Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Eletrônica e Sistemas

Graduação em Engenharia Eletrônica

Uma Implementação Computacional de Estimativa Harmônica Baseada na Série de Fourier Quantizada com Interface Gráfica

Diego Felipe Gomes Coelho

Trabalho de Graduação

Recife
7 de Dezembro de 2012

Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Eletrônica e Sistemas

Diego Felipe Gomes Coelho

Uma Implementação Computacional de Estimativa Harmônica Baseada na Série de Fourier Quantizada com Interface Gráfica

Trabalho apresentado ao Programa de Graduação em Engenharia Eletrônica do Departamento de Eletrônica e Sistemas da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Engenheiro Eletrônico.

Orientador: *Prof. Dr. Renato J. Cintra*
Co-orientador: *Prof. Dr. Hélio Magalhães de Oliveira*

Recife
7 de Dezembro de 2012

A meus pais, Décio Coelho de Araújo e
Geny G. da Silva Coelho, e meus irmãos,
Glauber D. G. Coelho e Alana L. G. Coelho.

Agradecimentos

Agradeço aos meus pais, em especial minha mãe, por em seu modo muito especial me forçar a estudar até mesmo nas férias enquanto meus primos alegremente brincavam. Agradeço também a todos os infortúnios e mazelas da vida, que apesar de tudo, me fortaleceram o suficiente para poder chegar a momentos como este, a de conclusão do curso de Engenharia Eletrônica na UFPE, que para mim representa uma conquista sem tamanho.

Resumo

Este trabalho explora a proposta feita em [H. M. de Oliveira D. F. Souza, R. J. de Sobral Cintra. Uma Ferramenta para Análise de Sons Musicais: A Série Quantizada de Fourier. *XXII Simpósio Brasileiro de Telecomunicações*, sep 2005] do uso da base quantizada de sinais $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=1}^{\infty}$ no contextos de detecção harmônica. Este trabalho aborda aspectos teóricos do desenvolvimento da série de Fourier clássica e da proposta série de Fourier Quantizada. Um aplicativo escrito para a plataforma Matlab® que implementa a estimação de harmônicos baseada na série de Fourier Quantizada foi desenvolvido.

Palavras-chave: Série de Fourier Clássica, Série de Fourier Quantizada, Implementação Computacional, Interface Gráfica, Estimativa Harmônica, Análise de Fourier, Álgebra Linear, Processamento de Sinais.

Abstract

This work analyzes the Fourier quantized base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=1}^{\infty}$ proposed in [H. M. de Oliveira D. F. Souza, R. J. de Sobral Cintra. Uma Ferramenta para Análise de Sons Musicais: A Série Quantizada de Fourier. *XXII Simpósio Brasileiro de Telecomunicações*, sep 2005] applied in the context of harmonic detection. A Matlab® language written software that implements a harmonic estimation based on quantized Fourier series is presented. Results are compared with those furnished by the usual Fourier series analysis.

Keywords: Classical Fourier Series, Quantized Fourier Series, Computational Implementation, Graphical Interface, Harmonic Estimation, Fourier Analysis, Linear Algebra, Signal Processing.

Sumário

Lista de Figuras

Lista de Tabelas

CAPÍTULO 1

Introdução

1.1 Contextualização

Por décadas, o processamento de sinais tem desempenhado um papel central em áreas como telecomunicações, engenharia médica e clínica, acústica, sísmica, exploração e prospecção petrolífera, instrumentação, robótica, eletrônica de consumo e muitas outras áreas. Hoje, elaborados algoritmos e sistemas de processamento de sinais tem sido amplamente usados, desde em sistemas de entretenimento, como a televisão e sistemas de áudio, até sistemas críticos para controle de sistemas aviônicos e de manutenção da vida de pacientes em estado crítico de saúde. Historicamente, a área de processamento de sinais tem sido beneficiada pela íntima relação entre teoria e aplicações que implementam sistemas baseados no desenvolvimento teórico. Até meados de 1960, toda a tecnologia no campo de processamento de sinais foi preponderantemente ligada a sistemas de tempo contínuo, contudo, a evolução dos computadores digitais e microprocessadores em conjunto com avanços teóricos da época levaram à ascensão dos sistemas de processamento de sinais digitais.

A fundamental diferença entre processamento de sinais de tempo contínuo e discreto é que este trata do processamento de sinais cuja a componente do tempo assume apenas valores discretos, e não mais contínuos. Não obstante, o processamento digital de sinais é um caso especial do processamento de sinais de tempo discreto. Ao passo que o processamento de sinais de tempo discreto trata de sinais cuja a única componente discreta é o tempo, o processamento digital de sinais trata de sinais cuja ambas componentes, tempo e amplitude, assumem valores discretos.

Embora haja diversas situações em que os sinais a serem processados são naturalmente de tempo discreto, este não é caso geral. Usualmente, sinais de tempo discreto representam sinais de tempo contínuo que devem ser processados de modo a se obter uma outra sequência ou extrair alguma informação. Nestas situações, os sinais de tempo contínuo são transformados em sinais de tempo discreto por meio de um processo chamado de amostragem [28]. Durante tal processo, são geradas amostras do sinal de tempo contínuo. Obedecendo a taxa de Nyquist [28], o sinal de tempo discreto obtido pode bem representar o sinal de tempo contínuo original; e daquele este pode ser reconstruído. Não raro, sistemas que implementam tais técnicas são projetados de modo que operem em tempo real, ou seja, que o sinal de tempo discreto obtido seja processado a mesma taxa que o sinal de tempo contínuo é amostrado.

Maior parte das implementações de sistemas de processamento de sinais envolve processar um sinal de modo a se obter outro. Contudo, em uma outra classe de problemas, nem sempre há o interesse em obter uma outra sequência numérica, mas, sim, extrair informações da sequência de entrada. Este tipo de problema ocorre frequentemente em sistemas de reconhecimento de

voz e imagem [18, 29, 30]. Usualmente, tais sistemas são compostos de uma fase de pré-processamento seguidas de um bloco de reconhecimento de padrão.

Os problemas de processamento de sinais não estão limitados a sinais de uma única dimensão como dissertado até o momento. De fato, problemas relacionados com imagem constantemente demandam sistemas capazes de manipular sinais multidimensionais. Para ilustrar, pode-se citar codificação de vídeo [7], imagens médicas [18], imagens de satélites [22] e processamento de sinais sísmicos [35]. Contudo, problemas que tratam de sinais multidimensionais podem ser analisados e decompostos em problemas de uma dimensão, exceto por algumas adaptações.

Uma ferramenta matemática fundamental em processamento de sinais é a série de Fourier [16, 19, 28]. Esta técnica faz parte de um conjunto maior de técnicas coletivamente denominadas de Análise de Fourier.

A série de Fourier surgiu com o estudo de ondas de calor feito Joseph Fourier (1768-1830) publicado inicialmente em *Mémoire sur la propagation de la chaleur dans les corps solides* (Tratado da propagação de calor em corpos rígidos) em 1807 e *Théorie analytique de la chaleur* (Teoria analítica do calor) em 1822 [17]. Embora a série de Fourier tenha sido motivada originalmente pelo estudo da resolução da equação do calor, ficou evidente posteriormente que as mesmas técnicas poderiam ser aplicadas a uma grande variedade de problemas matemáticos e físicos, especialmente aqueles envolvendo equações diferenciais ordinárias com coeficientes constantes, para a qual as soluções envolvam ondas sinusoidais. Uma digressão sobre a aplicação da série de Fourier em equações diferenciais pode ser encontrado no Apêndice D.

1.2 Objetivos e Contribuições

Este trabalho se desenvolve em contextos em que a série de Fourier é aplicada para detecção de sinais. Tais situações podem surgir em cenários de aplicações tais como reconhecimento de voz, reconhecimento de padrões de atividade sísmica e reconhecimento e processamento de imagem e vídeo. Em tais situações não se faz necessário uma computação exata dos coeficientes da série de Fourier, haja visto que o objetivo se limita a detecção. Com a eliminação do requisito da computação exata dos coeficientes, pode-se, em detrimento da precisão, aumentar a velocidade e massa de dados processados. Com o objetivo de tirar proveito desta vantagem, este trabalho tem como finalidade implementar um analisador de espectro implementado em *software* capaz de realizar detecção baseado na quantização do núcleo da expansão em série de Fourier, como proposto em [11].

De modo semelhante ao que foi feito em [11], neste trabalho é adotada a base quantizada de Fourier $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ para estimação harmônica. Contudo, embora em [11] se tenha considerando por simplificação a base proposta como sendo ortogonal, neste trabalho não é feita tal consideração. De fato, a base proposta é não-ortogonal, como demonstrado na Seção 3.2.1. Embora desconsiderar a não-ortogonalidade da base proposta possa ser útil em alguns cenários em que há um baixo requisito de precisão, neste trabalho não o fazemos. Todos os cálculos em busca da aproximação dos coeficientes da série de Fourier clássica em função da base proposta são realizados considerando a sua não-ortogonalidade.

Embora considerar a base proposta como ortogonal resulte em uma simplificação dos cál-

culos envolvidos, como feito em [11], tal consideração resulta em uma aproximação que pode ser considerada de baixa precisão. Proceder com os cálculos com a base proposta considerando a sua não-ortogonalidade pode levar a um aumento da complexidade das operações envolvidas para se estimar os coeficientes da série de Fourier clássica. Não obstante, acompanhado do aumento da complexidade das operações envolvidas, é obtida uma aproximação mais precisa e próxima dos coeficientes da série de Fourier clássica. Obviamente, o quão precisa deve ser a estimação harmônica deve ser ditada por cada aplicação particular.

O aplicativo resultante será escrito em linguagem de Matlab®. O *software* poderá ser usado nas aplicações acimas discutidas com pequenas modificações, a depender da aplicação escolhida.

Embora diversas outras ferramentas tenham sido desenvolvidas com o objetivo de manipular sinais não-estacionários, como por exemplo a transformada *Wavelet* [12], neste trabalho são tratados apenas de sinais estacionários. Tal suposição permite aplicar a série de Fourier.

1.3 Estrutura do Trabalho

O trabalho se desdobra como segue.

Capítulo 2 Formaliza o conceito da série de Fourier e trata de questões como a representação de um sinal, ortonormalidade, fenômeno de Gibbs e convergência da série de Fourier;

Capítulo 3 Apresenta e formaliza a representação de um sinal por meio da série quantizada de Fourier. Neste Capítulo, são tratados aspectos como a não-ortogonalidade da série quantizada de Fourier e a relação entre os coeficientes de Fourier e os coeficientes da série quantizada de Fourier. Ainda é apresentada uma forma de se obter uma aproximação para os coeficientes de Fourier a partir dos coeficientes da série quantizada de Fourier;

Capítulo 4 Este capítulo trás uma explanação a respeito de algumas simulações da obtenção dos coeficientes da série quantizada de Fourier e dos coeficientes da série de Fourier a partir da série quantizada para alguns sinais em especial;

Capítulo 5 Investiga algumas formas de otimização computacional que podem ser aplicadas a implementações em *hardware* e *software* com o objetivo de se minimizar o número de operações necessárias para a obtenção de uma estimativa dos coeficientes da série de Fourier a partir da série quantizada de Fourier;

Capítulo 7 Aborda aspectos de projeto de interfaces gráficas com o uso da linguagem Matlab® e descreve o aplicativo confeccionado durante este trabalho de graduação;

Capítulo 8 contém as considerações finais a cerca do aplicativo construído com este trabalho de graduação, além de endereçar possíveis trabalhos futuros para enriquecimento do *software* criado;

Seção A Contém os principais códigos usados durante a fase de simulação;

Seção B Contém os principais códigos escritos na implementação do *software*;

Seção C Contém imagens da interface gráfica do *software* criado em algumas situações comuns durante seu uso;

Seção D Aborda a aplicação da série de Fourier a resolução de problemas envolvendo equações diferenciais;

Seção D Apresenta uma digressão sobre o estudo de séries infinitas usada para fundamentar o estudo da convergência da série de Fourier.

CAPÍTULO 2

As Séries de Fourier

A série de Fourier é um caso especial de séries infinitas: as séries trigonométricas. Suas aplicações já foram discutidas no capítulo anterior. Neste capítulo, estamos interessados em descrever a série de Fourier e formalizá-la.

Seja um sinal descrito por uma função real $f(t)$, contínua, definida na reta real e que seja periódica com período T, ou seja, $f(t+T) = f(t)$ para todo $t \in \mathbb{R}$. Admita que $f(t)$ seja absolutamente integrável em todo intervalo fechado na reta real [2].

Definição 1. *A série de Fourier de $f(t)$ é dada por:*

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)), \tag{2.1}$$

em que $\omega_0 \triangleq \frac{2\pi}{T}$ é chamada de frequência fundamental e a_0 e $\{a_n, b_n\}_{n=1}^{\infty}$ são coeficiente reais dados por:

$$a_0 = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \mathrm{d}t, \tag{2.2}$$

$$a_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cos(n\omega_0 t) \mathrm{d}t, \tag{2.3}$$

e

$$b_n = \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \sin(n\omega_0 t) \mathrm{d}t, \tag{2.4}$$

para $n \in \mathbb{N}$, em que $\mathbb{N}$ representa o conjunto dos números naturais, excetuando-se o 0.

O conjunto $\{1, \cos(n\omega_0 t), \sin(n\omega_0 t)\}_{n=1}^{\infty}$ forma uma base para a representação de funções periódicas com período fundamental de $2\pi/\omega_0$ [13]. Com esta nova base de funções periódicas, pode-se definir o produto interno entre duas funções quaisquer da seguinte forma.

Definição 2. *Seja $f(t)$ e $g(t)$ duas funções reais, definidas na reta real e periódicas, ambas com período T e ambas integráveis à Riemann. Define-se o produto interno entre $f(t)$ e $g(t)$:*

$$\langle f(t), g(t) \rangle \triangleq \frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) \cdot g(t) \mathrm{d}t, \tag{2.5}$$

Este produto interno induz a seguinte norma para funções:

$$\| f(t) \| = \sqrt{\langle f(t), f(t) \rangle} = \sqrt{\frac{2}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f^2(t) \mathrm{d}t}, \tag{2.6}$$

Sejam $g(t)$ e $h(t)$ funções com as característica exigidas na Definição 1. As propriedades comuns a produtos internos de vetores de um espaço vetorial são facilmente demonstradas usando as propriedades das integrais [32], a saber:

1. Comutatividade
$$\langle f(t), g(t)\rangle = \langle g(t), f(t)\rangle;$$

2. Distributividade com relação a soma
$$\langle f(t), g(t)+h(t)\rangle = \langle f(t), g(t)\rangle + \langle f(t), h(t)\rangle;$$

3. Bilinearidade
$$\langle \alpha \cdot f(t), g(t)\rangle = \alpha\langle f(t), g(t)\rangle = \langle f(t), \alpha \cdot g(t)\rangle,$$
em que $\alpha \in \mathbb{R}$;

4. Não-negatividade
$$\langle f(t), f(t)\rangle = \|f(t)\| \geq 0$$

5. Nulidade da norma
$$\|f(t)\| = 0 \Leftrightarrow f(t) = 0.$$

O produto interno como dado na Definição 2 é útil para a compreensão da série de Fourier e suas propriedades. A seção a seguir trata da ortogonalidade, uma importante propriedade presente na base da série de Fourier, cujo estudo depende da noção de produto interno.

2.1 Ortogonalidade da Série de Fourier

A série de Fourier goza de propriedades importantes e úteis na análise de sinais. Uma dessas propriedades é a ortogonalidade. No caso em particular da série de Fourier, significa dizer que a base de Fourier $\{1, \cos(n\omega_0 t), \sin(n\omega_0 t)\}_{n=1}^{\infty}$ é uma base ortogonal com respeito ao produto interno na Definição 2. O teorema a seguir detalha essa característica.

Teorema 1. *A base de Fourier,* $\{1, \cos(n\omega_0 t), \sin(n\omega_0 t)\}_{n=1}^{\infty}$*, é uma base ortogonal [16].*

Demonstração. Primeiro, analisemos os casos triviais $\langle 1, \cos(n\omega_0 t)\rangle$ e $\langle 1, \sin(n\omega_0 t)\rangle$, em que $n \in \mathbb{N}$.

$$\langle 1, \cos(n\omega_0 t)\rangle = \frac{\omega_0}{\pi}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \cos(n\omega_0 t)\mathrm{d}t \tag{2.7}$$
$$= \frac{\omega_0}{\pi}\frac{\sin(n\omega_0 t)}{n\omega_0}\Bigg|_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} = 0$$

$$\langle 1, \cos(n\omega_0 t)\rangle = \frac{\omega_0}{\pi}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \sin(n\omega_0 t)\mathrm{d}t \tag{2.8}$$
$$= -\frac{\omega_0}{\pi}\frac{\cos(n\omega_0 t)}{n\omega_0}\Bigg|_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} = 0$$

Assim, as funções $\sin(n\omega_0 t)$ e $\cos(n\omega_0 t)$ são ortogonais em relação a função constante 1. Considerando n, $m \in \mathbb{N}$, tem-se que:

$$\langle \cos(n\omega_0 t), \cos(m\omega_0 t)\rangle = \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \cos((n+m)\omega_0 t)\mathrm{d}t + \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \cos((n-m)\omega_0 t)\mathrm{d}t \tag{2.9}$$
$$= 0 + \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \cos((n-m)\omega_0 t)\mathrm{d}t.$$

A integral do termo correspondente a $\cos((n+m)\omega_0 t)$ sempre corresponderá a zero para quaisquer naturais n e m [13]. Para o termo $\cos((n-m)\omega_0 t)$, tem-se que se $n \neq m$, então, pelo mesmo motivo mencionado, a integral deste termo será zero, e apenas se $n = m$, então $\cos((n-m)\omega_0 t) = \cos(0\omega_0 t) = 1$. Assim, vem que :

$$\frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} 1\mathrm{d}t = \frac{\omega_0}{\pi}t\Bigg|_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} = 1. \tag{2.10}$$

Para os elementos da base na forma $\sin(m\omega_0 t)$, $m \in \mathbb{N}$, tem-se que:

$$\langle \sin(n\omega_0 t), \sin(m\omega_0 t)\rangle = \frac{\omega_0}{\pi}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \sin(n\omega_0 t)\cdot\sin(m\omega_0 t)\mathrm{d}t \tag{2.11}$$
$$= \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \cos((n-m)\omega_0 t)\mathrm{d}t - \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \cos((n+m)\omega_0 t)\mathrm{d}t$$
$$= \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \cos((n-m)\omega_0 t)\mathrm{d}t - 0,$$

em que mais uma vez o termo $\cos((n+m)\omega_0 t)$, por sempre ser periódico, resulta em uma integral de valor nulo. O termo $\cos((n-m)\omega_0 t)$, para $n \neq m$ resulta em uma integral também nula, e para $n = m$ resulta em $\cos((n-m)\omega_0 t) = \cos(0\omega_0 t) = 1$, o que leva a

$$\frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} 1\mathrm{d}t = \frac{\omega_0}{\pi}t\Bigg|_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} = 1. \tag{2.12}$$

Resta analisar os produtos internos da forma $\langle \cos(n\omega_0 t), \sin(m\omega_0 t)\rangle$.

$$\begin{aligned}\langle \sin(n\omega_0 t), \cos(m\omega_0 t)\rangle &= \frac{\omega_0}{\pi}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \sin(n\omega_0 t)\cdot\cos(m\omega_0 t)\mathrm{d}t \\ &= \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \sin((n+m)\omega_0 t)\mathrm{d}t + \frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \sin((n-m)\omega_0 t)\mathrm{d}t \\ &= 0 + \frac{\omega_0}{\pi}\frac{1}{2}\int_{-\frac{\pi}{\omega_0}}^{\frac{\pi}{\omega_0}} \sin((n-m)\omega_0 t)\mathrm{d}t.\end{aligned} \tag{2.13}$$

O termo $\sin((n+m)\omega_0 t)$, por ser periódico, resulta em uma integral de valor nulo. Semelhantemente, a integral do termo $\sin((n-m)\omega_0 t)$ sempre resultará em um valor nulo para quaisquer n e $m \in \mathbb{N}$.

Desta forma, para qualquer n e m inteiros diferentes de zero, tem-se:

$$\langle \cos(n\omega_0 t), \cos(m\omega_0 t)\rangle = \begin{cases} 1, & \text{se } n = m, \\ 0, & \text{caso contrário;}\end{cases} \tag{2.14}$$

$$\langle \sin(n\omega_0 t), \cos(m\omega_0 t)\rangle = 0, \forall n \text{ e } m; \tag{2.15}$$

$$\langle \sin(n\omega_0 t), \sin(m\omega_0 t)\rangle = \begin{cases} 1, & \text{se } n = m, \\ 0, & \text{caso contrário;}\end{cases} \tag{2.16}$$

$$\langle 1, \sin(n\omega_0 t)\rangle = \langle 1, \cos(n\omega_0 t)\rangle = 0, \forall n \geq 1. \tag{2.17}$$

□

A Definição 2, que trata do produto interno entre duas funções, pode ser útil para a determinação de algumas outras propriedades da série de Fourier de sinais com determinadas simetrias. A seção seguinte aborda quais as propriedades da série de Fourier dos sinais analisados quando estes apresentam simetrias do tipo ímpar, par e $f(t) = \pm f(t \pm T/2)$.

2.2 Os Coeficientes da Série de Fourier e o Produto Interno

Tendo agora a noção de produto interno e ortogonalidade, podemos re-expressar a Definição 1 da seguinte forma.

Definição 3. *A série trigonométrica de Fourier de $f(t)$ é dada por:*

$$\frac{a_0}{2} + \sum_{n=1}^{\infty}(a_n\cos(n\omega_0 t) + b_n\sin(n\omega_0 t)), \tag{2.18}$$

em que $\omega_0 = 2\pi/T$ é chamada de frequência fundamental e $\{a_n, b_n\}_{n=1}^{\infty}$ são coeficiente reais dados por:

$$a_n = \langle f(t), \cos(n\omega_0 t)\rangle, \tag{2.19}$$

$$a_0 = \langle f(t), 1 \rangle, \tag{2.20}$$

e

$$b_n = \langle f(t), \sin(n\omega_0 t \rangle, \tag{2.21}$$

para $n \geq 0$

Usando (2.19), (2.20) e (2.21), é possível extrair algumas propriedades dos coeficientes da expansão em série de Fourier.

Se a função $f(t)$ é ímpar, usando mudança de variáveis na integral do segundo membro de $t' = -t$ e considerando que a função cosseno é par, é imediato que

$$\begin{aligned}
a_n &= \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\cos(n\omega_0 t)\mathrm{d}t \\
&= \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t - \frac{2}{T}\int_{0}^{-\frac{T}{2}} f(-t')\cos(-n\omega_0 t')\mathrm{d}t' \\
&= \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t + \frac{2}{T}\int_{0}^{-\frac{T}{2}} f(t')\cos(n\omega_0 t')\mathrm{d}t' \\
&= \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t - \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t')\cos(n\omega_0 t')\mathrm{d}t' \\
&= 0.
\end{aligned}$$

Assim, $f(t)$, quando expressa por meio da série de Fourier, apresenta coeficientes a_n nulos, para todo $n \geq 0$.

Considerando agora o caso em que $f(t)$ é uma função par, tem-se que

$$\begin{aligned}
b_n &= \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\sin(n\omega_0 t)\mathrm{d}t \\
&= \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t - \frac{2}{T}\int_{0}^{-\frac{T}{2}} f(t')\sin(-n\omega_0 t')\mathrm{d}t' \\
&= \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t + \frac{2}{T}\int_{0}^{-\frac{T}{2}} f(t')\sin(n\omega_0 t')\mathrm{d}t' \\
&= \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t - \frac{2}{T}\int_{-\frac{T}{2}}^{0} f(t')\sin(n\omega_0 t')\mathrm{d}t' \\
&= 0.
\end{aligned}$$

Outros dois casos interessantes são quando $f(t)$ apresenta simetria do tipo $f(t) = f(t \pm T/2)$ e $f(t) = -f(t \pm T/2)$. Quando $f(t) = f(t \pm T/2)$, tem-se

$$a_n = \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\cos(n\omega_0 t)\mathrm{d}t = \int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t + \int_{0}^{\frac{T}{2}} f(t)\cos(n\omega_0 t)\mathrm{d}t.$$

Fazendo-se a mudança de variável $t' = t - T/2$ na segunda integral, tem-se que

$$\begin{aligned} a_n &= \int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t + \int_{0}^{\frac{T}{2}} f(t)\cos(n\omega_0 t)\mathrm{d}t \\ &= \int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t + \int_{-\frac{T}{2}}^{0} f(t'+T/2)\cos(n\omega_0(t'+T/2))\mathrm{d}t'. \end{aligned}$$

Como $n\omega_0 T/2 = n\pi$, tem-se que $\cos(n\omega_0(t'+T/2)) = \sin(n\omega_0 t')(-1)^n$, logo

$$\begin{aligned} a_n &= \int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t + \int_{-\frac{T}{2}}^{0} f(t'+T/2)\sin(n\omega_0 t')(-1)^n\mathrm{d}t' \\ &= \int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t + (-1)^n \int_{-\frac{T}{2}}^{0} f(t')\cos(n\omega_0 t')\mathrm{d}t' \\ &= (1+(-1)^n)\int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t. \end{aligned}$$

Então, se $n = 2k+1$, segue-se que $a_n = 0$.

De maneira análoga, considerando os coeficientes b_n, $n \in \mathbb{N}$, tem-se que

$$\begin{aligned} b_n &= \int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t + \int_{-\frac{T}{2}}^{0} f(t'+T/2)\sin(n\omega_0(t'+T/2))\mathrm{d}t' \\ &= \int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t + (-1)^n \int_{-\frac{T}{2}}^{0} f(t')\sin(n\omega_0 t')\mathrm{d}t' \\ &= (1+(-1)^n)\int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t. \end{aligned}$$

De modo similar, se $n = 2k+1$, tem-se que $b_n = 0$. Se tomarmos o caso $f(t) = f(t - T/2)$, obteremos o mesmo resultado. Para o caso $f(t) = -f(t \pm T/2)$, tomamos $f(t) = -f(t + T/2)$, e obtêm-se

$$\begin{aligned} a_n &= \int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t + \int_{-\frac{T}{2}}^{0} f(t'+T/2)\cos(n\omega_0 t')(-1)^n\mathrm{d}t' \\ &= \int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t - (-1)^n \int_{-\frac{T}{2}}^{0} f(t')\cos(n\omega_0 t')\mathrm{d}t' \\ &= (1-(-1)^n)\int_{-\frac{T}{2}}^{0} f(t)\cos(n\omega_0 t)\mathrm{d}t, \end{aligned}$$

o que leva a $a_n = 0$ se $n = 2k$. Para b_n, nestas condições, tem-se que

$$\begin{aligned} b_n &= \int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t + \int_{-\frac{T}{2}}^{0} f(t'+T/2)\sin(n\omega_0(t'+T/2))\mathrm{d}t' \\ &= \int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t - (-1)^n \int_{-\frac{T}{2}}^{0} f(t')\sin(n\omega_0 t')\mathrm{d}t' \\ &= (1-(-1)^n)\int_{-\frac{T}{2}}^{0} f(t)\sin(n\omega_0 t)\mathrm{d}t. \end{aligned}$$

Tabela 2.1 Propriedades dos coeficientes da série de Fourier

Condição	Coeficientes
$f(t) = f(-t)$	$b_n = 0\ , n \geq 1$
$f(t) = -f(-t)$	$a_n = 0\ , n \geq 0$
$f(t) = f(t \pm T/2)$	$a_{2k+1} = b_{2k+1} = 0\ , n \geq 0$
$f(t) = -f(t \pm T/2)$	$a_{2k} = b_{2k} = 0\ , n \geq 1$

Isto implica que $b_n = 0$ se n é par. Se o caso de simetria $f(t) = -f(t - T/2)$ for considerado, obtêm-se os mesmos resultados para os coeficientes a_n e b_n. A Tabela 2.1 resume as propriedades aqui demonstradas.

Comumente a série de Fourier de uma dada função é representada em sua forma exponencial. Esta é a representação baseada na equação de Leonard Euler [1],

$$e^{j\theta} = \cos(\theta) + j\sin(\theta), \tag{2.22}$$

em que j representa a unidade imaginária $\sqrt{-1}$ [1]. Usando (2.22) e a paridade das funções $\cos(\cdot)$ e $\sin(\cdot)$, podemos obter:

$$\cos(n\omega_0 t) = \frac{1}{2}(e^{jn\omega_0 t} + e^{-jn\omega_0 t}) \tag{2.23}$$

e

$$\sin(n\omega_0 t) = \frac{1}{2j}(e^{jn\omega_0 t} - e^{-jn\omega_0 t}). \tag{2.24}$$

Assim sendo, definindo

$$c_n = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)e^{-jn\omega_0 t}\mathrm{d}t, \tag{2.25}$$

em que $n \in \mathbb{Z}$, obtemos as seguintes relações entre c_n e os coeficientes a_n e b_n previamente discutidos:

$$\begin{aligned} a_n &= \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\cos(n\omega_0 t)\mathrm{d}t \\ &= \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\frac{1}{2}(e^{jn\omega_0 t} + e^{-jn\omega_0 t})\mathrm{d}t \\ &= (c_{-n} + c_n), \end{aligned} \tag{2.26}$$

e

$$\begin{aligned} b_n &= \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\sin(n\omega_0 t)\mathrm{d}t \\ &= \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\frac{1}{2j}(e^{jn\omega_0 t} - e^{-jn\omega_0 t})\mathrm{d}t \\ &= j(c_n - c_{-n}). \end{aligned} \tag{2.27}$$

Desse modo, tem-se que

$$c_n = \frac{1}{2}(a_n - jb_n) \tag{2.28}$$

e

$$c_{-n} = \frac{1}{2}(a_n + jb_n). \tag{2.29}$$

Observe que os resultados acima obtidos correspondem ao coeficiente b_n ser uma função ímpar de n, e a_n uma função par. Observe também que por meio desta definição dos coeficientes da série de Fourier, podemos incluir o termo constante, a_0. De fato, se $n = 0$ tem-se

$$c_0 = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)e^{0\omega_0 t}\mathrm{d}t = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)\mathrm{d}t = a_0. \tag{2.30}$$

É interessante salientar que agora n não se limita a ser um número natural, mas um inteiro, em decorrência da existência dos coeficientes de índice negativo. Para tanto, tomamos o limite do somatório sobre todo os inteiros $\mathbb{Z}$. Sendo assim, a série de Fourier da função $f(t)$ se torna

$$f(t) \cong \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_0 t}, \tag{2.31}$$

em que

$$c_n = \langle f(t), e^{jn\omega_0 t}\rangle = \frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(t)e^{jn\omega_0 t}\mathrm{d}t, \tag{2.32}$$

em que $n \in \mathbb{Z}$.

Uma propriedade de interesse na série de Fourier de uma função com os requisitos listados na Definição 1 é que os coeficientes a_n, b_n e c_n tendem a zero a medida que n cresce. Essa característica desempenha papel fundamental na análise da convergência da série de Fourier, que será tratada na próxima seção. Isso pode ser demonstrado pelo uso da desigualdade de Bessel.

Desigualdade de Bessel: Se $f(t)$ é periódica com período T, integrável no sentido de Riemann no intervalo $[-T/2, T/2]$ e os coeficientes da série de Fourier são definidos por (2.25), então

$$\sum_{n=-\infty}^{\infty} |c_n|^2 \leq \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}} |f(t)|^2\mathrm{d}t. \tag{2.33}$$

Demonstração. Sendo $|z|^2 = z\bar{z}$ para qualquer complexo [36] e $f(t)$ real, tem-se que

$$\begin{aligned}\left| f(t) - \sum_{n=-N}^{N} c_n e^{jn\omega_0 t}\right|^2 &= \left(f(t) - \sum_{n=-N}^{N} c_n e^{jn\omega_0 t}\right)\left(\overline{f(t)} - \sum_{n=-N}^{N} \overline{c_n} e^{-jn\omega_0 t}\right) \\ &= |f(t)|^2 - f(t)\sum_{n=-N}^{N}[c_n e^{jn\omega_0 t} + \overline{c_n} e^{-jn\omega_0 t}] + \sum_{n=-N}^{N}\sum_{m=-N}^{N} c_m \overline{c_n} e^{j(m-n)\omega_0 t}.\end{aligned} \tag{2.34}$$

Dividindo ambos lados da expressão acima por $1/T$, integrando de $-T/2$ até $T/2$ e usando (2.25), tem-se que:

$$\begin{aligned}\frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}\left|f(t)-\sum_{n=-N}^{N}c_n e^{jn\omega_0 t}\right|^2 \mathrm{d}t =& \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}|f(t)|^2\mathrm{d}t \\ &- \sum_{n=-N}^{N}\left[c_n\frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}f(t)e^{jn\omega_0 t}\mathrm{d}t+\overline{c_n}\frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}f(t)e^{-jn\omega_0 t}\mathrm{d}t\right] \\ &+ \sum_{n=-N}^{N}\sum_{m=-N}^{N}c_m\overline{c_n}\frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}e^{j(m-n)\omega_0 t}\mathrm{d}t \\ =& \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}|f(t)|^2\mathrm{d}t-\sum_{n=-N}^{N}[c_n\overline{c_n}+\overline{c_n}c_n]+\sum_{n=-N}^{N}c_n\overline{c_n} \\ =& \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}|f(t)|^2\mathrm{d}t-\sum_{n=-N}^{N}|c_n|^2.\end{aligned}$$

Contudo, o termo quadrático do membro à esquerda da equação é maior ou igual a zero, logo

$$\sum_{n=-N}^{N}|c_n|^2 \leq \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}|f(t)|^2\mathrm{d}t.$$

Fazendo $N\to\infty$, obtemos que

$$\sum_{n=-\infty}^{\infty}|c_n|^2 \leq \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}|f(t)|^2\mathrm{d}t.$$

□

A desigualdade de Bessel desempenha um papel importante no entendimento do comportamento dos coeficientes da série de Fourier quando o número de harmônicos é consideravelmente elevado. Os coeficientes da série de Fourier de uma função com as características exigidas na Definição 1 tendem a zero à medida que o número de harmônicos tende a infinito [16]. Com o objetivo de demonstrar tal afirmação, faremos uso da desigualdade de Bessel em conjunto com as séries absolutamente convergentes [6].

Considere a série $\sum_{n=-\infty}^{\infty}|c_n|^2$. Segundo [31], por se tratar de uma série de valores absolutos, há apenas duas possibilidades com respeito a convergência, ou a série converge ou diverge para infinito. Suponha por absurdo que a série $\sum_{n=-\infty}^{\infty}|c_n|^2$ diverge para infinito. Logo, a desigualdade de Bessel força

$$\frac{1}{T}\int_{\frac{-T}{2}}^{\frac{T}{2}}|f(t)|^2\mathrm{d}t=\infty, \tag{2.35}$$

o que significa que $f(t)$ não é absolutamente integrável, contradizendo a suposição inicial na Definição 1 de que $f(t)$ é absolutamente integrável. Logo, a série $\sum_{n=-\infty}^{\infty}|c_n|^2$ converge. Observe que $\sum_{n=-\infty}^{\infty}|c_n|^2$ é uma série absolutamente convergente. Os resultados a seguir fornecem mais informações sobre cada coeficiente c_n.

Teorema 2. *Toda série absolutamente convergente é convergente [31].*

Teorema 3. *Se a série* $\sum_{n=-N}^{N} a_n$ *converge, então* $\lim a_n = 0$ *[31].*

Aplicando estes resultados à série $\sum_{n=-\infty}^{\infty} |c_n|^2$, tem-se que $\lim c_n = 0$. Este resultado será muito útil na demonstração da convergência da série de Fourier. Considerando a importância deste resultado para este trabalho, apresentamos este resultado e suas consequências sobre a_n e b_n como teorema.

Teorema 4. *Os coeficientes* a_n*,* b_n *e* c_n *da expansão em série de Fourier de uma função* $f(t)$ *que obedece os requisitos de* (1) *tendem a zero a medida em que* $n \to \infty$*.*

A convergência de a_n e b_n a zero quando $n \to \infty$ pode ser verificada através da relação entre a_n e c_n em (2.26), e b_n e c_n em (2.27).

Todas as análises feitas até então foram realizadas supondo que todas as funções periódicas que satisfazem as condições da Definição 1 podem ser representadas por meio da série de Fourier. Contudo, apesar das condições exigidas na Definição 1 serem suficientes para a expansão em série de Fourier de um sinal representado por uma função $f(t)$, estas condições não são necessárias. Avaliemos as condições necessárias sobre a função $f(t)$ para ser expansível em série de Fourier.

2.3 Convergência da Série de Fourier

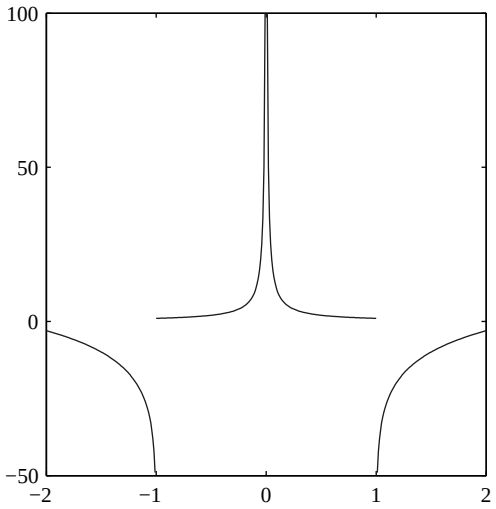
Uma forma de se avaliar a convergência de uma dada série é por considerar a soma parcial dos seus termos [32]. Com a informação do comportamento da soma parcial dos termos da série quando o somatório tende ao infinito é possível verificar se a série é convergente ou não. Para tanto, avaliar a convergência da série de Fourier pode ser reduzido ao problema de se analisar o comportamento da série truncada de Fourier quando o número de termos da soma parcial tende ao infinito. Uma explanação mais detalhada desta abordagem com uma digressão sobre séries infinitas pode ser verificada no Apêndice E.

Considerando a série de Fourier até o N-ésimo termo, e usando (2.18) e (2.31), define-se

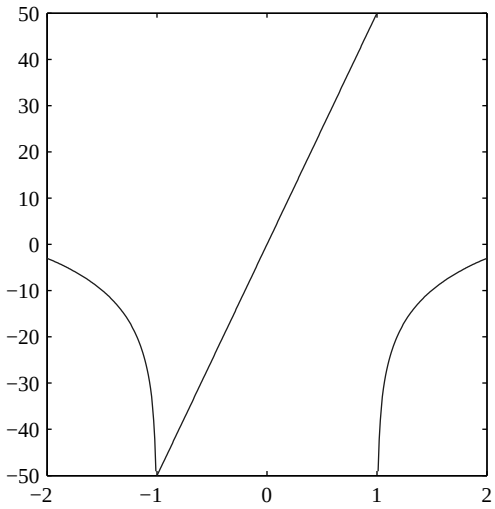
$$S_N^f(t) \triangleq a_0 + \sum_{n=1}^{N} (a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)) = \sum_{n=-N}^{N} c_n e^{jn\omega_0 t}, \tag{2.36}$$

em que $S_N^f(t)$ representa a série de Fourier truncada no N-ésimo termo. Desta forma, analisar a convergência se limita a avaliar se $\lim_{N\to\infty} S_N^f(t) = S^f(t)$, em que representa $S^f(t)$ a série de Fourier infinita, ou, simplesmente, a série de Fourier de $f(t)$.

Antes de analisar a série de Fourier é necessário fazer algumas observações sobre o estudo de funções reais. Uma propriedade fundamental no estudo de funções é a continuidade e a suavidade. De fato, a condição de continuidade apresentada no início deste capítulo para a representação de uma função em série de Fourier não é necessária. Com respeito a continuidade, basta que a função seja contínua por partes. É apresentado a seguir duas definições que são requisitos para as funções reais apresentarem convergência de suas séries de Fourier.



(a) Função não suave por partes, pois sua derivada na origem não existe.



(b) Função suave por partes.

Figura 2.1 Exemplo de funções suaves e não suaves por partes.

Definição 4. *Uma função $f(t)$ é uma função contínua por partes em um intervalo fechado $[a,b]$, em que $-\infty < a < b < \infty$, se $f(t)$ é contínua em $[a,b]$ exceto em um número finito de pontos, $x_1, x_2, \ldots, x_k$, e os limites laterais de $f(t)$ nestes pontos existem [2].*

Uma exceção surge com essa definição. Quando um dos pontos de descontinuidade, $x_1, x_2, \ldots, x_k$, é um ponto extremo do intervalo fechado $[a,b]$, é necessário apenas que o limite lateral pela vizinhança contida no intervalo exista. Por exemplo, se b é um ponto de descontinuidade, basta que o limite lateral pela esquerda de b exista. De modo semelhante, se a é um ponto de descontinuidade, basta que o limite lateral pela direita de a exista. Por simplificação, será denotado por $CP(a,b)$ o conjunto de todas as funções reais contínuas por partes no intervalo fechado $[a,b]$.

A segunda definição necessária para demonstrar a convergência da série de Fourier é a suavidade por partes.

Definição 5. *Uma função $f(t)$ é uma função suave por partes no intervalo $[a,b]$ se $f(t)$ e sua primeira derivada, $f'(t)$, pertencem ao conjunto $CP(a,b)$.*

O conjunto de todas as funções suaves por partes no intervalo fechado $[a,b]$ é denotado por $SP(a,b)$. Quando uma função $f(t) \in CP(a,b)$ e $f(t) \in SP(a,b)$ para todo intervalo limitado $[a,b]$ de $\mathbb{R}$, dize-se que $f(t)$ é contínua e suave por partes em $\mathbb{R}$. A propriedade da suavidade recebe esse nome por causa do que é visto no gráfico das funções com tais características. A Figura 2.1 exibe o exemplo de uma função suave e não suave.

Retornando para as funções periódicas, a partir de (2.36), usando o resultado da segunda

igualdade, e o resultado de (2.25), tem-se

$$S_N^f(t) = \sum_{n=-N}^{N} \left(\frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(\psi) e^{-jn\omega_0 \psi} d\psi \right) e^{jn\omega_0 t} = \frac{1}{T} \sum_{n=-N}^{N} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(\psi) e^{jn\omega_0 (t-\psi)} d\psi.$$

Realizando a mudança de variável $\phi = \psi - t$ e substituindo $-n$ por n, tem-se

$$S_N^f(t) = \frac{1}{T} \sum_{n=-N}^{N} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(\phi + t) e^{jn\omega_0 \phi} d\phi.$$

Usando a propriedade da linearidade da integral [2], tem-se

$$S_N^f(t) = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} \sum_{n=-N}^{N} f(\phi + t) e^{jn\omega_0 \phi} d\phi.$$

Observando que a soma é em n, logo pode-se reescrever

$$\begin{aligned}
S_N^f(t) =& \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} \sum_{n=-N}^{N} f(\phi + t) e^{jn\omega_0 \phi} d\phi \\
=& \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} [f(\phi + t) e^{-jN\omega_0 \phi} + f(\phi + t) e^{j(1-N)\omega_0 \phi} + \dots \\
& + f(\phi + t) e^{j(N-1)\omega_0 \phi} + f(\phi + t) e^{jN\omega_0 \phi}] d\phi \\
=& \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(\phi + t) \sum_{n=-N}^{N} e^{jn\omega_0 \phi} d\phi \\
=& \frac{2\pi}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(\phi + t) \frac{1}{2\pi} \sum_{n=-N}^{N} e^{jn\omega_0 \phi} d\phi \\
=& \omega_0 \int_{-\frac{T}{2}}^{\frac{T}{2}} f(\phi + t) D_N(\omega_0 \phi) d\phi,
\end{aligned}$$

em que

$$D_N(\omega_0 \phi) = \frac{1}{2\pi} \sum_{n=-N}^{N} e^{jn\omega_0 \phi},$$

representa o N-ésimo núcleo de Dirichlet [16].

Pode-se reescrever o N-ésimo núcleo de Dirichlet de forma compacta, reconhecendo que esta é uma série geométrica de razão $e^{j\omega_0 \phi}$ e termo inicial $e^{-jN\omega_0 \phi}/2\pi$. Assim, usando as relações decorrentes de (2.22), tem-se

$$D_N(\omega_0 \phi) = \frac{1}{2\pi} \frac{\sin((N + 1/2)\omega_0 \phi)}{\sin(\frac{1}{2}\omega_0 \phi)}.$$

Antes de prosseguir, é importante verificar duas igualdades que serão usadas na demonstração da convergência. Considerando o N-ésimo núcleo de Dirichlet, tem-se que

$$\int D_N(\omega_0 \phi) d\phi = \frac{1}{\omega_0} \left[\frac{\phi}{2\pi} + \frac{1}{\omega_0 \pi} \sum_{n=1}^{N} \sin(n\omega_0 \phi) \right] + C,$$

em que C representa uma constante real qualquer. Assim, tem-se

$$\int_0^{\frac{T}{2}} D_N(\omega_0\phi)d\phi = \frac{1}{\omega_0}\left[\frac{\phi}{2\pi} + \frac{1}{\omega_0\pi}\sum_{n=1}^{N}\sin(n\omega_0\phi)\right]\Bigg|_0^{\pi} = \frac{1}{2}$$

e

$$\int_{\frac{T}{2}}^{0} D_N(\omega_0\phi)d\phi = \frac{1}{\omega_0}\left[\frac{\phi}{2\pi} + \frac{1}{\omega_0\pi}\sum_{n=1}^{N}\sin(n\omega_0\phi)\right]\Bigg|_{-\pi}^{0} = \frac{1}{2}.$$

Foi assumido nas seções anteriores que para que uma função $f(t)$ tivesse uma expansão em série de Fourier ela deveria ser contínua sobre todo o seu período. De fato, isto não é necessário. É assumido agora que a função $f(t)$ é apenas contínua por partes, e que o valor da série de Fourier de $f(t)$ no período vale $1/2[f(t_+) + f(t_-)]$, em que $f(t_+)$ e $f(t_-)$ denotam os limites laterais pela direita e esquerda de $f(t)$, respectivamente. Observe que com essa definição a série de Fourier de $f(t)$, $S^f(t)$, fica definida para todos os pontos do intervalo. O interesse agora é demonstrar que $S_N^f(t)$ converge para $S^f(t)$ quando $N \to \infty$. Assim,

$$\begin{aligned} S_N^f(t) - \frac{1}{2}[f(t_+) + f(t_-)] &= \int_{\frac{-T}{2}}^{0}[f(\phi+t) - f(t_-)]D_N(\omega_0\phi)d\phi \\ &\quad + \int_0^{\frac{T}{2}}[f(\phi+t) - f(t_+)]D_N(\omega_0\phi)d\phi. \end{aligned}$$

É definido uma nova função $g(\phi)$, em função de $f(t)$, da seguinte forma

$$g(\phi) = \begin{cases} \frac{f(\phi+t)-f(\phi_-)}{e^{j\omega_0\phi}-1} & \text{para} \quad \frac{-T}{2} < \phi < 0, \\ \frac{f(\phi+t)-f(\phi_+)}{e^{j\omega_0\phi}-1} & \text{para} \quad 0 < \phi < \frac{T}{2}. \end{cases}$$

A função $g(\phi)$ é tão suave e contínua quanto $f(t)$ em $[-T/2, T/2]$, exceto em $\phi = 0$, em que $e^{j\omega_0\phi} - 1$ se aproxima de 0, onde $g(t)$ é descontínua. Contudo, pela regra de l'Hôspital [2]

$$\lim_{\phi\to 0_+} g(\phi) = \lim_{\phi\to 0_+} \frac{f(\phi+t) - f(t_+)}{e^{j\omega_0\phi} - 1} = \lim_{\phi\to 0_+} \frac{f'(\phi+t)}{j\omega_0 e^{j\omega_0\phi}} = \frac{f'(t_+)}{j\omega_0}.$$

O mesmo pode ser verificado para o limite pela esquerda, resultando em

$$\lim_{\phi\to 0_-} g(\phi) = \frac{f'(t_-)}{j\omega_0}.$$

Logo, $g(t)$ é contínua por partes em $[-T/2, T/2]$, e obedece as condições necessárias estabelecidas para se ter uma representação em série de Fourier. Assim, a igualdade se torna

$$\begin{aligned} S_N^f(t) - \frac{1}{2}[f(t_+) + f(t_-)] &= \int_{\frac{-T}{2}}^{0} g(\phi)[e^{j(N+1)\omega_0\phi} + e^{-jN\omega_0\phi}]d\phi \\ &\quad + \int_0^{\frac{T}{2}} g(\phi)[e^{j(N+1)\omega_0\phi} + e^{-jN\omega_0\phi}]d\phi \\ &= \int_{\frac{-T}{2}}^{\frac{T}{2}} g(\phi)[e^{j(N+1)\omega_0\phi} + e^{-jN\omega_0\phi}]d\phi, \end{aligned}$$

usando definição de c_n em (2.25), tem-se

$$S_N^f(t) - \frac{1}{2}[f(t_+) + f(t_-)] = c^g_{-(N+1)} - c^g_N,$$

em que $c^g_{-(N+1)}$ e c^g_N representam o $-(N+1)$ e N-ésimo coeficiente c_n da série de Fourier da função $g(t)$. Para concluir a demonstração é aplicado o Teorema 4.

$$\begin{aligned}\lim_{N\to\infty} S_N^f(t) - \frac{1}{2}[f(t_+) + f(t_-)] &= \lim_{N\to\infty}(c^g_{-(N+1)} - c^g_N) \\ &= \lim_{N\to\infty} c^g_{-(N+1)} - \lim_{N\to\infty} c^g_N = 0.\end{aligned}$$

Logo, vem que

$$\lim_{N\to\infty} S_N^f(t) = S^f(t).$$

CAPÍTULO 3

A Série de Fourier Quantizada

Neste capítulo, a série de Fourier Quantizada proposta por D. F. Souza, R. J. Cintra e H. M. Oliveira [11] é introduzida e analisada. Sua relação com a série de Fourier clássica é investigada, e uma forma de obtê-la em função da série de Fourier Quantizada é apresentada.

3.1 Motivação

A série de Fourier clássica é largamente aplicada em sistema de detecção harmônica [13]. Usualmente, em tais sistemas é processada uma grande massa de dados. Não obstante, é exigido de tais sistemas respostas rápidas às entradas aos quais os sistemas são expostos. Por envolver naturalmente a função $\cos(\cdot)$ e $\sin(\cdot)$ no cálculo dos coeficientes, analisar os sinais de entrada com tal ferramenta demanda operações em aritmética de ponto flutuante [16].

Em contextos em que outras ferramentas para processamento de sinais são usadas, como por exemplo a DFT, técnicas de quantização das quantidades envolvidas nos cálculos para o processamento dos dados tem sido desenvolvidas pela comunidade com a finalidade de reduzir a complexidade das operações envolvidas [23, 28].

Neste trabalho é proposta a base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=1}^{\infty}$ para representação de sinais periódicos com as propriedades exigidas na Definição 1. Esta representação é capaz de fornecer aproximações para os coeficientes da série de Fourier em contextos de detecção harmônica. Tais aproximações são computadas de tal modo a não exigir aritmética em ponto flutuante. Com esta representação é de interesse aplicá-la em situações que exijam a computação de um grande volume de dados e de forma rápida. Ao eliminar a multiplicação por pontos flutuantes, como consequência, o tempo de computação é reduzido em detrimento da perda de precisão na representação do valor desejado. Em verdade, nesta representação não é obtida a série de Fourier clássica do sinal de entrada, contudo, uma aproximação dela.

Antes de definir o que é a série de Fourier Quantizada, serão investigadas algumas propriedades da base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=1}^{\infty}$ fundamentais para o compreensão da série de Fourier Quantizada. A seção seguinte trata de algumas propriedades da série de Fourier Quantizada.

3.2 Propriedades da Base Quantizada de Fourier

Diferentemente da base da série de Fourier clássica, as funções da base proposta podem assumir apenas quantidades finitas. As funções da base proposta assumem os valores 1 e -1, que são exatamente representados em aritmética de ponto fixo. Tal propriedade permite à base

proposta ser usada em situações em que seja necessário reduzir a complexidade das operações envolvidas.

A Figura 3.1 exibe algumas funções da base proposta para o intervalo $[0, 2\pi]$. O conhecimento da teoria das funções é de ajuda para compreender o comportamento dos elementos da base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ [2, 5]. Considerando que as funções da base proposta nada mais são do que funções resultantes de composições de funções trigonométricas $\cos(\cdot)$ e $\sin(\cdot)$ com a função $\text{sign}(\cdot)$, é de se esperar que as funções da base proposta sejam funções periódicas com o mesmo período fundamental do argumento da função $\text{sign}(\cdot)$.

A série de Fourier Quantizada não goza das mesmas propriedades da série de Fourier clássica. A seguir é apresentada uma propriedade presente na base da série de Fourier clássica que não o é na base proposta da série de Fourier Quantizada. Esta propriedade é a ortogonalidade. Por reduzir o número de operações para a obtenção computacional da representação de sinais, a ortogonalidade comumente é uma propriedade de interesse. A ausência de ortogonalidade da base proposta, $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$, faz com que sejam necessários alguns cálculos adicionais antes de exprimir o sinal nesta nova base. Ao contrário da série de Fourier clássica, os coeficientes não são obtidos diretamente após a computação dos produtos internos de $f(t)$ pelos elementos da base proposta.

3.2.1 Não-Ortogonalidade da Base da Série de Fourier Quantizada

A ortogonalidade, uma propriedade que a base da série de Fourier clássica goza, não está igualmente presente na base da série de Fourier Quantizada. Para verificar isto, é preciso analisar o comportamento dos elementos desta base de acordo com o produto interno proposto em (2). Há quatro possibilidades:

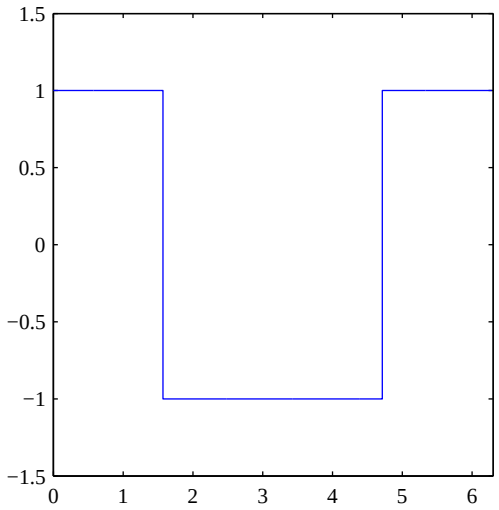
1. $\langle \text{sign}(\cos(n\omega_0 t)), \text{sign}(\cos(k\omega_0 t)) \rangle$,
2. $\langle \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(k\omega_0 t)) \rangle$,
3. $\langle \text{sign}(\sin(n\omega_0 t)), \text{sign}(\sin(k\omega_0 t)) \rangle$ e,
4. o produto interno entre a unidade e as funções $\cos(n\omega_0 t))$ e $\text{sign}(\sin(k\omega_0 t))$,

em que $n, k \in \mathbb{N}$. Para simplificar e facilitar a análise, é utilizada a expansão em série de Fourier dos sinais $\text{sign}(\cos(n\omega_0 t))$ e $\text{sign}(\sin(n\omega_0 t))$, dadas por

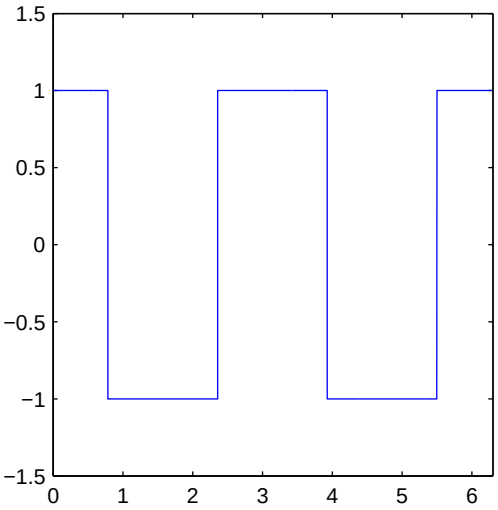
$$\text{sign}(\cos(n\omega_0 t)) = \sum_{l=1}^{\infty} \alpha_l \cos((2l+1)n\omega_0 t), \tag{3.1}$$

em que

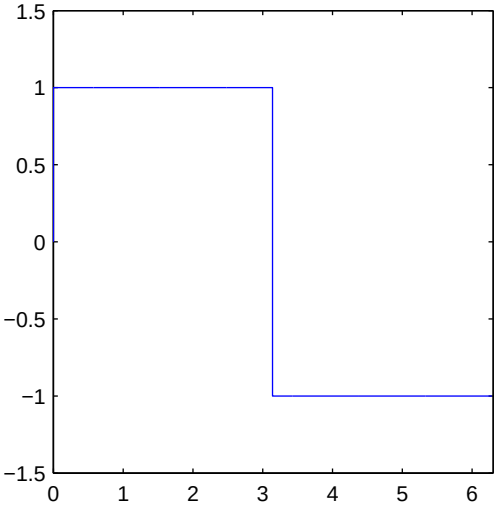
$$\alpha_l = \frac{4}{(2l+1)\pi}(-1)^l, \tag{3.2}$$



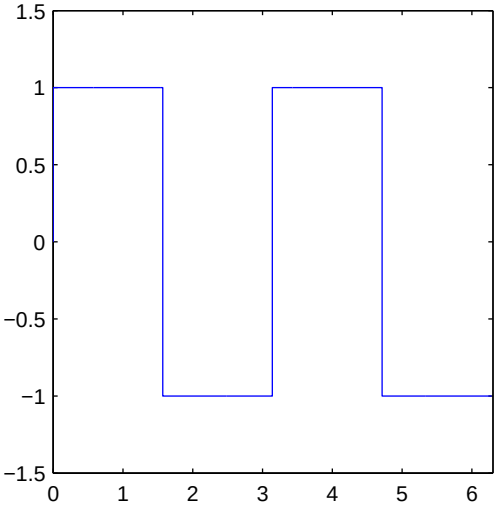
(a) Função sign(cos(t)) para $t \in [0, 2\pi]$



(b) Função sign(cos(2t)) para $t \in [0, 2\pi]$



(c) Função sign(sin(t)) para $t \in [0, 2\pi]$



(d) Função sign(sin(2t)) para $t \in [0, 2\pi]$

Figura 3.1 Algumas funções da base quantizada proposta.

e

$$\text{sign}(\sin(n\omega_0 t)) = \sum_{l=1}^{\infty} \beta_l \sin((2l+1)n\omega_0 t), \tag{3.3}$$

em que

$$\beta_l = \frac{4}{(2l+1)\pi}, \tag{3.4}$$

e o Teorema 1 em conjunto com o seguinte resultado do Teorema 5.

Teorema 5. *Seja $f(t)$ uma função Riemann integrável [2], periódica e ímpar. Então, tem-se que*

$$\int_T f(t)dt = 0. \tag{3.5}$$

Demonstração. Usando as suposições mencionadas e mudanças de variáveis, obtêm-se

$$\begin{aligned}\int_T f(t)dt &= \int_{\frac{-T}{2}}^{\frac{T}{2}} f(t)dt = \int_{\frac{-T}{2}}^{0} f(t)dt + \int_0^{\frac{T}{2}} f(t)dt \\ &= \int_{\frac{T}{2}}^{0} f(-t)(-dt) + \int_0^{\frac{T}{2}} f(t)dt = \int_{\frac{T}{2}}^{0} -f(t)(-dt) + \int_0^{\frac{T}{2}} f(t)dt \\ &= \int_{\frac{T}{2}}^{0} f(t)dt + \int_0^{\frac{T}{2}} f(t)dt = -\int_0^{\frac{T}{2}} f(t)dt + \int_0^{\frac{T}{2}} f(t)dt = 0. \end{aligned} \tag{3.6}$$

□

De acordo com a teoria das funções, considerando a que a função $\cos(n\omega_0 t)$ é uma função par, a função $\text{sign}(\cos(n\omega_0 t))$ se trata também de uma função par. Sendo assim, os coeficientes de Fourier dos elementos de base $\sin(n\omega_0 t)$ da função $\text{sign}(\cos(n\omega_0 t))$ são todos zero [2,5]. De modo semelhante, se tratando a função $\text{sign}(\sin(n\omega_0 t))$ de uma função ímpar, os coeficientes de Fourier dos termos de $\cos(n\omega_0 t))$ são todos nulos [19].

Analisando cada caso individualmente, pode-se estabelecer a não-ortogonalidade da base proposta.

1. Seja considerado o primeiro dos casos, $\langle \text{sign}(\cos(n\omega_0 t)), \text{sign}(\cos(k\omega_0 t))\rangle$.

$$
\begin{aligned}
&\langle \text{sign}(\cos(n\omega_0 t)), \text{sign}(\cos(k\omega_0 t))\rangle = \\
&\frac{2}{T}\int_T \text{sign}(\cos(n\omega_0 t))\,\text{sign}(\cos(k\omega_0 t))\mathrm{d}t \\
&\frac{2}{T}\int_T \Big(\sum_{l=1}^{\infty}\alpha_l \cos((2l+1)n\omega_0 t)\Big)\Big(\sum_{p=1}^{\infty}\alpha_p \cos((2p+1)k\omega_0 t)\Big)\mathrm{d}t \\
&\sum_{l,p=1}^{\infty}\alpha_l\alpha_p\frac{2}{T}\int_T \cos((2l+1)n\omega_0 t)\cos((2p+1)k\omega_0 t)\mathrm{d}t \\
&\sum_{l,p=1}^{\infty}\alpha_l\alpha_p\langle\cos((2l+1)n\omega_0 t),\cos((2p+1)k\omega_0 t)\rangle \\
&\left(\frac{4}{\pi}\right)^2\sum_{l,p=1}^{\infty}\frac{(-1)^{l+p}}{(2l+1)(2p+1)}\langle\cos((2l+1)n\omega_0 t),\cos((2p+1)k\omega_0 t)\rangle.
\end{aligned}
\tag{3.7}
$$

Empregando-se (2.14) e (2.17), nota-se que o produto interno tem valor diferente de zero apenas quando $(2l+1)n = (2p+1)k$. Observe que a razão entre n/k deve gerar uma fração irredutível, cujos numerador e denominador devem ser ímpares. Usamos assim a definição da função chi-fração dada a seguir [11].

Definição 6. *Sejam n e k dois inteiros que formam a fração irredutível*

$$\frac{n}{k} = \frac{p}{q}. \tag{3.8}$$

A função chi-fração é definida por

$$\chi(n,k) \triangleq \chi\left(\frac{n}{k}\right) = \chi\left(\frac{p}{q}\right) = \begin{cases} \frac{1}{p\cdot q}, & \textit{provido que } p \textit{ e } q \textit{ são ímpares,} \\ 0, & \textit{caso contrário.} \end{cases} \tag{3.9}$$

Aplicando a Definição 6, em que $\frac{2p+1}{2l+1}$ é a fração irredutível de $\frac{n}{k}$,

$$
\begin{aligned}
&\langle \text{sign}(\cos(n\omega_0 t)), \text{sign}(\cos(k\omega_0 t))\rangle = \\
&= \left(\frac{4}{\pi}\right)^2\sum_{l=1}^{\infty}\sum_{p=1}^{\infty}\frac{(-1)^{l+p}}{(2l+1)(2p+1)}\langle\cos((2l+1)n\omega_0 t),\cos((2p+1)k\omega_0 t)\rangle \\
&= (-1)^{l+p}\left(\frac{4}{\pi}\right)^2\left(\chi(n,k)\frac{1}{1^2}+\chi(n,k)\frac{1}{3^2}+\chi(n,k)\frac{1}{5^2}\cdots\right) \\
&= (-1)^{l+p}\left(\frac{4}{\pi}\right)^2\chi(n,k)\sum_{z=0}^{\infty}\frac{1}{(2z+1)^2}.
\end{aligned}
$$

em que $\frac{2p+1}{2l+1}$ é a fração irredutível de $\frac{n}{k}$. Lembrando que $\sum_{z=0}^{\infty}\frac{1}{(2z+1)^2}=\frac{\pi^2}{8}$, tem-se que

$$\begin{aligned}\langle \text{sign}(\cos(n\omega_0 t)),\text{sign}(\cos(k\omega_0 t))\rangle =&(-1)^{l+p}\left(\frac{4}{\pi}\right)^2\chi(n,k)\sum_{z=0}^{\infty}\frac{1}{(2z+1)^2} \qquad (3.10)\\ =&(-1)^{l+p}\left(\frac{4}{\pi}\right)^2\chi(n,k)\frac{\pi^2}{8},\end{aligned}$$

o que leva a

$$\langle \text{sign}(\cos(n\omega_0 t)),\text{sign}(\cos(k\omega_0 t))\rangle = 2(-1)^{l+p}\chi(n,k), \qquad (3.11)$$

em que $\frac{2p+1}{2l+1}$ é a fração irredutível de $\frac{n}{k}$. Assim, as funções $\text{sign}(\cos(n\omega_0 t))$ e $\text{sign}(\cos(k\omega_0 t))$ da base proposta não são ortogonais entre si.

2. Seja considerado o caso de $\langle \text{sign}(\cos(n\omega_0 t)),\text{sign}(\sin(k\omega_0 t))\rangle$.

$$\begin{aligned}&\langle \text{sign}(\cos(n\omega_0 t)),\text{sign}(\sin(k\omega_0 t))\rangle =\\ &=\frac{2}{T}\int_T \text{sign}(\cos(n\omega_0 t))\,\text{sign}(\sin(k\omega_0 t))\,\mathrm{d}t\\ &=\frac{2}{T}\int_T\Big(\sum_{l=1}^{\infty}\alpha_l\cos((2l+1)n\omega_0 t)\Big)\Big(\sum_{p=1}^{\infty}\beta_p\sin((2p+1)k\omega_0 t)\Big)\mathrm{d}t\\ &=\sum_{l,p=1}^{\infty}\alpha_l\beta_p\frac{2}{T}\int_T\cos((2l+1)n\omega_0 t)\sin((2p+1)k\omega_0 t)\mathrm{d}t\\ &=\sum_{l,p=1}^{\infty}\alpha_l\beta_p\langle\cos((2l+1)n\omega_0 t),\sin((2p+1)k\omega_0 t)\rangle.\end{aligned}$$

Usando (2.15) e (2.17), obtêm-se

$$\langle \text{sign}(\cos(n\omega_0 t)),\text{sign}(\sin(k\omega_0 t))\rangle = 0. \qquad (3.12)$$

Logo, as funções $\text{sign}(\cos(n\omega_0 t))$ e $\text{sign}(\sin(k\omega_0 t))$ da base proposta são sempre ortogonais entre si, independentemente de n e k.

3. Considerando o caso $\langle \text{sign}(\sin(n\omega_0 t)),\text{sign}(\sin(k\omega_0 t))\rangle$, obtêm-se

$$\begin{aligned}&\langle \text{sign}(\sin(n\omega_0 t)),\text{sign}(\sin(k\omega_0 t))\rangle =\\ &=\frac{2}{T}\int_T \text{sign}(\sin(n\omega_0 t))\,\text{sign}(\sin(k\omega_0 t))\mathrm{d}t\\ &=\frac{2}{T}\int_T\Big(\sum_{l=1}^{\infty}\beta_l\sin((2l+1)n\omega_0 t)\Big)\Big(\sum_{p=1}^{\infty}\beta_p\sin((2p+1)k\omega_0 t)\Big)\mathrm{d}t\\ &=\sum_{l,p=1}^{\infty}\beta_l\beta_p\frac{2}{T}\int_T\sin((2l+1)n\omega_0 t)\sin((2p+1)k\omega_0 t)\mathrm{d}t\\ &=\sum_{l,p=1}^{\infty}\beta_l\beta_p\langle\sin((2l+1)n\omega_0 t),\sin((2p+1)k\omega_0 t)\rangle.\end{aligned}$$

Novamente, usando (2.16) e (2.17) em conjunto com a Definição 6, tem-se que

$$\begin{aligned}
&\langle \text{sign}(\sin(n\omega_0 t)), \text{sign}(\sin(k\omega_0 t))\rangle = \\
&= \left(\frac{4}{\pi}\right)^2 \sum_{l,p=1|l=p}^{\infty} \frac{1}{(2l+1)(2p+1)} \langle \sin((2l+1)n\omega_0 t), \sin((2p+1)k\omega_0 t)\rangle \\
&= \left(\frac{4}{\pi}\right)^2 \chi(n,k)\{\frac{1}{1^2}+\frac{1}{3^2}+\frac{1}{5^2}+\ldots\} \\
&= \left(\frac{4}{\pi}\right)^2 \chi(n,k)\left(\frac{\pi^2}{8}\right), \qquad (3.13)
\end{aligned}$$

o que resulta em

$$\langle \text{sign}(\sin(n\omega_0 t)), \text{sign}(\sin(k\omega_0 t))\rangle = 2\chi(n,k). \qquad (3.14)$$

Logo, as funções $\text{sign}(\cos(n\omega_0 t))$ e $\text{sign}(\cos(k\omega_0 t))$ da base proposta não são ortogonais entre si.

4. Por último, sejam considerados os casos $\langle \text{sign}(\cos(n\omega_0 t)), 1\rangle$ e $\langle \text{sign}(\sin(n\omega_0 t)), 1\rangle$.

$$\begin{aligned}
\langle \text{sign}(\cos(n\omega_0 t)), 1\rangle =& \frac{2}{T}\int_T \text{sign}(\cos(n\omega_0 t))1\mathrm{d}t \\
=& \frac{2}{T}\int_T \Big(\sum_{l=1}^{\infty} \alpha_l \cos((2l+1)n\omega_0 t)\Big)\mathrm{d}t \\
=& \frac{2}{T}\sum_{l=1}^{\infty} \alpha_l \int_T \cos((2l+1)n\omega_0 t)\mathrm{d}t = 0, \qquad (3.15)
\end{aligned}$$

e

$$\begin{aligned}
\langle \text{sign}(\sin(n\omega_0 t)), 1\rangle =& \frac{2}{T}\int_T \text{sign}(\sin(n\omega_0 t))1\mathrm{d}t \\
=& \frac{2}{T}\int_T \Big(\sum_{l=1}^{\infty} \alpha_l \sin((2l+1)n\omega_0 t)\Big)\mathrm{d}t \\
=& \frac{2}{T}\sum_{l=1}^{\infty} \alpha_l \int_T \sin((2l+1)n\omega_0 t)\mathrm{d}t = 0. \qquad (3.16)
\end{aligned}$$

Logo, os elementos da base proposta $\text{sign}(\cos(n\omega_0 t))$ e $\text{sign}(\sin(n\omega_0 t))$ sempre são ortogonais ao vetor 1, independentemente de n.

Conclui-se assim que $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ não forma uma base ortogonal.

Por questão de simplificação de notação, chamaremos as grandezas dadas por (3.11) e (3.13) de

$$A_{n,k} = 2(-1)^{l+p}\chi(n,k), \qquad (3.17)$$

em que $\frac{2p+1}{2l+1}$ é a fração irredutível de $\frac{n}{k}$, e

$$B_{n,k} = 2\chi(n,k). \qquad (3.18)$$

Esta notação será útil mais tarde.

3.3 A Série de Fourier Quantizada

Seja $f(t)$ uma função real, contínua, definida na reta real e que seja periódica com período T, ou seja, $f(t+T) = f(t)$ para todo $t \in \mathbb{R}$. Admita que $f(t)$ seja absolutamente integrável em todo intervalo fechado na reta real $\mathbb{R}$ [2].

Definição 7. *A série de Fourier Quantizada de $f(t)$ é dada por:*

$$\frac{\hat{a}_0}{2} + \sum_{n=1}^{\infty} (\hat{a}_n \operatorname{sign}(\cos(n\omega_0 t)) + \hat{b}_n \operatorname{sign}(\sin(n\omega_0 t))), \tag{3.19}$$

em que $\omega_0 = \frac{2\pi}{T}$ é chamada de frequência fundamental e $\hat{a}_0$ e $\{\hat{a}_n, \hat{b}_n\}_{n=1}^{\infty}$ são coeficiente reais.

A definição acima, diferentemente da definição da série de Fourier clássica, não especifica como os coeficientes da expensão em série de Fourier Quantizada podem ser encontrados. O cálculo para a obtenção dos coeficientes da série de Fourier Quantizada, $\hat{a}_n$ e $\hat{b}_n$, será abordado na Seção 3.4.

Diferentemente do que ocorre com os coeficientes da série de Fourier clássica, os coeficientes da série de Fourier Quantizada não podem ser encontrado diretamente pelo simples produtos interno da função analisada com o respectivo elemento da base proposta. Embora isso estritamente seja verdade, D. F. Souza, R. J. Cintra e H. M. Oliveira [11] ao proporem o uso da base $\{1, \operatorname{sign}(\cos(n\omega_0 t)), \operatorname{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ procederam com os cálculos para estimativa harmônica como se esta formasse uma base ortogonal. Sem dúvida, isto é uma aproximação que, em determinados contextos e aplicações, pode ser aplicada e cujos erros provenientes das aproximações sejam toleráveis. Embora essa seja uma forma de endereçar o problema da detecção harmônica, isto não é feito neste trabalho.

Para que sejamos capazes de computar os coeficientes da série de Fourier, a_n e b_n, devemos ter em mãos os coeficientes da série de Fourier Quantizada, $\hat{a}_n$ e $\hat{b}_n$. Na próxima seção é proposta uma forma de se obter os coeficientes da série de Fourier Quantizada com uma formulação matricial do problema.

3.4 Formulação Matricial e os Coeficientes $\hat{a}_n$ e $\hat{b}_n$

Como discutido nas seções anteriores, considerando que a base da série de Fourier Quantizada, $\{1, \operatorname{sign}(\cos(n\omega_0 t)), \operatorname{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$, não é ortogonal, para obter cada $\hat{a}_n$ e $\hat{b}_n$ não pode-se simplesmente computar os produtos internos $\langle f(t), \operatorname{sign}(\cos(n\omega_0 t))\rangle$ e $\langle f(t), \operatorname{sign}(\sin(n\omega_0 t))\rangle$. Segue agora a investigação de como obter cada $\hat{a}_n$ e $\hat{b}_n$.

Sendo

$$f(t) \triangleq \frac{\hat{a}_0}{2} + \sum_{n=0}^{\infty} (\hat{a}_n \operatorname{sign}(\cos(n\omega_0 t)) + \hat{b}_n \operatorname{sign}(\sin(n\omega_0 t))), \tag{3.20}$$

a expansão da função $f(t)$ na base $\{1, \operatorname{sign}(\cos(n\omega_0 t)), \operatorname{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ nas condições da Definição 7, realizando o produto interno $\langle f(t), \operatorname{sign}(\cos(k\omega_0 t))\rangle$ e usando (3.11) e (3.12)

obtêm-se

$$\langle f(t), \text{sign}(\cos(k\omega_0 t))\rangle = \sum_{n=0}^{\infty} \hat{a}_n A_{n,k}. \tag{3.21}$$

Realizando o produto interno $\langle f(t), \text{sign}(\sin(k\omega_0 t))\rangle$ e usando (3.13) e (3.12) obtêm-se

$$\langle f(t), \text{sign}(\sin(k\omega_0 t))\rangle = \sum_{n=0}^{\infty} \hat{b}_n B_{n,k}. \tag{3.22}$$

O produto interno $\langle f(t), 1\rangle$ resulta em

$$\langle f(t), 1\rangle = \hat{a}_0. \tag{3.23}$$

Logo, para obter cada $\hat{a}_n$, exceto $\hat{a}_0$, considerando a aproximação até o N-ésimo harmônico, deve-se resolver o sistema linear

$$\mathbf{M}_{A_N} \cdot \hat{\mathbf{a}}_N = \mathbf{v}_N, \tag{3.24}$$

em que $\mathbf{M}_{A_N}$ é a matriz simétrica que considera a expansão de $f(t)$ até o N-ésimo harmônico,

$$\mathbf{M}_{A_N} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,N} \\ A_{2,1} & A_{2,2} & \dots & A_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N,1} & A_{N,2} & \dots & A_{N,N} \end{bmatrix}, \tag{3.25}$$

em que $A_{n,k}$ é dado por (3.17),

$$\mathbf{v}_N = \begin{bmatrix} \langle f(t), \text{sign}(\cos(\omega_0 t))\rangle \\ \langle f(t), \text{sign}(\cos(2\omega_0 t))\rangle \\ \vdots \\ \langle f(t), \text{sign}(\cos(N\omega_0 t))\rangle \end{bmatrix}, \tag{3.26}$$

e

$$\hat{\mathbf{a}}_N = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_N \end{bmatrix}. \tag{3.27}$$

De modo semelhante, para obter $\hat{b}_n$, deve-se resolver o sistema

$$\mathbf{M}_{B_N} \cdot \hat{\mathbf{b}}_N = \mathbf{w}_N, \tag{3.28}$$

em que $\mathbf{M}_{B_N}$ também é uma matriz simétrica que considera a expansão de $f(t)$ até o N-ésimo harmônico,

$$\mathbf{M}_{B_N} = \begin{bmatrix} B_{1,1} & B_{1,2} & \dots & B_{1,N} \\ B_{2,1} & B_{2,2} & \dots & B_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ B_{N,1} & B_{N,2} & \dots & B_{N,N} \end{bmatrix}, \tag{3.29}$$

em que $B_{n,k}$ é dado por (3.18),

$$\mathbf{w}_N = \begin{bmatrix} \langle f(t), \text{sign}(\sin(\omega_0 t))\rangle \\ \langle f(t), \text{sign}(\sin(2\omega_0 t))\rangle \\ \vdots \\ \langle f(t), \text{sign}(\sin(N\omega_0 t))\rangle \end{bmatrix}, \tag{3.30}$$

e

$$\hat{\mathbf{b}}_N = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \vdots \\ \hat{b}_N \end{bmatrix}. \tag{3.31}$$

Neste trabalho, a base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ será considerada como de fato ela é: não-ortogonal. O *software* confeccionado neste trabalho considera esta base como não-ortogonal. A seguir é derivada a relação entre os coeficientes da série de Fourier clássica e os coeficientes da série de Fourier Quantizada.

3.4.1 A Relação entre os Coeficientes de Fourier Clássica e da Série de Fourier Quantizada

Sendo $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ uma base do espaço de funções que obedecem as condições da Definição 1, qualquer função nestas condições pode ser expressa como uma combinação linear dos elementos dessa nova base. Assim, tem-se que

$$f(t) \triangleq \frac{\hat{a}_0}{2} + \sum_{n=1}^{\infty} (\hat{a}_n \,\text{sign}(\cos(n\omega_0 t)) + \hat{b}_n \,\text{sign}(\sin(n\omega_0 t))), \tag{3.32}$$

em que $n \in \mathbb{N}$.

Dado os coeficientes da série de Fourier que representa uma dada função, estamos interessados em obtê-los a partir dos coeficientes da expansão em série da mesma função considerando a base proposta. Para tanto, analisa-se a seguir a relação entre os coeficientes da série de Fourier e os coeficientes da expansão em série considerando a base $\{1, \text{sign}(\cos(n\omega_0 t))$, $\text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$. Sendo $k \in \mathbb{N}$, tem-se:

1. Como primeiro caso, a relação entre os coeficientes $\hat{a}_0$, da série de Fourier Quantizada, e a_0, da série de Fourier clássica, é dada como se segue.

$$\begin{aligned} a_0 =& \langle f(t), 1\rangle \\ & \langle \frac{\hat{a}_0}{2}, 1\rangle + \sum_{n=0}^{\infty} (\hat{a}_n \langle \text{sign}(\cos(n\omega_0 t)), 1\rangle + \hat{b}_n \langle \text{sign}(\sin(n\omega_0 t)), 1\rangle), \end{aligned}$$

usando (3.16) e (3.15), obtêm-se

$$a_0 = \hat{a}_0. \tag{3.33}$$

Isso significa que, na representação da série de Fourier Quantizada, o termo responsável por representar o valor constante do sinal, representado pela função $f(t)$, não se altera em relação a série de Fourier clássica.

2. A relação entre os coeficientes $\hat{a}_k$ da série de Fourier Quantizada e a_k da série de Fourier clássica é obtida através do produto interno $\langle f(t), \cos(k\omega_0 t)\rangle$, como se segue.

$$a_k = \langle f(t), \cos(k\omega_0 t)\rangle$$
$$\sum_{n=0}^{\infty} (\hat{a}_n \langle \text{sign}(\cos(n\omega_0 t)), \cos(k\omega_0 t)\rangle + \hat{b}_n \langle \text{sign}(\sin(n\omega_0 t)), \cos(k\omega_0 t)\rangle).$$

Usando o fato de que $\text{sign}(\sin(n\omega_0 t))$ é impar e $\cos(k\omega_0 t)$ é par, os termos $\langle \text{sign}(\sin(n\omega_0 t)), \cos(k\omega_0 t)\rangle$ valem todos zero. Usando a expansão em série de Fourier de $\text{sign}(\cos(n\omega_0 t))$, tem-se que

$$a_k = \sum_{n=0}^{\infty} \hat{a}_n \langle \sum_{l=1}^{\infty} \alpha_l \cos((2l+1)n\omega_0 t), \cos(k\omega_0 t)\rangle$$
$$\sum_{n=0}^{\infty} \hat{a}_n \sum_{l=1}^{\infty} \alpha_l \langle \cos((2l+1)n\omega_0 t), \cos(k\omega_0 t)\rangle.$$

Usando (2.14), nota-se que toda vez que $(2l+1)n = k$, o produto interno $\langle \cos((2l+1)\omega_0 t), \cos(k\omega_0 t)\rangle$ resulta em 1, e 0 caso contrário, logo

$$a_k = \sum_{n|k} \hat{a}_n \alpha_{(\frac{k}{n}-1)/2}. \tag{3.34}$$

3. Por último, de modo semelhante ao que foi feito para os coeficientes $\hat{a}_k$ e a_k, a relação entre os coeficientes $\hat{b}_k$ da série de Fourier Quantizada e b_k série de Fourier clássica é obtida através do produto interno $\langle f(t), \sin(k\omega_0 t)\rangle$, como se segue.

$$b_k = \langle f(t), \sin(k\omega_0 t)\rangle$$
$$\sum_{n=0}^{\infty} (\hat{a}_n \langle \text{sign}(\cos(n\omega_0 t)), \sin(k\omega_0 t)\rangle + \hat{b}_n \langle \text{sign}(\sin(n\omega_0 t)), \sin(k\omega_0 t)\rangle), \tag{3.35}$$

usando o fato de que $\text{sign}(\cos(n\omega_0 t))$ é par e $\sin(k\omega_0 t)$ é impar, os termos $\langle \text{sign}(\cos(n\omega_0 t)), \sin(k\omega_0 t)\rangle$ valem todos zero. Usando a expansão em série de Fourier de $\text{sign}(\sin(n\omega_0 t))$, tem-se que

$$b_k = \sum_{n=0}^{\infty} \hat{b}_n \langle \sum_{l=1}^{\infty} \beta_l \sin((2l+1)n\omega_0 t), \sin(k\omega_0 t)\rangle$$
$$\sum_{n=0}^{\infty} \hat{b}_n \sum_{l=1}^{\infty} \beta_l \langle \sin((2l+1)n\omega_0 t), \sin(k\omega_0 t)\rangle.$$

Usando (2.14), nota-se que toda vez que $(2l+1) = k$, o produto interno $\langle \sin((2l+1)\omega_0 t), \sin(k\omega_0 t)\rangle$ resulta em 1, e 0 caso contrário, logo

$$b_k = \sum_{n|k}^{\infty} \hat{b}_n \beta_{(\frac{k}{n}-1)/2}. \tag{3.36}$$

É obtido assim uma relação entre cada a_n e $\hat{a}_n$ e entre cada b_n e $\hat{b}_n$ a partir de cada α_l e β_l, respectivamente.

CAPÍTULO 4

Simulação Computacional

Neste capítulo são descritas algumas simulações computacionais no que diz respeito a representação de sinais por meio da série de Fourier Quantizada e a série de Fourier obtida a partir da série de Fourier Quantizada.

4.1 Algoritmo para Obtenção de $\hat{\mathbf{a}}_N$ e $\hat{\mathbf{b}}_N$

O desenvolvimento teórico-matemático feito no Capítulo 3 leva, naturalmente, a um algoritmo ou conjunto de passos finito que pode orientar a análise do sinal. Este algoritmo está exposto no Algoritmo 1.

Algoritmo 1 Algoritmo de obtenção dos coeficientes da série de Fourier quantizada.

Entrada: O sinal $f(t)$ e número de harmônicos N
Saída: Vetores $\hat{\mathbf{b}}_N$ e $\hat{\mathbf{a}}_N$ contendo os coeficientes da série de Fourier quantizada $\hat{a}_n$ e $\hat{b}_n$ para $0 \leq n \leq N$

Gerar $\mathbf{v}_N$ e $\mathbf{w}_N$
Gerar $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$
Resolver os sistemas $\mathbf{M}_{A_N} \cdot \hat{\mathbf{a}}_N = \mathbf{v}_N$ e $\mathbf{M}_{B_N} \cdot \hat{\mathbf{b}}_N = \mathbf{w}_N$
Retorna $\hat{\mathbf{a}}_N$ e $\hat{\mathbf{b}}_N$

O Algoritmo 1 necessita da geração das matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$. Isso pode ser facilmente implementado seguindo (3.25) e (3.29). Além disso, a função chi-fração pode ser implementada seguindo diretamente a Definição 6.

A linguagem usada para simulação computacional foi a linguagem da plataforma Matlab®. O método usado para resolver os sistemas em (3.24) e (3.28) foi o método de eliminação de Gauss [24]. Este método foi escolhido por alcançar uma alta acurácia computacional quando comparado com outros métodos diretos existentes [24]. Obviamente, quando se trata de um sistema de tempo real, deve-se explorar as propriedades das matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$, que neste momento não são relevantes, de modo a se reduzir o número de operações necessárias para se encontrar a solução do sistema, reduzindo tempo de computação.

4.2 Simulação para Obtenção de $\hat{\mathbf{a}}_N$ e $\hat{\mathbf{b}}_N$

Sinais com forma de onda de rampa e parábola foram usados para simulação computacional para a obtenção dos vetores coluna $\hat{\mathbf{a}}_N$ e $\hat{\mathbf{b}}_N$.

Pode-se observar na Figura 4.1 um exemplo computacional para uma função rampa, em que $f(t) = t$ para $t \in [0,1]$. Na Figura 4.2, pode-se observar o mesmo para uma parábola, em que $f(t) = t^2$ para $t \in [0,1]$. Em ambas as simulações foram usados sinais com mil amostras.

4.3 Simulação para Obtenção de a_n e b_n a partir de $\mathbf{a}_N$ e $\mathbf{b}_N$

Após obtidos os coeficientes $\hat{a}_n$ e $\hat{b}_n$, pode-se usar (3.34) e (3.36) de modo a obter uma aproximação para a_n e b_n. Observe que a_0 é obtido de forma exata, sem aproximações.

Procedendo desta forma simples, obtêm-se estimativas para os coeficientes de Fourier. A Figura 4.3 exibe a representação da função rampa $f(t) = t$ para $t \in [0,1]$ em série de Fourier, obtida a partir da série de Fourier Quantizada considerando a base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}$. O mesmo é feito para a parábola $f(t) = t^2$ para $t \in [0,1]$ na Figura 4.4.

Seja $f(t)$ um sinal em que se esteja interessado em analisar a acurácia da sua representação por meio da série de Fourier Quantizada e da série de Fourier clássica obtida a partir da série de Fourier Quantizada. Por questão de notação, será definido como série de Fourier Aproximada a série de Fourier clássica obtida a partir da série de Fourier Quantizada.

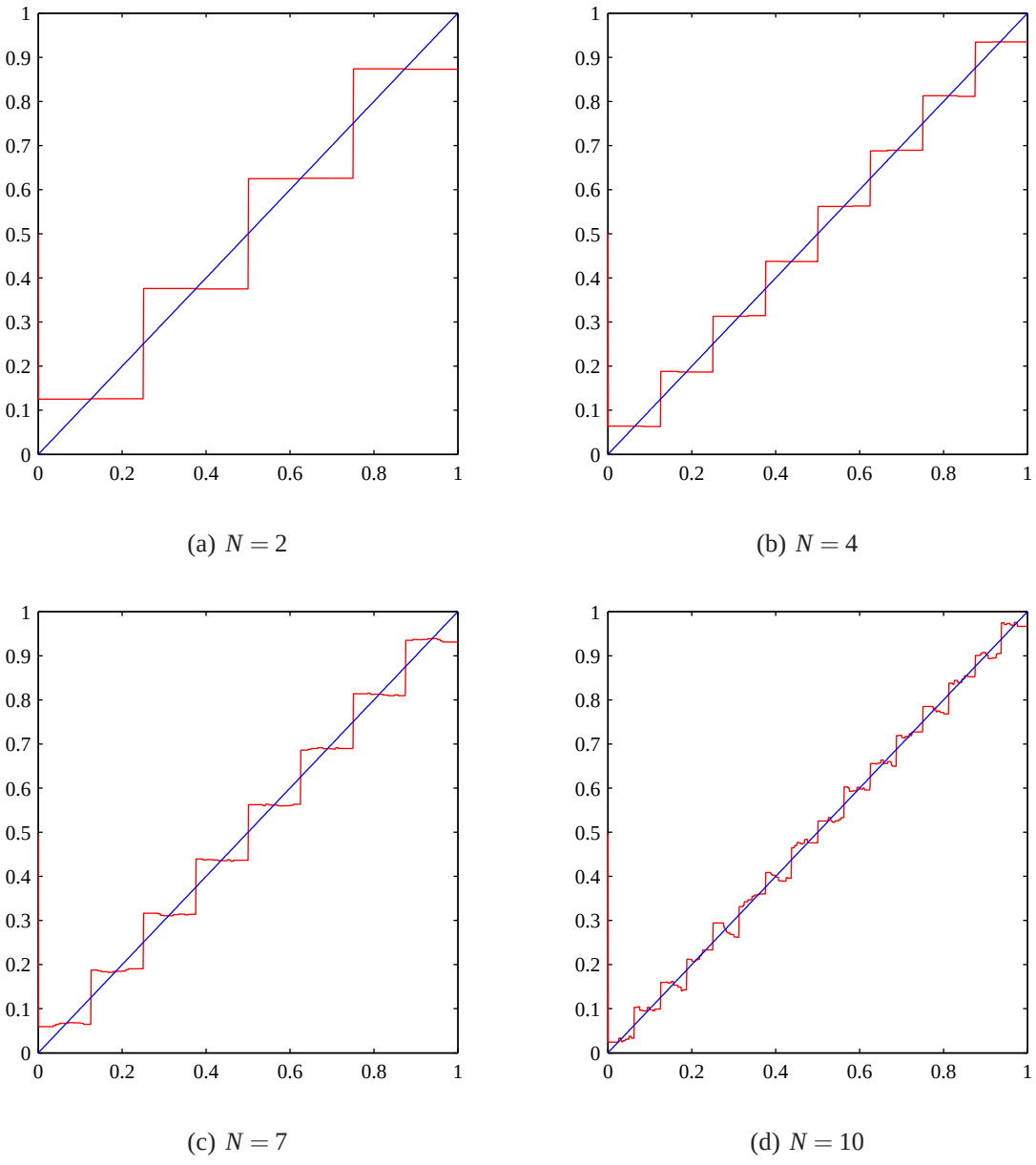
Uma forma de se medir o quão próximos dois sinais são um do outro é por avaliar o erro quadrático médio entre eles [24]. Considerando dois sinais com L amostras, cada, uma possível forma de se expressar o *MSE* entre eles é dada por

$$\text{MSE} \triangleq \sum_{k=1}^{L} r_k^2,$$

em que r_k representa a diferença entre as k-ésimas amostras dos sinais avaliados.

Considerando a série de Fourier Aproximada, é definido como MSE_{Aprox} o *MSE* entre a série de Fourier Aproximada de um dado sinal e sua série de Fourier clássica obtida do modo tradicional. Além disso, considerando a série de Fourier Quantizada, é definido como MSE_{Quant} o *MSE* entre a série de Fourier Quantizada e a série de Fourier clássica obtida do modo tradicional.

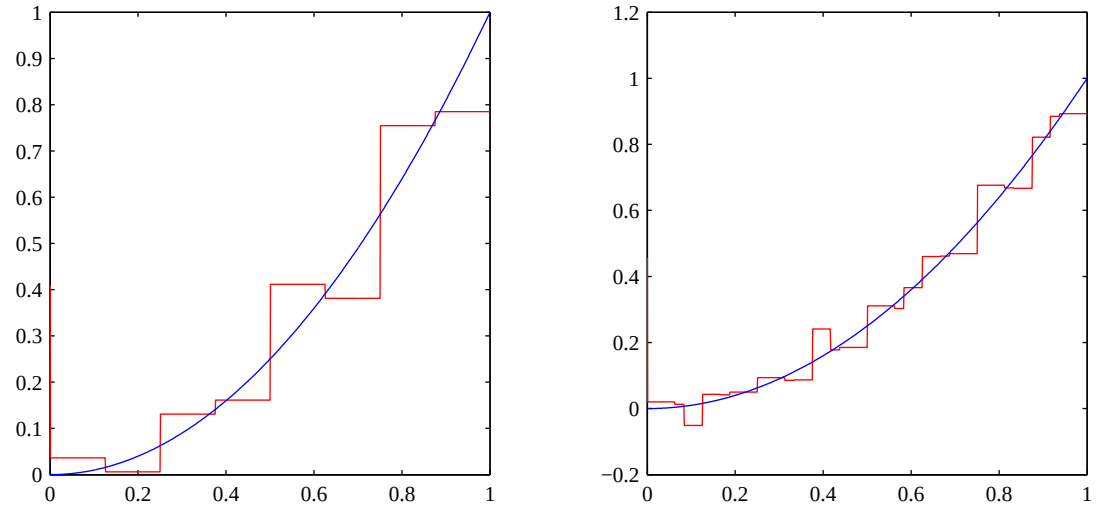
As Tabelas 4.1, 4.2, 4.3 e 4.4 exibem alguns valores de erros quadráticos médio (*MSE*) da representação em série de Fourier Quantizada e série de Fourier Aproximada a partir da série de Fourier Quantizada de sinais em forma de rampa, parábola, cosseno e gaussiana em relação a série de Fourier clássica obtida da forma tradicional.



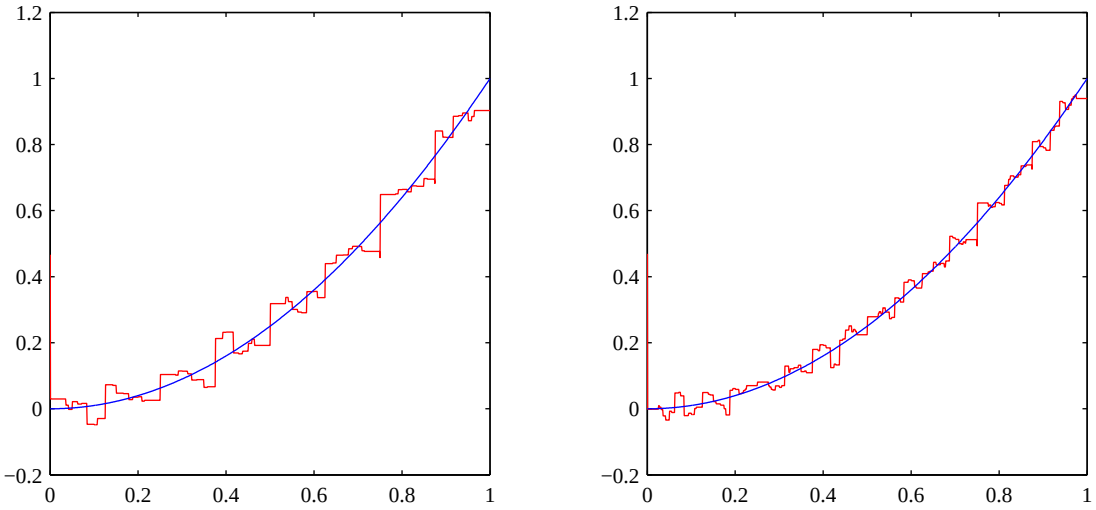
(a) $N = 2$ (b) $N = 4$

(c) $N = 7$ (d) $N = 10$

Figura 4.1 Representação da expansão de $f(t) = t$ para $t \in [0,1]$ na base da série de Fourier Quantizada com 2, 4, 7 e 10 harmônicos, respectivamente.

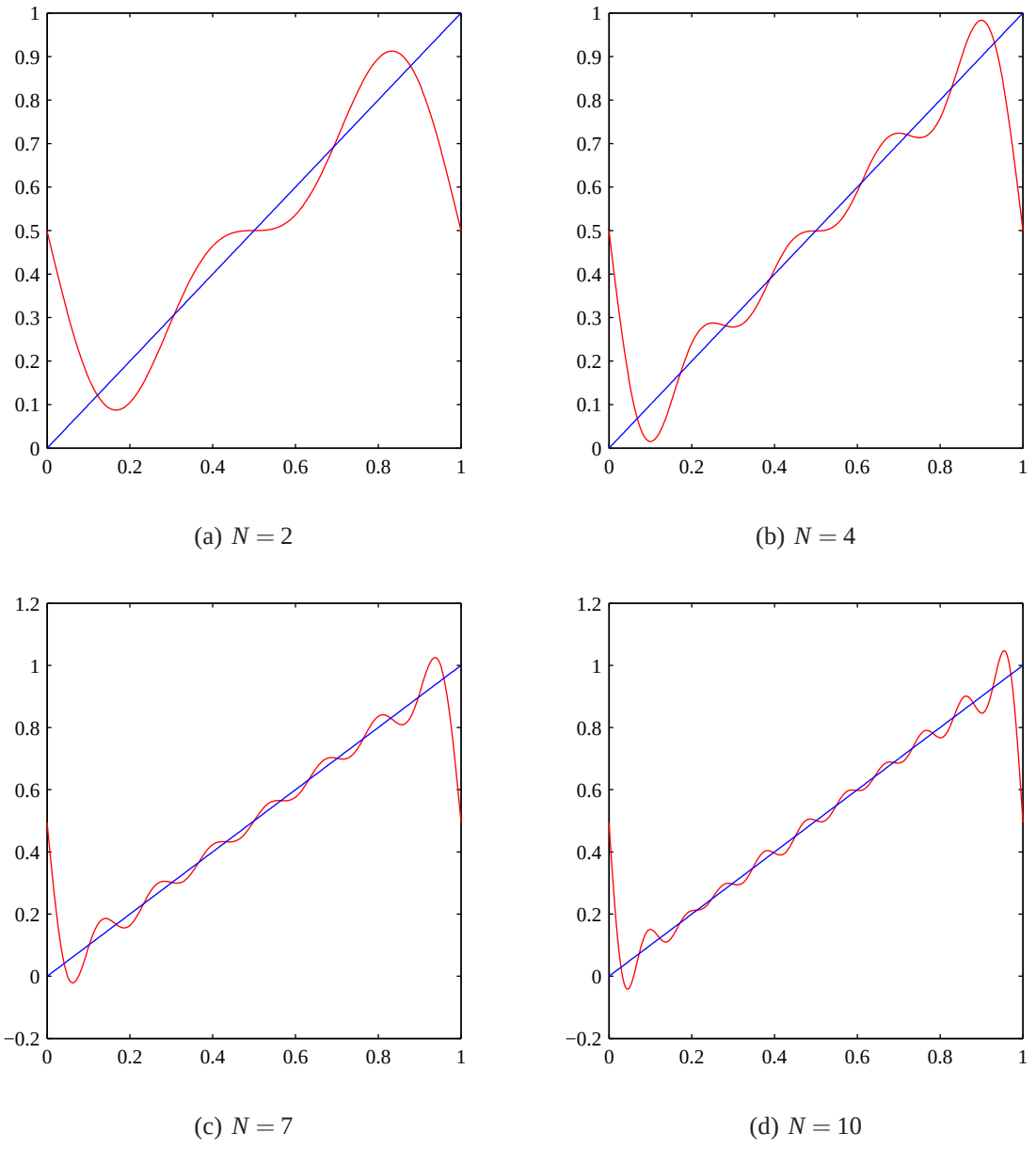


(a) $N = 2$ (b) $N = 4$



(c) $N = 7$ (d) $N = 10$

Figura 4.2 Representação da expansão de $f(t) = t^2$ para $t \in [0,1]$ na base da série de Fourier Quantizada com 2, 4, 7 e 10 harmônicos, respectivamente.



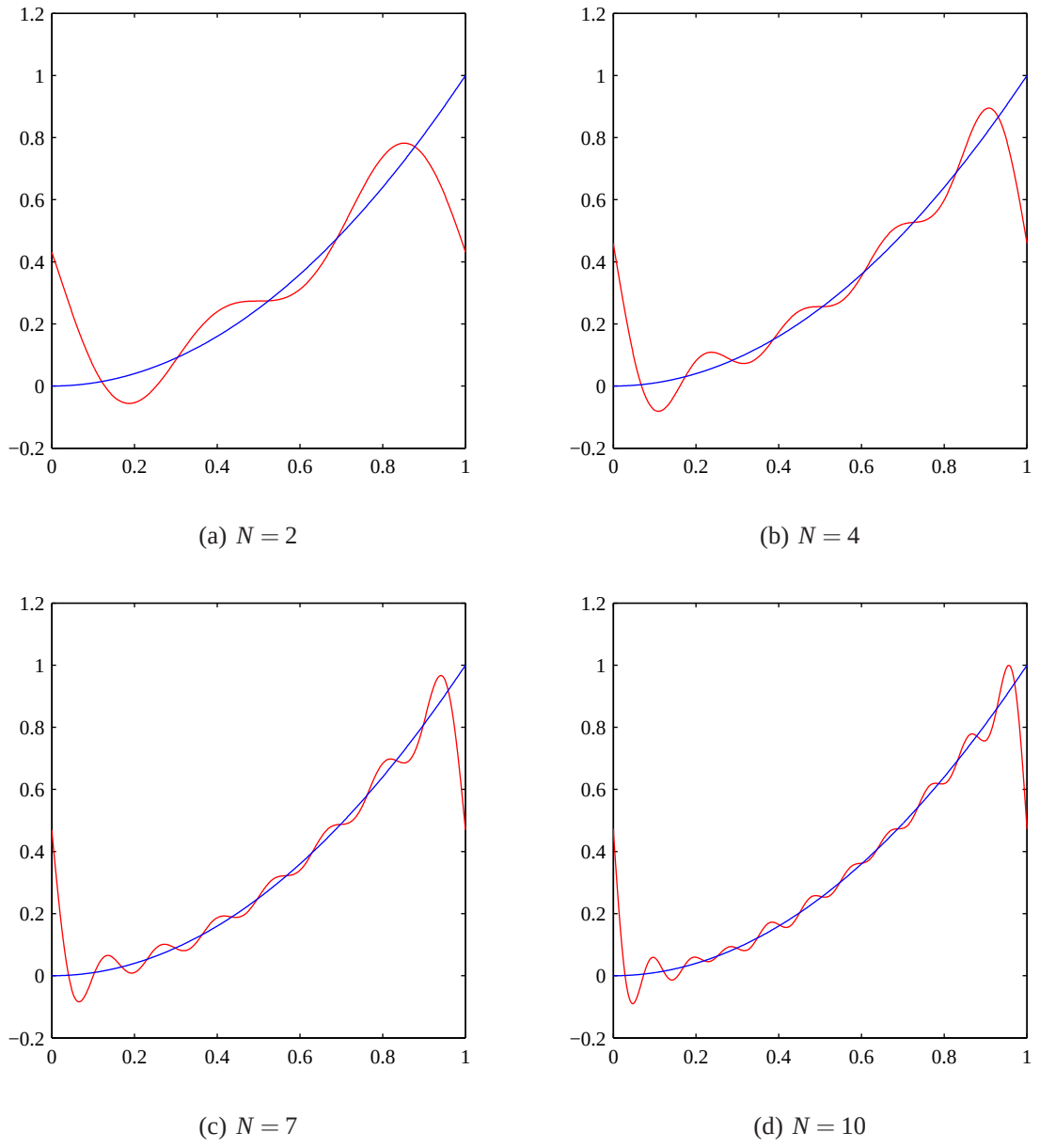
(a) $N = 2$

(b) $N = 4$

(c) $N = 7$

(d) $N = 10$

Figura 4.3 Representação da expansão de $f(t) = t$ para $t \in [0,1]$ em série de Fourier Aproximada com 2, 4, 7 e 10 harmônicos, respectivamente, obtida a partir da série de Fourier Quantizada.



(a) $N = 2$ (b) $N = 4$

(c) $N = 7$ (d) $N = 10$

Figura 4.4 Representação da expansão de $f(t) = t^2$ para $t \in [0,1]$ em série de Fourier Aproximada com 2, 7, 4 e 10 harmônicos, respectivamente, obtida a partir da série de Fourier Quantizada.

Tabela 4.1 MSE_{Quant} e MSE_{Aprox} para o sinal $f(t) = t$ para $t \in [0,1]$ com mil amostras

N	MSE_{Quant}	MSE_{Aprox}
2	$3.8340 \cdot 10^{-3}$	$2.8435 \cdot 10^{-5}$
4	$3.1339 \cdot 10^{-3}$	$3.9085 \cdot 10^{-5}$
8	$2.3753 \cdot 10^{-3}$	$1.7027 \cdot 10^{-4}$
10	$2.1346 \cdot 10^{-3}$	$2.2846 \cdot 10^{-4}$
16	$1.7827 \cdot 10^{-3}$	$3.0531 \cdot 10^{-4}$
32	$1.3732 \cdot 10^{-3}$	$4.5591 \cdot 10^{-4}$

Tabela 4.2 MSE_{Quant} e MSE_{Aprox} para o sinal $f(t) = t^2$ para $t \in [0,1]$ com mil amostras

N	MSE_{Quant}	MSE_{Aprox}
2	$3.9676 \cdot 10^{-3}$	$5.1896 \cdot 10^{-4}$
4	$3.217 \cdot 10^{-3}$	$2.4718 \cdot 10^{-4}$
8	$2.4204 \cdot 10^{-3}$	$2.0383 \cdot 10^{-4}$
10	$2.1779 \cdot 10^{-3}$	$2.1714 \cdot 10^{-4}$
16	$1.798 \cdot 10^{-3}$	$2.559 \cdot 10^{-4}$
32	$1.3546 \cdot 10^{-3}$	$3.6494 \cdot 10^{-4}$

Tabela 4.3 MSE_{Quant} e MSE_{Aprox} para o sinal $f(t) = \cos(2\pi t)$ para $t \in [0,1]$ com mil amostras

N	MSE_{Quant}	MSE_{Aprox}
2	$9.7292 \cdot 10^{-3}$	$4.2361 \cdot 10^{-3}$
4	$6.6242 \cdot 10^{-3}$	$1.9589 \cdot 10^{-3}$
8	$4.3533 \cdot 10^{-3}$	$1.1433 \cdot 10^{-3}$
10	$4.3606 \cdot 10^{-3}$	$1.168 \cdot 10^{-3}$
16	$3.3324 \cdot 10^{-3}$	$1.1174 \cdot 10^{-3}$
32	$2.5203 \cdot 10^{-3}$	$1.2268 \cdot 10^{-3}$

Tabela 4.4 MSE_{Quant} e MSE_{Aprox} para o sinal $f(t) = e^{-t^2}$ para $t \in [0,1]$ com mil amostras

N	MSE_{Quant}	MSE_{Aprox}
2	$2.6937 \cdot 10^{-3}$	$2.275 \cdot 10^{-4}$
4	$2.1523 \cdot 10^{-3}$	$1.0785 \cdot 10^{-4}$
8	$1.5874 \cdot 10^{-3}$	$2.5101 \cdot 10^{-4}$
10	$1.435 \cdot 10^{-3}$	$2.8744 \cdot 10^{-4}$
16	$1.1992 \cdot 10^{-3}$	$4.0157 \cdot 10^{-4}$
32	$1.0242 \cdot 10^{-3}$	$6.2589 \cdot 10^{-4}$

CAPÍTULO 5

Técnicas de Implementação

Este capítulo descreve algumas técnicas de implementação usadas durante a implementação do Analisador de Espectro via *software*.

Em alguns cenários, as sequências de entradas são passadas em tempo real. O sistema que realiza a análise não tem acesso a completa sequência de entrada ao iniciar a análise. Tal disponibilidade implicaria em um aumento do tempo total de processamento, visto que a aplicação necessitaria aguardar o término de toda a sequência de entrada para então iniciar o processamento. Além disso, restrições de capacidade de memória, forçam o projetista a estruturar o sistema de tal modo que este seja capaz de lidar com trechos da sequência de entrada, a medida que esta é recebida, e então simultaneamente fornecer um prévio resultado do processamento do sinal até então recebido. Não raro, tal tipo de abordagem introduz alguns resultados indesejados no sinal de saída, como descontinuidades.

O Analisador de Espectro projetado e implementado neste trabalho de graduação assume que o todo o sinal de entrada já está disponível para análise, contudo, será tratado aqui o processamento de um sinal com um número alto de amostras por quebra em blocos. Tal abordagem possibilitará futuros acréscimos de funcionalidade do Analisador de Espectro, por exemplo, processamento em tempo real com *stream* de dados.

5.1 Processamento em Blocos de Curto Comprimento

Tal tipo de abordagem não se limita a situações em que o sistema não tem acesso ao sinal completo para processamento ou limitação em memória. Aplicações em que a quantidade de amostras de uma dada sequência que deve ser analisada é alta pode usufruir do processamento por parte. Este tipo de processamento é especialmente útil em situações em que as máquinas computacionais usadas são dotadas de múltiplos processadores. Em tais circunstâncias, o processamento de trechos da sequência original pode ser feito com o uso de paralelismo [21, 27]. Mesmo em situações que as máquinas computacionais encarregadas do processamento não sejam dotadas de mais de um processador, essa abordagem ainda é útil. De fato, lidar com sequência de pequeno comprimento, do ponto de vista dos computadores digitais, é bem mais simples, tendo em vista os problemas oriundos com uma grande massa de dados carregada simultaneamente em memória [33, 34].

As funções **smallblocklengthsf**, **smallblocklengthsignal** e **smallblocklengthsignalsf** implementam a metodologia do processamento em blocos de curto comprimento. Os seu códigos fontes podem ser vistos no Apêndice B. A função **smallblocklengthsf** implementa a obtenção da representação em série de Fourier clássica de um dado sinal de entrada. A função **small-**

blocklengthsignal desempenha um papel semelhante, porém o fazendo para a série de Fourier Quantizada. A última função, **smallblocklengthsignalsf**, implementa a obtenção da representação de um dado sinal de entrada em série de Fourier Aproximada, isto é, a série de Fourier clássica obtida a partir da série de Fourier Quantizada.

A técnica de processamento por blocos de comprimento curto pode ainda ser combinada com uma outra técnica baseada nas características do processo de obtenção dos coeficientes da série de Fourier Quantizada e obtenção dos coeficientes da série de Fourier a partir da série de Fourier Quantizada. Tal técnica se baseia no cálculo a priori de quantidades previamente conhecidas e que são independentes do sinal de entrada. Além disso, ao se aplicar a filosofia do cálculo a priori, é buscada a minimização das redundâncias computacionais. Tal estratégia, que já é bem conhecida pela comunidade de processamento de sinais [28], é abordada na seção seguinte.

5.2 Cálculo a priori e Minimização de Redundâncias

Comumente, é de interesse que o sistema seja capaz de processar o maior número de possível de dados de forma econômica. Para tanto, usualmente, são procuradas redundâncias computacionais durante a fase de processamento dos dados, de modo que estas possam ser agrupada e minimizadas ao máximo, e se possível, eliminadas. Esta é a técnica de cálculo a priori de quantidades já conhecidas, independentes do sinal de entrada. De fato, esta é uma técnica bem conhecida e largamente usada pela comunidade de processamento de sinais. Tal abordagem levou naturalmente ao surgimento dos chamados algoritmos rápidos [8, 28].

Algoritmos rápidos nada mais são do que algoritmos que eliminam as redundâncias computacionais durante o cálculo de determinada quantidade. Usualmente, algoritmos rápidos são aplicados no cálculo de transformadas discretas, como a transformada discreta de Fourier (DFT), do cosseno (DCT) e de Hartley (DHT) [8]. Tais técnicas podem ser aplicadas na obtenção da série de Fourier clássica a partir da série de Fourier Quantizada como descrito a seguir.

Seja $\alpha_{\mathbf{k}}$ o vetor de todos os índices $\alpha_{(\frac{k}{n}-1)/2}$, como descrito em (3.34) e $\beta_{\mathbf{k}}$ o vetor de todos os índices $\beta_{(\frac{k}{n}-1)/2}$, como descrito em (3.36), em que n é um divisor de k. É fácil observar que os coeficientes a_k e b_k da série de Fourier podem ser obtidos como resultado do produto interno entre os vetores $\alpha_{\mathbf{k}}$ e $\beta_{\mathbf{k}}$ e os vetores formados pelos coeficientes $\hat{a}_n$ e $\hat{b}_n$ em que n é um divisor de k. Além disso, os vetores $\alpha_{\mathbf{k}}$ e $\beta_{\mathbf{k}}$ são independentes do sinal de entrada, o que permite assim que estes sejam calculados a priori. Além disso, é notório que os coeficientes $\alpha_{\mathbf{k}}$ e $\beta_{\mathbf{k}}$, em que n é um divisor de k, assumem uma forma que torna fácil o seu cômputo, como se segue. Substituindo $l = (k/n-1)/2$ em (3.2) e (3.4), obtêm-se:

$$\alpha_{(\frac{k}{n}-1)/2} = \frac{4n}{k\pi}(-1)^{(\frac{k}{n}-1)/2}, \tag{5.1}$$

e

$$\beta_{(\frac{k}{n}-1)/2} = \frac{4n}{k\pi}, \tag{5.2}$$

em que n é um divisor de k.

Além disso, pode-se observar que para todo $l \in \mathbb{N}$

$$\alpha_l = \beta_l \cdot (-1)^l. \tag{5.3}$$

Tal relação é muito bem-vinda, tendo em vista que esta permite "enxugar"o número de operações durante a fase de preparação para o processamento. Contudo, tal ação de encontrar os coeficientes da série de Fourier a partir da série de Fourier Quantizada, só é possível tendo conhecimento destes. Para tanto, antes de se realizar a operação mencionada, é preciso resolver os sistemas constantes em (3.24) e (3.28). A seção seguinte trata do problema de dada as matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$ e os produtos internos em (3.26) e (3.30), como encontrar de modo rápido e econômico, do ponto de vista computacional, os coeficientes $\hat{a}_n$ e $\hat{b}_n$.

Em certos contextos, além do interesse de detecção, há a intenção de quantificar exatamente o que está acontecendo com o sinal analisado no espectro da frequência. Em muitos outros momentos, a detecção nada mais é do que uma busca em primeira aproximação, cujo objetivo é analisar se o sinal tem ou não determinado comportamento no espaço frequencial. Sendo detectado algum harmônico que sugira que o sinal tem tal comportamento de interesse no espaço frequencial, é em seguida feita uma análise mais acurada, sendo assim computados os coeficientes da série de Fourier clássica com base na série de Fourier Quantizada. Um modo de computar os coeficientes da série de Fourier partindo dos coeficientes da série de Fourier Quantizada é usando o método de Dempster-McLeod [14].

5.3 Método de Dempster-McLeod

Usualmente é de interesse detectar a presença de harmônicos em um dado sinal de entrada. A depender da aplicação, não há necessidade de se tomar conhecimento exato de quanto de energia cada harmônico transporta. Em detrimento da precisão, pode-se obter consideráveis ganhos na quantidade de dados que podem ser processados no mesmo espaço de tempo em relação ao cálculo usual dos coeficientes da série de Fourier clássica. Uma das formas de se alcançar isso é eliminando as operações de multiplicação por números flutuantes.

Os coeficientes das matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$ são os $A_{n,k}$ e $B_{n,k}$ descritos em (3.17) e (3.18). Os coeficientes $A_{n,k}$ e $B_{n,k}$, a menos de multiplicações por escalares, são funções única e exclusivamente da função chi-fração. Visto que os únicos parâmetros da função chi-fração são apenas os índices n e k, conclui-se que os coeficientes são independentes da entrada, o que faz com que as matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$ também o sejam.

No Algoritmo 1, na fase de simulação, foi usado o método de eliminação de Gauss [24] para resolver os sistemas (3.24) e (3.28). Ao aplicar o método de Gauss, são necessárias $N(N-1)$ multiplicações e N divisões [24], além das somas e subtrações. Esta ainda não é a forma mais rápida para se resolver os sistemas em questão.

Dentre diversos métodos existentes, como decomposição LU e QR das matrizes dos coeficientes do sistema [15] e de Choleski [26], por ser o método que apresenta maior acurácia computacional, o método de eliminação de Gauss foi escolhido para resolver os sistemas em (3.24) e (3.28) na fase de simulação computacional, como foi descrito no Capítulo 4. Embora

essa seja uma abordagem eficaz e possível para se obter uma estimativa dos coeficientes da série Fourier, esta não é a mais eficiente de todas. Uma propriedade das matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$ que pode ser extremamente útil é que não são dependentes do sinal de entrada.

As matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$ serem independentes da entrada tem um forte impacto na forma como os coeficientes da série de Fourier Quantizada, $\hat{a}_n$ e $\hat{b}_n$, e em consequência os coeficientes de Fourier, são obtidos. Considerando tal propriedade, pode-se calcular a priori as matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ e então usá-las par obter os coeficientes da série de Fourier Quantizada e da série de Fourier clássica. A princípio, pode-se questionar que esta forma de se resolver o sistema, sendo já dadas as matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$, aumente o número de multiplicações de $N(N-1)$ para N^2, além da complexidade computacional naturalmente oriunda do processo de inversão matricial. Embora isso seja verdade, esta forma de se resolver o sistema, calculando a matriz inversa a priori, elimina as N divisões. De fato, que está se realizando é uma troca de N divisões por N multiplicações. Além disso, a complexidade do processo de inversão matricial não é problema para aplicações que exigem uma grande massa cálculos, pois, como exposto, as matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ podem ser calculadas antes de se iniciar o processamento dos dados, e então armazenadas para serem utilizadas quando necessário. Toda a complexidade computacional no momento do processamento se limita as N^2 multiplicações. Não obstante, considerando a constituição das matrizes $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$, nota-se que o número de multiplicações não é exatamente N^2. As matrizes $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ apresentam um pequeno grau de esparsidade, que é favorável a redução da complexidade computacional do processo de obtenção dos coeficientes da série de Fourier Quantizada, $\hat{a}_n$ e $\hat{b}_n$, em função dos produtos internos $\langle f(t), \text{sign}(\cos(n\omega_0 t))\rangle$ e $\langle f(t), \text{sign}(\sin(n\omega_0 t))\rangle$. As equações (5.4), (5.6), (5.5) e (5.7) exibem as matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ para $N = 3$ e 5, respectivamente.

$$MA_3^{-1} = \begin{bmatrix} \frac{9}{8} & 0 & \frac{3}{8} \\ 0 & 1 & 0 \\ \frac{3}{8} & 0 & \frac{9}{8} \end{bmatrix} \tag{5.4}$$

$$MB_3^{-1} = \begin{bmatrix} \frac{9}{8} & 0 & \frac{-3}{8} \\ 0 & 1 & 0 \\ \frac{-3}{8} & 0 & \frac{9}{8} \end{bmatrix} \tag{5.5}$$

$$MA_5^{-1} = \begin{bmatrix} \frac{7}{6} & 0 & \frac{3}{8} & 0 & \frac{-5}{24} \\ 0 & 1 & 0 & 0 & 0 \\ \frac{3}{8} & 0 & \frac{9}{8} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{-5}{24} & 0 & 0 & 0 & \frac{25}{24} \end{bmatrix} \tag{5.6}$$

$$MB_5^{-1} = \begin{bmatrix} \frac{7}{6} & 0 & \frac{-3}{8} & 0 & \frac{-5}{24} \\ 0 & 1 & 0 & 0 & 0 \\ \frac{-3}{8} & 0 & \frac{9}{8} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{-5}{24} & 0 & 0 & 0 & \frac{25}{24} \end{bmatrix} \tag{5.7}$$

O método denominado de método de Dempster-McLeod foi introduzido em [14]. O método consiste em implementar a multiplicação de números em ponto flutuante por constantes inteiras de modo econômico do ponto de vista computacional. No problema apresentado neste trabalho, as constantes não são de natureza inteira, fazendo-se assim necessário uma pequena adaptação.

Neste momento é importante notar que os coeficientes de uma matrizes inversa de uma dada matriz pode ser obtido por uma composição de sucessivas operações de soma, subtração, multiplicação e divisão entre os elementos e cofatores da matriz original [9, 15]. Considerando as matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$ com base nas equações (3.25) e (3.29), e em (3.17) e (3.18), todos os seus coeficientes são números racionais, logo, os coeficientes das matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ também o são.

Dado que os coeficientes de $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ são números racionais, estes podem ser exatamente representados com um número finito de *bits*. Em geral, tal representação, quando usada no cenário computacional para a obtenção dos coeficientes da série de Fourier Quantizada, incorrerá na necessidade de divisões por números inteiros quaisquer. Certamente, isso demanda bastante recurso computacional.

Com o objetivo de se representar do melhor modo possível as matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ podemos aproximar seus elementos por números ádicos racionais da forma $m/2^n$, em que $0 \leq n \leq N$, $m \in \mathbb{Z}$ [10] e N representa o número de *bits* usados. Tal aproximação é vantajosa do ponto de vista computacional, pois a divisão que seria realizada tendo como divisor uma quantidade qualquer em ponto flutuante, que certamente é computacionalmente custoso, é substituída por uma divisão de um inteiro em potência de 2, que pode ser implementada simplesmente por um deslocamento de bits.

Tal tipo de abordagem para se reduzir o tempo de computação deve ser adequado para cenários em que a implementação da estimativa harmônica através da série de Fourier Aproximada esteja sendo feita em nível de *hardware*. Obviamente, se a implementação estiver sendo executada em nível de *software*, tal abordagem pode ser bem vinda, desde que o computador digital usado suporte a divisão por inteiros que são potência de 2 por meio de deslocamento de bits. A título ilustrativo, algumas aplicações em que isso pode acontecer são aplicações nas quais o *hardware* usado são microcontroladores, DSP's (Digital Signal Processor) ou FPGA's (Field-Programmable Gate Array), dispositivos nos quais os projetistas podem ter pleno controle sobre cada *bit*.

CAPÍTULO 6

Otimização Computacional no Contexto de Detecção

O capítulo anterior tratou de assuntos como cálculo a priori de algumas quantidades e minimização de redundâncias computacionais. Neste capítulo é tratada mais algumas otimizações e simplificações no cálculo de estimação dos harmônicos da série de Fourier em contextos em que não é necessário a computação exata dos coeficientes de Fourier, mas sim, da relação entre eles. Além disso, neste capítulo é tratado como obter os coeficientes da série de Fourier a partir da estimação deles no contexto de detecção.

6.1 Método do Fator de Escalonamento I

A partir do cálculo a priori das matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ é possível encontrar os coeficientes da série de Fourier Quantizada, dado que foram computados os produtos internos presente em (3.26) e (3.30). Não obstante, o cômputo dos produtos matriz-vetor presentes em (3.24) e (3.28) pode ser demasiadamente custoso e desnecessário no sentido computacional em cenários em que não há interesse em calcular exatamente os coeficientes de Fourier. O motivo disso pode ser claramente visto observando as matrizes em (5.4), (5.6), (5.5) e (5.7), em que facilmente percebe-se a representação em ponto flutuante dos coeficientes das matrizes. Uma forma de se solucionar este problema é procurar uma forma de substituir as matrizes $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ por duas outras matrizes tais que as proporções entre os coeficientes $\hat{a}_n$ e $\hat{b}_n$ se preservem.

As máquinas computacionais são capazes apenas de representar quantidades finitas. Claramente, os coeficientes das matrizes $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ são representados por números racionais. Para evitar o problema da multiplicação por números em ponto flutuante, pode-se definir um fator de escalonamento ρ como sendo o menor inteiro capaz de, quando multiplicado por $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$, gerar matrizes $\rho \cdot \mathbf{M}_{A_N}^{-1}$ e $\rho \cdot \mathbf{M}_{B_N}^{-1}$ com coeficientes apenas no conjunto dos inteiros. Este método é conhecido como método do fator de escalonamento.

Para o processamento em tempo real, e aproveitamento da eliminação de multiplicação por números flutuantes, as inversas escalonadas $\rho \cdot \mathbf{M}_{A_N}^{-1}$ e $\rho \cdot \mathbf{M}_{B_N}^{-1}$ devem ser calculadas a priori, e então usadas no momento da análise. Uma rotina chamada **scaleFactor** foi criada com o objetivo de encontrar coeficientes com a menor magnitude possível, de modo que os coeficientes de $\rho \cdot \mathbf{M}_{A_N}^{-1}$ e $\rho \cdot \mathbf{M}_{B_N}^{-1}$ se tornassem inteiros, ou se aproximasse de um tão bem quanto desejado. O código fonte da função **scaleFactor** pode ser visto na Apêndice A.

Simulações computacionais foram realizadas com as matrizes para alguns números de harmônicos da série de Fourier Quantizada. A Tabela 6.1 exibe os coeficientes de escalona-

Tabela 6.1 Erro decorrente da aplicação do método do fator de escalonamento

N	α	$MSE_{\mathbf{M}_{A_N}}$	$MSE_{\mathbf{M}_{B_N}}$
2	1	0.0000	0.0000
3	3	0.0000	0.0000
4	3	0.0000	0.0000
5	15	0.0000	0.0000
6	15	0.0000	0.0000
7	105	0.0000	0.0000
8	105	0.0000	0.0000
9	315	0.0000	0.0000
10	315	0.0000	0.0000
12	3465	0.0000	0.0000
16	45045	$6.2804 \cdot 10^{-16}$	$6.2804 \cdot 10^{-16}$

Tabela 6.2 Quantidades que devem ser calculadas a priori

Tipo	Quantidade
Matrizes	$\mathbf{M}_{A_N}^{-1}$, $\mathbf{M}_{B_N}^{-1}$ $\rho \cdot \mathbf{M}_{A_N}^{-1}$, $\rho \cdot \mathbf{M}_{B_N}^{-1}$
Coeficientes	α_l, β_l

mento encontrados para estes números de harmônicos e os respectivos erros quadrático médio (*MSE*) [24]. Note que os valores de erro quadrático médio que são endereçados como zero na Tabela 6.1 significam que a máquina computacional usada não suporta uma precisão tão pequena a ponto de capturar a magnitude do erro quadrático médio. A versão do Matlab® usada possui precisão de até 2.220410^{-16}.

Com respeito a minimização de redundâncias e cálculo a priori de algumas quantidades, a Tabela 6.2 exibe exemplos de quantidades que devem ser calculadas a priori de modo a se reduzir o número de operações realizadas durante o processamento.

Embora o escalonamento das matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ seja útil e reduza a complexidade das operações em contextos de detecção, ainda pode-se otimizar a forma como os coeficientes a_k e b_k da série de Fourier são encontrados a partir de $\hat{a}_n$ e $\hat{b}_n$. Mais uma vez, não estamos interessados em computar exatamente apenas os coeficientes a_k e b_k, mas sim, preservar as relações entre eles e buscar otimizar as operações segundo o relaxamento deste critério. A seguir é dada uma adaptação do método do fator de escalonamento para os coeficientes $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$.

6.2 Método do Fator de Escalonamento II

Por meio de (3.2) e (3.4) pode-se observar que α_l e β_l podem ser calculados a priori. Além disso, quando $l = (k/n-1)/2$ para $k \in \mathbb{N}$ e n é um divisor de k, pode-se notar em (5.1) e (5.2) que os coeficientes $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$ são facilmente obtidos, e que α_l e β_l guardam uma íntima relação, de modo que α_l pode ser obtido de β_l por uma eventual troca de sinal.

Contudo, observando mais atentamente a (5.1) e (5.2) pode-se observar que a menos do número $k\pi$ que divide o numerador, os coeficientes $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$ podem assumir valores inteiros. Tal situação é bem desejada e procurada quando se trata de detecção harmônica, pois assim pode-se reduzir consideravelmente a complexidade das operações durante o processamento. De tal forma, na verdade, o que se está fazendo evitar uma divisão por um número representado em ponto flutuante, o que certamente é custoso do ponto de vista computacional, haja visto a irracionalidade de π [19].

Uma forma de se alcançar tal situação é por encontrar um fator de escalonamento para os coeficientes $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$ de tal modo que os mesmo sejam representados por meio de números inteiros. Considerando a situação em que o processamento feito abrange o sinal até o N-ésimo harmônico, deve-se, para tanto, multiplicar os coeficientes $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$ por $\pi N!/4$. Nesta situação, cada termo k no numerador, que vai de 1 até N, é cancelado pelo correspondente termo presente no fatorial. O termo π é responsável por cancelar a divisão por π, tendo em vista o fator que aparece no denominador da expressão dos coeficientes, como em (5.1) e (5.2). Sendo assim, cada coeficiente $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$ se tornará

$$\frac{\pi N!}{4}\alpha_{(\frac{k}{n}-1)/2} = n\left(\frac{N!}{k}\right)(-1)^{(\frac{k}{n}-1)/2} \tag{6.1}$$

e

$$\frac{\pi N!}{4}\beta_{(\frac{k}{n}-1)/2} = n\left(\frac{N!}{k}\right). \tag{6.2}$$

Aplicando as equações acima em (3.34) e (3.36) obtêm-se

$$\begin{aligned}\sum_{n|k}\hat{a}_n\frac{\pi N!}{4}\alpha_{(\frac{k}{n}-1)/2} &= \sum_{n|k}\hat{a}_n n\left(\frac{N!}{k}\right)(-1)^{(\frac{k}{n}-1)/2} \\ &= \frac{N!}{k}\sum_{n|k}\hat{a}_n n(-1)^{(\frac{k}{n}-1)/2}\end{aligned} \tag{6.3}$$

e

$$\begin{aligned}\sum_{n|k}^{\infty}\hat{b}_n\beta_{(\frac{k}{n}-1)/2} &= \sum_{n|k}^{\infty}\hat{b}_n n\left(\frac{N!}{k}\right) \\ &= \frac{N!}{k}\sum_{n|k}^{\infty}\hat{b}_n n.\end{aligned} \tag{6.4}$$

Tal representação escalonada dos coeficientes $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$ combinada com as matrizes inversas escalonadas $\rho \cdot \mathbf{M}_{A_N}^{-1}$ e $\rho \cdot \mathbf{M}_{B_N}^{-1}$, exploradas na seção anterior, formam um conjunto adequado para serem aplicados em contextos que envolvem detecção harmônica. Embora o principal objetivo em situações deste tipo seja estabelecer uma estimativa para os coeficientes da série de Fourier, é possível ainda obtê-los a partir da estimativa aqui descrita. A seção seguinte aborda uma forma de se obter os coeficientes da série de Fourier após os sucessivos escalonamentos.

6.3 Coeficientes da Série de Fourier e a Detecção Harmônica

Usualmente, em problemas que envolvem detecção harmônica, há o interesse em processar o sinal de entrada de acordo com certas características apresentadas após a etapa de detecção harmônica. É proposto que ao invés de se computar a série de Fourier do modo clássico, os coeficientes sejam obtidos a partir da estimativa harmônica previamente apresentada.

Nas seções anteriores foi estabelecido como encontrar uma estimativa para os coeficientes da série de Fourier em contextos em que o objetivo se limitava a realizar a detecção harmônica. Nesta seção é estabelecida uma forma de se obter os coeficientes da série de Fourier a partir da estimativa acima descrita. Esta forma de se obter os coeficientes da série de Fourier se baseia na forma em que a estimativa destes coeficientes foi realizada.

Os sucessivos escalonamento dos coeficientes em (5.1) e (5.2), e das matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$, resulta em um conjunto de coeficientes de Fourier escalonados pelo fator

$$\lambda = \pi \frac{N!}{4} \rho. \tag{6.5}$$

Todo o processo de estimativa dos coeficientes da série de Fourier feitos até então tinha como objetivo manter a relação entre os coeficientes da série de Fourier, levando a um consequente escalonamento dos coeficientes por fatores constantes. Desta forma, obter os coeficientes da série de Fourier a partir da estimativa feita nada mais é do que retirar os fatores de escalonamento introduzidos durante o processo de detecção. Assim, obter os coeficientes se resume a dividir os coeficientes escalonados obtidos durante a detecção pelo fator em (6.5).

A seção seguinte descreve alguns aspectos básicos considerando na confecção do *software*, como por exemplo, interface gráfica e detalhes de implementação. Os códigos escritos para implementação podem ser vistos no Apêndice B.

CAPÍTULO 7

Software

Neste capítulo são endereçados alguns aspectos básicos do projeto de interfaces gráficas e então aplicados ao uso do Matlab®. Além disso, neste capítulo ainda é detalhado o projeto do Analisador de Espectro. O aplicativo foi implementado com a versão R2012a do Matlab®. Este pode ser encontrado para download no link `https://docs.google.com/folder/d/0B7E88fvrkno_NG13QXI2SmwzVkk/edit`.

7.1 Fundamentos de Projeto de Interface Gráfica

Com a evolução dos microcomputadores e microprocessadores, e o consequente avanço e aumento da complexidade dos aplicativos usados, se fez necessário algo que pudesse "simplificar a vida" do usuário final, sem que este tivesse a necessidade de entender o que de fato acontece por baixo da camada de interação. Tal abordagem, facilitaria a difusão do uso de aplicações que possuíssem suporte a algum mecanismo que facilitasse a vida do usuário. O termo Interface Gráfica de Usuário, ou GUI (Graphical User Interface), se refere a ideia de ícones, botões e caixas de diálogos que são apresentados como *front-end*, que nada mais é do que a última camada ou nível de abstração de uma aplicação o qual interage diretamente com o usuário final com a finalidade de facilitar a operação e uso da aplicação.

Há diversas razões pelas quais alguém poderia estar interessado em usar GUI's. Por exemplo, alguém pode estar interessado em automatizar uma função que é usada constantemente, ou talvez deseje compartilhar a aplicação com outras pessoas que não precisam ou não sabem operar a aplicação com muitos detalhes.

Não obstante, o projeto de GUI's comumente é separado em três fases [25] distintas. Estas são Análise, Projeto e Consideração do Usuário.

A primeira fase se refere ao que é feito antes mesmo de lançar mão de papel e lápis para projetar a GUI. Esta fase trata de quem e como usará a aplicação que está sendo construída. Levando em consideração o usuário final, o programador é capaz de entender o que o usuário deseja ou o que poderia ser desnecessário. No caso em questão, é de interesse construir uma interface gráfica para facilitar a análise de sinais. Comumente, os usuários de tais aplicações não são pessoas cuja formação é especializada na área de processamento de sinais, contudo, estes tem um conhecimento básico do assunto, de modo que eles sejam capazes de entender o que é uma série de Fourier.

A segunda fase trata do projeto da GUI, podendo ser realizado apenas com o uso de papel e lápis ou auxiliado por computador. Esta fase só pode ser iniciada após a compreensão das necessidades do usuário final. Esta etapa endereça as funcionalidades, tarefas e componentes

que devem estar presentes na GUI em desenvolvimento. Esta fase engloba a prototipação da GUI.

A terceira e última fase está intimamente relacionada com a primeira. De fato esta fase é uma extensão da primeira. Neste momento, é considerado o tipo de usuário intencionado para a aplicação com base nas suas habilidades visuais, cognitivas e físicas. O primeiro aspecto nos leva a pensar sobre como estarão dispostos os botões e ícones presentes na GUI. O programador deve tomar devida atenção a este aspecto para que as telas de interface com o usuário não fiquem poluídas com excessivos detalhes que poderiam dificultar a operação do aplicativo. O aspecto das habilidades cognitivas remete o programador a como se deve organizar as funcionalidades. É útil organizar em grupos as funcionalidades e tarefas disponíveis na aplicação, de modo que facilite ao usuário a busca por tais serviços. Além disso, o programador deve se preocupar em quando e onde apresentar informações relevantes sobre a aplicação. Usualmente, os programadores e equipes mais experientes procuram padronizar algumas funcionalidades comuns em diversas aplicações, como os locais dos botões para salvar algo ou encerrar a aplicação. O último aspecto é o das habilidades físicas. O programador nunca deve esquecer que o usuário interage com a máquina por meio de uma interface física, o teclado ou mouse. Por esse motivo, é de fundamental importância definir da melhor forma possível, por exemplo, atalhos, ou a forma como determinada tarefa pode ser requisitada da aplicação.

7.2 GUI's com Matlab®

O Matlab® fornece aos programadores um ambiente de desenvolvimento e um conjunto de funções pré-definidas que permite, a mesmo aqueles que tem pouco conhecimento da linguagem, em pouco tempo e com um pouco de esforço, criar GUI's úteis. Esta ferramenta é chamada de *Graphical User Interface Development Environment*, ou GUIDE.

O GUIDE permite ao programador compor sua GUI com o uso de basicamente dois tipos de objetos, *User Interface Control Elements*, chamados de objetos *uicontrol*, e *User Interface Menu*, chamados de objetos uimenu. Os objetos *uicontrol* são objetos que podem ser colocados nas telas de interação com o usuário, e dividem-se em dez tipos, como exposto na Tabela 7.1.

O Matlab® ainda possui um conjunto de janelas pre-definidas que podem ser facilmente manipuladas. Estas são janelas de ajuda (**helpdlg**), mensagem comum (**msgdlg**), mensagem de erro (**errordlg**), aviso (**warndlg**), entrada de dados (**inputdlg**), questionamento com resposta binária (**questdlg**), seleção de arquivos ou diretórios (**uigetfile**) e geração de arquivos (**uiputfile**).

Mais detalhes sobre a constituição de cada tipo de janela e mais informações sobre a ferramenta GUIDE podem ser encontradas em [25] e na própria documentação de ajuda do O Matlab®.

7.3 Implementação em Software com Linguagem Matlab®

Esta seção descreve a composição básica do aplicativo criado que implementa o Analisador de Espectral baseado na série de Fourier Quantizada.

Tabela 7.1 Tipos de objetos *uicontrol*

Tipo de UI Control	Descrição
Check Box	Indica o estado de uma opção ou atributo
Editable Text	Caixa de texto editável
Frame	Usado para visualizar grupos de objetos *uicontrol*
Pop-up Menu	Gera uma lista de opções mutuamente excludentes
List Box	Gera uma lista de opções selecionáveis
Push Buttom	Invoca um evento quando acionado
Radio Button	Indica uma opção que pode selecionada
Toggle Button	Gera botão com apenas dois estados possíveis, on ou off
Slider	Gera um botão que representa um intervalo de valores
Static Text	Caixa de texto não editável

Com o objetivo de facilitar a implementação do *software*, a expansão do sinal de entrada na série de Fourier Aproximada, ou na série de Fourier Quantizada, foi feito por passos. O desenvolvimento matemático feito no Capítulo 3 leva, naturalmente, a um algoritmo ou conjunto de passo finito que pode orientar a análise do sinal.

Para se obter por fim a série de Fourier clássica a partir da série de Fourier Quantizada, isto é, a série de Fourier Aproximada, é preciso conhecer os coeficientes da série de Fourier Quantizada. Diferentemente do que foi proposto por D. F. Souza, R. J. Cintra e H. M. Oliveira [11], neste trabalho a base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}$ é considerada como de fato é, não-ortogonal. A menos de operações usuais de truncamento e arredondamento das máquinas computacionais digitais, tal consideração resulta no cálculo exato dos coeficientes da série de Fourier Quantizada. O conhecimento dos coeficientes da série de Fourier Quantizada é feito mediante o cômputo dos produtos internos do sinal de entrada $f(t)$ e dos elementos da base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}$. A partir dos produtos internos, que são obtidos computacionalmente, pois depende de cada sinal $f(t)$ de entrada, e das matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$ junto com os coeficientes da série de Fourier Quantizada, que são desconhecidos, forma-se um sistema linear, que é resolvido mediante o uso das matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ previamente calculadas. Tal sistema linear possui como incógnitas os coeficientes da série de Fourier Quantizada. Sendo resolvido o sistema, os coeficientes da série de Fourier Quantizada são encontrados. Após encontrar a solução para o sistema, os coeficientes da série de Fourier Quantizada são ponderados segundo a equações descritas em (3.34) e (3.36). Tal ponderação resulta na estimativa dos coeficientes da série de Fourier Aproximada descrita em (2.18).

Tal cálculo pode ser executado pelas funções **sf_signal_coeff**, que computa os coeficientes da série de Fourier Quantizada do sinal completo resolvendo o sistema da forma já descrita, e **smallblocklengthsignal**, que estima os harmônicos da série de Fourier Quantizada a partir blocos que curto comprimento, como descrito na Seção 5.1.

Após conhecidos os valores dos coeficientes $\hat{a}_n$ e $\hat{b}_n$, da série de Fourier Quantizada, pode-se encontrar os coeficientes da série de Fourier. Isso pode ser feito seguindo as equações (5.1)

Algoritmo 2 Algoritmo de obtenção dos coeficientes da série de Fourier Quantizada.

Entrada: O sinal $f(t)$ e número de harmônicos N
Saída: Vetores $\hat{\mathbf{b}}_N$ e $\hat{\mathbf{a}}_N$ contendo os coeficientes da série de Fourier Quantizada $\hat{a}_n$ e $\hat{b}_n$ para $0 \leq n \leq N$

Gerar $\mathbf{v}_N$ e $\mathbf{w}_N$
Carregar as matrizes $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$
$\hat{\mathbf{a}}_N \leftarrow \mathbf{M}_{A_N}^{-1} \cdot \mathbf{v}_N$
$\hat{\mathbf{b}}_N \leftarrow \mathbf{M}_{A_N}^{-1} \cdot \mathbf{w}_N$
Retorna $\hat{\mathbf{a}}_N$ e $\hat{\mathbf{b}}_N$

e (5.2). Mais uma vez, quantidades independentes da forma de onda do sinal de entrada podem ser calculadas a priori. De fato, os coeficientes $\alpha_{(\frac{k}{n}-1)/2}$ e $\beta_{(\frac{k}{n}-1)/2}$ podem ser calculados antes de se iniciar o processamento dos sinais de entrada. Sendo assim, o número de multiplicações em ponto flutuante necessárias para computar os coeficientes de Fourier até o N-ésimo harmônico, já conhecidos os coeficientes da série de Fourier Quantizada, é dado por:

$$\sum_{n=1}^{N} \sigma_0(k), \tag{7.1}$$

em que $\sigma_0(k)$ é um caso especial da função divisor [4, 19] e denota o número de divisores de k. A função responsável pela transformação dos coeficientes da série de Fourier Quantizada nos coeficientes da série de Fourier é chamada de **sig2sf_coeff_alt**. Tal função apresenta como entrada os coeficientes da série de Fourier Quantizada e retorna os coeficientes de Fourier. Mais detalhes podem ser vistos no Apêndice B.

Procedendo da forma como descrita acima, pode-se encontrar diretamente os coeficientes da série de Fourier a partir dos coeficientes da série de Fourier Quantizada. Contudo, em contextos em que há o interesse apenas em se realizar detecção, as simplificações exploradas no Capítulo 5 podem ser usadas para reduzir o tempo de computação. De fato, as multiplicações em ponto flutuante, como descrito no parágrafo anterior, se tornam multiplicações por números inteiros. As funções **divisorswithoddresult**, **signsvectorcoeffs** e **sig2sf_coef_alt_detection** desempenham este papel dentro do *software* criado. Mais detalhes podem ser vistos no Apêndice B.

CAPÍTULO 8

Conclusões

Neste trabalho abordou-se uma forma alternativa de se realizar o processamento de sinais de uma dimensão em contextos de detecção harmônica. Foi apresentada uma forma alternativa de se estimar os coeficientes da série de Fourier baseado no trabalho de D. F. de Souza, R. J. de Sobral Cintra e H. M. de Oliveira em [11]. A abordagem realizada em [11] aproxima os coeficientes de Fourier por aproximações do coeficientes da série de Fourier Quantizada, considerando o núcleo $\text{sign}(\cos(n\omega_0 t))$ e $\text{sign}(\sin(n\omega_0 t))$. A aproximação proposta é feita tomando a base $\{1, \text{sign}(\cos(n\omega_0 t)), \text{sign}(\sin(n\omega_0 t))\}_{n=0}^{\infty}$ como sendo ortogonal. Tal suposição, em sentido estrito, não é verdade, como demonstrado no Capítulo 3, contudo, provê uma boa aproximação sob certas condições.

Neste trabalho, particularmente, optou-se por encontrar os coeficientes de Fourier a partir dos coeficientes da série de Fourier Quantizada sem aproximações por resolver os sistemas em (3.24) e (3.28) e então aplicar os coeficientes da série de Fourier Quantizada em (3.34) e (3.36). Algumas considerações sobre possíveis simplificações da complexidade computacional das operações envolvidas para se obter os coeficientes da série de Fourier, a_k e b_k, a partir dos coeficientes da série de Fourier Quantizada, $\hat{a}_n$ e $\hat{b}_n$, foram exploradas. No Capítulo 5 foram abordadas algumas possíveis otimizações para o processamento dos sinais de entrada com quebra em blocos de comprimento curto; cálculo a priori e redução de redundâncias; método de Dempster-McLeod e método do fator de escalonamento.

Em especial, é interessante ressaltar a aplicação do método do fator de escalonamento aos coeficientes α_l e β_l em (6.1) e (6.2), e às matrizes inversas $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$ em (3.25) e (3.29). A aplicação de tal método sobre tais quantidades resulta em uma redução da complexidade das operações envolvidas para se estimar os coeficientes de Fourier em contextos de detecção de sinais. Obviamente, tal redução é bem vinda e implica na redução do tempo de computação, haja visto que tais simplificações são calculadas a priori.

Em alguns casos, após a detecção da presença de determinado harmônico, há o interesse em se processar o sinal de entrada. Ao invés de usar a abordagem clássica, é possível obter os coeficientes da série de Fourier de modo aproximado anulando o efeito dos sucessivos escalonamentos aplicados às quantidades acima descritas. Tal anulamento pode ser implementado pela normalização dos coeficientes escalonados obtidos na fase de detecção, já que o coeficiente que expressa o efeito final sobre os coeficientes pode ser facilmente obtido, como em (6.5).

Como trabalhos futuros, pretende-se estender o Analisador de Espectro para sinais bidimensionais, oferecendo suporte a análise de sinais de imagem e vídeo. Além disso, é esperado poder implementar a execução de análise e salvamento automático de diversos arquivos, de modo sequencial. Esta particularidade é de interesse para situações que os dados a serem analisados estejam espelhados por diversos arquivos menores. Não obstante, haja visto a evolução

dos computadores digitais atuais, futuros enriquecimentos do *software* poderão oferecer suporte a processamento paralelo, como levemente abordado na Seção 5.1. Tal tipo de implementação deverá fazer uso do processamento em quebra por blocos de curto comprimento.

APÊNDICE A

Códigos e Funções Usadas para Simulação Computacional

Este capítulo contém os códigos usados para simulação durante a confecção deste trabalho de graduação escrito em linguagem de Matlab®.

1. Função que implementa a função chi-fração.

```
function output = chifrac( n, k )
%CHIFRAC implements the chi-fraction function.

%This function return the inverse of the product between the
    denominator
%and numerator of the irreductable fraction p/q of n/k, provided that
    p and
%q are odds. Otherwise, it return zero.

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Sep 06/2012

%Change Log:

%Requirements:
%   1)The input arguments must be integers.

output = 0;

[p q] = irrfrac(n, k);

if(mod(p,2) == 1 && mod(q,2)==1)
    output = 1/(p*q);
end


end
```

2. Função que implementa a função chi-fração sinalizada.

```
function output = chifracsign( n, k )
```

```
%CHIFRAC implements the chi-fraction function signalized.

%This function return the inverse of the product between the
    denominator
%and numerator of the irreductable fraction p/q of n/k, provided that
    p and
%q are odds multiplied by (-1)^((p+q)2-1). Otherwise, it return zero.

%Since p and q are odds, the sumation p+q is always even, so the
%calculation (p+q)/2-1 always return a integers value.

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Sep 06/2012

%Change Log:

%Requirements:
%   1)The input arguments must be integers.

output = 0;

[p q] = irrfrac(n, k);

if(mod(p,2) == 1 && mod(q,2)==1)
    output = (-1)^((p+q)/2-1)/(p*q);
end



end
```

3. Função que retorna a fração n/k simplificada.

```
function [p q] = irrfrac( n, k )
%IRRFRAC returns the denominator and numaretor of the irredeuctable
    fraction
%between n and k.

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Sep 06/2012

%Change Log:

%Requirements:
```

```
%   1)The input arguments must be integers.

n = round(n);
k = round(k);

mdc = gcd(n,k);

p = n/mdc;
q = k/mdc;

end
```

4. Função que retorna as matrizes $\mathbf{M}_{A_N}$ e $\mathbf{M}_{B_N}$.

```
function [A B] = mtrxgen_signal( N )
%MTRXGEN Returns the matrxies MA_N and MB_N used to obtain the Fourier
    -like
%coefficients, using the kernel sign(cos) and sign(sin).

%This function was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%September 08/2012

%Change Log:


%Requirements:
%   1)The vector f and t must have the same size. This requirement is
%   imposed by the use of vectorized product '.*'.

%Sanity check:
%The input argumet N must be a non-negative integer.

if(N < 0)
    disp('Error! There is no negative frequency.');
    return;
end

%The matrix of An,k's and Bn,k's:

A = [];
B = [];

for n = 1:N
    for k = 1:N
        A(n,k) = chifracsign(n,k);
        B(n,k) = chifrac(n,k);
    end
end
```

```
38
39
40 end
```

5. Função que retorna a série de Fourier.

```
1  function f = sf( coeffCos, coeffSin, t )
2  %SF_SIGNAL Summary of this function goes here
3  %   Detailed explanation goes here
4  %This function returns the values of the Fourier-like serie of a given
5  %mathematical function at the values of parameter t, using the
       fundamental
6  %frequency freq and the Fourier-like serie coefficients passed as
7  %parameters. The input arguments are the Fourier-like serie
       coefficients as
8  %respect to sign(cos) and sign(sin) given as row vectors in coeffCos
       and
9  %coeffSin, respectively, the fundamental frequency freq, given in Hz,
       and
10 %the time t, as a row vector. The output argument is only the serie
       value
11 %evaluated at the time t.
12
13 %This functions was created as a part of End Course Project of
14 %the author, Electronics Engineering student at Universidade
15 %Federal de Pernambuco, Brazil.
16
17 %Author: Diego F. G. Coelho
18 %June 05/2012
19
20 %Change Log:
21 %June 05/2012: It does not work properly.
22 %August 29/2012: Problem fixed. The problem, indeed, was in the
       function
23 %sf_coeff, which its output is input of the present function.
24
25 %Requirements:
26 %   1)The vectors coeffCos and coeffSin must have the same size.
27
28 %Sanity check:
29 %Verify if coeffCos and coeffSin has the same length.
30 if(length(coeffCos)~=length(coeffSin))
31     disp('Error! The coefficients vectors must have the same size.');
32     return;
33 end
34
35 %Create the vector of harmonics:
36 n = 0:1:length(coeffCos)-1;
37
38 %The fundamental frequency:
39 omega_0 = 2*pi/(t(end)-t(1));
40
```

```
%Create the matrix which contain the elements basis. Each line contain
    the
%base element evaluated at each harmonic.
sfCos = cos(omega_0*(t'*n));
sfSin = sin(omega_0*(t'*n));

%Create the output argument:
f = (sfCos*(coeffCos') + sfSin*(coeffSin'))';

end
```

6. Função que retorna os coeficientes da série de Fourier.

```
function [coeffCos coeffSin] = sf_coeff( f, t, N)
%SF Summary of this function goes here
%   Detailed explanation goes here
%This function returns the coefficients of Fourier serie
%The coefficients are returned in a matrix which the lines
%contain the coeficients of the expansion due the cos(.) and
%sin(.) kernels, respectively. The
%input arguments are the function f to be expanded, the time vector t,
%which are both row vectors and the integer harmonic parameter N that
%represent the Nth harmonic in what the representation should be
    truncated.

%This functions was created as a part of End Course Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%June 04/2012

%Requirements:
%   1)The vector f and t must have the same size. This requirement is
%   imposed by the use of vectorized product '.*'.

%Sanity check:
%Verify if N is greater is positive. N can be zero, it means that the
    user
%is interested in the mean.
if(N < 0)
    disp('Error! There is no negative frequency.');
    return;
end

%The coeffcients vector due cos(.) and sin(.) repectively:
coeffCos = [];
coeffSin = [];

%The fundamental ferequency:
omega_0 = 2*(pi/(t(end)-t(1)));

```

```
%Frequency-time product:
freq_time = omega_0.*t;

for k = 0:N
    coeffCos = [coeffCos, trapz(t,f.* cos(k.*freq_time))];
    coeffSin = [coeffSin, trapz(t,f.* sin(k.*freq_time))];
end

coeffCos = (omega_0/pi)*coeffCos;
%By the definition of coefficient of cos(.) of 0th harminic:
coeffCos(1) = coeffCos(1)/2;
coeffSin = (omega_0/pi)*coeffSin;

end
```

7. Função que retorna a série de Fourier Quantizada.

```
function f = sf_signal( coeffCos, coeffSin, t)
%SF_SIGNAL Return the Fourier-like serie of f expanded using the
    signal
%function of sin and cos function as kernel function.
%   Detailed explanation goes here
%This function returns the values of the Fourier-like serie of a given
%mathematical function at the values of parameter t, using the
    fundamental
%frequency freq and the Fourier-like serie coefficients passed as
%parameters. The input arguments are the Fourier-like serie
    coefficients as
%respect to sign(cos) and sign(sin) given as row vectors in coeffCos
    and
%coeffSin, respectively, the fundamental frequency freq, given in Hz,
    and
%the time t, as a row vector. The output argument is only the serie
    value
%evaluated at the time t.

%This functions was created as a part of End Course Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%June 04/2012

%Change Log:
%June 05/2012: It does not work properly.
%August 29/2012: Problem fixed. The written script was expand the
    function
%using the usual kernel, cos(.) and sin(.).

%Requirements:
%   1)The vectors coeffCos and coeffSin must have the same size.

```

```
%Sanity check:
%Verify if coeffCos and coeffSin has the same length.
if(length(coeffCos)~=length(coeffSin))
    disp('Error! The coefficients vectors must have the same size.');
    return;
end

%Create the vector of harmonics:
n = 0:1:length(coeffCos)-1;

%The fundamental frequency:
omega_0 = 2*pi/(t(end)-t(1));

%Create the matrix which contain the elements basis. Each line contain
    the
%base element evaluated at each harmonic.
sfCos = sign(cos(omega_0*(t'*n)));
sfSin = sign(sin(omega_0*(t'*n)));

%Create the output argument:
f = (sfCos*(coeffCos') + sfSin*(coeffSin'))';

end
```

8. Função que retorna os coeficientes da série de Fourier Quantizada.

```
function [coeffCos coeffSin] = sf_signal_coeff( f, t, N)
%SF_SIGNAL Return the Fourier-like serie of f expanded using the
    signal
%function of sin and cos function as kernel function.
%   Detailed explanation goes here
%This function returns the Fourier-like serie expanded using the
    signal
%function. The coefficients are returned in a matrix which the lines
%contain the coeficients of the expansion due the signal(cos) and
%signal(sin) kernels, respectively. The
%input arguments are the function f to be expanded, the time vector t,
%which are both row vectors and the integer harmonic parameter N that
%represent the Nth harmonic in what the representation should be
    trucated.

%This functions was created as a part of End Course Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%June 04/2012

%Change Log:
%August 30/2012: It does not work. Theoretical errors.
%September 06/2012:Theoretical erros corrected. It is working properly
    .
```

```
23
24 %Requirements:
25 %   1)The vector f and t must have the same size. This requirement is
26 %   imposed by the use of vectorized product '.*'.
27
28 %Sanity check:
29 %Verify if N is greater is positive. N can be zero, it means that the
       user
30 %is interested in the mean.
31 if(N < 0)
32     disp('Error! There is no negative frequency.');
33     return;
34 end
35
36 %The coeffcients vector due singal(cos) adn signal(sin) repectively:
37 coeffCos = [];
38 coeffSin = [];
39
40 %The inner product values:
41 fsigncos = [];
42 fsignsin = [];
43
44 %The fundamental ferequency:
45 omega_0 = 2*(pi/(t(end)-t(1)));
46
47 %Frequency-time product:
48 freq_time = omega_0.*t;
49
50 %The matrix of An,k's and Bn,k's:
51
52 [A B] = mtrxgen_signal(N);
53
54 %This calculation of inner product use the definition given by D. F.
       Souza,
55 %R. Cintra and de Oliveira. Observe that it is multiplied by 1/T,
       where T is
56 %the fundamental period, not by 2/T.
57
58 %These are column vectors.
59 for k = 1:N
60     fsigncos = [fsigncos; trapz(t,f.* sign(cos(k.*freq_time)))];
61     fsignsin = [fsignsin; trapz(t,f.* sign(sin(k.*freq_time)))];
62 end
63
64 fsigncos = (omega_0/(2*pi))*fsigncos;
65 fsignsin = (omega_0/(2*pi))*fsignsin;
66
67 %This solve the linear system in order to find the fourier-like
       coeffcients
68 %using the base sign(.)
69 coeffCos = (A\fsigncos)';
70 %This include the 0th harmonic.
```

```
71 coeffCos = [trapz(t,f), coeffCos];
72
73 coeffSin = (B\fsignsin)';
74 %This include the 0th harmonic.
75 coeffSin = [0, coeffSin];
76
77 end
```

9. Função que converte os coeficientes da série de Fourier Quantizada para os da série de Fourier clássica.

```
1  function [ coeffCos coeffSin ] = sig2sf_coeff( coeffSigCos,
       coeffSigSin )
2  %SIG2SF Summary of this function goes here
3  %   Detailed explanation goes here
4  %This function performs the convertion from the coeffcients of Fourier
       -like
5  %serie using sign(cos) and sign(sin) functions as kernel to Fourier
6  %serie coefficients.The input arguments coeffSigCos and coeffSigSin
       are row
7  %vector which contain the coefficients of Fourier-like serie using
8  %sign(cos) and sign(sin) functions as kernel, respectively, and
       coeffCos
9  %and coeffSin are row vectors which contains the Fourier coefficients
       using
10 %cos and sin functions as kernel.
11
12 %This functions was created as a part of End Course Project of
13 %the author, Electronics Engineering student at Universidade
14 %Federal de Pernambuco, Brazil.
15
16 %Author: Diego F. G. Coelho
17 %June 04/2012
18
19 %Change Log:
20 %June 04/2012: It does not work properly.
21 %September 06/2012: The problem was due theoretical problems. It is
       fixed.
22 %It is working properly.
23
24 %Sanity check:
25 %The coefficients must have the same length
26
27 if(length(coeffSigCos)~=length(coeffSigSin))
28     disp('Error! The coeffcients vector must have the same length.');
29     return;
30 end
31
32 %The coefficients of Fourier serie:
33 coeffCos = [];
34 coeffSin = [];
35
```

```
for k = 1:(length(coeffSigCos)-1)
    a_k = 0;
    b_k = 0;
    for n = 1:k
        if(mod(k,n)==0)
            if(mod(k/n,2)==1)
                frac = k/n;
                a_k = a_k + coeffSigCos(n+1)*(((-1)^((frac-1)/2))/frac
                    );
                b_k = b_k + coeffSigSin(n+1)/frac;
            end
        end
    end
    coeffCos = [coeffCos, a_k];
    coeffSin = [coeffSin, b_k];
end

coeffCos = (4/pi).*coeffCos;
coeffSin = (4/pi).*coeffSin;

%The mean value is the same and the first value with respect to sin is
%always zero:

coeffCos = [coeffSigCos(1), coeffCos];
coeffSin = [coeffSigSin(1), coeffSin];

end
```

10. Função que retorna fator de escalonamento α das matrizes $\mathbf{M}_{A_N}^{-1}$ e $\mathbf{M}_{B_N}^{-1}$.

```
function [Ascaled Bscaled scale MSEA MSEB] = scaleFactor( A , B, prec)
%SCALEFACTOR This function return a matrix Ascaled = scale*.A and
    Bscaled =
%scale*.B, where Ascaled and Bscaled are matrixes with only integer
    values
%and scale is a scalar number.

%The input argument prec is the precision used. The errors must be at
    most
%equals to prec.

%This functions was created as a part of End Course Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Sep 07/2012

%Change Log:
%Sep 08/2012: Adapt the function to perofrm the same at two matrixes
    at the
%same time.
```

```
%Oct 17/2012: Correct the MSE calculation. It must be divided by the
    number
%of elements, that is, since the matrix is squared, linsA^2.

%Requirements:
%   1)The input argument prec must be different from zero.
%   2)The matrixes A and B must have the same dimensions.

if(prec <= 0)
    disp('Error! The input argument prec must be greater than zero.');
    return;
end

%The numbers of lines and columns of the matrix A
[linsA clumnsA] = size(A);

%The numbers of lines and columns of the matrix B
[linsB clumnsB] = size(B);

if(linsA ~= linsB || clumnsA ~= clumnsB)
    disp('Error! The input argument matrixes A and B must have the
        same dimension.');
    return;
end


%Flag used to tell if all the compon enets are with the error within
%specified by the input argument prec.
flag = false;

%The initial scale factor value
scale = 0;

%The output matrix A and B
Ascaled = [];
Bscaled = [];

while ~flag
    scale = scale + 1;
    Ascaled = scale.*A;
    Bscaled = scale.*B;
    errA = Ascaled-round(Ascaled);
    errB = Bscaled-round(Bscaled);
    if(max(abs(errA)) <= prec & max(abs(errB)) <= prec)
        flag = true;
        Ascaled = round(Ascaled);
        Bscaled = round(Bscaled);
    end
end

MSEA = sqrt(sum(sum(errA.^2)))/linsA^2;
MSEB = sqrt(sum(sum(errB.^2)))/linsB^2;
```

```
69
70 end
```

APÊNDICE B

Códigos e Funções Usadas para a Implementação do Software

Este capítulo contém os códigos usados para a implementação do software proposto neste trabalho de graduação escrito em linguagem de Matlab®. Alguma funções presentes nesta seção são uma adaptação, ou otimização, de algumas usadas na seção anterior. As funções que permaneceram inalteradas, mas que foram usadas na implementação, não estão listadas nesta seção.

1. Arquivo do programa principal da interface gráfica.

```
function varargout = Analisador(varargin)
% ANALISADOR MATLAB code for Analisador.fig
%      ANALISADOR, by itself, creates a new ANALISADOR or raises the existing
%      singleton*.
%
%      H = ANALISADOR returns the handle to a new ANALISADOR or the handle to
%      the existing singleton*.
%
%      ANALISADOR('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in ANALISADOR.M with the given input arguments.
%
%      ANALISADOR('Property','Value',...) creates a new ANALISADOR or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Analisador_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Analisador_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Analisador
```

```
24
25 % Last Modified by GUIDE v2.5 14-Oct-2012 17:56:41
26
27 % Begin initialization code - DO NOT EDIT
28 gui_Singleton = 1;
29 gui_State = struct('gui_Name',       mfilename, ...
30                    'gui_Singleton',  gui_Singleton, ...
31                    'gui_OpeningFcn', @Analisador_OpeningFcn, ...
32                    'gui_OutputFcn',  @Analisador_OutputFcn, ...
33                    'gui_LayoutFcn',  [] , ...
34                    'gui_Callback',   []);
35 if nargin && ischar(varargin{1})
36     gui_State.gui_Callback = str2func(varargin{1});
37 end
38
39 if nargout
40     [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41 else
42     gui_mainfcn(gui_State, varargin{:});
43 end
44 % End initialization code - DO NOT EDIT
45
46 end
47
48 % --- Executes just before Analisador is made visible.
49 function Analisador_OpeningFcn(hObject, eventdata, handles, varargin)
50 % This function has no output args, see OutputFcn.
51 % hObject    handle to figure
52 % eventdata  reserved - to be defined in a future version of MATLAB
53 % handles    structure with handles and user data (see GUIDATA)
54 % varargin   command line arguments to Analisador (see VARARGIN)
55
56 % Choose default command line output for Analisador
57 handles.output = hObject;
58
59 handles.file = '';
60 handles.flag = 1;
61 handles.prec = 10^(-2);
62 handles.lngth = 'Comp.';
63 handles.N = 1;
64 handles.sfplot = true;
65 handles.save = false;
66 handles.discrete = false;
67 handles.toplot2 = [];
68 handles.n = [];
69 handles.t = [];
70 handles.kbetal = [];
71 handles.kalphal = [];
72 handles.multipliers = [];
73
74 % Update handles structure
75 guidata(hObject, handles);
```

```
76
77 % UIWAIT makes Analisador wait for user response (see UIRESUME)
78 % uiwait(handles.figure1);
79
80 end
81
82 % --- Outputs from this function are returned to the command line.
83 function varargout = Analisador_OutputFcn(hObject, eventdata, handles)
84 % varargout  cell array for returning output args (see VARARGOUT);
85 % hObject    handle to figure
86 % eventdata  reserved - to be defined in a future version of MATLAB
87 % handles    structure with handles and user data (see GUIDATA)
88
89 % Get default command line output from handles structure
90 varargout{1} = handles.output;
91
92 end
93
94 function edit1_Callback(hObject, eventdata, handles)
95 % hObject    handle to edit1 (see GCBO)
96 % eventdata  reserved - to be defined in a future version of MATLAB
97 % handles    structure with handles and user data (see GUIDATA)
98
99 % Hints: get(hObject,'String') returns contents of edit1 as text
100 %        str2double(get(hObject,'String')) returns contents of edit1
    as a double
101
102 end
103
104 % --- Executes during object creation, after setting all properties.
105 function edit1_CreateFcn(hObject, eventdata, handles)
106 % hObject    handle to edit1 (see GCBO)
107 % eventdata  reserved - to be defined in a future version of MATLAB
108 % handles    empty - handles not created until after all CreateFcns
    called
109
110 % Hint: edit controls usually have a white background on Windows.
111 %       See ISPC and COMPUTER.
112 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
113     set(hObject,'BackgroundColor','white');
114 end
115
116 end
117
118 % --- Executes on button press in pushbutton1.
119 function pushbutton1_Callback(hObject, eventdata, handles)
120 % hObject    handle to pushbutton1 (see GCBO)
121 % eventdata  reserved - to be defined in a future version of MATLAB
122 % handles    structure with handles and user data (see GUIDATA)
123
124 file = handles.file;
```

```
125
126 load(file,'t','y');
127 N = handles.N;
128 A = handles.A;
129 B = handles.B;
130 lngth = handles.lngth;
131
132 if(2*N > lngth)
133     errordlg('O nmero de harmnicos deve ser no mximo a metade do
        nmero de amostras.','File Error');
134     return;
135 end
136
137 switch handles.flag
138     case 1
139         if(isempty(lngth))
140             [coeffSigCos coeffSigSin] = sf_signal_coeff(y, t , N, A, B
                );
141             [coeffCos coeffSin] = sig2sf_coeff_alt(coeffSigCos,
                coeffSigSin, ...
142                 handles.kalphal, handles.kbetal);
143             toplot2 = sf(coeffCos, coeffSin, t);
144         else
145             toplot2 = smallblocklengthsignalsf(y, t, N, lngth, A, B);
146         end
147
148         %plot in axes2
149         axes(handles.axes2);
150         plot(gca, t, toplot2);
151         title('Srie de Fourier arpoximada');
152         xlabel('t');
153
154         %compute and plot the canonical fourier serie if the variable
155         %sfplot is true.
156         if(handles.sfplot == true)
157             if(isempty(lngth))
158                 [coeffCos coeffSin] = sf_coeff(y, t , N);
159                 toplot1 = sf(coeffCos, coeffSin, t);
160             else
161                 toplot1 = smallblocklengthsf(y, t, N, lngth);
162             end
163         %plot in axes1
164         axes(handles.axes1);
165         plot(gca, t, toplot1);
166         title('Srie de Fourier');
167         xlabel('t');
168         end
169
170         handles.toplot2 = toplot2;
171         handles.t = t;
172         handles.discrete = false;
173
```

```
174         % Update handles structure
175         guidata(hObject, handles);
176 
177     case 2
178         if(isempty(lngth))
179             [coeffSigCos coeffSigSin] = sf_signal_coeff(y, t , N, A, B
                );
180             toplot2 = sf_signal(coeffSigCos, coeffSigSin, t);
181         else
182             toplot2 = smallblocklengthsignal(y, t, N, lngth, A, B);
183         end
184         %plot in axes2
185         axes(handles.axes2);
186         plot(gca, t, toplot2);
187         title('Série quantizada');
188         xlabel('t');
189 
190         %compute and plot the canonical fourier serie if the variable
191         %sfplot is true.
192         if(handles.sfplot == true)
193             if(isempty(lngth))
194                 [coeffCos coeffSin] = sf_coeff(y, t , N);
195                 toplot1 = sf(coeffCos, coeffSin, t);
196             else
197                 %lngth = str2num(lngth);
198                 toplot1 = smallblocklengthsf(y, t, N, lngth);
199             end
200         %plot in axes1
201         axes(handles.axes1);
202         plot(gca, t, toplot1);
203         title('Série de Fourier');
204         xlabel('t');
205         end
206 
207         handles.toplot2 = toplot2;
208         handles.t = t;
209         handles.discrete = false;
210 
211         % Update handles structure
212         guidata(hObject, handles);
213 
214 
215     case 3
216         if(isempty(lngth))
217             [coeffSigCos coeffSigSin] = sf_signal_coeff(y, t , N, A, B
                );
218             [coeffCos coeffSin] = sig2sf_coeff_alt(coeffSigCos,
                    coeffSigSin,...
219                 handles.kalphal, handles.kbetal);
220         else
221             [coeffCos coeffSin] = smallblocklengthsignalsfcoeff(y, t,
                N, lngth, A, B);
```

```
222         end
223         toplot2 = coeffCos.^2+coeffSin.^2;
224         toplot2(1) = 0;
225         %plot in axes2
226         axes(handles.axes2);
227         n = 0:length(coeffCos)-1;
228         stem(gca, n, toplot2);
229         title('Coeficientes de Fourier aproximados');
230         ylabel('a_n^2+b_n^2');
231         xlabel('n');
232
233         %compute and plot the canonical fourier coefficients if the
                variable
234         %sfplot is true.
235         if(handles.sfplot == true)
236             if(isempty(lngth))
237                 [coeffCos coeffSin] = sf_coeff(y, t , N);
238             else
239                 [coeffCos coeffSin] = smallblocklengthsfcoeff(y, t, N,
                        lngth, A, B);
240             end
241             toplot1 = coeffCos.^2 + coeffSin.^2;
242             toplot1(1) = 0;
243             %plot in axes1
244             axes(handles.axes1);
245             stem(gca, n, toplot1);
246             title('Coeficientes de Fourier');
247             ylabel('a_n^2+b_n^2');
248             xlabel('n');
249         end
250
251         handles.toplot2 = toplot2;
252         handles.n = n;
253         handles.discrete = true;
254
255         % Update handles structure
256         guidata(hObject, handles);
257
258
259     case 4
260         if(isempty(lngth))
261             [coeffSigCos coeffSigSin] = sf_signal_coeff(y, t , N, A, B
                    );
262         else
263             [coeffSigCos coeffSigSin] = smallblocklengthsignalcoeff(y,
                    t, N, lngth, A, B);
264         end
265         toplot2 = coeffSigCos.^2+coeffSigSin.^2;
266         toplot2(1) = 0;
267         %plot in axes2
268         axes(handles.axes2);
269         n = 0:length(coeffSigCos)-1;
```

```
270         stem(gca, n, toplot2);
271         title('Coeficientes da  s rie  quantizada');
272         ylabel('a_n^2+b_n^2');
273         xlabel('n');
274 
275         %compute and plot the canonical fourier coefficients if the
                variable
276         %sfplot is true.
277         if(handles.sfplot == true)
278             if(isempty(lngth))
279                 [coeffCos coeffSin] = sf_coeff(y, t , N);
280             else
281                 [coeffCos coeffSin] = smallblocklengthsfcoeff(y, t, N,
                        lngth, A, B);
282             end
283             toplot1 = coeffCos.^2 + coeffSin.^2;
284             toplot1(1) = 0;
285             %plot in axes1
286             axes(handles.axes1);
287             stem(gca, n, toplot1);
288             title('Coeficientes de Fourier');
289             ylabel('a_n^2+b_n^2');
290             xlabel('n');
291         end
292 
293         handles.toplot2 = toplot2;
294         handles.n = n;
295         handles.discrete = true;
296 
297         % Update handles structure
298         guidata(hObject, handles);
299 
300     case 5
301         handles.A = A;
302         handles.B = B;
303         handles.N = N;
304         kbetal = divisorswithoddresult(N);
305         handles.kbetal = kbetal;
306         handles.kalphal = signsvectorcoeffs(kbetal);
307 
308         % Update handles structure
309         guidata(hObject, handles);
310 
311         set(handles.edit2, 'String', num2str(handles.N));
312         set(handles.edit4, 'String', num2str(handles.lngth));
313         set(handles.edit3, 'String', num2str(handles.prec));
314 
315         if(isempty(lngth))
316             [coeffSigCos coeffSigSin] = sf_signal_coeff(y, t , N, A, B
                    );
317             [coeffCos coeffSin] = sig2sf_coeff_alt_detection(
                    coeffSigCos,...
```

```
318                 coeffSigSin, handles.kalphal, handles.kbetal, handles.
                        multipliers);
319             %Scaling the a_0:
320             coeffCos(1) = handles.scale*coeffCos(1);
321         else
322             [coeffCos coeffSin] =
                    smallblocklengthsignalsfcoeffdetection(y,...
323                 t, N, lngth, A, B, handles.kalphal, handles.kbetal,
                        handles.multipliers);
324             coeffCos(1) = 0;
325
326         end
327
328 %         [coeffSigCos coeffSigSin] = sf_signal_coeff(y, t , N, A, B);
329 %         [coeffCos coeffSin] = sig2sf_coeff_alt_detection(coeffSigCos
    ,...
330 %             coeffSigSin, handles.kalphal, handles.kbetal, handles.
    multipliers);
331 %         %Scaling the a_0:
332 %         coeffCos(1) = handles.scale*coeffCos(1);
333         toplot2 = coeffCos.^2+coeffSin.^2;
334         %plot in axes2
335         axes(handles.axes2);
336         n = 0:length(coeffCos)-1;
337         stem(gca, n, toplot2);
338         title('Coef. de Fourier aproximados escalonados');
339         ylabel('\alpha^2(a_n^2+b_n^2)');
340         xlabel('n');
341
342         %compute and plot the canonical fourier serie if the variable
343         %sfplot is true.
344         if(handles.sfplot == true)
345             if(isempty(lngth))
346                 [coeffCos coeffSin] = sf_coeff(y, t , N);
347             else
348                 [coeffCos coeffSin] = smallblocklengthsfcoeff(y, t, N,
                        lngth, A, B);
349                 coeffCos(1) = 0;
350             end
351             %[coeffCos coeffSin] = sf_coeff(y, t , N);
352             toplot1 = coeffCos.^2 + coeffSin.^2;
353             %plot in axes1
354             axes(handles.axes1);
355             stem(gca, n, toplot1);
356             title('Coeficientes de Fourier');
357             ylabel('a_n^2+b_n^2');
358             xlabel('n');
359         end
360         handles.toplot2 = toplot2;
361         handles.n = n;
362         handles.discrete = true;
363 end
```

```
364
365         % Update handles structure
366         guidata(hObject, handles);
367
368 end
369
370 function edit2_Callback(hObject, eventdata, handles)
371 % hObject    handle to edit2 (see GCBO)
372 % eventdata  reserved - to be defined in a future version of MATLAB
373 % handles    structure with handles and user data (see GUIDATA)
374
375 % Hints: get(hObject,'String') returns contents of edit2 as text
376 %        str2double(get(hObject,'String')) returns contents of edit2
    as a double
377
378 end
379
380 % --- Executes during object creation, after setting all properties.
381 function edit2_CreateFcn(hObject, eventdata, handles)
382 % hObject    handle to edit2 (see GCBO)
383 % eventdata  reserved - to be defined in a future version of MATLAB
384 % handles    empty - handles not created until after all CreateFcns
    called
385
386 % Hint: edit controls usually have a white background on Windows.
387 %       See ISPC and COMPUTER.
388 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
389     set(hObject,'BackgroundColor','white');
390 end
391 end
392
393 % --------------------------------------------------------------------
394 function analyses_1D_Callback(hObject, eventdata, handles)
395 % hObject    handle to analyses_1D (see GCBO)
396 % eventdata  reserved - to be defined in a future version of MATLAB
397 % handles    structure with handles and user data (see GUIDATA)
398
399 end
400
401 % --------------------------------------------------------------------
402 function seriefouriercanonica_1D_Callback(hObject, eventdata, handles)
403 % hObject    handle to seriefouriercanonica_1D (see GCBO)
404 % eventdata  reserved - to be defined in a future version of MATLAB
405 % handles    structure with handles and user data (see GUIDATA)
406 handles.flag = 1;
407
408 [A B] = mtrxgen_signal(handles.N);
409
410 handles.A = A;
411 handles.B = B;
412 kbetal = divisorswithoddresult(handles.N);
```

```
413 handles.kbetal = kbetal;
414 handles.kalphal = signsvectorcoeffs(kbetal);
415
416 % Update handles structure
417 guidata(hObject, handles);
418
419 end
420
421 % --------------------------------------------------------------------
422 function seriefourierquantizada_1D_Callback(hObject, eventdata,
    handles)
423 % hObject    handle to seriefourierquantizada_1D (see GCBO)
424 % eventdata  reserved - to be defined in a future version of MATLAB
425 % handles    structure with handles and user data (see GUIDATA)
426 handles.flag = 2;
427
428 [A B] = mtrxgen_signal(handles.N);
429
430 handles.A = A;
431 handles.B = B;
432 kbetal = divisorswithoddresult(handles.N);
433 handles.kbetal = kbetal;
434 handles.kalphal = signsvectorcoeffs(kbetal);
435
436
437 % Update handles structure
438 guidata(hObject, handles);
439
440 end
441
442 % --------------------------------------------------------------------
443 function coeficientefourier_1D_Callback(hObject, eventdata, handles)
444 % hObject    handle to coeficientefourier_1D (see GCBO)
445 % eventdata  reserved - to be defined in a future version of MATLAB
446 % handles    structure with handles and user data (see GUIDATA)
447 handles.flag = 3;
448
449 [A B] = mtrxgen_signal(handles.N);
450
451 handles.A = A;
452 handles.B = B;
453 kbetal = divisorswithoddresult(handles.N);
454 handles.kbetal = kbetal;
455 handles.kalphal = signsvectorcoeffs(kbetal);
456
457
458 % Update handles structure
459 guidata(hObject, handles);
460
461 end
462
463 % --------------------------------------------------------------------
```

```
464 function coeficientequantizada_1D_Callback(hObject, eventdata, handles
    )
465 % hObject    handle to coeficientequantizada_1D (see GCBO)
466 % eventdata  reserved - to be defined in a future version of MATLAB
467 % handles    structure with handles and user data (see GUIDATA)
468 handles.flag = 4;
469
470 [A B] = mtrxgen_signal(handles.N);
471
472 handles.A = A;
473 handles.B = B;
474 kbetal = divisorswithoddresult(handles.N);
475 handles.kbetal = kbetal;
476 handles.kalphal = signsvectorcoeffs(kbetal);
477
478
479 % Update handles structure
480 guidata(hObject, handles);
481
482 end
483
484 % --------------------------------------------------------------------
485 function deteccao_1D_Callback(hObject, eventdata, handles)
486 % hObject    handle to deteccao_1D (see GCBO)
487 % eventdata  reserved - to be defined in a future version of MATLAB
488 % handles    structure with handles and user data (see GUIDATA)
489 handles.flag = 5;
490
491 [A B] = mtrxgen_signal(handles.N);
492 [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
493
494 handles.scale = scaleF;
495 handles.A = A;
496 handles.B = B;
497 kbetal = divisorswithoddresult(handles.N);
498 handles.kbetal = kbetal;
499 handles.kalphal = signsvectorcoeffs(kbetal);
500 handles.multipliers = multipliers(handles.N);
501
502 % Update handles structure
503 guidata(hObject, handles);
504
505 end
506
507 % --------------------------------------------------------------------
508 function audio_Callback(hObject, eventdata, handles)
509 % hObject    handle to audio (see GCBO)
510 % eventdata  reserved - to be defined in a future version of MATLAB
511 % handles    structure with handles and user data (see GUIDATA)
512
513 end
514
```

```
515 %--------------------------------------------------------------------
516 function analyses_2D_Callback(hObject, eventdata, handles)
517 % hObject    handle to analyses_2D (see GCBO)
518 % eventdata  reserved - to be defined in a future version of MATLAB
519 % handles    structure with handles and user data (see GUIDATA)
520
521 end
522
523 % --- Executes on key press with focus on listbox1 and none of its
        controls.
524 function listbox1_KeyPressFcn(hObject, eventdata, handles)
525 % hObject    handle to listbox1 (see GCBO)
526 % eventdata  structure with the following fields (see UICONTROL)
527 %   Key: name of the key that was pressed, in lower case
528 %   Character: character interpretation of the key(s) that was pressed
529 %   Modifier: name(s) of the modifier key(s) (i.e., control, shift)
        pressed
530 % handles    structure with handles and user data (see GUIDATA)
531
532 end
533
534 %--------------------------------------------------------------------
535 function seriefouriercanonica_2D_Callback(hObject, eventdata, handles)
536 % hObject    handle to seriefouriercanonica_2D (see GCBO)
537 % eventdata  reserved - to be defined in a future version of MATLAB
538 % handles    structure with handles and user data (see GUIDATA)
539
540 end
541
542 %--------------------------------------------------------------------
543 function seriefourierquantizada_2D_Callback(hObject, eventdata,
        handles)
544 % hObject    handle to seriefourierquantizada_2D (see GCBO)
545 % eventdata  reserved - to be defined in a future version of MATLAB
546 % handles    structure with handles and user data (see GUIDATA)
547
548 end
549
550 %--------------------------------------------------------------------
551 function coeficientefourier_2D_Callback(hObject, eventdata, handles)
552 % hObject    handle to coeficientefourier_2D (see GCBO)
553 % eventdata  reserved - to be defined in a future version of MATLAB
554 % handles    structure with handles and user data (see GUIDATA)
555
556 end
557
558 %--------------------------------------------------------------------
559 function coeficientequantizada_2D_Callback(hObject, eventdata, handles
        )
560 % hObject    handle to coeficientequantizada_2D (see GCBO)
561 % eventdata  reserved - to be defined in a future version of MATLAB
562 % handles    structure with handles and user data (see GUIDATA)
```

```
563
564 end
565
566 % --------------------------------------------------------------------
567 function deteccao_2D_Callback(hObject, eventdata, handles)
568 % hObject    handle to deteccao_2D (see GCBO)
569 % eventdata  reserved - to be defined in a future version of MATLAB
570 % handles    structure with handles and user data (see GUIDATA)
571
572 end
573
574 function edit3_Callback(hObject, eventdata, handles)
575 % hObject    handle to edit3 (see GCBO)
576 % eventdata  reserved - to be defined in a future version of MATLAB
577 % handles    structure with handles and user data (see GUIDATA)
578
579 % Hints: get(hObject,'String') returns contents of edit3 as text
580 %        str2double(get(hObject,'String')) returns contents of edit3
    as a double
581
582 handles.prec = str2num(get(hObject, 'String'));
583
584 % Update handles structure
585 guidata(hObject, handles);
586
587 end
588
589 % --- Executes during object creation, after setting all properties.
590 function edit3_CreateFcn(hObject, eventdata, handles)
591 % hObject    handle to edit3 (see GCBO)
592 % eventdata  reserved - to be defined in a future version of MATLAB
593 % handles    empty - handles not created until after all CreateFcns
    called
594
595 % Hint: edit controls usually have a white background on Windows.
596 %       See ISPC and COMPUTER.
597 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
598     set(hObject,'BackgroundColor','white');
599 end
600
601 end
602
603 % --------------------------------------------------------------------
604 function comparison_Callback(hObject, eventdata, handles)
605 % hObject    handle to comparison (see GCBO)
606 % eventdata  reserved - to be defined in a future version of MATLAB
607 % handles    structure with handles and user data (see GUIDATA)
608
609 pergunta = 'Deseja comparar o resultado obtido a partir da s rie de
    Fourier Quantizada com a s rie de Fourier Cannica?';
610 comp = questdlg(pergunta, ...
```

```
611         'Ativa de Compara', 'Sim', 'No', 'No');
612
613     if(strcmp('Sim', comp))
614         %Activate comparison variable
615         handles.sfplot = true;
616         %Turn on the axes1 visibility
617         set(handles.axes1,'Visible', 'on');
618         set(handles.axes2, 'Position', [24 11.4 60 9]);
619     else
620         %Deactivate comparison variable
621         handles.sfplot = false;
622         %Clear and Turn off the axes1 visibility
623         cla reset;
624         set(handles.axes1,'Visible', 'off');
625         set(handles.axes2, 'Position', [18.6 15 71.6 15.23]);
626     end
627
628 % Update handles structure
629 guidata(hObject, handles);
630
631 end
632
633 % --------------------------------------------------------------------
634 function save_Callback(hObject, eventdata, handles)
635 % hObject    handle to save (see GCBO)
636 % eventdata  reserved - to be defined in a future version of MATLAB
637 % handles    structure with handles and user data (see GUIDATA)
638
639 %file = inputdlg({'Nome do arquivo:'}, 'Salvar Anlise', [1], {'Nome
        do arquivo sem extenso'});
640 [file, pathname] = uiputfile('Default.mat', 'Salvar Anlise');
641
642 if(file == 0)
643     return;
644 else
645     filetobesaved = strcat(pathname, file);
646
647     toplot2 = handles.toplot2;
648     n = handles.n;
649     t = handles.t;
650
651     if(handles.discrete == true)
652         save(filetobesaved, 'toplot2', 'n');
653     else
654         save(filetobesaved, 'toplot2', 't');
655     end
656 end
657
658 end
659
660 % --- Executes on button press in pushbutton4.
661 function pushbutton4_Callback(hObject, eventdata, handles)
```

```
662 % hObject    handle to pushbutton4 (see GCBO)
663 % eventdata  reserved - to be defined in a future version of MATLAB
664 % handles    structure with handles and user data (see GUIDATA)
665
666 [file, pathname] = uigetfile('*.mat', 'Arquivo para Anlise');
667
668 if(file ~= 0)
669     file = strcat(pathname, file);
670     set(handles.edit1, 'String', file);
671     handles.file = file;
672     % Update handles structure
673     guidata(hObject, handles);
674 end
675
676 end
677
678 % ---------------------------------------------------------------------
679 function configs_Callback(hObject, eventdata, handles)
680 % hObject    handle to configs (see GCBO)
681 % eventdata  reserved - to be defined in a future version of MATLAB
682 % handles    structure with handles and user data (see GUIDATA)
683
684 configs = inputdlg({'Informe o nmero de harmnicos', ...
685     'Informe o comprimento de cada bloco', ...
686     'Informe a preciso usada para encontrar a matriz aproximada para
        detec'}, ...
687     'Configuraes para Anlise', [1 1 1], {'Deve ser um inteiro maior
        que 0', ...
688     'Deve ser um inteiro maior que 0', '10^(-2)'});
689
690 if(~isempty(configs))
691     N = str2num(configs{1});
692     [A B] = mtrxgen_signal(N);
693
694     if(handles.flag == 5)
695         [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
696         handles.scale = scaleF;
697         handles.multipliers = multipliers(N);
698     end
699
700     handles.A = A;
701     handles.B = B;
702     handles.N = N;
703     handles.lngth = str2num(configs{2});
704     handles.prec = str2num(configs{3});
705     kbetal = divisorswithoddresult(N);
706     handles.kbetal = kbetal;
707     handles.kalphal = signsvectorcoeffs(kbetal);
708
709     % Update handles structure
710     guidata(hObject, handles);
711
```

```
712     set(handles.edit2, 'String', num2str(handles.N));
713     set(handles.edit4, 'String', num2str(handles.lngth));
714     set(handles.edit3, 'String', num2str(handles.prec));
715 end
716
717 end
718
719 function edit4_Callback(hObject, eventdata, handles)
720 % hObject    handle to edit4 (see GCBO)
721 % eventdata  reserved - to be defined in a future version of MATLAB
722 % handles    structure with handles and user data (see GUIDATA)
723
724 % Hints: get(hObject,'String') returns contents of edit4 as text
725 %        str2double(get(hObject,'String')) returns contents of edit4
        as a double
726
727 end
728
729 % --- Executes during object creation, after setting all properties.
730 function edit4_CreateFcn(hObject, eventdata, handles)
731 % hObject    handle to edit4 (see GCBO)
732 % eventdata  reserved - to be defined in a future version of MATLAB
733 % handles    empty - handles not created until after all CreateFcns
        called
734
735 % Hint: edit controls usually have a white background on Windows.
736 %       See ISPC and COMPUTER.
737 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
        defaultUicontrolBackgroundColor'))
738     set(hObject,'BackgroundColor','white');
739 end
740
741 end
742
743 % --------------------------------------------------------------------
744 function exemplos_Callback(hObject, eventdata, handles)
745 % hObject    handle to exemplos (see GCBO)
746 % eventdata  reserved - to be defined in a future version of MATLAB
747 % handles    structure with handles and user data (see GUIDATA)
748
749 end
750
751 % --------------------------------------------------------------------
752 function parabola_Callback(hObject, eventdata, handles)
753 % hObject    handle to parabola (see GCBO)
754 % eventdata  reserved - to be defined in a future version of MATLAB
755 % handles    structure with handles and user data (see GUIDATA)
756
757 file = 'examples/parabola';
758
759 set(handles.edit1, 'String', file);
760
```

```
761 handles.file = file;
762 % Update handles structure
763 guidata(hObject, handles);
764
765 %configs_Callback(hObject, eventdata, handles);
766
767 N = 5;
768 [A B] = mtrxgen_signal(N);
769
770 if(handles.flag == 5)
771     [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
772     handles.scale = scaleF;
773 end
774
775 handles.A = A;
776 handles.B = B;
777 handles.N = N;
778 handles.lngth = str2num('');
779 handles.prec = 10^(-2);
780 kbetal = divisorswithoddresult(N);
781 handles.kbetal = kbetal;
782 handles.kalphal = signsvectorcoeffs(kbetal);
783
784 % Update handles structure
785 guidata(hObject, handles);
786
787 set(handles.edit2, 'String', num2str(handles.N));
788 set(handles.edit4, 'String', num2str(handles.lngth));
789 set(handles.edit3, 'String', num2str(handles.prec));
790
791 pushbutton1_Callback(hObject, eventdata, handles);
792 str1 = 'Este exemplo foi construdo tendo como sinal de entrada para
      anlise um sinal';
793 str2 = ' parablico, de 0 at 1, com perodo de fundamental de 0.5,
    com 1001 amostras. Na anlise foi considerado';
794 str3 = ' at o quinto harmnico, sem quebra em blocos. O grfico na
    parte superior';
795 str4 = ' representa a srie de Fourier, e';
796 str5 = ' o grfico da parte inferior representa a srie de Fourier
    quantizada.';
797 str6 = ' Caso deseje alterar as configuraes realizar outras anlise
    com este';
798 str7 = ' mesmo sinal, altere as configuraes clicando no boto de
     configuraes na barra de menu na parte superior';
799 str8 = ' da janela. Caso deseje obter os coeficientes, quer sejam de
    Fourier, da';
800 str9 = ' srie quantizada, ou apenas realizar uma estima dos
    coeficientes, clique';
801 str10 = ' no boto de anlise -1D na barra de menu, na parte superior
    da janela.';
802 str11 = 'O perodo considerado foi 0.5 para que se tornasse evidente a
     peiodicidade do sinal.';
```

```
803 msgbox(strcat(str1, str2, str3, str4, str5, str6, str7, str8, str9,
        str10, str11)...
804      , 'Informativo sobre exemplo de sinal parablico');
805
806 end
807
808 % --------------------------------------------------------------------
809 function cosseno_Callback(hObject, eventdata, handles)
810 % hObject    handle to cosseno (see GCBO)
811 % eventdata  reserved - to be defined in a future version of MATLAB
812 % handles    structure with handles and user data (see GUIDATA)
813
814 file = 'examples/cosseno';
815
816 set(handles.edit1, 'String', file);
817
818 handles.file = file;
819 % Update handles structure
820 guidata(hObject, handles);
821
822 %configs_Callback(hObject, eventdata, handles);
823
824 N = 5;
825 [A B] = mtrxgen_signal(N);
826
827 if(handles.flag == 5)
828     [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
829     handles.scale = scaleF;
830 end
831
832 handles.A = A;
833 handles.B = B;
834 handles.N = N;
835 handles.lngth = str2num('');
836 handles.prec = 10^(-2);
837 kbetal = divisorswithoddresult(N);
838 handles.kbetal = kbetal;
839 handles.kalphal = signsvectorcoeffs(kbetal);
840 % Update handles structure
841 guidata(hObject, handles);
842
843 set(handles.edit2, 'String', num2str(handles.N));
844 set(handles.edit4, 'String', num2str(handles.lngth));
845 set(handles.edit3, 'String', num2str(handles.prec));
846
847 pushbutton1_Callback(hObject, eventdata, handles);
848 str1 = 'Este exemplo foi construdo tendo como sinal de entrada para
        anlise um sinal';
849 str2 = ' cosseno, de 0 at 2*pi, com 629 amostras. Na anlise foi
        considerado';
850 str3 = ' at o quinto harmnico, sem quebra em blocos. O grfico na
        parte superior';
```

```
851 str4 = ' representa a  s rie  de Fourier, e';
852 str5 = ' o  g r fico  da parte inferior representa a  s rie  de Fourier
    quantizada.';
853 str6 = ' Caso deseje alterar as configuraes realizar outras  a n lise
    com este';
854 str7 = ' mesmo sinal, altere as configuraes clicando no boto de
    configuraes na barra de menu na parte superior';
855 str8 = ' da janela. Caso deseje obter os coeficientes, quer sejam de
    Fourier, da';
856 str9 = '  s rie  quantizada, ou apenas realizar uma estima  dos
    coeficientes, clique';
857 str10 = ' no boto de  a n lise -1D na barra de menu, na parte superior
    da janela.';
858 msgbox(strcat(str1, str2, str3, str4, str5, str6, str7, str8, str9,
    str10)...
859     , 'Informativo sobre exemplo de sinal cosseno');
860
861 end
862
863 % --------------------------------------------------------------------
864 function sinaldeaudio_Callback(hObject, eventdata, handles)
865 % hObject    handle to sinaldeaudio (see GCBO)
866 % eventdata  reserved - to be defined in a future version of MATLAB
867 % handles    structure with handles and user data (see GUIDATA)
868
869 file = 'examples/aleluia';
870
871 set(handles.edit1, 'String', file);
872
873 handles.file = file;
874 % Update handles structure
875 guidata(hObject, handles);
876
877 %configs_Callback(hObject, eventdata, handles);
878
879 N = 5;
880 [A B] = mtrxgen_signal(N);
881
882 if(handles.flag == 5)
883     [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
884     handles.scale = scaleF;
885 end
886
887 handles.A = A;
888 handles.B = B;
889 handles.N = N;
890 handles.lngth = str2num('64');
891 handles.prec = 10^(-2);
892 kbetal = divisorswithoddresult(N);
893 handles.kbetal = kbetal;
894 handles.kalphal = signsvectorcoeffs(kbetal);
895
```

```
896 % Update handles structure
897 guidata(hObject, handles);
898
899 set(handles.edit2, 'String', num2str(handles.N));
900 set(handles.edit4, 'String', num2str(handles.lngth));
901 set(handles.edit3, 'String', num2str(handles.prec));
902
903 pushbutton1_Callback(hObject, eventdata, handles);
904 str1 = 'Este exemplo foi construdo tendo como sinal de entrada para
        anlise um sinal';
905 str2 = ' de audio, amostrado com 8192Hz, com 73.113 amostras. Na
        anlise foi considerado';
906 str3 = ' at o quinto harmnico, com o sinal quebrado em blocos com 64
        amostras de comprimento. O grfico na parte superior';
907 str4 = ' representa a srie de Fourier, e';
908 str5 = ' o grfico da parte inferior representa a srie de Fourier
       quantizada.';
909 str6 = ' Caso deseje alterar as configuraes realizar outras anlise
       com este';
910 str7 = ' mesmo sinal, altere as configuraes clicando no boto de
       configuraes na barra de menu na parte superior';
911 str8 = ' da janela. Caso deseje obter os coeficientes, quer sejam de
       Fourier, da';
912 str9 = ' srie quantizada, ou apenas realizar uma estima dos
       coeficientes, clique';
913 str10 = ' no boto de anlise-1D na barra de menu, na parte superior
       da janela.';
914 msgbox(strcat(str1, str2, str3, str4, str5, str6, str7, str8, str9,
       str10)...
915      , 'Informativo sobre exemplo de sinal de audio');
916
917 end
918
919 % --------------------------------------------------------------------
920 function gaussiana_Callback(hObject, eventdata, handles)
921 % hObject    handle to gaussiana (see GCBO)
922 % eventdata  reserved - to be defined in a future version of MATLAB
923 % handles    structure with handles and user data (see GUIDATA)
924
925 file = 'examples/gaussiana';
926
927 set(handles.edit1, 'String', file);
928
929 handles.file = file;
930 % Update handles structure
931 guidata(hObject, handles);
932
933 %configs_Callback(hObject, eventdata, handles);
934
935 N = 5;
936 [A B] = mtrxgen_signal(N);
937
```

```
938 if(handles.flag == 5)
939     [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
940     handles.scale = scaleF;
941 end
942
943 handles.A = A;
944 handles.B = B;
945 handles.N = N;
946 handles.lngth = str2num('');
947 handles.prec = 10^(-2);
948 kbetal = divisorswithoddresult(N);
949 handles.kbetal = kbetal;
950 handles.kalphal = signsvectorcoeffs(kbetal);
951
952 % Update handles structure
953 guidata(hObject, handles);
954
955 set(handles.edit2, 'String', num2str(handles.N));
956 set(handles.edit4, 'String', num2str(handles.lngth));
957 set(handles.edit3, 'String', num2str(handles.prec));
958
959 pushbutton1_Callback(hObject, eventdata, handles);
960 str1 = 'Este exemplo foi construdo tendo como sinal de entrada para
      anlise um sinal';
961 str2 = ' gaussiano, de 0 at 4, com perodo fundamental de 2, com 4001
      amostras. Na anlise foi considerado';
962 str3 = ' at o quinto harmnico, sem quebra em blocos. O grfico na
     parte superior';
963 str4 = ' representa a srie de Fourier, e';
964 str5 = ' o grfico da parte inferior representa a srie de Fourier
     quantizada.';
965 str6 = ' Caso deseje alterar as configuraes realizar outras anlise
     com este';
966 str7 = ' mesmo sinal, altere as configuraes clicando no boto de
     configuraes na barra de menu na parte superior';
967 str8 = ' da janela. Caso deseje obter os coeficientes, quer sejam de
     Fourier, da';
968 str9 = ' srie quantizada, ou apenas realizar uma estima dos
     coeficientes, clique';
969 str10 = ' no boto de anlise-1D na barra de menu, na parte superior
     da janela.';
970 str11 = 'O perodo considerado foi 2 para que se tornasse evidente a
     peiodicidade do sinal.';
971 msgbox(strcat(str1, str2, str3, str4, str5, str6, str7, str8, str9,
     str10, str11)...
972      , 'Informativo sobre exemplo de sinal gaussiano');
973
974 end
975
976 % --------------------------------------------------------------------
977 function rampa_Callback(hObject, eventdata, handles)
978 % hObject    handle to rampa (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


file = 'examples/rampa';

set(handles.edit1, 'String', file);

handles.file = file;
% Update handles structure
guidata(hObject, handles);

%configs_Callback(hObject, eventdata, handles);

N = 5;
[A B] = mtrxgen_signal(N);

if(handles.flag == 5)
    [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
    handles.scale = scaleF;
end

handles.A = A;
handles.B = B;
handles.N = N;
handles.lngth = str2num('');
handles.prec = 10^(-2);
kbetal = divisorswithoddresult(N);
handles.kbetal = kbetal;
handles.kalphal = signsvectorcoeffs(kbetal);

% Update handles structure
guidata(hObject, handles);

set(handles.edit2, 'String', num2str(handles.N));
set(handles.edit4, 'String', num2str(handles.lngth));
set(handles.edit3, 'String', num2str(handles.prec));

pushbutton1_Callback(hObject, eventdata, handles);
str1 = 'Este exemplo foi construdo tendo como sinal de entrada para
    anlise um sinal';
str2 = ' rampa, de 0 at 1, com perodo fundamental de 0.5, com 1001
    amostras. Na anlise foi considerado';
str3 = ' at o quinto harmnico, sem quebra em blocos. O grfico na
    parte superior';
str4 = ' representa a srie de Fourier, e';
str5 = ' o grfico da parte inferior representa a srie de Fourier
    quantizada.';
str6 = ' Caso deseje alterar as configuraes realizar outras anlise
    com este';
str7 = ' mesmo sinal, altere as configuraes clicando no boto de
    configuraes na barra de menu na parte superior';
```

```
1025 str8 = ' da janela. Caso deseje obter os coeficientes, quer sejam de
         Fourier, da';
1026 str9 = ' s rie quantizada, ou apenas realizar uma estima dos
         coeficientes, clique';
1027 str10 = ' no boto de an lise -1D na barra de menu, na parte superior
         da janela.';
1028 str11 = 'O perodo considerado foi 0.5 para que se tornasse evidente a
          peiodicidade do sinal.';
1029 msgbox(strcat(str1, str2, str3, str4, str5, str6, str7, str8, str9,
         str10, str11)...
1030      , 'Informativo sobre exemplo de sinal rampa');
1031
1032 end
1033
1034 %--------------------------------------------------------------------
1035 function seno_Callback(hObject, eventdata, handles)
1036 % hObject    handle to seno (see GCBO)
1037 % eventdata  reserved - to be defined in a future version of MATLAB
1038 % handles    structure with handles and user data (see GUIDATA)
1039
1040
1041 file = 'examples/seno';
1042
1043 set(handles.edit1, 'String', file);
1044
1045 handles.file = file;
1046 % Update handles structure
1047 guidata(hObject, handles);
1048
1049 %configs_Callback(hObject, eventdata, handles);
1050
1051 N = 5;
1052 [A B] = mtrxgen_signal(N);
1053
1054 if(handles.flag == 5)
1055     [A B scaleF MSEA MSEB] = scaleFactor(A, B, handles.prec);
1056     handles.scale = scaleF;
1057 end
1058
1059 handles.A = A;
1060 handles.B = B;
1061 handles.N = N;
1062 handles.lngth = str2num('');
1063 handles.prec = 10^(-2);
1064 kbetal = divisorswithoddresult(N);
1065 handles.kbetal = kbetal;
1066 handles.kalphal = signsvectorcoeffs(kbetal);
1067
1068 % Update handles structure
1069 guidata(hObject, handles);
1070
1071 set(handles.edit2, 'String', num2str(handles.N));
```

```
1072 set(handles.edit4, 'String', num2str(handles.lngth));
1073 set(handles.edit3, 'String', num2str(handles.prec));
1074
1075 pushbutton1_Callback(hObject, eventdata, handles);
1076 str1 = 'Este exemplo foi construdo tendo como sinal de entrada para
     anlise um sinal';
1077 str2 = ' seno, de 0 at 2*pi, com 629 amostras. Na anlise foi
    considerado';
1078 str3 = ' at o quinto harmnico, sem quebra em blocos. O grfico na
    parte superior';
1079 str4 = ' representa a srie de Fourier, e';
1080 str5 = ' o grfico da parte inferior representa a srie de Fourier
    quantizada.';
1081 str6 = ' Caso deseje alterar as configuraes realizar outras anlise
    com este';
1082 str7 = ' mesmo sinal, altere as configuraes clicando no boto de
    configuraes na barra de menu na parte superior';
1083 str8 = ' da janela. Caso deseje obter os coeficientes, quer sejam de
    Fourier, da';
1084 str9 = ' srie quantizada, ou apenas realizar uma estima dos
    coeficientes, clique';
1085 str10 = ' no boto de anlise -1D na barra de menu, na parte superior
    da janela.';
1086 msgbox(strcat(str1, str2, str3, str4, str5, str6, str7, str8, str9,
    str10)...
1087     , 'Informativo sobre exemplo de sinal seno');
1088
1089 end
```

2. Função que retorna as matrizes MA_N^{-1} e MB_N^{-1}.

```
1  function [A B] = mtrxgen_signal( N )
2  %MTRXGEN Return the matrxies MA_N and MB_N used to obtain the Fourier-
       like
3  %coefficients, using the kernel sign(cos) and sign(sin).
4
5  %This function was created as a part of Senior Project of
6  %the author, Electronics Engineering student at Universidade
7  %Federal de Pernambuco, Brazil.
8
9  %Author: Diego F. G. Coelho
10 %September 08/2012
11
12 %Change Log:
13
14
15 %Requirements:
16 %   1)The vector f and t must have the same size. This requirement is
17 %   imposed by the use of vectorized product '.*'.
18
19 %Sanity check:
20 %The input argumet N must be a non-negative integer.
```

```
21
22 if(N < 0)
23     disp('Error! There is no negative frequency.');
24     return;
25 end
26
27 %The matrix of An,k's and Bn,k's:
28
29 A = [];
30 B = [];
31
32 %Wait box
33 h = waitbar(0, 'Por favor, espere, as matrizes MA_N^{-1} e MB_N^{-1}
       esto sendo geradas  ...');
34
35 for n = 1:N
36     for k = 1:N
37         A(n,k) = chifracsign(n,k);
38         B(n,k) = chifrac(n,k);
39     end
40     p = n/N;
41     waitbar(p, h);
42 end
43
44 A = inv(A);
45 B = inv(B);
46
47 close(h);
48 end
```

3. Função que retorna os coeficientes da série de Fourier Quantizada.

```
1  function [coeffCos coeffSin] = sf_signal_coeff( f, t, N, A, B)
2  %SF_SIGNAL Return the Fourier-like serie of f expanded using the
       signal
3  %function of sin and cos function as kernel function.
4  %   Detailed explanation goes here
5  %This function return the Fourier-like serie expanded using the signal
6  %function. The coefficients are returned in a matrix which the lines
7  %contain the coeficients of the expansion due the signal(cos) and
8  %signal(sin) kernels, respectively. The
9  %input arguments are the function f to be expanded, the time vector t,
10 %which are both row vectors and the integer harmonic parameter N that
11 %represent the Nth harmonic in what the representation should be
       trucated.
12 %Also the matrixes A and B that represent MA_N^(-1) and MB_N^(-1)
13 %respectively, area passed by as input arguments.
14
15 %This functions was created as a part of Senior Project of
16 %the author, Electronics Engineering student at Universidade
17 %Federal de Pernambuco, Brazil.
18
```

```
19 %Author: Diego F. G. Coelho
20 %June 04/2012
21
22 %Change Log:
23 %August 30/2012: It does not work. Theoretical errors.
24 %September 06/2012:Theoretical erros corrected. It is working properly
       .
25
26 %Requirements:
27 %   1)The vector f and t must have the same size. This requirement is
28 %   imposed by the use of vectorized product '.*'.
29
30 %Sanity check:
31 %Verify if N is greater is positive. N can be zero, it means that the
       user
32 %is interested in the mean.
33
34 if(N < 0)
35     disp('Error! There is no negative frequency.');
36     return;
37 end
38
39 %The coeffcients vector due singal(cos) adn signal(sin) repectively:
40 coeffCos = [];
41 coeffSin = [];
42
43 %The inner product values:
44 fsigncos = [];
45 fsignsin = [];
46
47 %The fundamental frequency:
48 omega_0 = 2*(pi/(t(end)-t(1)));
49
50 %Frequency-time product:
51 freq_time = omega_0.*t;
52
53 %This calculation of inner product use the definition given by D. F.
       Souza,
54 %R. Cintra and de Oliveira. Observe that it is multiplied by 1/T,
       where T is
55 %the fundamental period, not by 2/T.
56
57 %Wait box
58 h = waitbar(0, 'Por favor, espere, os coeficientes esto sendo
       calculados ...');
59
60 %These are column vectors.
61 for k = 1:N
62     fsigncos = [fsigncos; trapz(t,f.* sign(cos(k.*freq_time)))];
63     fsignsin = [fsignsin; trapz(t,f.* sign(sin(k.*freq_time)))];
64     %Wait box
65     p = k/N;
```

```
66      waitbar(p, h);
67  end
68
69  fsigncos = (omega_0/(2*pi))*fsigncos;
70  fsignsin = (omega_0/(2*pi))*fsignsin;
71
72  %This solve the linear system in order to find the fourier-like
        coeffcients
73  %using the base sign(.)
74  coeffCos = (A*fsigncos)';
75  %This include the 0th harmonic.
76  coeffCos = [(omega_0/(2*pi))*trapz(t,f), coeffCos];
77  %coeffCos = [(omega_0/(2*pi))*trapz(t,f), fsigncos'];
78
79  coeffSin = (B*fsignsin)';
80  %This include the 0th harmonic.
81  coeffSin = [0, coeffSin];
82  %coeffSin = [0, fsignsin'];
83
84  close(h);
85
86  end
```

4. Função que retorna os coeficientes da série de Fourier Aproximada.

```
1   function [ coeffCos coeffSin ] = sig2sf_coeff( coeffSigCos,
        coeffSigSin )
2   %SIG2SF Summary of this function goes here
3   %   Detailed explanation goes here
4   %This function perform the convertion from the coeffcients of Fourier-
        like
5   %serie using sign(cos) and sign(sin) functions as kernel to Fourier
6   %serie coefficients.The input arguments coeffSigCos and coeffSigSin
        are row
7   %vector which contain the coefficients of Fourier-like serie using
8   %sign(cos) and sign(sin) functions as kernel, respectively, and
        coeffCos
9   %and coeffSin are row vectors which contains the Fourier coefficients
        using
10  %cos and sin functions as kernel.
11
12  %This functions was created as a part of Senior Project of
13  %the author, Electronics Engineering student at Universidade
14  %Federal de Pernambuco, Brazil.
15
16  %Author: Diego F. G. Coelho
17  %June 04/2012
18
19  %Change Log:
20  %June 04/2012: It does not work properly.
21  %September 06/2012: The problem was due theoretical problems. It is
        fixed.
```

```
%It is working properly.

%Sanity check:
%The coefficients must have the same length

if(length(coeffSigCos)~=length(coeffSigSin))
    disp('Error! The coeffcients vector must have the same length.');
    return;
end

%The coefficients of Fourier serie:
coeffCos = [];
coeffSin = [];

%Wait box
h = waitbar(0, 'Por favor, espere, os coeficientes esto sendo
    convertidos ...');

for k = 1:(length(coeffSigCos)-1)
    a_k = 0;
    b_k = 0;
    for n = 1:k
        if(mod(k,n)==0)
            if(mod(k/n,2)==1)
                frac = k/n;
                a_k = a_k + coeffSigCos(n+1)*(((-1)^((frac-1)/2))/frac
                    );
                b_k = b_k + coeffSigSin(n+1)/frac;
            end
        end
    end
    coeffCos = [coeffCos, a_k];
    coeffSin = [coeffSin, b_k];
    %Wait box
    p = k/(length(coeffSigCos)-1);
    waitbar(p, h);
end

coeffCos = (4/pi).*coeffCos;
coeffSin = (4/pi).*coeffSin;

%The mean value is the same and the first value with respect to sin is
%always zero:

coeffCos = [coeffSigCos(1), coeffCos];
coeffSin = [coeffSigSin(1), coeffSin];
close(h);
end
```

5. Função que retorna a representação de um sinal em série de Fourier clássica usando blocos de curto comprimento.

```
function  f_sf = smallblocklengthsf( f, t, N, lngth )
%SMALLBLOCKLENGTHSF calculate the Fourier Serie of a function f(t)
    with N
%harmonics breaking the sequence in small blocks of length lngth.

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Sep 25/2012

%Sanity Check:
%1) The input arfument lngth must be greater than zero
if(lngth <= 0)
    disp('Error" Te input argument lnght must be greater than zero.');
    return;
end

Nblocks = floor(length(f)/lngth);

f_sf = [];

%Wait box
h = waitbar(0, 'Por favor, espere, a s rie de Fourier est sendo
    calculada em blocos ...');

for i = 1:Nblocks-1
   [coeffCos coeffSin] = sf_coeff_nwaitbar(f((i-1)*lngth+1:i*lngth), t
       ((i-1)*lngth+1:i*lngth), N);
   f_sf = [f_sf, sf(coeffCos, coeffSin, t((i-1)*lngth+1:i*lngth))];
   %Wait box
    p = i/Nblocks;
    waitbar(p, h);
end

[coeffCos coeffSin] = sf_coeff_nwaitbar(f((Nblocks-1)*lngth+1:end), t
    ((Nblocks-1)*lngth+1:end), N);
f_sf = [f_sf, sf(coeffCos, coeffSin, t((Nblocks-1)*lngth+1:end))];
close(h);
```

6. Função que retorna a representação de um sinal em série de Fourier Quantizada usando blocos de curto comprimento.

```
function  f_sf = smallblocklengthsignal( f, t, N, lngth, A, B)
%SMALLBLOCKLENGTHSIGNAL calculate the Fourier like-Serie of a function
    f(t)
%with N harmonics breaking the sequence in small blocks of length
    lngth
%using the sign kernel.

```

```
%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Sep 25/2012

%Sanity Check:
%1) The input argument lngth must be greater than zero
if(lngth <= 0)
    disp('Error" Te input argument lnght must be greater than zero.');
    return;
end

Nblocks = floor(length(f)/lngth);

f_sf = [];

%Wait box
h = waitbar(0, 'Por favor, espere, a s rie quantizada est sendo
    calculada em blocos ...');

for i = 1:(Nblocks-1)
   [coeffSigCos coeffSigSin] = sf_signal_coeff_nwaitbar(f((i-1)*lngth
       +1:i*lngth), t((i-1)*lngth+1:i*lngth), N, A, B);
    f_sf = [f_sf, sf(coeffSigCos, coeffSigSin, t((i-1)*lngth+1:i*lngth)
       )];
   %Wait box
    p = i/Nblocks;
    waitbar(p, h);
end

[coeffSigCos coeffSigSin] = sf_signal_coeff_nwaitbar(f((Nblocks-1)*
    lngth+1:end), t((Nblocks-1)*lngth+1:end), N, A, B);
f_sf = [f_sf, sf(coeffSigCos, coeffSigSin, t((Nblocks-1)*lngth+1:end))
    ];

close(h);
```

7. Função que retorna a representação de um sinal em série de Fourier Aproximada usando blocos de curto comprimento.

```
function  f_sf = smallblocklengthsignalsf( f, t, N, lngth, A, B)
%SMALLBLOCKLENGTHSIGNALSF calculate the Fourier Serie of a function f(
    t)
%with N harmonics breaking the sequence in small blocks of length
    lngth
%using the sign kernel.

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.
```

```
9
10 %Author: Diego F. G. Coelho
11 %Sep 25/2012
12
13 %Sanity Check:
14 %1) The input arfument lngth must be greater than zero
15 if(lngth <= 0)
16     disp('Error" The input argument lnght must be greater than zero.')
           ;
17     return;
18 end
19
20 Nblocks = floor(length(f)/lngth);
21
22 coeffCos = [];
23 coeffSin = [];
24
25 f_sf = [];
26
27 %Wait box
28 h = waitbar(0, 'Por favor, espere, a  s rie  de Fourier aproximada est
       sendo calculada em blocos ...');
29
30 for i = 1:Nblocks-1
31    [coeffSigCos coeffSigSin] = sf_signal_coeff_nwaitbar(f((i-1)*lngth
          +1:i*lngth), t((i-1)*lngth+1:i*lngth), N, A, B);
32    [coeffCos coeffSin] = sig2sf_coeff_nwaitbar(coeffSigCos,
          coeffSigSin);
33    f_sf = [f_sf, sf(coeffCos, coeffSin, t((i-1)*lngth+1:i*lngth))];
34    %Wait box
35     p = i/Nblocks;
36     waitbar(p, h);
37 end
38
39 [coeffSigCos coeffSigSin] = sf_signal_coeff_nwaitbar(f((Nblocks-1)*
       lngth+1:end), t((Nblocks-1)*lngth+1:end), N, A, B);
40 [coeffCos coeffSin] = sig2sf_coeff(coeffSigCos, coeffSigSin);
41 f_sf = [f_sf, sf(coeffCos, coeffSin, t((Nblocks-1)*lngth+1:end))];
42 close(h);
```

8. Função que retorna os coeficientes da série de Fourier clássica usando blocos de curto comprimento.

```
1 function  [coeffCos coeffSin] = smallblocklengthsfcoeff( f, t, N,
       lngth, A, B )
2 %SMALLBLOCKLENGTHSFCOEFF calculate the Fourier serie coefficients of a
3 %function f(t) with N harmonics breaking the sequence in small blocks
       of
4 %length lngth.
5
6 %This functions was created as a part of Senior Project of
7 %the author, Electronics Engineering student at Universidade
```

```
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Oct 18/2012

%Sanity Check:
%1) The input arfument lngth must be greater than zero
if(lngth <= 0)
    disp('Error" Te input argument lnght must be greater than zero.');
    return;
end

Nblocks = floor(length(f)/lngth);

coeffCos = [];
coeffSin = [];

for i = 1:(Nblocks-1)
   [coeffCosAux coeffSinAux] = sf_coeff(f((i-1)*lngth+1:i*lngth), t((i
       -1)*lngth+1:i*lngth), N);
   coeffCos = [coeffCos; coeffCosAux];
   coeffSin = [coeffSin; coeffSinAux];
end

[coeffCosAux coeffSinAux] = sf_coeff(f((Nblocks-1)*lngth+1:end), t((
    Nblocks-1)*lngth+1:end), N);
coeffCos = [coeffCos; coeffCosAux];
coeffSin = [coeffSin; coeffSinAux];

coeffCos = mean(coeffCos);
coeffSin = mean(coeffSin);
```

9. Função que retorna os coeficientes da série de Fourier Quantizada usando blocos de curto comprimento.

```
function [coeffSigCos coeffSigSin] = smallblocklengthsignalcoeff( f,
    t, N, lngth, A, B )
%SMALLBLOCKLENGTHSIGNALCOEFF calculate the Fourier-like serie
    coefficients
%of a function f(t) with N harmonics breaking the sequence in small
    blocks
%of length lngth, based on the base sign(.).

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Oct 18/2012

%Sanity Check:
%1) The input argument lngth must be greater than zero
```

```
if(lngth <= 0)
    disp('Error" Te input argument lnght must be greater than zero.');
    return;
end

Nblocks = floor(length(f)/lngth);

coeffSigCos = [];
coeffSigSin = [];

for i = 1:(Nblocks-1)
   [coeffSigCosAux coeffSigSinAux] = sf_signal_coeff(...
       f((i-1)*lngth+1:i*lngth), t((i-1)*lngth+1:i*lngth), N, A, B);
   coeffSigCos = [coeffSigCos; coeffSigCosAux];
   coeffSigSin = [coeffSigSin; coeffSigSinAux];

end

[coeffSigCosAux coeffSigSinAux] = sf_signal_coeff(...
    f((Nblocks-1)*lngth+1:end), t((Nblocks-1)*lngth+1:end), N, A, B);
coeffSigCos = [coeffSigCos; coeffSigCosAux];
coeffSigSin = [coeffSigSin; coeffSigSinAux];

coeffSigCos = mean(coeffSigCos);
coeffSigSin = mean(coeffSigSin);
```

10. Função que retorna os coeficientes da série de Fourier Aproximada usando blocos de curto comprimento.

```
function [coeffCos coeffSin] = smallblocklengthsignalsfcoeff( f, t, N
    , lngth, A, B )
%SMALLBLOCKLENGTHSIGNALSFCOEFF calculate the Fourier serie
    coefficients of
%a function f(t) with N harmonics breaking the sequence in small
    blocks of
%length lngth using the sign kernel.

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Oct 18/2012

%Sanity Check:
%1) The input arfument lngth must be greater than zero
if(lngth <= 0)
    disp('Error" Te input argument lnght must be greater than zero.');
    return;
end

Nblocks = floor(length(f)/lngth);
```

```
21
22 coeffCos = [];
23 coeffSin = [];
24
25 for i = 1:(Nblocks-1)
26    [coeffSigCos coeffSigSin] = sf_signal_coeff(f((i-1)*lngth+1:i*lngth
       ),...
27        t((i-1)*lngth+1:i*lngth), N, A, B);
28    [coeffCosAux coeffSinAux] = sig2sf_coeff(coeffSigCos, coeffSigSin);
29    coeffCos = [coeffCos; coeffCosAux];
30    coeffSin = [coeffSin; coeffSinAux];
31 end
32
33 [coeffSigCos coeffSigSin] = sf_signal_coeff(f((Nblocks-1)*lngth+1:end)
     ,...
34     t((Nblocks-1)*lngth+1:end), N, A, B);
35 [coeffCosAux coeffSinAux] = sig2sf_coeff(coeffSigCos, coeffSigSin);
36 coeffCos = [coeffCos; coeffCosAux];
37 coeffSin = [coeffSin; coeffSinAux];
38
39 coeffCos = mean(coeffCos);
40 coeffSin = mean(coeffSin);
```

11. Função que retorna os coeficientes da série de Fourier Aproximada escalonados usando blocos de curto comprimento.

```
1  function [coeffCos coeffSin] = smallblocklengthsignalsfcoeffdetection
       (f,...
2      t, N, lngth, A, B, kalphal, kbetal, multipliers)
3  %SMALLBLOCKLENGTHSIGNALSFCOEFFDETECTION estimate the escaled Fourier
       serie
4  %coefficients of a function f(t) with N harmonics breaking the
       sequence in
5  %small blocks of length lngth using the sign kernel.
6
7  %This functions was created as a part of Senior Project of
8  %the author, Electronics Engineering student at Universidade
9  %Federal de Pernambuco, Brazil.
10
11 %Author: Diego F. G. Coelho
12 %Nov 13/2012
13
14 %Sanity Check:
15 %1) The input arfument lngth must be greater than zero
16 if(lngth <= 0)
17     disp('Error" Te input argument lnght must be greater than zero.');
18     return;
19 end
20
21 Nblocks = floor(length(f)/lngth);
22
23 coeffCos = [];
```

```
coeffSin = [];

for i = 1:(Nblocks-1)
   [coeffSigCos coeffSigSin] = sf_signal_coeff(f((i-1)*lngth+1:i*lngth),...
       t((i-1)*lngth+1:i*lngth), N, A, B);
   [coeffCosAux coeffSinAux] = sig2sf_coeff_alt_detection(coeffSigCos,...
       coeffSigSin, kalphal, kbetal, multipliers);
   coeffCos = [coeffCos; coeffCosAux];
   coeffSin = [coeffSin; coeffSinAux];
end

[coeffSigCos coeffSigSin] = sf_signal_coeff(f((Nblocks-1)*lngth+1:end),...
    t((Nblocks-1)*lngth+1:end), N, A, B);
[coeffCosAux coeffSinAux] = sig2sf_coeff_alt_detection(coeffSigCos, coeffSigSin,...
    kalphal, kbetal, multipliers);
coeffCos = [coeffCos; coeffCosAux];
coeffSin = [coeffSin; coeffSinAux];

coeffCos = mean(coeffCos);
coeffSin = mean(coeffSin);
```

12. Função que retorna as quantidades $N!/k$ para escalonamento e cômputo a priori dos coeficientes $\alpha_{(k/n-1)/2}$.

```
function divisors = divisorswithoddresult(v)
%DIVISORS returns a cell array in which each element is a row vector whose
%elements are the divisors of the index of the each row vector in the cell
%array which result in a odd number.



%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
%Oct 18/2012

%Change Log:
%Oct 18/2012:  Change the function name from divisors to
%divisorswithoddresult. This best decribe the role performed by this
%function within the simulations and spectral analyzer.

%Sanity check:
%Ensure that the input argument v is a integer number
v = round(v);
```

```
23
24 %The output argument
25 divisors = {};
26
27 for i = 1:v
28     aux = [];
29     for k = 1:i
30         if(mod(i, k) == 0)
31             if(mod(i/k, 2) == 1)
32                 aux = [aux, k];
33             end
34         end
35     end
36     divisors{i} = aux;
37 end
38
39 end
```

13. Função que retorna as quantidades $N!/k(-1)^{(k/n-1)/2}$ para escalonamento dos coeficientes $\beta_{(k/n-1)/2}$.

```
1  function signsvectorcoeffs = signsvectorcoeffs( div )
2  %SIGNSVECTORCOEFFS return a cell array in which each element is a row
3  %vector whose elements are the divisors of the index of the each row
       vector
4  %in the cell array which result in a odd number. The values of each
       element
5  %of each row vector is pondered by a (-1)^(k/n-1), where k is the
       position
6  %of the row vector in the cell array and n is the possition of the
       element
7  %in the row vector.
8
9  %The output correspond to the beta coefficients used to compute the
       Fourier
10 %coeffiicents escaled by a factor k, where k is a the index position
       of
11 %the row vector in the cell array.
12
13 %This functions was created as a part of Senior Project of
14 %the author, Electronics Engineering student at Universidade
15 %Federal de Pernambuco, Brazil.
16
17 %Author: Diego F. G. Coelho
18 %Oct 18/2012
19
20 %Change Log:
21 %Oct 18/2012:  Change the function name from divisors to
22 %divisorswithoddresult. This best decribe the role performed by this
23 %function within the simulations and spectral analyzer.
24
25 %Sanity check:
```

```
%There is no need

len = length(div);

signsvectorcoeffs = {};

for i = 1:len
    signsvectorcoeffs{i} = div{i}.*(-1).^((i./div{i}-1)/2);
end


end
```

14. Função que retorna os coeficientes da série de Fourier Aproximada escalonados.

```
function [ coeffCos coeffSin ] = sig2sf_coeff_alt_detection(
    coeffSigCos, coeffSigSin, kalphal, kbetal, multipliers )
%SIG2SF Summary of this function goes here
%   Detailed explanation goes here
%This function perform the convertion from the coeffcients of Fourier-
    like
%serie using sign(cos) and sign(sin) functions as kernel to Fourier
%serie coefficients.The input arguments coeffSigCos and coeffSigSin
    are row
%vector which contain the coefficients of Fourier-like serie using
%sign(cos) and sign(sin) functions as kernel, respectively, and
    coeffCos
%and coeffSin are row vectors which contains the Fourier coefficients
    using
%cos and sin functions as kernel. The input arguments kalphal and
    kbetal
%represent the alpha and beta coefficients escaled by k used to
    compute the
%Fourier series coefficients. In order to compute it correctly, the
%coefficients kalpha and kbeta must be divided by k, as done in this
    code.
%Due the fact that the coefficients kalpha and kbeta do not depende on
     the
%input signal tha tis being analyzed, they may be calculated
    previously
%during the cofiguration procedures. The iput argument multipliers s a
     row
%vector containg, at index n, the produtc between all natural number
    from 1
%to N (harmonics), except n. This is used to keep the proporcionality
%between the coefficients.

%This functions was created as a part of Senior Project of
%the author, Electronics Engineering student at Universidade
%Federal de Pernambuco, Brazil.

%Author: Diego F. G. Coelho
```

```
26 %Oct 18/2012
27
28 %Change Log:
29
30 %Sanity check:
31 %The coefficients must have the same length
32
33 if(length(coeffSigCos)~=length(coeffSigSin))
34     disp('Error! The coeffcients vector must have the same length.');
35     return;
36 end
37
38 %The coefficients of Fourier serie:
39 coeffCos = [];
40 coeffSin = [];
41
42 for k = 1:(length(coeffSigCos)-1)
43     coeffCos = [coeffCos, multipliers(k)*dot(coeffSigCos(1+kbetal{k}),
           kalphal{k})];
44     coeffSin = [coeffSin, multipliers(k)*dot(coeffSigSin(1+kbetal{k}),
           kbetal{k})];
45 end
46
47 %The mean value is the same and the first value with respect to sin is
48 %always zero:
49
50 %Multiply by pi/4*M, M = multipliers(end)*length(coeffCos), avoid
51 %discrepancy at the zero frequency component. Fortunatelly M is equals
       to
52 %the factorial(N-1), where N is the total (including 0) number of
53 %harmonics, since length(coeffCos) is the number of harmonics (
       excepting 0).
54 p = (pi/4)*(multipliers(end)*length(coeffCos));
55
56 coeffCos = [p*coeffSigCos(1), coeffCos];
57 coeffSin = [p*coeffSigSin(1), coeffSin];
58
59 end
```

APÊNDICE C

Apêndice de Imagens do Software Produzido

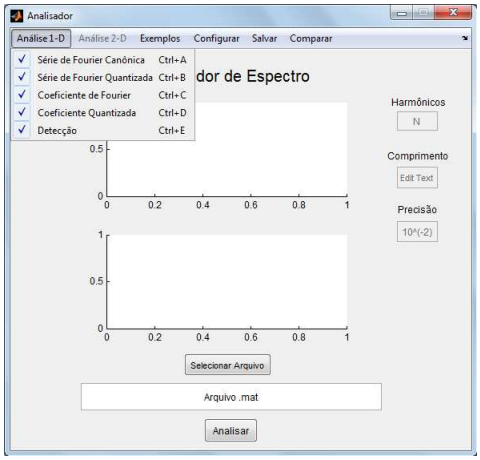


Figura C.1 Imagem do Analisador de Espectro ao ser iniciado.

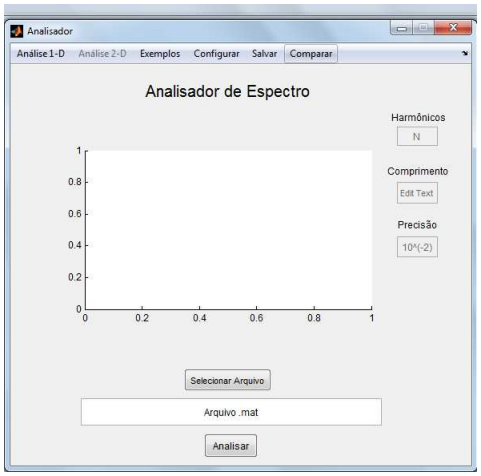


Figura C.2 Imagem do Analisador de Espectro ao ser usado no modo sem comparação.

Figura C.3 Imagem do Analisador de Espectro ao ser configurado.

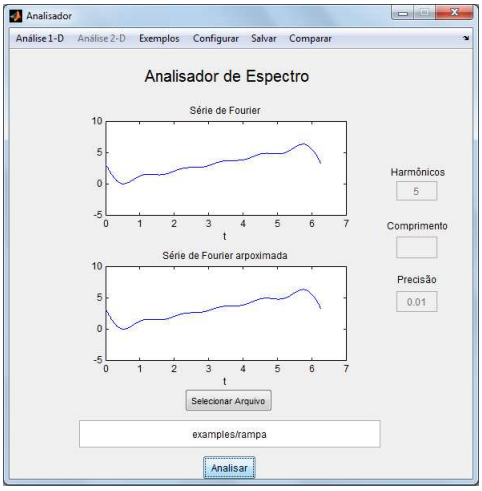


Figura C.4 Imagem do Analisador de Espectro analisando um sinal rampa com 5 harmônicos.

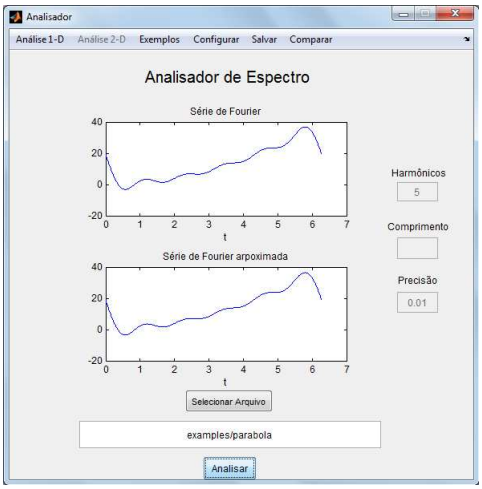


Figura C.5 Imagem do Analisador de Espectro um sinal parabólico com 5 harmônicos.

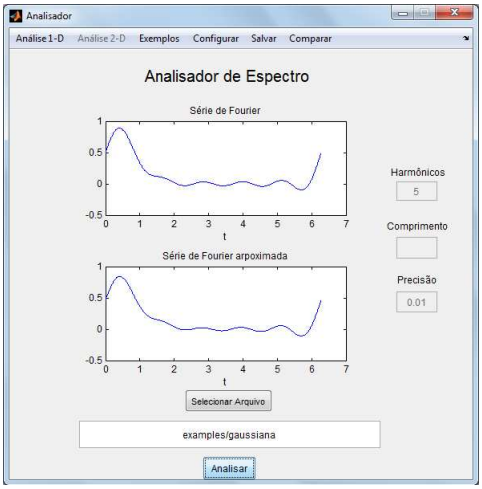


Figura C.6 Imagem do Analisador de Espectro um sinal gaussiano com 5 harmônicos.

APÊNDICE D

Aplicações da Série de Fourier em Equações Diferenciais

A série de Fourier surgiu com o estudo de ondas de calor feito Joseph Fourier (1768-1830) publicado inicialmente em *Mémoire sur la propagation de la chaleur dans les corps solides* (Tratado da propagação de calor em corpos rígidos) em 1807 e *Théorie analytique de la chaleur* (Teoria analítica do calor) em 1822 [17]. Embora a série de Fourier tenha sido motivada originalmente pelo estudo da resolução da equação do calor, ficou evidente posteriormente que as mesmas técnicas poderiam ser aplicadas a uma grande variedade de problemas matemáticos e físicos, especialmente aqueles envolvendo equações diferenciais ordinárias com coeficientes constantes, para a qual as soluções envolvam ondas sinusoidais. Dentre essas aplicações, podemos citar [16]:

1. Vibração em cordas tensionadas;
2. Vibrações em membranas;
3. Vibrações em colunas de ar;
4. Ondas em fluidos incompressíveis;
5. Ondas mecânicas em meios elásticos;
6. Ondas eletromagnéticas.

As equações de tais problemas comumente envolvem um operador linear diferencial conhecido como Laplaciano [16]:

$$\nabla^2 u(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{n} \frac{\partial^2 u(x_1, x_2, \ldots, x_n)}{\partial x_k^2},$$

em que u é uma função de variáveis reais $x_1, \ldots, x_n$ de classe $C^{(2)}$ [3].

O problema do estudo de ondas de calor em corpos rígidos se baseia na solução da equação

$$\frac{\partial u(t, \underline{\mathbf{x}})}{\partial t} = k\nabla^2 u(t, \underline{\mathbf{x}}),$$

em que u é uma função de quatro variáveis, o tempo t e posição $\underline{\mathbf{x}} = \{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$, e k é uma constante real. Semelhantemente a este problema, o estudo da propagação de ondas, em geral, são modelados por equações do tipo

$$\frac{\partial^2 u(t, \underline{\mathbf{x}})}{\partial t^2} = c^2\nabla^2 u(t, \underline{\mathbf{x}}).$$

As soluções mais gerais de tais equações são descritas como combinação linear de funções sinusoidais dependentes do tempo t e do espaço $\underline{\mathbf{x}}$

$$u(t,\mathbf{x}) = \sum_{i=0}^{n} a_i \phi(t,\underline{\mathbf{x}})_i,$$

em que cada $\phi(t,\underline{\mathbf{x}})_i$ é uma função contínua de parâmetros t e $\underline{\mathbf{x}}$.

Um caso importante das equações mencionadas é quando o problema não depende do tempo, dando origem ao problema da equação de Laplace:

$$\nabla^2 u(t,\underline{\mathbf{x}}) = 0.$$

Um dos mais notórios problemas que são resolvidos usando a equação de Laplace é o problema de Dirichlet [36], em que consiste em resolver a equação de Laplace para a função u no interior de uma dada região D sujeita ao vínculo de que $u = f$ para a fronteira de D [36]. Outro importante problema é o de, ao invés de especificar u no contorno de D, especificar-se $\nabla u \cdot \mathbf{n}$ no contorno de D, em que $\mathbf{n}$ representa o vetor ortonormal ao contorno de D. Tal problema é conhecido como problema de Von Neumann [6, 37].

Todos esse problemas são casos restritos de um problema mais geral, que é equação fundamental da física quântica [20], a equação de onda de *Schrödinger*:

$$-\frac{\hbar^2}{2m}\nabla^2 u + V(\mathbf{x})u = Eu,$$

em que $\hbar$ é a constante de Planck normalizada [20], m é a massa da partícula, $V(\mathbf{x})$ é a energia potencial da partícula em função da posição e E é a energia da partícula.

APÊNDICE E

Digressão sobre Séries Infinitas

Para analisar a série de Fourier é necessário ter bem fundamentado alguns conceitos sobre o estudo de séries. Um elemento fundamental no estudo das séries são as sequências. Séries são construídas com base em sequências. No caso em questão, é de interesse analisar séries cujos elementos estejam sobre o conjunto dos reais. Para tanto, pode-se definir uma sequência da seguinte forma.

Definição 8. *Seja $S = \{n \in \mathbb{Z} | n \geq m\}$ tal que $m \in \mathbb{Z}$. Uma sequência infinita de números reais é uma função*

$$\begin{aligned} S &\to \mathbb{R} \\ n &\mapsto s_n. \end{aligned} \tag{E.1}$$

Usualmente toma-se $m = \{0, 1\}$, contudo, no caso em estudo, será aceito que m pode assumir qualquer valor inteiro. Para se referir a tais sequências será usada a notação $(s_n)_{n=m}^{\infty}$, indicando que esta é uma sequência real infinita que parte de $n = m$. Define-se convergência de uma sequência infinita como se segue [6, 32].

Definição 9. *Uma sequência infinita de números reais é dita convergente para um número real s se para cada $\varepsilon > 0$ existir um número inteiro N tal que $n > N$ implique $|s_n - s| < \varepsilon$.*

Se a sequência $(s_n)_{n=m}^{\infty}$ converge para s, escreveremos como notação que $\lim_{n\to\infty} s_n = s$, ou simplesmente $s_n \to s$.

Uma série infinita pode ser construída a partir de uma sequência infinita. Pode-se definir uma série infinita da seguinte forma.

Definição 10. *A sequência infinita $(s_n)_{n=1}^{\infty}$ nas quais seus termos formam a soma parcial de um conjunto de números reais ordenados, a_n, dada por*

$$\begin{aligned} s_1 &= a_1 \\ s_2 &= a_1 + a_2 \\ &\vdots \\ s_n &= \sum_{k=1}^{n} a_k, \end{aligned} \tag{E.2}$$

é chamada de série.

Usando a Definição 10, de que uma série é uma sequência de números reais tais que esse números representam somas parciais, pode-se afirmar que para toda série infinita há uma sequência infinita associada. Sendo assim, semelhantemente às sequências, as séries podem ser convergentes ou divergentes. A convergência de uma série é dada como consequência da Definição 9. Este resultado é apresentado como teorema a seguir.

Teorema 6. *Uma série infinita , $\sum_{k=1}^{n} a_k$, é dita convergir para s se e somente se sua sequência associada, $(s_n)_{n=1}^{\infty}$, convergir para s.*

Isso corresponde ao fato de que se $\lim_{n\to\infty} s_n = s$, então, como $s_n = \sum_{k=1}^{n} a_k$, pela Definição 10, $\lim_{n\to\infty} \sum_{k=1}^{n} a_k = s$.

Estudar a convergência de uma série pode ser resumido ao estudo do limite da soma parcial dos termos quando $n \to \infty$. Desta forma, para analisar a convergência da série de Fourier basta considerar o limite da soma parcial da série de Fourier truncada quando o número de termos tende ao infinito.

Referências Bibliográficas

[1] Lars V. Ahlfors. *Complex Analysis An Introduction To The Theory Of Analytic Functions Of One Complex Variable*. Mc-Graw Hill, 2nd edition, 1966.

[2] Tom M. Apostol. *Calculus*, Volume 1. John Wiley & Sons, Inc., Pasadena, CA, 2nd edition, sep 1966.

[3] Tom M. Apostol. *Calculus*, Volume 2. John Wiley & Sons, Inc., Pasadena, CA, 2nd edition, sep 1968.

[4] Tom M. Apostol. *Introduction to Analytic Number Theory*. Springer-Verlag, Pasadena, CA, 1976.

[5] Tom M. Apostol. *Mathematical Analysis*. Addison-Wesley Publishing Company, Reading, MA, 1981.

[6] George B. Arfken and Hans J. Weber. *Mathematical Methods for Physicists*. Elsevier Academic Press, Miami University, Oxford, OH and University of Virginia, Charlottesville, VA, 3rd edition, 2005.

[7] Vasudev Bhaskaran and Konstantinos Konstantinides. *Image and Video Compression Standards: Algorithms and Architectures*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[8] R. E. Blahut. *Fast Algorithms for Digital Signal Processing*. Addison-Wesley, 1985.

[9] José Luiz Boldrini, Sueli I. Rodrigues Costa, Vera Lúcia Figueiredo, and Henry G. Wetzler. *Álgebra Linear*. HARBRA, 3ª edição, 1980.

[10] V. Britanak, P. C. Yip, and K. R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic Press Inc. Elsevier Science, 2006.

[11] H. M. de Oliveira D. F. Souza, R. J. de Sobral Cintra. Uma Ferramenta para Análise de Sons Musicais: A Série Quantizada de Fourier. *XXII Simpósio Brasileiro de Telecomunicações*, sep 2005.

[12] Hélio Magalhães de Oliveira. *Análise de sinais para engenheiros: uma abordagem via Wavelets*. Brasport Livros e Multimídia Ltda, Rio de Janeiro, RJ, 2007.

[13] Hélio Magalhães de Oliveira. *Engenharia de Telecomunicações*. Notas de Aula, 2010.

[14] A. G. Dempster and M. D. Macleod. Constant integer multiplication using minimum adders. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 141(5):407 – 413, 1994.

[15] José Dias dos Santos and Zanoni Carvalho da Silva. *Métodos Numéricos*. Editora Universitária, 2ª edição, 2009.

[16] Gerald B. Folland. *Fourier Analysis and Its Applications*. American Mathematical Society, Seattle, WA, 2000.

[17] Joseph Fourier. *Théorie Analytique de La Chaleur*. Éditions Jacques Gabay, Paris, FRA, 1822.

[18] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, 2nd edition, 2002.

[19] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley Publishing Company, Reading, MA, 1989.

[20] David Halliday, Robert Resnick, and Jearl Walker. *Fundamentals os Physics*, Volume 1 and 2 and 3 and 4. John Willey and Sons Inc., 7rd edition, 2005.

[21] George Em Karniadakis and Robert M. Kirby II. *Parallel Scientific Computing in C++ and MPI*. Cambridge University Press, 2003.

[22] Gottfried Konecny. *Geoinformation: remote sensing, photogrammetry and geographic information systems*. Taylor & Francis, oct 2002.

[23] Michael P. Lamoureux. The Poorman's Transform: Approximating the Fourier Transform without Multiplication. *IEEE Transaction on Signal Processing*, 41(3), mar 1993.

[24] Jamal T. Manassah. *Elementary Mathematical and Computational Tools for Electrical and Computer Engineers Using Matlab* . CRC Press, New York, NY, 2001.

[25] Patrick Marchand and Thomas Holland. *Graphics and GUIs with MATLAB*. CHAPMAN & HALL/CRC, 3rd edition, 2003.

[26] J. C. Nash. *Compact Numerical Methods for Computers*. Adam Hilger, 1990.

[27] Scott J. Norton and Mark D. Dipasquale. *The Multithreaded Programming Guide Thread Time*. Prentice-Hall, Inc., 1997.

[28] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-time signal processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, 2nd edition, 1999.

[29] Gordon E. Pelton. *Voice Processing*. Mc-Graw Hill, Inc., 1992.

[30] Lawrence Rabiner and Juang Bung-Hwang. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., 1993.

[31] Kenneth A. Ross. *Elementary Analysis: The Theory of Calculus*. Springer, OR, Eugene, 1980.

[32] B. A. Sethuraman. *Rings, Fields, and Vector Spaces, An Introduction to Abstract Algebra via Geometric Constructibility*. Springer, New York, 175 Fifth Avenue, NY, USA, 1996.

[33] Andrew S. Tanenbaum. *Organização Estruturada de Computadores*. LTC Livros Técnicos e Científicos, Vrije Universiteit, Amsterdam, Holanda, 4th edition, 2001.

[34] Andrew S. Tanenbaum and Albert S. Woodhull. *Operating Systems*. Prentice-Hall, Inc., 1997.

[35] Özdogan Yilmaz. *Seismic Data Analysis*, Volume II. Society of Exploration Geophysicists, 2001.

[36] Geraldo Ávila. *Funções de uma Variável complexa*. LTC, 1974.

[37] Geraldo Ávila. *Variáveis complexas e aplicações*. LTC, 2000.