**Isabelle Reis Lund**

# Contribuições à geração de tráfego fractal por meio da transformada *wavelet*

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

São Paulo
2008

**Isabelle Reis Lund**

# Contribuições à geração de tráfego fractal por meio da transformada *wavelet*

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Elétrica.

Área de concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. José Roberto de Almeida Amazonas

São Paulo
2008

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com anuência de seu orientador.

São Paulo, 10 de Julho de 2008.

Assinatura do autor

Assinatura do orientador

**Ficha Catalográfica**

Lund, Isabelle Reis
Contribuições à geração de tráfego fractal por meio da transformada *wavelet*. São Paulo, 2008. 153 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1. Fractal 2. Análise de séries temporais 3. Rede de Comunicação I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle. II. Título.

Para minha filha Emanuelle, minha mãe Gisele e Amarildo

# Agradecimentos

À minha filha Emanuelle por ser a razão da minha vida.

À minha mãe Gisele por ser um exemplo a seguir.

Ao meu marido Amarildo pelo apoio em todos os momentos.

Ao meu orientador Prof. Dr. José Roberto Amazonas por todo apoio e amizade.

Ao amigo Alexandre Barbosa de Lima pela presença, ajuda, solidariedade e amizade nos momentos mais críticos.

Ao amigo Fernando Lemos de Mello pelo apoio no início deste trabalho.

# Resumo

Estudos mostraram que o tráfego nas redes de dados tanto locais quanto de grande área, possui propriedades fractais como dependência de longa duração - *Long-Range Dependence* (LRD) e auto-similaridade. Devido à heterogeneidade de aplicações nessas redes, os *traces* de tráfego podem apresentar dependência de longa duração - *Long Range Dependence* (LRD), dependência de curta duração - *Short Range Dependence* (SRD) ou uma mistura de LRD com SRD.

Sendo assim, este trabalho tem como objetivo sintetizar séries temporais gaussianas com flexibilidade de processamento no plano tempo-freqüência a serem inseridas num gerador de tráfego com as características estatísticas específicas do tráfego encontrado em redes por comutação de pacotes reais, como auto-similaridade, LRD e SRD.

Para isto foram desenvolvidos dois métodos para síntese de séries temporais gaussianas com LRD e simultânea introdução de SRD em diferentes faixas de freqüência: *Discrete Wavelet Tansform* (DWT) com mapa de variâncias e *Discrete Wavelet Packet Tansform* (DWPT). Estes métodos utilizaram o mapa de variâncias cujo conceito foi desenvolvido neste trabalho.

A validação dos métodos foi feita através de análise estatística e comparação com resultados de séries geradas pelo método *Discrete Wavelet Transfom* (DWT) de Bäckar utilizado em [1]. Além disso, também foi validada a idéia de que a DWPT é mais interessante que a DWT por ser mais flexível e prover uma maior flexibilidade de processamento no plano tempo-freqüência.

# Abstract

Studies demonstrated that the data network traffic of Local Area Network (LAN) and Wide Area Network has fractal properties as long range dependence (LRD) and self-similarity. The traffic traces can show long range dependence, short range dependence or the both behaviors because of applications heterogeneity in these networks.

This work objective is to synthetisize gaussian time series with processor flexibility in the time-frequency plan to be inserted in a traffic generator with the specific statistical traffic characteristics of real packet networks such as self-similarity, long range dependence (LRD) and short range dependence (SRD).

Two methods were developed for the gaussian time series with LRD and SRD synthesis: *Discrete Wavelet Tansform* (DWT) with variance map and *Discrete Wavelet Packet Tansform* (DWPT). These methods used the variance map which concept was developed in this work.

The methods validation was done by statistic analysis and comparison with the time series generated by the Bäckar *Discrete Wavelet Transfom* (DWT) used by [1]. Besides of this, the idea that the DWPT is more interesting than the DWT because of its processing flexibility in the time-frequency plan was validated.

# Sumário

**Lista de Figuras**

**Lista de Tabelas**

**Lista de Abreviaturas**

**1 Introdução** **1**

1.1 Objetivo . . . . . 1

1.2 Motivação . . . . . 3

1.3 Organização do Trabalho . . . . . 6

**2 Revisão Teórica e Trabalhos Correlacionados** **7**

2.1 Geração de Tráfego . . . . . 7

2.2 Caracterização e Síntese de tráfego . . . . . 11

2.2.1 Fractais . . . . . 12

2.2.2 O Expoente de Hurst . . . . . 14

2.2.3 LRD e Auto-Similaridade . . . . . 15

2.2.4 Processo FGN . . . . . 19

2.3 *Wavelet* . . . . . 21

2.3.1 Transformada *Wavelet* Contínua - *Continuous Wavelet Transform* - CWT . . . . . 21

2.3.2 Transformada *Wavelet* Discreta - *Discrete Wavelet Transform* DWT . . . . . 23

2.3.3 Maximal Overlap DWT - MODWT . . . . . 42

2.3.4 *Discrete Wavelet Packet Transform* - DWPT . . . . . 46

2.3.5 Mapa de Variâncias . . . . . 55

2.3.6 Considerações finais . . . . . 59

**3 Geração de Séries Temporais** **60**

3.1 Síntese de Séries Temporais via *Discrete Wavelet Transform* (DWT) . . . . . 60

3.1.1 Geração de Séries Temporais Gaussianas com LRD pelo método DWT de Bäckar . . . . . 63

3.1.2 Geração de Séries Temporais Gaussianas com LRD pelo método DWT com mapa de variâncias . . . . . 64

3.1.3 Geração de Séries Temporais Gaussinas com LRD com periodicidade pelo método DWT com mapa de variâncias . . 67

3.2 Análise de Séries Temporais via DWT . . . . . 69

3.3 Síntese de Séries Temporais via *Discrete Wavelet Packet Transform* (DWPT) . . . . . 71

3.3.1 Geração de Séries Temporais Gaussianas com LRD pelo método DWPT . . . . . 72

3.3.2 Geração de Séries Temporais Gaussinas com LRD com periodicidade pelo método DWPT . . . . . 76

3.4 Análise de Séries Temporais via DWPT . . . . . 81

**4 Validações** **82**

4.1 Comparação dos métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT . . . . . 83

4.2 Comparação entre as séries geradas sem e com SRD via DWT com mapa de variâncias e DWPT . . . . . 87

4.2.1 Faixa de freqüência: 0,15 - 0,25 . . . . . 88

4.2.2 Faixa de freqüência: 0,25 - 0,35: . . . . . 90

4.2.3 Faixa de freqüência: 0,35 - 0,45: . . . . . 90

4.3 Comparação entre as séries geradas com periodicidade pelo DWT com mapa de variâncias e DWPT . . . . . 94

4.3.1 Método DWT com mapa de variâncias . . . . . 97

4.3.2 Método DWPT . . . . . 97

**5 Conclusões e Trabalhos Futuros** **104**

**Referências** **106**

**Apêndice A – Códigos MATLAB - Síntese pelo método DWT de Bäckar** **110**

**Apêndice B – Códigos MATLAB - Síntese pelo método DWT com mapa de variâncias** **117**

**Apêndice C – Códigos MATLAB - Síntese pelo método DWPT** **127**

**Apêndice D – Códigos MATLAB - Análise DWT e DWPT** **140**

**Apêndice E – Códigos MATLAB - Síntese de séries com periodicidade** **144**

# Lista de Figuras


1.1 Árvore multiescala do tráfego. Cada nó é a soma dos seus dois nós filhos. Fonte: [2] página 1008. . . . . 5

2.1 Diagrama de Blocos do Gerador de Tráfego. . . . . 10

2.3 Ilustração das cinco primeiras iterações para obtenção do conjunto de Cantor. Repare-se que o formato de uma determinada parte do objeto, quando magnificada, se assemelha ao formato do todo (auto-similaridade). . . . . 14

2.4 Visualização do tráfego *Fast* Ethernet (100 Mbps) coletado num servidor WWW/Email/FTP da Universidade de Drexel [3] em quatro diferentes níveis de agregação: 10 ms, 100 ms, 1 s e 10 s (de cima para baixo). Este *trace* tem um tamanho total de 3 h. . . . . 15

2.5 DEPs para modelos AR(4) e FD(0,4) de mesma potência. . . . . 16

2.6 Vetores linha $\mathcal{W}_{n\bullet}^T$ da matriz transformada *wavelet* discreta $\mathcal{W}$ baseada na *wavelet* de Haar para $N = 16$ e $n = 0$ a 7 (de cima para baixo à esquerda) e $n = 8$ a 15 (direita) - fonte: [4] página 57. 24

2.7 Vetores linha $\mathcal{W}_{n\bullet}^T$ da matriz transformada *wavelet* discreta $\mathcal{W}$ baseada na *wavelet* D(4) para $N = 16$ e $n = 0$ a 7 (de cima para baixo à esquerda) e $n = 8$ a 15 (direita). Fonte: [4] página 60. . . 27

2.8 Funções ganho quadrado para o filtro *wavelet* de Haar (gráfico acima à esquerda), filtro de escala de Haar (gráfico acima à direita. Filtro *wavelet* de D(4) (gráfico abaixo à esquerda) e filtro de escala D(4) (gráfico abaixo à direita). As linhas pontilhadas marcam a freqüência $f = 1/4$. Fonte: [4] página 73. . . . . 31

2.9 Decomposição Piramidal. Um sinal de entrada com quatro bandas de interesse é decomposto resultando no aumento na resolução espectral dos componentes de baixa freqüência. Fonte: [5] página 19. . . . . 34

2.10 Reconstrução Piramidal. Fonte: [5] página 19. . . . . 34

2.11 Análise de Multirresolução. Fonte: [5] página 20. . . . . 35

2.12 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_1$ e $\mathbf{V}_1$. A série temporal no vetor $\mathbf{X}$ de tamanho $N$ é circularmente filtrada utilizando o filtro *wavelet* $H(\cdot)$ periodizado ao tamanho $N$ (a forma desse filtro no domínio de freqüência é dada por $\{H(\frac{k}{N}) : k = 0, \ldots, N-1\}$). Todos os valores de índices ímpares da série filtrada são utilizados para formar o vetor $\mathbf{W}_1$ de tamanho $N/2$ contendo os coeficientes *wavelet* de nível $j = 1$("$\downarrow 2$" indica subamostragem por dois); de maneira similar, o vetor $\mathbf{V}_1$ de tamanho $N/2$ contendo os coeficientes de escala de nível $j = 1$ é obtido pela subamostragem da saída da filtragem de $\mathbf{X}$ com o filtro de escala $G(\cdot)$ periodizado ao tamanho $N$. Fonte: [4] página 80. . . . . 36

2.13 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_1$, $\mathbf{W}_2$ e $\mathbf{V}_2$. Fonte: [4] página 89. . . . . 38

2.14 Diagrama de fluxo ilustrando a síntese de $\mathbf{X}$ a partir de $\mathbf{W}_1$, $\mathbf{W}_2$ e $\mathbf{V}_2$. Fonte: [4] página 90. . . . . 39

2.15 Diagrama de fluxo ilustrando a análise de $\mathbf{V}_{j-1}$ em $\mathbf{W}_j$, e $\mathbf{V}_j$ seguido pela síntese de $\mathbf{V}_{j-1}$ a partir de $\mathbf{W}_j$, e $\mathbf{V}_j$ - fonte: [4] página 94. . . . . 40

2.16 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ nos coeficientes *wavelet* e de escala MODWT $\widetilde{\mathbf{W}}_1$ e $\widetilde{\mathbf{V}}_1$ de primeiro nível, seguido da síntese de $\mathbf{X}$ a parir de $\widetilde{\mathbf{W}}_1$ e $\widetilde{\mathbf{V}}_1$. Fonte: [4] página 169. . . . . 45

2.17 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{2,0}$, $\mathbf{W}_{2,1}$, $\mathbf{W}_{2,2}$, $\mathbf{W}_{2,3}$ (ordem de seqüência). $N_1 \equiv N/2$. Fonte: [4] página 210. . . 49

2.18 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{2,0}$, $\mathbf{W}_{2,1}$, $\mathbf{W}'_{2,2}$, $\mathbf{W}'_{2,3}$ (ordem natural). $N_1 \equiv N/2$. Fonte: [4] página 210. . . . . 50

2.19 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{3,0}$, ..., $\mathbf{W}_{3,7}$. $N_j \equiv N/2^j$. Fonte: [4] página 212. . . . . 51

2.20 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{3,0}$, $\mathbf{W}_{3,1}$, $\mathbf{W}_{2,1}$, $\mathbf{W}_{1,1}$ o qual é idêntico à DWT parcial de nível $J_0 = 3$. Fonte: [4] página 212. . . . . 52

2.21 Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{2,0}$, $\mathbf{W}_{3,2}$, $\mathbf{W}_{3,3}$, $\mathbf{W}_{1,1}$, uma decomposição diátia disjunta. Fonte: [4] página 213. . 53

3.1 Algoritmo de reconstrução baseado na IDWT . . . . . 62

3.2 Realização de um processo $1/f$ obtida através do método DWT de Bäckar implementado em [1] utilizando função de Haar, sorteio IID Gaussiano dos coeficientes *wavelet* e escolha de $\alpha = 0{,}9$ (H = 0,95) para controle de variância das variáveis aleatórias Gaussianas. 65

3.3 Realização de um processo $1/f$ de 4096 pontos obtida através do método DWT com mapa de variâncias utilizando função de Haar, sorteio IID Gaussiano dos coeficientes *wavelet* e escolha de $\alpha = 0{,}9$ (H = 0,95) para inicialização do cálculo do mapa de variância. . . 67

3.4 Realização de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWT com mapa de variâncias utilizando função de Haar, sorteio IID Gaussiano dos coeficientes *wavelet* e escolha de $\alpha = 0{,}9$ (H = 0,95) para inicialização do cálculo do mapa de variância. . . . . 70

3.5 Realização de um processo $1/f$ de 4096 pontos obtida através do método DWPT utilizando função de Haar, $alpha = 0{,}9$ (H = 0,95) para inicialização do cálculo do mapa de variâncias e reconstrução inicializada pelo coeficiente $\mathbf{W}_{12,4096}$. . . . . 76

3.6 Realização de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWPT utilizando função de Haar, $alpha = 0{,}9$ (H = 0,95) para inicialização do cálculo do mapa de variâncias e reconstrução inicializada pelo coeficiente $\mathbf{W}_{12,4096}$. . . . . 80

4.4 Ganho SRD constante = 10 e variável = 10-1-10 . . . . . 88

# Lista de Tabelas

2.1 Mapa de variâncias DWPT para síntese de um FGN com 3 níveis ($J$=3) e parâmetro $\alpha$=0,9, normalizado pelo coeficiente $\mathbf{W}_{3,1}$ . . . 58

4.1 Estimativas do parâmetro de Hurst pelos métodos de Whittle e do periodograma das séries geradas pelos métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT . . . . . 87

# Lista de Abreviaturas

**AR** *Auto Regressive*

**ATM** *Asynchronous Transfer Mode*

**CAC** Controle de Admissão de Conexões

**CBR** *Constant Bit Rate*

**CQS** *Classify, Queue and Schedule*

**CRM** *Customer Relationship Management*

**CSMA/CD** *Carrier Sense Multiple Access with Collision Detection*

**CWT** *Continuous Wavelet Transform*

**DEP** Densidade Espectral de Potência

**DFBM** *Discrete Fractional Brownian motion*

**DiffServ** *Differentiated Services*

**D-ITG** *Distributed Internet Traffic Generator*

**DW** *Data Warehouse*

**DWT** *Discrete Wavelet Transform*

**DWPT** *Discrete Wavelet Packet Transform*

**ERP** *Enterprise Resource Planning*

**FARIMA** *Fractional Auto-Regressive Integrated Moving Average*

**fBM** *Fractional Brownian Motion*

**FD** *Fractionally Differenced Process*

**FFT** *Fast Fourier Transform*

**FGN** *Fractional Gaussian Noise*

**FIR** *Finite Impulse Response*

**GPS** *Generalized Processor Sharing*

**GSMM** *Gaussian Spectral Synthesis Method*

**ICWT** *Inverse Continuous Wavelet Transform*

**IDWT** *Inverse Discrete Wavelet Transform*

**IDWPT** *Inverse Discrete Wavelet Packet Transform*

**IETF** *Internet Engineering Task Force*

**IIR** *Infinite Impulse Response*

**IntServ** *Integrated Services*

**IP** *Internet Protocol*

**LSPs** *label switched paths*

**LRD** *Long Range Dependence*

**MPLS** *Multiprotocol Label Switching*

**MRA** *MultiResolution Analysis*

**MTU** *Maximum Transfer Unit*

**MWM** *Multifractal Wavelet Model*

**NS** *Network Simulator*

**ODFT** *Ortogonal Discrete Fourier Tansform*

**PDF** *Probability Density Function*

**PHB** *per-hop behaviours*

**PPL** *Pure Power Law Process*

**PPP** *Point-to-Point Protocol*

**QoS** *Quality of Service*

**RED** *Random Early Detection*

**RSVP** *Resource Reservation Protocol*

**SNMP** *Simple Network Management Protocol*

**SRD** *Short Range Dependence*

**TI** Tecnologia da Informação

**VBR** *Variable Bit Rate*

**WAN** *Wide Area Network*

**WFT** *Windowed Fourier Transform*

**WWW** *World Wide Web*

# 1 Introdução

Este capítulo apresenta os objetivos, as contribuições, a motivação e a organização deste trabalho.

## 1.1 Objetivo

O objetivo deste trabalho é modelar e implementar um gerador de tráfego para a geração de séries temporais que sintetizem tráfego com as características estatísticas específicas do tráfego encontrado em redes por comutação de pacotes reais, como auto-similaridade, dependência de longa duração - LRD e dependência de curta duração - SRD. Este trabalho efetua a síntese de realizações do processo aleatório auto-similar *Fractional Gaussian Noise* (FGN) utilizando três métodos *wavelet*: a *Discrete Wavelet Transform* (DWT) de Bäckar utilizado no trabalho de [1], DWT com mapa de variâncias e a *Discrete Wavelet Packet Transform* (DWPT). Os métodos DWT com mapa de variâncias e DWPT foram desenvolvidos neste trabalho. Como o modelo FGN gera séries que apresenta LRD, foi introduzido um ganho à série numa determinada faixa de freqüência a fim de gerar séries que apresentem característica mista (LRD e SRD) apresentada por tráfegos reais. A introdução deste ganho numa faixa de freqüência específica foi possível nos métodos desenvolvidos neste trabalho já que a utilização do mapa de variâncias possibilita explorar o espectro de freqüência da série a ser sintetizada.

Para a realização deste trabalho foram utilizados pacotes de ferramentas como Matlab e S-Plus.

Este trabalho abordou a modelagem de tráfego via *wavelets*. A transformada *wavelet* é indicada para a síntese de sinais $1/f^{\alpha}$, $0 < \alpha < 1$, [6] [4] porque:

- os coeficientes *wavelet* de um processo $1/f^{\alpha}$ são aproximadamente não correlacionados no plano tempo-escala. Portanto, a modelagem e o processamento desses sinais naquele domínio podem ser realizados de maneira

eficiente. Mais precisamente, pode-se afirmar que não há correlação entre coeficientes *wavelet* de uma mesma escala e que a correlação entre escalas diferentes é fraca (neste caso, a correlação é maior entre escalas adjacentes [7]).

- é o método mais eficiente do ponto de vista computacional. A complexidade (em termos de operações de adição e multiplicação) associada à geração de uma realização de M amostras é O(M). Métodos baseados na FFT (*Fast Fourier Transform*) [8], [9] possuem complexidade O(MlogM). A técnica de geração no domínio do tempo de Hosking [10], baseada nas recursões de Levinson-Durbin, requer $O(M^2)$ operações.
- oferece a possibilidade de síntese de tráfego não-gaussiano.
- a noção de escala temporal é intrínseca à definição da transformada.

As principais contribuições dessa pesquisa são:

1. Implementação da Análise da série temporal gerada via *Discrete Wavelet Packet Transform* (DWPT) a fim de encontrar os coeficientes DWPT de todos os níveis desta série.
2. Desenvolvimento do conceito do mapa de variâncias.
3. Implementação do método DWT com mapa de variâncias para gerar uma série temporal a partir da síntese de realizações aproximadas dos processos aleatórios auto-similares FGN com introdução de SRD numa faixa de freqüência desejada.
4. Implementação do método DWPT para gerar uma série temporal a partir da síntese de realizações aproximadas dos processos aleatórios auto-similares FGN com introdução de SRD numa faixa de freqüência desejada.
5. Implementação da método DWT com mapa de variâncias para gerar uma série temporal a partir da síntese de realizações aproximadas dos processos aleatórios auto-similares FGN com uma periodicidade.
6. Implementação do método DWPT para gerar uma série temporal a partir da síntese de realizações aproximadas dos processos aleatórios auto-similares FGN com uma periodicidade.
7. Avaliação do gerador de tráfego por meio de análise estatística.

## 1.2 Motivação

Com os avanços tecnológicos na área de telecomunicações, redes e aplicações, a *Internet* e as redes de dados corporativas passaram a ser a infra-estrutura para diversos tipos de serviços com diferentes características de tráfego. São exemplos de tais serviços as aplicações mais tradicionais como navegar na *web* e as aplicações multimídia como voz sobre IP, transmissão de vídeo e mensagens instantâneas. Assim, as atuais redes são caracterizadas pela diversidade de tipos de tráfego os quais possuem diversas propriedades estatísticas como LRD e SRD e distribuição não-gaussiana.

Sendo assim, modelar tráfego e entendê-lo é importante para o projeto e simulação de redes, para prover qualidade de serviço (QoS) para diversas aplicações e para o gerenciamento e controle das redes. Vários modelos foram propostos no passado para modelar o tráfego de redes como o processo aleatório de Poisson [11], [12]. Entretanto, é necessário modelar o tráfego de redes heterogêneo que possui duas propriedades estatísticas pertinentes: presença simultânea de LRD e SRD e distribuição marginal que resultam da complexidade das redes IP e da atual diversidade de aplicações. Aplicações como voz sobre IP (*VoIp*) e *Variable Bit Rate* (VBR) *video traces* geram tráfego com a propriedade SRD, enquanto aplicações como requisição de uma página *web* geram tráfego LRD. De fato, tráfego de redes como VBR podem exibir uma presença simultânea de SRD e LRD [7]. Trabalhos feitos na área com medidas tanto em redes locais [13] quanto em redes de grande área [14] mostraram que o tráfego das redes possui propriedades fractais tais como memória longa LRD e auto-similaridade. Foi constatado que as aplicações de tempo real são caracterizadas pela SRD. Já aplicações que não são de tempo real como *video-on-demand*, comunicação de dados e algumas tarefas de gerenciamento de redes devem ser modeladas por modelos de tráfego que capturem a dependência temporal em largas escalas de tempo (LRD) [7].

Modelos como processo FGN podem capturar a LRD mas não a SRD. Modelos desenvolvidos para caracterizar tanto a SRD quanto a LRD incluem *Fractional Auto-Regressive Integrated Moving Average* (FARIMA) [15], um modelo baseado no procedimento de Hosking [16], o modelo *scene-based* [17], o processo modulado de Markov [18], o processo *fractal point* [19], e o modelo M/G/$\infty$ [20]. Dentre esses modelos, o *scene-based* e o processo modulado de Markov, provêem um modelo interpretado fisicamente que inclui tanto LRD quanto SRD. No entanto, devido à natureza estocástica do tráfego de redes, é difícil definir e segmentar exatamente o tráfego de rede em diferentes estados no domínio do tempo. O modelo

FARIMA não é computacionalmente eficiente. O modelo FARIMA-T (FARIMA truncado) desenvolido por Lima é computacionalmente eficiente [21]. O modelo M/G/$\infty$ tem um número moderado de parâmetros e é um processo pontual [7].

O desafio de modelar esse tráfego heterogêneo está em desenvolver um modelo de tráfego que possa caracterizar essas propriedades estatísticas, seja computacionalmente eficiente e fácil de analisar. [7] desenvolveu um modelo de tráfego *wavelet* que cuida dessas questões.

Outra propriedade estatística importante do tráfego heterogêneo é a distribuição não-gaussiana. Tanto tráfego de vídeo quanto de dados têm função densidade de probabilidade marginal não-gaussiana de cauda pesada.

Alguns modelos de tráfego sintetizam a LRD em tráfego Internet real de maneira satisfatória, mas diferem em outras propriedades de tráfego. De acordo com [2], são eles:

1. *Fractional Brownian Motion*

   O *Fractional Brownian Motion* (FBM) é o único processo gaussiano com incrementos estacionários e com a seguinte propriedade de escala para todo a > 0 , t $\in \mathbb{R}$ , e 0 < H < 1:

   $$B_{at} \stackrel{d}{=} a^H B_t \tag{1.1}$$

   O símbolo $\stackrel{d}{=}$ denota igualdade em distribuição.

2. *Fractional Gaussian Noise*

   O *Fractional Gaussian Noise* (FGN) corresponde à primeira diferença de um processo estocástico de tempo contínuo conhecido como movimento Browniano fracionário (FBM) $\{B_H(t) : 0 \leq t \leq \infty\}$ com parâmetro de Hurst $0 < H < 1$ [4, pág.279], ou seja,

   $$X_t = \Delta B_H(t) = B_H(t+1) - B_H(t), \quad t = 0, 1, 2, \ldots \quad . \tag{1.2}$$

3. *Wavelet-Domain Independent Gaussian Model*

   *Wavelet-Domain Independent Gaussian Model* (WIG) é o modelo de tráfego gaussiano capaz de aproximar o fBm e o FGN como processos com escala mais geral que 1.1 e 1.2. Utiliza uma árvore de multiescala para modelar o tráfego no intervalo de tempo [0,T] [22], [23].Os nós $V_{j,k}$ na árvore correspondem ao tráfego total no intervalo de tempo [$k2^{-j}$T, ($k$+1)$2^{-j}$T], $k$ = 0,1,...,$2^j$ - 1. Veja Fig. 1.1.

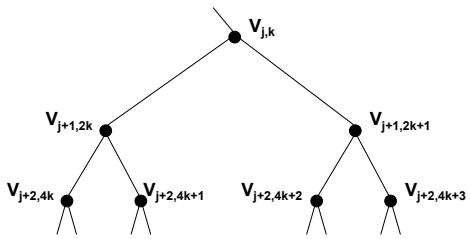



**Figura 1.1:** Árvore multiescala do tráfego. Cada nó é a soma dos seus dois nós filhos. Fonte: [2] página 1008.

Iniciando no nó $V_{j,k}$, o WIG modela nós $V_{j+1,2k}$ e $V_{j+1,2k+1}$ usando inovações aleatórias aditivas independentes $Z_{j,k}$ através:

$$V_{j+1,2k} = (V_{j,k} + Z_{j,k})/2 \quad \text{e} \quad V_{j+1,2k+1} = (V_{j,k} - Z_{j,k})/2 \tag{1.3}$$

Na prática, a árvore WIG de tamanho finito $n$ é utilizada para obter um processo discreto no tempo $V_{n,k}$. O $Z_{j,k}$ tem a mesma variância dentro de cada escala j, assim garantindo que $V_{n,k}$ é um processo estacionário de primeira ordem. A raiz $V_{0,0}$ e todo $Z_{j,k}$ são gaussianos o que garante que todos os nós da árvore sejam gaussianos.

Especificar um modelo de tráfego significa escolher seus parâmetros tanto para atingir estatísticas-chave para o tráfego observado quanto para garantir que o modelo tenha certas propriedades estatísticas pré-definidas. Especificar o WIG envolve a escolha de seus parâmetros para obter a progressão da variância requerida de var($V_{j,k}$). O WIG pode prover uma aproximação gaussiana para qualquer processo estacionário de tempo discreto X; isto é, o WIG pode ser especificado para obter

$$var(V_{n-j,k}) = var(K^{\{X\}}[2^j]) \tag{1.4}$$

4. *Multifractal Wavelet Model*

   O *Multifractal Wavelet Model* (MWM) é um modelo não-gaussiano baseado na árvore de multi escala que, assim como o WIG, permite um comportamento de escala mais genérico da variância dos nós da árvore que o FGN [24]. Diferentemente do WIG, assegura positividade em todas escalas de tempo, uma propriedade intrínseca do tráfego de dados real que é freqüentemente mal aproximado pelos modelos gaussianos. Sendo $V_{0,0} \geq 0$, o MWM usa

inovacões multiplicativas independentes $U_{j,k} \in [0,1]$ para modelar os dois filhos do nó $V_{j,k}$ por meio de:

$$V_{j+1,2k} := V_{j,k}U_{j,k} \quad \text{e} \quad V_{j+1,2k+1} := V_{j,k}(1 - U_{j,k}). \tag{1.5}$$

Como o produto das variáveis aleatórias independentes converge para uma distribuição lognormal pelo teorema do limite central, os nós $V_{j,k}$ se tornam aproximadamente lognormal ao aumentar j.

Segundo [24], $U_{j,k}$ e $V_{0,0}$ são modelados como variáveis aleatórias beta simétricas. O nó da árvore $V_{j,k}$ é o produto de várias variáveis aleatórias beta independentes.

Especificar o MWM envolve escolher seus parâmetros para obter a progressão da variância requerida de var($V_{j,k}$). O MWM pode modelar qualquer processo estacionário de tempo discreto X com autocovariância positiva no sentido de 1.4.

Enquanto os modelos WIG e MWM são estacionários de primeira ordem, não são estacionários de segunda ordem. Isto é aparente na Fig. 1.1. Observe que $V_{j+2,4k}$ e $V_{j+2,4k+1}$ têm o mesmo nó pai enquanto $V_{j+2,4k+1}$ e $V_{j+2,4k+2}$ não têm. Então, a correlação de $V_{j+2,4k+1}$ com seus dois vizinhos, $V_{j+2,4k}$ e $V_{j+2,4k+2}$ é diferente. Ambos os modelos, no entanto, têm uma estrutura de correlação com média no tempo próxima da do processo estacionário X que modelam.

## 1.3 Organização do Trabalho

O Capítulo 2 faz uma revisão teórica dos tópicos: geração de tráfego, caracterização e síntese de tráfego e transformadas *wavelet*.

O Capítulo 3 descreve os métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT para geração de séries temporais a partir da síntese de realizações aproximadas dos processos aleatórios auto-similares FGN.

O Capítulo 4 apresenta os resultados da geração de séries pelos métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT, fazendo uma análise comparativa dos resultados a fim de validar os métodos DWT com mapa de variâncias e DWPT.

O Capítulo 5 apresenta as conclusões e sugestões para trabalhos futuros.

# 2 Revisão Teórica e Trabalhos Correlacionados

## 2.1 Geração de Tráfego

Diversos trabalhos relacionados à geração de tráfego para ambientes de simulação foram desenvolvidos, porém os métodos apresentados na literatura tem limitações e será feita uma breve descrição dos mesmos:

- Via traces reais: À primeira vista foi considerado reproduzir *traces* de tráfego capturados em redes reais. Essa solução tem um efeito indesejado quando o tráfego agregado atinge níveis de congestionamento que acionam mecanismos de controle de congestionamento e algoritmos de gerência de filas. Assim, as fontes de tráfego que reconhecem a ação desses mecanismos (ex. fonte TCP mediante perda de pacotes) tendem a reduzir sua taxa de transmissão buscando aliviar a condição de congestionamento [25]. Usuários reais poderiam desistir de suas conexões e aplicações adaptativas poderiam mudar o seu comportamento, tornando o *trace* de tráfego específico para a situação em que foi coletado.

- Via simulação: Em uma simulação com características diferentes das do ambiente real, a interação entre as fontes de tráfego e os mecanismos de controle de recursos de redes deve ser distinta. Espera-se que as fontes de tráfego apresentem um comportamento de injeção de pacotes diferente do capturado pelo *trace*, dependendo das condições particulares de cada cenário de teste e simulação.

[25] apresenta algumas estratégias para contornar as dificuldades em simular a *Internet*. Em uma das estratégias, a busca por invariantes, que significa buscar comportamentos empiricamente demonstráveis para uma grande variedade de ambientes, [25] destaca o fenômeno da *auto-similaridade*, freqüentemente presente nos padrões de tráfego de redes de dados.

Uma outra estratégia mencionada em [25] enfatiza o cuidado que deve ser tomado ao explorar o espaço de parâmetros de uma simulação ou experimento. A recomendação é sempre analisar resultados para um grande intervalo de valores de entrada. Essa estratégia se acopla à grande heterogeneidade e às constantes mudanças da *Internet*.

A principal dificuldade em se encontrar um gerador de tráfego realista, pronto para ser utilizado em *testbeds* e simulações, está no fato da modelagem e caracterização de tráfego em redes de dados ainda ser um tema em estudo.

A auto-similaridade é uma característica estatística que deseja-se reproduzir, uma vez que é encontrada em *traces* de tráfego reais e bem aceita pelos pesquisadores na área de redes de dados. Em [26], encontra-se uma boa explicação de auto-similaridade, são citados alguns métodos que permitem a síntese de séries auto-similares e são indicadas algumas maneiras existentes para a estimação do parâmetro de *Hurst* em uma série auto-similar. Em seu trabalho, [26] estabeleceu que as análises efetuadas no estudo de uma disciplina de serviço *Generalized Processor Sharing* (GPS) seriam feitas mediante modelos de tráfego auto-similares, reforçando a importância desse fenômeno.

Em [27] foi introduzido o método DHM (Davies and Harte *Method*) de geração de realizações de processos estacionários Gaussianos de média nula. O método é baseado na *Fast Fourier Transform* (FFT). [9] também publicou um método baseado na FFT denominado *Gaussian Synthesis Method* (GSSM). Em [8] foi proposto um método de geração de realizações aproximadas do FGN baseado na FFT. Paxson demonstrou que o seu método gera séries estatísticamente indistinguíveis de realizações FGN e que essas séries podem ser utilizadas na geração de tráfego de redes auto-similares.

A transformada *wavelet* foi utilizada para obter um método rápido e preciso de sintetizar séries que aproximam realizações de um processo FBM em [28] .

O modelo MWM proposto em [24] permite sintetizar séries não-Gaussianas com propriedades multifractais que possuem aplicação na geração de tráfego de redes. Em [29] é proposto um método para geração de tráfego em redes que consiste de dois estágios. O primeiro estágio gera uma série multifractal com distribuição de probabilidade marginal de cauda longa por meio da transformada *wavelet*. No segundo estágio é utilizada uma técnica de decomposição em componentes de freqüência por filtros Volterra para ajustar as características de dependência de curta duração.

A tese [30] tem o objetivo de sintetizar tráfego em "tempo-real" semelhante ao encontrado em redes reais. É mencionado um possível uso para o tráfego sintetizado com características reais e sua vantagem: utilização em ambientes de teste para comparação de desempenho de equipamentos de rede obtendo resultados mais realistas. Nesse estudo, afirma-se que a auto-similaridade, a ocorrência de rajadas e a "dependência de longa duração" caracterizam o tráfego encontrado nas redes de dados.

Efetuando-se buscas por ferramentas livres para geração de tráfego em *testbeds*, encontrou-se uma quantidade relevante de ferramentas com a função de geração de pacotes. Algumas delas são mais voltadas para a geração de tráfego HTTP, indicado para realizar testes de carga especificamente em servidores *Web*. Embora modelos auto-similares sejam aceitos para emulação de tráfego em redes de dados, nenhuma das ferramentas encontradas incorpora esse tipo de modelo.

Essa constatação reforça a importância de modelar e implementar um gerador de tráfego baseado em modelos estatísticos adequados para redes de dados. Ao identificar as principais dificuldades para realizar a implementação de um gerador de tráfego, percebe-se que o primeiro desafio é a geração de séries temporais auto-similares que possam representar a contagem de *bytes* por unidade de tempo. Em seqüência, deve-se investigar alternativas para a conversão de uma série temporal (*bytes/sec*) em uma série com indicações de tamanho e tempo entre pacotes (série de pacotes) que permita o envio de pacotes na rede. Em [1], a geração de séries temporais que se assemelham às realizações de um processo FGN e a geração de séries como especificado pelo modelo MWM são implementadas da mesma maneira descrita por [30]. Posteriormente, as séries geradas são convertidas em séries de pacotes por dois métodos alternativos. Em um deles o tamanho dos pacotes enviados é mantido constante, variando-se o tempo entre pacotes característico de cada *bin* de transmissão. No segundo, os instantes de envio de blocos de pacotes são determinados e eqüiespaçados, variando-se o tamanho dos blocos e dos pacotes enviados.

A Fig. 2.1 apresenta o diagrama de blocos proposto para o gerador de tráfego implementado em [1] sendo a interface do gerador feita com o *software* simulador de redes *Network Simulator* (NS-2).

Na solução proposta, o usuário pode controlar o nível médio do tráfego gerado e a amplitude dos surtos através dos parâmetros de entrada da função de *geração da série temporal*. A série gerada é encaminhada ao módulo *gerador de pacotes* que monta os pacotes no tamanho especificado pela série sintetizada e com as

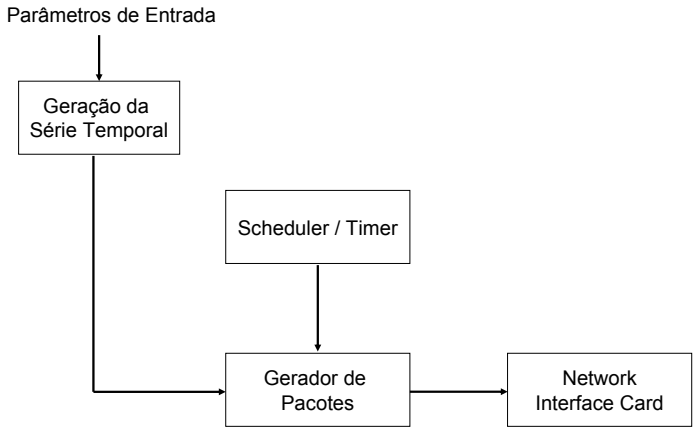



**Figura 2.1:** Diagrama de Blocos do Gerador de Tráfego.

informações de controle necessárias.

O método proposto por [1] faz a síntese de realizações aproximadas dos processos aleatórios auto-similares FGN e MWM via transformada *wavelet*. As séries temporais geradas possuem propriedades fractais como *auto-similaridade* e LRD que são propriedades identificadas no tráfego multiserviço. Porém, também foi constatado que o tráfego pode apresentar (SRD) em algumas escalas temporais [1]. Devido a essa constatação, o método proposto também foi capaz de sintetizar séries Gaussianas (FGN) e não-Gaussianas (MWM) com espectros mais genéricos do que 1/f, ou seja, séries que também apresentam dependência de curta duração. Para isto a geração foi feita em dois estágios como em [29]. O primeiro gera uma realização aproximada do FGN ou do MWM via Transformada *Wavelet* Discreta (DWT). O segundo estágio introduz SRD através de uma filtragem *Infinite Impulse Response* IIR passa-bandas da saída do primeiro estágio. Assim, o gerador implementado por [1] é mais flexível que o de [30], uma vez que a sua saída pode ser $1/f$, $1/f$ e SRD ou somente SRD. Foi efetuada uma caracterização detalhada das séries resultantes, utilizando momentos estatísticos de $2^{\underline{a}}$, $3^{\underline{a}}$ e $4^{\underline{a}}$ ordens, além de testes estatísticos específicos para séries auto-similares.

No módulo *gerador de pacotes* foram apresentadas duas alternativas para converter as séries temporais geradas em séries de pacotes. As séries de pacotes foram novamente analisadas a fim de identificar se o método de conversão introduziu distorção nas características auto-similares das séries sintetizadas. Foi mostrado que as séries de pacotes auto-similares podem ser utilizadas em *softwares* simuladores

de redes, ou para se injetar pacotes em redes de teste.

[1] utilizou recursos do simulador NS-2 para introduzir as séries de pacotes sintetizadas em cenários de simulação adequados. Os resultados (medidas de atraso médio, perda de pacotes para o tráfego de interesse e tamanho da fila) dos cenários com tráfego interferente correspondente às séries de pacotes baseadas em modelos FGN e MWM foram comparados com resultados obtidos em cenários cujo tráfego interferente foi gerado com modelo Poisson.

Em relação às ferramentas de geração de tráfego disponibilizadas para a comunidade de pesquisa, pode-se destacar a plataforma D-ITG proposta em [31] e [32]. Essas são duas publicações abordando uma plataforma para geração de tráfego que, segundo os autores, adere com precisão aos padrões estabelecidos por processos estocásticos que representem o tempo entre envios de pacotes e o tamanho de pacotes. Essa solução permite que um conjunto significativo de distribuições de probabilidade para a geração dos processos estocásticos de tempo entre envios de pacotes e tamanho de pacotes possa ser escolhida: constante, uniforme, exponencial, normal, Pareto, Poisson etc. Adicionalmente, protocolos bastante utilizados nas redes IP estão mapeados em alguns modelos propostos, permitindo ao usuário da ferramenta escolher um protocolo em vez das distribuições de probabilidades disponíveis. O D-ITG não possui modelos específicos para a geração de processos auto-similares.

O trabalho apresentado em [31] e [32] inclui uma explicação sobre cada um dos componentes da plataforma de geração de tráfego: *ITGSend* (módulo que envia os pacotes), *ITGRecv* (módulo que recebe os pacotes), *ITGLog* (recebe as informações de vários *senders e receivers* e armazena essas informações em arquivos de log) e *ITGManager* (possibilita controlar os módulos geradores de pacotes). Em [32], é feita uma comparação entre o D-ITG e alguns geradores disponibilizados na *Internet*, baseada no *throughput* de cada gerador. Além do D-ITG, os geradores envolvidos são: Iperf, Mgen, Rudecrude, UDPgenerator e Mtools.

## 2.2 Caracterização e Síntese de tráfego

Esta seção introduz brevemente a noção de fractal e formula os conceitos de LRD, SRD e auto-similaridade. O trabalho de [21] será utilizado como base para as definições desses conceitos.

Nesta seção também será descrito o FGN que foi o modelo utilizado para

gerar as séries temporais e validar os objetivos deste trabalho.

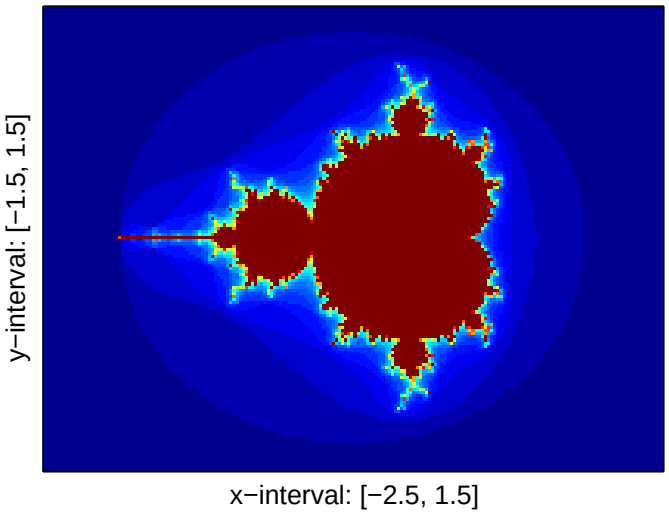
### 2.2.1 Fractais

As formas da geometria clássica - triângulos, círculos, esferas, etc. - perdem suas estruturas quando ampliadas (isto é, quando se dá um *zoom* numa determinada região da figura). Por exemplo, uma pessoa na superfície da Terra tem a impressão de que a mesma é plana (já um astronauta em órbita vê uma Terra redonda). Suponha que um indivíduo não tenha sido informado de que foi posicionado num ponto de um círculo com raio de curvatura da ordem de centenas de quilômetros. Esse observador percebe o círculo como sendo uma reta, apesar disto não ser verdade.

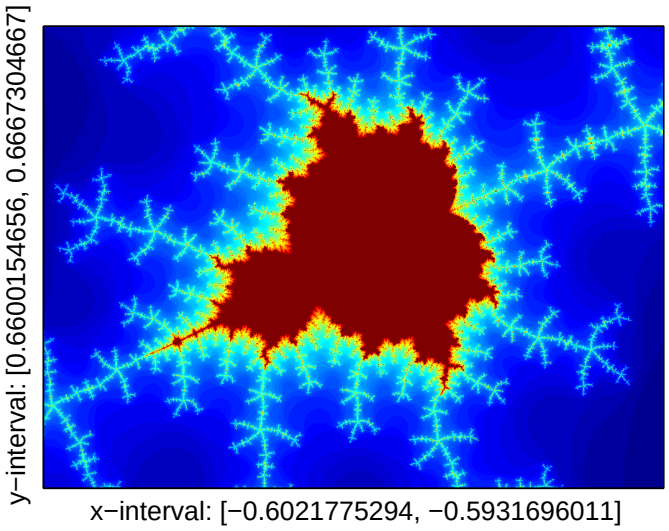
Benoit B. Mandelbrot propôs em 1975 [33] o termo **fractal** (do latim *fractus*, que tem o significado de fraturado, quebrado) para designar objetos matemáticos que possuem uma estrutura rica em detalhes ao longo de muitas escalas de observação. O conjunto de Mandelbrot ou o "boneco de pão-de-mel" da Fig. 2.2 é um fractal matemático com estrutura detalhada (ou seja, é altamente irregular) ao longo de uma série infinita de escalas. Note-se que a Fig. 2.2 (b), obtida por meio de um *zoom* de uma determinada região da Fig. 2.2 (a), mostra que existe um sub-boneco de pão-de-mel embebido no boneco original. Esta característica ilustra uma outra propriedade dos fractais conhecida como **auto-similaridade**: um objeto auto-similar contém réplicas menores de si mesmo em todas as escalas. O conjunto de Mandelbrot é um exemplo de fractal determinístico, pois é exatamente auto-similar.

Um fractal matemático bastante conhecido é o conjunto de Cantor, descrito em 1883 pelo matemático alemão Georg Cantor [35] apud [36], o qual é constituído por um número infinito não-enumerável de pontos no segmento $[0, 1]$ da reta real. O conjunto de Cantor pode ser obtido de acordo com os seguintes passos: 1) comece com o intervalo $[0, 1]$, 2) remova o intervalo aberto $(1/3, 2/3)$, ou seja, remova o terço do meio de $[0, 1]$, mas não os números $1/3$ e $2/3$ e 3) a cada nova iteração, remova o terço do meio (intervalo aberto) dos segmentos resultantes do passo anterior. A Fig. 2.3 mostra as cinco primeiras iterações do procedimento acima descrito.

A Fig. 2.4 ilustra que o tráfego Ethernet coletado numa rede local da Universidade de Drexel é auto-similar e altamente impulsivo em quatro escalas temporais de agregação (10 ms, 100 ms, 1 s e 10 s). A série na escala de 100 ms foi obtida por meio de uma agregação da série na escala de 10 ms, ou seja, uma ordenada




**(a)**




**(b)**

**Figura 2.2:** A Fig. (a) é o conjunto de Mandelbrot, também conhecido como "boneco de pão-de-mel". A Fig (b) mostra um sub-boneco de pão-de-mel que está embebido na Fig. (a). As figuras foram obtidas via `MATLAB` utilizando-se o código de A. Klimke [34], da Universidade de Stuttgart.

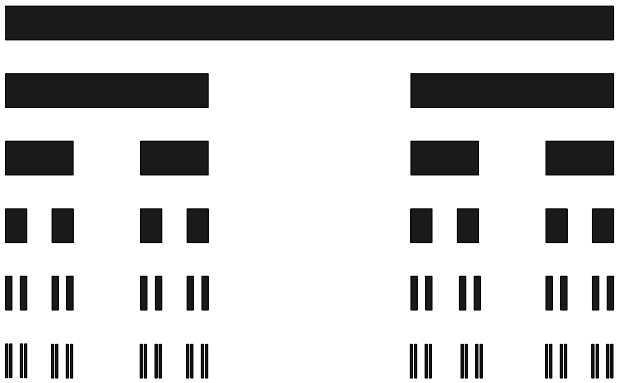


**Figura 2.3:** Ilustração das cinco primeiras iterações para obtenção do conjunto de Cantor. Repare-se que o formato de uma determinada parte do objeto, quando magnificada, se assemelha ao formato do todo (auto-similaridade).

da série de 100 ms corresponde à soma dos *bytes* em 10 *bins* consecutivos da série na escala de 10 ms. O mesmo procedimento foi aplicado para obtenção das séries nas escalas de 1 s e 10 s, que são agregações das séries nas escalas de 100 ms e 1 s, respectivamente. É surpreendente observar que agregações sucessivas não suavizam o tráfego (a suavização aconteceria se o tráfego fosse bem modelado pelo processo de Poisson). Note-se que a alternância de períodos de surtos e de suavidade é preservada nas quatro escalas de tempo em questão. Isto permite afirmar que não há uma escala de tempo característica para a ocorrência de surtos. Esta propriedade de invariância com relação à mudança de escala temporal é conhecida na literatura como "*scaling invariance*" ou "*scaling behavior*" (que devem ser entendidos como sinônimos de auto-similaridade) [37], [38].

### 2.2.2 O Expoente de Hurst

Por razões históricas, o grau de persistência (LRD) de uma série temporal é caracterizado pelo parâmetro $H$, $0 < H < 1$, de Hurst[1] [39]. Uma série temporal é LRD (auto-similar) quando $1/2 < H < 1$ e tem memória curta ou dependência de curta duração (SRD) para $0 < H \leq 1/2$ [40]. Quanto mais próximo $H$ estiver de 1, maior será o grau de persistência da série. Informalmente, diz-se que uma série é monofractal quando $H$ é invariante no tempo e multifractal quando $H$ varia no tempo de forma determinística ou aleatória. Os artigos [37], [41], [42], [43] mostraram que o tráfego *Wide Area Network* (WAN) pode ser multifractal (com distribuição marginal não-Gaussiana) em escalas refinadas de tempo (até centenas de milissegundos), em contraste com o comportamento monofractal

[1]Hurst, um hidrologista, observou que a série histórica do nível mínimo anual do rio Nilo possui memória longa. Esta série tem sido registrada há centenas de anos.

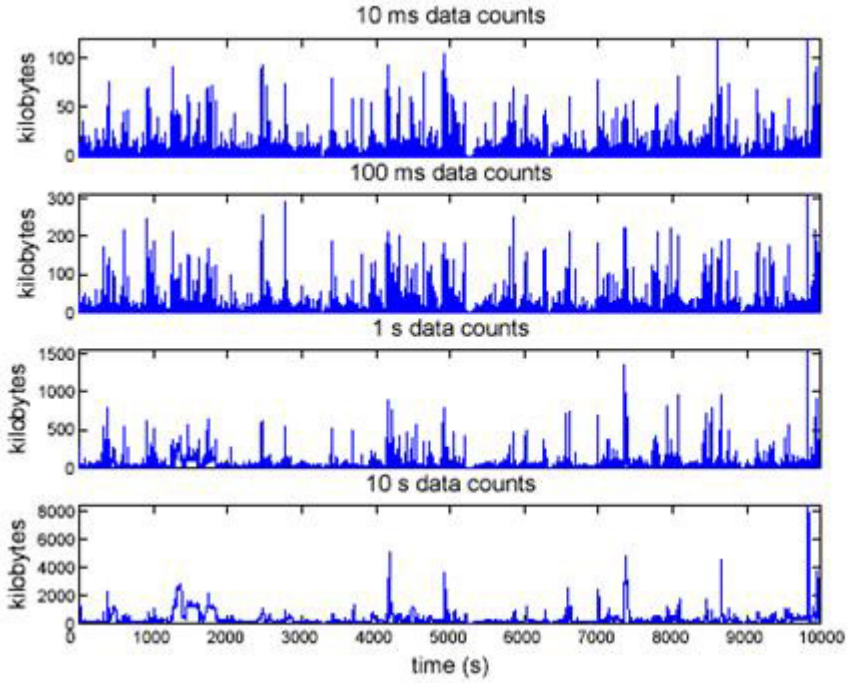



**Figura 2.4:** Visualização do tráfego *Fast* Ethernet (100 Mbps) coletado num servidor WWW/Email/FTP da Universidade de Drexel [3] em quatro diferentes níveis de agregação: 10 ms, 100 ms, 1 s e 10 s (de cima para baixo). Este *trace* tem um tamanho total de 3 h.

(ou auto-similar) que foi observado para o tráfego LAN [13].

A memória longa é caracterizada no domínio da freqüência pela existência de uma singularidade do tipo $1/f^{\alpha}$, $0 < \alpha < 1$ ($\alpha = 2H - 1$), para $f \to 0$. A Fig. 2.5 mostra que a Densidade Espectral de Potência (DEP) de um modelo LRD da classe *Fractionally Differenced Process* (FD)[2] [44], [45] com parâmetro $d = 0,4$ ($d = H - 1/2$) apresenta o comportamento $1/f^{\alpha}$ na origem do espectro, ao passo que um modelo *Auto Regressive* (AR) de ordem 4 (AR(4)) não. A literatura costuma referir-se aos processos LRD como "ruídos $1/f$".

## 2.2.3 LRD e Auto-Similaridade

### 2.2.3.1 Dependência de Longa Duração

Considere um processo aleatório estacionário $X_t$, $t \in \mathbb{Z}$, com média $\mu_X$ e variância $\sigma_X^2$. Sejam $X_1, X_2, \ldots, X_N$ observações de uma realização $X_t$. Se as variáveis aleatórias $X_1, X_2, \ldots, X_N$ forem independentes ou não-correlacionadas, então, a

[2] O modelo FD($d$) corresponde ao processo FARIMA($p, d, q$) com $p = q = 0$, em que $p$ denota o parâmetro autoregressivo e $q$ o de média móvel.

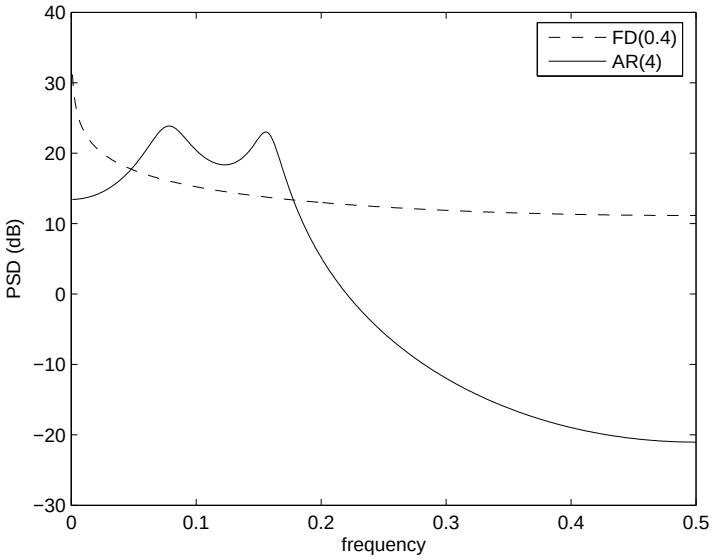



**Figura 2.5:** DEPs para modelos AR(4) e FD(0,4) de mesma potência.

variância de $\bar{X}$ (média amostral) será dada por

$$\sigma_{\bar{X}}^2 = \frac{\sigma_X^2}{N}. \tag{2.1}$$

Se a amostra for suficientemente grande, a distribuição amostral do estimador $\bar{X}$ será normal. A expressão do intervalo de confiança para $\mu_X$, ao nível de confiança $(1-\beta)$, é dada por[3]

$$\bar{X} - z_{\beta/2}\frac{\sigma_X}{\sqrt{N}} \leq \mu_X \leq \bar{X} + z_{\beta/2}\frac{\sigma_X}{\sqrt{N}}, \tag{2.2}$$

em que $z_{\beta/2}$ denota o quantil[4] $q_{(1-\beta/2)}$ da distribuição normal padrão [40].

**Definição 2.1** (Dependência de Longa Duração)**.** $X_t$ é um processo com dependência de longa duração ou memória longa se existirem constantes $\alpha$ e $C_S$, satisfazendo $0 < \alpha < 1$ e $C_S > 0$, tais que [40], [4], [47]

$$\lim_{f \to 0} \frac{S_X(f)}{C_S|f|^{-\alpha}} = 1\,, \tag{2.3}$$

em que $S_X(f)$ denota a DEP de $X_t$ e $f$ representa a freqüência normalizada $(-1/2 \leq f \leq 1/2)$, em ciclos/amostra.

Portanto, a DEP de processos LRD tende a infinito na freqüência zero (singularidade do tipo $1/f^\alpha$ na origem do espectro). Note-se que a definição 2.1 é

[3]Dada uma probabilidade $1-\beta$, encontra-se um valor $z_{\beta/2}$ tal que $P\{-z_{\beta/2} < Z < z_{\beta/2}\} = 1-\beta$ ($z_{\beta/2} = 1,96$ para $1-\beta = 95\%$).

[4]O quantil $q_\alpha$ de uma função de distribuição $F_x$ é o valor para o qual se tem $F_{q_\alpha} = \alpha$ [46, pág.181]. A mediana, por exemplo, corresponde ao quantil $q_{0,5}$.

assintótica, pois o formato da DEP em freqüências afastadas da origem não é especificado. É mais comum caracterizar-se a memória longa pelo parâmetro $H$ de Hurst:

$$H = \frac{\alpha + 1}{2}, \qquad 1/2 < H < 1\,. \tag{2.4}$$

Quanto maior o valor de $H$, maior é o grau de memória longa do processo.

Uma definição alternativa pode ser dada no domínio do tempo, em termos da autocorrelação $R_X(\tau)$. $X_t$ é um processo do tipo $1/f^\alpha$ se a sua autocorrelação $R_X(\tau)$, para valores suficientemente grandes do *lag* $\tau$, decresce segundo uma função potência (isto é, o decaimento para zero é extremamente lento e do tipo hiperbólico):

$$\lim_{\tau\to\infty} \frac{R_X(\tau)}{C_R \tau^{-(1-\alpha)}} = 1\,, \tag{2.5}$$

em que $C_R > 0$.

A singularidade na origem do espectro implica a validade da relação

$$\sum_{\tau=-\infty}^{\infty} R_X(\tau) = \infty\,, \tag{2.6}$$

para $1/2 < H < 1$, ou seja, as autocorrelações decaem para zero tão lentamente que não são somáveis. A Eq. (2.6) afirma que a dependência estatística entre eventos distantes diminui muito lentamente com o aumento do *lag* $\tau$. Deste modo, a razão entre autocorrelações para valores suficientemente grandes do *lag* não se altera de modo apreciável.

Se $X_t$ for LRD, a variância de $\bar{X}$ decresce com o tamanho $N$ da amostra mais lentamente do que no caso tradicional (variáveis independentes ou não-correlacionadas) e da seguinte maneira [40, pág.6]

$$\sigma^2_{\bar{X}} \approx \sigma^2_X c(\rho_X) N^{\alpha-1}, \tag{2.7}$$

em que $c(\rho_X)$ é definido por

$$\lim_{N\to\infty} N^{-(1+\alpha)} \sum_{i\neq j} \rho_X(i,j). \tag{2.8}$$

Neste caso, a distribuição de $\bar{X}$ é assintoticamente Gaussiana, com $E[\bar{X}] = \mu_X$.

#### 2.2.3.2 Auto-Similaridade

**Definição 2.2** (Processo $\boldsymbol{H}$-ss)**.** Um processo estocástico $\{Y_t\}_{t\in\mathbb{R}}$ é auto-similar com parâmetro $0 < H < 1$, ou seja, é $H$-ss (*self-similar with parameter* $H$) se,

para qualquer $a > 0$,

$$\{Y(t)\} \stackrel{d}{=} \{a^{-H}Y(at)\}, \tag{2.9}$$

em que $\stackrel{d}{=}$ denota igualdade entre as distribuições finito-dimensionais [47].

Um processo $H$-ss é LRD se $1/2 < H < 1$. O processo Movimento Browniano (de tempo contínuo) [48], também conhecido como processo de Wiener, satisfaz a definição 2.2, sendo auto-similar com $H = 1/2$ (mas não é LRD). Se o processo $X_t = \Delta Y_t$, denominado processo de incrementos de $Y_t$ ou primeira diferença de $Y_t$, for estacionário, então $Y_t$ é denominado $H$-sssi (*H self-similar with stationary increments* - auto-similar com incrementos estacionários). Neste caso, o processo $H$-sssi $Y_t$ é um processo integrado de ordem 1, $Y_t \sim I(1)$.

Se os momentos de $Y(t)$ de ordem menor ou igual a $q$ existirem, pode-se concluir a partir de (2.9) que [49]

$$E|Y_t|^q = E|Y_1|^q|t|^{qH}. \tag{2.10}$$

Portanto, o processo $Y_t \sim I(1)$ não pode ser estacionário.

Assumindo-se $E[Y_t] = 0$ com o intuito de simplificar a notação[5], demonstra-se que a autocovariância de $Y_t$ é dada por [40]

$$C_Y(t,s) = E[Y_tY_s] = \frac{\sigma_X^2}{2}[t^{2H} + s^{2H} - (t-s)^{2H}]. \tag{2.11}$$

em que $\sigma_X^2 = E[(Y_t - Y_{t-1})^2] = E[Y^2(1)]$ é a variância do processo de incrementos $X_t$.

Considere a versão amostrada $Y_t$, $t \in \mathbb{Z}$, de um processo $Y_t$ $H$-sssi, em que o período de amostragem é unitário. Existem vários processos não-Gaussianos $Y_t$ $H$-sssi. Entretanto, para cada valor de $H \in (0,1)$ há exatamente um único processo Gaussiano $Y_t$ $H$-sssi, denominado movimento Browniano fracionário de tempo discreto (*Discrete Fractional Brownian motion* (DFBM))[40], [4]. O ruído Gaussiano fracionário (FGN), proposto por Mandelbrot e van Ness em 1968 [48], corresponde ao processo de incrementos do DFBM. O FGN é um modelo bastante utilizado em simulações de tráfego LRD [50], [8], [30] [6].

**Definição 2.3** (Auto-Similaridade Exata de Segunda Ordem)**.** Seja o processo estacionário de tempo discreto $X_t = Y_t - Y_{t-1}$. $X_t$ é um processo exatamente auto-similar de segunda ordem com parâmetro de Hurst $H$ $(1/2 < H < 1)$ se a

[5] O que implica $E[X_t] = E[Y_t - Y_{t-1}] = 0$

[6] Os modelos DFBM e FGN são não-paramétricos e não são utilizados para se fazer previsão de valores futuros de tráfego. O processo FGN é definido no domínio da freqüência a partir de uma prescrição específica de DEP [8].

sua autocovariância existe e for dada por [40]

$$C_X(\tau) = \frac{\sigma_X^2}{2}[|\tau+1|^{2H} - 2|\tau|^{2H} + |\tau-1|^{2H}], \quad \tau = \ldots, -1, 0, 1, \ldots \quad . \tag{2.12}$$

Pode-se demonstrar que a autocovariância dada por (2.12) satisfaz [40]

$$\lim_{\tau\to\infty} \frac{C_X(\tau)}{\sigma_X^2 \tau^{2H-2} H(2H-1)} = 1, \tag{2.13}$$

ou seja, $C_X(\tau)$ tem um decaimento hiperbólico. Portanto, auto-similaridade de segunda ordem implica LRD quando $1/2 < H < 1$.

Considere o processo agregado $X_t^{(M)}$ de um processo exatamente auto-similar de segunda ordem $X_t$, ao nível de agregação $M$. Demonstra-se que

$$C_X^{(M)}(\tau) = C_X(\tau), \qquad M = 2, 3, \ldots \tag{2.14}$$

A Eq. (2.14) diz que as estatísticas de segunda ordem do processo original não mudam com a mudança de escala, o que justifica o termo "exatamente auto-similar de segunda ordem".

**Definição 2.4** (Auto-Similaridade Assintótica de Segunda Ordem)**.** Um processo $X_t$ é assintoticamente auto-similar de segunda ordem com parâmetro de Hurst $H$ $(1/2 < H < 1)$ se a sua autocovariância e a autocovariância do seu processo agregado estão relacionadas por [49]

$$\lim_{M\to\infty} C_X^{(M)}(\tau) = C_X(\tau). \tag{2.15}$$

### 2.2.4 Processo FGN

Em termos históricos, o processo FGN, proposto por Mandelbrot e Van Ness em 1968 [48] para modelagem de séries hidrológicas LRD, é o primeiro modelo importante de memória longa que aparece na literatura. Por definição, se $\{X_t\}_{t\in\mathbb{Z}}$ é um FGN, então $X_t$ é um processo estacionário com autocovariância dada por (2.12) (ou seja, $C_X(\tau) = \frac{\sigma_X^2}{2}[|\tau+1|^{2H} - 2|\tau|^{2H} + |\tau-1|^{2H}]$, $\tau = \ldots, -1, 0, 1, \ldots$). O FGN corresponde à primeira diferença de um processo estocástico de tempo contínuo conhecido como movimento Browniano fracionário (FBM) $\{B_H(t) : 0 \leq t \leq \infty\}$ com parâmetro de Hurst $0 < H < 1$ [4, pág.279], ou seja,

$$X_t = \Delta B_H(t) = B_H(t+1) - B_H(t), \quad t = 0, 1, 2, \ldots \quad . \tag{2.16}$$

Beran fornece uma definição formal do proceso FBM em [40]. O FBM tem

uma denominação especial quando $H = 1/2$: movimento Browniano (*Brownian motion*) e é designado por $B_{1/2}(t)$. Neste caso, $X_1, X_2, ...$ são variáveis aleatórias Gaussianas independentes.

Pode-se criar um FBM de tempo discreto (DFBM), denotado por $B_t$, por meio da soma cumulativa de amostras do FGN $\{X_t\}$:

$$B_t \equiv B_H(t) = \sum_{u=0}^{t-1} X_u, \quad t = 1, 2, \ldots \quad . \tag{2.17}$$

A DEP do DFBM é dada pela fórmula [4, pág. 280]

$$S_{B_t}(f) = \sigma_X^2 C_H \sum_{j=-\infty}^{\infty} \frac{1}{|f+j|^{2H+1}}, \quad -\frac{1}{2} \leq f \leq \frac{1}{2}, \tag{2.18}$$

em que $\sigma_X^2$ é a potência de um FGN de média nula, $C_H = \frac{\Gamma(2H+1)\sin(\pi H)}{2\pi^{2H+1}}$ e $0 < H < 1$. De acordo com (2.18), a DEP do DFBM possui uma singularidade do tipo $|f|^{-\alpha}$, $0 < \alpha < 1$, na origem, pois

$$S_{B_t} \propto |f|^{1-2H}, \quad f \to 0\,. \tag{2.19}$$

O FGN e o DFBM estão relacionados pela função de transferência

$$H(z) = \frac{X(z)}{B(z)} = 1 - z^{-1}, \tag{2.20}$$

em que $X(z)$ e $B(z)$ denotam as transformadas $z$ de $X_t$ e $B_t$, respectivamente. A resposta em freqüência associada a (2.20) é

$$H(f) = H(z)|_{z=e^{j2\pi f}} = 1 - e^{-j2\pi f}\,. \tag{2.21}$$

Como a relação saída/entrada em termos das DEPs é igual a [51, pág. 351]

$$S_X(f) = |H(f)|^2 S_{B_t}(f)\,, \tag{2.22}$$

em que $|H(f)|^2$ é dado por,

$$|H(f)|^2 = G(f) = 4\sin^2(\pi f)\,, \tag{2.23}$$

então a DEP do FGN é igual a

$$S_X(f) = 4\sin^2(\pi f) S_{B_t}(f)\,. \tag{2.24}$$

Assim, (2.18) e (2.24) mostram que a DEP do FGN é caracterizada por somente dois parâmetros: $\sigma_X^2$ e $H$ (responsável pela forma do espectro). Além

disso, é importante se ter em mente que o FGN é completamente especificado pela sua média e pela sua DEP, pois é Gaussiano.

Em [8], é mostrado que (2.24) pode ser reescrita na forma:

$$S_X(f) = A(f,H)[|2\pi f|^{-2H-1} + B(f,H)]\,, \tag{2.25}$$

em que $A(f,H) = 2\sin(\pi H)\Gamma(2H+1)(1-\cos(2\pi f))$ e $B(f,H) = \sum_{j=1}^{\infty}[(2\pi j + 2\pi f)^{-2H-1} + (2\pi j - 2\pi f)^{-2H-1}]$. Para pequenos valores de $f$ tem-se que $S_X(f) \propto |f|^{1-2H}$ .

## 2.3 *Wavelet*

Uma *wavelet* é uma função de duração finita com média temporal nula. Existem decomposições *wavelet* em tempo contínuo e em tempo discreto. A ferramenta básica para estudo de séries temporais via *wavelets* é a versão discreta, a *Discrete Wavelet Transform* (DWT) [1].

Bases ortonormais construídas a partir de funções *wavelet* são utilizadas para descrever sinais no plano tempo-freqüência. A transformada *wavelet* é uma solução natural para a análise de séries temporais auto-similares porque a sua aplicação envolve "dilações"[7] (expansões) de bandas espectrais. Portanto, a DWT possui resolução temporal variável. Além disso, os coeficientes da DWT de um sinal $1/f$ são praticamente não-correlacionados (intra e inter-escalas) [4]. Por essa razão, as *wavelets* têm sido amplamente empregadas nas análise e síntese (simulação) de sinais fractais [52].

### 2.3.1 Transformada *Wavelet* Contínua - *Continuous Wavelet Transform* - CWT

*Wavelets* servem para representar sinais. Utilizam como ponto de partida as *wavelets* "mãe" $\psi(t)$[8] definidas como funções de suporte essencialmente compacto que obedecem a três propriedades [4]:

- A sua integral é zero,
$$\int_{-\infty}^{\infty} \psi(u)du = 0; \tag{2.26}$$
- São do tipo quadrado integrável (integral da função ao quadrado é igual à

[7] *Dilations*, em inglês.
[8] Exemplos de função $\psi$ serão encontrados no final deste item.

um)

$$\int_{-\infty}^{\infty} |\psi(u)|^2 du = 1; \tag{2.27}$$

- Satisfazem a condição de admissibilidade, pela qual a sua transformada de Fourier definida por

$$\Psi(f) \triangleq \int_{-\infty}^{\infty} \psi(u) e^{-i2\pi f u} du \tag{2.28}$$

é tal que existe uma constante finita

$$C_\psi \triangleq \int_{0}^{\infty} \frac{|\Psi(f)|^2}{f} df > 0. \tag{2.29}$$

As Eqs. 2.26 e 2.27 estão intimamente relacionadas com o fato das *wavelets* serem de suporte compacto, ou seja, sinais de duração finita que oscilam ao menos uma vez em torno da reta $x = 0$ [4]. Já a última condição, Eq. 2.28, é essencial para permitir a reconstrução de uma função $x(t)$ a partir da sua transformada *Wavelet* contínua - *Continuous Wavelet Transform* (CWT) que é definida como

$$W(\lambda, t) \triangleq \int_{-\infty}^{\infty} \psi_{\lambda,t}(u) x(u)\, du, \quad \text{em que} \quad \psi_{\lambda,t}(u) \triangleq \frac{1}{\sqrt{\lambda}} \psi\left(\frac{u-t}{\lambda}\right) \tag{2.30}$$

Deste modo, o sinal $x(t)$ é recuperado completamente mediante a transformada *Wavelet* contínua inversa - *Inverse Continuous Wavelet Transform* (ICWT) dada por:

$$x(t) = \frac{1}{C_\psi} \int_{0}^{\infty} \left[ \int_{-\infty}^{\infty} W(\lambda, u) \frac{1}{\sqrt{\lambda}} \psi\left(\frac{t-u}{\lambda}\right) du \right] \frac{d\lambda}{\lambda^2} \tag{2.31}$$

A função *wavelet* mais antiga é a função *wavelet* de Haar definida por:

$$\psi^{(H)}(u) \equiv \begin{cases} -1/\sqrt{2}, -1 < u \leq 0; \\ 1/\sqrt{2},\ 0 < u \leq 1; \\ 0,\ \text{qualquer outro u} \end{cases} \tag{2.32}$$

O que caracteriza a transformada *Wavelet* contínua é sua excepcional redundância, na medida que um sinal unidimensional acaba sendo representado por um plano, isto é, de uma dimensão passa-se a duas dimensões. A transformada *Wavelet* discreta - *Discrete Wavelet Transform* (DWT) definida em tempo e escala discretos minimiza esta redundância de representação por ser equivalente a uma subamostragem da CWT [5].

### 2.3.2 Transformada *Wavelet* Discreta - *Discrete Wavelet Transform* DWT

A DWT é a ferramenta básica para o estudo de séries temporais via *wavelet* e tem uma função análoga à da transformada de Fourier discreta na análise espectral [4].

A DWT pode ser apresentada como uma espécie de "subamostragem" da CWT pois, apesar de apresentar as mesmas propriedades desta, decompõe o sinal apenas em escalas múltiplas de 2 ($\lambda = 2^{j-1}$, j=1,2,3,. . . )[9]. Outra maneira de introduzir a DWT, a mais utilizada na literatura[10], é representá-la por meio de equações envolvendo operadores, i. e., matrizes ortogonais[11] [4].

A transformada *wavelet* discreta (DWT) da série temporal $\{X_t\}$ é uma transformada ortonormal assim como a transformada de Fourier discreta - *Ortogonal Discrete Fourier Tansform* (ODFT). Seja $\{W_n : n = 0, ..., N-1\}$ os coeficientes DWT, então, pode-se dizer que $\mathbf{W} = \mathcal{W}\mathbf{X}$, equação conhecida como equação de análise ou de decomposição, em que $\mathbf{W}$ é o vetor coluna de tamanho $N = 2^J$ do qual o $n$-ésimo elemento é o $n$-ésimo coeficiente DWT $W_n$, e $\mathcal{W}$ é uma matriz de transformação ortogonal de valor real $N$ x $N$ que define a DWT e satisfaz $\mathcal{W}^T\mathcal{W}$ $= I_N$. A ortonormalidade implica em $\mathbf{X} = \mathcal{W}^T\mathbf{W}$ e $||\mathbf{W}||^2 = ||\mathbf{X}||^2$. O sinal no tempo contido em X pode ser completamente recuperado por esta equação que é conhecida como equação de síntese ou de reconstrução. $W_n^2$ representa a contribuição à energia atribuída ao coeficiente DWT de índice $n$ [4].

Assim como coeficientes ODFT são associados com freqüências[12], o $n$-ésimo coeficiente *wavelet* $W_n$ é associado à uma certa escala e a um certo intervalo de tempo. A título ilustrativo será utilizada a DWT de Haar com $N = 16$. A Fig. 2.6 mostra os elementos de $\mathcal{W}$ num formato linha a linha. Nesta figura, os quadrados na linha horizontal representam elementos de $\mathcal{W}$ que são zero, no entanto, cada linha tem alguns elementos que não são iguais a zero em locais diferentes (relacionado ao conjunto de tempos). Então, as linhas desta matriz para $n = 0$, 8, 12, 14 e 15 são:

$$\mathcal{W}_{0\bullet}^{T\,13} = [-\tfrac{1}{\sqrt{2}}, \tfrac{1}{\sqrt{2}}, \underbrace{0, \ldots, 0}_{14\ zeros}]$$

[9]Diz-se que a DWT trabalha com escalas "diádicas", em inglês: "dyadic scales".

[10]Na Engenharia a descrição via banco de filtros é a mais usual.

[11]Uma matriz $\mathcal{O}_{N\times N}$ de valores reais é dita ortogonal se sua transposta for igual à sua inversa, isto é, se $\mathcal{O}\mathcal{O}^T = I_N$, em que $I_N$ é a matriz identidade de ordem $N$. A mesma definição pode ser estendida para matrizes complexas substituindo "$T$" por "$H$" (transposição Hermitiana). Matrizes "complexas ortogonais" são chamadas de unitárias.

[12]Número de ciclos por amostra.

[13]A notação $\mathcal{W}_{j\bullet}^T$ significa vetor transposto contendo elementos da $j$-ésima linha da matriz

$$\mathcal{W}_{8\bullet}^T = [-\tfrac{1}{2},-\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\underbrace{0,\ldots,0}_{12\ zeros}]$$

$$\mathcal{W}_{12\bullet}^T = [\underbrace{-\frac{1}{\sqrt{8}},\ldots,-\frac{1}{\sqrt{8}}}_{4\ desses},\underbrace{\frac{1}{\sqrt{8}},\ldots,\frac{1}{\sqrt{8}}}_{4\ desses},\underbrace{0,\ldots,0}_{8\ zeros}]$$

$$\mathcal{W}_{14\bullet}^T = [\underbrace{-\frac{1}{4},\ldots,-\frac{1}{4}}_{8\ desses},\underbrace{\frac{1}{4},\ldots,\frac{1}{4}}_{8\ desses}]$$

$$\mathcal{W}_{15\bullet}^T = [\underbrace{\frac{1}{4},\ldots,\frac{1}{4}}_{16\ desses}]$$

As onze demais linhas são versões deslocadas das acima:

$\mathcal{W}_{1\bullet}$[14] $= \Gamma^2\mathcal{W}_{0\bullet}$ $\mathcal{W}_{2\bullet} = \Gamma^4\mathcal{W}_{0\bullet}$ $\ldots\mathcal{W}_{7\bullet} = \Gamma^{14}\mathcal{W}_{0\bullet}$

$\mathcal{W}_{9\bullet} = \Gamma^4\mathcal{W}_{8\bullet}$ $\mathcal{W}_{10\bullet} = \Gamma^8\mathcal{W}_{8\bullet}$ $\mathcal{W}_{11\bullet} = \Gamma^{12}\mathcal{W}_{8\bullet}$

$\mathcal{W}_{13\bullet} = \Gamma^8\mathcal{W}_{12\bullet}$

Pela figura é fácil ver que o produto interno de duas linhas diferentes é zero o que estabelece que $\mathcal{W}$ é ortonormal.

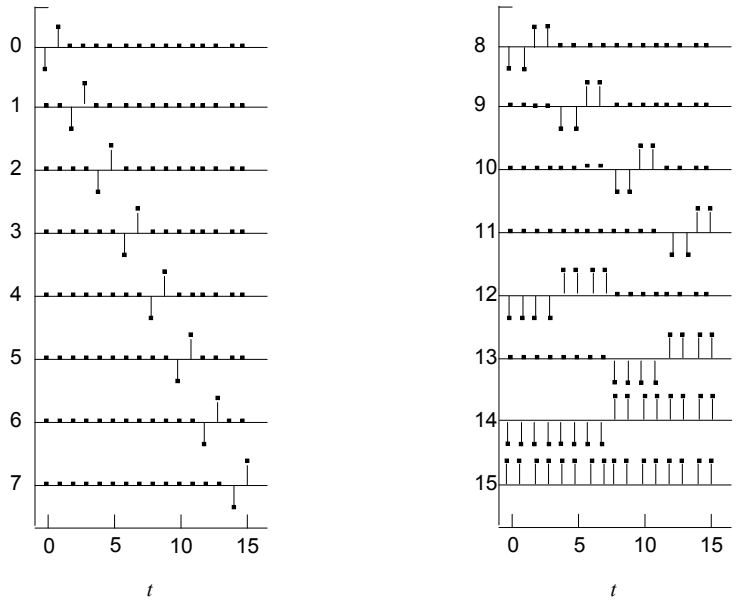



**Figura 2.6:** Vetores linha $\mathcal{W}_{n\bullet}^T$ da matriz transformada *wavelet* discreta $\mathcal{W}$ baseada na *wavelet* de Haar para $N = 16$ e $n = 0$ a 7 (de cima para baixo à esquerda) e $n = 8$ a 15 (direita) - fonte: [4] página 57.

Definindo exatamente o que a noção de escala significa, para um inteiro posi-

$N \times N$ $\mathcal{W}$.

[14] A notação $\mathcal{W}_{j\bullet}$ significa vetor contendo elementos da $j$-ésima linha da matriz $N \times N$ $\mathcal{W}$.

tivo $\lambda$:

$$\overline{X}_t(\lambda) \equiv \frac{1}{\lambda}\sum_{l=0}^{\lambda-1} X_{t-l} \tag{2.33}$$

A equação 2.33 representa a média dos valores de dados contínuos $\lambda$ com índices de t - $\lambda$ + 1 a t (note que $\overline{X}_t(1) = X_t$, pode ser considerado como "média de um único ponto", e que $\overline{X}_{N-1}(N) = \overline{X}$, é a média da amostra de todos valores N). Refere-se a $\overline{X}_t(\lambda)$ como a média da amostra para escala $\lambda$ sobre o intervalo de tempo t - $\lambda$ + 1 a t. Desde que $\mathbf{W} = \mathcal{W}\mathbf{X}$, levando-se em consideração as linhas da matriz $\mathcal{W}$ para a DWT de Haar com $N = 16$, pode-se escrever:

$$\mathbf{W} = \begin{bmatrix} W_0 \\ \vdots \\ W_7 \\ W_8 \\ \vdots \\ W_{11} \\ W_{12} \\ W_{13} \\ W_{14} \\ W_{15} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}}(X_1 - X_0) \\ \vdots \\ \frac{1}{\sqrt{2}}(X_{15} - X_{14}) \\ \frac{1}{2}(X_3 + X_2 - X_1 - X_0) \\ \vdots \\ \frac{1}{2}(X_{15} + X_{14} - X_{13} - X_{12}) \\ \frac{1}{\sqrt{8}}(X_7 + ... + X_4 - X_3 - ... - X_0) \\ \frac{1}{\sqrt{8}}(X_{15} + ... + X_{12} - X_{11} - ... - X_8) \\ \frac{1}{4}(X_{15} + ... + X_8 - X_7 - ... - X_0) \\ \frac{1}{4}(X_{15} + ... + X_0) \end{bmatrix} \tag{2.34}$$

Utilizando a definição da equação 2.33 para $\overline{X}_t(\lambda)$, pode-se reescrever $W_n$ como:

$W_0 = \frac{1}{\sqrt{2}}[\overline{X}_1(1)$ - $\overline{X}_0(1)]$,..., $W_7 = \frac{1}{\sqrt{2}}[\overline{X}_{15}(1)$ - $\overline{X}_{14}(1)]$

$W_8 = \overline{X}_3(2)$ - $\overline{X}_1(2)$,..., $W_{11} = \overline{X}_{15}(2)$ - $\overline{X}_{13}(2)$

$W_{12} = \sqrt{2}[\overline{X}_7(4)$ - $\overline{X}_3(4)]$, $W_{13} = \sqrt{2}[\overline{X}_{15}(4)$ - $\overline{X}_{11}(4)]$

$W_{14} = 2[\overline{X}_{15}(8)$ - $\overline{X}_7(8)]$

$W_{15} = 4\overline{X}_{15}(16)$

Nota-se que os primeiros 8 coeficientes DWT $W_0$,...,$W_7$, são proporcionais às diferenças (mudanças) das médias adjacentes de $X_t$ numa escala unitária; os próximos quatro, $W_8$,..., $W_{11}$, são diferenças das médias adjacentes numa escala de dois; $W_{12}$ e $W_{13}$ são proporcionais às diferenças numa escala de quatro; $W_{14}$ é proporcional à diferença numa escala de oito; e o coeficiente final $W_{15}$ é propor-

cional à média de todos os dados [4].

Para $N$=$2^J$ genérico e para a função de Haar e outras DWTs, os elementos de $\mathcal{W}$ podem ser organizados tal que os primeiros $N/2$ coeficientes DWT são associados com mudanças de escala unitária; os próximos $N/4$ coeficientes com mudanças na escala de dois; e assim por diante até chegar aos coeficientes $W_{N-4}$ e $W_{N-3}$ os quais são associados a mudanças na escala de $N/4$; o coeficiente $W_{N-2}$ é associado à uma mudança na escala de $N/2$; e finalmente $W_{N-1}$ é novamente proporcional à média de todos os dados. Há exatamente $N/(2\tau_j)$ coeficientes na DWT associada com mudanças na escala $\tau_j$, onde $\tau_j \equiv 2^{j-1}$ para j = 1,..., J[15]; adicionalmente, existe um coeficiente $W_{N-1}$ associado com a média na escala N. Os $N$ - 1 coeficientes que são associados a mudanças em várias escalas são chamados coeficientes *wavelet* e $W_{N-1}$ é chamado de coeficiente de escala.

É importante notar que $\tau_j$ é uma escala padronizada sem unidade pré-definida.

Cada coeficiente *wavelet* em cada escala também é localizado no tempo. No exemplo acima, $W_0$ envolve somente tempos t = 0 e 1, por outro lado $W_8$ envolve tempos t = 0, 1, 2 e 3. Em comparação, os coeficientes ODFT não são localizados no tempo o que é uma importante distinção entre uma transformada "global" como a ODFT e uma transformada "local" como a DWT. A noção de que os coeficientes *wavelet* são relacionados a diferenças (de várias ordens) de valores médios de porções de $\{X_t\}$ concentradas no tempo não é somente válida para a transformada *wavelet* de Haar e sim fundamental para todas transformadas *wavelet* [4].

Outro exemplo de DWT é a *wavelet* D(4) que é um exemplo de filtro *wavelet Finite Impulse Response* (FIR). A Fig. 2.7 mostra a matriz $\mathcal{W}$ da *wavelet* D(4) com $N = 16$. As oito primeiras linhas dessa matriz correspondem a escala unitária. Cada uma das linhas está baseada em quatro valores diferentes de zero:

$$h_0 = \frac{1-\sqrt{3}}{4\sqrt{2}}, h_1 = \frac{-3+\sqrt{3}}{4\sqrt{2}}, h_2 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad \text{e} \quad h_3 = \frac{-1-\sqrt{3}}{4\sqrt{2}} \tag{2.35}$$

Por exemplo, tem-se:

$\mathcal{W}_{0\bullet}^T = [h_1,h_0,\underbrace{0,\ldots,0}_{12\ zeros},h_3,h_2]$ e $\mathcal{W}_{1\bullet}^T = [h_3,h_2,h_1,h_0,\underbrace{0,\ldots,0}_{12\ zeros}]$;

ou seja, $\mathcal{W}_{1\bullet} = \Gamma^2\mathcal{W}_{0\bullet}$. A ortonormalidade de $\mathcal{W}$ requer que:

[15]Note que $\tau_1 = 1$ e $\tau_J = N/2$.

$$\|\mathcal{W}_{0\bullet}\|^2 = h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1 \quad \text{e} \quad \langle \mathcal{W}_{0\bullet}, \mathcal{W}_{1\bullet} \rangle = h_0 h_2 + h_1 h_3 = 0 \tag{2.36}$$

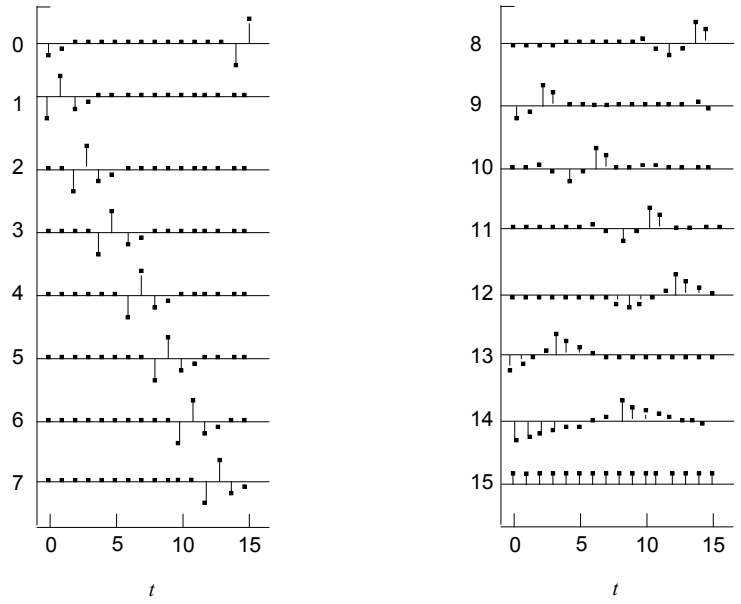



**Figura 2.7:** Vetores linha $\mathcal{W}_{n\bullet}^T$ da matriz transformada *wavelet* discreta $\mathcal{W}$ baseada na *wavelet* D(4) para $N = 16$ e $n = 0$ a 7 (de cima para baixo à esquerda) e $n = 8$ a 15 (direita). Fonte: [4] página 60.

Como foi mencionado anteriormente, cada coeficiente *wavelet* de Haar na escala unitária pode ser obtido mudando de escala a primeira diferença entre as "médias de um único ponto" adjacentes tal que $W_n \propto X_{2n+1} - X_{2n}$ para $n =$ 0,...,7. Por comparação, cada coeficiente *wavelet* D(4) na escala unitária pode ser obtido mudando de escala apropriadamente a segunda diferença da média ponderada de dois pontos consecutivos, ou seja, primeiro forma-se a média ponderada $Y_t \equiv aX_t + bX_{t-1}$ e então a segunda diferença de $Y_t$ que, por definição, é a primeira diferença da primeira diferença. Então, se $Y_t^{(1)} \equiv Y_t - Y_{t-1}$ representa a primeira diferença de $\{Y_t\}$, então:

$Y_t^{(2)} \equiv Y_t^{(1)} - Y_{t-1}^{(1)} = Y_t - 2Y_{t-1} + Y_{t-2}$ é a segunda diferença. Com uma escolha particular de a e b, os coeficientes *wavelet* da D(4) na escala unitária são dados por:

$$
\begin{aligned}
W_n &= Y_{2n+1} - 2Y_{2n} + Y_{2n-1} \\
&= aX_{2n+1} + bX_{2n} - 2(aX_{2n} + bX_{2n-1}) + aX_{2n-1} + bX_{2n-2} \\
&= aX_{2n+1} + (b-2a)X_{2n} + (a-2b)X_{2n-1} + bX_{2n-2} \\
&= h_0X_{2n+1} + h_1X_{2n} + h_2X_{2n-1} + h_3X_{2n-2}
\end{aligned} \tag{2.37}
$$

para $n = 0,\ldots,7$ (onde define-se que $X_{-1} \equiv X_{15}$ e $X_{-2} \equiv X_{14}$ pela circularidade). As duas condições da equação 2.36 podem ser utilizadas para encontrar soluções para a e b das quais os valores dados para $h_0$,$h_1$,$h_2$ e $h_3$ na equação 2.35 são uma possibilidade.

Pode-se observar um padrão para as DWT de Haar e D(4) que são válidas para todas *wavelets* de Daubechies. Para todo $L$ par e $L < N$, $h_0$, $h_1$,..., $h_{L-1}$ representam os elementos diferentes de zero das linhas de $\mathcal{W}$ associadas à escala unitária, então, cada coeficiente *wavelet* da escala unitária pode ser obtido pela escala apropriada da $L$-ésima/2 diferença da média ponderada dos $L$-ésimos/2 pontos consecutivos. Os pesos $L/2$ e os $h_l$s podem ser determinados pelas $L/2$ condições análogas à equação 2.36, ou seja,$\sum h_l^2 = 1$ e:

$$
\begin{aligned}
h_0h_2 + h_1h_3 + \ldots + h_{L-3}h_{L-1} &= 0 \\
h_0h_4 + h_1h_5 + \ldots + h_{L-5}h_{L-1} &= 0 \\
&\vdots \\
h_0h_{L-2} + h_1h_{L-1} &= 0
\end{aligned} \tag{2.38}
$$

ou seja, $\{h_l\}$ é ortogonal aos seus deslocamentos pares.

Assumindo que a série temporal $\mathbf{X}$ tem tamanho $N = 2^J$, as linhas da matriz $\mathcal{W}$ podem ser agrupadas em $J + 1$ sub-matrizes, o que leva a decomposição dos elementos do vetor $\mathbf{W}$ em J + 1 sub-vetores. Os primeiros $J$ sub-vetores são denotados como $\mathbf{W}_j$, j = 1,...,J e o $j$-ésimo sub-vetor contém todos os coeficientes DWT para escala $\tau_j$. Note que $\mathbf{W}_j$ é um vetor coluna com $N/2^j$ elementos. O sub-vetor final é denotado como $\mathbf{V}_J$ e contém somente o coeficiente de escala $W_{N-1}$. Então, pode-se escrever:

$$\mathcal{W}\mathbf{X} = \begin{bmatrix} \mathcal{W}_1 \\ \mathcal{W}_2 \\ \vdots \\ \mathcal{W}_J \\ \mathcal{V}_J \end{bmatrix} \mathbf{X} = \begin{bmatrix} \mathcal{W}_1\mathbf{X} \\ \mathcal{W}_2\mathbf{X} \\ \vdots \\ \mathcal{W}_J\mathbf{X} \\ \mathcal{V}_J\mathbf{X} \end{bmatrix} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_J \\ \mathbf{V}_J \end{bmatrix} = \mathbf{W} \tag{2.39}$$

As submatrizes $\mathcal{W}_j$ são de dimensão $\frac{N}{2^j} \times N$; $\mathcal{V}_J$ é de dimensão $1 \times N$; $\mathbf{W}_j$ é vetor coluna de tamanho $\frac{N}{2^j}$; e $\mathbf{V}_J$ contém o último elemento de $\mathbf{W}$. Em cada submatriz $\mathcal{W}_j$, as linhas são circularmente deslocadas em relação à anterior. Os coeficientes *wavelet* do vetor $\mathbf{W}_j$ são associados com diferenças (de várias ordens) de médias adjacentes na escala de $\tau_j = 2^{j-1}$, enquanto o coeficiente de escala em $\mathbf{V}_J$ é igual a $\sqrt{N}$ vezes a média da amostra $\overline{X}$ de $\mathbf{X}$.

Quando N = $2^J = 16$, então J = 4:

$\mathbf{W}_1^T = [W_0, W_1, W_2, W_3, W_4, W_5, W_6, W_7]$

$\mathbf{W}_2^T = [W_8, W_9, W_{10}, W_{11}]$

$\mathbf{W}_3^T = [W_{12}, W_{13}]$

$\mathbf{W}_4^T = [W_{14}]$

$\mathbf{V}_4^T = [W_{15}]$

Devido à ortonormalidade de $\mathcal{W}$ e à forma de $\mathbf{V}_J$, pode-se decompor a variância da amostra (potência empírica) de $\mathbf{X}$ em pedaços que são associados com escalas $\tau_1,\ldots,\tau_J$:

$$\hat{\sigma}_X^2 \equiv \frac{1}{N}\|\mathbf{X}\|^2 - \overline{X}^2 = \frac{1}{N}\|\mathbf{W}\|^2 - \overline{X}^2 = \frac{1}{N}\sum_{j=1}^{J}\|\mathbf{W}_j\|^2 \equiv \sum_{j=1}^{J} P_{\mathcal{W}}(\tau_j) \tag{2.40}$$

em que $\|\mathbf{W}_j\|^2/N$ representa a contribuição da variância da amostra de $\{X_t\}$ devido às mudanças na escala $\tau_j$ e a seqüência $\{P_{\mathcal{W}}(\tau_j)\}$ é o espectro da potência empírica da *wavelet* discreta.

A matriz $\mathcal{W}$ também pode ser construída a partir dos coeficientes dos filtros *wavelet* "$h_l$"(passa alta) e de escala "$g_l$" (passa baixa) discretos (Fig. 2.8). As diferentes funções *wavelets* (Haar, Daubechies, coiflets, etc) são definidas nesta matriz. O filtro *wavelet* de valor real $\{h_l{:}l = 0,\ldots,L\text{-}1\}$ deve ter soma nula, ter energia unitária e ser ortogonal a deslocamentos pares conforme as duas pro-

priedades seguintes:

$$(i)\sum_{l=0}^{L-1} h_l = 0 \quad \text{e} \quad (ii)\sum_{l=-\infty}^{\infty} h_l h_{l+2n} = \begin{cases} 1, & n = 0 \\ 0, & n = \pm 1, \pm 2, \ldots \end{cases} \tag{2.41}$$

em que "L" é a ordem do filtro "$h_l$" caso ele seja de comprimento finito, $h_0 \neq 0$ e $h_{L-1} \neq 0$, $h_l \equiv 0$ para $l < 0$ e $l \geq L$ e $n$ é qualquer valor inteiro diferente de zero. Neste caso a relação entre "$h_l$" e "$g_l$" é dada por

$$\begin{aligned} g_l &\equiv (-1)^{l+1} h_{L-1-l} \\ h_l &= (-1)^l g_{L-1-l}. \end{aligned} \tag{2.42}$$

O filtro de escala "$g_l$" deve satisfazer as condições:

$$(i)\sum_{l=0}^{L-1} g_l = \sqrt{2} \quad , \quad (ii)\sum_{l=-\infty}^{\infty} g_l g_{l+2n} = \begin{cases} 1, & n = 0 \\ 0, & n = \pm 1, \pm 2, \ldots \end{cases} \quad \text{e} \quad (iii)\sum_{l=-\infty}^{\infty} g_l h_{l+2n'} = 0. \tag{2.43}$$

para todo $n$ inteiro e diferente de zero e todos inteiros $n'$.

Os $N/2$ coeficientes *wavelet* para escala unitária são definidos como:

$$\begin{aligned} W_{1,t} &\equiv 2^{\frac{1}{2}} \widetilde{W}_{1,2t+1}, t = 0, \ldots, \frac{N}{2} - 1, \quad \text{com} \\ 2^{\frac{1}{2}} \widetilde{W}_{1,t} &\equiv \sum_{l=0}^{L-1} h_l X_{t-l mod N}, t = 0, \ldots, N-1; \end{aligned} \tag{2.44}$$

ou seja, $\{X_t\}$ é filtrado com $\{h_l\}$ para obter $\{2^{\frac{1}{2}} \widetilde{W}_{1,t}\}$, o qual é subamostrado para obter os coeficientes *wavelet*. Também pode-se obter $W_{1,t}$ diretamente sem precisar passar por $2^{\frac{1}{2}} \widetilde{W}_{1,t}$:

$$W_{1,t} = \sum_{l=0}^{L-1} h_l X_{2t+1-l mod N} = \sum_{l=0}^{N-1} h_l^{\circ} X_{2t+1-l mod N}, t = 0, \ldots, \frac{N}{2} - 1, \tag{2.45}$$

em que $\{h_l^{\circ}\}$ é $\{h_l\}$ periodizado ao tamanho N. Esses coeficientes formam os primeiros elementos de $\mathbf{W} = \mathcal{W}\mathbf{X}$, ou seja, os elementos do subvetor $\mathbf{W}_1 = \mathcal{W}_1\mathbf{X}$, em que $\mathcal{W}_1$ é a matriz $\frac{N}{2} \times N$ das $\frac{N}{2}$ primeiras linhas de $\mathcal{W}$. A primeira linha de $\mathcal{W}_1$ é dada por:

$\mathcal{W}_{0\bullet}^T = [h_1^{\circ}, h_0^{\circ}, h_{N-1}^{\circ}, h_{N-2}^{\circ}, \ldots, h_2^{\circ}]$, enquanto as demais $\frac{N}{2} - 1$ linhas são provenientes do deslocamento circular de $\mathcal{W}_{0\bullet}^T$. Sendo assim, $\mathcal{W}_{t\bullet}^T = [\Gamma^{2t}\mathcal{W}_{0\bullet}]^T$, t =

1,...,$\frac{N}{2}$ - 1. As propriedades de energia unitária e ortogonalidade para deslocamentos pares dos filtros *wavelet* implicam que as linhas de $\mathcal{W}_1$ sejam um conjunto de $\frac{N}{2}$ vetores ortonormais. Examinando os filtros *wavelet* de Daubechies (de Haar e o D(4)) na Fig. 2.8 verifica-se que na prática esses filtros *wavelet* são filtros passa-alta com uma banda nominal definida por $|f| \in [1/4, 1/2]$.

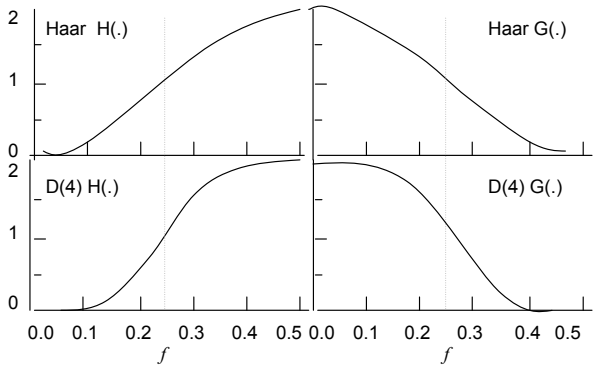



**Figura 2.8:** Funções ganho quadrado para o filtro *wavelet* de Haar (gráfico acima à esquerda), filtro de escala de Haar (gráfico acima à direita. Filtro *wavelet* de D(4) (gráfico abaixo à esquerda) e filtro de escala D(4) (gráfico abaixo à direita). As linhas pontilhadas marcam a freqüência $f = 1/4$. Fonte: [4] página 73.

Os $N/2$ coeficientes de escala de primeiro nível são definidos como:

$$\begin{aligned} V_{1,t} &\equiv 2^{\frac{1}{2}}\widetilde{V}_{1,2t+1}, t = 0, \ldots, \frac{N}{2} - 1, \quad \text{com} \\ 2^{\frac{1}{2}}\widetilde{V}_{1,t} &\equiv \sum_{l=0}^{L-1} g_l X_{t-l mod N}, t = 0, \ldots, N-1; \end{aligned} \tag{2.46}$$

Também pode-se obter $V_{1,t}$ diretamente sem precisar passar por $2^{\frac{1}{2}}\widetilde{V}_{1,t}$:

$$V_{1,t} = \sum_{l=0}^{L-1} g_l X_{2t+1-lmodN} = \sum_{l=0}^{N-1} g_l^{\circ} X_{2t+1-lmodN}, t = 0, \ldots, \frac{N}{2} - 1, \tag{2.47}$$

em que $\{g_l^{\circ}\}$ é $\{g_l\}$ periodizado ao tamanho N. Esses coeficientes formam os $N/2$ elementos de $\mathbf{V}_1 = \mathcal{V}_1\mathbf{X}$, em que $\mathcal{V}_1$ é a matriz $\frac{N}{2} \times N$ da qual a primeira linha é dada por:

$\mathcal{V}_{0\bullet}^T = [g_1^{\circ}, g_0^{\circ}, g_{N-1}^{\circ}, g_{N-2}^{\circ}, \ldots, g_2^{\circ}]$, enquanto as demais $\frac{N}{2} - 1$ linhas são provenientes do deslocamento circular de $\mathcal{V}_{0\bullet}^T$. Sendo assim, $\mathcal{V}_{t\bullet}^T = [\Gamma^{2t}\mathcal{V}_{0\bullet}]^T$, t = 1,. . .,$\frac{N}{2}$ - 1. As propriedades de energia unitária e ortogonalidade para deslocamentos pares dos filtros de escala implicam que as linhas de $\mathcal{V}_1$ sejam um conjunto de $\frac{N}{2}$ vetores ortonormais. O fato adicional de que o filtro de escala é ortogonal ao filtro *wavelet* e todos os seus deslocamentos pares leva a $\mathcal{V}_1$ e $\mathcal{W}_1$ serem ortogonais, ou seja, podem formar a matriz $N \times N$ ortonormal resultante:

$$\mathcal{P}_1 \equiv \begin{bmatrix} \mathcal{W}_1 \\ \mathcal{V}_1 \end{bmatrix} \tag{2.48}$$

Examinando os filtros de escala (de Haar e o D(4)) na Fig. 2.8 verifica-se que na prática esses filtros de escala são filtros passa-baixa com uma banda nominal definida por $|f| \in [0, 1/4]$.

Supondo ainda que se deseje decompor um sinal temporal

$$\mathbf{X} = [x(0), x(1), \ldots, x(k), \ldots, x(N-1)]^T$$

em $J_0$ escalas ($j = 0, 1, \ldots, J_0$), o vetor de coeficientes transformados $\mathbf{W}_{N\times 1}$ e a matriz ortogonal $\mathcal{W}_{N\times N}$ podem ser particionados em subvetores e submatrizes, respectivamente, de maneira que:

$$\mathcal{W} = \begin{bmatrix} \mathcal{W}_1 \\ \mathcal{W}_2 \\ \vdots \\ \mathcal{W}_{J_0} \\ \mathcal{V}_{J_0} \end{bmatrix} \quad \text{e} \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \vdots \\ \mathbf{W}_{J_0} \\ \mathbf{V}_{J_0} \end{bmatrix} \tag{2.49}$$

As submatrizes $\mathcal{W}_j$ são de dimensão $\frac{N}{2^j} \times N$ e suas linhas são construídas com os filtros "$h_l$", enquanto $\mathcal{V}_{J_0}$ é uma matriz $\frac{N}{2^J} \times N$ construída com filtros "$g_l$".

Desse modo, a equação de decomposição $\mathbf{W} = \mathcal{W}\mathbf{X}$ equivale à filtrar circular-

mente a série temporal representada por $\mathbf{X}$ e a reter todos os valores ímpares da saída. A equação de reconstrução $\mathbf{X} = \mathcal{W}^T\mathbf{W}$ equivale exatamente ao contrário, ou seja, sobreamostra-se o vetor de coeficientes transformados e filtra-se o resultado de forma a se recuperar a série temporal original [5].

Foi utilizando esta interpretação que Mallat [53] propôs um algoritmo de construção da DWT chamado de piramidal e ilustrado nas Figs. 2.9 e 2.10 em que [16]$H(\frac{k}{N/2^{j-1}})$ e $G(\frac{k}{N/2^{j-1}})$ representam, respectivamente, a função de transferência dos filtros circulares passa-alta ("*Wavelet filter*") e passa-baixa ("*scaling filter*") na escala $j$. O símbolos "$\downarrow 2$" e "$\uparrow 2$" indicam subamostragem e sobreamostragem do sinal. Os filtros de reconstrução são respectivamente $H^*(\frac{k}{N/2^{j-1}})$ e $G^*(\frac{k}{N/2^{j-1}})$, ou seja, o conjugado complexo dos filtros usados na decomposição. O algoritmo calcula a DWT do sinal sem formar completamente a matriz $\mathcal{W}$ requerendo um número de operações menor (ordem $O(N)$)[17] que o obtido por meio da multiplicação direta de matrizes (ordem $O(N^2)$).

Um efeito da subamostragem no espectro do sinal original é o "alargamento" do espectro filtrado, resultando em uma expansão ("dilação") do espectro. No entanto, na saída dos filtros passa-alta, este alargamento aliado ao fenômeno de "aliasing" ainda causa uma reversão do espectro de interesse conforme mostrado pelo gráficos da Fig. 2.9. O número de decomposições ideal depende, entre outras coisas, do intervalo espectral que se deseja analisar. No caso de um sinal com 4 faixas espectrais de interesse como o sinal da Fig. 2.9, não há necessidade de uma decomposição maior que 2 níveis, pois o espectro está completamente decomposto nesta escala [5].

Ainda, por meio das equações de decomposição e reconstrução, verifica-se que a análise via DWT define uma verdadeira análise de multirresolução (MRA) de $\mathbf{X}$, decompondo um vetor temporal em $J_0$ somas de vetores $\mathcal{D}_j$ (chamado de detalhe pois contém as componentes de alta freqüência) mais o vetor $\mathcal{S}_{J_0}$ (chamado de componente liso do sinal por possuir baixas freqüências). A Fig. 2.11 ilustra esse fato ao mostrar como as diferentes funções de transferência dos filtros da Fig. 2.9 decompõem o espectro do sinal original. O sinal original $\mathbf{X}$ neste caso é igual à soma dos detalhes mais a escala.

[16] Aqui o índice "$k''$ das funções de transferência é diferente do índice "$k''$ da série temporal discreta.

[17] A notação $O(a_N)$ significa que o número de operações necessárias $N_{op}$, a menos de uma constante multiplicativa $C$, é menor ou igual a $a_N$ para qualquer $N$ ($N_{op} \leq Ca_N$). A DWT, portanto, exige menos operações que a própria transformada rápida de Fourier (FFT) que é de ordem $O(N \log_2 N)$.

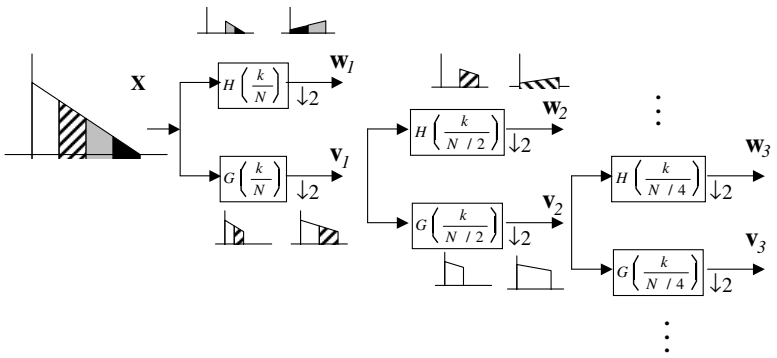



**Figura 2.9:** Decomposição Piramidal. Um sinal de entrada com quatro bandas de interesse é decomposto resultando no aumento na resolução espectral dos componentes de baixa freqüência. Fonte: [5] página 19.

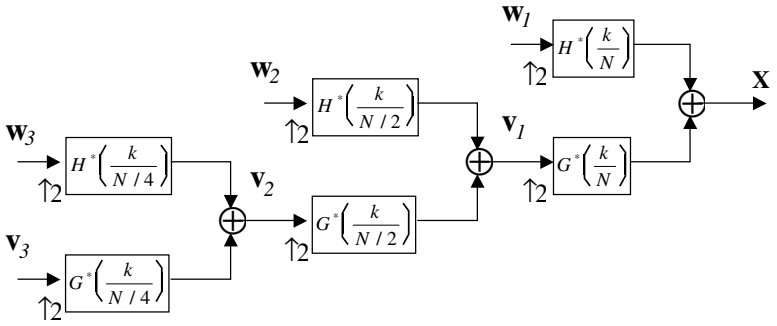



**Figura 2.10:** Reconstrução Piramidal. Fonte: [5] página 19.

$$\mathbf{X} = \mathcal{W}^T\mathbf{W} = \sum_{j=1}^{J_0} \mathcal{W}_j^T\mathbf{W}_j + \mathcal{V}_{J_0}^T\mathbf{V}_{J_0} \triangleq \sum_{j=1}^{J_0} \mathcal{D}_j + \mathcal{S}_{J_0} \tag{2.50}$$

Observando a definição da DWT verifica-se que a mesma exige um número de amostras sempre múltiplo de 2 ($N = 2^J, J \in \mathcal{N}$). Além disso, os coeficientes da DWT variam quando o vetor de entrada é deslocado temporalmente (por conta das subamostragens, cada escala possui um número de coeficientes diferente). Estas duas deficiências da DWT são corrigidas pela DWT modificada - MODWT [5].

#### 2.3.2.1 Primeiro estágio do algoritmo da pirâmide

O primeiro estágio do algoritmo da pirâmide para computar a DWT consiste em transformar a série temporal $\mathbf{X}$ de tamanho $N = 2^J$ nos $N/2$ coeficientes *wavelet* $\mathbf{W}_1$ e nos $N/2$ coeficientes de escala $\mathbf{V}_1$ de primeiro nível. A Fig. 2.12 ilustra essa transformação do ponto de vista de filtros. Existem $J$ - 1 estágios do algoritmo da pirâmide. Para $j = 2,\ldots,J$, o $j$-ésimo estágio transforma o vetor

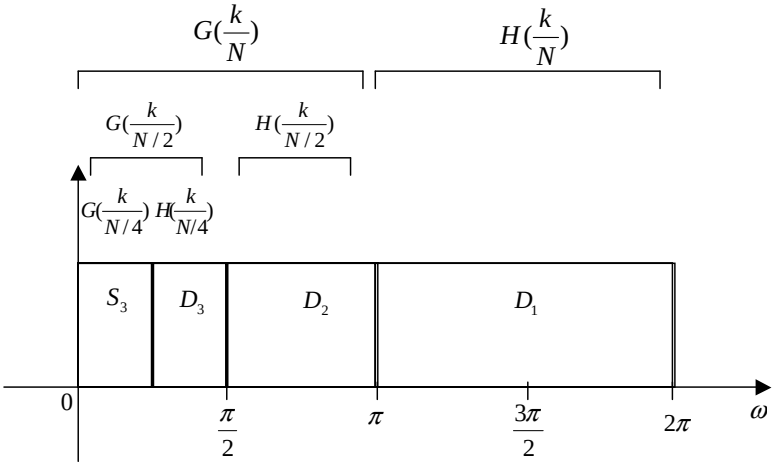



**Figura 2.11:** Análise de Multirresolução. Fonte: [5] página 20.

$\mathbf{V}_{j-1}$ de tamanho $N/2^{j-1}$ nos vetores $\mathbf{W}_j$ e $\mathbf{V}_j$, cada um de tamanho $N/2^j$. No $j$-ésimo estágio o vetor $\mathbf{V}_{j-1}$ é tratado exatamente como $\mathbf{X}$ no primeiro estágio: os elementos de $\mathbf{V}_{j-1}$ são filtrados separadamente com $\{h_l\}$ e $\{g_l\}$, e a saída do filtro é subamostrada para formar, respectivamente, $\mathbf{W}_j$ e $\mathbf{V}_j$. Os elementos de $\mathbf{V}_j$ são chamados de coeficientes de escala de nível $j$ e os de $\mathbf{W}_j$ são chamados de coeficientes *wavelet* para o nível $j$. No final do $J$-ésimo estágio, forma-se o vetor de coeficiente DWT $\mathbf{W}$ pela concatenação dos $J$+1 vetores $\mathbf{W}_1$,...,$\mathbf{W}_J$ e $\mathbf{V}_J$ conforme a equação 2.39.

Assim, o primeiro estágio do algoritmo da pirâmide decompõe ortogonalmente a série temporal $\{X_t{:}t{=}0,\ldots,N\text{-}1\}$ e em duas novas séries: $\{W_{1,t}{:}t{=}0,\ldots,\frac{N}{2}\text{-}1\}$ e $\{V_{1,t}{:}t{=}0,\ldots,\frac{N}{2}\text{-}1\}$.

Os coeficientes *wavelet* de primeiro nível $\{W_{1,t}\}$ são associados com:

1. $\mathcal{W}_1$, matriz $\frac{N}{2} \times N$ que satisfaz $\mathcal{W}_1\mathcal{W}_1^T = I_{\frac{N}{2}}$ e consiste das primeira $N/2$ linhas da matriz DWT $\mathcal{W}$ (cada linha de $\mathcal{W}_1$ contém os elementos de $\{h_l^\circ\}$ $\longleftrightarrow \{H(\frac{k}{N})\}$, ou seja, o filtro *wavelet* periodizado ao tamanho $N$; a primeira linha é $\left[h_1^\circ, h_0^\circ, h_{N-1}^\circ, h_{N-2}^\circ, \ldots, h_2^\circ\right]$, e as demais $\frac{N}{2}$-1 linhas são formadas pelo deslocamento circular da primeira linha para direita por, respectivamente, 2,4,6,...,$N$-2 unidades);

2. Os elementos do vetor $\mathbf{W}_1 = \mathcal{W}_1\mathbf{X}$ de tamanho $N/2$;

3. O detalhe de primeiro nível $\mathcal{D}_1 = \mathcal{W}_1^T\mathbf{W}_1 = \mathcal{W}_1^T\mathcal{W}_1\mathbf{X}$, o qual pode ser formado pela sobreamostragem de $\mathbf{W}_1$ por dois e então filtrá-lo com $\{H^*(\frac{k}{N}\}$;

4. Mudanças nas médias de escala $\tau_1 = 1$;

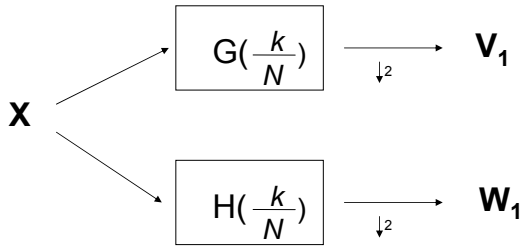



**Figura 2.12:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_1$ e $\mathbf{V}_1$. A série temporal no vetor $\mathbf{X}$ de tamanho $N$ é circularmente filtrada utilizando o filtro *wavelet* $H(\cdot)$ periodizado ao tamanho $N$ (a forma desse filtro no domínio de freqüência é dada por $\{H(\frac{k}{N}) : k = 0, \ldots, N-1\}$). Todos os valores de índices ímpares da série filtrada são utilizados para formar o vetor $\mathbf{W}_1$ de tamanho $N/2$ contendo os coeficientes *wavelet* de nível $j = 1$("$\downarrow 2$" indica subamostragem por dois); de maneira similar, o vetor $\mathbf{V}_1$ de tamanho $N/2$ contendo os coeficientes de escala de nível $j = 1$ é obtido pela subamostragem da saída da filtragem de $\mathbf{X}$ com o filtro de escala $G(\cdot)$ periodizado ao tamanho $N$. Fonte: [4] página 80.

5. Subamostragem por dois do processo de meia-banda associado com as freqüências altas [1/4,1/2] na série temporal $\{X_t\}$; e

6. A porção da DFT de $\{X_t\}$ com freqüências no intervalo [1/4,1/2], com uma ordem reversa.

Os coeficientes de escala de primeiro nível $\{V_{1,t}\}$ estão associados com:

1. $\mathcal{V}_1$, matriz $\frac{N}{2} \times N$ que satisfaz ambos $\mathcal{V}_1\mathcal{V}_1^T = I_{\frac{N}{2}}$ e $\mathcal{W}_1\mathcal{V}_1^T = \mathcal{V}_1\mathcal{W}_1^T = 0_{\frac{N}{2}}$ e da qual as linhas varrem o mesmo subespaço de $R^N$ como são varridas pelas últimas $N/2$ linhas da matriz DWT $\mathcal{W}$ ($\mathcal{V}_1$ tem a mesma estrutura que $\mathcal{W}_1$: só é necessário substituir cada $h_l^\circ$ por $g_l^\circ$, sendo $\{g_l^\circ\}$ o filtro de escala $\{g_l\}$ periodizado ao tamanho $N$, ou seja, $\{g_l^\circ\} \longleftrightarrow \{G(\frac{k}{N})\}$;

2. Os elementos do vetor $\mathbf{V}_1 = \mathcal{V}_1\mathbf{X}$ de tamanho $N/2$;

3. O componente liso do sinal de primeiro nível $\mathcal{S}_1 = \mathcal{V}_1^T \mathbf{V}_1 = \mathcal{V}_1^T \mathcal{V}_1 \mathbf{X}$, o qual pode ser formado pela sobreamostragem de $\mathbf{V}_1$ por dois e então filtrá-lo com $\{G^*(\frac{k}{N}\}$;

4. Médias de escala $\lambda_1 = 2$;

5. Subamostragem por dois do processo de meia-banda associado com as freqüências baixas [0,1/4] na série temporal $\{X_t\}$; e

6. A porção da DFT de $\{X_t\}$ com freqüências no intervalo [0,1/4].

A série $\{W_{1,t}\}$ constitui a primeira metade do vetor dos coeficientes *wavelet* $\mathbf{W}$; os demais coeficientes *wavelet* são obtidos a partir de $\{V_{1,t}\}$ nos estágios sucessivos do algoritmo da pirâmide.

#### 2.3.2.2 Segundo estágio do algoritmo da pirâmide

Conforme mencionado na Seção 2.3.2.1, o segundo estágio do algoritmo da pirâmide consiste em tratar $\{V_{1,t}\}$ da mesma maneira que $\{X_t\}$ foi tratado no primeiro estágio. Intuitivamente esse é um caminho razoável pois $\{X_t\}$ e $\{V_{1,t}\}$ são similares, já que $\{X_t\}$ pode ser considerado uma "média" na escala unitária enquanto os coeficientes de escala de primeiro nível podem ser considerados como uma média sobre a escala de dois. Assim, $\{V_{1,t}\}$ é filtrado circularmente e separadamente com $\{h_l\}$ e $\{g_l\}$, e a saída do filtro é subamostrada para produzir duas novas séries:

$$W_{2,t} = \sum_{l=0}^{L-1} h_l V_{1,2t+1-l mod \frac{N}{2}} \quad \text{e} \quad V_{2,t} = \sum_{l=0}^{L-1} g_l V_{1,2t+1-l mod \frac{N}{2}}, \tag{2.51}$$

t = 0,...,$\frac{N}{4}$ - 1. Nota-se que a equação acima tem o mesmo formato de filtro das equações 2.45 e 2.47 exceto pelo "mod $N$" que é substituído pelo "mod $\frac{N}{2}$"

As Figs. 2.13 e 2.14 ilustram a análise e síntese de X no segundo nível.

Assim, o segundo estágio do algoritmo da pirâmide decompõe o vetor $\mathcal{V}_1$ de tamanho $N/2$ em dois conjuntos de vetores de tamanho $N/4$, $\mathcal{W}_2$=$\mathcal{B}_2\mathcal{V}_1$=$\mathcal{B}_2\mathcal{A}_1$ e $\mathcal{V}_2$=$\mathcal{A}_2\mathcal{V}_1$=$\mathcal{A}_2\mathcal{A}_1$, sendo $\mathcal{A}_1$=$\mathcal{V}_1$. Cada linha de $\mathcal{B}_2$ contém os elementos do filtro *wavelet* $\{h_l\}$ periodizado ao tamanho $N/2$ (ou seja, a DFT inversa de $\{H(\frac{k}{\frac{N}{2}})$:$k$=0,...,$\frac{N}{2}$-1$\}$), com linhas adjacentes diferentes pelo deslocamento circular de duas unidades; da mesma forma, cada linha de $\mathcal{A}_2$ contém os elementos do filtro de escala $\{g_l\}$ periodizado ao tamanho $N/2$. Este estágio transforma $\{V_{1,t}$:$t$=0,...,$\frac{N}{2}$-1$\}$ em duas novas séries: $\{W_{2,t}$:$t$=0,...,$\frac{N}{4}$-1$\}$ e $\{V_{2,t}$:$t$=0,...,$\frac{N}{4}$-1$\}$.

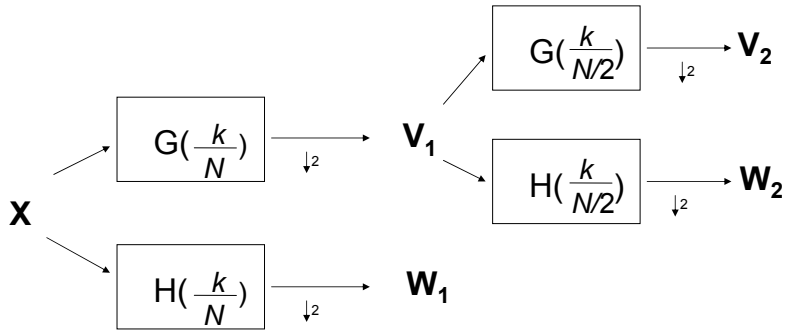



**Figura 2.13:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_1$, $\mathbf{W}_2$ e $\mathbf{V}_2$. Fonte: [4] página 89.

Assim como $\{X_t\}$ foi particionado ortogonalmente nas componentes passa-alta $\{W_{1,t}\}$ e passa-baixa $\{V_{1,t}\}$ pelos filtros *wavelet* e de escala, $\{V_{1,t}\}$ é particionado em $\{W_{2,t}\}$ e $\{V_{2,t}\}$; no entanto, a divisão de $\{V_{1,t}\}$ em passa-baixa e passa-alta corresponde à divisão da parte passa-baixa de $\{X_t\}$ em duas partes, uma correspondendo a $1/8 \leq |f| \leq 1/4$ (associada com os coeficientes *wavelet* de escala dois) e a outra a $0 \leq |f| \leq 1/8$.

Os coeficientes *wavelet* de segundo nível $\{W_{2,t}\}$ são associados com:

1. $\mathcal{W}_2 = \mathcal{B}_2\mathcal{A}_1$, matriz $\frac{N}{4} \times N$ que satisfaz $\mathcal{W}_2\mathcal{W}_2^T = I_{\frac{N}{4}}$ e compreende linhas $\frac{N}{2}$ a $\frac{3N}{4}$ - 1 da matriz DWT $\mathcal{W}$ (cada linha de $\mathcal{W}_2$ contém os elementos de $\{h^\circ_{2,l}\} \longleftrightarrow \{H(\frac{k}{N/2})\}\{G(\frac{k}{N})\}$, ou seja, o filtro *wavelet* de segundo nível $\{h_{2,l}\}$ periodizado ao tamanho $N$; a primeira linha é $\left[h^\circ_{2,3}, h^\circ_{2,2}, h^\circ_{2,1}, h^\circ_{2,0}, h^\circ_{2,N-1}, h^\circ_{2,N-2}, \ldots, h^\circ_{2,5}, h^\circ_{2,4}\right] = \mathcal{W}^T_{\frac{N}{2}\bullet}$, e as demais $\frac{N}{4}$-1 linhas são formadas pelo deslocamento circular da primeira linha para direita por, respectivamente, 4,8,12,...,$N$-4 unidades);

2. Os elementos do vetor $\mathbf{W}_2 = \mathcal{W}_2\mathbf{X}$ de tamanho $N/4$;

3. O detalhe de segundo nível $\mathcal{D}_2 = \mathcal{W}_2^T\mathbf{W}_2 = \mathcal{W}_2^T\mathcal{W}_2\mathbf{X}$;

4. Mudanças nas médias de escala $\tau_2 = 2$;

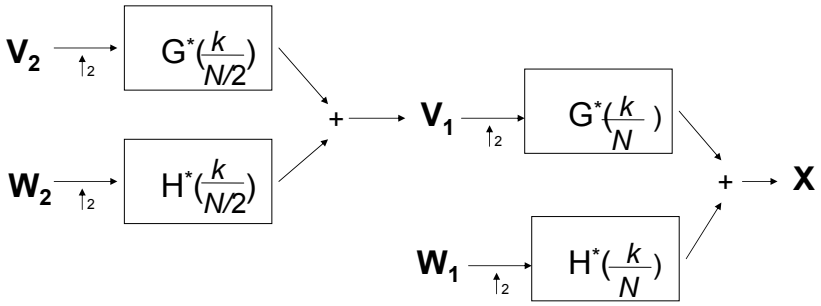



**Figura 2.14:** Diagrama de fluxo ilustrando a síntese de $\mathbf{X}$ a partir de $\mathbf{W}_1$, $\mathbf{W}_2$ e $\mathbf{V}_2$. Fonte: [4] página 90.

5. Subamostragem por dois do processo de meia-banda associado com as freqüências altas [1/4,1/2] nos coeficientes de escala de primeiro nível $\{V_{1,t}\}$; e

6. A porção da DFT de $\{X_t\}$ com freqüências no intervalo [1/8,1/4], com uma ordem reversa.

Os coeficientes de escala de segundo nível $\{V_{2,t}\}$ estão associados com:

1. $\mathcal{V}_2 = \mathcal{A}_2\mathcal{A}_1$, matriz $\frac{N}{4} \times N$ que satisfaz ambos $\mathcal{V}_2\mathcal{V}_2^T = I_{\frac{N}{4}}$ e $\mathcal{W}_2\mathcal{V}_2^T = \mathcal{V}_2\mathcal{W}_2^T = 0_{\frac{N}{4}}$ e da qual as linhas varrem o mesmo subespaço de $R^N$ como são varridas pelas últimas $N/4$ linhas da matriz DWT $\mathcal{W}$ ($\mathcal{V}_2$ tem a mesma estrutura que $\mathcal{W}_2$: só é necessário substituir cada $h^\circ_{2,l}$ por $g^\circ_{2,l}$, sendo $\{g^\circ_{2,l}\}$ o filtro de escala $\{g_{2,l}\}$ periodizado ao tamanho $N$, ou seja,
$\{g^\circ_{2,l}\} \longleftrightarrow \{G(\frac{k}{N/2})\}\{G(\frac{k}{N})\}$;

2. Os elementos do vetor $\mathbf{V}_2 = \mathcal{V}_2\mathbf{X}$ de tamanho $N/4$;

3. O componente liso do sinal de primeiro nível $\mathcal{S}_2 = \mathcal{V}_2^T\mathbf{V}_2 = \mathcal{V}_2^T\mathcal{V}_2\mathbf{X}$;

4. Médias de escala $\lambda_2 = 4$;

5. Subamostragem por dois do processo de meia-banda associado com as freqüências baixas [0,1/4] nos coeficientes de escala do primeiro nível $\{V_{1,t}\}$; e

6. A porção da DFT de $\{X_t\}$ com freqüências no intervalo [0,1/8].

#### 2.3.2.3 Estágio genérico do algoritmo da pirâmide

Conforme visto nas seções 2.3.2.1 e 2.3.2.2, pode-se definir o $j$-ésimo estágio do algoritmo da pirâmide, sendo $j = 1,\ldots,J$ (o tamanho da amostra N = $2^J$). Sendo $\{V_{0,t}\}$ definido como sendo o $\{X_t\}$, a entrada do $j$-ésimo estágio é $\{V_{j-1,t}{:}t{=}0,\ldots,N_{j-1}\text{-}1\}$, sendo $N_j \equiv N/2^j$. Esta entrada são os coeficientes de escala associados com médias sobre a escala $\lambda_{j-1} \equiv 2^{j-1}$. A saída do $j$-ésimo estágio são os coeficientes *wavelet* e de escala:

$$W_{j,t} \equiv \sum_{l=0}^{L-1} h_l V_{j-1,2t+1-l mod N_{j-1}} \quad \text{e} \quad V_{j,t} \equiv \sum_{l=0}^{L-1} g_l V_{j-1,2t+1-l mod N_{j-1}}, \tag{2.52}$$

t = 0,...,$N_{j-1}$ - 1.

A Fig. 2.15 ilustra a análise e síntese de $\mathbf{V}_{j-1}$.

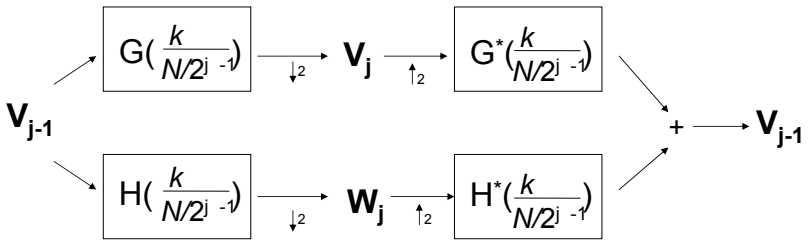



**Figura 2.15:** Diagrama de fluxo ilustrando a análise de $\mathbf{V}_{j-1}$ em $\mathbf{W}_j$, e $\mathbf{V}_j$ seguido pela síntese de $\mathbf{V}_{j-1}$ a partir de $\mathbf{W}_j$, e $\mathbf{V}_j$ - fonte: [4] página 94.

Assim, sendo o tamanho da amostra $N{=}2^J$ para um inteiro $J$, para $j = 1,\ldots,J$ com $N_j \equiv N/2^j$, o $j$-ésimo estágio do algoritmo da pirâmide decompõe os $N_{j-1}$ vetores linha na matriz $\mathcal{V}_{j-1}$ de tamanho $N_{j-1} \times N$ em dois conjuntos de vetores linha de tamanho $N_j$, $\mathcal{W}_j{=}\mathcal{B}_j\mathcal{V}_{j-1}$ e $\mathcal{V}_j{=}\mathcal{A}_j\mathcal{V}_{j-1}$, sendo $\mathcal{B}_j$ e $\mathcal{A}_j$ matrizes

$N_j \times N_{j-1}$ contendo, respectivamente, os filtros *wavelet* $\{h_l\}$ e de escala $\{g_l\}$ periodizados ao tamanho $N_{j-1}$ (aqui define-se que $\mathcal{V}_0 = I_N$). Os vetores linha $\mathcal{A}_j$ e $\mathcal{B}_j$ juntos formam um conjunto de $N_{j-1}$ vetores ortonormais. Com $V_{0,t} \equiv X_t$, a transformada do $j$-ésimo estágio transforma $\{V_{j-1,t}$:$t$=0,...,$N_{j-1}$-1$\}$ em duas novas séries: a dos coeficientes *wavelet* do $j$-ésimo nível $\{W_{j,t}$:$t$=0,...,$N_j$-1$\}$ e a dos coeficientes de escala do $j$-ésimo nível $\{V_{j,t}$:$t$=0,...,$N_j$-1$\}$.

Os coeficientes *wavelet* do $j$-ésimo nível $\{W_{j,t}\}$ são associados com:

1. $\mathcal{W}_j = \mathcal{B}_j\mathcal{A}_{j-1}\ldots\mathcal{A}_1$, matriz $N_j \times N$ que satisfaz $\mathcal{W}_j\mathcal{W}_j^T = I_{N_j}$ e compreende linhas $\sum_{k=1}^{j-1} N_k$ a $\sum_{k=1}^{j} N_k$ - 1 da matriz DWT $\mathcal{W}$ (a primeira dessas somas é considerada 0 quando $j$=1), com as linhas de $\mathcal{W}_j$ contendo versões de deslocamento circular de $\{h_{j,l}\}$ periodizado ao tamanho $N$, ou seja, $\{h^{\circ}_{j,l}\} \longleftrightarrow \{H(\frac{2^{j-1}k}{N})G(\frac{2^{j-2}k}{N})\ldots G(\frac{k}{N})\}$ (a seqüência $h^{\circ}_{j,l}$ aparece em ordem reversa nas linhas de $\mathcal{W}_j$, e as linhas adjacentes são formadas pelo deslocamento circular de $2^j$ unidades);

2. Os elementos do vetor $\mathbf{W}_j = \mathcal{W}_j\mathbf{X}$ de tamanho $N_j$;

3. O detalhe de $j$-ésimo nível $\mathcal{D}_j = \mathcal{W}_j^T\mathbf{W}_j = \mathcal{W}_j^T\mathcal{W}_j\mathbf{X}$;

4. Mudanças nas médias de escala $\tau_j = 2^{j-1}$;

5. Subamostragem por dois do processo de meia-banda associado com as freqüências altas [1/4,1/2] nos coeficientes de escala $\{V_{j-1,t}\}$; e

6. A porção da DFT de $\{X_t\}$ com freqüências no intervalo $[1/2^{j+1},1/2^j]$, com uma ordem reversa.

Os coeficientes de escala do $j$-ésimo nível $\{V_{j,t}\}$ estão associados com:

1. $\mathcal{V}_j = \mathcal{A}_j\mathcal{A}_{j-1}\ldots\mathcal{A}_1$, matriz $N_j \times N$ que satisfaz ambos $\mathcal{V}_j\mathcal{V}_j^T = I_{N_j}$ e $\mathcal{W}_j\mathcal{V}_j^T = \mathcal{V}_j\mathcal{W}_j^T = 0_{N_j}$ e da qual as linhas varrem o mesmo subespaço de $R^N$ como são varridas pelas últimas $N_j$ linhas da matriz DWT $\mathcal{W}$, com as linhas de $\mathcal{V}_j$ contendo a versão deslocada circularmente por $\{g_{j,l}\}$ periodizado ao tamanho $N$, ou seja, $\{g^{\circ}_{j,l}\} \longleftrightarrow \{G(\frac{2^{j-1}k}{N})G(\frac{2^{j-2}k}{N})\ldots G(\frac{k}{N})\}$ (a seqüência $g^{\circ}_{j,l}$ aparece em ordem reversa nas linhas de $\mathcal{V}_j$);

2. Os elementos do vetor $\mathbf{V}_j = \mathcal{V}_j\mathbf{X}$ de tamanho $N_j$;

3. O componente liso do sinal de $j$-ésimo nível $\mathcal{S}_j = \mathcal{V}_j^T\mathbf{V}_j = \mathcal{V}_j^T\mathcal{V}_j\mathbf{X}$;

4. Médias de escala $\lambda_j = 2^j$;

5. Subamostragem por dois do processo de meia-banda associado com as freqüências baixas [0,1/4] nos coeficientes de escala do nível $\{V_{j-1,t}\}$; e

6. A porção da DFT de $\{X_t\}$ com freqüências no intervalo $[0,1/2^{j+1}]$.

### 2.3.3 Maximal Overlap DWT - MODWT

A MODWT é uma versão modificada da DWT. Transformadas iguais ou similares à MODWT podem ser encontradas na literatura com os nomes de "*Undecimated DWT*", "*Translation invariant DWT*", "*shift invariant DWT*", "*wavelet frames*", "*stationary DWT*", "*time invariant DWT*", "*non-decimated DWT*" [4]. Esta variante da DWT procura utilizar os coeficientes que antes eram desprezados no cálculo da DWT. O resultado é uma transformada não mais ortogonal como a DWT, mas que ainda permite MRA, que trabalha com amostras de tamanho qualquer e cujos coeficientes são invariantes ao deslocamento do sinal de entrada. O preço pago é o esforço computacional maior que chega a $O(Nlog_2N)$, quando o tamanho da amostra é múltiplo de potências de dois.

A MODWT de nível $J_0$ para uma série temporal $\mathbf{X}$ é uma transformada não ortogonal com vetores coluna $\widetilde{\mathbf{W}}_1$, $\widetilde{\mathbf{W}}_2$, ..., $\widetilde{\mathbf{W}}_{J_0}$ e $\widetilde{\mathbf{V}}_{J_0}$ de dimensão $N$. O vetor $\widetilde{\mathbf{W}}_j$ contém os coeficientes *wavelet* MODWT associados a mudanças em $\mathbf{X}$ na escala de $\tau_j = 2^{j-1}$, enquanto $\widetilde{\mathbf{V}}_{J_0}$ contém os coeficientes de escala MODWT associados a variações na escala $\lambda_{J_0} = 2^{J_0}$ e acima. Assim como a DWT, a MODWT é definida em termos do esforço computacional do algoritmo da pirâmide.

De acordo com [4], as propriedades que distinguem a MODWT da DWT são:

1. Enquanto a DWT parcial de nível $J_0$ restringe o tamanho da amostra à um inteiro múltiplo de $2^{J_0}$, a MODWT de nível $J_0$ é bem definida para qualquer amostra de tamanho $N$. Quando $N$ for um múltiplo inteiro de $2^{J_0}$, a DWT parcial pode ser computada utilizando $O(N)$ multiplicações, no entanto, a MODWT correspondente requer $O(Nlog_2N)$ multiplicações. Sendo assim, há um custo computacional ao utilizar a MODWT, sendo que o esforço computacional é o mesmo que o amplamente utilizado no algoritmo da *fast Fourier transform* que é normalmente aceitável.

2. Assim como para a DWT, a MODWT pode ser utilizada para formar uma análise de multiresolução (MRA). A diferença é que os detalhes ($\widetilde{\mathcal{D}}_j$) e os componentes liso do sinal ($\widetilde{\mathcal{S}}_{J_0}$) da MRA do MODWT são tais que deslocamentos circulares na série temporal por uma determinada quantidade,

irá deslocar circularmente cada detalhe e componente liso pela quantidade correspondente.

3. Ao contrário da DWT, os detalhes e o componentes lisos do sinal da MODWT são associados com filtros de fase zero, sendo fácil assim, a associação de características na MRA com a série temporal original.

4. Assim como é verdade para a DWT, a MODWT pode ser utilizada para formar uma análise da variância (ANOVA) baseada nos coeficientes *wavelet* e de *escala*; ao contrário da DWT, os detalhes e componentes liso do sinal da MODWT não podem ser utilizados para formar tal análise.

5. Visto que uma série temporal e um deslocamento circular das séries podem ter diferentes potências espectrais empíricas baseadas na DWT, os espectros baseados na MODWT correspondente são os mesmos. De fato, pode-se obter a MODWT de uma série temporal circularmente deslocada pela simples aplicação de um deslocamento similar de cada um dos componentes $\widetilde{\mathbf{W}}_J$ e $\widetilde{\mathbf{V}}_{J_0}$ da MODWT da série original; ao contrário, a DWT de uma série obtida a partir do deslocamento circular não pode ser obtida por qualquer deslocamento circular dos componentes $\mathbf{W}_J$ e $\mathbf{V}_{J_0}$ da DWT da série original.

A fim de relacionar a DWT e a MODWT é conveniente definir o filtro *wavelet* MODWT $\{\widetilde{h}_l\}$ como $\widetilde{h}_l \equiv h_l/\sqrt{2}$ e o filtro de escala MODWT $\{\widetilde{g}_l\}$ como $\widetilde{g}_l \equiv g_l/\sqrt{2}$. Como $\{\widetilde{h}_l\}$ é simplesmente uma versão reescalada do filtro *wavelet*, então valem as seguintes condições provenientes das equações 2.41:

$$(i) \sum_{l=0}^{L-1} \widetilde{h}_l = 0 \quad \text{e} \quad (ii) \sum_{l=-\infty}^{\infty} \widetilde{h}_l \widetilde{h}_{l+2n} = \begin{cases} \frac{1}{2}, & n = 0 \\ 0, & n = \pm 1, \pm 2, \ldots \end{cases} \tag{2.53}$$

Similarmente, o filtro de escala MODWT tem as seguintes condições provenientes das equações 2.43:

$$(i) \sum_{l=0}^{L-1} \widetilde{g}_l = 1 \quad , \quad (ii) \sum_{l=-\infty}^{\infty} \widetilde{g}_l \widetilde{g}_{l+2n} = \begin{cases} \frac{1}{2}, & n = 0 \\ 0, & n = \pm 1, \pm 2, \ldots \end{cases} \quad \text{e} \quad (iii) \sum_{l=-\infty}^{\infty} \widetilde{g}_l \widetilde{h}_{l+2n'} = 0. \tag{2.54}$$

para todo $n$ inteiro dieferente de zero e para todo $n'$ inteiro.

A relação da equação 2.42 também vale para os filtros da MODWT:

$$\begin{aligned} \widetilde{g}_l &\equiv (-1)^{l+1}\widetilde{h}_{L-1-l} \\ \widetilde{h}_l &= (-1)^{l}\widetilde{g}_{L-1-l} \end{aligned} \tag{2.55}$$

Assim, pode-se escrever a partir das equações 2.44 e 2.46:

$$\widetilde{W}_{1,t} \equiv \sum_{l=0}^{L-1} \widetilde{h}_l X_{t-l mod N}, t = 0, \ldots, N-1; \quad \text{e} \quad \widetilde{V}_{1,t} \equiv \sum_{l=0}^{L-1} \widetilde{g}_l X_{t-l mod N}, t = 0, \ldots, N-1; \tag{2.56}$$

Então as seqüências $\{\widetilde{W}_{1,t}\}$ e $\{\widetilde{V}_{1,t}\}$ são obtidas pela filtragem circular de $\{X_t\}$ com, respectivamente, os filtros *wavelet* e de escala MODWT.

A motivação para formular a MODWT é essencialmente definir a transformada que se comporte o máximo possível como a DWT mas não sofra com a sensitividade da DWT com a escolha do ponto inicial da série temporal. Essa sensitividade está relacionada às subamostragens das saídas dos filtros *wavelet* e de escala em cada estágio do algoritmo da pirâmide. Para obter um grau de insensitividade em relação ao ponto inicial, é necessário eliminar essa subamostragem preservando a habilidade de realizar a ANOVA e a MRA.

O primeiro passo é notar que a saída do filtro que era normalmente descartada no primeiro estágio do algoritmo da pirâmide da DWT pode ser obtida ao aplicar o algoritmo da pirâmide da DWT no vetor deslocado circularmente $\Gamma\mathbf{X}$ ao invés de $\mathbf{X}$. Com isso, se utiliza o seguinte procedimento para definir o primeiro estágio do algoritmo da pirâmide para MODWT quando $N$ é uma amostra de tamanho par. A idéia é aplicar o algoritmo da pirâmide DWT duas vezes, uma para $\mathbf{X}$ e outra para $\Gamma\mathbf{X}$ e, depois, juntar os dois conjuntos de coeficientes DWT.

O vetor $\widetilde{\mathbf{W}}_1$ de dimensão $N$ contém os coeficientes *wavelet* MODWT de nível $j$ = 1. O $t$-ésimo elemento de $\widetilde{\mathbf{W}}_1$ é $\widetilde{W}_{1,t}$, $t$ = 0,1,. . .,N - 1 e é formado pela filtragem circular da série temporal $\{X_t\}$ com o filtro *wavelet* MODWT $\{\widetilde{h}_l \equiv h_l/\sqrt{2}\}$ conforme a equação 2.56; equivalentemente, pode-se considerar $\{\widetilde{W}_{1,t}\}$ como o resultado de filtrar circularmente $\{X_t\}$ com o filtro $\{\widetilde{h}_l^{\circ}\}$ o qual é $\{\widetilde{h}_l\}$ periodizado ao tamanho $N$. Similarmente, os coeficientes de escala $\widetilde{\mathbf{V}}_1$ são obtidos pela filtragem de $\{X_t\}$ com o filtro de escala MODWT $\{\widetilde{g}_l \equiv g_l/\sqrt{2}\}$; equivalentemente, pode-se considerar $\{\widetilde{V}_{1,t}\}$ como o resultado de filtrar $\{X_t\}$ com o filtro $\{\widetilde{g}_l^{\circ}\}$ o qual é $\{\widetilde{g}_l\}$ periodizado ao tamanho $N$. Cada linha da matriz $N \times N$ $\widetilde{\mathcal{B}}_1$ contém o filtro periodizado $\{\widetilde{h}_l^{\circ}\}$, com a diferença entre as linhas dada pelo deslocamento circular; similarmente, $\widetilde{\mathcal{A}}_1$ tem linhas cujos elementos são o filtro periodizado $\{\widetilde{g}_l^{\circ}\}$. Assim:

$$\|\mathbf{X}\| = \left\|\widetilde{\mathbf{W}}_1\right\|^2 + \left\|\widetilde{\mathbf{V}}_1\right\|^2 \quad \text{e} \quad \mathbf{X} = \widetilde{\mathcal{B}}_1^T\widetilde{\mathbf{W}}_1 + \widetilde{\mathcal{A}}_1^T\widetilde{\mathbf{V}}_1 \equiv \widetilde{\mathcal{D}}_1 + \widetilde{\mathcal{S}}_1 \tag{2.57}$$

fornecem uma decomposição de energia e uma decomposição aditiva em termos do primeiro nível do detalhe MODWT $\widetilde{\mathcal{D}}_1$ e o primeiro nível do componente liso do sinal $\widetilde{\mathcal{S}}_1$, respectivamente. O diagrama de fluxo da análise e síntese de $\mathbf{X}$ em termos dos coeficientes da MODWT é apresentado na Fig. 2.16.

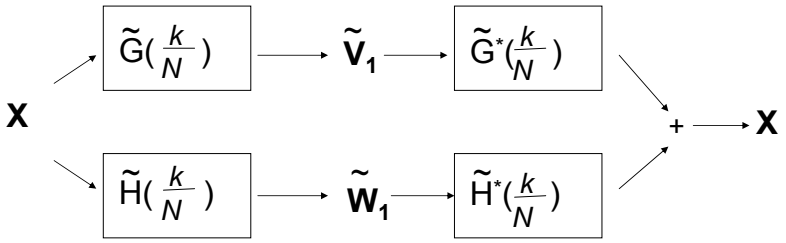



**Figura 2.16:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ nos coeficientes *wavelet* e de escala MODWT $\widetilde{\mathbf{W}}_1$ e $\widetilde{\mathbf{V}}_1$ de primeiro nível, seguido da síntese de $\mathbf{X}$ a parir de $\widetilde{\mathbf{W}}_1$ e $\widetilde{\mathbf{V}}_1$. Fonte: [4] página 169.

Para uma amostra de tamanho $N$, os coeficientes *wavelet* e de escala do $j$-ésimo nível da MODWT são definidos como os vetores $\widetilde{\mathbf{W}}_j$ e $\widetilde{\mathbf{V}}_j$ de dimensão $N$ cujos elementos são calculados por:

$$\widetilde{W}_{j,t} \equiv \sum_{l=0}^{L_j-1} \widetilde{h}_{j,l} X_{t-l mod N}, t = 0, \ldots, N-1; \quad \text{e} \quad \widetilde{V}_{j,t} \equiv \sum_{l=0}^{L_j-1} \widetilde{g}_{j,l} X_{t-l mod N}, t = 0, \ldots, N-1; \tag{2.58}$$

sendo $\widetilde{h}_{j,l} \equiv h_{j,l}/2^{j/2}$; $\widetilde{g}_{j,l} \equiv g_{j,l}/2^{j/2}$ e $L_j \equiv (2^j - 1)(L-1) + 1$. Os filtros $\{\widetilde{h}_{j,l}\}$ e $\{\widetilde{g}_{j,l}\}$ são chamados, respectivamente, de filtros *wavelet* e de escala do $j$-ésimo nível da MODWT. $L_j \equiv (2^j - 1)(L-1) + 1$

Em notação matricial:

$$\widetilde{\mathbf{W}}_j = \widetilde{\mathcal{W}}_j \mathbf{X} \quad \text{e} \quad \widetilde{\mathbf{V}}_j = \widetilde{\mathcal{V}}_j \mathbf{X} \tag{2.59}$$

sendo as linhas de $\widetilde{\mathbf{W}}_j$ versões deslocadas circularmente de $\{\widetilde{h}^{\circ}_{j,l}\}$ (ou seja, $\{\widetilde{h}_{j,l}\}$ periodizado ao tamanho $N$), enquanto as linhas de $\widetilde{\mathbf{V}}_j$ contêm versões deslocadas circularmente de $\{\widetilde{g}^{\circ}_{j,l}\}$.

### 2.3.4 *Discrete Wavelet Packet Transform* - DWPT

A DWT decompõe a série temporal $\mathbf{X}$ em coeficientes que podem ser associados com diferentes escalas e tempos. Assim, pode-se considerar a DWT de $\mathbf{X}$ como uma decomposição em tempo/escala. Os coeficientes *wavelet* para uma dada escala $\tau_j \equiv 2^{j-1}$ mostram como médias de $\mathbf{X}$ variam de um período para outro. A escala $\tau_j$ dá a largura no tempo (i.e. grau de localização) das médias. Como a DWT pode ser formulada através de filtros, pode-se relacionar a noção de escala a certas bandas de freqüências. O filtro equivalente que gera os coeficientes *wavelet* para escala $\tau_j$, é aproximadamente um filtro passa-faixa dado por $[1/2^{j+1},1/2^j]$. Para um tamanho de amostra $N = 2^J$, os coeficientes *wavelet* $N$ - 1 constituem uma decomposição da banda de uma oitava do intervalo de freqüência $[1/2^{J+1},1/2]$, enquanto o coeficiente de escala é associado ao intervalo $[0,1/2^{J+1}]$. Como um todo, os coeficientes DWT decompõem o intervalo de freqüência [0,1/2] em intervalos individuais adjacentes [4].

A DWPT pode ser considerada como uma coleção de transformadas ortonormais, sendo que cada uma delas pode ser rapidamente computada utilizando uma modificação muito simples do algoritmo da pirâmide para a DWT. Cada DWPT é associada com um nível $j$, e o $j$-ésimo nível da DWPT decompõe o intervalo de freqüência [0,1/2] em intervalos $2^j$ iguais e individuais. Devido à decomposição do nível $J$-1 dividir [0,1/2] em $N/2{=}2^{J-1}$ intervalos iguais, há uma DWPT que simula, mas não é a mesma, que a decomposição de [0,1/2] dada pela transformada discreta de Fourier (DFT). Quando $j = 1,...,J$-1, a DWPT resulta no que se pode chamar de decomposição tempo-freqüência porque cada coeficiente DWPT pode ser localizado numa banda de freqüência específica e num intervalo de tempo específico[18]. Também pode-se criar uma grande coleção de transformadas ortonormais através do agrupamento cuidadoso de bases vetoriais selecionadas de diferentes DWPTs. De fato, a DWT e todas DWTs parciais podem ser formadas utilizando bases vetoriais de diferentes DWPTs. Então, esse

[18]É similar ao espírito da chamada transformada de Fourier de tempo curto, a qual essencialmente é computada pela aplicação da DFT a sub-séries extraídas de $\mathbf{X}$.

esquema leva a uma coleção de transformadas flexível que serve como ponte entre decomposições tempo/escala e tempo/freqüência [4].

Conforme visto em 2.3.2, $\mathbf{W} = \mathcal{W}\mathbf{X}$ representa os coeficientes *wavelet* obtidos pela transformada de $\mathbf{X}$ utilizando a matriz DWT ortonormal $N \times N$ $\mathcal{W}$[19]. Na prática, a matriz $\mathcal{W}$ é gerada implicitamente pelo algoritmo da pirâmide. Conforme [4], o primeiro estágio do algoritmo pode ser descrito na notação matricial como:

$$\mathcal{P}_1\mathbf{X} = \begin{bmatrix} \mathcal{B}_1 \\ \mathcal{A}_1 \end{bmatrix} \mathbf{X} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{V}_1 \end{bmatrix} \equiv \begin{bmatrix} \mathbf{W}_{1,1} \\ \mathbf{W}_{1,0} \end{bmatrix} \tag{2.60}$$

sendo $\mathbf{W}_{1,1} \equiv \mathbf{W}_1$ e $\mathbf{W}_{1,0} \equiv \mathbf{V}_1$. A matriz $\mathcal{P}_1$ é ortonormal pois:

$$\mathcal{P}_1\mathcal{P}_1^T = \begin{bmatrix} \mathcal{B}_1 \\ \mathcal{A}_1 \end{bmatrix} \begin{bmatrix} \mathcal{B}_1^T, & \mathcal{A}_1^T \end{bmatrix} = \begin{bmatrix} \mathcal{B}_1\mathcal{B}_1^T & \mathcal{B}_1\mathcal{A}_1^T \\ \mathcal{A}_1\mathcal{B}_1^T & \mathcal{A}_1\mathcal{A}_1^T \end{bmatrix} = \begin{bmatrix} \mathcal{I}_{\frac{N}{2}} & 0_{\frac{N}{2}} \\ 0_{\frac{N}{2}} & \mathcal{I}_{\frac{N}{2}} \end{bmatrix} = \mathcal{I}_N \tag{2.61}$$

já que foram construídas as matrizes $\frac{N}{2} \times N$ $\mathcal{B}_1$ e $\mathcal{A}_1$ tais que:

$$\mathcal{B}_1\mathcal{B}_1^T = \mathcal{A}_1\mathcal{A}_1^T = \mathcal{I}_{\frac{N}{2}} \quad \text{e} \quad \mathcal{B}_1\mathcal{A}_1^T = \mathcal{A}_1\mathcal{B}_1^T = 0_{\frac{N}{2}} \tag{2.62}$$

sendo $\mathcal{I}_{\frac{N}{2}}$ a matriz identidade $\frac{N}{2} \times \frac{N}{2}$ enquanto $0_{\frac{N}{2}}$ representa a matriz $\frac{N}{2} \times \frac{N}{2}$ cujos elementos são zero.

Com um certo grau de aproximação, os coeficientes *wavelet* em $\mathbf{W}_{1,1}$ representam o conteúdo em freqüência de $\mathbf{X}$ para freqüências $f$ tais que $|f| \in [\frac{1}{4}, \frac{1}{2}]$, enquanto $\mathbf{W}_{1,0}$ representam freqüências $f$ tais que $|f| \in [0, \frac{1}{4}]$. A transformada $\mathcal{P}_1$ corresponde ao nível parcial da DWT $J_0 = 1$.

Similarmente, no final do segundo estágio do algoritmo da pirâmide, tem-se a transformada correspondente ao nível parcial da DWT $J_0 = 2$ que pode ser descrita como:

$$\begin{bmatrix} \mathcal{B}_1 \\ \mathcal{B}_2\mathcal{A}_1 \\ \mathcal{A}_2\mathcal{A}_1 \end{bmatrix} \mathbf{X} = \begin{bmatrix} \mathbf{W}_1 \\ \mathbf{W}_2 \\ \mathbf{V}_2 \end{bmatrix} \equiv \begin{bmatrix} \mathbf{W}_{1,1} \\ \mathbf{W}_{2,1} \\ \mathbf{W}_{2,0} \end{bmatrix} \tag{2.63}$$

sendo $\mathbf{W}_{1,1} \equiv \mathbf{W}_1$ como anteriormente, enquanto $\mathbf{W}_{2,1} \equiv \mathbf{W}_2$ e $\mathbf{W}_{2,0} \equiv \mathbf{V}_2$. Por comparação:

[19] Assume-se que $N = 2^J$ para qualquer inteiro $J$.

$$\mathcal{B}_2\mathcal{B}_2^T = \mathcal{A}_2\mathcal{A}_2^T = \mathcal{I}_{\frac{N}{4}} \quad \text{e} \quad \mathcal{B}_2\mathcal{A}_2^T = \mathcal{A}_2\mathcal{B}_2^T = 0_{\frac{N}{4}} \tag{2.64}$$

então, segue que a transformada é ortonormal pois:

$$\begin{aligned}
\begin{bmatrix} \mathcal{B}_1 \\ \mathcal{B}_2\mathcal{A}_1 \\ \mathcal{A}_2\mathcal{A}_1 \end{bmatrix} \begin{bmatrix} \mathcal{B}_1^T, & \mathcal{A}_1^T\mathcal{B}_2^T, & \mathcal{A}_1^T\mathcal{A}_2^T \end{bmatrix} &= \begin{bmatrix} \mathcal{B}_1\mathcal{B}_1^T & \mathcal{B}_1\mathcal{A}_1^T\mathcal{B}_2^T & \mathcal{B}_1\mathcal{A}_1^T\mathcal{A}_2^T \\ \mathcal{B}_2\mathcal{A}_1\mathcal{B}_1^T & \mathcal{B}_2\mathcal{A}_1\mathcal{A}_1^T\mathcal{B}_2^T & \mathcal{B}_2\mathcal{A}_1\mathcal{A}_1^T\mathcal{A}_2^T \\ \mathcal{A}_2\mathcal{A}_1\mathcal{B}_1^T & \mathcal{A}_2\mathcal{A}_1\mathcal{A}_1^T\mathcal{B}_2^T & \mathcal{A}_2\mathcal{A}_1\mathcal{A}_1^T\mathcal{A}_2^T \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{I}_{\frac{N}{2}} & 0_{\frac{N}{2}}\mathcal{B}_2^T & 0_{\frac{N}{2}}\mathcal{A}_2^T \\ \mathcal{B}_2 0_{\frac{N}{2}} & \mathcal{B}_2\mathcal{B}_2^T & \mathcal{B}_2\mathcal{A}_2^T \\ \mathcal{A}_2 0_{\frac{N}{2}} & \mathcal{A}_2\mathcal{B}_2^T & \mathcal{A}_2\mathcal{A}_2^T \end{bmatrix} \\
&= \begin{bmatrix} \mathcal{I}_{\frac{N}{2}} & 0_{\frac{N}{2},\frac{N}{4}} & 0_{\frac{N}{2},\frac{N}{4}} \\ 0_{\frac{N}{4},\frac{N}{2}} & \mathcal{I}_{\frac{N}{4}} & 0_{\frac{N}{4}} \\ 0_{\frac{N}{4},\frac{N}{2}} & 0_{\frac{N}{4}} & \mathcal{I}_{\frac{N}{4}} \end{bmatrix} = \mathcal{I}_N
\end{aligned} \tag{2.65}$$

sendo $0_{\frac{N}{2},\frac{N}{4}}$ uma matriz $\frac{N}{2} \times \frac{N}{4}$ de zeros.

No domínio da freqüência, $\mathbf{W}_{1,1}$ está relacionado ao intervalo de freqüência $[\frac{1}{4}, \frac{1}{2}]$; $\mathbf{W}_{2,1}$ ao intervalo $[\frac{1}{8}, \frac{1}{4}]$; e $\mathbf{W}_{2,0}$ ao intervalo $[0, \frac{1}{8}]$.

Combinando todos os resultados do segundo nível têm-se:

$$\begin{bmatrix} \mathbf{W}_{2,3} \\ \mathbf{W}_{2,2} \\ \mathbf{W}_{2,1} \\ \mathbf{W}_{2,0} \end{bmatrix} = \begin{bmatrix} \mathcal{A}_2\mathcal{B}_1 \\ \mathcal{B}_2\mathcal{B}_1 \\ \mathcal{B}_2\mathcal{A}_1 \\ \mathcal{A}_2\mathcal{A}_1 \end{bmatrix} \mathbf{X} \tag{2.66}$$

A ortonormalidade implica em:

$$\begin{aligned}
\mathbf{X} &= \begin{bmatrix} \mathcal{B}_1^T\mathcal{A}_2^T, & \mathcal{B}_1^T\mathcal{B}_2^T, & \mathcal{A}_1^T\mathcal{B}_2^T, & \mathcal{A}_1^T\mathcal{A}_2^T \end{bmatrix} \begin{bmatrix} \mathbf{W}_{2,3} \\ \mathbf{W}_{2,2} \\ \mathbf{W}_{2,1} \\ \mathbf{W}_{2,0} \end{bmatrix} \\
&= \mathcal{B}_1^T\mathcal{A}_2^T\mathbf{W}_{2,3} + \mathcal{B}_1^T\mathcal{B}_2^T\mathbf{W}_{2,2} + \mathcal{A}_1^T\mathcal{B}_2^T\mathbf{W}_{2,1} + \mathcal{A}_1^T\mathcal{A}_2^T\mathbf{W}_{2,0}
\end{aligned} \tag{2.67}$$

A decomposição de $\mathbf{X}$ que se assemelha a análise de multiresolução da DWT usual pode produzir uma análise de variância (ANOVA) similar à baseada na

DWT:

$$\|\mathbf{X}\|^2 = \|\mathbf{W}_{2,0}\|^2 + \|\mathbf{W}_{2,1}\|^2 + \|\mathbf{W}_{2,2}\|^2 + \|\mathbf{W}_{2,3}\|^2 \tag{2.68}$$

A Fig. 2.17 descreve a análise de $\mathbf{X}$ em $\mathbf{W}_{2,0}$, $\mathbf{W}_{2,1}$, $\mathbf{W}_{2,2}$, $\mathbf{W}_{2,3}$.

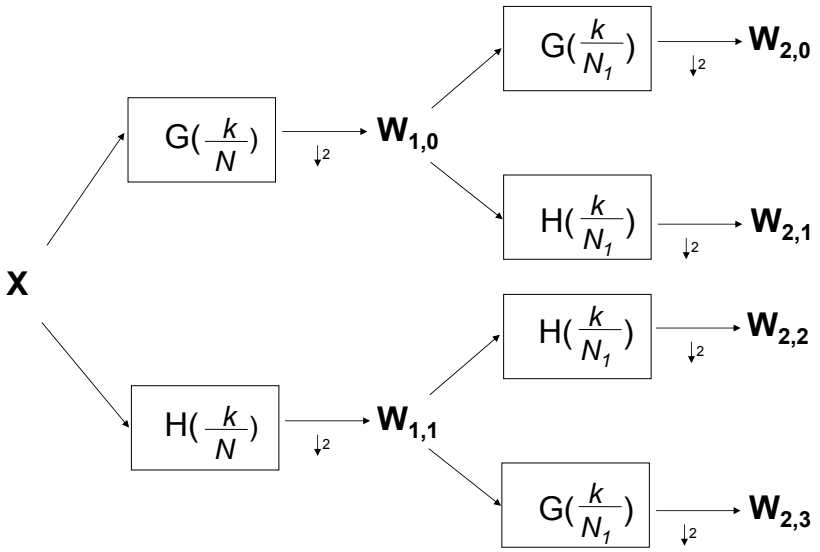



**Figura 2.17:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{2,0}$, $\mathbf{W}_{2,1}$, $\mathbf{W}_{2,2}$, $\mathbf{W}_{2,3}$ (ordem de seqüência). $N_1 \equiv N/2$. Fonte: [4] página 210.

A ordem de $\mathbf{W}_{2,n}$ na equação 2.66 e na Fig. 2.17 é particularmente útil na medida que $n$ corresponde à ordem das freqüências: $\mathbf{W}_{2,0}$ está relacionado ao intervalo de freqüência $[0,\frac{1}{8}]$; $\mathbf{W}_{2,1}$ ao intervalo de freqüência $[\frac{1}{8},\frac{1}{4}]$; $\mathbf{W}_{2,2}$ ao intervalo de freqüência $[\frac{1}{4},\frac{3}{8}]$; $\mathbf{W}_{2,3}$ ao intervalo de freqüência $[\frac{3}{8},\frac{1}{2}]$. Esta ordem é chamada de ordem de seqüência em [54]. Nota-se que os passos de filtragem da Fig. 2.17 são: passa-baixa, passa-alta, passa-alta e passa-baixa para $\mathbf{W}_{2,n}$, $n = 0,\ldots,3$, respectivamente. Os seguintes fatos importantes explicam esta ordem dos filtros:

- Por um lado, filtrar circularmente $\mathbf{X}$ com o filtro de escala $\{g_l\}$ e sub-amostrar por 2 leva a série $\mathbf{V}_1 = \mathbf{W}_{1,0}$ cujo conteúdo em freqüência sobre $f \in [0,\frac{1}{2}]$ está relacionado ao conteúdo em freqüência de $\mathbf{X}$ sobre $f \in [0,\frac{1}{4}]$;

- Por outro lado, filtrar circularmente $\mathbf{X}$ com o filtro *wavelet* $\{h_l\}$ e sub-amostrar por 2 leva a série $\mathbf{W}_1 = \mathbf{W}_{1,1}$ cujo conteúdo em freqüência sobre

$f \in [0,\frac{1}{2}]$ está relacionado ao conteúdo em freqüência de $\mathbf{X}$ sobre $f \in [\frac{1}{4},\frac{1}{2}]$ mas em ordem reversa;

A necessidade de reverter as operações dos filtros passa-alta e passa-baixa para obter a correspondência apropriada das freqüências em $\mathbf{W}_{2,j}$, $j = 0,\ldots,3$ é devido a essa ordem reversa das freqüências na representação de $\mathbf{W}_{1,1}$ quando comparada à representação de $\mathbf{X}$.

A Fig. 2.18 apresenta uma ordem alternativa das operações de filtro que devem ser mais intuitivas porque para cada divisão no diagrama de fluxo, o ramo de cima é resultante da operação do filtro passa-baixa $G(\cdot)$ e o ramo de baixo é resultante da operação do filtro passa-alta $H(\cdot)$. [54] se refere a esta ordem como ordem natural mas a ordem de freqüência não está diretamente refletida no segundo índice.

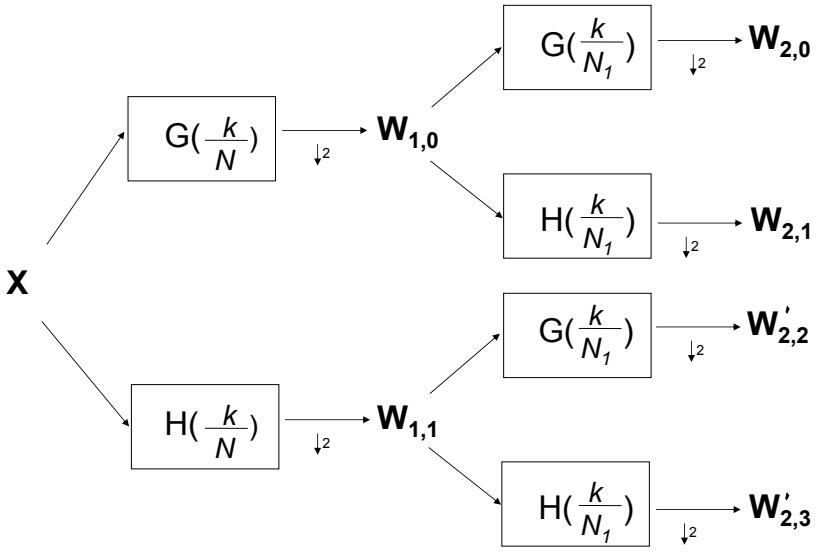



**Figura 2.18:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{2,0}$, $\mathbf{W}_{2,1}$, $\mathbf{W}'_{2,2}$, $\mathbf{W}'_{2,3}$ (ordem natural). $N_1 \equiv N/2$. Fonte: [4] página 210.

Os resultados acima podem ser generalizados seguindo as seguintes regras:

1. Se $n$ em $\mathbf{W}_{j-1,n}$ for par, o filtro passa-baixa $G(\cdot)$ deve ser utilizado para obter $\mathbf{W}_{j,2n}$ e o filtro passa-alta $H(\cdot)$ deve ser utilizado para obter $\mathbf{W}_{j,2n+1}$. Em notação matricial:

$$\mathbf{W}_{j,2n} = \mathcal{A}_j \mathbf{W}_{j-1,n} \quad \text{e} \quad \mathbf{W}_{j,2n+1} = \mathcal{B}_j \mathbf{W}_{j-1,n} \tag{2.69}$$

2. Se $n$ em $\mathbf{W}_{j-1,n}$ for ímpar, o filtro passa-alta $H(\cdot)$ deve ser utilizado para obter $\mathbf{W}_{j,2n}$ e o filtro passa-baixa $G(\cdot)$ deve ser utilizado para obter $\mathbf{W}_{j,2n+1}$. Em notação matricial:

$$\mathbf{W}_{j,2n} = \mathcal{B}_j \mathbf{W}_{j-1,n} \quad \text{e} \quad \mathbf{W}_{j,2n+1} = \mathcal{A}_j \mathbf{W}_{j-1,n} \tag{2.70}$$

Com essas regras é possível construir $2^j$ vetores no nível $j$, $\mathbf{W}_{j,n}$, $n = 0,\ldots,2^j - 1$. O vetor $\mathbf{W}_{j,n}$ é nominalmente associado com freqüências no intervalo $\mathcal{I}_{j,n} \equiv [\frac{n}{2^{j+1}}, \frac{n+1}{2^{j+1}}]$. Esta regra está ilustrada na Fig. 2.19 que é um exemplo de *wavelet packet table* ou *wavelet packet tree*. Já que o ponto inicial da análise é a série temporal $\mathbf{X}$, define-se que $\mathbf{W}_{0,0} \equiv \mathbf{X}$. Assim, $\mathbf{X}$ está associado com uma *doublet* $(j,n)$, no caso (0,0).

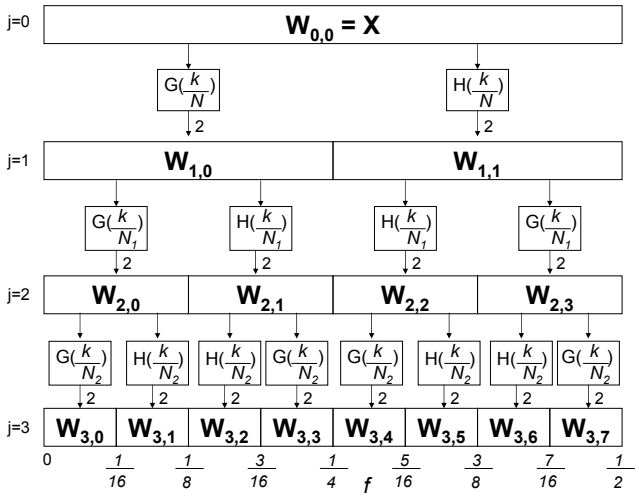



**Figura 2.19:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{3,0}$, ..., $\mathbf{W}_{3,7}$. $N_j \equiv N/2^j$. Fonte: [4] página 212.

A transformada que leva $\mathbf{X}$ a $\mathbf{W}_{j,n}$, $n = 0,\ldots,2^j$-1, para qualquer $j$ entre 0 e $J$ é chamada de *Discrete Wavelet Packet Transform* (DWPT). Esta transformada é ortonormal. Sendo assim, chama-se $\mathbf{W}_{j-1,n}$ de "pai" de seus $\mathbf{W}_{j,2n}$ e $\mathbf{W}_{j,2n+1}$ "filhos". Ao sair de uma DWPT de nível $j$-1 para uma de nível $j$, substitui-se cada "pai" por seus dois "filhos". Como descrito nas equações 2.69 e 2.70, a divisão de um nó "pai" é uma transformada ortonormal. Desde que isto seja verdadeiro para todo nó "pai" no nível $j$-1, a ortonormalidade da DWPT do nível $j$-1 implica na ortonormalidade da transformada do nível $j$.

A coleção de *doublets* $(j,n)$ formando os índices dos nós da tabela serão denotados por $\mathcal{N} \equiv \{(j,n) : j = 0,\ldots,J_0; n = 0,\ldots,2^j - 1\}$, sendo possível pegar qualquer $J_0$ que satisfaz $J_0 \leq J$ (como também é verdadeiro para DWTs parciais, se $J_0 < J$ pode-se relaxar a estipulação $N = 2^J$ e permitir que $N$ seja

qualquer inteiro múltiplo de $2^{J_0}$). As *doublets* $(j,n)$ que formam os índices dos coeficientes WP correspondentes a uma transformada ortonormal serão denotadas por $\mathcal{C}$; por exemplo, $\mathcal{C} = \{(3,n){:}n{=}0,\ldots,7\}$ para a transformada DWPT de nível $j = 3$ levando a $\mathbf{W}_{3,0}, \ldots, \mathbf{W}_{3,7}$. Claramente, $\mathcal{C} \subset \mathcal{N}$.

É interessante notar que além das DWPTs, é possível extrair outras transformadas ortonormais da tabela WP. Por exemplo, todas DWTs parciais podem ser retiradas dessa tabela. Isto é ilustrado pela Fig. 2.20 na qual a DWT parcial de nível $J_0 = 3$ consiste de quatro nós da tabela WP e são eles: $\mathcal{C} = \{(3,0),(3,1),(2,1),(1,1)\}$. Nota-se que o intervalo de freqüência para estes quatro nós formam uma partição completa de $[0,\frac{1}{2}]$. Mais genericamente, define-se a decomposição diática disjunta como aquela na qual em cada ponto de divisão (nó "pai" em potencial) na estrutura da tabela da Fig. 2.19, a divisão é feita tanto por filtros $G(\cdot)$ quanto filtros $H(\cdot)$. Tal decomposição também é ortonormal e está associada a uma partição completa de $[0,\frac{1}{2}]$. A Fig. 2.21 mostra uma decomposição diática disjunta arbitrária na qual $\mathcal{C}$ contém as *doublets* (2,0), (3,2), (3,3), e (1,1).

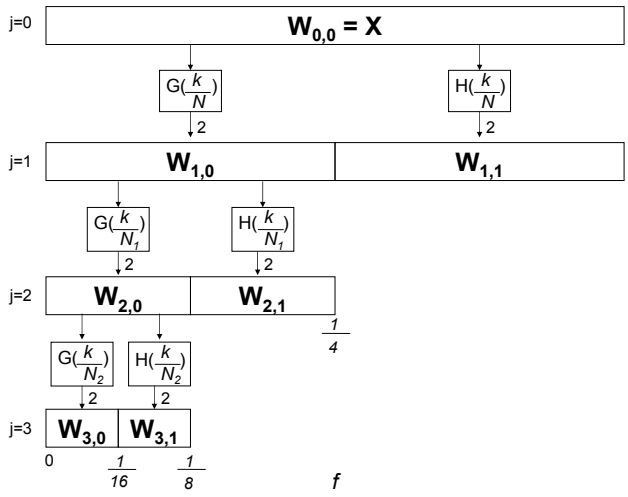



**Figura 2.20:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{3,0}$, $\mathbf{W}_{3,1}$, $\mathbf{W}_{2,1}$, $\mathbf{W}_{1,1}$ o qual é idêntico à DWT parcial de nível $J_0 = 3$. Fonte: [4] página 212.

A tabela WP foi completamente especificada quando foram selecionados filtros *wavelet* $\{h_l\}$ e seus respectivos filtros de escala $\{g_l\}$. Qualquer transformada ortonormal que possa ser extraída de uma tabela WP é especificada pelas *doublets* em $\mathcal{C}$. Para a transformada ortonormal da Fig. 2.21:

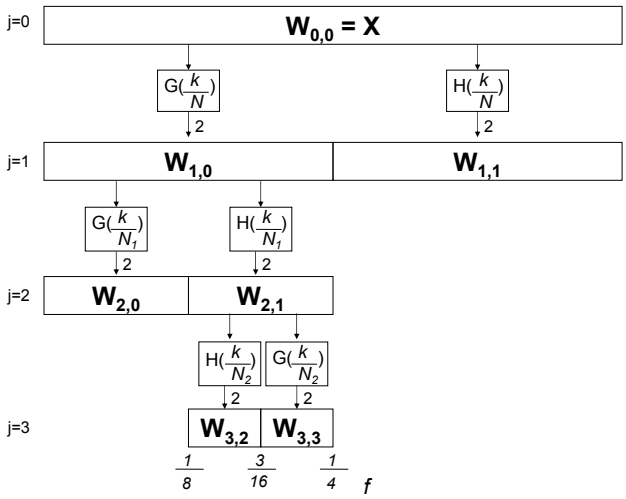



**Figura 2.21:** Diagrama de fluxo ilustrando a análise de $\mathbf{X}$ em $\mathbf{W}_{2,0}$, $\mathbf{W}_{3,2}$, $\mathbf{W}_{3,3}$, $\mathbf{W}_{1,1}$, uma decomposição diátia disjunta. Fonte: [4] página 213.

$$\begin{bmatrix} \mathbf{W}_{1,1} \\ \mathbf{W}_{3,3} \\ \mathbf{W}_{3,2} \\ \mathbf{W}_{2,0} \end{bmatrix} = \begin{bmatrix} \mathcal{B}_1 \\ \mathcal{A}_3\mathcal{B}_2\mathcal{A}_1 \\ \mathcal{B}_3\mathcal{B}_2\mathcal{A}_1 \\ \mathcal{A}_2\mathcal{A}_1 \end{bmatrix} \mathbf{X} \tag{2.71}$$

Devido à ortonormalidade desta transformada, têm-se uma decomposição aditiva:

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} \mathcal{B}_1^T, & \mathcal{A}_1^T\mathcal{B}_2^T\mathcal{A}_3^T, & \mathcal{A}_1^T\mathcal{B}_2^T\mathcal{B}_3^T, & \mathcal{A}_1^T\mathcal{A}_2^T \end{bmatrix} \begin{bmatrix} \mathbf{W}_{1,1} \\ \mathbf{W}_{3,3} \\ \mathbf{W}_{3,2} \\ \mathbf{W}_{2,0} \end{bmatrix} \\ &= \mathcal{B}_1^T\mathbf{W}_{1,1} + \mathcal{A}_1^T\mathcal{B}_2^T\mathcal{A}_3^T\mathbf{W}_{3,3} + \mathcal{A}_1^T\mathcal{B}_2^T\mathcal{B}_3^T\mathbf{W}_{3,2} + \mathcal{A}_1^T\mathcal{A}_2^T\mathbf{W}_{2,0} \end{aligned} \tag{2.72}$$

e a decomposição de energia (ANOVA):

$$\|\mathbf{X}\|^2 = \|\mathbf{W}_{1,1}\|^2 + \|\mathbf{W}_{3,3}\|^2 + \|\mathbf{W}_{3,2}\|^2 + \|\mathbf{W}_{2,0}\|^2 \tag{2.73}$$

Para computar os coeficientes DWPT para os níveis $j = 1,\ldots,J_0$, os coeficientes DWPT são recursivamente filtrados num nível anterior. Seja $W_{j-1,n,t}$ o $t$-ésimo elemento de $\mathbf{W}_{j-1,n}$, $n = 0,\ldots, 2^{j-1}$-1. Dado este vetor de tamanho $N_{j-1}$ e utilizando as regras das equações 2.69 e 2.70 são produzidos os elementos de $\mathbf{W}_{j,n}$:

$$W_{j,2n,t} \equiv \sum_{l=0}^{L-1} a_{n,l} W_{j-1,n,2t+1-l mod N_{j-1}},$$
$$W_{j,2n+1,t} \equiv \sum_{l=0}^{L-1} b_{n,l} W_{j-1,n,2t+1-l mod N_{j-1}}, \tag{2.74}$$

sendo t = 0,...,$N_j - 1$,

$$a_{n,l} = \begin{cases} \mathrm{g}_l, \text{ se } n \text{ for par;} \\ \mathrm{h}_l, \text{ se } n \text{ for ímpar,} \end{cases} \quad \text{e } b_{n,l} = \begin{cases} \mathrm{h}_l, \text{ se } n \text{ for par;} \\ \mathrm{g}_l, \text{ se } n \text{ for ímpar.} \end{cases}$$

A equação 2.74 pode ser escrita mais compactamente como:

$$W_{j,n,t} = \sum_{l=0}^{L-1} u_{n,l} W_{j-1,\lfloor \frac{n}{2} \rfloor,2t+1-l mod N_{j-1}}, \tag{2.75}$$

sendo t = 0,...,$N_j - 1$,

$$u_{n,l} = \begin{cases} \mathrm{g}_l, \text{ se } n \text{ mod } 4 = 0 \text{ ou } 3; \\ \mathrm{h}_l, \text{ se } n \text{ mod } 4 = 1 \text{ ou } 2, \end{cases} \quad \text{e } \lfloor \cdot \rfloor \text{ denota o operador de parte inteira.}$$

Também pode-se escrever $\mathbf{W}_{j,n}$ em termos da filtragem de $\mathbf{X}$ seguida de uma subamostragem apropriada [20]. Primeiramente, formula-se a forma correta do filtro $\{u_{j,n,l} : l = 0, \ldots, L_j - 1\}$ correspondente ao nó $(j,n)$ na tabela WP ($L_j \equiv$ ($2^j$ - 1)($L$ -1) + 1). Seja $u_{1,0,l} \equiv g_l$ e $u_{1,1,l} \equiv h_l$, para os nós $(j, n)$ com $j > 1$:

$$u_{j,n,l} \equiv \sum_{k=0}^{L-1} u_{n,k} u_{j-1,\lfloor \frac{n}{2} \rfloor,l-2^{j-1}k}, l = 0, , L_j - 1 \tag{2.76}$$

normalmente, define-se $u_{j,n,l} = 0$ para $l < 0$ e $l \geq L_j$. Por exemplo, para $j = 2$:

$u_{2,0,l} = \sum_{k=0}^{L-1} u_{0,k} u_{1,0,l-2k} = \sum_{k=0}^{L-1} g_k g_{l-2k}$

$u_{2,1,l} = \sum_{k=0}^{L-1} u_{1,k} u_{1,0,l-2k} = \sum_{k=0}^{L-1} h_k g_{l-2k}$

$u_{2,2,l} = \sum_{k=0}^{L-1} u_{2,k} u_{1,1,l-2k} = \sum_{k=0}^{L-1} h_k h_{l-2k}$

$u_{2,3,l} = \sum_{k=0}^{L-1} u_{3,k} u_{1,1,l-2k} = \sum_{k=0}^{L-1} g_k h_{l-2k}$

Então, para $j = 1,\ldots,J_0$, pode-se escrever os elementos de $\mathbf{W}_{j,n}$ em termos da filtragem de $\mathbf{X}$ com a subamostragem apropriada :

[20] O argumento para construir o filtro que leva $\mathbf{X}$ a $\mathbf{W}_{j,n}$ é similar ao apresentado na Seção 2.3.2 para DWT.

$$W_{j,n,t} = \sum_{l=0}^{L_j-1} u_{j,n,l} X_{2^j[t+1]-1-l mod N}, \tag{2.77}$$

sendo t = 0,1,. . .,$N_j - 1$.

Conforme visto anteriormente, cada $\mathbf{W}_{j,n}$ no $j$-ésimo nível da DWPT está nominalmente associado a uma largura de banda $1/2^{j+1}$, ou seja, a largura do seu intervalo de freqüência $\mathcal{I}_{j.n}$. Nota-se também que $\mathbf{W}_{j,0}$ é exatamente o mesmo vetor que $\mathbf{V}_j$ do coeficiente de escala do nível $j$ da DWT. Estes coeficientes de escala estão associados com a escala $\lambda_j = 2^j$. Então pode-se associar uma "largura de tempo" nominal de $\lambda_j$ com $\mathbf{W}_{j,0}$. Desde que os filtros equivalentes utilizados para criar todos $\mathbf{W}_{j,n}$ no nível $j$ tenham exatamente o mesmo tamanho $L_j$ e a mesma largura de banda nominal, pode-se associar a mesma "largura de tempo" $\lambda_j$ para cada $\mathbf{W}_{j,n}$ (nota-se que desde que hajam $N_j$ = N/$2^j$ = N/$\lambda_j$ coeficientes em cada $\mathbf{W}_{j,n}$, o tamanho nominal de $\lambda_j$ para cada coeficiente leva a uma cobertura no tempo total de $N_j \times \lambda_j = N$; ou seja, coletivamente os coeficientes em $\mathbf{W}_{j,n}$ cobrem todo o tempo sobre o qual $\mathbf{X}$ foi registrado como é intuitivamente razoável). Para $j = 0$, a "largura de tempo" é unitária e a largura de banda é 1/2 a qual é sensível dado que o nível $j = 0$ da DWPT é a transformada identidade. Considerando o outro extremo, ou seja, o nível $J$ da DWPT de uma série com tamanho $N = 2^J$, a "largura de tempo" é $N$, no entanto a largura de banda é $1/(2N)$. Para todos níveis $j$, o produto da "largura de tempo" pela largura de banda é uma constante igual a 1/2 para manter-se a "relação de reciprocidade" entre os domínios do tempo e freqüência [4].

### 2.3.5 Mapa de Variâncias

Nas seções 2.3.2 e 2.3.4 foi visto que a variância de uma série temporal pode ser decomposta nas escalas da DWT de acordo com (2.40) e nos níveis da DWPT por (2.73). Nesta seção a análise de variância baseada em *wavelet* é mais detalhada pela definição da variância *wavelet*, também conhecida como espectro *wavelet*.

A variância *wavelet* está relacionada com a densidade espectral de potência (DEP) da série temporal; assim sendo, é possível estimar o espectro da série a partir da variância *wavelet*.

Sendo $\{X_t\}$ um processo estacionário com DEP $S_X(f)$ definida na faixa de freqüência [-1/2,1/2] e média nula, segundo a propriedade fundamental da DEP, [30]:

$$\int_{-1/2}^{1/2} S_X(f)df = \sigma_X^2 \equiv varX_t \tag{2.78}$$

ou seja, a DEP decompõe a variância do processo pelas freqüências.

Por analogia a esse resultado fundamental, a variância *wavelet* é:

$$\sum_{j=1}^{\infty} \upsilon_X^2(\tau_j) = \sigma_X^2 \tag{2.79}$$

ou seja, a variância *wavelet* decompõe a variância do processo $\sigma_X^2$ pela escala $\tau_j$.

Nota-se que esta equação é análoga à equação do espectro de potência empírico da *wavelet* discreta $\{P_{\mathcal{W}}(\tau_j)\}$ definida em 2.40. Sendo assim, a soma das variâncias dos coeficientes *wavelet* das escalas de decomposição $j$ é igual à variância do processo.

Como visto na Seção 2.3.2, o coeficiente *wavelet* da escala $\tau_j$ é nominalmente associado a uma faixa de freqüência $[1/2^{j+1},1/2^j]$. Como $\upsilon_X^2(\tau_j)$ é a variância do coeficiente *wavelet* da escala $\tau_j$:

$$\upsilon_X^2(\tau_j) \approx 2\int_{1/2^{j+1}}^{1/2^j} S_X(f)df \tag{2.80}$$

Nota-se que o fator 2 que multiplica a integral acima é devido à DEP $S_X(.)$ ser uma função par de $f$ no intervalo [-1/2,1/2].

Pode-se estimar $S_X(.)$ com a função $\overline{S}_X(.)$ que é constante em cada intervalo $[1/2^{j+1},1/2^j]$ para $j$=1,...,$J$, ou seja, assume-se:

$\overline{S}_X(f) = C_j$ quando $\frac{1}{2^{j+1}} < f \leq \frac{1}{2^j}$,

sendo $C_j$ uma constante tal que:

$\int_{1/2^{j+1}}^{1/2^j} S_X(f)df = \int_{1/2^{j+1}}^{1/2^j} \overline{S}_X(f)df = \frac{C_j}{2^{j+1}}$

Utilizando a equação 2.80:

$$\upsilon_X^2(\tau_j) \approx \frac{C_j}{2^j} \tag{2.81}$$

Então pode-se utilizar $\hat{C}_j \equiv 2^j\hat{\upsilon}_X^2(\tau_j)$ para estimar os níveis da DEP.

Por analogia ao cálculo da variância dos coeficientes *wavelet* da DWT, encontra-se a variância dos coeficientes da DWPT a fim de obter um mapa de variâncias

que será utilizado na síntese de séries temporais.

Como visto na Seção 2.3.4 o coeficiente de cada nível $j$ é nominalmente associado a uma faixa de freqüência $[n/2^{j+1}, n+1/2^{j+1}]$. Como $v_X^2(j,n)$ é a variância do coeficiente $n$ do nível $j$:

$$v_X^2(j,n) \approx 2^j \int_{n/2^{j+1}}^{n+1/2^{j+1}} S_X(f)df \tag{2.82}$$

Como o objetivo deste trabalho é gerar séries temporais baseadas no modelo fGN, é necessário calcular o mapa de variâncias para os coeficientes DWPT que será utilizado na síntese de séries FGN via DWT e DWPT.

A DEP de um FGN pode ser aproximada por:

$$S_X(f) \propto f^{-\alpha} \tag{2.83}$$

Substituindo 2.83 em 2.82, obtém-se a expressão para variância dos coeficiente DWPT para um processo FGN:

$$\begin{aligned} v_X^2(j,n) &\approx 2^j \int_{n/2^{j+1}}^{n+1/2^{j+1}} f^{-\alpha} df \\ v_X^2(j,n) &\approx \frac{2^j}{1-\alpha}\left(\left(\frac{n+1}{2^{j+1}}\right)^{1-\alpha} - \left(\frac{n}{2^{j+1}}\right)^{1-\alpha}\right) \\ v_X^2(j,n) &\approx \frac{2^j}{1-\alpha}\left(\frac{1}{2^{j+1}}\right)^{1-\alpha}\left((n+1)^{1-\alpha} - n^{1-\alpha}\right) \end{aligned} \tag{2.84}$$

Como a DWT pode ser extraída da tabela WP, o mapa de variâncias da DWT também pode ser extraído do mapa de variâncias da DWPT. Os coeficientes *wavelet* das escalas $j$ da DWT são os coeficientes $n$=1 dos níveis $j$ da DWPT. Então, substituindo $n$=1 na expressão 2.84, obtém-se a expressão para a variância do coeficiente *wavelet* de cada escala $j$:

$$
\begin{aligned}
\upsilon_X^2(\tau_j) &= \upsilon_X^2(j,1) \\
\upsilon_X^2(\tau_j) &\approx \frac{2^j}{1-\alpha}\left(\frac{1}{2^{j+1}}\right)^{1-\alpha}\left(2^{1-\alpha}-1^{1-\alpha}\right) \\
\upsilon_X^2(\tau_j) &\approx \frac{1}{1-\alpha}\left(2^j\left(\frac{1}{2^{j(1-\alpha)}2^{1-\alpha}}\right)\left(2^{1-\alpha}-1\right)\right) \\
\upsilon_X^2(\tau_j) &\approx \frac{1}{1-\alpha}\left(2^j\left(\frac{1}{2^j 2^{-\alpha j}2^{1-\alpha}}\right)\left(2^{1-\alpha}-1\right)\right) \\
\upsilon_X^2(\tau_j) &\approx \frac{1}{1-\alpha}\left(\frac{1}{2^{-\alpha j}2^{1-\alpha}}\left(2^{1-\alpha}-1\right)\right) \qquad (2.85) \\
\upsilon_X^2(\tau_j) &\approx \frac{1}{1-\alpha}\left(\frac{1}{2^{-\alpha j}}-\frac{1}{2^{-\alpha j}2^{1-\alpha}}\right) \\
\upsilon_X^2(\tau_j) &\approx \frac{1}{1-\alpha}\frac{1}{2^{-\alpha j}}\left(1-\frac{1}{2^{1-\alpha}}\right) \\
\text{sendo} \quad \frac{1}{1-\alpha}\left(1-\frac{1}{2^{1-\alpha}}\right) &= C \quad \text{onde C é uma constante} \\
\upsilon_X^2(\tau_j) &\approx C * 2^{\alpha j}
\end{aligned}
$$

Pela expressão 2.85, observa-se que a variância dos coeficientes *wavelet* de um processo FGN é exponencialmente relacionada à escala $j$ como foi utilizado em [1] que se baseou nos trabalhos de Kaplan e Kuo [55] e de Flandrin [56] [57] que obtiveram o mesmo resultado.

A tabela 2.1 é um exemplo numérico de um mapa de variâncias DWPT para síntese de um FGN com 3 níveis ($J$=3) e parâmetro $\alpha$=0,9 normalizado pelo coeficiente *wavelet* da DWT no último nível a fim de que esse coeficiente tenha variância 1 na escala menos refinada $J$=3.

**Tabela 2.1:** Mapa de variâncias DWPT para síntese de um FGN com 3 níveis ($J$=3) e parâmetro $\alpha$=0,9, normalizado pelo coeficiente $\mathbf{W}_{3,1}$

| Nível | Coeficientes DWPT | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | $W_{0,0}$ | | | | | | | |
| | 2,1441 | | | | | | | |
| 1 | $W_{1,0}$ | | | | $W_{1,1}$ | | | |
| | 4,0011 | | | | 0,2872 | | | |
| 2 | $W_{2,0}$ | | $W_{2,1}$ | | $W_{2,2}$ | | $W_{2,3}$ | |
| | 7,4664 | | 0,5359 | | 0,3311 | | 0,2432 | |
| 3 | $W_{3,0}$ | $W_{3,1}$ | $W_{3,2}$ | $W_{3,3}$ | $W_{3,4}$ | $W_{3,5}$ | $W_{3,6}$ | $W_{3,7}$ |
| | 13,9327 | 1,0000 | 0,6179 | 0,4539 | 0,3611 | 0,3011 | 0,2589 | 0,2275 |

### 2.3.6 Considerações finais

Após o estudo das diferentes transformadas *wavelet* concluiu-se que a DWT seria uma melhor opção para geração de séries temporais por ser uma espécie de "subamostragem" da CWT o que minimiza sua excepcional redundância.

Após o estudo da DWT, verificou-se que a mesma exige um número de amostras sempre múltiplo de 2 e que há um número diferente de coeficientes para cada escala devido a subamostragem o que pode não ser muito interessante para geração de séries temporais para algumas aplicações. Para corrigir tais deficiências têm-se uma versão modificada da DWT, a MODWT que trabalha com amostras de qualquer tamanho e elimina a subamostragem utilizada na DWT.

Ao estudar a DWPT observou-se que esta transformada pode ser considerada uma coleção de transformadas ortonormais DWT. O que faz da transformada DWPT interessante para a geração de séries temporais neste trabalho é o fato de cada coeficiente DWPT poder ser localizado numa banda de freqüência específica e num intervalo de tempo especifico. A possibilidade de formar DWTs utilizando bases vetoriais de diferentes DWPTs leva a uma coleção de transformadas flexível servindo como ponte entre decomposições tempo/escala das DWTs e tempo/freqüência.

O estudo da variância *wavelet* levou à dedução da variância *wavelet packet* que junto com o conceito de *wavelet packet table* gerou o conceito e a definição do mapa de variâncias apresentado em 2.3.5.

# 3 Geração de Séries Temporais

O objetivo deste trabalho é gerar séries temporais Gaussianas com LRD e introduzir SRD em diferentes faixas de freqüência. Foram desenvolvidos dois métodos para geração de tais séries, o DWT com mapa de variâncias e o DWPT. Para introduzir o comportamento SRD às séries LRD geradas numa determinada faixa de freqüência, foi utilizado o conceito do mapa de variâncias, somando o ganho SRD à variância dos coeficientes referentes à tal faixa de freqüência. Também foi introduzido o comportamento SRD através da introdução de periodicidade às séries temporais geradas através da soma dos coeficientes *wavelet* da série a ser gerada com os coeficientes *wavelet* de uma série temporal referente a um coseno com a freqüência onde se pretendia introduzir o SRD. Essa periodicidade foi introduzida por ser observada na prática em [58]. Como apresentado em 2.3.4, a DWPT resulta no que se pode chamar de decomposição tempo-freqüência pois cada coeficiente DWPT pode ser localizado numa banda de freqüência específica e num intervalo de tempo específico. Assim, foi verificado que a DWPT foi o método mais eficaz desenvolvido deste trabalho por oferecer maior flexibilidade em relação às freqüências.

## 3.1 Síntese de Séries Temporais via *Discrete Wavelet Transform* (DWT)

Na Seção 2.3.2, a Transformada *Wavelet* Discreta (DWT) foi apresentada de duas formas: por meio de matrizes ortogonais e por meio de filtros *wavelet* (passa-alta) e de escala (passa-baixa).

Foi apresentado o algoritmo da pirâmide de Mallat para a decomposição da série nos coeficientes da DWT (análise) e respectiva reconstrução da série a partir destes coeficientes (síntese). Este algoritmo é computacionalmente mais eficiente que a multiplicação direta de matrizes.

Por meio das equações de decomposição e de reconstrução do algoritmo de

Mallat verificou-se que a análise via DWT define uma verdadeira análise de multiresolução (MRA) da série temporal $\mathbf{X}$, decompondo o vetor temporal da série em $J_0$ somas de vetores $\mathcal{D}_j$ (chamado de detalhe pois contém as componentes de alta freqüência - coeficientes *wavelet*) mais o vetor $\mathcal{S}_{J_0}$ (chamado de aproximação por possuir baixas freqüências - coeficiente de escala).

A MRA é implementada no algoritmo da pirâmide por bancos de filtros passa-alta e passa-baixa adequadamente posicionados para separar recursivamente os sinais de detalhe (coeficientes *wavelet*) e de aproximação (coeficientes de escala) respectivamente. Como os coeficientes dos filtros *wavelet* e de escala são reais para a *wavelet* de Haar utilizada neste trabalho, a relação entre os coeficientes de um filtro de reconstrução e os coeficientes do respectivo filtro de decomposição é simples, bastando inverter a ordem dos coeficientes para passar de um filtro a outro (função *fliplr* do *software Matlab*) [1].

Sendo as seqüências $\{g_n\}$ para os coeficientes do filtro de escala de reconstrução (passa-baixa) e $\{h_n\}$ para os coeficientes do filtro *wavelet* de reconstrução (passa-alta), pode-se obter as relações de análise que indicam como os coeficientes de escala e os coeficientes *wavelet* de um sinal em uma escala $j$ são calculados a partir dos coeficientes da escala $j$ - 1 [30]:

$$u_{j,k} = \sum_{n=0}^{2N-1} g_n u_{j-1,n+2k} \;\text{ e }\; w_{j,k} = \sum_{n=0}^{2N-1} h_n u_{j-1,n+2k}\,, \tag{3.1}$$

em que N é o número de momentos nulos do filtro *wavelet* adotado.

Os coeficientes $\{g_n\}$ e $\{h_n\}$ dos filtros são chamados de seqüências geradoras por [59] e apresentam $2N$ elementos não nulos (N=1 para *wavelet* de Haar). $\{g_n\}$ e $\{h_n\}$ estão tabelados [4, pág.109], sendo que a seqüência $\{g_n\}$ pode ser obtida através da função *dbaux* do *Matlab* para as *wavelets* de Daubechies [1].

Percival [4] apresenta a relação (3.2) que permite calcular os coeficientes do filtro *wavelet* $\{h_n\}$, quando são conhecidos os coeficientes do filtro de escala:

$$h_n = (-1)^n g_{2N-1-n}\,. \tag{3.2}$$

Bäckar [30] também apresenta a *equação de reconstrução* (3.3), necessária para obter os coeficientes de aproximação em uma escala mais refinada a partir de uma escala de menor resolução:

$$u_{j,k} = \sum_{l=\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor - N+1} g_{k-2l} u_{j+1,l} + h_{k-2l} w_{j+1,l} \,. \tag{3.3}$$

A primeira somatória de (3.3) pode ser calculada utilizando-se a operação de convolução entre a série $\{g_n\}$ e uma série obtida pela intercalação de zeros nos coeficientes de aproximação apropriados (*upsampling*). Similarmente, na segunda somatória efetua-se a convolução entre a série $\{h_n\}$ e os coeficientes *wavelet* apropriados após *upsampling*. No *software Matlab* utiliza-se a função *conv* para efetuar a operação de convolução e a função *dyadup* para a operação *upsampling*.

O diagrama da Fig. 3.1 [1] mostra como as operações de *upsampling* e de convolução dos coeficientes de escala e *wavelet* com os coeficientes dos filtros passa-baixas e passa-altas são recursivamente combinados para se obter coeficientes de aproximação em escalas mais refinadas. Esse diagrama de reconstrução equivale à *Inverse Discrete Wavelet Transform* (IDWT) aliada à teoria MRA, implementado através do algoritmo da pirâmide de Mallat [1].

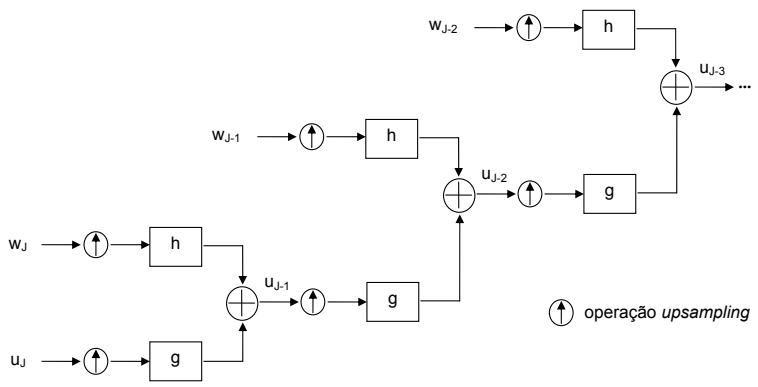



**Figura 3.1:** Algoritmo de reconstrução baseado na IDWT

Tanto o sinal de detalhe quanto o sinal de aproximação de uma determinada escala podem ser reconstruídos a partir dos respectivos coeficientes *wavelet* ou de aproximação passando-se esses coeficientes através do banco de filtros do algoritmo da pirâmide, porém mantendo-se os valores dos demais coeficientes em zero. A adição de todos os sinais de detalhe mais o sinal de aproximação menos refinado forma o sinal original ou sinal sintetizado.

[1] para maior clareza a Fig.2.10 está repetida para adaptar a notação

### 3.1.1 Geração de Séries Temporais Gaussianas com LRD pelo método DWT de Bäckar

Pode-se resumir o procedimento de geração de séries temporais gaussianas com LRD pelo método DWT de Bäckar em [1] através dos seguintes passos:

1. Determina-se o número total de escalas $J+1$, em que $j = J$ será a escala menos refinada e $j = 0$ será a escala mais refinada;
2. Determina-se o número total de pontos $M$ desejados ao final da síntese;
3. Computam-se os coeficientes dos filtros $\{g_n\}$ (*dbaux*) e $\{h_n\}$ (3.2) dependendo da *wavelet* escolhida;
4. Calcula-se quantos pontos os coeficientes de cada escala $j$ deverá ter;
5. Determinam-se os coeficientes de aproximação na escala menos refinada $j = J$ como zero;
6. Determinam-se os coeficientes *wavelet* com distribuição Gaussiana independente e identicamente distribuída de média 0 e variância 1,0 na escala menos refinada $j = J$. A variância dos coeficientes *wavelet* decai de uma escala para outra pelo fator $2^{\alpha}$;
7. Efetuam-se as operações de *upsampling* (*dyadup*) e convolução (*conv*);
8. Adiciona-se as séries dos coeficientes *wavelet* e de escala da escala $j = J$ após *upsampling* e convolução para obter os coeficientes de aproximação na escala $j = J - 1$;
9. Repete-se os 4 últimos passos, substituindo-se $j$ por $j - 1$ até que os coeficientes de aproximação na escala $j = 0$ sejam obtidos.

Mello [1] utilizou $2^{\alpha}$ como fator de decaimento da variância dos coeficientes *wavelet* de uma escala menos refinada $j$ para uma escala mais refinada $j$-1 baseado no trabalho de Kaplan e Kuo [55] que mostrou que a aplicação da DWT com a base de Haar ao FGN produz coeficientes *wavelet* pouco correlacionados, com uma variância que é exponencialmente relacionada à escala:

$$\mathrm{Var}\{w_{j,k}\} \propto 2^{j\alpha}\,. \tag{3.4}$$

Mello [1] implementou o gerador de tráfego como feito em [30] seguindo a prescrição de Kaplan e Kuo e utilizando o algoritmo da pirâmide. Assim as séries dependem:

- do filtro *wavelet*;
- do número de escalas da MRA;
- do processo estocástico que gera a realização $\{u_J(k)\}$;
- do processo estocástico que gera as realizações $\{w_j(k)\}_{j=J,J-1,\ldots,1}$.

Portanto, para garantir que as séries geradas correspondam a realizações de processos com DEP $1/f$, o processo estocástico que controla a criação dos coeficientes *wavelet* $\{w_j(k)\}_{j=J,J-1,\ldots,1}$ deve ser escolhido de modo que $\eta_j = \frac{Var[w_j]}{Var[w_{j-1}]} = 2^{\alpha}$ , $j \leq J$. Ou seja, o decaimento da energia dos coeficientes *wavelet* entre escalas consecutivas é constante.

Neste trabalho foi verificado o por que da utilização de $2^{\alpha}$ como fator de decaimento da variância dos coeficientes *wavelet* de uma escala para outra por meio do mapa de variâncias na Seção 2.3.5.

A Fig. 3.2 mostra uma realização $1/f$ com $\alpha = 0,9$ ($H = 0,95$), $N = 4096$, Nr. *Vanishing Moment* = 1 (filtro de Haar) e número de escalas $J = 12$ que foi gerada pelo algoritmo da pirâmide implementado em [1].

### 3.1.2 Geração de Séries Temporais Gaussianas com LRD pelo método DWT com mapa de variâncias

Para que seja possível a introdução de SRD em séries temporais geradas via DWT, o código desenvolvido em [1] foi alterado para utilizar a variância do coeficiente *wavelet* de um mapa de variâncias previamente calculado. Neste mapa, são calculadas as variâncias dos coeficientes do nível menos refinado ($j = J+1$) de uma *wavelet packet table* (2.3.4) e normalizadas pela variância do coeficiente *wavelet* da DWT (coeficiente $\mathbf{W}_{J+1,2}$). As variâncias dos coeficientes dos demais níveis ($j$) são calculadas a partir das variâncias dos coeficientes do nível ($j+1$). Após o cálculo do mapa de variâncias completo, é introduzida a SRD nas variâncias dos coeficientes do nível menos refinado do mapa referentes ao intervalo de freqüência requerido gerando um mapa de variâncias com SRD.

Este código gera ao mesmo tempo séries temporais gaussianas com LRD e tais séries com a introdução de SRD no intervalo de freqüência solicitado.

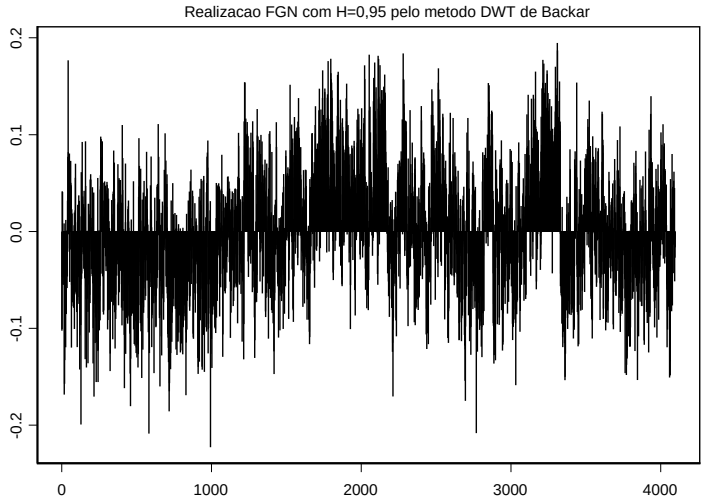



**Figura 3.2:** Realização de um processo $1/f$ obtida através do método DWT de Bäckar implementado em [1] utilizando função de Haar, sorteio IID Gaussiano dos coeficientes *wavelet* e escolha de $\alpha = 0{,}9$ (H = 0,95) para controle de variância das variáveis aleatórias Gaussianas.

As características que controlam a geração de séries temporais Gaussianas com LRD e densidade espectral de potência (DEP) $1/f$ com introdução de SRD pelo método DWT com mapa de variâncias são:

- O número de escalas que serão utilizadas para síntese da série $J$ ($J+1$ é a escala menos refinada e 1 a mais refinada) ;
- O número de momentos nulos que determina qual filtro *wavelet* será utilizado (Haar, Daubechies de ordem N);
- Número do coeficiente do nível menos refinado referente ao início do intervalo de freqüência a ser introduzida a SRD;
- Número do coeficiente do nível menos refinado referente ao fim do intervalo de freqüência a ser introduzida a SRD;
- Número de pontos que a série gerada deverá ter;
- $\alpha$ - parâmetro que controla o decaimento de energia dos coeficientes *wavelet* utilizado no cálculo das variâncias dos coeficientes do nível menos refinado do mapa de variâncias;
- Coeficiente de aproximação na escala menos refinada configurado para o valor zero;

- Coeficiente *wavelet* com distribuição Gaussiana independente e identicamente distribuída de média 0 e variância 1,0 na escala $J + 1$.

Pode-se resumir o procedimento de geração de séries temporais gaussianas com LRD com introdução de SRD pelo método DWT com mapa de variâncias através dos seguintes passos:

1. Determina-se o número total de escalas $J + 1$, em que $j = J + 1$ será a escala menos refinada e $j = 1$ será a escala mais refinada (série temporal gerada);
2. Determina-se o número total de pontos $M$ desejados ao final da síntese;
3. Computam-se os coeficientes dos filtros $\{g_n\}$ (*dbaux*) e $\{h_n\}$ (3.2) dependendo da *wavelet* escolhida;
4. Calcula-se quantos pontos os coeficientes de cada escala $j$ deverá ter;
5. Calcula-se o mapa de variâncias utilizando $\alpha$ como parâmetro para o cálculo da variância dos coeficientes do nível menos refinado; Tais variâncias são normalizadas pela variância do coeficiente *wavelet* da DWT (coeficiente $\mathbf{W}_{J+1,2}$) para obter uma distribuição Gaussiana independente e identicamente distribuída de média 0 e variância 1,0 na escala menos refinada $j = J + 1$;
6. Calcula-se o mapa de variâncias SRD, com a introdução de SRD nas variâncias dos coeficientes do nível menos refinado que estejam entre os coeficientes passados referentes ao início e fim da faixa de freqüência a ser introduzida a SRD.
7. Determinam-se os coeficientes de aproximação na escala menos refinada $j = J + 1$ como zero;
8. Os coeficientes *wavelet* dos níveis são obtidos através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que o coeficiente *wavelet* deverá ter naquele nível;
9. Efetuam-se as operações de *upsampling* (*dyadup*) e convolução (*conv*);
10. Adiciona-se as séries dos coeficientes *wavelet* e de escala da escala $j = J+1$ após *upsampling* e convolução para obter os coeficientes de aproximação na escala $j = (J + 1) - 1$;

11. Repete-se os 4 últimos passos, substituindo-se $j$ por $j-1$ até que os coeficientes de aproximação na escala $j = 1$ (série temporal) sejam obtidos.

A Fig. 3.3 mostra uma realização $1/f$ sem a introdução de SRD com $\alpha = 0,9$ ($H = 0,95$), $N = 4096$, Nr. *Vanishing Moment* $= 1$ (filtro de Haar) e número de escalas $J = 12$ que foi gerada pelo algoritmo da pirâmide implementado utilizando o mapa de variância dos coeficientes *wavelet* da DWT. Nota-se que essa realização é visualmente parecida com a série gerada pelo método da DWT de Bäckar da Fig. 3.2 gerada por [1].

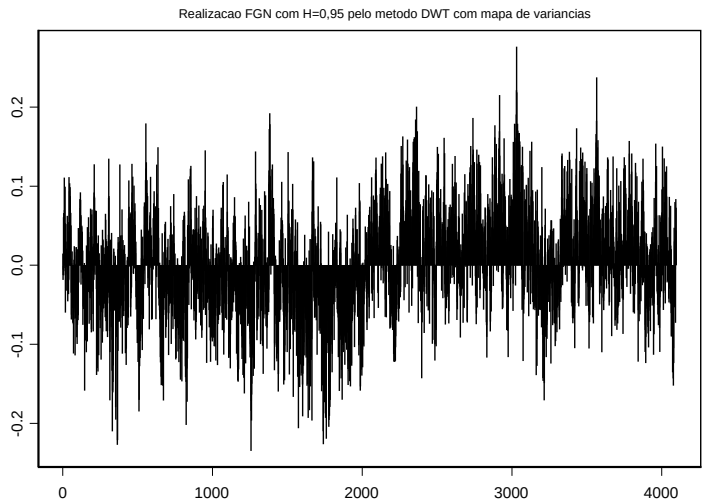



**Figura 3.3:** Realização de um processo $1/f$ de 4096 pontos obtida através do método DWT com mapa de variâncias utilizando função de Haar, sorteio IID Gaussiano dos coeficientes *wavelet* e escolha de $\alpha$ = 0,9 (H = 0,95) para inicialização do cálculo do mapa de variância.

### 3.1.3 Geração de Séries Temporais Gaussinas com LRD com periodicidade pelo método DWT com mapa de variâncias

A periodicidade foi introduzida na geração de séries temporais gaussinas com LRD pelo método DWT com mapa de variâncias assim caracterizando a série como uma série mista (LRD e SRD).

As características que controlam a geração de séries temporais Gaussianas com LRD com periodicidade pelo método DWT com mapa de variâncias:

- Coeficientes DWT da série referente ao sinal do cosseno com amplitude, freqüência e número de pontos desejados;

- O número de escalas que serão utilizadas para síntese da série $J$ ($J+1$ é a escala menos refinada e 1 a mais refinada) ;
- O número de momentos nulos que determina qual filtro *wavelet* será utilizado (Haar, Daubechies de ordem N);
- Número de pontos que a série gerada deverá ter;
- $\alpha$ - parâmetro que controla o decaimento de energia dos coeficientes *wavelet* utilizado no cálculo das variâncias dos coeficientes do nível menos refinado do mapa de variâncias;
- Coeficiente de aproximação na escala menos refinada configurado para o valor zero;
- Coeficiente *wavelet* com distribuição Gaussiana independente e identicamente distribuída de média 0 e variância 1,0 na escala $J+1$ somado ao coeficiente *wavelet* equivalente da série do sinal cosseno utilizado para introduzir periodicidade à série.

Pode-se resumir o procedimento de geração de séries temporais gaussianas com LRD com periodicidade pelo método DWT com mapa de variâncias através dos seguintes passos:

1. Gera-se um sinal cosseno com amplitude e freqüência desejadas e com o mesmo número de pontos que a série que será introduzida a periodicidade;
2. Calculam-se os coeficientes *wavelet* da série gerada do sinal cosseno com o código de análise DWT;
3. Determina-se o número total de escalas $J+1$, em que $j = J+1$ será a escala menos refinada e $j = 1$ será a escala mais refinada (série temporal gerada);
4. Determina-se o número total de pontos $M$ desejados ao final da síntese;
5. Computam-se os coeficientes dos filtros $\{g_n\}$ (*dbaux*) e $\{h_n\}$ (3.2) dependendo da *wavelet* escolhida;
6. Calcula-se quantos pontos os coeficientes de cada escala $j$ deverá ter;
7. Calcula-se o mapa de variâncias utilizando $\alpha$ como parâmetro para o cálculo da variância dos coeficientes do nível menos refinado; Tais variâncias são normalizadas pela variância do coeficiente *wavelet* da DWT (coeficiente

$\mathbf{W}_{J+1,2}$) para obter uma distribuição Gaussiana independente e identicamente distribuída de média 0 e variância 1,0 na escala menos refinada $j = J + 1$;

8. Determinam-se os coeficientes de aproximação na escala menos refinada $j = J + 1$ como zero;

9. Carrega-se o arquivo com os coeficientes *wavelet* da série do sinal cosseno;

10. Os coeficientes *wavelet* dos níveis são obtidos através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que o coeficiente *wavelet* deverá ter naquele nível somado ao coeficiente *wavelet* equivalente da série do cosseno utilizado para introduzir periodicidade à série. ;

11. Efetuam-se as operações de *upsampling* (*dyadup*) e convolução (*conv*);

12. Adiciona-se as séries dos coeficientes *wavelet* e de escala da escala $j = J+1$ após *upsampling* e convolução para obter os coeficientes de aproximação na escala $j = (J + 1) - 1$;

13. Repete-se os 4 últimos passos, substituindo-se $j$ por $j - 1$ até que os coeficientes de aproximação na escala $j = 1$ (série temporal) sejam obtidos.

A Fig. 3.4 mostra uma realização $1/f$ com periodicidade de amplitude 0,008, $\alpha = 0,9$ ($H = 0,95$), $N = 4096$, Nr. *Vanishing Moment* $= 1$ (filtro de Haar) e número de escalas $J = 12$ que foi gerada pelo algoritmo da pirâmide implementado utilizando o mapa de variância dos coeficientes *wavelet* da DWT.

## 3.2 Análise de Séries Temporais via DWT

Foi desenvolvido um código para fazer a análise de séries temporais (decomposição da série nos coeficientes *wavelet*) como apoio ao desenvolvimento do código de síntese de séries temporais (reconstrução do sinal a partir dos coeficientes *wavelet*).

Pode-se resumir o procedimento de análise de séries via DWT através dos seguintes passos:

1. Os seguintes parâmetros são passados para o programa de análise da série temporal via DWT:

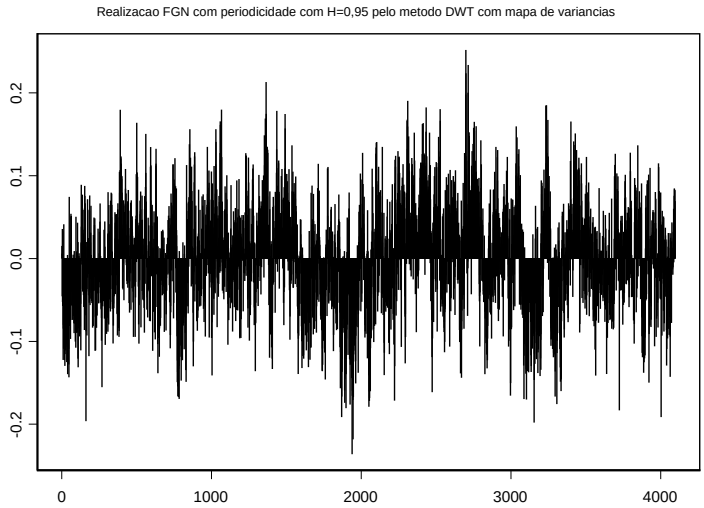



**Figura 3.4:** Realização de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWT com mapa de variâncias utilizando função de Haar, sorteio IID Gaussiano dos coeficientes *wavelet* e escolha de $\alpha = 0{,}9$ (H = 0,95) para inicialização do cálculo do mapa de variância.

- O número de escalas $J$ que serão utilizadas para análise da série;
- O número de momentos nulos que determina qual filtro *wavelet* será utilizado (Haar, Daubechies);
- O nome do arquivo com a série temporal a ser analisada;

2. Computam-se os coeficientes dos filtros passa-baixa $\{g_n\}$ (*dbaux*) e passa-alta $\{h_n\}$ dependendo da *wavelet* escolhida;

3. Carrega-se a série temporal num vetor;

4. Calcula-se o tamanho da série temporal, ou seja, o número de pontos dessa série;

5. Calcula-se o número de pontos dos coeficientes para cada escala $j$ da análise da série;

6. Na primeira escala ($j = 1$) efetua-se a operação de convolução (*conv*) da série com os filtros passa-baixa $\{g_n\}$ e passa-alta $\{h_n\}$ e em seguida a operação de *downsampling* (*dyaddown*) obtendo os coeficientes de aproximação e de detalhe da primeira escala respectivamente;

7. Efetua-se a operação de convolução (*conv*) do coeficiente de aproximação com os filtros passa-baixa $\{g_n\}$ e passa-alta $\{h_n\}$ e em seguida a operação

de *downsampling* (*dyaddown*) obtendo, respectivamente, os coeficientes de aproximação e de detalhe da escala seguinte;

8. Repete-se o último passo, substituindo-se $j$ por $j+1$ até que os coeficientes de aproximação e de detalhe da escala $J$ sejam obtidos.

## 3.3 Síntese de Séries Temporais via *Discrete Wavelet Packet Transform* (DWPT)

Em 2.3.4, foi visto que a DWPT pode ser considerada como uma coleção de transformadas ortonormais, sendo que cada uma delas pode ser rapidamente computada utilizando uma modificação do algoritmo da pirâmide para a DWT. Cada DWPT é associada com um nível $j$, e o $j$-ésimo nível da DWPT decompõe o intervalo de freqüência [0,1/2] em $2^j$ intervalos iguais e individuais. A DWPT resulta no que se pode chamar de decomposição tempo-freqüência pois cada coeficiente DWPT pode ser localizado numa banda de freqüência específica e num intervalo de tempo específico. Pode-se criar uma grande coleção de transformadas ortonormais através do agrupamento cuidadoso de bases vetoriais selecionadas de diferentes DWPTs. A DWT pode ser formada utilizando bases vetoriais da DWPT como visto na Fig. 2.20.

Com as regras apresentadas em 2.3.4, são construídos os $2^j$ coeficientes de cada nível $j$ até o nível menos refinado, assim construindo a *wavelet packet table* ou *wavelet packet tree* apresentada na Fig. 2.19. Os coeficientes $\mathbf{W}_{j,n}$ são nominalmente associados com freqüências no intervalo $\mathcal{I}_{j,n} \equiv [\frac{n}{2^{j+1}}, \frac{n+1}{2^{j+1}}]$.

A relação entre os coeficientes do filtro de reconstrução utilizado e os coeficientes do respectivo filtro de decomposição é a mesma que a mencionada em 3.1.

Os coeficientes $\{g_n\}$ e $\{h_n\}$ dos filtros passa-baixa e passa-alta respectivamente são obtidos da mesma forma que em 3.1.

A *equação de reconstrução* de Bäckar [30] apresentada anteriormente na Seção 3.1 também é utilizada para obter os coeficientes da DWPT seguindo as regras apresentadas na Seção 2.3.4 que identifica qual filtro deverá ser utilizado para cada coeficiente de um determinado nível para gerar o coeficiente do nível mais refinado.

### 3.3.1 Geração de Séries Temporais Gaussianas com LRD pelo método DWPT

Para a geração de séries pelo método DWPT é necessário calcular o mapa de variâncias conforme já foi descrito na Seção 3.1.2 para obter:

- o coeficiente de inicialização $n$ e seu par $n+1$ ou $n-1$ do nível menos refinado através da multiplicação do desvio padrão (raiz da variância retirada do mapa) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes do nível menos refinado deverão ter para iniciar a reconstrução;
- o coeficiente do nível $j$ que não foi reconstruído a partir dos coeficientes do nível $j+1$ mas que será necessário para a reconstrução do coeficiente do nível $j-1$.

O mapa de variâncias com a introdução de SRD também será calculado conforme descrito na Seção 3.1.2 a fim de gerar a mesma série temporal com a introdução do SRD no intervalo de freqüências solicitado.

As características que controlam a geração de séries temporais Gaussianas com LRD e densidade espectral de potência (DEP) $1/f$ com introdução de SRD pelo método DWPT são:

- O número de níveis que serão utilizadas para síntese da série $J$ ($J+1$ é a escala menos refinada e 1 a mais refinada) ;
- O número de momentos nulos que determina qual filtro *wavelet* será utilizado (Haar, Daubechies de ordem N);
- O número do coeficiente do nível menos refinado por onde a reconstrução deverá ser iniciada;
- O número do coeficiente do nível menos refinado referente ao início do intervalo de freqüência a ser introduzida a SRD;
- O número do coeficiente do nível menos refinado referente ao fim do intervalo de freqüência a ser introduzida a SRD;
- $\alpha$ - parâmetro que controla o decaimento de energia dos coeficientes *wavelet* utilizado no cálculo das variâncias dos coeficientes do nível menos refinado do mapa de variâncias;

- Coeficientes de inicialização do nível menos refinado calculados pela multiplicação do desvio padrão (raiz da variância retirada do mapa) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes do nível menos refinado deverão ter para iniciar a reconstrução;

Pode-se resumir o procedimento de geração de séries temporais gaussianas com LRD e DEP $1/f$ com introdução de SRD pelo método DWPT através dos seguintes passos:

1. Determina-se o número total de níveis $J+1$, em que $j = J+1$ será o nível menos refinado e $j = 1$ será o nível mais refinado (série temporal gerada);
2. Calcula-se o número de total de pontos $M$, que a série gerada deverá ter no final da síntese, a partir do número de níveis $J$ passado para o programa;
3. Computam-se os coeficientes dos filtros $\{g_n\}$ (*dbaux*) e $\{h_n\}$
4. Calcula-se quantos pontos os coeficientes de cada escala $j$ deverá ter;
5. Calcula-se o mapa de variâncias utilizando $\alpha$ como parâmetro para o cálculo da variância dos coeficientes do nível menos refinado; Tais variâncias são normalizadas pela variância do coeficiente *wavelet* da DWT (coeficiente $\mathbf{W}_{J+1,2}$) para obter uma distribuição Gaussiana independente e identicamente distribuída de média 0 e variância 1,0 na escala menos refinada $j = J$;
6. Calcula-se o mapa de variâncias SRD, com a introdução do ganho SRD nas variâncias dos coeficientes do nível menos refinado que estejam entre os coeficientes passados referentes ao início e fim do intervalo a ser introduzida a SRD;
7. Obtém-se o coeficiente de inicialização ($n$) do nível menos refinado ($j = J+1$) passado pelo programa e seu respectivo par (se $n$ for par será $n-1$, se $n$ for ímpar será $n+1$) através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter;
8. Iniciando em $j = (J+1) - 1$:
   - Calcula-se o número do coeficiente a ser reconstruído ($m$) no nível $j$; Verifica-se se o coeficiente a ser reconstruído ($m$) é par ou ímpar:
     - Se ($m$) for par:

    * o coeficiente ímpar do nível menos refinado $j+1$ será convoluído com o filtro passa-alta $\{h_n\}$ e o coeficiente par desse mesmo nível será convoluído com o filtro passa-baixa $\{g_n\}$.
    * Se o nível for $j = (J+1) - 1$:
      · calculam-se os coeficientes do nível menos refinado ($j = J+1$) que serão necessários para a reconstrução do coeficiente ímpar $(m-1)$ do nível $j = (J+1) - 1$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter;
    * Se o nível for mais refinado que $j = (J+1) - 1$:
      · calculam-se os coeficientes do nível menos refinado ($j = J+1$) que serão necessários para a reconstrução dos coeficientes dos níveis mais refinados até chegar ao coeficiente $(m-1)$ do nível $j$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter;
      · calculam-se recursivamente do nível $(i = (J+1) - 1)$ até o nível $i = j+1$ os coeficientes necessários para reconstruir o coeficiente ímpar $(m-1)$ do nível $j$. Os coeficientes ímpares serão calculados através da soma do coeficiente par do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$ com o coeficiente ímpar do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-baixa $\{g_n\}$. Os coeficientes pares serão calculados através da soma do coeficiente par do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-baixa $\{g_n\}$ com o coeficiente ímpar do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$. ;
    * calcula-se o coeficiente ímpar $(m-1)$ do nível $j$
  - Se for ímpar:
    * o coeficiente par do nível menos refinado $j+1$ será convoluído com o filtro passa-alta $\{h_n\}$ e o coeficiente ímpar desse mesmo nível será convoluído com o filtro passa-baixa $\{g_n\}$.
    * Se o nível for $j = (J+1) - 1$:

- · calculam-se os coeficientes do nível menos refinado ($j = J+1$) que serão necessários para a reconstrução do coeficiente par ($m+1$) do nível $j = (J+1)-1$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter;

* Se o nível for mais refinado que $j = (J+1)-1$:
  - · calculam-se os coeficientes do nível menos refinado ($j = J+1$) que serão necessários para a reconstrução dos coeficientes dos níveis mais refinados até chegar ao coeficiente ($m+1$) do nível $j$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter;
  - · calculam-se recursivamente do nível ($i = (J+1)-1$) até o nível $i = j+1$ os coeficientes necessários para reconstruir o coeficiente par ($m+1$) do nível $j$. Os coeficientes ímpares serão calculados através da soma do coeficiente par do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$ com o coeficiente ímpar do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-baixa $\{g_n\}$. Os coeficientes pares serão calculados através da soma do coeficiente par do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-baixa $\{g_n\}$ com o coeficiente ímpar do nível $i = i+1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$. ;
* calcula-se o coeficiente par ($m+1$) do nível $j$

9. Repete-se o último passo, substituindo-se $j$ por $j-1$ até que o coeficiente do nível $j = 1$ (série temporal) seja obtido.

A Fig. 3.5 mostra uma realização $1/f$ sem a introdução de SRD com $\alpha = 0,9$ ($H = 0,95$), $N = 4096$, Nr. *Vanishing Moment* $= 1$ (filtro de Haar), número de escalas $J = 12$ e coeficiente de inicialização $\mathbf{W}_{12,4096}$, que foi gerada pelo algoritmo da pirâmide modificado para a DWPT que utiliza o mapa de variância dos coeficientes dos coeficientes da DWPT. Nota-se que essa realização é visualmente parecida com a série gerada pelo método da DWT de Bäckar da Fig. 3.2

gerada por [1] e com a série gerada pelo método da DWT com mapa de variâncias desenvolvido neste trabalho da Fig. 3.3.

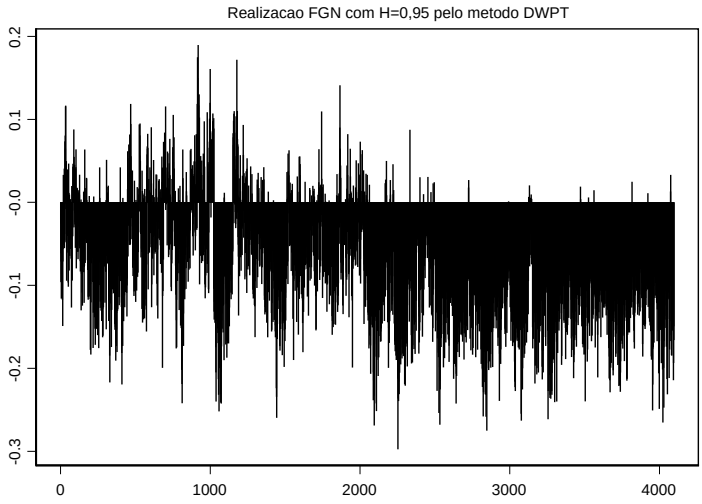



**Figura 3.5:** Realização de um processo $1/f$ de 4096 pontos obtida através do método DWPT utilizando função de Haar, $alpha = 0{,}9$ (H = 0,95) para inicialização do cálculo do mapa de variâncias e reconstrução inicializada pelo coeficiente $\mathbf{W}_{12,4096}$.

### 3.3.2 Geração de Séries Temporais Gaussinas com LRD com periodicidade pelo método DWPT

A periodicidade foi introduzida na geração de séries temporais gaussians com LRD pelo método DWPT assim caracterizando a série como uma série mista (LRD e SRD) da mesma forma que em 3.1.3.

As características que controlam a geração de séries temporais Gaussianas com LRD com periodicidade pelo método DWPT são:

- Coeficientes DWPT da série referente ao sinal do cosseno com amplitude, freqüência e número de pontos desejados;
- O número de níveis que serão utilizadas para síntese da série $J$ ($J + 1$ é a escala menos refinada e 1 a mais refinada) ;
- O número de momentos nulos que determina qual filtro *wavelet* será utilizado (Haar, Daubechies de ordem N);
- O número do coeficiente do nível menos refinado por onde a reconstrução deverá ser iniciada;

- $\alpha$ - parâmetro que controla o decaimento de energia dos coeficientes *wavelet* utilizado no cálculo das variâncias dos coeficientes do nível menos refinado do mapa de variâncias;

- Coeficientes de inicialização do nível menos refinado calculados pela multiplicação do desvio padrão (raiz da variância retirada do mapa) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes do nível menos refinado deverão ter para iniciar a reconstrução somado ao coeficiente DWPT equivalente da série do sinal cosseno utilizado para introduzir periodicidade à série.

Pode-se resumir o procedimento de geração de séries temporais gaussianas com LRD e DEP $1/f$ com periodicidade pelo método DWPT através dos seguintes passos:

1. Gera-se um sinal cosseno com amplitude e freqüência desejadas e com o mesmo número de pontos que a série que será introduzida a periodicidade;

2. Calculam-se os coeficientes DWPT da série gerada do sinal cosseno com o código de análise DWPT;

3. Determina-se o número total de níveis $J+1$, em que $j = J+1$ será o nível menos refinado e $j = 1$ será o nível mais refinado (série temporal gerada);

4. Calcula-se o número de total de pontos $M$, que a série gerada deverá ter no final da síntese, a partir do número de níveis $J$ passado para o programa;

5. Computam-se os coeficientes dos filtros $\{g_n\}$ (*dbaux*) e $\{h_n\}$

6. Calcula-se quantos pontos os coeficientes de cada escala $j$ deverá ter;

7. Calcula-se o mapa de variâncias utilizando $\alpha$ como parâmetro para o cálculo da variância dos coeficientes do nível menos refinado; Tais variâncias são normalizadas pela variância do coeficiente *wavelet* da DWT (coeficiente $\mathbf{W}_{J+1,2}$) para obter uma distribuição Gaussiana independente e identicamente distribuída de média 0 e variância 1,0 na escala menos refinada $j = J$;

8. Carrega-se o arquivo com os coeficientes DWPT da série do sinal cosseno;

9. Obtém-se o coeficiente de inicialização ($n$) do nível menos refinado ($j = J+1$) passado pelo programa e seu respectivo par (se $n$ for par será $n-1$, se $n$ for ímpar será $n+1$) através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números

aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter somado ao coeficiente DWPT equivalente da série do cosseno utilizado para introduzir periodicidade à série;

10. Iniciando em $j = (J + 1) - 1$:

    - Calcula-se o número do coeficiente a ser reconstruído $(m)$ no nível $j$; Verifica-se se o coeficiente a ser reconstruído $(m)$ é par ou ímpar:
      - Se $(m)$ for par:
        * o coeficiente ímpar do nível menos refinado $j + 1$ será convoluído com o filtro passa-alta $\{h_n\}$ e o coeficiente par desse mesmo nível será convoluído com o filtro passa-baixa $\{g_n\}$.
        * Se o nível for $j = (J + 1) - 1$:
          · calculam-se os coeficientes do nível menos refinado $(j = J + 1)$ que serão necessários para a reconstrução do coeficiente ímpar $(m - 1)$ do nível $j = (J + 1) - 1$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter somado ao coeficiente DWPT equivalente da série do cosseno utilizado para introduzir periodicidade à série;
        * Se o nível for mais refinado que $j = (J + 1) - 1$:
          · calculam-se os coeficientes do nível menos refinado $(j = J + 1)$ que serão necessários para a reconstrução dos coeficientes dos níveis mais refinados até chegar ao coeficiente $(m-1)$ do nível $j$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter somado ao coeficiente DWPT equivalente da série do cosseno utilizado para introduzir periodicidade à série;
          · calculam-se recursivamente do nível $(i = (J + 1) - 1)$ até o nível $i = j + 1$ os coeficientes necessários para reconstruir o coeficiente ímpar $(m-1)$ do nível $j$. Os coeficientes ímpares serão calculados através da soma do coeficiente par do nível $i = i + 1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$ com o coeficiente ímpar do nível $i = i + 1$ após

*upsampling* e convolução com o filtro passa-baixa $\{g_n\}$. Os coeficientes pares serão calculados através da soma do coeficiente par do nível $i = i + 1$ após *upsampling* e convolução com o filtro passa-baixa $\{g_n\}$ com o coeficiente ímpar do nível $i = i + 1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$. ;

* calcula-se o coeficiente ímpar $(m - 1)$ do nível $j$

– Se for ímpar:

* o coeficiente par do nível menos refinado $j+1$ será convoluído com o filtro passa-alta $\{h_n\}$ e o coeficiente ímpar desse mesmo nível será convoluído com o filtro passa-baixa $\{g_n\}$.
* Se o nível for $j = (J + 1) - 1$:
  · calculam-se os coeficientes do nível menos refinado $(j = J + 1)$ que serão necessários para a reconstrução do coeficiente par $(m+1)$ do nível $j = (J+1)-1$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter somado ao coeficiente DWPT equivalente da série do cosseno utilizado para introduzir periodicidade à série;
* Se o nível for mais refinado que $j = (J + 1) - 1$:
  · calculam-se os coeficientes do nível menos refinado $(j = J + 1)$ que serão necessários para a reconstrução dos coeficientes dos níveis mais refinados até chegar ao coeficiente $(m+1)$ do nível $j$ através da multiplicação do desvio padrão (raiz da variância retirada do mapa de variâncias) por um vetor de números aleatórios de dimensão igual ao número de pontos que os coeficientes daquele nível deverão ter somado ao coeficiente DWPT equivalente da série do cosseno utilizado para introduzir periodicidade à série;
  · calculam-se recursivamente do nível $(i = (J + 1) - 1)$ até o nível $i = j + 1$ os coeficientes necessários para reconstruir o coeficiente par $(m + 1)$ do nível $j$. Os coeficientes ímpares serão calculados através da soma do coeficiente par do nível $i = i + 1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$ com o coeficiente ímpar do nível $i = i + 1$ após

*upsampling* e convolução com o filtro passa-baixa $\{g_n\}$. Os coeficientes pares serão calculados através da soma do coeficiente par do nível $i = i + 1$ após *upsampling* e convolução com o filtro passa-baixa $\{g_n\}$ com o coeficiente ímpar do nível $i = i + 1$ após *upsampling* e convolução com o filtro passa-alta $\{h_n\}$. ;

* calcula-se o coeficiente par $(m + 1)$ do nível $j$

11. Repete-se o último passo, substituindo-se $j$ por $j - 1$ até que o coeficiente do nível $j = 1$ (série temporal) seja obtido.

A Fig. 3.6 mostra uma realização $1/f$ com periodicidade de amplitude 0,008 com $\alpha = 0,9$ ($H = 0,95$), $N = 4096$, Nr. *Vanishing Moment* = 1 (filtro de Haar), número de escalas $J = 12$ e coeficiente de inicialização $\mathbf{W}_{12,4096}$, que foi gerada pelo algoritmo da pirâmide modificado para a DWPT que utiliza o mapa de variância dos coeficientes da DWPT. Nota-se que essa realização é visualmente parecida com a série gerada pelo método da DWT de Bäckar da Fig. 3.2 gerada por [1] e com a série gerada pelo método da DWT com mapa de variâncias desenvolvido neste trabalho da Fig. 3.3.

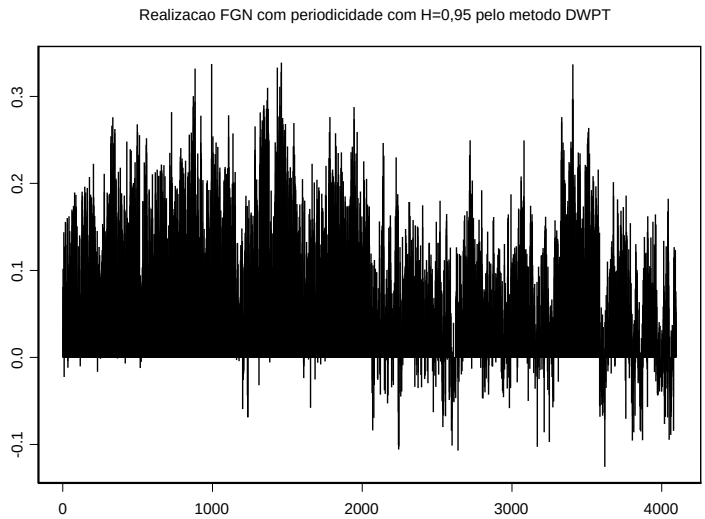



**Figura 3.6:** Realização de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWPT utilizando função de Haar, *alpha* = 0,9 (H = 0,95) para inicialização do cálculo do mapa de variâncias e reconstrução inicializada pelo coeficiente $\mathbf{W}_{12,4096}$.

## 3.4 Análise de Séries Temporais via DWPT

Foi desenvolvido um código para fazer a análise de séries temporais (decomposição da série nos coeficientes DWPT) como apoio ao desenvolvimento do código de síntese de séries temporais (reconstrução do sinal a partir dos coeficientes DWPT)

Pode-se resumir o procedimento de análise de séries via DWPT através dos seguintes passos:

1. Os seguintes parâmetros são passados para o programa de análise da série temporal via DWPT:
   - O número de escalas $J$ que serão utilizadas para análise da série;
   - O número de momentos nulos que determina qual filtro *wavelet* será utilizado (Haar, Daubechies);
   - O nome do arquivo com a série temporal a ser analisada;
2. Computam-se os coeficientes dos filtros passa-baixa $\{g_n\}$ (*dbaux*) e passa-alta $\{h_n\}$ dependendo da *wavelet* escolhida;
3. Carrega-se a série temporal num vetor;
4. Calcula-se o tamanho da série temporal, ou seja, o número de pontos dessa série;
5. Calcula-se o número de pontos dos coeficientes para cada escala $j$ da análise da série;
6. Na primeira escala ($j = 1$) efetua-se a operação de convolução (*conv*) da série com os filtros passa-baixa $\{g_n\}$ e passa-alta $\{h_n\}$ e em seguida a operação de *downsampling* (*dyaddown*) obtendo os coeficientes ($n = 1$) e ($n = 2$) respectivamente da primeira escala;
7. Efetua-se a operação de convolução (*conv*) dos coeficientes da primeira escala com os filtros passa-baixa $\{g_n\}$ e passa-alta $\{h_n\}$ e em seguida a operação de *downsampling* (*dyaddown*) obtendo, os coeficientes ímpar e par respectivamente da escala seguinte se o coeficiente da primeira escala era ímpar e os coeficientes par e ímpar respectivamente da escala seguinte se o coeficiente da primeira escala era par;
8. Repete-se o último passo, substituindo-se $j$ por $j+1$ até que os coeficientes da escala $J$ solicitada sejam obtidos.

# 4 Validações

Neste capítulo os métodos DWT com mapa de variâncias e DWPT serão validados. O trabalho [1] que faz a geração de séries temporais Gaussianas com LRD pelo método DWT de Bäckar foi utilizado como comparação às gerações de séries pelos métodos DWT com mapa de variâncias e DWPT desenvolvidas neste trabalho a fim de validar que tais métodos geram séries temporais gaussianas com LRD.

Como este trabalho está apoiado no trabalho de [1] que já demonstrou que o método de geração de séries temporais via transformada *wavelet* realmente gera séries gaussianas com LRD, a validação deste trabalho ficou limitada à análise do periodograma, da função de autocorrelação e estimativa do parâmetro de Hurst pelos métodos de Whittle e do periodograma.

A análise estatística das séries Gaussinas foi feita utilizando as seguintes estatísticas [60] [1]:

1. *função de autocorrelação* $\rho_X(\tau)$;
2. DEP $P_X(f)$;
3. $H$, pelos métodos de Whittle e do periodograma;

O estimador da função de autocorrelação é dado por:

$$\hat{\rho}_X(\tau) = \frac{1}{M s_X^2} \sum_{k=m+1}^{M} (X_k - \hat{\mu})(X_{k-m} - \hat{\mu}) \,, \tag{4.1}$$

em que $M$ é o número de amostras, $s_X^2$ é a variância amostral e $\hat{\mu}$ é o estimador da média $\mu$ de $X$. Note-se que $-1 \leq \rho \leq 1$.

O estimador $\hat{P}_X(f)$ da DEP é obtido pelo método *não-paramétrico*[1] do *periodograma* [61], com *janelamento de dados* (*data tapering*, para redução de *vaza-*

[1]Os métodos *paramétricos* de análise espectral são baseados em modelos AR, MA e ARMA. Portanto não devem ser aplicados para estimação da DEP de um ruído $1/f$.

*mento* de potência) e suavização (*smoothing*, para redução da variabilidade de $\hat{P}_X(f)$). O periodograma é calculado via[2]:

$$\hat{P}_X(f) = \frac{1}{M}|X(f)|^2 . \tag{4.2}$$

O método de estimação do parâmetro H de Whittle [62] é baseado numa estimação de máxima verossimilhança no domínio da freqüência do modelo FD(d). Esse método usa o periodograma. O método de estimação de H pelo periodograma baseia-se no fato de que $P_X(f) \propto f^{2H-1}$ para freqüências próximas de zero.

Essa validação foi feita em três partes:

1. Efetuou-se uma análise comparativa entre as séries geradas sem SRD pelos métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT a fim de validar que as séries geradas pelos métodos DWT com mapa de variâncias e DWPT são $1/f$ e LRD assim como as geradas pelo método DWT de Bäckar desenvolvido em [1];

2. Comparou-se as séries geradas pelos métodos DWT com mapa de variâncias e DWPT antes e depois da inserção do ganho SRD pela análise dos periodogramas;

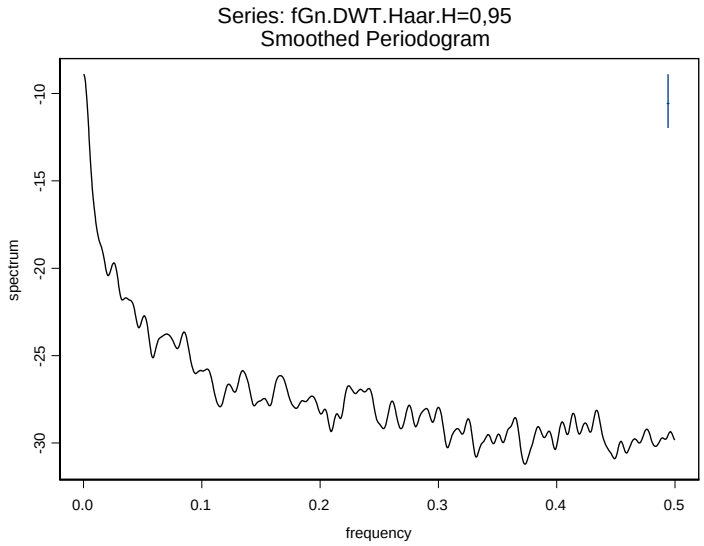
3. Comparou-se as séries geradas com periodicidade pelos métodos DWT com mapa de variâncias e DWPT.

## 4.1 Comparação dos métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT

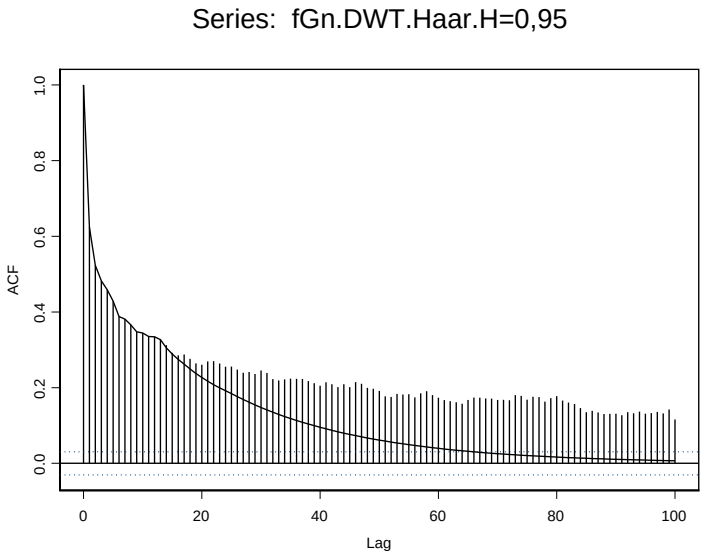
As Figs. 4.1, 4.2 e 4.3 mostram gráficos do periodograma suavizado e das funções de autocorrelação para séries simuladas pelos métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT respectivamente. Os periodogramas mostram que as DEPs das séries simuladas com H=0,95 de todos os métodos têm pólos na origem, ou seja, que as séries são $1/f$. Pelas funções de autocorrelação também pode-se observar que as séries são $1/f$. Verifica-se que as séries geradas pelo método DWT com mapa de variâncias são similares às geradas pelo método DWT de Bäckar com a vantagem de ser mais flexível pois o mapa pode ser alterado para séries com diferentes espectros além de possibilitar a introdução

[2] A definição foi dada sem incluir o janelamento e a suavização, para melhor compreensão da natureza essencial do estimador.

de SRD em diferentes faixas de freqüência. Também verificou-se que as séries geradas pelo método DWPT são similares às geradas pelos métodos DWT de Bäckar e DWT com mapa de variâncias com a vantagem de poderem ser geradas a partir de qualquer coeficiente DWPT sendo assim ainda mais flexível em relação à introdução de SRD em diferentes faixas de freqüência que o método DWT com mapa de variâncias.



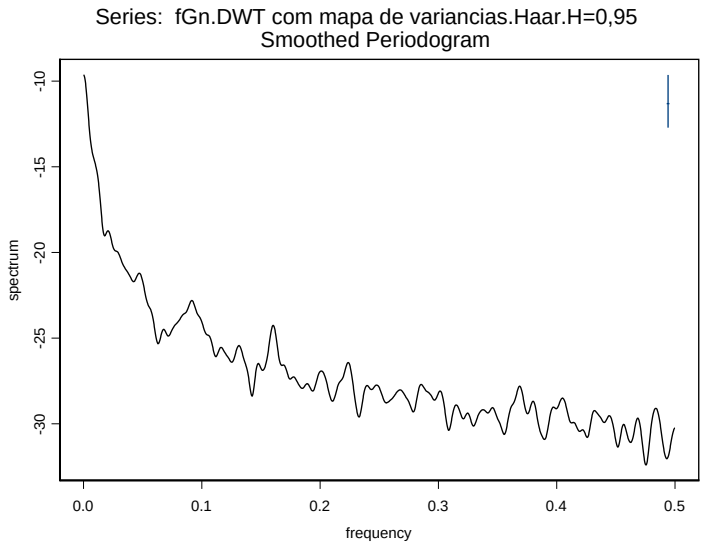

**(a)**



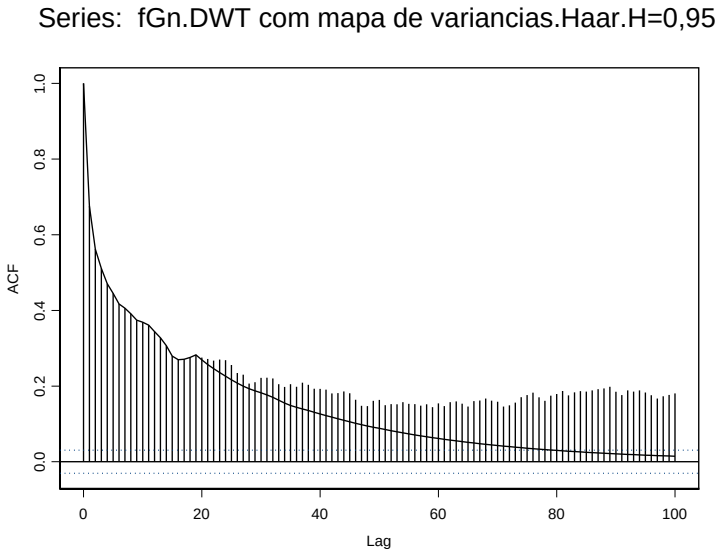

**(b)**

**Figura 4.1:** A Fig. (a) é o periodograma suavizado e a Fig. (b) é a função de autocorrelação da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWT de Bäckar utilizando função de Haar e $\alpha = 0{,}9$ (H = 0,95).

Além da análise do periodograma e da função de autocorrelação, também foram feitas análises estatísticas dos três métodos utilizando a estimativa de H

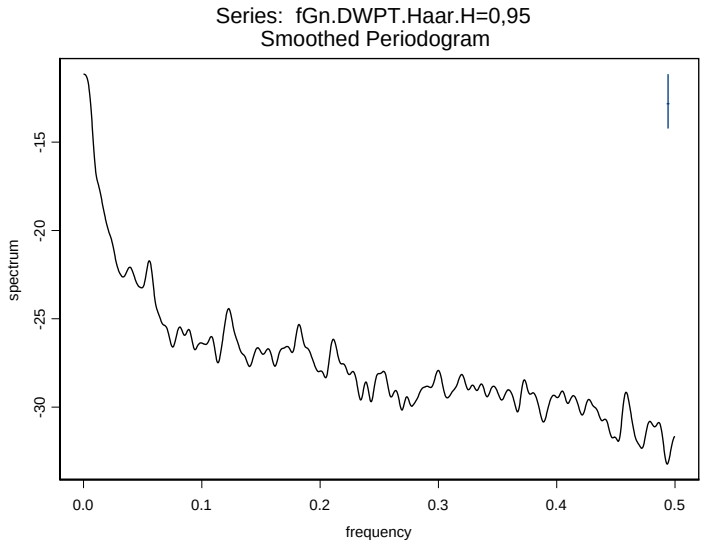



**(a)**

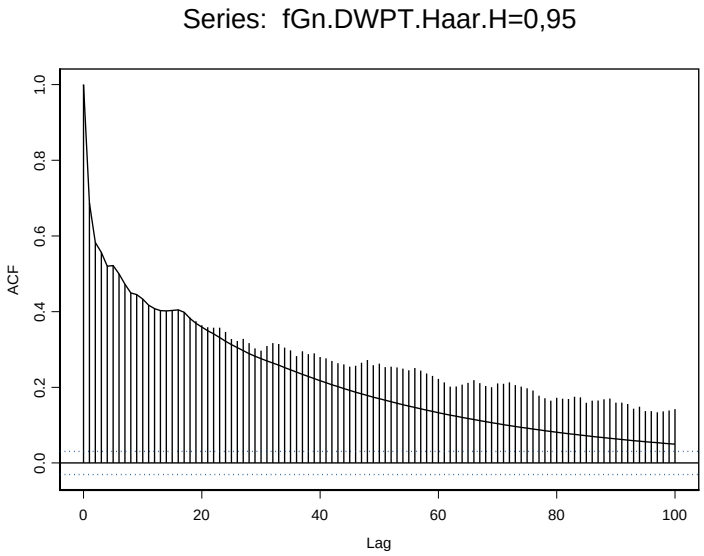



**(b)**

**Figura 4.2:** A Fig. (a) é o periodograma suavizado e a Fig. (b) é a função de autocorrelação da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWT com mapa de variâncias utilizando função de Haar e $\alpha$ = 0,9 (H = 0,95).




**(a)**




**(b)**

**Figura 4.3:** A Fig. (a) é o periodograma suavizado e a Fig. (b) é a função de autocorrelação da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWPT utilizando função de Haar e $\alpha = 0{,}9$ (H = 0,95).

pelos métodos de Whittle e do periodograma. Foram geradas dez series com os mesmos parâmetros - 4096 pontos, função de Haar e $\alpha$ = 0,9 (H = 0,95) para cada método e foi calculada a estimativa de H pelos métodos de Whittle e do periodograma. A Tabela 4.1 mostra as médias dessas estimativas para essas 10 gerações. De acordo com Paxson [8], o método e Whittle é um bom método de estimação do parâmetro de Hurst H para séries que apresentam LRD e como pode-se observar na tabela, esse método é melhor que o do periodograma para todas os métodos de geração de séries. Comparando-se os valores do parâmetro de Hurst estimados pelo método de Whittle nos três metódos de geração de séries, conclui-se que o método DWPT é o que se aproxima mais do parâmetro de Hurst real que foi inserido como parâmetro para geração das séries ($\alpha$ = 0,9 (H = 0,95)). Pode-se verificar que a estimativa no método DWT com mapa de variâncias é muito próxima da estimativa no método de Bäckar pois são a mesma transformada.

**Tabela 4.1:** Estimativas do parâmetro de Hurst pelos métodos de Whittle e do periodograma das séries geradas pelos métodos DWT de Bäckar, DWT com mapa de variâncias e DWPT

| Método | **H Whittle** | **H periodograma** |
|---|---|---|
| DWT de Bäckar | 0.92989941 | 0.93369818 |
| DWT com mapa de variâncias | 0.936262928 | 0.90977339 |
| DWPT | 0.953252836 | 0.970969597 |

## 4.2 Comparação entre as séries geradas sem e com SRD via DWT com mapa de variâncias e DWPT

Foram geradas séries gaussianas com LRD com os parâmetros $\alpha = 0,9$ ($H = 0,95$), $N = 4096$, Nr. *Vanishing Moment* = 1 (filtro de Haar), número de escalas $J = 12$ pelo método da DWT com mapa de variâncias e pelo método DWPT com coeficiente de inicialização $\mathbf{W}_{12,2048}$. Simultaneamente foram introduzidas nestas séries, SRD com ganho constante igual a 10 e SRD com ganho variável igual a 10-1-10 em três faixas de freqüência diferentes conforme as funções ganho SRD da Fig. 4.4. No total foram 18 séries geradas, 6 séries com LRD e 6 séries mistas com LRD e introdução de ganho SRD constante e 6 séries mistas com LRD e introdução de ganho SRD variável (9 realizações pelo método DWT com mapa de variâncias e 9 realizações pelo método DWPT).

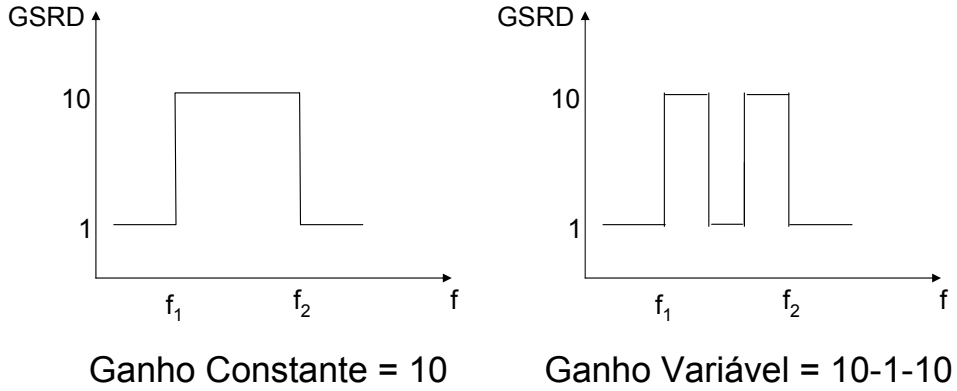



**Figura 4.4:** Ganho SRD constante = 10 e variável = 10-1-10

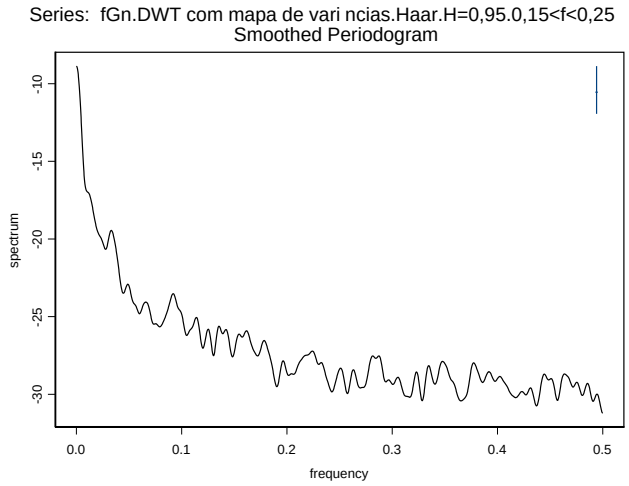
### 4.2.1 Faixa de freqüência: 0,15 - 0,25

#### 4.2.1.1 Método DWT com mapa de variâncias:

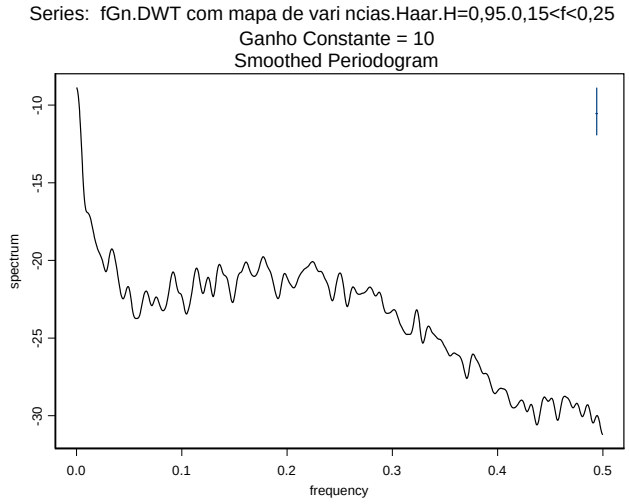
A Fig. 4.5 mostra os periodogramas da série LRD pura e da mesma série com a introdução de SRD com ganho constante igual à 10 e com ganho variável igual à 10-1-10 na faixa de freqüência 0,15 - 0,25 Hz geradas pelo método DWT com mapa de variâncias. A partir da comparação desses gráficos observa-se o efeito da introdução de SRD nessa faixa de freqüência de forma clara. Observa-se que a diferença entre o ganho constante e o ganho variável não é percebida neste método.

#### 4.2.1.2 Método DWPT

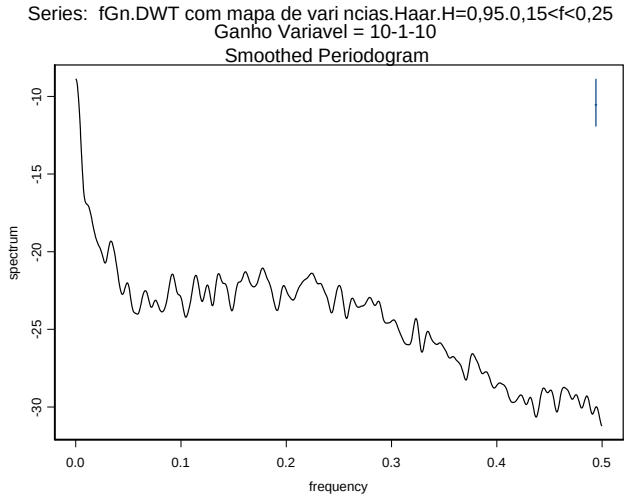
A Fig. 4.6 mostra os periodogramas da série LRD pura e da mesma série com a introdução de SRD com ganho constante igual à 10 e com ganho variável igual à 10-1-10 na faixa de freqüência 0,15 - 0,25 Hz geradas pelo método DWPT. A partir da comparação desses gráficos observa-se o efeito da introdução de SRD nessa faixa de freqüência de forma clara. Comparando-se esses gráficos com os gráficos da Fig. 4.5 observa-se que o efeito da introdução de SRD pelo método




**(a)**




**(b)**




**(c)**

**Figura 4.5:** A Fig. (a) é o periodograma suavizado da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWT com mapa de variâncias utilizando função de Haar e $\alpha$=0,9 (H=0,95); A Fig. (b) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD constante=10 na faixa de freqüência 0,15-0,25; A Fig. (c) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD variável=10-1-10 na faixa de freqüência 0,15-0,25.

DWPT é apresentado de forma um pouco mais evidente que pelo método DWT com mapa de variâncias. Observa-se que a diferença entre o ganho constante e o ganho variável é percebida neste método.

### 4.2.2 Faixa de freqüência: 0,25 - 0,35:

#### 4.2.2.1 Método DWT com mapa de variâncias

A Fig. 4.7 mostra os periodogramas da série LRD pura e da mesma série com a introdução de SRD com ganho constante igual à 10 e com ganho variável igual à 10-1-10 na faixa de freqüência 0,25 - 0,35 Hz geradas pelo método DWT com mapa de variâncias. A partir da comparação desses gráficos observa-se o efeito da introdução de SRD nessa faixa de freqüência de forma mais discreta que na faixa de freqüência 0,15 - 0,25 Hz como pode-se observar na Fig. 4.5. Observa-se que a diferença entre o ganho constante e o ganho variável não é percebida neste método.

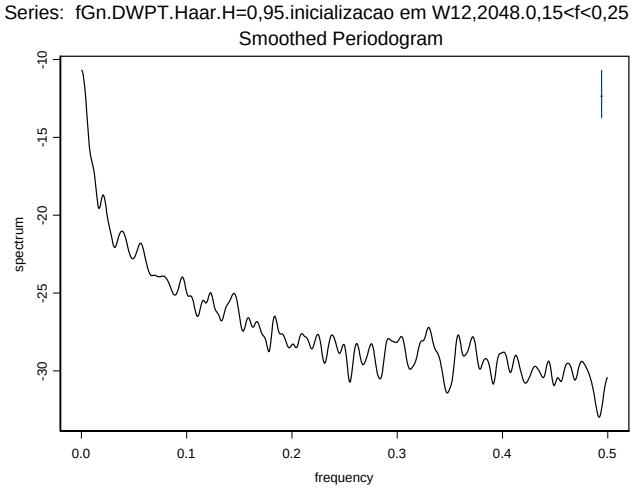
#### 4.2.2.2 Método DWPT

A Fig. 4.8 mostra os periodogramas da série LRD pura e da mesma série com a introdução de SRD com ganho constante igual à 10 e com ganho variável igual à 10-1-10 na faixa de freqüência 0,25 - 0,35 Hz geradas pelo método DWPT. A partir da comparação desses gráficos observa-se o efeito da introdução de SRD nessa faixa de freqüência de forma clara. Comparando-se esses gráficos com os gráficos da Fig. 4.7 observa-se que o efeito da introdução de SRD pelo método DWPT é apresentado de forma mais evidente que pelo método DWT com mapa de variâncias. A partir da comparação desses gráficos com os da faixa de freqüência 0,15 - 0,25 Hz da Fig. 4.6 observa-se que o efeito da introdução de SRD nessa faixa de freqüência se dá da mesma forma que na faixa de freqüência mais baixa (0,15 - 0,25 Hz). Observa-se que a diferença entre o ganho constante e o ganho variável é percebida neste método.

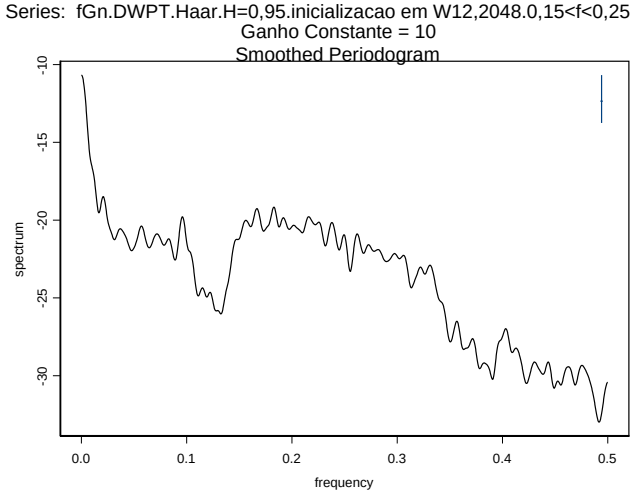
### 4.2.3 Faixa de freqüência: 0,35 - 0,45:

#### 4.2.3.1 Metodo DWT com mapa de variâncias

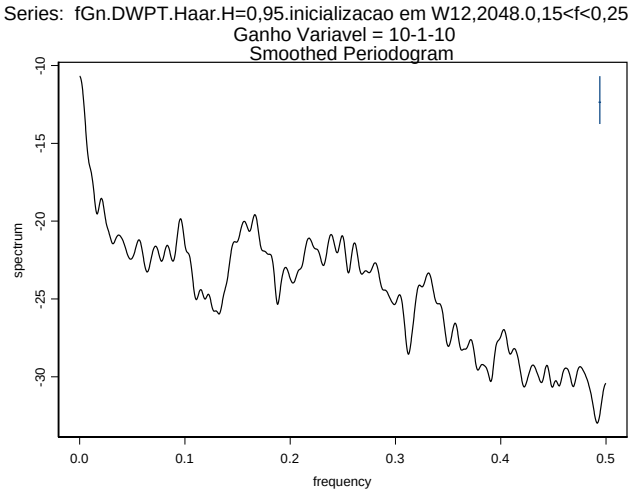
A Fig. 4.9 mostra os periodogramas da série LRD pura e da mesma série com a introdução de SRD com ganho constante igual à 10 e com ganho variável igual à 10-1-10 na faixa de freqüência 0,35 - 0,45 Hz geradas pelo método DWT com mapa




**(a)**

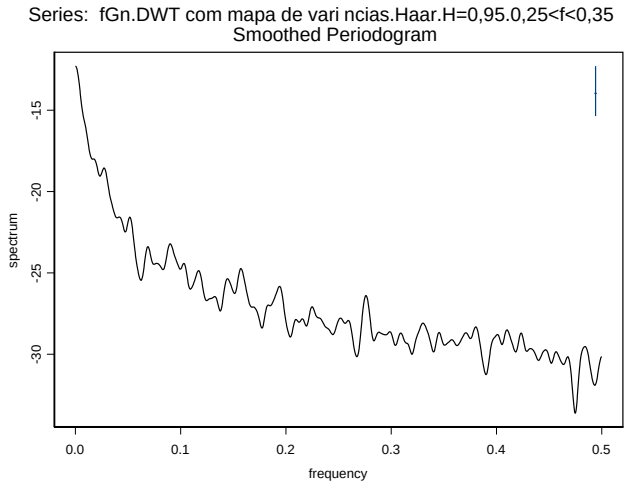



**(b)**

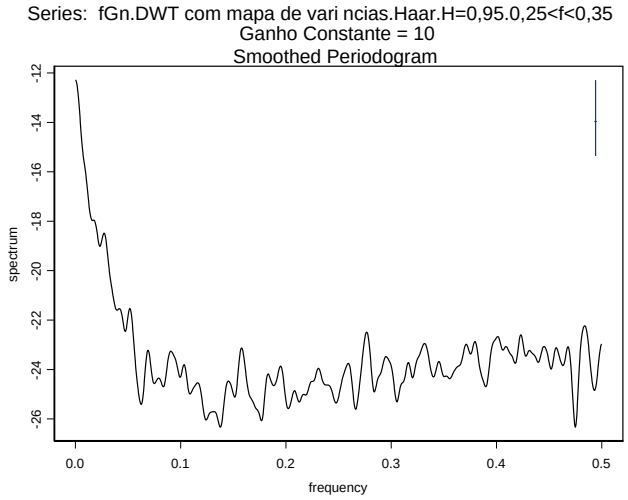



**(c)**

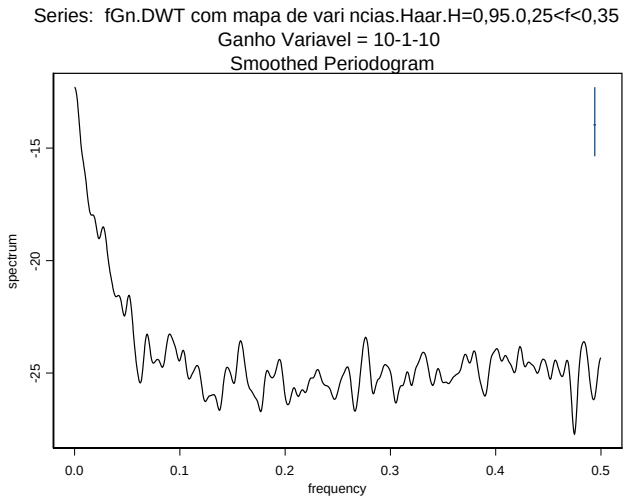
**Figura 4.6:** A Fig. (a) é o periodograma suavizado da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWPT utilizando função de Haar, $\alpha$=0,9 (H=0,95) e coeficiente de inicialização=$\mathbf{W}_{12,2048}$; A Fig. (b) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD constante=10 na faixa de freqüência 0,15-0,25; A Fig. (c) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD variável=10-1-10 na faixa de freqüência 0,15-0,25.




**(a)**

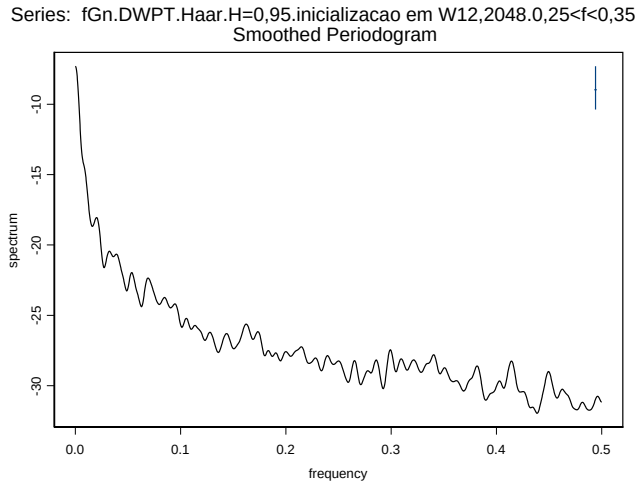



**(b)**

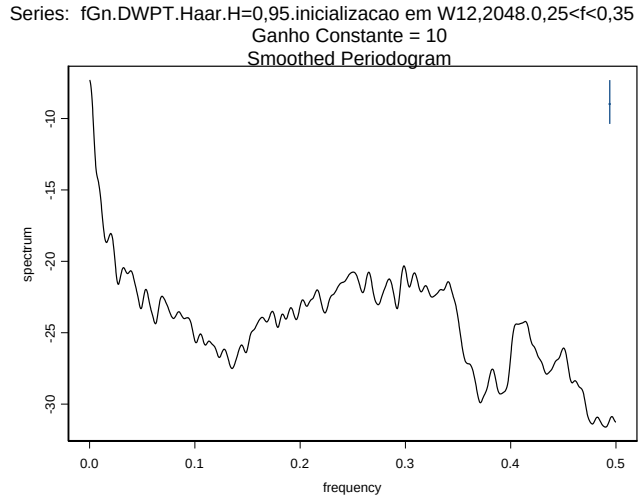



**(c)**

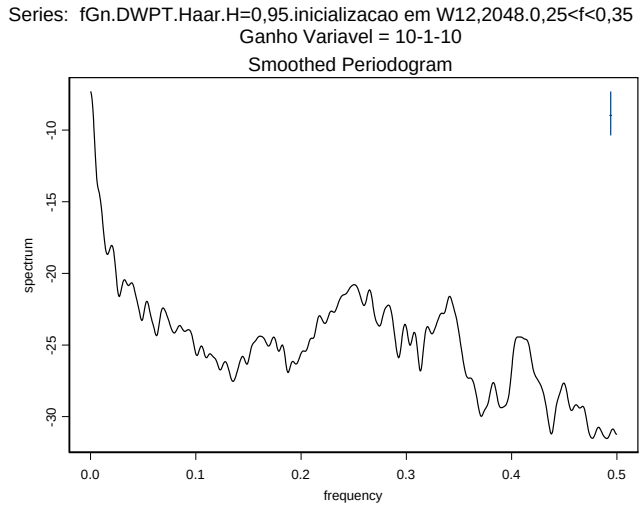
**Figura 4.7:** A Fig. (a) é o periodograma suavizado da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWT com mapa de variâncias utilizando função de Haar e $\alpha$=0,9 (H=0,95); A Fig. (b) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD constante=10 na faixa de freqüência 0,25-0,35; A Fig. (c) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD variável=10-1-10 na faixa de freqüência 0,25-0,35.




**(a)**




**(b)**




**(c)**

**Figura 4.8:** A Fig. (a) é o periodograma suavizado da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWPT utilizando função de Haar, $\alpha$=0,9 (H=0,95) e coeficiente de inicialização=$\mathbf{W}_{12,2048}$; A Fig. (b) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD constante=10 na faixa de freqüência 0,25-0,35; A Fig. (c) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD variável=10-1-10 na faixa de freqüência 0,25-0,35.

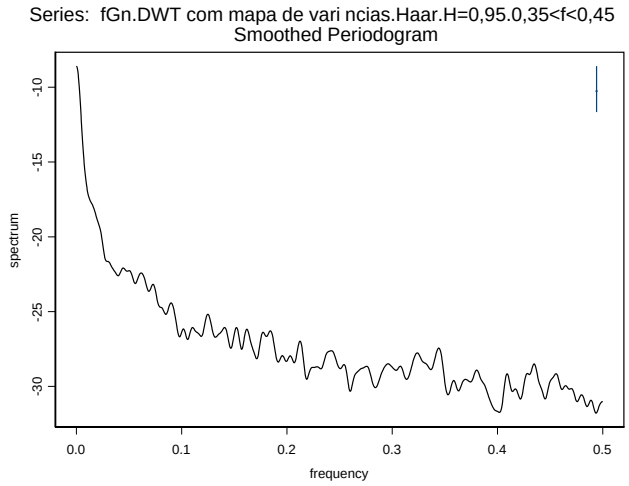
de variâncias. A partir da comparação desses gráficos observa-se que o efeito da introdução de SRD nessa faixa de freqüência ocorre de forma muito discreta e quase não pode ser observado. Assim, quanto maior a faixa de freqüência onde é introduzida a SRD, menor o efeito observado no periodograma. Isso se deve ao fato de o método de geração via DWT não especificar a forma da DEP em freqüências médias e altas. Sendo assim, este método não é bom para efeitos em médias e altas freqüências. Observa-se que a diferença entre o ganho constante e o ganho variável não é percebida neste método.

#### 4.2.3.2 Método DWPT

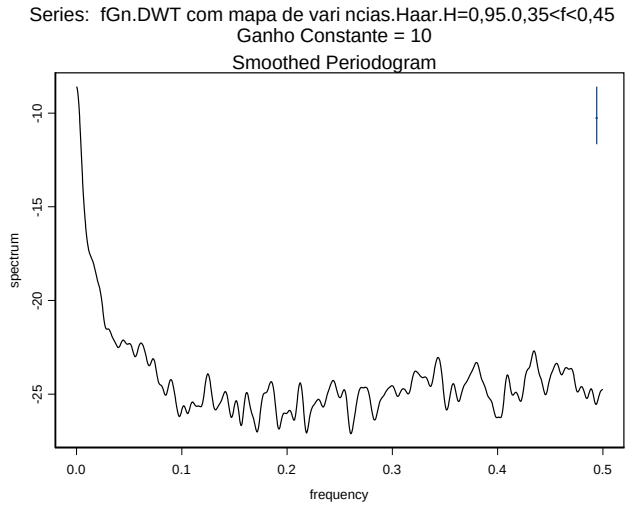
A Fig. 4.10 mostra os periodogramas da série LRD pura e da mesma série com a introdução de SRD com ganho constante igual à 10 e com ganho variável igual à 10-1-10 na faixa de freqüência 0,35 - 0,45 Hz geradas pelo método DWPT. A partir da comparação desses gráficos observa-se o efeito da introdução de SRD nessa faixa de freqüência de forma clara. Comparando-se esses gráficos com os gráficos da Fig. 4.9 observa-se que o efeito da introdução de SRD pelo método DWPT é apresentado de forma muito mais evidente que pelo método DWT com mapa de variâncias. A partir da comparação desses gráficos com os da faixa de freqüência 0,15 - 0,25 Hz das Figs. 4.6 observa-se que o efeito da introdução de SRD nessa faixa de freqüência se dá da mesma forma que na faixa de freqüência mais baixa (0,15 - 0,25 Hz). Isso se deve ao método de geração via DWPT ser flexível em relação às freqüências e assim especificar a forma da DEP em freqüências médias e altas. Portanto, este método é bom para efeitos em médias e altas freqüências e apresenta uma maior flexibilidade em relação ao método da DWT com mapa de variâncias. Observa-se que a diferença entre o ganho constante e o ganho variável é percebida neste método.

## 4.3 Comparação entre as séries geradas com periodicidade pelo DWT com mapa de variâncias e DWPT

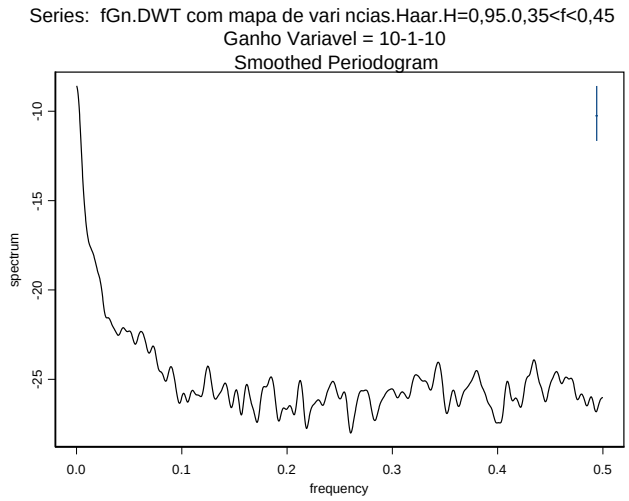
Esta seção apresenta os resultados das séries gaussianas com LRD e DEP $1/f$ com periodicidade de amplitude 0,008 e parâmetros $\alpha = 0,9$ ($H = 0,95$), $N = 4096$, Nr. *Vanishing Moment* = 1 (filtro de Haar), número de escalas $J = 12$ geradas pelos métodos DWT com mapa de variâncias e DWPT com quatro coeficientes de inicialização diferentes. No total foram 5 séries geradas, 1 série gerada pelo




**(a)**

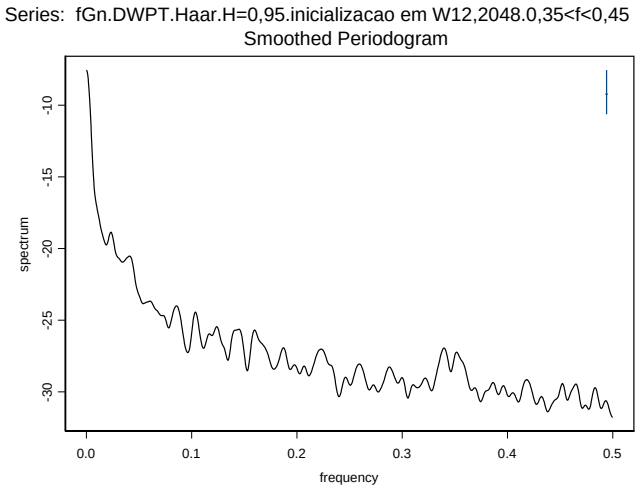



**(b)**

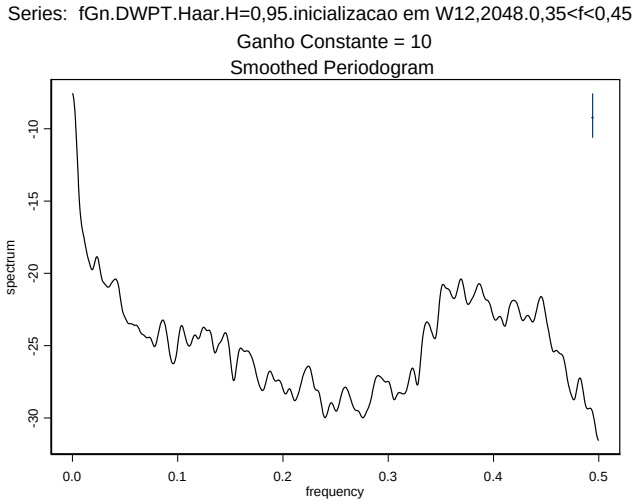



**(c)**

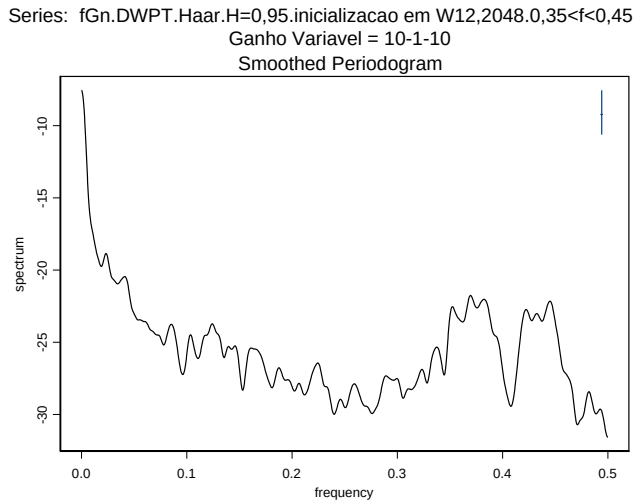
**Figura 4.9:** A Fig. (a) é o periodograma suavizado da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWT com mapa de variâncias utilizando função de Haar e $\alpha$=0,9 (H=0,95); A Fig. (b) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD constante=10 na faixa de freqüência 0,35-0,45; A Fig. (c) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD variável=10-1-10 na faixa de freqüência 0,35-0,45.




**(a)**




**(b)**




**(c)**

**Figura 4.10:** A Fig. (a) é o periodograma suavizado da série temporal de um processo $1/f$ de 4096 pontos obtida através do método DWPT utilizando função de Haar, $\alpha$=0,9(H=0,95) e coeficiente de inicialização=$\mathbf{W}_{12,2048}$; A Fig. (b) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD constante=10 na faixa de freqüência 0,35-0,45; A Fig. (c) é o periodograma da mesma série temporal da Fig. (a) com a introdução de um ganho SRD variável=10-1-10 na faixa de freqüência 0,35-0,45.

método DWT com mapa de variâncias e 4 pelo método DWPT.

### 4.3.1 Método DWT com mapa de variâncias

A Fig. 4.11 mostra o sinal, a função de autocorrelação e o periodograma da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 gerada pelo método DWT com mapa de variâncias. A partir da análise destes gráficos, pode-se observar a presença da periodicidade do cosseno. No periodograma da Fig. 4.11c observa-se o comportamento misto da série LRD com SRD na freqüência 0,35 introduzida pela periodicidade.

### 4.3.2 Método DWPT

Esta seção apresenta os resultados das séries gaussianas com LRD e DEP $1/f$ com periodicidade de amplitude 0,008 e parâmetros $\alpha = 0,9$ ($H = 0,95$), $N =$ 4096, Nr. *Vanishing Moment* = 1 (filtro de Haar), número de escalas $J = 12$ geradas pelo método DWPT com quatro coeficientes de inicialização diferentes.

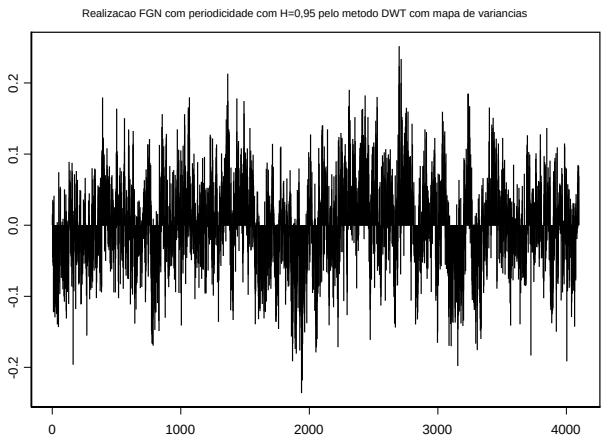
Foram feitas análises da função de auto-correlação e do peridograma das séries geradas inicializadas pelos 4 coeficientes. A partir da análise dos gráficos pode-se observar a presença da periodicidade do cosseno. Comparando-se os gráficos da função de auto-correlação observa-se que o decaimento é mais suave conforme o coeficiente de inicialização aumenta. Observando os gráficos do periodograma verifica-se que o comportamento misto da série LRD com SRD na freqüência 0,35 introduzida pela periodicidade ocorre independente do coeficiente de inicialização. Nota-se que os espectros gerados são semelhantes mas não são idênticos. Portanto, neste caso, foi possível controlar o espectro do sinal gerado a partir de qualquer ponto de início (coeficiente de inicialização).

#### 4.3.2.1 Coeficiente de inicialização $\mathbf{W_{12,1}}$

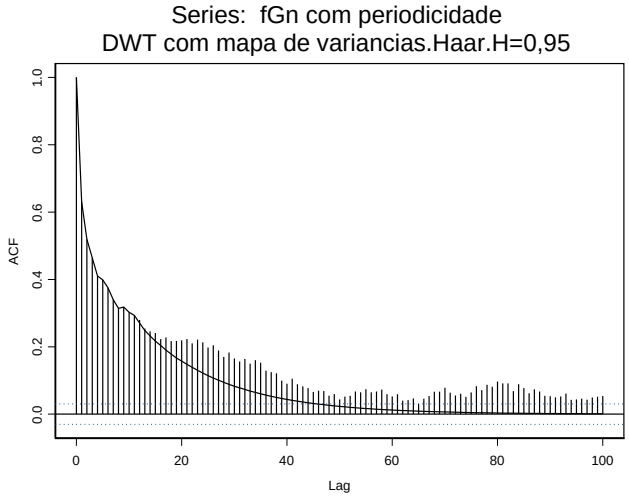
A Fig. 4.12 mostra o sinal, a função de autocorrelação e o periodograma da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 gerada pelo método DWPT inicializado pelo coeficiente $\mathbf{W}_{12,1}$.

#### 4.3.2.2 Coeficiente de inicialização $\mathbf{W_{12,2048}}$

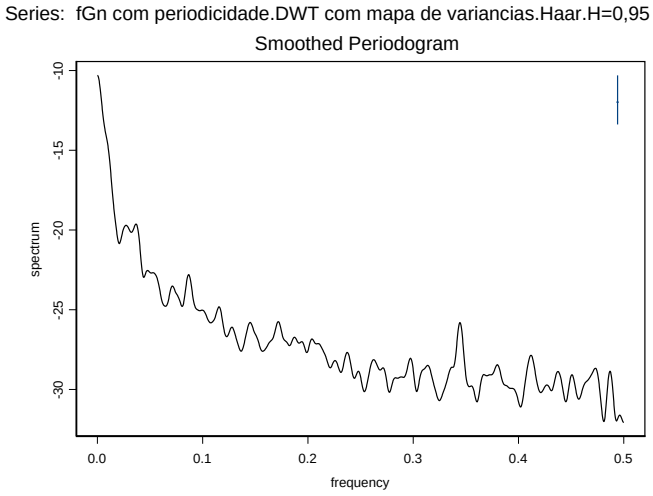
A Fig. 4.13 mostra o sinal, a função de autocorrelação e o periodograma da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude




**(a)**

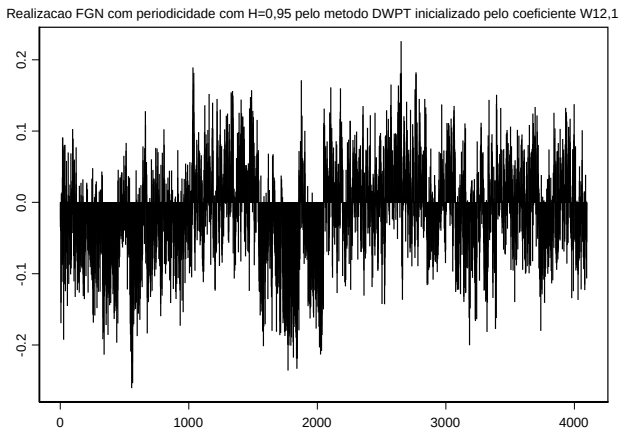



**(b)**

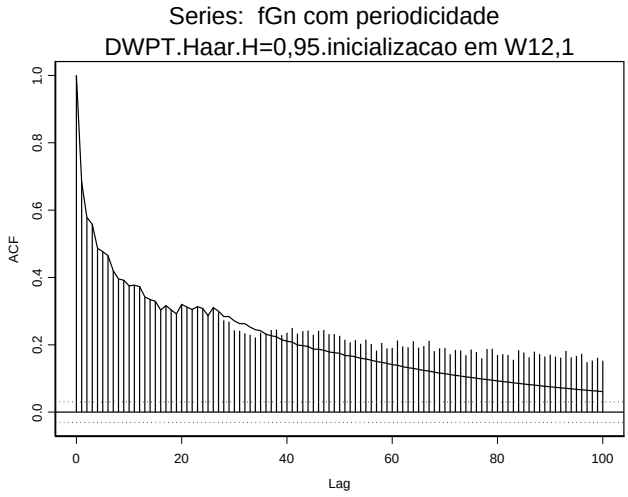



**(c)**

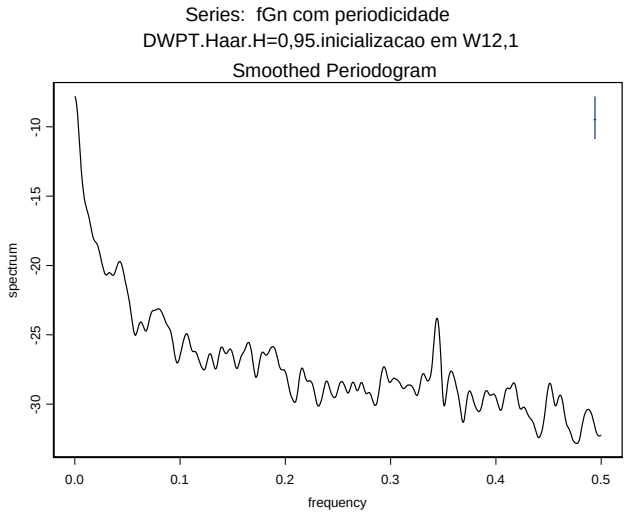
**Figura 4.11:** A Fig. (a) é o sinal da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWT com mapa de variâncias utilizando função de Haar e $\alpha = 0{,}9$ (H = 0,95); A Fig. (b) é a função de autocorrelação da série; A Fig. (c) é o periodograma suavizado da série temporal.




**(a)**




**(b)**




**(c)**

**Figura 4.12:** A Fig. (a) é o sinal da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWPT utilizando função de Haar e $\alpha = 0{,}9$ (H = 0,95); A Fig. (b) é a função de autocorrelação da série; A Fig. (c) é o periodograma suavizado da série temporal.

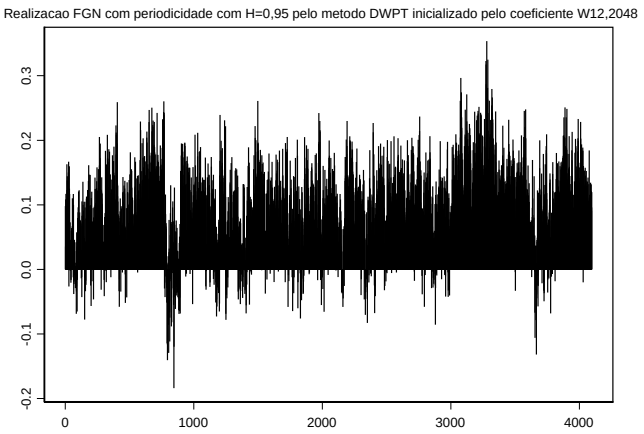
0,008 gerada pelo método DWPT inicializado pelo coeficiente $\mathbf{W}_{12,2048}$. A partir da análise destes gráficos, pode-se observar a presença da periodicidade do cosseno. No gráfico da Fig. 4.13c observa-se o comportamento misto da série LRD com SRD na freqüência 0,35 introduzida pela periodicidade.

#### 4.3.2.3 Coeficiente de inicialização $\mathbf{W}_{12,2867}$

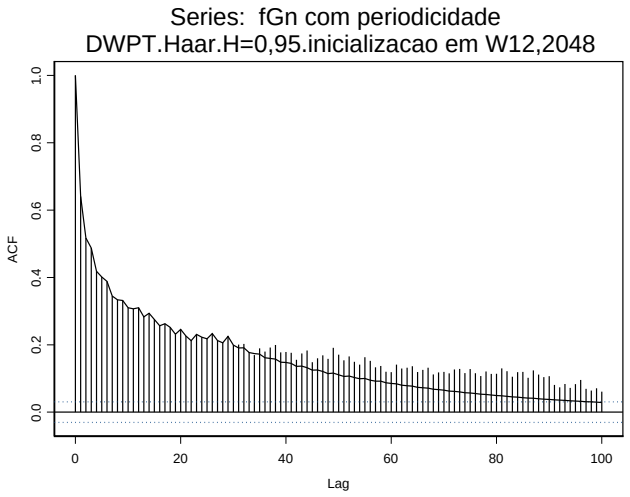
A Fig. 4.14 mostra o sinal, a função de autocorrelação e o periodograma da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 gerada pelo método DWPT inicializado pelo coeficiente $\mathbf{W}_{12,2867}$. A partir da análise destes gráficos, pode-se observar a presença da periodicidade do cosseno. No gráfico da Fig. 4.14c observa-se o comportamento misto da série LRD com SRD na freqüência 0,35 introduzida pela periodicidade.

#### 4.3.2.4 Coeficiente de inicialização $\mathbf{W}_{12,4096}$

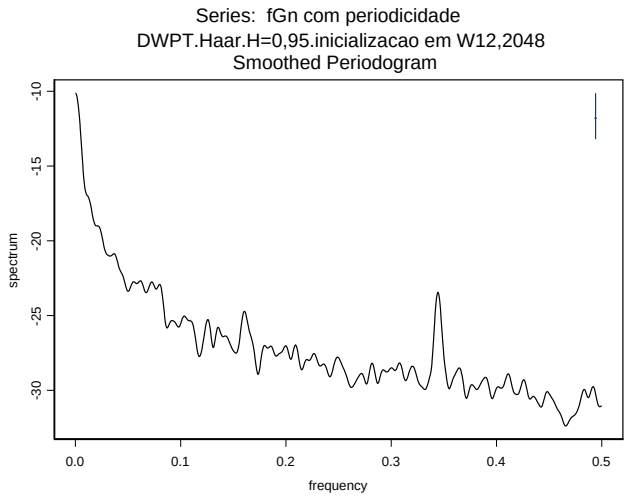
A Fig. 4.15 mostra o sinal, a função de autocorrelação e o periodograma da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 gerada pelo método DWPT inicializado pelo coeficiente $\mathbf{W}_{12,4096}$. A partir da análise destes gráficos, pode-se observar a presença da periodicidade do cosseno. No gráfico da Fig. 4.15c observa-se o comportamento misto da série LRD com SRD na freqüência 0,35 introduzida pela periodicidade.




**(a)**

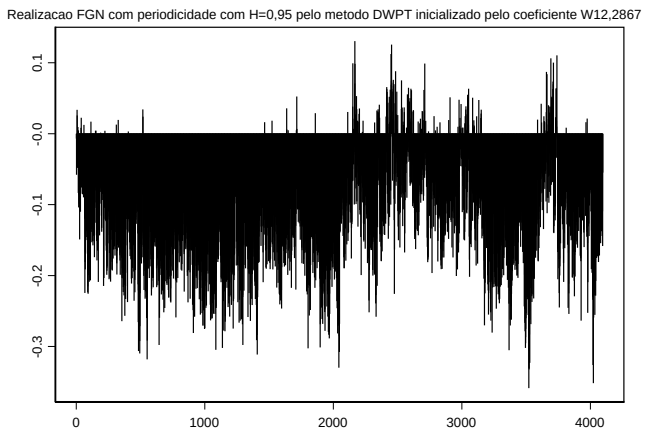



**(b)**

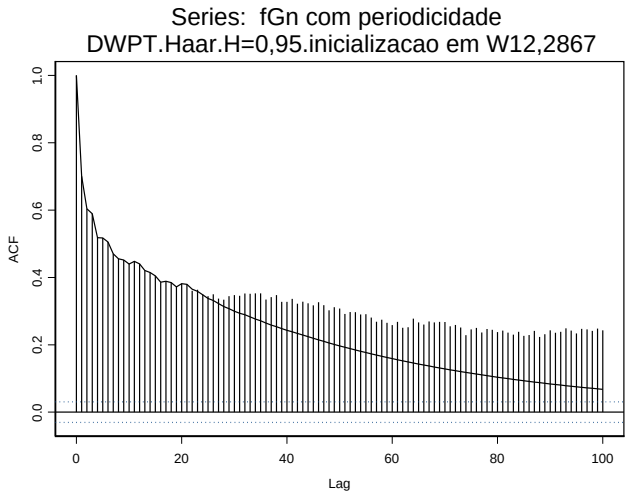



**(c)**

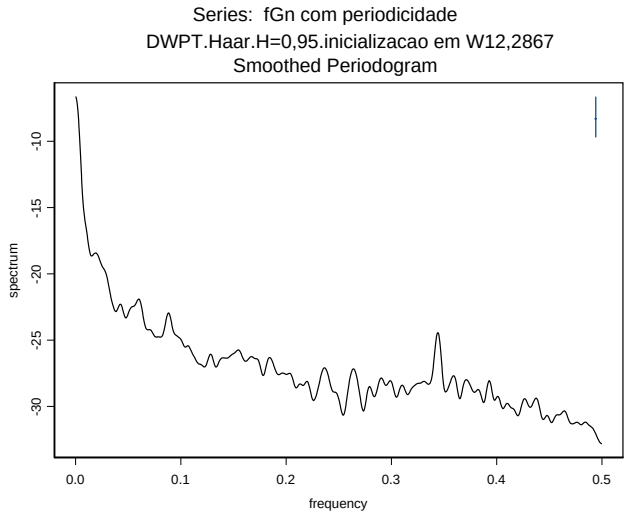
**Figura 4.13:** A Fig. (a) é o sinal da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWPT utilizando função de Haar e $\alpha = 0{,}9$ (H = 0,95); A Fig. (b) é a função de autocorrelação da série; A Fig. (c) é o periodograma suavizado da série temporal.




**(a)**

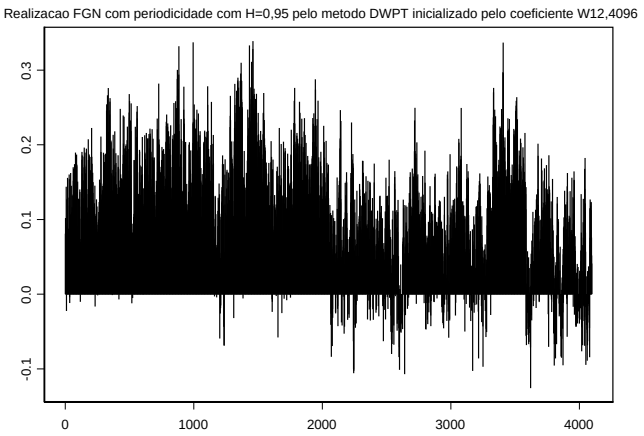



**(b)**

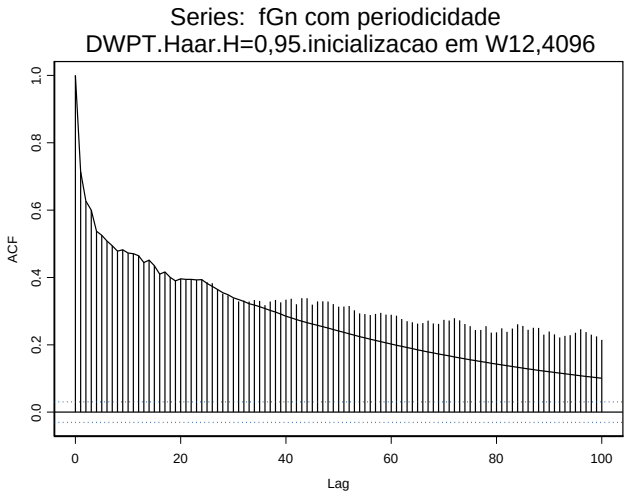



**(c)**

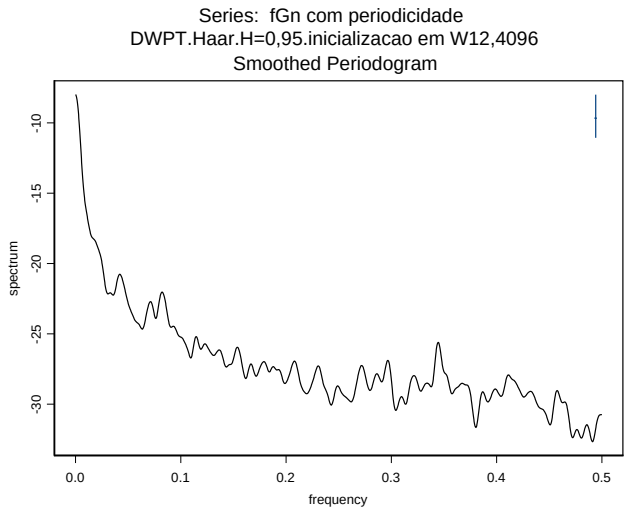
**Figura 4.14:** A Fig. (a) é o sinal da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWPT utilizando função de Haar e $\alpha = 0{,}9$ (H = 0,95); A Fig. (b) é a função de autocorrelação da série; A Fig. (c) é o periodograma suavizado da série temporal.




**(a)**




**(b)**




**(c)**

**Figura 4.15:** A Fig. (a) é o sinal da série temporal de um processo $1/f$ de 4096 pontos com periodicidade de amplitude 0,008 obtida através do método DWPT utilizando função de Haar e $\alpha = 0{,}9$ (H = 0,95); A Fig. (b) é a função de autocorrelação da série; A Fig. (c) é o periodograma suavizado da série temporal.

# 5 Conclusões e Trabalhos Futuros

A geração de tráfego realista é um tema importante de pesquisa em redes para propiciar um método de simulação e avaliação de desempenho quando o objetivo primordial é o de assegurar a QoS fim-a-fim. A utilização de *traces* ou de geradores sintéticos, que não levam em conta a LRD, não permitem reproduzir com fidelidade o comportamento real das redes. Assim sendo, este trabalho desenvolveu um método mais eficiente que explora a flexibilidade de processamento no plano tempo-freqüência para gerar tráfego que apresentem simultaneamente comportamentos LRD, SRD e periodicidade.

Este trabalho iniciou a partir do estudo do trabalho desenvolvido por Mello [1], utilizando o gerador desenvolvido no mesmo para gerar uma série temporal a partir da síntese de realizações aproximadas dos processos aleatórios auto-similares *Fractional Gaussian Noise* (FGN) utilizando o método DWT de Bäckar. O passo seguinte foi implementar a análise da série temporal gerada via DWT a fim de encontrar os coeficientes DWT de todos os níveis dessa série. A partir dessa implementação, foi implementada a análise da série temporal gerada via DWPT a fim de encontrar os coeficientes DWPT de todos os níveis dessa série, a qual foi feita por meio das alterações necessárias no algoritmo da pirâmide de Mallat que foi utilizado na implementação da análise via DWT.

A fim de obter flexibilidade de processamento no plano tempo-freqüência, foi desenvolvido o conceito do mapa de variâncias a partir do estudo do cálculo da variância *wavelet* e da *wavelet packet table* da DWPT. A partir desse conceito, foram implementados os métodos DWT com mapa de variâncias e DWPT para gerar séries temporais a partir da síntese de realizações aproximadas dos processos aleatórios auto-similares FGN com introdução de SRD numa faixa de freqüência desejada.

Também foram implementados os métodos DWT com mapa de variâncias e DWPT para gerar uma série temporal a partir da síntese de realizações aproxi-

madas dos processos aleatórios auto-similares FGN com uma periodicidade devido aos estudos que verificaram na prática a presença de periodicidade em tráfego real [58]

O presente trabalho reproduziu e comparou com sucesso os mecanimos de geração de séries temporais geradas pelos métodos DWT de Bäckar utilizado pelo trabalho de [1], DWT com mapa de variâncias e DWPT, que mostraram-se eficientes para síntese de séries temporais $1/f$. O método de Bäckar não garante que as séries geradas sejam FGN, mas apenas que essas têm comportamento $1/f$ (o que é suficiente para geração de séries temporais para simulação de teletráfego). Ressalta-se que o trabalho de Bäckar só gera séries LRD e não faz a inserção de SRD.

O trabalho de Mello [1] fez a inserção de SRD em dois estágios, filtrando a série temporal gerada pelo método Bäckar .

Neste trabalho foram desenvolvidos os métodos DWT com mapa de variâncias e DWPT que utilizaram o conceito de mapa de variâncias para a inserção de SRD em diferentes faixas de freqüência assim explorando melhor as funcionalidades de processamento no plano tempo-freqüência por meio de *wavelets*.

Esses métodos mostraram-se aplicáveis à geração de séries temporais com DEPs arbitrárias.

Os resultados deste trabalho demostraram que o método DWPT é mais eficiente que o método DWT com mapa de variâncias para a geração de séries com a inserção de SRD por ser mais flexível e prover uma maior flexibilidade de processamento no plano tempo-freqüência.

Trata-se de um trabalho inicial que precisa ser ampliado com a utilização de curvas de ganho SRD mais realistas. Também deverá ser ampliado para geracão de séries MWM e séries com distribuições não gaussianas, por exemplo distribuições $\alpha$-estáveis.

# Referências


[1] MELLO, F. L. de. *Estudo e Implementação de um Gerador de Tráfego com Dependência de Longa Duração*. Tese (Dissertação de Mestrado) — Escola Politécnica, USP, 2006.

[2] RIBEIRO, R. H. R. V. J.; BARANIUK, R. G. Multiscale queueing analysis. *IEEE/ACM Transactions on Networking, vol 14, No. 5*, 2006.

[3] CAPPE, O. et al. Long-range dependence and heavy-tail modeling for teletraffic data. *IEEE Signal Processing Magazine*, v. 19, p. 14–27, May 2002.

[4] PERCIVAL, D. B.; WALDEN, A. T. *Wavelet Methods for Time Series Analysis*. [S.l.]: Cambridge University Press, 2000.

[5] SILVA, G. H. da. *Análise de Causalidade entre Séries Temporais Via Decomposição Wavelet*. Tese (Dissertação de Mestrado) — Escola Politécnica, USP, 2004.

[6] DAUBECHIES, I. *Ten Lectures on Wavelets*. [S.l.]: Capital City Press, 1992.

[7] MA, S.; JI, C. Modeling heterogeneous network traffic in wavelet domain. *IEEE/ACM Transactions on Networking, vol 9, No. 5*, 2001.

[8] PAXSON, V. Fast, aproximate synthesis of fractional gaussian noise for generating self-similar network traffic. *Computer Communication review, v. 27, p. 5-18*, 1997.

[9] PERCIVAL, D. B. Simulating gaussian random processes with specified spectra. *Computer Science and Statistics, vol 24, pp. 534-538*, 1992.

[10] HOSKING, J. R. M. Modeling persistence in hydrological time series using fractional differencing. *Water Resources Research, vol. 20, pp. 1898-1908*, 1984.

[11] SATO, H. Teletraffic technologies in atm networks. *Artech House*, 1994.

[12] SCHWARTZ, M. *Broadband Integrated Networks*. 1st.. ed. [S.l.]: Prentice Hall, 1996.

[13] LELAND, W. E.; WILLINGER, W. et al. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Networking*, 1994.

[14] PAXSON, V.; FLOYD, S. Wide-area traffic: The failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, v. 3, n. 3, p. 226–244, June 1995.

[15] GARRET, M. W.; WILLINGER, W. Analyis, modeling and generation of self-similar vbr vido traffic. *SIGCOM, London, U.K., pp. 269-279*, 1994.

[16] HUANG M. DEVETSIKIOTIS, I. L. C.; KAYE, A. R. Modeling and simulation of self-similar variable bit rate compressed video: A unified approach. *SIGCOM, London, Stanford, CA pp. 114-125*, 1996.

[17] JELENKOVIC, A. A. L. P. R.; SEMRET, N. The effect of multiple time scales and subexponentiality in mpeg video streams on queueing behavior. *IEEE J. Select. Areas Commun., vol. 15, pp. 1052-1071*, 1997.

[18] ANDERSEN, A. T.; NIELSEN, B. F. An application of superpositions of two state markovian sources to the modeling of self-similar behavior. *in proc. IEEE INFOCOM, Kobe, Japan, PP. 196-204*, 1997.

[19] RYU, B. K. *Fractal Network Traffic: From understanding to implications.* Dissertação (Mestrado) — Columbia Univ., NY, 1996.

[20] KRUNZ, M. M.; MAKOWSKI, A. M. Modeling video traffic using $m/g/\infty$ input processes: A compromisse between markovian and lrd models. *IEEE J. Select. Areas Commun., pp. 733-748*, 1998.

[21] LIMA, A. B. *Contribuições à Modelagem de Teletráfego fractal.* Tese (Doutorado) — Escola Politécnica, USP, 2008.

[22] MA, S.; JI, C. Modeling video traffic in the wavelet domain. *IEEE INFOCOM, PP. 201-208*, 1998.

[23] MA, S. *Network Traffic Modeling and Analysis.* Dissertação (Mestrado) — Rensselaer Polytechnic Inst., Troy, NY, 1998.

[24] RIEDI, R. H. et al. A multifractal wavelet model with application to network traffic. *IEEE Transactions on Information Theory*, 1999.

[25] FLOYD, S.; PAXSON, V. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 2001.

[26] PEREIRA, F. M. *Análise de Desempenho da Disciplina de Serviço 'Generalized Processor Sharing' sob Tráfego Auto-Similar.* Dissertação (Mestrado) — Faculdade de Engenharia Elétrica e de Computação, Unicamp, 2002.

[27] DAVIES, R. B.; HARTE, D. S. Tests for hurst effect. *Biometrika, vol. 74, pp. 95-101*, 1987.

[28] FAN, Z.; MARS, P. Self-similar traffic generation and parameter estimation using wavelet transform. *IEEE Global Telecommunications Conference*, 1997.

[29] SHIMIZU, J. A. A multifractal traffic generation method using wavelet and volterra filtering. *Sapporo, Japan [s.n.]*, 2004.

[30] BäCKAR, J.-A. *A Framework for implementing fractal traffic models in real-time.* Dissertação (Mestrado) — SERC, 2000.

[31] EMMA, D.; PESCAPE, A.; VENTRE, G. Analysis and experimentation of an open distributed platform for synthetic traffic generation. In: . [S.l.]: 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004.

[32] EMMA, D. et al. D-itg distributed internet traffic generator. In: . [S.l.]: First International Conference on the Quantitative Evaluation of Systems, 2004.

[33] MANDELBROT, B. B. *The Fractal Geometry of Nature.* New York: WH Freeman, 1977.

[34] KLIMKE, A. *Mandelbrot Set GUI (for MATLAB)*. 2003. Disponível em: <http://matlabdb.mathematik.uni-stuttgart.de/index.jsp>.

[35] CANTOR, G. Über unendliche, lineare punktmannigfaltigkeiten v. *Mathematische Annalen*, v. 21, p. 545–591, 1883.

[36] PEITGEN, H.-O.; JüRGENS, H.; SAUPE, D. *Chaos and Fractals: New Frontiers of Science*. [S.l.]: Springer-Verlag, 1992.

[37] FELDMANN, A.; GILBERT, A. C.; WILLINGER, W. Data networks as cascades: investigating the multifractal nature of internet wan traffic. *Computer Communication review*, v. 28, n. 4, p. 42–55, 1998.

[38] PARK, C. et al. Long-range dependence in a changing internet traffic mix. *Computer Networks*, v. 48, p. 401–422, 2005.

[39] HURST, H. E. Long-term storage capacity of reservoirs. *Trans. Am. Soc. Civil Engineers*, v. 116, p. 770–799, 1951.

[40] BERAN, J. *Statistics for Long-Memory Processes*. [S.l.]: Chapman & Hall, 1994.

[41] RIEDI, R.; VéHEL, J. L. *Multifractal properties of TCP traffic: a numerical study, Tech. Rep. 3129*. [S.l.], 1997. Disponível em: <http://citeseer.ist.psu.edu/riedi97multifractal.html>.

[42] GILBERT, A. C.; WILLINGER, W.; FELDMANN, A. Scaling analysis of conservative cascades, with applications to network traffic. *IEEE Transactions on Information Theory*, v. 45, n. 3, p. 971–991, April 1999. Special Issue on "Multiscale Statistical Signal Analysis and its Applications".

[43] RIBEIRO, V. et al. Simulation of nonGaussian long-range dependent traffic using wavelets. In: *SIGMETRICS'99*. [S.l.: s.n.], 1999.

[44] GRANGER, C. W. J.; JOYEUX, R. An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis*, v. 1, p. 15–29, Oct. 1980.

[45] HOSKING, J. R. M. Fractional differencing. *Biometrika*, v. 68, p. 165–176, Oct. 1981.

[46] JAIN, R. *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Simulation, and Modeling*. [S.l.]: John Wiley & Sons, Inc., 1991.

[47] TAQQU, M. S. Fractional brownian motion and long-range dependence. In: DOUKAN, P.; OPPENHEIM, G.; TAQQU, M. S. (Ed.). *Theory and Applications of Long Range Dependence*. Boston, MA: Birkhäuser, 2003.

[48] MANDELBROT, B. B.; NESS, J. V. Fractional brownian motions, fractional noises and applications. *SIAM Rev.*, v. 10, p. 422–437, Feb. 1968.

[49] PETROPULU, A. P.; YANG, X. Self-similarity and long-range dependence through the wavelet lens. In: BARNER, K. E.; ARCE, G. R. (Ed.). *Nonlinear Signal and Image Processing - Theory, Methods, and Applications*. [S.l.]: CRC Press, 2004.

[50] MELLO, F. L. de et al. Geração de séries auto-similares Gaussianas via wavelets para uso em simulações de tráfego. *IEEE Latin America Transactions*, v. 5, n. 1, Março 2007.

[51] STARK, H.; WOODS, J. W. *Probability and Random Processes with Applications to Signal Processing*. 3rd. ed. Upper Saddle River, NY: Prentice Hall, 2002.

[52] ABRY, P.; VEITCH, D. Wavelet analysis of long-range dependent traffic. *IEEE Transactions on Information Theory, v. 4, n. 1, p. 2-15*, 1998.

[53] MALLAT, S. G. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 11, p. 674-693*, 1989.

[54] WICKERHAUSER, M. V. *Adapted Wavelet Analysis from Theory to Software*. [S.l.]: Wellesley, 1994.

[55] KAPLAN, L. M.; KUO, C.-C. J. Fractal estimation from noisy data via discrete fractional gaussian noise (dfgn) and the haar basis. *IEEE Transactions on Signal Processing*, v. 12, p. 3554–3562, 1993.

[56] FLANDRIN, P. On the spectrum of fractional brownian motions. *IEEE Transactions on Information Theory*, v. 35, p. 197–199, 1989.

[57] FLANDRIN, P. Wavelet analysis and synthesis of fractional brownian motion. *IEEE Transactions on Information Theory*, v. 38, n. 2, p. 910–917, 1992.

[58] STOEV MURAD S. TAQQU, C. P. J. M. S.

[59] VEITCH, D.; BäCKAR, J.-A. et al. On-line generation of fractal and multifractal traffic. *PAM2000 Workshop on Passive and Active Networking, Hamilton, New Zeland*, 2000.

[60] TSAY, R. S. *Analysis of Financial Time Series*. [S.l.]: Wiley Interscience, 2005.

[61] PERCIVAL, D. B.; WALDEN, A. T. *Spectral Analysis for Physical Applications*. New York, NY: Cambridge, 1993.

[62] ZIVOT, E.; WANG, J. *Modeling Financial Time Series with S-PLUS*. [S.l.]: Springer, 2003.

# Apêndice A – Códigos MATLAB - Síntese pelo método DWT de Bäckar

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Modelos:
% 1 - distr. IID para aprox. fGn
% 2 - MWM
% 3 - distr. AR(1) para aprox. fGn
% 9 - Reconstrucao Analise Matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametros:
% TopLev = 3; % Nivel menos refinado de todos
% NrOfVM = 2; %numero de Vanishing Moments da Wavelet de Daubechie
% modelo = 1; % 1 - distr. IID para aprox. fGn
% Lm = 16; % numero de amostras que desejamos obter no final - m=modelo
% alpha = 0.5; % parametro que controla decaimento da potencia dos
%              % coeficientes de detalhe
% p = 0.05; % parametro que seta var[R] = 1/(2p+1). p nao poderia ser nulo (p>0),
%           % mas faremos com que p = 0 signifique que devera ser feito o
%           % calculo de p em funcao de H = (alpha+1)/2.
%           % Nessa situacao, para j->0 (escala de alta resolucao):
%           % p = (2^(2H-1) - 1) / (2 - 2^(2H-1)), onde alpha = 2H-1
% phi = 0.8 % parametro do processo AR(1) (coeficiente filtro)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% uso: [Data, nome_arquivo]=Generate(TopLev, NrOfVM, modelo, varargin)
%      ou [Data]=Generate(TopLev, NrOfVM, 9, varargin) - modelo 9
% onde: 1- varargin sera dois parametros:      "Lm, alpha"                            para o modelo 1
%       2- e varargin sera dois parametros:    "alpha, p"                             para o modelo 2
%       3- e varargin sera tres parametros:    "Lm, alpha, phi"                       para o modelo 3
%       9- e varargin serao cinco parametros:  "Lm, vetor_coefs (C), L, TopLev, wname"  para o modelo 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Data, nome_arquivo] = Generate(TopLev, NrOfVM, modelo, varargin)

%TopLev = 3; % Nivel menos refinado de todos
%NrOfVM = 2; %numero de Vanishing Moments da Wavelet de Daubechie
%modelo = 1; % 1 - distr. IID para aprox. fGn

%Lm = 16; % numero de amostras que desejamos obter no final - m=modelo


% Set Daubechies wavelet name.
%wname = 'db2';
N = int2str(NrOfVM); %N e uma string
wname = 'db'; %falta concatenar com N para obter dbN
wname = strcat(wname,N)
```

```
%hnorm=h*sqrt(2)

% Compute the corresponding scaling filter.
g = dbwavf(wname);
% uso da funcao dbaux seria mais simples pois aceitaria N como inteiro em vez de exigir a string dbN
g = g * sqrt(2)
Lf = length(g) % precisa ser igual a 2N
h = fliplr(g); h = (-1).^(0:Lf-1).*h
% representando a formula 75b do livro de wavelets indicado pelo prof. Baccala
% de qq modo a formula acima condiz com
% os coeficientes apresentados pelo pacote Wavelet do Matlab

% aqui vamos chamar a funcao Model que de acordo com um modelo,
% coeficientes dos filtros, TopLev e NrOfVM (outros parametros passados opcionalmente) ira produzir
% a sintese ou aproximacao na escala 0.

[Data, nome_arquivo] = Model(TopLev, NrOfVM, h, g, modelo, varargin{:});

%indices = Data{0+(1)}.kp(1):Data{0+(1)}.kp(2)
%plot(indices,Data{0+(1)}.app)
%stem(Data{1}.kp(1):Data{1}.kp(2),Data{1}.app)
```

```
function [Data,nome_arquivo] = Model(TopLev, NrOfVM, h, g, modelo, varargin)
%Model.m esta na pagina 84

%modelo = 1; % 1 - distr. IID para aprox. fGn
%modelo = 2; % 2 - MWM
%TopLev = 3; % Nivel menos refinado de todos
%NrOfVM = 2; %numero de Vanishing Moments da Wavelet de Daubechie

% Lm deve vir atraves do varargin:
% Lm = 16; % numero de amostras que desejamos obter no final - m=modelo
if modelo ~= 2, % nao sendo o MWM um dos dados de entrada passa a ser o Lm:
 Lm = varargin{1};
 % deletando a celula {1} para que a celula {2} (alpha) passe a ser
 % a celula {1}:
 varargin(1) = [];
 % o proximo dado de entrada passa a ser o alpha ficando armazenado em varargin{1}
 %(so nao sera alpha caso seja o modelo 9)
 switch modelo
  case 1,
   %gera um nome de arquivo para armazenar a saida da serie posteriormente:
   nome_arquivo
   = ['fGn_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(To
  case 3,
   %gera um nome de arquivo para armazenar a saida da serie posteriormente:
   nome_arquivo =['fGn_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_phi(',num2str(varargin{2}),')_NrOfV
  case 9,
  %gera um nome de arquivo vazio:
   nome_arquivo = [];
 end

else %2 - MWM:
 if NrOfVM ~= 1,
  error('Modelo 2 (MWM) exige que se escolha NrOfVM = 1 para usar a Wavelet de Haar');
 end
  Lm = 2^TopLev; %parte de um unico coeficiente de aprox. U0,0 (massa)
  % o Lm passa a ser controlado pelo numero de refinamentos
  % para o MWM o varargin{1} sera o alpha.

  %gera um nome de arquivo para armazenar a saida da serie MWM posteriormente:
  nome_arquivo = ['MWM_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_p(',num2str(varargin{2}),')_TopLev(
end

Lf = 2*NrOfVM; %comprimento do filtro
kn = [0 Lm-1];
kp = kn;
Data{0+(1)}.kp = kn;
%ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0)
kc = kn;
%ira armazenar dois extremos da serie temporarios para a proxima escala
kd = kn;
%ira armazenar dois extremos da serie consolidados para a proxima escala
Data{0+(1)}.kd = kn;

%efetuando o calculo dos coeficientes em cada um dos extremos e que serao
%necessarios:
for j=1:TopLev,
```

```
 %kd(1) e o kminimo da escala anterior
 %kd(2) e o kmax da escala anterior
 %Lf - 1 = 2N - 1
 kc = [kd(1)-(Lf-1) kd(2)];
 kd = fix (kc/2);
 % fix arredonda para zero... e na verdade deveria ser arredonda para baixo, nao? (floor)
 % kd agora passou a conter os indices de kc, porem divididos por 2:
 %kd = [fix(kmin/2 - N + 1/2)) fix(kmax/2)] - isto parece, embora nao exatamente, com
 %as formulas 7.9 e 7.10 onde kd passa a representar uma escala menos
 %refinada. As diferencas estao no fix / floor e no 1/2 / 1

 kn = floor([0 kn(2)-(Lf-1)]/2);
 kp = floor(kp/2) - [(NrOfVM - 1) 0];

 Data{j+(1)}.kd = kd;
 Data{j+(1)}.kp = kp;
 Data{j+(1)}.kn = kn;
end

%Ate aqui calculamos os indices kmin e kmax de cada escala
%Alem do numero final de coeficientes desejado e do numero de refinamentos (Toplevel) que devem ser passados
%como input para o algoritmo, ainda nos resta calcular qtos coeficientes sao necessarios no nivel menos
%refinado para dar inicio ao algoritmo de %reconstrucao. Obviamente este numero pode ser extraido de:
%Data{TopLev+(1)}.kp que neste instante estaria no vetor kp.

%calculo dos coeficientes da funcao de escala na escala menos refinada:
 Data{TopLev+(1)}.app  = appProcess(diff(kp)+1, modelo, varargin{:});
 % passamos junto parametros adicionais dependentes do modelo (varargin) que devem ser tratados
 % dentro da appProcess.

 for j=TopLev:-1:1,
  %estamos calculando [Data{j+(1)}.det] e qdo for modelo MWM tb. o
  %valor do parametro adicional p em cada escala:
  [Data] = detProcess(j,TopLev,Data,modelo,varargin{:});% calcula o detalhe da escala atual
  Data{j-1+(1)}.app = Recon(j,NrOfVM,Data,h,g);
  %reconstroi os coef. de aprox. do nivel imediatamente inferior a escala atual ???
 end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%interna:
function approxs = appProcess(n, modelo, varargin)
%function approxs = appProcess(n, modelo, testevariaveis)
switch modelo
 case 1, approxs = zeros(1,n); %tudo zero para aprox. do fGn
 case 2, approxs = 4+randn(1,n);
 %Gaussiana com media = 2; var = 1; -> mudamos para gaussiana com media 4.
 case 3, approxs = zeros(1,n); %tudo zero para aprox. do fGn
 case 9,
 %varargin{.} = Lm, vetor_coefs, TopLev, wname
  vetor_coefs = varargin{1};
  L=varargin{2};
  TopLev = varargin{3};
  wname = varargin{4};
  coefsA = appcoef(vetor_coefs,L,wname,TopLev);
  if length(coefsA) == n ,
     approxs = coefsA; % no nivel mais alto os coeficientes de aproximacao
```

```
    else
      warning('Numero de coeficientes da dwt nao bate com o necessario para reconstituir o sinal');
      approxs = coefsA(1:n);
    end
  otherwise, disp('Modelo nao implementado'); approxs = [];
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%interna:
function [Data] = detProcess(level, TopLev, Data, modelo, varargin)
  %estamos calculando [Data{j+(1)}.det] e qdo for modelo MWM tb. o
  %valor do parametro adicional p em cada escala:

  %teste varargin:
  %a = varargin{1}
  %b = varargin{2}
  %c = varargin{3}

  % todos os meus modelos (menos o 9) demandam um parametro alpha que neste momento
  % precisa ser necessariamente a primeira celula do cell array varargin:
  if modelo ~= 9,
    alpha = varargin{1};
    % nao vamos testar alpha por enquanto pois vamos assumir um valor cte para todos os levels (escalar)

     % Lembrar que:
     % Var[aX] = a^2 Var[X] (e se a media de X nao for nula????) -> lembrar
     % de estocasticos.
     % queremos que a^2 = 2^( (level - TopLev) * alpha ) - ou seja, va
     % decaindo de escala para escala com um fator 2^alpha - qto. melhor a
     % resolucao de uma escala queremos menor variancia nos detalhes (cai a
     % energia dos detalhes???)
     % a^2 = 2^( (level - TopLev) * alpha ) =>
     % a = 2^( (level - TopLev) * alpha/2 )
     % a = ctrvariance
     ctrvariance = 2^( (level - TopLev) * alpha/2 )
  end

  n = diff( Data{level+(1)}.kp ) + 1;

  switch modelo
   case 1, Data{level+(1)}.det = ctrvariance*randn(1,n);
   %iid com a variancia do processo caindo com o fator ilustrado acima em funcao de um parametro alpha
   %randn(1,n) sao n realizacoes de uma distribuicao normal de media
   %nula e variancia 1.0. Logo teremos variancia dos detalhes < 1 para levels ~= de TopLev
   case 2, %MWM:
     p = varargin{2};
     if (level == TopLev) & (p == 0),
     % Calcular p em cada uma das escalas partindo da de melhor resolucao (0):
       %p(0) ou p para j -> -inf conforme paper MWM:
       p = (2^alpha - 1)/(2 - 2^alpha);
       Data{0+(1)}.p = p;
       % todos os outros valores de p de acordo com a relacao
       % (8.8) em outro formato:
       for i = 1:TopLev,
         p = (2*p+1)/2^alpha - 1;
         Data{i+(1)}.p = p;
```

```
      end
    elseif (level == TopLev) & (p > 0),
      Data{TopLev+(1)}.p = p;
      for i = TopLev-1:-1:0,
       p = (2^alpha*(p+1)-1)/2;
       Data{i+(1)}.p = p;
      end
      p = Data{TopLev+(1)}.p;
    elseif level ~= TopLev,
     % p ja deve estar calculado para todas as escalas:
      p = Data{level+(1)}.p;
    end
    % sorteio da distribuicao beta:
    % R = betarnd(A,B,m,n) generates an m-by-n matrix containing
    % random numbers from the beta distribution with parameters A and B.:
    % no Matlab: they are nonzero only on the interval (0 1).
    R01 = betarnd(p,p,1,n);
    % queremos obter a = 2 x - 1 para obter imagem no intervalo (-1,1), logo:
    R = 2.*R01-1;
    Data{level+(1)}.det = R.*Data{level+(1)}.app;
  case 3, phi = varargin{2};
    % AR(1) com a variancia do processo caindo com o fator ilustrado acima em funcao de um parametro alpha
    %randn(1,n) sao n realizacoes de uma distribuicao normal de media
    %nula e variancia 1.0.
     inject = ctrvariance*randn(1,n);
     Data{level+(1)}.det(1) = phi*0 + inject(1);
     for i=2:n,
       Data{level+(1)}.det(i) = phi*Data{level+(1)}.det(i-1) + inject(i);
     end
     clear inject;
  case 9,
     %varargin{.} = Lm, vetor_coefs, TopLev, wname
     vetor_coefs = varargin{1};
     L = varargin{2};
     coefsD = detcoef(vetor_coefs,L,level);
     if length(coefsD) == n ,
        Data{level+(1)}.det = coefsD;
        % no nivel level os coeficientes de detalhe gerados pela decomposicao do Matlab
     else
        warning('Numero de coeficientes da dwt nao bate com o necessario para reconstituir o sinal');
        Data{level+(1)}.det = coefsD(1:n);
     end

   otherwise, error(['Modelo ', int2str(modelo), ' nao implementado!!!']); %disp('Modelo nao implementado');
end
```

```
% Reconstrucao do nivel imediatamente inferior:

function approxs = Recon(level, NrOfVM, Data, h, g) % h will be high-pass filter coefficients.
  Lf = length(h);

  kp = Data{level + (1)}.kp; % coeficiente minimo e maximo na escala level

  %A seguir estao implementadas as equacoes 7.11, 7.12, 7.13 e 7.14 da
  %pagina 49:
  %%%%%%%%%%%%%%%%%

  ku = 2*kp; % coeficiente minimo e maximo na escala level apos fazermos o upsampling

  kc = ku + [0 Lf-1]; % coeficiente minimo e maximo gerados na escala level-1 logo apos a convolucao

  %%%%%%%%%%%%%%%%%%

  kp = Data{level-1 + (1)}.kp; % coeficiente minimo e maximo na escala level-1 (sao os que vamos manter)

  % coeficientes de approximacao e detalhe na escala level que serao
  % upsampled e convoluidos com os coeficientes dos filtros apropriados:
  appro =  Data{level + (1)}.app;
  detail =  Data{level + (1)}.det;

  % dyadup e do pacote de wavelets do Matlab (wavelet toolbox) e faz
  % exatamente o upsampling
  appup = dyadup(appro,0);
  detup = dyadup(detail,0);

  % A convolucao:
  appconv = conv(appup,g); % coefc. de approximacao convoluidos com os coeficientes do filtro passa-baixas
  detconv = conv(detup,h); % coefc. de detalhe convoluidos com os coeficientes do filtro passa-altas

  % indices = [dif1+(1):dif2+(1)], onde dif1 e a diferenca entre o kmindesejado e
  % kminconv resultando no shift que devemos dar no vetor inicial.
  % ja dif2 sera o maximo que queremos obter - o indice inicial do vetor
  % apos convolucao:
  indices = [kp(1)-kc(1)+(1):kp(2)-kc(1)+(1)];

  appro = appconv(indices) + detconv(indices);

  approxs = appro;
```

# Apêndice B – Códigos MATLAB - Síntese pelo método DWT com mapa de variâncias

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Modelos:
% 1 - distr. IID para aprox. fGn
% 2 - MWM
% 3 - distr. AR(1) para aprox. fGn
% 9 - Reconstrucao Analise Matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Parametros:
% TopLev = 3; % Nivel menos refinado de todos
% NrOfVM = 2; %numero de Vanishing Moments da Wavelet de Daubechie
% modelo = 1; % 1 - distr. IID para aprox. fGn
% Lm = 16; % numero de amostras que desejamos obter no final - m=modelo
% alpha = 0.5; % parametro que controla decaimento da potencia dos
%              % coeficientes de detalhe
% p = 0.05; % parametro que seta var[R] = 1/(2p+1). p nao poderia ser nulo (p>0),
%           % mas faremos com que p = 0 signifique que devera ser feito o
%           % calculo de p em funcao de H = (alpha+1)/2.
%           % Nessa situacao, para j->0 (escala de alta resolucao):
%           % p = (2^(2H-1) - 1) / (2 - 2^(2H-1)), onde alpha = 2H-1
% phi = 0.8 % parametro do processo AR(1) (coeficiente filtro)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% uso: [Data, nome_arquivo]=Generate(TopLev, NrOfVM, modelo, varargin)
%      ou [Data]=Generate(TopLev, NrOfVM, 9, varargin) - modelo 9
% onde: 1- varargin sera dois parametros:      "Lm, alpha"                           para o modelo 1
%       2- e varargin sera dois parametros:    "alpha, p"                            para o modelo 2
%       3- e varargin sera tres parametros:    "Lm, alpha, phi"                      para o modelo 3
%       9- e varargin serao cinco parametros:  "Lm, vetor_coefs (C), L, TopLev, wname"  para o modelo 9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [Data,mapa,DataSRD,mapaSRD,DataSRD2,mapaSRD2,nome_arquivo,nome_arquivo_1,nome_arquivo_2,nome_arquivo_3

%precisamos obter os coeficientes h (funcao de escala) e g (funcao wavelet)

% Set Daubechies wavelet name.
%wname = 'db2';
N = int2str(NrOfVM); %N e uma string
wname = 'db'; %falta concatenar com N para obter dbN
wname = strcat(wname,N)
```

```
% Compute the corresponding scaling filter.
g = dbwavf(wname);
% uso da funcao dbaux seria mais simples pois aceitaria N como inteiro em vez de exigir a string dbN
g = g * sqrt(2)
Lf = length(g) % precisa ser igual a 2N
h = fliplr(g); h = (-1).^(0:Lf-1).*h
% de qq modo a formula acima condiz com
% os coeficientes apresentados pelo pacote Wavelet do Matlab

[Data,mapa,DataSRD,mapaSRD,DataSRD2,mapaSRD2,nome_arquivo,nome_arquivo_1,nome_arquivo_2,nome_arquivo_3,nome_arq

if (exist('nome_arquivo')==1),
 if length(nome_arquivo)>1,
        for j=1:+1:TopLev+1,
        Data{j}.app;
        ans';
        nome_arquivo=[nome_arquivo,'_nível',int2str(j),'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo);
        eval(comando2ex);
        nome_arquivo = ['fGn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{2}),')_NrOfVM(',int2str(NrOfVM),')_
        end
    end
end

if (exist('nome_arquivo_1')==1),
  if length(nome_arquivo_1)>1,
     for j=1:+1:TopLev+1,
         mapa{j}=[mapa{j,1}];
     for n=2:+1:2^(j-1),
         mapa{j}=[mapa{j} mapa{j,n}];
     end
   mapa{j};
   ans';
   nome_arquivo_1=[nome_arquivo_1,'_nível',int2str(j),'.txt'];
   nome_arquivo=[nome_arquivo,'.txt'];
   comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_1);
   eval(comando2ex);
   nome_arquivo_1 = ['mapa_Variancias_fgn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{2}),')_NrOfVM(',int2st
end
end
end

if (exist('nome_arquivo_2')==1),
    if length(nome_arquivo_2)>1,
        for j=1:+1:TopLev+1,
        DataSRD{j}.app;
        ans';
        nome_arquivo_2=[nome_arquivo_2,'_nível',int2str(j),'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_2);
        eval(comando2ex);
        nome_arquivo_2 = ['fGn1SRD_(',int2str(Lm),')pts_alpha(',num2str(varargin{2}),')_NrOfVM(',int2str(NrOfVM
        end
    end
end

if (exist('nome_arquivo_3')==1),
  if length(nome_arquivo_3)>1,
```

```
    for j=1:+1:TopLev+1,
         mapaSRD{j}=[mapaSRD{j,1}];
    for n=2:+1:2^(j-1),
            mapaSRD{j}=[mapaSRD{j} mapaSRD{j,n}];
    end
   mapaSRD{j};
   ans';
   nome_arquivo_3=[nome_arquivo_3,'_nível',int2str(j),'.txt'];
  comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_3);
  eval(comando2ex);
  nome_arquivo_3 = ['mapaSRD_Variancias_fgn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{2}),')_NrOfVM(',int2
end
end
end

if (exist('nome_arquivo_5')==1),
    if length(nome_arquivo_5)>1,
        for j=1:+1:TopLev+1,
        DataSRD2{j}.app;
        ans';
        nome_arquivo_5=[nome_arquivo_5,'_nível',int2str(j),'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_5);
        eval(comando2ex);
        nome_arquivo_5 = ['fGn1SRD2_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int2str(NrOfV
        end
    end
end

if (exist('nome_arquivo_6')==1),
  if length(nome_arquivo_6)>1,
    for j=1:+1:TopLev+1,
         mapaSRD2{j}=[mapaSRD2{j,1}];
    for n=2:+1:2^(j-1),
            mapaSRD2{j}=[mapaSRD2{j} mapaSRD2{j,n}];
    end
   mapaSRD2{j};
   ans';
   nome_arquivo_6=[nome_arquivo_6,'_nível',int2str(j),'.txt'];
  comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_6);
  eval(comando2ex);
  nome_arquivo_6 = ['mapaSRD2_Variancias_fgn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int
end
end
end
```

```
function [Data,mapa,DataSRD,mapaSRD,DataSRD2,mapaSRD2,nome_arquivo,nome_arquivo_1,nome_arquivo_2,nome_arquivo_3

    % Lm deve vir atraves do varargin:
        Lm = varargin{1};
        % deletando a celula {1} para que a celula {2} (alpha) passe a ser
        % a celula {1}:
        varargin(1) = []; % o proximo dado de entrada passa a ser o alpha ficando armazenado em varargin{1} (so
                %gera um nome de arquivo para armazenar a saida da serie posteriormente:
                 nome_arquivo = ['fGn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int2str(Nr
                 nome_arquivo_1 = ['mapa_Variancias_fgn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_N
                 nome_arquivo_2 = ['fGn1SRD_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int2s
                 nome_arquivo_3 = ['mapaSRD_Variancias_fgn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),'
                 nome_arquivo_4 = ['fGn1Det_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int2s
                 nome_arquivo_5 = ['fGn1SRD2_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int2
                 nome_arquivo_6 = ['mapaSRD2_Variancias_fgn1_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),

  Lf = 2*NrOfVM; %comprimento do filtro
kn = [0 Lm-1]; %aqui dois extremos da serie que estamos interessados no nivel mais refinado - ira armazenar som
kp = kn; % ira armazenar extremos contendo non-polluted e polluted coefficients. Todos os coeficientes entre es
Data{0+(1)}.kp = kn; %ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0)
    DataSRD{0+(1)}.kp = kn; %ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0)
    DataSRD2{0+(1)}.kp = kn; %ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0
kc = kn;  %ira armazenar dois extremos da serie temporarios para a proxima escala
kd = kn; %ira armazenar dois extremos da serie consolidados para a proxima escala (desnecessario ja que temos o
Data{0+(1)}.kd = kn;
DataSRD{0+(1)}.kd = kn;
DataSRD2{0+(1)}.kd = kn;

%efetuando o calculo dos coeficientes em cada um dos extremos e que serao
%necessarios:
for j=1:TopLev,
        %kd(1) e o kminimo da escala anterior
        %kd(2) e o kmax da escala anterior
        %Lf - 1 = 2N - 1
        kc = [kd(1)-(Lf-1) kd(2)];
        kd = fix (kc/2); % fix arredonda para zero... e na verdade deveria ser arredonda para baixo, nao? (floo
        % kd agora passou a conter os indices de kc, porem divididos por 2:
        %kd = [fix(kmin/2 - N + 1/2)) fix(kmax/2)] - isto parece, embora nao exatamente, com
        %as formulas 7.9 e 7.10 onde kd passa a representar uma escala menos
        %refinada. As diferencas estao no fix / floor e no 1/2 / 1

        kn = floor([0 kn(2)-(Lf-1)]/2); %[0 floor(kmax/2 - N + 1/2)]!!! -> nao temos mais nenhuma formula que i
        kp = floor(kp/2) - [(NrOfVM - 1) 0];


        Data{j+(1)}.kd = kd
        Data{j+(1)}.kp = kp
        Data{j+(1)}.kn = kn
        DataSRD{j+(1)}.kd = kd
        DataSRD{j+(1)}.kp = kp
        DataSRD{j+(1)}.kn = kn
        DataSRD2{j+(1)}.kd = kd
        DataSRD2{j+(1)}.kp = kp
        DataSRD2{j+(1)}.kn = kn

    end
```

```
    %Ate aqui calculamos os indices kmin e kmax de cada escala
%Alem do numero final de coeficientes desejado e do numero de refinamentos (Toplevel) que devem ser passados co
%algoritmo, ainda nos resta calcular qtos coeficientes sao necessarios no nivel menos refinado para dar inicio
%reconstrucao. Obviamente este numero pode ser extraido de:
%Data{TopLev+(1)}.kp que neste instante estaria no vetor kp.


    %%Calculo do mapa de variâncias
    for j=TopLev+1:-1:1,
    f = 0.5/2^(j-1)
    NJ= diff( Data{j}.kp ) + 1
    Cs=1
    GSRD1=10
    GSRD2=1
    GSRD3=5

    for n=1:+1:2^(j-1),
      alpha = varargin{1};
      n
      if j==TopLev+1,
            mapa{j,n} = (2^(j-1))*(Cs/(1-alpha)*f^(1-alpha)*(n^(1-alpha)-(n-1)^(1-alpha)));
      else
          if mod(n,2) == 0,
             m=2*n;
             mapa{j,n} = (mapa{j+1,m-1}+mapa{j+1,m})/2;
          else
             m=(2*n-1);
             mapa{j,n} = (mapa{j+1,m}+mapa{j+1,m+1})/2;
          end
          %mapa{j,n} = mapa{j+1,n}*2^(-alpha);
      end
    end
    %mapa normalizado
    if j==TopLev+1,
        a=mapa{j,2}
        for n=1:+1:2^(j-1),
            mapa{j,n} = mapa{j,n}/a;
        end
    end

    for n=1:+1:2^(j-1),
      if j==TopLev+1,
      if n>=nsrdd & n<=nsrdt,
        mapaSRD2{j,n}=mapa{j,n}*GSRD1;
        for n=nsrdd:+1:nsrdd+((nsrdt-nsrdd)/3),
            mapaSRD{j,n}=mapa{j,n}*GSRD1;
        end

        for n=nsrdd+((nsrdt-nsrdd)/3)+1:+1:nsrdd+2*((nsrdt-nsrdd)/3),
            mapaSRD{j,n}=mapa{j,n}*GSRD3;
        end

        for n=nsrdd+2*((nsrdt-nsrdd)/3)+1:+1:nsrdd+3*((nsrdt-nsrdd)/3),
            mapaSRD{j,n}=mapa{j,n}*GSRD1;
        end
```

```
      else
        mapaSRD{j,n}=mapa{j,n}*GSRD2;
        mapaSRD2{j,n}=mapa{j,n}*GSRD2;

      end
      else
            if mod(n,2) == 0,
               m=2*n;
               mapaSRD{j,n} = (mapaSRD{j+1,m-1}+mapaSRD{j+1,m})/2;
               mapaSRD2{j,n} = (mapaSRD2{j+1,m-1}+mapaSRD2{j+1,m})/2;
            else
               m=(2*n-1);
               mapaSRD{j,n} = (mapaSRD{j+1,m}+mapaSRD{j+1,m+1})/2;
               mapaSRD2{j,n} = (mapaSRD2{j+1,m}+mapaSRD2{j+1,m+1})/2;
            end
      end
    end
    end


%calculo dos coeficientes da funcao de escala na escala menos refinada:
[Data{TopLev+(1)}.app, DataSRD{TopLev+(1)}.app, DataSRD2{TopLev+(1)}.app]  = appProcess(diff(kp)+1, modelo, var

    for j=TopLev:-1:1,

        [Data,mapa,DataSRD,mapaSRD,DataSRD2,mapaSRD2] = detProcess(j,TopLev,Data,mapa,DataSRD,mapaSRD,DataSRD2,
        Data{j-1+(1)}.app = Recon(j,NrOfVM,Data,h,g); %reconstroi os coef. de aprox. do nivel imediatamente inf
        DataSRD{j-1+(1)}.app = ReconSRD(j,NrOfVM,DataSRD,h,g); %reconstroi os coef. de aprox. do nivel imediata
        DataSRD2{j-1+(1)}.app = ReconSRD2(j,NrOfVM,DataSRD2,h,g); %reconstroi os coef. de aprox. do nivel imedi
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%interna:
function [approxs, approxsSRD, approxsSRD2] = appProcess(n, modelo, varargin, mapa, mapaSRD, mapaSRD2,TopLev)

                approxs = zeros(1,n); %tudo zero para aprox. do fGn
                approxsSRD = zeros(1,n); %tudo zero para aprox. do fGn
                approxsSRD2 = zeros(1,n); %tudo zero para aprox. do fGn


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%interna:
function [Data,mapa,DataSRD,mapaSRD,DataSRD2,mapaSRD2] = detProcess(level, TopLev, Data,mapa,DataSRD,mapaSRD,Da

        alpha = varargin{1}; % nao vamos testar alpha por enquanto pois vamos assumir um valor cte para todos o

        ctrvariance = (mapa{(level+1),2})^(0.5);
        ctrvarianceSRD = (mapaSRD{(level+1),2})^(0.5);
        ctrvarianceSRD2 = (mapaSRD2{(level+1),2})^(0.5);
        %ctrvariance = (mapa{(level+1),2})^(0.5);
        %ctrvarianceSRD = (mapaSRD{(level+1),2})^(0.5);

    n = diff( Data{level+(1)}.kp ) + 1;

            A=randn(1,n);
```

```
Data{level+(1)}.det = ctrvariance*A;
%iid com a variancia do processo caindo com o fator ilustrado acima em funcao de um parametro alph
DataSRD{level+(1)}.det = ctrvarianceSRD*A;
%iid com a variancia do processo caindo com o fator ilustrado acima em funcao de um parametro alph
DataSRD2{level+(1)}.det = ctrvarianceSRD2*A;
%iid com a variancia do processo caindo com o fator ilustrado acima em funcao de um parametro alph
```

```
% Reconstrucao do nivel imediatamente inferior:

function approxs = Recon(level, NrOfVM, Data, h, g) % h will be high-pass filter coefficients.
    Lf = length(h);

    kp = Data{level + (1)}.kp; % coeficiente minimo e maximo na escala level

    %A seguir estao implementadas as equacoes 7.11, 7.12, 7.13 e 7.14 da
    %pagina 49:
    %%%%%%%%%%%%%%%%%

    ku = 2*kp; % coeficiente minimo e maximo na escala level apos fazermos o upsampling

    kc = ku + [0 Lf-1]; % coeficiente minimo e maximo gerados na escala level-1 logo apos a convolucao

    %%%%%%%%%%%%%%%%%%

    kp = Data{level-1 + (1)}.kp; % coeficiente minimo e maximo na escala level-1 (sao os que vamos manter)

    % coeficientes de approximacao e detalhe na escala level que serao
    % upsampled e convoluidos com os coeficientes dos filtros apropriados:
    appro =  Data{level + (1)}.app;
    detail =  Data{level + (1)}.det;

    % dyadup e do pacote de wavelets do Matlab (wavelet toolbox) e faz
    % exatamente o upsampling
    appup = dyadup(appro,0);
    detup = dyadup(detail,0);

    % A convolucao:
    appconv = conv(appup,g); % coefc. de approximacao convoluidos com os coeficientes do filtro passa-baixas
    detconv = conv(detup,h); % coefc. de detalhe convoluidos com os coeficientes do filtro passa-altas

    % indices = [dif1+(1):dif2+(1)], onde dif1 e a diferenca entre o kmindesejado e
    % kminconv resultando no shift que devemos dar no vetor inicial.
    % ja dif2 sera o maximo que queremos obter - o indice inicial do vetor
    % apos convolucao:
    indices = [kp(1)-kc(1)+(1):kp(2)-kc(1)+(1)];

    appro = appconv(indices) + detconv(indices);

    approxs = appro;
```

```
% Recon sinal com SRD com ganho variável via DWT com mapa de variâncias
% Reconstrucao do nivel imediatamente inferior:

function approxs = ReconSRD(level, NrOfVM, DataSRD, h, g) % h will be high-pass filter coefficients.
    Lf = length(h);

    kp = DataSRD{level + (1)}.kp; % coeficiente minimo e maximo na escala level

    %A seguir estao implementadas as equacoes 7.11, 7.12, 7.13 e 7.14 da
    %pagina 49:
    %%%%%%%%%%%%%%%%%

    ku = 2*kp; % coeficiente minimo e maximo na escala level apos fazermos o upsampling

    kc = ku + [0 Lf-1]; % coeficiente minimo e maximo gerados na escala level-1 logo apos a convolucao

    %%%%%%%%%%%%%%%%%%

    kp = DataSRD{level-1 + (1)}.kp; % coeficiente minimo e maximo na escala level-1 (sao os que vamos manter)

    % coeficientes de approximacao e detalhe na escala level que serao
    % upsampled e convoluidos com os coeficientes dos filtros apropriados:
    appro =  DataSRD{level + (1)}.app;
    detail =  DataSRD{level + (1)}.det;

    % dyadup e do pacote de wavelets do Matlab (wavelet toolbox) e faz
    % exatamente o upsampling
    appup = dyadup(appro,0);
    detup = dyadup(detail,0);

    % A convolucao:
    appconv = conv(appup,g); % coefc. de approximacao convoluidos com os coeficientes do filtro passa-baixas
    detconv = conv(detup,h); % coefc. de detalhe convoluidos com os coeficientes do filtro passa-altas

    % indices = [dif1+(1):dif2+(1)], onde dif1 e a diferenca entre o kmindesejado e
    % kminconv resultando no shift que devemos dar no vetor inicial.
    % ja dif2 sera o maximo que queremos obter - o indice inicial do vetor
    % apos convolucao:
    indices = [kp(1)-kc(1)+(1):kp(2)-kc(1)+(1)];

    appro = appconv(indices) + detconv(indices);

    approxs = appro;
```

```
% Recon sinal com SRD com ganho constante via DWT com mapa de variâncias
% Reconstrucao do nivel imediatamente inferior:

function approxs = ReconSRD2(level, NrOfVM, DataSRD2, h, g) % h will be high-pass filter coefficients.
    Lf = length(h);

    kp = DataSRD2{level + (1)}.kp; % coeficiente minimo e maximo na escala level

    %A seguir estao implementadas as equacoes 7.11, 7.12, 7.13 e 7.14 da
    %pagina 49:
    %%%%%%%%%%%%%%%%%

    ku = 2*kp; % coeficiente minimo e maximo na escala level apos fazermos o upsampling

    kc = ku + [0 Lf-1]; % coeficiente minimo e maximo gerados na escala level-1 logo apos a convolucao

    %%%%%%%%%%%%%%%%%%

    kp = DataSRD2{level-1 + (1)}.kp; % coeficiente minimo e maximo na escala level-1 (sao os que vamos manter)

    % coeficientes de approximacao e detalhe na escala level que serao
    % upsampled e convoluidos com os coeficientes dos filtros apropriados:
    appro =  DataSRD2{level + (1)}.app;
    detail =  DataSRD2{level + (1)}.det;

    % dyadup e do pacote de wavelets do Matlab (wavelet toolbox) e faz
    % exatamente o upsampling
    appup = dyadup(appro,0);
    detup = dyadup(detail,0);

    % A convolucao:
    appconv = conv(appup,g); % coefc. de approximacao convoluidos com os coeficientes do filtro passa-baixas
    detconv = conv(detup,h); % coefc. de detalhe convoluidos com os coeficientes do filtro passa-altas

    % indices = [dif1+(1):dif2+(1)], onde dif1 e a diferenca entre o kmindesejado e
    % kminconv resultando no shift que devemos dar no vetor inicial.
    % ja dif2 sera o maximo que queremos obter - o indice inicial do vetor
    % apos convolucao:
    indices = [kp(1)-kc(1)+(1):kp(2)-kc(1)+(1)];

    appro = appconv(indices) + detconv(indices);

    approxs = appro;
```

# Apêndice C – Códigos MATLAB - Síntese pelo método DWPT

```
function [DataB,DataC,DataD,DataE,DataF,DataG] = Sintese_DWPT_fgn_14(TopLev, NrOfVM, ninic, nsrdd, nsrdt, alpha


Lm=2^TopLev;


N = int2str(NrOfVM); %N e uma string
wname = 'db'; %falta concatenar com N para obter dbN
wname = strcat(wname,N)

g = dbwavf(wname); % uso da funcao dbaux seria mais simples pois aceitaria N como inteiro em vez de exigir a st
g = g * sqrt(2)
Lf = length(g) % precisa ser igual a 2N
h = fliplr(g); h = (-1).^(0:Lf-1).*h % representando a formula 75b do livro de wavelets

nome_arquivo = ['fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(TopLev),')_alpha('

nome_arquivo_1 = ['mapa_Variancias_fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(

nome_arquivo_2 = ['fgnDWPT14_SRD_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(TopLev),')_al

nome_arquivo_3 = ['mapaSRD_Variancias_fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2s

nome_arquivo_4 = ['fgnDWPT14_SRD2_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(TopLev),')_

nome_arquivo_5 = ['mapaSRD2_Variancias_fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2

    Lf = 2*NrOfVM; %comprimento do filtro
kn = [0 Lm-1]; %aqui dois extremos da serie que estamos interessados no nivel mais refinado - ira armazenar som
kp = kn; % ira armazenar extremos contendo non-polluted e polluted coefficients. Todos os coeficientes entre es
DataB{0+(1)}.kp = kn; %ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0)
    DataD{0+(1)}.kp = kn; %ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0)
kc = kn;  %ira armazenar dois extremos da serie temporarios para a proxima escala
kd = kn; %ira armazenar dois extremos da serie consolidados para a proxima escala
DataB{0+(1)}.kd = kn;
   DataD{0+(1)}.kd = kn;

%efetuando o calculo dos coeficientes em cada um dos extremos e que serao
%necessarios:
    for j=1:TopLev,
%for j=2:TopLev+1,
```

```
    %kd(1) e o kminimo da escala anterior
    %kd(2) e o kmax da escala anterior
    %Lf - 1 = 2N - 1
    kc = [kd(1)-(Lf-1) kd(2)];  % nao encontrei nenhuma formula que usa esses indices - temporario... nao a
    kd = fix (kc/2); % fix arredonda para zero... e na verdade deveria ser arredonda para baixo, nao? (floo
    % kd agora passou a conter os indices de kc, porem divididos por 2:
    %kd = [fix(kmin/2 - N + 1/2)) fix(kmax/2)] - isto parece, embora nao exatamente, com
    %as formulas 7.9 e 7.10 onde kd passa a representar uma escala menos
    %refinada. As diferencas estao no fix / floor e no 1/2 / 1

    kn = floor([0 kn(2)-(Lf-1)]/2); %[0 floor(kmax/2 - N + 1/2)]!!! -> nao temos mais nenhuma formula que i
    kp = floor(kp/2) - [(NrOfVM - 1) 0]; %nossa!!! o proprio kp, que continha o kn inicial ficando assim: k

     DataB{j+(1)}.kd = kd;
     DataB{j+(1)}.kp = kp;
     DataB{j+(1)}.kn = kn;
     DataD{j+(1)}.kd = kd;
     DataD{j+(1)}.kp = kp;
     DataD{j+(1)}.kn = kn;
end


%%Calculo do mapa de variâncias
for j=TopLev+1:-1:1,
f = 0.5/2^(j-1)
Cs=1
GSRD1=10
GSRD2=1
GSRD3=5

for n=1:+1:2^(j-1),
  if j==TopLev+1,
      mapa{j,n} = (2^(j-1))*(Cs/(1-alpha)*f^(1-alpha)*(n^(1-alpha)-(n-1)^(1-alpha)));
  else
      if mod(n,2) == 0,
         m=2*n;
         mapa{j,n} = (mapa{j+1,m-1}+mapa{j+1,m})/2;
      else
         m=(2*n-1);
         mapa{j,n} = (mapa{j+1,m}+mapa{j+1,m+1})/2;
      end
  end
end
%mapa normalizado
if j==TopLev+1,
 Y=mapa{j,2}
 for n=1:+1:2^(j-1),
   mapa{j,n} = mapa{j,n}/Y;
 end
end


for n=1:+1:2^(j-1),
  if j==TopLev+1,
    if n>=nsrdd & n<=nsrdt,
        mapaSRD2{j,n}=mapa{j,n}*GSRD1;
        for n=nsrdd:+1:nsrdd+((nsrdt-nsrdd)/3),
```

```
                mapaSRD{j,n}=mapa{j,n}*GSRD1;
            end

            for n=nsrdd+((nsrdt-nsrdd)/3)+1:+1:nsrdd+2*((nsrdt-nsrdd)/3),
                mapaSRD{j,n}=mapa{j,n}*GSRD3;
            end

            for n=nsrdd+2*((nsrdt-nsrdd)/3)+1:+1:nsrdd+3*((nsrdt-nsrdd)/3),
                mapaSRD{j,n}=mapa{j,n}*GSRD1;
            end

        else
            mapaSRD2{j,n}=mapa{j,n}*GSRD2;
            mapaSRD{j,n}=mapa{j,n}*GSRD2;
        end
      else
        if mod(n,2) == 0,
            m=2*n;
            mapaSRD{j,n} = (mapaSRD{j+1,m-1}+mapaSRD{j+1,m})/2;
            mapaSRD2{j,n} = (mapaSRD2{j+1,m-1}+mapaSRD2{j+1,m})/2;
        else
            m=(2*n-1);
            mapaSRD{j,n} = (mapaSRD{j+1,m}+mapaSRD{j+1,m+1})/2;
            mapaSRD2{j,n} = (mapaSRD2{j+1,m}+mapaSRD2{j+1,m+1})/2;
        end
        %end
      end
    %end
    end
    end


   if ninic <= 2^TopLev,
    if mod(ninic,2) == 0,
        m=ninic/2
    else
        m=(ninic+1)/2
    end

    for j=TopLev+1:-1:1,

        j
        o = diff( DataB{j}.kp ) + 1
        if j==TopLev+1,
            p = diff( DataB{j}.kp ) + 1
        end
        m
        f = 0.5/2^(j-1)
         if j == TopLev+1,
             if mod(m,2) == 0,
                if mod(ninic,2) == 0,
              ctrvariance{j,ninic} = mapa{j,ninic}^(0.5);
              ctrvarianceSRD{j,ninic}=mapaSRD{j,ninic}^(0.5);
              ctrvarianceSRD2{j,ninic}=mapaSRD2{j,ninic}^(0.5);
              A=randn(1,o);
              W{j,ninic} = ctrvariance{j,ninic}*A;
              WSRD{j,ninic} = ctrvarianceSRD{j,ninic}*A;
```

```
WSRD2{j,ninic} = ctrvarianceSRD2{j,ninic}*A;
ctrvariance{j,ninic-1}=mapa{j,ninic-1}^(0.5);
ctrvarianceSRD{j,ninic-1}=mapaSRD{j,ninic-1}^(0.5);
ctrvarianceSRD2{j,ninic-1}=mapaSRD2{j,ninic-1}^(0.5);
B=randn(1,o);
W{j,ninic-1} = ctrvariance{j,ninic-1}*B;
WSRD{j,ninic-1} = ctrvarianceSRD{j,ninic-1}*B;
WSRD2{j,ninic-1} = ctrvarianceSRD2{j,ninic-1}*B;

      DataB{j}=[W{j,ninic-1} W{j,ninic}];
      DataD{j}=[WSRD{j,ninic-1} WSRD{j,ninic}];
      DataF{j}=[WSRD2{j,ninic-1} WSRD2{j,ninic}];

  else
    A=randn(1,o);
    ctrvariance{j,ninic} = mapa{j,ninic}^(0.5);
    ctrvarianceSRD{j,ninic}=mapaSRD{j,ninic}^(0.5);
    ctrvarianceSRD2{j,ninic}=mapaSRD2{j,ninic}^(0.5);
    W{j,ninic} = ctrvariance{j,ninic}*A;
    WSRD{j,ninic} = ctrvarianceSRD{j,ninic}*A;
    WSRD2{j,ninic} = ctrvarianceSRD2{j,ninic}*A;
    ctrvariance{j,ninic+1}=mapa{j,ninic+1}^(0.5);
    ctrvarianceSRD{j,ninic+1}=mapaSRD{j,ninic+1}^(0.5);
    ctrvarianceSRD2{j,ninic+1}=mapaSRD2{j,ninic+1}^(0.5);
    B=randn(1,o);
    W{j,ninic+1} = ctrvariance{j,ninic+1}*B;
    WSRD{j,ninic+1} = ctrvarianceSRD{j,ninic+1}*B;
    WSRD2{j,ninic+1} = ctrvarianceSRD2{j,ninic+1}*B;

      DataB{j}=[W{j,ninic} W{j,ninic+1}];
      DataD{j}=[WSRD{j,ninic} WSRD{j,ninic+1}];
      DataF{j}=[WSRD2{j,ninic} WSRD2{j,ninic+1}];


  end

else
  if mod(ninic,2) == 0,

  ctrvariance{j,ninic}=mapa{j,ninic}^(0.5);
  ctrvarianceSRD{j,ninic}=mapaSRD{j,ninic}^(0.5);
  ctrvarianceSRD2{j,ninic}=mapaSRD2{j,ninic}^(0.5);
  A=randn(1,o);
  W{j,ninic} = ctrvariance{j,ninic}*A;
  WSRD{j,ninic} = ctrvarianceSRD{j,ninic}*A;
  WSRD2{j,ninic} = ctrvarianceSRD2{j,ninic}*A;
    ctrvariance{j,ninic-1}=mapa{j,ninic-1}^(0.5);
    ctrvarianceSRD{j,ninic-1}=mapaSRD{j,ninic-1}^(0.5);
    ctrvarianceSRD2{j,ninic-1}=mapaSRD2{j,ninic-1}^(0.5);
    B=randn(1,o);
    W{j,ninic-1}=ctrvariance{j,ninic-1}*B;
    WSRD{j,ninic-1} = ctrvarianceSRD{j,ninic-1}*B;
    WSRD2{j,ninic-1} = ctrvarianceSRD2{j,ninic-1}*B;

      DataB{j}=[W{j,ninic-1} W{j,ninic}];
      DataD{j}=[WSRD{j,ninic-1} WSRD{j,ninic}];
      DataF{j}=[WSRD2{j,ninic-1} WSRD2{j,ninic}];
```

```
        else

            ctrvariance{j,ninic} = mapa{j,ninic}^(0.5);
            ctrvarianceSRD{j,ninic}=mapaSRD{j,ninic}^(0.5);
            ctrvarianceSRD2{j,ninic}=mapaSRD2{j,ninic}^(0.5);
            A=randn(1,o);
            W{j,ninic} = ctrvariance{j,ninic}*A;
            WSRD{j,ninic} = ctrvarianceSRD{j,ninic}*A;
            WSRD2{j,ninic} = ctrvarianceSRD2{j,ninic}*A;
            ctrvariance{j,ninic+1} = mapa{j,ninic+1}^(0.5);
            ctrvarianceSRD{j,ninic+1}=mapaSRD{j,ninic+1}^(0.5);
            ctrvarianceSRD2{j,ninic+1}=mapaSRD2{j,ninic+1}^(0.5);
            B=randn(1,o);
        W{j,ninic+1}=ctrvariance{j,ninic+1}*B;
        WSRD{j,ninic+1} = ctrvarianceSRD{j,ninic+1}*B;
        WSRD2{j,ninic+1} = ctrvarianceSRD2{j,ninic+1}*B;

            DataB{j}=[W{j,ninic} W{j,ninic+1}];
            DataD{j}=[WSRD{j,ninic} WSRD{j,ninic+1}];
            DataF{j}=[WSRD2{j,ninic} WSRD2{j,ninic+1}];

        end

    end
else

    if mod(m,2) == 0,

        Wup{j+1,2*m-1} = dyadup(W{j+1,2*m-1},0);
        Wup{j+1,2*m} = dyadup(W{j+1,2*m},0);
        Wconv{j+1,2*m-1} = conv(Wup{j+1,2*m-1},h);
        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
        Wconv{j+1,2*m} = conv(Wup{j+1,2*m},g);
        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
        W{j,m}=Wconv{j+1,2*m-1}+Wconv{j+1,2*m};

        %SRD
        WSRDup{j+1,2*m-1} = dyadup(WSRD{j+1,2*m-1},0);
        WSRDup{j+1,2*m} = dyadup(WSRD{j+1,2*m},0);
        WSRDconv{j+1,2*m-1} = conv(WSRDup{j+1,2*m-1},h);
        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
        WSRDconv{j+1,2*m} = conv(WSRDup{j+1,2*m},g);
        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
        WSRD{j,m}=WSRDconv{j+1,2*m-1}+WSRDconv{j+1,2*m};

        %SRD-ganho constante
        WSRD2up{j+1,2*m-1} = dyadup(WSRD2{j+1,2*m-1},0);
        WSRD2up{j+1,2*m} = dyadup(WSRD2{j+1,2*m},0);
        WSRD2conv{j+1,2*m-1} = conv(WSRD2up{j+1,2*m-1},h);
        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
        WSRD2conv{j+1,2*m} = conv(WSRD2up{j+1,2*m},g);
        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
        WSRD2{j,m}=WSRD2conv{j+1,2*m-1}+WSRD2conv{j+1,2*m};


        if j==(TopLev+1)-1,
```

```
        %p = diff( DataB{(TopLev+1)}.kp ) + 1
        C=randn(1,p);
        ctrvariance{j+1,2*(m-1)-1} = mapa{j+1,2*(m-1)-1}^(0.5);
        W{j+1,2*(m-1)-1} = ctrvariance{j+1,2*(m-1)-1}*C;
        ctrvarianceSRD{j+1,2*(m-1)-1} = mapaSRD{j+1,2*(m-1)-1}^(0.5);
        ctrvarianceSRD2{j+1,2*(m-1)-1} = mapaSRD2{j+1,2*(m-1)-1}^(0.5);
        WSRD{j+1,2*(m-1)-1} = ctrvarianceSRD{j+1,2*(m-1)-1}*C;
        WSRD2{j+1,2*(m-1)-1} = ctrvarianceSRD2{j+1,2*(m-1)-1}*C;

        D=randn(1,p);
        ctrvariance{j+1,2*(m-1)} = mapa{j+1,2*(m-1)}^(0.5);
        W{j+1,2*(m-1)} = ctrvariance{j+1,2*(m-1)}*D;
        ctrvarianceSRD{j+1,2*(m-1)} = mapaSRD{j+1,2*(m-1)}^(0.5);
        ctrvarianceSRD2{j+1,2*(m-1)} = mapaSRD2{j+1,2*(m-1)}^(0.5);
        WSRD{j+1,2*(m-1)} = ctrvarianceSRD{j+1,2*(m-1)}*D;
        WSRD2{j+1,2*(m-1)} = ctrvarianceSRD2{j+1,2*(m-1)}*D;

    else
        for i=TopLev+1:-1:j+1,
            if i==TopLev+1,
                for l=(2^(i-j)*(m-1)):-1:(2^(i-j)*(m-1)-2^(i-j)+1),
                   % p = diff( DataB{(TopLev+1)}.kp ) + 1
                    C=randn(1,p);
                    ctrvariance{i,l} = mapa{i,l}^(0.5);
                    W{i,l} = ctrvariance{i,l}*C;
                    ctrvarianceSRD{i,l} = mapaSRD{i,l}^(0.5);
                    ctrvarianceSRD2{i,l} = mapaSRD2{i,l}^(0.5);
                    WSRD{i,l} = ctrvarianceSRD{i,l}*C;
                    WSRD2{i,l} = ctrvarianceSRD2{i,l}*C;
                end
            else
                for l=(2^(i-j)*(m-1)):-1:(2^(i-j)*(m-1)-2^(i-j)+1),
                    if mod(l,2) == 0,
                        Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                        Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                        Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},h);
                        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                        Wconv{i+1,2*l} = conv(Wup{i+1,2*l},g);
                        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                        W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};
                        %SRD
                        WSRDup{i+1,2*l-1} = dyadup(WSRD{i+1,2*l-1},0);
                        WSRDup{i+1,2*l} = dyadup(WSRD{i+1,2*l},0);
                        WSRDconv{i+1,2*l-1} = conv(WSRDup{i+1,2*l-1},h);
                        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                        WSRDconv{i+1,2*l} = conv(WSRDup{i+1,2*l},g);
                        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                        WSRD{i,l}=WSRDconv{i+1,2*l-1}+WSRDconv{i+1,2*l};

                        %SRD com ganho constante
                        WSRD2up{i+1,2*l-1} = dyadup(WSRD2{i+1,2*l-1},0);
                        WSRD2up{i+1,2*l} = dyadup(WSRD2{i+1,2*l},0);
                        WSRD2conv{i+1,2*l-1} = conv(WSRD2up{i+1,2*l-1},h);
                        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                        WSRD2conv{i+1,2*l} = conv(WSRD2up{i+1,2*l},g);
                        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                        WSRD2{i,l}=WSRD2conv{i+1,2*l-1}+WSRD2conv{i+1,2*l};
```

```
                else
                    Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                    Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                    Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},g);
                    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                    Wconv{i+1,2*l} = conv(Wup{i+1,2*l},h);
                    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                    W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};
                    %SRD
                    WSRDup{i+1,2*l-1} = dyadup(WSRD{i+1,2*l-1},0);
                    WSRDup{i+1,2*l} = dyadup(WSRD{i+1,2*l},0);
                    WSRDconv{i+1,2*l-1} = conv(WSRDup{i+1,2*l-1},g);
                    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                    WSRDconv{i+1,2*l} = conv(WSRDup{i+1,2*l},h);
                    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                    WSRD{i,l}=WSRDconv{i+1,2*l-1}+WSRDconv{i+1,2*l};

                    %SRD com ganho constante
                    WSRD2up{i+1,2*l-1} = dyadup(WSRD2{i+1,2*l-1},0);
                    WSRD2up{i+1,2*l} = dyadup(WSRD2{i+1,2*l},0);
                    WSRD2conv{i+1,2*l-1} = conv(WSRD2up{i+1,2*l-1},g);
                    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                    WSRD2conv{i+1,2*l} = conv(WSRD2up{i+1,2*l},h);
                    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                    WSRD2{i,l}=WSRD2conv{i+1,2*l-1}+WSRD2conv{i+1,2*l};

                end
            end
        end
    end
end

Wup{j+1,2*(m-1)-1} = dyadup(W{j+1,2*(m-1)-1},0);
Wup{j+1,2*(m-1)} = dyadup(W{j+1,2*(m-1)},0);
Wconv{j+1,2*(m-1)-1} = conv(Wup{j+1,2*(m-1)-1},g);
% coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
Wconv{j+1,2*(m-1)} = conv(Wup{j+1,2*(m-1)},h);
% coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
W{j,m-1}=Wconv{j+1,2*(m-1)-1}+Wconv{j+1,2*(m-1)};

%SRD
WSRDup{j+1,2*(m-1)-1} = dyadup(WSRD{j+1,2*(m-1)-1},0);
WSRDup{j+1,2*(m-1)} = dyadup(WSRD{j+1,2*(m-1)},0);
WSRDconv{j+1,2*(m-1)-1} = conv(WSRDup{j+1,2*(m-1)-1},g);
% coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
WSRDconv{j+1,2*(m-1)} = conv(WSRDup{j+1,2*(m-1)},h);
% coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
WSRD{j,m-1}=WSRDconv{j+1,2*(m-1)-1}+WSRDconv{j+1,2*(m-1)};

%SRD com ganho constante
WSRD2up{j+1,2*(m-1)-1} = dyadup(WSRD2{j+1,2*(m-1)-1},0);
WSRD2up{j+1,2*(m-1)} = dyadup(WSRD2{j+1,2*(m-1)},0);
WSRD2conv{j+1,2*(m-1)-1} = conv(WSRD2up{j+1,2*(m-1)-1},g);
% coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
WSRD2conv{j+1,2*(m-1)} = conv(WSRD2up{j+1,2*(m-1)},h);
```

```
    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
    WSRD2{j,m-1}=WSRD2conv{j+1,2*(m-1)-1}+WSRD2conv{j+1,2*(m-1)};



    DataB{j}=[W{j,m-1} W{j,m}];
    DataD{j}=[WSRD{j,m-1} WSRD{j,m}];
    DataF{j}=[WSRD2{j,m-1} WSRD2{j,m}];

   m=m/2;

  else

    Wup{j+1,2*m-1} = dyadup(W{j+1,2*m-1},0);
    Wup{j+1,2*m} = dyadup(W{j+1,2*m},0);
    Wconv{j+1,2*m-1} = conv(Wup{j+1,2*m-1},g);
    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
    Wconv{j+1,2*m} = conv(Wup{j+1,2*m},h);
    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
    W{j,m}=Wconv{j+1,2*m-1}+Wconv{j+1,2*m};
    %SRD
    WSRDup{j+1,2*m-1} = dyadup(WSRD{j+1,2*m-1},0);
    WSRDup{j+1,2*m} = dyadup(WSRD{j+1,2*m},0);
    WSRDconv{j+1,2*m-1} = conv(WSRDup{j+1,2*m-1},g);
    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
    WSRDconv{j+1,2*m} = conv(WSRDup{j+1,2*m},h);
    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
    WSRD{j,m}=WSRDconv{j+1,2*m-1}+WSRDconv{j+1,2*m};

    %SRD com ganho constante
    WSRD2up{j+1,2*m-1} = dyadup(WSRD2{j+1,2*m-1},0);
    WSRD2up{j+1,2*m} = dyadup(WSRD2{j+1,2*m},0);
    WSRD2conv{j+1,2*m-1} = conv(WSRD2up{j+1,2*m-1},g);
    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
    WSRD2conv{j+1,2*m} = conv(WSRD2up{j+1,2*m},h);
    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
    WSRD2{j,m}=WSRD2conv{j+1,2*m-1}+WSRD2conv{j+1,2*m};

  if j>1,
  if j==(TopLev+1)-1,
        C=randn(1,p)
        ctrvariance{j+1,2*(m+1)-1} = mapa{j+1,2*(m+1)-1}^(0.5);
        W{j+1,2*(m+1)-1} = ctrvariance{j+1,2*(m+1)-1}*C;
        ctrvarianceSRD{j+1,2*(m+1)-1} = mapaSRD{j+1,2*(m+1)-1}^(0.5);
        WSRD{j+1,2*(m+1)-1} = ctrvarianceSRD{j+1,2*(m+1)-1}*C;
        ctrvarianceSRD2{j+1,2*(m+1)-1} = mapaSRD2{j+1,2*(m+1)-1}^(0.5);
        WSRD2{j+1,2*(m+1)-1} = ctrvarianceSRD2{j+1,2*(m+1)-1}*C;

        D=randn(1,p);
        ctrvariance{j+1,2*(m+1)} = mapa{j+1,2*(m+1)}^(0.5);
        W{j+1,2*(m+1)} = ctrvariance{j+1,2*(m+1)}*D;
        ctrvarianceSRD{j+1,2*(m+1)} = mapaSRD{j+1,2*(m+1)}^(0.5);
        WSRD{j+1,2*(m+1)} = ctrvarianceSRD{j+1,2*(m+1)}*D;
        ctrvarianceSRD2{j+1,2*(m+1)} = mapaSRD2{j+1,2*(m+1)}^(0.5);
        WSRD2{j+1,2*(m+1)} = ctrvarianceSRD2{j+1,2*(m+1)}*D;

  else
```

```
for i=TopLev+1:-1:j+1,
     if i==TopLev+1,
         for l=(2^(i-j)*(m+1)):-1:(2^(i-j)*(m+1)-2^(i-j)+1),
             %p = diff( DataB{(TopLev+1)}.kp ) + 1
             C=randn(1,p);
             ctrvariance{i,l} = mapa{i,l}^(0.5);
             W{i,l} = ctrvariance{i,l}*C;
             ctrvarianceSRD{i,l} = mapaSRD{i,l}^(0.5);
             ctrvarianceSRD2{i,l} = mapaSRD2{i,l}^(0.5);
             WSRD{i,l} = ctrvarianceSRD{i,l}*C;
             WSRD2{i,l} = ctrvarianceSRD2{i,l}*C;
         end
     else
         for l=(2^(i-j)*(m+1)):-1:(2^(i-j)*(m+1)-2^(i-j)+1),
             if mod(l,2) == 0,
                 Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                 Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                 Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},h);
                 % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                 Wconv{i+1,2*l} = conv(Wup{i+1,2*l},g);
                 % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                 W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};
                 %SRD
                 WSRDup{i+1,2*l-1} = dyadup(WSRD{i+1,2*l-1},0);
                 WSRDup{i+1,2*l} = dyadup(WSRD{i+1,2*l},0);
                 WSRDconv{i+1,2*l-1} = conv(WSRDup{i+1,2*l-1},h);
                 % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                 WSRDconv{i+1,2*l} = conv(WSRDup{i+1,2*l},g);
                 % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                 WSRD{i,l}=WSRDconv{i+1,2*l-1}+WSRDconv{i+1,2*l};

                 %SRD com ganho constante
                 WSRD2up{i+1,2*l-1} = dyadup(WSRD2{i+1,2*l-1},0);
                 WSRD2up{i+1,2*l} = dyadup(WSRD2{i+1,2*l},0);
                 WSRD2conv{i+1,2*l-1} = conv(WSRD2up{i+1,2*l-1},h);
                 % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                 WSRD2conv{i+1,2*l} = conv(WSRD2up{i+1,2*l},g);
                 % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                 WSRD2{i,l}=WSRD2conv{i+1,2*l-1}+WSRD2conv{i+1,2*l};

             else
                 Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                 Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                 Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},g);
                 % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                 Wconv{i+1,2*l} = conv(Wup{i+1,2*l},h);
                 % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                 W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};
                 %SRD
                 WSRDup{i+1,2*l-1} = dyadup(WSRD{i+1,2*l-1},0);
                 WSRDup{i+1,2*l} = dyadup(WSRD{i+1,2*l},0);
                 WSRDconv{i+1,2*l-1} = conv(WSRDup{i+1,2*l-1},g);
                 % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                 WSRDconv{i+1,2*l} = conv(WSRDup{i+1,2*l},h);
                 % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                 WSRD{i,l}=WSRDconv{i+1,2*l-1}+WSRDconv{i+1,2*l};
```

```
                                %SRD com ganho constante
                                WSRD2up{i+1,2*l-1} = dyadup(WSRD2{i+1,2*l-1},0);
                                WSRD2up{i+1,2*l} = dyadup(WSRD2{i+1,2*l},0);
                                WSRD2conv{i+1,2*l-1} = conv(WSRD2up{i+1,2*l-1},g);
                                % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                                WSRD2conv{i+1,2*l} = conv(WSRD2up{i+1,2*l},h);
                                % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                                WSRD2{i,l}=WSRD2conv{i+1,2*l-1}+WSRD2conv{i+1,2*l};
                            end
                        end
                    end
                end
            end


            Wup{j+1,2*(m+1)-1} = dyadup(W{j+1,2*(m+1)-1},0);
            Wup{j+1,2*(m+1)} = dyadup(W{j+1,2*(m+1)},0);
            Wconv{j+1,2*(m+1)-1} = conv(Wup{j+1,2*(m+1)-1},h);
            % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
            Wconv{j+1,2*(m+1)} = conv(Wup{j+1,2*(m+1)},g);
            % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
            W{j,m+1}=Wconv{j+1,2*(m+1)-1}+Wconv{j+1,2*(m+1)};

            %SRD
            WSRDup{j+1,2*(m+1)-1} = dyadup(WSRD{j+1,2*(m+1)-1},0);
            WSRDup{j+1,2*(m+1)} = dyadup(WSRD{j+1,2*(m+1)},0);
            WSRDconv{j+1,2*(m+1)-1} = conv(WSRDup{j+1,2*(m+1)-1},h);
            % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
            WSRDconv{j+1,2*(m+1)} = conv(WSRDup{j+1,2*(m+1)},g);
            % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
            WSRD{j,m+1}=WSRDconv{j+1,2*(m+1)-1}+WSRDconv{j+1,2*(m+1)};

            %SRD com ganho constante
            WSRD2up{j+1,2*(m+1)-1} = dyadup(WSRD2{j+1,2*(m+1)-1},0);
            WSRD2up{j+1,2*(m+1)} = dyadup(WSRD2{j+1,2*(m+1)},0);
            WSRD2conv{j+1,2*(m+1)-1} = conv(WSRD2up{j+1,2*(m+1)-1},h);
            % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
            WSRD2conv{j+1,2*(m+1)} = conv(WSRD2up{j+1,2*(m+1)},g);
            % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
            WSRD2{j,m+1}=WSRD2conv{j+1,2*(m+1)-1}+WSRD2conv{j+1,2*(m+1)};
            end

            if j==1,
                DataB{j}=[W{j,m}];
                DataD{j}=[WSRD{j,m}];
                DataF{j}=[WSRD2{j,m}];
            else
                DataB{j}=[W{j,m} W{j,m+1}];
                DataD{j}=[WSRD{j,m} WSRD{j,m+1}];
                DataF{j}=[WSRD2{j,m} WSRD2{j,m+1}];
            end
            m=(m+1)/2;

        end

    end
end
```

```
    else

        warning('O coeficiente escolhido não está dentro dos coeficientes de último nível para a DWPT');
    end

          for j=1:+1:TopLev+1,
          if j==1,
              DataC{j} =mapa{j,1};
              DataE{j} =mapaSRD{j,1};
              DataG{j} =mapaSRD2{j,1};
          else
              DataC{j} =mapa{j,1};
              DataE{j} =mapaSRD{j,1};
              DataG{j} =mapaSRD2{j,1};
              for nl=2:+1:2^(j-1),
              DataC{j} =[DataC{j} mapa{j,nl}];
              DataE{j} =[DataE{j} mapaSRD{j,nl}];
              DataG{j} =[DataG{j} mapaSRD2{j,nl}];
              end
          end
          end

if (exist('nome_arquivo')==1),
    if length(nome_arquivo)>1,
        for j=1:+1:TopLev+1,
        DataB{j};
        ans';
        nome_arquivo=[nome_arquivo,'_nível',int2str(j),'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo);
        eval(comando2ex);
        nome_arquivo = ['fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(TopLev),')_


       end
    end
end


if (exist('nome_arquivo_2')==1),
    if length(nome_arquivo_2)>1,
        for j=1:+1:TopLev+1,
        DataD{j};
        ans';
        nome_arquivo_2=[nome_arquivo_2,'_nível',int2str(j),'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_2);
        eval(comando2ex);
        nome_arquivo_2 = ['fgnDWPT14_SRD_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(TopL

        end
    end
end


if (exist('nome_arquivo_1')==1),
```

```
    if length(nome_arquivo_1)>1,
        for j=1:+1:TopLev+1,
          DataC{j};
          ans';
          nome_arquivo_1=[nome_arquivo_1,'_nível',int2str(j),'.txt'];
          comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_1);
          eval(comando2ex);
          nome_arquivo_1 = ['mapa_Variancias_fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(

        end
    end
end


if (exist('nome_arquivo_3')==1),
    if length(nome_arquivo_3)>1,
        for j=1:+1:TopLev+1,
          DataE{j};
          ans';
          nome_arquivo_3=[nome_arquivo_3,'_nível',int2str(j),'.txt'];
          comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_3);
          eval(comando2ex);
          nome_arquivo_3 = ['mapaSRD_Variancias_fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopL

        end
    end
 end

if (exist('nome_arquivo_4')==1),
    if length(nome_arquivo_4)>1,
        for j=1:+1:TopLev+1,
        DataF{j};
        ans';
        nome_arquivo_4=[nome_arquivo_4,'_nível',int2str(j),'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_4);
        eval(comando2ex);
        nome_arquivo_4 = ['fgnDWPT14_SRD2_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(Top

        end
    end
end

if (exist('nome_arquivo_5')==1),
    if length(nome_arquivo_5)>1,
        for j=1:+1:TopLev+1,
        DataG{j};
        ans';
        nome_arquivo_5=[nome_arquivo_5,'_nível',int2str(j),'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo_5);
        eval(comando2ex);
        nome_arquivo_5 = ['mapaSRD2_Variancias_fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLe

      end
    end
end
```

# Apêndice D – Códigos MATLAB - Análise DWT e DWPT

```
% Análise DWT:

function [DataB] = Analise3(level, NrOfVM, sinal) % h will be high-pass filter coefficients.

N = int2str(NrOfVM); %N e uma string
wname = 'db'; %falta concatenar com N para obter dbN
wname = strcat(wname,N)

g = dbwavf(wname);
% uso da funcao dbaux seria mais simples pois aceitaria N como inteiro em vez de exigir a string dbN
g = g * sqrt(2)
Lf = length(g) % precisa ser igual a 2N
h = fliplr(g); h = (-1).^(0:Lf-1).*h
% de qq modo a formula acima condiz com
% os coeficientes apresentados pelo pacote Wavelet do Matlab


%carrega o sinal
A = load(sinal);
j=1;
DataB{j}=A.';
nome_arquivo = [sinal,'DWT'];
%Lf = length(h);
%Calcula o número de pontos do sinal
Lm = length(A);

     for j=1:+1:level,
        % A convolucao:
        kp = Lm/(2^j);
        if j == 1,
            appconv{j+(1)} = conv(A,g);
            % coefc. de approximacao convoluidos com os coeficientes do filtro passa-baixas
            detconv{j+(1)} = conv(A,h);
            % coefc. de detalhe convoluidos com os coeficientes do filtro passa-altas
        else
            appconv{j+(1)} = conv(appdown{j},g);
            % coefc. de approximacao convoluidos com os coeficientes do filtro passa-baixas
            detconv{j+(1)} = conv(appdown{j},h);
            % coefc. de detalhe convoluidos com os coeficientes do filtro passa-altas
        end
        % dyaddown e do pacote de wavelets do Matlab (wavelet toolbox) e faz
        % exatamente o downsampling
            appdown{j+(1)} = dyaddown(appconv{j+(1)},0);
```

```
            detdown{j+(1)} = dyaddown(detconv{j+(1)},0)*(-1);
        if j == level,
            DataB{j+(2)} = appdown{j+(1)}.';
            DataB{j+(1)} = detdown{j+(1)}.';
        else
            DataB{j+(1)} = detdown{j+(1)}.';
        end

     end


if (exist('nome_arquivo')==1),
    if length(nome_arquivo)>1,
      nome_arquivo=[nome_arquivo,'.txt'];
      fid = fopen(nome_arquivo,'w');
      for j=1:+1:level+2,
       y = DataB{j};
       fprintf(fid,'%12.4f\n',y);
       fprintf(fid,'\r\n',' ');
      end
      fclose(fid);
      nome_arquivo=[];
    end
end




% Análise DWPT:

function [DataC] = Analise_DWPT2(level, NrOfVM, sinal) % h will be high-pass filter coefficients.

N = int2str(NrOfVM); %N e uma string
wname = 'db'; %falta concatenar com N para obter dbN
wname = strcat(wname,N)

g = dbwavf(wname);
% uso da funcao dbaux seria mais simples pois aceitaria N como inteiro em vez de exigir a string dbN
g = g * sqrt(2)
Lf = length(g) % precisa ser igual a 2N
h = fliplr(g); h = (-1).^(0:Lf-1).*h
% de qq modo a formula acima condiz com
% os coeficientes apresentados pelo pacote Wavelet do Matlab


%carrega o sinal
A = load(sinal);
T = length(A)
j=1;
DataC{j}=A.';
%DataD{j}=A;
%DataE{j}=A;
nome_arquivo = [sinal,'DWPT'];
nome_arquivot = [sinal,'DWPTtransposto'];
%Lf = length(h);
%Calcula o número de pontos do sinal
Lm = length(A);
```

```
     n = 1; %inicia a variável n como zero pois o sinal é o mesmo que o coeficiente W(0,0)
     for j=1:+1:level,
        % A convolucao:
        kp = Lm/(2^j);
         if j == 1,
             Wconv{j,2*n-1} = conv(A,g);
             % sinal W(0,0) convoluido com os coeficientes do filtro passa-baixas e feita a subamostragem
             Wconv{j,2*n} = conv(A,h);
             % sinal W(0,0) convoluído com os coeficientes do filtro passa-altas e feita a subamostragem e reve
             Wdown{j,2*n-1} = dyaddown(Wconv{j,2*n-1},0);
             Wdown{j,2*n} = dyaddown(Wconv{j,2*n},0)*(-1);
             DataC{j+(1),2*n-1} = Wdown{j,2*n-1}.';
             DataC{j+(1),2*n} = Wdown{j,2*n}.';
         else
           for n=1:+1:2^(j-1),
             if mod(n,2) == 0,
                Wconv{j,2*n-1} = conv(Wdown{j-1,n},h);
                % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
                Wconv{j,2*n} = conv(Wdown{j-1,n},g);
                % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
                Wdown{j,2*n-1} = dyaddown(Wconv{j,2*n-1},0)*(-1);
                Wdown{j,2*n} = dyaddown(Wconv{j,2*n},0);
             else
                Wconv{j,2*n-1} = conv(Wdown{j-1,n},g);
                % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
                Wconv{j,2*n} = conv(Wdown{j-1,n},h);
                % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
                Wdown{j,2*n-1} = dyaddown(Wconv{j,2*n-1},0);
                Wdown{j,2*n} = dyaddown(Wconv{j,2*n},0)*(-1);
             end
             DataC{j+(1),2*n-1} = Wdown{j,2*n-1}.';
             DataC{j+(1),2*n} = Wdown{j,2*n}.';
           end
         end


     end

if (exist('nome_arquivo')==1),
    if length(nome_arquivo)>1,
      nome_arquivo=[nome_arquivo,'.txt'];
      fid = fopen(nome_arquivo,'w');
      for j=1:+1:level+1,
       for n=1:+1:2^(j-1),
        y = DataC{j,n};
        fprintf(fid,'%12.4f\n',y);
       end
       fprintf(fid,'\r\n',' ');
      end
      fclose(fid);
      nome_arquivo=[];
    end
end


if (exist('nome_arquivot')==1),
    if length(nome_arquivot)>1,
```

```
      nome_arquivot=[nome_arquivot,'.txt'];
      fidt = fopen(nome_arquivot,'w');
      for j=1:+1:level+1,
        for n=1:+1:2^(j-1),
            yt = DataC{j,n}.';
            fprintf(fidt,'%12.4f\r\n',yt);
        end
        fprintf(fidt,'\r\n',' ');
      end
      fclose(fidt);
      nome_arquivot=[];
    end
end
```

# Apêndice E – Códigos MATLAB - Síntese de séries com periodicidade

```
% Gerador de cosseno
function [w,nome_arquivo] = cosseno_isabelle3(N,f,AM)
%f - frequencia do cosseno
%N - numero de pontos
%t - periodo da serie
t=N;
x=2*pi*f;
a=1/N;
k=0:a:1-a;
w=AM*cos(x*k);
plot(k,w), grid on;

nome_arquivo = ['cosseno_isabelle3_(N=',int2str(N),')_(f=',int2str(f),')_(AM=',num2str(AM),')'];


if (exist('nome_arquivo')==1),
    if length(nome_arquivo)>1,
        w;
        ans';
        nome_arquivo=[nome_arquivo,'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo);
        eval(comando2ex);
        nome_arquivo=[];
    end
end



% Sintese fGn com periodicidade via inserção de cosseno pelo método da DWT com mapa de variâncias

function [DataB] = Sintese_cosseno_fgn(TopLev, NrOfVM, Lm, alpha, sinal) % h will be high-pass filter coefficie

N = int2str(NrOfVM); %N e uma string
wname = 'db'; %falta concatenar com N para obter dbN
wname = strcat(wname,N)

g = dbwavf(wname); % uso da funcao dbaux seria mais simples pois aceitaria N como inteiro em vez de exigir a st
g = g * sqrt(2)
Lf = length(g) % precisa ser igual a 2N
h = fliplr(g); h = (-1).^(0:Lf-1).*h
% de qq modo a formula acima condiz com
% os coeficientes apresentados pelo pacote Wavelet do Matlab
```

```
nome_arquivo = ['Cosseno_(',sinal,')+fgn_(',int2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(TopL
            %nome_arquivo = ['fGn_(',int2str(Lm),')pts_alpha(',num2str(varargin{1}),')_NrOfVM(',int2str(NrOfVM)

A=load(sinal);

%sinal é o arquivo com os coeficientes DWT do cosseno


    Lf = 2*NrOfVM; %comprimento do filtro
kn = [0 Lm-1]; %aqui dois extremos da serie que estamos interessados no nivel mais refinado - ira armazenar som
kp = kn; % ira armazenar extremos contendo non-polluted e polluted coefficients. Todos os coeficientes entre es
DataB{0+(1)}.kp = kn; %ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0)
kc = kn;  %ira armazenar dois extremos da serie temporarios para a proxima escala
kd = kn; %ira armazenar dois extremos da serie consolidados para a proxima escala (desnecessario ja que temos o
DataB{0+(1)}.kd = kn;

%efetuando o calculo dos coeficientes em cada um dos extremos e que serao
%necessarios:
for j=1:TopLev,
        %kd(1) e o kminimo da escala anterior
        %kd(2) e o kmax da escala anterior
        %Lf - 1 = 2N - 1
        kc = [kd(1)-(Lf-1) kd(2)];  % nao encontrei nenhuma formula que usa esses indices - temporario... nao a
        kd = fix (kc/2); % fix arredonda para zero... e na verdade deveria ser arredonda para baixo, nao? (floo
        % kd agora passou a conter os indices de kc, porem divididos por 2:
        %kd = [fix(kmin/2 - N + 1/2)) fix(kmax/2)] - isto parece, embora nao exatamente, com
        %as formulas 7.9 e 7.10 onde kd passa a representar uma escala menos
        %refinada. As diferencas estao no fix / floor e no 1/2 / 1

        kn = floor([0 kn(2)-(Lf-1)]/2); %[0 floor(kmax/2 - N + 1/2)]!!! -> nao temos mais nenhuma formula que i
        kp = floor(kp/2) - [(NrOfVM - 1) 0]; %nossa!!! o proprio kp, que continha o kn inicial ficando assim: k

        DataB{j+(1)}.kd = kd;
        DataB{j+(1)}.kp = kp;
        DataB{j+(1)}.kn = kn;
end

%Ate aqui calculamos os indices kmin e kmax de cada escala
%Alem do numero final de coeficientes desejado e do numero de refinamentos (Toplevel) que devem ser passados co
%algoritmo, ainda nos resta calcular qtos coeficientes sao necessarios no nivel menos refinado para dar inicio
%reconstrucao. Obviamente este numero pode ser extraido de:
%Data{TopLev+(1)}.kp que neste instante estaria no vetor kp.


%%Calculo do mapa de variâncias
    for j=TopLev+1:-1:1,
    f = 0.5/2^(j-1)
    NJ= diff( Data{j}.kp ) + 1
    Cs=1
    GSRD1=10
    GSRD2=1
    GSRD3=5

    for n=1:+1:2^(j-1),

      n
```

```
      if j==TopLev+1,
            mapa{j,n} = (2^(j-1))*(Cs/(1-alpha)*f^(1-alpha)*(n^(1-alpha)-(n-1)^(1-alpha)));
              else
          if mod(n,2) == 0,
             m=2*n;
             mapa{j,n} = (mapa{j+1,m-1}+mapa{j+1,m})/2;
          else
             m=(2*n-1);
             mapa{j,n} = (mapa{j+1,m}+mapa{j+1,m+1})/2;
          end
       end
    end
    %mapa normalizado
    if j==TopLev+1,
        a=mapa{j,2}
        for n=1:+1:2^(j-1),
            mapa{j,n} = mapa{j,n}/a;
        end
    end



%calculo dos coeficientes da funcao de escala na escala menos refinada:
     DataB{TopLev+(1)}.app  = appProcess(diff(kp)+1, Lm, alpha);
     % passamos junto parametros adicionais dependentes do modelo (varargin) que devem ser tratados dentro da a

        for j=TopLev:-1:1,
            [DataB] = detProcess(j,TopLev,DataB,Lm,alpha,sinal);% calcula o detalhe da escala atual
            DataB{j-1+(1)}.app = Recon(j,NrOfVM,DataB,h,g); %reconstroi os coef. de aprox. do nivel imediatamen
        end

if (exist('nome_arquivo')==1),
    if length(nome_arquivo)>1,
        DataB{1}.app;
        ans';
        nome_arquivo=[nome_arquivo,'.txt'];
        comando2ex = sprintf('save %s ans -ASCII',nome_arquivo);
        eval(comando2ex);
        nome_arquivo=[];
    end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%interna:
function approxs = appProcess(n, p, alpha)
%function approxs = appProcess(n, modelo, testevariaveis)
        approxs = zeros(1,n); %tudo zero para aprox. do fGn

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%interna:
function [DataB] = detProcess(level, TopLev, DataB, p, alpha, sinal)


        ctrvariance = (mapa{(level+1),2})^(0.5);
```

```
    n = diff( DataB{level+(1)}.kp ) + 1;

    A=load(sinal);


   DataB{level+(1)}.det = ctrvariance*randn(1,n)+A(2^(TopLev+(1))-n-(n-1):2^(TopLev+(1))-n).'; %iid com a varia
```

```
% Sintese fGn com periodicidade via inserção de cosseno pelo método da DWPT

function [DataB] = Sintese_DWPT_cosseno_fgn_14(TopLev, NrOfVM, ninic, nsrdd, nsrdt, alpha, sinal) % h will be h

Lm=2^TopLev;

N = int2str(NrOfVM); %N e uma string
wname = 'db'; %falta concatenar com N para obter dbN
wname = strcat(wname,N)

g = dbwavf(wname); % uso da funcao dbaux seria mais simples pois aceitaria N como inteiro em vez de exigir a st
g = g * sqrt(2)
Lf = length(g) % precisa ser igual a 2N
h = fliplr(g); h = (-1).^(0:Lf-1).*h % representando a formula 75b do livro de wavelets

nome_arquivo = ['Cosseno+fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(TopLev),')

X=load(sinal);

  Lf = 2*NrOfVM; %comprimento do filtro
kn = [0 Lm-1]; %aqui dois extremos da serie que estamos interessados no nivel mais refinado - ira armazenar som
kp = kn; % ira armazenar extremos contendo non-polluted e polluted coefficients. Todos os coeficientes entre es
DataB{0+(1)}.kp = kn; %ou seja, esses sao os extremos que deverao ser gerados na escala mais refinada (0)
kc = kn;  %ira armazenar dois extremos da serie temporarios para a proxima escala
kd = kn; %ira armazenar dois extremos da serie consolidados para a proxima escala
DataB{0+(1)}.kd = kn;

%efetuando o calculo dos coeficientes em cada um dos extremos e que serao
%necessarios:
    for j=1:TopLev,
%for j=2:TopLev+1,
        %kd(1) e o kminimo da escala anterior
        %kd(2) e o kmax da escala anterior
        %Lf - 1 = 2N - 1
        kc = [kd(1)-(Lf-1) kd(2)];  % nao encontrei nenhuma formula que usa esses indices - temporario... nao a
        kd = fix (kc/2); % fix arredonda para zero... e na verdade deveria ser arredonda para baixo, nao? (floo
        % kd agora passou a conter os indices de kc, porem divididos por 2:
        %kd = [fix(kmin/2 - N + 1/2)) fix(kmax/2)] - isto parece, embora nao exatamente, com
        %as formulas 7.9 e 7.10 onde kd passa a representar uma escala menos
        %refinada. As diferencas estao no fix / floor e no 1/2 / 1

        kn = floor([0 kn(2)-(Lf-1)]/2); %[0 floor(kmax/2 - N + 1/2)]!!! -> nao temos mais nenhuma formula que i
        kp = floor(kp/2) - [(NrOfVM - 1) 0]; %nossa!!! o proprio kp, que continha o kn inicial ficando assim: k

         DataB{j+(1)}.kd = kd;
         DataB{j+(1)}.kp = kp;
         DataB{j+(1)}.kn = kn;
    end
```

```
%Ate aqui calculamos os indices kmin e kmax de cada escala
%Alem do numero final de coeficientes desejado e do numero de refinamentos (Toplevel) que devem ser passados co
%algoritmo, ainda nos resta calcular qtos coeficientes sao necessarios no nivel menos refinado para dar inicio
%reconstrucao. Obviamente este numero pode ser extraido de:
%Data{TopLev+(1)}.kp que neste instante estaria no vetor kp.

    %         n = 1; %inicia a variável n como zero pois o sinal é o mesmo que o coeficiente W(0,0)

    %ctrvariance = 2^( (level - TopLev) * alpha/2 );

    %%Calculo do mapa de variâncias
    for j=TopLev+1:-1:1,
    f = 0.5/2^(j-1)
    Cs=1
    GSRD1=10
    GSRD2=1
    GSRD3=5
    for n=1:+1:2^(j-1),
      if j==TopLev+1,
          mapa{j,n} = (2^(j-1))*(Cs/(1-alpha)*f^(1-alpha)*(n^(1-alpha)-(n-1)^(1-alpha)));
      else
          if mod(n,2) == 0,
             m=2*n;
             mapa{j,n} = (mapa{j+1,m-1}+mapa{j+1,m})/2;
          else
             m=(2*n-1);
             mapa{j,n} = (mapa{j+1,m}+mapa{j+1,m+1})/2;
          end
          %mapa{j,n} = mapa{j+1,n}*2^(-alpha);
      end
    end
    %mapa normalizado
    if j==TopLev+1,
     Y=mapa{j,2}
     for n=1:+1:2^(j-1),
       mapa{j,n} = mapa{j,n}/Y;
     end
    end


   if ninic <= 2^TopLev,
    if mod(ninic,2) == 0,
        m=ninic/2
    else
        m=(ninic+1)/2
    end

    for j=TopLev+1:-1:1,
        j
        o = diff( DataB{j}.kp ) + 1
        if j==TopLev+1,
            p = diff( DataB{j}.kp ) + 1
        end
        m
        f = 0.5/2^(j-1)
         % A convolucao:
```

```
if j == TopLev+1,

    if mod(m,2) == 0,
       if mod(ninic,2) == 0,
      ctrvariance{j,ninic} = mapa{j,ninic}^(0.5);
      A=randn(1,o);
      W{j,ninic} = ctrvariance{j,ninic}*A+X(Lm*TopLev+ninic);
      ctrvariance{j,ninic-1}=mapa{j,ninic-1}^(0.5);
      B=randn(1,o);
      W{j,ninic-1} = ctrvariance{j,ninic-1}*B+X(Lm*TopLev+ninic-1).';
      DataB{j}=[W{j,ninic-1} W{j,ninic}];

       else
         A=randn(1,o);
         ctrvariance{j,ninic} = mapa{j,ninic}^(0.5);
         W{j,ninic} = ctrvariance{j,ninic}*A+X(Lm*TopLev+ninic).';
         ctrvariance{j,ninic+1}=mapa{j,ninic+1}^(0.5);
         B=randn(1,o);
         W{j,ninic+1} = ctrvariance{j,ninic+1}*B+X(Lm*TopLev+ninic+1).';
         DataB{j}=[W{j,ninic} W{j,ninic+1}];
       end

    else
       if mod(ninic,2) == 0,

       ctrvariance{j,ninic}=mapa{j,ninic}^(0.5);
         A=randn(1,o);
       W{j,ninic} = ctrvariance{j,ninic}*A+X(Lm*TopLev+ninic).';
         ctrvariance{j,ninic-1}=mapa{j,ninic-1}^(0.5);
         B=randn(1,o);
         W{j,ninic-1}=ctrvariance{j,ninic-1}*B+X(Lm*TopLev+ninic-1).';
         DataB{j}=[W{j,ninic-1} W{j,ninic}];

        else
       ctrvariance{j,ninic} = mapa{j,ninic}^(0.5);
           A=randn(1,o);
           W{j,ninic} = ctrvariance{j,ninic}*A+X(Lm*TopLev+ninic).';
           ctrvariance{j,ninic+1} = mapa{j,ninic+1}^(0.5);
      B=randn(1,o);
      W{j,ninic+1}=ctrvariance{j,ninic+1}*B+X(Lm*TopLev+ninic+1).';
          DataB{j}=[W{j,ninic} W{j,ninic+1}];

       end
  end



else


    if mod(m,2) == 0,

       Wup{j+1,2*m-1} = dyadup(W{j+1,2*m-1},0);
       Wup{j+1,2*m} = dyadup(W{j+1,2*m},0);
       Wconv{j+1,2*m-1} = conv(Wup{j+1,2*m-1},h);
       % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostra
       Wconv{j+1,2*m} = conv(Wup{j+1,2*m},g);
```

```
% coefiiciente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
W{j,m}=Wconv{j+1,2*m-1}+Wconv{j+1,2*m};


if j==(TopLev+1)-1,
    %p = diff( DataB{(TopLev+1)}.kp ) + 1
    C=randn(1,p);
    ctrvariance{j+1,2*(m-1)-1} = mapa{j+1,2*(m-1)-1}^(0.5);
    W{j+1,2*(m-1)-1} = ctrvariance{j+1,2*(m-1)-1}*C+X(Lm*(j)+2*(m-1)-1).';

    D=randn(1,p);
    ctrvariance{j+1,2*(m-1)} = mapa{j+1,2*(m-1)}^(0.5);
    W{j+1,2*(m-1)} = ctrvariance{j+1,2*(m-1)}*D+X(Lm*(j)+2*(m-1)).';

else
    for i=TopLev+1:-1:j+1,
        if i==TopLev+1,
            for l=(2^(i-j)*(m-1)):-1:(2^(i-j)*(m-1)-2^(i-j)+1),
               % p = diff( DataB{(TopLev+1)}.kp ) + 1
                C=randn(1,p);
                ctrvariance{i,l} = mapa{i,l}^(0.5);
                W{i,l} = ctrvariance{i,l}*C+X(Lm*(i-1)+l).';
            end
        else
            for l=(2^(i-j)*(m-1)):-1:(2^(i-j)*(m-1)-2^(i-j)+1),
                if mod(l,2) == 0,
                    Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                    Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                    Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},h);
                    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                    Wconv{i+1,2*l} = conv(Wup{i+1,2*l},g);
                    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                    W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};
                else
                    Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                    Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                    Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},g);
                    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                    Wconv{i+1,2*l} = conv(Wup{i+1,2*l},h);
                    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                    W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};

                end
            end
        end
    end
end

Wup{j+1,2*(m-1)-1} = dyadup(W{j+1,2*(m-1)-1},0);
Wup{j+1,2*(m-1)} = dyadup(W{j+1,2*(m-1)},0);
Wconv{j+1,2*(m-1)-1} = conv(Wup{j+1,2*(m-1)-1},g);
% coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
Wconv{j+1,2*(m-1)} = conv(Wup{j+1,2*(m-1)},h);
% coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
W{j,m-1}=Wconv{j+1,2*(m-1)-1}+Wconv{j+1,2*(m-1)};
```

```
    DataB{j}=[W{j,m-1} W{j,m}];
    DataD{j}=[WSRD{j,m-1} WSRD{j,m}];

   m=m/2;

  else

    Wup{j+1,2*m-1} = dyadup(W{j+1,2*m-1},0);
    Wup{j+1,2*m} = dyadup(W{j+1,2*m},0);
    Wconv{j+1,2*m-1} = conv(Wup{j+1,2*m-1},g);
    % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
    Wconv{j+1,2*m} = conv(Wup{j+1,2*m},h);
    % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
    W{j,m}=Wconv{j+1,2*m-1}+Wconv{j+1,2*m};

  if j>1,
  if j==(TopLev+1)-1,
        %p = diff( DataB{}.kp ) + 1
        C=randn(1,p)
        ctrvariance{j+1,2*(m+1)-1} = mapa{j+1,2*(m+1)-1}^(0.5);
        W{j+1,2*(m+1)-1} = ctrvariance{j+1,2*(m+1)-1}*C+X(Lm*(j)+2*(m+1)-1).';

        D=randn(1,p);
        ctrvariance{j+1,2*(m+1)} = mapa{j+1,2*(m+1)}^(0.5);
        W{j+1,2*(m+1)} = ctrvariance{j+1,2*(m+1)}*D+X(Lm*(j)+2*(m+1)).';
  else
       for i=TopLev+1:-1:j+1,
            if i==TopLev+1,
                for l=(2^(i-j)*(m+1)):-1:(2^(i-j)*(m+1)-2^(i-j)+1),
                    %p = diff( DataB{(TopLev+1)}.kp ) + 1
                    C=randn(1,p);
                    ctrvariance{i,l} = mapa{i,l}^(0.5);
                    W{i,l} = ctrvariance{i,l}*C+X(Lm*(i-1)+l).';
                end
            else
                for l=(2^(i-j)*(m+1)):-1:(2^(i-j)*(m+1)-2^(i-j)+1),
                    if mod(l,2) == 0,
                        Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                        Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                        Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},h);
                        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                        Wconv{i+1,2*l} = conv(Wup{i+1,2*l},g);
                        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                        W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};

                    else
                        Wup{i+1,2*l-1} = dyadup(W{i+1,2*l-1},0);
                        Wup{i+1,2*l} = dyadup(W{i+1,2*l},0);
                        Wconv{i+1,2*l-1} = conv(Wup{i+1,2*l-1},g);
                        % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas
                        Wconv{i+1,2*l} = conv(Wup{i+1,2*l},h);
                        % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixa
                        W{i,l}=Wconv{i+1,2*l-1}+Wconv{i+1,2*l};
                    end
                end
            end
       end
```

```
                end


                  Wup{j+1,2*(m+1)-1} = dyadup(W{j+1,2*(m+1)-1},0);
                  Wup{j+1,2*(m+1)} = dyadup(W{j+1,2*(m+1)},0);
                  Wconv{j+1,2*(m+1)-1} = conv(Wup{j+1,2*(m+1)-1},h);
                  % coeficiente W(j-1,n) convoluido com os coeficientes do filtro passa-baixas e feita a subamost
                  Wconv{j+1,2*(m+1)} = conv(Wup{j+1,2*(m+1)},g);
                  % coeficiente W(j-1,n) convoluído com os coeficientes do filtro passa-altas e feita a subamostr
                  W{j,m+1}=Wconv{j+1,2*(m+1)-1}+Wconv{j+1,2*(m+1)};

                end

                 if j==1,
                     DataB{j}=[W{j,m}];
                 else
                     DataB{j}=[W{j,m} W{j,m+1}];
                 end
                 m=(m+1)/2;

               end

            end
        end




        else

            warning('O coeficiente escolhido não está dentro dos coeficientes de último nível para a DWPT');
        end

              for j=1:+1:TopLev+1,
              if j==1,
                  DataC{j} =mapa{j,1};
              else
                  DataC{j} =mapa{j,1};
                  for nl=2:+1:2^(j-1),
                  DataC{j} =[DataC{j} mapa{j,nl}];
                  end
              end
              end

    if (exist('nome_arquivo')==1),
        if length(nome_arquivo)>1,
            for j=1:+1:TopLev+1,
            DataB{j};
            ans';
            nome_arquivo=[nome_arquivo,'_nível',int2str(j),'.txt'];
            comando2ex = sprintf('save %s ans -ASCII',nome_arquivo);
            eval(comando2ex);
            nome_arquivo = ['Cosseno+fgnDWPT14_(',num2str(Lm),')pts_NrOfVM(',int2str(NrOfVM),')_TopLev(',int2str(To


            end
        end
```

```
end
```