEdit;
    texto_Te: TEdit;
    texto_Tr: TEdit;
    texto_Tm: TEdit;
```

texto_w1: TEdit;
label_Vrf: TLabel;
label_Vrt: TLabel;
label_Vrv: TLabel;
label_k: TLabel;
label_k1: TLabel;
label_Trf: TLabel;
label_Trt: TLabel;
label_Trv: TLabel;
label_Te: TLabel;
label_Tr: TLabel;
label_Tm: TLabel;
label_w1: TLabel;
Button1: TButton;
Button2: TButton;
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
Edit4: TEdit;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
Label1: TLabel;
Label3: TLabel;
Label4: TLabel;
Label2: TLabel;
Edit5: TEdit;
MainMenu1: TMainMenu;
Iniciarcontrole1: TMenuItem;
raargrfico1: TMenuItem;

```
    Sair1: TMenuItem;
    procedure FormCreate(Sender: TObject);
    function le_ad(canal: integer): double;
    procedure pwm(piref,pireal: double; var ptsup,ptinf: integer;
flags,peso_sup,peso_inf: word; var saida_atual,verificar:word);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Iniciarcontrole1Click(Sender: TObject);
    procedure raargrfico1Click(Sender: TObject);
    procedure Sair1Click(Sender: TObject);
  private
    { Private declarations }
  public
    op: integer;

    { Public declarations }

  end;
("Declaração das variáveis públicas")
const
max: double = 16350;
pi: double =3.141593;
var
  principal_form: Tprincipal_form;
  i,escolha,simular: integer;
  aux: string;
  e9,wref,a,ie9d,ml,Vrv,dt,Trv,s9,ie9a,mdref,soma: double;
  p,vml,md,imr,sd: array [0..3] of double;
  ie7d,Vrt,Trt,e7,isqref,ie7a,imrref,e8,ie8d,Vrf,Tdesc,Tdesc1: double;
  Trf,isdref,ie8a,isaref,isbref,isarefa,sinal,w1:double;
  k1,Te,ma,limite,isbrefa,isa,isb,m,entrada:double;
```

```
  k,ws,Tr,wmr,ie3d,a3,ie3a,degrau,min_ml,max_ml: double;
  Tm,isd,isq,pu_w,pu_corrente,pu_wref: double;
  arqX : textFile;
  EndBase,saida,cuidado1,cuidado2,cuidado3: word;
  Vt,is1real,is2real,is3real,is1ref,is2ref,is3ref: double;
  t1,t3,t5,t4,t6,t2,morto:integer;
  w,regulador: array [0..3] of double;

implementation

Uses grafico,diretorio;

{$R *.DFM}
function inportb(EndPorta: Integer): BYTE stdcall; external 'inpout32.DLL' name
'Inp32';
procedure outportb(EndPorta: integer; Valor:BYTE); stdcall; external 'inpout32.DLL'
name 'Out32';

(“Procedimento de configuração da placa de aquisição de dados na inicialização
do programa”)
procedure Tprincipal_form.FormCreate(Sender: TObject);
begin
decimalseparator:='.';
EndBase:=$220; // Endereco base da placa de aquisicao.
outportb ((EndBase+11),1);// Habilita conversao por software.
outportb((EndBase+10),10); // Selecao do canal 0 do A/D.
outportb((EndBase+9),0); // Ganho unitario.
simular:=0;
end;
```

```
("Função para leitura do conversor A/D da placa de aquisição de dados")
function Tprincipal_form.le_ad(canal: integer):double;
var
dado,dadoMSB,dadoLSB: word;
begin
outportb((EndBase+10),canal); // Selecao do canal do A/D.
outportb((EndBase+9),0); // Ganho unitario.
outportb((EndBase+12),0); // Inicia conversao do A/D.
dado:=0;
repeat
    dadoMSB:=inportb(EndBase+5); //Ler bits MSB do A/D.
    dado:=dadoMSB and $10; // Flag de conversao.
until (dado=0); // Teste de conversao.
dadoLSB:=inportb(EndBase+4);
dado:= (dadoMSB shl 8) + dadoLSB; // Dado em decimal.
Vt:= 10.0 * dado / 4095.0 - 5.0; // Valor em tensao. */
le_ad:=Vt;
end;
("Procedimento para a implementação da histerese de corrente e atualização das saídas")
procedure Tprincipal_form.pwm(piref,pireal: double; var ptsup,ptinf: integer; flags,peso_sup,peso_inf: word; var saida_atual,verificar:word);
var        //declaração de variaveis locais
er,perro:double;
ptsup_ant,ptinf_ant: integer;
aux: word;
begin
ptsup_ant:=ptsup;     //salva a situação dos IGBT´s
ptinf_ant:=ptinf;
perro:=0.01;          // erro de 1%
er:=piref-pireal;     // cálculo da diferença entre a corrente real e a de referência
```

```
if (piref=0) or (abs(er)<0.003) then er:=0   //cálculo da porcentagem de erro entre
else                                         //a corrente real e a de referência
      er:=er/abs(piref);
if (abs(er)>perro) and (piref>pireal) then    //se o erro for maior que 1% e a
begin                                         //corrente de referência for maior
      ptsup:=1;                               //que a real então liga-se o IGBT
      ptinf:=0;                               //superior e desliga-se o inferior
end;
if (abs(er)>perro) and (piref<pireal) then    //se o erro for maior que 1% e a
begin                                         //corrente de referência for menor
       ptsup:=0;                              //que a real então liga-se o IGBT
       ptinf:=1;                              //inferior e desliga-se o superior
end;

if (ptsup<>ptsup_ant) or (ptinf<>ptinf_ant) then
begin
        //se for necessário fazer a mudança do estado dos IGBT's, com a inserção
        //do tempo morto
      saida_atual:=saida_atual and flags;
      outportb(EndBase+13,saida_atual);
      for aux:=0 to morto do;
      if ptsup=1 then saida_atual:=saida_atual+peso_sup;
      if ptinf=1 then saida_atual:=saida_atual+peso_inf;
      outportb(EndBase+13,saida_atual);
      verificar:=0;
end
else
      verificar:=verificar+1;
end;
("Procedimento que inicia a função de controle do motor")
procedure Tprincipal_form.Button1Click(Sender: TObject);
```

```
var       //Declaração das variáveis locais
i,j,vez,faz_nada,transfere,contar,contar2: integer;
vetimr,vetis1real,vetis2real,vetis3real,vetwref,vetw,vetisd,vetisq: array [0..15100] of
double;
JI,T,Ta: double;
X1,X2,X0: word;
begin
//Inicialização das variáveis
Button1.enabled:=false;
Button2.Caption:='Parar';
Trt:=strtofloat(texto_trt.text);
Trv:=strtofloat(texto_trv.text);
Te:=strtofloat(texto_te.text);
Tr:=strtofloat(texto_tr.text);
Tm:=strtofloat(texto_tm.text);
W1:=strtofloat(texto_w1.text);
Vrf:=strtofloat(texto_vrf.text);
Vrt:=strtofloat(texto_vrt.text);
Vrv:=strtofloat(texto_vrv.text);
K:=strtofloat(texto_k.text);
K1:=strtofloat(texto_k1.text);
Trf:=strtofloat(texto_trf.text);
simular:=1;
cuidado1:=0;
cuidado2:=0;
cuidado3:=0;
j:=0;
i:=0;
dt:=0.000164;
w[0]:=0.0;
sinal:=1;
```

```
vez:=0;
ie9a:=0.0;
md[0]:=0.0;
ie7a:=0.0;
imr[0]:=1.0;
ie8a:=0.0;
p[0]:=pi/3.0;
ie3a:=1.0;
soma:=0.0;
escolha:=1;
a:=0;
is1real:=le_ad(0);
is2real:=le_ad(1);
is3real:=le_ad(2);
w[0]:=le_ad(3);
wref:=le_ad(4);
transfere:=0;
i:=1;
pu_w:=2.65;
pu_corrente:=2.52;
pu_wref:=5;
morto:=2;
saida:=0;
t1:=0;
t3:=0;
t5:=0;
t4:=0;
t6:=0;
t2:=0;
contar:=0;
contar2:=0;
```

```
("Início do loop de controle que implementa a malha proposta no trabalho")
repeat
  e9:=(wref - w[i-1]);                    //regulador de velocidade
  if abs(e9)<0.003 then e9:=0;
  ie9d:=dt*e9*(Vrv/Trv) + ie9a;
  s9:=e9*Vrv+ie9d;

  if((s9*imr[i-1]<=-3)or(s9*imr[i-1]>=3))then    ie9d:=ie9a;    // congela integrador
//limitador de OV
  ie9a:=ie9d;

  if((s9*imr[i-1]>-2.5)and(s9*imr[i-1]<2.5))then  mdref:=s9*imr[i-1];  // bloco limitador
  if(s9*imr[i-1]<=-2.5)then  mdref:=-2.5;
  if(s9*imr[i-1]>=2.5)then  mdref:=2.5;
  e7:=mdref-md[i-1];                          // regulador de torque
  ie7d:=dt*(Vrt/Trt)*e7+ie7a;
  isqref:=e7*Vrt+ie7d;
  ie7a:=ie7d;

  imrref:=1.0;                  // field weakening (simplificação)

  e8:=(imrref-imr[i-1]);        // regulador de fluxo
  ie8d:=dt*e8*(Vrf/Trf)+ie8a;
  isdref:=e8*Vrf+ie8d;
  ie8a:=ie8d;

  //bloco demodulador
  isaref:=isdref*cos(p[i-1])-isqref*sin(p[i-1]);
  isbref:=isqref*cos(p[i-1])+isdref*sin(p[i-1]);

  isarefa:=isaref-w1*Te*isbref;                          // bloco de atraso
```

```
isbrefa:=isbref+w1*Te*isaref;

is1ref:=(2.0/3.0)*isarefa;   //continuação bloco demodulador
is2ref:=(1.0/sqrt(3.0))*isbrefa-(1.0/3.0)*isarefa;
is3ref:=(-1.0/3.0)*isarefa-(1.0/sqrt(3.0))*isbrefa;
if is1ref>=2.5 then is1ref:=2.5;     //Congela o valor das correntes de referência
if is1ref<=-2.5 then is1ref:=-2.5;   //do estator no máximo 2.5 P.U.
if is2ref>=2.5 then is2ref:=2.5;
if is2ref<=-2.5 then is2ref:=-2.5;
if is3ref>=2.5 then is3ref:=2.5;
if is3ref<=-2.5 then is3ref:=-2.5;
is1real:=le_ad(0);     //le as correntes reais do motor
is2real:=le_ad(1);
is3real:=le_ad(2);
w[i]:=le_ad(3);
wref:=le_ad(4);        //le a velocidade real do motor

is1real:=is1real/pu_corrente;    //normaliza as correntes
is2real:=is2real/pu_corrente;
is3real:=is3real/pu_corrente;

if wref>0 then
begin
    pu_w:=2.7;
    pu_wref:=4.8;
end
else
begin
    pu_w:=2.65;
    pu_wref:=5;
end;
```

```
  w[i]:=w[i]/pu_w;              //normaliza a velocidade
  wref:=wref/pu_wref;

  pwm(is1ref,is1real,t1,t4,60,1,2,saida,cuidado1);       //atualização pwm
  pwm(is2ref,is2real,t3,t6,51,4,8,saida,cuidado2);
  pwm(is3ref,is3real,t5,t2,15,16,32,saida,cuidado3);

//bloco modulador
  isa:=(3.0/2.0)*is1real;
  isb:=(sqrt(3.0)/2.0)*(is2real-is3real);

  isd:=isa*cos(p[i-1]) + isb*sin(p[i-1]);
  isq:=isb*cos(p[i-1]) - isa*sin(p[i-1]);

  a3:=(isd-imr[i-1])*(1.0/Tr);
  ie3d:=dt*a3+ie3a;
  ie3a:=ie3d;
  imr[i]:=ie3a;   //estimação de imr
  md[i]:=k*isq*imr[i];

  ws:=(isq/(Tr*imr[i]))/k1;

  wmr:=k1*(ws+w[i]);
  p[i]:=p[i-1] + dt*wmr;
  if(p[i]>2*pi)then  p[i]:=p[i]-2*pi;
  if(p[i]<-2*pi)then  p[i]:=p[i]+2*pi;      //estimação do argumento de imr

  //salva os valores necessários em vetores
  if (contar<15000) and (contar2=11) then
  begin
```

```
  vetimr[contar]:=imr[i];
  vetis1real[contar]:=is1real;
  vetis2real[contar]:=is2real;
  vetis3real[contar]:=is3real;
  vetwref[contar]:=wref;
  vetw[contar]:=w[i];
  vetisd[contar]:=isd;
  vetisq[contar]:=isq;

  contar:=contar+1;
  contar2:=0;
  end;

  contar2:=contar2+1;
  //atualização dos valores anteriores para atuais
  w[i-1]:=w[i];
  imr[i-1]:=imr[i];
  md[i-1]:=md[i];
  p[i-1]:=p[i];
Application.ProcessMessages;
until simular=0;  //fim do loop de controle

(“Funções para armazenar os vetores salvos em arquivos de texto”)
outportb(EndBase+13,0);
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\imr.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
begin
      WriteLn(arqX,vetimr[faz_nada]);
end;
CloseFile(arqX);
```

```
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\is1real.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
begin
      WriteLn(arqX,vetis1real[faz_nada]);
end;
CloseFile(arqX);
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\is2real.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
begin
      WriteLn(arqX,vetis2real[faz_nada]);
end;
CloseFile(arqX);
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\is3real.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
begin
      WriteLn(arqX,vetis3real[faz_nada]);
end;
CloseFile(arqX);
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\wref.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
begin
      WriteLn(arqX,vetwref[faz_nada]);
end;
CloseFile(arqX);
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\w.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
```

```
begin
      WriteLn(arqX,vetw[faz_nada]);
end;
CloseFile(arqX);

AssignFile(arqX,diretorio_form.texto_salvdado.text+'\isd.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
begin
      WriteLn(arqX,vetisd[faz_nada]);
end;
CloseFile(arqX);
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\isq.txt');
Rewrite(arqX);
for faz_nada:=0 to contar do
begin
      WriteLn(arqX,vetisq[faz_nada]);
end;
CloseFile(arqX);
grafico_form.Visible:=true;
grafico_form.button1.visible:=true;

end;

procedure Tprincipal_form.Button2Click(Sender: TObject);
begin
outportb(EndBase+13,0);
simular:=0;
Button2.Caption:='Parado';
end;
```

```
procedure Tprincipal_form.Iniciarcontrole1Click(Sender: TObject);
begin
diretorio_form.visible:=true;
op:=1;
principal_form.Enabled:=false;
end;

procedure Tprincipal_form.raargrfico1Click(Sender: TObject);
begin
op:=2;
grafico_form.visible:=true;
diretorio_form.visible:=true;
grafico_form.enabled:=false;
end;

procedure Tprincipal_form.Sair1Click(Sender: TObject);
begin
Close;
end;

end.
```

b)Código fonte do form responsável por “plotar” os gráficos resultantes de controles realizados anteriormente

```
unit grafico;

interface
```

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, TeeProcs, TeEngine, Chart, StdCtrls, Series, Menus;

type
  Tgrafico_form = class(TForm)
    Chart1: TChart;
    Chart2: TChart;
    Chart3: TChart;
    Chart4: TChart;
    Series1: TLineSeries;
    Series2: TLineSeries;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    Series6: TLineSeries;
    Series7: TLineSeries;
    Series8: TLineSeries;
    Button1: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormShow(Sender: TObject);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  grafico_form: Tgrafico_form;

implementation

Uses diretorio, principal;
{$R *.dfm}

procedure Tgrafico_form.FormCreate(Sender: TObject);
begin
decimalseparator:='.';
end;

procedure Tgrafico_form.Button1Click(Sender: TObject);
var
arqX : textFile;
imr,is1real,is2real,is3real,w,wref,isd,isq: array [0..15100] of double;
i,n_amostras,j: integer;
aux: string;
begin
button1.Visible:=false;
Chart1.Series[0].clear;
Chart1.Series[1].clear;
Chart2.Series[0].clear;
Chart2.Series[1].clear;
```

```
Chart2.Series[2].clear;
Chart3.Series[0].clear;
Chart3.Series[1].clear;
Chart4.Series[0].clear;

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\imr.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
    imr[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);
n_amostras:=i;

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\is1real.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
    is1real[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\is2real.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
```

```
    is2real[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\is3real.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
    is3real[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\wref.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
    wref[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\w.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
    w[i]:=strtofloat(aux);
```

```
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\isd.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
    isd[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);

i:=0;
AssignFile(arqX,diretorio_form.texto_salvdado.text+'\isq.txt');
Reset(arqX);
repeat
    ReadLn(arqX,aux);
    isq[i]:=strtofloat(aux);
    i:=i+1;
until EOF(arqX);
CloseFile(arqX);

i:=0;
repeat
   Chart1.Series[0].AddXY ((i*0.000164*11),wref[i],'',clTeeColor);
   Chart1.Series[1].AddXY ((i*0.000164*11),w[i],'',clTeeColor);
   Chart2.Series[0].AddXY ((i*0.000164*11),is1real[i],'',clTeeColor);
   Chart2.Series[1].AddXY ((i*0.000164*11),is2real[i],'',clTeeColor);
   Chart2.Series[2].AddXY ((i*0.000164*11),is3real[i],'',clTeeColor);
```

```
    Chart3.Series[0].AddXY ((i*0.000164*11),isd[i],'',clTeeColor);
    Chart3.Series[1].AddXY ((i*0.000164*11),isq[i],'',clTeeColor);
    Chart4.Series[0].AddXY ((i*0.000164*11),imr[i],'',clTeeColor);


  i:=i+1;
until i>=n_amostras-1;
end;

procedure Tgrafico_form.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
button1.Visible:=false;
principal_form.enabled:=true;
end;

procedure Tgrafico_form.FormShow(Sender: TObject);
begin
Chart1.Series[0].clear;
Chart1.Series[1].clear;
Chart2.Series[0].clear;
Chart2.Series[1].clear;
Chart2.Series[2].clear;
Chart3.Series[0].clear;
Chart3.Series[1].clear;
Chart4.Series[0].clear;
principal_form.enabled:=false;
end;

end.
```

c)Código fonte do form para leitura do caminho onde os arquivos serão armazenados ou carregados.

```
unit diretorio;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  Tdiretorio_form = class(TForm)
    texto_salvdado: TEdit;
    Button1: TButton;
    Label1: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  diretorio_form: Tdiretorio_form;

implementation

uses principal, grafico;
```

```
{$R *.dfm}

procedure Tdiretorio_form.Button1Click(Sender: TObject);
begin

if principal_form.op= 1 then
begin
principal_form.Button1.Enabled:=true;
principal_form.Enabled:=true;
end;
if principal_form.op= 2 then
begin
grafico_form.Button1.Visible:=true;
grafico_form.Enabled:=true;
end;
diretorio_form.Visible:=false;
end;

procedure Tdiretorio_form.FormCreate(Sender: TObject);
begin

end;

end.
```