View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repositório Institucional da UFSC

EDUARDO LUIZ ORTIZ BATISTA

FILTROS VOLTERRA ADAPTATIVOS: ESTRUTURAS INTERPOLADAS E MODELOS ESTOCÁSTICOS

FLORIANÓPOLIS

2009

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

FILTROS VOLTERRA ADAPTATIVOS: ESTRUTURAS INTERPOLADAS E MODELOS ESTOCÁSTICOS

Tese submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Doutor em Engenharia Elétrica.

EDUARDO LUIZ ORTIZ BATISTA

Florianópolis, Outubro de 2009.

FILTROS VOLTERRA ADAPTATIVOS: ESTRUTURAS INTERPOLADAS E MODELOS ESTOCÁSTICOS

Eduardo Luiz Ortiz Batista

'Esta Tese foi julgada adequada para obtenção do Título de Doutor em Engenharia Elétrica, Área de Concentração em *Comunicações e Processamento de Sinais*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Prof. Rui Seara, Dr.
Orientador

Prof. Roberto de Souza Salgado, Ph.D.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Rui Seara, Dr.
Presidente

Prof. Orlando José Tobias, Dr.
Co-Orientador

Prof. João Marcos Travassos Romano, Dr.

Prof. João César Moura Mota, Dr.

Prof. Hans Helmut Zürn, Ph.D.

Prof. Sidnei Noceti Filho, Dr.

AGRADECIMENTOS

Ao meu orientador, Prof. Rui Seara, por todos os ensinamentos, confiança, apoio e por todas as oportunidades que me proporcionou em todos os anos de trabalho em conjunto.

Ao meu co-orientador, Prof. Orlando J. Tobias, por todos os ensinamentos, apoio e excelente relacionamento ao longo dos últimos anos.

Aos membros da banca examinadora, por todas as críticas e sugestões que resultaram no aperfeiçoamento do trabalho final e dos trabalhos futuros.

Aos meus pais, Antônio e Joana, por todo amor, incentivo e por tudo que me ensinaram e continuam ensinando. Sem os valores e a educação que me proporcionaram, este trabalho não seria possível.

À minha mulher, Fran, por todo amor, carinho, companheirismo, apoio, compreensão e por estar ao meu lado todos os dias incentivando para que eu me torne uma pessoa melhor.

Ao meu irmão, Paulo, pela amizade, incentivo e companheirismo.

Ao amigo Elton L. Fontão, pelo essencial apoio e pelas oportunidades proporcionadas ao longo dos meus muitos anos de trabalho no LINSE.

Ao Walter Gontijo que, em conjunto com o Rafael Demetri, contribuiu com as pesquisas de cancelamento de eco apresentadas neste trabalho.

Aos professores do Programa de Pós-Graduação em Engenharia Elétrica da UFSC que contribuíram com o desenvolvimento deste trabalho e das publicações que dele resultaram, em especial ao Prof. Hans H. Zürn e ao Prof. Sidnei Noceti Filho.

A todos os amigos e colegas do LINSE, pelos muitos anos de excelente convivência.

A toda minha família e a todos meus amigos que acreditaram no sucesso deste trabalho.

Ao CNPq e à CAPES, pelo apoio financeiro.

Resumo da Tese submetida à Universidade Federal de Santa Catarina como parte dos requisitos necessários para obtenção do grau de Doutor em Engenharia Elétrica.

FILTROS VOLTERRA ADAPTATIVOS: ESTRUTURAS INTERPOLADAS E MODELOS ESTOCÁSTICOS

Eduardo Luiz Ortiz Batista

Outubro/2009

Orientador: Prof. Rui Seara, Dr.
Co-orientador: Prof. Orlando J. Tobias, Dr.
Área de Concentração: Comunicações e Processamento de Sinais.
Palavras-chave: Filtros adaptativos; filtros não-lineares; filtros Volterra; interpolação.
Número de páginas: 165.

RESUMO: Este trabalho de pesquisa visa o estudo de filtros Volterra adaptativos objetivando desenvolver novas estruturas que apresentem um bom compromisso entre desempenho e complexidade computacional. Nesse contexto, abordagens interpoladas são consideradas em filtros Volterra adaptativos. Tais abordagens utilizam um filtro esparso (visando reduzir a complexidade) combinado com um interpolador para recriar os coeficientes zerados do filtro esparso. Dessa forma, diversas estruturas para a implementação eficiente de filtros Volterra adaptativos são obtidas. Algumas características importantes das estruturas interpoladas, como, por exemplo, a existência de um efeito de borda indesejável, são evidenciadas e estudas. Com vistas ao efeito de borda, um procedimento para a sua remoção é apresentado, melhorando consideravelmente o desempenho de filtros interpolados adaptativos tanto nos casos lineares quanto nos Volterra. Adicionalmente, são estudas estruturas interpoladas inteiramente adaptativas considerando filtros lineares como também filtros Volterra. Análises estatísticas de algumas estruturas Volterra adaptativas são também apresentadas com o objetivo de obter modelos estocásticos de primeira e segunda ordens para predizer o comportamento de tais estruturas. Exemplos de aplicações são considerados visando avaliar tanto os novos algoritmos desenvolvidos quanto seus modelos estatísticos.

Abstract of Thesis presented to Federal University of Santa Catarina as a partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.

ADAPTIVE VOLTERRA FILTERS: INTERPOLATED STRUCTURES AND STOCHASTIC MODELS

Eduardo Luiz Ortiz Batista

October/2009

Advisor: Prof. Rui Seara, Dr.
Co-advisor: Prof. Orlando J. Tobias, Dr.
Area of Concentration: Communications and Signal Processing.
Keywords: Adaptive filters; nonlinear filters; Volterra filters; interpolation.
Number of Pages: 165.

ABSTRACT: This research work aims to study adaptive Volterra filters for obtaining new structures presenting a good trade-off between performance and computational complexity. In this context, interpolated approaches are considered for adaptive Volterra filtering. Such approaches use a sparse filter (for complexity reduction) associated with an interpolator for recreating the zeroed coefficients of the sparse filter. Thus, different adaptive Volterra structures are obtained aiming at efficient implementation. Some important features of the interpolated structures, such as the existence of an undesirable boundary effect, are highlighted and studied. Concerning the boundary effect, a procedure for its removal is presented, improving considerably the performance of the adaptive interpolated filters for both linear and Volterra cases. In addition, fully adaptive interpolated structures are studied considering linear filters as well as Volterra filters. Statistical analyses of some adaptive Volterra structures are also presented to obtain first and second order stochastic models for predicting the behavior of such structures. Application examples are considered to assess the developed algorithms as well as their statistical models for performance.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABIFIR	*adaptive boundaryless interpolated finite impulse response*
AFIR	*adaptive finite impulse response*
AIFIR	*adaptive interpolated finite impulse response*
AP	*affine projection*
APIV	*adaptive partially interpolated Volterra*
AR	*autoregressive*
ATA	adaptador de telefonia analógica
BIFIR	*boundaryless interpolated finite impulse response*
CFAIFIR	*complete fully adaptive interpolated finite impulse response*
DSL	*digital subscriber line*
DSP	*digital signal processor*
EQM	erro quadrático médio
ERLE	*echo return loss enhancement*
FABIFIR	*fully adaptive boundaryless interpolated finite impulse response*
FAIFIR	*fully adaptive interpolated finite impulse response*
FAIV	*fully adaptive interpolated Volterra*
FAPIV	*fully adaptive partially interpolated Volterra*
FFT	*fast Fourier transform*
FIR	*finite impulse response*
IEEE	*Institute of Electrical and Electronics Engineers*
IFIR	*interpolated finite impulse response*
IIR	*infinite impulse response*
IP	*internet protocol*
ITU	*International Telecommunication Union*
LMS	*least-mean-square*
MFABIFIR	*modified fully adaptive boundaryless interpolated finite impulse response*
MIT	*Massachusetts Institute of Technology*
NLMS	*normalized least-mean-square*
OFDM	*orthogonal frequency-division multiplexing*
PAM	*pulse amplitude modulation*
RLS	*recursive least-squares*
VoIP	*voice over internet protocol*

CAPÍTULO 1

Introdução

A filtragem adaptativa é um tópico de processamento digital de sinais com diversas aplicações práticas [1]-[2], a saber identificação de sistemas, equalização de canais, cancelamento de interferências, cancelamento de eco e controle ativo de ruído e vibração. O surgimento de processadores digitais de sinais (*digital signal processors* - DSPs) de alto desempenho, ocorrido nas últimas décadas, vem propiciando o uso de filtros adaptativos em uma gama cada vez maior de aplicações. Um bom exemplo é a utilização de tais filtros em problemas envolvendo não-linearidades, que são, por natureza, de maior complexidade. Aliado a tal avanço tecnológico resta também um grande interesse no desenvolvimento de pesquisas, o que tem levado ao constante surgimento de soluções adaptativas inovadoras. Assim, os filtros adaptativos têm ocupado um espaço cada vez maior na área de processamento digital de sinais devido, principalmente, a sua crescente aplicabilidade.

1.1 Filtros Adaptativos

No contexto de processamento de sinais, os filtros podem ser entendidos como dispositivos capazes de remover componentes indesejados de um sinal. Dessa maneira, conhecidas as características do sinal a ser filtrado (sinal de entrada), as especificações da filtragem requerida e a estrutura ou tipo do filtro a ser utilizado, é possível obter o projeto de um filtro ótimo satisfazendo algum dado critério de desempenho. Entretanto, em muitas aplicações práticas o conhecimento das características do sinal de entrada não é totalmente disponível, tornando assim inviável o projeto de um filtro ótimo *a priori*. A solução para tal problema pode ser o uso de um filtro adaptativo. Esse filtro possui a capacidade de se adaptar ou se autoprojetar através do uso de um algoritmo adaptativo recursivo. Dessa forma, torna-se possível que o filtro apresente um desempenho mais satisfatório em um ambiente em que as características do sinal a ser filtrado não são completamente conhecidas ou, ainda, onde tais características sejam variantes no tempo.

Na Figura 1.1, é apresentado um diagrama típico de um filtro adaptativo aplicado a um problema de estimação de sinal. Para esse problema, deseja-se estimar o sinal $d(n)$ a partir de um sinal de entrada $x(n)$, assumindo que existe alguma correlação entre ambos. Assim, utiliza-se um algoritmo adaptativo (recursivo) para otimizar os parâmetros (coeficientes) do filtro em relação a algum dado critério de desempenho.

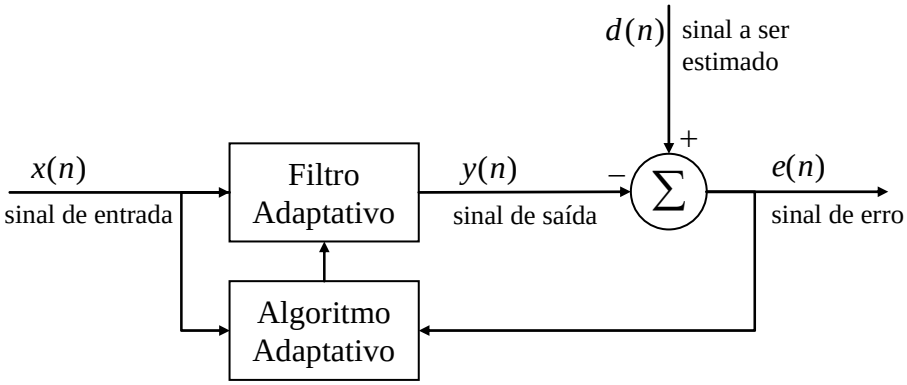


Figura 1.1. Diagrama de blocos de um filtro adaptativo no contexto de um problema de estimação de sinal a partir de um sinal de entrada.

Os primeiros trabalhos de pesquisa na área de filtragem adaptativa foram desenvolvidos por Bernard Widrow [3] e deram origem ao algoritmo *least-mean-square* (LMS). Tal algoritmo e suas variações são ainda hoje muito utilizados e estudados. Desse algoritmo temos um bom exemplo de critério de desempenho (função custo) utilizado para a otimização dos coeficientes do filtro adaptativo. Trata-se do valor quadrático médio do sinal de erro ou o erro quadrático médio (EQM).

1.2 Filtros Adaptativos Lineares

Um sistema é considerado linear se ele satisfaz o princípio da superposição [4]. Conforme comentado na seção anterior, o uso de filtros adaptativos envolve a aplicação de um algoritmo recursivo para ajustar os coeficientes a cada iteração. Como conseqüência, tem-se que um filtro adaptativo não é, a rigor, um sistema linear uma vez que, devido à contínua variação dos seus coeficientes, o princípio da superposição [4] não é satisfeito. Apesar disso, os filtros adaptativos são comumente classificados como lineares se sua relação de entrada e saída obedece ao princípio da superposição considerando fixos os valores de seus coeficientes [2]. Portanto, um filtro adaptativo é considerado linear se sua estrutura é linear desconsiderada a sua característica adaptativa.

Existem diversas estruturas para a implementação de filtros lineares. Tais estruturas podem ser classificadas em dois grupos [2]: estruturas com resposta ao impulso finita (*finite impulse response* - FIR) e estruturas com resposta ao impulso infinita (*infinite impulse response* - IIR). As estruturas FIR são as mais comumente utilizadas em filtragem adaptativa por não apresentarem problemas de estabilidade oriundos das estruturas IIR. Dentre as estruturas FIR, os filtros transversais ou filtros de linha de atrasos são os mais utilizados, sendo muitas vezes referidos apenas como filtros FIR. A relação de entrada e saída dessa estrutura é dada por

$$y(n) = \sum_{i=0}^{N-1} w(i,n)x(n-i) \tag{1.1}$$

onde N representa o tamanho de memória do filtro, $w(i,n)$ os coeficientes do filtro, $x(n)$ o sinal de entrada e $y(n)$ o sinal de saída. Esta relação pode ainda ser escrita na forma vetorial, resultando em

$$y(n) = \mathbf{w}^{\mathrm{T}}(n)\mathbf{x}(n) \tag{1.2}$$

com

$$\mathbf{w}(n) = [w(0,n) \;\; w(1,n) \;\; \cdots \;\; w(N-1,n)]^{\mathrm{T}} \tag{1.3}$$

e

$$\mathbf{x}(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-N+1)]^{\mathrm{T}}. \tag{1.4}$$

Considerando seja (1.1) ou (1.2), é possível notar que o filtro FIR transversal constitui-se de uma estrutura bastante simples com grande facilidade de implementação. Tal característica, aliada a uma base teórica muito bem estabelecida, tornou o filtro transversal uma estrutura essencial no processamento digital de sinais, sendo utilizado em uma grande variedade de aplicações com desempenho bastante satisfatório [1]-[2]. Por outro lado, em aplicações nas quais valores elevados de tamanho de memória N são requeridos, o uso dos filtros FIR transversais sofre sérias restrições. Isso ocorre pois, nesses casos, os filtros transversais apresentam um grande número de coeficientes, resultando em um elevado custo computacional de implementação. Nesse contexto, implementações com complexidade reduzida de filtros FIR, como, por exemplo, as implementações interpoladas [5], têm um papel muito importante, permitindo o uso de tais filtros em um maior contexto de aplicações.

1.3 Filtros Adaptativos Não-Lineares

Como apresentado na seção anterior, os filtros adaptativos lineares são bastante simples do ponto de vista de implementação e por isso sua aplicação é bem difundida e estudada. No entanto, algumas aplicações requerendo o uso de filtros adaptativos possuem dispositivos não-lineares em sua estrutura. Um bom exemplo são aplicações que envolvem elementos apresentando características de saturação. Para esse tipo de estrutura um filtro adaptativo não-linear passa a ser uma solução muito mais indicada ou mesmo indispensável dependendo dos requisitos de desempenho pretendidos. Alguns exemplos de aplicações nas quais os filtros adaptativos não-lineares são atualmente utilizados com sucesso são [6]: equalização de sistemas de comunicação por satélite, processamento de sinais dependente da percepção, estudo das distorções harmônicas em alto-falantes, tratamento de imagens com ruído, estudo do movimento de navios ancorados devido às ondas oceânicas, estudo de distorções em sistemas de gravação magnéticos, dentre outras.

Um sistema não-linear é definido como um sistema que não obedece ao princípio da superposição [4]. Trata-se de uma definição bastante ampla e, como conseqüência, é praticamente impossível desenvolver uma base teórica que seja aplicável a todos os sistemas não-lineares. Assim, os sistemas não-lineares são divididos em classes, em geral baseadas nas funções não-lineares que descrevem o seu funcionamento, cada uma apresentando características, aplicações e desenvolvimento teórico próprios. Alguns exemplos de tais classes de sistemas considerados em engenharia são os seguintes [6]:

- Sistemas Homomórficos
- Filtros Morfológicos
- Redes Neurais
- Filtros Polinomiais

No contexto adaptativo, a procura por parâmetros que otimizem a função custo de um sistema ou filtro não-linear é geralmente lenta e difícil. Isso ocorre pois soluções analíticas simples geralmente não são possíveis, ao contrário do que acontece com os filtros lineares. Assim, as implementações adaptativas de filtros não-lineares geralmente requerem um grande esforço computacional. Com o aumento da capacidade de processamento dos DSPs ocorrido nas últimas décadas, tornou-se possível a aplicação de filtragem adaptativa não-linear em maior escala. No entanto, muitos filtros adaptativos não-lineares têm sua complexidade exponencialmente dependente do tamanho de memória

de tal forma que, mesmo com uma necessidade de memória relativamente pequena, uma quantidade muito grande de operações por amostra é requerida. Esse é um problema que afeta sobremaneira os filtros não-lineares polinomiais, como, por exemplo, os filtros Volterra [6]-[7]. Outro problema enfrentado no uso dos filtros polinomiais e, particularmente, dos filtros Volterra, é a baixa taxa de convergência que os algoritmos adaptativos tendem a apresentar. Além disso, a escassez de ferramentas de análise e síntese também dificulta o uso de tais filtros. Para resolver esses problemas muito esforço de pesquisa ainda precisa ser realizado. O desenvolvimento de implementações de complexidade reduzida [8], algoritmos com melhor desempenho de convergência e análises teóricas dos filtros adaptativos não-lineares [9]-[10] são bons exemplos do que pode ser feito para prover pesquisadores e engenheiros de ferramentas comparáveis às existentes para filtros lineares.

1.4 Objetivos do Trabalho

O objetivo deste trabalho de pesquisa é tratar algumas das dificuldades ainda existentes para o uso dos filtros Volterra em problemas de filtragem adaptativa. Dentre tais dificuldades, destaca-se, primeiramente, a necessidade do desenvolvimento de uma base teórica mais abrangente para os filtros Volterra adaptativos, sendo esse o assunto principal do próximo capítulo deste trabalho. Outra importante dificuldade encontrada em muitas aplicações adaptativas dos filtros Volterra e, algumas vezes, dos filtros FIR, é a elevada carga computacional requerida. Com vistas a tal dificuldade, o foco deste trabalho é o desenvolvimento e o aperfeiçoamento das abordagens interpoladas para implementação de filtros FIR e Volterra adaptativos. Nesse contexto, busca-se obter melhorarias de desempenho e também o desenvolvimento de uma sólida base teórica para, com isso, ampliar a aplicabilidade dos filtros interpolados adaptativos.

1.5 Organização do Trabalho

Este trabalho está organizado conforme descrito a seguir. No Capítulo 2, os filtros Volterra são apresentados em detalhes, procurando destacar as suas principais características e também as diferentes formas de representação da sua relação de entrada e saída. Ainda no Capítulo 2, uma análise estatística do filtro Volterra adaptado pelo algoritmo LMS é apresentada. O Capítulo 3 trata dos filtros Volterra interpolados

adaptativos, com destaque inicial para a descrição matemática da sua relação de entrada e saída e para uma análise descrevendo a forma como os coeficientes são recriados através do processo de interpolação. São apresentados, também no Capítulo 3, um procedimento para implementação dos filtros interpolados sem o efeito de borda indesejável, resultante do processo de interpolação, e ainda uma análise estatística dos filtros Volterra interpolados adaptativos baseada em uma abordagem com restrições. O Capítulo 4 trata das implementações inteiramente adaptativas dos filtros interpolados, nas quais tanto o interpolador quanto o filtro esparso, que compõem a estrutura interpolada, são adaptados. Nesse contexto, uma base teórica para o problema da adaptação de filtros FIR em cascata é apresentada, com sua posterior aplicação para o desenvolvimento de implementações de estruturas interpoladas inteiramente adaptativas mais eficientes de filtros FIR e Volterra. No Capítulo 5, as diferentes abordagens e a base teórica desenvolvida nos Capítulos 2 a 4 são utilizadas para o desenvolvimento de implementações interpoladas específicas visando o cancelamento de eco de linha em comunicações de dados e voz. Finalmente, no Capítulo 6, as conclusões gerais do trabalho são apresentadas.

1.6 Publicações da Tese

Este trabalho de tese originou às seguintes publicações:

I. E. L. O. Batista, O. J. Tobias, and R. Seara, “A mathematical framework to describe interpolated adaptive Volterra filters,” in *Proc. IEEE Int. Telecomm. Symp*., Fortaleza, Brazil, Sept. 2006, pp. 144-149.

II. E. L. O. Batista, O. J. Tobias, and R. Seara, “Border effect removal for IFIR and interpolated Volterra filters,” in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Honolulu, USA, vol. 3, Apr. 2007, pp. 1329-1332.

III. E. L. O. Batista, O. J. Tobias, and R. Seara, “Stochastic model of the LMS Volterra filter,” in *Proc. European Signal Processing Conf. (EUSIPCO)*, Poznan, Poland, Sept. 2007, pp. 1721-1725.

IV. E. L. O. Batista, O. J. Tobias, and R. Seara, “New insights in adaptive cascaded FIR structure: application to fully adaptive interpolated FIR structures,” in *Proc. European Signal Processing Conf. (EUSIPCO)*, Poznan, Poland, Sept. 2007, pp. 370-374.

V. E. L. O. Batista, O. J. Tobias, and Rui Seara, “Filtros FIR adaptativos em cascata: aplicação em filtros IFIR adaptativos,” in *Anais do XXV Simpósio Brasileiro de Telecomunicações*, Recife-CE, Brazil, CD-ROM, Sept. 2007.

VI. E. L. O. Batista, O. J. Tobias, and Rui Seara, “Filtros IFIR Inteiramente Adaptativos sem Efeito de Borda,” in *Anais do XXV Simpósio Brasileiro de Telecomunicações*, Recife-CE, Brazil, CD-ROM, Sept. 2007.

VII. E. L. O. Batista, O. J. Tobias, and R. Seara, “A fully adaptive IFIR filter with removed border effect,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Apr. 2008, pp. 3821-3824.

VIII. E. L. O. Batista, O. J. Tobias, and R. Seara, “A fully LMS adaptive interpolated Volterra structure,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Apr. 2008, pp. 3613-3616.

IX. E. L. O. Batista, O. J. Tobias, and R. Seara, “Fully adaptive LMS/NLMS interpolated Volterra filters with removed boundary effect,” in *Proc. European Signal Processing Conf. (EUSIPCO)*, Glasgow, Scotland, Aug. 2009, pp. 1725-1729.

X. E. L. O. Batista, O. J. Tobias, and R. Seara, “Filtros FIR-IFIR sem Efeito de Borda para o Cancelamento de Eco de Linha em Sistemas DSL,” in *Anais do XXV Simpósio Brasileiro de Telecomunicações*, Blumenau-SC, Brazil, CD-ROM, Sept. 2009.

XI. E. L. O. Batista, O. J. Tobias, and R. Seara, “A Sparse-Interpolated Scheme for Implementing Adaptive Volterra Filters,” *IEEE Trans. on Signal Processing*, vol. 58, no. 4, pp. 2022-2035, Apr. 2010.

CAPÍTULO 2

Filtro Volterra Adaptativo

Em filtragem adaptativa, uma das estruturas mais utilizadas para lidar com problemas envolvendo sistemas não-lineares é o filtro Volterra. Neste capítulo, as principais características de tal filtro são descritas como também uma análise de seu comportamento quando adaptado pelo algoritmo LMS é apresentada. Como introdução, uma breve descrição histórica da origem e aplicação do filtro Volterra adaptativo é apresentada na próxima seção.

2.1 História

A história do desenvolvimento do filtro Volterra adaptativo pode ser ilustrada, de forma resumida, pela linha do tempo apresentada na Figura 2.1. Pelo que conhecemos, o primeiro passo foi dado no final do século XIX pelo matemático italiano Vito Volterra com o desenvolvimento do que hoje se denomina série de Volterra [11]. O uso de tal série para o estudo de sistemas não-lineares ocorreu na década de 1940 com os trabalhos desenvolvidos por Norbert Wiener [12]-[13] no *Massachussets Institute of Technology* (MIT). Nas décadas de 1960 e 1970 surgiram os primeiros artigos discutindo o uso de expansões em série de Volterra visando o estudo de sistemas discretos [6], [14]. No início dos anos 1980, os primeiros trabalhos considerando o filtro Volterra no contexto adaptativo foram publicados, com destaque para a sua primeira aplicação associada ao algoritmo LMS, concebida por Coker e Simkins [15]. Desde então, uma grande evolução vem sendo observada no uso do filtro Volterra adaptativo. Alguns exemplos de aplicações para as quais filtros Volterra adaptativos têm sido atualmente utilizados com sucesso são: cancelamento de eco acústico [16]-[17], controle ativo de processos [7], equalização de canais de satélite [18], identificação e redução de distorções em alto-falantes [19], e compensação dos efeitos não-lineares em transmissores OFDM [20].

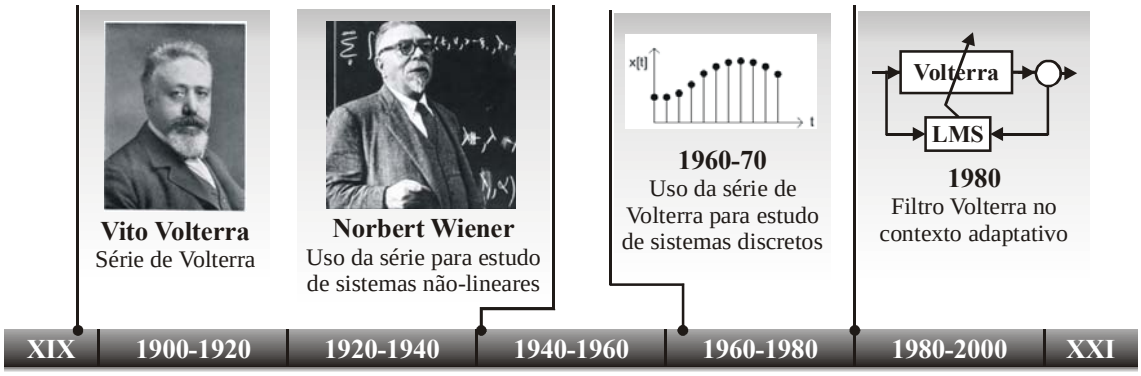


Figura 2.1. Linha de tempo: da série de Volterra ao filtro Volterra adaptativo.

2.2 Filtro Volterra

A relação de entrada e saída de um filtro Volterra causal e discreto é dada por [6]

$$\begin{aligned} y(n) &= h_0 + \sum_{m_1=0}^{N-1} h_1(m_1)x(n-m_1) \\ &+ \sum_{m_1=0}^{N-1}\sum_{m_2=0}^{N-1} h_2(m_1,m_2)x(n-m_1)x(n-m_2) + \dots \\ &+ \sum_{m_1=0}^{N-1}\cdots\sum_{m_P=0}^{N-1} h_P(m_1,\cdots,m_P)x(n-m_1)\cdots x(n-m_P) \end{aligned} \tag{2.1}$$

onde $x(n)$ e $y(n)$ são, respectivamente, os sinais de entrada e saída; $h_p(m_1,\cdots,m_p)$ representa os coeficientes de ordem p; N é o tamanho de memória do filtro; e P, a ordem do filtro. Conforme descrito em [6], no contexto adaptativo o termo h_0 em (2.1) pode geralmente ser tratado em separado da estrutura de filtragem. Assim, em geral assume-se $h_0 = 0$ sem perda de generalidade [6]. Ao longo deste trabalho, a relação de entrada e saída do filtro Volterra na forma de (2.1) é denominada representação convencional em distinção às outras representações posteriormente apresentadas.

2.2.1 Estrutura em Blocos

A partir de (2.1), é possível constatar que o filtro Volterra é composto por um bloco de primeira ordem (linear), equivalente a um filtro transversal [1] e cujos coeficientes são denotados por $h_1(m_1)$, seguido de blocos não-lineares cujos coeficientes são dados por $h_2(m_1,m_2)$, $h_3(m_1,m_2,m_3),\cdots,$ $h_P(m_1,\cdots,m_P)$. Essa estruturação em blocos de um filtro Volterra, evidenciada a partir de (2.1), é ilustrada na Figura 2.2.

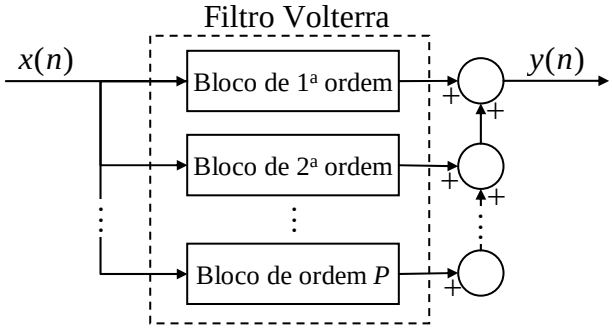


Figura 2.2. Estrutura em blocos de um Filtro Volterra.

Denotando a saída de um bloco de ordem p por $y_p(n)$, podemos reescrever (2.1) como segue:

$$y(n) = \sum_{p=1}^{P} y_p(n) \tag{2.2}$$

com $y_p(n)$ dado por

$$y_p(n) = \sum_{m_1=0}^{N-1} \sum_{m_2=0}^{N-1} \cdots \sum_{m_p=0}^{N-1} h_p(m_1, m_2, \ldots, m_p) \times \prod_{k=1}^{p} x(n-m_k). \tag{2.3}$$

Do ponto de vista prático, a estrutura em blocos do filtro Volterra é bastante vantajosa. Por exemplo, em algumas aplicações é possível desconsiderar blocos que se mostrem desnecessários. Além disso, estratégias de redução de complexidade podem ser aplicadas apenas aos blocos que demandem um maior número de coeficientes, preservando, assim, as características dos demais blocos.

2.2.2 Relação de Entrada e Saída Vetorial

A saída de cada um dos blocos de um filtro Volterra pode ser escrita na forma vetorial. Essa abordagem é bem conhecida para o bloco de primeira ordem (filtro transversal), resultando em

$$y_1(n) = \mathbf{h}_1^{\mathrm{T}} \mathbf{x}_1(n) \tag{2.4}$$

onde $\mathbf{x}_1(n)$ representa o vetor de entrada de primeira ordem, dado por

$$\mathbf{x}_1(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-N+1)]^{\mathrm{T}} \tag{2.5}$$

e $\mathbf{h}_1$, o vetor de coeficientes, dado por

$$\mathbf{h}_1 = [h_1(0) \;\; h_1(1) \;\; \cdots \;\; h_1(N-1)]^{\mathrm{T}}. \tag{2.6}$$

Para os demais blocos, um procedimento análogo é utilizado [6], considerando que o vetor de entrada de ordem p pode ser obtido recursivamente por

$$\mathbf{x}_p(n) = \mathbf{x}_1(n) \otimes \mathbf{x}_{p-1}(n) \tag{2.7}$$

onde $\otimes$ representa o produto de Kronecker [6], [21]. Com isso, o vetor de entrada de segunda ordem pode ser escrito como

$$\mathbf{x}_2(n) = \mathbf{x}_1(n) \otimes \mathbf{x}_1(n) \tag{2.8}$$

o que resulta em

$$\begin{aligned}\mathbf{x}_2(n) = [&x^2(n) \;\; x(n)x(n-1) \;\; \cdots \;\; x(n)x(n-N+1) \\ &x(n-1)x(n) \;\; x^2(n-1) \;\; \cdots \;\; x^2(n-N+1)]^{\mathrm{T}}.\end{aligned} \tag{2.9}$$

O vetor de coeficientes correspondente é, agora, expresso como

$$\mathbf{h}_2 = [h_2(0,0) \;\; h_2(0,1) \;\; \cdots \;\; h_2(0,N-1) \;\; h_2(1,0) \;\; h_2(1,1) \;\; \cdots \;\; h_2(N-1,N-1)]^{\mathrm{T}} \tag{2.10}$$

resultando na seguinte relação de entrada e saída para o bloco de segunda ordem:

$$y_2(n) = \mathbf{x}_2^{\mathrm{T}}(n)\mathbf{h}_2. \tag{2.11}$$

O mesmo procedimento pode ser realizado para os demais blocos não-lineares do filtro Volterra, resultando, para um bloco de ordem p qualquer, na seguinte relação:

$$y_p(n) = \mathbf{h}_p^{\mathrm{T}}\mathbf{x}_p(n) \tag{2.12}$$

com $y_p(n)$ representando o sinal de saída do bloco de ordem p e $\mathbf{h}_p$ o vetor de coeficientes desse bloco. Definindo então o vetor de entrada do filtro Volterra como

$$\mathbf{x}_{\mathrm{V}}(n) = [\mathbf{x}_1^{\mathrm{T}}(n) \;\; \mathbf{x}_2^{\mathrm{T}}(n) \;\; \cdots \;\; \mathbf{x}_P^{\mathrm{T}}(n)]^{\mathrm{T}} \tag{2.13}$$

e o vetor de coeficientes por

$$\mathbf{h}_{\mathrm{V}} = [\mathbf{h}_1^{\mathrm{T}} \;\; \mathbf{h}_2^{\mathrm{T}} \;\; \cdots \;\; \mathbf{h}_P^{\mathrm{T}}]^{\mathrm{T}} \tag{2.14}$$

a relação de entrada e saída (2.1) pode ser reescrita da seguinte maneira:

$$y(n) = \mathbf{h}_{\mathrm{V}}^{\mathrm{T}}\mathbf{x}_{\mathrm{V}}(n). \tag{2.15}$$

Nota-se, a partir de (2.15), que a operação de filtragem do filtro Volterra pode ser descrita através do produto interno entre dois vetores. Essa característica é muito importante por simplificar consideravelmente o tratamento matemático em procedimentos de análise de tal filtro. Além disso, observa-se, dessa expressão, a relação linear entre o sinal de saída $y(n)$ e o vetor de coeficientes $\mathbf{h}_{\mathrm{V}}$, permitindo a aplicação direta dos algoritmos adaptativos, usualmente considerados em filtros lineares também em filtros Volterra.

2.2.3 Representação Espacial dos Blocos

Uma maneira alternativa de visualizar os coeficientes dos blocos do filtro Volterra é através da representação espacial. Tal representação é obtida considerando os índices $m_1, m_2, \ldots, m_P$ dos coeficientes como coordenadas de um espaço euclidiano [22]. Assim, a representação espacial de cada bloco possui dimensão igual à ordem do correspondente bloco. Para o bloco de primeira ordem, a representação espacial é unidimensional, coincidindo com a representação convencional de (2.6). No caso do bloco de segunda ordem, cada coeficiente possui dois índices, sendo sua representação espacial bidimensional na forma de uma matriz de coeficientes. Assim, a partir de (2.10), obtém-se

$$\mathbf{H}_2 = \begin{bmatrix} h_2(0,0) & h_2(0,1) & \cdots & h_2(0,N-1) \\ h_2(1,0) & h_2(1,1) & \cdots & h_2(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ h_2(N-1,0) & h_2(N-1,1) & \cdots & h_2(N-1,N-1) \end{bmatrix}. \tag{2.16}$$

É possível ainda estabelecer uma relação matemática entre (2.10) e (2.16) [6]. Tal relação é dada por

$$\mathbf{h}_2 = \mathrm{vec}\left(\mathbf{H}_2^{\mathrm{T}}\right) \tag{2.17}$$

onde $\mathrm{vec}(\cdot)$ é um operador que, a partir da realização de um empilhamento de colunas, converte uma matriz em um vetor (coluna) [21], [23]. Para o bloco de terceira ordem, a representação espacial é tridimensional na forma de um cubo. Para exemplificar, é ilustrada na Figura 2.3 a representação espacial dos coeficientes do bloco de terceira ordem de um filtro Volterra com tamanho de memória $N = 3$. Apesar de tal representação (tridimensional) não possibilitar um tratamento matemático simples como o verificado para os blocos de primeira e segunda ordens, ela é ainda bastante interessante do ponto de vista analítico, pois, muitas vezes, permite observar características dos sistemas que não estão evidenciadas a partir da análise apenas do vetor de coeficientes de terceira ordem. Para blocos de ordens superiores, as estruturas hipercúbicas (com quatro ou mais dimensões) resultantes da representação espacial não são mais passíveis de serem visualizadas. Porém, mesmo para esses casos, o conceito da representação espacial é ainda importante do ponto de vista analítico.

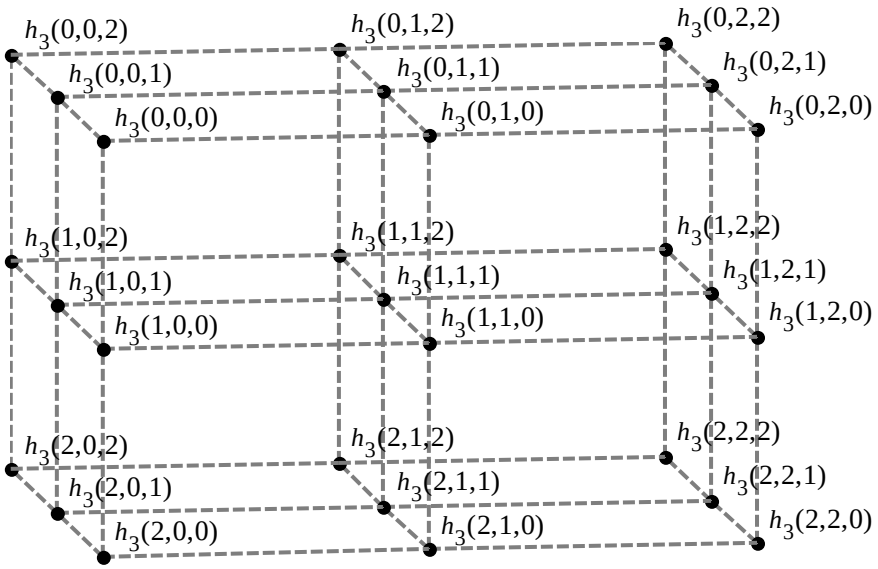


Figura 2.3. Representação espacial dos coeficientes de um bloco de terceira ordem de um filtro Volterra.

2.2.4 Complexidade Computacional

Como podemos observar em (2.6), o bloco de primeira ordem do filtro Volterra apresenta um número de coeficientes igual ao tamanho de memória do filtro N. Considerando o bloco de segunda ordem, temos, a partir de (2.8), que o seu vetor de entrada é obtido através do produto Kronecker do vetor de primeira ordem com ele próprio. De [6], temos que o produto Kronecker entre uma matriz $\mathbf{A}$ de dimensão $L_1 \times M_1$ e uma matriz $\mathbf{B}$ de dimensão $L_2 \times M_2$ resulta em uma matriz de dimensão $L_1 L_2 \times M_1 M_2$. Assim, levando em conta que o vetor de entrada de primeira ordem $\mathbf{x}_1(n)$ possui dimensão N, o vetor de entrada de segunda ordem $\mathbf{x}_2(n)$ terá dimensão N^2. Dessa forma, o vetor de coeficientes de segunda ordem possui N^2 coeficientes. Generalizando, o número de coeficientes para um bloco de ordem p em função do tamanho de memória N do filtro é

$$D_p(N) = N^p. \tag{2.18}$$

Assim, o número total de coeficientes de um filtro Volterra de ordem P e tamanho de memória N é

$$D_{\mathrm{V}}(N,P) = \sum_{k=1}^{P} N^k = \frac{N^{P+1} - N}{N-1}. \tag{2.19}$$

A partir de (2.18) e (2.19), é possível verificar que o número de coeficientes de um filtro Volterra cresce exponencialmente com o aumento do tamanho de memória. Como é de conhecimento geral, a complexidade computacional necessária para a implementação de um filtro ou de sua versão adaptativa está intimamente associada com o número total de coeficientes desse filtro. Dessa forma, algumas vezes os requisitos de tamanho de memória e ordem do filtro para uma determinada aplicação podem dificultar ou até mesmo inviabilizar o uso de uma estrutura Volterra. Esse número elevado de coeficientes pode também restringir o uso de algoritmos adaptativos de maior complexidade, tais como o algoritmo RLS (*recursive least-squares*) [2] e o algoritmo de projeções afins (AP) [2].

2.2.5 Representação Triangular Equivalente e Redução de Complexidade

A partir de (2.1), nota-se que em um bloco de ordem p de um filtro Volterra cada permutação dos índices $m_1, m_2, ..., m_p$ resulta em um diferente coeficiente. Entretanto, em uma operação de filtragem esses coeficientes com índices permutados são multiplicados pelo mesmo produto de amostras do sinal de entrada, dado por $x(n-m_1)x(n-m_2)\cdots x(n-m_p)$. Assim, é possível considerar a soma dos coeficientes cujos índices são dados pelas diferentes permutações de $m_1, m_2, ..., m_p$ como um único coeficiente que multiplica o produto de amostras $x(n-m_1)x(n-m_2)\cdots x(n-m_p)$. Para exemplificar, o bloco de segunda ordem possui os coeficientes $h_2(0,1)$ e $h_2(1,0)$, ambos multiplicados por $x(n)x(n-1) = x(n-1)x(n)$. Definindo, agora, $\underline{h}_2(0,1) = h_2(0,1) + h_2(1,0)$, obtém-se

$$h_2(0,1)x(n)x(n-1) + h_2(1,0)x(n)x(n-1) = \underline{h}_2(0,1)x(n)x(n-1). \tag{2.20}$$

Assim, de maneira geral, a relação de entrada e saída do filtro Volterra (2.1) pode ser agora reescrita como

$$\begin{aligned} y(n) = & \sum_{m_1=0}^{N-1} \underline{h}_1(m_1)x(n-m_1) \\ & + \sum_{m_1=0}^{N-1}\sum_{m_2=m_1}^{N-1} \underline{h}_2(m_1,m_2)x(n-m_1)x(n-m_2) + \dots \\ & + \sum_{m_1=0}^{N-1}\cdots\sum_{m_P=m_{P-1}}^{N-1} \underline{h}_P(m_1,\cdots,m_P)x(n-m_1)\cdots x(n-m_P) \end{aligned} \tag{2.21}$$

com

$$\underline{h}_p(m_1, m_2, ..., m_p) = \sum_{\varphi(\cdot)} h_p(m_{\varphi(1)}, m_{\varphi(2)}, ..., m_{\varphi(p)}) \tag{2.22}$$

onde o somatório em $\varphi(\cdot)$ de (2.22) representa a soma de todos os coeficientes da representação convencional da relação de entrada e saída do filtro Volterra cujos índices são dados pelas diferentes permutações $\varphi(m_1, m_2, ..., m_p)$ de $m_1, m_2, ..., m_p$. Assim, a relação de entrada e saída de um bloco de ordem p é

$$y_p(n) = \sum_{m_1=0}^{N-1} \cdots \sum_{m_p=m_{p-1}}^{N-1} \underline{h}_p(m_1, \cdots, m_p) x(n-m_1) \cdots x(n-m_p). \tag{2.23}$$

Considerando, por exemplo, um bloco de segunda ordem e organizando os coeficientes de (2.23) de acordo com a representação espacial descrita na Seção 2.2.3, a seguinte matriz de coeficientes é obtida:

$$\underline{\mathbf{H}}_2 = \begin{bmatrix} \underline{h}_2(0,0) & \underline{h}_2(0,1) & \cdots & \underline{h}_2(0,N-1) \\ 0 & \underline{h}_2(1,1) & \cdots & \underline{h}_2(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \underline{h}_2(N-1,N-1) \end{bmatrix}. \tag{2.24}$$

Em (2.24), observa-se que a representação espacial dos coeficientes de segunda ordem de (2.23) resulta em uma matriz triangular superior, enquanto os coeficientes da representação convencional quando organizados na forma espacial resultam em (2.16). Como conseqüência, a relação de entrada e saída do filtro Volterra dada por (2.21), (2.22) e (2.23) é denominada, de maneira geral, representação triangular [6].

Também no caso da representação triangular, é possível escrever a relação de entrada e saída dos diferentes blocos como produtos internos de dois vetores. Para o bloco de primeira ordem, tal relação é igual à da representação convencional, uma vez que o vetor de entrada $\underline{\mathbf{x}}_1(n)$ e o vetor de coeficientes $\underline{\mathbf{h}}_1$ da representação triangular são iguais, respectivamente, aos vetores de entrada $\mathbf{x}_1(n)$ e de coeficientes $\mathbf{h}_1$ da representação convencional. Isso se deve ao fato de os coeficientes do bloco de primeira ordem possuírem apenas o índice m_1, não havendo assim permutações de índices. Para os demais blocos, é necessário primeiro definir os vetores de entrada e de coeficientes considerando (2.21). Exemplificando, para o bloco de segunda ordem temos

$$\underline{\mathbf{h}}_2 = [\underline{h}_2(0,0) \;\; \underline{h}_2(0,1) \;\; \cdots \;\; \underline{h}_2(0,N-1) \;\; \underline{h}_2(1,1) \;\; \cdots \;\; \underline{h}_2(N-1,N-1)]^{\mathrm{T}} \tag{2.25}$$

e

$$\underline{\mathbf{x}}_2(n) = [x^2(n) \;\; x(n)x(n-1) \;\; \cdots \;\; x(n)x(n-N+1) \;\; x^2(n-1) \;\; \cdots \;\; x^2(n-N+1)]^{\mathrm{T}} \tag{2.26}$$

resultando na seguinte relação de entrada e saída:

$$y_2(n) = \mathbf{x}_2^{\mathrm{T}}(n)\mathbf{h}_2 = \underline{\mathbf{x}}_2^{\mathrm{T}}(n)\underline{\mathbf{h}}_2. \tag{2.27}$$

É importante ressaltar que, apesar da diferença existente entre $\mathbf{h}_2$ e $\underline{\mathbf{h}}_2$ e também entre $\mathbf{x}_2(n)$ e $\underline{\mathbf{x}}_2(n)$, observa-se em (2.27) a igualdade $\mathbf{x}_2^{\mathrm{T}}(n)\mathbf{h}_2 = \underline{\mathbf{x}}_2^{\mathrm{T}}(n)\underline{\mathbf{h}}_2$, ilustrando assim a equivalência das representações convencional e triangular. De maneira geral, para um bloco de ordem p é possível definir vetores $\underline{\mathbf{h}}_p$ e $\underline{\mathbf{x}}_p(n)$ análogos a (2.25) e (2.26), de tal forma que

$$y_p(n) = \mathbf{h}_p^{\mathrm{T}}\mathbf{x}_p(n) = \underline{\mathbf{h}}_p^{\mathrm{T}}\underline{\mathbf{x}}_p(n). \tag{2.28}$$

Assim, é possível definir o vetor de entrada

$$\underline{\mathbf{x}}_{\mathrm{V}}(n) = [\underline{\mathbf{x}}_1^{\mathrm{T}}(n)\ \underline{\mathbf{x}}_2^{\mathrm{T}}(n)\ \ldots\ \underline{\mathbf{x}}_P^{\mathrm{T}}(n)]^{\mathrm{T}} \tag{2.29}$$

e o vetor de coeficientes

$$\underline{\mathbf{h}}_{\mathrm{V}} = [\underline{\mathbf{h}}_1^{\mathrm{T}}\ \underline{\mathbf{h}}_2^{\mathrm{T}}\ \ldots\ \underline{\mathbf{h}}_P^{\mathrm{T}}]^{\mathrm{T}} \tag{2.30}$$

que resultam na seguinte relação de entrada e saída do filtro Volterra:

$$y(n) = \mathbf{h}_{\mathrm{V}}^{\mathrm{T}}\mathbf{x}_{\mathrm{V}}(n) = \underline{\mathbf{h}}_{\mathrm{V}}^{\mathrm{T}}\underline{\mathbf{x}}_{\mathrm{V}}(n). \tag{2.31}$$

A principal vantagem da representação triangular é o seu reduzido número de coeficientes em comparação com a representação convencional. Isso pode ser facilmente constatado comparando as matrizes (2.16) da representação convencional e (2.24) da representação triangular como também comparando as representações espaciais dos blocos de terceira ordem convencional (Figura 2.3) e triangular (Figura 2.4).

De maneira geral, considerando que no caso da representação triangular existe apenas um coeficiente para cada combinação de ordem p de amostras do sinal de entrada, o número de coeficientes para o bloco de ordem p em função do tamanho de memória N é dado por [7]

$$\underline{D}_p(N) = \frac{(N+p-1)!}{(N-1)!\,p!} \tag{2.32}$$

e o número total de coeficientes do filtro Volterra de ordem P e tamanho de memória N é

$$\underline{D}_{\mathrm{V}}(N,P) = \frac{(N+P)!}{N!P!} - 1. \tag{2.33}$$

Na Figura 2.5, são apresentadas as curvas do número de coeficientes requerido para implementação de blocos de segunda e terceira ordens usando as representações convencional e triangular em função do tamanho de memória. A partir dessas curvas fica

evidente a vantagem, em termos de número de coeficientes e, conseqüentemente, complexidade computacional, do uso da representação triangular para a implementação do filtro Volterra.

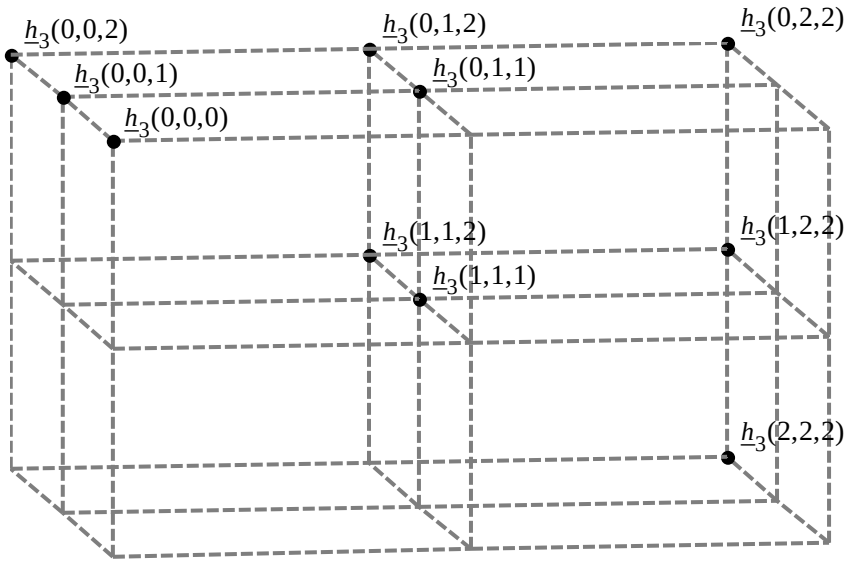
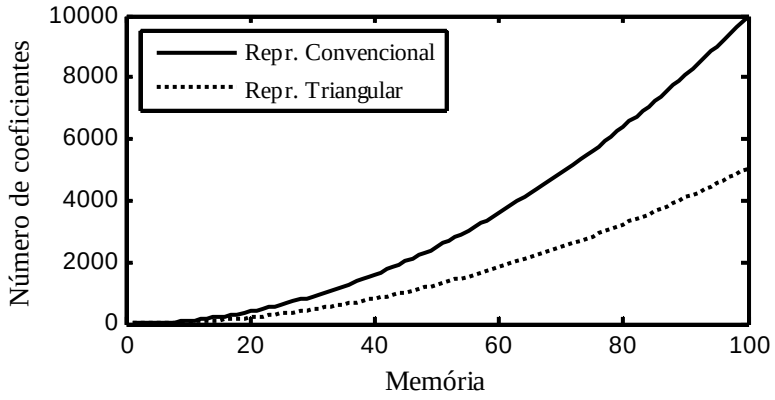
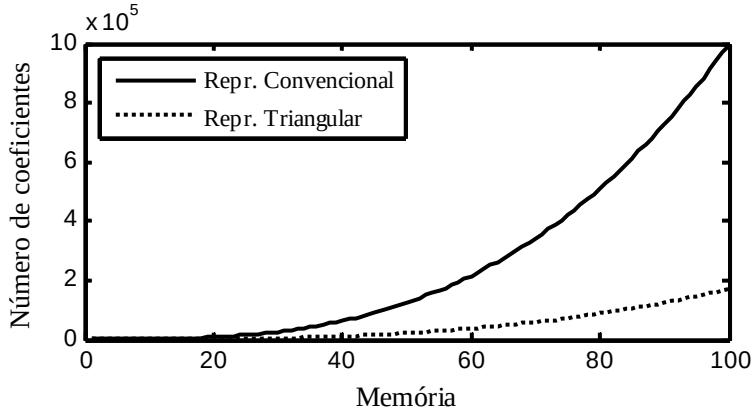


Figura 2.4. Representação espacial dos coeficientes da representação triangular de um bloco de terceira ordem do filtro Volterra.

Por outro lado, o uso da representação triangular para fins de análise de implementações do filtro Volterra geralmente resulta em um tratamento matemático bastante complicado. Isso se deve principalmente à ausência, no caso da representação triangular, de uma expressão análoga a (2.7) que estabeleça uma relação simples entre os vetores de entrada dos blocos de diferentes ordens do filtro Volterra. Assim, muitas vezes, torna-se necessário o uso da representação convencional para fins de análise enquanto para fins de implementação a representação triangular é a melhor opção devido ao seu menor número de coeficientes.



(a)



(b)

Figura 2.5. Número de coeficientes requerido para implementação de diferentes blocos do filtro Volterra usando as representações convencional e triangular em função do tamanho de memória. (a) Bloco de segunda ordem. (b) Bloco de terceira ordem.

2.2.6 Representação em Coordenadas Diagonais

A representação em coordenadas diagonais da relação de entrada e saída do filtro Volterra é obtida introduzindo a seguinte mudança de índices em (2.23) [24], [6]:

$$\begin{aligned} m_1 &= s \\ m_i &= s + r_{i-1}, \quad i = 2, \ldots, p. \end{aligned} \tag{2.34}$$

Adotando tal modificação, (2.23) pode ser reescrita como

$$y_p(n) = \sum_{s=0}^{N-1} \sum_{r_1=0}^{N-1-s} \sum_{r_2=r_1}^{N-1-s} \cdots \sum_{r_{p-1}=r_{p-2}}^{N-1-s} \underline{h}_p(s, s+r_1, \cdots, s+r_{p-1}) x(n-s) \prod_{i=1}^{p-1} x(n-s-r_i). \tag{2.35}$$

É importante observar que os coeficientes de (2.35) são os mesmos de (2.23) uma vez que apenas uma mudança de coordenadas é realizada. A diferença entre os somatórios de (2.23) e (2.35) está apenas na seqüência na qual os termos são somados. Tal diferença é ilustrada na Figura 2.6 para um filtro Volterra de segunda ordem com $N = 5$. Nessa figura, a matriz de coeficientes de segunda ordem é apresentada em conjunto com setas indicando

a seqüência na qual cada um dos coeficientes aparece nos somatórios de (2.23) e (2.35). Observe que, enquanto no caso da representação triangular a soma se dá ao longo das linhas da matriz, no caso da representação diagonal a soma ocorre ao longo das diagonais da matriz. Para blocos de ordens superiores o resultado é análogo ao do bloco de segunda ordem. A vantagem da representação em coordenadas diagonais é a possibilidade de implementar cada somatório ao longo de uma diagonal como um filtro FIR linear com uma não-linearidade sem memória na entrada [6]. Exemplificando, para o caso considerado na Figura 2.6 (filtro Volterra de segunda ordem com $N = 5$), o somatório ao longo da diagonal principal é dado por

$$\begin{aligned} y_{\mathrm{d}_0}(n) = {} & h_2(0,0)x^2(n) + h_2(1,1)x^2(n-1) \\ & + h_2(2,2)x^2(n-2) + h_2(3,3)x^2(n-3) + h_2(4,4)x^2(n-4). \end{aligned} \tag{2.36}$$

podendo o cálculo desse somatório ser implementado elevando o sinal de entrada ao quadrado e filtrando tal sinal usando um filtro FIR com coeficientes dados por $\{h_2(0,0), h_2(1,1), \ldots, h_2(4,4)\}$. Assim, a implementação dos somatórios ao longo das diferentes diagonais resulta em um banco de filtros FIR com não-linearidades sem memória em suas entradas. Conforme discutido em [24] e [25], essa forma de implementação é interessante do ponto de vista prático e pode ser explorada para elaborar implementações com complexidade reduzida em algumas aplicações.

$$\underline{\mathbf{H}}_2 = \begin{bmatrix} \underline{h}_2(0,0) & \underline{h}_2(0,1) & \underline{h}_2(0,2) & \underline{h}_2(0,3) & \underline{h}_2(0,4) \\ 0 & \underline{h}_2(1,1) & \underline{h}_2(1,2) & \underline{h}_2(1,3) & \underline{h}_2(1,4) \\ 0 & 0 & \underline{h}_2(2,2) & \underline{h}_2(2,3) & \underline{h}_2(2,4) \\ 0 & 0 & 0 & \underline{h}_2(3,3) & \underline{h}_2(3,4) \\ 0 & 0 & 0 & 0 & \underline{h}_2(4,4) \end{bmatrix}$$

(a)

$$\underline{\mathbf{H}}_2 = \begin{bmatrix} \underline{h}_2(0,0) & \underline{h}_2(0,1) & \underline{h}_2(0,2) & \underline{h}_2(0,3) & \underline{h}_2(0,4) \\ 0 & \underline{h}_2(1,1) & \underline{h}_2(1,2) & \underline{h}_2(1,3) & \underline{h}_2(1,4) \\ 0 & 0 & \underline{h}_2(2,2) & \underline{h}_2(2,3) & \underline{h}_2(2,4) \\ 0 & 0 & 0 & \underline{h}_2(3,3) & \underline{h}_2(3,4) \\ 0 & 0 & 0 & 0 & \underline{h}_2(4,4) \end{bmatrix}$$

(b)

Figura 2.6. Ilustração da seqüência na qual cada coeficiente aparece no somatório da relação de entrada e saída de segunda ordem em diferentes representações do filtro Volterra. (a) Representação triangular (2.23). (b) Representação em coordenadas diagonais (2.35).

2.3 Análise Estatística do Filtro Volterra Adaptado pelo Algoritmo LMS

Conforme mencionado na seção anterior, os algoritmos adaptativos tradicionalmente aplicados aos filtros FIR lineares também podem ser utilizados para adaptação do filtro Volterra [6]. No entanto, o elevado número de coeficientes apresentado por tal filtro implica um alto custo computacional que dificulta sobremaneira a sua aplicação no contexto adaptativo. Como conseqüência, implementações adaptativas do filtro Volterra são geralmente realizadas usando algoritmos adaptativos de menor complexidade como, por exemplo, o algoritmo LMS. Outro problema encontrado para a aplicação do filtro Volterra no contexto adaptativo é a falta de modelos matemáticos que descrevam o seu comportamento ao longo do processo de adaptação e que, com isso, possam ser usados como ferramentas efetivas de projeto. No caso dos filtros Volterra adaptados pelo algoritmo LMS (Volterra-LMS), são encontrados na literatura alguns trabalhos dedicados a tal modelagem. Em um dos primeiros trabalhos tratando esse problema, os autores apresentam resultados analíticos restritos a filtros Volterra-LMS de segunda ordem com sinais de entrada gaussianos [26]. Outros trabalhos são dedicados ao estudo da convergência e estabilidade [27]-[29], ou focados em filtros com procedimentos de ortogonalização de entrada [30]. Em geral, tais trabalhos são relacionados a filtros Volterra com restrições de ordem como também casos em que o modelo probabilístico do sinal de entrada é pré-determinado. Contribuindo nesse contexto, expressões que descrevem a curva de aprendizagem e o primeiro e segundo momentos do filtro Volterra-LMS são apresentadas nesta seção. O desenvolvimento dessas expressões é realizado sem a imposição de restrições quanto à ordem do filtro nem quanto ao modelo probabilístico do sinal de entrada, resultando em um modelo simples e generalizado do comportamento do filtro Volterra-LMS. Os resultados apresentados nesta seção constituem-se na primeira contribuição original desta tese [10], correspondendo a uma evolução dos resultados apresentados em [9] e [31].

2.3.1 Vetor de Coeficientes Ótimo

O algoritmo LMS é uma implementação estocástica do método *steepest descent* [1]. A idéia por trás de tal algoritmo é minimizar o erro quadrático médio $\xi(n) = E[e^2(n)]$ a partir da sua estimativa instantânea $\hat{\xi}(n) = e^2(n)$ [1]. Assim, o primeiro passo para a

análise do algoritmo LMS aplicado ao filtro Volterra é obter o vetor de coeficientes ótimo no sentido do mínimo erro quadrático médio. A Figura 2.7 apresenta o diagrama de blocos de um filtro Volterra, representado por $\mathbf{h}_{\mathrm{V}}$, no contexto da estimação de um sinal $d(n)$ correlacionado com o sinal de entrada $x(n)$ e, conseqüentemente, com o vetor de entrada $\mathbf{x}_{\mathrm{V}}(n)$ do filtro Volterra. O sinal $e(n)$ representa o erro de estimação.

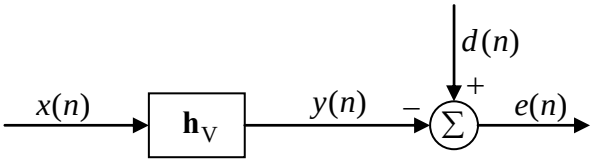


Figura 2.7. Diagrama de blocos de um filtro Volterra aplicado a um problema de estimação de um sinal.

A partir da Figura 2.7 e considerando a representação convencional da relação de entrada e saída do filtro Volterra dada por (2.15), o erro instantâneo pode ser escrito como

$$e(n) = d(n) - y(n) = d(n) - \mathbf{h}_{\mathrm{V}}^{\mathrm{T}}\mathbf{x}_{\mathrm{V}}(n). \tag{2.37}$$

Elevando (2.37) ao quadrado e tomando o valor esperado de ambos os lados da expressão resultante, obtém-se

$$\begin{aligned}\xi = E[e^2(n)] &= E[d^2(n)] - 2E[d(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)]\mathbf{h}_{\mathrm{V}} + \mathbf{h}_{\mathrm{V}}^{\mathrm{T}}E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)]\mathbf{h}_{\mathrm{V}} \\ &= E[d^2(n)] - 2\mathbf{p}_{\mathrm{V}}^{\mathrm{T}}\mathbf{h}_{\mathrm{V}} + \mathbf{h}_{\mathrm{V}}^{\mathrm{T}}\mathbf{R}_{\mathrm{VV}}\mathbf{h}_{\mathrm{V}}\end{aligned} \tag{2.38}$$

onde os valores esperados $\mathbf{R}_{\mathrm{VV}} = E[\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)]$ e $\mathbf{p}_{\mathrm{V}} = E[d(n)\mathbf{x}_{\mathrm{V}}(n)]$ são, respectivamente, a matriz de autocorrelação de entrada e o vetor de correlação cruzada entre o sinal a ser estimado $d(n)$ e o vetor de entrada. Considerando (2.38) e fazendo $\nabla_{\mathbf{h}_{\mathrm{V}}}\xi = 0,$ obtém-se

$$\mathbf{p}_{\mathrm{V}} = \mathbf{R}_{\mathrm{VV}}\mathbf{h}_{\mathrm{Vo}}. \tag{2.39}$$

onde $\mathbf{h}_{\mathrm{Vo}}$ é o vetor de coeficientes ótimo. A solução do sistema de equações representado por (2.39) de forma a se obter o valor de $\mathbf{h}_{\mathrm{Vo}}$ não é trivial uma vez que a matriz $\mathbf{R}_{\mathrm{VV}}$ não possui inversa. Isso se deve ao fato de o vetor de entrada da representação convencional do filtro Volterra apresentar elementos com valores repetidos, como, por exemplo, $x(n)x(n-1)$ e $x(n-1)x(n)$ na parte de segunda ordem. Com isso, a matriz de autocorrelação $\mathbf{R}_{\mathrm{VV}}$ apresenta linhas e colunas repetidas sendo, portanto, não-invertível (singular) devido ao seu posto deficiente. Por outro lado, o vetor de entrada da representação triangular do filtro Volterra não possui os elementos repetidos apresentados

pelo vetor da representação convencional. Assim, a partir da representação triangular é possível obter o vetor de coeficientes ótimo. Fazendo um desenvolvimento similar a (2.37)-(2.39) e considerando também (2.31), obtém-se

$$\underline{\mathbf{p}}_{\mathrm{V}} = \underline{\mathbf{R}}_{\mathrm{VV}}\underline{\mathbf{h}}_{\mathrm{Vo}} \tag{2.40}$$

com $\underline{\mathbf{R}}_{\mathrm{VV}} = E[\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)]$ e $\underline{\mathbf{p}}_{\mathrm{V}} = E[d(n)\underline{\mathbf{x}}_{\mathrm{V}}(n)]$. Como agora a matriz $\underline{\mathbf{R}}_{\mathrm{VV}}$ é invertível (não-singular), obtém-se, a partir de (2.40), a seguinte expressão para o vetor de coeficientes ótimo:

$$\underline{\mathbf{h}}_{\mathrm{Vo}} = \underline{\mathbf{R}}_{\mathrm{VV}}^{-1}\underline{\mathbf{p}}_{\mathrm{V}}. \tag{2.41}$$

Adicionalmente, substituindo (2.41) na expressão do erro quadrático médio (2.38) e usando a representação triangular, a seguinte expressão para o valor mínimo do erro quadrático médio é obtida:

$$\xi_{\min} = E[d^2(n)] - \underline{\mathbf{p}}_{\mathrm{V}}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vo}}. \tag{2.42}$$

O fato de o vetor de coeficientes ótimos de (2.41) ser obtido a partir da representação triangular não implica perda de generalidade. Tal vetor pode, por exemplo, ser convertido para a representação convencional utilizando como base a representação simétrica descrita em [6]. Além disso, como a implementação de filtros Volterra adaptativos é geralmente realizada usando a representação triangular devido ao seu menor custo computacional, o vetor ótimo que se deseja obter para fins de análise é normalmente o de tal representação. Outro fato importante a ser destacado com vistas a (2.41) é a sua similaridade com a expressão de cálculo do vetor ótimo do filtro FIR linear transversal [1]. Isso se deve à similaridade existente entre (2.31) e a relação de entrada e saída do filtro transversal [1]. No entanto, as estruturas de $\underline{\mathbf{R}}_{\mathrm{VV}}$ e $\underline{\mathbf{p}}_{\mathrm{V}}$ são bastante diferentes das matrizes correspondentes para o caso linear, implicando diferenças importantes entre os algoritmos.

2.3.2 Comportamento Médio dos Coeficientes

A equação de adaptação dos coeficientes do filtro Volterra-LMS é dada por [6]

$$\underline{\mathbf{h}}_{\mathrm{V}}(n+1) = \underline{\mathbf{h}}_{\mathrm{V}}(n) + 2\mu e(n)\underline{\mathbf{x}}_{\mathrm{V}}(n). \tag{2.43}$$

Se diferentes valores de passo de adaptação são considerados para cada bloco ou para cada coeficiente, a expressão de adaptação é escrita como

$$\underline{\mathbf{h}}_{\mathrm{V}}(n+1) = \underline{\mathbf{h}}_{\mathrm{V}}(n) + 2\mathbf{M}e(n)\underline{\mathbf{x}}_{\mathrm{V}}(n) \tag{2.44}$$

onde $\mathbf{M}$ é uma matriz diagonal contendo os diferentes valores do passo de adaptação para cada coeficiente. Por simplicidade, o desenvolvimento apresentado aqui considera o caso de adaptação usando um único valor de passo dado por (2.43). A extensão para o caso com múltiplos valores é trivial.

O objetivo agora é determinar uma expressão que descreva (modele) o comportamento do primeiro momento do vetor de coeficientes, ou seja, o comportamento médio dos coeficientes de (2.43). Para tal, um ruído de medição $z(n)$ (gaussiano, com média igual a zero e descorrelacionado dos demais sinais do sistema) é adicionado a $d(n)$. Com isso, o erro instantâneo é dado por

$$e(n) = d(n) + z(n) - y(n) = d(n) + z(n) - \underline{\mathbf{h}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{x}}_{\mathrm{V}}(n). \tag{2.45}$$

Agora, substituindo (2.45) em (2.43) e tomando o valor esperado de ambos os lados da equação resultante, obtém-se

$$\begin{aligned} E[\underline{\mathbf{h}}_{\mathrm{V}}(n+1)] = E[\underline{\mathbf{h}}_{\mathrm{V}}(n)] + 2\mu E[d(n)\underline{\mathbf{x}}_{\mathrm{V}}(n)] \\ +2\mu E[z(n)\underline{\mathbf{x}}_{\mathrm{V}}(n)] - 2\mu E[\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{V}}(n)]. \end{aligned} \tag{2.46}$$

Para determinar os valores esperados de (2.46), uma condição de adaptação lenta é assumida [1], [2]. Apesar de tal condição ser comumente considerada para o caso dos filtros FIR lineares, é razoável considerá-la também válida para o caso de filtros Volterra uma vez que esses últimos apresentam uma convergência geralmente mais lenta em comparação com os primeiros [6]. Assim, sob a condição de adaptação lenta, a correlação entre os vetores de coeficientes e de entrada pode ser desprezada e o último termo de (2.46) é aproximado por

$$E[\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{V}}(n)] \approx E[\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)]E[\underline{\mathbf{h}}_{\mathrm{V}}(n)]. \tag{2.47}$$

Como o ruído de medição possui média nula e é descorrelacionado de $\underline{\mathbf{x}}_{\mathrm{V}}(n)$, tem-se

$$E[z(n)\underline{\mathbf{x}}_{\mathrm{V}}(n)] = E[z(n)]E[\underline{\mathbf{x}}_{\mathrm{V}}(n)] = 0. \tag{2.48}$$

A partir de (2.47) e (2.48), a expressão recursiva para o comportamento médio dos coeficientes é dada por

$$E[\underline{\mathbf{h}}_{\mathrm{V}}(n+1)] = (\mathbf{I} - 2\mu\underline{\mathbf{R}}_{\mathrm{VV}})E[\underline{\mathbf{h}}_{\mathrm{V}}(n)] + 2\mu\underline{\mathbf{p}}_{\mathrm{V}} \tag{2.49}$$

onde $\mathbf{I}$ é a matriz identidade. Também é possível utilizar o vetor de erro dos coeficientes, o qual é dado por

$$\underline{\mathbf{v}}(n) = \underline{\mathbf{h}}_{\mathrm{V}}(n) - \underline{\mathbf{h}}_{\mathrm{Vo}} \tag{2.50}$$

para descrever o comportamento dos coeficientes do filtro. Assim, considerando que a partir de (2.50) tem-se $\underline{\mathbf{h}}_{\mathrm{V}}(n) = \underline{\mathbf{v}}(n) + \underline{\mathbf{h}}_{\mathrm{Vo}}$ e substituindo essa expressão em (2.49), obtém-se

$$E[\underline{\mathbf{v}}(n+1)] = (\mathbf{I} - 2\mu\underline{\mathbf{R}}_{\mathrm{VV}})E[\underline{\mathbf{v}}(n)]. \tag{2.51}$$

2.3.3 Matriz de Autocorrelação de Entrada

A partir dos resultados apresentados na seção anterior, observa-se que a evolução dos coeficientes ao longo do processo adaptativo do filtro Volterra-LMS é influenciada pela matriz $\underline{\mathbf{R}}_{\mathrm{VV}}$. Assim, o conhecimento da estrutura de tal matriz tem um papel importante na modelagem do algoritmo. Relembrando a definição de $\underline{\mathbf{R}}_{\mathrm{VV}}$, dada por

$$\underline{\mathbf{R}}_{\mathrm{VV}} = E[\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)] \tag{2.52}$$

a estrutura de tal matriz, considerando um filtro Volterra de ordem P cujo vetor de entrada é dado por (2.30), pode ser escrita alternativamente como

$$\underline{\mathbf{R}}_{\mathrm{VV}} = \begin{bmatrix} \underline{\mathbf{R}}_{11} & \underline{\mathbf{R}}_{12} & \cdots & \underline{\mathbf{R}}_{1P} \\ \underline{\mathbf{R}}_{21} & \underline{\mathbf{R}}_{22} & \cdots & \underline{\mathbf{R}}_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ \underline{\mathbf{R}}_{P1} & \underline{\mathbf{R}}_{P2} & \cdots & \underline{\mathbf{R}}_{PP} \end{bmatrix} \tag{2.53}$$

onde cada submatriz de $\underline{\mathbf{R}}_{\mathrm{VV}}$ é dada por

$$\underline{\mathbf{R}}_{p_1 p_2} = E[\underline{\mathbf{x}}_{p_1}(n)\underline{\mathbf{x}}_{p_2}^{\mathrm{T}}(n)] \tag{2.54}$$

e $\underline{\mathbf{x}}_{p_1}(n)$ e $\underline{\mathbf{x}}_{p_2}(n)$ são os vetores de entrada de ordem p_1 e p_2, respectivamente, da representação triangular do filtro Volterra. A partir de (2.53) observa-se que a matriz de autocorrelação de entrada possui uma estrutura mais complexa no caso do filtro Volterra do que no caso do filtro linear [1]. Algumas das implicações de tal estrutura no comportamento do algoritmo Volterra-LMS serão discutidas na seqüência considerando inicialmente o caso de um sinal branco gaussiano de entrada.

No caso do filtro linear (o qual corresponde a um filtro Volterra de ordem $P = 1$), $\underline{\mathbf{R}}_{\mathrm{VV}}$ é composta apenas pela submatriz $\underline{\mathbf{R}}_{11}$ cuja dimensão é $N \times N$. Assim,

$$\underline{\mathbf{R}}_{\mathrm{VV}} = \underline{\mathbf{R}}_{11} = \begin{bmatrix} \sigma_x^2 & 0 & \cdots & 0 \\ 0 & \sigma_x^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_x^2 \end{bmatrix} \tag{2.55}$$

para um sinal de entrada branco gaussiano com variância σ_x^2. Note que (2.55) é uma matriz diagonal, o que permite reescrever (2.51) como um conjunto de equações independentes ou desacopladas. Assim, a expressão que modela o comportamento do k-ésimo coeficiente do bloco de primeira ordem pode ser escrita como

$$E[\underline{v}_1(k,n+1)] = E[\underline{v}_1(k,n)] - 2\mu\sigma_x^2 E[\underline{v}_1(k,n)]. \tag{2.56}$$

Agora, considerando um filtro Volterra de segunda ordem $(P=2)$, $\underline{\mathbf{R}}_{\mathrm{VV}}$ é composta pelas submatrizes $\underline{\mathbf{R}}_{11}$, $\underline{\mathbf{R}}_{12}$, $\underline{\mathbf{R}}_{21}$, e $\underline{\mathbf{R}}_{22}$. Novamente, para o caso de um sinal de entrada branco gaussiano com variância σ_x^2, $\underline{\mathbf{R}}_{11}$ é dada pela matriz (2.55),

$$\underline{\mathbf{R}}_{12} = \underline{\mathbf{R}}_{21}^{\mathrm{T}} = \mathbf{0} \tag{2.57}$$

e

$$\underline{\mathbf{R}}_{22} = \begin{bmatrix} 3\sigma_x^4 & 0 & \cdots & 0 & \sigma_x^4 & 0 & \cdots & \sigma_x^4 \\ 0 & \sigma_x^4 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_x^4 & 0 & 0 & \cdots & 0 \\ \sigma_x^4 & 0 & \cdots & 0 & 3\sigma_x^4 & 0 & \cdots & \sigma_x^4 \\ 0 & 0 & \cdots & 0 & 0 & \sigma_x^4 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_x^4 & 0 & \cdots & 0 & \sigma_x^4 & 0 & \cdots & 3\sigma_x^4 \end{bmatrix}. \tag{2.58}$$

Assim, nesse caso a estrutura de $\underline{\mathbf{R}}_{\mathrm{VV}}$ é

$$\underline{\mathbf{R}}_{\mathrm{VV}} = \begin{bmatrix} \underline{\mathbf{R}}_{11} & \mathbf{0} \\ \mathbf{0} & \underline{\mathbf{R}}_{22} \end{bmatrix} \tag{2.59}$$

com $\underline{\mathbf{R}}_{11}$ dada por (2.56) e $\underline{\mathbf{R}}_{22}$ por (2.59). A partir de (2.58) e (2.59) é possível observar que, para esse caso, $\underline{\mathbf{R}}_{\mathrm{VV}}$ não é uma matriz contendo elementos apenas na diagonal principal. Como resultado, o comportamento dos coeficientes não pode ser descrito por um conjunto de equações desacopladas. Exemplificando para o caso do um filtro Volterra-LMS com tamanho de memória $N=3$ e um sinal de entrada branco gaussiano, a equação que descreve o comportamento do primeiro coeficiente do bloco de segunda ordem é dada por

$$\begin{aligned} E[\underline{v}_2(0,0,n+1)] = E[\underline{v}_2(0,0,n)] - 6\mu\sigma_x^4 E[\underline{v}_2(0,0,n)] \\ -2\mu\sigma_x^4 E[\underline{v}_2(1,1,n)] - 2\mu\sigma_x^4 E[\underline{v}_2(2,2,n)] \end{aligned} \tag{2.60}$$

de onde é possível notar que o comportamento do coeficiente $E[\underline{v}_2(0,0,n)]$ também depende do comportamento dos outros coeficientes $E[\underline{v}_2(1,1,n)]$ e $E[\underline{v}_2(2,2,n)]$. Adicionalmente, considerando (2.59), (2.51) pode ser reescrita como

$$E\left\{\begin{bmatrix}\underline{\mathbf{v}}_1(n+1)\\ \underline{\mathbf{v}}_2(n+1)\end{bmatrix}\right\} = \begin{Bmatrix}E[\underline{\mathbf{v}}_1(n)]\\ E[\underline{\mathbf{v}}_2(n)]\end{Bmatrix} - 2\mu\begin{Bmatrix}\underline{\mathbf{R}}_{11}E[\underline{\mathbf{v}}_1(n)]\\ \underline{\mathbf{R}}_{22}E[\underline{\mathbf{v}}_2(n)]\end{Bmatrix} \tag{2.61}$$

onde $\underline{\mathbf{v}}_1(n)$ e $\underline{\mathbf{v}}_2(n)$ correspondem aos vetores de erro dos coeficientes de primeira e segunda ordens, respectivamente. A partir de (2.61), observa-se que, nesse caso, o comportamento dos blocos de primeira e segunda ordens é independente ou desacoplado. Tal independência entre blocos é interessante do ponto de vista prático uma vez que se torna possível, por exemplo, a obtenção de diferentes limites de estabilidade para o valor do passo de adaptação, facilitando o uso de passos diferentes para cada um dos blocos de forma a alterar o processo de convergência [32].

Analisando um filtro de terceira ordem com tamanho de memória igual a 3 e novamente com um sinal de entrada branco gaussiano com variância σ_x^2, obtêm-se as mesmas submatrizes apresentadas anteriormente, a saber $\underline{\mathbf{R}}_{11}$, $\underline{\mathbf{R}}_{12}$, $\underline{\mathbf{R}}_{21}$ e $\underline{\mathbf{R}}_{22}$, e, ainda, as seguintes submatrizes:

$$\underline{\mathbf{R}}_{13} = \begin{bmatrix} 3\sigma_x^4 & 0 & 0 & \sigma_x^4 & 0 & \sigma_x^4 & 0 & 0 & 0 & 0 \\ 0 & \sigma_x^4 & 0 & 0 & 0 & 0 & 3\sigma_x^4 & 0 & \sigma_x^4 & 0 \\ 0 & 0 & \sigma_x^4 & 0 & 0 & 0 & 0 & \sigma_x^4 & 0 & 3\sigma_x^4 \end{bmatrix} \tag{2.62}$$

$$\underline{\mathbf{R}}_{23} = \underline{\mathbf{R}}_{32}^{\mathrm{T}} = \mathbf{0} \tag{2.63}$$

e

$$\underline{\mathbf{R}}_{33} = \begin{bmatrix} 15\sigma_x^6 & 0 & 0 & 3\sigma_x^6 & 0 & 3\sigma_x^6 & 0 & 0 & 0 & 0 \\ 0 & 3\sigma_x^6 & 0 & 0 & 0 & 0 & 3\sigma_x^6 & 0 & \sigma_x^6 & 0 \\ 0 & 0 & 3\sigma_x^6 & 0 & 0 & 0 & 0 & \sigma_x^6 & 0 & 3\sigma_x^6 \\ 3\sigma_x^6 & 0 & 0 & 3\sigma_x^6 & 0 & \sigma_x^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_x^6 & 0 & 0 & 0 & 0 & 0 \\ 3\sigma_x^6 & 0 & 0 & \sigma_x^2 & 0 & 3\sigma_x^6 & 0 & 0 & 0 & 0 \\ 0 & 3\sigma_x^6 & 0 & 0 & 0 & 0 & 15\sigma_x^6 & 0 & 3\sigma_x^6 & 0 \\ 0 & 0 & \sigma_x^6 & 0 & 0 & 0 & 0 & 3\sigma_x^6 & 0 & 3\sigma_x^6 \\ 0 & \sigma_x^6 & 0 & 0 & 0 & 0 & 3\sigma_x^6 & 0 & 3\sigma_x^6 & 0 \\ 0 & 0 & 3\sigma_x^6 & 0 & 0 & 0 & 0 & 3\sigma_x^6 & 0 & 15\sigma_x^6 \end{bmatrix}. \tag{2.64}$$

Assim, (2.51) pode ser reescrita como

$$E\left\{\begin{bmatrix}\underline{\mathbf{v}}_1(n+1)\\ \underline{\mathbf{v}}_2(n+1)\\ \underline{\mathbf{v}}_3(n+1)\end{bmatrix}\right\}=\begin{Bmatrix}E[\underline{\mathbf{v}}_1(n)]\\ E[\underline{\mathbf{v}}_2(n)]\\ E[\underline{\mathbf{v}}_3(n)]\end{Bmatrix}-2\mu\begin{Bmatrix}\underline{\mathbf{R}}_{11}E[\underline{\mathbf{v}}_1(n)]+\underline{\mathbf{R}}_{13}E[\underline{\mathbf{v}}_3(n)]\\ \underline{\mathbf{R}}_{22}E[\underline{\mathbf{v}}_2(n)]\\ \underline{\mathbf{R}}_{31}E[\underline{\mathbf{v}}_1(n)]+\underline{\mathbf{R}}_{33}E[\underline{\mathbf{v}}_3(n)]\end{Bmatrix} \quad (2.65)$$

Nota-se, a partir de (2.65), que agora o acoplamento não ocorre apenas entre os coeficientes dentro de um mesmo bloco não-linear, mas também entre os blocos de primeira e terceira ordens. Assim, como o comportamento desses blocos é interdependente, o uso de diferentes passos de adaptação para cada um deles não é uma abordagem recomendável [32].

Para o caso de sinais de entrada gaussianos coloridos de média zero, muitos dos elementos de $\underline{\mathbf{R}}_{\mathrm{VV}}$ que têm valores nulos para um sinal de entrada branco passam a ter valores diferentes de zero. Isso implica um maior grau de acoplamento entre o comportamento dos coeficientes como também em uma convergência mais lenta. Por outro lado, os acoplamentos entre os blocos são os mesmos tanto para sinais de entrada gaussianos brancos quanto coloridos. Isso se deve ao fato de as matrizes cujos elementos são momentos ímpares do sinal de entrada, como, por exemplo, $\underline{\mathbf{R}}_{12}$, $\underline{\mathbf{R}}_{23}$ e $\underline{\mathbf{R}}_{34}$, serem sempre nulas para quaisquer sinais de entrada gaussianos de média zero.

Considerando sinais de entrada com outras distribuições de probabilidade, uma análise similar à descrita anteriormente para sinais gaussianos deve ser realizada levando-se em conta as especificidades das distribuições envolvidas. Em geral, um grau de acoplamento pequeno entre os coeficientes é verificado para sinais brancos e um maior grau de acoplamento é verificado para sinais coloridos. Além disso, a maior complexidade da matriz de autocorrelação de entrada do filtro Volterra-LMS em comparação com a do filtro linear, geralmente leva a um maior grau de acoplamento como também uma convergência mais lenta.

2.3.4 Influência da Matriz de Autocorrelação de Entrada no Valor do Passo de Adaptação

Conforme discutido em [1], o valor máximo do passo de adaptação do algoritmo LMS no caso de um filtro FIR linear é inversamente proporcional à soma dos autovalores da matriz de autocorrelação de entrada. Assim, considerando que o traço de tal matriz é igual à soma dos seus autovalores [1], quanto maior for o traço, menor será o limite superior para o passo de adaptação. É razoável esperar esse mesmo tipo de dependência entre o traço de

$\underline{\mathbf{R}}_{\mathrm{VV}}$ e o passo de adaptação no caso do filtro Volterra-LMS, uma vez que os algoritmos são de mesma natureza. Dessa forma, a partir de (2.53), observa-se que

$$\mathrm{tr}[\underline{\mathbf{R}}_{\mathrm{VV}}] = \mathrm{tr}[\underline{\mathbf{R}}_{11}] + \mathrm{tr}[\underline{\mathbf{R}}_{22}] + \cdots + \mathrm{tr}[\underline{\mathbf{R}}_{PP}] \tag{2.66}$$

com $\mathrm{tr}[\cdot]$ denotando o operador traço da matriz. Como as matrizes de correlação são sempre definidas positivas, é possível afirmar que

$$\mathrm{tr}[\underline{\mathbf{R}}_{11}] < \mathrm{tr}[\underline{\mathbf{R}}_{\mathrm{VV}}]. \tag{2.67}$$

A partir de (2.67), considerando o mesmo sinal de entrada, é possível concluir que o limite superior de estabilidade do passo de adaptação do filtro Volterra-LMS será sempre menor do que o do filtro linear. Tal afirmação também é suportada pela consideração à otimalidade de H^{∞} de filtros adaptados pelo algoritmo LMS [2]. Nesse caso, o limite de estabilidade superior do passo de adaptação é dado por

$$\mu < \min_{1 \le n \le K} \frac{1}{\left\| \mathbf{u}(n) \right\|^2} \tag{2.68}$$

onde $\mathbf{u}(n)$ é o vetor de entrada do algoritmo e K é o número de iterações em uma realização do algoritmo. A partir de (2.29), é possível verificar que

$$\left\| \underline{\mathbf{x}}_1(n) \right\|^2 < \left\| \underline{\mathbf{x}}_{\mathrm{V}}(n) \right\|^2 = \left\| \underline{\mathbf{x}}_1(n) \right\|^2 + \left\| \underline{\mathbf{x}}_2(n) \right\|^2 + \ldots + \left\| \underline{\mathbf{x}}_P(n) \right\|^2 . \tag{2.69}$$

Assim, a partir de (2.68) e (2.69), para um mesmo sinal de entrada, novamente se observa que o limite superior de estabilidade do passo de adaptação do filtro Volterra-LMS será sempre menor do que o do filtro linear. Conseqüentemente, em aplicações práticas, valores absolutos pequenos do passo de adaptação são necessários para assegurar a convergência e um funcionamento adequado do filtro Volterra-LMS.

2.3.5 Erro Quadrático Médio

Para determinar a expressão que modela a curva de aprendizagem do filtro Volterra-LMS, primeiramente (2.50) é utilizada para reescrever o valor instantâneo do erro. Assim,

$$e(n) = d(n) + z(n) - \underline{\mathbf{v}}^{\mathrm{T}}(n)\underline{\mathbf{x}}_{\mathrm{V}}(n) - \underline{\mathbf{h}}_{\mathrm{Vo}}^{\mathrm{T}}\underline{\mathbf{x}}_{\mathrm{V}}(n) = e_{\mathrm{o}}(n) - \underline{\mathbf{v}}^{\mathrm{T}}(n)\underline{\mathbf{x}}_{\mathrm{V}}(n) \tag{2.70}$$

onde

$$e_{\mathrm{o}}(n) = d(n) + z(n) - \underline{\mathbf{h}}_{\mathrm{Vo}}^{\mathrm{T}}\underline{\mathbf{x}}_{\mathrm{V}}(n) \tag{2.71}$$

é o erro de estimação ótimo. Elevando (2.70) ao quadrado, tomando o valor esperado de ambos os lados da expressão resultante e usando o princípio da ortogonalidade [2], obtém-se

$$\xi(n) = \xi_{\min} + \mathrm{tr}\left[\underline{\mathbf{K}}(n)\underline{\mathbf{R}}_{\mathrm{VV}}\right] \tag{2.72}$$

onde $\xi(n) = E[e^2(n)]$, $\xi_{\min} = E[e_{\mathrm{o}}^2(n)]$, e $\underline{\mathbf{K}}(n) = E[\underline{\mathbf{v}}(n)\underline{\mathbf{v}}^{\mathrm{T}}(n)]$ denota a matriz de covariância do vetor de erro dos coeficientes (momento de segunda ordem). A partir de (2.52) e das informações apresentadas na Seção 2.3.3, é possível observar que $\underline{\mathbf{R}}_{\mathrm{VV}}$ é uma matriz hermitiana [1]. Assim, a decomposição $\underline{\mathbf{R}}_{\mathrm{VV}} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}}$ é factível, com $\mathbf{Q}$ representando a matriz de autovetores de $\underline{\mathbf{R}}_{\mathrm{VV}}$ e Λ uma matriz diagonal composta pelos autovalores de $\underline{\mathbf{R}}_{\mathrm{VV}}$. Considerando tal decomposição, obtém-se, a partir de (2.72), a seguinte expressão para a curva de aprendizagem:

$$\xi(n) = \xi_{\min} + \mathrm{tr}\left[\underline{\mathbf{K}}(n)\underline{\mathbf{R}}_{\mathrm{VV}}\right] = \xi_{\min} + \boldsymbol{\lambda}^{\mathrm{T}}\underline{\bar{\mathbf{k}}}(n) \tag{2.73}$$

onde $\boldsymbol{\lambda}$ é um vetor composto pelos autovalores de $\underline{\mathbf{R}}_{\mathrm{VV}}$ e $\underline{\bar{\mathbf{k}}}(n)$, um vetor contendo os elementos da diagonal principal da matriz $\underline{\bar{\mathbf{K}}}(n) = \mathbf{Q}^{\mathrm{T}}\underline{\mathbf{K}}(n)\mathbf{Q}$.

2.3.6 Modelo para o Momento de Segunda Ordem

A partir de (2.73), nota-se que, para obter a curva de aprendizagem, o conhecimento do comportamento da matriz $\underline{\mathbf{K}}(n)$ ou do vetor $\underline{\bar{\mathbf{k}}}(n)$ é requerido. Assim, nesta seção expressões para modelagem de tais elementos são desenvolvidas. Então, substituindo (2.70) em (2.43) e manipulando-se a expressão resultante, obtém-se

$$\underline{\mathbf{v}}(n+1) = \underline{\mathbf{v}}(n) + 2\mu\underline{\mathbf{x}}_{\mathrm{V}}(n)e_{\mathrm{o}}(n) - 2\mu\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{v}}(n). \tag{2.74}$$

Agora, fazendo $\underline{\mathbf{v}}(n+1)\underline{\mathbf{v}}^{\mathrm{T}}(n+1)$ e tomando o valor esperado da expressão resultante, obtém-se

$$\begin{aligned}
E[\underline{\mathbf{v}}(n+1)\underline{\mathbf{v}}^{\mathrm{T}}(n+1)] = {} & E[\underline{\mathbf{v}}(n)\underline{\mathbf{v}}^{\mathrm{T}}(n)] \\
& -2\mu E[\underline{\mathbf{v}}(n)\underline{\mathbf{v}}^{\mathrm{T}}(n)\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)] \\
& -2\mu E[\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{v}}(n)\underline{\mathbf{v}}^{\mathrm{T}}(n)] \\
& +4\mu^2 E[\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{v}}(n)\underline{\mathbf{v}}^{\mathrm{T}}(n)\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)] \\
& +4\mu^2 E[\underline{\mathbf{x}}_{\mathrm{V}}(n)e_{\mathrm{o}}(n)e_{\mathrm{o}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)] \\
& +2\mu E\{[\mathbf{I}_{\mathrm{d}} - 2\mu\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)]\underline{\mathbf{v}}(n)e_{\mathrm{o}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\} \\
& +2\mu E\{\underline{\mathbf{x}}_{\mathrm{V}}(n)e_{\mathrm{o}}(n)\underline{\mathbf{v}}^{\mathrm{T}}(n)[\mathbf{I}_{\mathrm{d}} - 2\mu\underline{\mathbf{x}}_{\mathrm{V}}(n)\underline{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)]\}.
\end{aligned} \tag{2.75}$$

Considerando o princípio da ortogonalidade [1], os dois últimos termos à direita de (2.75) são iguais a zero. O quarto termo do lado direto de (2.75) é desprezado devido às seguintes razões: (a) tal termo depende de μ^2, o qual tem valor muito pequeno uma vez que μ é geralmente pequeno; (b) também depende de $\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)$, o qual se torna pequeno uma vez que $\mathbf{v}(n)$ tende a zero quando a convergência é alcançada; (c) por simplicidade e generalização do modelo, uma vez que o cálculo de momentos de alta ordem de $\mathbf{x}_{\mathrm{V}}(n)$ são evitados. Por outro lado, tal aproximação pode se tornar ineficiente na fase transiente do processo adaptativo caso o produto $\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)\mathbf{v}(n)\mathbf{v}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{V}}(n)\mathbf{x}_{\mathrm{V}}^{\mathrm{T}}(n)$ resulte em uma matriz com valores elevados, o que pode ocorrer para filtros Volterra com um número muito grande de coeficientes. Entretanto, como o uso de filtros Volterra com tamanhos de memória elevados é bastante restrito devido à carga computacional envolvida, a aproximação aqui realizada apresenta geralmente um bom desempenho para casos de aplicações práticas. Assim, determinando os demais termos em (2.75), a seguinte expressão para o segundo momento é obtida:

$$\underline{\mathbf{K}}(n+1) = \underline{\mathbf{K}}(n) - 2\mu\left[\underline{\mathbf{K}}(n)\mathbf{R}_{\mathrm{VV}} + \underline{\mathbf{R}}_{\mathrm{VV}}\underline{\mathbf{K}}(n)\right] + 4\mu^2\xi_{\min}\underline{\mathbf{R}}_{\mathrm{VV}}. \tag{2.76}$$

Agora, pré-multiplicando (2.76) por $\mathbf{Q}^{\mathrm{T}}$, pós-multiplicando por $\mathbf{Q}$, e levando-se em conta que $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{Q}\mathbf{Q}^{\mathrm{T}} = \mathbf{I}$, obtém-se

$$\underline{\bar{\mathbf{K}}}(n+1) = \underline{\bar{\mathbf{K}}}(n) - 2\mu\left[\underline{\bar{\mathbf{K}}}(n)\Lambda + \Lambda\underline{\bar{\mathbf{K}}}(n)\right] + 4\mu^2\xi_{\min}\Lambda. \tag{2.77}$$

Considerando que a matriz Λ é uma matriz diagonal, tem-se a diagonal principal de $\underline{\bar{\mathbf{K}}}(n)\Lambda$ igual à diagonal principal de $\Lambda\underline{\bar{\mathbf{K}}}(n)$, resultando na seguinte expressão recursiva para o vetor $\underline{\bar{\mathbf{k}}}(n)$:

$$\underline{\bar{\mathbf{k}}}(n+1) = (\mathbf{I} - 4\mu\Lambda)\underline{\bar{\mathbf{k}}}(n) + 4\mu^2\xi_{\min}\boldsymbol{\lambda}. \tag{2.78}$$

Combinando (2.73) e (2.78), um modelo para a curva de aprendizagem do filtro Volterra adaptado pelo algoritmo LMS é obtido. É importante notar que a implementação de tal modelo é realizada com um custo computacional bastante reduzido uma vez que os produtos matriciais de (2.78) envolvem apenas matrizes diagonais e vetores.

2.3.7 Desajuste

O desajuste é definido como

$$\mathcal{M} = \xi_{\mathrm{excess}} / \xi_{\min} \tag{2.79}$$

onde ξ_{excess} é o erro quadrático médio em excesso, ou seja, a diferença entre o valor em regime permanente do erro quadrático médio e seu valor ótimo. Assim, subtraindo ξ_{min} de (2.73), considerando a convergência do algoritmo, ou seja, $\lim_{n\to\infty} \underline{\mathbf{K}}(n+1) = \lim_{n\to\infty} \underline{\mathbf{K}}(n) = \underline{\mathbf{K}}_{\infty}$ e usando (2.77), obtém-se

$$\xi_{excess} = \mathrm{tr}[\underline{\mathbf{K}}_{\infty}\underline{\mathbf{R}}_{VV}] = \mu\xi_{min}\ \mathrm{tr}[\underline{\mathbf{R}}_{VV}]. \tag{2.80}$$

Finalmente, substituindo (2.80) em (2.79), a expressão para o desajuste é obtida. Assim,

$$\mathcal{M} = \mu\ \mathrm{tr}[\underline{\mathbf{R}}_{VV}]. \tag{2.81}$$

2.3.8 Resultados de Simulação

Para verificar a precisão do modelo desenvolvido, alguns resultados de simulação serão apresentados nesta seção considerando um problema de identificação de sistema [1] usando diferentes sinais de entrada. Nos exemplos considerados, resultados obtidos através de simulação Monte Carlo (média de 200 realizações) são comparados com os resultados obtidos com o modelo proposto. Diferentes valores para o passo de adaptação são usados, todos relacionados com um valor crítico μ_{crit} obtido experimentalmente considerando o critério dado por (2.68) [2]. Assim, μ_{crit} é aproximadamente igual ao valor mínimo do inverso da norma quadrática do vetor de entrada obtido ao longo de 200 realizações de simulação de Monte Carlo. Em todos os casos, a variância do ruído de medição $z(n)$ é $\sigma_z^2 = 10^{-6}$. As curvas do comportamento médio dos pesos são apresentadas apenas no Exemplo 1 devido à redundância dos resultados, enquanto as curvas de aprendizagem são apresentadas em todos os exemplos considerados.

Exemplo 2.1: Neste exemplo, a planta tem um tamanho de memória $N = 7$ e sua saída é dada por

$$\begin{aligned} d(n) = {} & x(n) + 0,5x(n-1) + 0,1x(n-2) - 0,2x(n-3) - 0,3x(n-4) \\ & -0,1x(n-5) + 0,08x(n-6) + 0,7x^2(n) + 0,3x(n)x(n-1) \\ & -0,1x(n)x(n-3) + 0,05x(n)x(n-6) + 0,5x^2(n-1) \\ & +0,2x(n-1)x(n-2) - 0,2x(n-1)x(n-5) - 0,35x^2(n-2) \\ & -0,05x(n-2)x(n-4) + 0,02x(n-2)x(n-5)\ -0,2x^2(n-3) \\ & +0,05x(n-3)x(n-5) + 0,1x^2(n-4) + 0,05x(n-4)x(n-5) \\ & +0,02x(n-4)x(n-6) - 0,1x^2(n-5) + 0,05x^2(n-6). \end{aligned} \tag{2.82}$$

O sinal de entrada é branco gaussiano com variância $\sigma_x^2 = 1$. Na Figura 2.8 o comportamento médio dos coeficientes é apresentado, enquanto a Figura 2.9 ilustra a curva de aprendizagem, ambos obtidos usando $\mu = \mu_{crit}$ com $\mu_{crit} = 0,001$. Nos resultados apresentados, apenas pequenas diferenças são verificadas durante a fase transiente entre os resultados de simulação Monte Carlo e os obtidos através do modelo proposto. Tais diferenças tornam-se mais evidentes para as curvas de aprendizagem devido às aproximações consideradas. Reduzindo o passo de adaptação para $\mu = \mu_{crit}/2$, melhores resultados são verificados conforme mostrado nas Figuras 2.10 e 2.11. Como em aplicações práticas geralmente são utilizados valores de μ menores do que o limite de estabilidade, os resultados obtidos atestam a aplicabilidade e a efetividade do modelo proposto para o caso estudado.

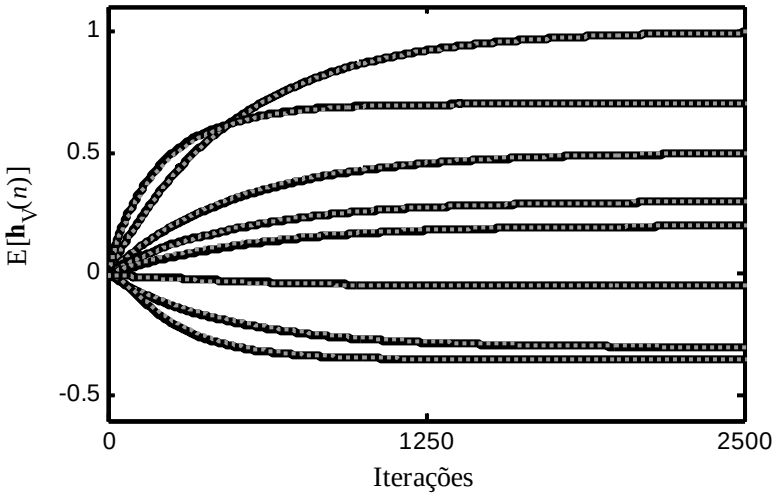


Figura 2.8. Exemplo 2.1. Evolução de $E[\mathbf{h}_V(n)]$ com $\mu = \mu_{crit}$. (Linhas escuras) simulação Monte Carlo. (Linhas cinzas pontilhadas) modelo proposto, obtido a partir de (2.49).

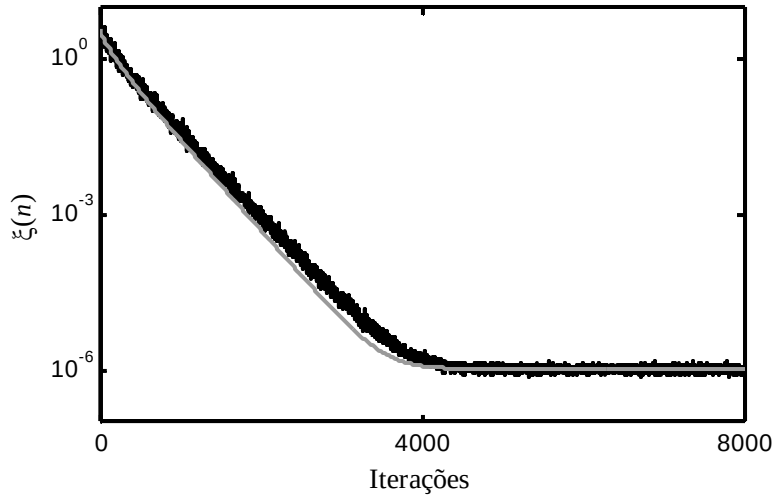


Figura 2.9. Exemplo 2.1. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

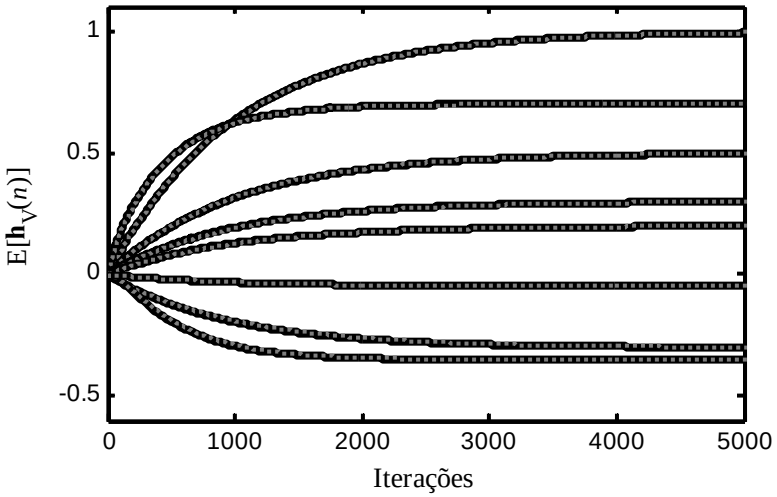


Figura 2.10. Exemplo 2.1. Evolução de $E[\mathbf{h}_V(n)]$ com $\mu = \mu_{crit}/2$. (Linhas escuras) simulação Monte Carlo. (Linhas cinzas pontilhadas) modelo proposto, obtido a partir de (2.49).

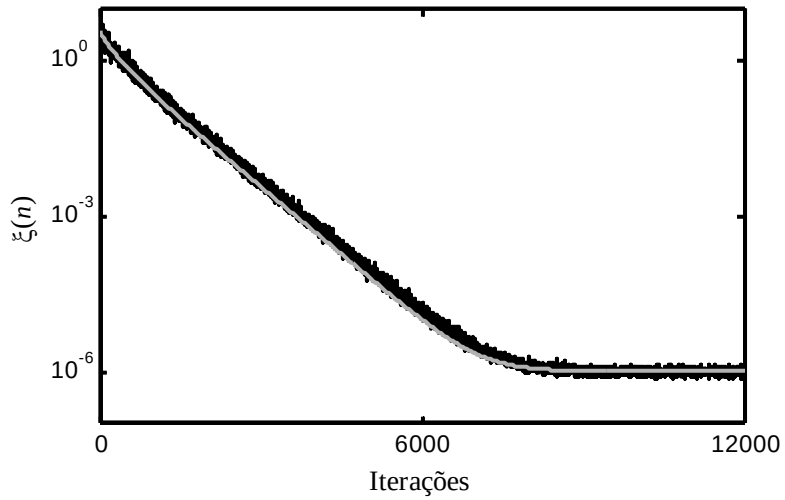


Figura 2.11. Exemplo 2.1. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}/2$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

Exemplo 2.2: Para este exemplo, a mesma planta do Exemplo 2.1 é considerada, porém agora com um sinal de entrada obtido a partir de um processo AR dado por $x(n) = \alpha\, x(n-1) + \sqrt{1-\alpha^2}\, u(n)$, onde $u(n)$ é um sinal branco gaussiano com variância unitária e $\alpha = 0,5$. Na Figura 2.12, curvas do erro quadrático médio obtidas usando $\mu = \mu_{crit}$ com $\mu_{crit} = 0,00033$ são apresentadas. Novamente, verifica-se uma similaridade muito boa entre os resultados de simulação Monte Carlo e os do modelo proposto.

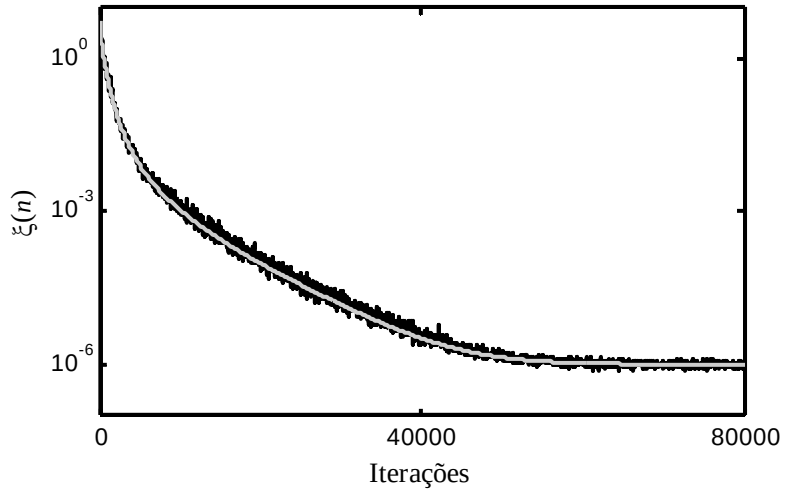


Figura 2.12. Exemplo 2.2. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

Exemplo 2.3: A planta deste exemplo é a mesma dos exemplos anteriores. A diferença agora está no sinal de entrada que é branco e apresenta distribuição uniforme de probabilidade entre -1 e $+1$. Os resultados obtidos com $\mu = \mu_{crit}$ e $\mu_{crit} = 0,0385$ estão apresentados na Figura 2.13. Dessa figura, nota-se que, apesar da boa concordância na fase de regime permanente, uma pequena diferença na fase transiente é observada. Tal diferença é resultado do alto valor absoluto de μ, o que nesse caso é uma conseqüência da variância baixa do sinal de entrada. Considerando um valor de μ menor, dado por $\mu = \mu_{crit}/3$, resultados muito bons são obtidos conforme mostrado na Figura 2.14.

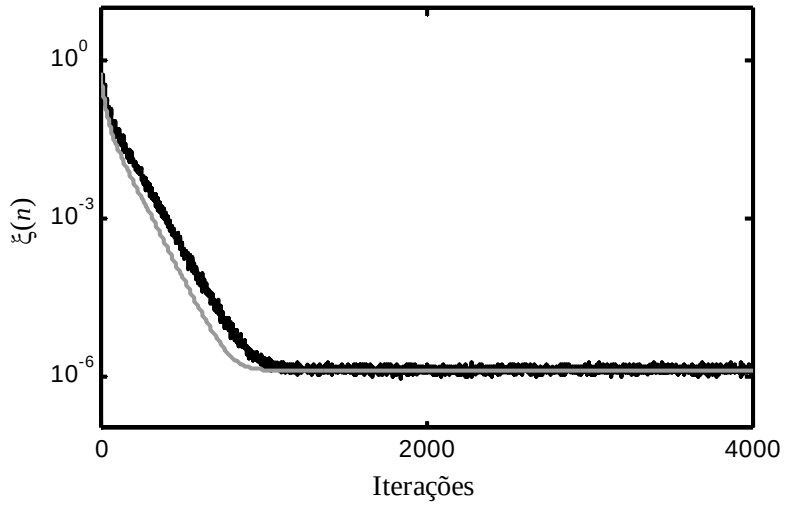


Figura 2.13. Exemplo 2.3. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

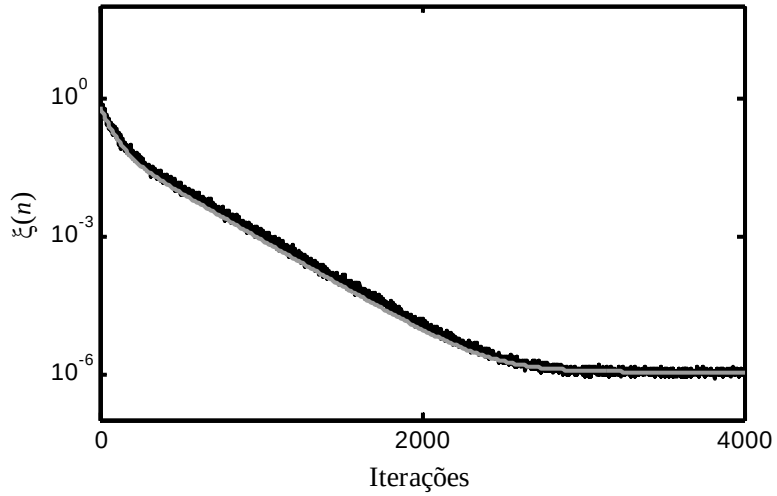


Figura 2.14. Exemplo 2.3. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}/3$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

Exemplo 2.4: Para este exemplo, a planta é a mesma dos exemplos anteriores, mas o sinal de entrada é branco com distribuição de probabilidade de Rayleigh [21] com um fator $\gamma = 0,4$ (média aproximadamente igual a $0,5$ e variância $0,0687$). O passo de adaptação é dado por $\mu = \mu_{crit} = 0,017$ e os resultados obtidos estão apresentados na Figura 2.15. Como pode ser observada nessa figura, uma concordância muito boa entre os resultados de simulação e o do modelo proposto é novamente obtida.

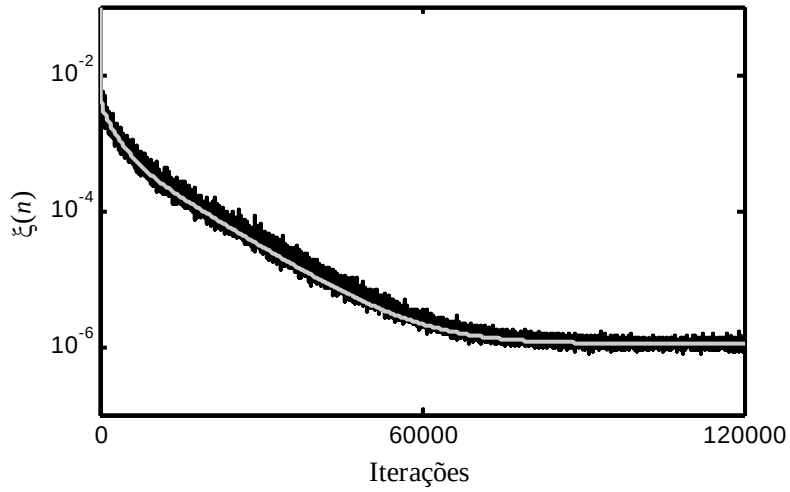


Figura 2.15. Exemplo 2.4. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

Exemplo 2.5: Neste exemplo, é utilizada a mesma planta dos exemplos anteriores e o sinal de entrada é agora composto dos valores 0 e +1 com igual probabilidade (distribuição discreta e uniforme). O valor de μ_{crit} é $0,028$ e o passo de adaptação adotado é $\mu = \mu_{crit}/2$. Os resultados obtidos são apresentados na Figura 2.16, de onde se observa um casamento muito bom entre os resultados de simulação e o do modelo proposto.

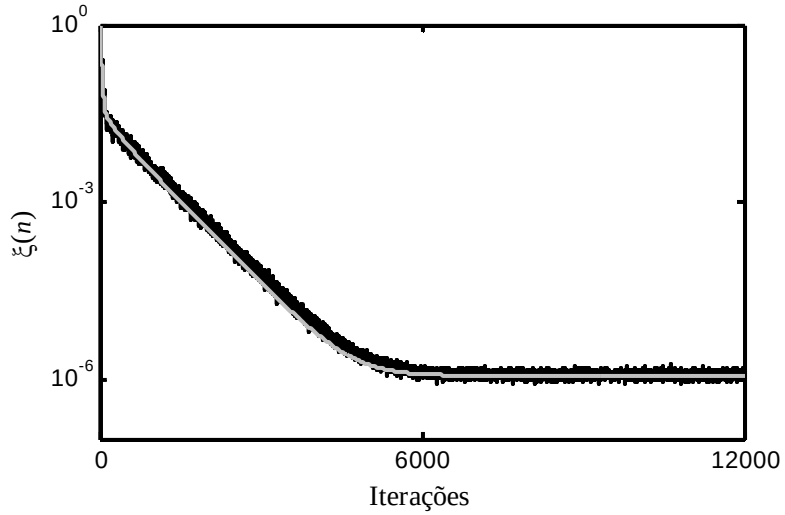


Figura 2.16. Exemplo 2.5. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}/2$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

Exemplo 2.6: Para este exemplo, uma planta Volterra de terceira ordem com tamanho de memória igual a 3 é considerada. Os vetores de coeficientes da planta são dados por

$$\mathbf{h}_1 = [-0,84 \quad -1,59 \quad -0,1200]^T$$
$$\mathbf{h}_2 = [-0,45 \quad 0,15 \quad -0,12 \quad 0,38 \quad -0,63 \quad 0,48]^T$$
$$\mathbf{h}_3 = [0,14 \quad -0,09 \quad 0,9 \quad -0,36 \quad -0,64 \quad -0,23 \quad -0,25 \quad -0,41 \quad 0,24 \quad 0,26]^T.$$

O sinal de entrada é branco gaussiano com variância $\sigma_x^2 = 0,5$. O passo de adaptação é $\mu = \mu_{crit}$ com $\mu_{crit} = 0,0003$. Os resultados obtidos são mostrados na Figura 2.17, de onde se observa um bom casamento entre simulação e modelo.

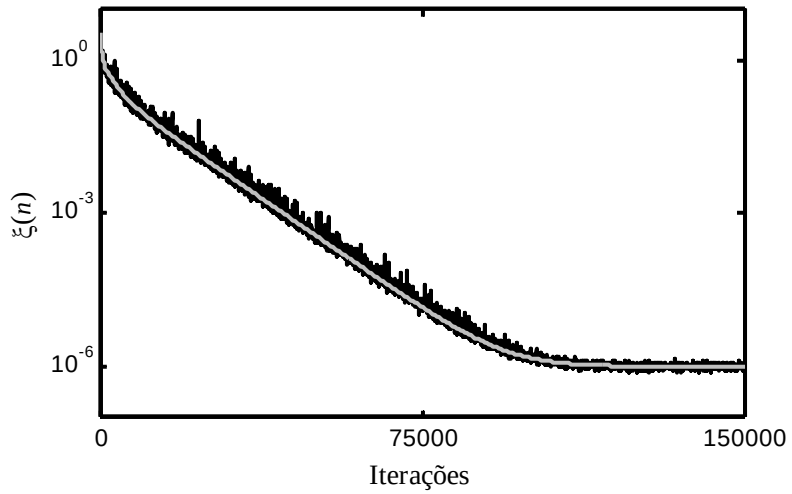


Figura 2.17. Exemplo 2.6. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

Exemplo 2.7: Neste exemplo, uma planta composta por um bloco de primeira ordem e um bloco de quarta ordem, ambos com tamanho de memória $N = 7$ (total de 217 coeficientes), é considerada. O sinal de entrada é branco com distribuição uniforme de probabilidade entre -1 e $+1$. O passo de adaptação usado é $\mu = \mu_{crit}$ com $\mu_{crit} = 0,007$. As curvas do erro quadrático médio obtidas são apresentadas na Figura 2.18. Como pode ser notado, uma concordância muito boa entre os resultados de simulação e o do modelo proposto é novamente verificada.

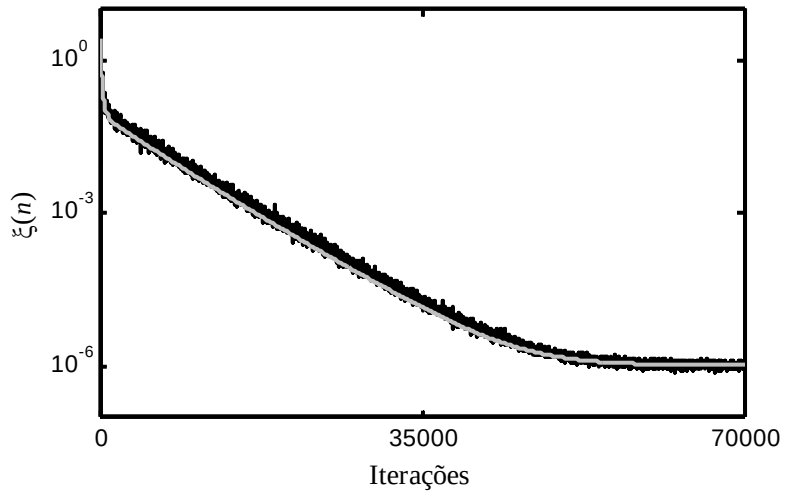


Figura 2.18. Exemplo 2.7. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

Exemplo 2.8: A planta para este exemplo é um filtro Volterra de segunda ordem com tamanho de memória 50 e, conseqüentemente, 1325 coeficientes. O uso de filtros Volterra com um número tão elevado de coeficientes é bastante restrito devido à alta complexidade computacional requerida. Assim, o objetivo aqui é avaliar o desempenho do modelo proposto e das aproximações realizadas nessa situação crítica. O sinal de entrada é branco gaussiano com variância $\sigma_x^2 = 1$. O valor obtido para μ_{crit} é $0,0001$ e os resultados são obtidos considerando $\mu = \mu_{\text{crit}}$ e $\mu = \mu_{\text{crit}}/3$. Tais resultados são mostrados nas Figuras 2.19 e 2.20, respectivamente. Como pode ser observado, uma pequena diferença na fase transiente entre a simulação e o modelo, que é conseqüência das aproximações adotadas, é verificada. Entretanto, tal diferença é quase desprezível para $\mu = \mu_{\text{crit}}/3$, permitindo o uso do modelo com uma precisão razoável mesmo para essa situação limite.

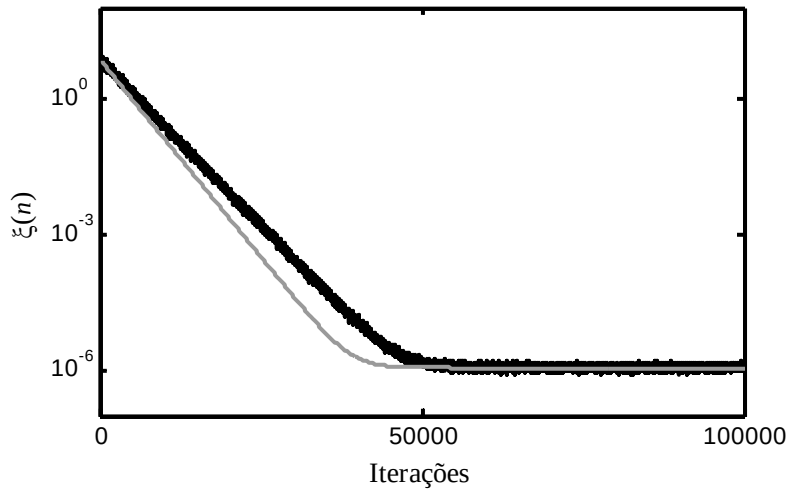


Figura 2.19. Exemplo 2.8. Evolução de $\xi(n)$ usando $\mu = \mu_{\text{crit}}$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

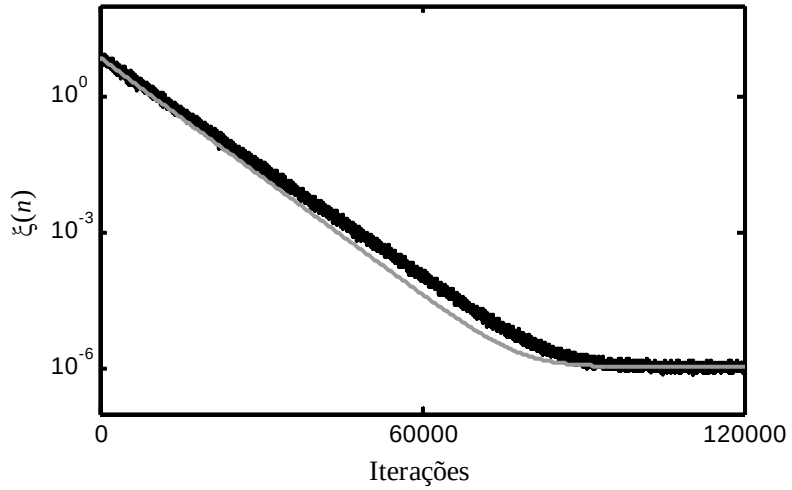


Figura 2.20. Exemplo 2.8. Evolução de $\xi(n)$ usando $\mu = \mu_{crit}/3$. (Linha irregular escura) simulação Monte Carlo. (Linha cinza) modelo proposto, obtido a partir de (2.73) e (2.78).

2.4 Considerações

Neste capítulo, as principais características do filtro Volterra e suas implicações práticas foram apresentadas. Tais características serão exploradas nos próximos capítulos com o objetivo de produzir novos resultados que venham a contribuir para a aplicação dos filtros Volterra em filtragem adaptativa. Além disso, expressões de modelagem para a curva de aprendizagem e para o primeiro e segundo momentos do filtro Volterra adaptado pelo algoritmo LMS também foram apresentadas neste capítulo. O modelo obtido apresenta um bom desempenho, tendo ainda a vantagem de sua aplicação não sofrer restrições quanto à ordem do filtro nem quanto ao modelo probabilístico do sinal de entrada.

CAPÍTULO 3

Filtros Volterra Interpolados Adaptativos

Como conseqüência da alta carga computacional requerida para implementação dos filtros Volterra adaptativos, diversas estruturas para sua implementação com redução de complexidade computacional vêm sendo desenvolvidas. Este capítulo trata de uma dessas classes de estruturas, a dos filtros Volterra interpolados [31], [33]-[35]. Nesse contexto, as principais características e a motivação para o desenvolvimento de tais filtros são aqui apresentadas. Adicionalmente, três estudos originais, relacionados aos filtros Volterra interpolados, são apresentados neste capítulo. O primeiro deles é a descrição da estrutura equivalente e do processo de recriação dos coeficientes por interpolação nas correspondentes estruturas [8]. O segundo estudo consiste no desenvolvimento de um procedimento para implementação de filtros interpolados com remoção do efeito de borda [36], proporcionando ganhos importantes de desempenho. Um terceiro e último estudo trata da modelagem estatística dos filtros Volterra interpolados adaptativos, sendo descritos o vetor ótimo de coeficientes, o erro quadrático médio mínimo e o comportamento de tais filtros quando adaptados pelo algoritmo LMS.

3.1 Implementações com Complexidade Reduzida do Filtro Volterra

As estratégias para implementação com complexidade reduzida do filtro Volterra e sua versão adaptativa discutidas na literatura podem ser divididas em três classes principais: implementação no domínio da freqüência, implementação baseada na decomposição da matriz de coeficientes e implementação esparsa. Recentemente, uma quarta classe, a de filtros Volterra interpolados, tem sido discutida na literatura [31], [33]-[34]. Nesta seção, as características gerais dessas diferentes classes são apresentadas e discutidas.

3.1.1 Implementação no Domínio da Freqüência

Uma das primeiras implementações no domínio da freqüência de filtros Volterra é apresentada em [37], envolvendo um filtro Volterra de segunda ordem adaptado pelo algoritmo LMS. Outra abordagem, descrevendo apenas a operação de filtragem sem considerar adaptação dos coeficientes, é dada em [38], com destaque para a possibilidade do uso de processamento paralelo para implementação da estrutura proposta. Outras abordagens, discutindo a operação de filtragem [39]-[40] ou também no contexto adaptativo [41]-[43] consideram a implementação no domínio da freqüência em conjunto com os métodos *overlap-and-save* e *overlap-and-add* [40] para redução de complexidade. Em geral, a principal vantagem das estruturas no domínio da freqüência está associada mais à velocidade de convergência (no contexto adaptativo) do que propriamente com a redução de complexidade computacional, uma vez que tal redução é geralmente obtida apenas nos casos em que filtros com tamanhos de memória muito grandes são considerados. Isso se deve em parte à necessidade de realização de transformadas de Fourier, através da FFT (*fast Fourier transform*) [1], para implementação dos filtros. Uma conseqüência direta do uso da FFT é a necessidade dos filtros apresentarem tamanhos de memória que são potências de dois como, por exemplo, 2^2, 2^3, 2^4, etc. Essa restrição é necessária para que a transformada possa ser implementada com menor impacto na complexidade computacional total do algoritmo.

3.1.2 Implementação Baseada na Decomposição da Matriz de Coeficientes

Conforme descrito em [44], implementações baseadas na decomposição da matriz de coeficientes são obtidas a partir da representação da relação de entrada e saída de um bloco não-linear do filtro Volterra como um produto entre dois vetores de entrada e uma matriz de coeficientes. A partir de tal representação, é possível realizar uma decomposição da matriz de coeficientes de forma a obter estruturas compostas por blocos conectados em paralelo [44]. Para obter implementações com complexidade reduzida, alguns desses blocos paralelos são desconsiderados de acordo com as características da decomposição matricial utilizada e do problema a ser tratado. Por exemplo, se a decomposição em valores singulares [21] é aplicada, é possível desconsiderar os blocos paralelos correspondentes aos menores autovalores da matriz de coeficientes [44]. Esse tipo de implementação foi originalmente proposto em [45], sendo também discutido em [44]. Sua aplicação no

contexto adaptativo é descrita em [46]. Adicionalmente, uma implementação no domínio da freqüência baseada na decomposição da matriz de coeficientes é dada em [47].

3.1.3 Implementação Esparsa

A idéia por trás das implementações esparsas é remover os coeficientes desnecessários ou menos importantes do filtro Volterra em função das características e requisitos de desempenho em uma determinada aplicação. Com isso, busca-se aumentar a eficiência do filtro do ponto de vista de complexidade computacional, mesmo que se tenha que admitir, freqüentemente, alguma perda de desempenho. Tais implementações podem ser subdivididas em duas classes: com estimação de regiões dominantes e truncadas em número de diagonais.

3.1.3.1 Implementações com Estimação de Regiões Dominantes

As implementações com estimação de regiões dominantes são, em geral, realizadas a partir da estimação dos coeficientes de maior importância (maior magnitude) do filtro Volterra. Feito isso, são introduzidos atrasos no sinal de entrada para tornar possível a implementação apenas da região principal do vetor de coeficientes. Conseqüentemente, uma considerável redução no número de coeficientes é obtida a custa de alguma perda de desempenho. As principais implementações desse tipo são discutidas em [48]-[51]. Apesar de tais implementações apresentarem muito bom desempenho em aplicações específicas, elas apresentam a desvantagem da necessidade de se detectar ou estimar os coeficientes mais importantes, o que implica custos em termos de complexidade computacional.

3.1.3.2 Implementações Truncadas em Número de Diagonais

As implementações truncadas em número de diagonais são baseadas na representação em coordenadas diagonais da relação de entrada e saída do filtro Volterra, descrita na Seção 2.2.6. Conforme discutido em [25], existem aplicações em que os coeficientes de maior importância do filtro Volterra são os mais próximos das diagonais principais dos blocos. Assim, nesses casos é possível desconsiderar os coeficientes das diagonais mais distantes da diagonal principal sem levar a perdas de desempenho significativas. Um dos primeiro trabalhos que tratam de tal conceito é apresentado em [25]. Implementações propostas em alguns trabalhos anteriores [26], [52] apresentam implicitamente tal característica. A implementação descrita em [53] e os *power filters* propostos em [54] também são baseados nesse conceito, sendo que, no caso dos *power*

filters, apenas a diagonal principal de cada bloco é considerada [54]. Ainda, em [55], a implementação de filtros Volterra no domínio da freqüência utilizando também um truncamento do número de diagonais é apresentada. De maneira geral, a idéia por trás das implementações truncadas em número de diagonais é interessante, apresentando bom desempenho em algumas aplicações específicas. No entanto, de forma análoga às implementações com estimação de regiões dominantes, uma escolha inapropriada do número de diagonais a ser considerado pode levar a importantes perdas de desempenho. Esse problema é ainda mais crítico no caso dos *power filters*, nos quais apenas a diagonal principal é considerada.

3.1.4 Implementações Interpoladas

Nas abordagens interpoladas, a redução de complexidade computacional também é obtida a partir do uso de estruturas esparsas [34]. No entanto, a presença de um filtro interpolador implica na recriação de coeficientes e em estruturas equivalentes não-esparsas análogas aos filtros FIR interpolados [5], [56]. Assim, as implementações interpoladas podem ser consideradas constituintes de uma terceira classe de estruturas para implementação com complexidade reduzida de filtros Volterra. As principais características de tal classe de estruturas são descritas ao longo deste capítulo, sendo que, na próxima seção, são apresentadas as definições básicas das estruturas que motivaram o seu desenvolvimento, a saber, os filtros FIR interpolados.

3.2 Definições Básicas de Filtros FIR Interpolados

Um filtro interpolado de reposta ao impulso finita (*interpolated* FIR - IFIR) é uma realização com número reduzido de coeficientes de um filtro FIR convencional [5], [56]. A idéia básica por trás dos filtros IFIR é remover alguns dos coeficientes do filtro FIR e depois recriá-los usando um filtro interpolador. O diagrama de blocos de tal estrutura está ilustrado na Figura 3.1, onde estão mostrados o filtro FIR convencional $\mathbf{w}$ e sua versão interpolada, formada pelo interpolador $\mathbf{g}$ em série com o filtro esparso $\mathbf{w}_s$. Os sinais $x(n)$ e $y(n)$ representam, respectivamente, os sinais de entrada e saída do filtro FIR padrão, resultando na seguinte relação de entrada e saída:

$$y(n) = \mathbf{x}^{\mathrm{T}}(n)\mathbf{w} \tag{3.1}$$

com

$$\mathbf{w} = [w(0) \;\; w(1) \;\; \cdots \;\; w(N-1)]^{\mathrm{T}} \tag{3.2}$$

e

$$\mathbf{x}(n) = \{x(n) \;\; x(n-1) \;\; x(n-2) \;\; \cdots \;\; x(n-N+1)\}^{\mathrm{T}}. \tag{3.3}$$

O filtro interpolador é um filtro FIR com M coeficientes e resposta ao impulso definida pelo vetor $\mathbf{g} = [g(0) \;\; g(1) \;\; \cdots \;\; g(M-1)]^{\mathrm{T}}$. Assim, a saída $\tilde{x}(n)$ do interpolador é relacionada ao sinal de entrada por

$$\tilde{x}(n) = \sum_{j=0}^{M-1} g(j)x(n-j). \tag{3.4}$$

O fator que determina a esparsidade de $\mathbf{w}_{\mathrm{s}}$ é denominado fator de interpolação e é denotado por L [56]. O vetor de coeficientes do filtro esparso $\mathbf{w}_{\mathrm{s}}$ é então obtido colocando para zero $(L-1)$ amostras de cada L amostras consecutivas do vetor de coeficientes (3.2) do filtro FIR padrão. Dessa maneira, são colocados para zero os coeficientes cujos índices não são múltiplos de L, resultando no seguinte vetor de dimensão $(N \times 1)$:

$$\mathbf{w}_{\mathrm{s}} = \{w(0) \;\; 0 \;\; \cdots \;\; w(L) \;\; 0 \;\; \cdots \;\; w(2L) \;\; 0 \;\; \cdots \;\; w[(N_{\mathrm{s}}-1)L] \;\; 0 \;\; \cdots \;\; 0\}^{\mathrm{T}}. \tag{3.5}$$

O vetor de entrada correspondente é dado por

$$\tilde{\mathbf{x}}(n) = \{\tilde{x}(n) \;\; \tilde{x}(n-1) \;\; \tilde{x}(n-2) \;\; \cdots \;\; \tilde{x}(n-N+1)\}^{\mathrm{T}}. \tag{3.6}$$

Assim, o sinal de saída $\hat{y}(n)$ do filtro IFIR, que é uma versão aproximada de $y(n)$ visto que a estrutura interpolada é uma aproximação do filtro FIR padrão, é dado por

$$\hat{y}(n) = \tilde{\mathbf{x}}^{\mathrm{T}}(n)\mathbf{w}_{\mathrm{s}}. \tag{3.7}$$

O número de coeficientes diferentes de zero em (3.5) é dado pela expressão

$$N_{\mathrm{s}} = \left\lfloor \frac{N-1}{L} \right\rfloor + 1 \tag{3.8}$$

com $\lfloor \cdot \rfloor$ denotando a operação de truncamento e N, o tamanho de memória do filtro padrão.

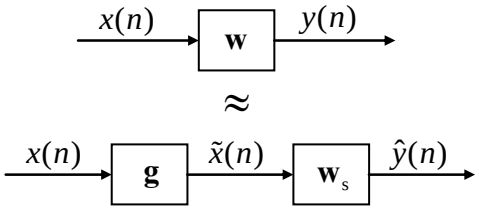


Figura 3.1. Diagrama de blocos de uma estrutura IFIR.

O filtro IFIR também pode ser implementado invertendo as posições do interpolador e do filtro esparso na estrutura ilustrada na Figura 3.1. Assim, obtém-se a estrutura apresentada na Figura 3.2, onde estão presentes os mesmos elementos apresentados na Figura 3.1 e, ainda, $\hat{x}(n)$ representando o sinal de entrada filtrado pelo filtro esparso. Para esse caso, a saída do filtro é

$$\hat{y}(n) = \hat{\mathbf{x}}^{\mathrm{T}}(n)\mathbf{g} \tag{3.9}$$

com

$$\hat{\mathbf{x}}(n) = \left\{\hat{x}(n)\ \ \hat{x}(n-1)\ \ \hat{x}(n-2)\ \ \cdots\ \ \hat{x}(n-M+1)\right\}^{\mathrm{T}} \tag{3.10}$$

e

$$\hat{x}(n) = \mathbf{w}_{\mathrm{s}}^{\mathrm{T}}\mathbf{x}(n). \tag{3.11}$$

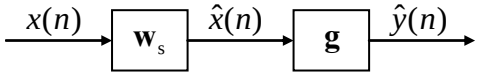


Figura 3.2. Diagrama de blocos de uma estrutura IFIR com interpolador na saída.

Em comum, as estruturas da Figura 3.1 e da Figura 3.2 apresentam a mesma função de transferência e o mesmo vetor de coeficientes equivalente. Tal vetor, denotado por $\mathbf{w}_{\mathrm{i}}$, é obtido a partir da convolução das respostas ao impulso do filtro esparso $\mathbf{w}_{\mathrm{s}}$ e do interpolador $\mathbf{g}$, possuindo, assim, $N + M - 1$ coeficientes.

Conforme mencionado, o papel do filtro interpolador em uma estrutura IFIR é recriar os coeficientes zerados em (3.5). Geralmente, esses coeficientes são recriados em função dos valores de seus coeficientes vizinhos. Tal processo é aqui ilustrado através de um exemplo com $L = 2$ e usando um interpolador linear, cujo vetor de coeficientes é dado por $\mathbf{g} = [0{,}5\ \ 1\ \ 0{,}5]^{\mathrm{T}}$. Para esse caso, considerando um filtro esparso com $N = 5$ e vetor de coeficientes dado por

$$\mathbf{w}_{\mathrm{s}} = [w(0)\ \ 0\ \ w(2)\ \ 0\ \ w(4)]^{\mathrm{T}} \tag{3.12}$$

o vetor equivalente, obtido convolvendo as respostas de $\mathbf{g}$ e $\mathbf{w}_{\mathrm{s}}$, é dado por

$$\mathbf{w}_{\mathrm{i}} = \left[\underline{0,5w(0)}\ w(0)\ \boxed{0,5w(0)+0,5w(2)}\ w(2)\ \boxed{0,5w(2)+0,5w(4)}\ w(4)\ \underline{0,5w(4)} \right]^{\mathrm{T}}. \quad (3.13)$$

A partir de (3.13), verifica-se que o procedimento de recriação dos coeficientes no filtro IFIR ocorre como segue: os coeficientes iguais a zero em (3.12) são reconstruídos em função dos valores de seus vizinhos, conforme indicado pelos retângulos em (3.13). Nesse caso, como $\mathbf{g} = [0,5\ \ 1\ \ 0,5]^{\mathrm{T}}$, os coeficientes iguais a zero são reconstruídos como a média aritmética dos seus vizinhos. Além disso, os coeficientes sublinhados em (3.13) surgem como efeito de borda do processo de interpolação. Assim, fica evidente que a estrutura interpolada constitui-se de uma aproximação do filtro convencional, proporcionando melhores resultados se o interpolador produzir coeficientes com valores mais próximos daqueles que foram colocados para zero para obtenção de (3.12). Substituindo, agora, o interpolador linear por um interpolador genérico com $\mathbf{g} = \left[g(0)\ \ g(1)\ \ g(2) \right]^{\mathrm{T}}$, obtém-se

$$\begin{aligned} \mathbf{w}_{\mathrm{i}} = \Big[& \underline{g(0)w(0)}\ \ \mathrm{g}(1)w(0)\ \ \boxed{g(2)w(0)+g(0)w(2)}\ \ \mathrm{g}(1)w(2) \\ & \boxed{g(2)w(2)+g(0)w(4)}\ \ \mathrm{g}(1)w(4)\ \ \underline{g(2)w(4)} \Big]^{\mathrm{T}}. \end{aligned} \quad (3.14)$$

Comparando (3.13) com (3.14), observa-se que os coeficientes do filtro interpolador têm papéis bastante distintos. Enquanto o coeficiente central $g(1)$ tem a função de replicar os coeficientes diferentes de zero do filtro esparso, os coeficientes $g(0)$ e $g(2)$ são os responsáveis pela recriação dos coeficientes que estão zerados a partir de seus vizinhos. Assim, de maneira geral, o interpolador deve possuir um coeficiente central e $2(L-1)$ coeficientes laterais, de forma que os coeficientes que estão zerados possam ser recriados em função de seus vizinhos. Portanto, o tamanho de memória do interpolador em função do fator de interpolação L é

$$M(L) = 1 + 2(L-1) = 2L-1. \quad (3.15)$$

Por exemplo, um fator de interpolação $L = 3$ resulta em $M = 5$ e $\mathbf{g} = [g(0)\ \mathrm{g}(1)\ \mathrm{g}(2)\ \mathrm{g}(3)\ \mathrm{g}(4)]^{\mathrm{T}}$. Considerando, ainda, um filtro esparso com $N = 7$ $(N_{\mathrm{s}} = 3)$, temos

$$\mathbf{w}_{\mathrm{s}} = [w(0)\ \ 0\ \ 0\ \ w(3)\ \ 0\ \ 0\ \ w(6)]^{\mathrm{T}} \quad (3.16)$$

e um vetor equivalente dado por

$$\mathbf{w}_{\mathrm{i}} = \Big[\underline{g(0)w(0)} \;\; \underline{g(1)w(0)} \;\; g(2)w(0) \;\; \boxed{g(3)w(0)+g(0)w(3)} \;\; \boxed{g(4)w(0)+g(1)w(3)} \;\; g(2)w(3) \;\; \boxed{g(3)w(3)+g(0)w(6)} \;\; \boxed{g(4)w(3)+g(1)w(6)} \;\; g(2)w(6) \;\; \underline{g(3)w(6)} \;\; \underline{g(4)w(6)} \Big]^{\mathrm{T}}. \tag{3.17}$$

Em (3.17), constata-se a presença dos coeficientes recriados (destacados por caixas) e também aqueles resultantes do efeito de borda (sublinhados). Nota-se, ainda, a replicação dos coeficientes diferentes de zero do vetor esparso (3.16), aqui multiplicados pelo coeficiente central do interpolador $[g(2)]$.

A partir das informações apresentadas nesta seção, observa-se que o uso de filtros IFIR para implementação de filtros FIR leva a uma considerável redução no número total de coeficientes requerido. Do ponto de vista prático, tal redução pode também ser considerada uma redução no tamanho de memória do filtro uma vez que o filtro esparso da estrutura interpolada é uma versão dizimada do filtro FIR convencional. Essa característica é bastante interessante especialmente no contexto adaptativo, pois proporciona implementações com menor custo computacional e, em alguns casos, com melhores taxas de convergência [57].

3.3 Filtros Volterra Interpolados Adaptativos

No caso do filtro Volterra, uma redução no tamanho de memória do filtro leva a uma redução ainda maior no número de coeficientes, o que pode ser facilmente observado a partir de (2.19) e (2.33). Assim, o uso de uma abordagem interpolada, similar à descrita para os filtros IFIR na seção anterior, torna-se bastante sedutora. Nesta seção, os filtros Volterra interpolados são apresentados destacando a descrição de sua estrutura, a aplicação no contexto adaptativo e também a avaliação da complexidade computacional requerida para sua implementação.

3.3.1 Implementação Interpolada do Filtro Volterra

Conforme apresentado em [34], a estrutura usada para implementação do filtro Volterra interpolado, ilustrada na Figura 3.3, é similar a dos filtros IFIR com interpolador na entrada. Na parte superior de tal figura, é mostrada a forma simplificada da estrutura do filtro Volterra interpolado, onde $x(n)$ é o sinal de entrada, $\mathbf{g}$ representa o filtro interpolador, $\tilde{\mathbf{x}}_{\mathrm{V}}(n)$ é o vetor de entrada Volterra interpolado, o qual é similar a (2.13), porém obtido a partir do sinal de entrada interpolado, resultando em

$$\tilde{\mathbf{x}}_{\mathrm{V}}(n) = \left[\tilde{\mathbf{x}}_1(n) \;\; \tilde{\mathbf{x}}_2(n) \;\; \cdots \;\; \tilde{\mathbf{x}}_P(n)\right]^{\mathrm{T}} \tag{3.18}$$

$\mathbf{h}_{\mathrm{Vs}}$ é o vetor de coeficientes Volterra esparso, dado por

$$\mathbf{h}_{\mathrm{Vs}} = \left[\mathbf{h}_{1\mathrm{s}} \;\; \mathbf{h}_{2\mathrm{s}} \;\; \cdots \;\; \mathbf{h}_{P\mathrm{s}}\right]^{\mathrm{T}} \tag{3.19}$$

e $\hat{y}(n)$ representa o sinal de saída, obtido por

$$\hat{y}(n) = \tilde{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{Vs}}\,. \tag{3.20}$$

Na parte inferior da Figura 3.3, a estrutura de blocos do filtro Volterra interpolado é apresentada de forma expandida, com $\tilde{\mathbf{x}}_p(n)$ representando o vetor de entrada interpolado para cada bloco $\mathbf{h}_{p\mathrm{s}}$ de ordem p, e $\hat{y}_p(n)$ a saída de cada bloco esparso também de ordem p.

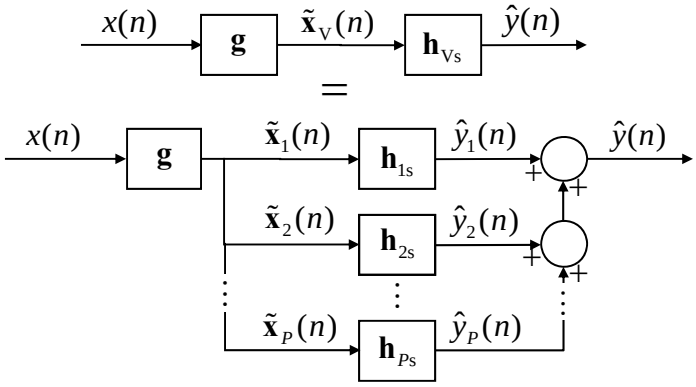


Figura 3.3. Diagrama de blocos do filtro Volterra interpolado.

De maneira similar aos filtros IFIR, o bloco esparso de primeira ordem do filtro Volterra interpolado possui o seguinte vetor de entrada:

$$\tilde{\mathbf{x}}_1(n) = \{\tilde{x}(n) \;\; \tilde{x}(n-1) \;\; \tilde{x}(n-2) \;\; \cdots \;\; \tilde{x}[n-N+1]\}^{\mathrm{T}} \tag{3.21}$$

com o respectivo vetor de coeficientes dado por

$$\mathbf{h}_{1\mathrm{s}} = \left\{h_1(0) \;\; 0 \;\; \cdots \;\; h_1(L) \;\; 0 \;\; \cdots \;\; h_1[(N_{\mathrm{s}}-1)L] \;\; 0 \;\; \cdots \;\; 0\right\}^{\mathrm{T}} \tag{3.22}$$

e relação de entrada e saída por

$$\hat{y}_1(n) = \tilde{\mathbf{x}}_1^{\mathrm{T}}(n)\mathbf{h}_{1\mathrm{s}}. \tag{3.23}$$

Considerando agora o bloco de segunda ordem, o vetor de entrada é obtido a partir de (2.7), resultando em

$$\tilde{\mathbf{x}}_2(n) = \tilde{\mathbf{x}}_1(n) \otimes \tilde{\mathbf{x}}_1(n). \tag{3.24}$$

O processo de derivação do vetor de coeficientes esparso de segunda ordem é melhor visualizado quando expressado usando a representação espacial dos blocos descrita na

Seção 2.2.3. Tomando como exemplo um filtro Volterra convencional com memória igual a 3, tem-se a seguinte matriz de coeficientes de segunda ordem:

$$\mathbf{H}_2 = \begin{bmatrix} h_2(0,0) & h_2(0,1) & h_2(0,2) \\ h_2(1,0) & h_2(1,1) & h_2(1,2) \\ h_2(2,0) & h_2(2,1) & h_2(2,2) \end{bmatrix}. \tag{3.25}$$

A versão esparsa de (3.25) é obtida de forma análoga à adotada para obtenção de (3.5) e (3.22). Desse modo, são zerados os coeficientes de (3.25) que possuem ao menos um índice que não é múltiplo de L. Por exemplo, se $L = 2,$ a partir de (3.25), obtém-se

$$\mathbf{H}_{2s} = \begin{bmatrix} h_2(0,0) & 0 & h_2(0,2) \\ 0 & 0 & 0 \\ h_2(2,0) & 0 & h_2(2,2) \end{bmatrix}. \tag{3.26}$$

Escrevendo (3.26) na forma vetorial, tem-se

$$\mathbf{h}_{2s} = [h_2(0,0) \;\; 0 \;\; h_2(0,2) \;\; 0 \;\; 0 \;\; 0 \;\; h_2(2,0) \;\; 0 \;\; h_2(2,2)]^{\mathrm{T}}. \tag{3.27}$$

Considerando então (3.24) e (3.27), a relação de entrada e saída do bloco esparso de segunda ordem é

$$\hat{y}_2(n) = \tilde{\mathbf{x}}_2^{\mathrm{T}}(n)\mathbf{h}_{2s}. \tag{3.28}$$

Os demais blocos possuem relações de entrada e saída similares à do bloco de segunda ordem, resultando em

$$\hat{y}_p(n) = \tilde{\mathbf{x}}_p^{\mathrm{T}}(n)\mathbf{h}_{ps} \tag{3.29}$$

com

$$\tilde{\mathbf{x}}_p(n) = \tilde{\mathbf{x}}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}(n) \tag{3.30}$$

e vetor de coeficientes esparso obtido de maneira similar à descrita para os blocos de primeira e segunda ordens, ou seja, zerando os coeficientes que apresentam ao menos um índice que não é múltiplo de L.

3.3.2 Contexto Adaptativo

A implementação dos filtros Volterra interpolados no contexto adaptativo é comumente realizada com a adaptação apenas dos coeficientes do filtro esparso. Essa característica, aliada ainda à representação triangular da relação de entrada e saída, torna bastante interessante o uso de tais filtros uma vez que a carga computacional é reduzida consideravelmente.

3.3.2.1 Filtro Volterra Interpolado Adaptativo

Como um exemplo de filtro Volterra interpolado adaptativo, podemos considerar uma estrutura Volterra interpolada de segunda ordem (Figura 3.3) adaptada pelo algoritmo LMS, aplicada a um problema de estimação de sinal (Figura 1.1). Tal esquema é ilustrado na Figura 3.4, com $e(n)$ representando o sinal de erro e $d(n)$ o sinal a ser estimado. A extensão desse problema para filtros de maior ordem é direta, apenas incluindo os demais blocos à estrutura.

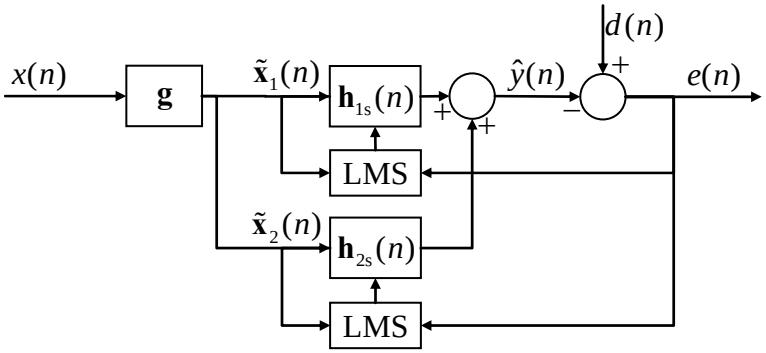


Figura 3.4. Estrutura de um filtro Volterra interpolado adaptativo de segunda ordem.

Assim, considerando o algoritmo LMS [1] e a representação triangular da relação de entrada e saída Volterra descrita na Seção 2.2.5, a equação de atualização dos coeficientes é escrita como

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \underline{\mathbf{h}}_{\mathrm{Vs}}(n) + 2\mu e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) \tag{3.31}$$

onde μ é o passo de adaptação, $\underline{\mathbf{h}}_{\mathrm{Vs}}$ é o vetor de coeficientes esparso na representação triangular e $\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)$ é o vetor de entrada correspondente da representação triangular obtido considerando o sinal de entrada interpolado. Agora, se utilizado o algoritmo NLMS [1] para o mesmo problema, tem-se

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \underline{\mathbf{h}}_{\mathrm{Vs}}(n) + \frac{\alpha}{\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) + \psi} e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) \tag{3.32}$$

onde α é o parâmetro de controle do algoritmos NLMS e ψ é uma constante pequena e positiva que impede que α seja dividido por valores muito pequenos quando $\tilde{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\tilde{\mathbf{x}}_{\mathrm{V}}(n)$ é próximo a zero. Outros algoritmos adaptativos também podem ser considerados, como, por exemplo, o algoritmo RLS [2] ou o algoritmo de projeções afins (AP) [2].

3.3.2.2 Filtro Volterra Parcialmente Interpolado Adaptativo

Uma outra estrutura interpolada para a implementação de filtros Volterra no contexto adaptativo é obtida levando em conta a estrutura de blocos do filtro Volterra. Uma vez que os blocos não-lineares são os que apresentam maior número de coeficientes, é possível obter praticamente a mesma redução de complexidade alcançada no caso anterior interpolando somente os coeficientes dos blocos de ordem superior. Esse procedimento resulta no filtro Volterra parcialmente interpolado adaptativo. Uma estrutura desse tipo é ilustrada na Figura 3.5 considerando um problema de estimação de sinal por um filtro de segunda ordem. Destaca-se, nessa figura, que o bloco de primeira ordem não apresenta esparsidade e, portanto, o seu sinal de entrada não é interpolado, enquanto o bloco de segunda ordem é esparso tendo o seu sinal de entrada interpolado. Como no caso anterior, a extensão para filtros de maior ordem é realizada de forma direta.

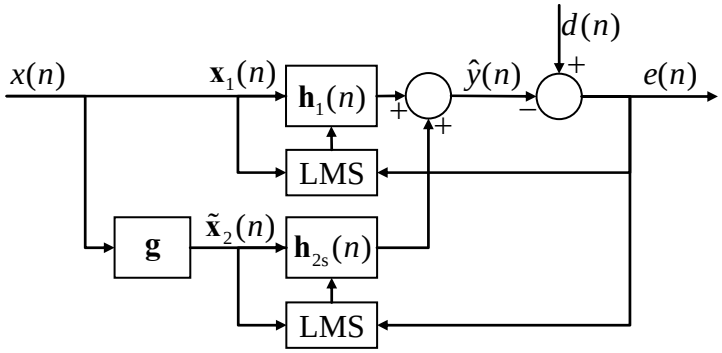


Figura 3.5. Estrutura de um filtro Volterra parcialmente interpolado adaptativo de segunda ordem.

O processo de adaptação para este caso possui diferenças apenas no bloco de primeira ordem (sem interpolação). Assim, definindo um vetor de coeficientes parcialmente esparso, usando a representação triangular da relação de entrada e saída do filtro Volterra, como

$$\underline{\mathbf{h}}_{\mathrm{VPs}} = \left[\underline{\mathbf{h}}_1 \;\; \underline{\mathbf{h}}_{2\mathrm{s}} \;\; \cdots \;\; \underline{\mathbf{h}}_{P\mathrm{s}}\right]^{\mathrm{T}} \tag{3.33}$$

e um vetor de entrada parcialmente interpolado como

$$\underline{\tilde{\mathbf{x}}}_{\mathrm{VP}}(n) = \left[\underline{\mathbf{x}}_1(n) \;\; \underline{\tilde{\mathbf{x}}}_2(n) \;\; \cdots \;\; \underline{\tilde{\mathbf{x}}}_P(n)\right]^{\mathrm{T}} \tag{3.34}$$

a equação de adaptação dos coeficientes usando o algoritmo LMS [1] é dada por

$$\underline{\mathbf{h}}_{\mathrm{VPs}}(n+1) = \underline{\mathbf{h}}_{\mathrm{VPs}}(n) + 2\mu e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{VP}}(n). \tag{3.35}$$

De forma análoga, usando o algoritmo NLMS [1], tem-se

$$\underline{\mathbf{h}}_{\mathrm{VPs}}(n+1) = \underline{\mathbf{h}}_{\mathrm{VPs}}(n) + \frac{\alpha}{\underline{\tilde{\mathbf{x}}}_{\mathrm{VP}}^{\mathrm{T}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{VP}}(n) + \psi} e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{VP}}(n). \tag{3.36}$$

Novamente destaca-se que outros algoritmos adaptativos também podem ser utilizados.

3.3.3 Complexidade Computacional

Com o objetivo de analisar a carga computacional requerida para implementação de um filtro Volterra interpolado, o custo computacional do filtro interpolador pode ser desconsiderado uma vez que tal filtro não participa do processo adaptativo e também possui um número pequeno de coeficientes. Assim, a comparação do custo computacional para implementação de filtros Volterra convencionais e interpolados pode ser realizada diretamente em termos do número de coeficientes que participam do processo adaptativo. Para o filtro Volterra interpolado, tal número é igual ao número de coeficientes do filtro esparso, o qual, a partir de (2.33), é dado por

$$\underline{D}_{\mathrm{Vi}}(N_{\mathrm{s}}, P) = \frac{(N_{\mathrm{s}} + P)!}{N_{\mathrm{s}}!P!} - 1. \tag{3.37}$$

Com relação à implementação parcialmente interpolada, o número de coeficientes que participam do processo adaptativo é igual ao dado por (3.37) acrescido de $(N - N_{\mathrm{s}})$ coeficientes devido à natureza não-esparsa do bloco de primeira ordem. Assim,

$$\underline{D}_{\mathrm{VPi}}(N_{\mathrm{s}}, P) = \frac{(N_{\mathrm{s}} + P)!}{N_{\mathrm{s}}!P!} - 1 + N - N_{\mathrm{s}}. \tag{3.38}$$

A Figura 3.6 apresenta curvas que ilustram o número de coeficientes em função do tamanho de memória para diferentes implementações do filtro Volterra de segunda e terceira ordens. Dessa figura, verifica-se que as implementações interpoladas proporcionam uma considerável redução no número de coeficientes quando comparadas às implementações convencionais. Também se observa que as duas implementações interpoladas consideradas apresentam praticamente a mesma redução de complexidade. Adicionalmente, o número de coeficientes requerido pelas diferentes implementações do filtro Volterra para alguns valores de tamanho de memória são apresentados nas Tabelas 3-1 e 3-2 de forma a deixar mais claras as informações apresentadas na Figura 3.6.

Tabela 3-1. Número de coeficientes requerido pelas diferentes implementações de filtros Volterra de segunda ordem para alguns valores de tamanho de memória.

Tipo de implementação	Tamanho de memória				
	20	40	60	80	100
Volterra repr. convencional	420	1640	3660	6480	10100
Volterra repr. triangular	230	860	1890	3320	5150
Volterra interpolado	65	230	495	860	1325
Volterra parc. interpolado	75	250	525	900	1375

Tabela 3-2. Número de coeficientes requerido pelas diferentes implementações de filtros Volterra de terceira ordem para alguns valores de tamanho de memória.

Tipo de implementação	Tamanho de memória				
	20	40	60	80	100
Volterra repr. convencional	8420	65640	219660	518480	1010100
Volterra repr. triangular	1770	12340	39710	91880	176850
Volterra interpolado	285	1770	5455	12340	23425
Volterra parc. interpolado	295	1790	5485	12380	23475

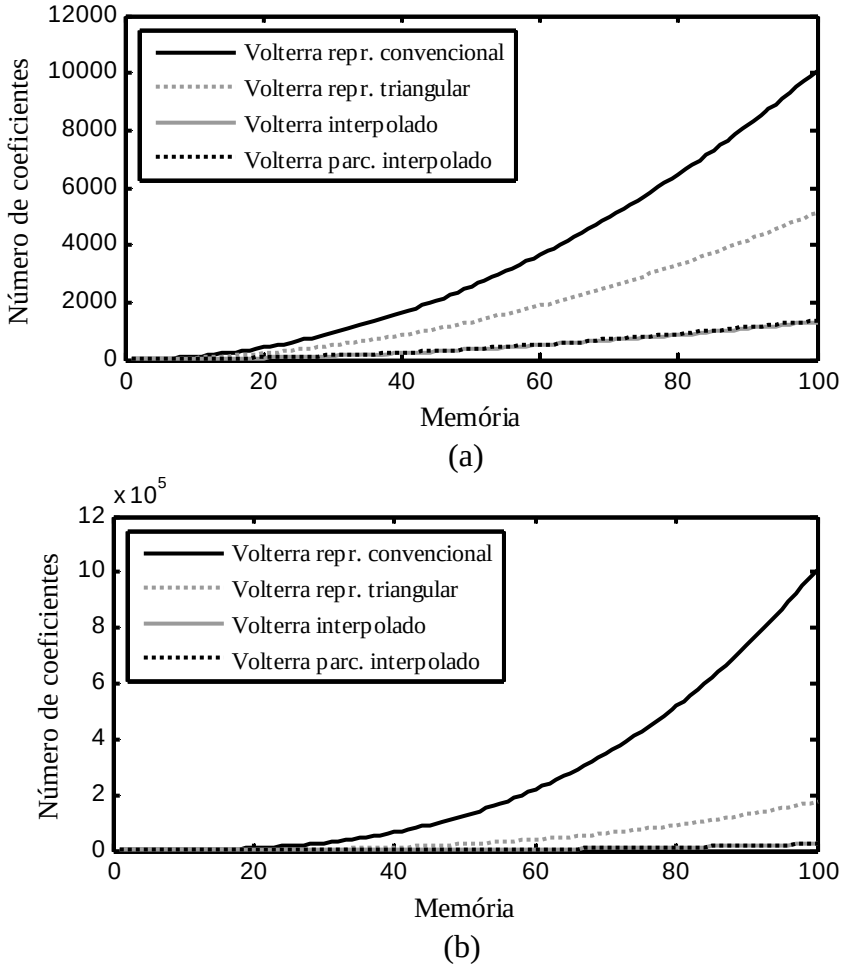


Figura 3.6. Número de coeficientes em função do tamanho de memória para diferentes implementações de um filtro Volterra. (a) Implementações de segunda ordem. (b) Implementações de terceira ordem.

3.4 Descrição de Filtros Volterra Interpolados

Conforme descrito na seção anterior, os filtros Volterra interpolados apresentam um grande potencial de aplicação devido, principalmente, a sua considerável redução de carga computacional, quando comparada a das implementações Volterra convencionais. No entanto, alguns problemas ainda limitam o uso de tais estruturas. Um desses problemas é a falta de uma base teórica mais sólida, uma vez que existem apenas procedimentos práticos relativos a sua implementação [34]. Contribuindo nesse sentido, uma descrição dos filtros Volterra interpolados [8] é apresentada nesta seção. O que se busca é o desenvolvimento de expressões matemáticas que permitam descrever a relação de entrada e saída de tais filtros de uma maneira simples. Com isso, é possível verificar, dentre outras características, a maneira como os coeficientes são recriados através do processo de interpolação.

3.4.1 Função de Transferência Global de Filtros IFIR

Conforme descrito na Seção 3.2, o vetor de coeficientes equivalente $\mathbf{w}_{\mathrm{i}}$ da estrutura IFIR é obtido convolvendo a resposta ao impulso do interpolador $\mathbf{g}$ (M elementos) com a do filtro esparso $\mathbf{w}_{\mathrm{s}}$ (N elementos). Como conseqüência, tal vetor apresenta $(N+M-1)$ elementos. De forma a facilitar o desenvolvimento matemático, é interessante obter $\mathbf{w}_{\mathrm{i}}$ usando um produto matricial. Isso é realizado definindo a seguinte matriz de interpolação

$$\mathbf{G}=\begin{bmatrix} g(0) & 0 & 0 & \cdots & 0 \\ g(1) & g(0) & 0 & \cdots & 0 \\ g(2) & g(1) & g(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g(M-1) & g(M-2) & g(M-3) & \cdots & g(0) \\ 0 & g(M-1) & g(M-2) & \cdots & g(1) \\ 0 & 0 & g(M-1) & \cdots & g(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & g(M-1) \end{bmatrix} \tag{3.39}$$

com dimensões $(N+M-1)\times N$ e cujos elementos são obtidos do vetor de interpolação $\mathbf{g}=[g(0)\ \ g(1)\ \ \cdots\ \ g(M-1)]^{\mathrm{T}}$. Considerando tal matriz, o vetor de coeficientes equivalente pode ser obtido fazendo

$$\mathbf{w}_{\mathrm{i}}=\mathbf{G}\mathbf{w}_{\mathrm{s}}. \tag{3.40}$$

Alternativamente, definindo a seguinte matriz

$$\mathbf{W}_{\mathrm{s}}=\begin{Bmatrix} \begin{bmatrix} \\ \mathbf{w}_{\mathrm{s}} \\ \\ \end{bmatrix} & 0 & 0 & \cdots & 0 \\ 0 & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{s}} \\ \\ \end{bmatrix} & 0 & \cdots & 0 \\ 0 & 0 & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{s}} \\ \\ \end{bmatrix} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{s}} \\ \\ \end{bmatrix} \end{Bmatrix} \tag{3.41}$$

com dimensões $(N+M-1)\times M$ é possível, ainda, reescrever (3.40) como

$$\mathbf{w}_{\mathrm{i}}=\mathbf{W}_{\mathrm{s}}\mathbf{g}=\mathbf{G}\mathbf{w}_{\mathrm{s}}. \tag{3.42}$$

Definindo-se, agora, um vetor de entrada estendido com $(N+M-1)$ amostras do sinal de entrada, como

$$\mathbf{x}_{\mathrm{e}}(n)=\left[x(n)\ \ x(n-1)\ \ \cdots\ \ x(n-N-M+2)\right]^{\mathrm{T}} \tag{3.43}$$

é possível reescrever (3.6) e (3.10) como

$$\tilde{\mathbf{x}}(n) = \mathbf{G}^{\mathrm{T}}\mathbf{x}_{\mathrm{e}}(n) \tag{3.44}$$

e

$$\hat{\mathbf{x}}(n) = \mathbf{W}_{\mathrm{s}}^{\mathrm{T}}\mathbf{x}_{\mathrm{e}}(n) \tag{3.45}$$

respectivamente. Assim, a relação de entrada e saída global do filtro IFIR pode ser expressa como

$$\hat{y}(n) = \mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)\mathbf{w}_{\mathrm{i}} = \mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}\mathbf{w}_{\mathrm{s}} = \mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)\mathbf{W}_{\mathrm{s}}\mathbf{g}. \tag{3.46}$$

3.4.2 Extensão para Filtros Volterra Interpolados

A extensão do procedimento da seção anterior para o bloco de primeira ordem do filtro Volterra interpolado é realizada de maneira direta. Assim, considerando o vetor de coeficientes de primeira ordem dado por (3.22) e o vetor de entrada por (3.21), é possível reescrever (3.21) como

$$\tilde{\mathbf{x}}_1(n) = \mathbf{G}^{\mathrm{T}}\mathbf{x}_{\mathrm{e}}(n) \tag{3.47}$$

resultando na seguinte relação de entrada e saída:

$$\hat{y}_1(n) = \mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}\mathbf{h}_{1\mathrm{s}}. \tag{3.48}$$

Deste modo, o vetor de coeficientes equivalente é dado por

$$\mathbf{h}_{1\mathrm{i}} = \mathbf{G}\mathbf{h}_{1\mathrm{s}}. \tag{3.49}$$

Considerando, agora, o bloco de segunda ordem, o vetor de entrada é obtido através de (3.24). Assim, substituindo (3.47) em (3.24), obtém-se

$$\tilde{\mathbf{x}}_2(n) = \tilde{\mathbf{x}}_1(n) \otimes \tilde{\mathbf{x}}_1(n) = \mathbf{G}^{\mathrm{T}}\mathbf{x}_{\mathrm{e}}(n) \otimes \mathbf{G}^{\mathrm{T}}\mathbf{x}_{\mathrm{e}}(n)\,. \tag{3.50}$$

A partir da propriedade do produto de Kronecker cruzado [6], tem-se

$$\mathbf{AC} \otimes \mathbf{BD} = (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) \tag{3.51}$$

onde $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ e $\mathbf{D}$ representam matrizes genéricas. Então, considerando (3.51), é possível reescrever (3.50) como

$$\tilde{\mathbf{x}}_2(n) = \left(\mathbf{G}^{\mathrm{T}} \otimes \mathbf{G}^{\mathrm{T}}\right)\left(\mathbf{x}_{\mathrm{e}}(n) \otimes \mathbf{x}_{\mathrm{e}}(n)\right) = \mathbf{G}_2^{\mathrm{T}}\mathbf{x}_{2\mathrm{e}}(n) \tag{3.52}$$

sendo $\mathbf{x}_{2\mathrm{e}}(n) = \mathbf{x}_{\mathrm{e}}(n) \otimes \mathbf{x}_{\mathrm{e}}(n)$ e $\mathbf{G}_2 = \mathbf{G} \otimes \mathbf{G}$. Dessa forma, a saída do bloco de segunda ordem pode agora ser reescrita em função do vetor $\mathbf{x}_{\mathrm{e}}(n)$, resultando em

$$\hat{y}_2(n) = \mathbf{x}_{2\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}_2\mathbf{h}_{2\mathrm{s}}. \tag{3.53}$$

De maneira similar a (3.40), o vetor equivalente de segunda ordem é dado por

$$\mathbf{h}_{2\mathrm{i}} = \mathbf{G}_2\mathbf{h}_{2\mathrm{s}}. \tag{3.54}$$

Por exemplo, com $L=2,\ N=3$ e usando o interpolador linear $\mathbf{g}=[0{,}5\ \ 1\ \ 0{,}5]^{\mathrm{T}}$, o vetor de coeficientes equivalente de segunda ordem (3.54) escrito na forma matricial é

$$\mathbf{H}_{2\mathrm{i}}=\begin{Bmatrix} \underline{0{,}25h_2(0,0)} & \underline{0{,}5h_2(0,0)} & \underline{0{,}25h_2(0,0)+0{,}25h_2(0,2)} & \underline{0{,}5h_2(0,2)} & \underline{0{,}25h_2(0,2)} \\ \underline{0{,}5h_2(0,0)} & h_2(0,0) & \boxed{0{,}5h_2(0,0)+0{,}5h_2(0,2)} & h_2(0,2) & \underline{0{,}5h_2(0,2)} \\ \underline{\boxed{0{,}25h_2(0,0)+0{,}25h_2(2,0)}} & \boxed{0{,}5h_2(0,0)+0{,}5h_2(2,0)} & \boxed{0{,}25h_2(0,0)+0{,}25h_2(0,2)+0{,}25h_2(2,2)+0{,}25h_2(2,0)} & \boxed{0{,}5h_2(0,2)+0{,}5h_2(2,2)} & \underline{\boxed{0{,}25h_2(0,2)+0{,}25h_2(2,2)}} \\ \underline{0{,}5h_2(2,0)} & h_2(2,0) & \boxed{0{,}5h_2(2,0)+0{,}5h_2(2,2)} & h_2(2,2) & \underline{0{,}5h_2(2,2)} \\ \underline{0{,}25h_2(2,0)} & \underline{0{,}5h_2(2,0)} & \underline{0{,}25h_2(2,0)+0{,}25h_2(2,2)} & \underline{0{,}5h_2(2,2)} & \underline{0{,}25h_2(2,2)} \end{Bmatrix}. \tag{3.55}$$

Verifica-se, em (3.55), como os coeficientes são recriados (indicados por retângulos) através do processo de interpolação. Por exemplo, o coeficiente central de (3.55) $[H_{2\mathrm{i}}(3,3)]$ corresponde à média aritmética dos seus quatro vizinhos da versão esparsa. Também neste caso, o efeito de borda está presente, sendo esse indicado pelos coeficientes sublinhados.

Para o caso do bloco de terceira ordem, o procedimento é similar, resultando nas seguintes expressões:

a) Vetor de entrada

$$\begin{aligned}\tilde{\mathbf{x}}_3(n)&=\tilde{\mathbf{x}}_1(n)\otimes\tilde{\mathbf{x}}_2(n)\\&=[\mathbf{G}^{\mathrm{T}}\mathbf{x}_{\mathrm{e}}(n)]\otimes[\mathbf{G}_2^{\mathrm{T}}\mathbf{x}_{2\mathrm{e}}(n)]\\&=[\mathbf{G}^{\mathrm{T}}\otimes\mathbf{G}_2^{\mathrm{T}}][\mathbf{x}_{\mathrm{e}}(n)\otimes\mathbf{x}_{2\mathrm{e}}(n)]\\&=\mathbf{G}_3^{\mathrm{T}}\mathbf{x}_{3\mathrm{e}}(n)\end{aligned} \tag{3.56}$$

com $\mathbf{G}_3=\mathbf{G}\otimes\mathbf{G}_2$ e $\mathbf{x}_{3\mathrm{e}}(n)=\mathbf{x}_{\mathrm{e}}(n)\otimes\mathbf{x}_{2\mathrm{e}}(n)$.

b) Relação de entrada e saída

$$\hat{y}_3(n)=\mathbf{x}_{3\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}_3\mathbf{h}_{3\mathrm{s}}. \tag{3.57}$$

c) Vetor de coeficientes equivalente

$$\mathbf{h}_{3\mathrm{i}}=\mathbf{G}_3\mathbf{h}_{3\mathrm{s}}. \tag{3.58}$$

O processo de recriação dos coeficientes para o bloco de terceira ordem é melhor observado a partir da representação espacial do vetor de coeficientes equivalente considerando um caso com $N=3$, $L=2$ e $\mathbf{g}=[0{,}5\ \ 1\ \ 0{,}5]^{\mathrm{T}}$. Tal representação é mostrada na Figura 3.7 sem a ilustração do efeito de borda, o que resultaria em uma casca externa à estrutura cúbica do desenho. Os círculos preenchidos em preto nos vértices do cubo representam os coeficientes do bloco esparso de terceira ordem, enquanto os círculos pequenos não preenchidos indicam os coeficientes recriados por interpolação. Conforme observado a partir da Figura 3.7, o processo de recriação ocorre de maneira análoga ao

discutido para os blocos de primeira e segunda ordens. Sendo assim, para esse caso com $\mathbf{g} = [0{,}5 \;\; 1 \;\; 0{,}5]^{\mathrm{T}}$, os coeficientes recriados serão sempre médias aritméticas de seus vizinhos. Por exemplo, o coeficiente no centro do cubo resulta da média aritmética dos oito coeficientes que são seus vizinhos. Os valores dos demais coeficientes são obtidos como resultado das médias aritméticas dos valores dos seus dois ou quatro vizinhos, dependendo da sua posição na estrutura. Generalizando, para um bloco de ordem p, a relação de entrada e saída é dada por

$$\hat{y}_p(n) = \mathbf{x}_{p\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}_p\mathbf{h}_{p\mathrm{s}} \tag{3.59}$$

com

$$\mathbf{x}_{p\mathrm{e}}(n) = \mathbf{x}_{\mathrm{e}}(n) \otimes \mathbf{x}_{(p-1)\mathrm{e}}(n) \tag{3.60}$$

e

$$\mathbf{G}_p = \mathbf{G} \otimes \mathbf{G}_{p-1}. \tag{3.61}$$

Ainda, tem-se

$$\tilde{\mathbf{x}}_p(n) = \mathbf{G}_p^{\mathrm{T}}\mathbf{x}_{p\mathrm{e}}(n) \tag{3.62}$$

e

$$\mathbf{h}_{p\mathrm{i}} = \mathbf{G}_p\mathbf{h}_{p\mathrm{s}}. \tag{3.63}$$

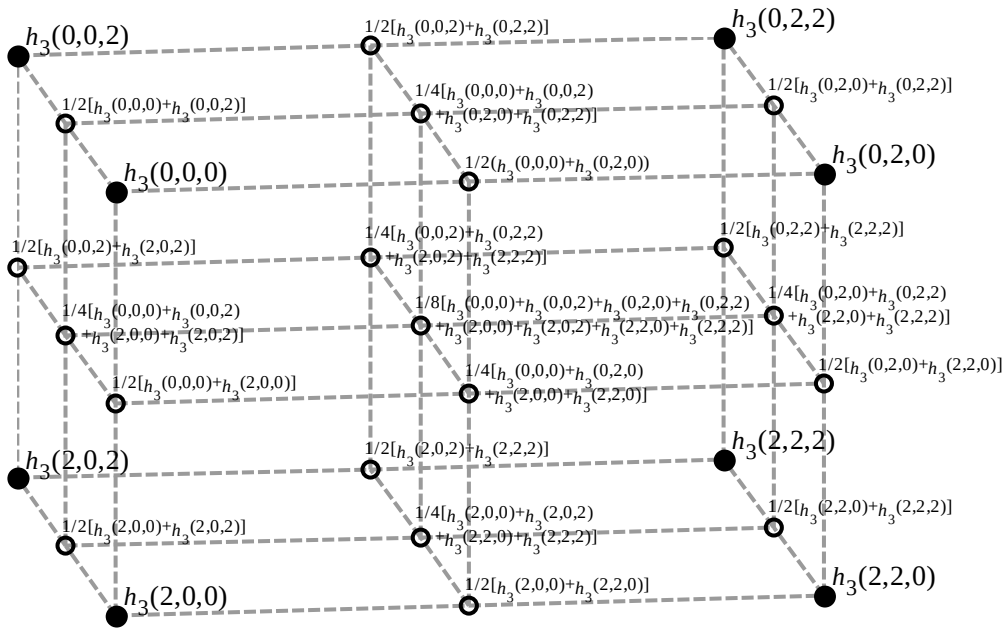


Figura 3.7. Representação espacial do processo de recriação dos coeficientes para um bloco de terceira ordem.

Finalmente, a relação de entrada e saída equivalente (em função do sinal de entrada) para o filtro Volterra interpolado é

$$\hat{y}(n) = \mathbf{x}_{\mathrm{Ve}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{Vi}} \tag{3.64}$$

com o vetor de entrada formado por

$$\mathbf{x}_{\mathrm{Ve}}(n) = [\mathbf{x}_{\mathrm{e}}^{\mathrm{T}}(n) \quad \mathbf{x}_{2\mathrm{e}}^{\mathrm{T}}(n) \quad \ldots \quad \mathbf{x}_{P\mathrm{e}}^{\mathrm{T}}(n)]^{\mathrm{T}} \tag{3.65}$$

e o vetor de coeficientes dado por

$$\mathbf{h}_{\mathrm{Vi}} = [\mathbf{h}_{1\mathrm{s}}^{\mathrm{T}}\mathbf{G}^{\mathrm{T}} \quad \mathbf{h}_{2\mathrm{s}}^{\mathrm{T}}\mathbf{G}_{2}^{\mathrm{T}} \quad \cdots \quad \mathbf{h}_{P\mathrm{s}}^{\mathrm{T}}\mathbf{G}_{P}^{\mathrm{T}}]^{\mathrm{T}}. \tag{3.66}$$

A expressão (3.64) está ilustrada na Figura 3.8 na forma de um diagrama de blocos, com o filtro Volterra interpolado apresentado no lado esquerdo e o seu equivalente no lado direito. Para o caso do filtro Volterra parcialmente interpolado, as seguintes expressões, análogas a (3.64)-(3.66), são obtidas:

$$\hat{y}(n) = \mathbf{x}_{\mathrm{VPe}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{VPi}} \tag{3.67}$$

com vetor de entrada

$$\mathbf{x}_{\mathrm{VPe}}(n) = [\mathbf{x}^{\mathrm{T}}(n) \quad \mathbf{x}_{2\mathrm{e}}^{\mathrm{T}}(n) \quad \ldots \quad \mathbf{x}_{P\mathrm{e}}^{\mathrm{T}}(n)]^{\mathrm{T}} \tag{3.68}$$

e vetor de coeficientes equivalente

$$\mathbf{h}_{\mathrm{VPi}} = \left[\mathbf{h}_{1}^{\mathrm{T}} \quad \mathbf{h}_{2\mathrm{s}}^{\mathrm{T}}\mathbf{G}_{2}^{\mathrm{T}} \quad \cdots \quad \mathbf{h}_{P\mathrm{s}}^{\mathrm{T}}\mathbf{G}_{P}^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{3.69}$$

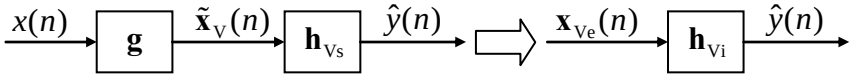


Figura 3.8. Ilustração de (3.64): filtro Volterra interpolado e sua estrutura equivalente.

3.4.3 Considerações sobre Estruturas Volterra Interpoladas com Interpolador na Saída

Conforme apresentado na Seção 3.3.1, os filtros Volterra interpolados são implementados através de uma estrutura com um interpolador de entrada seguido por um filtro Volterra esparso (Figura 3.3). Ainda, a partir dos resultados da Seção 3.4.2 verifica-se que, utilizando tal estrutura, o filtro equivalente resultante apresenta uma forma análoga à dos filtros IFIR, com o filtro esparso sendo responsável pela redução de complexidade e o interpolador com o objetivo de recriar os coeficientes zerados do filtro esparso a partir dos valores de seus vizinhos. No caso dos filtros IFIR, a troca das posições do interpolador e do filtro esparso não modifica a estrutura equivalente. No entanto, no caso dos filtros Volterra interpolados essa mudança produz uma modificação da estrutura equivalente por

envolver um filtro Volterra esparso que é não-linear. Assim, considerando uma estrutura composta por um filtro Volterra esparso seguido por um interpolador na saída, tem-se novamente o bloco equivalente de primeira ordem igual a um filtro IFIR. Com respeito ao bloco de segunda ordem, sua relação de entrada e saída é dada por

$$\hat{y}_2(n) = \sum_{k=0}^{M-1} g(k)\mathbf{h}_{2s}^{\mathrm{T}}\mathbf{x}_2(n-k) \tag{3.70}$$

onde g_k representa os coeficientes do interpolador, $\mathbf{h}_{2s}$ o vetor de coeficientes esparso de segunda ordem, $\hat{y}_2(n)$ a saída do bloco de segunda ordem e $\mathbf{x}_2(n)$ o vetor de entrada de segunda ordem, que nesse caso é composto por amostras do sinal de entrada e não por amostras interpoladas desse sinal. Analisando-se (3.70), observa-se que tal expressão corresponde a um somatório das saídas de blocos de segunda ordem esparsos, cada bloco apresentando um atraso diferente no sinal de entrada e o vetor de coeficientes multiplicado por um dos coeficientes do interpolador. Assim, para um caso com $N = 3$, $L = 2$ e $\mathbf{g} = [g(0)\ \ g(1)\ \ g(2)]^{\mathrm{T}}$, a matriz de coeficientes equivalente pode ser escrita como

$$\begin{aligned}\widehat{\mathbf{H}}_{2i} = g(0)&\begin{bmatrix} h_2(0,0) & 0 & h_2(0,2) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ h_2(2,0) & 0 & h_2(2,2) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + g(1)\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & h_2(0,0) & 0 & h_2(0,2) & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & h_2(2,0) & 0 & h_2(2,2) & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ &+ g(2)\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_2(0,0) & 0 & h_2(0,2) \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_2(2,0) & 0 & h_2(2,2) \end{bmatrix}\end{aligned} \tag{3.71}$$

o que resulta em

$$\widehat{\mathbf{H}}_{2i} = \begin{bmatrix} g(0)h_2(0,0) & 0 & g(0)h_2(0,2) & 0 & 0 \\ 0 & g(1)h_2(0,0) & 0 & g(1)h_2(0,2) & 0 \\ g(0)h_2(2,0) & 0 & g(0)h_2(2,2)+g(2)h_2(0,0) & 0 & g(2)h_2(0,2) \\ 0 & g(1)h_2(2,0) & 0 & g(1)h_2(2,2) & 0 \\ 0 & 0 & g(2)h_2(2,0) & 0 & g(2)h_2(2,2) \end{bmatrix}. \tag{3.72}$$

Nota-se, a partir de (3.72), que a estrutura equivalente a uma estrutura Volterra interpolada de segunda ordem com interpolador na saída apresenta uma característica esparsa. Essa característica também é observada nos blocos de ordens superiores, o que pode ser facilmente verificado desenvolvendo expressões similares a (3.70)-(3.72) para tais blocos.

O interesse aqui é lidar com estruturas interpoladas que não apresentem características esparsas de maneira análoga aos filtros IFIR. Portanto, as estruturas Volterra interpoladas com interpolador na saída não fazem parte desse cenário.

3.5 Remoção do Efeito de Borda em Filtros IFIR e Filtros Volterra Interpolados

Na seção anterior, foi demonstrado que as implementações interpoladas, tanto de filtros FIR lineares quanto de filtros Volterra, apresentam um indesejável efeito de borda. Apesar de tal característica perder importância com o aumento do tamanho de memória do filtro, em alguns casos ela pode afetar consideravelmente o desempenho ou até mesmo inviabilizar o uso de filtros interpolados. Um exemplo desse tipo de situação é a modelagem (usando um filtro interpolado) de uma planta cuja resposta ao impulso tem a forma de uma exponencial decrescente. Nesse caso, o filtro interpolado equivalente apresenta um erro de modelagem considerável devido à coincidência de seu primeiro coeficiente, que é resultante do efeito de borda, com o coeficiente de maior magnitude da planta. Outro problema apresentado pelas implementações interpoladas está relacionado com o seu tamanho de memória equivalente. Conforme descrito no capítulo anterior, o tamanho de memória de um filtro interpolado usado para substituir um filtro convencional com tamanho de memória N é igual a $(N+M-1)$, com M denotando o tamanho de memória do interpolador. Assim, nota-se que o filtro interpolado equivalente apresenta $M-1$ coeficientes desnecessários em aplicações para as quais o tamanho de memória requerido é N. Nesta seção, uma implementação mais eficiente e robusta de filtros interpolados [36] é desenvolvida a partir da remoção do seu efeito de borda e da conseqüente adequação do tamanho de memória equivalente.

3.5.1 Remoção do Efeito de Borda em Filtros IFIR

Conforme apresentado na Seção 3.2, um filtro IFIR com tamanho de memória $N=5$, fator de interpolação $L=2$ e interpolador $\mathbf{g}=[g(0)\ \ g(1)\ \ g(2)]^{\mathrm{T}}$, apresenta o seguinte vetor de coeficientes equivalente:

$$\mathbf{w}_{\mathrm{i}}=\mathbf{G}\mathbf{w}_{\mathrm{s}}=\Big[\underline{g(0)w(0)}\ \ g(1)w(0)\ \ \boxed{g(2)w(0)+g(0)w(2)}\ \ g(1)w(2)\ \ \boxed{g(2)w(2)+g(0)w(4)}\ \ g(1)w(4)\ \ \underline{g(2)w(4)}\Big]^{\mathrm{T}}. \tag{3.73}$$

Em (3.73), os coeficientes resultantes do efeito de borda estão sublinhados. Com a remoção de tal efeito, o que se deseja é que o filtro obtido apresente o seguinte vetor de coeficientes:

$$\mathbf{w}_{\mathrm{i}}' = \left[g(1)w(0) \;\; \underline{g(2)w(0)+g(0)w(2)} \;\; g(1)w(2) \;\; \underline{g(2)w(2)+g(0)w(4)} \;\; g(1)w(4) \right]^{\mathrm{T}}. \quad (3.74)$$

Para obter (3.74), determina-se a matriz de transformação $\mathbf{T}$ que satisfaz a seguinte relação:

$$\mathbf{w}_{\mathrm{i}}' = \mathbf{T}\mathbf{w}_{\mathrm{i}}. \quad (3.75)$$

Considerando as dimensões dos vetores envolvidos no exemplo (5×1 para $\mathbf{w}_{\mathrm{i}}'$ e 7×1 para $\mathbf{w}_{\mathrm{i}}$) e a natureza da transformação que se deseja realizar, a matriz $\mathbf{T}$ para esse caso é dada por

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \quad (3.76)$$

A matriz $\mathbf{T}$ pode ser generalizada para outros valores de N e L levando em conta que os $L-1$ primeiros e os $L-1$ últimos coeficientes do vetor equivalente $\mathbf{w}_{\mathrm{i}}$ correspondem ao efeito de borda, o que pode ser observado, por exemplo, em (3.17) e (3.73). Dessa forma, tem-se a seguinte matriz de transformação generalizada com dimensão $N\times(N+M-1)$:

$$\mathbf{T} = \begin{bmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \vdots & \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ \underbrace{0 \;\; \cdots \;\; 0}_{L-1 \text{ colunas}} & & & \underbrace{0 \;\; 0 \;\; \cdots \;\; 1}_{N \text{ colunas}} & & & & \underbrace{0 \;\; \cdots \;\; 0}_{L-1 \text{ colunas}} & & \end{bmatrix}. \quad (3.77)$$

Agora, substituindo (3.42) em (3.75), obtém-se

$$\mathbf{w}_{\mathrm{i}}' = \mathbf{T}\mathbf{G}\mathbf{w}_{\mathrm{s}} = \mathbf{T}\mathbf{W}_{\mathrm{s}}\mathbf{g}. \quad (3.78)$$

Nota-se, a partir de (3.78), que para remover o efeito de borda no filtro IFIR, a matriz de interpolação deve ser substituída por uma outra matriz de interpolação $\mathbf{G}'$ definida como

$$\mathbf{G}' = \mathbf{T}\mathbf{G} \quad (3.79)$$

ou, alternativamente, a matriz $\mathbf{W}_{\mathrm{s}}$ deve ser substituída por

$$\mathbf{W}_{\mathrm{s}}' = \mathbf{T}\mathbf{W}_{\mathrm{s}}. \quad (3.80)$$

Considerando o filtro IFIR com interpolador na entrada (Figura 3.1), o uso da matriz de interpolação modificada $\mathbf{G}'$ resulta em um vetor de entrada do filtro esparso, agora sem efeito de borda, dado por

$$\tilde{\mathbf{x}}'(n) = \mathbf{G}'^{\mathrm{T}}\mathbf{x}(n) = \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) \tag{3.81}$$

com

$$\mathbf{x}(n) = \left[x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-N+1)\right]^{\mathrm{T}}. \tag{3.82}$$

Para o caso do filtro IFIR com interpolador na saída (Figura 3.2), o uso de $\mathbf{W}_{\mathrm{s}}'$ resulta no seguinte vetor de entrada para o filtro interpolador de saída:

$$\hat{\mathbf{x}}'(n) = \mathbf{W}_{\mathrm{s}}'^{\mathrm{T}}\mathbf{x}(n) = \mathbf{W}_{\mathrm{s}}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n). \tag{3.83}$$

Assim, para remover o efeito de borda do filtro IFIR, basta substituir (3.6) por (3.81) no caso do filtro IFIR com interpolador de entrada, ou (3.10) por (3.83) no filtro IFIR com interpolador na saída. O custo computacional para implementar tal procedimento é pequeno. Por exemplo, considerando um filtro IFIR com interpolador na entrada, $L = 2$, $M = 3$, $N = 5$ e $\mathbf{g} = [g(0) \;\; g(1) \;\; g(2)]^{\mathrm{T}}$, o vetor de entrada modificado é dado por

$$\tilde{\mathbf{x}}'(n) = \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) = \begin{bmatrix} g(1)x(n) + g(2)x(n-1) \\ \mathbf{g}^{\mathrm{T}}\mathbf{x}_{\mathrm{m}}(n) \\ \mathbf{g}^{\mathrm{T}}\mathbf{x}_{\mathrm{m}}(n-1) \\ \mathbf{g}^{\mathrm{T}}\mathbf{x}_{\mathrm{m}}(n-2) \\ g(0)x(n-3) + g(1)x(n-4) \end{bmatrix}^{\mathrm{T}} \tag{3.84}$$

com

$$\mathbf{x}_{\mathrm{m}}(n) = [x(n) \;\; x(n-1) \;\; \cdots \;\; x(n-M+1)]^{\mathrm{T}}. \tag{3.85}$$

Para se obter (3.84), a cada iteração são necessárias as seguintes operações:

i) duas multiplicações e uma soma para obter o primeiro elemento para a iteração atual;
ii) uma multiplicação e uma soma para obter o segundo elemento da iteração atual a partir do primeiro elemento da iteração anterior;
iii) uma multiplicação e uma subtração para obter o último elemento da iteração atual a partir do penúltimo elemento da iteração anterior.

Portanto, no total, são necessárias 4 multiplicações e 3 somas para se determinar (3.84), enquanto no caso do filtro IFIR convencional são necessárias 3 multiplicações e 2 somas. Generalizando, tem-se um total de $M + L - 1$ multiplicações e $M + L - 2$ somas para a filtragem modificada do filtro IFIR sem efeito de borda, contra M multiplicações e $M - 1$

somas para a filtragem padrão do filtro IFIR. Assim, observa-se um incremento de apenas $2L-2$ operações por amostra. Como o valor de L é geralmente pequeno, tal incremento é praticamente desprezível. Para o caso do filtro IFIR com interpolador na saída, realiza-se a implementação de $\hat{\mathbf{x}}'(n) = \mathbf{W}_s^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)$ de forma similar à descrita para $\tilde{\mathbf{x}}'(n) = \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)$, resultando também em um acréscimo de apenas $2L-2$ operações por amostra.

3.5.2 Filtros Volterra Interpolados sem Efeito de Borda

O procedimento de remoção do efeito de borda para os filtros Volterra interpolados é similar ao adotado para os filtros IFIR com interpolador na entrada: a substituição do vetor de entrada por (3.81). Assim, para o bloco de primeira ordem os resultados são idênticos aos da seção anterior. Considerando o bloco de segunda ordem, o vetor de entrada do filtro esparso é obtido da seguinte maneira:

$$\begin{aligned}\tilde{\mathbf{x}}_2'(n) &= \tilde{\mathbf{x}}'(n) \otimes \tilde{\mathbf{x}}'(n) \\ &= \mathbf{G}'^{\mathrm{T}}\mathbf{x}(n) \otimes \mathbf{G}'^{\mathrm{T}}\mathbf{x}(n) \\ &= \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) \otimes \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n).\end{aligned} \tag{3.86}$$

Aplicando a propriedade do produto de Kronecker cruzado [6] em (3.86), obtém-se

$$\begin{aligned}\tilde{\mathbf{x}}_2'(n) &= \left(\mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}} \otimes \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\right) \\ &= \left[\left(\mathbf{G}^{\mathrm{T}} \otimes \mathbf{G}^{\mathrm{T}}\right)\left(\mathbf{T}^{\mathrm{T}} \otimes \mathbf{T}^{\mathrm{T}}\right)\right]\left[\mathbf{x}(n) \otimes \mathbf{x}(n)\right]\end{aligned} \tag{3.87}$$

o que resulta em

$$\tilde{\mathbf{x}}_2'(n) = \mathbf{G}_2^{\mathrm{T}}\mathbf{T}_2^{\mathrm{T}}\mathbf{x}_2(n) \tag{3.88}$$

com $\mathbf{T}_2 = \mathbf{T} \otimes \mathbf{T}$, $\mathbf{G}_2 = \mathbf{G} \otimes \mathbf{G}$ e $\mathbf{x}_2(n)$ como o vetor de entrada de segunda ordem do filtro Volterra de (2.8). Considerando (3.88), temos a relação de entrada e saída do bloco de segunda ordem dada por

$$\hat{y}_2'(n) = \mathbf{x}_2^{\mathrm{T}}(n)\mathbf{T}_2\mathbf{G}_2\mathbf{h}_{2s}. \tag{3.89}$$

Assim, o vetor de coeficientes equivalente do bloco de segunda ordem é dado por

$$\mathbf{h}_{2i}' = \mathbf{T}_2\mathbf{G}_2\mathbf{h}_{2s} = \mathbf{G}_2'\mathbf{h}_{2s} \tag{3.90}$$

com $\mathbf{G}_2' = \mathbf{G}' \otimes \mathbf{G}' = \mathbf{T}_2\mathbf{G}_2$. Calculando (3.90) para o caso de um filtro Volterra interpolado com $N=3$, $L=2$, $\mathbf{g} = [0{,}5 \;\; 1 \;\; 0{,}5]^{\mathrm{T}}$ e escrevendo o vetor resultante na forma matricial, obtém-se

$$\mathbf{H}'_{2\mathrm{i}} = \begin{Bmatrix} h_2(0,0) & 0,5h_2(0,0)+0,5h_2(0,2) & h_2(0,2) \\ \begin{bmatrix} 0,5h_2(0,0)+ \\ 0,5h_2(2,0) \end{bmatrix} & \begin{bmatrix} 0,25h_2(0,0)+0,25h_2(0,2)+ \\ 0,25h_2(2,0)+0,25h_2(2,2) \end{bmatrix} & \begin{bmatrix} 0,5h_2(0,2)+ \\ 0,5h_2(2,2) \end{bmatrix} \\ h_2(2,0) & 0,5h_2(2,0)+0,5h_2(2,2) & h_2(2,2) \end{Bmatrix}. \tag{3.91}$$

Nota-se agora, comparando (3.91) com (3.55), que o efeito de borda foi removido e o tamanho de memória equivalente é $N = 3$. Para o bloco de terceira ordem, temos o seguinte vetor de entrada do filtro esparso

$$\begin{aligned} \tilde{\mathbf{x}}'_3(n) &= \tilde{\mathbf{x}}'(n) \otimes \tilde{\mathbf{x}}'_2(n) \\ &= \mathbf{G}'^{\mathrm{T}}\mathbf{x}(n) \otimes \mathbf{G}'^{\mathrm{T}}_2\mathbf{x}(n) \\ &= \mathbf{G}^{\mathrm{T}}\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) \otimes \mathbf{G}^{\mathrm{T}}_2\mathbf{T}^{\mathrm{T}}_2\mathbf{x}(n) \end{aligned} \tag{3.92}$$

resultando em

$$\tilde{\mathbf{x}}'_3(n) = \mathbf{G}^{\mathrm{T}}_3\mathbf{T}^{\mathrm{T}}_3\mathbf{x}_3(n) = \mathbf{G}'^{\mathrm{T}}_3\mathbf{x}_3(n) \tag{3.93}$$

com $\mathbf{T}_3 = \mathbf{T} \otimes \mathbf{T}_2$ e $\mathbf{G}'_3 = \mathbf{T}_3\mathbf{G}_3$. Assim, de maneira similar ao bloco de segunda ordem, o vetor de coeficientes equivalente de terceira ordem sem efeito de borda é dado por

$$\mathbf{h}'_{3\mathrm{i}} = \mathbf{T}_3\mathbf{G}_3\mathbf{h}_{3\mathrm{s}} = \mathbf{G}'_3\mathbf{h}_{3\mathrm{s}}. \tag{3.94}$$

Calculando (3.94) para $N = 3$, $L = 2$ e $\mathbf{g} = [0,5 \;\; 1 \;\; 0,5]^{\mathrm{T}}$, obtém-se um vetor de coeficientes de terceira ordem cuja representação espacial é dada pela Figura 3.7. Note que tal figura é apresentada na Seção 3.4.2 para ilustrar o processo de recriação dos coeficientes sem ilustrar o efeito de borda. Agora, a representação apresentada na Figura 3.7 é exatamente a representação dos coeficientes obtidos com (3.94). De maneira geral, tem-se

$$\tilde{\mathbf{x}}'_p(n) = \mathbf{G}^{\mathrm{T}}_p\mathbf{T}^{\mathrm{T}}_p\mathbf{x}_p(n) = \mathbf{G}'^{\mathrm{T}}_p\mathbf{x}_p(n) \tag{3.95}$$

e

$$\mathbf{h}'_{p\mathrm{i}} = \mathbf{T}_p\mathbf{G}_p\mathbf{h}_{p\mathrm{s}} = \mathbf{G}'_p\mathbf{h}_{p\mathrm{s}} \tag{3.96}$$

com

$$\mathbf{T}_p = \mathbf{T} \otimes \mathbf{T}_{p-1} \tag{3.97}$$

$$\mathbf{G}'_p = \mathbf{T}_p\mathbf{G}_p \tag{3.98}$$

e $\mathbf{G}_p$ dado por (3.61). Assim, para um bloco de ordem qualquer do filtro Volterra interpolado, a substituição do vetor de entrada interpolado (3.30) pelo vetor de entrada interpolado modificado (3.95) produz uma estrutura equivalente sem efeito de borda. Também é importante ressaltar que os vetores de entrada modificados obtidos a partir de

(3.95) são os da representação convencional do filtro Volterra, enquanto para fins de implementação a representação triangular equivalente é utilizada devido ao seu menor custo em termos de complexidade computacional.

3.5.3 Simulações

Nesta seção, alguns resultados são apresentados para verificar o desempenho dos filtros interpolados usando o procedimento de remoção do efeito de borda. Os exemplos apresentados são problemas de identificação de sistemas [1]-[2] de segunda e terceira ordens utilizando os seguintes tipos de filtros adaptativos: filtro FIR linear, filtro Volterra convencional, e implementações adaptativas dos filtros Volterra interpolados e interpolados com a remoção do efeito de borda. Nas simulações considerando filtros de segunda ordem, gráficos dos valores dos coeficientes são apresentados com o objetivo de ilustrar a estrutura resultante e o efeito do procedimento de remoção do efeito de borda. É importante destacar que as diferentes implementações do filtro Volterra são realizadas usando a representação triangular, enquanto a apresentação das superfícies de segunda ordem é realizada usando a representação convencional. Não são apresentadas simulações com filtros IFIR uma vez que eles são similares aos blocos de primeira ordem dos filtros Volterra. O passo de adaptação adotado nos exemplos quando se utiliza o algoritmo LMS é $\mu = \mu_{\max}/2$, com $\mu_{\max}$ obtido a partir de (2.68). É importante ressaltar que, no caso das implementações interpoladas, o cálculo da norma quadrática do vetor de entrada necessário para computar (2.68) é realizado considerando apenas os elementos de tal vetor cujos coeficientes correspondentes são diferentes de zero.

Exemplo 3.5.3.1: Neste exemplo, o sinal de entrada é branco e gaussiano com variância $\sigma_x^2 = 1$ e o ruído aditivo tem variância $\sigma_z^2 = 10^{-4}$. O algoritmo adaptativo utilizado é o LMS, e os coeficientes da planta estão ilustrados na Figura 3.9. Observa-se que a planta possui, tanto para o bloco de primeira ordem quanto para o bloco de segunda ordem, uma resposta na forma exponencial decrescente. O tamanho de memória dos filtros adaptativos é $N = 11$, resultando em 11 coeficientes para o filtro linear, 77 para o filtro Volterra, 27 para o filtro Volterra interpolado e para sua implementação sem efeito de borda, e 32 para o filtro Volterra parcialmente interpolado e para sua implementação sem efeito de borda.

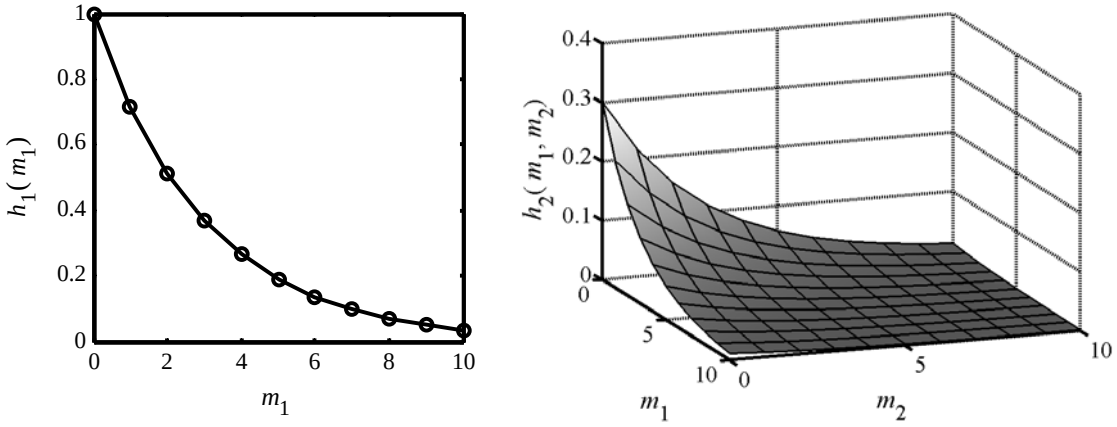


Figura 3.9. Planta para o Exemplo *3.5.3.1*.

Os resultados do erro quadrático médio (EQM) obtidos para os diferentes filtros estão apresentados na Figura 3.10. Dessa figura é possível notar que a remoção do efeito de borda tem grande impacto no desempenho dos filtros, uma vez que as implementações interpoladas sem efeito de borda apresentam desempenho muito superior ao das implementações interpoladas convencionais. A causa dessa diferença de desempenho fica evidente a partir das Figuras 3.11-3.14. Nas duas primeiras estão mostrados os coeficientes do filtro Volterra interpolado equivalente superpostos aos coeficientes da planta, enquanto nas outras duas, uma apresentação similar às primeiras é realizada para os coeficientes do filtro Volterra interpolado sem efeito de borda. A partir dessas figuras, nota-se, claramente, que a implementação sem efeito de borda é capaz de modelar com mais precisão a planta considerada neste exemplo. Além disso, é importante observar, a partir das curvas de tais figuras, que o filtro Volterra interpolado apresenta tamanho de memória maior do que a da planta, enquanto sua implementação com remoção do efeito de borda apresenta tamanho de memória igual ao da planta.

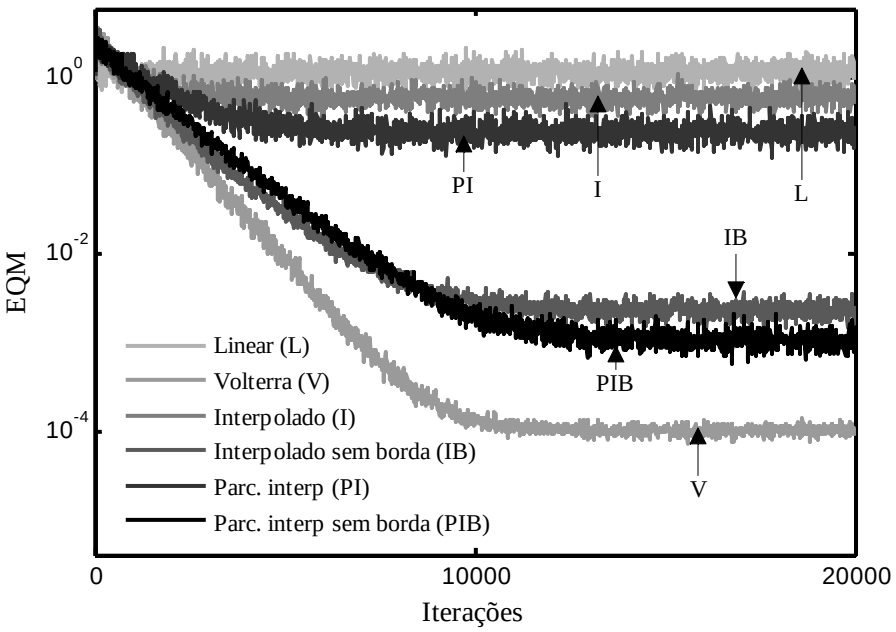


Figura 3.10. Curvas de erro quadrático médio obtidas para o Exemplo *3.5.3.1* (média de 200 realizações).

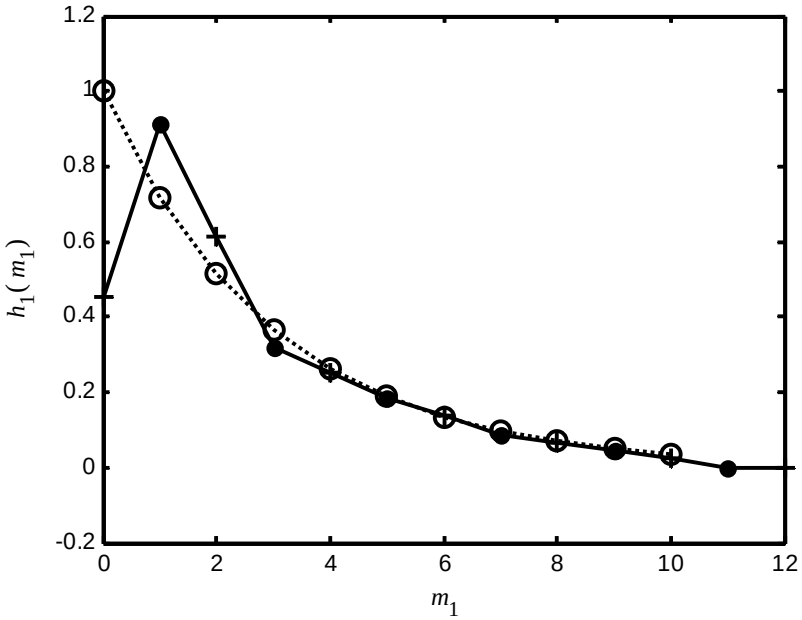


Figura 3.11. Exemplo *3.5.3.1*. Superposição de curvas de coeficientes de primeira ordem da planta (linha tracejada com marcadores circulares) com a curva do filtro Volterra interpolado (linha sólida com coeficientes indicados por marcadores circulares em preto e coeficientes recriados indicados por +).

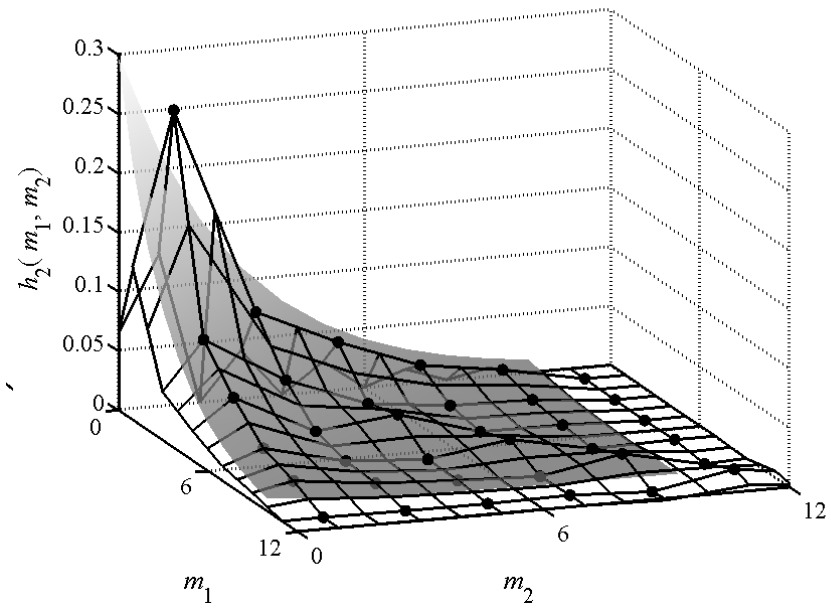


Figura 3.12. Exemplo *3.5.3.1*. Superposição das superfícies de coeficientes de segunda ordem da planta (superfície sólida) com a do filtro Volterra interpolado (grade com coeficientes do filtro esparso indicados por marcadores circulares em preto e com os demais vértices indicando os coeficientes recriados por interpolação).

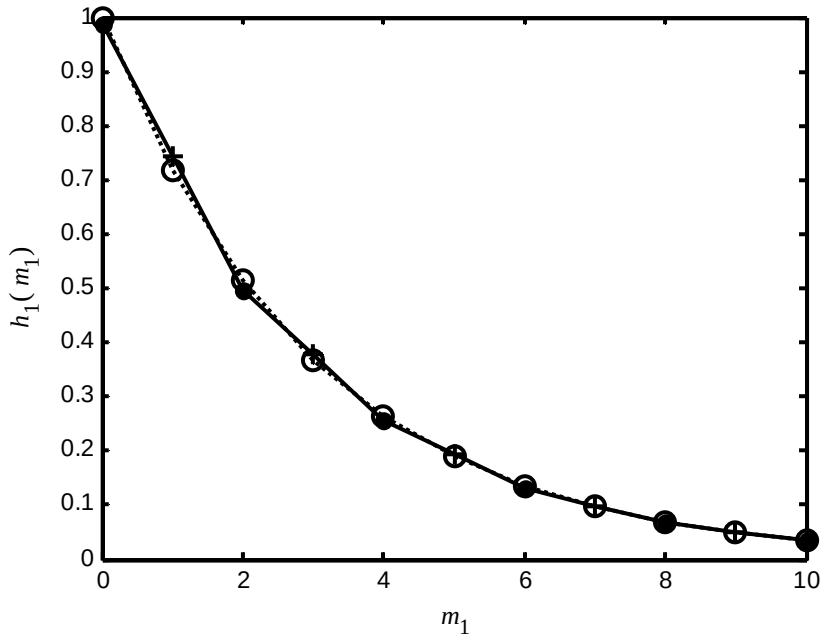


Figura 3.13. Exemplo *3.5.3.1*. Superposição de curvas de coeficientes de primeira ordem da planta (linha tracejada com marcadores circulares) com a curva do filtro Volterra interpolado sem efeito de borda (linha sólida com coeficientes indicados por marcadores circulares em preto e coeficientes recriados indicados por +).

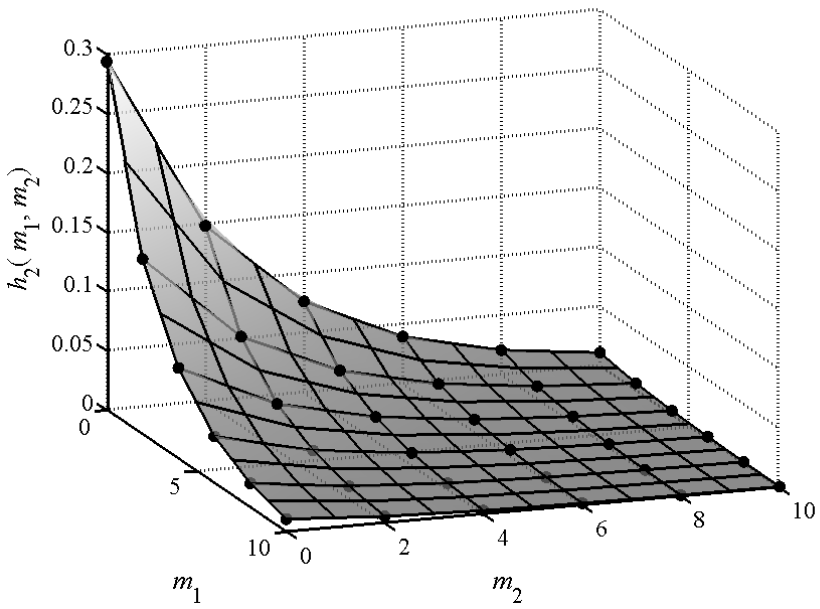


Figura 3.14. Exemplo *3.5.3.1*. Superposição das superfícies de coeficientes de segunda ordem da planta (superfície sólida) com a do filtro Volterra interpolado sem efeito de borda (grade com coeficientes do filtro esparso indicados por marcadores circulares em preto e com os demais vértices indicando os coeficientes recriados por interpolação).

Exemplo 3.5.3.2: Neste exemplo, os dados utilizados são os mesmos do Exemplo 4.1, com diferença apenas na planta, cujos coeficientes são ilustrados na Figura 3.15. Nota-se que agora a planta possui coeficientes iguais a zero na região de borda.

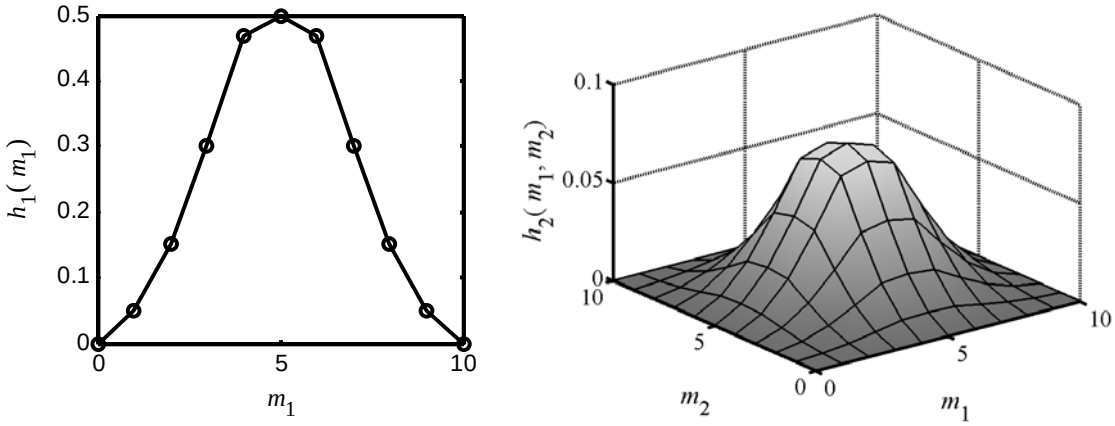


Figura 3.15. Planta para o Exemplo *3.5.3.2*.

As curvas do erro quadrático médio (EQM) obtidas para este exemplo estão mostradas na Figura 3.16. A partir dessa figura, nota-se que o desempenho dos filtros interpolados sem efeito de borda é melhor do que aquele dos filtros interpolados, apesar da diferença não ser tão grande quanto à obtida no Exemplo 3.1. Isso se deve ao fato de que os coeficientes de borda da planta atual apresentam valores menores do que os do exemplo anterior. Os

coeficientes obtidos em regime permanente pelo filtro Volterra interpolado são ilustrados em conjunto com os coeficientes da planta nas Figuras 3.17 e 3.18. As Figuras 3.19 e 3.20 são similares a 3.17 e 3.18, porém considerando o filtro interpolado sem efeito de borda.

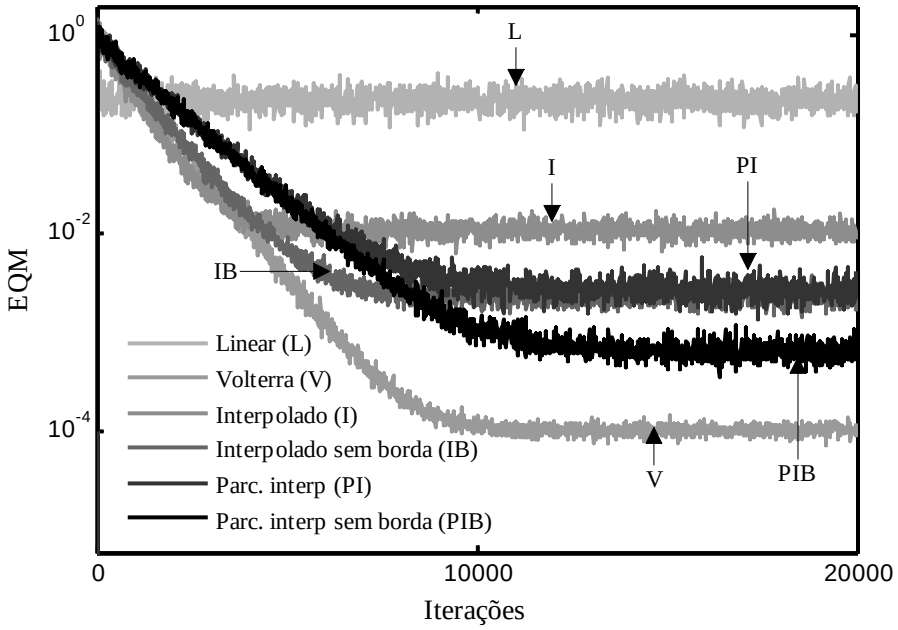


Figura 3.16. Curvas de erro quadrático médio obtidas para o Exemplo *3.5.3.2* (média de 200 realizações).

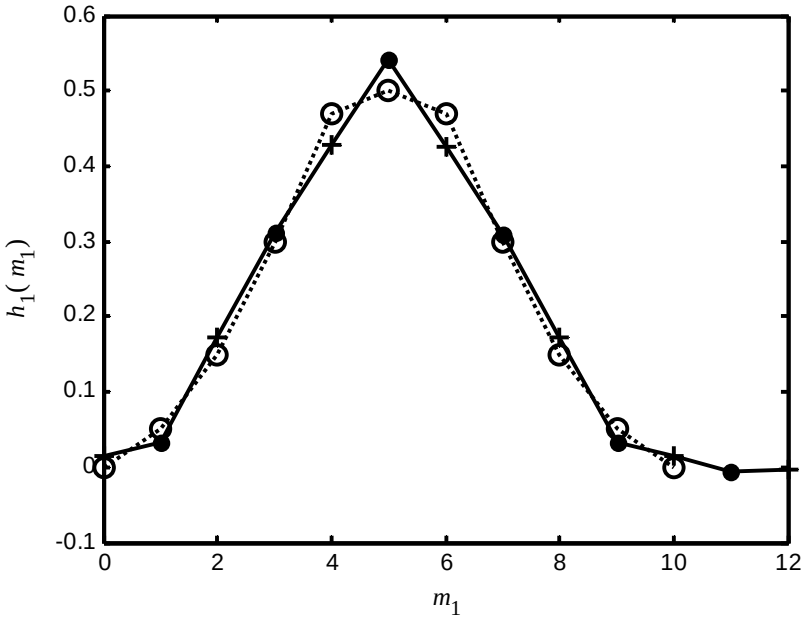


Figura 3.17. Exemplo *3.5.3.2*. Superposição de curvas de coeficientes de primeira ordem da planta (linha tracejada com marcadores circulares) com a curva do filtro Volterra interpolado (linha sólida com coeficientes indicados por marcadores circulares em preto e coeficientes recriados indicados por +).

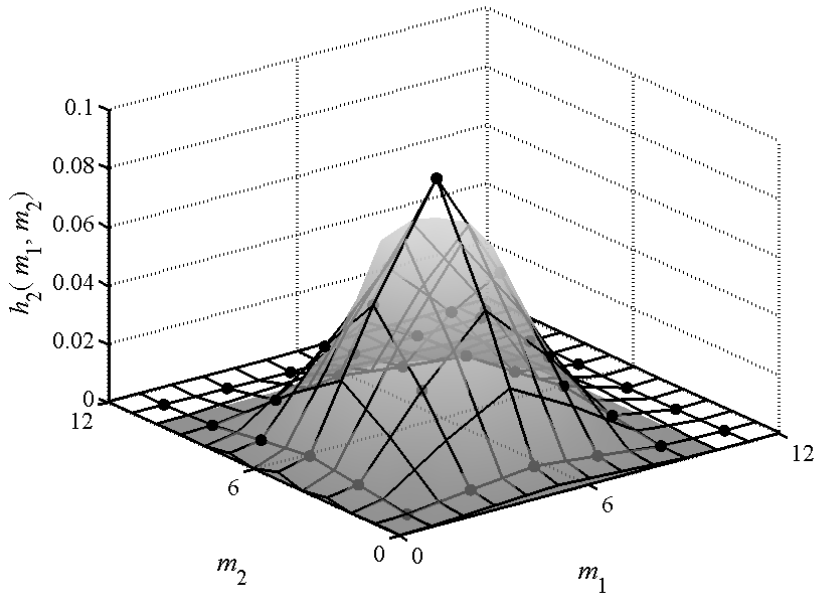


Figura 3.18. Exemplo *3.5.3.2*. Superposição das superfícies de coeficientes de segunda ordem da planta (superfície sólida) com a do filtro Volterra interpolado (grade com coeficientes do filtro esparso indicados por marcadores circulares em preto e com os demais vértices indicando os coeficientes recriados por interpolação).

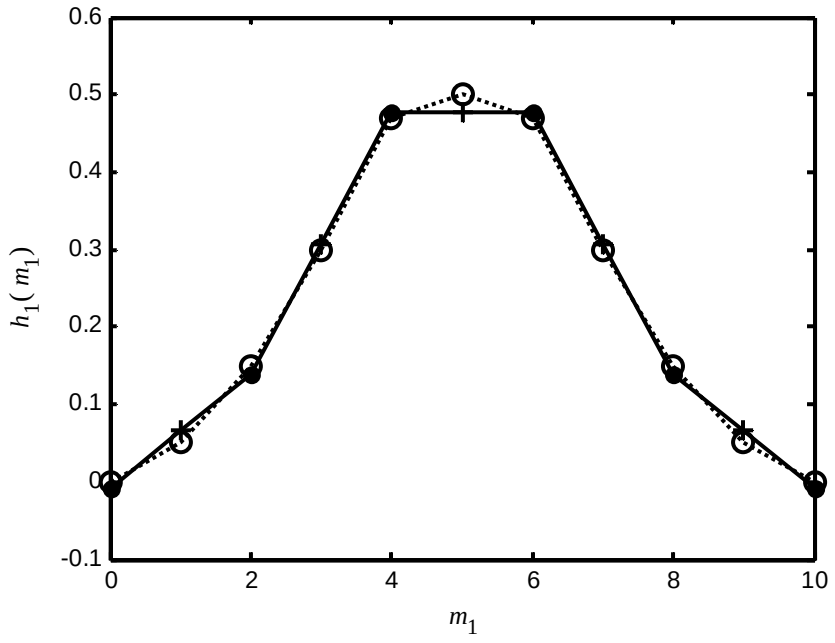


Figura 3.19. Exemplo *3.5.3.2*. Superposição de curvas de coeficientes de primeira ordem da planta (linha tracejada com marcadores circulares) com a curva do filtro Volterra interpolado sem efeito de borda (linha sólida com coeficientes indicados por marcadores circulares em preto e coeficientes recriados indicados por +).

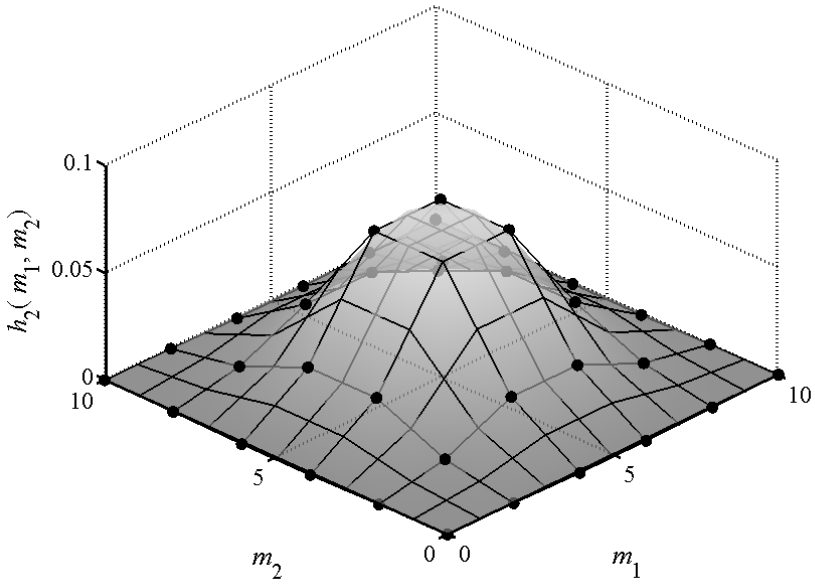
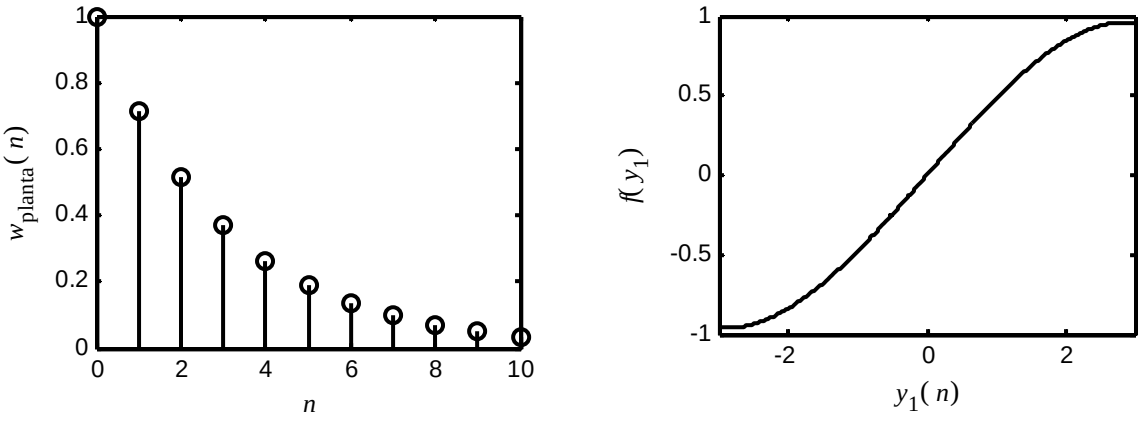


Figura 3.20. Exemplo *3.5.3.2*. Superposição das superfícies de coeficientes de segunda ordem da planta (superfície sólida) com a do filtro Volterra interpolado sem efeito de borda (grade com coeficientes do filtro esparso indicados por marcadores circulares em preto e com os demais vértices indicando os coeficientes recriados por interpolação).

Exemplo 3.5.3.3: Para este exemplo, a planta consiste de um filtro FIR seguido de uma não-linearidade sem memória, com características ilustradas na Figura 3.21. Tem-se, dessa figura, o filtro FIR com uma resposta exponencial decrescente e a não-linearidade com característica de saturação. O sinal de entrada é branco gaussiano com variância unitária, enquanto o ruído aditivo possui variância $\sigma_z^2 = 10^{-6}$. O algoritmo adaptativo é o NLMS [equação de adaptação dada por (3.32)], com $\alpha = 0.25$ e $\psi = 10^{-6}$. Todas as diferentes implementações dos filtros Volterra adaptativos utilizadas apresentam um bloco de primeira ordem (linear) e um bloco de terceira ordem devido à característica da não-linearidade da planta. Os filtros adaptativos têm tamanho de memória $N = 11$, resultando em 11 coeficientes para o filtro linear, 397 coeficientes para o filtro Volterra convencional e 67 coeficientes para o filtro Volterra parcialmente interpolado como também para o parcialmente interpolado sem efeito de borda. As curvas de erro quadrático médio obtidas para os diferentes filtros são mostradas na Figura 3.22. Nessa figura, observa-se que o filtro Volterra interpolado convencional apresenta um desempenho muito próximo ao do filtro linear, enquanto sua versão sem efeito de borda atinge um desempenho superior.



(a) (b)

Figura 3.21. Planta para o Exemplo *3.5.3.3*. (a) Resposta ao impulso da parte linear. (b) Não-linearidade sem memória.

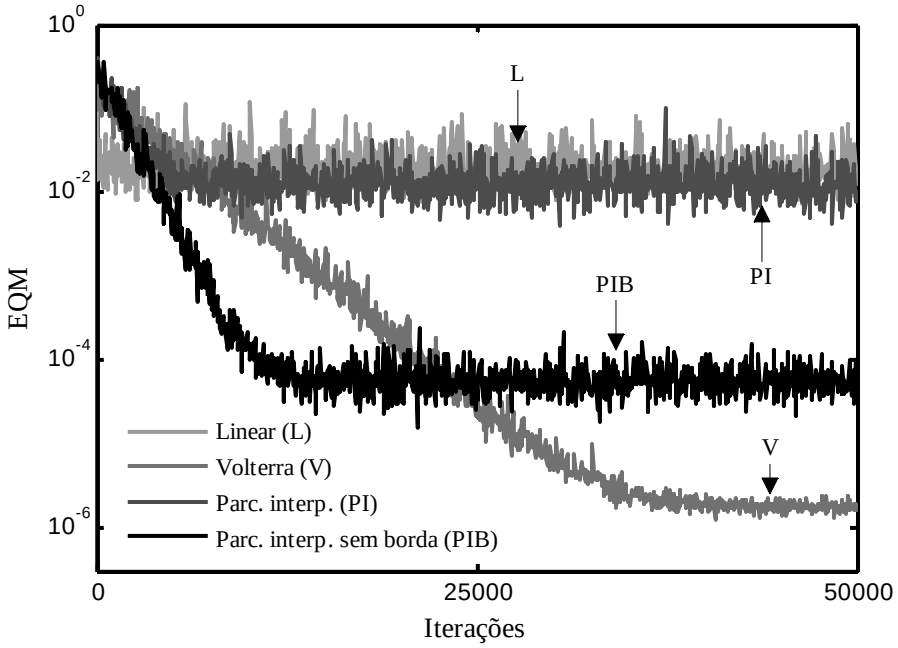


Figura 3.22. Curvas de erro quadrático médio obtidas para o Exemplo *3.5.3.3* (média de 200 realizações).

3.6 Abordagem com Restrições para Análise de Filtros Volterra Interpolados

Como descrito na Seção 2.3, uma das dificuldades para o projeto e estudo de filtros Volterra no contexto adaptativo é a falta de modelos matemáticos para determinar seu comportamento ao longo do processo adaptativo. Contribuindo nesse sentido, um modelo estatístico para o comportamento de filtros Volterra interpolados adaptados pelo algoritmo LMS é proposto nesta seção. Os resultados aqui discutidos são a evolução dos resultados apresentados em [31], podendo também serem utilizados para análise de filtros Volterra parcialmente interpolados e das suas implementações com remoção do efeito de borda.

3.6.1 Vetor de Coeficientes Ótimo e Desempenho em Regime Permanente

Uma questão importante que surge com o uso da abordagem interpolada para implementação do filtro Volterra é o impacto das aproximações envolvidas sobre o desempenho em regime permanente. Nesse contexto, expressões para o vetor de coeficientes ótimo do filtro esparso e para o valor mínimo do erro quadrático médio em regime permanente são desenvolvidas nesta seção. O primeiro passo para realizar tal análise é considerar que o filtro Volterra esparso é um filtro Volterra apresentando alguns de seus coeficientes com valores fixados em zero. Assim, de forma similar a [56] e [58], tal condição pode ser expressa na forma de restrições da seguinte maneira:

$$\mathbf{C}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vs}} = \mathbf{f} \tag{3.99}$$

onde $\mathbf{C}$ é a matriz de restrições, $\mathbf{f}$ é o vetor de reposta com todos os elementos iguais a zero e $\underline{\mathbf{h}}_{\mathrm{Vs}}$ é o vetor de coeficientes da representação triangular do filtro Volterra. Exemplificando, para o caso de um filtro Volterra interpolado com $P=1$, $N=5$ e $L=2$, o vetor de coeficientes esparso é dado por $\underline{\mathbf{h}}_{\mathrm{Vs}} = [h_1(0)\ \ 0\ \ h_1(2)\ \ 0\ \ h_1(4)]^{\mathrm{T}}$, a matriz de restrições por

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}} \tag{3.100}$$

e o vetor de resposta por $\mathbf{f} = [0\ \ 0]^{\mathrm{T}}$. Dessa forma, considera-se que os coeficientes com valores fixados em zero do vetor de coeficientes esparso são os que sofrem as restrições descritas por (3.99). Note que o vetor de coeficientes utilizado para esta análise é o da representação triangular. Isso se deve ao fato de que tal representação é a utilizada para

fins de implementação e também por que o uso da representação convencional leva a um problema sem solução trivial conforme descrito na Seção 2.3.1. Tem-se, então, um problema de minimização do erro quadrático médio $\xi(n) = E[e^2(n)]$ observando as restrições descritas por (3.99). Considerando o filtro Volterra interpolado aplicado ao problema de estimação de um sinal (Figura 2.7) e também (3.20) usando a representação triangular, o erro instantâneo pode ser escrito como

$$e(n) = d(n) - y(n) = d(n) - \underline{\mathbf{h}}_{\mathrm{Vs}}^{\mathrm{T}} \underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n). \tag{3.101}$$

Elevando (3.101) ao quadrado e tomando o valor esperado da expressão resultante, a seguinte expressão para o erro quadrático médio é obtida:

$$\xi(n) = E[e^2(n)] = E[d^2(n)] - 2\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}^{\mathrm{T}} \underline{\mathbf{h}}_{\mathrm{Vs}} + \underline{\mathbf{h}}_{\mathrm{Vs}}^{\mathrm{T}} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}} \underline{\mathbf{h}}_{\mathrm{Vs}} \tag{3.102}$$

com

$$\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}} = E[\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) \underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)] \tag{3.103}$$

e

$$\underline{\tilde{\mathbf{p}}}_{\mathrm{V}} = E[d(n) \underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)]. \tag{3.104}$$

Para incluir as restrições descritas por (3.99) na função custo (3.102), o método dos multiplicadores de Lagrange [58] é utilizado. Assim, adicionando o vetor de multiplicadores de Lagrange $\boldsymbol{\theta}$ a (3.102), obtém-se

$$J = E[d^2(n)] - 2\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}^{\mathrm{T}} \underline{\mathbf{h}}_{\mathrm{Vs}} + \underline{\mathbf{h}}_{\mathrm{Vs}}^{\mathrm{T}} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}} \underline{\mathbf{h}}_{\mathrm{Vs}} + \boldsymbol{\theta}^{\mathrm{T}} (\mathbf{C}^{\mathrm{T}} \underline{\mathbf{h}}_{\mathrm{Vs}} - \mathbf{f}). \tag{3.105}$$

Para obter o valor de $\underline{\mathbf{h}}_{\mathrm{Vs}}$ que minimiza (3.105), faz-se $\nabla_{\underline{\mathbf{h}}_{\mathrm{V}}} J = 0$. Com isso, o vetor de coeficientes ótimo esparso em função de $\boldsymbol{\theta}$ é dado por

$$\underline{\mathbf{h}}_{\mathrm{Vso}} = \frac{1}{2} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} (2\underline{\tilde{\mathbf{p}}}_{\mathrm{V}} - \mathbf{C}\boldsymbol{\theta}). \tag{3.106}$$

Como (3.106) também deve satisfazer (3.99), é possível escrever

$$\mathbf{C}^{\mathrm{T}} \underline{\mathbf{h}}_{\mathrm{Vso}} = \mathbf{f} = \frac{1}{2} \mathbf{C}^{\mathrm{T}} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} (2\underline{\tilde{\mathbf{p}}}_{\mathrm{V}} - \mathbf{C}\boldsymbol{\theta}). \tag{3.107}$$

Agora, isolando $\boldsymbol{\theta}$ e considerando que $\mathbf{f}$ é um vetor de zeros, obtém-se

$$\boldsymbol{\theta} = 2\left[\mathbf{C}^{\mathrm{T}} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} \mathbf{C}\right]^{-1} \mathbf{C}^{\mathrm{T}} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} \underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.108}$$

Finalmente, substituindo (3.108) em (3.106), a seguinte expressão para o vetor de coeficientes ótimo é obtida:

$$\underline{\mathbf{h}}_{\mathrm{Vso}} = \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} \underline{\tilde{\mathbf{p}}}_{\mathrm{V}} - \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} \mathbf{C}\left(\mathbf{C}^{\mathrm{T}} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} \mathbf{C}\right)^{-1} \mathbf{C}^{\mathrm{T}} \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}^{-1} \underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.109}$$

Uma vez obtido o do vetor de coeficientes ótimo, o valor mínimo do erro quadrático médio é determinado substituindo tal vetor em (3.102). Assim,

$$\xi_{\min} = E[d^2(n)] - 2\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vso}} + \underline{\mathbf{h}}_{\mathrm{Vso}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\underline{\mathbf{h}}_{\mathrm{Vso}}. \tag{3.110}$$

3.6.2 Algoritmo LMS

Conforme descrito na Seção 2.3.1, a idéia por trás do algoritmo LMS é minimizar a erro quadrático médio (função custo) a partir de sua estimativa instantânea. Assim, considerando agora a função custo com restrições, dada por (3.105), os coeficientes são adaptados no sentido contrário ao seu gradiente, o que resulta em

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \underline{\mathbf{h}}_{\mathrm{Vs}}(n) - \mu\nabla_{\underline{\mathbf{h}}_{\mathrm{Vs}}} J. \tag{3.111}$$

O valor instantâneo de (3.105) é dado por

$$\begin{aligned} J(n) = {} & d^2(n) - 2d(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{Vs}}(n) \\ & + \underline{\mathbf{h}}_{\mathrm{Vs}}^{\mathrm{T}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{Vs}}(n) + \boldsymbol{\theta}^{\mathrm{T}}(n)[\mathbf{C}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vs}}(n) - \mathbf{f}] \end{aligned} \tag{3.112}$$

e seu gradiente por

$$\nabla_{\underline{\mathbf{h}}_{\mathrm{Vs}}} J(n) = -2d(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) + 2\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{Vs}}(n) + \mathbf{C}\boldsymbol{\theta}(n). \tag{3.113}$$

Substituindo (3.113) em (3.111) e considerando (3.101), obtém-se

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \underline{\mathbf{h}}_{\mathrm{Vs}}(n) + 2\mu e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) - \mu\mathbf{C}\boldsymbol{\theta}(n). \tag{3.114}$$

Os multiplicadores de Lagrange são obtidos impondo as restrições de (3.99) a (3.114), o que resulta em

$$\mathbf{f} = \mathbf{C}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \mathbf{C}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vs}}(n) + 2\mu\mathbf{C}^{\mathrm{T}}e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) - \mu\mathbf{C}^{\mathrm{T}}\mathbf{C}\boldsymbol{\theta}(n). \tag{3.115}$$

Como $\mathbf{C}^{\mathrm{T}}\mathbf{C} = \mathbf{I}$ (matriz identidade), $\boldsymbol{\theta}(n)$ pode ser isolado em (3.115) obtendo

$$\boldsymbol{\theta}(n) = -\frac{1}{\mu}\left[\mathbf{f} - \mathbf{C}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vs}}(n) - 2\mu\mathbf{C}^{\mathrm{T}}e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\right]. \tag{3.116}$$

Finalmente, substituindo (3.116) em (3.114), a expressão de adaptação dos coeficientes com restrições é obtida. Assim,

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \mathbf{P}\left[\underline{\mathbf{h}}_{\mathrm{Vs}}(n) + 2\mu e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\right] \tag{3.117}$$

onde

$$\mathbf{P} = \mathbf{I} - \mathbf{C}\mathbf{C}^{\mathrm{T}}. \tag{3.118}$$

Analisando a matriz $\mathbf{P}$ observa-se que ela apresenta elementos apenas na diagonal principal e que tais elementos são iguais a zero ou 1, dependendo se o correspondente coeficiente de $\underline{\mathbf{h}}_{\mathrm{Vs}}(n)$ sofre ou não restrição. Assim, (3.117) corresponde estritamente à

aplicação do algoritmo LMS apenas aos coeficientes do vetor esparso que não sofrem restrição. Essa característica pode ser melhor observada considerando o caso de um filtro Volterra interpolado de 1ª ordem com $N = 5$ e $L = 2,$ para o qual a expressão de adaptação dos coeficientes com restrições, a partir de (3.117), é dada por

$$\underline{\mathbf{h}}_{1\mathrm{s}}(n+1) = \begin{bmatrix} \underline{h}_1(0,n) \\ 0 \\ \underline{h}_1(2,n) \\ 0 \\ \underline{h}_1(4,n) \end{bmatrix} + 2\mu e(n) \begin{bmatrix} \tilde{x}(n) \\ 0 \\ \tilde{x}(n-2) \\ 0 \\ \tilde{x}(n-4) \end{bmatrix}. \tag{3.119}$$

Outras características da matriz $\mathbf{P}$ são a simetria $(\mathbf{P} = \mathbf{P}^{\mathrm{T}})$ e a idempotência $(\mathbf{P}^2 = \mathbf{PP} = \mathbf{P})$.

3.6.2.1 Comportamento Médio dos Coeficientes

Comparando a equação de adaptação do filtro Volterra interpolado (3.117) com a do filtro Volterra convencional (2.43), observa-se que a única diferença está na multiplicação do lado direito da expressão pela matriz $\mathbf{P}$. Como essa matriz é constante ao longo do processo adaptativo, resultados similares aos apresentados na Seção 2.3.2 também podem ser obtidos considerando (3.117) [31]. Assim, a expressão que descreve o comportamento médio dos coeficientes do filtro Volterra esparso é

$$E[\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1)] = \mathbf{P}(\mathbf{I} - 2\mu\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}})E[\underline{\mathbf{h}}_{\mathrm{Vs}}(n)] + 2\mu\mathbf{P}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.120}$$

Ainda, considerando o vetor de erro dos coeficientes definido como

$$\underline{\mathbf{v}}_{\mathrm{s}}(n) = \underline{\mathbf{h}}_{\mathrm{Vs}}(n) - \underline{\mathbf{h}}_{\mathrm{Vso}} \tag{3.121}$$

obtém-se

$$E[\underline{\mathbf{v}}_{\mathrm{s}}(n+1)] = \mathbf{P}(\mathbf{I} - 2\mu\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}})E[\underline{\mathbf{v}}_{\mathrm{s}}(n)]. \tag{3.122}$$

3.6.2.2 Curva de Aprendizagem e Segundo Momento

Com relação à expressão do erro quadrático médio, realizando uma análise similar a apresentada na Seção 2.3.5, o erro instantâneo pode ser reescrito como

$$e(n) = e_{\mathrm{o}}(n) - \underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) \tag{3.123}$$

com

$$e_{\mathrm{o}}(n) = d(n) + z(n) - \underline{\mathbf{h}}_{\mathrm{Vso}}^{\mathrm{T}}\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n). \tag{3.124}$$

Elevando (3.123) ao quadrado, tomando o valor esperado da expressão resultante e aplicando o princípio da ortogonalidade [2], obtém-se

$$\xi(n) = \xi_{\min} + \mathrm{tr}\left\{\underline{\mathbf{K}}_{\mathrm{s}}(n)\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\right\} \tag{3.125}$$

com $\underline{\mathbf{K}}_{\mathrm{s}}(n) = E[\underline{\mathbf{v}}_{\mathrm{s}}(n)\underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n)]$. Para se obter a curva de aprendizagem a partir de (3.125) é necessário antes obter uma expressão recursiva para $\underline{\mathbf{K}}_{\mathrm{s}}(n)$. Assim, substituindo (3.121) em (3.117), obtém-se

$$\underline{\mathbf{v}}_{\mathrm{s}}(n+1) + \underline{\mathbf{h}}_{\mathrm{Vso}} = \mathbf{P}\left[\underline{\mathbf{v}}_{\mathrm{s}}(n) + \underline{\mathbf{h}}_{\mathrm{Vso}} + 2\mu e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\right]. \tag{3.126}$$

Considerando (3.123) e, ainda, que $\mathbf{P}\underline{\mathbf{h}}_{\mathrm{Vso}} = \underline{\mathbf{h}}_{\mathrm{Vso}}$, (3.126) pode ser reescrita como

$$\underline{\mathbf{v}}_{\mathrm{s}}(n+1) = \mathbf{P}\left[\underline{\mathbf{v}}_{\mathrm{s}}(n) + 2\mu e_{\mathrm{o}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) - 2\mu\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{v}}_{\mathrm{s}}(n)\right]. \tag{3.127}$$

Multiplicando (3.127) por sua versão transposta, tomando o valor esperado da expressão resultante e levando em conta que $\mathbf{P} = \mathbf{P}^{\mathrm{T}}$, obtém-se

$$\begin{aligned}
E[\underline{\mathbf{v}}_{\mathrm{s}}(n+1)\underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n+1)] &= \mathbf{P}E[\underline{\mathbf{v}}_{\mathrm{s}}(n)\underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n)]\mathbf{P} \\
&-2\mu\mathbf{P}E[\underline{\mathbf{v}}_{\mathrm{s}}(n)\underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)]\mathbf{P} \\
&-2\mu\mathbf{P}E[\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{v}}_{\mathrm{s}}(n)\underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n)]\mathbf{P} \\
&+4\mu^2\mathbf{P}E[\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{v}}_{\mathrm{s}}(n)\underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)]\mathbf{P} \\
&+4\mu^2\mathbf{P}E[\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)e_{\mathrm{o}}(n)e_{\mathrm{o}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)]\mathbf{P} \\
&+2\mu\mathbf{P}E\{[\mathbf{I}_{\mathrm{d}} - 2\mu\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)]\underline{\mathbf{v}}_{\mathrm{s}}(n)e_{\mathrm{o}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\}\mathbf{P} \\
&+2\mu\mathbf{P}E\{\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)e_{\mathrm{o}}(n)\underline{\mathbf{v}}_{\mathrm{s}}^{\mathrm{T}}(n)[\mathbf{I}_{\mathrm{d}} - 2\mu\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)]\}\mathbf{P}.
\end{aligned} \tag{3.128}$$

Comparando (2.75) com (3.128), observa-se que esta última tem a mesma forma da primeira pré- e pós-multiplicada pela matriz $\mathbf{P}$. Assim, é possível fazer um desenvolvimento similar ao realizado para (2.75), resultando na seguinte expressão recursiva para a matriz $\underline{\mathbf{K}}_{\mathrm{s}}(n)$:

$$\underline{\mathbf{K}}_{\mathrm{s}}(n+1) = \mathbf{P}\left\{\underline{\mathbf{K}}_{\mathrm{s}}(n) - 2\mu\left[\underline{\mathbf{K}}_{\mathrm{s}}(n)\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}} + \underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\underline{\mathbf{K}}_{\mathrm{s}}(n)\right] + 4\mu^2\xi_{\min}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\right\}\mathbf{P}. \tag{3.129}$$

Conforme comentado anteriormente, a matriz $\mathbf{P}$ é idempotente, ou seja, $\mathbf{PP} = \mathbf{P}$. Dessa forma e fazendo uma pré- e uma pós-multiplicação de (3.129) por $\mathbf{P}$, conclui-se que $\mathbf{P}\underline{\mathbf{K}}_{\mathrm{s}}(n+1)\mathbf{P} = \underline{\mathbf{K}}_{\mathrm{s}}(n+1)$ e, portanto,

$$\underline{\mathbf{K}}_{\mathrm{s}}(n) = \mathbf{P}\underline{\mathbf{K}}_{\mathrm{s}}(n)\mathbf{P}. \tag{3.130}$$

Além disso, pré-multiplicando (3.130) por $\mathbf{P}$ e manipulando a expressão resultante, obtém-se

$$\mathbf{P}\underline{\mathbf{K}}_{\mathrm{s}}(n) = \underline{\mathbf{K}}_{\mathrm{s}}(n)\mathbf{P}. \tag{3.131}$$

Considerando (3.130) e (3.131), (3.129) pode ser reescrita como

$$\underline{\mathbf{K}}_s(n+1) = \underline{\mathbf{K}}_s(n) - 2\mu\left[\underline{\mathbf{K}}_s(n)\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P} + \mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}\underline{\mathbf{K}}_s(n)\right] + 4\mu^2\xi_{min}\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}. \quad (3.132)$$

e, ainda, (3.125) como

$$\xi(n) = \xi_{min} + \mathrm{tr}\left[\mathbf{P}\underline{\mathbf{K}}_s(n)\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\right] = \xi_{min} + \mathrm{tr}\left[\underline{\mathbf{K}}_s(n)\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}\right]. \quad (3.133)$$

Como o produto $\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}$ resulta em uma matriz hermitiana [1], a decomposição $\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P} = \mathbf{Q_P}\tilde{\Lambda}\mathbf{Q_P}^T$ é possível, com $\mathbf{Q_P}$ representando a matriz de autovetores de $\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}$ e $\tilde{\Lambda}$ uma matriz diagonal composta pelos autovalores de $\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}$. Considerando tal decomposição, de maneira análoga a (2.73), obtém-se a seguinte expressão para o erro quadrático médio:

$$\xi(n) = \xi_{min} + \mathrm{tr}\left[\underline{\mathbf{K}}_s(n)\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}\right] = \xi_{min} + \tilde{\boldsymbol{\lambda}}^T\underline{\bar{\mathbf{k}}}_s(n) \quad (3.134)$$

onde $\tilde{\boldsymbol{\lambda}}$ é o vetor dos autovalores de $\mathbf{P}\underline{\tilde{\mathbf{R}}}_{VV}\mathbf{P}$ e $\underline{\bar{\mathbf{k}}}_s(n)$ é um vetor que contém os elementos da diagonal principal da matriz $\underline{\bar{\mathbf{K}}}_s(n) = \mathbf{Q_P}^T\underline{\mathbf{K}}_s(n)\mathbf{Q_P}$. Agora, pré-multiplicando (3.132) por $\mathbf{Q_P}^T$, pós-multiplicando por $\mathbf{Q_P}$ e levando-se em conta que $\mathbf{Q_P}^T\mathbf{Q_P} = \mathbf{Q_P}\mathbf{Q_P}^T = \mathbf{I}$ onde $\mathbf{I}$ é a matriz identidade, obtém-se

$$\underline{\bar{\mathbf{K}}}_s(n+1) = \underline{\bar{\mathbf{K}}}_s(n) - 2\mu\left[\underline{\bar{\mathbf{K}}}_s(n)\tilde{\Lambda} + \tilde{\Lambda}\underline{\bar{\mathbf{K}}}_s(n)\right] + 4\mu^2\xi_{min}\tilde{\Lambda}. \quad (3.135)$$

Considerando que $\tilde{\Lambda}$ é uma matriz diagonal, tem-se a diagonal principal de $\underline{\bar{\mathbf{K}}}_s(n)\tilde{\Lambda}$ igual a diagonal principal de $\tilde{\Lambda}\underline{\bar{\mathbf{K}}}_s(n)$, e a seguinte expressão recursiva para o vetor $\underline{\bar{\mathbf{k}}}_s(n)$ é obtida:

$$\underline{\bar{\mathbf{k}}}_s(n+1) = (\mathbf{I} - 4\mu\tilde{\Lambda})\underline{\bar{\mathbf{k}}}_s(n) + 4\mu^2\xi_{min}\tilde{\boldsymbol{\lambda}}. \quad (3.136)$$

Assim, o uso de (3.134) em conjunto com (3.136) permite obter as curvas de aprendizagem das diferentes implementações interpoladas do filtro Volterra adaptadas pelo algoritmo LMS.

3.6.3 Forma Alternativa de Cálculo do Vetor de Coeficientes Ótimo com Restrições

Conforme apresentado na Seção 3.6.1, o vetor de coeficientes ótimo do filtro Volterra interpolado pode ser calculado a partir de (3.109). Uma expressão mais simples para o cálculo de tal vetor é obtida considerando que, após a convergência, o vetor de coeficientes do filtro Volterra interpolado adaptado pelo algoritmo LMS é em média igual ao vetor de coeficientes ótimo. Assim,

$$E[\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1)] = E[\underline{\mathbf{h}}_{\mathrm{Vs}}(n)] = E[\underline{\mathbf{h}}_{\mathrm{Vs}}(\infty)] = \underline{\mathbf{h}}_{\mathrm{Vso}}. \tag{3.137}$$

Considerando (3.137), (3.120) pode ser reescrita como

$$\underline{\mathbf{h}}_{\mathrm{Vso}} = \mathbf{P}(\mathbf{I} - 2\mu\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}})\underline{\mathbf{h}}_{\mathrm{Vso}} + 2\mu\mathbf{P}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.138}$$

Pré-multiplicando (3.138) por $\mathbf{P}$ e considerando que $\mathbf{P} = \mathbf{PP}$, é fácil verificar que $\mathbf{P}\underline{\mathbf{h}}_{\mathrm{Vso}} = \underline{\mathbf{h}}_{\mathrm{Vso}}$. Dessa forma, manipulando (3.138) obtém-se

$$\mathbf{P}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{P}\underline{\mathbf{h}}_{\mathrm{Vso}} = \mathbf{P}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.139}$$

Conforme comentado anteriormente, $\mathbf{P}$ é uma matriz diagonal cujos elementos são iguais a um quando correspondem a coeficientes que não sofrem restrição e iguais a zero quando correspondem a coeficientes que sofrem restrição. Analisando a estrutura da matriz resultante do produto $\mathbf{CC}^{\mathrm{T}}$, observa-se que tal matriz também é diagonal e, ao contrário da matriz $\mathbf{P}$, apresenta elementos iguais a zero quando correspondem a coeficientes que não sofrem restrição e iguais a um quando correspondem a coeficientes que sofrem restrição. Para ilustrar tais características, considera-se um caso simples de um filtro Volterra interpolado de primeira ordem (equivalente a um filtro IFIR) com $N = 5$ e $L = 2$, onde a matriz $\mathbf{P}$ é dada por

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.140}$$

e a matriz $\mathbf{CC}^{\mathrm{T}}$ por

$$\mathbf{CC}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.141}$$

A partir de (3.140) e (3.141), observa-se que $\mathbf{P}$ é complementar a $\mathbf{CC}^{\mathrm{T}}$, o que também pode ser concluído a partir de (3.118). Além disso, de forma análoga a (3.141), é possível definir uma matriz $\mathbf{C}_{\mathrm{c}}$ tal que

$$\mathbf{P} = \mathbf{C}_{\mathrm{c}}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}. \tag{3.142}$$

A estrutura de $\mathbf{C}_{\mathrm{c}}$ é análoga e complementar à estrutura da matriz $\mathbf{C}$, isto é, enquanto $\mathbf{C}$ é a matriz utilizada para aplicar as restrições aos coeficientes iguais a zero do vetor de coeficientes esparso, $\mathbf{C}_{\mathrm{c}}$ é a matriz que aplicaria as restrições nos coeficientes diferentes

de zero do mesmo vetor. Para exemplificar, considerando o caso de um filtro Volterra interpolado de primeira ordem com $N = 5$ e $L = 2,$ tem-se

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}} \tag{3.143}$$

e

$$\mathbf{C}_{\mathrm{c}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}. \tag{3.144}$$

Agora, substituindo (3.142) em (3.139), obtém-se

$$\mathbf{C}_{\mathrm{c}}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{C}_{\mathrm{c}}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vso}} = \mathbf{C}_{\mathrm{c}}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.145}$$

Considerando que $\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\mathbf{C}_{\mathrm{c}} = \mathbf{I},$ a pré-multiplicação em ambos os lados de (3.145) por $\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}$ resulta em

$$\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{C}_{\mathrm{c}}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vso}} = \mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.146}$$

Manipulando (3.146), obtém-se

$$\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vso}} = \left(\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{C}_{\mathrm{c}}\right)^{-1}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.147}$$

Finalmente, pré-multiplicando ambos os lados de (3.147) por $\mathbf{C}_{\mathrm{c}},$ considerando (3.142) e, ainda, que $\mathbf{P}\underline{\mathbf{h}}_{\mathrm{Vso}} = \underline{\mathbf{h}}_{\mathrm{Vso}},$ a seguinte expressão para o vetor de coeficientes ótimo é obtida:

$$\underline{\mathbf{h}}_{\mathrm{Vso}} = \mathbf{C}_{\mathrm{c}}\left(\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{C}_{\mathrm{c}}\right)^{-1}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}. \tag{3.148}$$

Note que (3.148) é uma expressão mais simples do que (3.109) para o cálculo do vetor de coeficientes ótimo. Além disso, (3.148) possibilita uma nova interpretação para tal cálculo. Considerando a estrutura de $\mathbf{C}_{\mathrm{c}},$ observa-se que o produto $\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{C}_{\mathrm{c}}$ resulta na remoção das linhas e colunas de $\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}$ relacionadas aos coeficientes que sofrem restrição, com a conseqüente redução de dimensão. Portanto, o cálculo de $(\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{C}_{\mathrm{c}})^{-1}$ é o cálculo da inversa da matriz de autocorrelação de entrada obtida a partir de um vetor de entrada reduzido $\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)$ que possui apenas os elementos correspondentes aos coeficientes que não sofrem restrição. Além disso, o produto $\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}$ resulta em um vetor, também de dimensão reduzida, composto pelos elementos de $\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}$ correspondentes aos coeficientes que não sofrem restrição. Conseqüentemente, $(\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}\mathbf{C}_{\mathrm{c}})^{-1}\mathbf{C}_{\mathrm{c}}^{\mathrm{T}}\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}$ é o cálculo do vetor de coeficientes ótimo em um espaço vetorial reduzido coerente com o vetor de entrada

$\mathbf{C}_c^T \underline{\tilde{\mathbf{x}}}_V(n)$. A pré-multiplicação do vetor resultante de $(\mathbf{C}_c^T \underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c)^{-1} \mathbf{C}_c^T \underline{\tilde{\mathbf{p}}}_V$ por $\mathbf{C}_c$, que resulta em (3.148), implica apenas redimensionamento de tal vetor através da inserção de zeros nas posições correspondentes aos elementos que sofrem restrição. Outro aspecto importante de (3.148), que deve ser ressaltado, é a sua equivalência com (3.109). A prova de tal equivalência pode ser obtida igualando ambas as expressões, o que resulta em

$$\mathbf{C}_c(\mathbf{C}_c^T \underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c)^{-1} \mathbf{C}_c^T \underline{\tilde{\mathbf{p}}}_V = \underline{\tilde{\mathbf{R}}}_{VV}^{-1} \underline{\tilde{\mathbf{p}}}_V - \underline{\tilde{\mathbf{R}}}_{VV}^{-1} \mathbf{C}(\mathbf{C}^T \underline{\tilde{\mathbf{R}}}_{VV}^{-1} \mathbf{C})^{-1} \mathbf{C}^T \underline{\tilde{\mathbf{R}}}_{VV}^{-1} \underline{\tilde{\mathbf{p}}}_V. \tag{3.149}$$

Pré-multiplicando (3.149) por $\underline{\tilde{\mathbf{R}}}_{VV}$, obtém-se

$$\underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c(\mathbf{C}_c^T \underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c)^{-1} \mathbf{C}_c^T \underline{\tilde{\mathbf{p}}}_V = \underline{\tilde{\mathbf{p}}}_V - \mathbf{C}(\mathbf{C}^T \underline{\tilde{\mathbf{R}}}_{VV}^{-1} \mathbf{C})^{-1} \mathbf{C}^T \underline{\tilde{\mathbf{R}}}_{VV}^{-1} \underline{\tilde{\mathbf{p}}}_V. \tag{3.150}$$

A partir de (3.143) e (3.144) observa-se que o produto $\mathbf{C}_c^T \mathbf{C}$ resulta em uma matriz nula. Dessa forma, pré-multiplicando (3.150) por $\mathbf{C}_c^T$, obtém-se

$$\mathbf{C}_c^T \underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c(\mathbf{C}_c^T \underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c)^{-1} \mathbf{C}_c^T \underline{\tilde{\mathbf{p}}}_V = \mathbf{C}_c^T \underline{\tilde{\mathbf{p}}}_V \tag{3.151}$$

o que, considerando que $\mathbf{C}_c^T \underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c(\mathbf{C}_c^T \underline{\tilde{\mathbf{R}}}_{VV} \mathbf{C}_c)^{-1} = \mathbf{I}$, resulta em

$$\mathbf{C}_c^T \underline{\tilde{\mathbf{p}}}_V = \mathbf{C}_c^T \underline{\tilde{\mathbf{p}}}_V. \tag{3.152}$$

Como a igualdade expressa em (3.152) é sempre verdadeira, a igualdade entre (3.109) e (3.148) é comprovada.

3.6.4 Considerações sobre o Cálculo da Matriz de Autocorrelação de Entrada

Uma das maiores dificuldades para avaliar o desempenho dos filtros Volterra interpolados a partir das expressões desenvolvidas nas seções anteriores é o cálculo das matrizes $\underline{\tilde{\mathbf{R}}}_{VV}$ como também de $\underline{\tilde{\mathbf{p}}}_V$. Isso se deve ao fato de que tais matrizes dependem de momentos de ordens elevadas do sinal de entrada filtrado pelo interpolador. No entanto, é possível obter os valores de $\underline{\tilde{\mathbf{R}}}_{VV}$ e $\underline{\tilde{\mathbf{p}}}_V$ a partir das próprias estatísticas do sinal de entrada, simplificando, assim, esse processo. O primeiro passo nessa direção é considerar que $\underline{\tilde{\mathbf{R}}}_{VV}$ e $\underline{\tilde{\mathbf{p}}}_V$ podem ser obtidas a partir de $\tilde{\mathbf{R}}_{VV}$ e $\tilde{\mathbf{p}}_V$, que são matrizes similares, porém obtidas considerando a representação convencional da relação de entrada e saída do filtro Volterra. Para obter $\underline{\tilde{\mathbf{R}}}_{VV}$ é necessário apenas remover as linhas e colunas repetidas de $\tilde{\mathbf{R}}_{VV}$, enquanto o vetor $\underline{\tilde{\mathbf{p}}}_V$ é obtido removendo as linhas de $\tilde{\mathbf{p}}_V$ correspondentes àquelas extraídas de $\tilde{\mathbf{R}}_{VV}$ para obtenção de $\underline{\tilde{\mathbf{R}}}_{VV}$. Considerando tal característica, torna-se possível

utilizar as definições apresentadas na Seção 3.4 para o cálculo das matrizes de correlação. Assim, a partir de (3.62) é possível escrever cada submatriz de $\tilde{\mathbf{R}}_{\mathrm{VV}}$ como

$$\tilde{\mathbf{R}}_{p_1p_2} = E[\mathbf{G}_{p_1}^{\mathrm{T}}\mathbf{x}_{p_1\mathrm{e}}(n)\mathbf{x}_{p_2\mathrm{e}}^{\mathrm{T}}(n)\mathbf{G}_{p_2}] = \mathbf{G}_{p_1}^{\mathrm{T}}\mathbf{R}_{p_1p_2\mathrm{e}}\mathbf{G}_{p_2} \tag{3.153}$$

com $\mathbf{R}_{p_1p_2\mathrm{e}} = E[\mathbf{x}_{p_1\mathrm{e}}(n)\mathbf{x}_{p_2\mathrm{e}}^{\mathrm{T}}(n)]$. Considerando (3.153), a matriz $\tilde{\mathbf{R}}_{\mathrm{VV}}$ é dada por

$$\tilde{\mathbf{R}}_{\mathrm{VV}} = \begin{bmatrix} \mathbf{G}^{\mathrm{T}}\mathbf{R}_{11\mathrm{e}}\mathbf{G} & \mathbf{G}^{\mathrm{T}}\mathbf{R}_{12\mathrm{e}}\mathbf{G}_2 & \cdots & \mathbf{G}^{\mathrm{T}}\mathbf{R}_{1P\mathrm{e}}\mathbf{G}_P \\ \mathbf{G}_2^{\mathrm{T}}\mathbf{R}_{21\mathrm{e}}\mathbf{G} & \mathbf{G}_2^{\mathrm{T}}\mathbf{R}_{22\mathrm{e}}\mathbf{G}_2 & \cdots & \mathbf{G}_2^{\mathrm{T}}\mathbf{R}_{2P\mathrm{e}}\mathbf{G}_P \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_P^{\mathrm{T}}\mathbf{R}_{P1\mathrm{e}}\mathbf{G} & \mathbf{G}_P^{\mathrm{T}}\mathbf{R}_{P2\mathrm{e}}\mathbf{G}_2 & \cdots & \mathbf{G}_P^{\mathrm{T}}\mathbf{R}_{PP\mathrm{e}}\mathbf{G}_P \end{bmatrix}. \tag{3.154}$$

De maneira análoga a (3.155) e (3.156), é possível escrever

$$\tilde{\mathbf{p}}_p = E[d(n)\tilde{\mathbf{x}}_p(n)] \tag{3.157}$$

o que, a partir de (3.62), resulta em

$$\tilde{\mathbf{p}}_p = E[d(n)\mathbf{G}_p^{\mathrm{T}}\mathbf{x}_{p\mathrm{e}}(n)] = \mathbf{G}_p^{\mathrm{T}}E[d(n)\mathbf{x}_{p\mathrm{e}}(n)] = \mathbf{G}_p^{\mathrm{T}}\mathbf{p}_{p\mathrm{e}} \tag{3.158}$$

com $\mathbf{p}_{p\mathrm{e}} = E[d(n)\mathbf{x}_{p\mathrm{e}}(n)]$. Assim, considerando (3.158), obtém-se

$$\tilde{\mathbf{p}}_{\mathrm{V}} = [\mathbf{p}_{1\mathrm{e}}^{\mathrm{T}}\mathbf{G} \quad \mathbf{p}_{2\mathrm{e}}^{\mathrm{T}}\mathbf{G}_2 \quad \cdots \quad \mathbf{p}_{P\mathrm{e}}^{\mathrm{T}}\mathbf{G}_P]^{\mathrm{T}}. \tag{3.159}$$

Usando (3.154) e (3.159), a obtenção de $\underline{\tilde{\mathbf{R}}}_{\mathrm{VV}}$ e $\underline{\tilde{\mathbf{p}}}_{\mathrm{V}}$ torna-se bastante facilitada uma vez que só depende do modelo probabilístico do sinal de entrada. Adicionalmente, uma abordagem similar à utilizada para o cálculo de (3.154) e (3.159) pode ser adotada para obter as matrizes necessárias para análise dos filtros Volterra parcialmente interpolados ou das implementações interpoladas sem efeito de borda. Exemplificando, o cálculo de $\underline{\tilde{\mathbf{R}}}'_{\mathrm{VV}}$ e $\underline{\tilde{\mathbf{p}}}'_{\mathrm{V}}$ requeridos para a análise de filtros Volterra interpolados sem efeito de borda resulta em

$$\tilde{\mathbf{R}}'_{\mathrm{VV}} = \begin{bmatrix} \mathbf{G}'^{\mathrm{T}}\mathbf{R}_{11}\mathbf{G}' & \mathbf{G}'^{\mathrm{T}}\mathbf{R}_{12}\mathbf{G}'_2 & \cdots & \mathbf{G}'^{\mathrm{T}}\mathbf{R}_{1P}\mathbf{G}'_P \\ \mathbf{G}'^{\mathrm{T}}_2\mathbf{R}_{21}\mathbf{G}' & \mathbf{G}'^{\mathrm{T}}_2\mathbf{R}_{22}\mathbf{G}'_2 & \cdots & \mathbf{G}'^{\mathrm{T}}_2\mathbf{R}_{2P}\mathbf{G}'_P \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}'^{\mathrm{T}}_P\mathbf{R}_{P1}\mathbf{G}' & \mathbf{G}'^{\mathrm{T}}_P\mathbf{R}_{P2}\mathbf{G}'_2 & \cdots & \mathbf{G}'^{\mathrm{T}}_P\mathbf{R}_{PP}\mathbf{G}'_P \end{bmatrix} \tag{3.160}$$

e

$$\tilde{\mathbf{p}}'_{\mathrm{V}} = [\mathbf{p}_1^{\mathrm{T}}\mathbf{G}' \quad \mathbf{p}_2^{\mathrm{T}}\mathbf{G}'_2 \quad \cdots \quad \mathbf{p}_P^{\mathrm{T}}\mathbf{G}'_P]^{\mathrm{T}} \tag{3.161}$$

com $\mathbf{R}_{p_1p_2} = E[\mathbf{x}_{p_1}(n)\mathbf{x}_{p_2}^{\mathrm{T}}(n)]$ e $\mathbf{p}_p = E[d(n)\mathbf{x}_p(n)]$.

3.6.5 Simulações

Nesta seção, alguns resultados de simulações são apresentados para verificar o desempenho do modelo desenvolvido neste capítulo. Tais resultados envolvem a modelagem, a partir de (3.110), (3.134) e (3.136), das curvas de aprendizagem das diferentes implementações interpoladas consideradas nos exemplos apresentados na Seção 3.5.3. Além disso, os procedimentos descritos na Seção 3.6.4 são utilizados para o cálculo das matrizes de correlação necessárias. Resultados referentes à modelagem do comportamento médio dos coeficientes não serão apresentados aqui devido à sua similaridade com os resultados apresentados em [31] como também com os resultados da Seção 2.3.8. É importante ressaltar que, como as aproximações consideradas para o desenvolvimento do modelo com restrições são similares às utilizadas para o desenvolvimento do modelo do filtro Volterra convencional (Seção 2.3), as características e limitações de ambos os modelos também são similares.

Exemplo 3.6.5.1: Neste exemplo, as curvas de aprendizagem dos filtros interpolado, interpolado com remoção do efeito de borda, parcialmente interpolado e parcialmente interpolado com remoção do efeito de borda obtidas no Exemplo *3.5.3.1* são consideradas. Os resultados da modelagem de tais curvas a partir de (3.134) e (3.136) estão apresentados na Figura 3.23. A partir de tal figura, observa-se uma concordância muito boa entre as curvas obtidas com o modelo proposto e as obtidas por simulação de Monte Carlo (200 realizações), verificando, assim, a precisão do modelo desenvolvido.

Exemplo 3.6.5.2: Para este exemplo, são consideradas as curvas de aprendizagem das implementações interpoladas utilizadas no Exemplo *3.5.3.2*. Os resultados da modelagem estão apresentados na Figura 3.24. Novamente um muito bom casamento entre as curvas obtidas com o modelo proposto e as curvas obtidas por simulação de Monte Carlo (200 realizações) é observada.

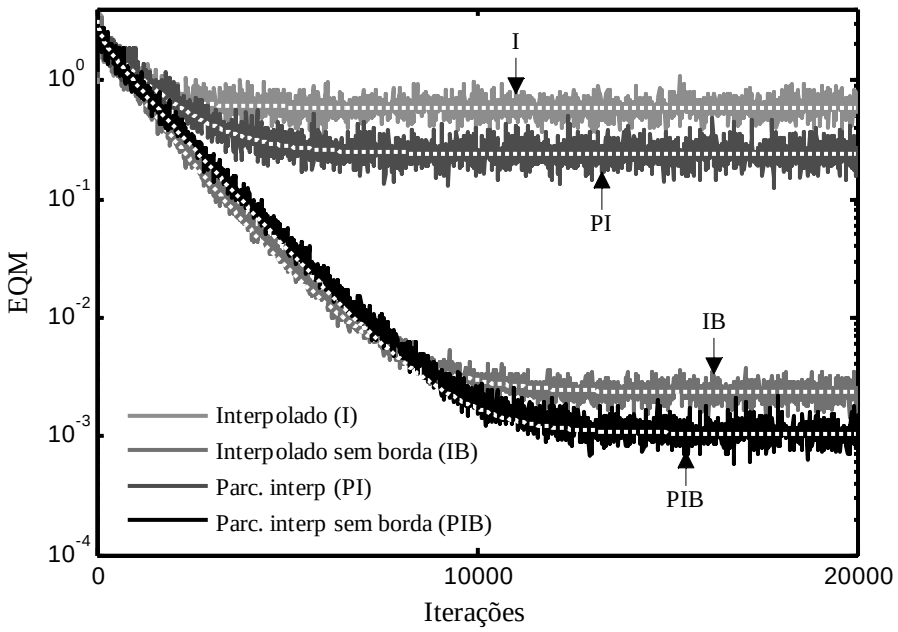


Figura 3.23. Exemplo *3.6.5.1*. Modelagem das curvas de aprendizagem do Exemplo *3.5.3.1*. (Linhas sólidas) simulações de Monte Carlo. (Linhas pontilhadas brancas) curvas do modelo proposto, obtidas a partir de (3.134) e (3.136).

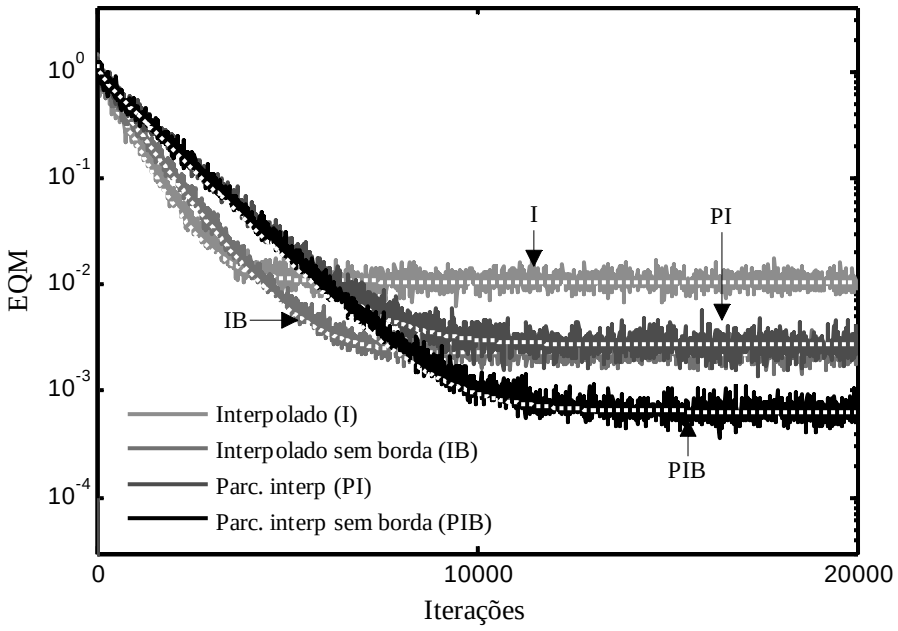


Figura 3.24. Exemplo *3.6.5.2*. Modelagem das curvas de aprendizagem do Exemplo *3.5.3.2*. (Linhas sólidas) simulações de Monte Carlo. (Linhas pontilhadas brancas) curvas do modelo proposto, obtidas a partir de (3.134) e (3.136).

Exemplo 3.6.5.3: Neste exemplo, as curvas de aprendizagem das abordagens interpoladas utilizadas no Exemplo *3.5.3.3* são apresentadas. Como o algoritmo NLMS foi o utilizado para obtenção de tais curvas, não é possível utilizar as expressões (3.134) e (3.136) uma vez que estas descrevem as curvas de aprendizagem para o algoritmo LMS. No entanto, uma análise do desempenho em regime permanente é possível a partir de (3.110). O resultado dessa análise está ilustrado na Figura 3.25. Nessa figura, observa-se uma boa estimativa do regime permanente a partir de (3.110). O pequeno descasamento observado é devido ao fato de o algoritmo NLMS apresentar um desajuste maior que o do LMS em regime permanente para passos de adaptação grandes (nesse caso $\alpha = 0,25$). Refazendo as simulações para um passo de adaptação menor, dado por $\alpha = 0,1$, resultados ainda melhores são obtidos, conforme mostrado na Figura 3.26.

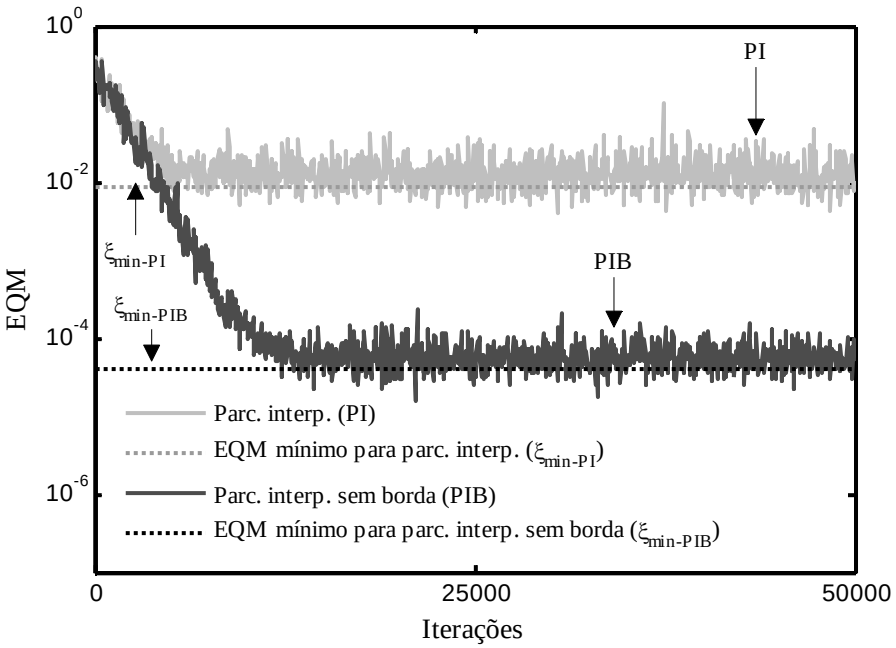


Figura 3.25. Exemplo *3.6.5.3*. Modelagem em regime permanente das implementações interpoladas do Exemplo *3.5.3.3* com $\alpha = 0.25$. (Linhas sólidas) simulações de Monte Carlo. (Linhas pontilhadas) curvas do modelo proposto, obtidas a partir de (3.110).

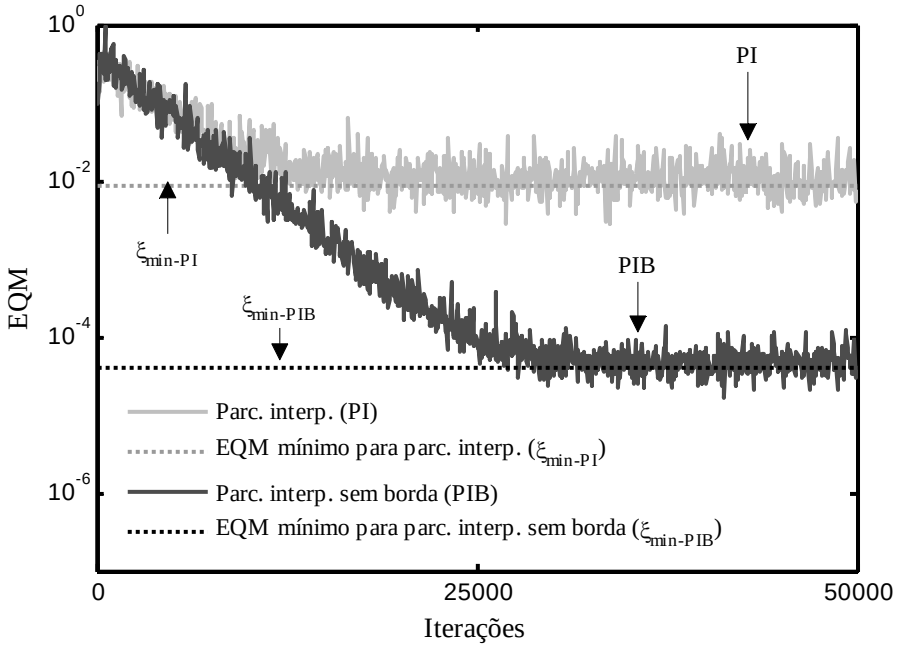


Figura 3.26. Exemplo *3.6.5.3*. Modelagem em regime permanente das implementações interpoladas do Exemplo *3.5.3.3* com $\alpha = 0.1$. (Linhas sólidas) simulações de Monte Carlo. (Linhas pontilhadas) curvas do modelo proposto, obtidas a partir de (3.110).

3.7 Considerações

Neste capítulo, os filtros Volterra interpolados foram descritos e diferentes estudos correlatos foram apresentados. Dentre os quais destacam-se, como contribuições originais, a descrição matemática dos filtros Volterra interpolados e a concepção de um procedimento para remoção do indesejável efeito de borda. Outra contribuição original apresentada neste capítulo, é o desenvolvimento de um modelo estatístico que descreve o desempenho em regime permanente do filtro Volterra interpolado adaptativo bem como o seu comportamento transitório quando adaptado pelo algoritmo LMS. Através dessas contribuições, o que se busca é aumentar o potencial de aplicação dos filtros Volterra interpolados através tanto da melhoria do seu desempenho quanto do desenvolvimento de importantes ferramentas de análise.

CAPÍTULO 4

Filtros Volterra Interpolados Inteiramente Adaptativos

Como descrito no capítulo anterior, a implementação de filtros Volterra adaptativos usando abordagens interpoladas permite uma redução de carga computacional quase sempre às expensas de alguma perda de desempenho. Os principais fatores que contribuem para tal perda de desempenho são: (a) a característica intrínseca de posto reduzido das estruturas interpoladas [57]; (b) o efeito de borda, que pode ser removido usando o procedimento apresentado no capítulo anterior; e (c) a escolha inapropriada dos coeficientes do filtro interpolador, os quais são responsáveis por recriar, na estrutura equivalente, os coeficientes zerados do filtro esparso. Para contornar este último problema, a implementação de filtros Volterra interpolados inteiramente adaptativos, em que tanto o interpolador quanto o filtro esparso têm seus coeficientes adaptados, é discutida neste capítulo. Assim, a adaptação completa de filtros FIR em cascata bem como de filtros FIR interpolados é inicialmente tratada, com posterior extensão dos resultados obtidos para a implementação de filtros Volterra interpolados.

4.1 Uma Abordagem Alternativa para a Adaptação de Filtros FIR em Cascata

Conforme mencionado na Seção 3.2, os filtros IFIR são compostos por dois filtros FIR, um interpolador e um filtro esparso, dispostos em cascata. Assim, o estudo do processo adaptativo de filtros FIR em cascata é de grande interesse para a análise e implementação de filtros IFIR inteiramente adaptativos [59], [60]. Na Figura 4.1, o diagrama de blocos de uma estrutura de dois filtros FIR em cascata é ilustrado. Nessa figura, $\mathbf{w}_{\mathrm{a}}$ representa o filtro de entrada, com tamanho de memória N_{a} e vetor de coeficientes

$$\mathbf{w}_{\mathrm{a}} = [w_{\mathrm{a}}(0) \;\; w_{\mathrm{a}}(1) \;\; \cdots \;\; w_{\mathrm{a}}(N_{\mathrm{a}}-1)]^{\mathrm{T}} \tag{4.1}$$

e $\mathbf{w}_{\mathrm{b}}$ representa o filtro de saída, com tamanho de memória N_{b} e vetor de coeficientes

$$\mathbf{w}_{\mathrm{b}} = [w_{\mathrm{b}}(0) \;\; w_{\mathrm{b}}(1) \;\; \cdots \;\; w_{\mathrm{b}}(N_{\mathrm{b}}-1)]^{\mathrm{T}}. \tag{4.2}$$

A variável $x(n)$ denota o sinal de entrada, $y_{\mathrm{a}}(n)$ representa o sinal intermediário e $y(n)$ é o sinal de saída. O filtro equivalente da estrutura em cascata é representado por $\mathbf{w}_{\mathrm{ab}}$ e seu vetor de coeficientes é obtido fazendo a convolução entre as repostas ao impulso de $\mathbf{w}_{\mathrm{a}}$ e $\mathbf{w}_{\mathrm{b}}$. Assim, o tamanho de memória do filtro equivalente $\mathbf{w}_{\mathrm{ab}}$ é dado por

$$N_{\mathrm{ab}} = N_{\mathrm{a}} + N_{\mathrm{b}} - 1. \tag{4.3}$$

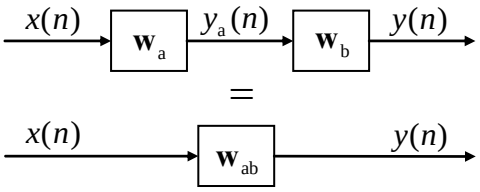


Figura 4.1. Diagrama de blocos de uma estrutura FIR em cascata e seu filtro equivalente.

Para facilitar o tratamento matemático, o cálculo de $\mathbf{w}_{\mathrm{ab}}$ pode ser escrito como um produto de matrizes de forma similar ao procedimento apresentado na Seção 3.4.1 para os filtros IFIR. Assim, as seguintes matrizes são definidas:

$$\mathbf{W}_{\mathrm{a}} = \begin{Bmatrix} \begin{bmatrix} \\ \mathbf{w}_{\mathrm{a}} \\ \\ \end{bmatrix} & 0 & 0 & \cdots & 0 \\ 0 & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{a}} \\ \\ \end{bmatrix} & 0 & \cdots & 0 \\ 0 & 0 & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{a}} \\ \\ \end{bmatrix} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{a}} \\ \\ \end{bmatrix} \end{Bmatrix} = \begin{bmatrix} w_{\mathrm{a}}(0) & 0 & 0 & \cdots & 0 \\ w_{\mathrm{a}}(1) & w_{\mathrm{a}}(0) & 0 & \cdots & 0 \\ w_{\mathrm{a}}(2) & w_{\mathrm{a}}(1) & w_{\mathrm{a}}(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{\mathrm{a}}(N_{\mathrm{a}}-1) & w_{\mathrm{a}}(N_{\mathrm{a}}-2) & w_{\mathrm{a}}(N_{\mathrm{a}}-3) & \cdots & w_{\mathrm{a}}(0) \\ 0 & w_{\mathrm{a}}(N_{\mathrm{a}}-1) & w_{\mathrm{a}}(N_{\mathrm{a}}-2) & \cdots & w_{\mathrm{a}}(1) \\ 0 & 0 & w_{\mathrm{a}}(N_{\mathrm{a}}-1) & \cdots & w_{\mathrm{a}}(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w_{\mathrm{a}}(N_{\mathrm{a}}-1) \end{bmatrix} \tag{4.4}$$

com dimensão $(N_{\mathrm{a}} + N_{\mathrm{b}} - 1) \times N_{\mathrm{b}}$, e

$$\mathbf{W}_{\mathrm{b}} = \begin{Bmatrix} \begin{bmatrix} \\ \mathbf{w}_{\mathrm{b}} \\ \\ \end{bmatrix} & 0 & 0 & \cdots & 0 \\ 0 & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{b}} \\ \\ \end{bmatrix} & 0 & \cdots & 0 \\ 0 & 0 & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{b}} \\ \\ \end{bmatrix} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & \begin{bmatrix} \\ \mathbf{w}_{\mathrm{b}} \\ \\ \end{bmatrix} \end{Bmatrix} = \begin{bmatrix} w_{\mathrm{b}}(0) & 0 & 0 & \cdots & 0 \\ w_{\mathrm{b}}(1) & w_{\mathrm{b}}(0) & 0 & \cdots & 0 \\ w_{\mathrm{b}}(2) & w_{\mathrm{b}}(1) & w_{\mathrm{b}}(0) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{\mathrm{b}}(N_{\mathrm{b}}-1) & w_{\mathrm{b}}(N_{\mathrm{b}}-2) & w_{\mathrm{b}}(N_{\mathrm{b}}-3) & \cdots & w_{\mathrm{b}}(0) \\ 0 & w_{\mathrm{b}}(N_{\mathrm{b}}-1) & w_{\mathrm{b}}(N_{\mathrm{b}}-2) & \cdots & w_{\mathrm{b}}(1) \\ 0 & 0 & w_{\mathrm{b}}(N_{\mathrm{b}}-1) & \cdots & w_{\mathrm{b}}(2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & w_{\mathrm{b}}(N_{\mathrm{b}}-1) \end{bmatrix} \tag{4.5}$$

com dimensão $(N_a + N_b - 1) \times N_a$. Com isso, torna-se possível escrever

$$\mathbf{w}_{ab} = \mathbf{W}_a \mathbf{w}_b = \mathbf{W}_b \mathbf{w}_a . \tag{4.6}$$

Levando em conta que o vetor equivalente $\mathbf{w}_{ab}$ possui a dimensão dada em (4.3), é possível definir um vetor de entrada $\mathbf{x}_{ab}(n)$ como

$$\mathbf{x}_{ab}(n) = \left[x(n) \;\; x(n-1) \;\; \cdots \;\; x(n - N_a - N_b + 2) \right]^T \tag{4.7}$$

de tal forma que a relação de entrada e saída da estrutura FIR em cascata possa ser escrita em termos do seguinte produto matricial:

$$y(n) = \mathbf{w}_{ab}^T \mathbf{x}_{ab}(n) = \mathbf{w}_b^T \mathbf{W}_a^T \mathbf{x}_{ab}(n) = \mathbf{w}_a^T \mathbf{W}_b^T \mathbf{x}_{ab}(n). \tag{4.8}$$

Considerando agora a estrutura FIR em cascata aplicada a um problema de estimação de sinal, adaptando ambos os filtros temos a situação ilustrada na Figura 4.2. Nessa figura, $\mathbf{w}_a(n)$ e $\mathbf{w}_b(n)$ representam as versões variantes no tempo de $\mathbf{w}_a$ e $\mathbf{w}_b$ da Figura 4.1, $d(n)$ denota o sinal a ser estimado (sinal desejado) e $e(n)$ representa o sinal de erro, dado por

$$e(n) = d(n) - y(n). \tag{4.9}$$

Os demais sinais da Figura 4.2 são os mesmos da Figura 4.1. Substituindo (4.8) em (4.9) e considerando que agora os coeficientes de ambos os filtros são variantes no tempo, obtém-se

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}_a^T(n) \mathbf{W}_b^T(n) \mathbf{x}_{ab}(n) \\ &= d(n) - \mathbf{w}_b^T(n) \mathbf{W}_a^T(n) \mathbf{x}_{ab}(n) \end{aligned} \tag{4.10}$$

onde $\mathbf{W}_a(n)$ e $\mathbf{W}_b(n)$ são as versões variantes no tempo de (4.4) e (4.5), respectivamente.

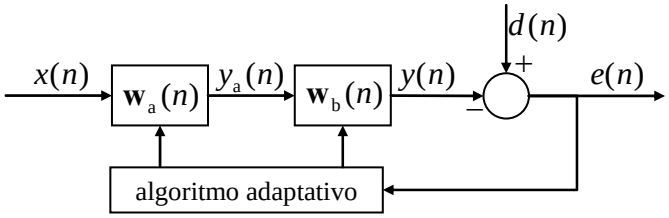


Figura 4.2. Diagrama de blocos da estrutura FIR em cascata inteiramente adaptativa.

4.1.1 Algoritmo LMS aplicado a Filtros FIR em Cascata

A função custo utilizada para o desenvolvimento do algoritmo LMS é a estimativa instantânea do erro quadrático médio [1], dada por

$$\hat{J}_{MSE}(n) = e^2(n) . \tag{4.11}$$

Considerando o esquema da Figura 4.2, os coeficientes de cada um dos filtros são adaptados usando o gradiente da função custo (4.11) conforme descrito em [2]. Assim, para o filtro de entrada $\mathbf{w}_a(n)$, a adaptação é realizada da seguinte maneira:

$$\mathbf{w}_a(n+1) = \mathbf{w}_a(n) - \mu_a \nabla_{\mathbf{w}_a} e^2(n) \tag{4.12}$$

com μ_a representando o passo de adaptação. Aplicando a regra da cadeia, o vetor gradiente de (4.12) pode ser escrito como

$$\nabla_{\mathbf{w}_a} e^2(n) = \frac{\partial e^2(n)}{\partial \mathbf{w}_a(n)} = \frac{\partial e^2(n)}{\partial e(n)} \frac{\partial e(n)}{\partial \mathbf{w}_a(n)}. \tag{4.13}$$

Dessa maneira, a partir de (4.10), os termos de (4.13) são dados por

$$\frac{\partial e^2(n)}{\partial e(n)} = 2e(n) \tag{4.14}$$

e

$$\frac{\partial e(n)}{\partial \mathbf{w}_a(n)} = -\mathbf{W}_b^{\mathrm{T}}(n)\mathbf{x}_{ab}(n). \tag{4.15}$$

Assim, a equação de adaptação para o filtro de entrada é dada por

$$\mathbf{w}_a(n+1) = \mathbf{w}_a(n) + 2\mu_a e(n)\mathbf{W}_b^{\mathrm{T}}(n)\mathbf{x}_{ab}(n). \tag{4.16}$$

Considerando o filtro de saída e usando também a regra do gradiente, tem-se

$$\mathbf{w}_b(n+1) = \mathbf{w}_b(n) - \mu_b \nabla_{\mathbf{w}_b} e^2(n) \tag{4.17}$$

onde μ_b é o passo de adaptação do filtro de saída. De forma análoga a (4.13), obtém-se

$$\nabla_{\mathbf{w}_b} e^2(n) = \frac{\partial e^2(n)}{\partial \mathbf{w}_b(n)} = \frac{\partial e^2(n)}{\partial e(n)} \frac{\partial e(n)}{\partial \mathbf{w}_b(n)} \tag{4.18}$$

resultando em (4.14) e

$$\frac{\partial e(n)}{\partial \mathbf{w}_b(n)} = -\mathbf{W}_a^{\mathrm{T}}(n)\mathbf{x}_{ab}(n). \tag{4.19}$$

Finalmente, a equação de adaptação para os coeficientes do filtro de saída é dada por

$$\mathbf{w}_b(n+1) = \mathbf{w}_b(n) + 2\mu_b e(n)\mathbf{W}_a^{\mathrm{T}}(n)\mathbf{x}_{ab}(n). \tag{4.20}$$

4.1.2 Filtros FIR em Cascata Adaptados pelo Algoritmo NLMS

As expressões para adaptação da estrutura em cascata usando o algoritmo NLMS são obtidas de maneira análoga ao procedimento realizado em [2] para obtenção do algoritmo NLMS para um filtro FIR convencional. Assim, considerando o filtro de entrada

da estrutura em cascata, a equação de adaptação é obtida minimizando a norma euclidiana de

$$\delta\mathbf{w}_a(n+1) = \mathbf{w}_a(n+1) - \mathbf{w}_a(n) \tag{4.21}$$

sujeita à seguinte restrição:

$$\mathbf{w}_a^T(n+1)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n) = d(n). \tag{4.22}$$

É importante ressaltar que a expressão de restrição (4.22) é ligeiramente diferente da apresentada em [2] devido às características particulares da estrutura em cascata. Aplicando o método dos multiplicadores de Lagrange [2] a (4.21) e (4.22), a função custo pode ser escrita como

$$J_{\mathbf{w}_a}(n) = \left\|\delta\mathbf{w}_a(n+1)\right\|^2 + \theta_a(n)\left[d(n) - \mathbf{w}_a^T(n+1)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n)\right] \tag{4.23}$$

onde $\theta_a(n)$ é um multiplicador de Lagrange escalar com valor real. Diferenciando (4.23) com respeito a $\mathbf{w}_a(n+1)$, obtém-se

$$\frac{\partial J_{\mathbf{w}_a}(n)}{\partial\mathbf{w}_a(n+1)} = 2\left[\mathbf{w}_a(n+1) - \mathbf{w}_a(n)\right] - \theta_a(n)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n). \tag{4.24}$$

Fazendo (4.24) igual a zero, tem-se

$$\mathbf{w}_a(n+1) = \mathbf{w}_a(n) + \frac{1}{2}\theta_a(n)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n). \tag{4.25}$$

A substituição de (4.25) em (4.22) resulta em

$$\begin{aligned} d(n) &= \left(\mathbf{w}_a(n) + \frac{1}{2}\theta_a(n)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n)\right)^T \mathbf{W}_b^T(n)\mathbf{x}_{ab}(n) \\ &= \mathbf{w}_a^T(n)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n) + \frac{1}{2}\theta_a(n)\left\|\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n)\right\|^2. \end{aligned} \tag{4.26}$$

Agora, resolvendo (4.26) para $\theta_a(n)$, obtém-se

$$\theta_a(n) = \frac{2\left[d(n) - \mathbf{w}_a^T(n)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n)\right]}{\left\|\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n)\right\|^2} = \frac{2e(n)}{\left\|\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n)\right\|^2}. \tag{4.27}$$

Finalmente, substituindo (4.27) em (4.25), e adicionando as constantes positivas α_a e ψ_a para controlar o processo adaptativo e prevenir a divisão por zero [2], a equação de adaptação dos coeficientes do filtro de entrada é

$$\mathbf{w}_a(n+1) = \mathbf{w}_a(n) + \frac{\alpha_a}{\left\|\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n)\right\|^2 + \psi_a} e(n)\mathbf{W}_b^T(n)\mathbf{x}_{ab}(n) \tag{4.28}$$

De forma similar a (4.21) e (4.22), para o filtro de saída, é necessário minimizar a norma euclidiana de

$$\delta\mathbf{w}_{\mathrm{b}}(n+1) = \mathbf{w}_{\mathrm{b}}(n+1) - \mathbf{w}_{\mathrm{b}}(n) \tag{4.29}$$

submetida à restrição

$$\mathbf{w}_{\mathrm{b}}^{\mathrm{T}}(n+1)\mathbf{W}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{ab}}(n) = d(n). \tag{4.30}$$

Novamente, aplicando o método dos multiplicadores de Lagrange [2], obtém-se

$$J_{\mathbf{w}_{\mathrm{b}}}(n) = \left\|\delta\mathbf{w}_{\mathrm{b}}(n+1)\right\|^2 + \theta_{\mathrm{b}}(n)\left[d(n) - \mathbf{w}_{\mathrm{b}}^{\mathrm{T}}(n+1)\mathbf{W}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{ab}}(n)\right]. \tag{4.31}$$

Note que (4.31) tem a mesma forma de (4.23). Assim, através de um procedimento similar ao realizado para se obter (4.28), a seguinte equação de adaptação é obtida:

$$\mathbf{w}_{\mathrm{b}}(n+1) = \mathbf{w}_{\mathrm{b}}(n) + \frac{\alpha_{\mathrm{b}}}{\left\|\mathbf{W}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{ab}}(n)\right\|^2 + \psi_{\mathrm{b}}} e(n)\mathbf{W}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{ab}}(n) \tag{4.32}$$

onde α_{b} e ψ_{b} são constantes positivas utilizadas para controlar o processo adaptativo e prevenir a divisão por zero [2].

4.1.3 Particularidades do Processo Adaptativo dos Filtros FIR em Cascata

A partir da análise das expressões para adaptação de filtros FIR em cascata apresentadas nas Seções 4.1.1 e 4.1.2, algumas situações particulares resultando em problemas para a implementação são observados. Dentre eles, destacam-se:

i) *Inicialização dos coeficientes*. No caso da inicialização de ambos os vetores de coeficientes [$\mathbf{w}_{\mathrm{a}}(n)$ e $\mathbf{w}_{\mathrm{b}}(n)$] com valores iguais a zero, o que é comumente realizado, o processo adaptativo não inicia. Tal característica, também destacada em [61] para o caso dos filtros IFIR, pode ser facilmente verificada a partir de (4.16) e (4.20) (algoritmo LMS) ou a partir de (4.28) e (4.32) (algoritmo NLMS).

ii) *Complexidade computacional*. Novamente a partir de (4.16) e (4.20) (algoritmo LMS) ou a partir de (4.28) e (4.32) (algoritmo NLMS), observa-se que o cálculo de dois produtos de matrizes [$\mathbf{W}_{\mathrm{b}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{ab}}(n)$ e $\mathbf{W}_{\mathrm{a}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{ab}}(n)$] é requerido a cada iteração, implicando um alto custo computacional mesmo considerando que alguns dos elementos de $\mathbf{W}_{\mathrm{a}}(n)$ e $\mathbf{W}_{\mathrm{b}}(n)$ são sempre iguais a zero.

iii) *Estabilidade do algoritmo*. No caso do algoritmo LMS, é necessário realizar uma escolha apropriada dos valores dos dois passos de adaptação (μ_{a} e μ_{b}) de forma a garantir a estabilidade do algoritmo. Contudo, a obtenção de limites de estabilidade analíticos não é uma tarefa trivial, sendo este um problema ainda sem solução conhecida na literatura. Uma

possível alternativa para lidar com tal problema é usar um valor conservativo do passo de adaptação idêntico para ambos os filtros, o qual pode ser obtido e ajustado através de simulações.

iv) *Convergência do vetor de coeficientes equivalente*. É fácil verificar, a partir de (4.6), que diferentes combinações dos vetores de coeficientes $\mathbf{w}_a$ e $\mathbf{w}_b$ podem resultar em um mesmo vetor de coeficientes equivalente $\mathbf{w}_{ab}$. Como exemplo, o uso de vetores de coeficientes $\bar{\mathbf{w}}_a = k\mathbf{w}_a$ e $\bar{\mathbf{w}}_b = (1/k)\mathbf{w}_b$, o que implica $\bar{\mathbf{W}}_a = k\mathbf{W}_a$ e $\bar{\mathbf{W}}_b = (1/k)\mathbf{W}_b$ com k representando um escalar qualquer, produz o mesmo vetor de coeficientes equivalente que $\mathbf{w}_a$ e $\mathbf{w}_b$, ou seja,

$$\mathbf{w}_{ab} = \mathbf{W}_a\mathbf{w}_b = \bar{\mathbf{W}}_a\bar{\mathbf{w}}_b = \left(k\mathbf{W}_a\right)\left(\frac{1}{k}\mathbf{w}_b\right). \tag{4.33}$$

Considerando (4.33), observa-se que, para um valor suficientemente grande de k, é possível obter o vetor equivalente $\mathbf{w}_{ab}$ a partir de um vetor $\bar{\mathbf{w}}_a = k\mathbf{w}_a$ com valores grandes de coeficientes e um vetor $\bar{\mathbf{w}}_b = (1/k)\mathbf{w}_b$ com valores pequenos. Tal característica é indesejável principalmente em implementações usando processadores com aritmética de ponto fixo. Nesse caso, devido à adoção de um número limitado de *bits* para representação de cada coeficiente (palavra de dados de comprimento fixo), valores muito pequenos implicam perda de precisão, enquanto valores muito grandes podem exceder o valor máximo admitido pela representação adotada (*overflow*).

A consideração dos fatores descritos acima é de fundamental importância para uma implementação eficiente de filtros FIR em cascata adaptativos e das estruturas adaptativas derivadas de tal filtro, como, por exemplo, os filtros IFIR inteiramente adaptativos.

4.2 Filtros IFIR Inteiramente Adaptativos

A forma mais comum de se implementar filtros IFIR adaptativos é através da adaptação apenas do filtro esparso. No entanto, tal estratégia muitas vezes resulta em um pobre desempenho, em termos do valor mínimo do erro quadrático médio, devido a uma recriação de coeficientes ineficiente em conseqüência de uma escolha inapropriada dos coeficientes do filtro interpolador. Uma solução para tal problema é o uso de uma estrutura IFIR inteiramente adaptativa, na qual o interpolador também é adaptativo. Tal classe de estruturas foi originalmente proposta em [61]. Posteriormente, outras estruturas IFIR

inteiramente adaptativas foram também propostas em [57] e em [62]. De forma alternativa às abordagens adotadas em [57], [61] e [62], as expressões para adaptação de filtros IFIR inteiramente adaptativos são desenvolvidas, nesta seção, com base nas expressões adaptativas para a estrutura FIR em cascata obtidas na Seção 4.1. A abordagem aqui utilizada é interessante por permitir observar algumas características do processo adaptativo não discutidas nos trabalhos anteriores, como, por exemplo, certas aproximações adotadas com o objetivo de reduzir a complexidade computacional [60].

4.2.1 Adaptação usando o Algoritmo LMS

Considerando a estrutura IFIR, descrita nas Seções 3.2 e 3.4, aplicada a um problema de estimação de sinal (Figura 4.2), é possível escrever o sinal de erro, de forma análoga a (4.10), como

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}_s^{\mathrm{T}}(n)\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n) \\ &= d(n) - \mathbf{g}^{\mathrm{T}}(n)\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{x}_e(n) \end{aligned} \tag{4.34}$$

com $\mathbf{x}_e(n)$ dado por (3.43). A partir de (4.34), realizando um procedimento similar ao apresentado na Seção 4.1.1, a seguinte expressão para adaptação do filtro interpolador da estrutura IFIR, usando algoritmo LMS, é obtida:

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_1 e(n)\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{x}_e(n). \tag{4.35}$$

Para o caso do filtro esparso, é necessário, além de considerar o procedimento da Seção 4.1.1, levar em conta as restrições, decorrentes da esparsidade, de maneira similar à apresentada na Seção 3.6. Assim, tais restrições são descritas por

$$\mathbf{C}^{\mathrm{T}}\mathbf{w}_s(n) = \mathbf{f} \tag{4.36}$$

onde $\mathbf{C}$ é a matriz de restrições, $\mathbf{f}$ é o vetor de reposta com todos os elementos iguais a zero e $\mathbf{w}_s(n)$ é o vetor de coeficientes do filtro esparso da estrutura IFIR. Adicionando tais restrições à função custo (4.11) através do método dos multiplicadores de Lagrange [63], obtém-se

$$\begin{aligned} J_{\mathbf{w}_s}(n) = {} & d^2(n) - 2d(n)\mathbf{w}_s^{\mathrm{T}}(n)\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n) \\ & + \mathbf{w}_s^{\mathrm{T}}(n)\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n)\mathbf{x}_e^{\mathrm{T}}(n)\mathbf{G}(n)\mathbf{w}_s(n) + \boldsymbol{\theta}^{\mathrm{T}}(n)[\mathbf{C}^{\mathrm{T}}\mathbf{w}_s(n) - \mathbf{f}]. \end{aligned} \tag{4.37}$$

Realizando um desenvolvimento similar ao apresentado na Seção 3.6.2, a seguinte expressão para adaptação do filtro esparso é obtida:

$$\mathbf{w}_s(n+1) = \mathbf{P}\left[\mathbf{w}_s(n) + 2\mu_2 e(n)\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n)\right] \tag{4.38}$$

onde $\mathbf{P} = \mathbf{I} - \mathbf{C}\mathbf{C}^{\mathrm{T}}$. Note que, como o lado direito de (4.38) está pré-multiplicado por $\mathbf{P}$, (4.38) corresponde ao algoritmo LMS adaptando apenas os coeficientes de $\mathbf{w}_{\mathrm{s}}(n)$ que não sofrem restrição de forma análoga a (3.117) [ou veja (3.119)]. É importante ressaltar que (4.35) e (4.38) são válidas para a adaptação tanto da estrutura IFIR com interpolador na entrada (Figura 3.1) quanto da estrutura IFIR com interpolador na saída (Figura 3.2).

De maneira similar ao que foi discutido na Seção 4.1.3 para a estrutura FIR em cascata, a implementação de (4.35) e (4.38) implica um elevado custo computacional devido aos produtos de matrizes envolvidos. Para contornar tal problema, é possível, considerando uma condição de adaptação lenta, admitir algumas simplificações de forma a reduzir o custo computacional. Isso é realizado levando-se em conta que o produto $\hat{\mathbf{x}}(n) = \mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{e}}(n),$ dado por

$$\hat{\mathbf{x}}(n) = \mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{e}}(n) = [\mathbf{w}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{s}}(n) \quad \mathbf{w}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{s}}(n-1) \quad \cdots \quad \mathbf{w}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{s}}(n-M+1)]^{\mathrm{T}} \tag{4.39}$$

pode ser aproximado por

$$\underset{\sim}{\hat{\mathbf{x}}}(n) = [\mathbf{w}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{s}}(n) \quad \mathbf{w}_{\mathrm{s}}^{\mathrm{T}}(n-1)\mathbf{x}_{\mathrm{s}}(n-1) \quad \cdots \quad \mathbf{w}_{\mathrm{s}}^{\mathrm{T}}(n-M+1)\mathbf{x}_{\mathrm{s}}(n-M+1)]^{\mathrm{T}} \tag{4.40}$$

onde $\mathbf{x}_{\mathrm{s}}(n),$ de (4.39) e (4.40), é dado por

$$\mathbf{x}_{\mathrm{s}}(n) = \left[x(n) \quad x(n-1) \quad \cdots \quad x(n-N+1)\right]^{\mathrm{T}}. \tag{4.41}$$

Para se obter os elementos de (4.39), o cálculo de todos os M produtos internos vetoriais presentes em tal expressão são necessários a cada iteração. Por outro lado, para se obter (4.40) a cada iteração, é necessário apenas o cálculo do primeiro elemento $\mathbf{w}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{s}}(n)$ e seu reuso nas iterações seguintes. Uma aproximação similar é realizada para calcular o produto $\tilde{\mathbf{x}}(n) = \mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{e}}(n)$ de (4.38), resultando no vetor $\underset{\sim}{\tilde{\mathbf{x}}}(n)$. Assim, a partir de (4.35) e (4.38), as equações de adaptação simplificadas para o filtro IFIR inteiramente adaptativo são dadas por

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_1 e(n)\underset{\sim}{\hat{\mathbf{x}}}(n) \tag{4.42}$$

e

$$\mathbf{w}_{\mathrm{s}}(n+1) = \mathbf{P}\left[\mathbf{w}_{\mathrm{s}}(n) + 2\mu_2 e(n)\underset{\sim}{\tilde{\mathbf{x}}}(n)\right]. \tag{4.43}$$

As expressões (4.42) e (4.43) para adaptação dos filtros IFIR inteiramente adaptativos usando o algoritmo LMS são iguais às apresentadas em [61] e [62]. No entanto, nesses trabalhos os efeitos das hipóteses simplificativas consideradas [uso das versões aproximadas $\underset{\sim}{\tilde{\mathbf{x}}}(n)$ e $\underset{\sim}{\hat{\mathbf{x}}}(n)$] não são destacados nem avaliados. É importante mencionar

que, quando tais simplificações são adotadas, um cuidado especial deve ser tomado para a escolha dos passos de adaptação de forma a garantir a estabilidade do algoritmo.

Além da adoção das aproximações descritas anteriormente, outra maneira de reduzir a complexidade computacional para implementação dos filtros IFIR inteiramente adaptativos é levar em conta a posição do interpolador na estrutura. Conforme descrito em [64], os filtros IFIR adaptativos com interpolador fixo devem ser implementados com o filtro interpolador na entrada uma vez que, com isso, um menor custo computacional é obtido. No caso dos filtros IFIR inteiramente adaptativos, a situação se modifica e uma menor complexidade computacional é obtida utilizando a estrutura IFIR com interpolador na saída. Isso é facilmente observado primeiramente verificando que em ambos os casos, com o interpolador na entrada ou na saída, os vetores $\hat{\mathbf{x}}(n)$ e $\tilde{\mathbf{x}}(n)$ ou suas versões simplificadas $\underset{\sim}{\hat{\mathbf{x}}}(n)$ e $\underset{\sim}{\tilde{\mathbf{x}}}(n)$ precisam ser calculados a cada interação para que a adaptação dos filtros seja realizada. Adicionalmente, a saída da estrutura IFIR também precisa ser calculada a cada iteração, o que resulta em $y(n) = \mathbf{w}_s^{\mathrm{T}}(n)\tilde{\mathbf{x}}(n) \approx \mathbf{w}_s^{\mathrm{T}}(n)\underset{\sim}{\tilde{\mathbf{x}}}(n)$ para o caso com interpolador na entrada ou $y(n) = \mathbf{g}^{\mathrm{T}}(n)\hat{\mathbf{x}}(n) \approx \mathbf{g}^{\mathrm{T}}(n)\underset{\sim}{\hat{\mathbf{x}}}(n)$ para o caso com o interpolador na saída. Como usualmente o interpolador apresenta um número de coeficientes muito menor do que o filtro esparso, o cálculo do sinal de saída para o caso com o interpolador na saída apresenta um custo computacional menor. As diferenças de complexidade computacional entre as diversas implementações dos filtros IFIR adaptativos e inteiramente adaptativos podem ser observadas mais claramente a partir das curvas do número de operações por amostra em função do tamanho de memória, apresentadas na Figura 4.3, considerando o fator de esparsidade $L = 2$. Dessa figura, observa-se que a implementação do filtro IFIR inteiramente adaptativo com simplificações e interpolador na saída apresenta complexidade computacional muito próxima daquela requerida para implementação do filtro IFIR adaptativo com interpolador fixo. Além disso, nota-se que o uso das implementações inteiramente adaptativas sem aproximações não é recomendado uma vez que elas apresentam complexidade computacional maior do que a própria implementação do filtro FIR convencional (sem interpolação e esparsidade).

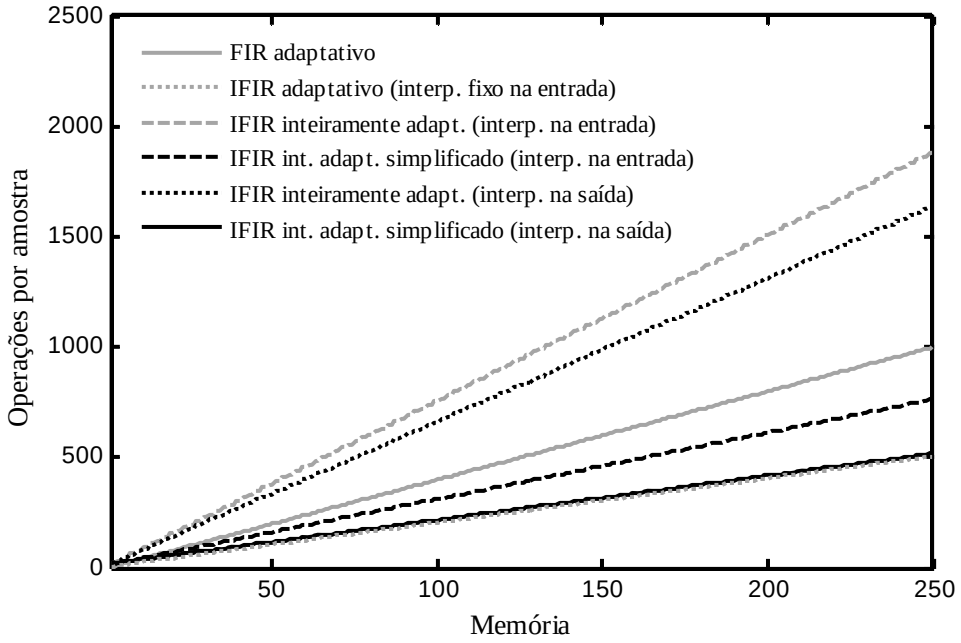


Figura 4.3. Número de operações por amostra em função do tamanho de memória requerido para implementação, usando o algoritmo LMS, de um filtro FIR adaptativo e de diferentes implementações adaptativas do filtro IFIR com $L = 2$.

4.2.2 Adaptação usando o Algoritmo NLMS

De maneira similar ao que foi apresentado na Seção 4.1.2, a expressão para adaptação do interpolador da estrutura IFIR usando o algoritmo NLMS é obtida minimizando a norma euclidiana de

$$\delta\mathbf{g}(n+1) = \mathbf{g}(n+1) - \mathbf{g}(n) \tag{4.44}$$

submetida à seguinte restrição:

$$\mathbf{g}^{\mathrm{T}}(n+1)\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{e}}(n) = d(n). \tag{4.45}$$

Adotando o mesmo procedimento da Seção 4.1.2, a expressão para adaptação do interpolador é

$$\mathbf{g}(n+1) = \mathbf{g}(n) + \frac{\alpha_1}{\left\|\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{e}}(n)\right\|^2 + \psi_1} e(n)\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{x}_{\mathrm{e}}(n) \tag{4.46}$$

onde α_1 e ψ_1 são parâmetros de controle do algoritmo. Para o caso do filtro esparso da estrutura interpolada, a obtenção de sua expressão de adaptação usando o algoritmo NLMS é um problema mais complexo. Isso se deve ao fato de tal expressão ser obtida minimizando a norma euclidiana de

$$\delta\mathbf{w}_{\mathrm{s}}(n+1) = \mathbf{w}_{\mathrm{s}}(n+1) - \mathbf{w}_{\mathrm{s}}(n) \tag{4.47}$$

sujeita a duas restrições, dadas por

$$\mathbf{w}_s^T(n+1)\mathbf{G}^T(n)\mathbf{x}_e(n) = d(n) \tag{4.48}$$

e

$$\mathbf{C}^T\mathbf{w}_s(n+1) = \mathbf{f}. \tag{4.49}$$

A restrição representada por (4.49) é similar a (4.36), com $\mathbf{C}$ representando a matriz de restrições e $\mathbf{f}$ o vetor de reposta com todos os elementos iguais a zero. Assim, aplicando o método dos multiplicadores de Lagrange, a função custo é escrita como

$$J_{\mathbf{w}_s}(n) = \left\|\delta\mathbf{w}_s(n+1)\right\|^2 + \theta_1(n)\left[d(n) - \mathbf{w}_s^T(n+1)\mathbf{G}^T(n)\mathbf{x}_e(n)\right] + \boldsymbol{\theta}_2^T(n)\left[\mathbf{C}^T\mathbf{w}_s(n+1) - \mathbf{f}\right]. \tag{4.50}$$

onde $\theta_1(n)$ é um multiplicador de Lagrange escalar com valor real e $\boldsymbol{\theta}_2(n)$ é um vetor de multiplicadores de Lagrange. Diferenciando (4.50) em relação a $\mathbf{w}_s(n+1)$, igualando a expressão resultante a zero e isolando $\mathbf{w}_s(n+1)$, obtém-se

$$\mathbf{w}_s(n+1) = \mathbf{w}_s(n) + \frac{1}{2}\theta_1(n)\mathbf{G}^T(n)\mathbf{x}_e(n) - \frac{1}{2}\mathbf{C}\boldsymbol{\theta}_2(n). \tag{4.51}$$

Substituindo (4.51) em (4.49), considerando que $\mathbf{C}^T\mathbf{C} = \mathbf{I}$ (matriz identidade) e isolando $\boldsymbol{\theta}_2(n)$, é possível obter

$$\boldsymbol{\theta}_2(n) = 2\mathbf{C}^T\left[\mathbf{w}_s(n) + \frac{1}{2}\theta_1(n)\mathbf{G}^T(n)\mathbf{x}_e(n)\right].. \tag{4.52}$$

Agora, substituindo (4.52) em (4.51) resulta em

$$\mathbf{w}_s(n+1) = \mathbf{P}\left[\mathbf{w}_s(n) + \frac{1}{2}\theta_1(n)\mathbf{G}^T(n)\mathbf{x}_e(n)\right] \tag{4.53}$$

com $\mathbf{P} = \mathbf{I} - \mathbf{C}\mathbf{C}^T$ dada por (3.118). Aplicando (4.53) em (4.48) e levando em conta que $\mathbf{P} = \mathbf{P}^T$, obtém-se

$$\theta_1(n)\mathbf{x}_e^T(n)\mathbf{G}(n)\mathbf{P}\mathbf{G}^T(n)\mathbf{x}_e(n) = 2\left[d(n) - \mathbf{w}_s^T(n)\mathbf{P}\mathbf{G}^T(n)\mathbf{x}_e(n)\right]. \tag{4.54}$$

Considerando a idempotência de $\mathbf{P}$, que implica $\mathbf{P} = \mathbf{P}\mathbf{P}$, é possível escrever

$$\mathbf{x}_e^T(n)\mathbf{G}(n)\mathbf{P}\mathbf{G}^T(n)\mathbf{x}_e(n) = \mathbf{x}_e^T(n)\mathbf{G}(n)\mathbf{P}\mathbf{P}\mathbf{G}^T(n)\mathbf{x}_e(n) = \left\|\mathbf{P}\mathbf{G}^T(n)\mathbf{x}_e(n)\right\|^2. \tag{4.55}$$

Além disso, é fácil verificar que $\mathbf{w}_s^T(n)\mathbf{P} = \mathbf{w}_s^T(n)$ uma vez que o produto de $\mathbf{w}_s^T(n)$ com $\mathbf{P}$ implica fixar em zero os coeficientes de $\mathbf{w}_s^T(n)$ que já são iguais a zero devido à sua natureza esparsa. Dessa forma, tem-se

$$\mathbf{w}_s^T(n)\mathbf{P}\mathbf{G}^T(n)\mathbf{x}_e(n) = \mathbf{w}_s^T(n)\mathbf{G}^T(n)\mathbf{x}_e(n) = \hat{y}(n). \tag{4.56}$$

Substituindo (4.55) e (4.56) em (4.54) e isolando $\theta_1(n)$ na expressão resultante, obtém-se

$$\theta_1(n) = \frac{2\left[d(n) - \mathbf{w}_s^{\mathrm{T}}(n)\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n) \right]}{\left\| \mathbf{P}\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n) \right\|^2} = \frac{2e(n)}{\left\| \mathbf{P}\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n) \right\|^2}. \tag{4.57}$$

Finalmente, substituindo (4.57) em (4.53) e adicionando os fatores de controle α_2 e ψ_2, a expressão para adaptação do filtro esparso da estrutura IFIR inteiramente adaptativa usando o algoritmo NLMS é

$$\mathbf{w}_s(n+1) = \mathbf{P}\mathbf{w}_s(n) + \frac{\alpha_2}{\left\| \mathbf{P}\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n) \right\|^2 + \psi_2} e(n)\mathbf{P}\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n). \tag{4.58}$$

Levando-se em conta a estrutura da matriz $\mathbf{P}$ e analisando (4.58), observa-se que tal expressão corresponde à adaptação apenas dos coeficientes do filtro esparso que são diferentes de zero e que, portanto, não sofrem restrição [ver (3.117) e (3.119)]. Outra característica importante que se pode observar a partir de (4.58), é a forma do termo de normalização, dado por $\left\| \mathbf{P}\mathbf{G}^{\mathrm{T}}(n)\mathbf{x}_e(n) \right\|^2$. Em outros trabalhos [57], [62], tal termo de normalização não é descrito de forma clara e sua derivação direta a partir das expressões apresentadas na Seção 4.2.2, sem considerar as restrições decorrentes da esparsidade, pode levar a implementações errôneas do algoritmo.

Adotando as mesmas aproximações utilizadas para obter (4.42) e (4.43), as expressões simplificadas para adaptação usando o algoritmo NLMS são dadas por

$$\mathbf{g}(n+1) = \mathbf{g}(n) + \frac{\alpha_1}{\left\| \underset{\sim}{\hat{\mathbf{x}}}(n) \right\|^2 + \psi_1} e(n)\underset{\sim}{\hat{\mathbf{x}}}(n) \tag{4.59}$$

e

$$\mathbf{w}_s(n+1) = \mathbf{P}\mathbf{w}_s(n) + \frac{\alpha_2}{\left\| \mathbf{P}\underset{\sim}{\tilde{\mathbf{x}}}(n) \right\|^2 + \psi_2} e(n)\mathbf{P}\underset{\sim}{\tilde{\mathbf{x}}}(n). \tag{4.60}$$

Também no caso do algoritmo NLMS aplicado às estruturas IFIR inteiramente interpoladas, a implementação com interpolador na saída resulta em um menor custo computacional. Adicionalmente, é importante destacar que, conforme descrito para (4.42) e (4.43), um cuidado especial deve ser tomado na escolha dos fatores de controle também de (4.59) e (4.60) devido às aproximações consideradas para o seu desenvolvimento.

4.2.3 Simulações

Nesta seção, resultados de simulação considerando problemas de identificação de sistemas [1] são apresentados com o objetivo de avaliar o desempenho das implementações inteiramente adaptativas dos filtros IFIR bem como o impacto das aproximações adotadas. As estruturas utilizadas nas simulações são o filtro FIR adaptativo convencional (AFIR), o filtro IFIR adaptativo com interpolador fixo (AIFIR), o filtro IFIR inteiramente adaptativo implementado sem aproximações [CFAIFIR, equações (4.35) e (4.38) para o algoritmo LMS e (4.46) e (4.58) para o NLMS] e o filtro IFIR inteiramente adaptativo implementado considerando aproximações [FAIFIR, equações (4.42) e (4.43) para o LMS e (4.59) e (4.60) para o NLMS]. As curvas do erro quadrático médio (EQM) obtidas com tais estruturas são comparadas para avaliação de desempenho. Para todos os casos, é adicionado a $d(n)$ um ruído de medição com variância $\sigma_v^2 = 10^{-6}$ (SNR = 60 dB). O fator de interpolação utilizado para os filtros IFIR adaptativos e inteiramente adaptativos é $L = 2$ em todos os exemplos. O interpolador fixo do filtro AIFIR é dado por $\mathbf{g} = [0,5 \;\; 1,0 \;\; 0,5]^{\mathrm{T}}$ e o interpolador variável dos filtros CFAIFIR e FAIFIR é inicializado com esse mesmo vetor de coeficientes.

Exemplo 4.2.3.1: Neste exemplo, os filtros adaptativos são utilizados para modelar uma planta cujo vetor de coeficientes é dado por

$$\mathbf{w}_1^{\mathrm{o}} = [0{,}5 \;\; 0{,}36 \;\; 0{,}26 \;\; 0{,}18 \;\; 0{,}13 \;\; 0{,}095 \;\; 0{,}068 \;\; 0{,}048 \;\; 0{,}035 \;\; 0{,}025 \;\; 0{,}018]^{\mathrm{T}} .$$

O sinal de entrada é branco e gaussiano com variância unitária. O algoritmo utilizado é o LMS e o mesmo valor do passo de adaptação $\mu = \mu_{\max} / 5$ é usado para todos os filtros, onde $\mu_{\max}$ é o valor máximo do passo de adaptação para o qual o algoritmo ainda converge (determinado experimentalmente). No caso das estruturas IFIR inteiramente adaptativas, o mesmo passo de adaptação é utilizado para o filtro esparso e para o interpolador, de forma a facilitar a obtenção de um valor de $\mu_{\max}$ que assegure a estabilidade do algoritmo. Os valores de $\mu_{\max}$ para esse exemplo são: $\mu_{\max} = 0,06$ para o AFIR, $\mu_{\max} = 0,03$ para o AIFIR, $\mu_{1\max} = \mu_{2\max} = 0,07$ para o CFAIFIR, e $\mu_{1\max} = \mu_{2\max} = 0,04$ para o FAIFIR. As curvas de EQM, obtidas a partir de simulações de Monte Carlo (média de 100 realizações independentes), são apresentadas na Figura 4.4. A partir dessa figura, observa-se que as estruturas IFIR inteiramente adaptativas apresentam melhor desempenho (termos do EQM

de regime permanente) do que aquele obtido com o filtro AIFIR. Além disso, a convergência do filtro CFAIFIR sem simplificações é mais rápida do que aquela obtida com o FAIFIR, com o mesmo valor de regime permanente. Isso se deve ao menor valor de μ_{max} obtido para o filtro FAIFIR, o que é uma conseqüência direta do uso de aproximações para o desenvolvimento de suas expressões de adaptação. Adicionalmente, escolhendo o passo de adaptação do filtro CFAIFIR igual ao do filtro FAIFIR ($\mu = \mu_{max}/5$ com $\mu_{1max} = \mu_{2max} = 0,04$) o mesmo desempenho para ambas as estruturas é obtido, o que é observado na Figura 4.5. Assim, conclui-se que uma das principais conseqüências do uso de aproximações para implementação dos filtros IFIR inteiramente interpolados adaptados pelo algoritmo LMS é a redução no limite máximo do passo de adaptação garantindo a sua estabilidade.

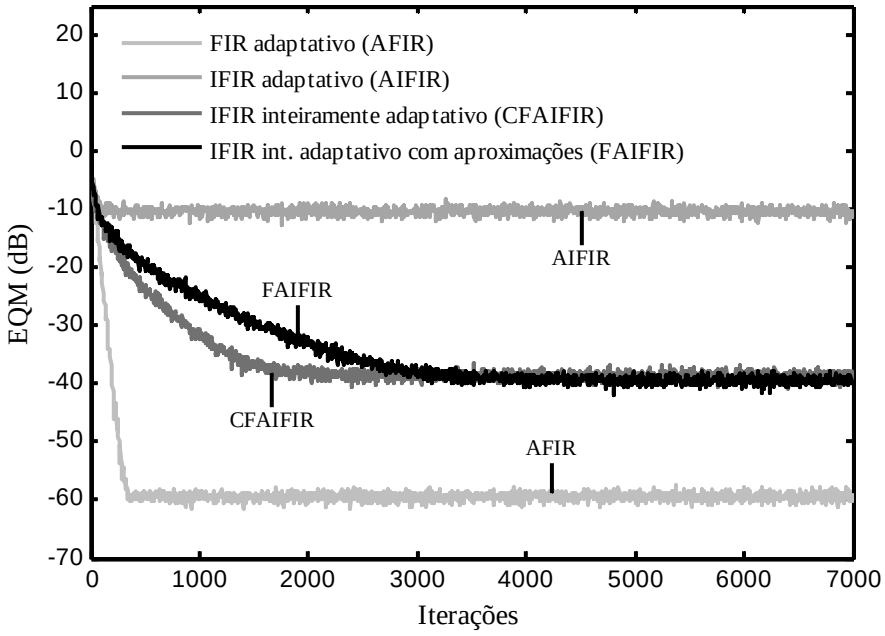


Figura 4.4. Curvas de erro quadrático médio obtidas para o Exemplo *4.2.3.1* (média de 100 realizações).

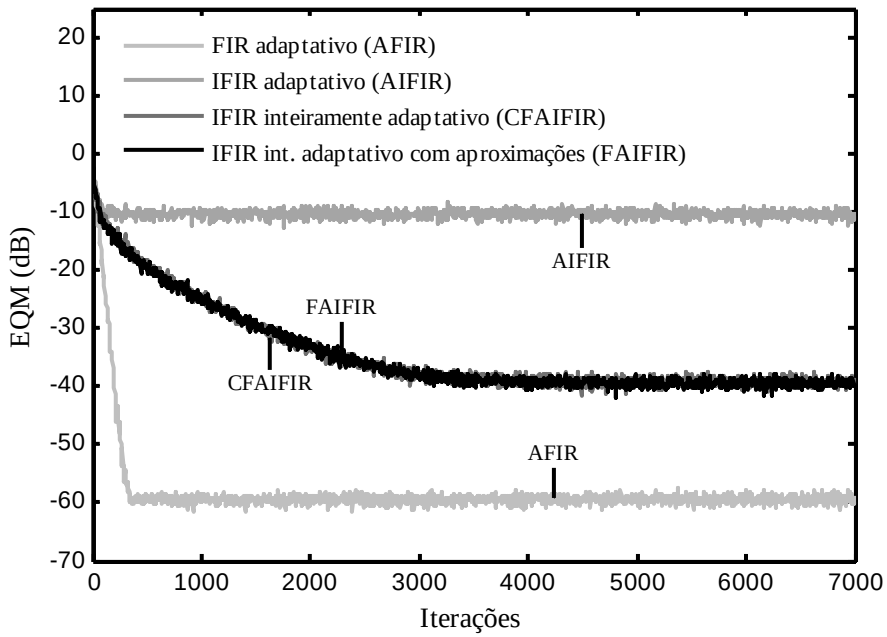


Figura 4.5. Curvas de erro quadrático médio obtidas para o Exemplo *4.2.3.1* (média de 100 realizações).

Exemplo 4.2.3.2: Para este exemplo, a planta tem 25 coeficientes dados por

$$\mathbf{w}_2^o = [0,01\ 0,03\ 0,07\ 0,12\ 0,16\ 0,24\ 0,28\ 0,37\ 0,39\ 0,48\ 0,46\ 0,54\ 0,49$$
$$0,55\ 0,46\ 0,48\ 0,39\ 0,37\ 0,28\ 0,24\ 0,16\ 0,12\ 0,07\ 0,036\ 0,013]^T.$$

O sinal de entrada é colorido e obtido a partir de um processo AR descrito por $x(n) = \beta\, x(n-1) + \sqrt{1-\beta^2}\, u(n)$, onde $u(n)$ é um sinal branco gaussiano com variância unitária e $\beta = 0,5$. Novamente, o algoritmo utilizado é o LMS com valores de passo de adaptação determinados de maneira similar ao Exemplo *4.2.3.1*. Os valores obtidos para μ_{max}, nesse caso, são: $\mu_{max} = 0,01$ para o filtro AFIR, $\mu_{max} = 0,009$ para o AIFIR, $\mu_{1max} = \mu_{2max} = 0,008$ para o filtro CFAIFIR e $\mu_{1max} = \mu_{2max} = 0,006$ para o filtro FAIFIR. As curvas de EQM obtidas são apresentadas na Figura 4.6, de onde novamente se observa o melhor desempenho dos filtros CFAIFIR e FAIFIR em relação ao AIFIR. Outra vez, o uso de passos de adaptação iguais para os filtros CFAIFIR e FAIFIR ($\mu = \mu_{max}/5$ com $\mu_{1max} = \mu_{2max} = 0,006$) leva a um desempenho bastante próximo para ambas as estruturas, o que pode ser observado na Figura 4.7. Com isso, verifica-se que a implementação com simplificações do filtro IFIR inteiramente adaptativo (FAIFIR) é a melhor opção em aplicações práticas em comparação com a implementação sem simplificações (CFAIFIR)

tanto devido ao alto custo computacional desta última quanto ao desempenho similar entre ambas.

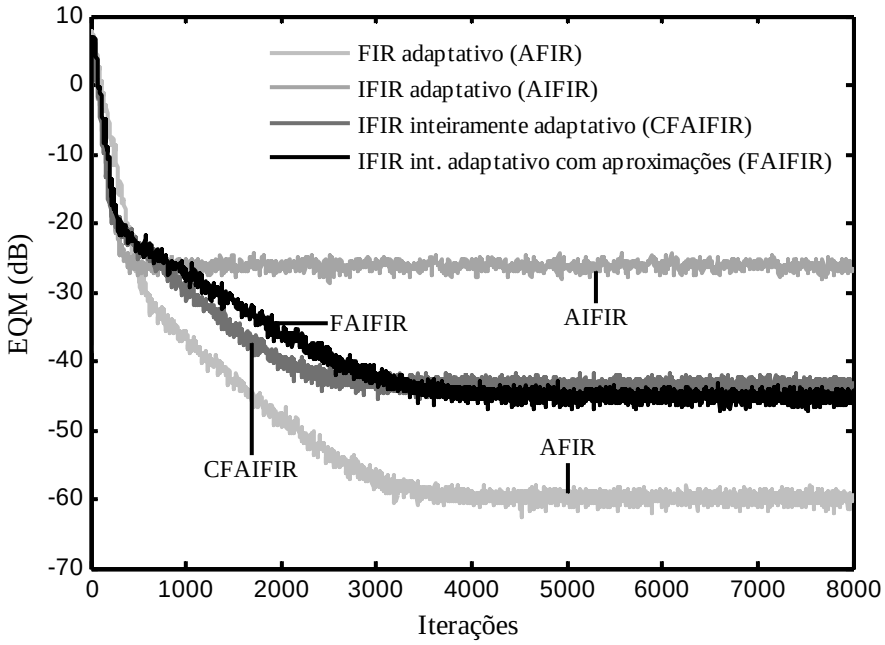


Figura 4.6. Curvas de erro quadrático médio obtidas para o Exemplo *4.2.3.2* (média de 100 realizações).

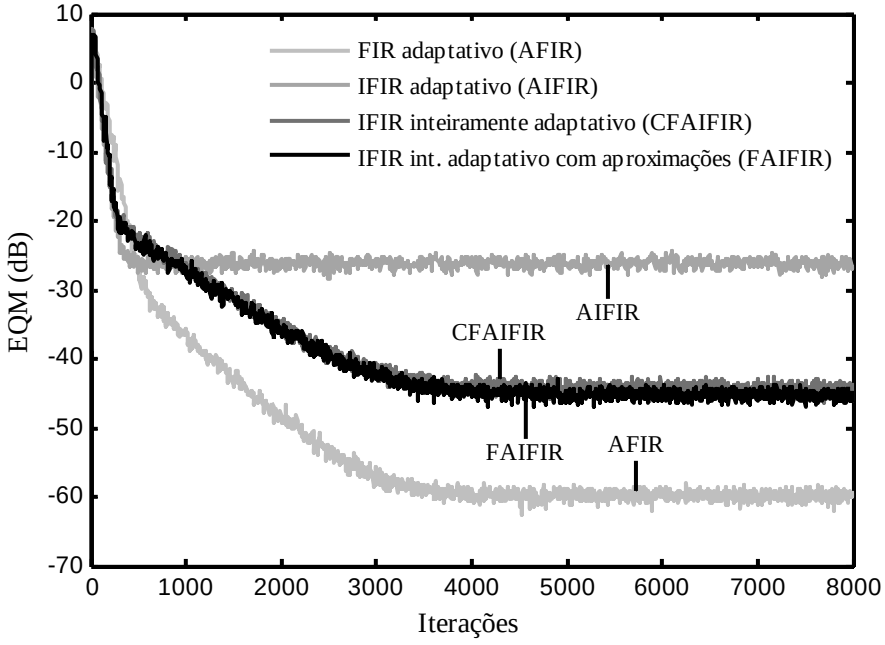


Figura 4.7. Curvas de erro quadrático médio obtidas para o Exemplo *4.2.3.2* (média de 100 realizações).

Exemplo 4.2.3.3: A planta deste exemplo é a mesma do Exemplo *4.2.3.1* e o sinal de entrada é branco e gaussiano com variância unitária. Agora, o algoritmo utilizado é o NLMS com $\alpha = \alpha_1 = \alpha_2 = 0,5$ e $\psi = \psi_1 = \psi_2 = 10^{-6}$. Os resultados de simulação obtidos estão mostrados na Figura 4.8. A partir desta figura observa-se que o desempenho dos filtros CFAIFIR e FAIFIR foi novamente melhor do que o do filtro AIFIR. Ainda, o filtro FAIFIR apresentou uma convergência mais lenta do que o filtro CFAIFIR, o que é conseqüência das simplificações adotadas. Um desempenho similar entre as estruturas CFAIFIR e FAIFIR é obtido modificando os parâmetros de controle. Por exemplo, usando $\alpha = \alpha_1 = \alpha_2 = 0,5$ e $\psi = \psi_1 = \psi_2 = 1$, as curvas apresentadas na Figura 4.9 são obtidas. Adicionalmente, modificando os parâmetros para $\alpha = \alpha_1 = \alpha_2 = 0,25$ e $\psi = \psi_1 = \psi_2 = 10^{-6}$ os resultados da Figura 4.10 são obtidos. A partir desses resultados, observa-se que existe a necessidade de um maior cuidado na escolha dos parâmetros para que a implementação simplificada do filtro IFIR inteiramente adaptativo (FAIFIR) apresente um desempenho similar ao da versão sem simplificações (CFAIFIR), destacando-se ainda que o alto custo em termos de complexidade computacional limita o uso da implementação sem simplificações em aplicações práticas.

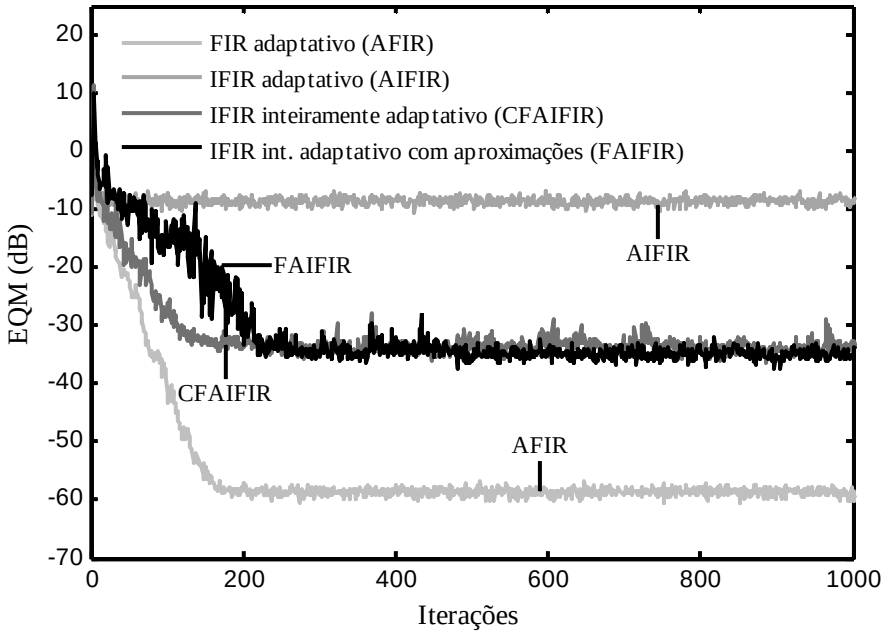


Figura 4.8. Curvas de erro quadrático médio obtidas para o Exemplo *4.2.3.3* (média de 100 realizações).

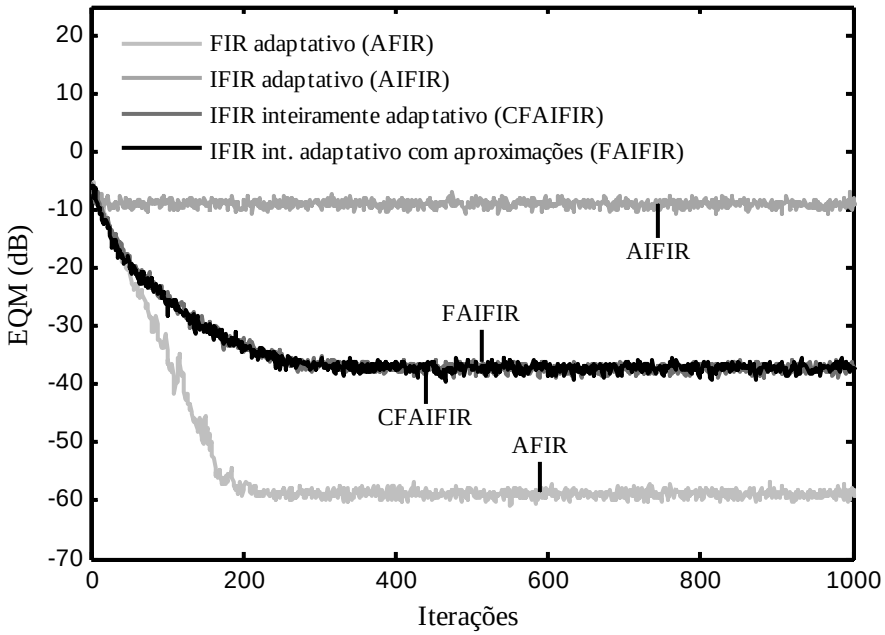


Figura 4.9. Curvas de erro quadrático médio obtidas para o Exemplo *4.2.3.3* (média de 100 realizações).

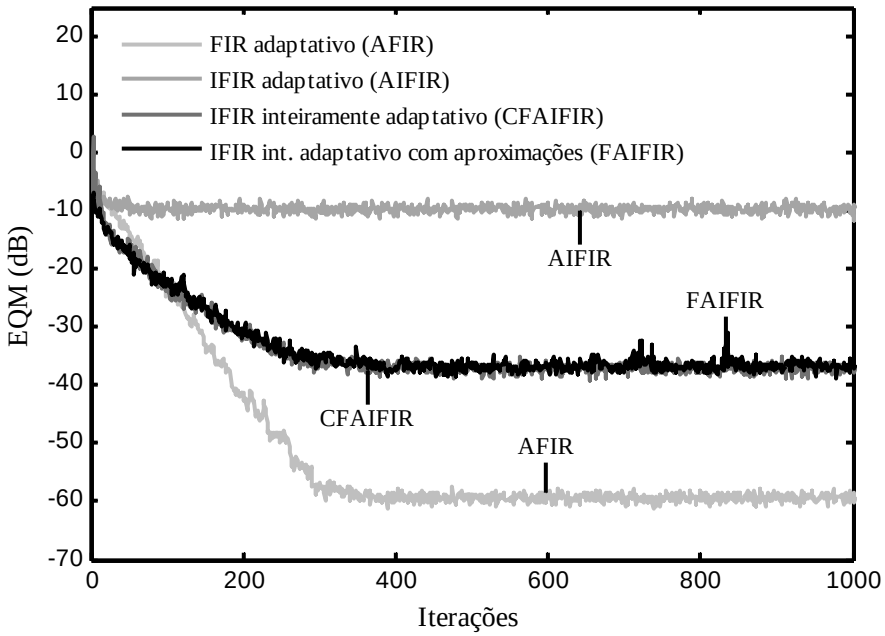


Figura 4.10. Curvas de erro quadrático médio obtidas para o Exemplo *4.2.3.3* (média de 100 realizações).

4.3 Filtros IFIR Inteiramente Adaptativos sem Efeito de Borda

A partir dos resultados apresentados na seção anterior, constata-se que os filtros IFIR inteiramente adaptativos em geral apresentam um desempenho em regime permanente superior ao dos filtros IFIR adaptativos com interpoladores fixos. No entanto, fatores como o efeito de borda limitam o desempenho dos filtros IFIR inteiramente adaptativos, restringindo assim a sua aplicação. Outros fatores que também dificultam a aplicação de tais filtros são intrínsecos às estruturas FIR em cascata adaptativas: maior complexidade computacional; necessidade de inicialização dos vetores de coeficientes com valores diferentes de zero; e necessidade de monitoramento dos vetores de coeficientes para evitar que um deles assuma valores muito pequenos e o outro valores muito grandes. Nesse contexto, a implementação com a remoção do efeito de borda dos filtros IFIR inteiramente adaptativos é descrita nesta seção. Adicionalmente, explorando algumas características particulares da implementação sem efeito de borda, uma estratégia para eliminação de alguns problemas de implementação associadas à adaptação da estrutura em cascata é apresentada. Dessa forma, novas e eficientes estratégias de implementação dos filtros IFIR inteiramente adaptativos são discutidas [65]-[66].

4.3.1 Remoção do Efeito de Borda em Filtros IFIR Inteiramente Adaptativos

Levando-se em conta que, no caso dos filtros IFIR inteiramente adaptativos, o algoritmo adaptativo realiza a otimização dos coeficientes tanto do interpolador quanto do filtro esparso com o objetivo de minimizar do erro quadrático médio, é comum supor que tal minimização seja capaz de tratar automaticamente o efeito de borda. De fato, uma redução parcial de tal efeito é observada, porém à custa de certa perda de desempenho no processo de recriação dos coeficientes. Nesse contexto, uma análise cuidadosa dos vetores de coeficientes equivalentes leva a conclusões bastante promissoras. É bem conhecido que quanto maior a correlação entre coeficientes, melhor serão os resultados obtidos através do processo de interpolação [5], [56]. Por outro lado, reescrevendo (3.14) como

$$\mathbf{w}_{\mathrm{i}} = \left[\underline{g_2 0 + g_0 w_0} \;\; \mathrm{g}_1 w_0 \;\; \boxed{g_2 w_0 + g_0 w_2} \;\; \mathrm{g}_1 w_2 \;\; \boxed{g_2 w_2 + g_0 w_4} \;\; \mathrm{g}_1 w_4 \;\; \underline{g_2 w_4 + g_0 0} \right]^{\mathrm{T}} \qquad (4.61)$$

torna-se evidente que os valores dos coeficientes relativos ao efeito de borda (sublinhados) envolvem supostos coeficientes iguais a zero que antecedem e sucedem, respectivamente, o primeiro e o último coeficiente do filtro esparso. Essa característica também é verificada em filtros interpolados com outros valores de fator de interpolação L, o que pode ser

observado, por exemplo, para $L=3$ a partir de (3.17). Assim, temos um efeito correspondente à existência de dois coeficientes iguais a zero que usualmente não são correlacionados com os demais, resultando em uma degradação de desempenho devido ao processo de interpolação. Tal problema é eliminado removendo o efeito de borda conforme descrito na Seção 3.5, o que resulta no seguinte vetor de coeficientes equivalente para o caso de (4.61):

$$\mathbf{w}_{\mathrm{i}}' = \left[g_1 w_0 \;\; \boxed{g_2 w_0 + g_0 w_2} \;\; \mathrm{g}_1 w_2 \;\; \boxed{g_2 w_2 + g_0 w_4} \;\; \mathrm{g}_1 w_4 \right]^{\mathrm{T}}. \tag{4.62}$$

Outro aspecto importante, observado em (4.61) e (4.62), é o papel do coeficiente central do interpolador, que aparece no vetor equivalente como um fator multiplicativo em relação aos coeficientes do filtro esparso. Assim, observa-se a sua função de reproduzir os coeficientes do filtro esparso no filtro equivalente, sem qualquer influência nos demais coeficientes recriados. Como conseqüência, é possível, por exemplo, fixar o valor desse coeficiente igual a 1 sem qualquer perda de generalidade da estrutura equivalente. Tal abordagem resolve alguns dos problemas associados ao processo adaptativo em cascata uma vez que elimina a necessidade de inicialização dos coeficientes com valores diferentes de zero e também o risco de o interpolador possuir coeficientes com valores muito grandes e o filtro esparso coeficientes com valores pequenos ou vice-versa. No entanto, o uso de um interpolador com o coeficiente central fixado em 1 não é interessante no caso do filtro IFIR convencional sem remoção do efeito de borda. Isso pode ser observado considerando uma situação na qual o sistema a ser modelado por um filtro IFIR, cujo vetor de coeficientes equivalente é dado por (4.61), apresenta o primeiro coeficiente com um valor maior do que o valor do segundo coeficiente. Nesse caso, o valor de w_0 em (4.61) será aumentado durante o processo adaptativo para tentar se ajustar ao valor do primeiro coeficiente e, como $g_1 = 1$, não será possível realizar uma compensação apropriada do valor do segundo coeficiente. No caso da implementação com remoção do efeito de borda, esse problema não ocorre e o uso do coeficiente central do interpolador igual a 1 torna-se vantajoso. Por outro lado, a fixação do valor de um dos coeficientes do interpolador pode afetar a convergência, tornando-a geralmente mais lenta, visto que os graus de liberdade do processo de otimização realizado pelo algoritmo adaptativo são reduzidos.

4.3.2 Adaptação usando o Algoritmo LMS

Considerando o filtro IFIR implementado utilizando o procedimento de remoção do efeito de borda, descrito na Seção 3.5, e a adaptação da estrutura FIR em cascata, é possível reescrever (4.10) como

$$\begin{aligned} e(n) &= d(n) - \mathbf{w}_s^{\mathrm{T}}(n)\mathbf{G}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) \\ &= d(n) - \mathbf{g}^{\mathrm{T}}(n)\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n). \end{aligned} \tag{4.63}$$

Realizando um procedimento similar ao da Seção 4.1.1, obtêm-se as seguintes expressões para adaptação do filtro IFIR com efeito de borda removido usando o algoritmo LMS:

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_1 e(n)\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) \tag{4.64}$$

e

$$\mathbf{w}_s(n+1) = \mathbf{P}\left[\mathbf{w}_s(n) + 2\mu_2 e(n)\mathbf{G}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right] \tag{4.65}$$

com a matriz $\mathbf{P}$ dada por (3.118). Adotando aproximações análogas às adotadas na seção anterior, as seguintes expressões simplificadas são obtidas:

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_1 e(n)\underset{\sim}{\hat{\mathbf{x}}}'(n) \tag{4.66}$$

e

$$\mathbf{w}_s(n+1) = \mathbf{P}\mathbf{w}_s(n) + 2\mu_2 e(n)\mathbf{P}\underset{\sim}{\tilde{\mathbf{x}}}'(n). \tag{4.67}$$

É importante ressaltar que (4.66) e (4.67) são válidas para adaptação tanto do filtro IFIR com interpolador na entrada quanto com interpolador na saída, sendo que, conforme descrito anteriormente, esta última implementação é de maior interesse considerando o custo computacional envolvido. Conforme mencionado anteriormente, a implementação do filtro IFIR inteiramente adaptativo com o valor do coeficiente central do interpolador fixado em 1 apresenta algumas vantagens práticas. Nesse caso, a expressão de adaptação é desenvolvida de forma similar a (4.38), porém considerando agora as restrições descritas por

$$\mathbf{c}_{\mathrm{i}}^{\mathrm{T}}\mathbf{g}(n+1) = f_{\mathrm{i}} \tag{4.68}$$

com

$$\mathbf{c}_{\mathrm{i}} = [\underbrace{0 \quad \cdots \quad 0}_{L-1 \text{ elementos}} \quad 1 \quad \underbrace{0 \quad \cdots \quad 0}_{L-1 \text{ elementos}}]^{\mathrm{T}} \tag{4.69}$$

e $f_{\mathrm{i}} = 1$ uma vez que o coeficiente que sofre restrição (coeficiente central do interpolador) tem seu valor fixado em 1. Então, efetuando um desenvolvimento similar ao realizado para obtenção de (3.117) e (4.38), a seguinte expressão para adaptação do interpolador é obtida:

$$\mathbf{g}(n+1) = \mathbf{P}_{\mathrm{i}}\mathbf{g}(n) + 2\mu_1 e(n)\mathbf{P}_{\mathrm{i}}\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) + \mathbf{c}_{\mathrm{i}} \tag{4.70}$$

com $\mathbf{P}_{\mathrm{i}} = \mathbf{I} - \mathbf{c}_{\mathrm{i}}\mathbf{c}_{\mathrm{i}}^{\mathrm{T}}$ de forma similar a (3.118). Adotando as aproximações consideradas para obtenção de (4.66), obtém-se

$$\mathbf{g}(n+1) = \mathbf{P}_{\mathrm{i}}\mathbf{g}(n) + 2\mu_1 e(n)\mathbf{P}_{\mathrm{i}}\hat{\underset{\sim}{\mathbf{x}}}'(n) + \mathbf{c}_{\mathrm{i}}. \tag{4.71}$$

Analisando (4.71) e considerando as estruturas de $\mathbf{P}_{\mathrm{i}}$ e $\mathbf{c}_{\mathrm{i}}$, observa-se que tal expressão corresponde à adaptação de todos os coeficientes de $\mathbf{g}(n)$ à exceção do coeficiente central, que tem seu valor fixado em 1.

4.3.3 Adaptação usando o Algoritmo NLMS

Para o caso do algoritmo NLMS, de forma análoga a (4.46) e (4.58) as seguintes expressões são obtidas:

$$\mathbf{g}(n+1) = \mathbf{g}(n) + \frac{\alpha_1}{\left\|\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right\|^2 + \psi_1} e(n)\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) \tag{4.72}$$

e

$$\mathbf{w}_{\mathrm{s}}(n+1) = \mathbf{P}\mathbf{w}_{\mathrm{s}}(n) + \frac{\alpha_2}{\left\|\mathbf{P}\mathbf{G}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right\|^2 + \psi_2} e(n)\mathbf{P}\mathbf{G}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n). \tag{4.73}$$

Ainda, de forma análoga a (4.59) e (4.60), as seguintes expressões são obtidas considerando as aproximações descritas anteriormente:

$$\mathbf{g}(n+1) = \mathbf{g}(n) + \frac{\alpha_1}{\left\|\hat{\underset{\sim}{\mathbf{x}}}'(n)\right\|^2 + \psi_1} e(n)\hat{\underset{\sim}{\mathbf{x}}}'(n) \tag{4.74}$$

e

$$\mathbf{w}_{\mathrm{s}}(n+1) = \mathbf{P}\mathbf{w}_{\mathrm{s}}(n) + \frac{\alpha_2}{\left\|\mathbf{P}\tilde{\underset{\sim}{\mathbf{x}}}'(n)\right\|^2 + \psi_2} e(n)\mathbf{P}\tilde{\underset{\sim}{\mathbf{x}}}'(n). \tag{4.75}$$

A expressão para adaptação do interpolador com coeficiente central fixado em 1 é obtida de forma análoga a (4.58) e (4.73), ou seja, minimizando a norma euclidiana de

$$\delta\mathbf{g}(n+1) = \mathbf{g}(n+1) - \mathbf{g}(n) \tag{4.76}$$

sujeito às restrições descritas por

$$\mathbf{g}(n+1)\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) = d(n) \tag{4.77}$$

e (4.68). Assim, a função custo é dada por

$$J_{\mathrm{i}}(n) = \left\|\delta\mathbf{g}(n+1)\right\|^2 + \theta_{1\mathrm{i}}(n)\left[d(n) - \mathbf{g}(n+1)\mathbf{W}_{\mathrm{s}}^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right] + \theta_{2\mathrm{i}}(n)\left[\mathbf{c}_{\mathrm{i}}^{\mathrm{T}}\mathbf{g}(n+1) - f_{\mathrm{i}}\right]. \tag{4.78}$$

com $\theta_{1i}(n)$ e $\theta_{2i}(n)$ representando multiplicadores de Lagrange escalares. Calculando o gradiente de (4.78) com respeito a $\mathbf{g}(n+1)$, igualando a expressão resultante a zero e isolando $\mathbf{g}(n+1)$, obtém-se

$$\mathbf{g}(n+1) = \mathbf{g}(n) + \frac{1}{2}\theta_{1i}(n)\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) - \frac{1}{2}\theta_{2i}(n)\mathbf{c}_i . \tag{4.79}$$

Substituindo (4.79) em (4.68), considerando $\mathbf{c}_i^{\mathrm{T}}\mathbf{c}_i = 1$ e isolando $\theta_{2i}(n)$, obtém-se

$$\theta_{2i}(n) = 2\mathbf{c}_i^{\mathrm{T}}\left[\mathbf{g}(n) + \frac{1}{2}\theta_{1i}(n)\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right] + \mathbf{c}_i . \tag{4.80}$$

Agora, substituindo (4.80) em (4.79), aplicando a expressão resultante em (4.77) e isolando $\theta_{1i}(n)$, obtém-se

$$\theta_{1i}(n) = \frac{2\left[d(n) - \mathbf{g}^{\mathrm{T}}(n)\mathbf{P}_i\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) - \mathbf{c}_i^{\mathrm{T}}\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right]}{\left\|\mathbf{P}_i\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right\|^2}. \tag{4.81}$$

Note que, nesse caso, $\mathbf{g}^{\mathrm{T}}(n+1)$ não é igual a $\mathbf{g}^{\mathrm{T}}(n+1)\mathbf{P}_i$ pois esse produto resulta em zerar o coeficiente central de $\mathbf{g}(n+1)$, sendo que o valor de tal coeficiente é igual a 1. No entanto, tem-se $\mathbf{g}^{\mathrm{T}}(n+1) = \mathbf{g}^{\mathrm{T}}(n+1)\mathbf{P}_i + \mathbf{c}_i$ visto que $\mathbf{c}_i$ é um vetor com o elemento correspondente ao coeficiente central de $\mathbf{g}(n+1)$ igual a 1. Assim, (4.81) pode ser escrita como

$$\theta_{1i}(n) = \frac{2\left[d(n) - \mathbf{g}^{\mathrm{T}}(n)\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right]}{\left\|\mathbf{P}_i\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right\|^2} = \frac{2e(n)}{\left\|\mathbf{P}_i\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right\|^2}. \tag{4.82}$$

Agora, substituindo (4.80) e (4.82) em (4.79), a seguinte expressão para adaptação do interpolador com coeficiente central fixado em 1 é obtida. Assim,

$$\mathbf{g}(n+1) = \mathbf{P}_i\mathbf{g}(n) + \frac{\alpha_1}{\left\|\mathbf{P}_i\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n)\right\|^2 + \psi_1}e(n)\mathbf{P}_i\mathbf{W}_s^{\mathrm{T}}(n)\mathbf{T}^{\mathrm{T}}\mathbf{x}(n) + \mathbf{c}_i . \tag{4.83}$$

Adotando as aproximações previamente discutidas, a partir de (4.83), a seguinte expressão é obtida:

$$\mathbf{g}(n+1) = \mathbf{P}_i\mathbf{g}(n) + \frac{\alpha_1}{\left\|\mathbf{P}_i\underset{\sim}{\hat{\mathbf{x}}}'(n)\right\|^2 + \psi_1}e(n)\mathbf{P}_i\underset{\sim}{\hat{\mathbf{x}}}'(n) + \mathbf{c}_i . \tag{4.84}$$

É importante ressaltar que as expressões acima são válidas para adaptação tanto do filtro IFIR com interpolador na entrada quanto o filtro IFIR com interpolador na saída. Com vistas à complexidade computacional, as características descritas na seção anterior

também são válidas para os filtros IFIR com remoção do efeito de borda. Assim, a estrutura inteiramente adaptativa de menor complexidade é obtida utilizando a estrutura IFIR com interpolador na saída.

4.3.4 Simulações

Com o objetivo de avaliar o desempenho dos filtros IFIR inteiramente adaptativos sem efeito de borda, alguns resultados de simulação são apresentados nesta seção. Todas as implementações de filtros IFIR inteiramente adaptativos utilizadas são realizadas com as expressões simplificadas, visto que tais implementações são as mais indicadas para aplicações práticas. São considerados problemas de identificação de sistemas [1] e o desempenho dos diferentes filtros são avaliados em função do erro quadrático médio (EQM). Os filtros adaptativos utilizados nas simulações são os seguintes: FIR adaptativo convencional (AFIR); IFIR adaptativo com interpolador fixo (AIFIR); IFIR adaptativo sem efeito de borda (ABIFIR); IFIR inteiramente adaptativo [FAIFIR, equações (4.42) e (4.43) para o LMS e (4.59) e (4.60) para o NLMS]; IFIR inteiramente adaptativo sem efeito de borda [FABIFIR, equações (4.66) e (4.67) para o LMS e (4.74) e (4.75) para o NLMS]; e IFIR inteiramente adaptativo sem efeito de borda modificado com coeficiente central do interpolador fixado em 1 [MFABIFIR, equações (4.71) e (4.67) para o LMS e (4.84) e (4.75) para o NLMS]. É importante ressaltar que as duas diferentes implementações do filtro IFIR inteiramente adaptativo sem efeito de borda (FABIFIR e MFABIFIR) são consideradas nas simulações para comparação de desempenho. Para todos os casos, o ruído de medição gaussiano, adicionado a $d(n)$, apresenta variância $\sigma_v^2 = 10^{-8}$ $(\text{SNR} = 80\,\text{dB})$. O fator de interpolação utilizado para os filtros IFIR adaptativos e inteiramente adaptativos é $L = 2$ em todos os exemplos, com o interpolador fixo do filtro AIFIR dado por $\mathbf{g} = [0,5\ \ 1,0\ \ 0,5]^{\mathrm{T}}$ e o interpolador variável dos filtros FAIFIR, FABIFIR e MFABIFIR inicializado com $\mathbf{g}(0) = [0\ \ 1\ \ 0]^{\mathrm{T}}$.

Exemplo 4.3.4.1: A planta a ser identificada neste exemplo apresenta uma resposta ao impulso exponencial decrescente e tamanho de memória $N = 11$, conforme ilustrado na Figura 4.11. O sinal de entrada é branco gaussiano com variância unitária. Os filtros adaptativos são adaptados usando o algoritmo LMS com $\mu = \mu_1 = \mu_2 = 0,5\mu_{\max}$, onde $\mu_{\max}$ é o valor máximo de μ para o qual o algoritmo é ainda estável (obtido experimentalmente). Os resultados obtidos são apresentados na Figura 4.12, a partir da

qual se verifica o excelente desempenho em regime permanente (EQM idêntico ao nível do ruído) dos filtros FABIFIR e MFABIFIR. Ainda, é importante observar que o filtro MFABIFIR apresenta uma melhor taxa de convergência do que o FABIFIR, ilustrando a efetividade da implementação com o valor do coeficiente central do interpolador fixado em 1.

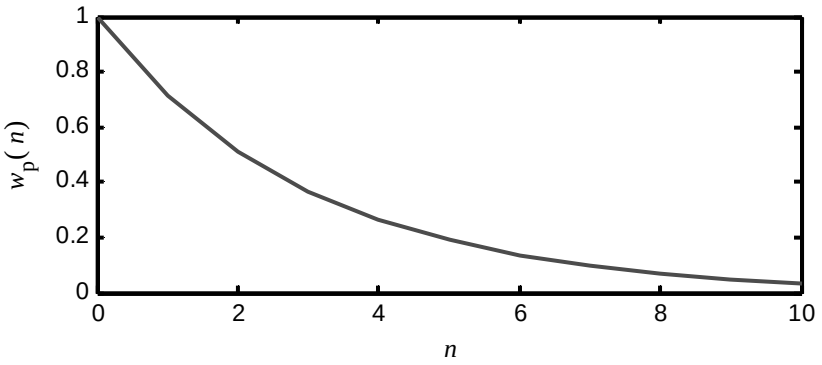


Figura 4.11. Resposta ao impulso da planta para o Exemplo *4.3.4.1*.

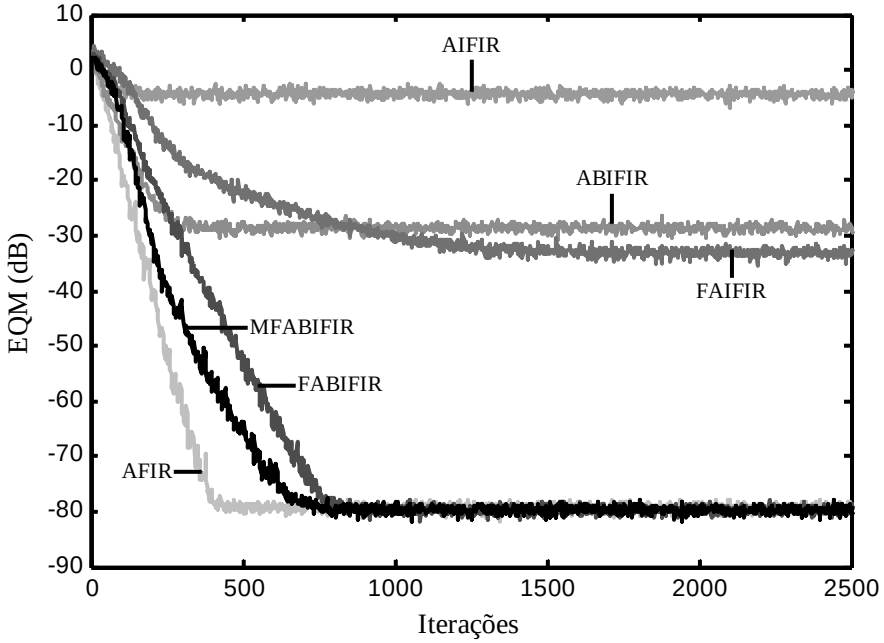


Figura 4.12. Curvas de erro quadrático médio obtidas para o Exemplo *4.3.4.1* (média de 100 realizações).

Exemplo 4.3.4.2: Para este exemplo, a planta a ser identificada apresenta novamente uma resposta ao impulso exponencial decrescente, porém com tamanho de memória $N = 101$, conforme ilustrado na Figura 4.13. O sinal de entrada é branco gaussiano com variância unitária e os filtros adaptativos são novamente adaptados usando o algoritmo LMS com o

passo de adaptação escolhido de maneira similar ao exemplo anterior. Na Figura 4.14, os resultados obtidos são apresentados. A partir dessa figura, novamente se observa o excelente desempenho dos filtros FABIFIR e MFABIFIR tanto com relação ao valor do EQM de regime permanente quanto à taxa de convergência. Ainda, é importante ressaltar que, nesse caso, o filtro FABIFIR é o que apresenta a melhor taxa de convergência.

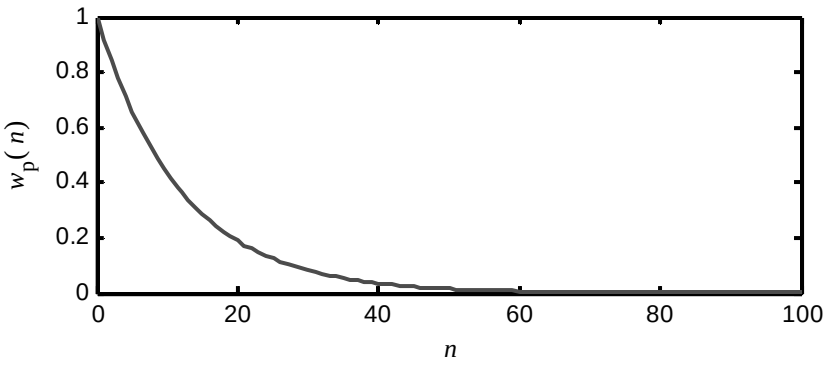


Figura 4.13. Resposta ao impulso da planta para o Exemplo *4.3.4.2*.

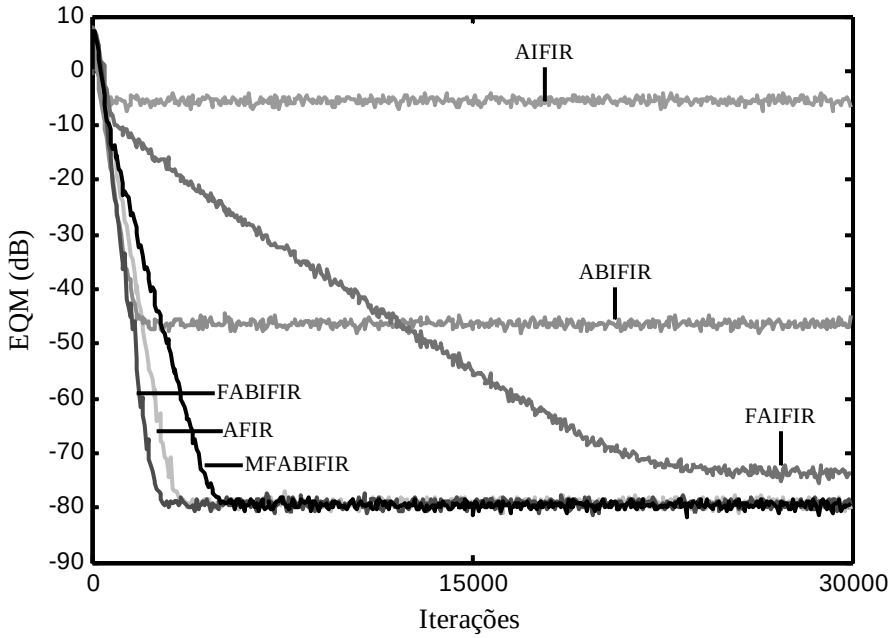


Figura 4.14. Curvas de erro quadrático médio obtidas para o Exemplo *4.3.4.2* (média de 100 realizações).

Exemplo 4.3.4.3: Neste exemplo, a planta apresenta tamanho de memória $N = 71$ e a resposta ao impulso ilustrada na Figura 4.15. O sinal de entrada é branco gaussiano com variância unitária. Os filtros adaptativos são adaptados usando o algoritmo NLMS. Os resultados obtidos com os parâmetros do NLMS $\alpha = \alpha_1 = \alpha_2 = 0,5$ e $\psi = \psi_1 = \psi_2 = 1$ são mostrados na Figura 4.12. A partir dessa figura, novamente verifica-se o excelente desempenho dos filtros IFIR inteiramente adaptativos sem efeito de borda (FABIFIR e MFABIFIR). Ainda, observa-se que, devido ao valor elevado do passo de adaptação tanto o filtro FABIFIR quanto o FAIFIR apresentam uma maior oscilação na fase transitória, enquanto o filtro MFABIFIR mostra um melhor desempenho. Dessa forma, verifica-se a efetividade da implementação com o valor do coeficiente central do interpolador fixado em 1, agora, para o caso do algoritmo NLMS. Modificando os parâmetros do algoritmo NLMS para $\alpha = \alpha_1 = \alpha_2 = 0,25$ e $\psi = \psi_1 = \psi_2 = 1$, observa-se um comportamento na fase transitória menos instável dos filtros FAIFIR e FABIFIR, e novamente o excelente desempenho do filtro MFABIFIR.

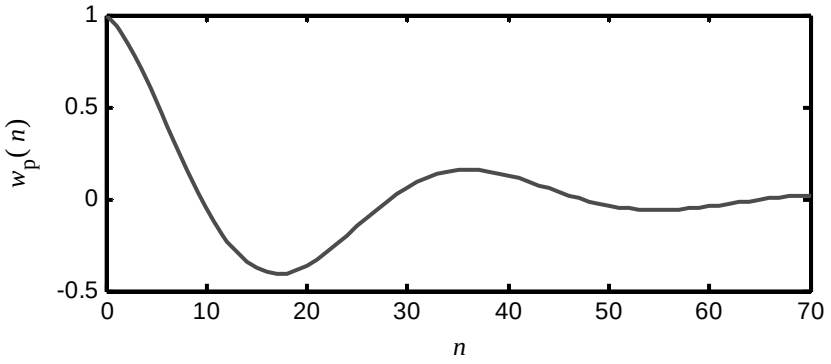


Figura 4.15. Resposta ao impulso da planta para o Exemplo *4.3.4.3*.

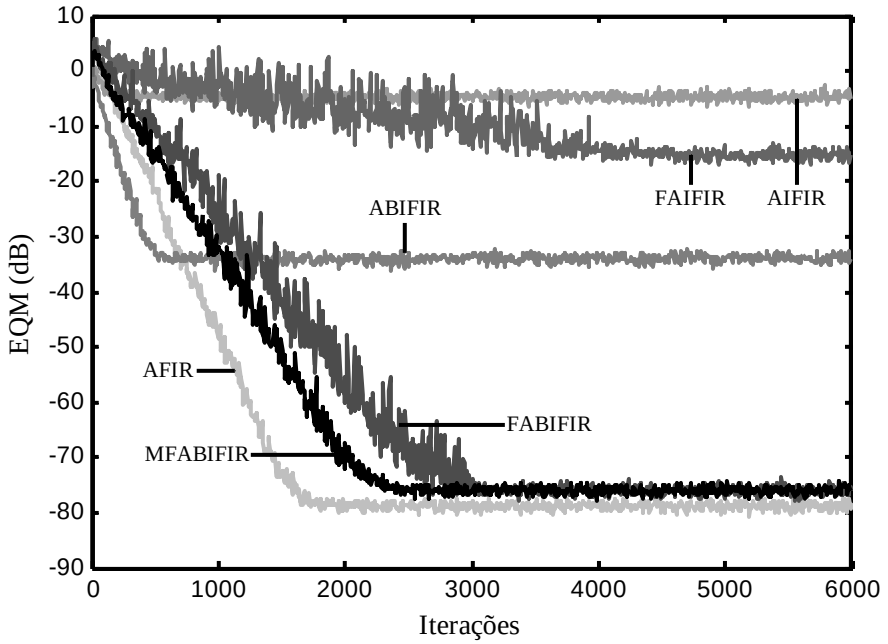


Figura 4.16. Curvas de erro quadrático médio obtidas para o Exemplo *4.3.4.3* (média de 100 realizações) com $\alpha = \alpha_1 = \alpha_2 = 0,5$ e $\psi = \psi_1 = \psi_2 = 1$.

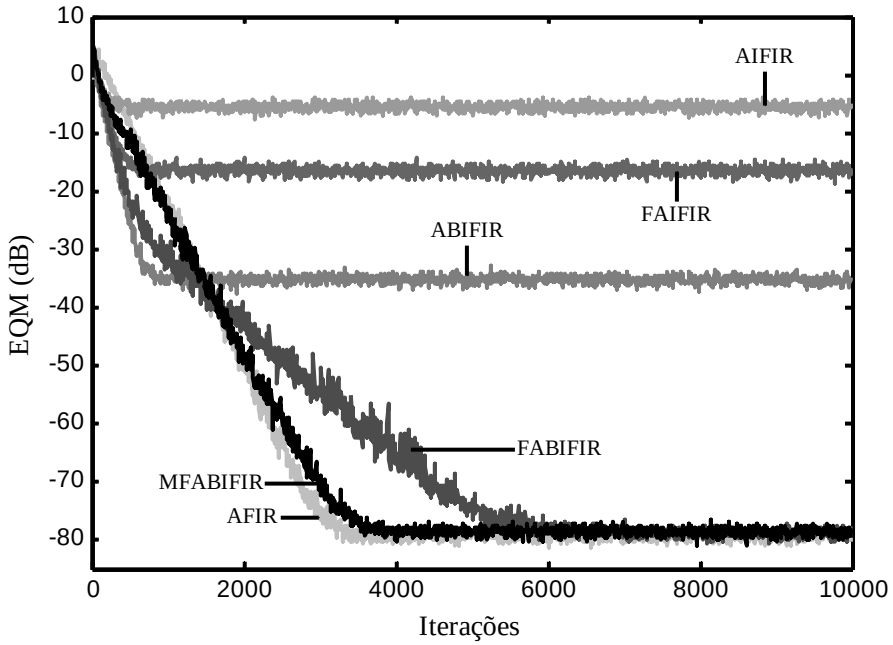


Figura 4.17. Curvas de erro quadrático médio obtidas para o Exemplo *4.3.4.3* (média de 100 realizações) com $\alpha = \alpha_1 = \alpha_2 = 0,25$ e $\psi = \psi_1 = \psi_2 = 1$.

Exemplo 4.3.4.4: A planta deste exemplo é a mesma do exemplo anterior. O sinal de entrada é colorido e obtido a partir de um processo AR dado por $x(n) = \beta\, x(n-1) + \sqrt{1-\beta^2}\, u(n)$, onde $u(n)$ é um sinal branco gaussiano com variância unitária e $\beta = 0,5$. O algoritmo adaptativo utilizado é o NLMS com $\alpha_1 = 0,25$, $\alpha_2 = 0,5$ e $\psi_1 = \psi_2 = 1$ para os filtros FAIFIR, FABIFIR e MFABIFIR; ainda, $\alpha = 0,5$ e $\psi = 1$ para os demais filtros. Os resultados obtidos estão apresentados na Figura 4.18, de onde se observa novamente o excelente desempenho das implementações inteiramente adaptativas sem efeito de borda (FABIFIR e MFABIFIR), com ligeira vantagem para a implementação com coeficiente central do interpolador com valor fixado em 1 (MFABIFIR).

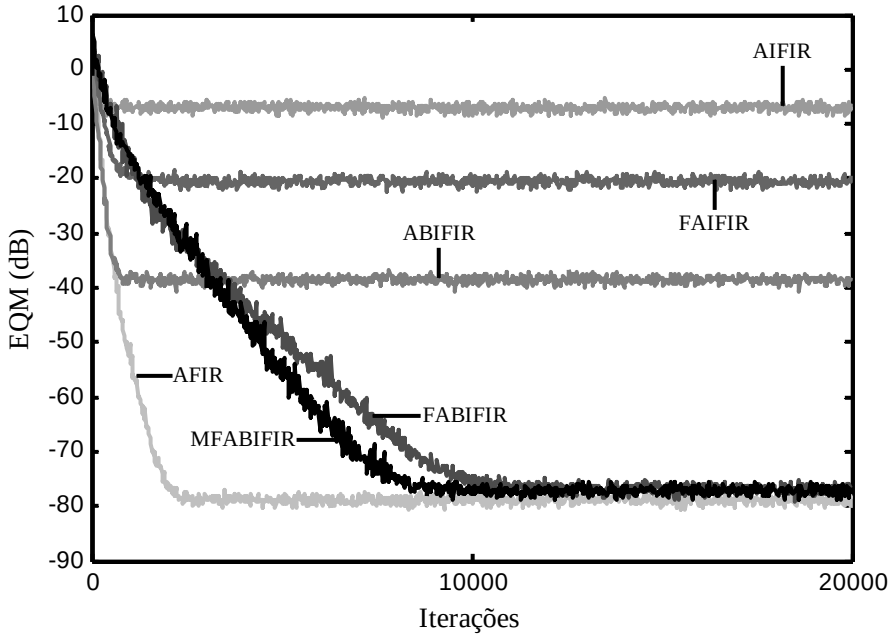


Figura 4.18. Curvas de erro quadrático médio obtidas para o Exemplo *4.3.4.4* (média de 100 realizações).

Exemplo 4.3.4.5: Para este exemplo, a planta apresenta a resposta ao impulso ilustrada na Figura 4.19 $(N = 101)$. Observe que agora os coeficientes da borda da planta são iguais a zero. O sinal de entrada é branco gaussiano com variância unitária. O algoritmo adaptativo é o NLMS com $\alpha_1 = 0,1$, $\alpha_2 = 0,5$ e $\psi_1 = \psi_2 = 10$ para os filtros FAIFIR, FABIFIR e MFABIFIR e $\alpha = 0,5$ e $\psi = 1$ para os filtros AIFIR e ABIFIR. Os resultados obtidos são mostrados na Figura 4.20, de onde se observa que, mesmo nesse caso em que o efeito de borda não é importante, o desempenho os filtros FABIFIR e MFABIFIR são os que apresentam o melhor desempenho.

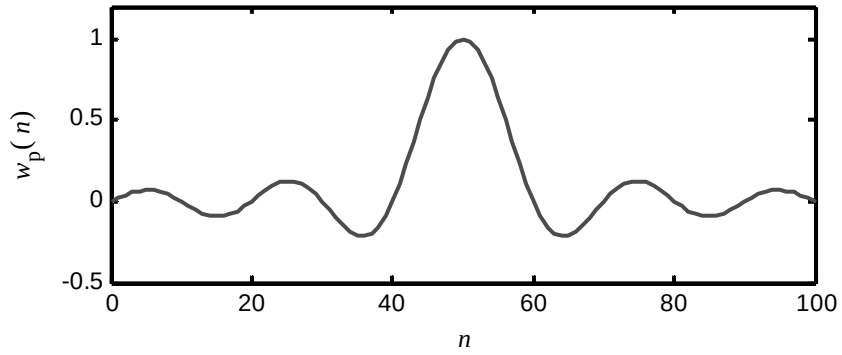


Figura 4.19. Resposta ao impulso da planta para o Exemplo *4.3.4.5*.

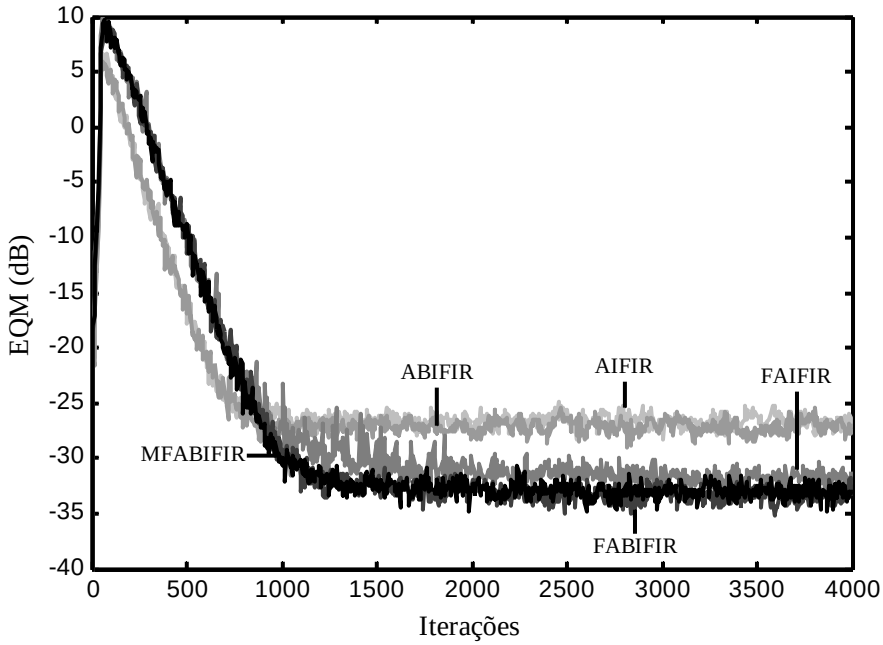


Figura 4.20. Curvas de erro quadrático médio obtidas para o Exemplo *4.3.4.5* (média de 100 realizações).

4.4 Filtros Volterra Interpolados Inteiramente Adaptativos

De forma análoga aos filtros IFIR adaptativos, uma escolha inapropriada dos coeficientes do interpolador em filtros Volterra interpolados adaptativos também pode levar a consideráveis perdas de desempenho. Nesse contexto, o uso de estratégias inteiramente adaptativas para implementação de tais filtros torna-se bastante interessante. No entanto, as dificuldades para implementação dos filtros Volterra inteiramente adaptativos são maiores por se tratarem de estruturas em cascata não-lineares em que o desenvolvimento das expressões de atualização torna-se mais complexo. Com o objetivo

de lidar com tais dificuldades, na primeira parte desta seção uma nova abordagem para descrever a relação de entrada e saída do filtro Volterra interpolado é apresentada. Tal abordagem torna possível o desenvolvimento da expressão de atualização para os coeficientes do interpolador, permitindo, com isso, a derivação do algoritmo LMS para a adaptação completa do filtro Volterra interpolado [67]. Adicionalmente, uma estratégia mista, usando o algoritmo LMS para adaptação do interpolador e o NLMS para adaptação do filtro esparso, é proposta, visando obter um algoritmo com melhor taxa de convergência como também baixo custo computacional [68]. Finalmente, implementações com remoção do efeito de borda são discutidas e resultados de simulação são apresentados para verificar o desempenho dos algoritmos propostos.

4.4.1 Sobre a Relação de Entrada e Saída do Filtro Volterra Interpolado

Considerando a estrutura do filtro Volterra interpolado apresentada na Figura 3.3, o sinal de saída do interpolador pode ser escrito como

$$\tilde{x}(n) = \mathbf{x}_{\mathrm{M}}^{\mathrm{T}}(n)\mathbf{g} \tag{4.85}$$

com

$$\mathbf{x}_{\mathrm{M}}(n) = \left[x(n)\;\; x(n-1)\;\; x(n-2)\;\; \cdots\;\; x(n-M+1)\right]^{\mathrm{T}}. \tag{4.86}$$

A partir de (4.85) e (4.86) e definindo a matriz de entrada de primeira ordem como

$$\mathbf{X}_1(n) = [\mathbf{x}_{\mathrm{M}}(n) \quad \mathbf{x}_{\mathrm{M}}(n-1) \quad \cdots \quad \mathbf{x}_{\mathrm{M}}(n-N+1)] \tag{4.87}$$

o vetor de entrada interpolado de primeira ordem do filtro Volterra interpolado [equação (3.21)] é obtido da seguinte maneira:

$$\tilde{\mathbf{x}}_1(n) = \mathbf{X}_1^{\mathrm{T}}(n)\mathbf{g}\,. \tag{4.88}$$

Assim, a saída do bloco esparso de primeira ordem [equação (3.23)] pode ser reescrita como

$$\hat{y}_1(n) = \tilde{\mathbf{x}}_1^{\mathrm{T}}(n)\mathbf{h}_{1\mathrm{s}} = \mathbf{g}^{\mathrm{T}}\mathbf{X}_1(n)\mathbf{h}_{1\mathrm{s}}. \tag{4.89}$$

Conforme apresentado na Seção 3.3.1, os vetores de entrada para os demais blocos são obtidos recursivamente fazendo

$$\tilde{\mathbf{x}}_p(n) = \tilde{\mathbf{x}}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}(n)\,. \tag{4.90}$$

Considerando (4.88), (4.90) e a regra do produto cruzado de Kronecker (*mixed-product rule* [6]), o vetor de entrada interpolado do bloco esparso de segunda ordem pode ser então escrito como

$$\tilde{\mathbf{x}}_2(n) = \left[\mathbf{X}_1^{\mathrm{T}}(n)\mathbf{g}\right] \otimes \left[\mathbf{X}_1^{\mathrm{T}}(n)\mathbf{g}\right] = \left[\mathbf{X}_1^{\mathrm{T}}(n) \otimes \mathbf{X}_1^{\mathrm{T}}(n)\right]\left[\mathbf{g} \otimes \mathbf{g}\right] = \mathbf{X}_2^{\mathrm{T}}(n)\mathbf{g}_2 \tag{4.91}$$

com $\mathbf{g}_2 = \mathbf{g} \otimes \mathbf{g}$ e $\mathbf{X}_2(n) = \mathbf{X}_1(n) \otimes \mathbf{X}_1(n)$. A partir de (4.91), a saída do bloco de segunda ordem [Equação (3.28)] pode ser expressa como

$$\hat{y}_2(n) = \mathbf{g}_2^{\mathrm{T}}\mathbf{X}_2(n)\mathbf{h}_{2\mathrm{s}}. \tag{4.92}$$

Generalizando, a saída do bloco esparso de ordem p da estrutura interpolada é dada por

$$\hat{y}_p(n) = \mathbf{g}_p^{\mathrm{T}}\mathbf{X}_p(n)\mathbf{h}_{p\mathrm{s}} \tag{4.93}$$

com

$$\mathbf{g}_p = \mathbf{g} \otimes \mathbf{g}_{p-1} \tag{4.94}$$

e

$$\mathbf{X}_p(n) = \mathbf{X}_1(n) \otimes \mathbf{X}_{p-1}(n). \tag{4.95}$$

Assim, é possível reescrever (3.18) como

$$\tilde{\mathbf{x}}_{\mathrm{V}}(n) = [\mathbf{g}^{\mathrm{T}}\mathbf{X}_1(n) \quad \mathbf{g}_2^{\mathrm{T}}\mathbf{X}_2(n) \quad \cdots \quad \mathbf{g}_P^{\mathrm{T}}\mathbf{X}_P(n)]^{\mathrm{T}} \tag{4.96}$$

e considerando que o vetor de coeficientes é

$$\mathbf{h}_{\mathrm{Vs}} = [\mathbf{h}_{1\mathrm{s}}^{\mathrm{T}} \quad \mathbf{h}_{2\mathrm{s}}^{\mathrm{T}} \quad \cdots \quad \mathbf{h}_{P\mathrm{s}}^{\mathrm{T}}]^{\mathrm{T}} \tag{4.97}$$

a saída do filtro Volterra interpolado é dada por

$$\hat{y}(n) = \sum_{p=1}^{P} \hat{y}_p(n) = \tilde{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{Vs}}. \tag{4.98}$$

É importante observar que (4.98) corresponde à representação convencional da relação de entrada e saída do filtro Volterra (Seção 2.2.2). No caso da representação triangular, comumente utilizada para fins de implementação, tem-se a relação de entrada e saída dada por

$$\hat{y}(n) = \underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{Vs}} \tag{4.99}$$

sendo, devido à equivalência entre as representações, possível escrever

$$\hat{y}(n) = \sum_{p=1}^{P} \hat{y}_p(n) = \tilde{\mathbf{x}}_{\mathrm{V}}^{\mathrm{T}}(n)\mathbf{h}_{\mathrm{Vs}} = \underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{Vs}}. \tag{4.100}$$

Adicionalmente, conforme apresentado na Seção 3.3.2.2, o filtro Volterra interpolado pode ser implementado na forma parcialmente interpolada na qual a esparsidade e interpolação são utilizadas apenas nos blocos não-lineares. Para esse caso, a saída do filtro é dada por

$$\hat{y}(n) = y_1(n) + \sum_{p=2}^{P} \hat{y}_p(n) = \underline{\tilde{\mathbf{x}}}_{\mathrm{VP}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{VPs}} \tag{4.101}$$

onde

$$y_1(n) = \mathbf{x}_1^{\mathrm{T}}(n)\mathbf{h}_1 \tag{4.102}$$

é a saída do bloco de primeira ordem convencional sem interpolação e $\underline{\mathbf{h}}_{\mathrm{VPs}}$ e $\underline{\tilde{\mathbf{x}}}_{\mathrm{VP}}(n)$ são dados por (3.33) e (3.34), respectivamente.

4.4.2 Adaptação usando o Algoritmo LMS

A partir de (4.98), o sinal de erro do algoritmo LMS para o filtro Volterra interpolado inteiramente adaptativo é escrito como

$$e(n) = d(n) - \hat{y}(n) = d(n) - \sum_{p=1}^{P} \hat{y}_p(n) = d(n) - \underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{Vs}} \tag{4.103}$$

onde $d(n)$ representa o sinal a ser estimado pelo filtro adaptativo. A equação para adaptação do filtro esparso é obtida de maneira similar à apresentada na Seção 3.6.2. Assim, a função custo é dada por

$$\hat{J}_{\mathbf{h}_{\mathrm{Vs}}}(n) = e^2(n) + \boldsymbol{\theta}^{\mathrm{T}}(n)[\mathbf{C}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vs}}(n) - \mathbf{f}] \tag{4.104}$$

o que resulta na seguinte equação de adaptação:

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \mathbf{P}\left[\underline{\mathbf{h}}_{\mathrm{Vs}}(n) + 2\mu_{\mathrm{V}} e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\right] \tag{4.105}$$

com a matriz $\mathbf{P}$ dada por (3.118). Note que, de forma análoga ao cálculo de $\hat{\mathbf{x}}(n)$ e $\tilde{\mathbf{x}}(n)$ descrito na Seção 4.2.1, o vetor $\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)$ deve ser completamente calculado a cada iteração devido à natureza agora variante no tempo do interpolador. Isso implica, por exemplo, a impossibilidade da implementação do cálculo do vetor $\underline{\tilde{\mathbf{x}}}_{\mathrm{1}}(n)$ como uma linha de atrasos, o que comumente é realizado. No entanto, assumindo uma variação lenta dos coeficientes e adotando aproximações similares às descritas na Seção 4.2 para os filtros IFIR inteiramente adaptativos [ver (4.39), (4.40) e (4.41)], uma versão aproximada de $\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)$, denotada por $\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)$, pode ser obtida com substancial redução de custo computacional. Assim, a seguinte expressão para adaptação do filtro esparso é determinada:

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \mathbf{P}\left[\underline{\mathbf{h}}_{\mathrm{Vs}}(n) + 2\mu_{\mathrm{V}} e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\right]. \tag{4.106}$$

Fazendo um desenvolvimento análogo ao realizado para obtenção de (4.106), a seguinte expressão para a adaptação do filtro Volterra parcialmente interpolado usando o algoritmo LMS é determinada:

$$\underline{\mathbf{h}}_{\mathrm{VPs}}(n+1) = \mathbf{P}\left[\underline{\mathbf{h}}_{\mathrm{VPs}}(n) + 2\mu_{\mathrm{V}} e(n)\underline{\tilde{\mathbf{x}}}_{\mathrm{PV}}(n)\right] \tag{4.107}$$

onde $\tilde{\underset{\smile}{\mathbf{x}}}_{\mathrm{PV}}(n)$ é a versão de $\tilde{\underline{\mathbf{x}}}_{\mathrm{PV}}(n)$ implementada com aproximações. É importante ressaltar ainda que existem diferenças na estrutura da matriz $\mathbf{P}$ em (4.106) e (4.107) devido às diferenças existentes entre os vetores esparsos e, conseqüentemente, entre as matrizes de restrições dos filtros interpolado e parcialmente interpolado.

De forma similar ao desenvolvimento apresentado na Seção 4.1.1, a adaptação do filtro interpolador é realizada da seguinte maneira:

$$\mathbf{g}(n+1) = \mathbf{g}(n) - \mu_{\mathrm{i}} \nabla_{\mathbf{g}} e^2(n) \tag{4.108}$$

onde μ_{i} é o passo de adaptação. A partir de (4.103) e considerando a regra da cadeia para o cálculo de derivadas, obtém-se

$$\nabla_{\mathbf{g}} e^2(n) = \frac{\partial e^2(n)}{\partial \mathbf{g}(n)} = \frac{\partial e^2(n)}{\partial e(n)} \frac{\partial e(n)}{\partial \mathbf{g}(n)}. \tag{4.109}$$

O cálculo do primeiro termo do lado direito de (4.109) resulta em

$$\frac{\partial e^2(n)}{\partial e(n)} = 2e(n) \tag{4.110}$$

enquanto o segundo termo pode ser escrito como

$$\frac{\partial e(n)}{\partial \mathbf{g}(n)} = -\frac{\partial \hat{y}(n)}{\partial \mathbf{g}(n)} = -\sum_{p=1}^{P} \frac{\partial \hat{y}_p(n)}{\partial \mathbf{g}(n)}. \tag{4.111}$$

O primeiro elemento do somatório em (4.111) é obtido considerando (4.89) e as regras de diferenciação vetorial descritas em [23]. Assim,

$$\frac{\partial \hat{y}_1(n)}{\partial \mathbf{g}(n)} = \frac{\partial \mathbf{g}^{\mathrm{T}}(n)}{\partial \mathbf{g}(n)} \mathbf{X}_1(n) \mathbf{h}_{1\mathrm{s}}(n) = \mathbf{X}_1(n) \mathbf{h}_{1\mathrm{s}}(n). \tag{4.112}$$

De maneira geral, considerando [23], a partir de (4.93) e (4.94), obtém-se

$$\frac{\partial \hat{y}_p(n)}{\partial \mathbf{g}(n)} = \frac{\partial \mathbf{g}_p^{\mathrm{T}}(n)}{\partial \mathbf{g}(n)} \mathbf{X}_p(n) \mathbf{h}_{p\mathrm{s}}(n) \tag{4.113}$$

com

$$\frac{\partial \mathbf{g}_p^{\mathrm{T}}(n)}{\partial \mathbf{g}(n)} = \mathbf{I}_{\mathrm{M}} \otimes \mathbf{g}_{p-1}^{\mathrm{T}}(n) + \mathbf{g}^{\mathrm{T}}(n) \otimes \frac{\partial \mathbf{g}_{p-1}^{\mathrm{T}}(n)}{\partial \mathbf{g}(n)} \tag{4.114}$$

onde $\mathbf{I}_{\mathrm{M}}$ é uma matriz identidade com dimensão $M \times M$. Considerando agora (4.95), (4.114) e a regra do produto cruzado de Kronecker [6], (4.113) pode ser reescrita como

$$\frac{\partial \hat{y}_p(n)}{\partial \mathbf{g}(n)} = \left[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}^{\mathrm{T}}(n) + \tilde{\mathbf{x}}_1^{\mathrm{T}}(n) \otimes \frac{\partial \mathbf{g}_{p-1}^{\mathrm{T}}(n)}{\partial \mathbf{g}(n)} \mathbf{X}_{p-1}(n) \right] \mathbf{h}_{ps}(n). \tag{4.115}$$

A partir de (4.115), o gradiente do sinal de saída do bloco de segunda ordem com respeito a $\mathbf{g}(n)$ é

$$\frac{\partial \hat{y}_2(n)}{\partial \mathbf{g}(n)} = [\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_1^{\mathrm{T}}(n) + \tilde{\mathbf{x}}_1^{\mathrm{T}}(n) \otimes \mathbf{X}_1(n)]\mathbf{h}_{2\mathrm{s}}(n). \tag{4.116}$$

Devido à característica de simetria de $\mathbf{h}_{2\mathrm{s}}$ [$h_2(j,k) = h_2(k,j)$ para qualquer valor de j e k em (3.27)], verifica-se que $[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_1^{\mathrm{T}}(n)]\mathbf{h}_{2\mathrm{s}}(n) = [\tilde{\mathbf{x}}_1^{\mathrm{T}}(n) \otimes \mathbf{X}_1(n)]\mathbf{h}_{2\mathrm{s}}(n)$ e também que (4.116) pode ser reescrita como

$$\frac{\partial \hat{y}_2(n)}{\partial \mathbf{g}(n)} = 2[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_1^{\mathrm{T}}(n)]\mathbf{h}_{2\mathrm{s}}(n). \tag{4.117}$$

A característica de simetria também é observada nos outros blocos não lineares, resultando em expressões similares. Assim, a expressão generalizada para o gradiente do sinal de saída de um bloco de ordem p em relação a $\mathbf{g}(n)$ é dada por

$$\frac{\partial \hat{y}_p(n)}{\partial \mathbf{g}(n)} = p[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}^{\mathrm{T}}(n)]\mathbf{h}_{p\mathrm{s}}(n) \tag{4.118}$$

o que permite reescrever (4.111) como

$$\frac{\partial e(n)}{\partial \mathbf{g}(n)} = -\sum_{p=1}^{P} p[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}^{\mathrm{T}}(n)]\mathbf{h}_{p\mathrm{s}}(n). \tag{4.119}$$

Finalmente, a partir de (4.108), (4.109), (4.110) e (4.119), a seguinte expressão para adaptação dos coeficientes do interpolador da estrutura Volterra interpolada é obtida:

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_{\mathrm{i}} e(n) \sum_{p=1}^{P} p[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}^{\mathrm{T}}(n)]\mathbf{h}_{p\mathrm{s}}(n) \tag{4.120}$$

com $\tilde{\mathbf{x}}_0 = 1$. Note que, uma vez que o interpolador é variante no tempo, $\tilde{\mathbf{x}}_{p-1}(n)$ deve ser inteiramente determinado a cada iteração, resultando em um alto custo computacional [por exemplo, $\tilde{\mathbf{x}}_1(n)$ não pode ser implementado como uma linha de atrasos]. No entanto, para reduzir a complexidade computacional, uma aproximação similar à adotada para calcular $\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)$ em (4.106) é aqui considerada, resultando em

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_{\mathrm{i}} e(n) \sum_{p=1}^{P} p[\mathbf{X}_1(n) \otimes \underline{\tilde{\mathbf{x}}}_{p-1}^{\mathrm{T}}(n)]\mathbf{h}_{p\mathrm{s}}(n). \tag{4.121}$$

Para o caso da implementação parcialmente interpolada, tem-se a relação de entrada e saída dada por (4.101) e como

$$\frac{\partial y_1(n)}{\partial \mathbf{g}(n)} = \frac{\partial \mathbf{x}_1^{\mathrm{T}}(n)\mathbf{h}_1}{\partial \mathbf{g}(n)} = \mathbf{0} \tag{4.122}$$

a expressão para adaptação do interpolador se torna

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_{\mathrm{i}} e(n) \sum_{p=2}^{P} p[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}^{\mathrm{T}}(n)]\mathbf{h}_{ps}(n). \tag{4.123}$$

Uma característica importante das equações de atualização dos coeficientes desenvolvidas nesta seção é o fato de a expressão para adaptação dos coeficientes do interpolador [(4.121) ou (4.123)] apresentar o vetor de entrada interpolado e o vetor de coeficientes esparso na representação Volterra convencional (Seção 2.2.2), enquanto tais vetores na expressão para adaptação do filtro esparso [(4.106) ou (4.107)] são os da representação triangular (Seção 2.2.5). Essa característica exige alguns cuidados para fins de implementação, não implicando, contudo, um aumento significativo de complexidade computacional.

4.4.3 Algoritmo LMS-NLMS

Uma outra forma de implementar filtros Volterra interpolados inteiramente adaptativos é usando o algoritmo NLMS para adaptação do filtro esparso e o LMS para adaptação do interpolador. Dessa forma, é possível obter algoritmos de fácil implementação, com carga computacional relativamente baixa e, ainda, apresentando taxas de convergência substancialmente melhores do que as obtidas usando apenas o algoritmo LMS. Assim, (4.121) é a expressão utilizada para a adaptação dos coeficientes do interpolador no caso da implementação interpolada enquanto (4.123) é a utilizada para o caso da implementação parcialmente interpolada. A expressão para a adaptação do filtro esparso, usando o algoritmo NLMS, é obtida a partir da otimização da norma euclidiana de

$$\delta\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) - \underline{\mathbf{h}}_{\mathrm{Vs}}(n) \tag{4.124}$$

sujeita às seguintes restrições:

$$\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}^{\mathrm{T}}(n)\underline{\mathbf{h}}_{\mathrm{Vs}}(n) = d(n) \tag{4.125}$$

e

$$\mathbf{C}^{\mathrm{T}}\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \mathbf{f}. \tag{4.126}$$

Adotando um procedimento similar ao apresentado na Seção 4.2.2 para os filtros IFIR inteiramente adaptativos, obtém-se

$$\underline{\mathbf{h}}_{\mathrm{Vs}}(n+1) = \mathbf{P}\underline{\mathbf{h}}_{\mathrm{Vs}}(n) + \frac{\alpha_{\mathrm{V}}}{\left\|\mathbf{P}\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n)\right\|^2 + \psi_{\mathrm{V}}} e(n)\mathbf{P}\underline{\tilde{\mathbf{x}}}_{\mathrm{V}}(n) \tag{4.127}$$

onde α_V e ψ_V são as constantes de controle do algoritmo NLMS. Considerando aproximações similares às adotadas para o desenvolvimento de (4.106), a seguinte equação de atualização dos coeficientes do filtro esparso é obtida:

$$\underline{\mathbf{h}}_{Vs}(n+1) = \mathbf{P}\underline{\mathbf{h}}_{Vs}(n) + \frac{\alpha_V}{\left\|\mathbf{P}\underline{\tilde{\mathbf{x}}}_V(n)\right\|^2 + \psi_V} e(n)\mathbf{P}\underline{\tilde{\mathbf{x}}}_V(n). \tag{4.128}$$

De forma análoga, para a implementação parcialmente interpolada tem-se

$$\underline{\mathbf{h}}_{VPs}(n+1) = \mathbf{P}\underline{\mathbf{h}}_{VPs}(n) + \frac{\alpha_V}{\left\|\mathbf{P}\underline{\tilde{\mathbf{x}}}_{VP}(n)\right\|^2 + \psi_V} e(n)\mathbf{P}\underline{\tilde{\mathbf{x}}}_{VP}(n). \tag{4.129}$$

4.4.4 Remoção do Efeito de Borda

A implementação de filtros Volterra interpolados adotando o procedimento de remoção do efeito de borda produz consideráveis ganhos de desempenho conforme discutido na Seção 3.5 deste trabalho. Tal procedimento é implementado substituindo o vetor de entrada do filtro Volterra esparso da estrutura Volterra interpolada por sua versão modificada obtida a partir de (3.95). Assim, as expressões para adaptação do filtro esparso podem ser derivadas diretamente a partir dos resultados das seções anteriores com modificação apenas no vetor de entrada do filtro esparso. Portanto, temos, para adaptação do filtro esparso da estrutura Volterra interpolada, a expressão do algoritmo LMS dada por

$$\underline{\mathbf{h}}_{Vs}(n+1) = \mathbf{P}\left[\underline{\mathbf{h}}_{Vs}(n) + 2\mu_V e(n)\underline{\tilde{\mathbf{x}}}'_V(n)\right] \tag{4.130}$$

e a expressão do algoritmo NLMS por

$$\underline{\mathbf{h}}_{Vs}(n+1) = \mathbf{P}\underline{\mathbf{h}}_{Vs}(n) + \frac{\alpha_V}{\left\|\mathbf{P}\underline{\tilde{\mathbf{x}}}'_V(n)\right\|^2 + \psi_V} e(n)\mathbf{P}\underline{\tilde{\mathbf{x}}}'_V(n). \tag{4.131}$$

Para o caso da estrutura parcialmente interpolada, tem-se

$$\underline{\mathbf{h}}_{VPs}(n+1) = \mathbf{P}\left[\underline{\mathbf{h}}_{VPs}(n) + 2\mu_V e(n)\underline{\tilde{\mathbf{x}}}'_{VP}(n)\right] \tag{4.132}$$

para o LMS e

$$\underline{\mathbf{h}}_{VPs}(n+1) = \mathbf{P}\underline{\mathbf{h}}_{VPs}(n) + \frac{\alpha_V}{\left\|\mathbf{P}\underline{\tilde{\mathbf{x}}}'_{VP}(n)\right\|^2 + \psi_V} e(n)\mathbf{P}\underline{\tilde{\mathbf{x}}}'_{VP}(n) \tag{4.133}$$

para o NLMS.

O desenvolvimento da expressão de adaptação para o interpolador é mais complexo e feito de forma similar ao que foi apresentado na Seção 4.4.2. Assim, é preciso obter uma representação da relação de entrada e saída de cada bloco do filtro Volterra interpolado

sem efeito de borda análoga a (4.93). Analisando (3.81), observa-se que tal expressão pode ser reescrita como

$$\tilde{\mathbf{x}}_1'(n) = \mathbf{X}_1'^{\mathrm{T}}(n)\mathbf{g} \tag{4.134}$$

onde $\mathbf{X}_1'(n)$ é uma matriz de Hankel não-quadrada [21] com dimensão $M \times N$, primeira coluna dada por

$$\mathbf{X}_1'^{\mathrm{T}}(n)\Big|_{\text{primeira coluna}} = [\underbrace{0 \ \cdots \ 0}_{L-1} \ x(n) \ x(n-1) \ \cdots \ x(n-M+L)]^{\mathrm{T}} \tag{4.135}$$

e última linha dada por

$$\mathbf{X}_1'^{\mathrm{T}}(n)\Big|_{\text{última linha}} = [x(n-M+L) \quad \cdots \ x(n-N+1) \ \underbrace{0 \ \cdots \ 0}_{L-1}]. \tag{4.136}$$

Os vetores (4.135) e (4.136) definem completamente $\mathbf{X}_1'(n)$, visto que uma matriz de Hankel é uma matriz com elementos iguais ao longo de qualquer diagonal com inclinação de sudoeste para nordeste [21]. Exemplificando, para um caso onde $L = 2$, $M = 3$ e $N = 5$, tem-se

$$\mathbf{X}_1'(n) = \begin{bmatrix} 0 & x(n) & x(n-1) & x(n-2) & x(n-3) \\ x(n) & x(n-1) & x(n-2) & x(n-3) & x(n-4) \\ x(n-1) & x(n-2) & x(n-3) & x(n-4) & 0 \end{bmatrix}. \tag{4.137}$$

A partir de (4.134), o sinal de saída do bloco de primeira ordem de um filtro Volterra interpolado sem efeito de borda é dado por

$$\hat{y}_1(n) = \mathbf{g}^{\mathrm{T}}\mathbf{X}_1'(n)\mathbf{h}_{1\mathrm{s}}. \tag{4.138}$$

Considerando (3.86) e (3.89), o sinal de saída do bloco de segunda ordem pode ser escrito como

$$\begin{aligned} \hat{y}_2(n) = \tilde{\mathbf{x}}_2'^{\mathrm{T}}(n)\mathbf{h}_{2\mathrm{s}} &= [\mathbf{g}^{\mathrm{T}}\mathbf{X}_1'(n) \otimes \mathbf{g}^{\mathrm{T}}\mathbf{X}_1'(n)]\mathbf{h}_{2\mathrm{s}} \\ &= (\mathbf{g}^{\mathrm{T}} \otimes \mathbf{g}^{\mathrm{T}})\left[\mathbf{X}_1'(n) \otimes \mathbf{X}_1'(n)\right]\mathbf{h}_{2\mathrm{s}} = \mathbf{g}_2^{\mathrm{T}}\mathbf{X}_2'(n)\mathbf{h}_{2\mathrm{s}} \end{aligned} \tag{4.139}$$

com $\mathbf{g}_2 = \mathbf{g} \otimes \mathbf{g}$ e $\mathbf{X}_2'(n) = \mathbf{X}_1'(n) \otimes \mathbf{X}_1'(n)$. Generalizando, tem-se

$$\hat{y}_p(n) = \mathbf{g}_p^{\mathrm{T}}\mathbf{X}_p'(n)\mathbf{h}_{p\mathrm{s}} \tag{4.140}$$

com

$$\mathbf{g}_p = \mathbf{g} \otimes \mathbf{g}_{p-1} \tag{4.141}$$

e

$$\mathbf{X}_p'(n) = \mathbf{X}_1'(n) \otimes \mathbf{X}_{p-1}'(n). \tag{4.142}$$

Considerando as definições acima, um desenvolvimento similar ao da Seção 4.4.2 pode ser obtido, resultando na expressão para adaptação dos coeficientes do interpolador do filtro Volterra interpolado sem efeito de borda. Assim,

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_{\mathrm{i}} e(n) \sum_{p=1}^{P} p[\mathbf{X}_1'(n) \otimes \tilde{\mathbf{x}}_{p-1}'^{\mathrm{T}}(n)]\mathbf{h}_{ps}(n). \tag{4.143}$$

No caso do filtro Volterra parcialmente interpolado, a adaptação do interpolador é dada por

$$\mathbf{g}(n+1) = \mathbf{g}(n) + 2\mu_{\mathrm{i}} e(n) \sum_{p=2}^{P} p[\mathbf{X}_1'(n) \otimes \tilde{\mathbf{x}}_{p-1}'^{\mathrm{T}}(n)]\mathbf{h}_{ps}(n). \tag{4.144}$$

Os filtros Volterra interpolados inteiramente adaptativos também podem ser implementados com o valor do coeficiente central do interpolador fixado em 1 de maneira similar ao procedimento apresentado na Seção 4.3 para os filtros IFIR inteiramente adaptativos. O desenvolvimento da expressão adaptativa nesse caso é feita a partir da seguinte função custo:

$$J_{\mathrm{i}}(n) = e^2(n) + \theta_{\mathrm{i}}^{\mathrm{T}}(n)\left[\mathbf{c}_{\mathrm{i}}^{\mathrm{T}}\mathbf{g}(n) - f_{\mathrm{i}}\right] \tag{4.145}$$

com $\mathbf{c}_{\mathrm{i}}$ dado por (4.69) e $f_{\mathrm{i}} = 1$. Fazendo um desenvolvimento similar aos apresentados nas Seções 4.3 e 4.4.2, a seguinte expressão é obtida para adaptação do interpolador do filtro Volterra interpolado:

$$\mathbf{g}(n+1) = \mathbf{P}_{\mathrm{i}}\mathbf{g}(n) + 2\mu_{\mathrm{i}} e(n) \sum_{p=1}^{P} p\mathbf{P}_{\mathrm{i}}[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}^{\mathrm{T}}(n)]\mathbf{h}_{ps}(n) + \mathbf{c}_{\mathrm{i}} \tag{4.146}$$

com $\mathbf{P}_{\mathrm{i}} = \mathbf{I}_{\mathrm{M}} - \mathbf{c}_{\mathrm{i}}\mathbf{c}_{\mathrm{i}}^{\mathrm{T}}$. Ainda, para o caso do filtro Volterra parcialmente interpolado, (4.146) torna-se

$$\mathbf{g}(n+1) = \mathbf{P}_{\mathrm{i}}\mathbf{g}(n) + 2\mu_{\mathrm{i}} e(n) \sum_{p=2}^{P} p\mathbf{P}_{\mathrm{i}}[\mathbf{X}_1(n) \otimes \tilde{\mathbf{x}}_{p-1}^{\mathrm{T}}(n)]\mathbf{h}_{ps}(n) + \mathbf{c}_{\mathrm{i}}. \tag{4.147}$$

4.4.5 Algoritmos e Complexidade Computacional

Os diversos algoritmos para implementação de filtros Volterra interpolados inteiramente adaptativos descritos nas Seções 4.4.2, 4.4.3 e 4.4.4 estão enumerados na Tabela 4-1. De maneira geral, o algoritmo que possui as melhores características dentre os indicados em tal tabela é a implementação parcialmente interpolada sem efeito de borda com coeficiente central do interpolador fixado em 1, identificada por XII na tabela. Isso ocorre visto que tal implementação incorpora a maioria dos aperfeiçoamentos descritos nas seções anteriores. No entanto, por se tratar de uma implementação usando os algoritmos

LMS e NLMS, o seu custo computacional é um tanto mais alto do que o das implementações usando o algoritmo LMS e também mais alto do que o das implementações interpoladas adaptativas com interpolador fixo. Para efeito de comparação, a Figura 4.21 apresenta curvas do número de operações por amostra em função do tamanho de memória requerido para implementação de versões de 2ª ordem com $L = 2$ dos algoritmos indicados por I e XII na Tabela 4-1, como também para implementações do filtro Volterra convencional adaptadas pelos algoritmos LMS e NLMS e, ainda, para o filtro Volterra interpolado adaptado pelo LMS (Seção 3.3). A complexidade computacional das demais implementações inteiramente adaptativas é próxima à das implementações I e XII cujas curvas estão apresentadas na Figura 4.21. Nessa figura, observa-se que o custo computacional das abordagens inteiramente adaptativas é maior do que o da abordagem com interpolador fixo. Porém, tal custo é ainda muito menor do que o custo das implementações convencionais do filtro Volterra.

Tabela 4-1. Algoritmos para a implementação de
filtros Volterra interpolados inteiramente adaptativos apresentados na Seção 4.4

Estruturas Inteiramente Adaptativas		Expressões de Atualização	
		Filtro Esparso	Interpolador
Algoritmo LMS			
I	Filtro Volterra interpolado	(4.106)	(4.121)
II	Filtro Volterra interpolado sem efeito de borda	(4.130)	(4.143)
III	Filtro Volterra interpolado sem efeito de borda modificado	(4.130)	(4.146)
IV	Filtro Volterra parcialmente interpolado	(4.107)	(4.123)
V	Filtro Volterra parc. interpolado sem efeito de borda	(4.131)	(4.144)
VI	Filtro Volterra parc. interp. sem efeito de borda modificado[1]	(4.131)	(4.147)
Algoritmo LMS-NLMS			
VII	Filtro Volterra interpolado	(4.128)	(4.121)
VIII	Filtro Volterra interpolado sem efeito de borda	(4.131)	(4.143)
IX	Filtro Volterra interpolado sem efeito de borda modificado	(4.131)	(4.146)
X	Filtro Volterra parcialmente interpolado	(4.129)	(4.123)
XI	Filtro Volterra parc. interpolado sem efeito de borda	(4.133)	(4.144)
XII	Filtro Volterra parc. interp. sem efeito de borda modificado[1]	(4.133)	(4.147)

[1] implementações com coeficiente central do interpolador fixado em 1.

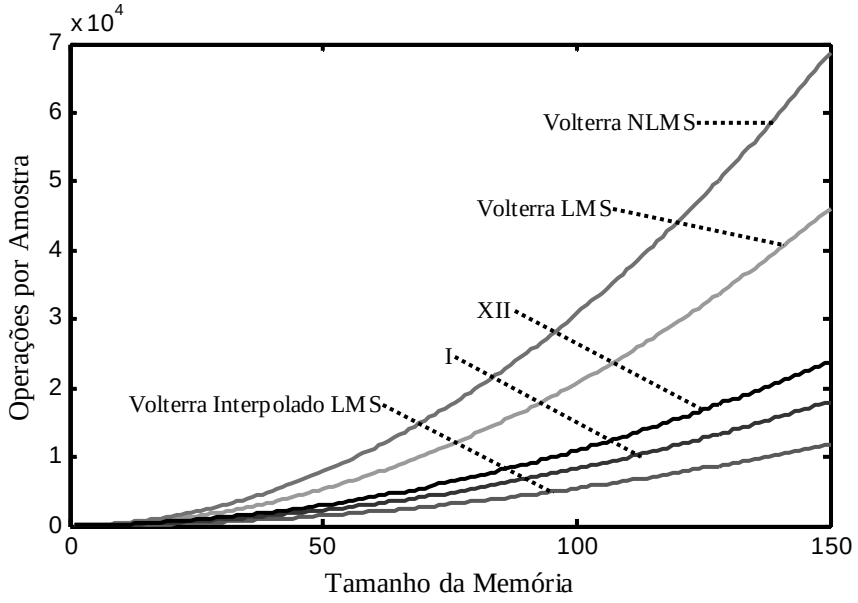


Figura 4.21. Complexidade computacional para diferentes implementações adaptativas de filtros Volterra.

4.4.6 Simulações

Nesta seção, resultados de simulação são apresentados com o objetivo de verificar o desempenho e as características das implementações inteiramente adaptativas dos filtros Volterra interpolados. Nos exemplos descritos a seguir, problemas de identificação de sistemas [1] são considerados e o desempenho dos diferentes algoritmos é avaliado em função das curvas do erro quadrático médio (EQM) obtidas por simulação de Monte Carlo (média de 100 realizações). Os filtros adaptativos utilizados nas simulações são filtro Volterra adaptado pelo algoritmo LMS (Volterra-LMS), filtro Volterra adaptado pelo NLMS (Volterra-NLMS), filtro Volterra parcialmente interpolado adaptado pelo LMS (APIV-LMS, Seção 3.3) e, ainda, as implementações dos filtros Volterra parcialmente interpoladas e inteiramente adaptativas indicadas na Tabela 4-1 por IV, V, VI, X, XII e XII. São consideradas apenas implementações parcialmente interpoladas devido ao seu melhor desempenho em comparação com as abordagens inteiramente interpoladas. Para todos os casos, o ruído de medição gaussiano, adicionado a $d(n)$, apresenta variância $\sigma_v^2 = 10^{-6}$ $(\text{SNR} = 60\,\text{dB})$. O fator de interpolação utilizado para todos os filtros interpolados é $L = 2$, com o interpolador fixo do filtro APIV-LMS dado por $\mathbf{g} = [0,5\ \ 1,0\ \ 0,5]^{\text{T}}$ e o interpolador variável dos demais filtros interpolados inicializado com $\mathbf{g}(0) = [0,5\ \ 1,0\ \ 0,5]^{\text{T}}$. Para os filtros Volterra-LMS e APIV-LMS, o passo de

adaptação escolhido é $\mu = \mu_{max}/2$ (μ_{max} é o valor máximo do passo de adaptação que ainda garante a convergência do algoritmo, obtido experimentalmente). Para o filtro Volterra-NLMS os parâmetros de controle são $\alpha = 0,5$ e $\psi = 10^{-6}$. Para as implementações interpoladas inteiramente adaptativas que usam o algoritmo LMS (I, II e III) adota-se $\mu_V = \mu_i = \mu_{max}/2$ (μ_{max} obtido experimentalmente com ambos os passos de adaptação iguais), enquanto para as implementações usando a combinação LMS-NLMS tem-se $\alpha_V = 0,5$, $\psi_V = 10$ e $\mu_i = \mu_{max}/2$ (μ_{max} obtido experimentalmente).

Exemplo 4.4.6.1: Neste exemplo, a planta é de segunda ordem com tamanho de memória $N = 11$ e coeficientes representados pelas curvas da Figura 3.9. Na Figura 4.22 são apresentadas as curvas de EQM obtidas com as implementações adaptadas pelo algoritmo LMS submetidas a um sinal branco gaussiano de entrada com variância unitária. Dessa figura, observa-se o excelente desempenho das implementações interpoladas inteiramente adaptativas, com destaque para as implementações sem efeito de borda (V e VI). Na Figura 4.23 estão apresentados resultados também obtidos com sinal de entrada branco gaussiano com variância unitária, porém para as implementações que utilizam os algoritmos NLMS e LMS-NLMS (Volterra-NLMS, X, XI e XII). Novamente o excelente desempenho das estruturas inteiramente adaptativas parcialmente interpoladas sem efeito de borda é verificado, sendo que agora tem-se uma convergência mais rápida do que a do filtro Volterra convencional adaptado pelo algoritmo NLMS com um custo computacional bem mais reduzido (veja Figura 4.21). Ainda, resultados obtidos com um sinal de entrada colorido e as implementações adaptadas pelo NLMS estão apresentados na Figura 4.24. O sinal colorido é obtido a partir de um processo AR descrito por $x(n) = \beta x(n-1) + \sqrt{1-\beta^2}u(n)$, onde $u(n)$ é um sinal branco gaussiano com variância unitária e $\beta = 0,5$. Novamente ótimos resultados são obtidos especialmente pelas implementações inteiramente adaptativas sem efeito de borda.

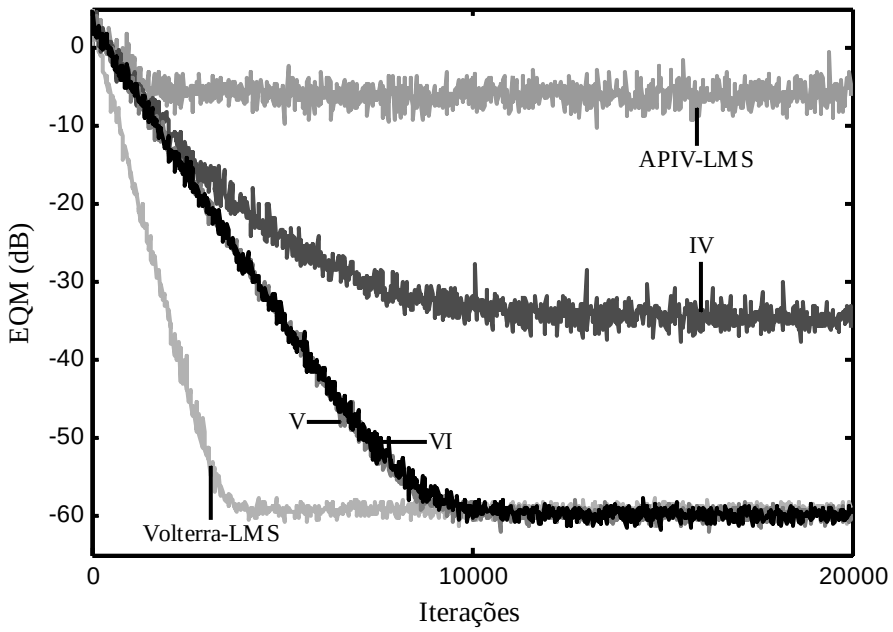


Figura 4.22. Exemplo *4.4.6.1*. Curvas de erro quadrático médio obtidas com um sinal branco de entrada (média de 100 realizações).

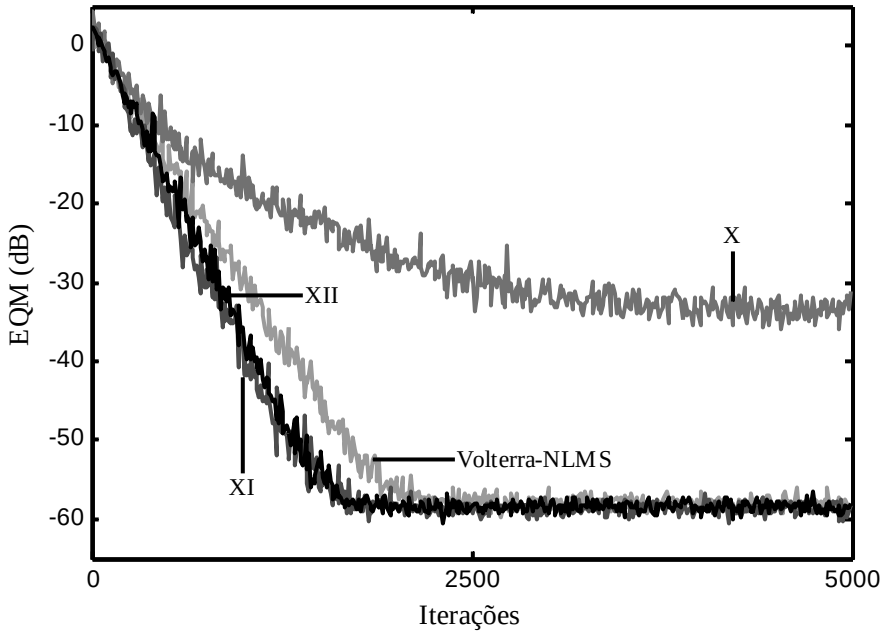


Figura 4.23. Exemplo *4.4.6.1*. Curvas de erro quadrático médio obtidas com um sinal branco de entrada (média de 100 realizações).

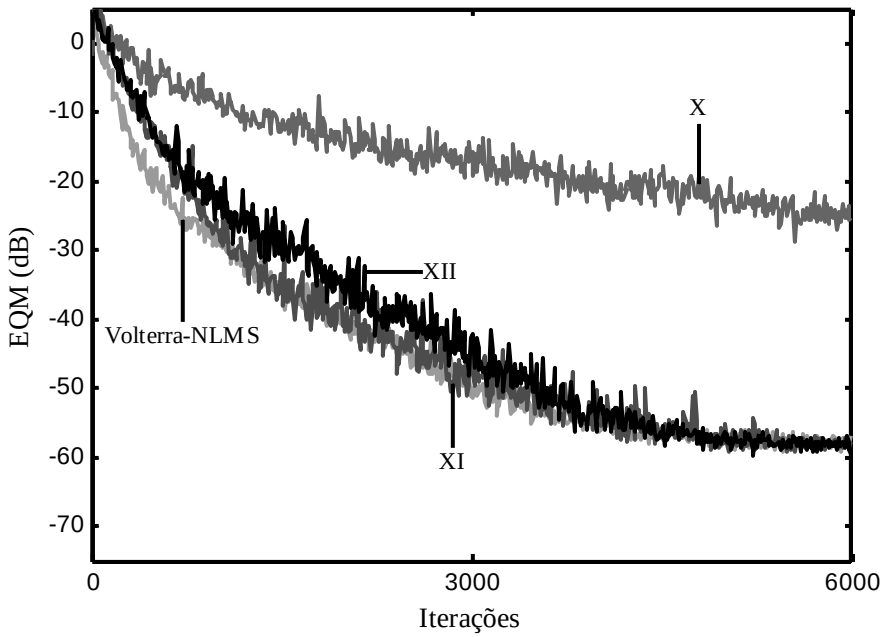


Figura 4.24. Exemplo *4.4.6.1*. Curvas de erro quadrático médio obtidas com um sinal colorido de entrada (média de 100 realizações).

Exemplo 4.4.6.2: A planta para este exemplo é a apresentada na Figura 3.15. Os resultados obtidos com um sinal branco gaussiano de variância unitária e as implementações que usam o algoritmo LMS estão apresentados na Figura 4.25. Observe que agora as implementações interpoladas inteiramente adaptativas apresentam um desempenho em regime permanente mais próximo do desempenho do filtro APIV-LMS, que possui um interpolador fixo. A razão disso é a característica da planta, que possui coeficientes pequenos nas bordas e compatibilidade com uma implementação interpolada com interpolador linear $\mathbf{g} = [0,5 \;\; 1,0 \;\; 0,5]^T$. Como regra geral, as implementações interpoladas inteiramente adaptativas apresentam um desempenho em regime permanente pelo menos igual ao das implementações interpoladas adaptativas com interpoladores fixos, sendo mais próximos do desempenho dos filtros Volterra convencionais em função das características da planta. Na Figura 4.26 estão apresentados os resultados das implementações que utilizam o algoritmo NLMS para um sinal de entrada branco gaussiano com variância unitária, enquanto na Figura 4.27 tem-se os resultados das mesmas implementações para um sinal de entrada colorido obtido de forma similar ao sinal colorido do exemplo anterior. Novamente o bom desempenho dos algoritmos propostos é verificado, com destaque para a implementação XII sem efeito de borda modificada.

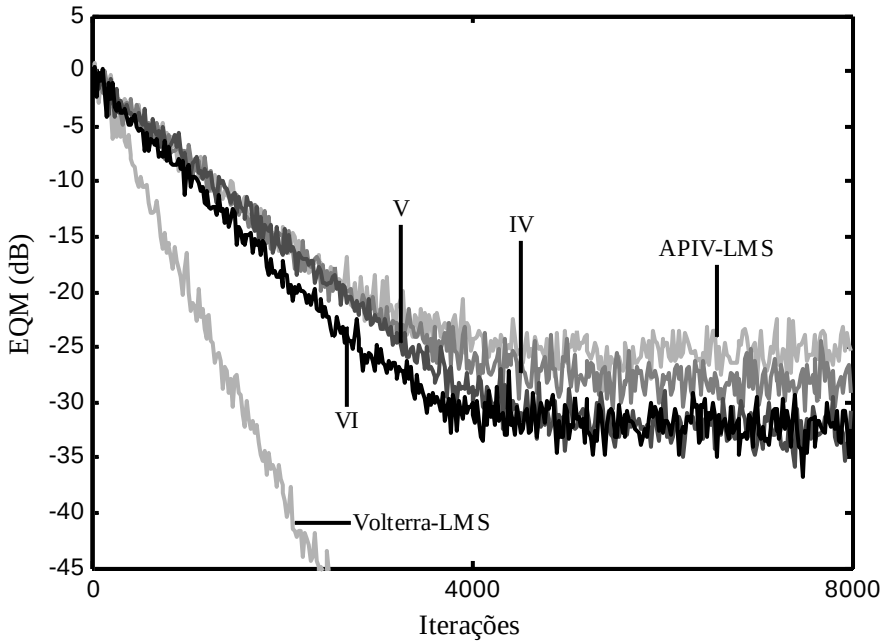


Figura 4.25. Exemplo *4.4.6.2*. Curvas de erro quadrático médio obtidas com um sinal branco de entrada (média de 100 realizações).

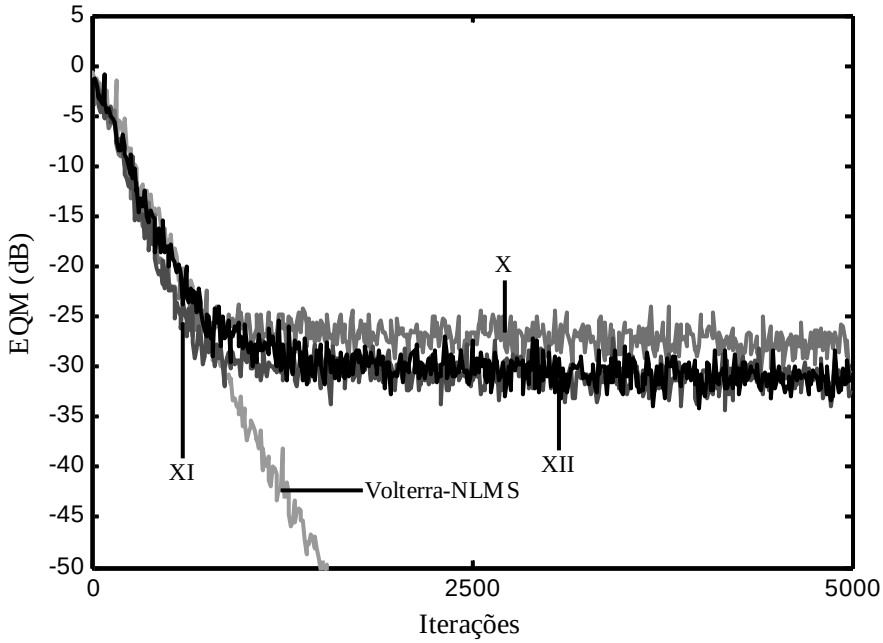


Figura 4.26. Exemplo *4.4.6.2*. Curvas de erro quadrático médio obtidas com um sinal branco de entrada (média de 100 realizações).

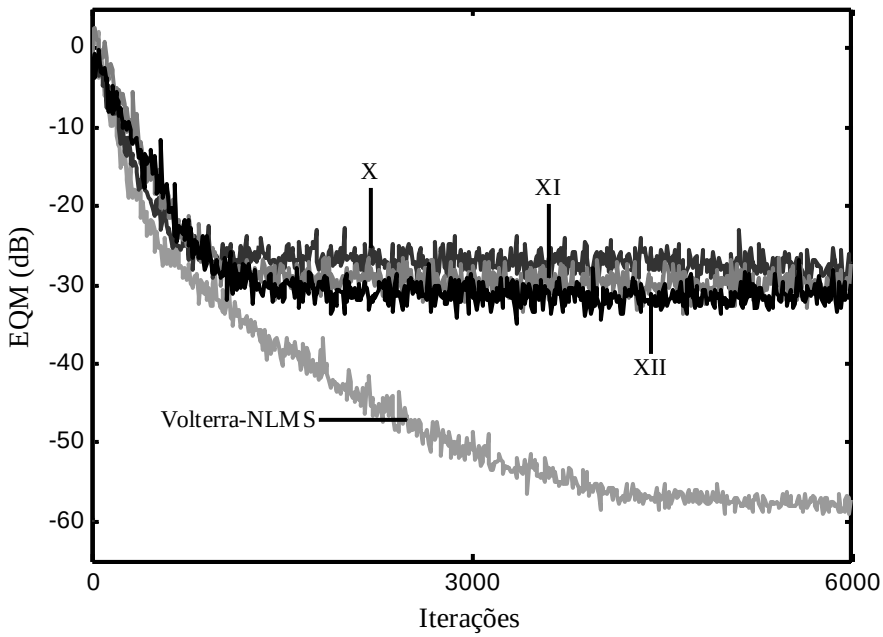


Figura 4.27. Exemplo *4.4.6.2*. Curvas de erro quadrático médio obtidas com um sinal colorido de entrada (média de 100 realizações).

4.5 Considerações

Este capítulo foi dedicado à implementação inteiramente adaptativa dos filtros IFIR e Volterra interpolados. O objetivo foi desenvolver implementações interpoladas mais eficientes nas quais a escolha dos coeficientes do interpolador não tenha grande influência no desempenho em regime permanente do filtro. Como resultado, foram apresentadas diversas abordagens originais para implementação dos filtros interpolados inteiramente adaptativos aliando baixo custo computacional com importantes ganhos de desempenho [59]-[60], [65]-[68]. Resultados de simulação permitiram verificar o bom desempenho das estruturas interpoladas inteiramente adaptativas, atestando assim o seu grande potencial de aplicação.

CAPÍTULO 5

Aplicações em Cancelamento de Eco de Linha

Um problema recorrente em comunicações através de linhas telefônicas é a presença de sinais de eco que, no caso de comunicação de voz em grandes distâncias (comunicação via satélite, por exemplo), podem resultar em desconforto e prejuízos para a fluência das conversações. Além disso, o eco tem um impacto bastante negativo no desempenho de sistemas de voz sobre IP (VoIP) [69] como também em sistemas que recorrem de tecnologia de reconhecimento de fala [70], os quais têm se popularizado significativamente nos últimos anos. Além das comunicações via voz, as transmissões de dados através de linhas telefônicas também são afetadas pela presença de eco [71]. Tal problema ganhou maior importância nas últimas décadas com a difusão dos sistemas DSL (*digital subscriber line* [72]) e a conseqüente transmissão de dados a taxas elevadas pela rede telefônica. Nesse contexto, a utilização de sistemas de cancelamento de eco torna-se indispensável para garantir comunicações confiáveis e de boa qualidade.

Os ecos encontrados em sistemas telefônicos são classificados como eco de linha e eco acústico. O primeiro é oriundo principalmente do descasamento de impedância na conversão de quatro para dois fios realizada pela híbrida [1], enquanto o segundo tem origem em sistemas de viva-voz, produzindo um indesejável acoplamento acústico entre o alto-falante e o microfone do aparelho telefônico [1]. Cada tipo de eco possui características específicas, causando diferentes impactos sobre as comunicações telefônicas. Em comum, ambos apresentam respostas relativamente longas [1] e, com certa freqüência, é constatada a presença de elementos não-lineares (alto-falantes ou híbridas, por exemplo) provocando distorções no caminho do eco [53], [71], [73]. Considerando que os sistemas de cancelamento de eco constituem-se de filtros que devem se ajustar às condições do sistema, é clara a necessidade de se utilizar filtros adaptativos lineares e não-lineares exibindo um grande número de coeficientes que dificulta sobremaneira sua implementação.

Neste capítulo, o problema de cancelamento de eco de linha em comunicações telefônicas é abordado. De maneira geral, canceladores de eco baseados em

implementações interpoladas de filtros FIR e filtros Volterra são aqui desenvolvidos visando obter soluções com bom desempenho e baixo custo computacional. Assim, o problema de cancelamento de eco de linha em sistemas DSL é inicialmente tratado, seguido pelo cancelamento de eco de linha linear e não-linear em comunicações de voz.

5.1 Filtros FIR-IFIR sem Efeito de Borda para o Cancelamento de Eco de Linha em Sistemas DSL

A reposta ao impulso de eco típica encontrada em sistemas DSL possui uma porção inicial curta e com variações rápidas e uma parte final longa e que varia lentamente denominada cauda [74]-[75]. Um exemplo de resposta desse tipo é ilustrado na Figura 5.1. Geralmente, filtros FIR adaptativos são utilizados para cancelamento de eco em sistemas DSL. Porém, como as respostas ao impulso de eco são usualmente muito longas, o filtro FIR utilizado para o cancelamento de eco possui tipicamente centenas de coeficientes [75]. Dessa forma, a complexidade computacional do cancelador de eco torna-se muito elevada, podendo inviabilizar sua implementação.

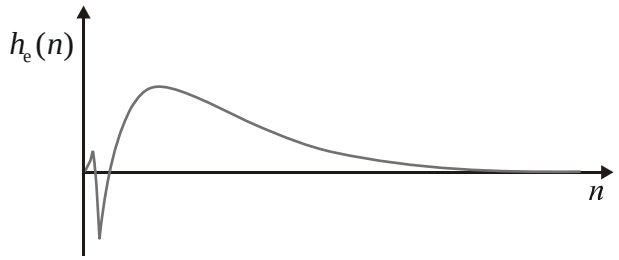


Figura 5.1. Resposta ao impulso de eco típica de uma híbrida em sistemas DSL.

Para reduzir a carga computacional dos canceladores de eco em sistemas DSL, abordagens baseadas em estruturas híbridas FIR-IFIR vêm sendo consideradas com sucesso [74]-[75]. A proposta, em tais abordagens, é usar um filtro FIR adaptativo para o cancelamento do eco associado à parte inicial da resposta ao impulso de eco e um filtro IFIR adaptativo para o cancelamento do eco associado à parte final. Trata-se de uma estratégia interessante, visto que a parte final da resposta ao impulso de eco típica de sistemas DSL apresenta variações lentas, conforme mostrado na Figura 5.1. Essa característica particular implica um maior grau de correlação entre amostras próximas e/ou vizinhas, permitindo uma modelagem eficiente com o uso de um filtro IFIR. O diagrama

de blocos de uma estrutura FIR-IFIR é mostrado na Figura 5.2, onde $\mathbf{w}_h$ representa um filtro FIR, $\mathbf{w}_t$, um filtro IFIR e z^{-D}, um atraso de D amostras.

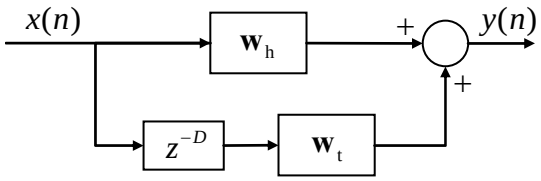


Figura 5.2. Estrutura de blocos do cancelador de eco FIR-IFIR.

A abordagem FIR-IFIR foi originalmente proposta em [74], sendo que, em tal trabalho, apenas o cancelamento do eco relacionado à parte final da resposta ao impulso de eco, realizado pelo filtro IFIR, é discutido. No entanto, as dificuldades decorrentes da associação dos filtros FIR e IFIR em paralelo para a síntese do cancelador de eco não são discutidas em [74]. A principal dificuldade está relacionada ao efeito de borda oriundo da operação de convolução entre o interpolador e o filtro esparso que compõem a estrutura de um filtro IFIR. Conforme descrito anteriormente, os valores dos coeficientes da estrutura equivalente de um filtro IFIR correspondentes ao efeito de borda são funções dos valores do primeiro ou do último coeficiente do filtro esparso e dos coeficientes do interpolador [ver (4.61)]. Assim, quando interpoladores lineares são utilizados, a resposta ao impulso equivalente de um filtro IFIR apresenta uma forma de rampa em seu início e seu final [75]. Como conseqüência, a associação do filtro FIR com o filtro IFIR, conforme sugerido em [74], conduzirá a uma resposta ao impulso na forma ilustrada na Figura 5.3. Nessa figura, observa-se uma importante diferença entre a resposta ao impulso de eco a ser modelada (curva em cinza) e a obtida com os filtros FIR (linha pontilhada) e IFIR (linha tracejada), o que leva a uma significativa degradação de desempenho do cancelador de eco FIR-IFIR.

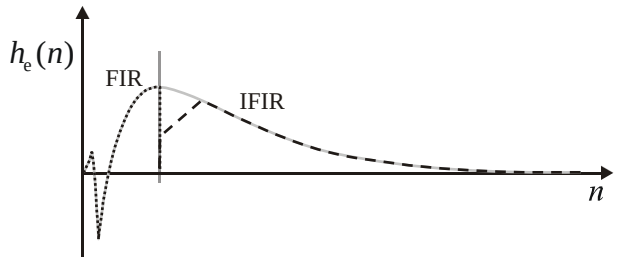


Figura 5.3. Resposta ao impulso do filtro FIR-IFIR proposto em [74].

Uma estratégia para mitigar o impacto da distorção causada pelo efeito de borda no desempenho do filtro FIR-IFIR é proposta em [75]. Tal estratégia, ilustrada na Figura 5.4,

consiste em aumentar o tamanho de memória do filtro FIR e impor uma superposição de parte de sua resposta ao impulso com a parte inicial da resposta do filtro IFIR correspondente ao efeito de borda. Assim, alguns coeficientes do filtro FIR são utilizados com o objetivo de compensar a distorção causada pelo efeito de borda do filtro IFIR. Apesar do bom desempenho obtido com a superposição das respostas ao impulso dos filtros FIR e IFIR, um aumento de complexidade computacional é verificado em conseqüência do aumento no tamanho de memória do filtro FIR. Além disso, observa-se, a partir dos resultados apresentados em [75], o uso de interpoladores com tamanhos de memória estendido de forma a garantir um desempenho satisfatório do algoritmo. Por exemplo, para os casos apresentados em [75], nos quais o fator de interpolação é $L = 4$, um interpolador com 23 coeficientes foi utilizado, enquanto usualmente 7 coeficientes seriam necessários [ver (3.15)]. Esse aumento do tamanho de memória do interpolador também implica aumento de carga computacional. Adicionalmente, em [75], menciona-se que uma redução na velocidade de convergência do algoritmo adaptativo é verificada em conseqüência da superposição das respostas ao impulso dos filtros FIR e IFIR. Tal redução é amenizada fixando em zero alguns dos coeficientes do filtro FIR na região de sua resposta ao impulso onde ocorre a superposição com a resposta do filtro IFIR [75].

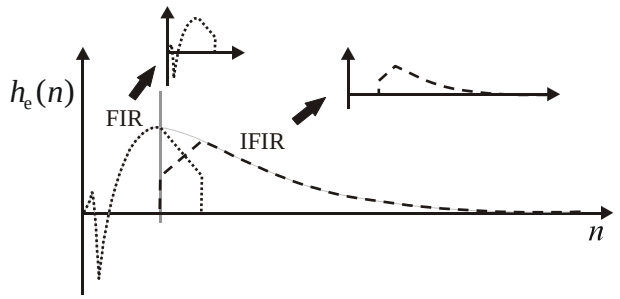


Figura 5.4. Resposta ao impulso do filtro FIR-IFIR proposto em [75].

Contribuindo neste contexto, uma nova abordagem para o cancelamento de eco em sistemas DSL é apresentada a seguir. A idéia central aqui é implementar o filtro IFIR da estrutura FIR-IFIR usando a implementação IFIR com remoção de efeito de borda (BIFIR - Seção 3.5). Tal abordagem, denominada FIR-BIFIR, permite eliminar a distorção causada pelo efeito de borda sem a necessidade da superposição dos coeficientes dos filtros FIR e IFIR, conforme ilustrado na Figura 5.5. Como conseqüência, obtém-se um cancelador de eco para sistemas DSL com menor custo computacional e também com melhor desempenho.

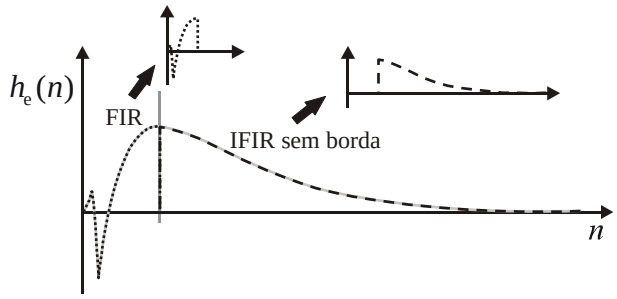


Figura 5.5. Resposta ao impulso da abordagem FIR-BIFIR proposta neste trabalho.

5.1.1 Abordagem FIR-BIFIR

O diagrama de blocos do filtro FIR-BIFIR é idêntico ao do filtro FIR-IFIR mostrado na Figura 5.2. O filtro FIR $\mathbf{w}_h$ possui tamanho de memória igual ao atraso de D amostras. Assim, seu vetor de coeficientes é dado por

$$\mathbf{w}_h = [w_h(0)\ \ w_h(1)\ \ w_h(2)\ \cdots w_h(D-1)]^T \tag{5.1}$$

e o vetor de entrada correspondente por

$$\mathbf{x}_h(n) = [x(n)\ \ x(n-1)\ \ x(n-2)\ \cdots\ x(n-D+1)]^T \tag{5.2}$$

o que resulta na seguinte relação de entrada e saída:

$$y_h(n) = \mathbf{w}_h^T \mathbf{x}_h(n). \tag{5.3}$$

Conforme mencionado anteriormente, a principal diferença entre as estruturas FIR-IFIR e FIR-BIFIR está na forma de implementação do filtro IFIR $\mathbf{w}_t$. No caso da abordagem FIR-BIFIR, $\mathbf{w}_t$ é implementado com a remoção do efeito de borda proposta na Seção 3.5. Assim, o vetor de coeficientes esparso é escrito como

$$\mathbf{w}_t = \{w_t(0)\ \ 0\ \cdots\ w_t(L)\ \ 0\ \cdots\ w_t(2L)\ \ 0\ \cdots\ w_t[(N_{ts}-1)L]\ \ 0\ \cdots\ 0\}^T \tag{5.4}$$

onde $N_{ts} = [(N-D-1)/L]+1$ denota o número de coeficientes diferentes de zero em (5.4) e L, o fator de interpolação (ou esparsidade). O vetor de entrada interpolado do filtro esparso é então dado por

$$\tilde{\mathbf{x}}_t'(n) = \mathbf{G}^T \mathbf{T}^T \mathbf{x}_t(n). \tag{5.5}$$

Em (5.5), $\mathbf{G}$ e $\mathbf{T}$ são, respectivamente, as matrizes de interpolação e de transformação definidas no Capítulo 3 e o vetor de entrada $\mathbf{x}_t(n)$ é dado por

$$\mathbf{x}_t(n) = [x(n-D)\ \ x(n-D-1)\ \ x(n-D-2)\ \cdots\ x(n-N+1)]^T. \tag{5.6}$$

Assim, a relação de entrada e saída do filtro BIFIR pode ser escrita como

$$y_t(n) = \mathbf{w}_t^T \tilde{\mathbf{x}}_t'(n). \tag{5.7}$$

Finalmente, considerando o algoritmo LMS para adaptação dos coeficientes, tem-se

$$\mathbf{w}_h(n) = \mathbf{w}_h(n-1) + 2\mu_h e(n)\mathbf{x}_h(n) \tag{5.8}$$

e

$$\mathbf{w}_t(n) = \mathbf{P}\left[\mathbf{w}_t(n-1) + 2\mu_t e(n)\tilde{\mathbf{x}}_t'(n)\right] \tag{5.9}$$

onde μ_h e μ_t são os passos de adaptação, $\mathbf{P}$ é uma matriz de projeção similar a (3.118), e

$$e(n) = d(n) - y(n) = d(n) - y_h(n) - y_t(n) \tag{5.10}$$

é o sinal de erro.

5.1.2 Complexidade Computacional

A complexidade computacional requerida para implementação do filtro FIR-BIFIR adaptado pelo algoritmo LMS, descrita em termos do número de operações por amostra, é apresentada na Tabela 5-1. Com o objetivo de comparar a carga computacional da abordagem FIR-BIFIR com àquela requerida pela estrutura FIR-IFIR de [75], a implementação utilizada para obter os resultados de simulação apresentados em [75] é aqui considerada. Assim, tem-se uma estrutura FIR-IFIR com $N = 250$, $L = 4$, $D = 31$, um filtro FIR com 50 coeficientes e um interpolador com 23 coeficientes. No caso da estrutura FIR-BIFIR, tem-se $N = 250$, $L = 4$ e $D = 31$, o que resulta em um filtro FIR com D coeficientes e um interpolador com $M = 7$ coeficientes. A Tabela 5-2 apresenta, para o caso descrito, uma comparação da carga computacional requerida para o filtro FIR convencional, o filtro FIR-IFIR de [74], o filtro FIR-IFIR de [75] e o FIR-BIFIR proposto. A partir dessa tabela, observa-se que as abordagens FIR-IFIR e FIR-BIFIR apresentam complexidade computacional muito inferior à do filtro FIR adaptativo. Adicionalmente, constata-se que a abordagem FIR-BIFIR proposta exibe complexidade computacional similar à do FIR-IFIR de [74] e uma redução de 18,5% de complexidade em relação à do filtro FIR-IFIR de [75].

Tabela 5-1. Número de operações por amostra
requerido para implementação do filtro FIR-BIFIR

	Filtro FIR		Filtro BIFIR	
	Filtragem	Adaptação	Filtragem	Adaptação
Multiplicações	D	D	$M+N_{\text{ts}}+L-1$	N_{ts}
Somas	$D-1$	D	$M+N_{\text{ts}}+L-3$	N_{ts}

Tabela 5-2. Comparação de complexidade computacional entre diferentes implementações de filtros FIR, FIR-IFIR e FIR-BIFIR com $N=250$, $L=4$ e $D=31$

	Operações por amostra	Percentuais relativos	
FIR convencional	999	**100%**	
FIR-IFIR [74]	355	35,5%	
FIR-IFIR [75]	443	44,3%	**100%**
FIR-BIFIR (proposto)	361	36,1%	81,5%

5.1.3 Resultados de Simulação

Com o objetivo de avaliar o desempenho do filtro FIR-BIFIR em comparação com outras implementações FIR-IFIR, nesta seção, são apresentados alguns resultados de simulação. Os diferentes filtros são aplicados a um problema de cancelamento de eco cuja resposta ao impulso, ilustrada na Figura 5.6, possui características similares às respostas consideradas em [75]. Os tamanhos de memória, fatores de esparsidade e valores de atraso considerados são os mesmos utilizados para a comparação de complexidade descrita na seção anterior. O desempenho é avaliado em termos da medida ERLE (*echo return loss enhancement*) definida como

$$\text{ERLE} = 10\log_{10}\frac{E[d^2(n)]}{E[e^2(n)]} \tag{5.11}$$

onde $d(n)$ é o sinal de retorno com eco e $e(n)$, o sinal com o eco cancelado. As curvas de ERLE foram obtidas por simulação de Monte Carlo (média de 100 realizações). De forma similar a [75], o sinal de entrada é um sinal codificado em 16-PAM com distribuição uniforme de probabilidade entre seus diferentes níveis de sinal.

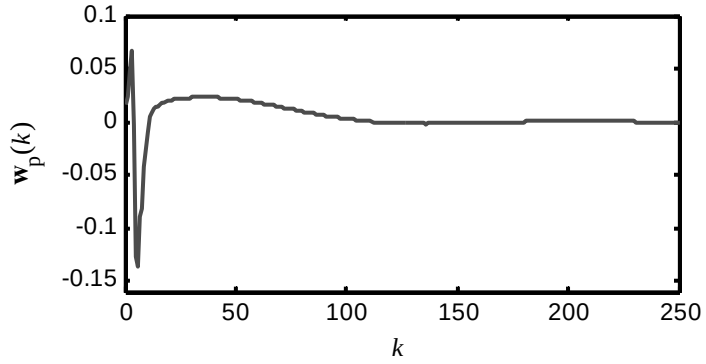


Figura 5.6. Resposta ao impulso de eco típica de um sistema de comunicação DSL.

Os resultados de simulação obtidos são apresentados na Figura 5.7. A Figura 5.7(a) mostra os resultados obtidos com as três implementações consideradas utilizando a estratégia descrita em [75] para escolha do passo de adaptação. Dessa forma, o intervalo total de adaptação de 12.000 amostras é dividido em cinco estágios e a cada estágio o passo de adaptação é reduzido à metade, sendo o valor inicial de tal passo igual a um valor crítico μ_{crit} que garante a estabilidade do algoritmo (obtido experimentalmente). Ainda, a Figura 5.7(b) apresenta os resultados de simulação obtidos mantendo os passos de adaptação iguais a μ_{crit} ao longo de todo o processo adaptativo, enquanto a Figura 5.7(c) apresenta os resultados obtidos usando um valor de passo para o filtro FIR e outro valor diferente para o filtro interpolado das estruturas FIR-IFIR e FIR-BIFIR. A partir desses resultados, observa-se que, independente da estratégia de escolha do passo de adaptação utilizada, o algoritmo FIR-BIFIR proposto apresenta um desempenho superior ao das demais implementações FIR-IFIR tanto com vistas ao valor máximo de ERLE quanto em relação à taxa de convergência.

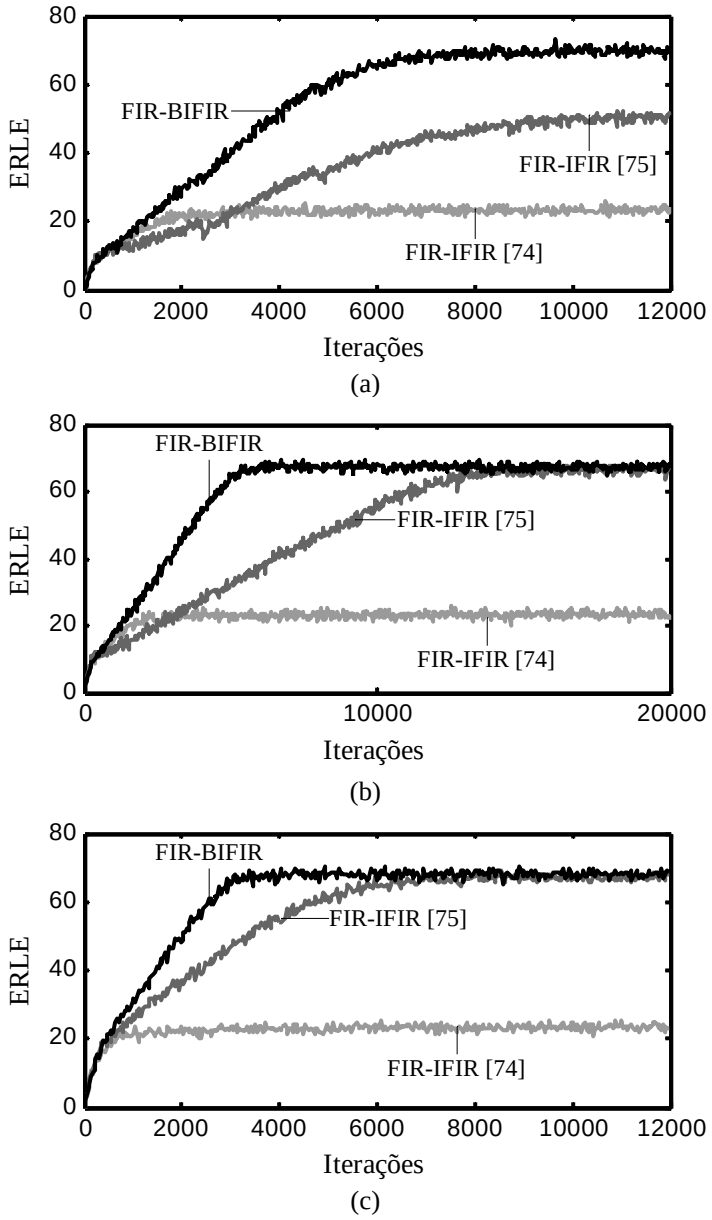


Figura 5.7. Curvas de ERLE (média de 100 realizações).

5.2 Cancelamento de Eco Linear de Linha em Comunicações de Voz

De acordo com a recomendação G.168 da *International Telecommunication Union* (ITU) [76], a forma mais comum encontrada de resposta ao impulso de eco em redes telefônicas da América do Norte é a resposta $\mathbf{m}_5(k)$ ilustrada na Figura 5.8. Observa-se, nessa figura, que $\mathbf{m}_5(k)$ apresenta uma parte inicial (indicada por A) onde a resposta é praticamente nula (atraso do eco), seguida de uma parte com variações rápidas (indicada por B) e uma parte final (indicada por C) na qual a resposta varia suavemente. Uma

resposta desse tipo também é verificada em uma simulação com sinais adquiridos por um adaptador de telefonia analógica (ATA) para redes VoIP. A resposta, apresentada na Figura 5.9, mostra um comprimento do atraso inicial de eco praticamente nulo como principal diferença com relação à resposta apresentada na Figura 5.8. Essa característica é bastante comum em aplicações envolvendo dispositivos ATA devido ao menor comprimento das linhas de transmissão envolvidas. Como conseqüência, os filtros FIR-BIFIR podem ser uma boa opção para implementação de canceladores de eco em tais aplicações, uma vez que as características das respostas ao impulso são similares às encontradas em sistemas DSL (descritas na Seção 5.1). Por outro lado, ao contrário dos casos envolvendo transmissões de dados nos quais o sinal de entrada é praticamente estacionário e existe uma fase de treinamento para o cancelador de eco [75], nos casos de comunicações de voz são requeridos algoritmos adaptativos mais robustos a variações das estatísticas e da potência do sinal de entrada. Assim, para esses casos, o algoritmo NLMS é uma melhor opção do que o algoritmo LMS.

A aplicação do filtro FIR-BIFIR ao problema de cancelamento de eco de linha em comunicações de voz é avaliada nesta seção. Com esse objetivo, a implementação de tal filtro usando o NLMS como algoritmo adaptativo é inicialmente discutida, sendo sua aplicação focada ao cancelamento de eco em ATAs para redes VoIP, a qual é avaliada a seguir através de simulações realizadas com sinais obtidos através de aquisições reais.

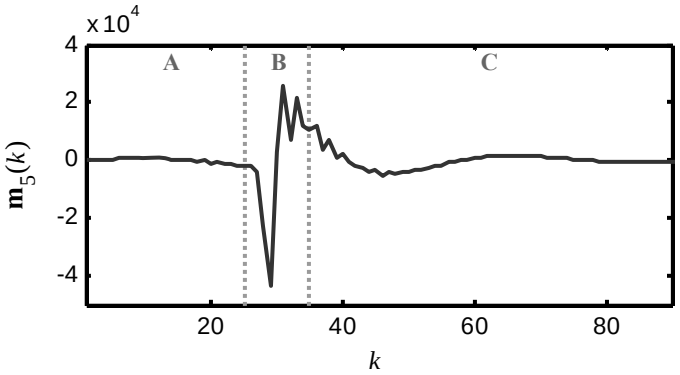


Figura 5.8. Resposta ao impulso de eco típica de redes telefônicas de acordo com a recomendação G.168 da ITU [76].

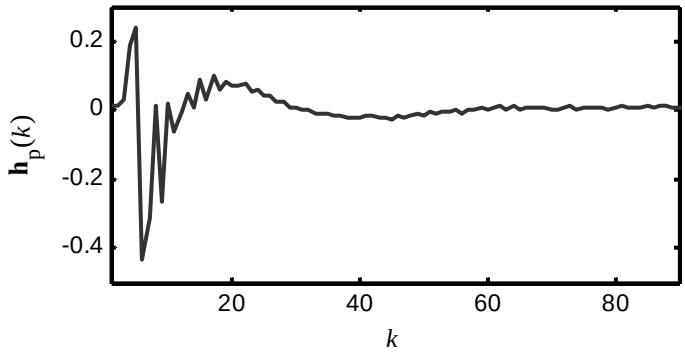


Figura 5.9. Resposta ao impulso de eco obtida em uma simulação com sinais reais.

5.2.1 Adaptação do Filtro FIR-BIFIR usando o Algoritmo NLMS

As expressões para a adaptação da estrutura FIR-BIFIR (Seção 5.1.1) usando o algoritmo NLMS são obtidas adotando uma abordagem similar à apresentada na Seção 4.2.2. Assim, obtém-se

$$\mathbf{w}_h(n) = \mathbf{w}_h(n-1) + \frac{\alpha}{\|\mathbf{x}_h(n)\|^2 + \|\mathbf{P}\tilde{\mathbf{x}}_t'(n)\|^2 + \psi} e(n)\mathbf{x}_h(n) \tag{5.12}$$

e

$$\mathbf{w}_t(n) = \mathbf{P}\mathbf{w}_t(n-1) + \frac{\alpha}{\|\mathbf{x}_h(n)\|^2 + \|\mathbf{P}\tilde{\mathbf{x}}_t'(n)\|^2 + \psi} e(n)\mathbf{P}\tilde{\mathbf{x}}_t'(n) \tag{5.13}$$

onde α e ψ são os parâmetros de controle do algoritmo NLMS, $\mathbf{P}$ é uma matriz de projeção similar a (3.118) e $e(n)$, o sinal de erro dado por (5.10).

A complexidade computacional requerida para implementação do filtro FIR-BIFIR adaptado pelo algoritmo NLMS é igual à complexidade requerida para implementação usando o algoritmo LMS (Tabela 5-1) acrescida da complexidade necessária para o cálculo do termo de normalização $\|\mathbf{x}_h(n)\|^2 + \|\mathbf{P}\tilde{\mathbf{x}}_t'(n)\|^2 + \psi$. A comparação de tal complexidade com a de um filtro FIR convencional e a de um filtro IFIR é apresentada na próxima seção no contexto de um problema real de cancelamento de eco de linha em comunicações de voz.

5.2.2 Resultados de Simulação com Sinais de Aquisições Reais

Para avaliar o desempenho do filtro FIR-BIFIR aplicado a um problema real de cancelamento de eco, são apresentados, nesta seção, alguns resultados de simulação considerando 3 aquisições de conversas telefônicas em um dispositivo ATA. Tais

aquisições não envolvem situações de *double-talk* [77]. Os parâmetros do filtro FIR-BIFIR usados em todas as simulações são $N = 101$, $D = 15$, $L = 4$ e $g = [0,25\ \ 0,5\ \ 0,75\ \ 1\ \ 0,75\ \ 0,5\ \ 0,25]^{\mathrm{T}}$. Para efeito de comparação, também são considerados nas simulações os seguintes filtros adaptativos: IFIR com tamanho de memória $N = 101$ e $L = 2$; FIR convencional com 101 coeficientes; e FIR convencional com 15 coeficientes. Este último filtro é igual ao filtro FIR da estrutura FIR-BIFIR com $D = 15$, o qual é incluído nas simulações com o objetivo de verificar o ganho de desempenho proporcionado pelo filtro BIFIR da estrutura FIR-BIFIR. A comparação entre os diferentes filtros é realizada em termos da medida de ERLE, definida em (5.11). Os parâmetros do algoritmo NLMS utilizados para adaptação de todos os filtros são $\alpha = 0,5$ e $\psi = 0,1$. A complexidade computacional requerida para implementação de cada um dos filtros aqui considerados, à exceção do filtro FIR com 15 coeficientes, é apresentada na Tabela 5-3 em termos do número de operações por amostra. A partir dessa tabela constata-se uma grande redução de carga computacional obtida com o filtro FIR-BIFIR, especialmente quando comparada à do filtro FIR convencional.

Tabela 5-3. Complexidade computacional requerida em diferentes implementações de filtros FIR, FIR-IFIR e FIR-BIFIR com $N = 101$

	Operações por amostra	Percentuais relativos	
FIR convencional	605	**100%**	
IFIR	310	51,2%	**100%**
FIR-BIFIR (proposto)	194	32,1%	62,6%

A Figura 5.10 mostra os resultados obtidos usando os sinais de uma primeira aquisição de pouco mais de 9 segundos de duração. A partir dessa figura, observa-se que o filtro FIR-BIFIR apresenta um desempenho muito próximo ao obtido com o filtro FIR convencional e superior ao do filtro IFIR. Os resultados obtidos com sinais de uma segunda aquisição, com a mesma duração da primeira, são mostrados na Figura 5.11. Novamente o desempenho do filtro FIR-BIFIR é bastante próximo ao obtido com o filtro FIR. Ainda, observa-se que não houve diferença substancial entre o desempenho do filtro FIR-BIFIR e o do filtro IFIR nesse caso. No entanto, é importante notar que o filtro IFIR aqui considerado $(L = 2)$ possui complexidade computacional superior ao do filtro FIR-BIFIR. Para efeito de comparação, resultados de simulação considerando o filtro IFIR com $L = 4$ são mostrados na Figura 5.12, onde um melhor desempenho do filtro

FIR-BIFIR é observado. Os resultados obtidos considerando sinais de uma terceira aquisição, agora de pouco menos de 5 segundos, são mostrados na Figura 5.13. Novamente, verifica-se que o desempenho do filtro FIR-BIFIR é bastante próximo ao do filtro FIR, sendo ainda superior ao do filtro IFIR. No caso das duas primeiras aquisições, o nível de ERLE máximo obtido para todos os filtros é aproximadamente 30 dB, enquanto para a terceira aquisição o nível de ERLE máximo é cerca de 20 dB. Uma análise mais minuciosa revelou que essa diferença de desempenho é causada pela presença de distorções não-lineares na resposta de eco do ATA utilizado na terceira aquisição. Assim, verifica-se que, em alguns casos, é necessário utilizar filtros adaptativos não-lineares para o cancelamento de eco de linha em comunicações de voz.

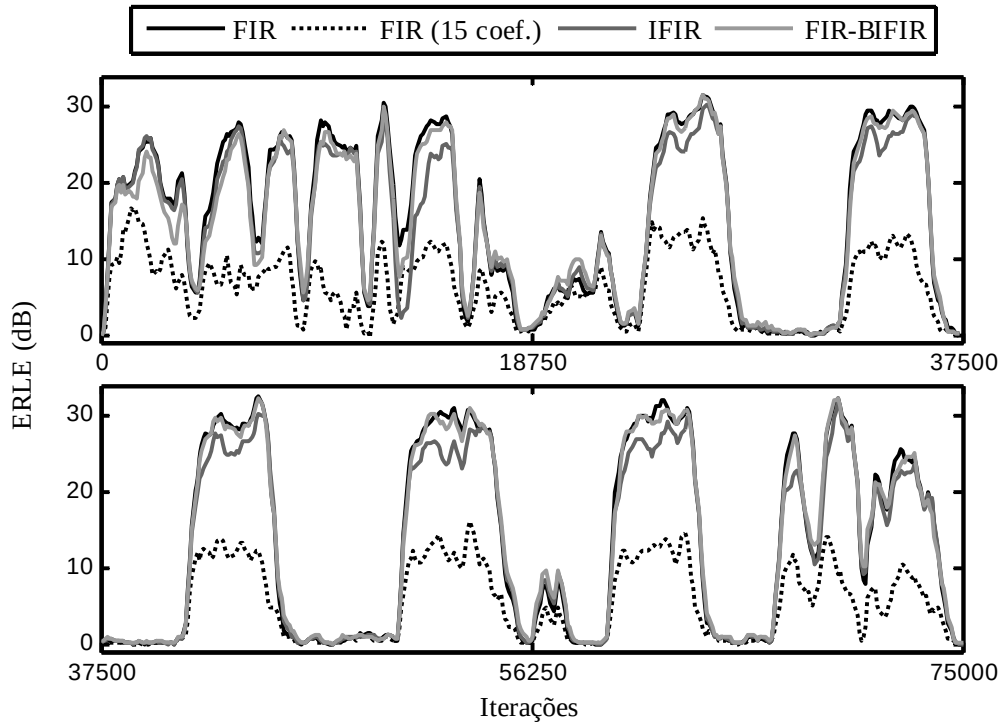


Figura 5.10. Curvas de ERLE obtidas usando sinais da primeira aquisição considerada na Seção 5.2.2.

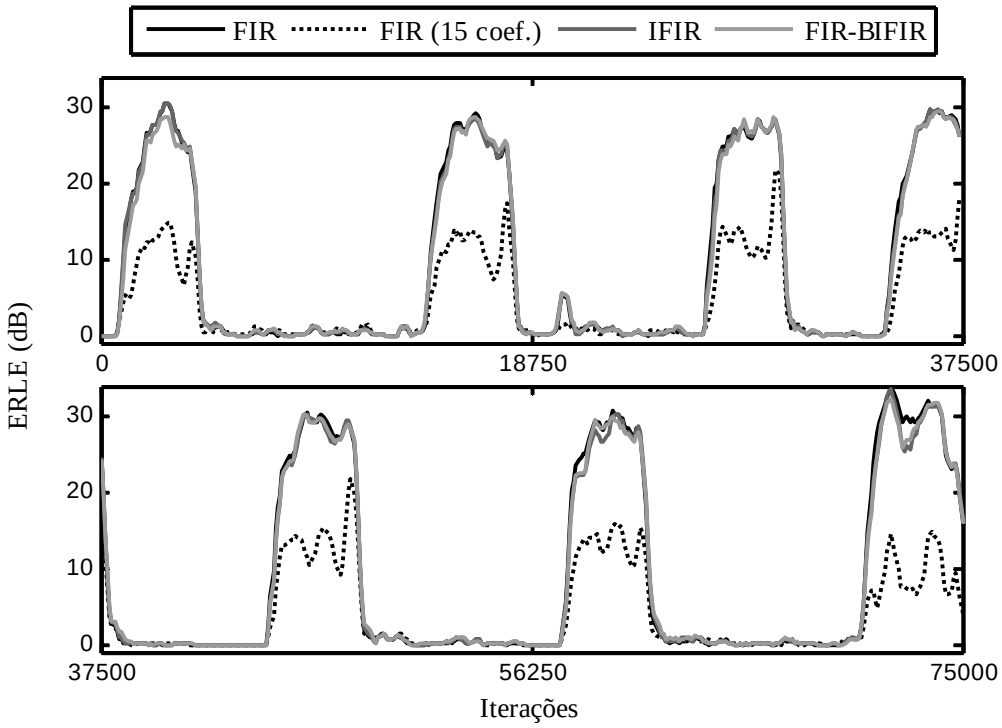


Figura 5.11. Curvas de ERLE obtidas usando sinais da segunda aquisição considerada na Seção 5.2.2. Filtro IFIR com $L = 2$.

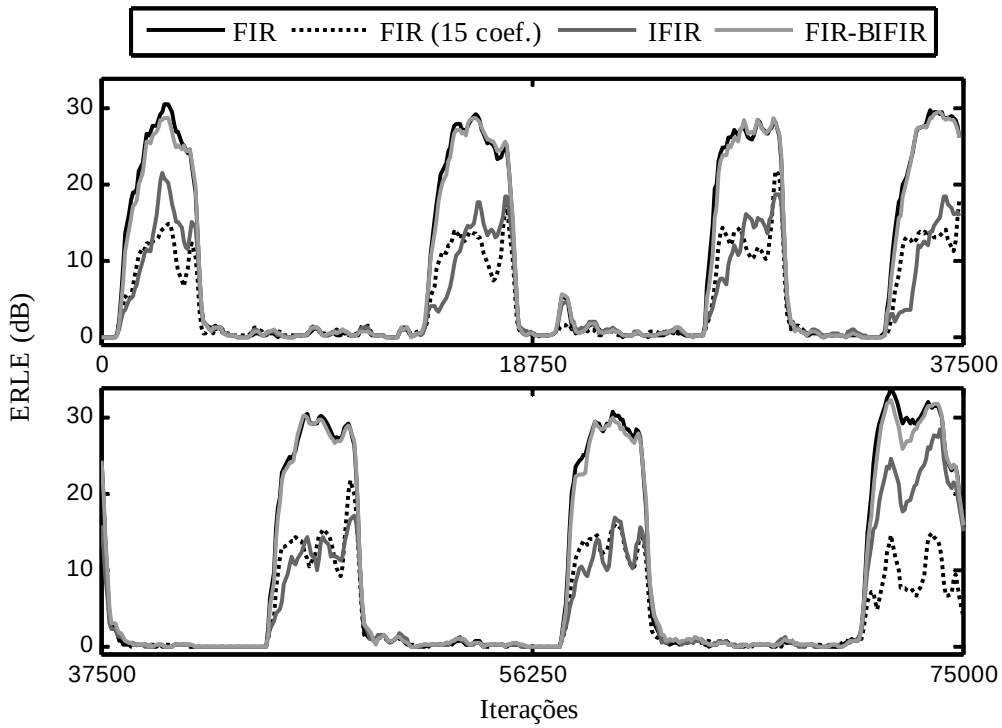


Figura 5.12. Curvas de ERLE obtidas usando sinais da segunda aquisição considerada na Seção 5.2.2. Filtro IFIR com $L = 4$.

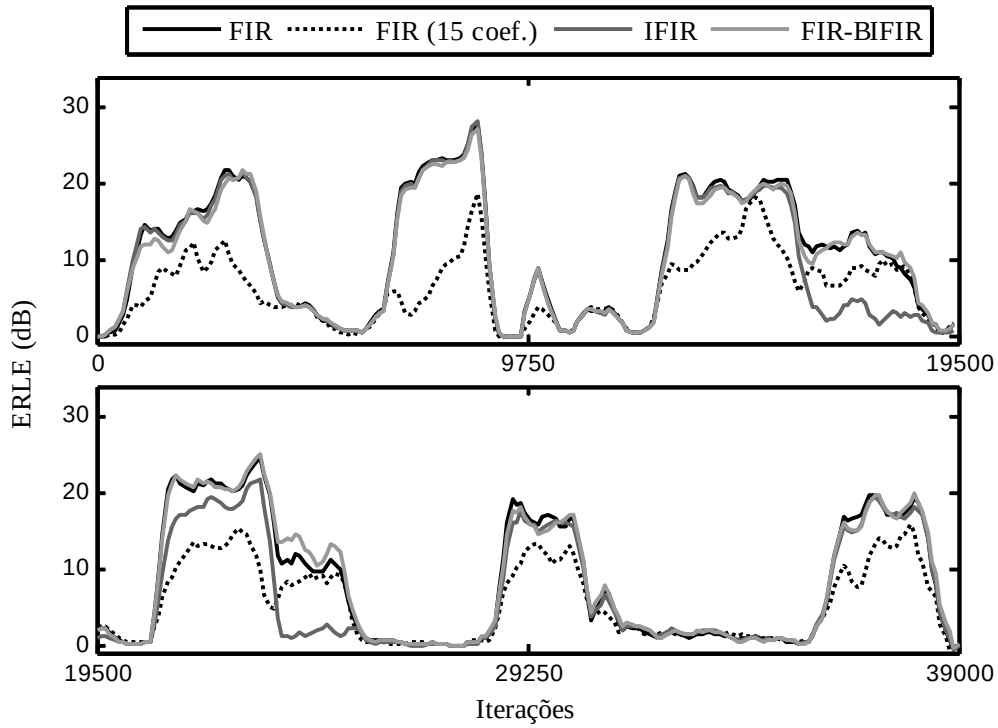


Figura 5.13. Curvas de ERLE obtidas usando sinais da terceira aquisição considerada na Seção 5.2.2.

5.3 Cancelamento de Eco Não-Linear de Linha em Comunicações de Voz

Em muitos problemas de cancelamento de eco, o uso de canceladores lineares não é suficiente para atender aos requisitos de desempenho devido à presença de distorções não-lineares no caminho do eco. Para resolver esse problema, implementações do filtro Volterra adaptativo vêm sendo utilizadas com sucesso [53], [71], [73]. No entanto, tais implementações sofrem importantes restrições de aplicação devido ao seu alto custo computacional. Como conseqüência, diversas estruturas de implementação com complexidade reduzida vêm sendo propostas e aplicadas a problemas de cancelamento de eco [53], [73]. Nesse contexto, filtros Volterra interpolados surgem como uma alternativa interessante devido à sua reduzida complexidade computacional. Nesta seção, implementações adaptativas e inteiramente adaptativas dos filtros Volterra interpolados são aplicadas ao problema de cancelamento de eco de linha. Adicionalmente, um novo algoritmo incorporando o filtro FIR-BIFIR à estrutura Volterra interpolada inteiramente adaptativa é proposta e avaliada.

5.3.1 Uso de Filtros FIR-BIFIR em Estruturas Volterra Interpoladas Inteiramente Adaptativas

Conforme descrito na Seção 3.3.2.2, uma maneira de melhorar o desempenho de um filtro Volterra interpolado é utilizar a sua implementação parcialmente interpolada, na qual o bloco de primeira ordem é implementado sem o uso de esparsidade e interpolação. Com isso, o custo computacional de implementação do filtro Volterra interpolado é pouco alterado. No entanto, em algumas aplicações a diferença de custo computacional entre as implementações interpolada e parcialmente interpolada do filtro Volterra pode ser significativa. Por exemplo, o cancelamento de eco de linha apresentado em [53] é realizado usando um filtro Volterra cujo bloco de primeira ordem possui tamanho de memória $N_1 = 100,$ enquanto o bloco de segunda ordem apresenta tamanho de memória $N_2 = 54.$ Nesse caso, o uso de uma abordagem interpolada (457 coeficientes) em comparação com o uso de uma abordagem parcialmente interpolada (506 coeficientes) implica uma diferença de cerca de 10% no número de coeficientes. Uma solução intermediária entre a abordagem interpolada e a parcialmente interpolada é obtida implementando o bloco de primeira ordem do filtro Volterra usando o filtro FIR-BIFIR proposto na Seção 5.1.1. Considerando, ainda, a abordagem Volterra interpolada inteiramente adaptativa (FAIV) para realização dos blocos não-lineares, obtém-se a estrutura FIR-BIFIR-FAIV cuja implementação de segunda ordem é mostrada na Figura 5.14.

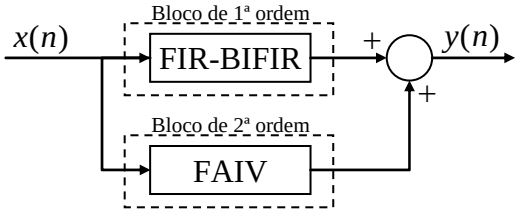


Figura 5.14. Diagrama de blocos da abordagem FIR-BIFIR-FAIV proposta.

A adaptação do filtro FIR-BIFIR-FAIV é feita com base nas expressões apresentadas na Seção 5.2.1 para o filtro FIR-BIFIR adaptado pelo algoritmo NLMS e nas expressões para adaptação do filtro Volterra interpolado usando o algoritmo LMS-NLMS (Seção 4.4.3). Assim, as seguintes expressões são consideradas:

$$\mathbf{w}_{\mathrm{h}}(n) = \mathbf{w}_{\mathrm{h}}(n-1) + \frac{\alpha}{q(n)+\psi} e(n)\mathbf{x}_{\mathrm{h}}(n) \tag{5.14}$$

$$\mathbf{w}_{\mathrm{t}}(n) = \mathbf{P}_1\mathbf{w}_{\mathrm{t}}(n-1) + \frac{\alpha}{q(n)+\psi} e(n)\mathbf{P}_1\tilde{\mathbf{x}}'_{\mathrm{t}}(n) \tag{5.15}$$

e

$$\underline{\mathbf{h}}_{2s}(n+1) = \mathbf{P}_2\underline{\mathbf{h}}_{2s}(n) + \frac{\alpha}{q(n)+\psi}e(n)\mathbf{P}_2\underline{\tilde{\mathbf{x}}}_2(n) \tag{5.16}$$

com o termo de normalização dado por

$$q(n) = \left\|\mathbf{x}_h(n)\right\|^2 + \left\|\mathbf{P}_1\tilde{\mathbf{x}}_t'(n)\right\|^2 + \left\|\mathbf{P}_2\underline{\tilde{\mathbf{x}}}_2(n)\right\|^2. \tag{5.17}$$

Em (5.14)-(5.16), α e ψ são parâmetros de controle e $\mathbf{P}_1$ e $\mathbf{P}_2$, matrizes de projeção análogas a (3.118). Além destas expressões, tem-se o interpolador da parte FAIV do filtro FIR-BIFIR-FAIV adaptado usando (4.123). A complexidade computacional requerida para implementação do filtro FIR-BIFIR-FAIV pode variar bastante em função dos parâmetros utilizados para sua implementação (fatores de esparsidade, atraso do filtro FIR-BIFIR, dentre outros). Assim, o cálculo de complexidade deve ser realizado para cada caso específico considerando o custo computacional de implementação do filtro FIR-BIFIR e do filtro Volterra interpolado inteiramente adaptativo. A comparação de complexidade entre o filtro FIR-BIFIR-FAIV e outras implementações do filtro Volterra é discutida na próxima seção, onde resultados de simulação com sinais reais são apresentados.

5.3.2 Resultados de Simulação com Sinais Reais

Nesta seção, resultados de simulação são apresentados com o objetivo de avaliar o desempenho de diferentes implementações interpoladas do filtro Volterra aplicadas ao cancelamento de eco. De forma similar à Seção 5.2.2, os resultados aqui apresentados são obtidos usando sinais de aquisições reais de conversação telefônica em dispositivos ATA sem a ocorrência de situações de *double-talk* [76]. Os seguintes filtros adaptativos são considerados nas simulações: filtro FIR convencional com $N = 101$; filtro Volterra convencional (V) com $N_1 = 101$ (tamanho de memória do bloco de primeira ordem) e $N_2 = 21$ (tamanho de memória do bloco de segunda ordem); filtro Volterra parcialmente interpolado adaptativo (APIV) com $N_1 = 101$, $N_2 = 21$, $L = 2$ e $\mathbf{g} = [0{,}5\ 1\ 0{,}5]^T$; filtro Volterra parcialmente interpolado inteiramente adaptativo (FAPIV) com $N_1 = 101$, $N_2 = 21$, $L = 2$ e $\mathbf{g}(0) = [0{,}5\ 1\ 0{,}5]^T$; e filtro FIR-BIFIR-FAIV possuindo um bloco de primeira ordem FIR-BIFIR com $N_1 = 101$, $D = 15$, $L_1 = 4$ e $\mathbf{g}_1 = [0{,}25\ 0{,}5\ 0{,}75\ 1\ 0{,}75\ 0{,}5\ 0{,}25]^T$ e, ainda, um bloco de segunda ordem FAIV com

$N_2 = 21$, $L_2 = 2$ e $\mathbf{g}_2(0) = [0,5\ 1\ 0,5]^T$. Os parâmetros do algoritmo NLMS usados para adaptação dos diferentes filtros são $\alpha = 0,5$ e $\psi = 0,1$. Além disso, para o algoritmo LMS usado na adaptação do interpolador adaptativo dos filtros FAPIV e FIR-BIFIR-FAIV, um passo de adaptação $\mu_i = 0,5$ é utilizado. A complexidade computacional dos diferentes filtros aqui considerados, em termos do número de operações por amostra, é apresentada na Tabela 5-4. Observa-se, a partir dessa tabela, a importante redução de complexidade obtida com o uso do filtro FIR-BIFIR-FAIV, sendo que tal complexidade é bastante inferior à requerida pelo filtro Volterra convencional.

Tabela 5-4. Complexidade computacional requerida para implementação dos filtros utilizados para obter os resultados de simulação apresentados na Seção 5.3.2

	Operações por amostra	Percentuais relativos
Volterra convencional adaptativo (V)	2018	**100%**
Volterra parc. interpolado adaptativo (APIV)	1028	50,9%
Volterra parc. interpolado inteiramente adaptativo (FAPIV)	1207	59,8%
FIR-BIFIR-FAIV	795	39,4%
FIR	609	30,2%

Os resultados obtidos com sinais de uma primeira aquisição, de pouco mais de 6 segundos, são mostrados na Figura 5.15, onde se observa o bom desempenho tanto do filtro FAPIV (proposto na Seção 4.4) quando do filtro FIR-BIFIR-FAIV (proposto na Seção 5.3.1). Em ambos os casos, o desempenho é bastante próximo ao do filtro Volterra convencional, requerendo um custo computacional consideravelmente reduzido conforme indicado na Tabela 5-4. Na comparação entre o filtro FAPIV e o FIR-BIFIR-FAIV, observa-se uma pequena vantagem, em termos do desempenho, a favor do filtro FAPIV. Os resultados obtidos com sinais de uma segunda aquisição, agora de pouco mais de 3 segundos de duração, estão mostrados na Figura 5.16. Novamente, observa-se o bom desempenho dos filtros FAPIV e FIR-BIFIR-FAIV, sendo que tal desempenho é muito próximo do obtido com o filtro Volterra convencional. Tais resultados atestam a efetividade e a aplicabilidade das abordagens interpoladas para implementação de filtros Volterra adaptativos.

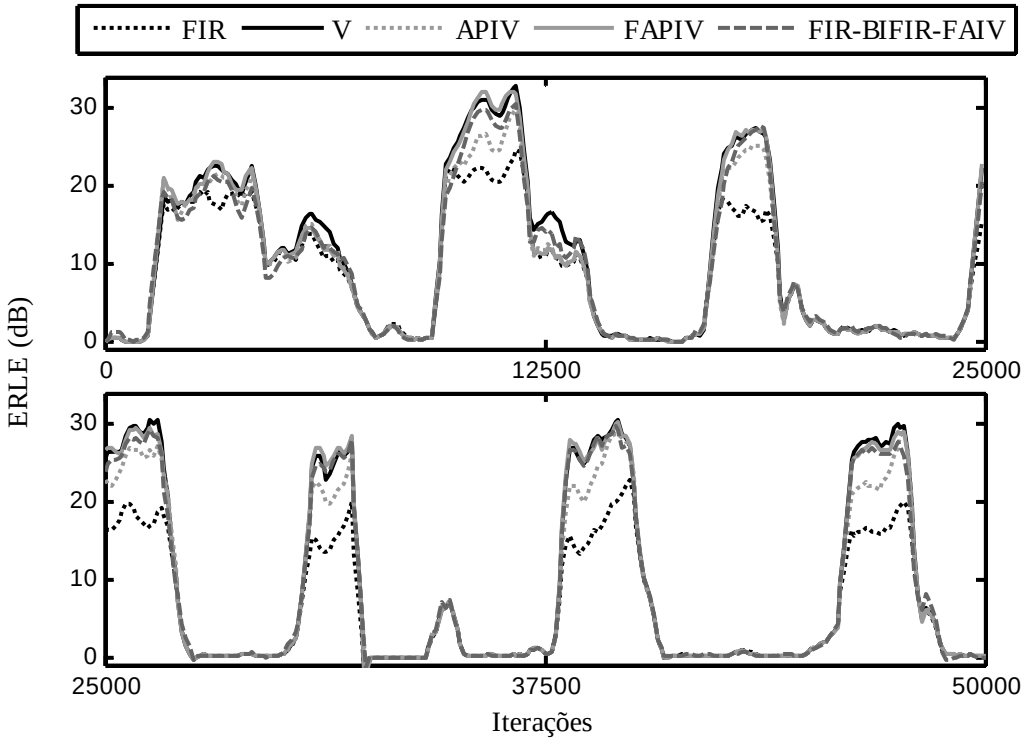


Figura 5.15. Curvas de ERLE obtidas usando sinais da primeira aquisição considerada na Seção 5.3.2.

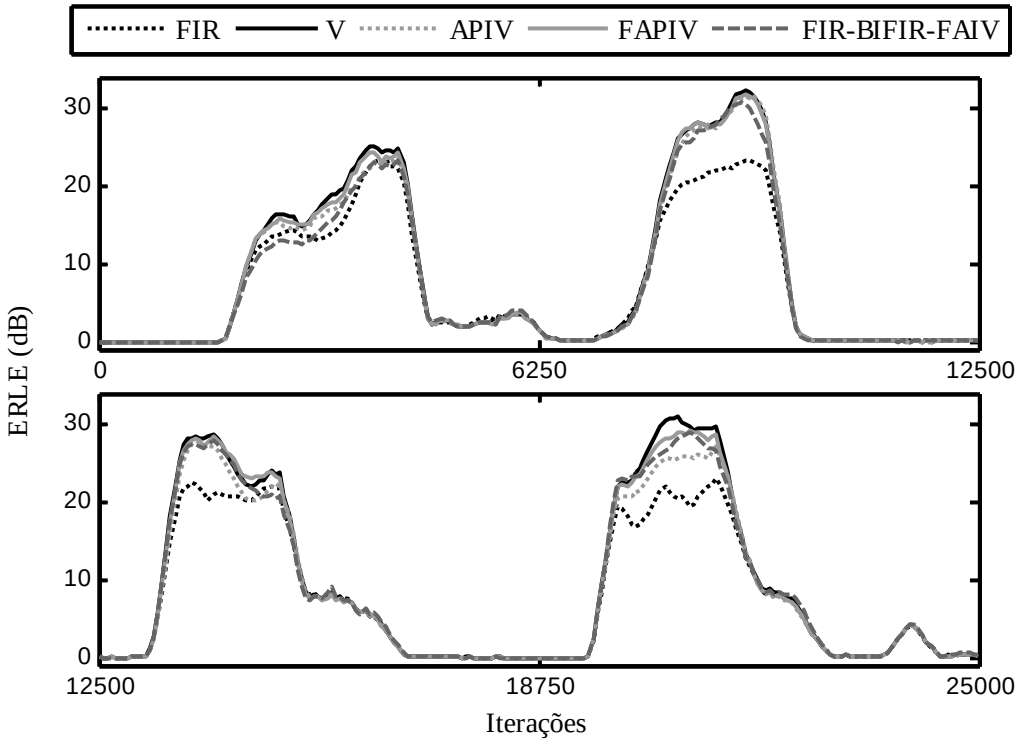


Figura 5.16. Curvas de ERLE obtidas usando sinais da segunda aquisição considerada na Seção 5.3.2.

5.4 Considerações

Neste capítulo, o problema de cancelamento de eco de linha para comunicações de voz e dados foi discutido. Para tal, abordagens interpoladas para implementação de filtros FIR e filtros Volterra adaptativos com complexidade reduzida foram consideradas. Como resultado, observou-se o grande potencial de tais abordagens para a implementação eficaz de canceladores de eco com complexidade menor do que a de outras estruturas concorrentes.

CAPÍTULO 6

Conclusões Finais

Neste capítulo, as conclusões finais deste trabalho de tese são apresentadas. Inicialmente, uma discussão dos resultados obtidos de cada capítulo é considerada, seguindo algumas sugestões para trabalhos futuros e as considerações finais.

6.1 Sumário e Discussão dos Resultados

O Capítulo 1 deste trabalho é dedicado a uma breve apresentação dos filtros adaptativos e à descrição do papel de tais filtros no contexto do processamento digital de sinais. Adicionalmente, as características que diferenciam os filtros adaptativos lineares dos não-lineares são brevemente descritas. O Capítulo 1 apresenta, ainda, os objetivos e as motivações para o desenvolvimento deste trabalho de pesquisa.

O Capítulo 2 é iniciado com uma breve descrição histórica do processo de desenvolvimento do filtro Volterra até suas primeiras aplicações no contexto adaptativo. Em seguida, tal filtro é apresentado destacando as suas principais características. Ainda no Capítulo 2, uma análise estatística do filtro Volterra adaptado pelo algoritmo LMS é apresentada. Como resultado dessa análise, expressões de modelagem do vetor de coeficientes ótimo, comportamento dos momentos de primeira e segunda ordens, curva de aprendizagem e desajuste do algoritmo são obtidas. Adicionalmente, uma descrição detalhada acerca da influência da matriz de autocorrelação de entrada no comportamento do filtro é apresentada. O modelo obtido a partir de tal análise é a primeira contribuição original deste trabalho, tendo ainda a vantagem de sua aplicação não sofrer restrições quanto à ordem do filtro nem quanto ao modelo probabilístico do sinal de entrada.

O Capítulo 3 trata dos filtros Volterra interpolados e das suas implementações adaptativas realizadas usando interpoladores fixos e filtros Volterra esparsos adaptativos. Nesse contexto, os filtros Volterra interpolados são inicialmente apresentados, sua implementação adaptativa é descrita e a complexidade computacional requerida é discutida. Em seguida, como outra contribuição original deste trabalho, uma descrição dos filtros Volterra interpolados é apresentada, permitindo verificar, por exemplo, a maneira

como os coeficientes são recriados através do processo de interpolação. Outra característica dos filtros Volterra interpolados, evidenciada a partir de sua descrição matemática, é o efeito de borda. Tal efeito pode produzir, em muitos casos, uma importante degradação de desempenho. Para eliminar esse problema, um procedimento para implementação de filtros Volterra interpolados com remoção do efeito de borda é proposto no Capítulo 3. Como resultado, novas implementações adaptativas de filtros IFIR e filtros Volterra interpolados são obtidas, apresentando desempenho notavelmente superior ao das implementações interpoladas usuais. O Capítulo 3 apresenta ainda uma análise estatística dos filtros Volterra interpolados adaptados pelo algoritmo LMS baseada em uma abordagem com restrições. Esse procedimento possibilita o uso do modelo obtido também para a análise de implementações esparsas do filtro Volterra. Adicionalmente, tal modelo não tem sua aplicação limitada pela ordem do filtro nem pelo modelo probabilístico do sinal de entrada.

O tema central do Capítulo 4 são as implementações inteiramente adaptativas dos filtros FIR e Volterra interpolados. Em tais implementações, os filtros dispostos em cascata que compõem a estrutura interpolada (interpolador e filtro esparso) são ambos adaptativos. A primeira contribuição apresentada no Capítulo 4 é o desenvolvimento, a partir de uma abordagem original, de expressões para adaptação de filtros FIR em cascata usando os algoritmos LMS e NLMS. A abordagem utilizada é interessante por facilitar o desenvolvimento de novas implementações inteiramente adaptativas dos filtros IFIR como também dos filtros Volterra interpolados. Assim, ainda no Capítulo 4, expressões para adaptação dos filtros IFIR são desenvolvidas associando uma abordagem com restrições com as expressões obtidas para adaptação dos filtros FIR em cascata. Como resultado, torna-se possível, a partir do equacionamento obtido, verificar características do processo de filtragem IFIR inteiramente adaptativo não descritas em trabalhos anteriores. Outra importante contribuição apresentada no Capítulo 4 é o desenvolvimento de implementações de filtros IFIR inteiramente adaptativos com remoção do efeito de borda. Em tais implementações, os principais problemas que afetam o desempenho dos filtros IFIR em aplicações adaptativas são superados, produzindo assim soluções com muito bom desempenho e baixo custo computacional. A parte final do Capítulo 4 é dedicada às implementações inteiramente adaptativas de filtros Volterra interpolados. O desenvolvimento dessas estruturas tornou-se possível da base teórica desenvolvida ao longo deste trabalho: partindo do estudo e análise do filtro Volterra adaptativo, passando

pela descrição matemática e remoção do efeito de borda dos filtros Volterra interpolados e chegando às novas abordagens para implementação inteiramente adaptativa dos filtros FIR em cascata e filtros IFIR. Como conseqüência de todo esse processo, as implementações inteiramente adaptativas de filtros Volterra interpolados obtidas apresentam um custo computacional de implementação baixo em comparação com o dos filtros Volterra convencionais aliada a um muito bom desempenho. Assim, tais implementações apresentam-se como alternativas interessantes aos filtros Volterra convencionais para muitas das aplicações em que limitações de custo computacional são enfrentadas.

No Capítulo 5, as abordagens interpoladas desenvolvidas neste trabalho são aplicadas a problemas de cancelamento de eco de linha em comunicações de dados e voz. O primeiro problema considerado é o cancelamento de eco em comunicações de dados em sistemas DSL. Para tal, uma nova estrutura FIR-IFIR é desenvolvida tendo como base a implementação com remoção do efeito de borda do filtro IFIR adaptado pelo algoritmo LMS, que está apresentada no Capítulo 3. Essa nova estrutura, denominada FIR-BIFIR, exibe um desempenho superior ao das demais estruturas FIR-IFIR disponíveis na literatura bem como um custo computacional praticamente igual ao da implementação concorrente de menor complexidade. Também no Capítulo 5, a aplicação da abordagem FIR-BIFIR a um problema de cancelamento de eco de linha em comunicações de voz é considerada. Para esse caso, expressões para adaptação de filtros FIR-BIFIR usando o algoritmo NLMS são apresentadas e o desempenho do algoritmo desenvolvido é avaliado em simulações realizadas usando sinais obtidos em aquisições reais. Os resultados de simulação mostrados atestam o alto potencial de aplicação do algoritmo proposto, sendo seu desempenho bastante próximo ao do obtido com filtros FIR convencionais, porém, a um custo computacional bem inferior. Ainda no Capítulo 5, uma nova abordagem híbrida, usando filtros FIR-BIFIR em estruturas Volterra interpoladas inteiramente adaptativas, é proposta e aplicada para um problema de cancelamento de eco de linha não-linear em comunicações de voz. Como resultado, um algoritmo com desempenho satisfatório e baixo custo computacional de implementação é obtido, sendo sua aplicabilidade verificada a partir de simulações realizadas com sinais obtidos em aquisições reais.

6.2 Sugestões para Trabalhos Futuros

Para trabalhos futuros, sugere-se:

- O uso das estruturas adaptativas interpoladas em outras aplicações como, por exemplo, o cancelamento acústico e o controle ativo de ruído e vibrações.
- Aplicação de outros algoritmos para adaptação dos filtros interpolados como, por exemplo, o algoritmo de projeções afins (AP) e o RLS.
- Estudo da característica de posto reduzido dos filtros interpolados.
- Desenvolvimento de uma análise estatística de filtros interpolados inteiramente adaptativos quando adaptados pelo algoritmo LMS.
- Desenvolvimento de uma análise estatística dos filtros interpolados adaptativos e inteiramente adaptativos usando outros algoritmos adaptativos.
- Desenvolvimento de novas estratégias para melhorar o desempenho de filtros interpolados implicando uma considerável redução de custos computacionais para a implementação convencional.

6.3 Considerações Finais

Neste trabalho foram apresentados diversos estudos relacionados aos filtros Volterra adaptativos. O objetivo principal foi aumentar a aplicabilidade de tais filtros através do aperfeiçoamento de sua base teórica bem como do desenvolvimento de estruturas e algoritmos eficientes do ponto de vista da complexidade computacional. Nesse contexto, um importante número de contribuições originais foi obtido e apresentado, sendo elas envolvendo tanto filtros Volterra quanto filtros FIR lineares adaptativos. Adicionalmente, os resultados teóricos obtidos foram, em geral, apresentados em conjunto com resultados de simulação para comprovar a efetividade e a aplicabilidade das estruturas e modelos desenvolvidos. Assim, acredita-se que o objetivo inicial foi atingido bem como novos caminhos para a obtenção de avanços nesta área foram abertos.

Referências Bibliográficas

[1] B. Farhang-Boroujeny, *Adaptive Filters Theory and Applications*, John Wiley & Sons Ltd., 1999.

[2] S. Haykin, *Adaptive Filter Theory*, 4th ed., Upper Saddle River, NJ: Prentice-Hall, 2002.

[3] IEEE History Center Biographies, "Bernard Widrow," Internet: http://www.ieee.org/web/aboutus/history_center/biography/widrow.html, [Aug. 13, 2009].

[4] A. V. Oppenheim and A. S. Willsky, *Signals & Systems*, Prentice Hall, 1997.

[5] Y. Neuvo, C. Y. Dong, and S. K. Mitra, "Interpolated finite impulse response digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 3, pp. 563-570, June 1984.

[6] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*, John Wiley & Sons Inc., 2000.

[7] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Trans. Signal Process.*, vol. 49, no. 8, pp. 1667-1676, Aug. 2001.

[8] E. L. O. Batista, O. J. Tobias, and R. Seara, "A mathematical framework to describe interpolated adaptive Volterra filters," in *Proc. IEEE Int. Telecomm. Symp.*, Fortaleza, Brazil, Sept. 2006, pp. 144-149.

[9] E. L. O. Batista, O. J. Tobias, and Rui Seara, "Análise Estatística de Filtros Volterra Adaptados pelo Algoritmo LMS," in *Anais do XXI Simpósio Brasileiro de Telecomunicações*, Belém-PA, Brazil, CD-ROM, Sept. 2004.

[10] E. L. O. Batista, O. J. Tobias, and R. Seara, "Stochastic model of the LMS Volterra filter," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Poznan, Poland, Sept. 2007, pp. 1721-1725.

[11] V. Volterra, *Theory of Functionals and of Integral and Integro-Differential Equations*, Dover Publications Inc., 1959.

[12] N. Wiener, “Response of a Nonlinear Device to Noise,” Radiation Laboratory, MIT, Cambridge, Massachusetts, Report. No. 129, Apr. 1942.

[13] N. Wiener, *Nonlinear Problems in Random Theory*, The MIT Press, Cambridge, Massachusetts, 1966.

[14] A. Alper, “A consideration of the discrete Volterra series,” *IEEE Trans. Automatic Control*, vol. 10, no. 3, pp. 322-327, July 1965.

[15] M. J. Coker and D. N. Simkins, “A nonlinear adaptive noise canceller,” in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Denver, USA, Apr. 1980, pp. 470-473.

[16] A. Stenger, L. Trautmann, and R. Rabenstein, “Nonlinear acoustic echo cancellation with second order adaptive Volterra filters,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Phoenix, AZ, USA, vol. 2, Mar. 1999, pp. 877-880.

[17] M. Zeller and W. Kellermann, “Iterated Coefficient Updates of Partitioned Block Frequency Domain Second-Order Volterra Filters for Nonlinear AEC,” in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Honolulu, USA, vol. 3, Apr. 2007, pp. 1425-1428.

[18] A. Gutierrez and W. E. Ryan, “Performance of adaptive Volterra equalizers on nonlinear satellite channels,” in *Proc. IEEE Int. Conf. Communications*, Seattle, WA, USA, vol. 1, June 1995, pp. 488-492.

[19] M. Tsujikawa, T. Shiozaki, Y. Kajikawa, and Y. Nomura, “Identification and elimination of second-order nonlinear distortion of loudspeaker systems using Volterra filters,” in *Proc. IEEE Int. Symp. Circ. and Systems*, Geneva, Switzerland, vol. 5, May 2000, pp. 249-252.

[20] J. Li and J. Ilow, “Adaptive Volterra Predistorters for Compensation of Non-linear Effects with Memory in OFDM Transmitters,” in *Proc. of the 4th Annual Commun. Networks and Services Research Conference*, Moncton, Canada, May 2006.

[21] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*, Prentice-Hall, 2000.

[22] Wikipedia, “Euclidean space,” Internet: http://en.wikipedia.org/wiki/Euclidean_space, [Aug. 13, 2009].

[23] J. W. Brewer, “Kronecker Products and Matrix Calculus in System Theory,” *IEEE Trans. on Circuits and Systems*, vol. CAS-25, no. 9, pp. 772-781, Sept. 1978.

[24] G. V. Raz and B. V. Veen, “Baseband Volterra filters for implementing carrier based nonlinearities,” *IEEE Trans. Signal Processing*, vol. 46, no. 1, pp.103-114, Jan. 1998.

[25] A. Fermo; A. Carini, and G. L. Sicuranza, “Simplified Volterra filters for acoustic echo cancellation in GSM receivers,” in *Proc. European Signal Processing Conf. (EUSIPCO)*, Tampere, Finland, Sept. 2000.

[26] T. Koh and E. Powers, “Second-order Volterra filtering and its application to nonlinear system identification,” *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 6, pp. 1445-1455, Dec. 1985.

[27] P. M. Clarkson and M. V. Dokic, “Stability and convergence behavior of second order LMS Volterra filter,” *Electronics Letters*, vol. 27, no. 5, pp. 441-443, Feb. 1991.

[28] M. V. Dokic and P. M. Clarkson, “On the performance of a second-order adaptive Volterra filter,” *IEEE Trans. Signal Process.*, vol. 41, no. 5, pp. 1944-1947, May 1993.

[29] J. Chao and A. Inomata, “A convergence analysis of Volterra adaptive filters,” in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Hong Kong, vol. 4, June 1997, pp. 2477-2480.

[30] T. Ogunfunmi and S. L. Chang, “Second order adaptive Volterra system identification based on discrete nonlinear Wiener model,” *IEE Proc. - Vision, Image, Signal Process.*, vol. 148, no. 1, pp. 21-29, Feb. 2001.

[31] E. L. O. Batista, “Filtro Volterra adaptativo: análise estatística e algoritmos simplificados,” Master’s Thesis, Universidade Federal de Santa Catarina, Florianópolis, SC, Brazil, 2004.

[32] D. W. Griffith and G. R. Arce, “Partially Decoupled Volterra Filters: Formulation and LMS Adaptation,” *IEEE Trans. Signal Process.*, vol. 45, no. 6, pp. 1485-1494, June 1997.

[33] E. L. O. Batista, O. J. Tobias, and Rui Seara, “Filtros Volterra adaptativos: interpolado e parcialmente interpolado,” in *Anais do XXI Simpósio Brasileiro de Telecomunicações*, Belém-PA, Brazil, CD-ROM, Sept. 2004.

[34] E. L. O. Batista, O. J. Tobias, and R. Seara, “Fully and partially interpolated adaptive Volterra filters,” in *Proc. European Signal Processing Conf.*, Antalya, Turkey, CD-ROM, paper no. 1507, Sept. 2005, pp. 1-4.

[35] E. L. O. Batista, O. J. Tobias, and R. Seara, "A Sparse-Interpolated Scheme for Implementing Adaptive Volterra Filters," *IEEE Trans. on Signal Processing*, vol. 58, no. 4, pp. 2022-2035, Apr. 2010.

[36] E. L. O. Batista, O. J. Tobias, and R. Seara, "Border effect removal for IFIR and interpolated Volterra filters," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Honolulu, USA, vol. 3, Apr. 2007, pp. 1329-1332.

[37] D. Mansour and A. Gray Jr., "Frequency domain non-linear adaptive filter," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Atlanta, USA, vol. 6, Apr. 1981, pp. 550-553.

[38] M. Morhac, "A fast algorithm of nonlinear Volterra filtering," *IEEE Trans. Signal Processing*, vol. 39, no. 10, pp. 2353-2356, Oct. 1991.

[39] S. Im and E. J. Powers, "A Fast Method of Discrete Third-Order Volterra Filtering," *IEEE Trans. Signal Processing*, vol. 44, no. 9, pp. 2195-2208, Sept. 1996.

[40] M. J. Reed and M. O. J. Hawksford, "Efficient implementation of the Volterra filter", *IEE Proc.-Vis. Image Signal Processing*, vol. 147, no. 2, pp. 109-114, Apr. 2000.

[41] S. Im, "A Normalized Block LMS Algorithm for Frequency-Domain Volterra Filters," in *Proc. IEEE Signal Process. Workshop on Higher-Order Statistics*, Banff, Canada, July 1997, pp. 152-156.

[42] S. Im and E. J. Powers, "A Block LMS Algorithm for Third-Order Frequency-Domain Volterra Filters," *IEEE Signal Process. Letters*, vol. 4, no. 3, pp. 75-78, Mar. 1997.

[43] F. Kuech and W. Kellermann, "Partitioned Block Frequency-Domain Adaptive Second-Order Volterra Filter," *IEEE Trans. Signal Processing*, vol. 53, no. 2, pp. 564-575, Feb. 2005.

[44] T. M. Panicker and V. J. Matthews, "Parallel-cascade realizations and approximations of truncated Volterra systems," *IEEE Trans. Signal Process.*, vol. 46, no. 10, pp. 2829-2832, Oct. 1998.

[45] H. H. Chang, C. L. Nikias, and A. N. Venetsanopoulos, "Efficient implementations of quadratic filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1511–1528, Dec. 1986.

[46] T. M. Panicker, V. J. Matthews, and G. L. Sicuranza, "Adaptive parallel-cascade truncated Volterra filters," *IEEE Trans. Signal Process.*, vol. 46, no. 10, pp. 2664-2673, Oct. 1998.

[47] L. Li and V. J. Matthews, "Efficient block-adaptive parallel-cascade quadratic filters," *IEEE Trans. Circ. and Syst. II-Analog and Digital Signal Processing*, vol. 46, no. 4, pp. 468-472, Apr. 1999.

[48] L. Yao, W. A. Sethares and Y.-H. Hu, "Identification of a nonlinear system modeled by a sparse Volterra series," in *Proc. IEEE Int. Conf. System Engineering*, Kobe, Japan, Sept. 1992, pp. 624-627.

[49] S. Im and M. J. Bae, "Nonlinear modeling of speech signals using a sparse Volterra model," in *Proc. 4th Int. Symp. Signal Process. and Its Applications*, Gold Coast, Australia, vol. 2, Aug. 1996, pp. 760-763.

[50] L. Yao, "Genetic algorithm based identification of nonlinear systems by sparse Volterra filters," *IEEE Trans. Signal Process.*, vol. 47, no. 12, pp. 3433-3435, Dec. 1999.

[51] L. Tan and J. Jiang, "An adaptive technique for modeling second-order Volterra systems with sparse kernels," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 12, pp. 1610-1615, Dec. 1998.

[52] W. A. Frank, "An efficient approximation to the quadratic Volterra filter and its application in real-time loudspeaker linearization," *Signal Process.*, vol. 45, no. 1, pp. 97-113, July 1995.

[53] F. Kuech and W. Kellermann, "Nonlinear line echo cancellation using a simplified second order Volterra filter," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Orlando, USA, vol. 2, May 2002, pp. 1117-1120.

[54] F. Kuech, A. Mitnacht and W. Kellermann, "Nonlinear acoustic echo cancellation using adaptive orthogonalized power filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Philadelphia, PA, USA, vol. 3, Mar. 2005, pp. 105-108.

[55] F. Kuech and W. Kellermann, "A novel multidelay adaptive algorithm for Volterra filters in diagonal coordinate representation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Montreal, Canada, vol. 2, May 2004, pp. 869-872.

[56] O. J. Tobias and R. Seara, "Analytical model for the first and second moments of an adaptive interpolated FIR filter using the constrained filtered-X LMS algorithm," *IEE Proc. – Vision, Image, Signal Process.*, vol. 148, no. 5, pp. 337-347, Oct. 2001.

[57] R. C. de Lamare and R. Sampaio-Neto, "Adaptive reduced-rank MMSE filtering with interpolated FIR filters and adaptive interpolators," *IEEE Signal Process. Letters*, vol. 12, no. 3, pp. 177-180, Mar. 2005.

[58] O. L. Frost III, "An algorithm for linearly constrained adaptive array processing," *Proc. IEEE*, v. 60, n. 8, p. 926-935, Aug. 1972.

[59] E. L. O. Batista, O. J. Tobias, and Rui Seara, "Filtros FIR adaptativos em cascata: aplicação em filtros IFIR adaptativos," in *Anais do XXV Simpósio Brasileiro de Telecomunicações*, Recife-CE, Brazil, CD-ROM, Sept. 2007.

[60] E. L. O. Batista, O. J. Tobias, and R. Seara, "New insights in adaptive cascaded FIR structure: application to fully adaptive interpolated FIR structures," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Poznan, Poland, Sept. 2007, pp. 370-374.

[61] M. D. Grosen, "New FIR structures for fixed and adaptive digital filters," Ph.D. Dissertation, University of California, Santa Barbara, CA, United States, 1987.

[62] R. C. Bilcu, P. Kuosmanen, and K. Egiazarian, "On adaptive interpolated FIR filters," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP)*, Montreal, Canada, vol. 2, May 2004, pp. 665-668.

[63] R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons Ltd., 1987.

[64] R. Seara, J. C. M. Bermudez, and E. Beck, "A new technique for the implementation of adaptive IFIR filters," in *Proc. Int. Symp. Signals, Systems, Electronics (ISSSE)*, Paris, France, vol. 2, Sept. 1992, pp. 644-647.

[65] E. L. O. Batista, O. J. Tobias, and Rui Seara, "Filtros IFIR Inteiramente Adaptativos sem Efeito de Borda," in *Anais do XXV Simpósio Brasileiro de Telecomunicações*, Recife-CE, Brazil, CD-ROM, Sept. 2007.

[66] E. L. O. Batista, O. J. Tobias, and R. Seara, "A fully adaptive IFIR filter with removed border effect," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Apr. 2008, pp. 3821-3824.

[67] E. L. O. Batista, O. J. Tobias, and R. Seara, "A fully LMS adaptive interpolated Volterra structure," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Las Vegas, NV, Apr. 2008, pp. 3613-3616.

[68] E. L. O. Batista, O. J. Tobias, and R. Seara, "Fully adaptive LMS/NLMS interpolated Volterra filters with removed boundary effect," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Glasgow, Scotland, Aug. 2009, pp. 1725-1729.

[69] J. H. James, B. Chen, and L. Garrison, “Implementing VoIP: a voice transmission performance progress report,” *IEEE Commun. Mag.*, vol. 42, no. 7, pp. 36-41, July 2004.

[70] X. Huang, A. Acero, and H. W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*, Upper Saddle River, Prentice-Hall, 2001.

[71] O. Agazzi and D. G. Messerschmitt, “Nonlinear echo cancellation of data signals,” *IEEE Trans. on Communications*, vol. 30, no. 11, pp. 2421-2433, Nov. 1982.

[72] K. Maxwell, “Asymmetric digital subscriber line: interim technology for the nextforty years,” *IEEE Commun. Mag.*, vol. 34, no. 10, pp. 100-106, Oct. 1996.

[73] A. Guerin, G. Faucon, and R. Le Bouquin-Jeannes, “Nonlinear acoustic echo cancellation based on Volterra filters,” *IEEE Trans. on Speech and Audio Process.*, vol. 11, no. 6, pp.672-684, Nov. 2003.

[74] A. Abousaada, T. Aboulnasr, and W. Steenaart, “An echo tail canceller based on adaptive interpolated FIR filtering,” *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 7, pp. 409-416, July 1992.

[75] S.-S. Lin and W.-R. Wu, “A low-complexity adaptive echo canceller for xDSL Applications,” *IEEE Trans. Signal Process.*, vol. 52, no. 5, pp. 1461-1465, May 2004.

[76] ITU-T Recommendation G.168, *Digital Network Echo Cancellers*, June 2002.

[77] E. Hansler and G. Schmidt, *Acoustic Echo and Noise Control: A Practical Approach*, John Wiley & Sons Ltd., 2004.