ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Energia e Automação Elétricas
PEA3411 - Introdução à Automação de Sistemas Elétricos

No. USP	Nome
11805850	Christian Palmerio Braga
11807167	Daniel Oliveira Milagre
11200608	Fernando Campos Chaim
11833671	Thiago Mendes Curto
Professor: Eduardo Lorenzetti Pellini	
Data: 21/07/2023	

Atividade S12: Condicionamento, filtragem, digitalização e cálculo fasorial

Parte 1: Sintetização de um sinal analógico real e a realização do seu condicionamento, proteção e filtragem para que esse possa ser processado digitalmente

Inicialmente, como o grupo não estava muito seguro em gerar um sinal através de algum programa (tal como o Audacity) para utilizá-lo na atividade, foi decidido que a melhor opção seria tentar simular um sinal real através da soma de dois sinais cossenoidais no Matlab, sendo uma delas a 1 Harmônica e a segunda - a terceira harmônica. Este sinal representa uma tensão AD em Volts, tendo a seguinte expressão

$$V(t) = \frac{220\cdot\sqrt{2}}{\sqrt{3}} \cdot cos(2\pi f_0 + 45°) + \frac{220\cdot\sqrt{2}}{\sqrt{3}} \cdot cos(2\pi \cdot 3f_0 + 15°)$$

Para simularmos uma tensão real do sistema elétrico, definiu-se que a frequência fundamental f_0 valeria 60 Hz.

$$V(t) = \frac{220\cdot\sqrt{2}}{\sqrt{3}} \cdot cos(2\pi 60 + 45°) + \frac{220\cdot\sqrt{2}}{\sqrt{3}} \cdot cos(2\pi \cdot 180 + 15°)$$

Como o enunciado determinou que a taxa de amostragem do sinal deveria ser elevada (entre 1.000 ou 10.000 amostras por ciclo da frequência fundamental do sinal), determinou-se que o valor da taxa de amostragem do sinal seria de 120kHz (2000 amostras por ciclo da frequência fundamental do sinal) e o sinal teria como tempo final 0,1 segundo.

```
%ETAPA 1 - CRIAÇÃO DO SINAL ANÁLOGICO E DEFINIÇÃO DO NÚMERO DE AMOSTRAS E FREQUÊNCIA DE AMOSTRAGEM:

f0 = 60; % Frequência fundamental do sinal (60 Hz)
tfinal = 0.1; % Duração do sinal (0.1 s)
m = 2000; % Número de amostras por ciclo da fundamental do sinal
fa = m*f0; % Frequência de amostragem para se criar o sinal analógico
t = 0:1/fa:tfinal; % Vetor de tempos para produzir as amostras desses sinais de tempo continuo
V1 = (220/sqrt(3))*sqrt(2); % Amplitude da fundamental do sinal de valor 220/sqrt(3) VRMS
Fase1 = deg2rad(45); %Fase da fundamental (45°)
V3 = V1/3; % Amplitude da terceira harmônica do sinal
Fase3 = deg2rad(15); %Fase da 3 Harmonica (15°)
fundamental = V1*cos(2*pi*f0*t+Fase1);
terceira = V3*cos(2*pi*3*f0*t+Fase3);
sinal = fundamental + terceira;
```

Figura 1 - Parte do código que define esse sinal

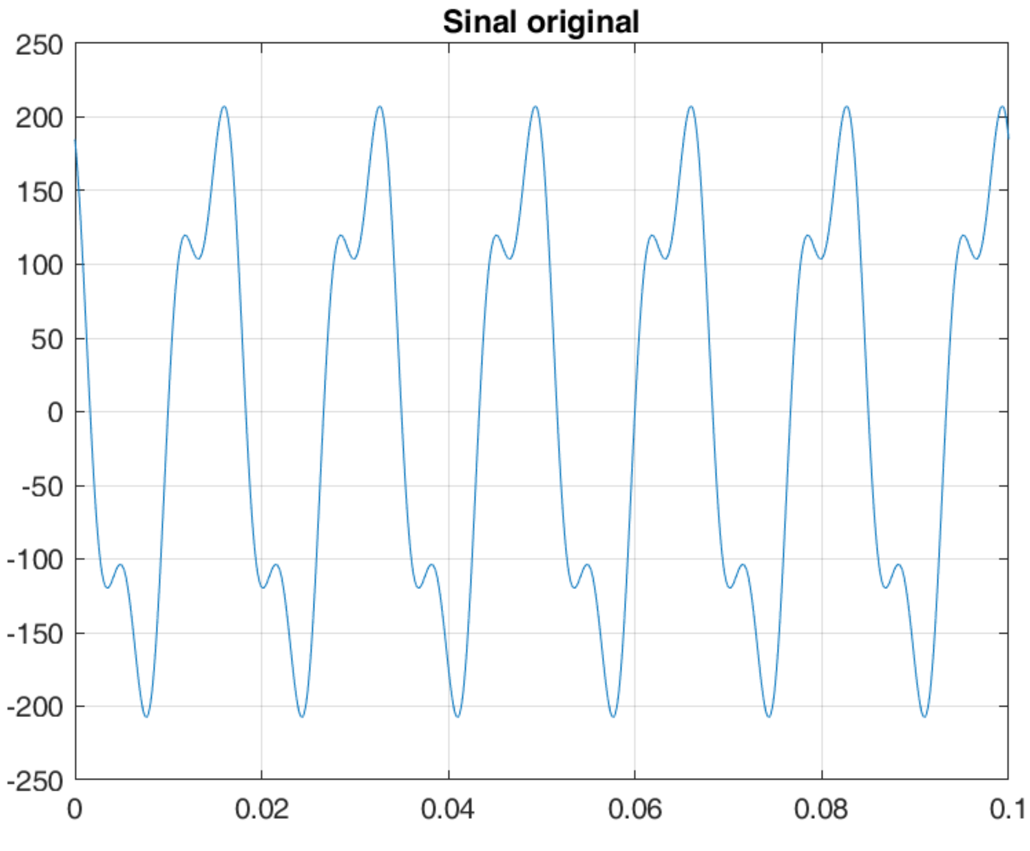

Figura 2 - Gráfico do sinal analógico definido no tempo

Depois de sintetizar o sinal analógico a ser analisado, o grupo definiu que o conversor AD utilizado seria do tipo unipolar com tensões de entrada que variam de 0 a 5 V, com taxa de amostragem $16 \cdot 60\ Hz = 960\ Hz$ e com resolução de 12 bits.

Após a definição das especificações do conversor AD, o grupo definiu como seria seria o condicionamento do sinal, que seria realizado por um TP com relação de espiras 6:220 e por um divisor de tensão resistivo, que reduz a tensão para um quinto do seu valor após passar pelo TP. Além disso, como estamos trabalhando com um conversor AD unipolar que trabalha com tensões entre 0 a 5 Volts, adicionou-se um offset de 2,5 Volts ao sinal.

```
% Simula a entrada desses dois sinais secundarios em entradas analogicas de um IED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1)Condicionamento: diminuir a intensidade dos sinais que vem do TP

sinalIn = sinal * (6/220);     %TP interno de 220:6V
% Ajustando a tensão com um divisor resistivo de 1/5
sinalIn = sinalIn * (1 / 5);

% Adicionando um offset aos canais para eles possuirem tensao analogica UNIPOLAR
% Admitindo eletronica com tensoes entre 0 a 5V, usaremos um offset de 2,5 V.
sinalIn = sinalIn + 2.5;
```

Figura 3 - Parte do código referente ao condicionamento do sinal

Com o condicionamento do sinal feito, definiu-se que a proteção do conversor AD seria realizada através de um circuito grampeador que limita que satura o sinal de entrada no conversor para um valor máximo de 5 Volts e um valor mínimo de 0 Volts. Entretanto, como visto nas aulas, tais circuitos grampeadores são construídos a partir de diodos, que, obviamente, não operam idealmente; logo, para simular uma incerteza ao circuito, permitiu-se que o circuito grampeador admitisse um valor máximo de 5,5 Volts e mínimo de -0,5 Volts.

```
% 2) Protecao dos sinais condicionados, para limitar os seus valores entre as necessidades
%   dos filtros e da eletronica. Supondo uma eletronica para sinais entre 0 e 5 V

% Simula o circuito de grampeamento
sinalInLim = min(sinalIn, 5.5);     % Valor maximo permitido eh 5 + 0.5 V
sinalInLim = max(sinalInLim,-0.5);  % Valor minimo permitido eh 0.0 - 0.5 V
```

Figura 4 - Parte do código referente ao circuito de proteção do conversor AD

Finalmente, definiu-se quais seriam os parâmetros do filtro analógico antialiasing, que realiza a limitação do espectro do sinal para as frequências desejadas.

A partir da leitura do enunciado, foi determinado que o filtro utilizado deveria ser um filtro do tipo “butterworth” e assim, o grupo definiu que os parâmetros do filtro deveriam ser:

- Frequência limite da banda de passagem (f_p): 60 Hz

  Esse valor foi escolhido para que se atenue todas as frequências maiores que a fundamental.
- Frequência limite da banda de rejeição (f_s): 180 Hz

  Esse valor foi escolhido para que se anulem todas as frequências maiores que a 3ª harmônica.
- Atenuação máxima admissível da banda de passagem em dB ($A_{máx}$): 3 dB

  Esse valor foi escolhido com base no conceito de frequência de corte
- Atenuação mínima admissível da banda de rejeição em dB (A_{min}): 40 dB

  Esse valor foi escolhido como um valor de atenuação que faz com que as harmônicas que estão na banda de rejeição sejam praticamente desprezíveis.

```
% 3) Filtragem: simula a filtragem passa baixa para fazer o anti-aliasing
%   Considerando uma frequencia de amostragem (que sera feita
%   posteriormente) com fa = 16*60 = 960 Hz (ou seja, amostragem de 16
%   amostras por ciclo)
%   Considerando um  conversor AD (que sera simulado posteriormente) com 12 bits n = 12

% Especificações do filtro
frequencia_banda_passagem = 60; % Frequência limite da banda de passagem em Hz (deseja-se atenuar todos as harmônicas fora a fundamental)
Amax = 3;            % Atenuação maxima admissivel da banda de passagem em dB (valor escolhido em
                                  % 3 dB por conta da definição de frequência de corte
frequencia_banda_rejeicao = 180; % Frequência limite da banda de rejeição em Hz
Amin = 40;           % Atenuação minima admissivel da banda de rejeicao em dB (40 dB foi escolhido por ser uma atenuação que faz com o sinal deixe de ser relevante)
                                  % Esse valor não precisa ser maior que 1/2^n

% Calculo das frequencias angulares
Wp = 2*pi*frequencia_banda_passagem;   % Frequencia da banda de passagem em rad/s
Ws = 2*pi*frequencia_banda_rejeicao;   % Frequencia da banda de rejeicao em rad/s

% Projeto do filtro butterworth com funcao de transferencia e componentes ideais
% Ordem do filtro segundo as especificacoes
[nFiltro, Wn] = buttord(Wp, Ws, Amax, Amin, 's');

% Projeto dos polinomios da funcao de transferencia do filtro
[num, den] = butter(nFiltro, Wn, 'low', 's');

% Montagem da funcao de transferencia
filtroPB = tf(num,den);

%Simulando o funcionamento dos filtros anti-aliasing passa baixa no sinal
sinalFil = lsim(filtroPB, sinalInLim, t);
```

Figura 5 - Parte do código referente ao filtro antialiasing passa-baixas do tipo butterworth

A partir disso, gerou-se uma figura que contém um gráfico que mostra o sinal após a processo de condicionamento e proteção e um gráfico que mostra o sinal após o processo de filtragem (que ocorre após o processo de condicionamento e proteção).

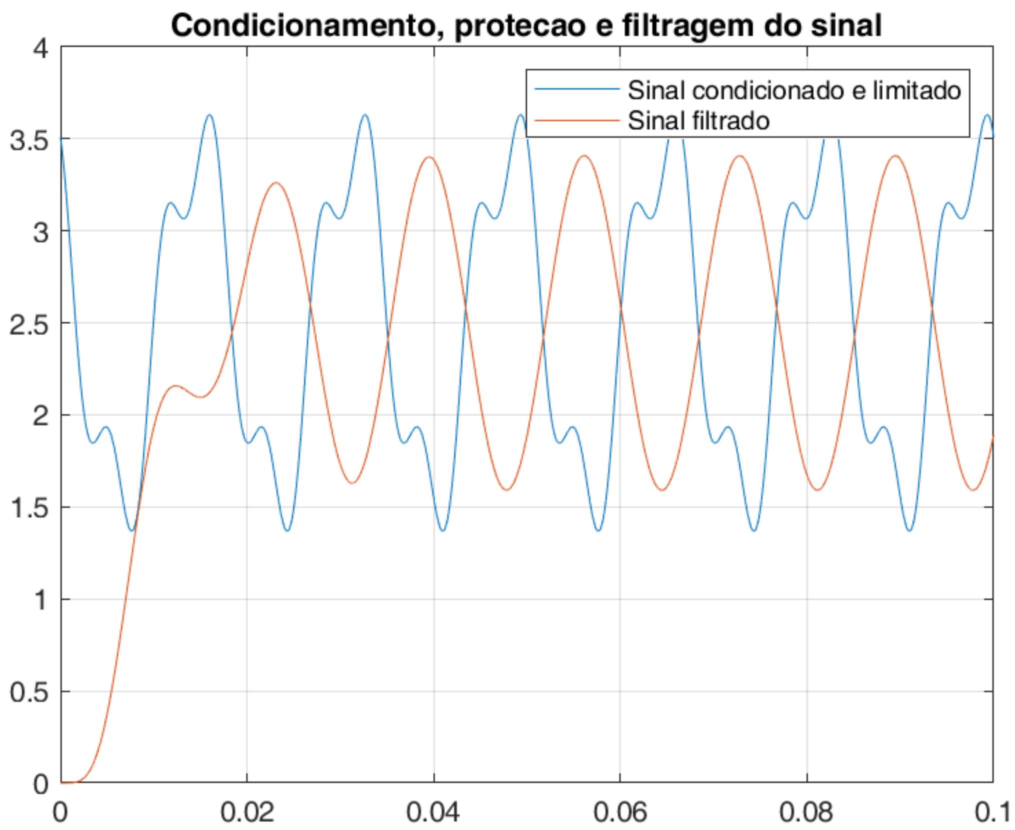

Figura 6 - Gráficos do sinal analógico no tempo após o condicionamento e proteção e após a filtragem

Observando-se a figura 6, observa-se que existe um atraso em relação ao sinal original (que possui a mesma forma de onda do sinal condicionado e grampeado, porém, com uma maior amplitude).

Parte 2: Obtenção dos fasores do sinal elaborado na parte 1

Após obtenção do sinal previamente condicionado, limitado e filtrado na parte 1 da atividade, o grupo iniciou a etapa 2 realizando a “digitalização” do sinal analógico através de um processo que simula a atuação do conversor AD especificado na etapa 1 da atividade.

```
%A) Simular o processo de amostragem (sample & hold)
m = 16; % Deseja-se ter 16 amostras por ciclo da onda fundamental
faFiltro = m*60; %freq de amostragem desejada [Hz]
taFiltro = 1/faFiltro;
% Fator de decimação - nesse caso que estamos emulando o comportamento do tempo
% continuo, vamos pegar amostras igualmente espacadas nos vetores dos sinais,
% a cada 'decimation_factor' numero de amostras
decimation_factor = round(fa/faFiltro);
% Inicialização do vetor de dados amostrados - prepara o processo de amostragem
Tsignal_smp = [];   %vetor para armazenar as amostras do tempo (auxiliar)
signal_smp = [];  %vetor para armazenar as amostras da tensao
% Loop para subamostragem ou decimacao
for i = 1:decimation_factor:length(sinalFil)    %varre o vetor de tempo continuo, saltando de decimation_factor
    signal_smp = [signal_smp, sinalFil(i)];   %pega uma amostra de tensao
    Tsignal_smp = [Tsignal_smp, t(i)];        %tambem amostro para saber o tempo de cada amostra
end

% Resultados
% Observacao do sinal no tempo contino e sinal amostrado por circuito de sample & hold
figure(3)
stairs(Tsignal_smp, signal_smp); hold on; grid;
stairs(t, sinalFil);
legend('Sinal amostrado', 'sinal continuo');
title('Processo de amostragem do S&H')

%B) Simulacao da digitalizacao com conversor AD
n=12;        %12 bits de resolucao - 2^12 simbolos diferentes
q=5/(2^n); % Quanta do conversor em Volts / simbolo

signal_dig = round(signal_smp/q);

figure(4)
stem(Tsignal_smp, signal_dig); hold on; grid;
title('Amostras digitalizadas com AD');
```

Figura 7 - Parte do código referente ao digitalização do sinal

A partir disso, gerou-se uma figura que contém um gráfico que mostra o analógico filtrado e o mesmo sinal após a digitalização.

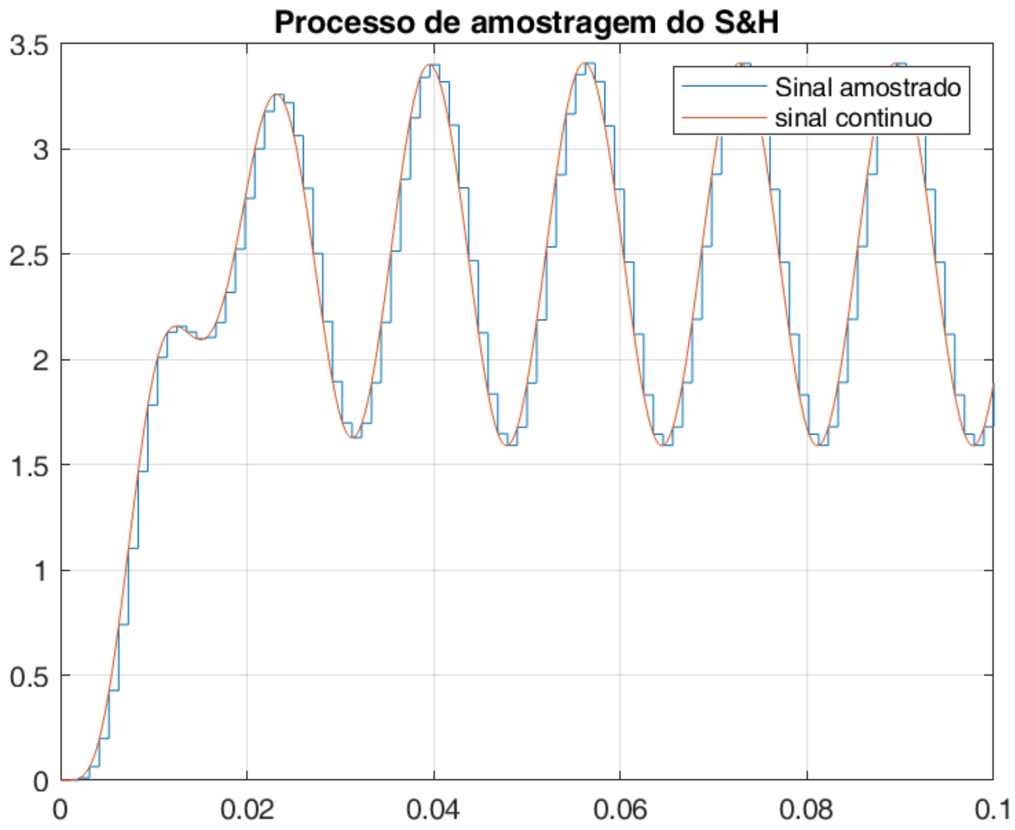

Figura 8 - Gráfico do sinal analógico no tempo após a filtragem e gráfico do sinal digitalizado

Uma observação importante a ser feita é em relação ao eixo x deste gráfico, que está em segundos. Isso obviamente, só faz sentido quando analisamos o sinal analógico, o mais correto seria criar uma figura para o sinal analógico observado no tempo e o sinal digital observado para cada uma das amostras, porém, quando se faz isso uma visualização comparativa entre os dois sinais fica prejudicada. Sendo assim, pode-se considerar que, nesse caso, o "tempo" para o gráfico digital corresponde ao número da amostra que corresponde àquele instante de tempo.

Depois de realizar a digitalização do sinal, iniciou-se a elaboração do algoritmo de processamento do sinal através da Transformada Discreta de Fourier (TDF), onde calculou-se os coeficientes da primeira e terceira harmônica usando a quantidade de amostras por ciclo como base, já que para esse cálculo foi utilizada as seguintes fórmulas:

$$Coef_{cosseno}h \rightarrow \frac{\sqrt{2}}{m} \cdot cos(\frac{2\pi \cdot i \cdot h}{m})$$

$$Coef_{seno}h \rightarrow \frac{\sqrt{2}}{m} \cdot sen(\frac{2\pi \cdot i \cdot h}{m})$$

Nos dois casos, haverá m coeficientes para cada caso, porém i irá variar de 0 até m-1 (em que m corresponde à especificação da quantidade de 16 amostras por ciclo escolhida para o conversor AD, ou seja, m = 16) e por isso, no algoritmo, definimos ele como (i-1), já que este cálculo é feito por um laço "for" que se inicia em 1 e vai até m.

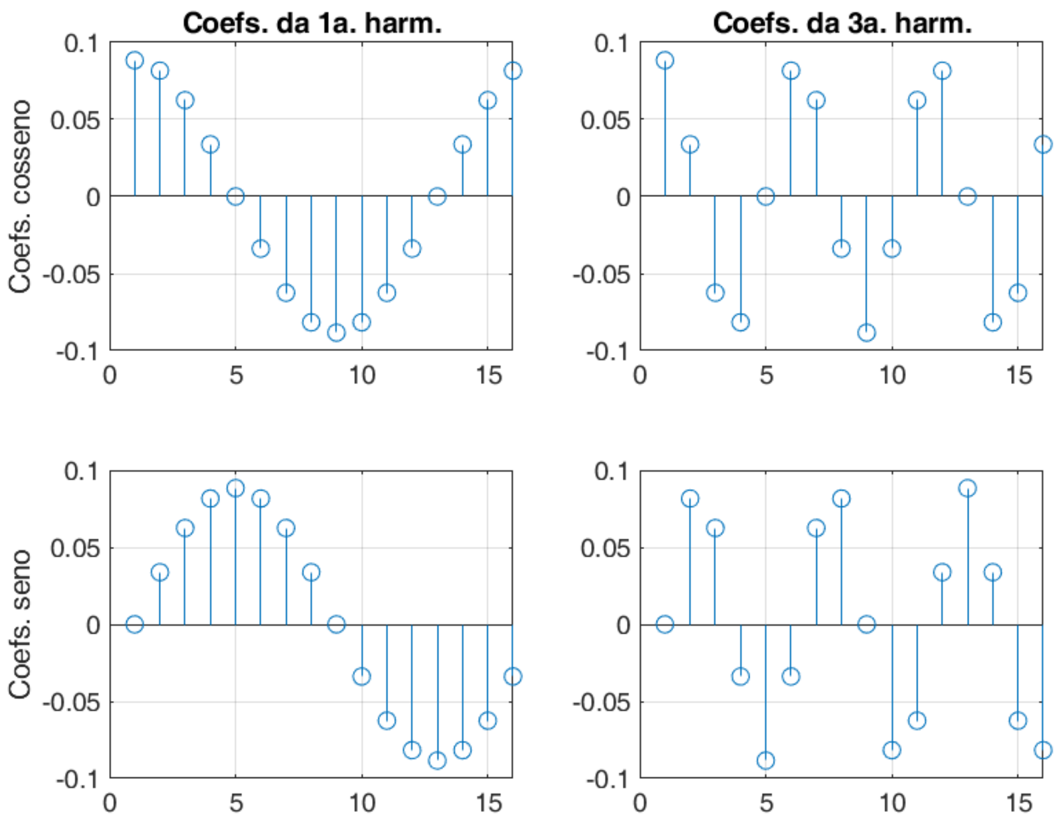

Figura 9 - Coeficientes das harmônicas para 16 amostras em um ciclo

Em seguida, um vetor X de tamanho m é criado contendo apenas zeros em seu conteúdo. É a partir desse vetor que é feita a varredura e as janelas deslizantes. Nesse sentido, há um laço for que contém todos os números das amostras do sinal (ou seja, é um vetor que vai do valor 1 até o número de amostras do sinal digitalizado), e dentro dele há outros dois laços.

O primeiro, realiza a janela deslizante, a cada nova iteração, o elemento z do vetor X passa para a posição z+1 e assim sucessivamente, até o fim do vetor - quando ocorre a finalização do laço. Então, imediatamente, após isso, a posição 1/inicial do vetor X é inicializada com o sinal digital de posição z, permitindo a construção dessa janela deslizante.

Já o segundo passo, realiza a obtenção de y. Para isso, há somas sucessivas realizadas m vezes, sendo que a cada iteração há uma multiplicação entre o coeficiente calculado e o vetor ambos na posição m, o resultado dessa multiplicação é somado com o resultado das iterações anteriores (de início essa soma começa com zero e a cada iteração há a multiplicação e o resultado da multiplicação é somado com os resultados obtidos anteriormente).[1] Dessa forma, obtemos a parte real e imaginária de y - a multiplicação utilizando os coeficientes cossenoidais nos dá a parte real de y e com os coeficientes senoidais temos a parte imaginária.

Ao fim do segundo for, é possível encontrar o módulo de y, já que temos ele na sua forma complexa. Com auxílio do octave, utilizamos a função ABS para encontrar o seu valor em módulo e a função ANGLE para encontrar o seu ângulo (em radianos). E com isso, formamos dois novos vetores, formados pelos módulos e fases de y para cada iteração z.

Esses passos, são repetidos para os coeficientes da terceira harmônica. De modo que ao final desse grande laço for, temos 4 vetores, dois para cada harmônica (módulo e fase):

O trecho do código referente ao procedimento de cálculo dos coeficientes da TDF está apresentado na tabela a seguir

```
% Transformada discreta de Fourier
for i=1:1:m %Coeficientes 1 Harmonica
Coef_b_cos1(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*1)/m);
Coef_b_sin1(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*1)/m);
end
for i=1:1:m %Coeficientes 3 Harmonica
Coef_b_cos3(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*3)/m);
Coef_b_sin3(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*3)/m);
end
for i=1:1:m %Coeficientes 5 Harmonica
Coef_b_cos5(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*5)/m);
Coef_b_sin5(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*5)/m);
end
for i=1:1:m %Coeficientes 7 Harmonica
Coef_b_cos7(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*7)/m);
Coef_b_sin7(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*7)/m);
end
%Código de plotagem dos gráficos retirado do exemplo do professor
figure(5);
subplot(221); stem(Coef_b_cos1); grid; ylabel('Coefs. cosseno');
title('Coefs. da 1a. harm.');
subplot(222); stem(Coef_b_cos3); grid;
title('Coefs. da 3a. harm.');
subplot(223); stem(Coef_b_sin1); grid; ylabel('Coefs. seno');
subplot(224); stem(Coef_b_sin3); grid;
x = zeros(1,m);
for z = 1:1:length(signal_dig)
%Zerando variaveis para evitar problemas na recursão
y_r1 = 0;
y_im1 = 0;
y_r3 = 0;
y_im3 = 0;
for j = 2:1:m
x(m-(j-2)) = x(m-(j-1));
%janela deslizante
```

[1] Neste caso, como a entrada dos dados do sinal digital ocorre no início do vetor (e não no fim - como nos foi ensinado em aula) a multiplicação pode ser feita de maneira direta sem precisar inverter o vetor dos coeficientes.

```
end
x(1) = signal_dig(z);
for k = 1:1:m
%Cálculo de Y para 1 Harmonica
y_r1 = x(k)*Coef_b_cos1(k)+y_r1;
y_im1 = x(k)*Coef_b_sin1(k)+y_im1;
%Cálculo de Y para 3 Harmonica
y_r3 = x(k)*Coef_b_cos3(k)+y_r3;
y_im3 = x(k)*Coef_b_sin3(k)+y_im3;
end
yy_complexo1 = complex(y_r1,y_im1);
yy1(z) = abs(yy_complexo1);
y_fase1(z) = angle(yy_complexo1);
yy_complexo3 = complex(y_r3,y_im3);
yy3(z) = abs(yy_complexo3);
y_fase3(z) = angle(yy_complexo3);
end
```

Tabela 1 - Parte do código referente ao cálculo dos coeficientes da TDF

Por último, se faz necessário apresentar os resultados e esta etapa, mostrando-se os gráficos do módulo do fasor em comparação com o sinal original sem e com offset.

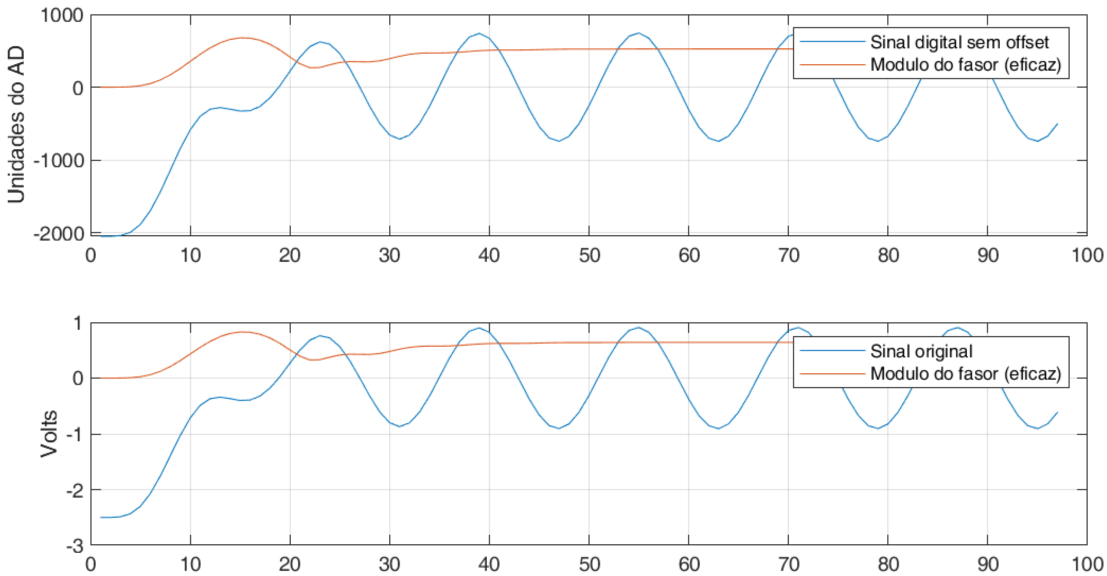

Figura 10 - Comparativo entre o Sinal Original e o Módulo do fasor eficaz

Além disso, há o gráfico comparativo da primeira e terceira harmônica.

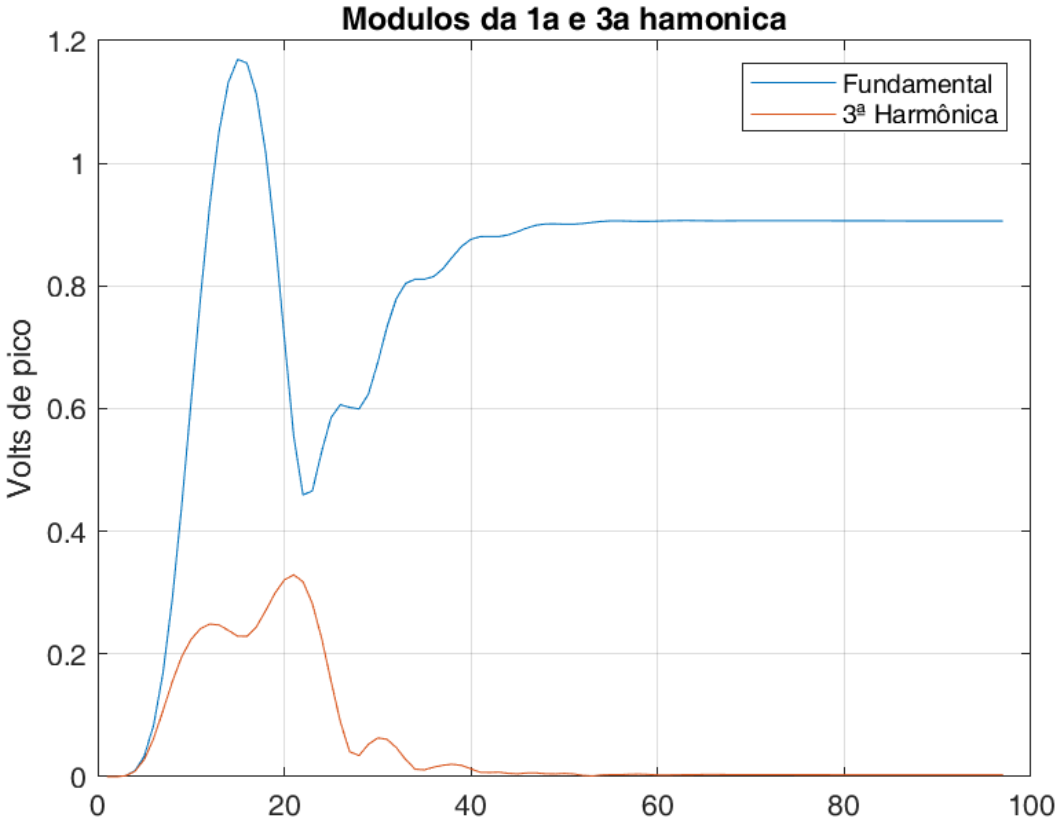

Figura 11 - Comparativo entre o módulo dos fasores da fundamental e da 3ª harmônica

Outrossim, realizou-se um teste para verificar se o módulo do fasor filtrado, está coerente com o módulo do cosseno que utilizamos na criação do sinal analógico - tanto para a primeira quanto para a terceira harmônica. Para isso, deve-se pegar um valor qualquer do módulo dos fasores após a estabilização e verificar se ele está próximo ao valor original - para isso basta selecionar uma amostra maior que a metade do número total de amostras, uma vez que nesse momento, o sistema já se encontra em regime permanente (pode-se observar isso vendo-se a figura 8). Além disso, para realizar essa verificação, basta multiplicar o resultado do módulo do fasor pela quantização (q) e por raiz(2) para passar o resultado de valor eficaz para volts):

Por último, se encontra uma informação que mostra em qual amostra o sinal atinge o regime permanente - o qual quando convertido torna-se o atraso digital.

```
Regime Permanente estabelecido na amostra
     1

Ou seja o tempo de atraso digital é (em segundos):
    0.0010

Módulo do fasor da 1a. harm.  na amostra 60
    0.9058

Módulo do fasor da 3a. harm.   na amostra 60
    0.0029
```

Figura 12 - Display que mostra o tempo de atraso digital e o valor do módulo do fasor obtidos na amostra 60

Sendo assim, conclui-se que os resultados obtidos estão fora do esperado para o sinal original, uma vez que esperava-se que o módulo do fasor da primeira harmônica fosse aproximadamente 179.63 ($\frac{220}{\sqrt{3}} \cdot \sqrt{2}$) e o da terceira harmônica fosse aproximadamente 59.87 ($\frac{220}{3 \cdot \sqrt{3}} \cdot \sqrt{2}$). Além disso, através da análise da figura 8, observa-se que, obviamente, o sinal não atinge o regime permanente logo na amostra 1, isso ocorre por volta da amostra cujo tempo contínuo é 0,04 segundo.

Curiosamente, quando utilizamos uma outra abordagem para se realizar a digitalização do sinal (baseada nas anotações presentes no link 2 da seção de anexos) sem realizar os procedimentos de condicionamento, limitação e filtragem do sinal, os resultados obtidos correspondem aos resultados esperados. A partir disso o grupo estará disponibilizando um arquivo alternativo ("tarefa2Alternativo.m") contendo o script que obtém-se os resultados esperados.

```
% APRESENTAÇÃO DOS RESULTADOS:
for i = 1:length(yy1)-1
    % Verificando se o elemento i é igual ao próximo elemento i+1
    if yy1(i) == yy1(i+1)
        tempoz = i; %amostra em que o regime permanente é estabelecido
        Tempoo = tempoz*taFiltro;
        disp("Regime Permanente estabelecido na amostra"); disp(i);
        disp("Ou seja o tempo de atraso digital é (em segundos):"); disp(Tempoo);
        break;
    end
end

na = 1:1:length(signal_dig);

% Para uma melhor padronização dos resultados, utilizamos o layout dos gráficos que o professor definiu em seu modelo.
figure(6); subplot(211);
plot(na, signal_dig-2048, na, yy1); grid; ylabel('Unidades do AD');
legend('Sinal digital sem offset', 'Modulo do fasor (eficaz)');
subplot(212);
plot(na, (signal_dig-2048)*q, na, yy1*q); grid; ylabel('Volts');
legend('Sinal original', 'Modulo do fasor (eficaz)');

figure(7);
plot(na, yy1*q*sqrt(2), na, yy3*q*sqrt(2)); grid;
title('Modulos da 1a e 3a hamonica');
legend('Fundamental','3ª harmônica');
ylabel('Volts de pico');

disp('Módulo do fasor da 1a. harm.  na amostra 60');  disp(yy1(60)*q*sqrt(2));
disp('Módulo do fasor da 3a. harm.   na amostra 60');  disp(yy3(60)*q*sqrt(2));
```

Figura 13 - Parte do código referente aos resultados finais

OBSERVAÇÃO: O algoritmo em Matlab elaborado para a parte 1 e o script elaborado para a digitalização do sinal elaborado para a etapa 2 desta atividade é totalmente baseado no script elaborado (e disponibilizado aos alunos) pelo professor Pellini durante a aula S11 (link para o arquivo disponível no item 1 da seção de anexos). Já os códigos referentes ao cálculo dos coeficientes da TDF foram baseados nas anotações e scripts elaborados pelo professor Pellini (e disponibilizados aos alunos) que estão disponíveis na página do professor do site “Notion” (item 2 da seção de anexos).

Anexos

1) Script elaborado pelo Professor Pellini na aula S11: https://edisciplinas.usp.br/mod/resource/view.php?id=4799094
2) Anotações e scripts sobre Transformada discreta de Fourier elaborados pelo professor Pellini: https://elpellini.notion.site/Transformada-discreta-de-Fourier-08858628447f45158eb1d05a1be4a00a
3) Script (para Matlab ou Octave) completo elaborado por nós alunos para essa atividade:

```
clc
clear
close all
%ETAPA 1 - CRIAÇÃO DO SINAL ANÁLOGICO E DEFINIÇÃO DO NÚMERO DE
AMOSTRAS E FREQUÊNCIA DE AMOSTRAGEM:
f0 = 60; % Frequência fundamental do sinal (60 Hz)
tfinal = 0.1; % Duração do sinal (0.1 s)
m = 2000; % Número de amostras por ciclo da fundamental do sinal
fa = m*f0; % Frequência de amostragem para se criar o sinal analógico
```

```
t = 0:1/fa:tfinal; % Vetor de tempos para produzir as amostras desses
sinais de tempo continuo
V1 = (220/sqrt(3))*sqrt(2); % Amplitude da fundamental do sinal de
valor 220/sqrt(3) VRMS
Fase1 = deg2rad(45); %Fase da fundamental (45°)
V3 = V1/3; % Amplitude da terceira harmônica do sinal
Fase3 = deg2rad(15); %Fase da 3 Harmonica (15°)
fundamental = V1*cos(2*pi*f0*t+Fase1);
terceira = V3*cos(2*pi*3*f0*t+Fase3);
sinal = fundamental + terceira;
% Simula a entrada desses dois sinais secundarios em entradas
analogicas de um IED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%
% 1)Condicionamento: diminuir a intensidade dos sinais que vem do TP
sinalIn = sinal * (6/220); %TP interno de 220:6V
% Ajustando a tensão com um divisor resistivo de 1/5
sinalIn = sinalIn * (1 / 5);
% Adicionando um offset aos canais para eles possuirem tensao
analogica UNIPOLAR
% Admitindo eletronica com tensoes entre 0 a 5V, usaremos um offset
de 2,5 V.
sinalIn = sinalIn + 2.5;
% 2) Protecao dos sinais condicionados, para limitar os seus valores
entre as necessidades
% dos filtros e da eletronica. Supondo uma eletronica para sinais
entre 0 e 5 V
% Simula o circuito de grampeamento
sinalInLim = min(sinalIn, 5.5); % Valor maximo permitido eh 5 + 0.5 V
sinalInLim = max(sinalInLim,-0.5); % Valor minimo permitido eh 0.0 -
0.5 V
% 3) Filtragem: simula a filtragem passa baixa para fazer o
anti-aliasing
% Considerando uma frequencia de amostragem (que sera feita
% posteriormente) com fa = 16*60 = 960 Hz (ou seja, amostragem de 16
% amostras por ciclo)
% Considerando um conversor AD (que sera simulado posteriormente) com
12 bits n = 12
% Especificações do filtro
frequencia_banda_passagem = 60; % Frequência limite da banda de
passagem em Hz (deseja-se atenuar todos as harmônicas fora a
fundamental)
Amax = 3; % Atenuação maxima admissivel da banda de passagem em dB
(valor escolhido em
% 3 dB por conta da definição de frequência de corte
frequencia_banda_rejeicao = 180; % Frequência limite da banda de
rejeição em Hz
Amin = 40; % Atenuação minima admissivel da banda de rejeicao em dB
(40 dB foi escolhido por ser uma atenuação que faz com o sinal deixe
de ser relevante)
% Esse valor não precisa ser maior que 1/2^n
% Calculo das frequencias angulares
Wp = 2*pi*frequencia_banda_passagem; % Frequencia da banda de
passagem em rad/s
Ws = 2*pi*frequencia_banda_rejeicao; % Frequencia da banda de
rejeicao em rad/s
% Projeto do filtro butterworth com funcao de transferencia e
componentes ideais
% Ordem do filtro segundo as especificacoes
[nFiltro, Wn] = buttord(Wp, Ws, Amax, Amin, 's');
```

```
% Projeto dos polinomios da funcao de transferencia do filtro
[num, den] = butter(nFiltro, Wn, 'low', 's');
% Montagem da funcao de transferencia
filtroPB = tf(num,den);
%Simulando o funcionamento dos filtros anti-aliasing passa baixa no
sinal
sinalFil = lsim(filtroPB, sinalInLim, t);
figure(1);
plot(t, sinal);
grid
title('Sinal original')
figure(2);
plot(t,sinalInLim, t, sinalFil);
grid
legend('Sinal condicionado e limitado', 'Sinal filtrado')
title('Condicionamento, protecao e filtragem do sinal')
% ETAPA 2 - Transformada Discreta de Fourier
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% Etapa de simulacao da amostragem e digitalização
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
%A) Simular o processo de amostragem (sample & hold)
m = 16; % Deseja-se ter 16 amostras por ciclo da onda fundamental
faFiltro = m*60; %freq de amostragem desejada [Hz]
taFiltro = 1/faFiltro;
% Fator de decimação - nesse caso que estamos emulando o
comportamento do tempo
% continuo, vamos pegar amostras igualmente espacadas nos vetores dos
sinais,
% a cada 'decimation_factor' numero de amostras
decimation_factor = round(fa/faFiltro);
% Inicialização do vetor de dados amostrados - prepara o processo de
amostragem
Tsignal_smp = []; %vetor para armazenar as amostras do tempo
(auxiliar)
signal_smp = []; %vetor para armazenar as amostras da tensao
% Loop para subamostragem ou decimacao
for i = 1:decimation_factor:length(sinalFil) %varre o vetor de tempo
continuo, saltando de decimation_factor
signal_smp = [signal_smp, sinalFil(i)]; %pega uma amostra de tensao
Tsignal_smp = [Tsignal_smp, t(i)]; %tambem amostro para saber o tempo
de cada amostra
end
% Resultados
% Observacao do sinal no tempo contino e sinal amostrado por circuito
de sample & hold
figure(3)
stairs(Tsignal_smp, signal_smp); hold on; grid;
stairs(t, sinalFil);
legend('Sinal amostrado', 'sinal continuo');
title('Processo de amostragem do S&H')
%B) Simulacao da digitalizacao com conversor AD
n=12; %12 bits de resolucao - 2^12 simbolos diferentes
q=5/(2^n); % Quanta do conversor em Volts / simbolo
signal_dig = round(signal_smp/q);
figure(4)
stem(Tsignal_smp, signal_dig); hold on; grid;
title('Amostras digitalizadas com AD');
% Transformada discreta de Fourier
```

```
for i=1:1:m %Coeficientes 1 Harmonica
Coef_b_cos1(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*1)/m);
Coef_b_sin1(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*1)/m);
end
for i=1:1:m %Coeficientes 3 Harmonica
Coef_b_cos3(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*3)/m);
Coef_b_sin3(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*3)/m);
end
for i=1:1:m %Coeficientes 5 Harmonica
Coef_b_cos5(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*5)/m);
Coef_b_sin5(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*5)/m);
end
for i=1:1:m %Coeficientes 7 Harmonica
Coef_b_cos7(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*7)/m);
Coef_b_sin7(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*7)/m);
end
%Código de plotagem dos gráficos retirado do exemplo do professor
figure(5);
subplot(221); stem(Coef_b_cos1); grid; ylabel('Coefs. cosseno');
title('Coefs. da 1a. harm.');
subplot(222); stem(Coef_b_cos3); grid;
title('Coefs. da 3a. harm.');
subplot(223); stem(Coef_b_sin1); grid; ylabel('Coefs. seno');
subplot(224); stem(Coef_b_sin3); grid;
x = zeros(1,m);
for z = 1:1:length(signal_dig)
%Zerando variaveis para evitar problemas na recursão
y_r1 = 0;
y_im1 = 0;
y_r3 = 0;
y_im3 = 0;
for j = 2:1:m
x(m-(j-2)) = x(m-(j-1));
%janela deslizante
end
x(1) = signal_dig(z);
for k = 1:1:m
%Cálculo de Y para 1 Harmonica
y_r1 = x(k)*Coef_b_cos1(k)+y_r1;
y_im1 = x(k)*Coef_b_sin1(k)+y_im1;
%Cálculo de Y para 3 Harmonica
y_r3 = x(k)*Coef_b_cos3(k)+y_r3;
y_im3 = x(k)*Coef_b_sin3(k)+y_im3;
end
yy_complexo1 = complex(y_r1,y_im1);
yy1(z) = abs(yy_complexo1);
y_fase1(z) = angle(yy_complexo1);
yy_complexo3 = complex(y_r3,y_im3);
yy3(z) = abs(yy_complexo3);
y_fase3(z) = angle(yy_complexo3);
end
% APRESENTAÇÃO DOS RESULTADOS:
for i = 1:length(yy1)-1
% Verificando se o elemento i é igual ao próximo elemento i+1
if yy1(i) == yy1(i+1)
tempoz = i; %amostra em que o regime permanente é estabelecido
Tempoo = tempoz*taFiltro;
disp("Regime Permanente estabelecido na amostra"); disp(i);
disp("Ou seja o tempo de atraso digital é (em segundos):");
disp(Tempoo);
```

```
break;
end
end
na = 1:1:length(signal_dig);
% Para uma melhor padronização dos resultados, utilizamos o layout
dos gráficos que o professor definiu em seu modelo.
figure(6); subplot(211);
plot(na, signal_dig-2048, na, yy1); grid; ylabel('Unidades do AD');
legend('Sinal digital sem offset', 'Modulo do fasor (eficaz)');
subplot(212);
plot(na, (signal_dig-2048)*q, na, yy1*q); grid; ylabel('Volts');
legend('Sinal original', 'Modulo do fasor (eficaz)');
figure(7);
plot(na, yy1*q*sqrt(2), na, yy3*q*sqrt(2)); grid;
title('Modulos da 1a e 3a hamonica');
legend('Fundamental','3ª harmônica');
ylabel('Volts de pico');
disp('Módulo do fasor da 1a. harm. na amostra 60');
disp(yy1(60)*q*sqrt(2));
disp('Módulo do fasor da 3a. harm. na amostra 60');
disp(yy3(60)*q*sqrt(2));
```

4) Script (para Matlab ou Octave) alternativo para a parte 2 da atividade:

```
clc
clear all
close all
%ETAPA 1 - CRIAÇÃO DO SINAL ANÁLOGICO E DEFINIÇÃO DO NÚMERO DE
AMOSTRAS E FREQUÊNCIA DE AMOSTRAGEM
na = 1 : 1 : 100; %100 AMOSTRAS em nosso sinal analógico
m = 16; %Quantidade de amostras por ciclo definida pelo operador
fa = m * 60; %Frequência de amostragem
ta = 1/fa; %Período de amostragem
A1 = (220/sqrt(3))*sqrt(2); %Amplitude da 1 Harmonica [V]
Fase1 = deg2rad(45); %Fase da 1 Harmonica
A3 = A1/3; %Amplitude da 3 Harmonica [V]
Fase3 = deg2rad(15); %Fase da 3 Harmonica
t = na*ta; %Vetor de tempo para cada Amostra
SinalAn = A1 * cos (2*pi*60*t + Fase1) + A3 * cos (2*pi*3*60*t +
Fase3); %vetor formado pela sinal senoidal analógico
SinalAn = SinalAn + 2.5; %Soma Metade do fundo de escala para
conversor unipolar (0 a 5 V)
q = 5/(2^12); %Bits por volts
SinalDig = round(SinalAn/q); %converte a tensao na representacao do
AD unipolar
%Plotagem do Sinal Analógico e Sinal Digital, utilizando na como o
eixo X
figure(1); subplot(211); plot(na, SinalAn); grid; ylabel('Tensao
AD');
subplot(212); stairs(na, SinalDig); grid; ylabel('Amostras AD');
%ETAPA 2 - Transformada Discreta de Fourier
for i=1:1:m %Coeficientes 1 Harmonica
Coef_b_cos1(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*1)/m);
Coef_b_sin1(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*1)/m);
end
for i=1:1:m %Coeficientes 3 Harmonica
Coef_b_cos3(i) = (sqrt(2)/m)*cos((2*pi*(i-1)*3)/m);
```

```
Coef_b_sin3(i) = (sqrt(2)/m)*sin((2*pi*(i-1)*3)/m);
end
%Código de plotagem dos gráficos retirado do exemplo do professor
figure(2);
subplot(221); stem(Coef_b_cos1); grid; ylabel('Coefs. cosseno');
title('Coefs. da 1a. harm.');
subplot(222); stem(Coef_b_cos3); grid;
title('Coefs. da 3a. harm.');
subplot(223); stem(Coef_b_sin1); grid; ylabel('Coefs. seno');
subplot(224); stem(Coef_b_sin3); grid;
x = zeros(1,m);
for z = 1:1:100
%Zerando variaveis para evitar problemas na recursão
y_r1 = 0;
y_im1 = 0;
y_r3 = 0;
y_im3 = 0;
for j = 2:1:m
x(m-(j-2)) = x(m-(j-1));
%janela deslizante
end
x(1) = SinalDig(z);
for k = 1:1:m
%Cálculo de Y para 1 Harmonica
y_r1 = x(k)*Coef_b_cos1(k)+y_r1;
y_im1 = x(k)*Coef_b_sin1(k)+y_im1;
%Cálculo de Y para 3 Harmonica
y_r3 = x(k)*Coef_b_cos3(k)+y_r3;
y_im3 = x(k)*Coef_b_sin3(k)+y_im3;
end
yy_complexo1 = complex(y_r1,y_im1);
yy1(z) = abs(yy_complexo1);
y_fase1(z) = angle(yy_complexo1);
yy_complexo3 = complex(y_r3,y_im3);
yy3(z) = abs(yy_complexo3);
y_fase3(z) = angle(yy_complexo3);
end
% ETAPA 3 - APRESENTAÇÃO DOS RESULTADOS
for i = 1:length(yy1)-1
% Verificando se o elemento i é igual ao próximo elemento i+1
if yy1(i) == yy1(i+1)
tempoz = i; %amostra em que o regime permanente é estabelecido
Tempoo = tempoz*ta;
disp("Regime Permanente estabelecido na amostra"); disp(i);
disp("Ou seja o tempo de atraso digital é (em segundos):");
disp(Tempoo);
break;
end
end
%Para uma melhor padronização dos resultados, utilizamos o layout dos
gráficos que o professor definiu em seu modelo.
figure(3); subplot(211);
plot(na, SinalDig-2048, na, yy1); grid; ylabel('Unidades do AD');
legend('Sinal digital sem offset', 'Modulo do fasor (eficaz)');
subplot(212);
plot(na, (SinalDig-2048)*q, na, yy1*q); grid; ylabel('Volts');
legend('Sinal original', 'Modulo do fasor (eficaz)');
figure(4);
plot(na, yy1*q*sqrt(2), na, yy3*q*sqrt(2)); grid;
title('Modulos da 1a e 3a hamonica');
```

```
ylabel('Volts de pico');
disp('Módulo do fasor da 1a. harm. na amostra 60');
disp(yy1(60)*q*sqrt(2));
disp('Módulo do fasor da 3a. harm. na amostra 60');
disp(yy3(60)*q*sqrt(2));
```