3.3 Sumário

Uma das vantagens da topologia de inversor escolhida para a implementação do protótipo, é a relativa flexibilidade na geração de formas de onda desequilibradas. A partir da modelagem matemática da planta, verifica-se que o quarto braço do inversor resultou em um modelo matemático, onde as tensões de saída da planta são dependentes apenas da tensão de entrada correspondente no filtro. Este quarto braço representa um quarto grau de liberdade, que possibilita também a condução de qualquer corrente elétrica pelo neutro. Deste modo, a Fonte de Potência CA proposta pode constituir-se num bom gerador de formas de onda e pode simular os diversos eventos relacionados que podem ocorrer no sistema de distribuição.

Na seção 3.2 foram apresentados alguns detalhes de operação do inversor, incluindo o algoritmo de modulação *space vector*. A descrição foi baseada no trabalho de Botterón *et al* [21].

CAPÍTULO 4

CONTROLADOR RMRAC (*Robust Model Reference Adaptive Control)*

Sistemas físicos reais são não-lineares, com modelo matemático muitas vezes difícil de ser obtido e com parâmetros variantes no tempo. Estes fatores implicam em variações na planta e que, do ponto de vista de sistemas de controle não são, em geral, modelados.

Em Fontes de Potência CA, dada a sua ampla faixa de geração de formas de onda, além da variação paramétrica da planta que está ligada, por exemplo, a variação da permeabilidade magnética de núcleos de indutores, tem-se as não-linearidades associadas à comutação, as dinâmicas não-modeladas formadas por resistências e elementos reativos parasitas e distúrbios externos. A modelagem de todas essas dinâmicas exigiria um grande esforço e em muitos casos inviável dada a complexidade do modelo obtido e ao fato do sistema real ser variante no tempo. Com o objetivo de buscar soluções de controle aplicáveis às Fontes de Potência CA, alguns estudos têm sido feitos no intuito de buscar soluções que garantam robustez, estabilidade e desempenho satisfatório a plantas sujeitas a dinâmicas não-modeladas e distúrbios externos.

Muitos autores utilizam controladores que garantam uma resposta rápida como em Jung *et al* [22], onde um controlador *Deadbeat* é empregado no controle das malhas interna e externa. Tzou *et al* em [11], aplicam um controlador por alocação de pólos via retroação de estados para melhorar o desempenho transitório. A fim de garantir um baixo erro em regime permanente, uma ação repetitiva tem sido adicionada. Em [9], Low propõe um controlador por modos deslizantes (*Sliding Mode Control*), o que garante robustez ao sistema mesmo com as variações paramétricas do sistema e distúrbios externos. Em [12] o mesmo autor utiliza um controlador preditivo generalizado (GPC-*Generalized Predictive Control*), o qual calcula uma seqüência prévia de sinais de controle, que é atualizada a cada intervalo de amostragem. Esta técnica utiliza o modelo da planta para o cálculo destas ações de controle, o que pode ser inviável em determinadas situações mais críticas principalmente em variações de carga.

A técnica de Controle Adaptativo Robusto por Modelo de Referência (RMRAC) (ver o diagrama de blocos da Figura 4.1) demanda um esforço computacional maior que as técnicas citadas anteriormente, entretanto as vantagens relativas à robustez e estabilidade fazem dela uma técnica atrativa para as Fontes de Potência CA. Controladores adaptativos possuem duas partes; a Lei de Controle e o Algoritmo de Adaptação Paramétrica. O objetivo do controlador RMRAC é calcular uma lei de controle "u" de modo que a saída da planta "y" siga a saída do modelo de referência "y_m" mesmo na presença de dinâmicas não modeladas.

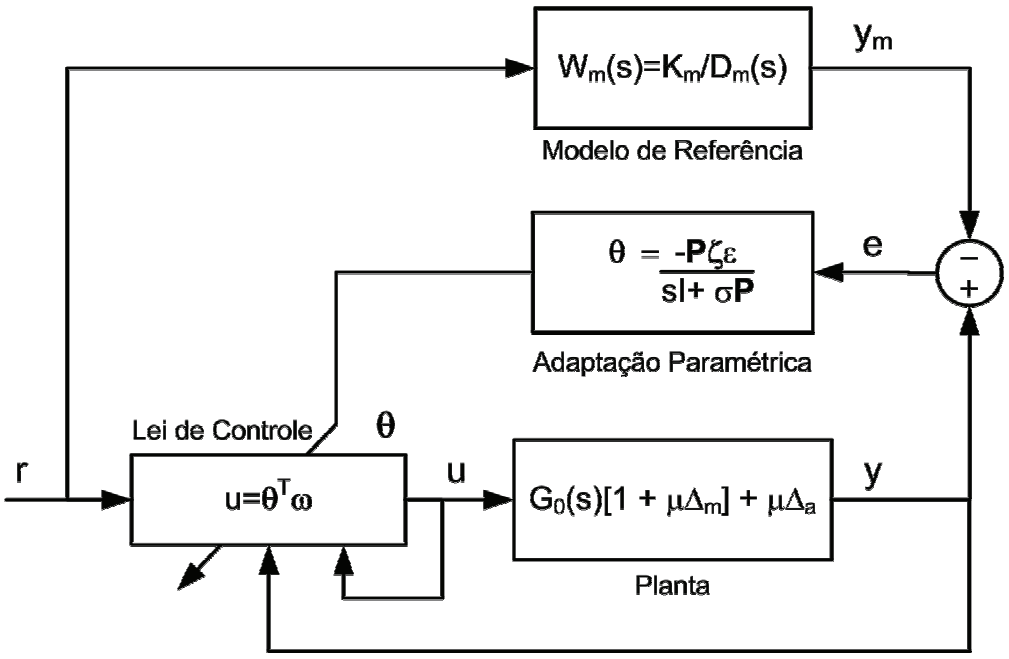


Figura 4.1: Controlador RMRAC

4.1 Descrição da Planta e do Modelo de Referência

Conforme mostrado no capítulo anterior, a planta pode ser modelada por três sistemas SISO idênticos (um sistema para cada fase de saída). Deste modo, todo o desenvolvimento que se segue considera uma única planta SISO.

A planta G(s) pode ser descrita por

$$\begin{aligned} &y(s)=G(s)u(s) \ \text{ ou} \\ &y(s)=\left(G_0(s)\left[1+\mu\Delta_m(s)\right]+\mu\Delta_a(s)\right)u(s) \end{aligned} \tag{4.1}$$

com

$$G(s)=k_p \frac{Z_0(s)}{R_0(s)} \tag{4.2}$$

onde $G_0(s)$ é a função de transferência da parte modelada da planta, $\mu\Delta_m$ e $\mu\Delta_a$ são dinâmicas multiplicativas e aditivas respectivamente e $Z_0(s)$ e $R_0(s)$ são polinômios de graus *m* e *n* respectivamente.

As seguintes suposições são feitas sobre a planta, para a parte modelada:

S1: $Z_0(s)$ é um polinômio mônico de Hurwitz de grau *m* ($\leq n-1$),

S2: $R_0(s)$ é um polinômio mônico de Hurwitz de grau *n*,

S3: O sinal de k_p e os valores de *m* e *n* são conhecidos. Sem perda de generalidade pode se considerar $k_p>0$,

Para a parte não modelada da planta assume-se que:

S4: $\Delta_a(s)$ é uma função de transferência estritamente própria e estável,

S5: $\Delta_m(s)$ é uma função de transferência estável,

S6: É conhecido um limite inferior p_0, para o qual os pólos de $\Delta_a(s\text{-}p)$ e $\Delta_m(s\text{-}p)$ são estáveis.

Seja o modelo de referência dado por

$$y_m(s)=W_m(s)r(s)=k_m \frac{1}{D_m(s)} r(s) \tag{4.3}$$

onde $D_m(s)$ é um polinômio mônico de Hurwitz de grau $n^*=n-m$ e r(s) é uma entrada externa limitada. O objetivo básico do controlador RMRAC é sintetizar uma ação de controle u(s), tal que o sistema em malha fechada seja estável para qualquer valor de $\mu \in \left[0,\mu^*\right)$ e $\mu^*>0$ e a saída da planta y(s) rastreie a saída do modelo de referência $y_m(s)$, mesmo na presença de dinâmicas não modeladas $\Delta_a(s) \text{ e } \Delta_m(s)$, desde que estas satisfaçam as hipóteses **S4-S6**.

4.2 Estrutura do Controlador

A lei de controle utilizada para o cálculo da ação de controle RMRAC possui a seguinte forma

$$u = \boldsymbol{\theta}^T \boldsymbol{\omega} \tag{4.4}$$

onde os vetores $\boldsymbol{\theta}$ e $\boldsymbol{\omega}$ são dados por $\boldsymbol{\theta}^T = \begin{bmatrix} \theta_1^T & \theta_2^T & \theta_3 & \theta_4 \end{bmatrix}$ e $\boldsymbol{\omega}^T = \begin{bmatrix} \omega_1^T & \omega_2^T & y & r \end{bmatrix}$, ambos de dimensão $(2n-1)$. Os sinais auxiliares ω_1 e ω_2 são dados por

$$\begin{aligned} \boldsymbol{\omega_1}(s) &= (s\mathbf{I}-\mathbf{F})^{-1} q u(s) \\ \boldsymbol{\omega}_2(s) &= (s\mathbf{I}-\mathbf{F})^{-1} q y(s) \end{aligned} \tag{4.5}$$

onde (F,q) é um par controlável.

Lema 4.1: O erro de rastreamento $e_1 = y - y_m$ é dado por

$$e_1 = \theta_4^{*-1} W_m(s) \boldsymbol{\phi}^T \boldsymbol{\omega} + \mu\eta(s) \tag{4.6}$$

com

$$\eta = \Delta(s) u \tag{4.7}$$

Prova: Dado o vetor $\boldsymbol{\theta}^{*T} = \begin{bmatrix} \theta_1^{*T} & \theta_2^{*T} & \theta_3^* & \theta_4^* \end{bmatrix}$, pode-se definir o erro de adaptação paramétrica $\boldsymbol{\phi} = \boldsymbol{\theta} - \boldsymbol{\theta}^*$.

Subtraindo o termo $\boldsymbol{\theta}^{*T}\boldsymbol{\omega}$ em ambos os lados de (4.4) têm-se

$$\boldsymbol{\phi}^T \boldsymbol{\omega} = u - \boldsymbol{\theta}^{*T} \boldsymbol{\omega} \tag{4.8}$$

Expandindo (4.8), obtém-se

$$\boldsymbol{\phi}^{T}\boldsymbol{\omega}+\theta_4^{*}r=\left[1-\theta_1^{*T}\left(sI-\mathbf{F}\right)^{-1}q-\theta_2^{*T}\left(sI-\mathbf{F}\right)^{-1}qG(s)-\theta_3^{*}G(s)\right]u \tag{4.9}$$

Definindo-se

$$F_1(s)=\boldsymbol{\theta}_1^{*T}\left(s\mathbf{I}-\mathbf{F}\right)^{-1}q \tag{4.10}$$

e

$$F_2(s)=\boldsymbol{\theta}_3^{*T}+\boldsymbol{\theta}_2^{*T}\left(s\mathbf{I}-\mathbf{F}\right)^{-1}q \tag{4.11}$$

obtém-se de (4.9)

$$\boldsymbol{\phi}^{T}\boldsymbol{\omega}+\boldsymbol{\theta}_4^{*}r=\left[1-F_1(s)-F_2(s)G(s)\right]u \tag{4.12}$$

Devido à controlabilidade de $G_0(s)$ existe um vetor $\boldsymbol{\theta}^{*}$ tal que $\boldsymbol{\phi}$ é um vetor nulo e então se pode escrever que $W_m(s)r=G_0(s)u$, ou seja, inverter a ordem $y_m \equiv y$.

Assim (4.12) torna-se:

$$\boldsymbol{\theta}_4^{*}W_m(s)^{-1}G_0(s)=\left[1-F_1(s)-F_2(s)G_0(s)\right] \tag{4.13}$$

De (4.13) se obtém

$$G_0(s)=\boldsymbol{\theta}_4^{*-1}W_m(s)\left[1-F_1(s)-F_2(s)G_0(s)\right] \tag{4.14}$$

Substituindo (4.14) em (4.1) têm-se

$$\begin{aligned} y=&[\theta_4^{*-1}W_m(s)\left[1-F_1(s)-F_2(s)G_0(s)\right]+\ldots \\ &\ldots\theta_4^{*-1}W_m(s)\left[1-F_1(s)-F_2(s)G_0(s)\right]\mu\Delta_m(s)+\mu\Delta_a(s)]u \end{aligned} \tag{4.15}$$

Somando e subtraindo $F_2(s)G(s)u$ em $\left[1-F_1(s)-F_2(s)G_0(s)\right]$ se obtém

$$y=[\theta_4^{*-1}W_m(s)\left[1-F_1(s)-F_2(s)G(s)\right]u+[\theta_4^{*-1}W_m(s)\left[-F_2(s)G_0(s)+F_2(s)G(s)\right]+\ldots \\ \ldots\theta_4^{*-1}W_m(s)\left[1-F_1(s)-F_2(s)G(s)\right]\mu\Delta_m(s)+\mu\Delta_a(s)]u \tag{4.16}$$

$$y=\theta_4^{*-1}W_m(s)\left[\boldsymbol{\phi}^T\boldsymbol{\omega}+\theta_4^{*}r\right]+[\theta_4^{*-1}W_m(s)F_2(s)\mu\Delta_a(s)+\ldots \\ \ldots\theta_4^{*-1}W_m(s)\left[1-F_1(s)\right]\mu\Delta_m(s)+\mu\Delta_a(s)u \tag{4.17}$$

Realizando operações algébricas em (4.17) obtém-se o erro de rastreamento

$$e_1=\theta_4^{*-1}W_m(s)\boldsymbol{\phi}^T\boldsymbol{\omega}+\mu\eta(s) \tag{4.18}$$

onde $\eta=\Delta(s)u$ com $\Delta(s)=\Delta_a(s)\left[1+\theta_4^{*-1}W_m(s)F_2(s)\right]+\theta_4^{*-1}W_m(s)\Delta_m(s)\left[1-F_1(s)\right]$.

Definindo $\varphi=\left(\theta_4\right)^{-1}-\left(\theta_4^{*}\right)^{-1}$ de (4.18) pode-se obter

$$e_1=\theta_4^{-1}W_m(s)\boldsymbol{\phi}^T\boldsymbol{\omega}-\varphi W_m(s)\boldsymbol{\phi}^T\boldsymbol{\omega}+\mu\eta(s) \tag{4.19}$$

Usando a assertiva:

$$\boldsymbol{\phi}^T W_m(s)\boldsymbol{\omega}-W_m(s)\boldsymbol{\phi}^T\boldsymbol{\omega}=\boldsymbol{\theta}^T W_m(s)\boldsymbol{\omega}-W_m(s)\boldsymbol{\theta}^T\boldsymbol{\omega} \tag{4.20}$$

e a equação do erro de rastreamento (4.18), obtém-se o erro aumentado ε_1 dado por

$$\varepsilon_1=e_1+\theta_4^{-1}\left[\boldsymbol{\theta}^T\boldsymbol{\zeta}-W_m(s)\boldsymbol{\theta}^T\boldsymbol{\omega}\right]=\theta_4^{-1}\boldsymbol{\phi}^T\boldsymbol{\zeta} \tag{4.21}$$

com $\boldsymbol{\zeta}=W_m(s)\mathbf{I}\boldsymbol{\omega}$.

Analisando a equação (4.21), observa-se que o erro aumentado considera não apenas o erro de rastreamento, mas também o erro de adaptação paramétrica $\boldsymbol{\phi}$.

4.3 Algoritmo de Adaptação Paramétrica

O algoritmo de adaptação paramétrica utilizado é do tipo gradiente da forma

$$\dot{\boldsymbol{\phi}} = \dot{\boldsymbol{\theta}} = -\mathbf{P}\boldsymbol{\zeta}\varepsilon - \sigma\mathbf{P}\boldsymbol{\theta} \tag{4.22}$$

onde $\mathbf{P}=\mathbf{P}^T$ e

$$\varepsilon(t)=\frac{\varepsilon_1(t)}{\overline{m}(t)}=\frac{e_1+\theta_4^{-1}\left[\boldsymbol{\theta}^T\boldsymbol{\zeta}-W_m(s)\boldsymbol{\theta}^T\boldsymbol{\omega}\right]}{\overline{m}(t)}$$
$$\overline{m}(t)=1+\alpha_1 m(t)^2 \tag{4.23}$$

e

$$\dot{m}(t)=\frac{-\delta_0}{1+\delta_m}m(t)+\delta_1\left(|u|+|y|+1\right),\ m(0)>\frac{\delta_1}{\delta_0},\ \delta_1\geq 1 \tag{4.24}$$

onde $\alpha_1, \delta_0, \delta_1, \lambda, \bar{\mu}$ e R^2 são constantes positivas e $0<\delta_0<q_0$. O parâmetro $q_0 \in \Re^+$ é tal que os pólos de $W_m(s-q_0)$ e os autovalores de $\left(q_0\mathbf{I}+\mathbf{F}\right)$ são estáveis, $\delta_m \geq 0$ é um parâmetro de projeto.

O parâmetro σ em (4.22) é dado por

$$\sigma=\begin{cases} 0 & \text{if} \quad \|\boldsymbol{\theta}\|<M_0 \\ \sigma_0\left(\|\boldsymbol{\theta}\|/M_0-1\right) & \text{if} \quad M_0\leq\|\boldsymbol{\theta}\|\leq 2M_0 \\ \sigma_0 & \text{if} \quad \|\boldsymbol{\theta}\|>2M_0 \end{cases} \tag{4.25}$$

onde $M_0>\|\theta^*\|$ e $\sigma_0>2\bar{\mu}^2/R^2\in\Re^+$ são parâmetros de projeto.

4.4 Sumário

Neste capítulo é proposta a aplicação de um controlador adaptativo para o controle de uma Fonte de Potência CA. Este controlador é aplicado a plantas que possuem dinâmicas não-modeladas obedecendo às propriedades descritas na primeira seção deste capítulo. É feita a obtenção da equação do erro de rastreamento e do erro aumentado que são utilizados pelo algoritmo de adaptação paramétrica do tipo mínimos quadrados modificado. Maiores detalhes sobre as provas de estabilidade deste algoritmo podem ser encontradas em Yoannou e Tsakalis [23].

CAPÍTULO 5

RESULTADOS DE SIMULAÇÃO

Antes da implementação experimental do protótipo, o sistema de controle da planta baseada no controlador RMRAC (capítulo anterior), foi simulado através de um programa implementado em ambiente MATLAB®. A fim de obter um resultado nas condições mais reais possíveis, o acionamento do inversor foi simulado. A estratégia de implementação da modulação PWM utilizada é baseada em um algoritmo de modulação *space vector* tal como apresentado em [21]. O modelo da planta e da estratégia de modulação utilizados para simulação são mostrados no Capítulo 3 e o controlador no Capítulo 4. A Figura 5.1 mostra o diagrama do sistema que foi simulado.

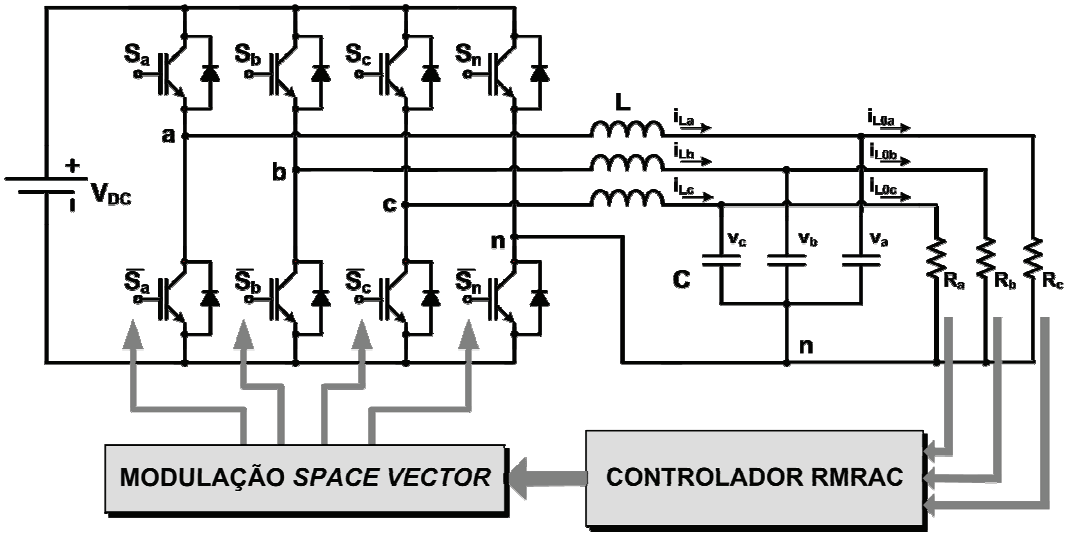


Figura 5.1: Sistema inversor, planta e controlador simulados

Os valores dos componentes da planta, do inversor e do controlador são dados nas Tabela 5.1 e Tabela 5.2.

5.1 Simulação da Planta e Controlador

Para o caso de fontes de potência com comutação, o estágio de saída possui um filtro LC para supressão do conteúdo harmônico oriundo da comutação das chaves. Para o controle

do sistema, o filtro e a carga formam a planta a ser controlada segundo a forma dada em (4.1) e (4.2), e tendo as propriedades descritas em $\mathbf{S_1}$-$\mathbf{S_6}$ (vide Capítulo 4).

O Modelo de Referência do controlador é um modelo que deve possuir dinâmicas semelhantes às da planta para que em malha-fechada, a planta rastreie sua saída apesar das dinâmicas não-modeladas existentes no sistema. Para a implementação deste controlador, tanto em simulação quanto experimentalmente, a Planta, o Modelo de Referência e as equações (4.1)–(4.5) e (4.22)–(4.25) foram discretizadas utilizando o Método de Euler.

A partir das equações em tempo discreto, é possível elaborar um programa para simular a planta em malha fechada. A Tabela 5.1 mostra os valores dos parâmetros da planta e de discretização utilizados na simulação para o filtro de saída, o inversor e o controlador discreto.

Como verificado no capítulo anterior, no esquema de controle RMRAC existe uma série de parâmetros de projeto cuja escolha requer conhecimento e experiência do projetista a fim de melhorar o desempenho do controlador, um bom ponto de partida para projeto é considerar $G_O(s)=G(s)$. A Tabela 5.2 mostra os parâmetros do controlador RMRAC utilizados. Apesar de existir um ponto ótimo na escolha dos valores combinando desempenho e robustez, muitos dos parâmetros possuem seleção de valores numa larga faixa, como por exemplo, o par (**F**,**q**) para a implementação de filtros internos, que respondem principalmente pela robustez do controlador.

Tabela 5.1 – Parâmetros da planta, inversor e controle

Parâmetro	Símbolo	Valor
Indutor	L	1mH
Capacitor	C	60μH
Barramento CC	V_{CC}	540
Período de amostragem	T_S	1/12000
Período de comutação	T_{SW}	1/12000

Os resultados de simulação têm por objetivo verificar o comportamento da planta e controlador em malha fechada numa etapa anterior à da implementação prática. Os aspectos analisados são relativos à estabilidade e robustez do controlador e desempenho na geração de formas de onda complexas. Neste sentido, diversas formas de onda foram obtidas, muitas delas relativas às normas citadas no Capítulo 2.

Tabela 5.2 – Parâmetros do controlador RMRAC e do algoritmo de adaptação

Parâmetro	Símbolo	Valor
Parâmetros adaptados	$\boldsymbol{\theta}^T = \begin{bmatrix} \theta_1^T & \theta_2^T & \theta_3 & \theta_4 \end{bmatrix}$	$\begin{bmatrix} -2 & 2 & 0.3 & 1 \end{bmatrix}^{(*)}$
Matriz de Covariância	$\mathbf{P}$	$10 \cdot \mathbf{I}_{4\times4}$
Vetor de regressão dos estados	ξ	$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^{(*)}$
Matriz de dinâmica dos filtros auxiliares	$\mathbf{F}$	3000
Matriz dos ganhos dos filtros auxiliares	$\mathbf{q}$	100
Limitador da dinâmica não modelada	$\bar{\mu}$	0,03
Limitador da Matriz de Covariância	R	20
Limitador da função *σ-modification*	M_0	4
Valor inicial do σ	σ_0	0,03
Constantes do RMRAC	δ_0	3
Constantes do RMRAC	δ_1	1
Constantes do RMRAC	α_1	1

Na implementação prática do sistema de controle existe limitação quanto ao processamento da lei de controle. Após simulações usando um algoritmo do tipo RLS, na implementação foi ajustado um algoritmo do tipo gradiente, o que diminui o número de operações matemáticas envolvidas no cálculo da Lei de Controle. Ou seja, A matriz de covariância **P** em (4.22) foi considerada constante, inclusive na obtenção dos resultados de simulação. A fim de mostrar primeiramente o comportamento característico do controlador RMRAC, foi realizado um teste simples, onde foi simulado o controle da planta com uma carga trifásica equilibrada resistiva de 24Ω. Na Figura 5.2 é mostrada as formas de onda da saída da planta e do modelo de referência para uma carga trifásica equilibrada.

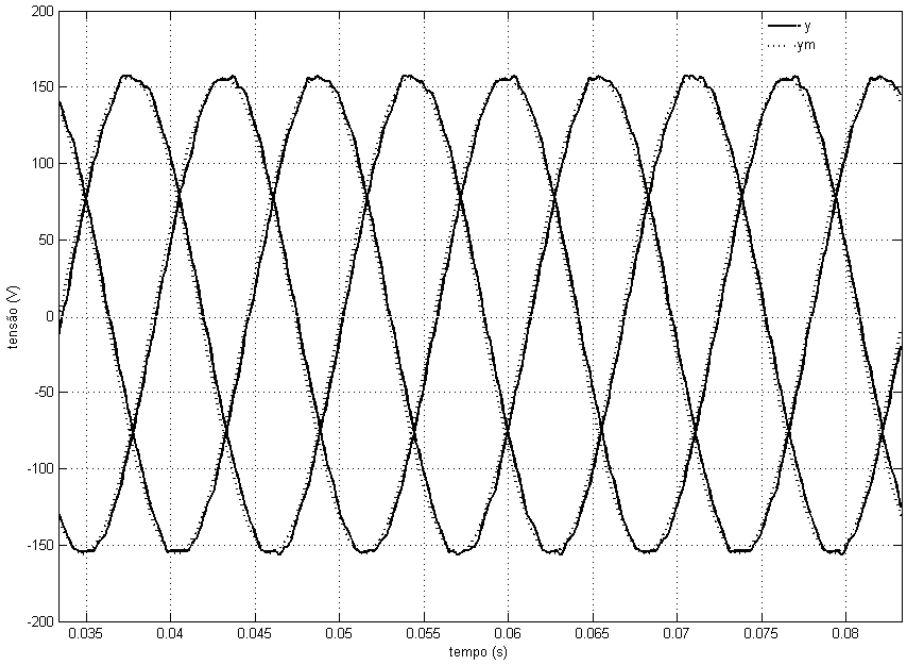


Figura 5.2: Tensões de saída da planta (—) e modelo de referência (····)

A Figura 5.3, apresenta a evolução dos ganhos $\theta_1, \theta_2, \theta_3$ e θ_4 sintonizados pelo algoritmo de adaptação paramétrica e a forma de onda do erro de trajetória normalizado pela tensão de base V_{CC} (ver Tabela 5.2). O resultado é relativo a uma das fases, já que o controle é baseado em três controladores SISO idênticos.

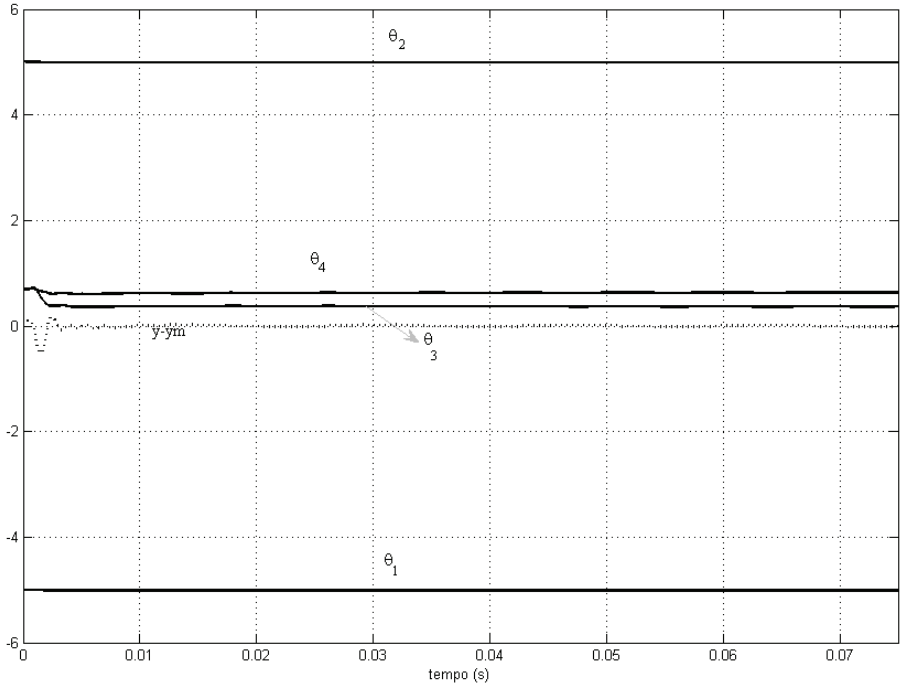


Figura 5.3: Parâmetros adaptados e erro de rastreamento

Como um dos objetivos deste trabalho é demonstrar a capacidade do protótipo implementado na geração de formas de onda genéricas. Foram obtidos alguns resultados incluindo formas de onda com harmônicas, formas de onda desbalanceadas e geração com cargas desequilibradas.

Formas de onda com componentes harmônicas na rede elétrica representam um grande problema, pois equipamentos a ela conectados podem apresentar mal-funcionamento por interferência eletromagnética conduzida. Outro problema é o desbalanceamento de tensão, que é causado por cargas desbalanceadas conectadas a circuitos polifásicos. Em ambos os casos, uma Fonte de Potência CA que consiga reproduzir estes problemas, pode ser um instrumento muito útil para o teste de Filtros Ativos de Potência (FAP) antes de sua instalação no sistema de distribuição. Apesar deste trabalho não apresentar nenhum resultado de ensaio para formas de onda geradas pela Fonte de Potência CA e compensadas por FAP's, foi realizado um ensaio no qual a geração de formas de onda complexas foram usadas para testar o desempenho de um algoritmo de detecção de fase. Os resultados estão mostrados no APÊNDICE A. Na Figura 5.4 é mostrada uma forma de onda trifásica gerando a referência $r(t)=0{,}5\sin(\omega t)+0{,}25\sin(5\omega t)+0{,}05\sin(7\omega t)$. Estas harmônicas foram escolhidas por serem bastante comuns na rede elétrica, a 11º harmônica não foi simulada devido à limitação do filtro LC de saída utilizado, o qual não pode ter uma banda passante muito alta sob pena de não atenuar suficientemente as harmônicas oriundas da comutação.

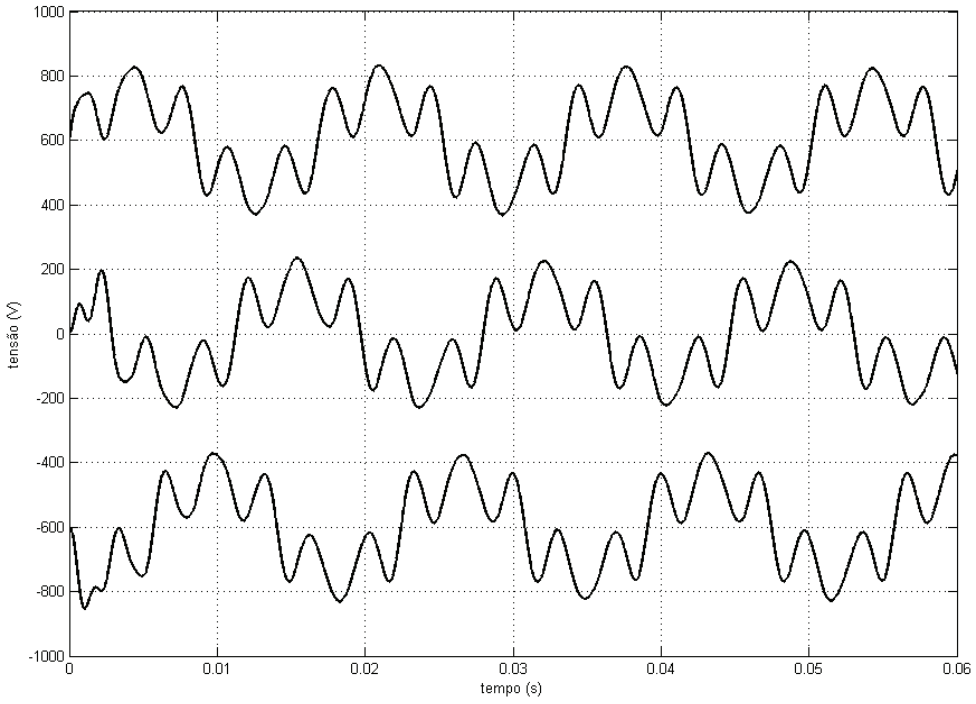


Figura 5.4: Forma de onda fundamental e harmônicas

A Figura 5.5 mostra uma forma de onda com freqüência fundamental de 300Hz para uma carga de 24Ω.

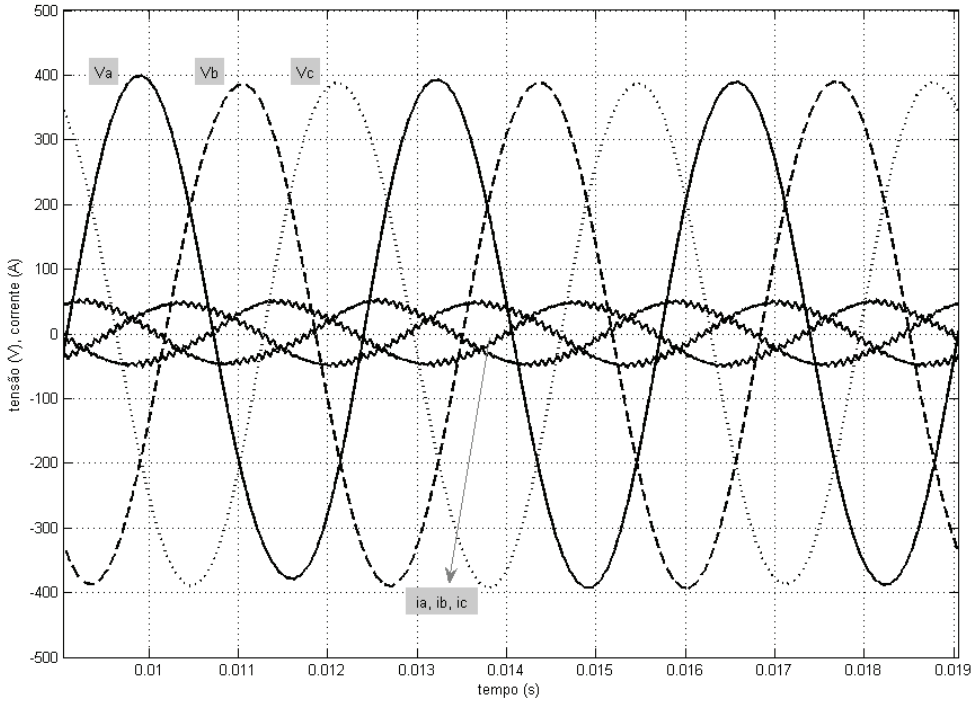


Figura 5.5: Tensões de saída e correntes nos indutores para uma fundamental de 300Hz

Na Figura 5.6 é mostrada uma forma de onda trifásica desequilibrada de 50Hz para uma carga balanceada de 24Ω. Na Figura 5.7 é mostrada uma forma de onda em condições opostas, gerando tensões de saída trifásicas equilibradas para uma carga desbalanceada de 10Ω, 24Ω e 48Ω.

Fenômenos de afundamento na tensão de saída são causados principalmente por surtos de carga. A Figura 5.8 mostra uma forma de onda com afundamento de 34% na tensão de saída durante dois ciclos.

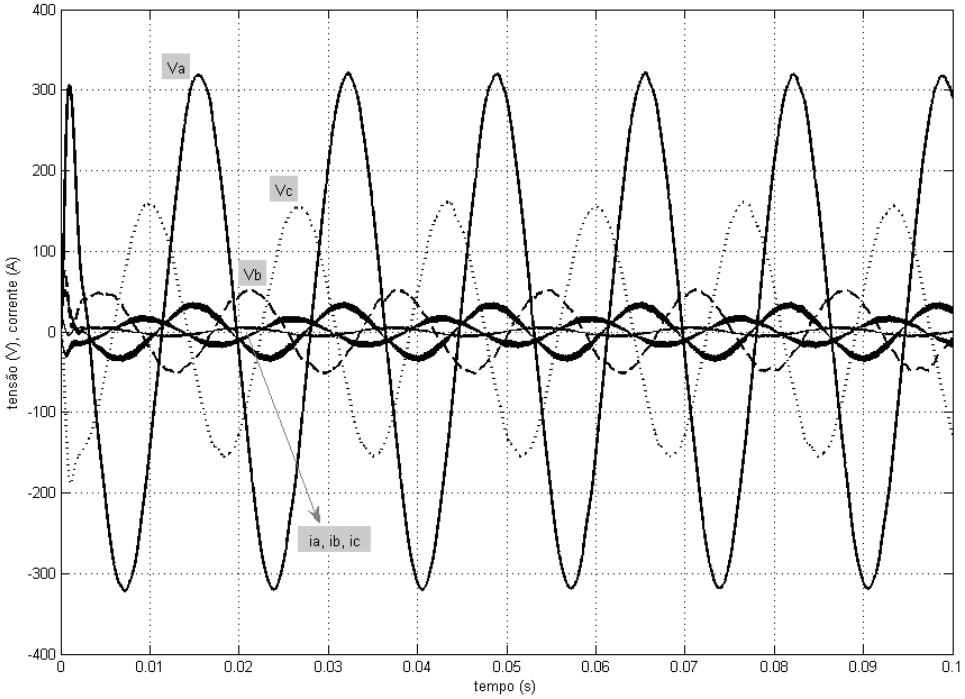


Figura 5.6: Formas de onda trifásicas desequilibradas de 50Hz para carga desbalanceada

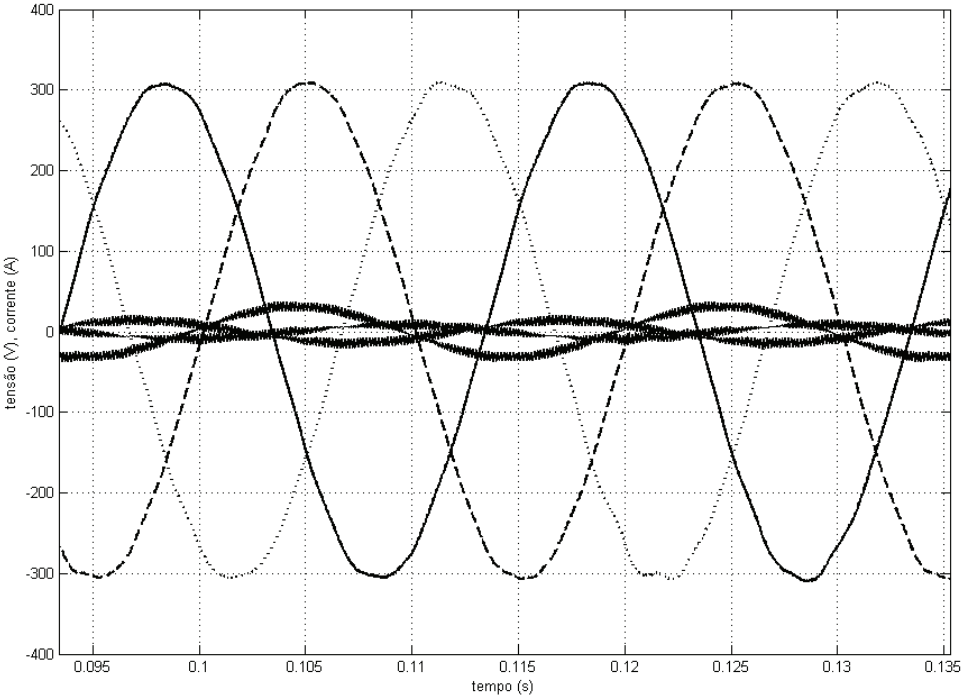


Figura 5.7: Tensões desequilibradas com cargas desbalanceadas em 50Hz

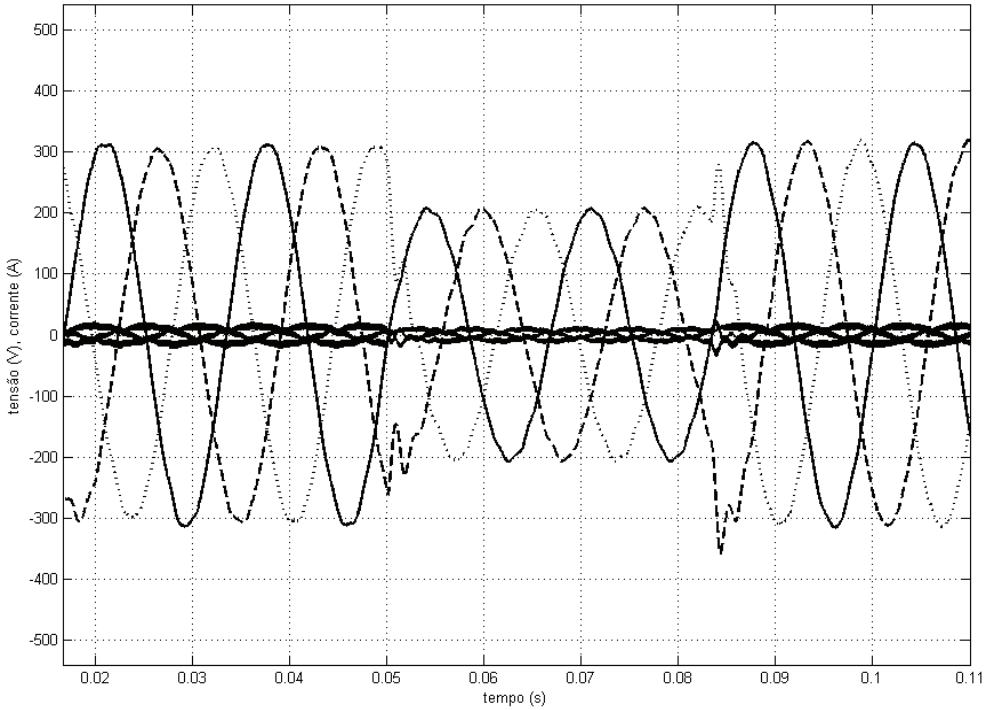


Figura 5.8: Afundamento de dois ciclos na tensão de saída

5.2 Sumário

Neste capítulo é apresentada a técnica de discretização do algoritmo RMRAC, os valores dos ganhos simulados e resultados obtidos por simulação. Apesar de a carga simulada ter sido resistiva, nesta etapa procurou-se apenas obter alguns resultados mostrando a possibilidade de geração de formas de onda desbalanceadas e com cargas desequilibradas. Entretanto, os resultados que incluem variações de amplitude mostraram que o controlador RMRAC possui um tempo de resposta que não é suficiente em casos onde se tenha que simular quedas e elevações súbitas de tensão. Ou seja, outros controladores ou ações que melhorem o tempo de resposta da planta em malha-fechada, são propostas a serem investigadas futuramente.

CAPÍTULO 6

PROTÓTIPO IMPLEMENTADO

A fim de validar a aplicação do sistema proposto e a obtenção experimental dos resultados até então obtidos por simulação, um protótipo de Fonte de Potência CA foi desenvolvida em laboratório. A Figura 6.1 mostra o diagrama de blocos básico do sistema. Detalhes da implementação serão apresentados a seguir.

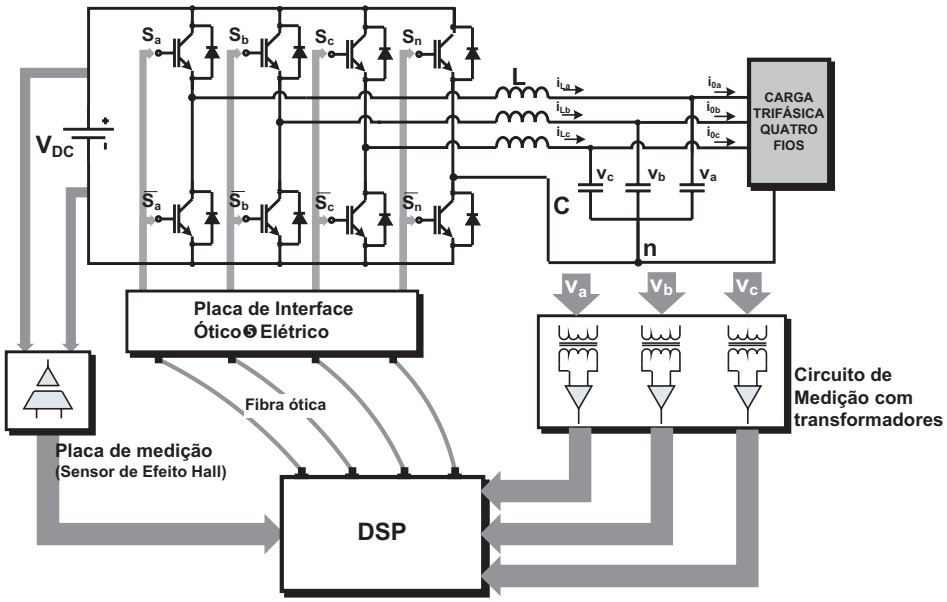


Figura 6.1: Diagrama de blocos do protótipo implementado

6.1 DSP TMS230C2812

Em Fontes de Potência CA a interface homem-máquina é uma característica presente em praticamente todos os equipamentos disponíveis hoje no mercado. Ela pode ser implementada por meio de um simples *display* com um teclado alfanumérico, ou por uma plataforma computadorizada, onde normalmente um programa permite ao usuário a seleção de vários tipos de testes pré-definidos, a visualização das formas de onda e a impressão de relatórios sobre o ensaio. Em ambos os casos, um processador eletrônico é utilizado para o cálculo das leis de controle, aquisição e processamento de sinais, interface com o usuário dentre outras. No protótipo implementado, o Processador Digital de Sinais ou DSP (*Digital*

Signal Processor) TMS320F2812, da Texas Instruments®, foi utilizado para a aquisição de sinais e o processamento da lei de controle proposta. Para acelerar o processo de desenvolvimento do protótipo foi utilizada a placa de desenvolvimento eZdsp™ F2812, cujo diagrama de blocos está mostrado na Figura 6.2. Esta placa já possui a interface necessária para um rápido desenvolvimento. Ela já vem equipada com conectores para alimentação, memória RAM externa de 64k de palavras de 16bits, DSP F2812, cristal de 30MHz, interface JTAG, conectores de expansão e conectores para os mais diversos pinos de entrada e saída do DSP.

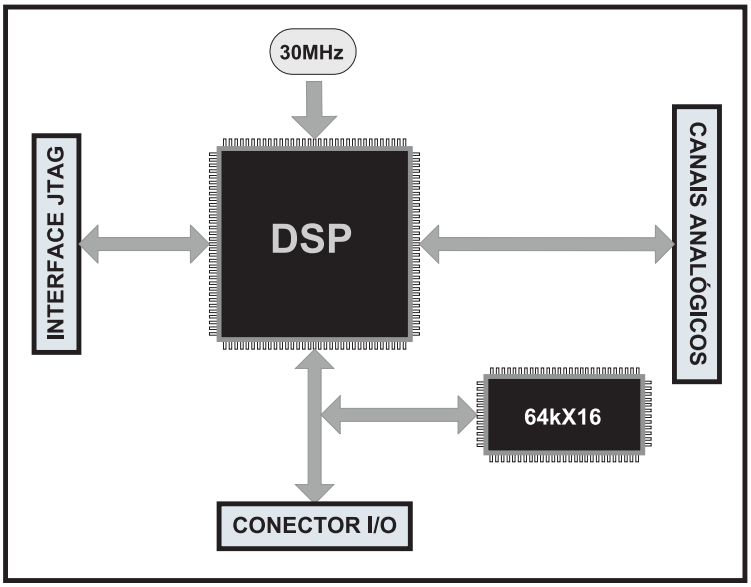


Figura 6.2: Diagrama de blocos simplificado da eZdsp™ F2812 (Fonte: Texas Instruments®)

A seguir será feita uma breve descrição da CPU e dos periféricos do DSP TMS320F2812.

6.1.1 Descrição da CPU

O DSP TMS320F2812 é um processador de 32 bits de ponto fixo e possui uma combinação de arquiteturas RISC e de microcontrolador. A arquitetura RISC permite a execução de uma instrução a cada ciclo de *clock*, tornando a CPU de 150MIPS, com as características de microcontrolador que inclui um conjunto de instruções bastante intuitivo, manipulação de bits e outras.

Alguns módulos principais da CPU são: *Pipeline*, espaço independente para registradores de controle do sistema, Unidade de Lógica e Aritmética (ALU) de 32 bits, *Barrel Shifter* e multiplicador também de 32 bits com resultado em 32 bits.

a) Pipeline: Cada instrução a ser executada passa por oito fases independentes que formam a *pipeline*. Deste modo, pode haver a qualquer momento oito instruções ativas em diferentes fases de execução. A *pipeline* possui um mecanismo que impede que leitura e escrita estejam sendo feitas ao mesmo tempo na mesma região da memória, e outro que maximiza o seu desempenho nos seus oito níveis de execução. As oito fases da *pipeline* são:

- Leitura 1 (F1 ou *Fetch1*): A CPU obtém o endereço da memória de programa em 22bits pelo barramento de endereço de programa (PAB(21:0));
- Leitura 2 (F2 ou *Fetch2*): Nesta fase a CPU adquire os dados da memória de programa pelo barramento de leitura dos dados de programa (PRDB(31:0));
- Decodificação 1 (D1): Identifica a instrução; se é ilegal e o tamanho da próxima instrução a ser executada. O C28x suporta instruções de 16 e de 32 bits;
- Decodificação 2 (D2): O *hardware* relacionado a esta fase, obtém a instrução e a carrega no registrador de instrução, onde a decodificação é completada. Se a instrução atingiu esta fase, ela está pronta a ser executada;
- Leitura 1 (R1): Se algum dado é necessário ser lido da memória, esta fase obtém o seu endereço;
- Leitura 2 (R2): Se existe dado a ser lido, o *hardware* relacionado a esta fase usa o endereço obtido do nível anterior (R1) e o lê;
- Execução (E): Realiza todas as multiplicações, deslocamentos e operações da ALU;
- Escrita (W): Escreve dados na memória, caso exista;

b) Espaço para Registradores Independentes: Estes registradores não são mapeados no espaço de dados. São registradores de controle do sistema, registradores matemáticos e ponteiros de dados;

c) Unidade de Lógica e Aritmética (ALU): A ALU realiza cálculos em 32 bits com complemento de 2's e operações lógicas Booleanas. A ALU pode também receber dados de registradores, memória de dados ou da lógica de controle do programa;

d) Barrel Shifter: Realiza as operações de deslocamento de dados, para a esquerda ou direita em até 16 bits;

e) Multiplicador: Realiza multiplicação em 32 bits com resultado em 64 bits. A multiplicação pode ser realizada entre números com sinal, sem sinal ou entre um com sinal e outro sem sinal;

Em se tratando de um processador cuja aplicação se refere principalmente à implementação de códigos mais complexos, incluindo multiplicações, FFT's e códigos maiores que seus antecessores da linha C24x, a memória interna é um item muito importante. A Tabela 6.1 mostra os blocos de memória da CPU do C28x. No *reset*, o ponteiro da pilha vai para o topo do bloco M1.

Tabela 6.1 – Memória interna do C28x

Bloco	Tamanho
Flash	128k x 16bits
2 blocos, L0 e L1 (SARAM)	4k x 16bits
2 blocos, M0 e M1 (SARAM)	1k x 16bits
1 bloco de RAM (SARAM)	8k x 16bits

6.1.2 Periféricos Principais – *Event Manager* e ADC

Esta seção se ocupa em descrever os módulos periféricos utilizados para a implementação do controle do protótipo. Serão abordados apenas alguns detalhes da configuração dos *timers* e dos conversores A/D para a geração de interrupções e dos sinais PWM.

a) Event Manager (EV): No TMS320F2812 há dois módulos *Event Manager*, o A (EV/A) e o B (EV/B). Devido ao fato de o EV/A e o EV/B serem idênticos e também aos vários periféricos existentes que não são utilizados no trabalho, será feita uma breve descrição acerca do programa que foi implementado.

O EV/A foi o periférico utilizado na geração de base de tempo para as interrupções e o cálculo da lei de controle e para a sincronização desta, com o início da aquisição de dados. A unidade do EV utilizada foi o *timer1* ou contador1. Este

contador serve como a base de tempo e foi programado no modo de contagem *Up-down*. Para este modo de contagem é necessário escrever o valor "0:1" nos bits [12:11] do registrador T1CON. Neste modo, o valor do contador do *timer* é incrementado de zero até um valor limite programado no registrador T1PR, e após decrementado para zero, sucessivamente. A forma de onda obtida tem a forma triangular mostrada na Figura 6.3, e o PWM (forma de onda em cinza) resultante é centrado no período. A interrupção ocorre sempre que o contador atinge o valor 0x000, passando o controle do programa para uma Rotina de Tratamento de Interrupção ou IRQ(*Interrupt Request Routine*), onde são feitas as aquisições pelos conversores A/D e o cálculo da lei de controle. Para a geração do PWM, a lei de controle calculada deve ser transformada no seu equivalente temporal, após, basta carregar o valor em cada um dos registradores T1CMPR, T1CMP, T2CMP e T3CMP. Cada registrador controla o comportamento de dois pinos de saída, que podem ser colocados no modo complementar, inclusive com um tempo morto programado, a fim de acionar um braço do inversor.

b) Conversor Analógico-Digital: O TMS320F2812 possui dois módulos de amostragem e retenção ou S/H, A e B (ver Figura 6.4), sendo que a tensão na entrada dos pinos analógicos selecionados é convertida em 12 bits. A presença de 2 módulos S/H permite que seja utilizado o modo de conversão Simultâneo, além de um modo Seqüencial. No modo de conversão Simultâneo os dois S/H A e B são utilizados paralelamente, ou seja, os canais ADCINAx e ADCINBx são convertidos paralelamente. Esta foi a configuração utilizada na implementação do protótipo. Deste modo os canal a conversão é feita aos pares da seguinte forma: ADCINA3/B3, ADCINA4/B4, ADCINA5/B5 e ADCINA7/B7. No modo de conversão Seqüencial, as entradas analógicas A0, A1,...,A7, B0, B1,..., B7 são convertidas ordenadamente. A seleção das entradas em cada módulo S/H é feita por multiplexadores analógicos. Este multiplexador é controlado pelos registradores CHSELSEQx (x=1, 2, 3, 4), de modo a dar a seqüência de conversões necessárias. Os resultados das conversões são armazenados nos registradores de resultados. Para o modo de conversão Simultâneo são utilizados os registradores ADCRESULT0/1, ADCRESULT2/3,..., ADCRESULT6/7 e para o modo Seqüencial os registradores ADCRESULT0, ADCRESULT1,...., ADCRESULT7.

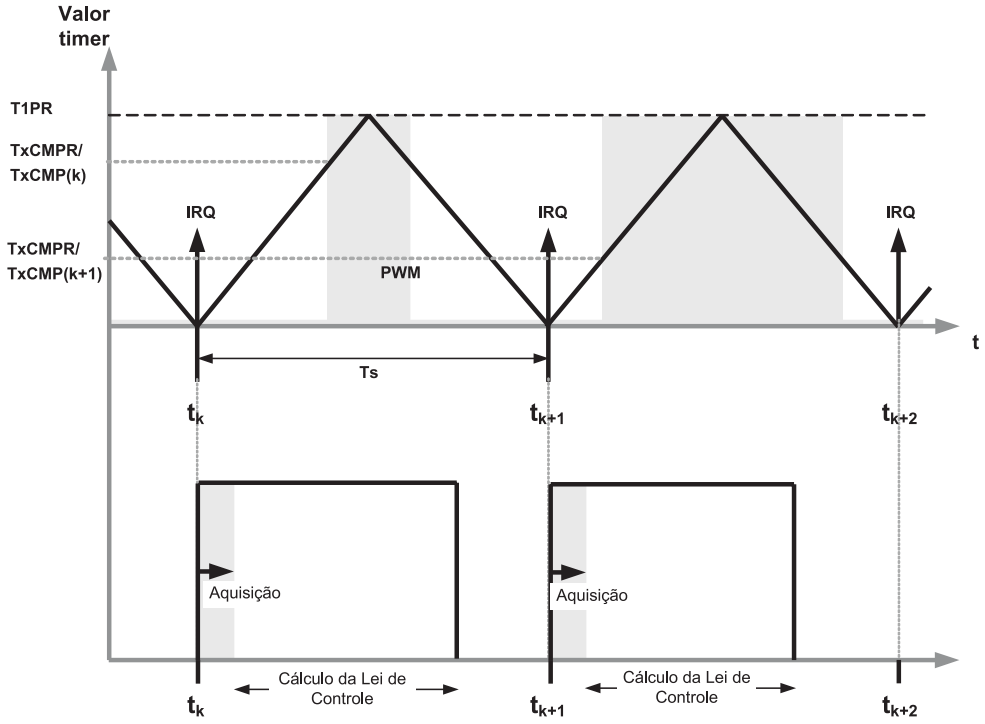


Figura 6.3: Operação do *timer*, geração do PWM e IRQ's

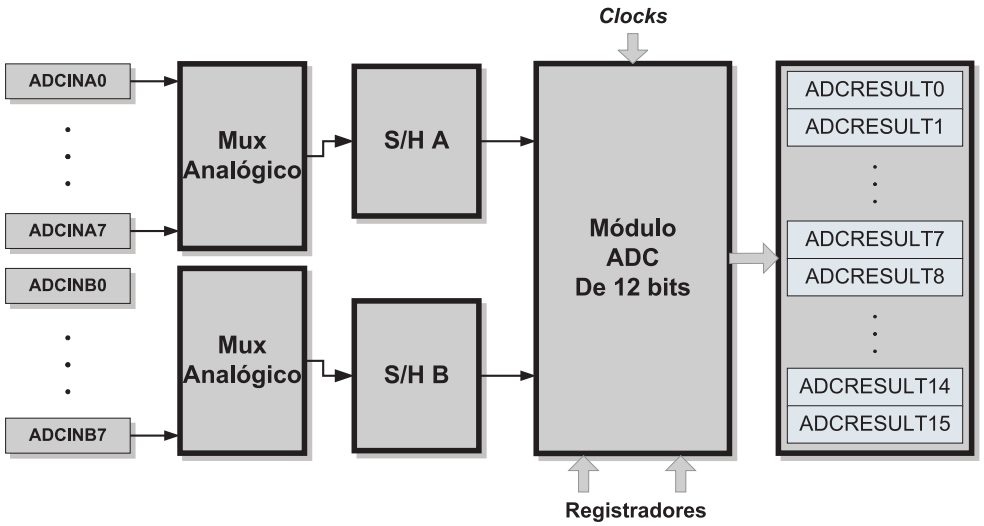


Figura 6.4: Diagrama de blocos do conversor A/D do TMS320F2810/12

Uma seqüência de conversão do A/D pode ser acionada por uma entrada externa, por uma operação de escrita ou pelo *Event Manager*. Para a implementação do protótipo, o acionamento do A/D foi feito pela interrupção do *timer1*. A seleção desta opção é feita pelo registrador ADCTRL2.

6.1.3 Programação

O programa para o controle da Fonte de Potência CA foi desenvolvido em linguagem C/C++, utilizando o compilador do programa Code Composer. O código desenvolvido é responsável pela execução da rotina de controle que implementa o controlador RMRAC, além de responder por configurações da CPU, dos periféricos e da memória externa.

Para os cálculos matemáticos utilizados para a lei de controle (ver APÊNDICE C), foi uma biblioteca de funções matemáticas de ponto fixo chamada IQmath. Estas bibliotecas fazem uso da ALU e do acumulador de 32bits para executarem operações aritméticas rápidas. Funções como seno, cosseno e módulo de números também podem ser utilizadas de modo rápido, e com poucos ciclos de *clock* necessários para a sua execução. As funções da IQmath trabalham com números com comprimento de 32bits. A Figura 6.5 mostra as etapas de processamento do código desenvolvido.

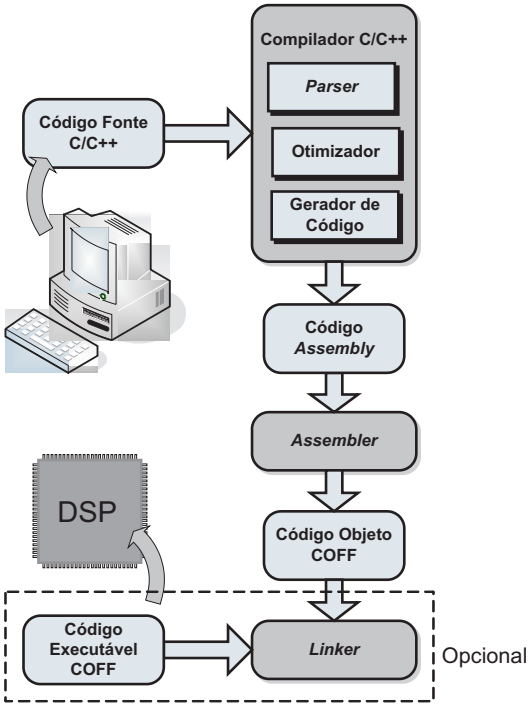


Figura 6.5: Etapas de implementação do programa desde o código fonte até o código executável.

a) Compilador: Realiza as tarefas de leitura do código fonte, verificação de sintaxe e produz um arquivo intermediário (*parser*). Após otimiza o código (otimizador) e finalmente, gera o código em *assembly*;

b) Assembler: Cria um código em linguagem de máquina que possui códigos em *assembly*, macros e diretivas;

c) Linker: Combina os arquivos objeto vindos do Assembler e cria um único módulo que pode ser alocado na memória e executado;

6.2 Módulo Retificador-Inversor

Na Figura 6.1, a tensão V_{DC} é obtida pela retificação da forma de onda CA de entrada da fonte. Esta tensão é processada pelo inversor de modo a sintetizar uma forma de onda modulada por largura de pulso. As etapas de retificação e inversão são realizadas pelo módulo retificador-inversor a quatro braços B6CI-E1IF-B6C da Semikron®. Este módulo também inclui circuitos de acionamento para o controle das chaves eletrônicas IGBT e monitoramento da sobrecorrente, para fins de proteção.

6.2.1 Retificador

O módulo retificador tem a função de converter a tensão alternada da rede elétrica em um valor CC. Para a aplicação em questão, o retificador é alimentado pela rede trifásica e uma topologia baseada em três braços de diodos foi usada. A tensão de saída é filtrada por um capacitor de valor 4800μF, a fim de reduzir a oscilação associada à etapa de retificação. Devido ao alto valor da capacitância e ao consumo pela carga, os picos de corrente na entrada do retificador podem ser elevados, o que implica no uso de algum filtro na entrada do retificador. Neste caso, um filtro passivo formado pelos indutores na Figura 6.6 foi utilizado.

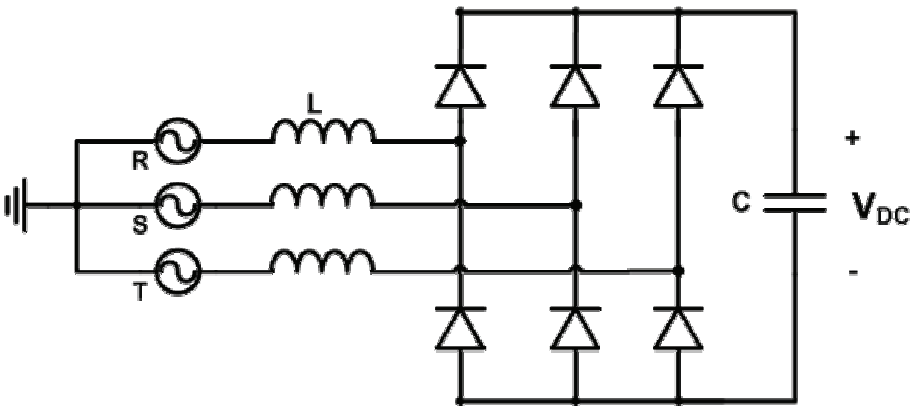


Figura 6.6: Retificador trifásico não-controlado

6.2.2 Inversor

Inversores de tensão sintetizam formas de onda moduladas por largura de pulso (PWM) a partir da comutação de chaves eletrônicas. Deste modo, a forma de onda modulada possui uma amplitude fixa igual ao valor do barramento CC. Porém a duração do pulso Δt da forma de onda PWM é variável dentro de um intervalo de tempo $T_{sw} = 1/f_{sw}$ fixo, como mostrado na Figura 6.7.

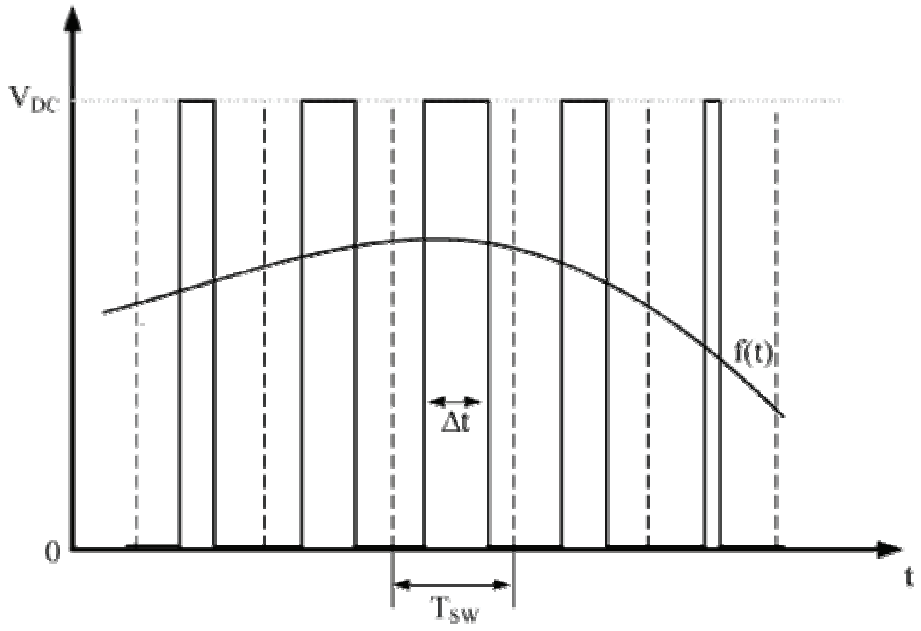


Figura 6.7: Forma de onda moduladora PWM e função f(t) modulada

O inversor utilizado é baseado em chaves semicondutoras IGBT SKM50GB123D de 50A. A Tabela 6.2 mostra os valores de tempo para operação em comutação. Os valores são válidos para um barramento CC de 600V, tensão no gatilho (V_{GE}) de ±15V e corrente nominal no coletor de 40A.

Na Figura 6.8 são mostradas as formas de onda para a tensão no gatilho que controla o acionamento do IGBT, a tensão entre coletor e emissor e a corrente no coletor. Pode-se verificar que a largura de pulso mínima para a chave utilizada, neste caso, é de aproximadamente 0,6us, ou seja, pulsos PWM com largura inferior a 0,6us não serão reproduzidos pelo IGBT. No caso da Fonte de Potência CA este desempenho é satisfatório, uma vez que o barramento CC e os níveis de corrente são ainda inferiores aos apresentados anteriormente nesta seção e a freqüência de comutação é de 12kHz.

Tabela 6.2 – Especificações para comutação

Símbolo	Valor
td(on)	70ns
tr(on)	60ns
ton	td(on)+tr(on)=130ns
td(off)	400ns
tf	45ns
toff	td(off)+toff=445ns

Fonte: Semikron®

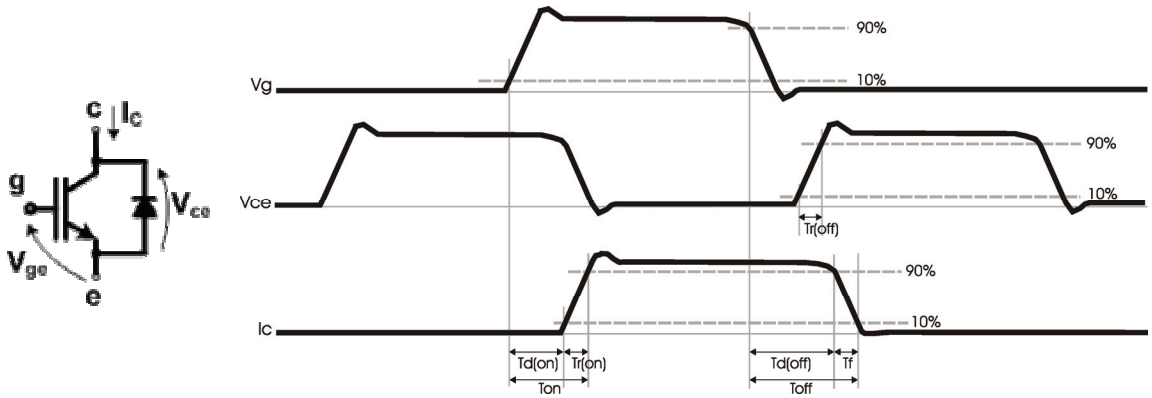


Figura 6.8: Formas de onda de um IGBT na comutação

6.2.3 Circuito de acionamento dos IGBT's

O acionamento dos IGBT's é feito por um circuito eletrônico, cuja função é adequar os níveis de tensão necessários ao acionamento da chave eletrônica. O módulo retificador-inversor utilizado possui o circuito de acionamento SKHI22BH4 da Semikron®, que são compostos basicamente de um dispositivo ASIC fixado numa placa de circuito impresso, cujo diagrama de blocos está representado na Figura 6.9. Além da adequação dos níveis de tensão para o acionamento dos IGBT's, o circuito de acionamento possui uma proteção de sobre-corrente e de falha na alimentação, além da possibilidade de geração de quatro valores diferentes de tempo-morto para as chaves. A proteção de sobrecorrente é feita com a utilização de um sensor entre o coletor e o emissor do IGBT enquanto que a proteção de subtensão consiste na medição da tensão de alimentação do ASIC e da placa. Outra característica é o isolamento feito por transformadores de pulso, entre a entrada (das placas de

interface com o DSP) e a saída (para o gatilho do IGBT). A Tabela 6.3 mostra algumas características elétricas do circuito de acionamento.

Tabela 6.3 – Características elétricas e de tempo do circuito de acionamento

Símbolo	Valor
Tempo morto	1,3 – 2,3 – 3,3, 4,3 μs
freq. máxima	50kHz
Tensão de alimentação (Vs)	15V
corrente consumida	80mA
Entradas TOP/BOT	0 – 15V
Tensões de acionamento (V_{on}/V_{off})	15/-7V

Fonte: Semikron®

6.3 Filtro LC

A topologia de filtro LC é atualmente a mais utilizada nas mais diversas aplicações de eletrônica de potência. Esta topologia é baseada em uma configuração indutor-capacitor como mostrado na Figura 6.1. Embora existam técnicas de projeto dos elementos capacitivo e indutivo para a minimização da energia reativa circulante e minimização da THD, como em Michels *et al* [24], neste trabalho, a escolha dos valores do filtro LC é realizada com base na banda passante necessária a cada teste. Esta foi a solução encontrada uma vez que limitações de instrumentação e das chaves eletrônicas, não permitiram um aumento da freqüência de comutação para ordem mais elevadas (dezenas de quilohertz). O filtro LC utilizado foi composto de um capacitor de 60uF e de um indutor de 1mH. A Figura 6.10 mostra a forma PWM e a forma de onda modulada obtidas pelo uso de um algoritmo de modulação *space vector*. Neste resultado, foi simulado um inversor comutando numa freqüência f_{sw}=12kHz e o PWM resultante possui uma componente fundamental de freqüência f_1=60Hz, o que resulta no *espectro* harmônico da Figura 6.11. Neste gráfico, o eixo das abscissas é formado pela freqüência normalizada $m_s = f/f_1$ e com o valor "1" representando a freqüência de 60Hz, e com grupos de freqüências nos múltiplos da freqüência f_{sw}. Assim, para o grupo de

freqüências próximas de ms=200, as harmônicas estão em torno da freqüência f_{sw}=12kHz (f=12kHz,12kHz/60 $\rightarrow$ 200), para ms=400, em torno de $2f_{sw}$ e assim sucessivamente.

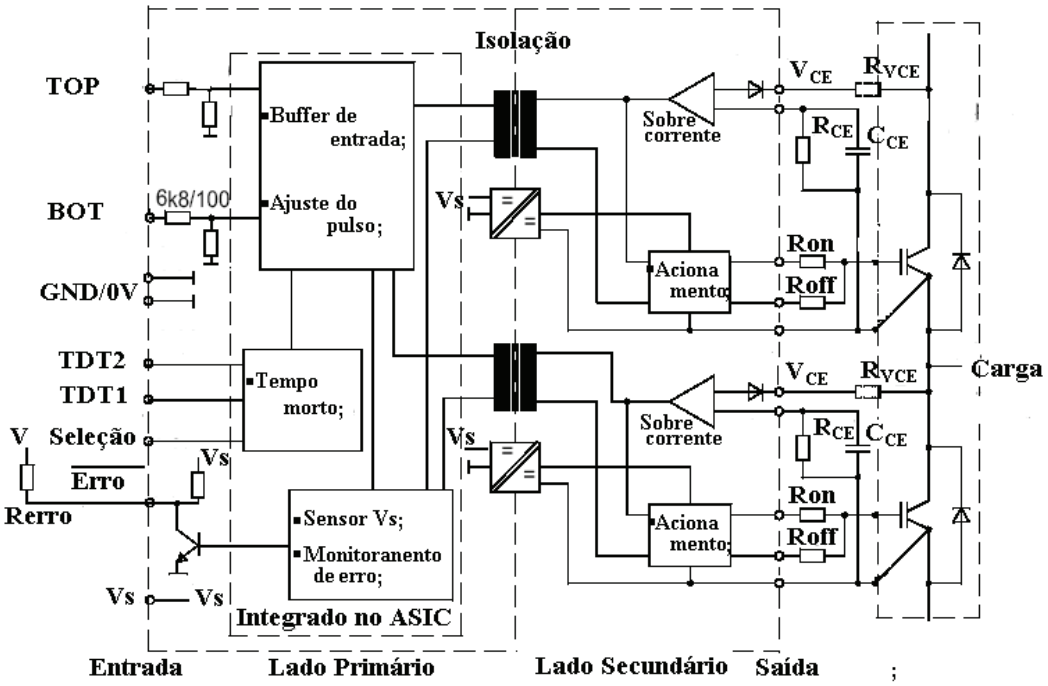


Figura 6.9: Diagrama de blocos do circuito de acionamento SKHI22BH4 (Fonte: Semikron®)

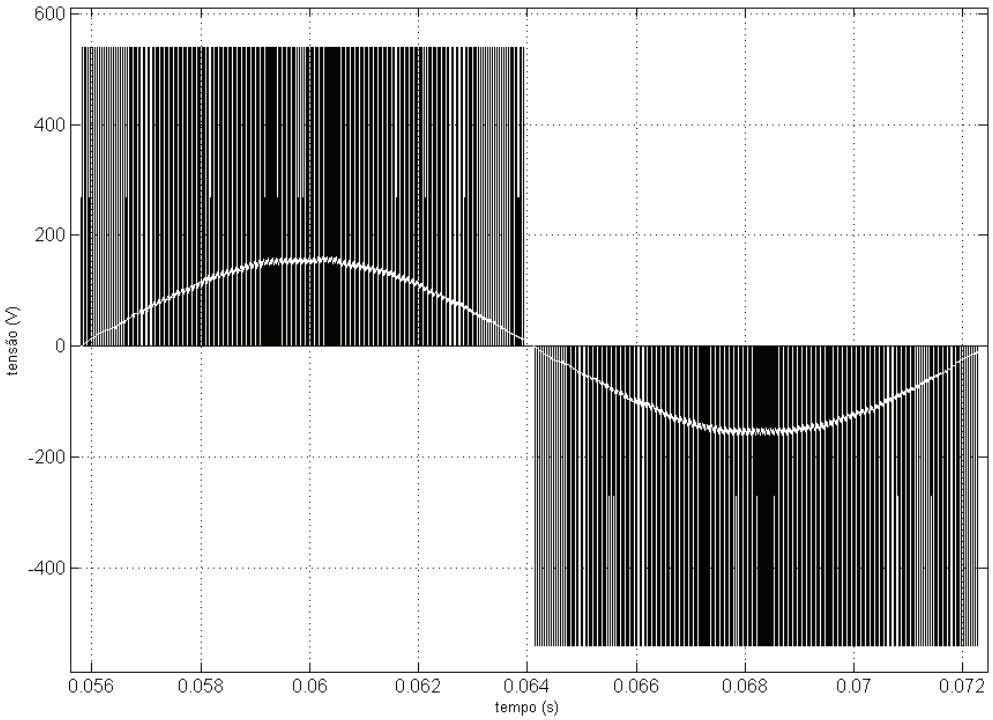


Figura 6.10: PWM de 12kHz e forma de onda modulada

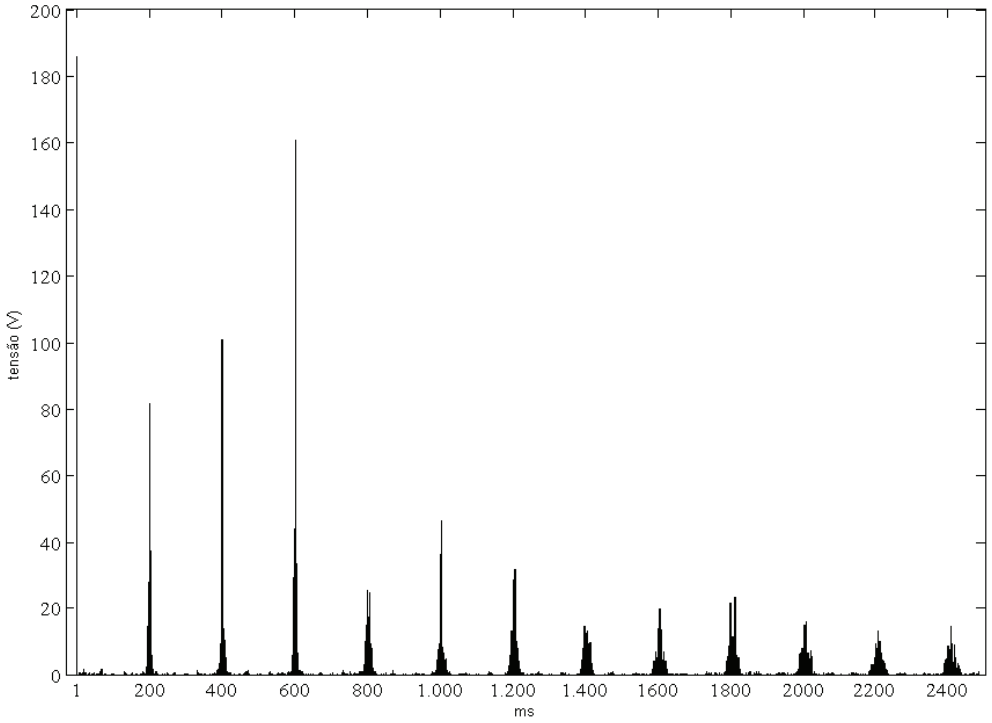


Figura 6.11: Espectro harmônico normalizado

6.4 Aquisição das Medidas para Controle da Fonte de Potência CA

O circuito de aquisição das tensões de saída produzidas pela Fonte de Potência CA é baseado em sensores isolados galvanicamente. A Figura 6.12 mostra o circuito utilizado.

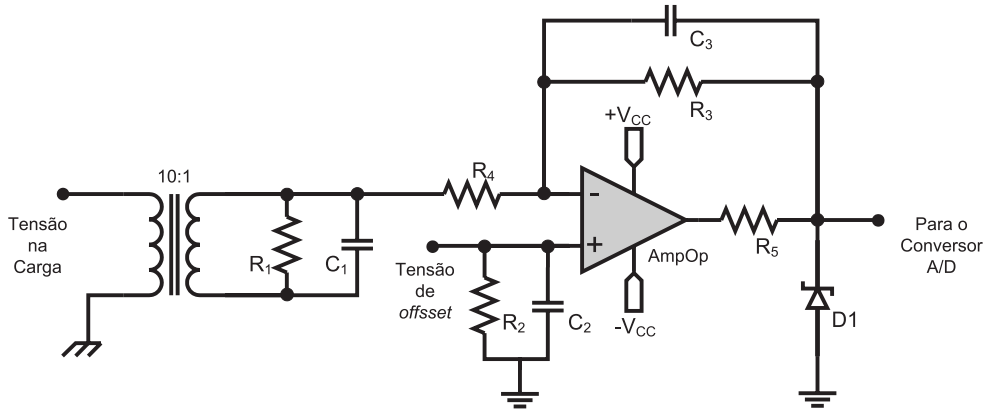


Figura 6.12: Circuito de isolação e condicionamento de sinais

O projeto envolveu a determinação dos valores dos componentes R_1 e C_1, R_3 e C_3 para a locação das freqüências de corte do circuito, e dos resistores R_3 e R_4, para o ganho final do sensor.

Para a medição do barramento CC, um circuito baseado num transdutor de efeito Hall da Lem® (ver Figura 6.13) foi utilizado. O uso de um transdutor deste tipo garante uma medição isolada para tensões da ordem de unidades de kV, sendo bastante adequado para as medições em baixa freqüência e valores DC. A operação básica consiste em fazer circular uma corrente elétrica pelo resistor R_P, sendo que a saída é uma corrente proporcional a corrente de entrada que circula sobre o resistor R_M, cuja tensão é usada para a medição.

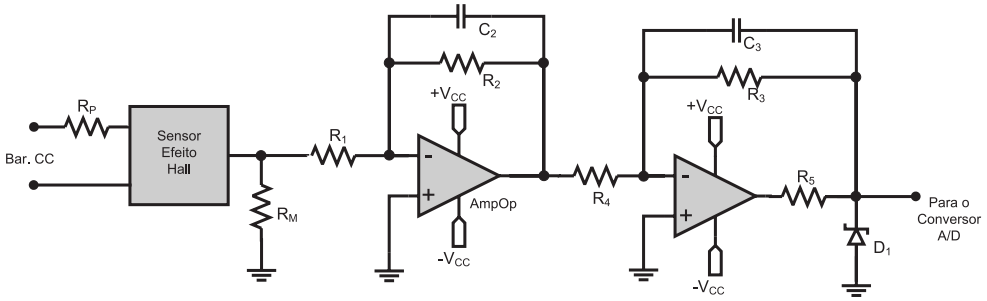


Figura 6.13: Circuito para medição do Barramento CC

6.5 Interface entre o DSP e o Circuito de Acionamento do Inversor

Na Figura 6.1, os sinais PWM gerados pelo DSP são enviados para a placa de interface por meio de fibras óticas. Esta placa recebe os sinais óticos na entrada e os converte em sinais elétricos com níveis de 0-15V. Para a ativação da placa de interface, um sinal de "Ativação" é utilizado. Deste modo, a qualquer momento o DSP pode, via este sinal, inibir os sinais para o inversor.

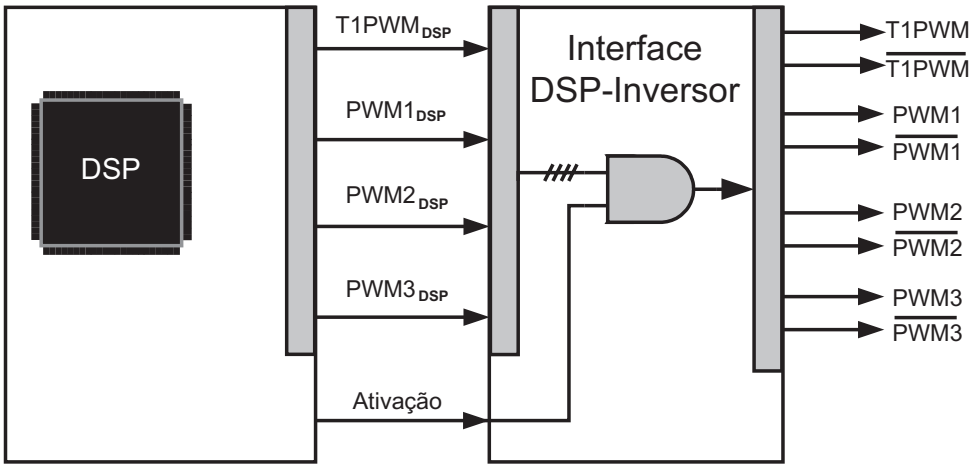


Figura 6.14: Envio dos sinais PWM do DSP e geração das saídas para as quatro pernas

6.6 Sumário

Este capítulo descreve as diversas etapas de implementação do protótipo. É feita uma descrição das diversas etapas que implementam o sistema, composto basicamente de um filtro LC, um módulo retificador-inversor e de um DSP para o acionamento das chaves semicondutoras do inversor. Uma etapa crítica na implementação é relacionada à qualidade na medição das tensões. Neste caso particular, verificou-se que o uso de um amplificador isolador par a obtenção de medidas diferenciais, resultou em uma menor qualidade da forma de onda adquirida se comparada aquela obtido com o uso de transformadores. O uso de transformadores, além de prover isolação galvânica, são lineares, mais confiáveis e não necessitam de implementação de fontes isoladas, como no caso dos amplificadores.

CAPÍTULO 7

RESULTADOS EXPERIMENTAIS

A fim de validar a lei de controle, resultados foram obtidos tanto em simulação quanto experimentalmente a partir da implementação do algoritmo RMRAC no controlador DSP TMS320F2812 da Texas Instruments®.

Os parâmetros da planta e os dados de simulação e implementação no DSP são mostrados nas Tabela 5.1 e Tabela 5.2.

O modelo de referência dado por (4.3) também pode ser representado por

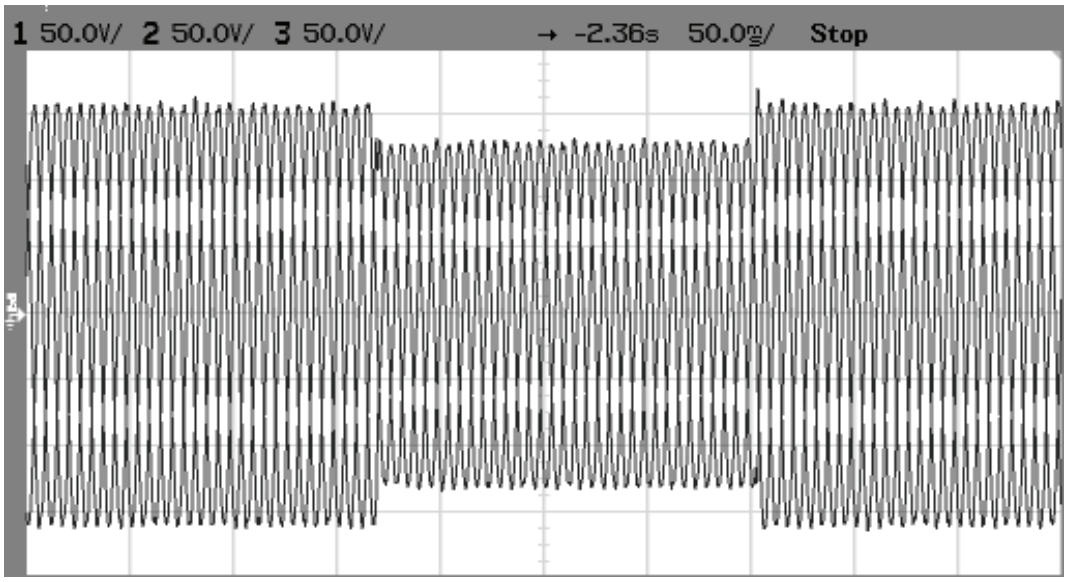
$$\frac{y_m(s)}{r(s)} = \frac{{\omega_m}^2}{s^2 + 2\zeta_m\omega_m + {\omega_m}^2} \tag{7.1}$$

onde $\zeta_m = 0{,}2$ e a freqüência ω_m foi escolhido para ter um valor 20% maior que o da planta ou $\omega_m = \frac{1.2}{\sqrt{LC}} = 780\text{Hz}$. Estes valores foram escolhidos de modo que o modelo de referência apresentasse uma resposta dinâmica semelhante a da planta em toda a faixa de freqüências de interesse.

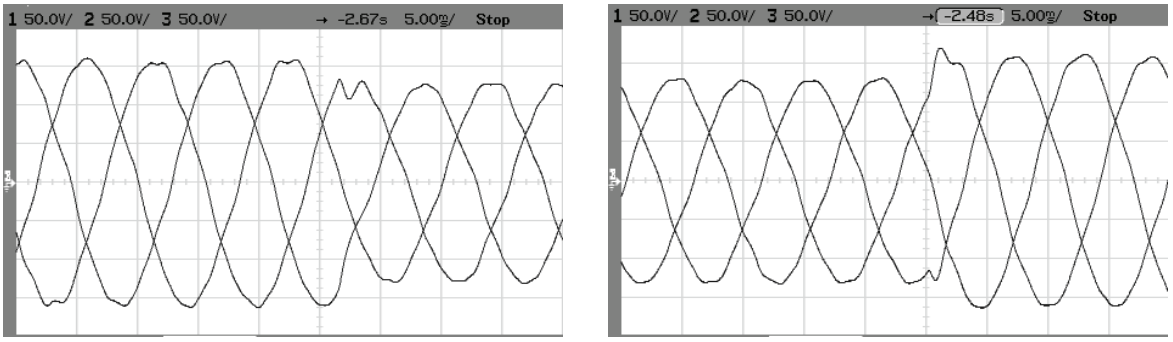
7.1 Resultados Experimentais

Algumas formas de onda foram obtidas a fim de demonstrar a capacidade de geração da Fonte de Potência CA quanto à geração de harmônicos e variações na amplitude.

A Figura 7.1 mostra o resultado de um afundamento na tensão das três fases que estão sendo geradas, com uma carga trifásica equilibrada de 48Ω.



(a)



(b)

Figura 7.1: Afundamento e Elevação da tensão de saída (a) e detalhes dos transitórios (b)
escalas horizontal de 5ms/div e vertical de 50V/div

Para simular a capacidade de geração de transitórios de curta duração, a Figura 7.2 mostra uma forma de onda na qual há uma variação periódica da tensão em uma fase durante um ciclo da fundamental de 60Hz. Formas de onda com harmônicas são muito úteis nos mais diversos ensaios. A Figura 7.3 mostra a geração da referência $r(t)=0,5\sin(\omega t)+0,3\sin(5\omega t)$.

A fim de demonstrar a capacidade de rastreamento da tensão de saída em relação à saída do modelo de referência, algumas formas de onda foram armazenadas na memória do DSP para posterior visualização, verificando-se assim o desempenho do controlador. Na Figura 7.4 a forma de onda da tensão de saída, obtida na Figura 7.3 é mostrada juntamente com a saída do modelo de referência.

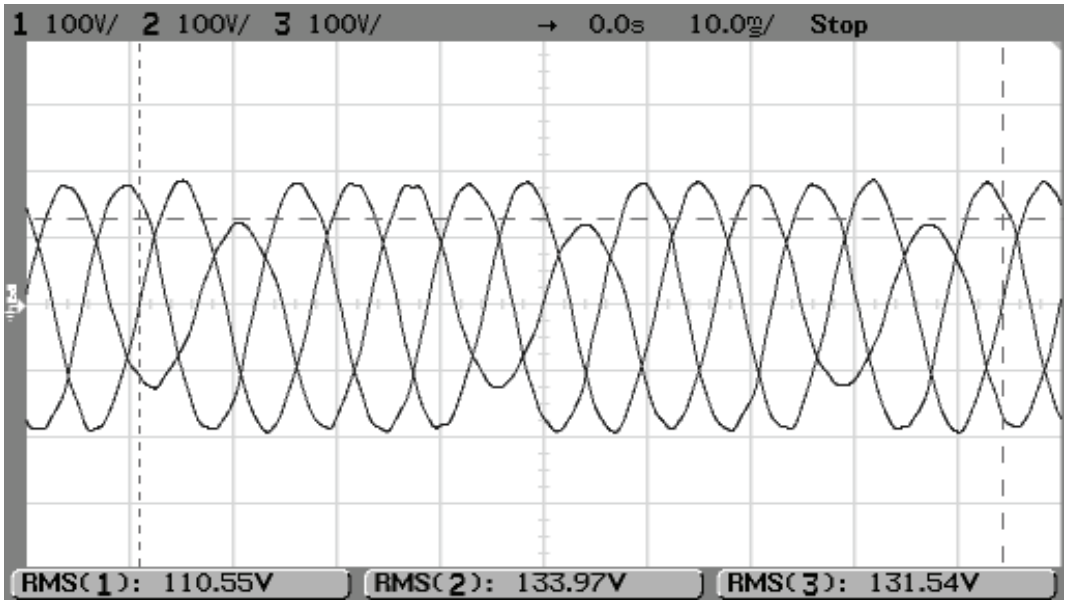


Figura 7.2: Variação periódica da tensão de saída com
FD=41,3% e THD=4,01%. Escala vertical de 100V/div

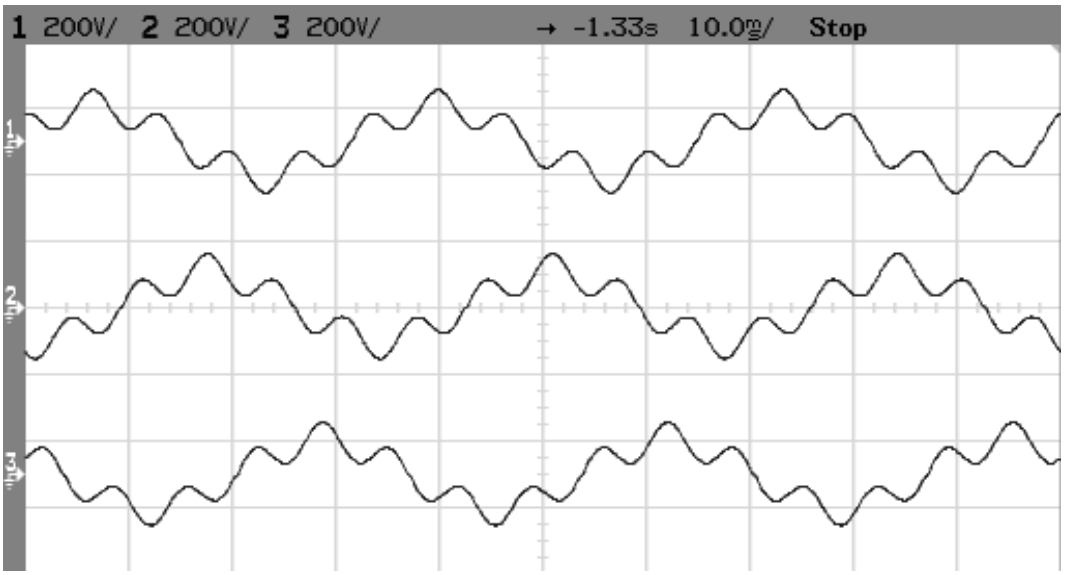


Figura 7.3: Fundamental em 60Hz e 5º harmônica. Escalas horizontal de 10ms/div e vertical de 200V/div

A fim de mostrar a capacidade de geração com carga desequilibrada, a Figura 7.5 mostra a obtenção de uma forma de onda trifásica equilibrada obtida para uma carga trifásica desequilibrada de 48Ω numa das fases e 24 Ω nas outras. Por fim, uma varredura AC foi realizada de modo a gerar diferentes freqüências. Foi feita uma variação lenta na freqüência e capturado alguns períodos para fins de demonstração. A Figura 7.6 mostra o resultado.

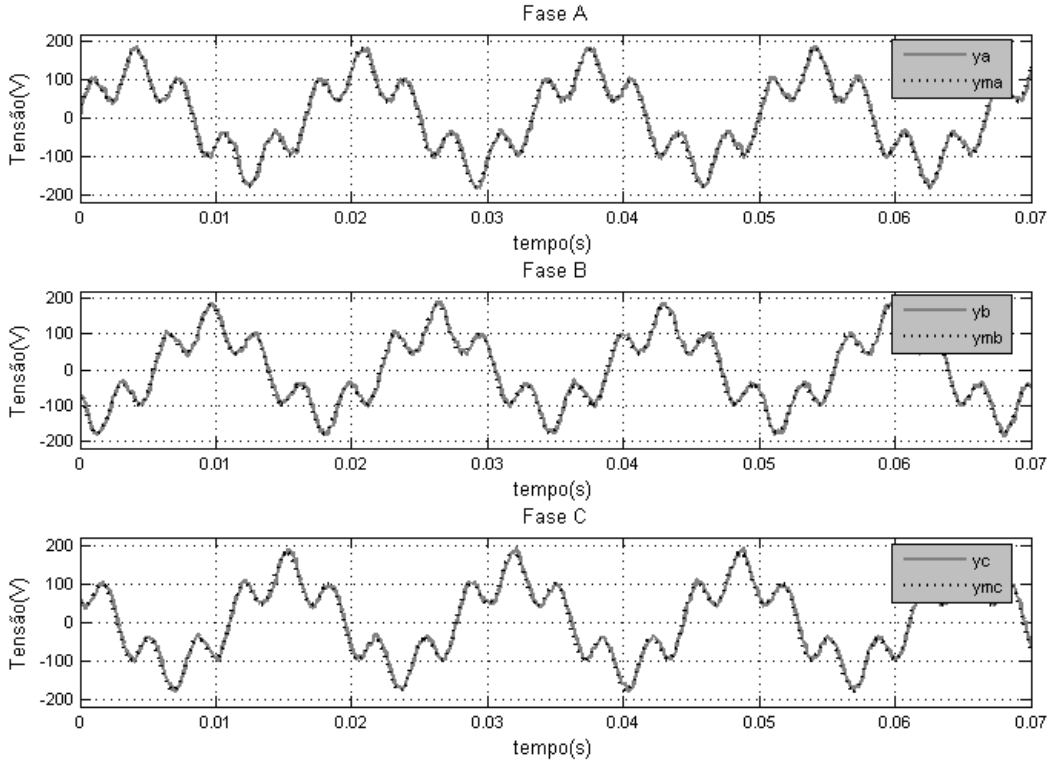


Figura 7.4: Saída da planta (y) e saída do modelo de referência (y_m). Escala vertical de 100V/div

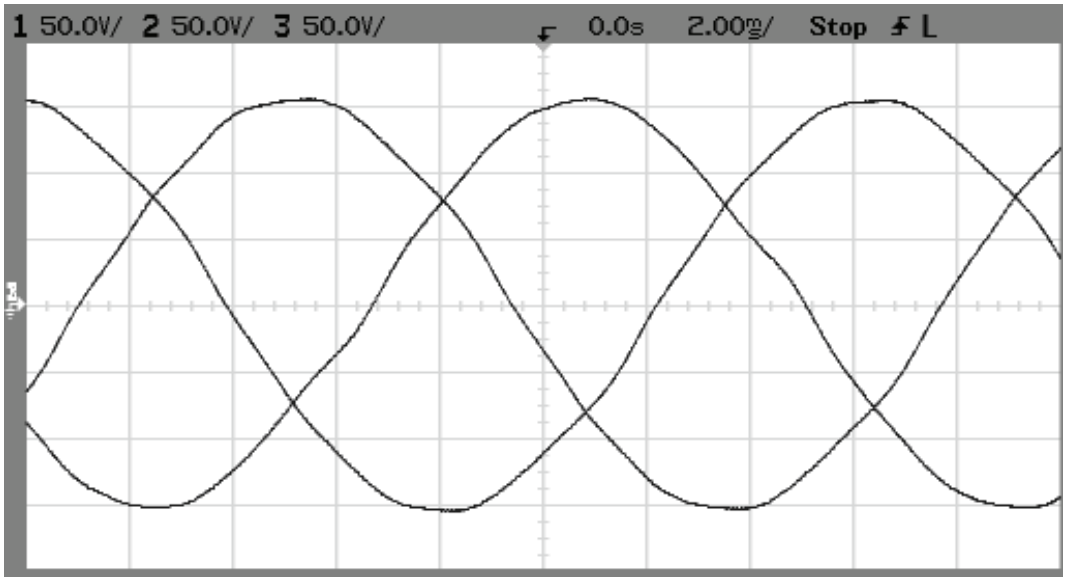


Figura 7.5: Geração trifásica equilibrada com uma das fases 50% mais carregada.

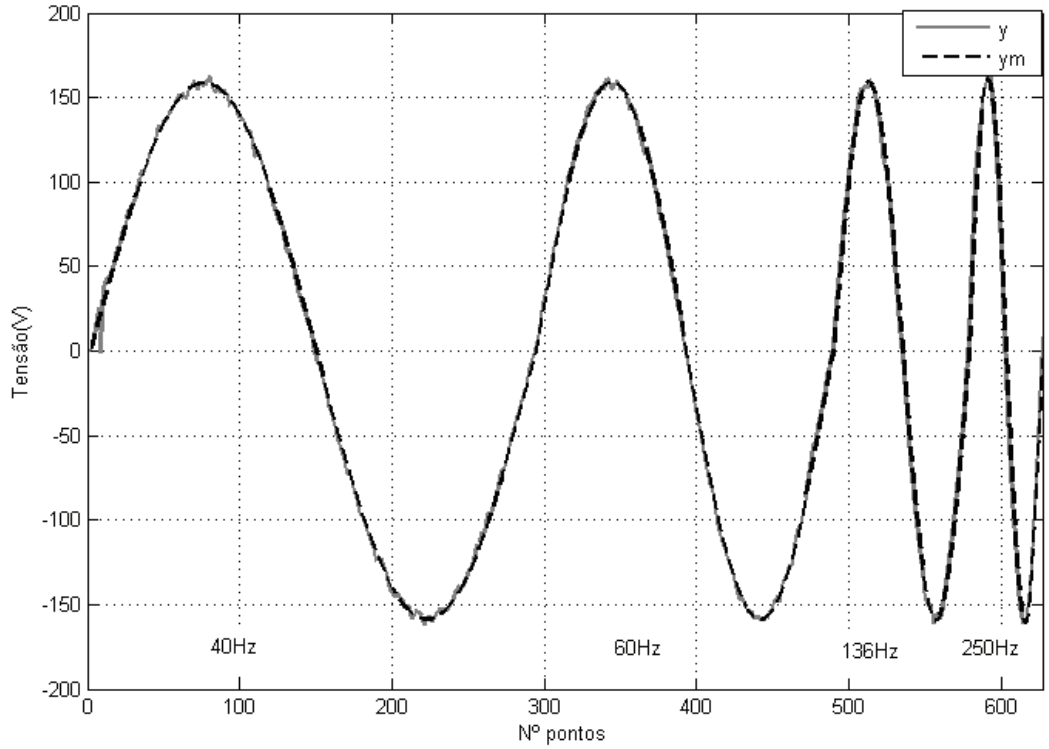


Figura 7.6: Varredura AC

7.2 Sumário

Este capítulo trata da obtenção de resultados práticos que podem ser úteis em tarefas de ensaios. Primeiramente os parâmetros da planta, modelo de referência e controlador foram mostrados, em seguida partiu-se para a obtenção das formas de onda para mostrar a capacidade de geração da Fonte de Potência CA proposta, incluindo formas de onda com variações em amplitude e injeção de harmônicas.

CAPÍTULO 8

CONCLUSÕES

Este trabalho apresenta uma contribuição ao estudo de Fontes de Potência CA. Como visto nos capítulos anteriores, este trabalho tem um caráter experimental uma vez que tratou de assuntos relativos à implementação de um protótipo e aplicações das Fontes de Potência CA comerciais. O Capítulo 2 mostra algumas normas, em cujas especificações de teste, as Fontes de Potência CA podem ser utilizadas. Foram poucas as referências acerca de topologias existentes e tecnologia associada a esta aplicação. Os dados obtidos apontam para uma forte tendência na implementação de equipamentos utilizando um inversor de tensão para a etapa de amplificação [25]. No Capítulo 3 é feita a modelagem da planta que é baseada nos trabalhos de Carati *et al* [13]. Também é apresentada a técnica de implementação da modulação PWM *space vector*, para o inversor a quatro pernas, sendo que esta topologia de inversor que já foi utilizada na implementação de filtros ativos [26] e UPS's vide [21], e outras aplicações [27]. No Capítulo 4 é apresentado o controlador utilizado. Entretanto, devido à limitação de processamento da plataforma utilizada para controle, a matriz de covariância **P** em (4.22) foi considerada constante, levando a um estimador do tipo Gradiente. Este fato leva a uma redução de desempenho na adaptação paramétrica do algoritmo em relação ao algoritmo RLS, onde a matriz **P** é atualizada, vide Carati *et al* [28] e [13]. No Capítulo 5 foram obtidos os resultados por simulação. Estes resultados mostraram a capacidade de geração de formas de onda trifásica desequilibradas, que é um dos objetivos deste trabalho, para a simulação de problemas que podem ocorrer na rede elétrica. Entretanto, problemas relativos à banda passante do filtro LC começaram a ser observados. Constatou-se que quanto maior a banda passante do filtro, maiores eram as amplitudes das componentes harmônicas oriundas da comutação do inversor, aumentando a THD da forma de onda de saída. No Capítulo 6 foi brevemente descrito o protótipo implementado. Um dos problemas enfrentados na execução do protótipo diz respeito à tecnologia envolvida na instrumentação, pois durante os testes do equipamento foram vários os problemas associados a ruídos irradiados e principalmente os conduzidos. Estes problemas causaram por várias vezes mal-funcionamento no processador e os sinais PWM na saída do deste, foram corrompidos por ruídos oriundos tanto da comutação das chaves eletrônicas, quanto dos campos

eletromagnéticos causados pelos indutores do filtro de saída. Isto ocorreu mesmo com a utilização de transformadores para a medição das tensões e de fibras óticas para o isolamento ente o DSP e o circuito de acionamento do inversor. Não foi possível a obtenção de resultados com amplitudes maiores que as obtidas. Melhorias no desempenho do sistema como um todo podem ser alcançadas com o uso de outra topologia de inversor e com o uso de filtros melhor sintonizados, ou mesmo de maior ordem. Quanto à questão do ruído, a solução é utilizar um isolamento da fonte de ruído (o inversor) e também dos circuitos eletrônicos de controle e de instrumentação. Além dessas melhorias, como sugestão para trabalhos futuros pode-se citar:

1. Utilização de uma topologia de inversor multiníveis, o que pode reduzir a THD do sinal modulado em PWM;
2. Implementação de um sistema supervisório em um microcomputador PC;
3. Investigação do comportamento de outras técnicas de controle frente às diversas condições de variação de freqüência e de carga, inclusive não-lineares;

A escolha de uma topologia de inversor adequada é a escolha mais importante quando da realização de trabalho futuros, um exemplo seria o uso conversores multiníveis, que são capazes de sintetizar formas de onda moduladas em PWM com conteúdo harmônico menor que o tradicional PWM três níveis para os mesmos valores de filtro de saída.

CAPÍTULO 9

REFERÊNCIAS BIBLIOGRÁFICAS

[1] SAKTHIVEL, K.N., DAS, S.K., KINI, K.R. **Importance of Quality AC Power Distribution and Understanding of EMC Standards IEC 61000-3-2, IEC 61000-3-3 & IEC 61000-3-11**. In: 8th International Conference on Electromagnetic Interference and Compatibility, p. 523-430, December, 2003.

[2] HAEFELY TECHNOLOGY. **Power Frequency Application Note 06, Variation of Power Immunity to Test to IEC 1000-4-28**. Switzerland.

[3] HAEFELY TECHNOLOGY. **Power Frequency Application Note 03, Voltage Dip, Interruption and Variation Immunity Testing to IEC 1000-4-11**. Switzerland.

[4] CHAND, S., CHAWLA, K. **EMC Evaluation and Analysis of UPS.** In: Proceedings of the International on Electromagnetic Interference and Compatibility, p. 37-42., 2002.

[5] JAEGER, D. **Generating and measuring 400Hz ac power for Military and Avionics Instrument manufacturers can be quiet challenging**. Hewlett-Packard Company.

[6] **Environmental Considerations in Development of Mobile Agricultural Electrical/Electronic Components**, ANSI/ASAE EP455, July 1991.

[7] CHEN, T. H., HUANG, K. C., LIAW, C. M. Random Vibration Test Control of an Inverter-Fed Electrodynamic Shaker. **IEEE Transactions on Industrial Electronics,** v. 49, n. 3, p.587-594, 2002.

[8] DELLA FLORA, L., GRÜNDLING, H.A. **Vibration acceleration control of an AC power source fed electrodynamic shaker**. In: 2005 IEEE 36th Conference on Power Electronics Specialists, p. 1175-1181, June, 2005.

[9] LOW K. S. A DSP-Based Single-Phase AC Power-Source. **IEEE Transactions on Industrial Electronics**. v. 46, n. 5, p. 936-941, 1999.

[10] BROECK, H. van der, LÜRKENS, P. **Programable AC power source**. 5th European Conference on Power Electronics and Application, Brighton. v. 13, p. 255-260, September, 1993.

[11] TZOU et al. High-Performance Programmable AC Power Source with Low Harmonic Distortion Using DSP-Based Repetitive Control Technique. **IEEE Transactions on Power Electronics**., v. 12, n. 4, p. 715-725, 1997.

[12] LOW K. S. A DSP-Based Variable AC Power Source. **IEEE Transactions on Instrumentation and Measurement**. v. 7, n. 4, p. 992-996, 1998.

[13] CARATI, E. G., RICHTER, C. M., GRÜNDLING, H. A. **A Three-Phase AC Power Source using Robust Model Reference Adaptive Control**. In: 9th Conference on Decision and Control, 2000, Sydnei, v. 4, p. 4078-4083, 2000.

[14] RICHTER, C.M. et al. **A Three-Phase AC Power Source using Multivariable Repetitive Robust Model Reference Adaptive Control**. In: Proceedings of the American Control Conference, Denver, p. 2300-2305, June, 2003.

[15] CALIFORNIA INSTRUMENTS. **Application Note 119 – Harmonics and Flicker Testing** IEC-1000-3-2 & IEC-1000-3-3. 1998.

[16] CALIFORNIA INSTRUMENTS. **PGUI32, P-Series, RP-Series User Manual – Revision B**. 2000.

[17] CALIFORNIA INSTRUMENTS. **CGUI32 / CGUI32 Training Manual – Revision 1.1**. 1998.

[18] ZHANG, R., BOROYEVICH, D., PRASAD, V.H. **A three-phase inverter with a neutral leg with space vector modulation**. In: IEEE Applied Power Electronics Conference. Atlanta, v. 2, p. 864-870, February, 1997.

[19] QUINN, C.A., MOHAN, N. **Active filtering of harmonic currents in three-phase, four-wire systems with three-phase and single-phase non-linear loads**. In: Applied Power Electronics Conference and Exposition/APEC'92, Boston. v.2, p. 829-835, February, 1992.

[20] BOWES, S.R. **New sinusoidal pulsewidth modulated inverter.** Proceedings on Insdustrial Electronics. v. 122, p. 1279-1285, 1975.

[21] BOTTERÓN, F. et al. **New limiting algorithms for space-vector modulated three-phase four-leg voltage source inverters**. IEE Proceedings on Electric Power Applications, v. 150, n. 6, 2003.

[22] JUNG, S.-L. et al. **DSP-Based Multiple-Loop Control Strategy for Single-Phase Inverters Used in AC Power Sources**. In: 28th IEEE Power Electronics Specialists Conference, v.1, p. 706–712, June, 1997.

[23] IOANNOU, P.A., TSAKALIS K., A Robust Direct Adaptive Control, **IEEE Transactions on Automatic Control**, vol. AC-3, no.11, p 1033-1044, 1986.

[24] MICHELS et al. Metodologia Generalizada de Projeto de Filtros de Saída de Segunda Ordem para Inversores de Tensão com Modulação PWM Digital SOBRAEP;

[25] STEFANELLO, M., CARATI, E. G. **Environment for Random and Sinusoidal Vibration Test Control of an Inverter-Fed Electrodyamic Shaker**. In: IEEE Intenational Symposium on Industrial Electronics. Rio de Janeiro. v. 2, p. 1093-1098. June, 2003.

[26] de CAMARGO, R.F., PINHEIRO, H. **New synchronization method for three-phase four-wire PWM converters under unbalanced and harmonics in the grid voltage**. In: Proceedings on EuroPES'05. 2005. CD-ROM.

[27] EL-BARBARI, S., HOFMAM, W. **Digital control of a four-leg inverter for standalone photovoltaic systems with unbalanced load**. In: 26th Conference of the IEEE Industrial Electronics Society, IECON, Nagoya. v. 1, p. 729-734, 2000.

[28] CARATI *et al*,.**Adaptive robust DSP-based single phase AC power source**. In: Conference on Control Applications. p. 24-28, September, 2001.

[29] de CAMARGO, R.F., PEREIRA, A.T., PINHEIRO, H. **New synchronization method for three-phase three-wire PWM converters under unbalanced and harmonics in the grid voltage**. In: IEEE Power Electronics Specialists Conference. p. 506-512, June, 2005;

[30] WELCH, G., BISHOP, G. **An Introduction to the Kalman Filter**. Technical Report 95-041. UNIVERSITY OF NORTH CAROLINA - Department of Computer Science, 2002.

APÊNDICE A

TESTE DE UM ALGORITMO DE DETECÇÃO DE FASE PARA UM FILTRO ATIVO PARALELO

A Fonte de Potência CA foi utilizada na geração de formas de onda diversas, para o teste de um algoritmo de sincronização de um Filtro Ativo. Estes algoritmos consistem em obter informações sobre a fase da forma de onda para a geração de referências. Algoritmos como estes, também podem ser usados em UPS's. O trabalho que apresenta a técnica de sincronização ensaiada pode ser encontrada em Camargo *et al* [26] e [29]. O diagrama de blocos do circuito para ensaio é mostrado na figura abaixo.

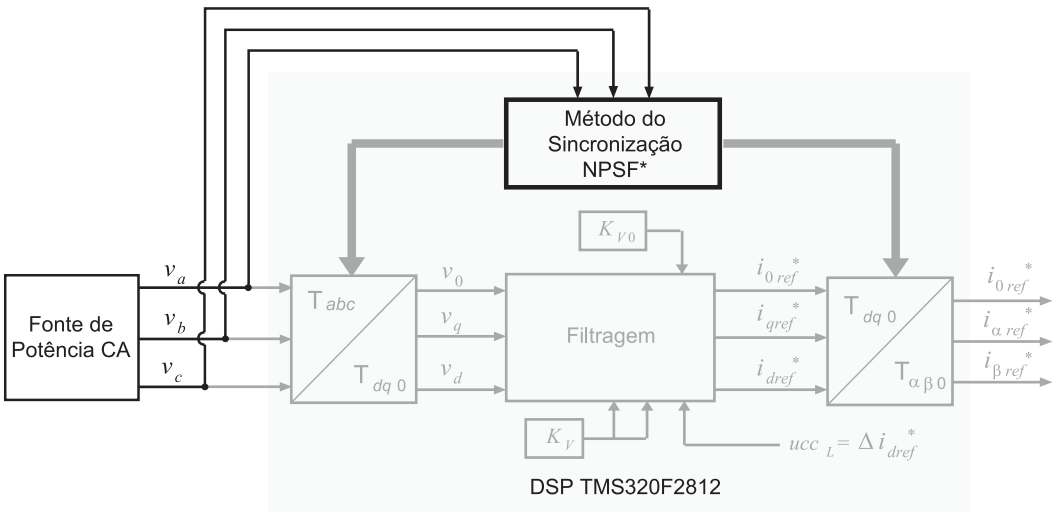


Figura A.1 – Circuito para teste do algoritmo de sincronização NPSF[4]

A seguir é mostrado um resultado experimental, com a forma de onda gerada pela Fonte de Potência CA e o resultado obtido na saída do algoritmo de sincronização.

[4] *Normalized Positive Sequence Frame*

Na Figura A.2 uma forma de onda incluindo a 5º harmônica foi gerada. A Figura A.3 mostra o conteúdo harmônico obtido e a Figura A.4 apresenta o seno e co-seno obtidos pelo algoritmo utilizado (NPSF) e o espectro harmônico destes sinais são mostrados na Figura A.5.

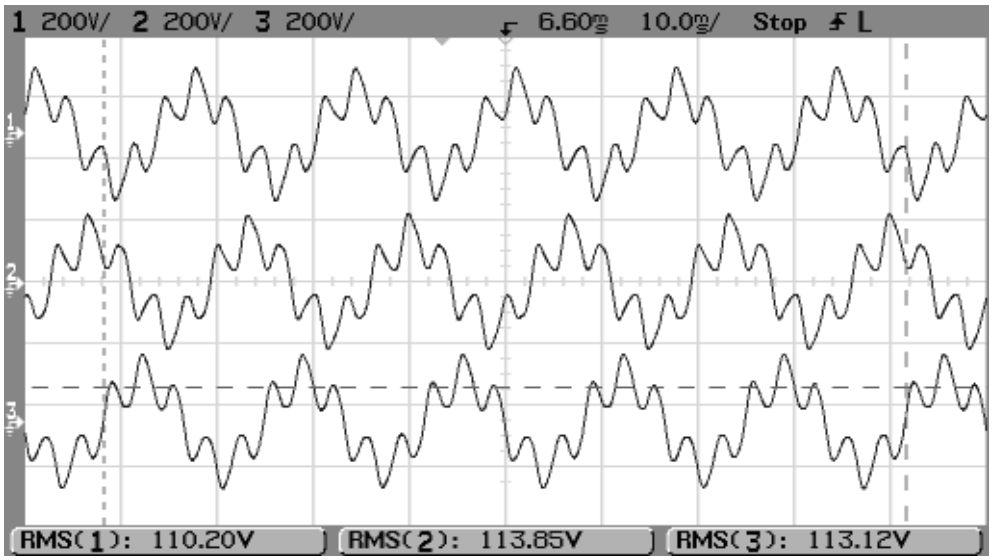


Figura A.2 – Forma de onda fundamental com 60Hertz e 5º harmônica gerada pela Fonte de Potência CA. Escala horizontal 10ms/div e vertical de 200V/div

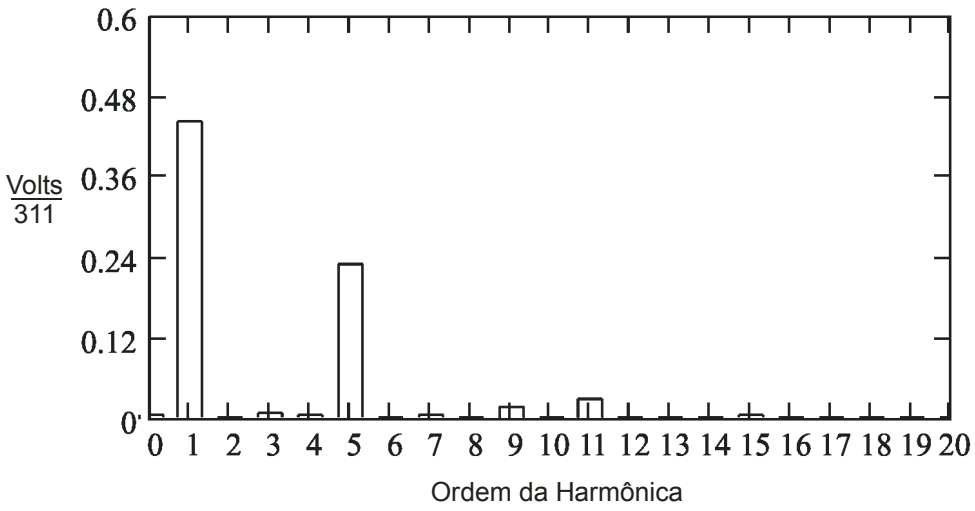


Figura A.3 – FFT da forma de onda da figura Figura A.2

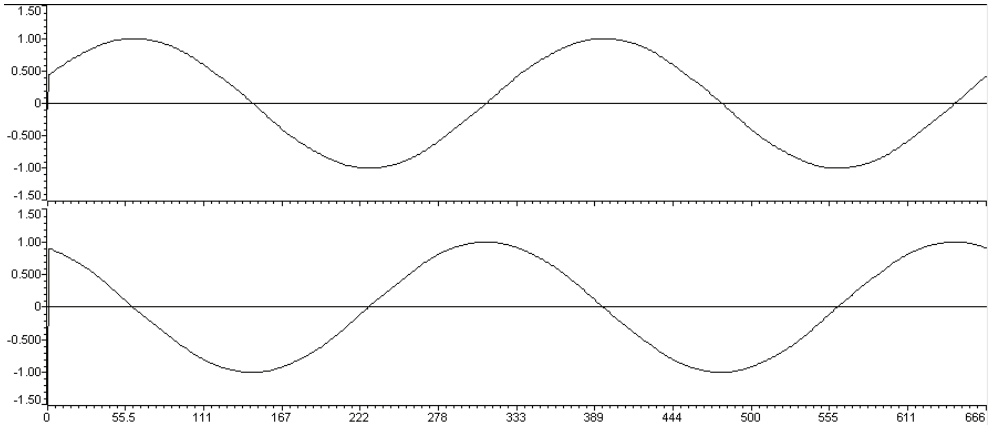


Figura A.4 – Formas de onda seno e cosseno obtidas pelo algoritmo de sincronização NPSF

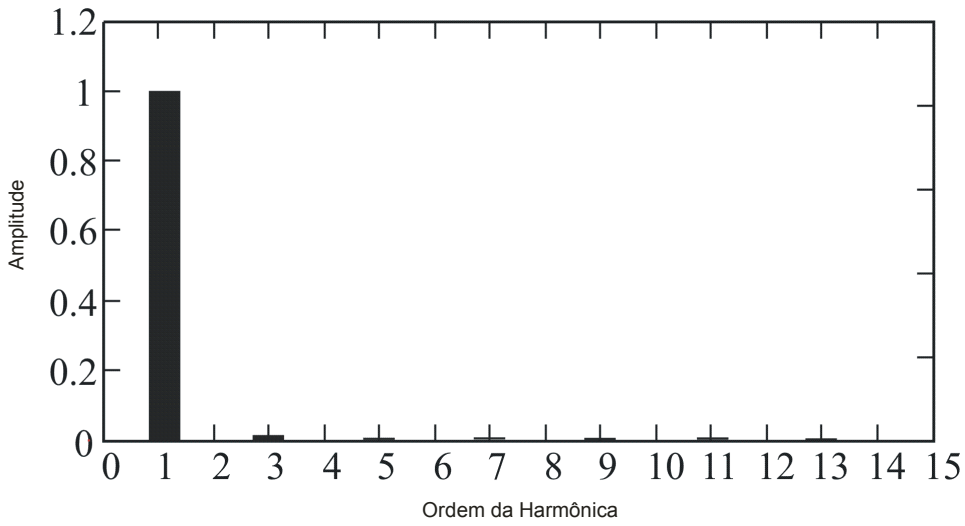


Figura A.5 – Espectro harmônico das formas de onda da Figura A.4

APÊNDICE B

FILTRO DE KALMAN

O filtro de Kalman é um conjunto de equações matemáticas que podem ser implementadas recursivamente na estimação de estados de um processo. É um filtro eficiente e aplicável mesmo na estimação de estados passados, presente e futuros, mesmo que a modelagem do sistema não seja totalmente conhecida.

No caso da aquisição dos sinais do sensor de tensão no protótipo implementado, o ruído é originado a partir das componentes de alta freqüência do sinal PWM, que não foram atenuadas de modo satisfatório pelo filtro LC.

O filtro de Kalman é usado na estimação do estado $x \in \Re^n$, de uma planta discreta governada pela equação diferencial estocástica

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{B.1}$$

com $\mathbf{u} \in \Re^l$ e $\mathbf{w} \in \Re^n$. O estado medido $y_k \in \Re^m$ dado por

$$y_k = \mathbf{H}\mathbf{x}_\mathbf{k} + \mathbf{v}_\mathbf{k} \tag{B.2}$$

Os vetores $\mathbf{w}_k$ e $\mathbf{v}_k$ representam os ruídos intrínsecos ao processo e o de medida respectivamente, e é assumido que são ruídos do tipo branco e com comportamento probabilístico normal. Em (B.1) e (B.2), as matriz **A, B** e **H** são matrizes de dimensão *n* x *n*, *n* x *l* e *m* x *n*, respectivamente.

Os ganhos do filtro de Kalman podem ser calculados vide Welch & Bishop [30] como

$$\begin{aligned} \mathbf{K}_k &= \mathbf{A}\mathbf{P}_k\mathbf{H}^\mathrm{T}\left(\mathbf{H}\mathbf{P}_k\mathbf{H}^\mathrm{T} + \mathbf{R}\right)^{-1}, \\ \mathbf{P}_{k+1} &= \mathbf{A}\mathbf{P}_k\mathbf{A}^\mathrm{T} - \mathbf{K}_k\mathbf{H}\mathbf{P}_k\mathbf{A}^\mathrm{T} + \mathbf{Q} \end{aligned} \tag{B.3}$$

Na prática, as matrizes de covariância dos ruídos do processo **Q** e a matriz de covariância dos ruídos de aquisição **R** variam no tempo, entretanto, neste trabalho elas foram

mantidas fixas baseadas . Para a implementação da equação de Kalman, as matrizes **K** e **P** foram previamente calculadas com base no modelo da planta, e posteriormente mantidas fixas no programa em tempo real. Os estados observados são calculados a partir de

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{A}\mathbf{K}\left(\mathrm{y} - \mathbf{H}\mathbf{x}_k\right) + \mathbf{B}\mathbf{u}_k \tag{B.4}$$

onde as matrizes **A**, **B** e **H** em (I.1) foram obtidas pela discretização da planta contínua dada por

$$\begin{aligned} \begin{bmatrix} \dot{v}_c \\ i_L \end{bmatrix} &= \begin{bmatrix} -1/RC & -1/C \\ 1/L & 0 \end{bmatrix} \cdot \begin{bmatrix} v_c \\ i_L \end{bmatrix} + \begin{bmatrix} 0 \\ 1/L \end{bmatrix} \cdot u \\ v_c &= \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} v_c \\ i_L \end{bmatrix} \end{aligned} \tag{B.5}$$

APÊNDICE C

ESTRUTURA DO PROGRAMA IMPLEMENTADO NO DSP PARA O CONTROLE DO PROTÓTIPO

O programa do controlador RMRAC foi implementado usando o compilador C/C++ do TMS320C28x. O sistema de desenvolvimento do programa Code Composer, permite dentre outras funções, a utilização de um editor para a programação do código, um compilador, emulador e permite gravar o código objeto no DSP. Do ponto de vista de geração de código, o programa foi desenvolvido a partir de uma estrutura de programa fornecida pela própria Texas Instruments®. Segue nas próximas páginas a descrição da estrutura do programa bem como algumas linhas de código mais importantes que foram implementadas.

C.1 Programa Principal

```
//===========================================================
//===========Inclusão dos arquivos de cabeçalho necessários ============
//===========================================================
#include "DSP28_Device.h"
#include "ADCcalibrationDriver.h"
#include <IQmathLib.h>
#include <Var_RMRAC4parm.h>

//===========================================================
//==============     Definição de Símbolos e Constantes      ==============
//===========================================================
#define     d0          5
#define     d1          1
#define     sigm0       0.01
#define     M0          2
#define     P0          30
#define     F           2000
#define     q           100
#define     t1          -2
#define     t2          2
#define     t3          0.3
#define     t4          1
```

```
#define         Ts              0.00008333

// Modelo de Referência para Lp=1mH e Rp=60uF
extern long     wm1 = _IQ28(1.62165289235263);
extern long     wm2 = _IQ28(-0.74127612877875);
extern long     wm3 = _IQ28(0.06280553273312);
extern long     wm4 = _IQ28(0.05681770369284);

// Planta para simulação
/*
extern long     wp1 = _IQ28(1.83239724459756);
extern long     wp2 = _IQ28(-0.94377228035834);
extern long     wp3 = _IQ28(0.05622667789856);
extern long     wp4 = _IQ28(0.05514835786221);
*/

// Filtros Auxiliares
extern long     I_TsF = _IQ28(1-Ts*F);
extern long     Tsq  = _IQ29(Ts*q);
extern long     I_Tsd0 = _IQ28(1-Ts*d0);
extern long     Tsd1 = _IQ29(Ts*d1);
extern long     M0 = _IQ(M0);
extern long     duasxM0 = 2*M0;
extern long     sigma0  = _IQ29(sigm0);
extern long     ts = _IQ30(Ts);

// Filtro de Kalmam (estados vc e iL)
// Filtro LC = 1mH/60uF e R=15
extern long A[2][2] = {   _IQ30(0.88862392249657),  _IQ30(1.32358559051665),
                          _IQ18(-0.07941513543100),  _IQ30(0.94377332210144)  };

extern long B[2]   = {    _IQ30(0.05622667789857), _IQ18(0.08175791367677)   };
// Para R=15
extern long K[2]   = { _IQ30(0.16846244863104),_IQ18(0.00193256140144) };

//RMRAC_VAR phA, phB, phC;
RMRAC phA = RMRAC_DEFAULTS;
RMRAC phB = RMRAC_DEFAULTS;
RMRAC phC = RMRAC_DEFAULTS;

//=============================================================
//====================    Definição de Variáveis    ===================
//=============================================================
long    Va, Vb, Vc, KDC;                    // Tensões das fases
long    V_offsset;                          // Tensões de offsset das medidas
long    VDC[64] = {     0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0,
```

```
                        0,0,0,0,0,0,0,0,
                        0,0,0,0,0,0,0,0  };        // Aquisição do barramento CC para média

// Definição da Matriz de mudança de base ABCN ---> abO00
extern void coeficientes();

// Aloca variáveis do driver de calibração:
ADC_CALIBRATION_DRIVER_VARS adc;

// Variáveis para teste, fluxo do programa e temporárias:
long a, a1, b, w, temp, temp1, temp2, temp3, M, Contador, Contador1, f_ini, f_end, freq;
Uint16 Cont, x, npts, ii;

// Modulação sVM e Mudança de base
long    MAT11, MAT12, MAT13, MAT21, MAT22, MAT23, MAT31, MAT32, MAT33;
long    INV_MAT11, INV_MAT12, INV_MAT13, INV_MAT21, INV_MAT22, INV_MAT23,INV_MAT31,
long    INV_MAT32,INV_MAT33;
long    s1, s2, s3, s4, s5, s6;
long    norma_elipse, ualfa, ubeta, uo;
long    M11, M12, M13, M21, M22, M23, M31, M32, M33;
long    first, second, third, fourth;
long    val_CMPR1, val_CMPR2, val_CMPR3, val_CMPR4;

// Buffers
extern  int16   y_buf[BUFFER_SIZE];
extern  int16   ym_buf[BUFFER_SIZE];
extern  int16   gp_buf[BUFFER_SIZE];

//===========================================================
//=====================   Programa Principal   ==============
//===========================================================
void main(void)
{
        // Inicilizações da RAM interna, Watchdog, Habilitação de Clocks para os periféricos
        InitSysCtrl();

        // Desabilita e limpa todos os vetores de interrupção:
        DINT;
        IER = 0x0000;
        IFR = 0x0000;

        // Inicializa pinos de GPIO
        EALLOW;
        GpioMuxRegs.GPAMUX.all = 0x006A;            // habilita GPIOA0(PWM1), PWM2,
                                                    // PWM4, PWM6 e T1PWM
        GpioMuxRegs.GPBMUX.all = 0x0000;            // Todos os pinos do EVB/GPIO são
                                                    // configurados como I/O
        GpioMuxRegs.GPBQUAL.all = 0x0000;
        GpioMuxRegs.GPADIR.bit.GPIOA0 = 1;          // GPIOA0(PWM1) como saída
        GpioDataRegs.GPASET.bit.GPIOA0 = 1;         // HAbilita o pino de Inibição, deste
                                                     // modo gera-se os sinais para os
```

```
                                        // gates dos IGBT's

GpioMuxRegs.GPFMUX.bit.SCITXDA_GPIOF4 = 0;      // Pino GPIOF4(SCITXDA)
                                                // como GPIO
GpioMuxRegs.GPFDIR.bit.GPIOF4 = 1;              // GPIOF4 como saída
GpioDataRegs.GPFCLEAR.bit.GPIOF4 = 1;           // Pino GPIOF4 é inicializado
                                                // em estado baixo
EDIS;

// Inicializa os registradores de Interrupção de periféricos em um valor conhecido
InitPieCtrl();

// Inicializa a tabela de vetores de interrupção
InitPieVectTable();

// Configura os canais de entrada analógica para conversão, o modo e as seqüências
ADCcalibrationInitAdc();

// Inicialização do drive de calibração do A/D
// ADCcalibrationDriverInit(&adc);

// Inicializa os timers desejados, seus modos de contagem, valores de período e outros
Init_PWM();

// Inicializa as matrizes para mudança de base
coeficientes();

// Configura os parâmetros de tempo para a zona6 da memória externa
xintf_zone6_timing();

// Inicializa a memória flash
//InitFlash();

// Inicializações de Variáveis do Controlador
phA.sigma = sigma0;
phB.sigma = sigma0;
phC.sigma = sigma0;

phA.P11 = _IQ(P0);
phA.P22 = _IQ(P0);
phA.P33 = _IQ(P0);
phA.P44 = _IQ(P0);
phB.P11 = _IQ(P0);
phB.P22 = _IQ(P0);
phB.P33 = _IQ(P0);
phB.P44 = _IQ(P0);
phC.P11 = _IQ(P0);
phC.P22 = _IQ(P0);
phC.P33 = _IQ(P0);
phC.P44 = _IQ(P0);
```

```
phA.teta1k = _IQ26(t1);
phA.teta1k_1 = _IQ26(t1);
phA.teta2k = _IQ26(t2);
phA.teta2k_1 = _IQ26(t2);
phA.teta3k = _IQ26(t3);
phA.teta3k_1 = _IQ26(t3);
phA.teta4k = _IQ26(t4);
phA.teta4k_1 = _IQ26(t4);
phB.teta1k = _IQ26(t1);
phB.teta1k_1 = _IQ26(t1);
phB.teta2k = _IQ26(t2);
phB.teta2k_1 = _IQ26(t2);
phB.teta3k = _IQ26(t3);
phB.teta3k_1 = _IQ26(t3);
phB.teta4k = _IQ26(t4);
phB.teta4k_1 = _IQ26(t4);
phC.teta1k = _IQ26(t1);
phC.teta1k_1 = _IQ26(t1);
phC.teta2k = _IQ26(t2);
phC.teta2k_1 = _IQ26(t2);
phC.teta3k = _IQ26(t3);
phC.teta3k_1 = _IQ26(t3);
phC.teta4k = _IQ26(t4);
phC.teta4k_1 = _IQ26(t4);

phA.mk  = _IQ(1.2*d1/d0);
phA.mk1 = _IQ(1.2*d1/d0);
phA.mbk = _IQ(1.2*d1/d0);
phB.mk  = _IQ(1.2*d1/d0);
phB.mk1 = _IQ(1.2*d1/d0);
phB.mbk = _IQ(1.2*d1/d0);
phC.mk  = _IQ(1.2*d1/d0);
phC.mk1 = _IQ(1.2*d1/d0);
phC.mbk = _IQ(1.2*d1/d0);

// Inicializações de variáveis para Controle do fluxo do programa
x = 1;
x_harm1 = 1;
Cont = 1;
Contador = 1;
Contador1 = 1;
npts = SWITCH_FREQ/FREQ;
npts_max = 48000;

// Cálculo da tensão de offsset
AdcRegs.ADCTRL2.bit.SOC_SEQ1=1;          // Inicia a conversão

while (AdcRegs.ADCST.bit.INT_SEQ1 == 0){}  // Espera o fim da conversão

temp1 = AdcRegs.ADCRESULT5;              // Obtêm os resultados
temp2 = AdcRegs.ADCRESULT7;
```

```
        temp3 = AdcRegs.ADCRESULT3;

        temp1 = temp1 << 6;
        temp2 = temp2 << 6;
        temp3 = temp3 << 6;
        temp = temp1 + temp2 + temp3;
        V_offsset = temp/3;                                  // Valor correspondente a
                                                    // tensão de offset

        // Reiniciliza o A/D para a próxima seqüência de conversão
        EALLOW;
        AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;         // Reinicializa a SEQ1
        AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;       // Limpa o flag de interrrupção da SEQ1
        EDIS;

        // Inicializa o timer1 é iniciado em 0x000 e inicia a contagem
        EvaRegs.T1CNT = 0x0000;
        EvaRegs.T1CON.bit.TENABLE = 1;

        // Loop infinito, esperando a interrupção de underflow do timer1
        for( ; ; );

}

//===========================================================
//==============      Rotina de tratamento da Interrupção     ==============
//===========================================================
interrupt void eva_timer1_isr(void)
{
        // Coloca o pino GPIOA6/T1PWM num valor alto marcando o início da rotina
        GpioDataRegs.GPFSET.bit.GPIOF4 = 1;

        // Atualiza a ação de Controle
        EvaRegs.CMPR1 = val_CMPR1;
        EvaRegs.CMPR2 = val_CMPR2;
        EvaRegs.CMPR3 = val_CMPR3;
        EvaRegs.T1CMPR = val_CMPR4;

        // Espera o fim da aquisição
        while (AdcRegs.ADCST.bit.INT_SEQ1 == 0){}

        // Barramento CC
        temp2 = 0;
        ii = 64;
        while(ii>0)
        {
                ii--;
                if (ii > 0)
                        VDC[ii] = VDC[ii-1];
                else
                        VDC[ii] = AdcRegs.ADCRESULT1;
```

```
        temp2 += VDC[ii];
}
temp2 = temp2 >> 6;             // Divide por 64
KDC = temp2*83.7;                       // Ajusta o valor para Q13

// Aquisição das tensões de fase
temp1 = AdcRegs.ADCRESULT5;
temp2 = AdcRegs.ADCRESULT7;
temp3 = AdcRegs.ADCRESULT3;

temp1 = temp1 << 6;
temp2 = temp2 << 6;
temp3 = temp3 << 6;

temp1 = temp1 - V_offsset;      // Subtrae o offsset
temp2 = temp2 - V_offsset;
temp3 = temp3 - V_offsset;

phA.yp_tmp = temp1 << 2;        // Ajuste na faixa =+/-0.8pu
phB.yp_tmp = temp2 << 2;
phC.yp_tmp = temp3 << 2;

// Reinicializa a Seq. de conversão do A/D
EALLOW;
AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;
AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
EDIS;

// Obtêm as referências de Tensão
a = 26353589.2666 * x / npts;           // 2*pi*f*t

temp1 = _IQ22(2.09439510239);           // Defasagem de 120º

phA.rk = _IQsin(a);
phB.rk = _IQsin(a - temp1);
phC.rk = _IQsin(a + temp1);

if (Contador > 4)
{
        Va = _IQ(0.8);
        Vb = _IQ(0.8);
        Vc = _IQ(0.8);
        Contador1++;
}
else
{
        Va = _IQ(0.3);          // Nos três primeiros ciclos aciona com amplitude
        Vb = _IQ(0.3);          //reduzida
        Vc = _IQ(0.3);
}
```

```
        phA.rk = _IQmpy(Va,phA.rk);
        phB.rk = _IQmpy(Vb,phB.rk);
        phC.rk = _IQmpy(Vc,phC.rk);

        // Contador equivalente ao tempo, para o cálculo das referências
        if(x == npts)
        {
                x = 1;
                Contador++;
        }
        else x++;

        // Funções para o cálculo dos controladores RMRAC para cada fase
        RMRAC_controller(&phA,'A');
        RMRAC_controller(&phB,'B');
        RMRAC_controller(&phC,'C');

        // Transformação em coordenadas alfa-beta-O
        ualfa = _IQmpy(MAT11,phA.uk) + _IQmpy(MAT12,phB.uk) + _IQmpy(MAT13,phC.uk);
        ubeta = _IQmpy(MAT21,phA.uk) + _IQmpy(MAT22,phB.uk) + _IQmpy(MAT23,phC.uk);
        uo = _IQmpy(MAT31,phA.uk) + _IQmpy(MAT32,phB.uk) + _IQmpy(MAT33,phC.uk);

          //===========================================================
          //=================       Modulação Space Vector       ===============
          //===========================================================

        // As linhas de código usadas para as tarefas relacionadas, foram omitidas:
          // a. Cálculo dos planos de separação s1, s2, s3, s4, s5 e s6
        // b. Identificação do Tetraedro ao qual a Lei de Controle pertence, baseado nisso, a
        //         função “matriz()” é utilizada para o cálculo dos tempos dos vetores
        // c. função “matriz()”;

          //===========================================================

        // Habilita a interrupção de "Underflow" e limpa o "flag" para o timer1
        EvaRegs.EVAIMRA.bit.T1UFINT = 1;
        EvaRegs.EVAIFRA.bit.T1UFINT = 1;

        // Habilita o bloco PIE2, que responde entre outras, pela interrupção de
        // underflow do GPtimer1 ( ver DSP28_PieVect.c linha 73          )
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP2;

        GpioDataRegs.GPFCLEAR.bit.GPIOF4 = 1;

    return;

}
```

C.2 Rotina de cálculo da Lei de Controle

```
//===================================================================
//===========Inclusão dos arquivos de cabeçalho necessários ===========
//===================================================================
#include "DSP28_Device.h"
#include "IQmathLib.h"
#include "Var_RMRAC4parm.h"

//===================================================================
//============          Definição de Símbolos, Constantes e Variáveis ==============
//===================================================================
extern long       wm1, wm2, wm3, wm4, wp1, wp2, wp3, wp4, I_TsF, Tsq, I_Tsd0, Tsd1, duasxM0, sigma0, ts, M0, KDC,
                                ...Amplitude;
extern long       A[2][2], B[2], K[2];
long      Ts_sigma_teta1, Ts_sigma_teta2, Ts_sigma_teta3, Ts_sigma_teta4;
long      Ts_ea_qsi1, Ts_ea_qsi2, Ts_ea_qsi3, Ts_ea_qsi4;
long      tmp1, tmp2, tmp3, tmp4, tmp5, tmp6, tmp7, tmp8, tmp, Mtmp;
float     tmp9;

//===================================================================
//================  Cálculo da Lei de Controel RMRAC       ==============
//===================================================================
void RMRAC_controller(RMRAC *p, char fase)
{
        // Calcula o Modelo de Referência
        p->ymk = _IQmpyIQX(wm1,28,p->ymk_1,22) + _IQmpyIQX(wm2,28,p->ymk_2,22) +…
                        _IQmpyIQX(wm3,28,p->rk_1,22) + _IQmpyIQX(wm4,28,p->rk_2,22);

        // Filtro de Kalmam incluindo o modelo da planta e a ação de controle "u"
        tmp1 = p->yp_tmp - p->V1k_1;
        p->V1k = _IQmpyIQX(A[0][0],30,p->V1k_1,22) + _IQmpyIQX(A[0][1],30,p->V2k_1,22) +…
                        _IQmpyIQX(K[0],30,tmp1,22) + _IQmpyIQX(B[0],30,p->uk_1,22);
        p->V2k = _IQmpyIQX(A[1][0],18,p->V1k_1,22) + _IQmpyIQX(A[1][1],30,p->V2k_1,22) +…
                        _IQmpyIQX(K[1],18,tmp1,22) + _IQmpyIQX(B[1],18,p->uk_1,22);
        p->yk = p->V1k;

           // Erro de rastreamento
        tmp9 = _IQtoF(p->yk) - _IQtoF(p->ymk);
          p->ek = _IQ30(tmp9);

        // LEI DE CONTROLE
        // uk = tetak'*[ w1k ; w2k ; yk ; rk ];
          p->uk = _IQmpyIQX(p->teta1k,26,p->w1k,29);
          p->uk += _IQmpyIQX(p->teta2k,26,p->w2k,29);
          p->uk += _IQmpyIQX(p->teta3k,26,p->yk,22);
          p->uk += _IQmpyIQX(p->teta4k,26,p->rk,22);

        // Filtros auxiliares
          p->w1k1 = _IQ29mpyIQX(I_TsF,28,p->w1k,29) + _IQ29mpyIQX(Tsq,29,p->uk,22);
```

```
p->w2k1 = _IQ29mpyIQX(I_TsF,28,p->w2k,29) + _IQ29mpyIQX(Tsq,29,p->yk,22);

// mk1 = B3*mk + B4*(modu+mody+1), e B3=B4=1;
  p->mbk = _IQmpy(p->mk,p->mk);

// Erro Aumentad0
// eak = ( ek + ( tetak'*qsik - vk )/tetak(4) )/mbk;
tmp1 = _IQ26mpyIQX(p->teta1k,26,p->qsi1k,28);
tmp1 += _IQ26mpyIQX(p->teta2k,26,p->qsi2k,28);
tmp1 += _IQ26mpyIQX(p->teta3k,26,p->qsi3k,28);
tmp1 += _IQ26mpyIQX(p->teta4k,26,p->qsi4k,28);
tmp1 -= p->vk >> 2;                  // tetak'*qsik - vk
tmp1 = (p->ek>>2) + _IQ28div(_IQ26toIQ(tmp1),_IQ26toIQ(p->teta4k));
p->eak = _IQ28div(_IQ28toIQ(tmp1),p->mbk);

   // v(k)=Wm*t'*w
p->vk1 = _IQ28mpyIQX(wm1,28,p->vk,28) + _IQ28mpyIQX(wm2,28,p->vk_1,28) +…
              _IQ28mpyIQX(wm3,28,p->uk,22) + _IQ28mpyIQX(wm4,28,p->uk_1,22);

  // Sigma Modification
tmp1 = _IQmpyIQX(p->teta1k,26,p->teta1k,26) + _IQmpyIQX(p->teta2k,26,p->teta2k,26) +…
              _IQmpyIQX(p->teta3k,26,p->teta3k,26) + _IQmpyIQX(p->teta4k,26,p->teta4k,26);
Mtmp = _IQsqrt(tmp1);

if (Mtmp > duasxM0)
            p->sigma = sigma0;
else
{
        if (Mtmp < M0)
                p->sigma = 0;
        else
        {
                tmp1 = _IQ29div(Mtmp,M0) - _IQ29(1);
                p->sigma = _IQ29mpy(_IQ29toIQ(sigma0),_IQ29toIQ(tmp1));
        }
}

// Adaptação Paramétrica
// tetak1 = ( eye(4) - Ts*sigma*P )*tetak - Ts*eak*P*qsik;
tmp = _IQ29mpyIQX(ts,30,p->sigma,29);                  // Ts*sigma

  tmp1 = (_IQ29(1) - _IQ29mpyIQX(tmp,29,p->P11,22));        // ( 1-Ts*sigma*Pii )
  tmp1 = _IQ26mpyIQX(tmp1,29,p->teta1k,26);          // ( 1-Ts*sigma*Pii )*tetaik

tmp2 = (_IQ29(1) - _IQ29mpyIQX(tmp,29,p->P22,22));  // ( 1-Ts*sigma*Pii )
tmp2 = _IQ26mpyIQX(tmp2,29,p->teta2k,26);          // ( 1-Ts*sigma*Pii )*tetaik

  tmp3 = (_IQ29(1) - _IQ29mpyIQX(tmp,29,p->P33,22));        // ( 1-Ts*sigma*Pii )
tmp3 = _IQ26mpyIQX(tmp3,29,p->teta3k,26);          // ( 1-Ts*sigma*Pii )*tetaik

tmp4 = (_IQ29(1) - _IQ29mpyIQX(tmp,29,p->P44,22));  // ( 1-Ts*sigma*Pii )
```

```
tmp4 = _IQ26mpyIQX(tmp4,29,p->teta4k,26);              // ( 1-Ts*sigma*Pii )*tetaik

tmp = _IQ28mpyIQX(ts,30,p->eak,28);                    // Ts*ea
   Ts_ea_qsi1 = _IQ28mpyIQX(tmp,28,p->qsi1k,28);
   Ts_ea_qsi2 = _IQ28mpyIQX(tmp,28,p->qsi2k,28);
   Ts_ea_qsi3 = _IQ28mpyIQX(tmp,28,p->qsi3k,28);
   Ts_ea_qsi4 = _IQ28mpyIQX(tmp,28,p->qsi4k,28);

   p->teta1k1 = tmp1 - _IQ26mpyIQX(Ts_ea_qsi1,28,p->P11,22);
   p->teta2k1 = tmp2 - _IQ26mpyIQX(Ts_ea_qsi2,28,p->P22,22);
   p->teta3k1 = tmp3 - _IQ26mpyIQX(Ts_ea_qsi3,28,p->P33,22);
   p->teta4k1 = tmp4 - _IQ26mpyIQX(Ts_ea_qsi4,28,p->P44,22);

// Ksi = Wm*W
// qsik1 = wm1*qsik + wm2*qsik_1 + wm3*[ w1k ; w2k ; yk ; rk ] + wm4*[ w1k_1 ; w2k_1 ; yk_1 ; rk_1 ];
p->qsi1k1 = _IQ28mpyIQX(wm1,28,p->qsi1k,28);
p->qsi1k1 += _IQ28mpyIQX(wm2,28,p->qsi1k_1,28);
p->qsi1k1 += _IQ28mpyIQX(wm3,28,p->w1k,29);
p->qsi1k1 += _IQ28mpyIQX(wm4,28,p->w1k_1,29);

   p->qsi2k1 = _IQ28mpyIQX(wm1,28,p->qsi2k,28);
p->qsi2k1 += _IQ28mpyIQX(wm2,28,p->qsi2k_1,28);
p->qsi2k1 += _IQ28mpyIQX(wm3,28,p->w2k,29);
p->qsi2k1 += _IQ28mpyIQX(wm4,28,p->w2k_1,29);

   p->qsi3k1 = _IQ28mpyIQX(wm1,28,p->qsi3k,28);
p->qsi3k1 += _IQ28mpyIQX(wm2,28,p->qsi3k_1,28);
p->qsi3k1 += _IQ28mpyIQX(wm3,28,p->yk,22);
p->qsi3k1 += _IQ28mpyIQX(wm4,28,p->yk_1,22);

   p->qsi4k1 = _IQ28mpyIQX(wm1,28,p->qsi4k,28);
p->qsi4k1 += _IQ28mpyIQX(wm2,28,p->qsi4k_1,28);
p->qsi4k1 += _IQ28mpyIQX(wm3,28,p->rk,22);
p->qsi4k1 += _IQ28mpyIQX(wm4,28,p->rk_1,22);

// m(k+1) = (1-Ts*d0)*m(k) + Ts*d1*(|uk|+|yk|+1)
tmp1 = _IQabs(p->uk) + _IQabs(p->yk) + _IQ(1);
p->mk1 = _IQmpyIQX(I_Tsd0,28,p->mk,22) + _IQmpyIQX(Tsd1,29,tmp1,22);

/*         P = (1+Ts*lb*mib^2)*P - Ts*( (P*qsik*qsik'*P )/mbk + mib^2*P^2/R^2);

// Atualizações
   p->rk_2 = p->rk_1;
   p->rk_1 = p->rk;
   p->yk_2 = p->yk_1;
    p->yk_1 = p->yk;
p->yp_tmp_2 = p->yp_tmp_1;
p->yp_tmp_1 = p->yp_tmp;
p->V1k_2 = p->V1k_1;
p->V1k_1 = p->V1k;
p->V2k_1 = p->V2k;
```

```
        p->V3k_1 = p->V3k;
        p->V4k_1 = p->V4k;
        p->V5k_1 = p->V5k;
        p->V6k_1 = p->V6k;
        p->ymk_2 = p->ymk_1;
        p->ymk_1 = p->ymk;
        p->vk_1 = p->vk;
        p->vk = p->vk1;
        p->uk_2 = p->uk_1;
        p->uk_1 = p->uk;
        p->w1k_1 = p->w1k;
        p->w2k_1 = p->w2k;
        p->w1k = p->w1k1;
        p->w2k = p->w2k1;
        p->teta1k_1 = p->teta1k;
        p->teta2k_1 = p->teta2k;
        p->teta3k_1 = p->teta3k;
        p->teta4k_1 = p->teta4k;
        p->teta1k = p->teta1k1;
        p->teta2k = p->teta2k1;
        p->teta3k = p->teta3k1;
        p->teta4k = p->teta4k1;
        p->qsi1k_1 = p->qsi1k;
        p->qsi2k_1 = p->qsi2k;
        p->qsi3k_1 = p->qsi3k;
        p->qsi4k_1 = p->qsi4k;
        p->qsi1k = p->qsi1k1;
        p->qsi2k = p->qsi2k1;
        p->qsi3k = p->qsi3k1;
        p->qsi4k = p->qsi4k1;
        p->mk = p->mk1;

return;
```