UNIVERSIDADE SÃO FRANCISCO

CURSO DE ENGENHARIA ELÉTRICA

ANÁLISE ESPECTRAL DE SINAIS UTILIZANDO FFT

Engenharia Elétrica – Modalidade Telecomunicações

André Luís de Souza
Autor

José Sindi Yamamoto, Prof. Dr.
Orientador

Itatiba (SP), Novembro de 2004

UNIVERSIDADE SÃO FRANCISCO

CURSO DE ENGENHARIA ELÉTRICA

ANÁLISE ESPECTRAL DE SINAIS UTILIZANDO FFT

Engenharia Elétrica – Modalidade Telecomunicações

André Luís de Souza
Autor

Relatório apresentado à Banca Examinadora do Trabalho de Conclusão do Curso de Engenharia Elétrica para análise e aprovação.
Orientador: José Sindi Yamamoto, Prof. Dr.

Itatiba (SP), Novembro de 2004

SUMÁRIO

LISTA DE ABREVIATURAS

TCC	Trabalho de Conclusão de Curso
USF	Universidade São Francisco
FT	*Fourier Transform* (Transformada de Fourier)
DFT	*Discrete Fourier Transform* (Transformada Discreta de Fourier)
FFT	*Fast Fourier Transform* (Transformada Rápida de Fourier)
A/D	Analógico para Digital
D/A	Digital para Analógico
SNR	*Signal Noise Ratio* (Relação Sinal/Ruído de Quantização)
DIF	Dizimação em freqüência
DIT	Dizimação no tempo
DIF-FFT	Algoritmo FFT por dizimação em freqüência
DIT-FFT	Algoritmo FFT por dizimação no tempo
PC	*Personal Computer* (Microcomputadores pessoais)
DSP	*Digital Signal Processor*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

RESUMO

Souza, André L. **Análise Espectral de Sinais utilizando FFT.** Itatiba, 2004. 58 f. Trabalho de Conclusão de Curso, Universidade São Francisco, Itatiba, 2004.

Neste trabalho são apresentados os modelos e programas desenvolvidos para análise espectral de sinais utilizando a ferramenta de software Simulink do MATLAB.
Adicionalmente, estão descritos os conceitos teóricos sobre aquisição de sinais, transformada de Fourier de sinais discretos, transformada rápida de Fourier (FFT) e Simulink, necessários para o entendimento e desenvolvimento do trabalho.

Palavras-chave: FFT. Analise Espectral. MATLAB. Simulink.

ABSTRACT

In this work the models and programs developed for spectral analysis of signals are presented using the tool of Simulink software of the MATLAB.
Additionally, the theoretical concepts are described on acquisition of signals, transformed of Fourier of discrete signals, transformed fast of Fourier (FFT) and Simulink, necessary for the agreement and development of the work.

***Keywords**: FFT. Spectral Analysis. MATLAB. Simulink.*

1. INTRODUÇÃO

Diversas formas podem ser empregadas para a realização de análise espectral de sinais, desde um banco de filtros até a utilização de transformadas de Fourier no domínio analógico ou discreto. Entretanto, a facilidade existente atualmente de aquisição de sinais em microcomputadores pessoais (PC), juntamente com a utilização da transformada discreta de Fourier (DFT), torna-se uma maneira muito atrativa para se realizar a analise espectral de sinais.

A DFT pode ser utilizada tanto para aplicações em tempo real como em aplicações em tempo não real. No primeiro caso, dependendo da largura de banda do sinal, torna-se necessária ainda a utilização de DSPs (*Digital Signal Processors*), enquanto que no segundo caso se torna necessária somente a utilização do processador do próprio PC e o armazenamento prévio do sinal que se deseja analisar. Tendo em vista a grande capacidade de armazenamento atualmente disponível nos PCs, a análise em tempo não real torna-se cada vez mais atrativa, para uma grande variedade de aplicações.

Assim, neste trabalho, são desenvolvidos os modelos e programas de DFT para análise espectral de sinais em tempo não real e em tempo real no caso de sinais de faixa estreita.

A DFT envolve muitos cálculos, sendo assim importante a utilização de métodos eficientes para a sua implementação prática. Os métodos conhecidos como transformada rápida de Fourier (FFT do Inglês: "*Fast Fourier Transform*") são utilizados neste trabalho para a implementação da DFT. Como ambiente de desenvolvimento, é utilizado o Simulink do MATLAB.

1.1. OBJETIVOS

1.1.1. Objetivo Geral

O objetivo geral deste trabalho é o desenvolvimento de um sistema para análise espectral de sinais em tempo real e não real, utilizando-se FFT e modelos do Simulink do MATLAB.

1.1.2. Objetivos Específicos

Os principais objetivos deste trabalho são:

- Estudar os problemas de aquisição de sinais
- Estudar os métodos rápidos de cálculo da DFT
- Desenvolver um sistema de análise espectral de sinais em tempo real e não real, baseado na aplicação da FFT através dos modelos do Simulink do MATLAB.

1.2. METODOLOGIA

Está descrita a seguir a metodologia utilizada para o desenvolvimento deste trabalho:

- Estudo da teoria de Processamento Digital de Sinais envolvida: Aquisição de sinais, Teorema de amostragem, Conversão Analógico-Digital, Relação Sinal/Ruído de Quantização (SNR), Transformada Discreta de Fourier (DFT) e Transformada Rápida de Fourier (FFT).
- Estudo do bloco denominado "FFT" do Simulink e da função denominada "fft" do MATLAB.
- Criação de um modelo no Simulink para exibir o espectro de freqüências de um sinal senoidal utilizando FFT.
- Criação de um modelo no Simulink para gravar sinais de áudio em formato *.wav.
- Criação de um modelo no Simulink para aplicar a FFT sobre o sinal de áudio previamente gravado em formato *.wav, para a visualização de seu espectro de freqüências.
- Criação de um modelo no Simulink para aplicar a FFT (Fast Fourier Transform) ou Transformada Rápida de Fourier sobre o sinal de áudio capturado de um microfone conectado a placa de som do computador, em tempo real, para a visualização de seu espectro de freqüências.

?? Demonstração dos resultados obtidos em gráficos

1.3. ESTRUTURA DO TRABALHO

Este trabalho está estruturado em uma parte teórica sobre processamento digital de sinais, abordando principalmente tópicos sobre FFT, e uma parte prática onde será desenvolvida uma metodologia para possibilitar a análise espectral de sinais utilizando os softwares MATLAB e Simulink.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. CONSIDERAÇÕES SOBRE O SOFTWARE MATLAB

O MATLAB é um sistema interativo baseado em matrizes de uso científico e de engenharia para computação numérica e visualização, desenvolvido pela MathWorks.

Seu ponto forte está em resolver problemas numéricos complexos fácil e rapidamente, com o uso de uma linguagem de programação semelhante ao Fortran ou C.

Ele também é poderoso do ponto de vista usual. Por possuir uma capacidade de usar uma programação relativamente simples, o MATLAB pode ser facilmente estendido, permitindo a criação de novos comandos e funções [1].

O MATLAB esta disponível para alguns ambientes computacionais: PCs utilizando DOS e Windows, Macintosh da Apple, estações de trabalho UNIX e várias máquinas paralelas.

O programa MATLAB básico está sempre sendo melhorado pela numerosa quantidade de *tollboxes* (uma coleção de funções específicas para um determinado assunto) ao longo dos anos.

Na Figura 1, encontra-se o ícone do software MATLAB.

Figura 1. Ícone do software MATLAB.

2.2. CONSIDERAÇÕES SOBRE O SOFTWARE SIMULINK

O Simulink é um aplicativo para o MATLAB e consiste em um ambiente interativo baseado em diagrama de blocos, voltado para modelamento, simulação e análise de sistemas dinâmicos contínuos, discretos ou híbridos [5].

O Simulink utiliza o ambiente numérico computacional do MATLAB como motor básico para a construção de modelos compostos por diagramas de blocos.

Sem muito rigor técnico, sistema dinâmico pode ser definido como um sistema cuja saída (resultado e estado) muda ao longo do tempo. A maioria dos eventos do mundo real são sistemas dinâmicos, por exemplo: circuitos elétricos, sistemas de amortecedores, sistemas de freios, sistemas termodinâmicos, etc.

O Simulink permite analisar o comportamento desses tipos de fenômeno (sistemas dinâmicos) a partir da construção do modelo matemático utilizando diagramas de blocos simulados pelo motor numérico do MATLAB.

2.3. CONVERSÃO ANALÓGICA-DIGITAL

Basicamente, o processo de conversão A/D de um sinal constituí-se de três passos [2], como descrito logo abaixo:

1. Amostragem – Esta é a conversão de um sinal contínuo no tempo em um sinal discreto no tempo obtido por *taking samples* ou amostras capturadas de um sinal contínuo no tempo em instantes discretos no tempo. Portanto, se $x_a(t)$ é o sinal de entrada a ser amostrado, o sinal de saída é $x_a(nT)$? $x(n)$, onde T é chamado de intervalo de amostragem.

2. Quantização – Esta é a conversão de um sinal de valor contínuo e discreto no tempo em um sinal (digital) de valor discreto e discreto no tempo. O valor de qualquer amostra deste sinal é representado por um valor selecionado dentre um range finito de possíveis valores. A diferença entre uma amostra não quantizada $x(n)$ e uma saída quantizada $x_q(n)$ é chamado de erro de quantização.

3. Codificação – No processo de codificação, qualquer valor discreto $x_q(n)$ é representado por uma seqüência binária.

O processo de conversão A/D pode ser observado na Figura 2.

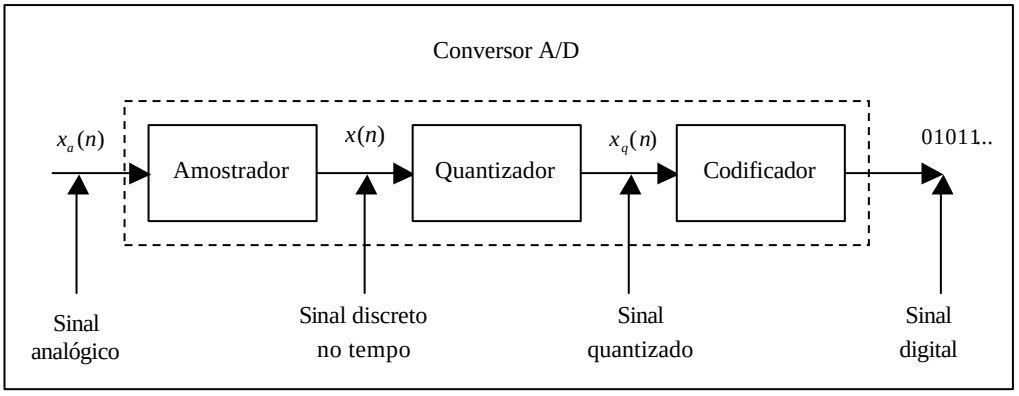

Figura 2. Partes básicas de um conversor A/D
Fonte: Adaptado de Proakis [2] (1996)

Embora seja possível modelar um conversor A/D como um amostrador seguido por um quantizador e um codificador, na prática a conversão A/D é realizada por um dispositivo simples que transforma o sinal $x_a(t)$ em números binários.

As operações de amostragem e quantização podem ser realizadas em qualquer ordem, mas, na prática, a amostragem é sempre realizada antes da quantização.

2.4. AMOSTRAGEM DE SINAIS ANALÓGICOS.

Existem muitas maneiras de amostrar um sinal analógico.

Na prática, as mais comuns são a amostragem periódica e a amostragem uniforme [2].

Elas são descritas pela relação como na Equação 1.

$$x(n) ? x_a(nT), \quad ? ? ? n ? ? \qquad \text{Equação 1}$$

Onde x(n) é o sinal discreto no tempo obtido por ***taking samples*** ou **amostras capturadas** do sinal analógico $x_a(t)$ a cada T segundos. Este procedimento é ilustrado na Figura 3. O intervalo de tempo T entre amostras sucessivas é chamado de período de amostragem ou intervalo de

amostragem e a recíproca 1/T = F_S é chamada de taxa de amostragem (amostras por segundo) ou freqüência de amostragem (hertz).

Amostragens periódicas estabelecem uma relação entre variáveis temporais t e n de sinais contínuos no tempo e sinais discretos no tempo, respectivamente. Realmente, estas variáveis são linearmente relatadas através do período de amostragem T ou, equivalentemente, através da taxa de amostragem $F_s = 1/T$, como pode ser observado na Equação 2.

$$t ? nT ? \frac{n}{F_s}$$ Equação 2

Como uma conseqüência da fórmula anterior, existe uma relação entre a freqüência variável F (ou ?) para sinais analógicos e a freqüência variável f (ou ?) para sinais discretos no tempo. Para estabelecer esta relação, consideremos um sinal analógico senoidal da Equação 3.

$$x_a(t) ? A\cos(2? Ft ? ?)$$ Equação 3

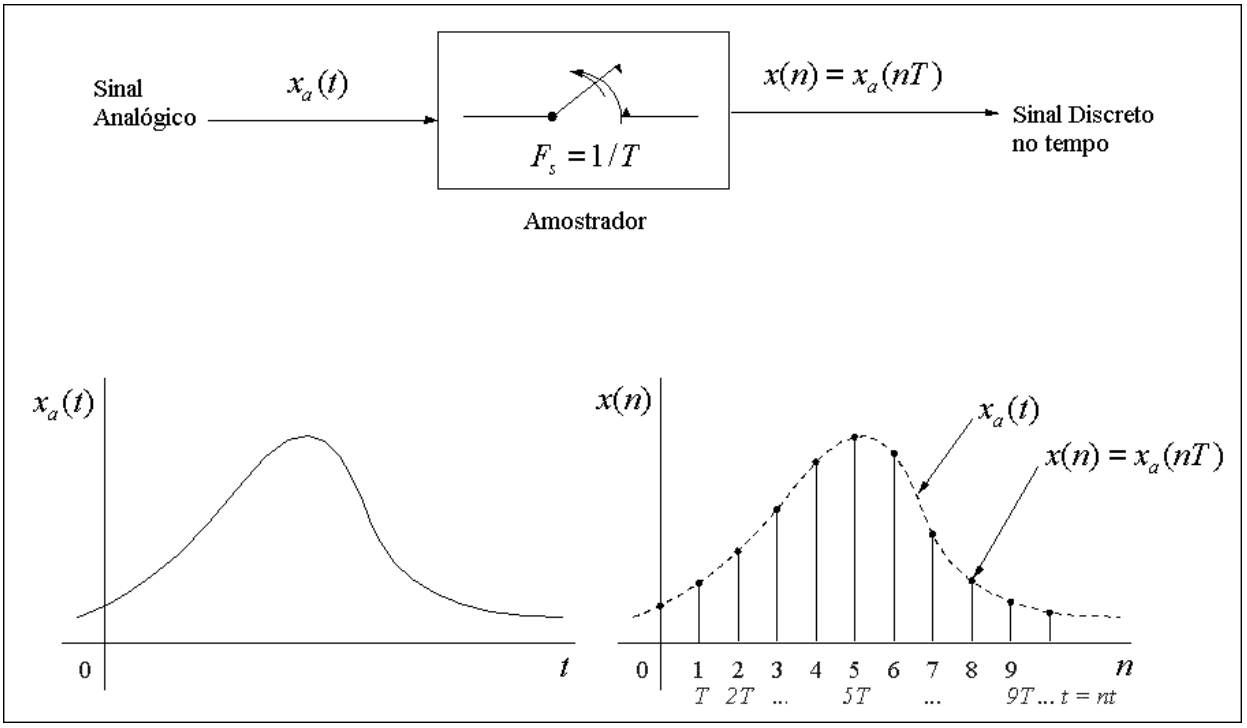

Figura 3. Amostragem periódica de um sinal analógico
Fonte: Adaptado de Proakis [2] (1996)

2.5. SINAIS DISCRETOS

Um sinal discretizado é obtido a partir de um sinal contínuo no tempo, através de uma amostragem em intervalos igualmente espaçados.

Chamando de s_n o sinal discretizado, então a amostragem de s(t) a cada T segundos fica como demonstrado na Equação 4.

$$s_n ? s(nT) \qquad n ? \ldots ? 1,0,1,2\ldots$$ Equação 4

Existe uma ambigüidade quando sinais contínuos no tempo são representados por um conjunto de amostras, a qual pode ser exemplificada tomando-se a amostragem de uma co-senóide de freqüência igual 2 Hz, obtida a uma taxa de cinco amostras por segundo; amostras idênticas poderiam ser obtidas se o sinal sob amostragem fosse uma co-senóide de freqüência 3Hz.

Seja um conjunto de amostras que esteja associado a um sinal contínuo que teve sua banda de freqüências limitada. Quando a taxa de amostragem de um sinal é maior que o dobro da maior freqüência presente no sinal, podemos afirmar que existirá somente um único sinal contínuo associado ao sinal discretizado.

O fenômeno de ter outros nomes para sinais contínuos com a mesma amostragem, como no caso em que a amostragem de uma co-senóide de freqüência 2Hz, e outra de uma co-senóide de 3 Hz, a uma taxa de cinco amostras por segundo, fornecendo os mesmos resultados, é chamado de ***aliasing*** ou **superposição espectral** [6].

Este fenômeno está exemplificado na Figura 4, onde aparece a co-senóide de freqüência 2 Hz na parte superior e a co-senóide 3 Hz na parte inferior da figura.

As quatro linhas horizontais ligando as figuras em pontos a, b, c e d, são os pontos de amostragem, além da origem, tomados a uma taxa de cinco amostras por segundo. Os valores de a, b, c e d, além da origem, são iguais para os dois gráficos, e valem:

origem: = 1
ponto a = ponto d = -0.809
ponto b = ponto c = 0.309

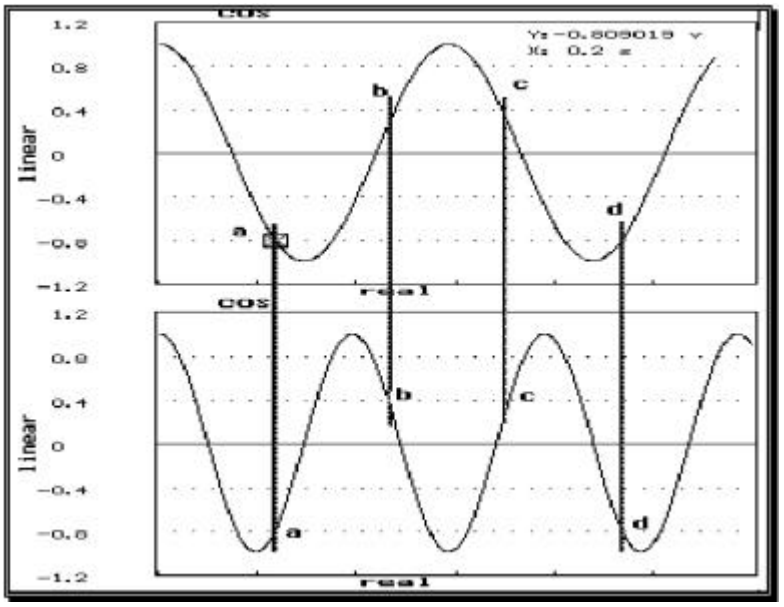

Figura 4. Exemplo de um caso de sobreposição espectral
Fonte: Adaptado de Scandelari [6]

2.6. TEOREMA DE AMOSTRAGEM

Qualquer sinal contém informações em amplitude, freqüência e fases de diversas componentes espectrais, mas estas informações em diversos casos não estão disponíveis para que se possa analisar o sinal.

Em particular, deve-se ter alguma informação geral, no domínio da freqüência, sobre o sinal a ser amostrado.

Para selecionar o período de amostragem T ou a equivalente taxa de amostragem F_s de qualquer sinal analógico, deve-se possuir algumas características sobre o sinal.

Sabendo-se a maior componente do espectro de freqüência de um sinal, é possível especificar a taxa de amostragem necessária para converter sinais analógicos em sinais digitais.

Supondo que qualquer sinal analógico pode ser representado por uma soma de senóides de diferentes amplitudes, freqüências e fases, como representado na Equação 5.

$$x_a(t) = \sum_{i=1}^{N} A_i \cos(2\pi F_i t + \theta_i) \qquad \text{Equação 5}$$

Onde N representa o número de componentes espectrais.

Num sinal de voz, por exemplo, as amplitudes, fases e freqüências geralmente mudam lentamente com o tempo de um segmento de tempo para outro.

Supondo que as freqüências não excedam um valor de freqüência conhecido, chamamos este valor conhecido de F_{max}, e que geralmente num sinal de voz, por exemplo, as maiores componentes do espectro de freqüência estão perto de 3000 Hz, pode-se utilizar um filtro que atenuará severamente as componentes espectrais acima de Fmax, ou seja, acima de 3000 Hz [2].

Para qualquer F_{max} conhecida é possível selecionar uma taxa de amostragem apropriada.

A maior freqüência em um sinal analógico que pode ser reconstruída não ambiguamente quando sinais são amostrados a taxas de $F_s = 1/T$ é $F_s/2$. Qualquer freqüência acima de $F_s/2$ ou abaixo de $–F_s/2$ resulta em amostras que são idênticas com um correspondente range de freqüências $–F_s/2 ? F ? F_s/2$.

Para evitar resultados ambíguos de sobreposição espectral, deve-se selecionar uma taxa de amostragem que seja suficiente alta. Deve-se então selecionar $F_s/2$ para que seja maior que F_{max}. Então, para eliminar o problema de sobreposição espectral, F_s é selecionada como mostrado na Equação 6.

$$F_s ? 2F_{max}$$ Equação 6

Onde F_{max} é a maior componente espectral do sinal analógico. Com a taxa de amostragem selecionada desta maneira, qualquer componente espectral, ou seja, $|F_i| < F_{max}$, no sinal analógico é mapeado em uma senóide discreta no tempo com uma freqüência como mostrado na Equação 7.

$$? \frac{1}{2} ? f_i ? \frac{F_i}{F_s} ? \frac{1}{2}$$ Equação 7

Ou equivalentemente, como mostrado na Equação 8.

$$? ? ? ? _i ? 2? f_i ? ?$$ Equação 8

Desde que f(x) – ½ ou |? | = p é a maior (única) freqüência no sinal discreto no tempo, a escolha da taxa de amostragem de acordo com $F_s > 2F_{max}$ elimina o problema de sobreposição espectral.

Em outras palavras a condição $F_s > 2F_{max}$ possibilita a medida de todas as componentes senoidais no sinal analógico estão mapeadas dentro das correspondentes componentes espectrais discretas no tempo com freqüências no intervalo fundamental. Todas as componentes espectrais do sinal analógico são representadas na forma de amostras sem ambigüidade, e assim o sinal analógico pode ser reconstruído sem distorção com os valores das amostras usando um método de interpolação apropriada (Conversão digital/analógica). A fórmula da interpolação ideal é especificada pelo teorema de amostragem.

2.6.1. Sobre o Teorema de Amostragem:

Se a maior freqüência contida num sinal analógico $x_a(t)$ é $F_{max} = B$ e o sinal é amostrado na taxa $F_s > 2F_{max} = 2B$, então $x_a(t)$ pode ser recuperado exatamente dos valores amostrados usados na função de interpolação como mostrado na Equação 9 [2].

$$g(t) ? \frac{\sin 2?Bt}{2?Bt}$$ Equação 9

Então $x_a(t)$ pode ser expresso como mostrado na Equação 10 abaixo:

$$x_a(t) ? \sum_{n???}^{?} x_a\left(\frac{n}{F_s}\right) g\left(t ? \frac{n}{F_s}\right)$$ Equação 10

onde, $x_a (n/F_s) = x_a(nT) = x(n)$ são amostras de $x_a(t)$.

Quando a amostragem de $x_a(t)$ é realizada na menor taxa de amostragem $F_s = 2B$, pode-se reconstruir a fórmula como mostrado na Equação 11.

$$x_a(t) ? \sum_{n???}^{?} x_a\left(\frac{n}{2B}\right) \frac{\sin 2?B(t ? n/2B)}{2?B(t ? n/2B)}$$ Equação 11

A taxa de amostragem FN = 2B = $2F_{max}$ é chamada de ***Nyquist rate*** ou **taxa de Nyquist**. A Figura 5 ilustra uma processo de conversão D/A ideal usando a função de interpolação (Equação 9).

Pode-se observar na Equação 10 ou na Equação 11, que a reconstrução de $x_a(t)$ da seqüência x(n) é um processo complicado, envolvendo uma grande soma de funções de interpolação g(t) e suas versões deslocadas no tempo g(t-nT) para -8 < n < 8, onde os fatores delimitantes são as amostras x(n).

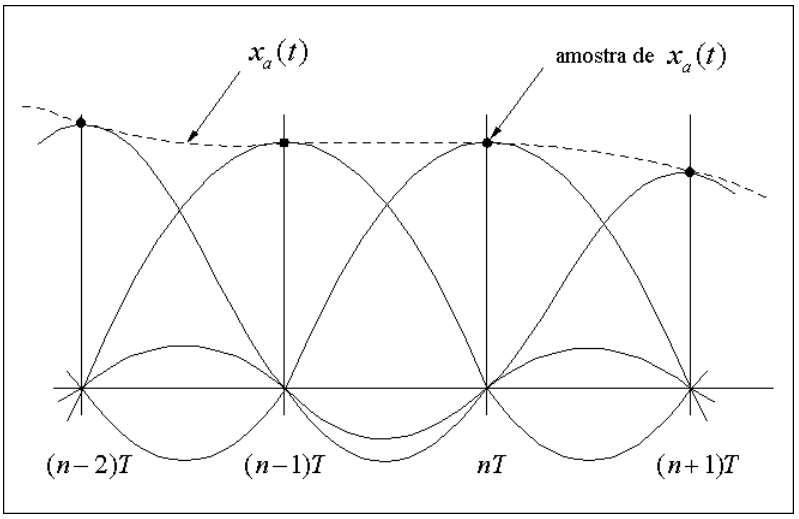

Figura 5. Conversão D/A ideal (interpolação)
Fonte: Adaptado de Proakis [2]

Por causa da complexidade e de um número infinito de amostras requeridas na Equação 10 e na Equação 11, estas equações de reconstrução são de maior interesse teórico do que prático.

2.7. QUANTIZAÇÃO DE SINAIS DE AMPLITUDE CONTÍNUA

Um sinal digital é uma seqüência de números (amostras) e qualquer um destes números é representado por um número finito de dígitos (precisão finita) [2].

O processo de conversão de um sinal discreto no tempo e com amplitude contínua em um sinal digital que expresse qualquer valor amostrado com um número finito de dígitos é chamado **quantização**. O erro introduzido na representação de um sinal de valor contínuo por um faixa finita de níveis discretos de valores é chamado de **erro de quantização** ou **ruído de quantização**.

Define-se a operação de quantização de amostras x(n) como Q[x(n)] e $x_q(n)$ denomina a seqüência de amostras quantizadas na saída do quantizador, então isto pode ser representado como na Equação 12.

$$x_q(n) ? Q[x(n)] \qquad \text{Equação 12}$$

Quando o erro de quantização é uma seqüência $e_q(n)$ definida como a diferença entre o valor quantizado e a amostra atual, então pode se representar como na Equação 13.

$$e_q(n) ? x_q(n) ? x(n) \qquad \text{Equação 13}$$

O Exemplo 1 a seguir ilustra o processo de quantização.

Exemplo 1:

Considerando o sinal discreto no tempo como representado na Equação 14.

$$x(n) ? \begin{cases} ?0.9^n, & n ? 0 \\ ?0, & n ? 0 \end{cases} \qquad \text{Equação 14}$$

Obtido por amostragem do sinal exponencial analógico $x_a(t) = 0.9t$, t = 0 com uma freqüência de amostragem F_S = 1Hz (ver Figura 6(a)), pode-se observar na Tabela 1 o valor das 10 primeiras amostras de x(n), o que revela que a descrição do valor da amostra x(n) requer n dígitos significantes.

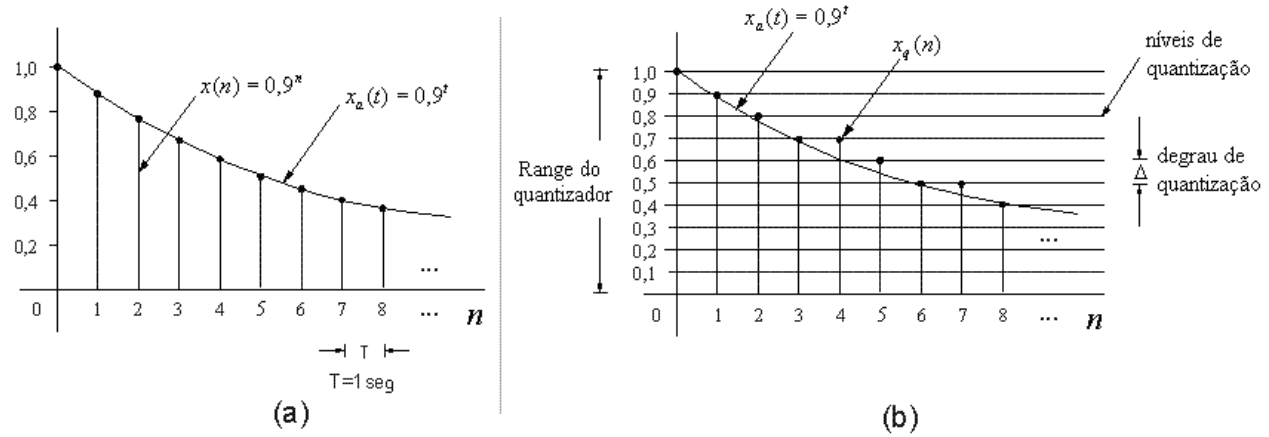

Figura 6. Amostragem de um sinal exponencial analógico. (a) Freqüência de Amostragem; (b) Níveis de Quantização.
Fonte: Adaptado de Proakis [2] (1996)

Tabela 1. Ilustração numérica da quantização com um digito significante usando truncamento e aproximação:

n	x(n) Sinal discreto no tempo	xq(n) (Truncado)	xq(n) (Aproximado)	eq(n) = xq(n) – x(n) (Aproximado)
0	1	1,0	1,0	0,0
1	0,9	0,9	0,9	0,0
2	0,81	0,8	0,8	-0,01
3	0,729	0,7	0,7	-0,029
4	0,6561	0,6	0,7	0,0439
5	0,59049	0,5	0,6	0,00951
6	0,531441	0,5	0,5	-0,031441
7	0,4782969	0,4	0,5	0,0217031
8	0,43046721	0,4	0,4	-0,03046721
9	0,387420489	0,3	0,4	0,012579511

Fonte: Adaptado de Proakis [2] (1996)

Fica claro que este sinal não pode ser processado usando uma calculadora ou um computador digital visto que apenas as primeiras amostras (que são poucas) podem ser armazenadas e manipuladas. Por exemplo, a maioria das calculadoras possuem apenas 8 dígitos significantes.

Por outro lado, pode-se assumir que se deseja apenas um digito significante. Para eliminar o excesso de dígitos, pode-se simplesmente descará-los (truncar), ou fazer uma aproximação (ou arredondamento) no número resultante. Os sinais quantizados resultantes $x_q(n)$ são mostrados na Tabela 1.

O processo de aproximação é ilustrado na Figura 6(b). Os valores permitidos no sinal digital são chamados de **níveis de quantização**, à distância ? entre dos níveis de quantização sucessivos é chamada de tamanho do **degrau de quantização** ou **resolução**. O quantizador aproximador associa qualquer amostra de x(n) ao nível de quantização mais próximo. Em contraste, o quantizador truncador associa qualquer amostra de x(n) ao nível de quantização abaixo dela. O erro de quantização eq(n) no quantizador aproximador é limitado à faixa de –?/2 à ?/2, como representado na Equação 15.

$$? \frac{?}{2} ? e_q(n) ? \frac{?}{2} \qquad \text{Equação 15}$$

Em outras palavras, o erro de quantização instantâneo não pode exceder a metade do degrau de quantização (ver Tabela 1).

Se x_{min} e x_{max} representam o menor e o maio valor de x(n) e L é o número de níveis de quantização, então se resulta na Equação 16.

$$? \; ? \; \frac{x_{max} \; ? \; x_{min}}{L \; ? \; 1} \qquad \text{Equação 16}$$

A definição ***dynamic range*** do sinal é representada como $x_{max} - x_{min}$. No Exemplo1 tem-se $x_{max} = 1$, $x_{min} = 0$, e L = 11, que resulta em ? = 0.1. Se o *dinamic range* é fixado, incrementando o número de níveis de quantização, L resulta num decremento do tamanho do degrau de quantização. Então o erro de quantização diminui e a precisão do quantizador aumenta. Na pratica pode-se reduzir o erro de quantização a um nível insignificante pela escolha de um número suficiente de níveis de quantização.

Teoricamente, a quantização de sinais analógicos sempre resulta em perda de informação. Este é um resultado da ambigüidade introduzida pela quantização. Na verdade, a quantização é um processo irreversível (por exemplo: mapeamento muitos-para-um) desde que todas as amostras estejam distanciadas de ?/2 certamente os níveis de quantização terão o mesmo valor. Esta ambigüidade torna a análise quantitativa exata da quantização extremamente difícil [2].

2.8. QUANTIZAÇÃO DE SINAIS SENOIDAIS

A Figura 7 ilustra a amostragem e quantização de um sinal analógico senoidal $x_a(t) \; ? \; A\cos ? \; _0 t$ usando uma grade retangular. As linhas horizontais com a faixa do quantizador indicam os níveis de quantização permitidos. As linhas verticais indicam o tempo de amostragem. Portanto, do sinal original analógico $x_a(t)$ pode-se obter um sinal discreto no tempo $x(n) = x_a(nT)$ por amostragem e um sinal discreto no tempo e na freqüência $x_q(nT)$ após a quantização[2].

Na prática, o sinal $x_q(t)$ pode ser obtido usando *zero-order hold*. Esta análise é usual porque senóides são usadas como sinais de teste em conversores A/D.

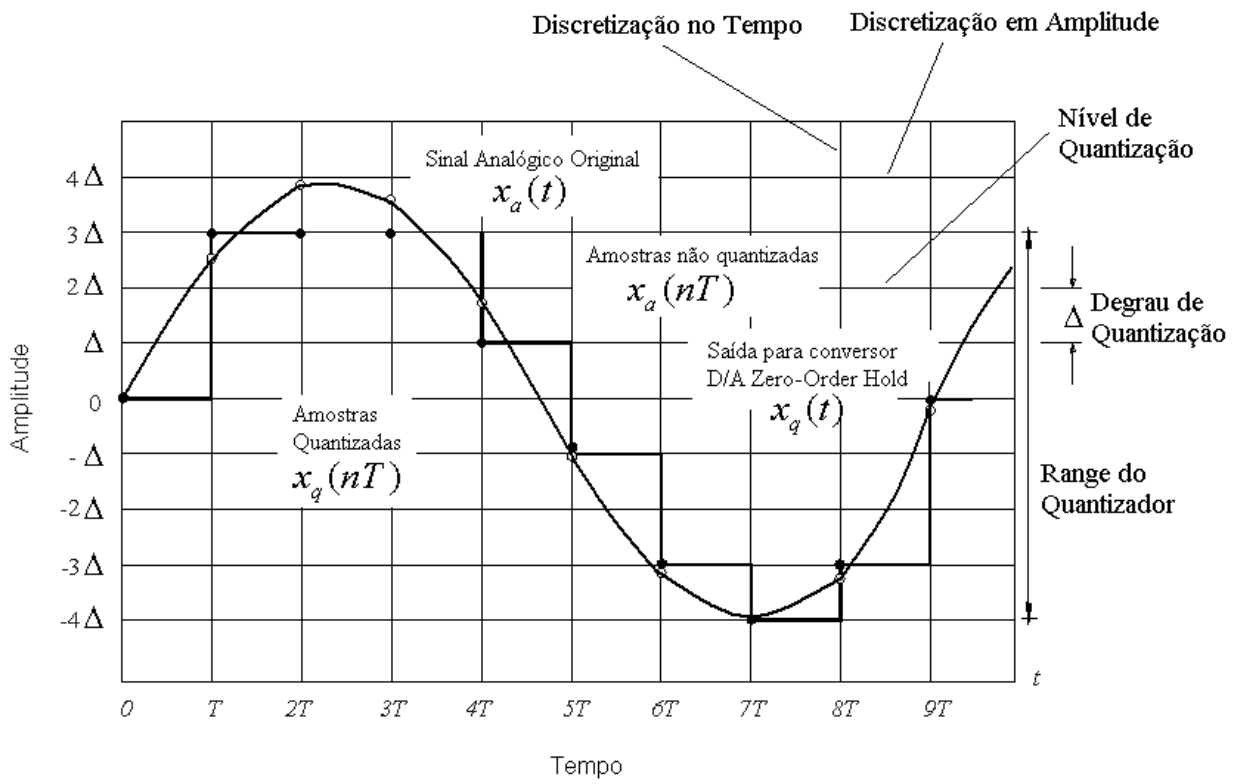

Figura 7. Amostragem e quantização de um sinal analógico senoidal.
Fonte: Adaptado de Proakis [2] (1996)

Se a taxa de amostragem F_s satisfaz o teorema de amostragem, a quantização é o único erro no processo de conversão A/D. Então, pode-se avaliar o erro de quantização pela quantização do sinal analógico $x_a(t)$ ao invés do sinal discreto no tempo $x(n) = x_a(nT)$. A Figura 7 indica que o sinal $x_a(t)$ é quase linear entre os níveis de quantização (ver Figura 8). O correspondente erro de quantização $e_q(t) = x_a(t) – x_q(t)$ é mostrado na Figura 8.

Ainda na Figura 8, ? indica o tempo que $x_a(t)$ permanece nos níveis de quantização.

A qualidade da saída do conversor A/D é usualmente medida pelo *Signal-to-quantization noise ratio* (SQNR) ou Relação Sinal/Ruído de quantização, que provê a razão entre a potência do sinal e a potência do ruído, como demonstrado na Equação 17.

$$SQNR = \frac{P_x}{P_q} = \frac{3}{2} 2^{2b} \qquad \text{Equação 17}$$

O SQNR pode ser expresso em decibéis (dB), como mostrado na Equação 18.

$$SQNR(dB) ? 10\log_{10} SQNR ? 1{,}76 ? 6{,}02b$$ Equação 18

Isto implica que o SQNR incrementa aproximadamente 6 dB para todo bit adicionado à largura da palavra, isto é, para cada duplicação de níveis de quantização.

Embora a Equação 18 seja demonstrada para sinais senoidais, resultados similares são obtidos para qualquer sinal cujo range dinâmico dura o range do quantizador.

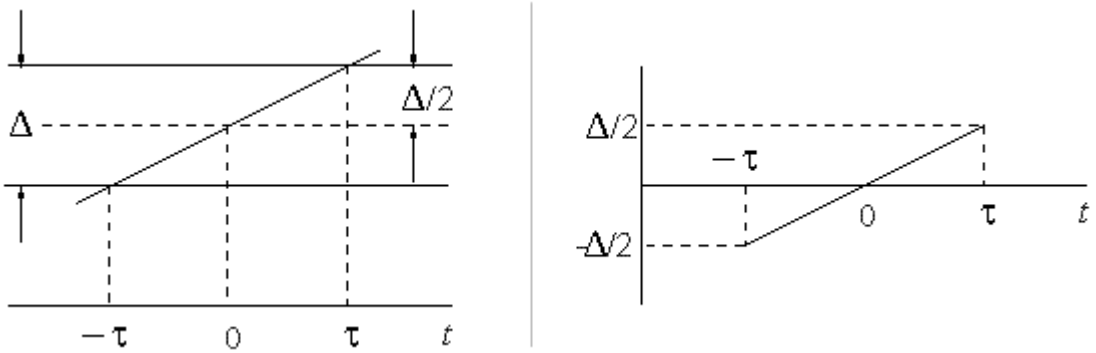

Figura 8. Erro de quantização.
Fonte: Adaptado de Proakis [2] (1996)

Esta relação é extremamente importante porque diz o número de bits requeridos por uma aplicação especifica para assegurar uma dada relação sinal/ruído. Por exemplo, a maioria dos *compact disc players* usam uma freqüência de amostragem de 44.1 kHz e 16 bit de resolução, o que implica em um SQNR de mais de 96 dB [2].

2.9. CODIFICAÇÃO DE AMOSTRAS QUANTIZADAS

O processo de codificação no conversor A/D associa um único número binário para cada nível de quantização. Se o quantizador possui L níveis, precisa-se de ao menos L diferentes números binários. Com uma largura de b bits, pode-se criar 2b diferentes números binários [2].

Desde que se tenha 2^b = L, ou equivalente, b = $\log_2$ L. Então o número de bits requeridos em um codificador é o menor número inteiro maior ou igual ao $\log_2$ L. No exemplo anterior pode-se observar que seria necessário um codificador com b = 4bits.

Comercialmente , pode-se obter conversores A/D com precisão finita de b = 16 ou menos. Geralmente, quanto maior velocidade de amostragem e maior a precisão de quantização, mais caros estes dispositivos se tornam.

2.10. TRANSFORMADA DE FOURIER DE UM SINAL DISCRETO

A transformada de Fourier de uma seqüência *x(n)*, tal que $\sum_{n=-\infty}^{\infty} |x(n)| < \infty$ pode ser definida como representado na Equação 19 [6].

$$X(w) = \sum_{n=-\infty}^{\infty} x(n)e^{-jwn} \qquad \text{Equação 19}$$

E a transformada inversa como representado na Equação 20.

$$x(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(w)e^{jwn}\,dw \qquad \text{Equação 20}$$

Esta transformada de Fourier é definida para todos os valores da variável de freqüência normalizada, w. Uma das principais diferenças entre a transformada de Fourier de um sinal contínuo e de um sinal discreto é o fato de que a transformada de Fourier de um sinal discreto é uma função periódica da freqüência variável w. Para valores inteiros de m, observar a Equação 21.

$$X(w - 2\pi m) = \sum_{n=-\infty}^{\infty} x_n e^{-j(w-2\pi m)n} = X(w) \qquad \text{Equação 21}$$

Esta periodicidade está presente pela mesma razão da **sobreposição espectral** discutida anteriormente. Na Equação 20, o sinal discreto está sendo representado como uma superposição de exponenciais com varias freqüências w na forma e^{jwn} . Uma exponencial com uma freqüência w+2.p.m (m inteiro) passará exatamente pelas mesmas amostragens que a exponencial com freqüência w, como representado na Equação 22.

$$e^{jwn} = e^{j(w-2\pi m)n} \qquad \text{Equação 22}$$

Por esta razão, a transformada de Fourier de um sinal discreto deve ser uma função periódica na freqüência [6].

2.11. A TRANSFORMADA DISCRETA DE FOURIER (DFT)

Pelo teorema de amostragem, a transformada de Fourier *X(k)* de uma seqüência x(n), pode ser amostrada em N pontos ou amostras eqüidistantes. O resultado desta amostragem é denominado de transformada discreta de Fourier (DFT, do Inglês: "*Discrete Fourier Transform*"), e pode ser definida como na Equação 23.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \qquad \text{Equação 23}$$

Onde $W_N^{nk} = e^{-j(2\pi/N)kn}$ constitui a função complexa básica, ou fatores cíclicos da DFT. Estes são periódicos e definem pontos no círculo unitário no plano complexo. Desde que a expansão de uma função periódica de período N, Equação 23 pode ser considerada como sendo a expansão em série de Fourier do sinal periódico $x_p(n)$ com um período N demonstrado na Equação 24 abaixo:

$$x_p(n) = \sum_{m=-\infty}^{\infty} x_r(n - mN) \qquad \text{Equação 24}$$

Onde:

$$x_r(n) = \begin{cases} x_n & 0 \le n \le N-1 \\ 0 & \text{demais valores} \end{cases} \qquad \text{Equação 25}$$

Então a Equação 23 pode ser vista em duas formas. Quando for utilizada para representar um sinal periódico, os coeficientes *X(k)* são também chamados de *Coeficientes discretos da Série de Fourier*. Quando for vista como uma expansão de *x(n)* no conjunto finito $0 \le n \le N-1$, será chamada de *Transformada Discreta de Fourier* (DFT).

A Transformada Discreta de Fourier juntamente com a convolução discreta, é uma das duas operações fundamentais em processamento digital de sinais.

A DFT é utilizada na descrição, representação e análise de sinais discretos. Também é usada juntamente com algoritmos eficientes para o cálculo rápido de convolução e correlação [6].

2.12. A TRANSFORMADA RÁPIDA DE FOURIER (FFT)

A DFT, representada pela Equação 26, é a única transformada que é discreta em ambos os domínios do tempo e freqüência, e é definido por uma seqüência de duração finita. Embora esta seja uma transformada que pode ser usada computacionalmente, a implementação da Equação 26 se mostra muito ineficiente, especialmente quando o tamanho da seqüência *N* é muito grande [1].

Em 1965 *Cooley* e *Tukey* demonstraram um procedimento que reduziu substancialmente a quantidade de cálculos computacionais envolvidos na DFT. Isto permitiu a explosão de aplicações da DFT, incluindo na área de processamento digital de sinais. Além do mais, isto permitiu o desenvolvimento de outros algoritmos eficientes. Todos estes algoritmos constituem um grupo conhecido como algoritmos de FFT (fast Fourier transform).

Considerando uma seqüência *x(n)* com *N* pontos, sua DFT pode ser representada como demonstrado na Equação 26 abaixo:

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \le k \le N-1 \qquad \text{Equação 26}$$

onde $W_N = e^{-j2\pi/N}$.

Para obter uma amostra de *X(k)*, é necessário *N* multiplicações complexas e (*N-1*) adições complexas. Portanto para obter um conjunto completo de coeficientes DFT, é necessário N^2 multiplicações complexas e $N(N-1) \approx N^2$ adições complexas. Além disto, devem ser armazenados N^2 coeficientes complexos W_N^{nk} (ou gera-los internamente com um processamento extra). Certamente, o número de cálculos DFT para uma seqüência de *N* amostras depende quadraticamente de *N*, como representado na Equação 27.

$$C_N = o(N^2) \qquad \text{Equação 27}$$

Na prática, para um grande N, $o(N^2)$ é inaceitável. Geralmente, o tempo de processamento para uma adição é muito menor que para uma multiplicação. Pode-se dizer que cada multiplicação complexa requer 4 multiplicações reais e 2 adições reais.

2.12.1. O objetivo de uma computação eficiente

Em um algoritmo eficiente, o número de cálculos computacionais deve ser constante por amostra de dados, e portanto o número total de cálculos computacionais deve ser linear com respeito a N [1].

A dependência quadrática de N pode ser reduzida na maioria dos cálculos computacionais (que são realizados repetidamente) utilizando as propriedades de periodicidade (Equação 28) e simetria (Equação 29) do fator W_N^{nk}.

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n} \qquad \text{Equação 28}$$

$$W_N^{kn+N/2} = -W_N^{kn} \qquad \text{Equação 29}$$

Um algoritmo que considera apenas a periodicidade de W_N^{nk}, é o algoritmo de *Goertzel* que também requer $C_N = o(N^2)$ multiplicações.

Para ilustrar as vantagens das propriedades de periodicidade e simetria, serão descritos e analisados dois algoritmos de FFT específicos, que requerem $C_N = o(N \log N)$ operações. Estes algoritmos são conhecidos como algoritmo de dizimação no tempo (DIT-FFT) e algoritmo de dizimação em freqüência (DIF-FFT). A seguir segue o Exemplo 2, para análise destes dois algoritmos.

Exemplo 2:

O objetivo deste exemplo [1] é desenvolver um algoritmo eficiente para realizar a computação de 4 pontos DFT, como descrito na Equação 30.

$$X(k) = \sum_{n=0}^{3} x(n)W_4^{nk}, \quad 0 \le k \le 3, \quad W_4 = e^{-j2\pi/4} = -j \qquad \text{Equação 30}$$

Os cálculos computacionais da Equação 30, podem ser demonstrados em forma de uma matriz que requer 16 multiplicações complexas, como mostrado na Equação 31.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad \text{Equação 31}$$

Logo em seguida, pode-se fazer uma aproximação eficiente utilizando a propriedade da periodicidade, como demonstrado na Equação 32.

$$\begin{aligned} W_4^0 = W_4^4 = 1 \quad ; \quad W_4^1 = W_4^9 = -j \\ W_4^2 = W_4^6 = -1 \quad ; \quad W_4^3 = j \end{aligned} \qquad \text{Equação 32}$$

Substituindo a Equação 32 na Equação 31, tem-se a Equação 33.

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} \qquad \text{Equação 33}$$

Usando a propriedade da simetria, resulta-se na Equação 34.

$$\begin{aligned} X(0) &= x(0) + x(1) + x(2) + x(3) &&= \underbrace{[x(0) + x(2)]}_{g_1} + [x(1) + x(3)] \\ X(1) &= x(0) - jx(1) - x(2) + jx(3) &&= [x(0) - x(2)] - j\underbrace{[x(1) - x(3)]}_{g_2} \\ X(2) &= x(0) - x(1) + x(2) - x(3) &&= \underbrace{[x(0) + x(2)]}_{h_1} - [x(1) + x(3)] \\ X(3) &= x(0) + jx(1) - x(2) - jx(3) &&= [x(0) - x(2)] + j\underbrace{[x(1) - x(3)]}_{h_2} \end{aligned} \qquad \text{Equação 34}$$

Portanto, um algoritmo eficiente para a Equação 34, está demonstrado na Equação 35.

$$\begin{array}{c|c} \textit{Step}1 & \textit{Step}2 \\ g_1 = x(0) + x(2) & X(0) = g_1 + g_2 \\ g_2 = x(1) + x(3) & X(1) = h_1 + jh_2 \\ h_1 = x(0) - x(2) & X(2) = g_1 - g_2 \\ h_2 = x(1) - x(3) & X(3) = h_1 - jh_2 \end{array} \qquad \text{Equação 35}$$

O algoritmo representado na Equação 35, requer apenas 2 multiplicações complexas, que é um número consideravelmente menor, mesmo para este exemplo. O fluxograma para o algoritmo da Equação 35 é demonstrado na Figura 9.

Uma outra interpretação:

O eficiente algoritmo representado na Equação 35, pode ser interpretado de outra forma. Primeiro a seqüência de 4 pontos *x(n)* é dividida em seqüências de 2 pontos, que devem ser dispostas em vetores em forma de colunas, como mostrado na Equação 36 [1].

$$\begin{bmatrix} x(0) \\ x(2) \end{bmatrix}, \begin{bmatrix} x(1) \\ x(3) \end{bmatrix} = \begin{bmatrix} x(0) & x(1) \\ x(2) & x(3) \end{bmatrix} \qquad \text{Equação 36}$$

A menor DFT de 2 pontos de cada coluna e selecionado, como representado na Equação 37.

$$W_2 \begin{bmatrix} x(0) & x(1) \\ x(2) & x(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x(0) & x(1) \\ x(2) & x(3) \end{bmatrix} = \begin{bmatrix} x(0) + x(2) & x(1) + x(3) \\ x(0) - x(2) & x(1) - x(3) \end{bmatrix} = \begin{bmatrix} g_1 & g_2 \\ h_1 & h_2 \end{bmatrix} \qquad \text{Equação 37}$$

Então cada elemento da matriz resultante é multiplicado por W_4^{pq}, onde p é o índice da linha e q é o índice da coluna. A Equação 38, representa o denominado *dot-product* resultante da Equação 37.

$$\begin{bmatrix} 1 & 1 \\ 1 & -j \end{bmatrix} \begin{bmatrix} g_1 & g_2 \\ h_1 & h_2 \end{bmatrix} = \begin{bmatrix} g_1 & g_2 \\ h_1 & -jh_2 \end{bmatrix} \qquad \text{Equação 38}$$

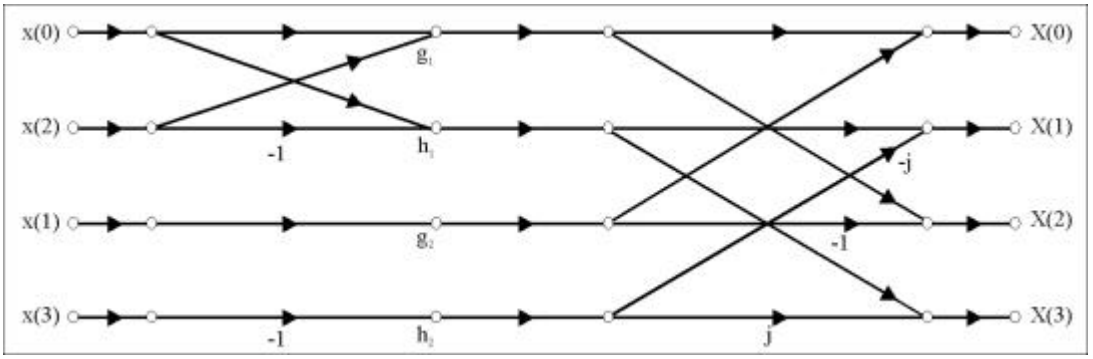

Figura 9. Fluxograma para o algoritmo da Equação 35
Fonte: Adaptado de Ingle [1] (1996)

Finalmente, as duas menores DFTs de dois pontos são selecionadas dos vetores linha, como demonstrado na Equação 39.

$$\begin{bmatrix} g_1 & g_2 \\ h_1 & jh_2 \end{bmatrix} W_2 \begin{bmatrix} g_1 & g_2 \\ h_1 & jh_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} g_1 & g_2 \\ h_1 & jh_2 \end{bmatrix} = \begin{bmatrix} X(0) & X(2) \\ X(1) & X(3) \end{bmatrix} \qquad \text{Equação 39}$$

Embora esta interpretação pareça possuir mais multiplicações que o algoritmo eficiente, ela sugere uma aproximação sistemática para computar um DFT maior, baseado em DFTs menores.

2.12.2. Aproximação por divisão e combinação.

Para reduzir a dependência quadrática computacional da DFT em *N*, deve-se escolher um número composto *N=LM* desde que atenda as especificações da Equação 40 [1].

$$L^2 + M^2 \ll N^2 \quad para\ N\ grande \qquad \text{Equação 40}$$

Agora, deve-se dividir a seqüência em *M* seqüências menores de largura *L*, tomando as menores *M* DFTs de *L* pontos, e então combiná-las em uma grande DFT usando as menores *L* DFTs de *M* pontos. Esta é a essência da aproximação por divisão e combinação. Estabelecendo *N = LM*, então os índices *n* e *k* na Equação 26, podem ser escritos como mostrado na Equação 41.

$$n = Mq + m, \quad 0 \le q \le L-1, \quad 0 \le m \le M-1$$
$$k = p + Lq, \quad 0 \le p \le L-1, \quad 0 \le q \le M-1 \qquad \text{Equação 41}$$

Deve-se então escrever as seqüências *x(n)* e *X(k)* como vetores *x(l,m)* e *X(p,q)*, respectivamente. Então a Equação 26 pode ser desenvolvida como na Equação 42.

$$\begin{aligned} X(p,q) &= \sum_{m=0}^{M-1}\sum_{l=0}^{L-1} x(l,m) W_N^{(Ml+m)(p+Lq)} \\ &= \sum_{m=0}^{M-1} W_N^{mp} \sum_{l=0}^{L-1} x(l,m) W_N^{Mlp} W_N^{Lmq} \\ &= \sum_{m=0}^{M-1} W_N^{mp} \sum_{l=0}^{L-1} x(l,m) W_L^{lp} W_M^{mq} \end{aligned} \qquad \text{Equação 42}$$

$$\underbrace{\qquad\qquad}_{L\text{-point DFT}} \qquad \underbrace{\qquad\qquad\qquad\qquad}_{M\text{-point DFT}}$$

Portanto, a Equação 42, pode ser implementada em três passos, como descrito a seguir.

1. Primeiro, deve-se computar o vetor DFT de *L* pontos para cada coluna *m*=0,...,*M*-1 , como descrito na Equação 43.

$$F(p,m) = \sum_{l=0}^{L-1} x(l,m) W_L^{lp}; \quad 0 \le p \le L-1 \qquad \text{Equação 43}$$

2. Segundo, deve-se modificar *F(p,m)* para obter outro vetor, como demonstrado na Equação 44.

$$G(p,m) = W_N^{pm} F(p,m), \quad \begin{matrix} 0 \le p \le L-1 \\ 0 \le m \le M-1 \end{matrix} \qquad \text{Equação 44}$$

O termo W_N^{pm} , na Equação 44, é chamado de termo *twiddle*.

3. Finalmente, deve-se computar as DFTs de *M* pontos para cada linha *p=0,...,L-1*, como demonstrado na Equação 45.

$$X(p,q) = \sum_{m=0}^{M-1} G(p,m) W_M^{mq}; \quad 0 \le q \le M-1 \qquad \text{Equação 45}$$

O número total de multiplicações complexas para esta aproximação pode ser encontrado através da Equação 46.

$$C_N ? ML^2 ? N ? LM^2 ? o(N^2) \qquad \text{Equação 46}$$

Este procedimento pode ser repetido se M ou L forem números compostos. Certamente, a algoritmo mais eficiente é obtido quando N é um número composto extremo, ou seja, $N ? R^?$. Estes algoritmos são conhecidos como algoritmos FFT *radix-R*. Quando $N ? R_1^{?1} R_2^{?2} \ldots$, então cada decomposição é conhecida como algoritmo FFT *mixed-radix*. O algoritmo mais fácil de programar e mais popular é o algoritmo FFT *radix-2* [1].

2.12.3. Algoritmo FFT Radix-2.

Tomando $N ? 2^?$; pode-se escolher $M=2$ e $L=N/2$ e dividir $x(n)$ em duas seqüências de $N/2$ pontos de acordo com a Equação 41, como demonstrado na Equação 47 [1].

$$\begin{matrix} g_1(n) ? x(2n) \\ g_2(n) ? x(2n ? 1) \end{matrix}; \quad 0 ? n ? \frac{N}{2} ? 1 \qquad \text{Equação 47}$$

A seqüência $g_1(n)$ contém amostras pares de $x(n)$ ordenadas, enquanto $g_2(n)$ possuí amostras ímpares de $x(n)$ ordenadas. Supondo $G_1(k)$ e $G_2(k)$ sendo DFTs de $N/2$ pontos de $g_1(n)$ e $g_2(n)$, respectivamente, então a Equação 42 pode ser reduzida como demonstrado na Equação 48.

$$X(k) ? G_1(k) ? W_N^k G_2(k), \quad 0 ? k ? N ? 1 \qquad \text{Equação 48}$$

A Equação 48 é chamada de *merging formula*, que combina duas DFTs de $N/2$ pontos em uma DFT de N pontos. O número total de multiplicações complexas é reduzido como demonstrado na Equação 49.

$$C_N ? \frac{N^2}{2} ? N ? o(N^2/2) \qquad \text{Equação 49}$$

Este procedimento pode ser repetido várias vezes. Em cada estágio as seqüências são dizimadas e as menores DFTs combinadas. Esta dizimação termina após *?* estágios quando se tem *N* seqüências de um ponto, que são também DFTs de um ponto. O procedimento resultante é chamado de *algoritmo FFT por dizimação no tempo (DIT-FFT)* [1], para o qual o número total de multiplicações complexas é demonstrado na Equação 50.

$$C_N \; ? \; N? \; ? \; N \log_2 N \qquad \text{Equação 50}$$

Certamente, se *N* for grande, então C_N é aproximadamente linear em *N*, este era o objetivo do algoritmo eficiente demonstrado anteriormente. Usando simetrias adicionais, C_N pode ser reduzido para $\frac{N}{2}\log_2 N$. O fluxograma para este algoritmo é mostrado na Figura 10.

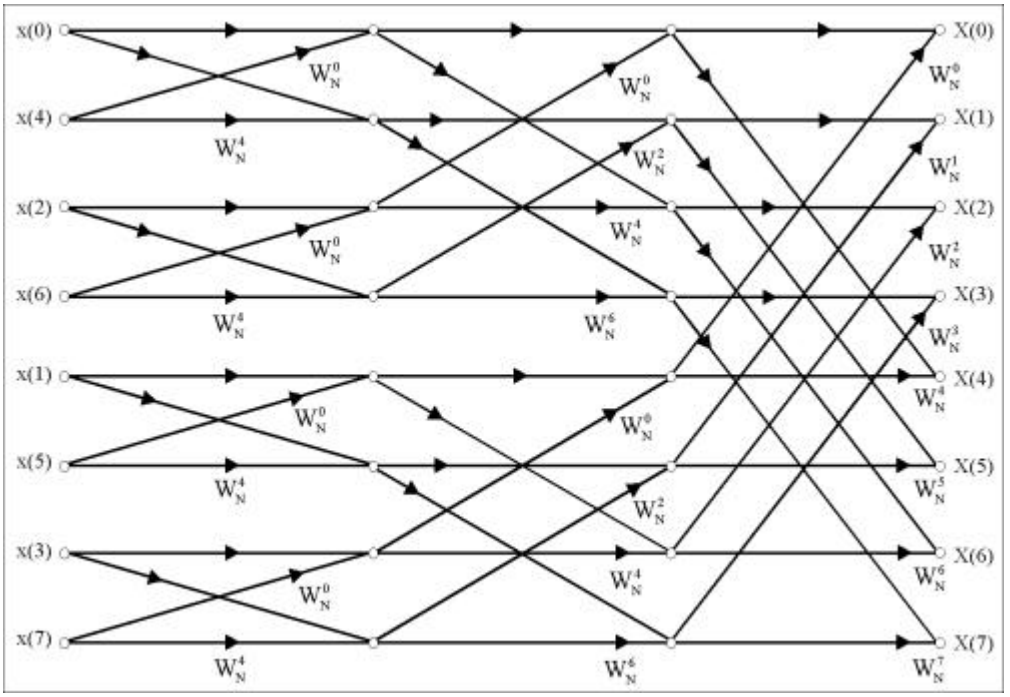

Figura 10. Estrutura de uma FFT por dizimação no tempo para *N* = 8.
Fonte: Adaptado de Ingle [1] (1996)

Em uma aproximação alternada, foi escolhido *L = 2*, *M = N/2* e foram seguindo os passos da Equação 42. As DFTs iniciais são DFTs de 2 pontos, que não contém multiplicações complexas.

Partindo da Equação 43, temos o desenvolvimento mostrado na Equação 51.

$$
\begin{aligned}
F(0,m) &= x(0,m) + x(1,m)W_2^0 \\
&= x(n) + x(n + N/2),\ 0 \le n \le N/2 \\
F(1,m) &= x(0,m) + x(1,m)W_2^1 \\
&= x(n) + x(n + N/2),\ 0 \le n \le N/2
\end{aligned}
$$

Equação 51

E partindo da Equação 44, temos o desenvolvimento mostrado na Equação 52.

$$
\begin{aligned}
G(0,m) &= F(0,m)W_N^0 \\
&= x(n) + x(n + N/2),\ 0 \le n \le N/2 \\
G(1,m) &= F(1,m)W_N^m \\
&= [x(n) + x(n + N/2)]W_N^n,\ 0 \le n \le N/2
\end{aligned}
$$

Equação 52

Estabelecendo $G(0,m) = d_1(n)$ e $G(1,m) = d_2(n)$ para $0 \le n \le N/2 - 1$ (desde que possa ser considerado como seqüências no domínio do tempo); então a partir da Equação 45 pode-se desenvolver a Equação 53.

$$
\begin{aligned}
X(0,q) &= X(2q) + D_1(q) \\
X(1,q) &= X(2q + 1) + D_2(q)
\end{aligned}
$$

Equação 53

Isto implica que os valores $X(k)$ da DFT são computados no modo de dizimação. Portanto esta aproximação é chamada de *algoritmo FFT por dizimação em freqüência (DIF-FFT)* [1]. O fluxograma da DIF-FFT é uma estrutura transposta da estrutura DIT-FFT, e sua complexidade computacional também é igual a $\frac{N}{2}\log_2 N$, como mostrado na Figura 11.

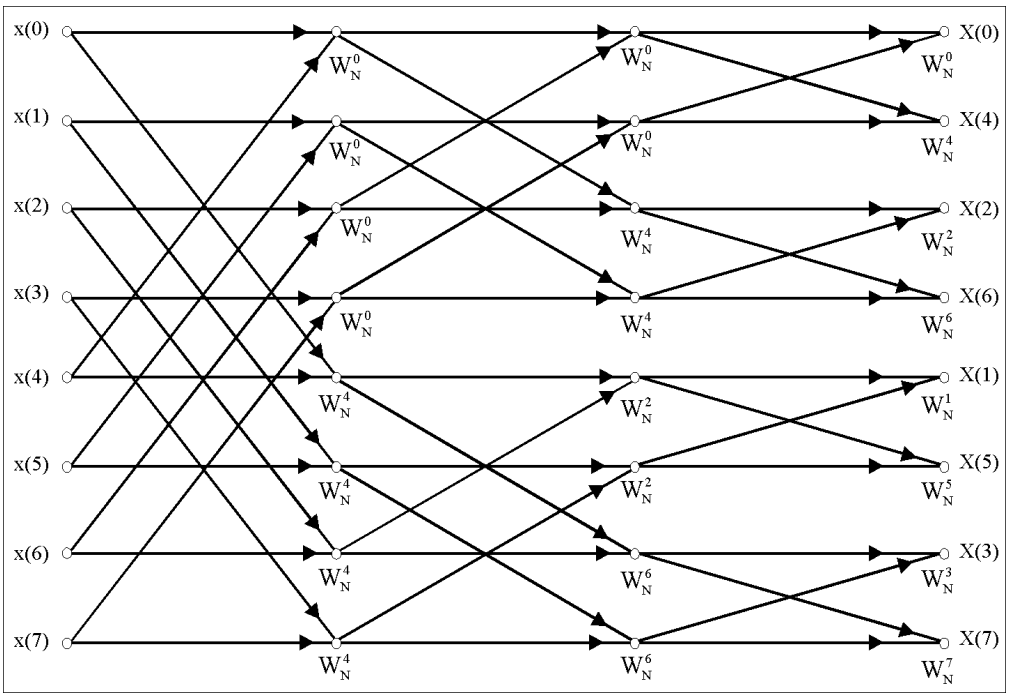

Figura 11. Estrutura de uma FFT por dizimação em freqüência para $N = 8$.

2.13.SELETIVIDADE EM FREQÜÊNCIA

Seletividade em freqüência é a habilidade de se distinguir diferentes componentes em freqüência do sinal de entrada [6]. Componentes entre dois pontos adjacentes discretos no espectro de freqüências não podem ser distinguidos.

A utilização de uma função de ponderação (janela de ponderação), a ser apresentada posteriormente, irá causar perdas na seletividade em freqüência. Porém, elas são necessárias para reduzir perdas resultantes do espalhamento espectral e da precisão finita do algoritmo utilizado, causados pelas características da função de saída do DFT, que contém um número finito de termos (janela retangular).

2.14.ESPALHAMENTO ESPECTRAL

Quando, para o cálculo da DFT, se toma N amostras para compor uma interpolação aproximada do sinal representado, gera-se uma descontinuidade nas extremidades da seqüência de tamanho finito devido à propriedade da periodicidade da DFT. Isto ocorre para todas as componentes dos sinais que não pertençam exatamente a um dos pontos discretos da série.

A Figura 12 demonstra este fenômeno, mostrando as amostragens resultantes se estas fossem obtidas diretamente do sinal versus a descontinuidade resultante da extensão periódica do sinal.

Esta descontinuidade exerce uma contribuição espectral (como um espalhamento de freqüências, de onde se origina o nome) através de todo o conjunto de valores de freqüência.

Funções de ponderação (janelas) reduzem a contribuição das amostragens próximas dos pontos limítrofes, e desta maneira reduzem a descontinuidade e seus efeitos na resposta em freqüência [6].

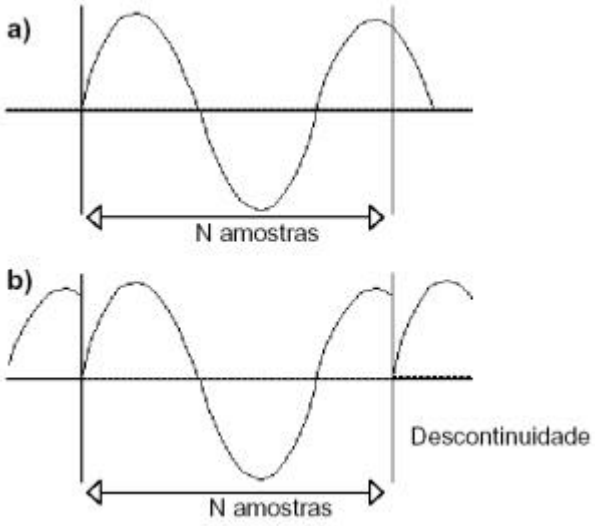

Figura 12. Descontinuidade resultante da extensão do sinal através do processo de DFT. (a) sinal contínuo; (b) sinal mostrando a descontinuidade assumida pelo processo de DFT.
Fonte: Adaptado de Scandelari [6]

A Figura 13 apresenta uma senóide de freqüência de 2 Hz, cuja extensão periódica no tempo não gera pontos de descontinuidade, e uma senóide de freqüência 2.25 Hz cuja extensão periódica no tempo gera um ponto de descontinuidade a cada 2.25 ciclos. Logo abaixo de cada senóide está representada a magnitude espectral da DFT de cada sinal. Notamos que a magnitude das componentes laterais do sinal de 2.25 Hz estão mais pronunciadas que as componentes laterais para o sinal de 2 Hz.

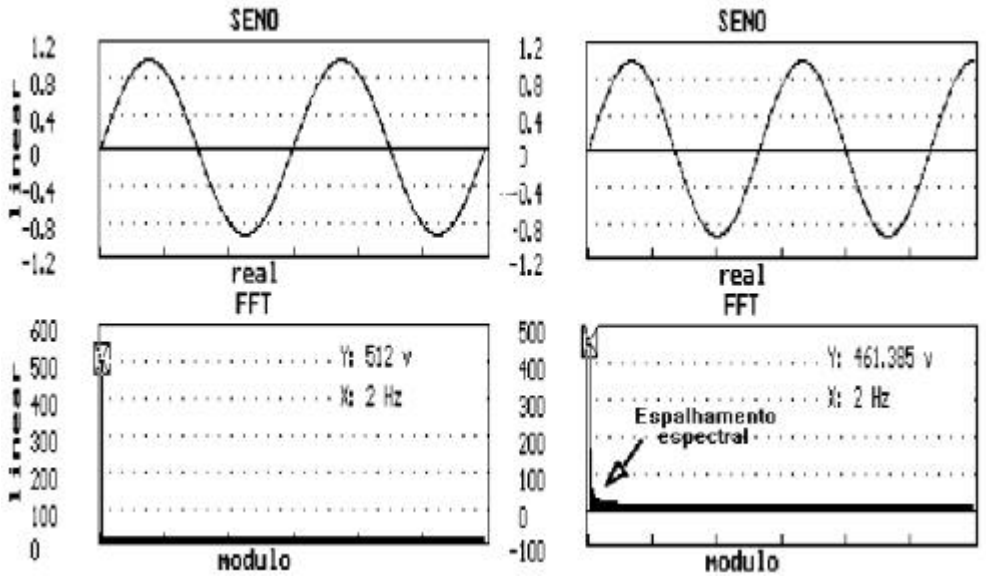

Figura 13. Dois sinais senoidais mostrando o espalhamento espectral.
Fonte: Adaptado de Scandelari [6]

2.15. JANELAS

As janelas ou funções de ponderação são utilizadas para modificar as características de resposta em freqüência dos algoritmos de DFT [6]. Funções de ponderação são usadas em conjunto com a análise espectral via DFT para reduzir o espalhamento em freqüência. A função de saída de uma DFT, resultante de uma função de entrada processada com uma janela, representa o produto de duas seqüências.

Desta forma, a saída do algoritmo de DFT é dado como descrito na Equação 54 abaixo:

$$Y_k = \sum_{n=0}^{N-1} a_n x_n W_N^{nk} \qquad \text{Equação 54}$$

onde a_n é a função de ponderação ou janela adotada.

Existem vários tipos de janelas. Cada qual com a sua respectiva função de ponderação e resposta em freqüência. Como exemplo podem ser citadas a janela de *Hamming*, a janela de *Hanning*, a janela de *Blackman-Harris*, a janela de *Kaiser*, a janela de *Barlett*, a janela Retangular e

a janela Triangular. Neste trabalho será abordada apenas a janela de *Hamming*, pois será utilizada na parte prática do trabalho.

2.15.1. Janela de HAMMING

A janela de *Hamming* utiliza os coeficientes 0,54 – 0,46 em sua equação para os termos em co-seno, como mostrado na Equação 55.

$$a_n = \begin{cases} 0{,}54 - 0{,}46\cos(2\pi n / N) & n = 0,1,2,\ldots, N-1 \\ 0 & \textit{demais} \end{cases} \qquad \text{Equação 55}$$

A resposta em freqüência está mostrada na Figura 14 e a curva da função na Figura 15. Nota-se que o lóbulo secundário está aproximadamente 43 dB abaixo do lóbulo principal.

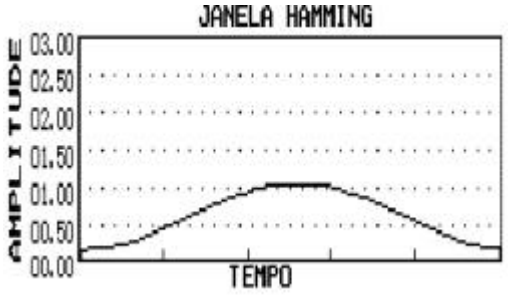

Figura 14. Janela de Hamming – Função de ponderação.
Fonte: Adaptado de Scandelari [6]

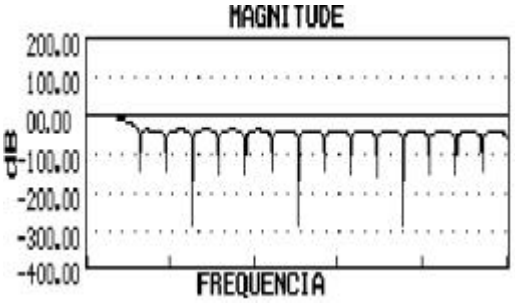

Figura 15. Janela de Hamming – Resposta em freqüência
Fonte: Adaptado de Scandelari [6]

3. PROJETO

3.1. IMPLEMENTAÇÃO DOS MODELOS NO SIMULINK.

3.1.1. Bloco "FFT" do Simulink e a função "fft" do MATLAB.

O bloco FFT demonstrado da Figura 16, realiza a operação da transformada rápida de Fourier em qualquer canal de uma entrada M por N ou uma entrada de largura M (esta entrada está representada na Figura 16 pela letra "u"), onde M deve ser uma potência de dois. Para trabalhar com outros tamanhos de entrada, deve-se utilizar o bloco *Zero Pad* para arredondar ou truncar a dimensão de largura M numa largura de potência de dois.

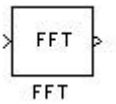

Figura 16. Bloco denominado “FFT”.

A saída do bloco FFT é equivalente à função fft do MATLAB, como mostrado na Figura 17.

```
>> Y = fft(u)     % Comando do MATLAB equivalente ao bloco FFT.
```

Figura 17. Comando do MATLAB equivalente ao bloco FFT.

A entrada de número = "k" do canal de saída de número = "l", ou seja y(k, l), é igual ao ponto de número = "k" da transformada discreta de Fourier (DFT) do canal de entrada de número = "l", como demonstrado na Equação 56.

$$Y(k,l) = \sum_{m=1}^{M} u(m,l)e^{-j2\pi(m-1)(k-1)/M} \qquad k = 1,...,M \qquad \text{Equação 56}$$

O bloco FFT suporta entradas reais e complexas de ponto flutuante e ponto fixo.

3.1.2. Algoritmos utilizados para a computação FFT pelo MATLAB e Simulink.

Dependendo se o bloco de entrada gera informações do tipo ponto fixo ou ponto flutuante, com valores reais ou complexos, e se desejar ter uma saída de ordem linear ou em bit invertido, o bloco deve utilizar um ou mais dos algoritmos descritos abaixo, como resumido na Tabela 2 e na Tabela 3 a seguir:

?? *Butterfly operation*

?? *Double-signal algorithm*

?? *Half-length algorithm*

?? *Radix-2 decimation-in-time (DIT) algorithm*

?? *Radix-2 decimation-in-frequency (DIF) algorithm*

Tabela 2. Algoritmos utilizados para sinais de entrada no formato de ponto flutuante:

Tipo da entrada	**Ordem da saída**	**Algoritmos utilizados para a computação da FFT**
Complexa	Linear	*Butterfly operation* e *Radix-2 DIT*
Complexa	Bit invertido	*Radix-2 DIF*
Real	Linear	*Butterfly operation* e *Radix-2 DIT* em conjunto com os algoritmos *Half-lenght* e *Double-signal*
Real	Bit invertido	*Radix-2 DIF* em conjunto com os algoritmos *Half-lenght* e *Double-signal*

Tabela 3. Algoritmos utilizados para sinais de entrada no formato de ponto fixo:

Tipo da entrada	**Ordem da saída**	**Algoritmos utilizados para a computação da FF**
Real ou complexa	Linear	*Butterfly operation* e *Radix-2 DIT*
Real ou complexa	Bit invertido	*Radix-2 DIF*

3.1.3. Modelo para a visualização do espectro de freqüências de um sinal senoidal.

Para validar o conceito de análise espectral de sinais utilizando FFT, foi inicialmente desenvolvido um modelo para análise espectral de um sinal senoidal, conforme mostrado na Figura 18.

O bloco denominado *Sine-Wave*, representa o sinal senoidal digitalizado a ser analisado no domínio da freqüência.

O bloco denominado *Buffer* converte amostras escalares em um frame de saída com uma taxa de amostragem baixa.

O bloco denominado *Short-Time FFT* que tem o nome genérico de *Periodogram* na biblioteca do Simulink , possui diversos parâmetros, como por exemplo, a escolha de uma função de janela para realizar uma seletividade de freqüências no sinal senoidal a ser analisado, neste caso foi utilizada a janela de *Hamming*. Este bloco também aplica a FFT sobre o sinal, assim como o bloco denominado "FFT" da Figura 16.

O bloco denominado *Short-Time Spectrum* que tem o nome genérico de *Spectrum Scope* na biblioteca do Simulink, tem a função de gerar um gráfico para que seja possível a visualização do espectro de freqüências do sinal senoidal.

A Figura 18 , ilustra o modelo denominado **"senoide.mld"**:

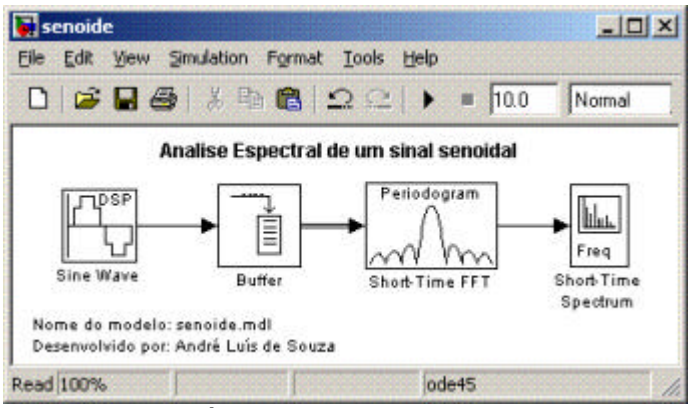

Figura 18. Modelo do Simulink para a análise espectral de um sinal senoidal.

3.1.4. Modelo para gravar sinais de áudio em formato *.wav.

Para gravar sinais de áudio em format *.wav, foi desenvolvido o modelo mostrado na Figura 19.

A função deste modelo foi possibilitar a gravação de um sinal de áudio através de um microfone conectado à placa de som do computador. Para isso, foi utilizado um bloco denominado *From Wave Device*, que tem a função de capturar o sinal de som proveniente do dispositivo de áudio instalado no computador, neste caso através de um microfone. Foi utilizado também um bloco denominado *To Wave File*, que tem a função de gravar o sinal capturado em um arquivo de tipo *.wav, neste caso o nome do arquivo foi definido como "**audio.wav"**.

A Figura 19, ilustra o modelo denominado "**recordwave.mdl**".

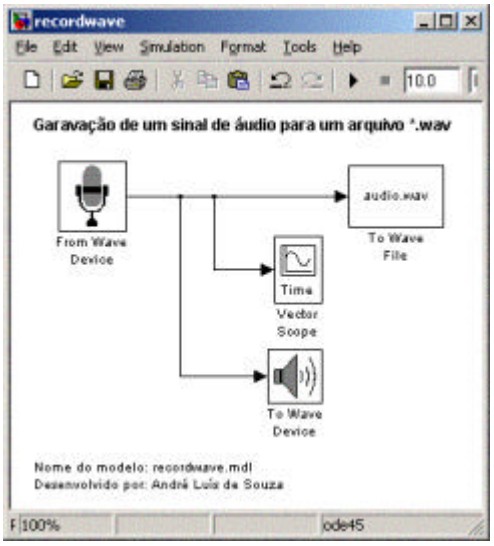

Figura 19. Modelo do Simulink para gravação de um sinal de áudio em um arquivo de formato *.wav.

3.1.5. Modelo para aplicar a FFT sobre o sinal de áudio previamente gravado.

A Figura 20, mostra o modelo desenvolvido para aplicar a FFT sobre um sinal de áudio previamente gravado em formato *.wav.

Para a simulação deste modelo, foi utilizado o sinal de áudio previamente gravado em formato *.wav, com a utilização do modelo denominado "**recordwave.mdl**".

O bloco denominado *Buffer* converte amostras escalares em um *frame* de saída com uma taxa de amostragem baixa.

Foi utilizada a janela de *Hamming* para reduzir o espalhamento em freqüência, e o comprimento da janela igual a 128. O parâmetro de escolha da função janela pode ser ajustado dentro do bloco denominado *Short-Time FFT*, assim também como os parâmetros sobre a aplicação da FFT.

A Figura 20 ilustra o modelo denominado **"aes_fft.mdl"**.

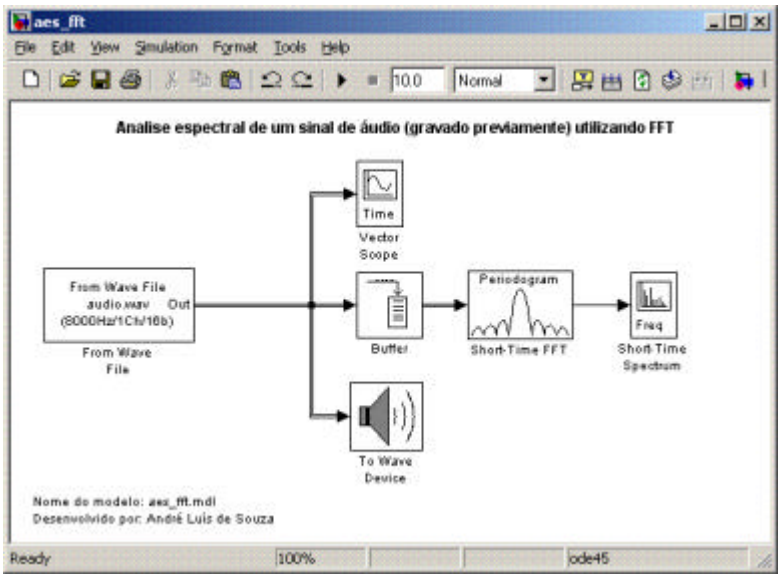

Figura 20. Modelo do Simulink para análise espectral utilizando FFT de um sinal de áudio previamente gravado em um arquivo de formato *.wav.

3.1.6. Modelo para aplicar a FFT sobre o sinal de áudio capturado em tempo real.

A Figura 21 mostra o modelo desenvolvido para análise espectral de sinais em tempo real. A viabilidade da utilização deste modelo depende da largura de banda do sinal.

Para a criação deste modelo, foi utilizado um microfone conectado a entrada denominada "**mic**" da placa de som do computador.

A diferença entre este modelo (**aes_fft_mic.mdl**) e o anterior (**aes_fft.mdl**), é que neste o sinal de áudio é analisado em tempo real, ou seja, não é necessário uma gravação prévia do sinal de áudio.

Isso foi possível, alterando o bloco do sinal de entrada (*From Wave File)* para o bloco *From Wave Device*, que captura o sinal do microfone que está conectado à placa de som do computador, em tempo real.

O sinal capturado do microfone passa pelo bloco denominado *Bufffer* converte amostras escalares em um *frame* de saída com uma taxa de amostragem baixa, logo em seguida as amostras passam pelo bloco denominado *Short-Time FFT* onde é aplicada uma função janela de *Hamming* com comprimento de 128, e logo após é aplicada a FFT sobre o sinal, para que assim seja possível visualizar seu espectro de freqüências.

A Figura 21 ilustra o modelo denominado **"aes_fft_mic.mdl"**.

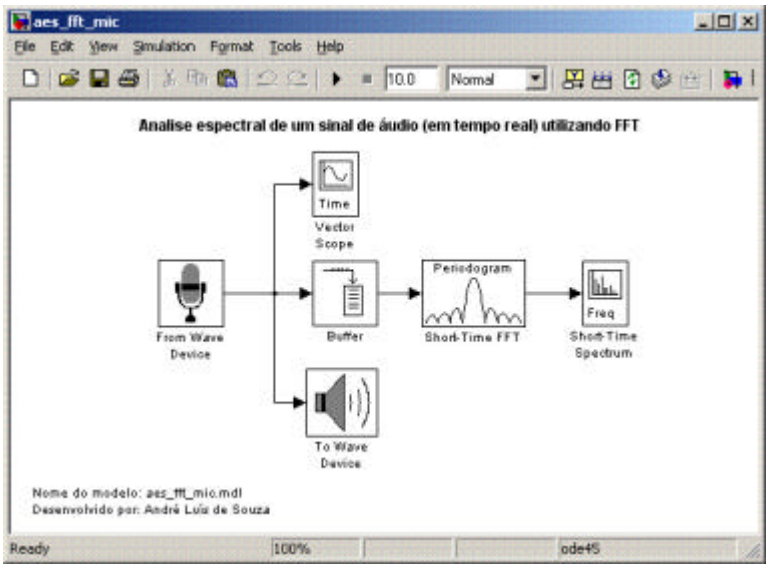

Figura 21. Modelo do Simulink para análise espectral de um sinal de áudio em tempo real.

3.2. DESCRIÇÃO DA FUNCIONALIDADE E PARÂMETROS DOS BLOCOS UTILIZADOS.

Segue a descrição da funcionalidade e parâmetros de cada bloco utilizado nos modelos desenvolvidos no Simulink.

3.2.1. Bloco - *Sine Wave*

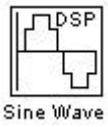

Figura 22. Bloco "*Sine Wave*".

Saída de amostras de um sinal senoidal. Para gerar mais que uma senóide simultaneamente, deve-se entrar com um vetor contendo valores para os parâmetros amplitude, freqüência e deslocamento de fase.

3.2.2. Bloco - *From Wave Device*

Figura 23. Bloco "*From Wave Device*".

Faz a leitura dos dados amostrados de um dispositivo de áudio padrão Windows, em tempo real.

Este bloco somente pode ser utilizado em plataformas Windows de 32-bit (WIN32).

3.2.3. Bloco - *From Wave File*

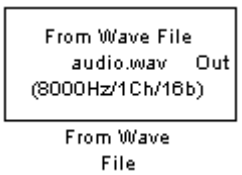

Figura 24. Bloco "*From Wave File*".

Faz a leitura dos dados amostrados de um arquivo de áudio padrão PCM Windows em formato ".WAV". Quando em *loop*, deve-se entrar o número de vezes a ser lido o arquivo de dados, ou entrar com o parâmetro "inf" para obter um loop infinito.

Este bloco somente pode ser utilizado em plataformas Windows de 32-bit (WIN32).

3.2.4. Bloco - *Vector Scope*

Figura 25. Bloco "*Vector Scope*".

Exibe um vetor ou matriz no domínio do tempo, domínio da freqüência ou dados especificados pelo usuário. Qualquer coluna de uma entrada tipo matriz 2-D é plotada como um canal de dados separado. Entrada 1-D são assumidas como um canal de dados simples.

Para operações no domínio da freqüência, a entrada deve vir de uma fonte como um bloco de Magnitude FFT, ou um bloco com uma equivalente organização dos dados.

3.2.5. Bloco - *To Wave Device*

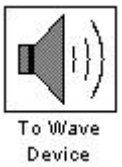

Figura 26. Bloco "*To Wave Device*".

Escreve os dados de amostras em um dispositivo de áudio padrão Windows em tempo real. Se pulsos aleatórios de áudio ocorrem, adiciona-se um atraso inicial na saída (para que os pulsos aleatórios comecem) ou duração da fila. Dados de entrada com precisão do tipo *double*, *single* e *int16* são reproduzidos usando amostragem de 16 bits; dados do tipo *uint8* são reproduzidos usando amostragem de 8 bits.

Este bloco somente pode ser utilizado em plataformas Windows de 32-bit (WIN32).

3.2.6. Bloco - To Wave File

Figura 27. Bloco "*To Wave File*".

Escreve os dados amostrados em um arquivo de áudio padrão PCM Windows em formato ".WAV".

Este bloco somente pode ser utilizado em plataformas Windows de 32-bit (WIN32).

3.2.7. Bloco - *Buffer*

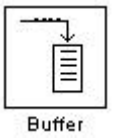

Figura 28. Bloco "*Buffer*".

Converte amostras escalares em um "frame" de saída com uma taxa de amostragem baixa. Pode-se converter um frame opcionalmente para um tamanho menor ou maior utilizando *overlap*.

Para cálculo do atraso da amostra, observar a função "*rebuffer_delay*".

3.2.8. Bloco – Periodogram

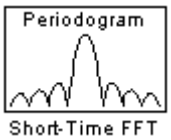

Figura 29. Bloco "Periodogram".

Estimação não paramétrica usando o método *Periodograma*. Este bloco possui internamente parâmetros para acerto da função de janela e FFT.

A Figura 30 mostra os parâmetros utilizados no bloco *Periodogram*.

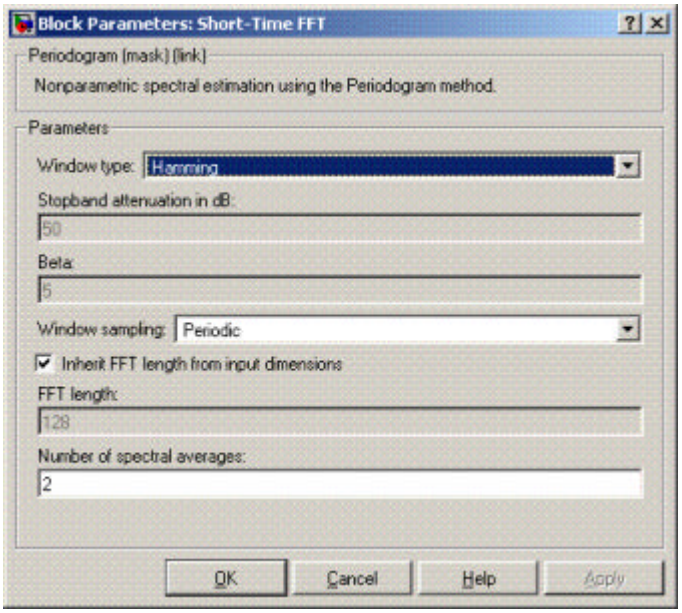

Figura 30. Parâmetros utilizados no bloco denominado *Periodogram.*

Neste trabalho, o bloco Periodogram, foi configurado com os seguintes parâmetros:

- Função Janela de *Hamming*
- *Stopband attenuation in dB*: 50
- Beta: 5
- *Window sampling: Periodic*
- *Inherit FFT length from input dimensions* (Selecionado)
- FFT *length*: 128
- *Number of spectral averages*: 2

3.2.9. Bloco - Spectrum Scope

Figura 31. Bloco "*Spectrum Scope*".

Computa e exibe o periodograma de qualquer sinal de entrada. Blocos de entrada não baseados em *frames* devem utilizar a opção de buferização.

3.3. VISUALIZAÇÃO DOS RESULTADOS ATRAVÉS DE GRÁFICOS.

3.3.1. Modelo “senoide.mdl”

As próximas figuras mostrarão os resultados obtidos dos modelos desenvolvidos neste trabalho.

A seguir, pode-se observar o resultado obtido com o modelo denominado “**senoide.mld**”, onde a Figura 32 demonstra a visualização das magnitudes do espectro de freqüências do sinal senoidal utilizado como entrada.

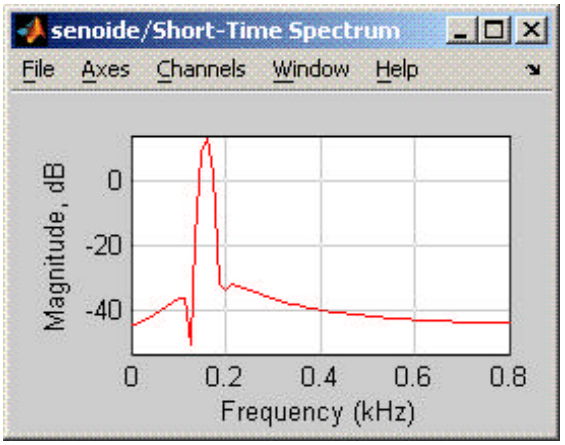

Figura 32. Magnitudes do espectro de freqüências do sinal senoidal utilizado com entrada no modelo “senoide.mld”.

3.3.2. Modelo “recordwave.mdl”

A Figura 33, exibe as amplitudes do sinal de áudio gravado em formato *.wav no domínio do tempo, com a utilização do modelo denominado “**recordwave.mdl**”.

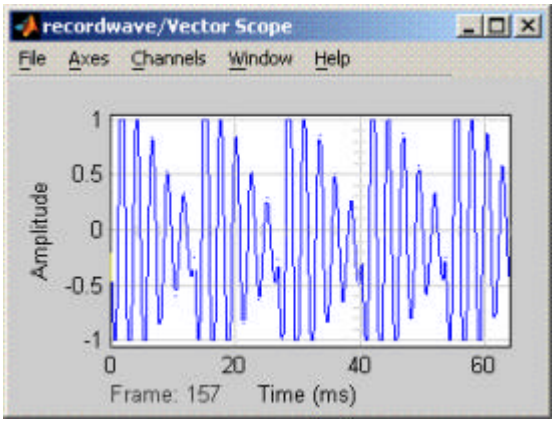

Figura 33. Amplitudes de sinal de áudio gravado em formato *.wav no domínio do tempo, com a utilização do modelo "recordwave.mdl".

3.3.3. Modelo "aes_fft.mdl"

Na Figura 34, é mostrado o espectro de freqüências do sinal utilizado como entrada no modelo denominado "**aes_fft.mdl**".

Pode-se observar a harmônica principal nas proximidades da freqüência de 800 Hz, a harmônica secundária nas proximidades da freqüência 2 kHz e muitas outras harmônicas provenientes de um sinal de voz.

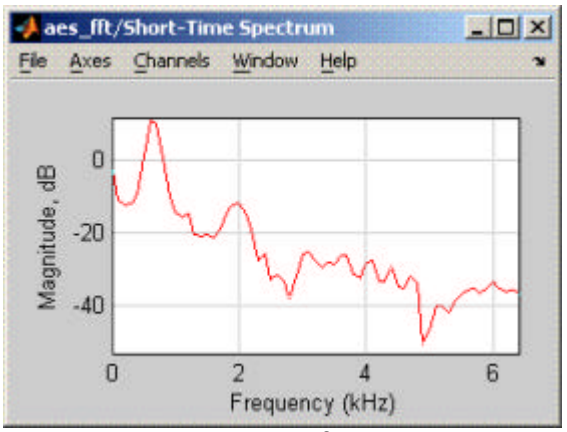

Figura 34. Espectro de freqüências do sinal gravado em formato *.wav, visualizado através do modelo "aes_fft.mdl".

3.3.4. Modelo “aes_fft_mic.mdl”

Na Figura 35, é possível observar as amplitudes do sinal de áudio capturado em tempo real pelo modelo denominado “**aes_fft_mic.mdl**”, através de microfone conectado à placa de som do computador.

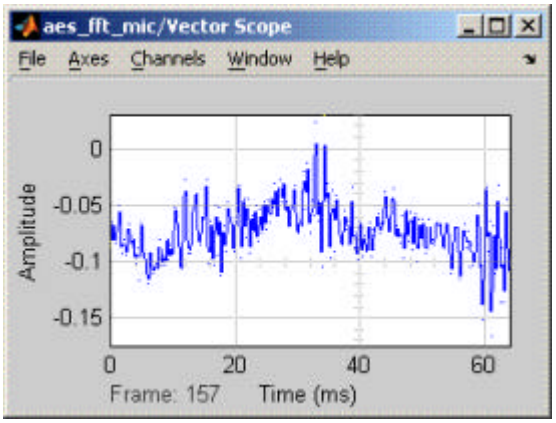

Figura 35. Amplitudes do sinal de áudio capturado em tempo real pelo modelo “aes_fft_mic.mdl”.

Na Figura 36, é mostrado o espectro de freqüências do sinal de áudio capturado em tempo real pelo modelo “aes_fft_mic.mdl”, através de um microfone conectado à placa de som do computador. Observa-se diversas harmônicas provenientes de um sinal de voz.

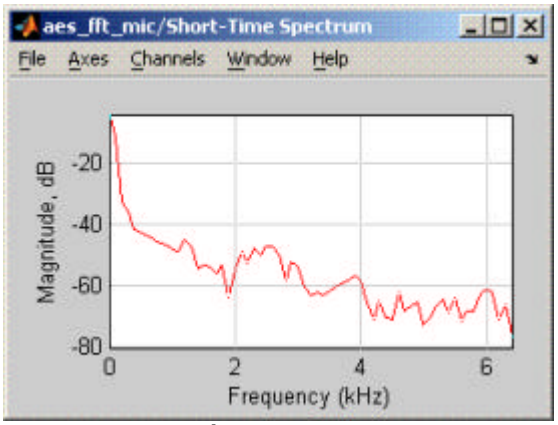

Figura 36. Espectro de freqüências do sinal de áudio (voz) capturado em tempo real pelo modelo “aes_fft_mic.mdl”.

4. CONCLUSÕES

Neste trabalho foi desenvolvido um sistema de análise espectral de sinais, em tempo real e não real, utilizando-se a transformada Rápida de Fourier a partir de modelos do Simulink do MATLAB.

Inicialmente, foi realizada uma revisão teórica dos principais aspectos relacionados com aquisição de sinais e cálculo de DFT. Esta revisão teórica mostrou-se essencial para o entendimento e utilização de diversos modelos funcionais do Simulink.

Foram realizadas diversas análises espectrais de sinais utilizando-se o sistema desenvolvido, o que possibilitou a obtenção dos resultados teoricamente esperados, tanto em tempo real quanto em tempo não real.

Finalmente, concluiu-se também que o Simulink e o MATLAB mostram-se duas ferramentas muito eficientes para análise espectral de sinais previamente gravados ou em tempo real, pois possuem funções e blocos que utilizam diversos tipos de algoritmos FFT, que são utilizados eficientemente para reduzir o número de cálculos computacionais, de acordo com o tipo de dado a ser analisado.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] – INGLE, Vinay K.; PROAKIS, John G.; *Digital Signal Processing Using MATLAB*; Bookware Companion Series TM; Brooks/Cole Thomson Learning.

[2] – PROAKIS, John G.; MANOLAKIS, Dimitris G.; *Digital Signal Processing – Principles, Algorithms, And Aplications*; Third Edition; Prentice Hall.

[3] - MATSUMOTO, Élia Y.*; MATLAB 6.5 – Fundamentos de Programação*; Editora Érica.

[4] – HANSELMAN, Duane; LITTLEFIELD, Bruce*; MATLAB 5 Guia Do Usuário*; Makron Books.

[5] - MATSUMOTO, Élia Y.; SIMULINK 5 – Fundamentos; 2ª Edição; Editora Érica.

[6] – SCANDELARI, Luciano; Transformada Rápida de Fourier; Centro Federal de Educação Tecnológica do Paraná.