PEA5855 - Embarcados para SEP

Prof. Dr. Eduardo Lorenzetti Pellini

Trabalho Final - 2022.1

Fernando Vendramini - NUSP: 13018571
Luiz Henrique Neves Rodrigues - NUSP: 10672506
Marcelo Monari Baccaro - NUSP: 8989262
Escola Politécnica da Universidade de São Paulo
PEA - Departamento de Engenharia de Energia e Automação Elétricas
Programa de Pós-Graduação em Engenharia Elétrica

Sumário

Capítulo 1

Desenvolvimento de um IED

1.1 Descrição do equipamento

O equipamento que foi desenvolvido pelo grupo é um dispositivo conhecido pela sigla IED (*Intelligent Electronic Device*) e que tem como objetivo a detecção de sobrecorrente e desligamento em uma linha de transmissão (LT), supervisionando dois conjuntos de dados trifásicos (tensões e correntes).

Um algoritmo foi implementado em duas etapas. A primeira trata da estimação da magnitude da componente fundamental de cada sinal, utilizando a Transformada Discreta de Fourier (DFT - *Discrete Fourier Transform*) com janela móvel de ciclo completo. A segunda etapa realiza a classificação do valor estimado na primeira etapa, e gera um sinal de *trip* de sobrecorrente ou de desligamento de uma fase para a saída do IED. Nosso algoritmo é enquadrado como **Digital Relaying**, que não é um classificador tão complexo como os utilizados em aplicações atuais (*Digital Distance Relaying*, *Adaptive Relaying* ou *Power Swing Detection*) [4,5,7], mas implementa a função proposta de maneira satisfatória.

Considerando que o dispositivo processa um total de seis grandezas elétricas (três correntes e três tensões), são necessários seis canais analógicos para a aquisição de dados. O processamento de cada fase é desacoplado das demais e gera saídas digitais individuais, resultando em seis canais digitais de saída. Além disso, nas melhores práticas de automação e proteção é comum utilizar sinais digitais individuais para a habilitação (*enable*) de canais digitais, assim teremos seis canais digitais de entrada. Adicionalmente, é necessário um canal digital de entrada para a habilitação geral do dispositivo.

Resumindo:

- 6 canais de entrada analógica com bloco de condicionamento, proteção e filtragem;
- 7 canais de entrada digital com opto acopladores de +24V e;
- 6 canais de saída digital com opto acopladores de +24V.

O IED é um dispositivo digital com processamento feito por um microcontrolador ou um DSP (*digital signal processor*), assim ele também possui alguns parâmetros referentes aos sistemas digitais:

- clock de 20MHz para o microcontrolador (ou DSP);
- registradores com tamanho de 16 bits;
- ADC com resolução de 10 bits;
- e ADC bipolares de -5V a +5V.

O nosso algoritmo combinado da estimação de magnitude da fundamental com a detecção de sobrecorrente ou desligamento também possui parâmetros:

- 0,15s de delay para estabilização durante a inicialização;
- amostragem com frequência de 480Hz (8 amostras por ciclo);
- DFT de ciclo completo com janela de 8 amostras;
- seno e cosseno calculados pelo CORDIC modo de rotação com 28 iterações.

1.2 Sinais de entrada analógica

Os sinais de entrada utilizados nas simulações são modificações dos dados originais obtidos do caso 4 dos arquivos COMTRADE fornecidos pelo professor. O conjunto trifásico de tensões correspondem aos VLTad's (VLTadA, VLTadB e VLTadC), que são os três primeiros sinais de tensão do arquivo "AD_GR_Rf25_F1_AT_90.dat". E o conjunto trifásico de correntes correspondem aos IAD's (IAD1, IAD2 e IAD3), que também são os três primeiros sinais de corrente desse mesmo arquivo.

Os dados desse arquivo já passaram por pré-processamento, em que eles foram limitados, amostrados e quantizados. Pode-se assumir que a amostragem foi feita com um período de amostragem (0,2ms) pequeno o suficiente para considerar os sinais como contínuos. Além disso, a quantização tem uma resolução de 12 bits (4096 níveis) e pode-se considerar que o ruído de quantização é insignificante. Assim, os dados brutos estão limpos.

O programa do apêndice A.1 apresenta o código em Matlab para gerar os sinais de entrada. O primeiro passo dele é extrair os sinais do arquivo .dat (que foi transformado em .csv por comodidade), juntamente com o tempo, que foi limitado a 0,8s ao invés do 1,5s original pois não há informação significativa nos 0,7s finais. Como os dados brutos estão quantizados em símbolos binários, pode-se atribuir qualquer valor aos níveis. Desta maneira, o segundo passo é o mapeamento linear (programa no apêndice A.2), em que nível 0 foi mapeado para o valor de pico negativo e o nível 4096 foi mapeado para o valor de pico positivo. Para a tensão, a amplitude do sinal é de $115/\sqrt{3} \approx 66,4$V. E, para a corrente, a amplitude do sinal é de 6A. O terceiro passo é adicionar um ruído gaussiano branco (WGN) com razão ruído-sinal (SNR) de 30dB para tornar o sinal de entrada mais interessante (programa no apêndice A.3). Por fim, os sinais de entrada são salvos num arquivo .csv para leitura pelo programa principal no apêndice A.4.

1.3 Condicionamento dos sinais de entrada e proteção interna

Como mostra o programa principal no apêndice A.4, o primeiro estágio dentro do IED a receber os sinais analógicos de entrada é o condicionamento de sinais, cuja função é amplificar (ou atenuar) os sinais para que eles fiquem com amplitudes próximas aos limites de tensão admitidos pelo ADC (conversor analógico-digital) através de um mapeamento linear (ou função afim), como mostra a equação (1.1) em que se mapeia o domínio $[b_1, a_1]$ na imagem $[b_2, a_2]$.

$$f(x) = \frac{a_2 - b_2}{a_1 - b_1} \cdot (x - a_1) + a_2 \tag{1.1}$$

Logo em seguida está o estágio de proteção interna, cuja função é limitar o sinais condicionados aos limites de tensão admitidos pelo ADC para evitar a queima deste por sobretensão, ou seja, ele atua como um saturador como descreve a equação (1.2).

$$f(x) = \begin{cases} x_{max}, \text{ se } x > x_{max} \\ x, \text{ se } x_{min} \leq x \leq x_{max} \\ x_{min}, \text{ se } x < x_{min} \end{cases} \tag{1.2}$$

Os estágios de condicionamento e proteção podem ser implementados como um estágio único pela eletrônica analógica, que seria um amplificador linear transistorizado que amplifica (ou atenua) ao mesmo tempo ceifa os sinais no limite de sua saturação, que corresponde às tensões (positiva e negativa) de alimentação. E é justamente os efeitos deste componente que o programa do apêndice A.2 busca implementar para sinais de saída com variação de -5V a +5V devido a característica bipolar (e não unipolar de 0V a +5V) do ADC.

1.4 Filtragem analógica passa-faixa

Em IEDs de proteção aplicados às linhas de transmissão, a magnitude da frequência fundamental é a informação de interessa para tomada de decisão pelo classificador de trip. Em outros sistemas de proteção, como de transformadores ou motores elétricos síncronos, as harmônicas também têm relevância. Assim, a filtragem analógica deve ser projetada de maneira a destacar os dados que carreguem informação sobre a fundamental. Isto é feito através do projeto em domínio de frequência de forma que a magnitude do filtro deve ser o mais próximo possível de 1 (ou 0 em dB) na faixa desta frequência. Como apenas a informação da fundamental é relevante para o classificador, então o filtro mais adequado é o passa-faixa.

Existem muitas formas de realizar um filtro passa-faixa. Uma das formas mais fáceis de realizá-lo é através de um filtro Butterworth utilizando a função "butter.m"do Matlab, em que se define a frequência central f_c e a faixa simétrica de frequência passante f_p. O programa no apêndice A.5 realiza-o com $f_c = 60$Hz e $f_p = 20$Hz (faixa passante de 50Hz até 70Hz) e ordem 1 (sistema dinâmico linear de segunda ordem) resultando na função de transferência (1.3), que apresenta um zero na origem ao contrário dos filtros Butterworth passa-baixa que possuem zeros apenas no infinito. O diagrama de Bode do filtro é mostrado na figura 1.1. O programa também aplica o filtro no sinal condicionado e limitado.

$$G_{butter}(s) = \frac{125,7 \cdot s}{s^2 + 125,7 \cdot s + 138174} \tag{1.3}$$

1.5 Digitalização pelo ADC

Certamente que o filtro analógico passa-faixa desempenha um papel importante em destacar a informação de interesse. Mas, em termos de aquisição de sinais digitais, ele possui um papel ainda mais importante: filtro anti-aliasing. Para que um sinal seja digitalizado, ou seja, discretizado e quantizado, a frequência de amostragem deve satisfazer o critério de Nyquist, que parte do principio que existe uma banda limitada de frequência no sinal amostrado. Como isto nem sempre é o caso de sinais reais, o filtro passa-faixa (ou passa-baixa) garante que esta condição está satisfeita.

Como o algoritmo do IED de proteção é digital, o passo de aquisição de dados deve acontecer pelo sistema conjunto de segurador de ordem zero (ZOH) com o ADC. O estágio de digitalização é executado pelos programas dos apêndices A.6, A.7 e A.8. O primeiro programa descreve a amostragem do sinal analógico filtrado com 8 amostras por ciclos (valor baixo escolhido propositalmente para testar o algoritmo) considerando que a frequência da fundamental é de 60Hz, ou seja, a frequência de amostragem é de $f_s = 8 \cdot 60 = 480$Hz (ou período de amostragem $T_s = 1/480 \approx 2.083$ms), o que satisfaz o critério de Nyquist. O segundo programa descreve um algoritmo de amostragem rápida pois o primeiro já calculou alguns parâmetros que não precisam ser calculados novamente. E o terceiro, é a quantização dos sinais com resolução de 10 bits.

1.6 Algoritmo do IED

Como descrito anteriormente, o objetivo do algoritmo do IED de proteção é detectar desligamento de fase, assim como sobrecorrente, num contexto de LT. Para tal finalidade, precisa-se estimar recursivamente a magnitude da componente fundamental de cada fase, o que é feito através da DFT.

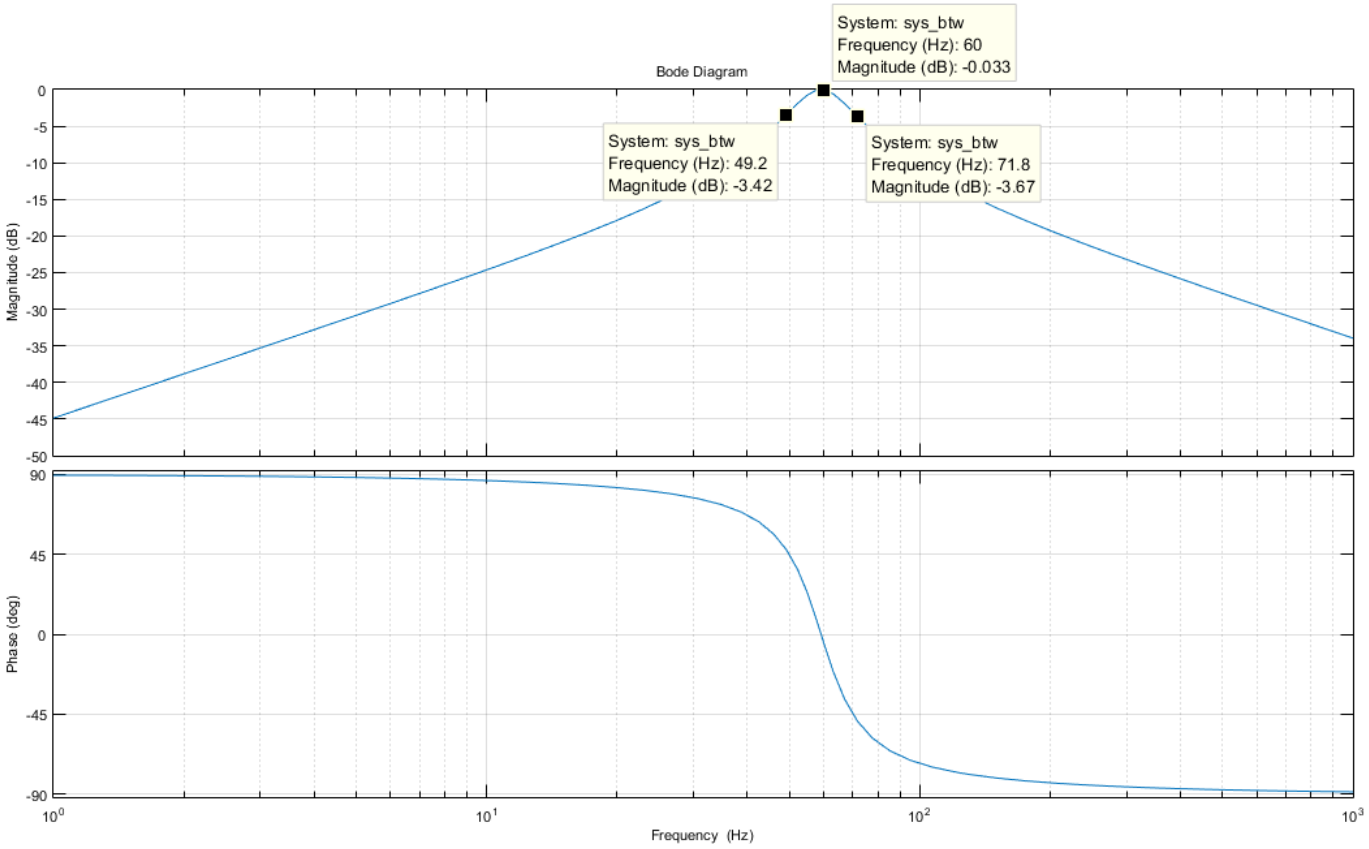

Figura 1.1: Diagrama de Bode para o filtro passa-faixa de 50Hz a 70Hz.

Obtido todos estes valores, um classificador decide quais serão os valores de saída digital que indicam comandos de trip para relês analógicos. Assim, o algoritmo completo tem dois estágios: estimação e detecção.

1.6.1 Estimação da componente fundamental

Quando se precisa estimar informações sobre o espectro de um sinal, o padrão em processamento de sinais, especialmente em telecomunicações, é utilizar a FFT (*Fast Fourier Transform*) ao invés da DFT, pois a primeira tem um custo computacional menor. Entretanto, esta vantagem só ocorre quando todas as raias espectrais possíveis são calculadas, ou seja, para um sinal com janela de amostragem K, a quantidade de raias espectrais estimadas também é K.

Porém, na área de sistemas de potência, apenas algumas harmônicas são de interesse como argumentam [6,10]. Geralmente, o interesse se limita à fundamental, especialmente em proteção de linhas de transmissão, e/ou a algumas harmônicas ímpares, especialmente em proteção de transformadores e motores assíncronos, afinal harmônicas pares causam assimetrias e são mais raras.

A seção 5.3.2 do livro [6] apresenta um estimador de ciclo completo para a componente harmônica p utilizando a DFT com janela de tamanho K, como mostra a equação (1.4). Este estimador basicamente adquire o sinal amostrado $x[k]$ e o correlaciona (convoluciona) com as funções ortogonais $\cos(\omega \cdot t)$ e $\sin(\omega \cdot t)$, entretanto para o tempo discretizado $t = k \cdot T_s$, para $k \in \mathbb{N}$. Assim, a entrada (ignorando o indicador de harmônica p) das funções trigonométricas se torna $\omega \cdot t = \omega \cdot T_s \cdot k = 2\pi(f/f_s) \cdot k = (2\pi/K) \cdot k$, em que f é a frequência da fundamental, f_s é a frequência de amostragem e elas se relacionam por $f_s = K \cdot f$, que indica como K não é apenas o tamanho da janela mas também a quantidade de pontos amostrados por ciclo. Mas isto é válido apenas para estimadores de

ciclo completo, sendo diferente para estimadores de meio ciclo.

$$\begin{aligned}\hat{X}^{(p)}(\omega) &= \hat{X}_c^{(p)} + j \cdot \hat{X}_s^{(p)} \\ &= \frac{2}{K}\sum_{k=1}^{K} x[k] \cdot \cos(p \cdot \omega \cdot T_s \cdot k) + j \cdot \frac{2}{K}\sum_{k=1}^{K} x[k] \cdot \sin(p \cdot \omega \cdot T_s \cdot k) \\ &= \frac{2}{K}\sum_{k=1}^{K} x[k] \cdot \cos\left(p \cdot \frac{2\pi}{K} \cdot k\right) + j \cdot \frac{2}{K}\sum_{k=1}^{K} x[k] \cdot \sin\left(p \cdot \frac{2\pi}{K} \cdot k\right)\end{aligned} \tag{1.4}$$

No contexto de sistemas embarcados, calcular funções trigonométricas representa um grande desafio. Uma maneira de tornar este processo realizável, seja em software ou diretamente em hardware, é utilizar o algoritmo CORDIC em modo de rotação [2] para aproximar estas funções. O programa do apêndice A.10 mostra como prosseguir por este caminho com um número fixo de iterações, que foi assumida 28, e tabelas pré-calculadas (*lookup tables* - LUT). E os seus valores são usados no programa do apêndice A.9 para atualizar os valores das funções trigonométricas a cada iteração do estimador. Mas, no caso da equação (1.4), não é realmente necessário utilizar este algoritmo pois os valores das funções trigonométricas se repetem a cada ciclo de estimação, assim basta usar tabelas (vetores) pré-calculadas. O que muda de fato a cada ciclo do estimador são os valores da janela (vetor) de tamanho K do sinal amostrado $x[k]$. O programa do apêndice A.11 atualiza estes valores através do conceito de vetor circular (*circular array*), em que ao invés de rotacionar o vetor inteiro (*array shift*) basta armazenar um apontador para a posição que deve receber o novo valor.

A magnitude e a fase da harmônica podem ser calculadas a partir de (1.4) como mostram, respectivamente, as equações (1.5) e (1.6). O programa do apêndice A.11 calcula apenas a magnitude da fundamental ($p = 1$) pois apenas esta é de interesse para o classificador de trip.

$$|\hat{X}^{(p)}(\omega)| = \sqrt{\left(\hat{X}_c^{(p)}\right)^2 + \left(\hat{X}_s^{(p)}\right)^2} \tag{1.5}$$

$$\angle\hat{X}^{(p)}(\omega) = \arctan\left(\frac{\hat{X}_s^{(p)}}{\hat{X}_c^{(p)}}\right) \tag{1.6}$$

1.6.2 Classificador para trip

Como escrito no começo deste capítulo, existem algoritmos complicadíssimos para realizar um classificador de trip. Mas, para a detecção de desligamento de fase ou sobrecorrente, pode-se implementar classificadores bem simples utilizando apenas um contador de eventos com um lógica de *reset* e limiares (*threshold*) fixos. Os programas dos apêndices A.12 e A.13 mostram como estão implementados os classificadores, respectivamente, do conjunto trifásico da tensão e da corrente.

Para a tensão, ocorre apenas a verificação do desligamento da fase, que é tido como uma sequência de 2 ciclos da fundamental, ou seja, por $2 \cdot K$ ciclos do estimador, com uma magnitude estimada abaixo de 10% do valor de pico nominal de $115/\sqrt{3} \approx 66,4$V, logo abaixo de 6,64V. Como o ADC está limitado em -5V a +5V, esse valor precisa ser mapeado para este conjunto, o que resulta em 0,5.

E, para a corrente, a verificação é tanto de desligamento quanto de sobrecorrente. O desligamento da fase é considerado como uma sequência de 2 ciclos da fundamental com uma magnitude estimada abaixo de 10% do valor de pico máximo, que vale 6A, logo abaixo de 0,6A. Como o ADC está limitado, este valor é mapeado para 0,5. Com relação à sobrecorrente, ela é considerada como uma sequência de meio ciclo da fundamental, ou seja, $K/2$ ciclos do estimador, com uma magnitude estimada 2,5 vezes maior que o valor de pico nominal, que vale 1A, logo acima de 2,5A. Como o ADC está limitado, este valor é mapeado para 2,0833.

Por fim, a saída de trip do classificador é habilitada apenas após 0,15s da inicialização do equipamento (*delay* programado) para evitar instabilidades e erros de classificação causados pelo transiente do filtro analógico e do estimador.

Capítulo 2

Resultados de simulação

As figuras dos gráficos deste capítulo são frutos da execução do programa no apêndice A.4 através da plotagem pelas funções dos apêndices A.14 e A.15. Cada figura tem uma indicação a qual fase ela pertence. Como o processamento, a estimação e a classificação de cada fase é feita separadamente, basta analisar uma fase pois a demais seguem a mesma lógica.

A fase VANs será usada para exemplificar a análise, cujas figuras são 2.1, 2.2 e 2.3, que indicam, respectivamente, os sinais processados até a digitalização pelo ADC, uma ampliação no processamento da fase, e a resposta do estimador e do classificador de *trip*.

A figura 2.1 tem o papel de mostrar como o sinal de entrada do IED é atenuado e limitado para não danificar o ADC. Além disso, ela mostra que o filtro analógico teve um tempo de resposta de aproximadamente 0,05s, o que corrobora a necessidade de um *delay* programado para habilitar a saída digital do classificador. Da mesma forma, com o desligamento da fase, o filtro gera um atraso até a sua desenergização.

A figura 2.2 mostra como o filtro analógico faz um ótimo trabalho em suavizar o sinal condicionado e limitado, embora ele gere um pequeno atraso. A quantização do ADC também se mostra razoável ao ser feita apenas com uma resolução de 10 bits, afinal o sinal digitalizado acompanha muito bem o sinal filtrado. Devido às propriedades no domínio da frequência do filtro, que se apresenta na figura 1.1, o sinal que é enviado ao ADC nunca ultrapassa os seus limites de tensão.

E a figura 2.3 mostra que o estimador também tem o seu atraso, embora ele seja pequeno devido ao tamanho da janela de dados acumulados também ser pequena. Isto é um *trade off* famoso: quanto maior a janela, mais preciso será o estimador, porém maior será o atraso. De qualquer forma, o estimador faz um excelente trabalho em manter a precisão. Entretanto, o que é mais crítico é o tempo de resposta do algoritmo completo ao evento de desligamento. A figura 2.8 mostra que o desligamento da fase ocorre no instante 0,6924s e a figura 2.3 mostra que a resposta de *trip* do classificador ocorre em 0,7618s, ou seja, $((0.7618 - 0.6924)/Ts)/K = (0.0694/0.0021)/8 \approx 4$ ciclos até a detecção. Este valor deveria ser o menor possível, preferencialmente, limitado a 2 ciclos. Mas o filtro analógico demora muito para desenergizar. Talvez um filtro mais rápido resolvesse. Ou então reduzir a sequência de detecção para apenas 1 ciclo.

A única diferença na análise das fases de tensão em relação às fases de corrente é que estas últimas possuem classificação com relação à sobrecorrente além do desligamento. Tomando a figura 2.9 como exemplo, pode-se verificar que o tempo de reposta do algoritmo é adequado para a detecção de sobrecorrente. Mas o problema do desligamento persiste.

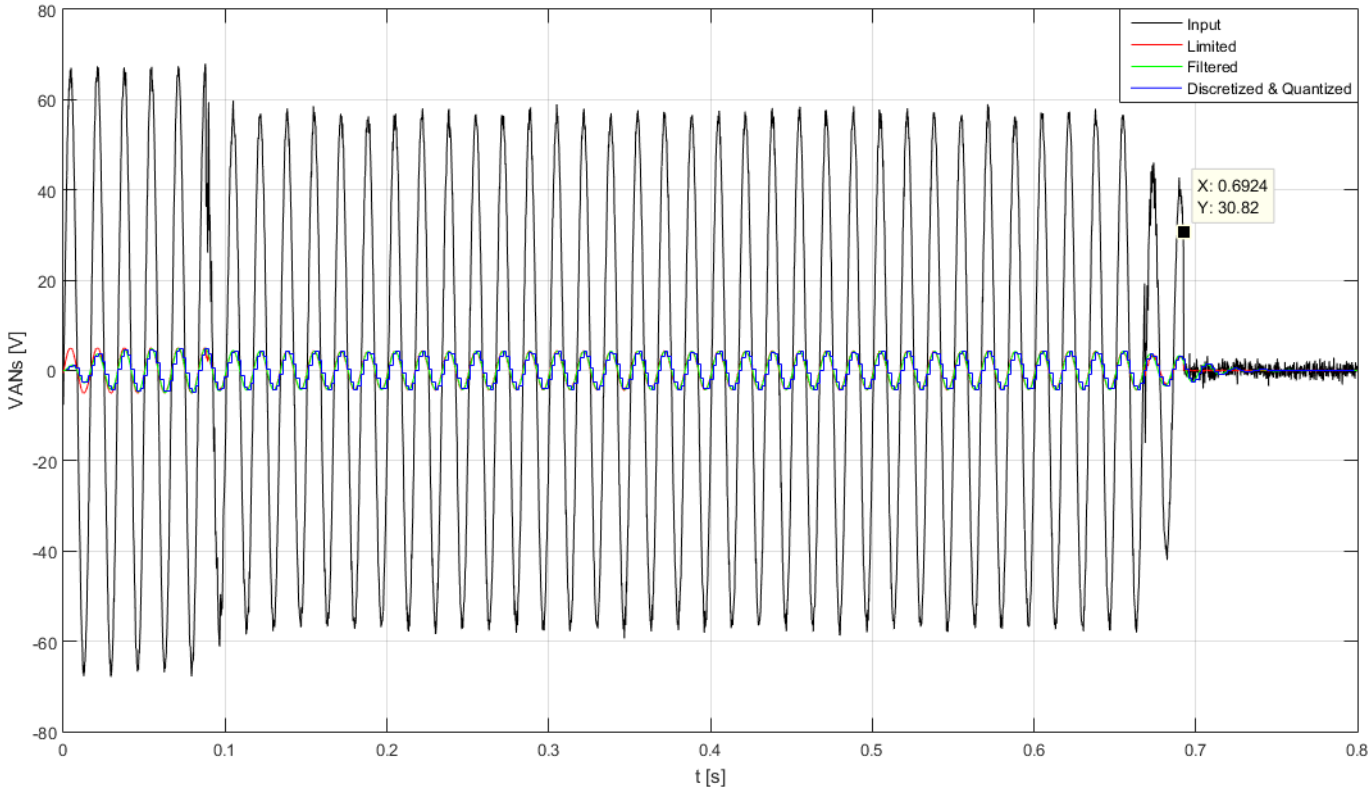

Figura 2.1: Processamento da fase VANs.

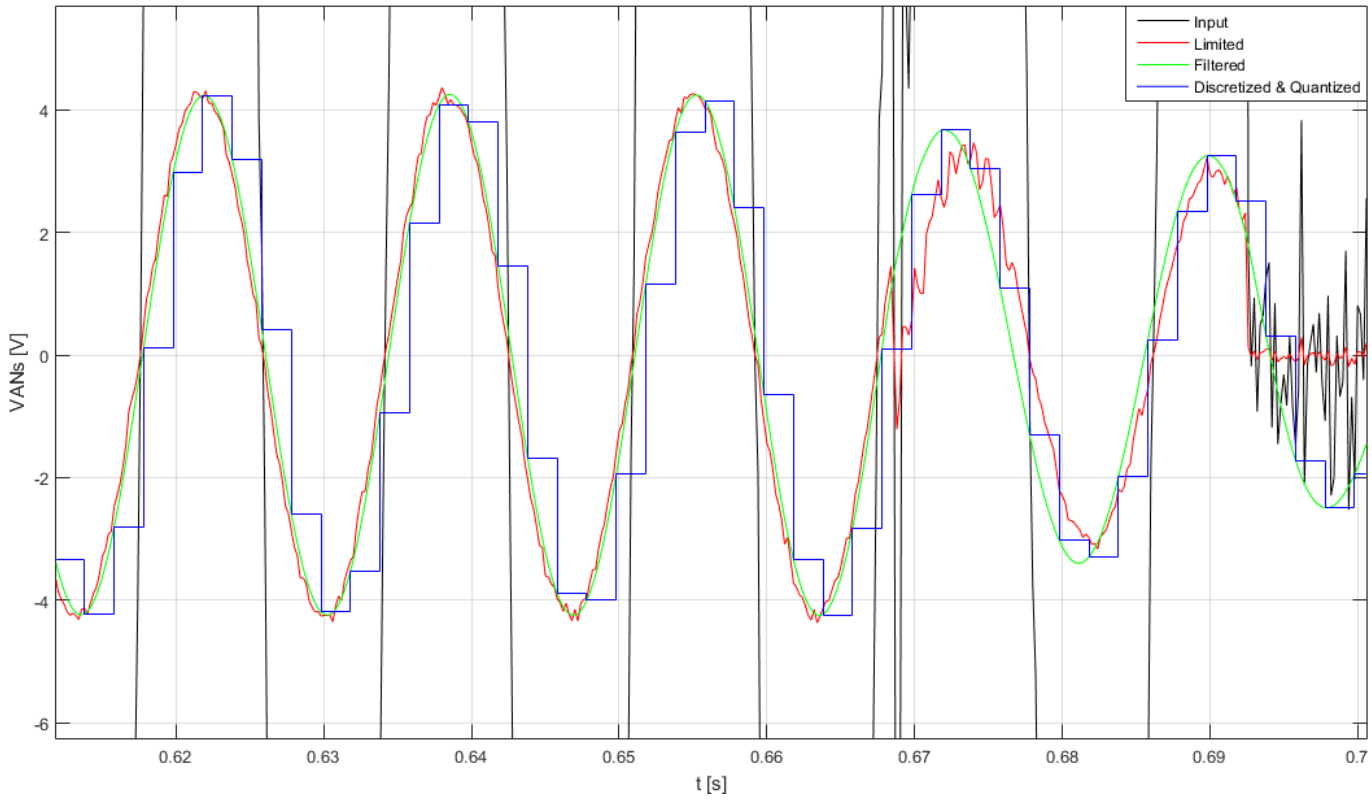

Figura 2.2: Zoom da fase VANs.

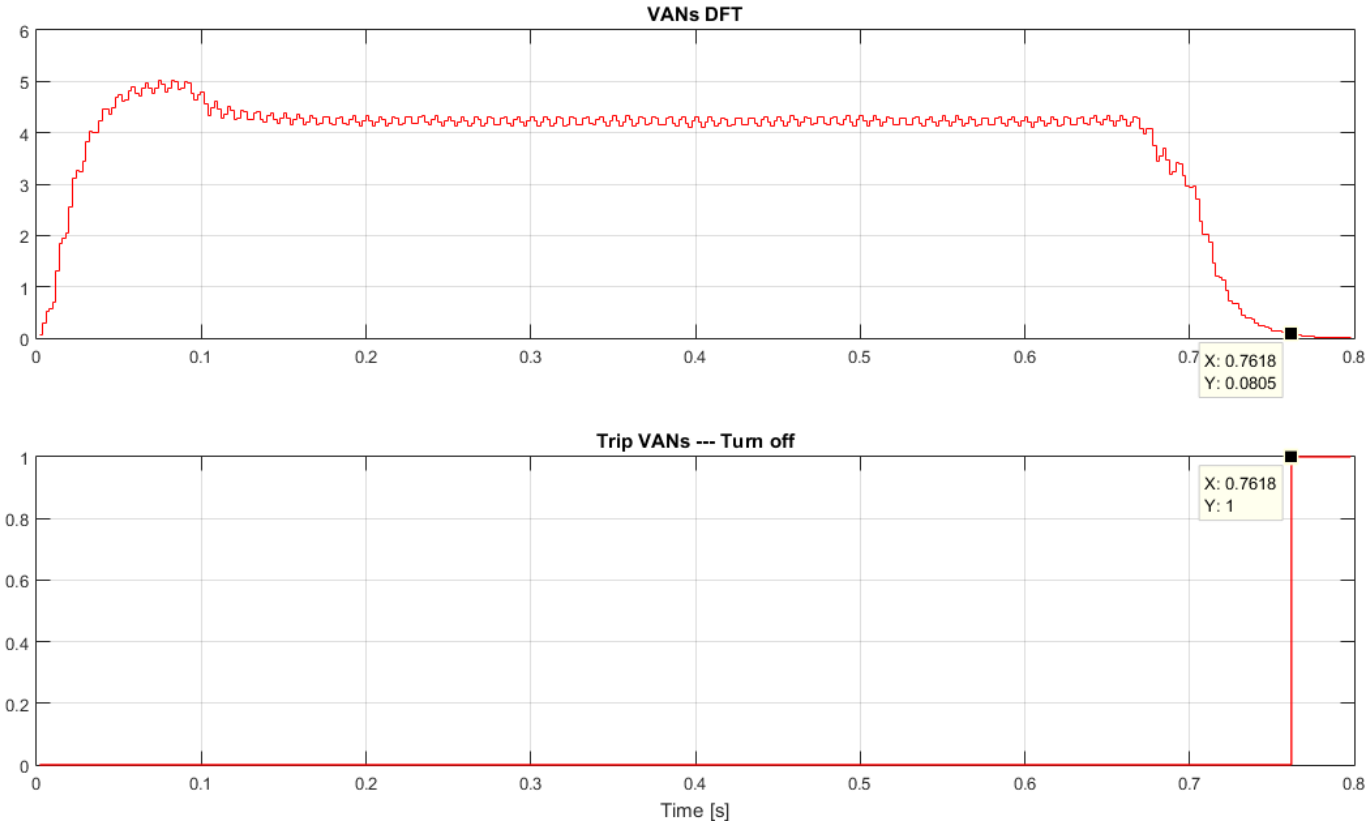

Figura 2.3: Estimação e classificação na fase VANs.

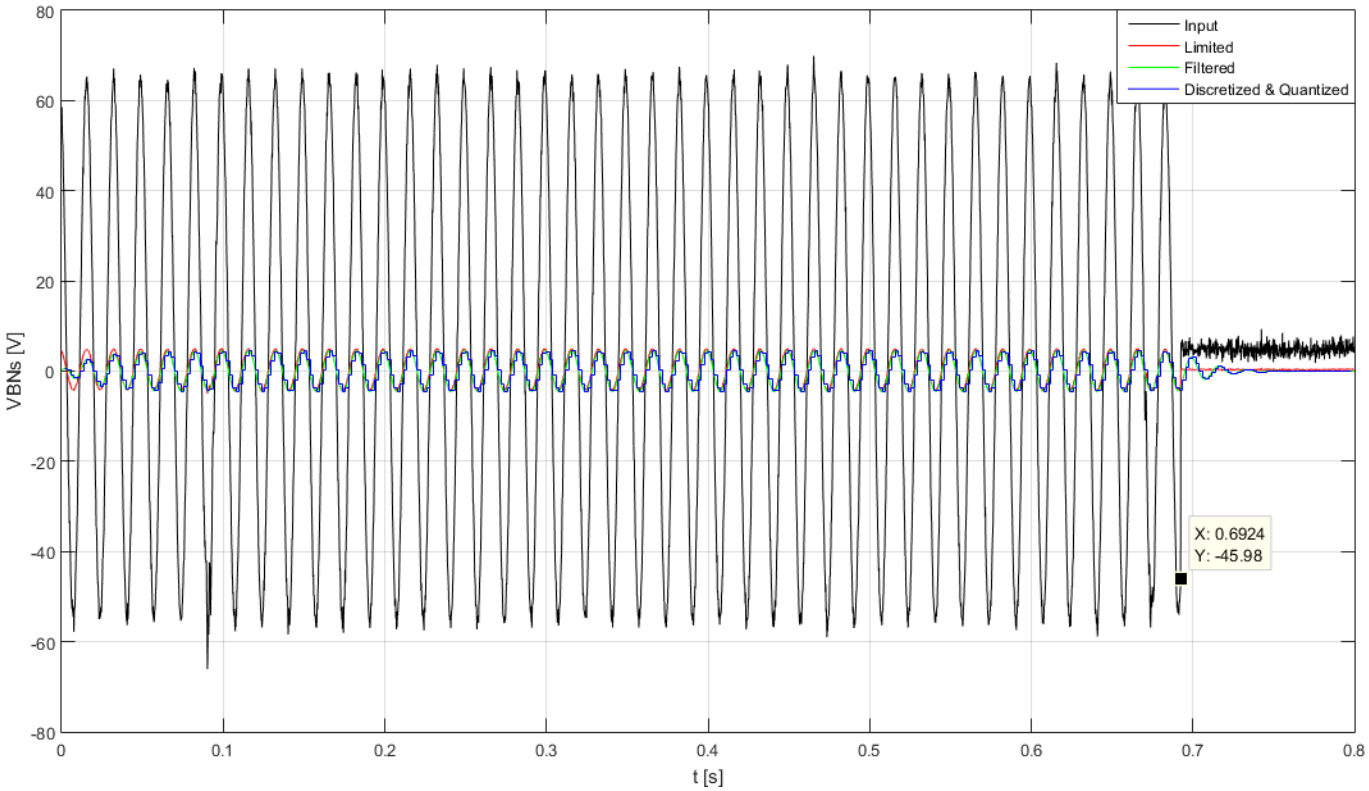

Figura 2.4: Processamento da fase VBNs.

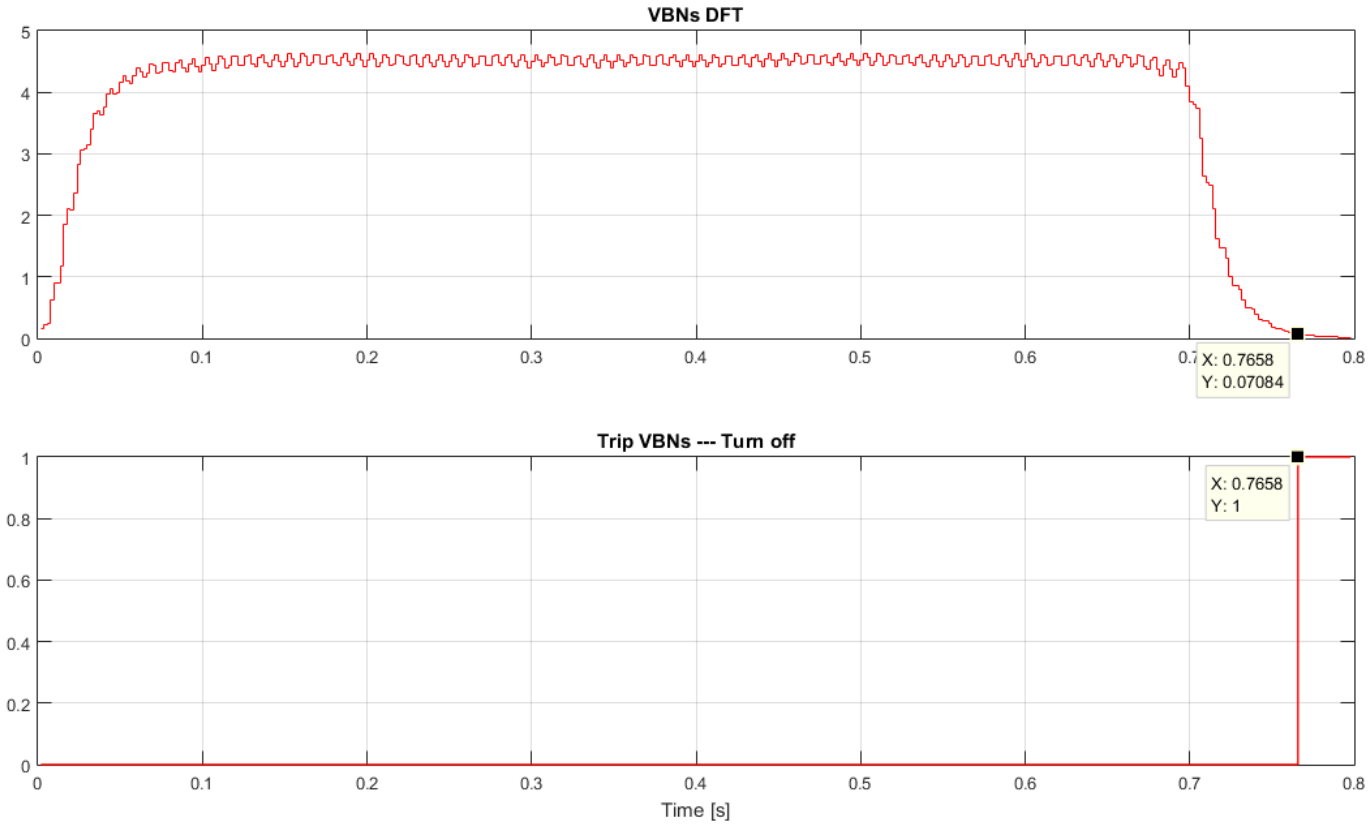

Figura 2.5: Estimação e classificação na fase VBNs.

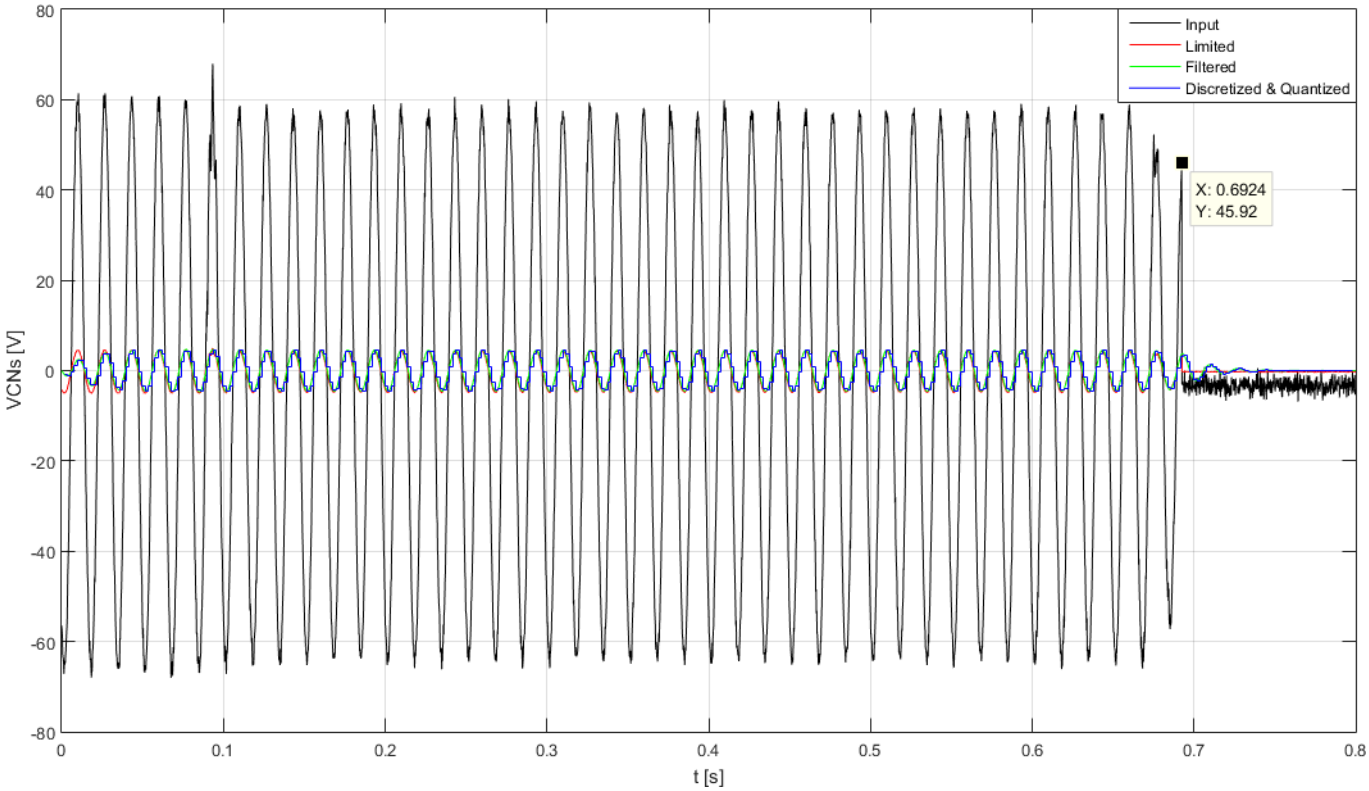

Figura 2.6: Processamento da fase VCNs.

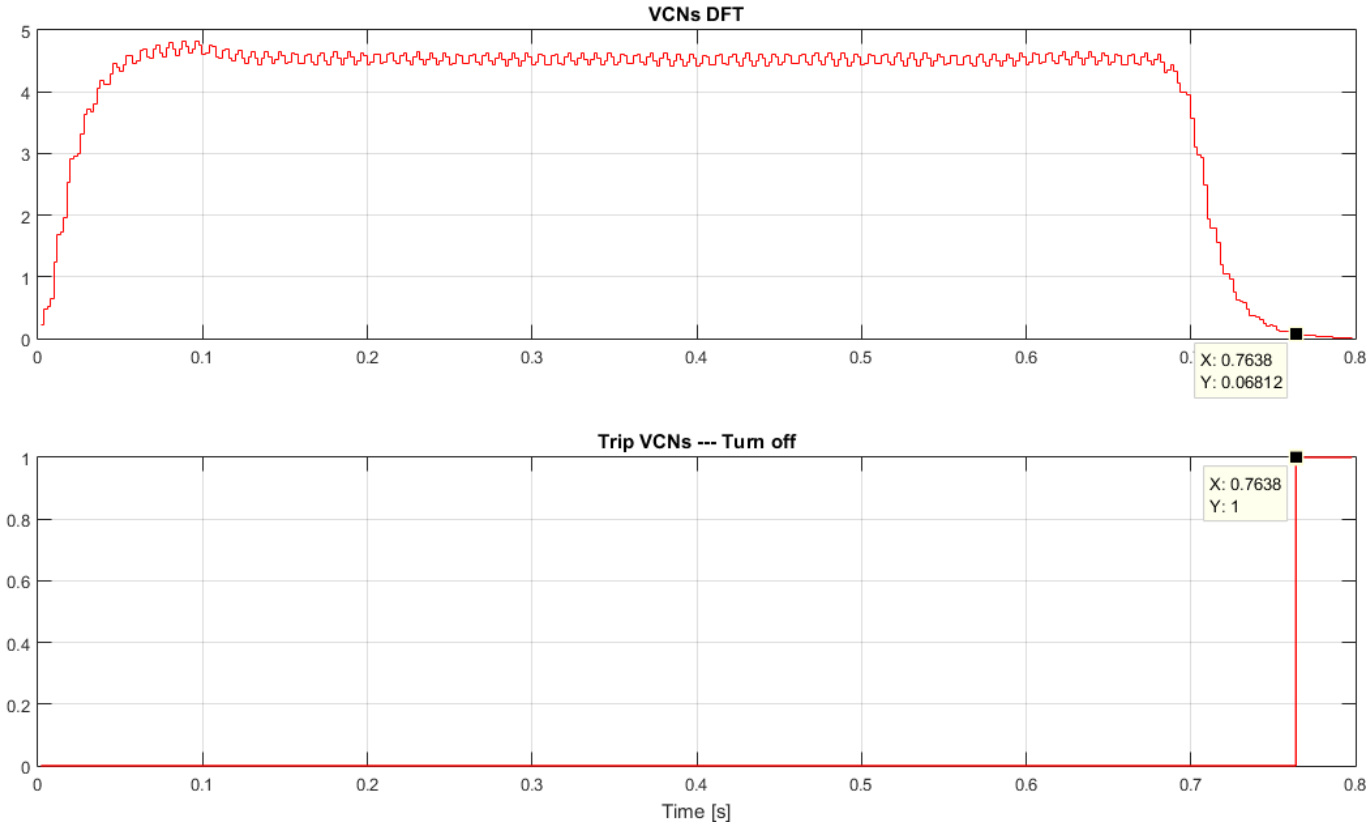

Figura 2.7: Estimação e classificação na fase VCNs.

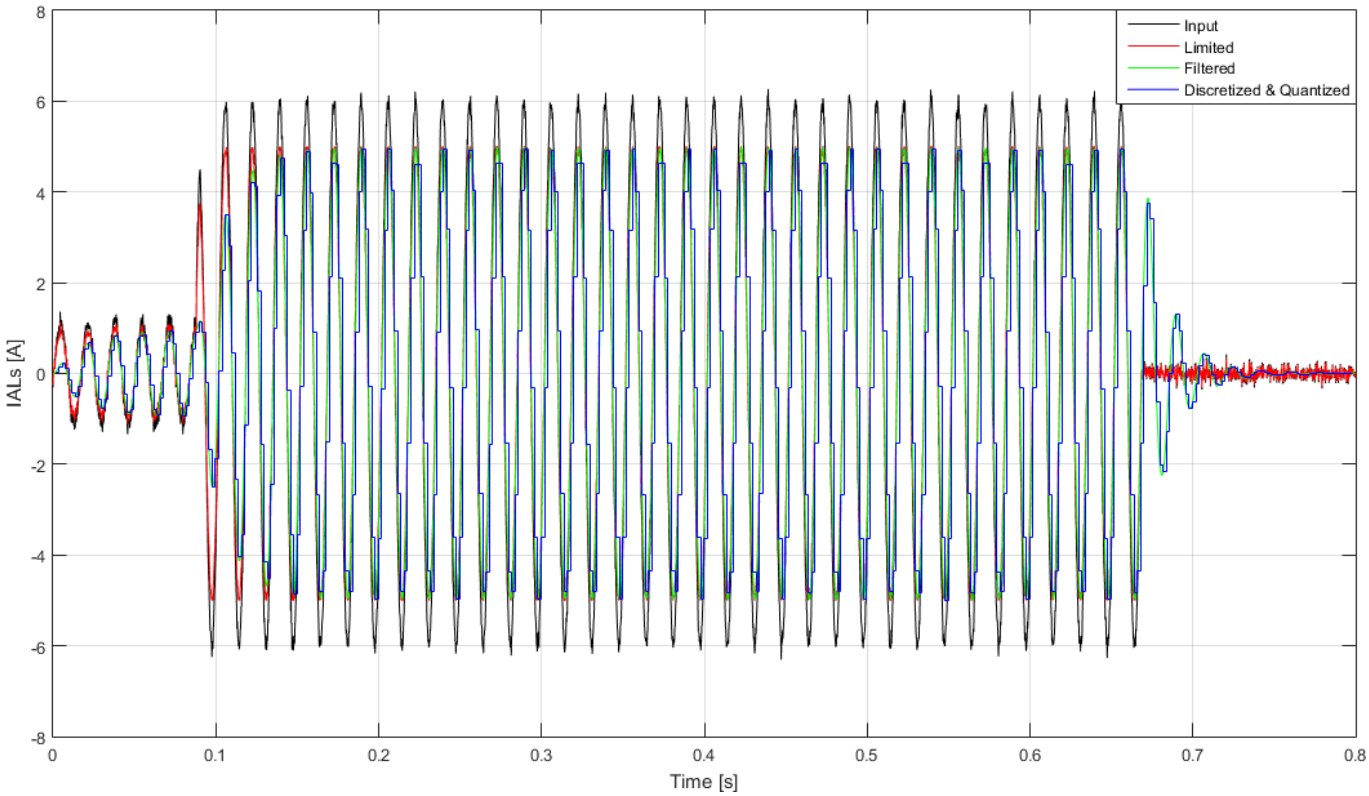

Figura 2.8: Processamento da fase IALs.

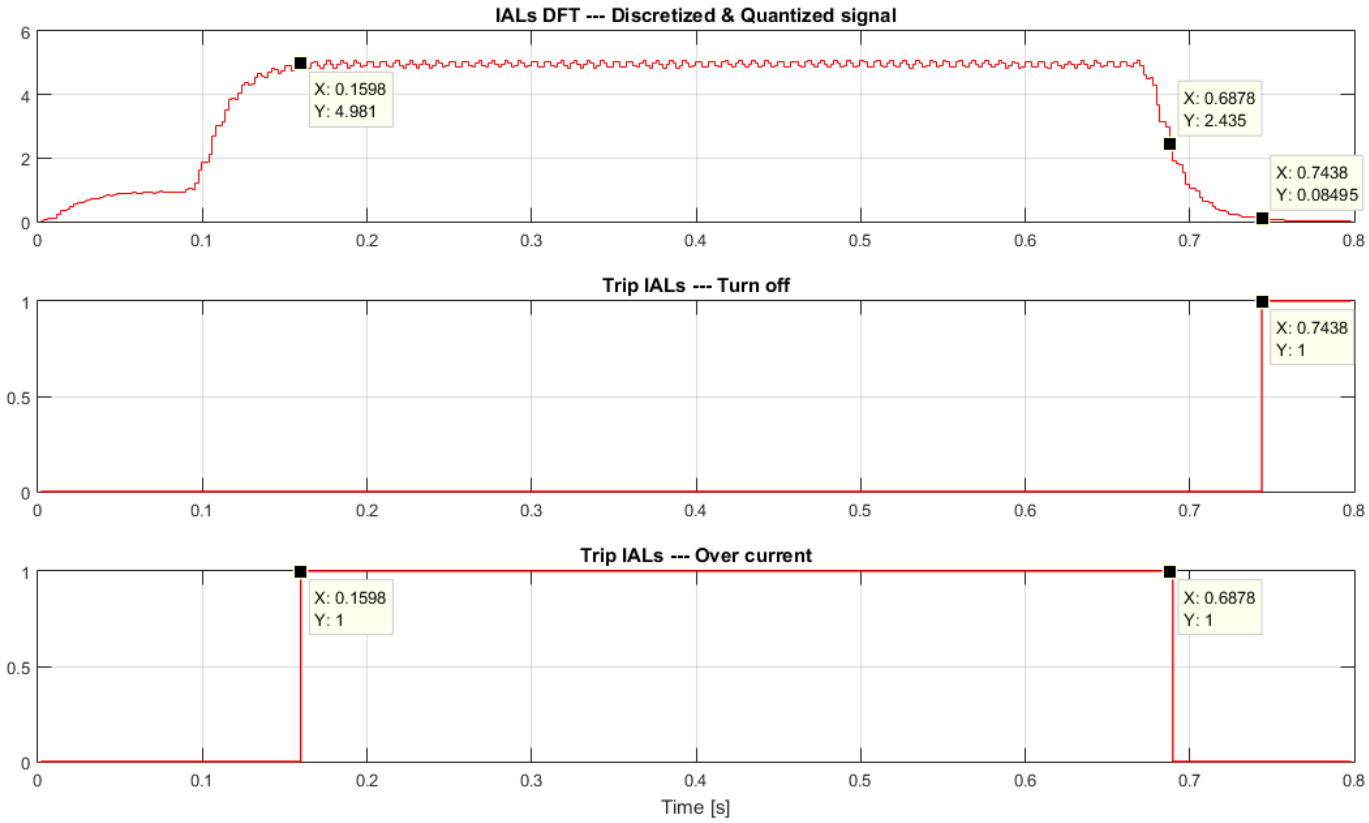

Figura 2.9: Estimação e classificação na fase IALs.

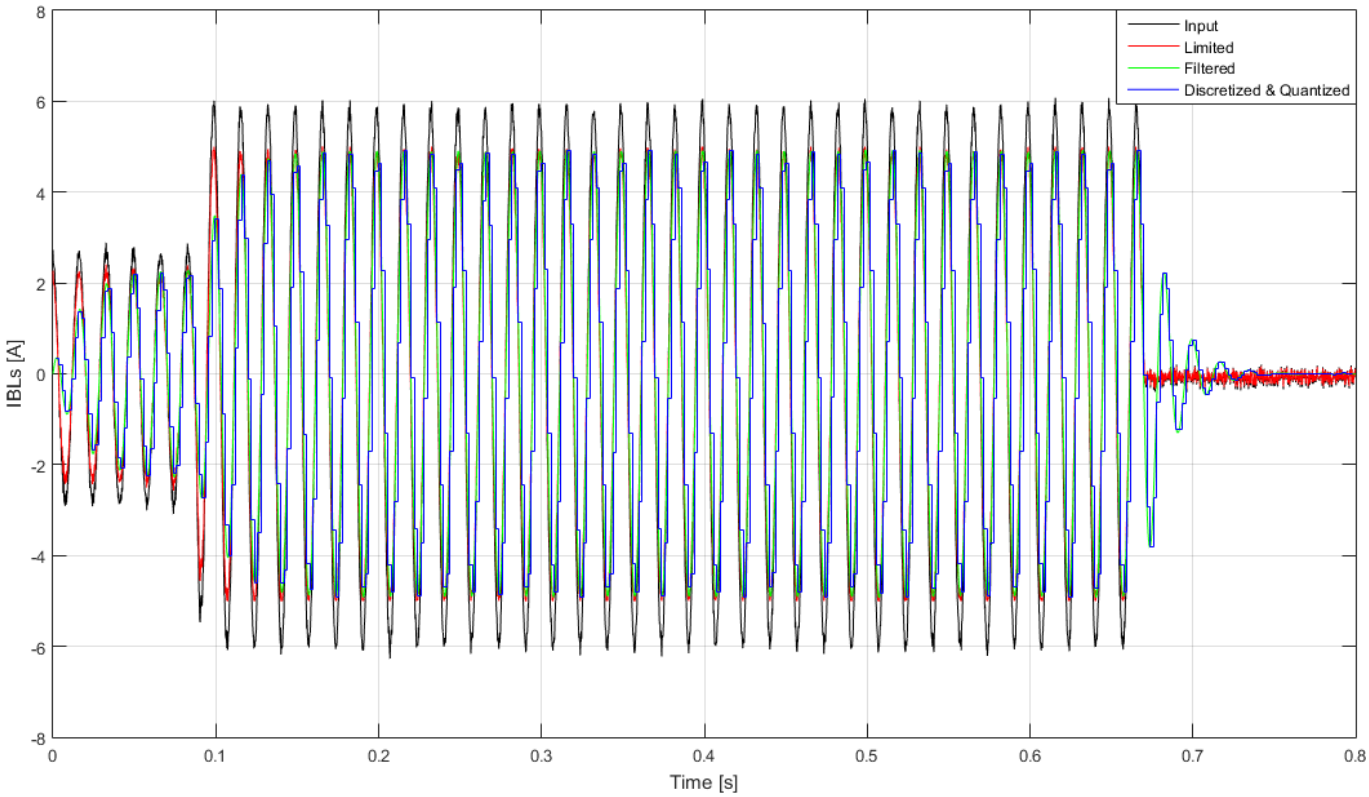

Figura 2.10: Processamento da fase IBLs.

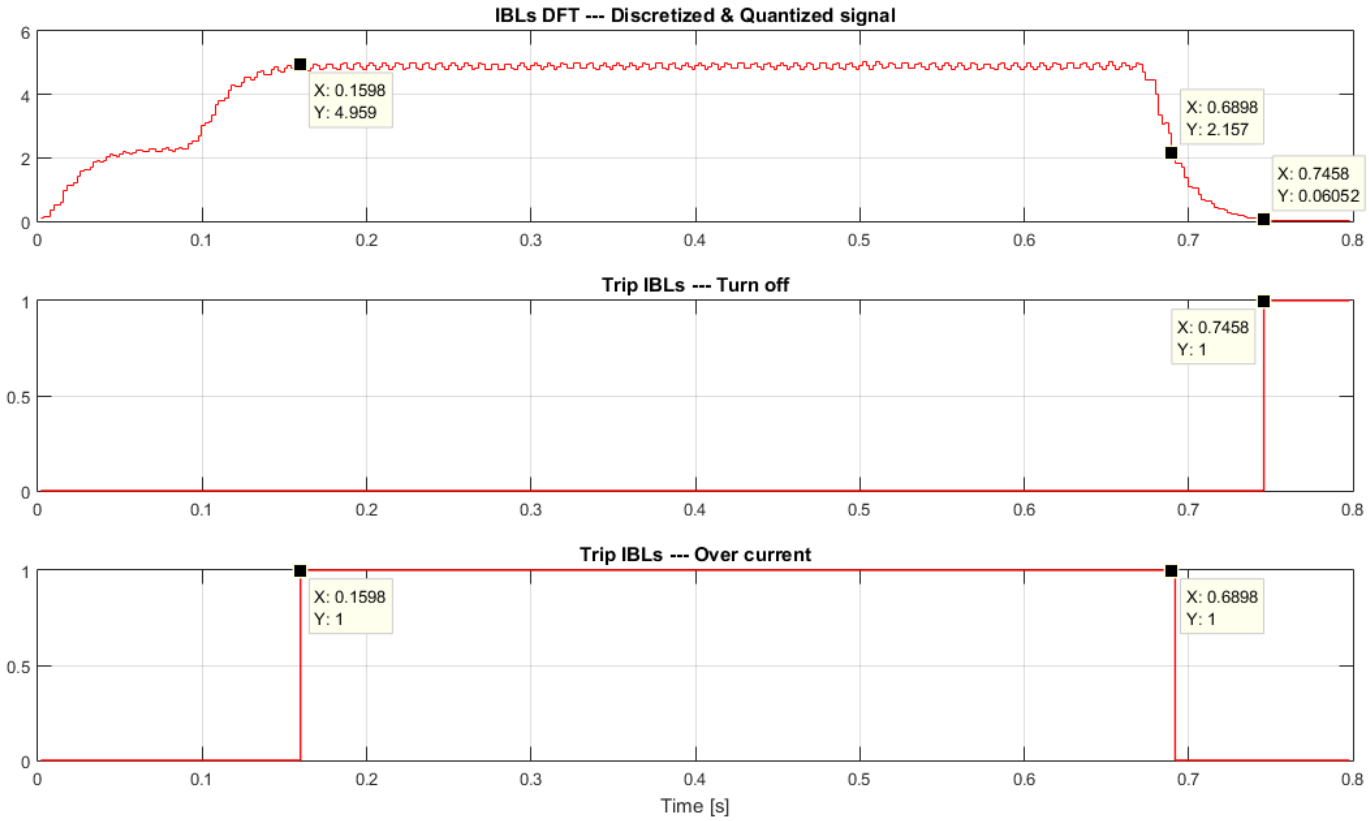

Figura 2.11: Estimação e classificação na fase IBLs.

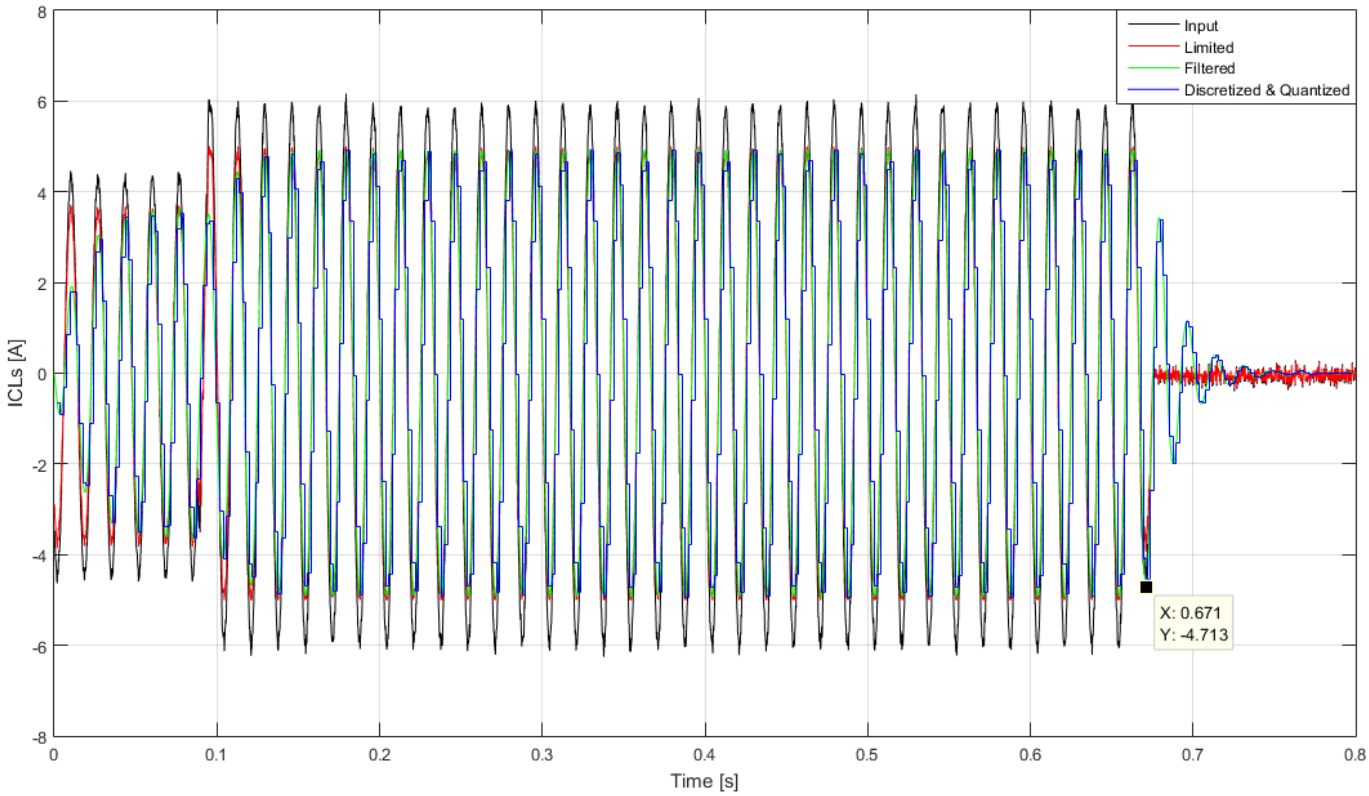

Figura 2.12: Processamento da fase ICLs.

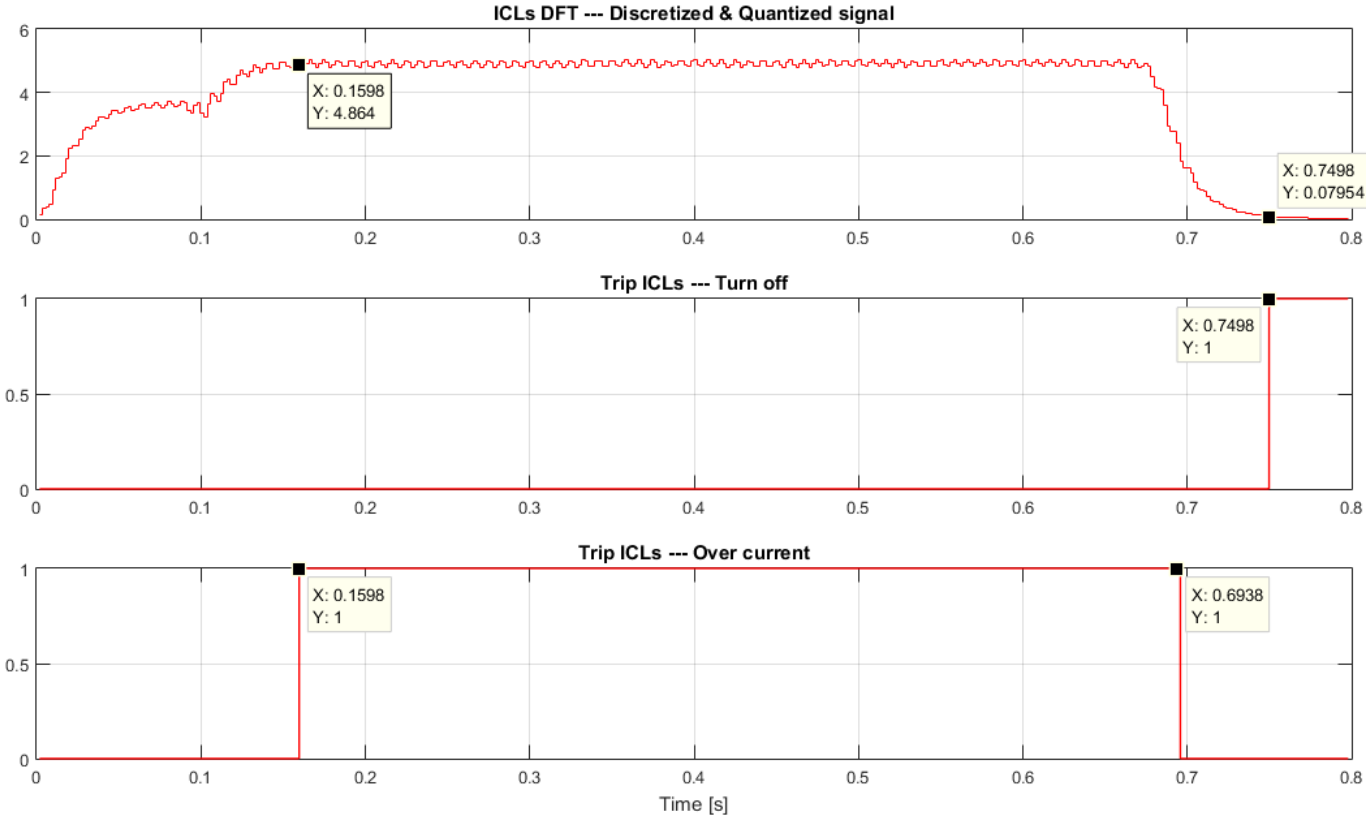

Figura 2.13: Estimação e classificação na fase ICLs.

Capítulo 3

Conclusão

Como a literatura clássica e técnica da área de relês digitais [6, 10] indica, é plenamente possível implementar sistemas digitais para lidar com a proteção de linha de transmissão. E com a vantagem de serem muito mais flexíveis em relação aos seus concorrentes analógicos.

Neste trabalho, não foi avaliado o impacto do tempo de processamento pelo microcontrolador (ou DSP) até a tomada de decisão, mas com o uso do CORDIC em hardware ou *lookup tables* o estimador de magnitude de harmônica se torna um algoritmo veloz. E o classificador de *trip* é formado por um pequeno conjunto de clausulas condicionais, o que também é veloz. Então, o atraso de processamento não deve ser algo preocupante.

A técnica de *digital relaying* certamente gera muitos parâmetros que precisam ser ajustados para garantir um tempo de resposta adequado. Além disso, o projeto do filtro analógico também demanda atenção pois ele pode atrasar o sinal de entrada no ADC e, consequente, atrasar a tomada de decisão. A exemplo dos 4 ciclos de resposta do *trip* de desligamento, algumas modificações neste filtro precisam ser feitas para reduzir a resposta para apenas 2 ciclos.

Embora o estimador implementado seja baseado em DFT janelada de ciclo completo, existem vários outras maneiras de projetar um estimado de harmônica. Um estimador recursivo muito utilizado para esta finalidade é o filtro de Kalman [3,9], que apresenta excelente robustez com relação a ruídos gaussianos, sinais correlacionados e perturbações constantes ou lentamente variáveis. Outro estimador é o PLL (*Phase Locked Loop*) [1,8], que pode ser implementado tanto em hardware quanto em software. Ele é um sistema de controle automático que permite a extração da informação sobre magnitude e fase de um sinal de maneira precisa.

Referências Bibliográficas

[1] A. Antonelli, S. Giarnetti, and F. Leccese. Pll system for harmonic analysis. In *2011 10th International Conference on Environment and Electrical Engineering*, pages 1–5, 2011.

[2] H. Huang and L. Xiao. Cordic based fast radix-2 dct algorithm. *IEEE Signal Processing Letters*, 20(5):483–486, 2013.

[3] K. Kennedy, G. Lightbody, and R. Yacamini. Power system harmonic analysis using the kalman filter. In *2003 IEEE Power Engineering Society General Meeting (IEEE Cat. No.03CH37491)*, volume 2, pages 752–757 Vol. 2, 2003.

[4] V. H. Makwana and B. R. Bhalja. *Transmission line protection using digital technology*. Springer, 1 edition, 2016.

[5] U. Patel, P. Bhatt, and N. Chothani. *Futuristic trends in numerical relaying for transmission line protections*. Springer, 1 edition, 2020.

[6] A. G. Phadke and J. S. Thorp. *Computer relaying for power systems*. Wiley, 2 edition, 2009.

[7] W. Rebizant, J. Szafran, and A. Wiszniewski. *Digital signal processing in power system protection and control*. Springer, 1 edition, 2011.

[8] C. H. G. Santos, R. V. Ferreira, S. M. Silva, and B. J. Cardoso Filho. Fourier-based pll applied for selective harmonic estimation in electric power systems,. *Journal of Power Electronics*, 13(5):884–895, 2013.

[9] M-C. Shin and C-H. Kim. A study on the digital distance relaying scheme using kalman filter. In UHI AHN, editor, *Power Systems and Power Plant Control 1989*, IFAC Symposia Series, pages 345–350. Pergamon, Oxford, 1990.

[10] G. Ziegler. *Numerical distance protection: Principles and application*. Publicis, 1 edition, 2011.

Apêndice A

Códigos em Matlab

A.1 Gerador dos sinais de entrada

```
close, clear, clc;
M = dlmread('Case4.csv');

m = 4000;
%m = max(size(M));
n = 1:m;

t = M(n,2) * 1e-6; % [s] ---- Medido originalmente em us

v_amp = 115 / sqrt(3); % [V]
vANs = M(n,3);
vANs = cond_prot(vANs, 0, 4096, -v_amp, v_amp);
vANs = noise_gen(vANs', 30);
vBNs = M(n,4);
vBNs = cond_prot(vBNs, 0, 4096, -v_amp, v_amp);
vBNs = noise_gen(vBNs', 30);
vCNs = M(n,5);
vCNs = cond_prot(vCNs, 0, 4096, -v_amp, v_amp);
vCNs = noise_gen(vCNs', 30);

i_amp = 6; % [A]
iALs = M(n,6);
iALs = cond_prot(iALs, 0, 4096, -i_amp, i_amp);
iALs = noise_gen(iALs', 30);
iBLs = M(n,7);
iBLs = cond_prot(iBLs, 0, 4096, -i_amp, i_amp);
iBLs = noise_gen(iBLs', 30);
iCLs = M(n,8);
iCLs = cond_prot(iCLs, 0, 4096, -i_amp, i_amp);
iCLs = noise_gen(iCLs', 30);

Mw = [t'; vANs; vBNs; vCNs; iALs; iBLs; iCLs];
dlmwrite('Case4_mod.csv',Mw);

figure(1);
subplot(211), plot(t,vANs, t,vBNs, t,vCNs), grid on;
subplot(212), plot(t,iALs, t,iBLs, t,iCLs), grid on;
```

A.2 Condicionamento e proteção interna

```
function y = cond_prot (x, inL, inU, outL, outU)
N = length(x);

%%% Conditioning --- Linear Mapping
% inL = Lower input voltage
% inU = Lower input voltage
% outL = Lower output voltage
% outU = Upper output voltage
y = (outU - outL) * (x - inL) / (inU - inL) + outL;

%%% Protection --- Limiter [y1, y2]
for i = 1:N
  if (y(i) < outL)
    y(i) = outL;
  elseif (y(i) > outU)
    y(i) = outU;
  end
end
```

A.3 Gerador de ruído

```
function [y, sd_v] = noise_gen (x, SNR_dB)
% SNR_dB = signal-to-noise ratio [dB]
N = length(x);
sd_v = sqrt(var(x) + mean(x)^2) * 10^(-SNR_dB/20);
y = x + sd_v * randn(1,N);
```

A.4 Programa principal

```
close, clear, clc;
%pkg load signal; % Octave

M = dlmread('Case4_mod.csv');
t = M(1,:); % [s]
VANs = M(2,:); % [V]
VBNs = M(3,:);
VCNs = M(4,:);
IALs = M(5,:); % [kA] --- [A]
IBLs = M(6,:);
ICLs = M(7,:);

%%% Conditioning & Protecting
v_amp = 115 / sqrt(3); % [V]
i_amp = 6; % [A]
in_amp = 5; % [V]
VANs_lim = cond_prot(VANs, -v_amp, v_amp, -in_amp, in_amp);
VBNs_lim = cond_prot(VBNs, -v_amp, v_amp, -in_amp, in_amp);
VCNs_lim = cond_prot(VCNs, -v_amp, v_amp, -in_amp, in_amp);
IALs_lim = cond_prot(IALs, -i_amp, i_amp, -in_amp, in_amp);
IBLs_lim = cond_prot(IBLs, -i_amp, i_amp, -in_amp, in_amp);
ICLs_lim = cond_prot(ICLs, -i_amp, i_amp, -in_amp, in_amp);

%%% Analog Filter
fn = 60; % fundamental frequency [Hz]
passband = 20;
VANs_fil = passband_butter_resp(VANs_lim, t, fn, passband);
VBNs_fil = passband_butter_resp(VBNs_lim, t, fn, passband);
VCNs_fil = passband_butter_resp(VCNs_lim, t, fn, passband);
IALs_fil = passband_butter_resp(IALs_lim, t, fn, passband);
IBLs_fil = passband_butter_resp(IBLs_lim, t, fn, passband);
ICLs_fil = passband_butter_resp(ICLs_lim, t, fn, passband);

%%% Sampling
sppc = 8; % sampled points per cycle
[td, VANs_fd, Ts, Ns, pd] = sampling(t, VANs_fil, sppc, fn);
VBNs_fd = samp_fast(VBNs_fil, Ns, pd);
VCNs_fd = samp_fast(VCNs_fil, Ns, pd);
IALs_fd = samp_fast(IALs_fil, Ns, pd);
IBLs_fd = samp_fast(IBLs_fil, Ns, pd);
ICLs_fd = samp_fast(ICLs_fil, Ns, pd);

%%% Quantization
nbits = 10;
VANs_fdq = quantization(VANs_fd, nbits, -in_amp, in_amp);
VBNs_fdq = quantization(VBNs_fd, nbits, -in_amp, in_amp);
VCNs_fdq = quantization(VCNs_fd, nbits, -in_amp, in_amp);
IALs_fdq = quantization(IALs_fd, nbits, -in_amp, in_amp);
IBLs_fdq = quantization(IBLs_fd, nbits, -in_amp, in_amp);
ICLs_fdq = quantization(ICLs_fd, nbits, -in_amp, in_amp);
```

```
%%% Harmonic Estimation
nw = sppc; % DFT window size
dft_cw = zeros(1,nw); % DFT cossine window
dft_sw = zeros(1,nw); % DFT sine window
VANs_dftw = zeros(1,nw); % DFT VANs window
VANs_dft = zeros(1,Ns);  % DFT VANs
VBNs_dftw = zeros(1,nw); % DFT VBNs window
VBNs_dft = zeros(1,Ns);  % DFT VBNs
VCNs_dftw = zeros(1,nw); % DFT VCNs window
VCNs_dft = zeros(1,Ns);  % DFT VCNs
IALs_dftw = zeros(1,nw); % DFT IALs window
IALs_dft = zeros(1,Ns);  % DFT IALs
IBLs_dftw = zeros(1,nw); % DFT IBLs window
IBLs_dft = zeros(1,Ns);  % DFT IBLs
ICLs_dftw = zeros(1,nw); % DFT ICLs window
ICLs_dft = zeros(1,Ns);  % DFT ICLs
for k = 1:Ns % DFT
  [na, dft_cw, dft_sw] = dft_cs(k, dft_cw, dft_sw, fn, Ts, nw);
  [VANs_dft(k), VANs_dftw] = dft_mag(VANs_fdq(k), VANs_dftw, ...
      dft_cw, dft_sw, nw, na);
  [VBNs_dft(k), VBNs_dftw] = dft_mag(VBNs_fdq(k), VBNs_dftw, ...
      dft_cw, dft_sw, nw, na);
  [VCNs_dft(k), VCNs_dftw] = dft_mag(VCNs_fdq(k), VCNs_dftw, ...
      dft_cw, dft_sw, nw, na);
  [IALs_dft(k), IALs_dftw] = dft_mag(IALs_fdq(k), IALs_dftw, ...
      dft_cw, dft_sw, nw, na);
  [IBLs_dft(k), IBLs_dftw] = dft_mag(IBLs_fdq(k), IBLs_dftw, ...
      dft_cw, dft_sw, nw, na);
  [ICLs_dft(k), ICLs_dftw] = dft_mag(ICLs_fdq(k), ICLs_dftw, ...
      dft_cw, dft_sw, nw, na);
end

%%% Tripping
% Voltage turn off
v_to = cond_prot(v_amp/10, -v_amp, v_amp, -in_amp, in_amp);
% Current turn off
i_to = cond_prot(i_amp/10, -i_amp, i_amp, -in_amp, in_amp);
pc_to = nw*2; % Points cycle turn off --- double cycle
i_oc = cond_prot(2.5, -i_amp, i_amp, -in_amp, in_amp); % Over current
pc_oc = nw/2; % Points cycle over current --- half cycle
t_init = 0.15; % Initializing time [us]
VANs_trip_to = trip_fault_v(td, VANs_dft, v_to, pc_to, t_init);
VBNs_trip_to = trip_fault_v(td, VBNs_dft, v_to, pc_to, t_init);
VCNs_trip_to = trip_fault_v(td, VCNs_dft, v_to, pc_to, t_init);
[IALs_trip_to, IALs_trip_oc] = trip_fault_i(td, IALs_dft, i_to, ...
    pc_to, i_oc, pc_oc, t_init);
[IBLs_trip_to, IBLs_trip_oc] = trip_fault_i(td, IBLs_dft, i_to, ...
    pc_to, i_oc, pc_oc, t_init);
[ICLs_trip_to, ICLs_trip_oc] = trip_fault_i(td, ICLs_dft, i_to, ...
    pc_to, i_oc, pc_oc, t_init);
```

```
%%% Plot
plot_v('VANs', 0, t, td, VANs, VANs_lim, VANs_fil, VANs_fdq, ...
    VANs_dft, VANs_trip_to);
plot_v('VBNs', 1, t, td, VBNs, VBNs_lim, VBNs_fil, VBNs_fdq, ...
    VBNs_dft, VBNs_trip_to);
plot_v('VCNs', 2, t, td, VCNs, VCNs_lim, VCNs_fil, VCNs_fdq, ...
    VCNs_dft, VCNs_trip_to);
plot_i('IALs', 3, t, td, IALs, IALs_lim, IALs_fil, IALs_fdq, ...
    IALs_dft, IALs_trip_to, IALs_trip_oc);
plot_i('IBLs', 4, t, td, IBLs, IBLs_lim, IBLs_fil, IBLs_fdq, ...
    IBLs_dft, IBLs_trip_to, IBLs_trip_oc);
plot_i('ICLs', 5, t, td, ICLs, ICLs_lim, ICLs_fil, ICLs_fdq, ...
    ICLs_dft, ICLs_trip_to, ICLs_trip_oc);
```

A.5 Filtro passa faixa

```
function y = passband_butter_resp (x, t, fcenter, passband)
%passband = frequency passband [Hz]
%fcenter = center frequency [Hz]
fcl = fcenter - passband/2; % lower cuttoff frequency Hz
fcu = fcenter + passband/2; % upper cuttoff frequency Hz
% Band-Pass 2nd order Butterworth analog filter
[num_btw, den_btw] = butter(1, 2*pi*[fcl, fcu], 's');
sys_btw = tf(num_btw, den_btw); % analog filter transfer function
y = lsim(sys_btw, x, t); % output signal
```

A.6 Amostragem

```
function [td, yd, Ts, Ns, pd] = sampling (t, y, pc, fb)
% pc = sampled points per cycle
% fb = max frequency boundary
N = length(t);
fs = pc * fb; % Sampling frequency
Ts = 1 / fs; % Sampling period
dt = t(N) - t(1); % time variation
Ns = round(dt * fs); % number of sampled points
pd = floor(N / Ns); % point distance
Ns = Ns + floor((t(N)-t(pd*Ns))/Ts); % adjust Ns
td = zeros(1,Ns); % discrete time
td(1) = t(1);
yd = zeros(1,Ns); % discrete signal
yd(1) = y(1);
for i = 1:Ns
    td(i) = t(pd*i);
    yd(i) = y(pd*i);
end
```

A.7 Amostragem rápida

```
function yd = samp_fast (y, Ns, pd)
yd = zeros(1,Ns); % discrete signal
yd(1) = y(1);
for i = 1:Ns
    yd(i) = y(pd*i);
end
```

A.8 Quantização

```
function ydq = quantization (yd, nbits, outL, outU)
%nbits = ADC bits
qLevels = 2^nbits; % 1024 qantization levels
% outL = Lower output voltage
% outU = Upper output voltage
scalingFactor = (outU - outL) / qLevels;
ydq = round(yd / scalingFactor) * scalingFactor;
```

A.9 Componentes seno e cosseno da DFT

```
function [n, uc, us] = dft_cs (k, uc, us, fn, Ts, nw)
psi = 2*pi*fn * Ts; % = 2*pi/nw
%cs = [cos(psi*k), sin(psi*k)];
cs = cordic(psi*k, 28);

% Array shift
%N = length(uc);
%uc = [uc(2:N), cs(1)]; % Shift left array
%us = [us(2:N), cs(2)]; % Shift left array

% Circular array
n = mod(k, nw);
if n == 0
    n = nw;
end
uc(n) = cs(1);
us(n) = cs(2);
```

A.10 CORDIC modo de rotação

```
function v = cordic(beta, n)
% This function computes v = [cos(beta), sin(beta)] (beta in radians)
% using n iterations. Increasing n will increase the precision.

if ((beta < -pi/2) || (beta > pi/2))
    if (beta < 0)
        v = cordic(beta + pi, n);
    else
        v = cordic(beta - pi, n);
    end
    v = -v; % flip the sign for second or third quadrant
    return
end

% Initialization of tables of constants used by CORDIC
% need a table of arctangents of negative powers of two, in radians:
% angles = atan(2.^-(0:27));
angles =  [...
0.78539816339745, 0.46364760900081, 0.24497866312686, 0.12435499454676, ...
0.06241880999596, 0.03123983343027, 0.01562372862048, 0.00781234106010, ...
0.00390623013197, 0.00195312251648, 0.00097656218956, 0.00048828121119, ...
0.00024414062015, 0.00012207031189, 0.00006103515617, 0.00003051757812, ...
0.00001525878906, 0.00000762939453, 0.00000381469727, 0.00000190734863, ...
0.00000095367432, 0.00000047683716, 0.00000023841858, 0.00000011920929, ...
0.00000005960464, 0.00000002980232, 0.00000001490116, 0.00000000745058];
% and a table of products of reciprocal lengths of vectors [1, 2^-2j]:
% Kvalues = cumprod(1./abs(1 + 1j*2.^(-(0:23))))
Kvalues = [...
0.70710678118655, 0.63245553203368, 0.61357199107790, 0.60883391251775, ...
0.60764825625617, 0.60735177014130, 0.60727764409353, 0.60725911229889, ...
0.60725447933256, 0.60725332108988, 0.60725303152913, 0.60725295913894, ...
0.60725294104140, 0.60725293651701, 0.60725293538591, 0.60725293510314, ...
0.60725293503245, 0.60725293501477, 0.60725293501035, 0.60725293500925, ...
0.60725293500897, 0.60725293500890, 0.60725293500889, 0.60725293500888];
Kn = Kvalues(min(n, length(Kvalues)));

% Initialize loop variables:
v = [1;0]; % start with 2-vector cosine and sine of zero
poweroftwo = 1;
angle = angles(1);

% Iterations
for j = 2:n;
    if beta < 0
        sigma = -1;
    else
        sigma = 1;
    end
    factor = sigma * poweroftwo;
    % Note the matrix multiplication can be done using scaling
```

```
    %   by powers of two and addition subtraction
    R = [1, -factor; factor, 1];
    v = R * v; % 2-by-2 matrix multiply
    beta = beta - sigma * angle; % update the remaining angle
    poweroftwo = poweroftwo / 2;
    % update the angle from table, or eventually by just dividing by two
    if j > length(angles)
        angle = angle / 2;
    else
        angle = angles(j);
    end
end

% Adjust length of output vector to be [cos(beta), sin(beta)]:
v = v' * Kn;
```

A.11 Estimação da componente fundamental por DFT

```
function [y, ux] = dft_mag (x, ux, uc, us, nw, na)
%ux = [ux(2:nw), x]; % Shift left array
ux(na) = x; % circular array
% Yc = 2 * (ux*uc') / nw
% Ys = 2 * (ux*us') / nw
% |Y| = sqrt(Yc^2 + Ys^2) =
y = 2 * sqrt((ux*uc')^2 + (ux*us')^2) / nw; % DFT
```

A.12 Trip por desligamento de tensão

```
function Trip_vto = trip_fault_v (t, V_dft, v_to, pc_to, t_init)
N = length(V_dft);
Trip_vto = zeros(1,N); % Trip voltage turn off
c = 0; % counter
for i = 1:N
  if t(i) > t_init
    if V_dft(i) < v_to
      c = c + 1;
      if c > pc_to-1
        Trip_vto(i) = 1; % Turn off detection
      end
    else
      c = 0; % counter reset
    end
  end
end
```

A.13 Trip por sobrecorrente ou desligamento de corrente

```
function [Trip_ito, Trip_ioc] = trip_fault_i (t, I_dft, i_to, ...
        pc_to, i_oc, pc_oc, t_init)
N = length(I_dft);
Trip_ito = zeros(1,N); % Trip current turn off
Trip_ioc = zeros(1,N); % Trip over current
c_to = 0; % counter turn off
c_oc = 0; % counter over current
for i = 1:N
  if t(i) > t_init
    if I_dft(i) < i_to
      c_to = c_to + 1;
      if c_to > pc_to-1
        Trip_ito(i) = 1; % Turn off detection
      end
    else
      c_to = 0; % counter reset
      if I_dft(i) > i_oc
        c_oc = c_oc + 1;
        if c_oc > pc_oc
          Trip_ioc(i) = 1; % Overcurrent detection
        end
      else
        c_oc = 0; % counter reset
      end
    end
  end
end
```

A.14 Gráficos relacionados aos sinais de tensão

```
function plot_v (vnam, n, t, td, V, V_lim, V_fil, V_fdq, V_dft, V_trip)
figure(2*n+1), set(gcf,'color','w');
plot(t,V,'k', t,V_lim,'r', t,V_fil,'g'), hold on, stairs(td,V_fdq,'b');
    hold off, grid on;
legend('Input','Limited','Filtered','Discretized & Quantized');
xlabel('t [s]'), ylabel([vnam ' [V]']);

figure(2*n+2), set(gcf,'color','w');
subplot(211), stairs(td,V_dft,'r'), grid on, title([vnam ' DFT']);
subplot(212), stairs(td,V_trip,'r','LineWidth',1), grid on;
    title(['Trip ' vnam ' --- Turn off']), xlabel('Time [s]');
```

A.15 Gráficos relacionados aos sinais de corrente

```
function plot_i (inam, n, t, td, I, I_lim, I_fil, I_fdq, I_dft, ...
    I_trip_to, I_trip_oc)
figure(2*n+1), set(gcf,'color','w');
plot(t,I,'k', t,I_lim,'r', t,I_fil,'g'), hold on, stairs(td,I_fdq,'b');
    hold off, grid on;
legend('Input','Limited','Filtered','Discretized & Quantized');
xlabel('Time [s]'), ylabel([inam ' [A]']);

figure(2*n+2), set(gcf,'color','w');
subplot(311), stairs(td,I_dft,'r'), grid on;
    title([inam ' DFT --- Discretized & Quantized signal']);
subplot(312), stairs(td,I_trip_to,'r','LineWidth',1), grid on;
    title(['Trip ' inam ' --- Turn off']);
subplot(313), stairs(td,I_trip_oc,'r','LineWidth',1), grid on;
    title(['Trip ' inam ' --- Over current']), xlabel('Time [s]');
```