ADAPTAÇÃO DIGITAL DE UM FILTRO A CAPACITOR CHAVEADO PROGRAMÁVEL

Eduardo Barbosa Moura Costa

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Mariane Rembold Petraglia, Ph.D.

Prof. Antonio Petraglia, Ph.D.

Prof. Antonio Carlos Moreirão de Queiroz, D.Sc.

Prof. Jacques Szczupak, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

FEVEREIRO DE 2005

COSTA, EDUARDO BARBOSA MOURA

Adaptação Digital de um Filtro a Capacitor Chaveado Programável [Rio de Janeiro] 2005

viii, 56 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Algoritmos adaptativos

I. COPPE/UFRJ II. Título (série).

Agradecimentos

Aos professores Antonio Petraglia e Mariane Rembold Petraglia pela orientação, paciência e dedicação indispensáveis demonstrados durante toda a elaboração deste trabalho.

Aos professores e funcionários do Programa de Engenharia Elétrica da COPPE/UFRJ por terem contribuído para a conclusão de mais uma etapa da minha vida acadêmica.

À minha família pelo apoio e incentivo em todos os momentos.

Aos meus amigos que, de qualquer modo, tenham contribuído para a conclusão deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ADAPTAÇÃO DIGITAL DE UM FILTRO A CAPACITOR CHAVEADO PROGRAMÁVEL

Eduardo Barbosa Moura Costa

Fevereiro/2005

Orientadores: Mariane Rembold Petraglia
Antonio Petraglia

Programa: Engenharia Elétrica

Neste trabalho são pesquisados métodos para a adaptação digital de um filtro a capacitor chaveado programável. A topologia do filtro é composta de blocos básicos formados por seções FIR de segunda ordem. Estuda-se a estrutura do filtro analógico e as possibilidades de adaptação na forma cascata e de variação na taxa de adaptação em relação à taxa de amostragem do sinal de entrada. Desenvolvem-se e analisam-se as superfícies de desempenho para estruturas de filtragem nas formas cascata e direta. Finalmente, sugere-se um algoritmo de adaptação considerando questões estruturais e de custo.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DIGITAL ADAPTATION OF A PROGRAMMABLE SWITCHED CAPACITOR FILTER

Eduardo Barbosa Moura Costa

February/2005

Advisors: Mariane Rembold Petraglia
Antonio Petraglia

Department: Electrical Engineering

In this work some methods for the digital adaptation of a programmable switched-capacitor filter are developed. The filter topology is composed of second-order FIR sections. The analog filter structure is studied and so are the possibilities of cascade form adaptation and variable rate adaptation with respect to the input signal sample rate. Performance surfaces are developed and analised for cascade and direct-form filter structures. Finally, an adaptive algorithm is suggested considering structural and cost issues.

Sumário

Lista de Figuras

Capítulo 1

Introdução

No decorrer do desenvolvimento de um projeto, o engenheiro deve estar ciente de todas as variáveis às quais seu sistema será submetido. Mas o que fazer quando estas variáveis são desconhecidas, ou pior, quando mudam a cada momento? O desafio reside em projetar um sistema que se adapte às novas condições e que se ajuste automaticamente, sem a necessidade de intervenção de um operador humano.

Algoritmos adaptativos são utilizados em diversas áreas: desde aplicações domésticas até sistemas militares. Com a popularização e o incremento no uso de sistemas de telecomunicações, tais algoritmos estão presentes no dia-a-dia das pessoas.

Alguns exemplos de aplicações nas quais se observam o uso de algoritmos adaptativos são: telefonia móvel e fixa, equalização de canais de comunicação, sistemas de cancelamento de ruído, entre outros.

O desenvolvimento de sistemas digitais tem-se popularizado nas últimas décadas como resultado dos avanços na produção de processadores de sinais digitais extremamente velozes. Este fato possibilitou o surgimento de uma ampla gama de produtos e aplicações, cujo custo é diretamente proporcional à sua velocidade.

No entanto, a utilização de sistemas analógicos ainda é freqüente como conseqüência de características bastante atraentes: baixo consumo de potência, menor área de integração e, conseqüentemente, menor custo. Além disso, apesar dos avanços obtidos no desenvolvimento de circuitos digitais, seus equivalentes analógicos ainda são a única solução

possível em aplicações com rigorosos requisitos de velocidade.

Filtros a capacitores chaveados possuem a característica de poderem ser programáveis, ou seja, seus coeficientes podem ser modificados durante sua operação. Por este motivo, o desenvolvimento de algoritmos adaptativos para esta classe de filtros analógicos torna-se bastante promissor.

Por outro lado, a implementação de algoritmos adaptativos no domínio analógico é dificultada pelo surgimento de *offsets* DC durante o processo de adaptação [1]. Desse modo, opta-se por uma estrutura analógica-digital mista, onde se pode processar (filtrar) sinais no domínio analógico e utilizar algoritmos adaptativos digitais ao mesmo tempo.

Esta estratégia gera a necessidade de conversores A/D e D/A operando em freqüências altas, aumentando o consumo e a área de integração. Surge, então, a necessidade de um algoritmo adaptativo que opere em uma taxa inferior à do sistema hospedeiro.

No decorrer deste trabalho, será apresentado um algoritmo adaptativo especialmente projetado para uma estrutura IIR cascata, com restrições quanto à obtenção do estado interno do filtro e que permite a subamostragem dos sinais de entrada e de erro. Desse modo, o requisito quanto à rapidez/consumo dos conversores A/D e D/A é relaxada uma vez que os únicos sinais analógicos necessários para o funcionamento do algoritmo serão subamostrados.

No Capítulo 2 será apresentada a teoria de algoritmos adaptativos, recordando conceitos de filtragem digital, passando pelo algoritmo *steepest descent* e citando dois dos principais tipos de algoritmos adaptativos utilizados atualmente. A seguir, serão abordadas as superfícies de desempenho para algoritmos baseados na minimização do erro quadrático, tópico este útil para a avaliação do funcionamento dos algoritmos nos capítulos que se seguem. O Capítulo 3 apresenta os casos de teste que foram utilizados para se confirmar qual a estrutura adaptativa adequada para a realização da adaptação do filtro descrito. O Capítulo 4 utiliza os resultados do Capítulo 3 para enunciar os requisitos, a teoria e o funcionamento do algoritmo adaptativo desenvolvido. Resultados de simulações efetuadas e discussões sobre o funcionamento do algoritmo também são apresentados no Capítulo 4, seguidos finalmente pelo capítulo final de conclusão.

Capítulo 2

Teoria básica de filtragem adaptativa

A classe de filtros digitais abordada neste texto restringe-se à dos filtros lineares com coeficientes reais. Deste modo, um filtro digital implementa a equação a diferenças:

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) - \sum_{i=1}^{M-1} a_i y(n-i) \tag{2.1}$$

ou, utilizando a transformada Z:

$$\frac{Y(z)}{X(z)} = \frac{B(z)}{A(z)} \tag{2.2}$$

onde $B(z) = \sum_{i=0}^{N-1} b_i z^{-i}$ e $A(z) = 1 + \sum_{i=1}^{M-1} a_i z^{-i}$.

Um algoritmo adaptativo possibilita a "regulagem" dos coeficientes a_i e b_i de determinado filtro digital. Para tanto, é necessária uma base de referência, ou seja, um sinal que diga ao algoritmo que direção o ajuste dos coeficientes deve tomar. O que parece mais lógico a se fazer é obter o sinal desejado $d(n)$ e compará-lo com o sinal de saída do filtro sendo adaptado, $y(n)$, gerando-se assim um sinal de erro, $e(n) = d(n) - y(n)$. Este sinal de erro servirá ao algoritmo adaptativo para reajustar os coeficientes do filtro (figura 2.1).

O procedimento continua até um número máximo pré-determinado de iterações, até que a função custo assuma um patamar considerado aceitável, ou mesmo indefinidamente.

A teoria utilizada no desenvolvimento de algoritmos adaptativos remonta à teoria de filtros Wiener [2], que são de uma classe de filtros lineares ótimos discretos no tempo. Estes filtros são otimizados levando-se em consideração uma *função custo* previamente definida.

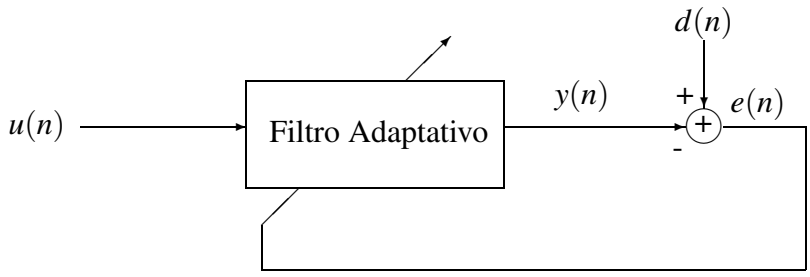


Figura 2.1: Estrutura básica de um filtro adaptativo

Com base no sinal de erro, o algoritmo adaptativo definirá uma função custo $J(e(n))$, que deverá ser minimizada, ou seja, o algoritmo deverá convergir para o ponto onde

$$\frac{\partial J(e(n))}{\partial \mathbf{w}(n)} = \mathbf{0} \tag{2.3}$$

sendo $\mathbf{w}(n)$ o vetor de coeficientes do filtro sendo adaptado. O desenvolvimento de cada algoritmo depende fortemente da forma da função custo $J(e(n))$.

A função custo normalmente é derivada a partir de um sinal de erro, $e(n) = d(n) - y(n)$ (figura 2.1), onde $y(n)$ é o sinal de saída do sistema e $d(n)$ é o sinal *desejado*, isto é, o sinal que o sistema deveria estar apresentando no instante de tempo n.

O sinal de erro $e(n)$ é utilizado porque ele nos indica o quão próximos (ou distantes) estamos da solução ótima. Como se trata de um problema de otimização, a forma que a função custo toma costuma ser uma função do sinal de erro. Mais especificamente, é comumente utilizado o valor esperado ($E[\cdot]$) do quadrado do sinal de erro, ou seja, o *erro médio quadrático*, $E[e^2(n)]$ como função custo do algoritmo adaptativo.

2.1 Algoritmo *steepest-descent*

Definida uma função custo como referência para avaliação da eficiência do filtro, deve-se desenvolver um método que a utilize para formar o algoritmo adaptativo.

O algoritmo *steepest-descent*[2] faz uso, a cada iteração, da função custo definida para o ajuste dos coeficientes do filtro em questão. Através de sucessivas iterações o algoritmo converge para um ponto ótimo, onde a função custo é minimizada. Ele consiste dos seguintes passos:

1. Estabelece-se um valor inicial (estimativa) para o vetor $\mathbf{w}(n)$ de coeficientes sendo adaptado.

2. Calcula-se o gradiente da função custo, $J(n)$, em relação ao vetor de coeficientes, $\mathbf{w}(n)$.

3. Incrementa-se $\mathbf{w}(n)$ na direção oposta à do gradiente calculado no item anterior.

4. Retorna ao passo 2 e repete-se o procedimento.

A equação de atualização utilizada pelo algoritmo *steepest-descent* é dada abaixo:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla J(n)] \tag{2.4}$$

O cálculo do vetor gradiente $\nabla J(n)$ leva em consideração uma estrutura FIR, ou seja:

$$y(n) = \sum_{i=0}^{M-1} w_i x(n-i) \tag{2.5}$$

Deste modo, o vetor gradiente $\nabla J(n)$ é obtido da seguinte forma:

$$\nabla J(n) = \left[\frac{\partial J(n)}{\partial w_0(n)} \quad \frac{\partial J(n)}{\partial w_1(n)} \quad \cdots \quad \frac{\partial J(n)}{\partial w_{M-1}(n)}\right]^T = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n) \tag{2.6}$$

onde

$$\mathbf{p} = E[\mathbf{x}(n)d(n)] \tag{2.7}$$

$$\mathbf{R} = E[\mathbf{x}(n)\mathbf{x}^T(n)] \tag{2.8}$$

$$\mathbf{x}(n) = \Big[x(n) \quad x(n-1) \quad \cdots \quad x(n-M+1)\Big]^T \tag{2.9}$$

Diversos algoritmos se utilizam do método *steepest-descent* para atualização de coeficientes, por sua extrema simplicidade. Pode-se citar o algoritmo de Newton como outro pertencente à mesma classe, mais complexo, porém mais rápido [2].

Embora à primeira vista o algoritmo *steepest-descent* pareça simples, ele necessita do conhecimento prévio da estatística do sinal de entrada, $x(n)$, que nem sempre é conhecida ou está disponível. Na seção a seguir serão apresentados os algoritmos LMS e RLS, que não fazem uso da estatística do sinal e que, por este motivo, são mais freqüentemente utilizados.

2.2 Algoritmos mais comuns

A tarefa dos algoritmos adaptativos de procurar pela solução ótima reduz-se à minimização de uma determinada função custo $J(n)$. Com base nesta função custo diversos tipos de algoritmos adaptativos vêm sendo desenvolvidos ao longo dos anos, cujo destaque pode ser dado aos dois mais utilizados:

- LMS – *Least Mean Square*
- RLS – *Recursive Least-Squares*

2.2.1 LMS – *Least Mean Square*

O algoritmo LMS, desenvolvido em 1960 por Widrow e Hoff, é utilizado como padrão de referência com o qual outros algoritmos adaptativos são comparados [2].

O algoritmo LMS procura minimizar o erro médio quadrático:

$$J(n) = E[(d(n) - y(n))^2] = E[e^2(n)] \tag{2.10}$$

No entanto, o conhecimento do valor esperado do sinal de erro não está disponível inicialmente, o que leva à consideração de estimativas instantâneas das estatísticas dos sinais. Sendo assim, o gradiente da função custo $J(n)$ pode ser calculado da seguinte forma:

$$\hat{\nabla} J(n) = -2\hat{\mathbf{p}} + \hat{\mathbf{R}}\mathbf{w} \tag{2.11}$$

onde

$$\hat{\mathbf{R}}(n) = \mathbf{x}(n)\mathbf{x}^T(n) \tag{2.12}$$

$$\hat{\mathbf{p}}(n) = \mathbf{x}(n)d(n) \tag{2.13}$$

Comparando com o algoritmo *steepest-descent*, observa-se a semelhança entre as definições de $\hat{\mathbf{R}}(n)$ e $\mathbf{R}(n)$, assim como entre $\hat{\mathbf{p}}(n)$ e $\mathbf{p}(n)$. A diferença reside na eliminação do operador de valor esperado, dado o desconhecimento da estatística do sinal de entrada,

$x(n)$. Portanto, $\hat{\mathbf{R}}(n)$ e $\hat{\mathbf{p}}(n)$ são apenas estimativas das matrizes $\mathbf{R}(n)$ e $\mathbf{p}(n)$, respectivamente. Utilizando a mesma regra iterativa do algoritmo *steepest descent*, a equação de atualização do algoritmo LMS é dada por:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e(n) \tag{2.14}$$

Dada a equação de atualização, verifica-se que o algoritmo LMS possui complexidade computacional $O(N)$, onde N é o número de coeficientes adaptados.

Critério de convergência

Pode ser mostrado que o algoritmo LMS converge para a solução ótima $\mathbf{w}_o$ quando [2]:

$$0 < \mu < \frac{2}{\sum_{i=0}^{M-1} E[|x(n-i)|^2]} \tag{2.15}$$

Desajuste

Define-se o erro médio quadrático em excesso $J_{\text{ex}}(n)$ como sendo a diferença entre o erro médio quadrático em um instante de tempo n, $J(n)$, e seu valor ótimo, J_{min}, obtido com o valor correto do vetor de coeficientes, $\mathbf{w}_o$:

$$J_{\text{ex}}(n) = J(n) - J_{\text{min}} \tag{2.16}$$

O desajuste do algoritmo LMS é obtido por:

$$\mathcal{M} = \frac{J_{\text{ex}}(\infty)}{J_{\text{min}}} = \frac{\mu}{2} \sum_{i=0}^{M-1} E[|x(n-i)|^2] \tag{2.17}$$

2.2.2 RLS – *Recursive Least-Squares*

O método RLS define como sendo sua função custo a expressão:

$$J(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2 \tag{2.18}$$

onde λ, uma constante positiva próxima, porém menor que 1, é definida como o "fator de esquecimento" do algoritmo. De fato, o parâmetro λ regula o "peso" de estimativas passadas do sinal de erro, considerando-as no resultado final da função custo $J(n)$.

O funcionamento do algoritmo RLS é dado abaixo:

1. $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$
2. $\mathbf{w}(0) = \mathbf{0}$
3. $\mathbf{k}(n) = \dfrac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{x}(n)}{1+\lambda^{-1}\mathbf{x}^T(n)\mathbf{P}(n-1)\mathbf{u}(n)}$
4. $e(n) = d(n) - \mathbf{w}^T(n-1)\mathbf{x}(n)$
5. $\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)e(n)$
6. $\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{x}^T(n)\mathbf{P}(n-1)$
7. voltar ao passo 3 e repetir iteração.

A complexidade computacional do algoritmo RLS, apesar de também crescer linearmente com o número de coeficientes adaptados, é superior à do algoritmo LMS por envolver um número maior de operações com matrizes, necessárias para os cálculos de $\mathbf{k}(n)$ e $\mathbf{P}(n)$. No entanto, o algoritmo RLS tende a convergir mais rapidamente que o LMS para a solução correta.

2.3 Adaptação IIR

Uma função de transferência é dita IIR quando sua resposta impulsional é infinita. Considere o sistema:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{N(z)}{D(z)} = \frac{\sum_{i=0}^{N-1} b_i z^{-i}}{1+\sum_{i=1}^{M-1} a_i z^{-i}} \tag{2.19}$$

ou, no domínio do tempo:

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) - \sum_{i=1}^{M-1} a_i y(n-i) \tag{2.20}$$

$$= \mathbf{b}^T\mathbf{x}(n) - \mathbf{a}^T\mathbf{y}(n-1) \tag{2.21}$$

onde $\mathbf{a} = \begin{bmatrix} a_1 & \cdots & a_{M-1} \end{bmatrix}^T$ e $\mathbf{b} = \begin{bmatrix} b_0 & \cdots & b_{N-1} \end{bmatrix}^T$.

Utilizando-se o método LMS, define-se a função custo:

$$J(n) = |e(n)|^2 \tag{2.22}$$

e o vetor $\mathbf{p}(n)$:

$$\mathbf{p}(n) = \begin{bmatrix} \mathbf{b}^T(n) & \mathbf{a}^T(n) \end{bmatrix}^T \tag{2.23}$$

A partir da equação 2.22, derivam-se os cálculos do vetor gradiente $\nabla J(n)$:

$$\begin{aligned} \nabla J(n) &= 2e(n)\frac{\partial e(n)}{\partial \mathbf{p}(n)} \\ &= -2e(n)\begin{bmatrix} \frac{\partial y(n)}{\partial b_0(n)} & \cdots & \frac{\partial y(n)}{\partial b_{N-1}(n)} & \frac{\partial y(n)}{\partial a_1(n)} & \cdots & \frac{\partial y(n)}{\partial a_{M-1}(n)} \end{bmatrix}^T \end{aligned} \tag{2.24}$$

O cálculo das derivadas parciais $\frac{\partial y(n)}{\partial b_j(n)}$ e $\frac{\partial y(n)}{\partial a_i(n)}$ é realizado da seguinte forma [3]:

$$\frac{\partial y(n)}{\partial a_i(n)} = -y(n-i) - \sum_{j=1}^{M-1} a_j(n)\frac{\partial y(n-j)}{\partial a_i(n)} \tag{2.25}$$

$$\frac{\partial y(n)}{\partial b_j(n)} = x(n-j) - \sum_{i=1}^{M-1} a_i(n)\frac{\partial y(n-i)}{\partial b_j(n)} \tag{2.26}$$

onde $i = 1,2,\ldots,M-1$ e $j = 0,1,\ldots,N-1$.

O cálculo das derivadas parciais $\frac{\partial y(n-i)}{\partial b_j(n)}$ e $\frac{\partial y(n-j)}{\partial a_i(n)}$ não é uma tarefa simples. Pode-se considerar que, se o passo de adaptação for suficientemente pequeno,

$$a_i(n) \approx a_i(n-j) \qquad \text{para } i,j = 1,2,\ldots,M \tag{2.27}$$

$$b_j(n) \approx b_j(n-i) \qquad \text{para } j = 1,2,\ldots,N \text{ e } i = 1,2,\ldots,M \tag{2.28}$$

Neste caso,

$$\frac{\partial y(n)}{\partial a_i(n)} \approx -y(n-i) - \sum_{j=1}^{M-1} a_j(n)\frac{\partial y(n-j)}{\partial a_i(n-j)} \tag{2.29}$$

$$\frac{\partial y(n)}{\partial b_j(n)} \approx x(n-j) - \sum_{i=1}^{M-1} a_i(n)\frac{\partial y(n-i)}{\partial b_j(n-i)} \tag{2.30}$$

As equações 2.29 e 2.30 tornam possível o cálculo das derivadas parciais através de equações a diferenças, com sinais de entrada $y(n)$ e $x(n)$, respectivamente. Equivalentemente,

$$s_{a_j}(n) = -y(n-j) - \sum_{i=1}^{M-1} a_i(n)s_{a_j}(n-i) \tag{2.31}$$

$$s_{b_j}(n) = x(n-j) - \sum_{i=1}^{M-1} a_i(n)s_{b_j}(n-i) \tag{2.32}$$

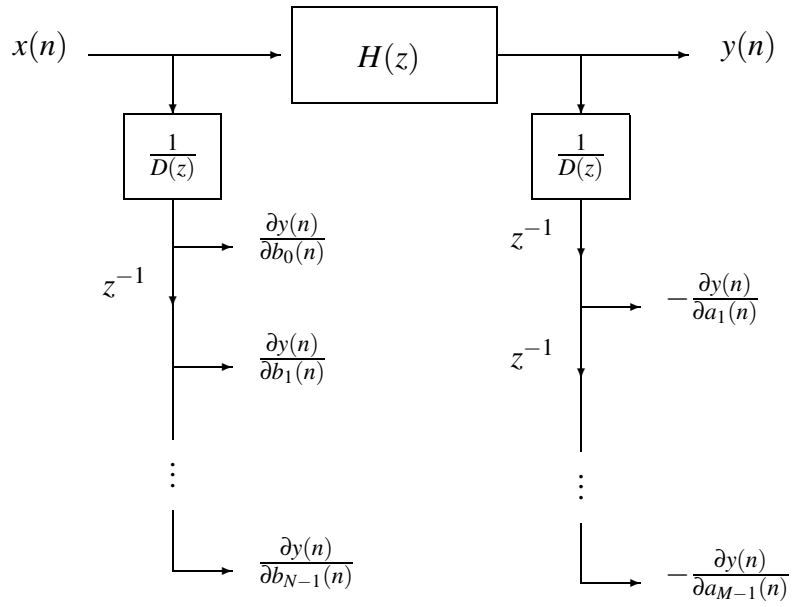


Figura 2.2: Cálculo das derivadas parciais em uma estrutura IIR

ou, utilizando a transformada Z:

$$S_{a_j(n)}(z) = \frac{-z^{-j}Y(z)}{1+\sum_{i=1}^{M-1} a_i(n)z^{-i}} = -z^{-j}\frac{Y(z)}{D(z)} \tag{2.33}$$

$$S_{b_j(n)}(z) = \frac{z^{-j}X(z)}{1+\sum_{i=1}^{M-1} a_i(n)z^{-i}} = z^{-j}\frac{X(z)}{D(z)} \tag{2.34}$$

Finalmente, pode-se obter o valor (aproximado) do vetor gradiente $\nabla J(n)$ pelas equações 2.33 e 2.34. A figura 2.2 mostra graficamente o cálculo das derivadas parciais de $y(n)$ em relação aos coeficientes $a_i(n)$ e $b_i(n)$.

A equação de adaptação é, portanto:

$$\mathbf{p}(n+1) = \mathbf{p}(n) + 2e(n)\mathbf{S}(n) \tag{2.35}$$

onde $\mathbf{S}(n) = \begin{bmatrix} -s_{b_0}(n) & \cdots & -s_{b_{N-1}}(n) & s_{a_1}(n) & \cdots & s_{a_{M-1}}(n) \end{bmatrix}^T$.

2.3.1 Adaptação IIR na forma cascata

Uma estrutura na forma cascata é obtida conectando-se seções IIR, de qualquer ordem, em série. Desta forma, a função de transferência resultante é representada pelo produto das funções de transferência individuais de cada seção.

Pode-se representar uma função de transferência $H(z)$, de ordem N, como o produto de seções de 2ª ordem. No caso de N ser ímpar, utilizam-se $\lfloor N/2 \rfloor$ seções de 2ª ordem mais uma seção de 1ª ordem.

No caso geral, tem-se que a função de transferência realizada numa estrutura cascata composta por K seções de 2ª ordem é dada por:

$$H(z) = \prod_{i=1}^{K} \frac{b_{0i}(n) + b_{1i}(n)z^{-1} + b_{2i}(n)z^{-2}}{1 + a_{0i}(n)z^{-1} + a_{1i}(n)z^{-2}} = \prod_{i=1}^{K} H_i(z) = \prod_{i=1}^{K} \frac{N_i(z)}{D_i(z)} \tag{2.36}$$

Utilizando os conceitos desenvolvidos na Seção 2.3, é possível derivar um algoritmo adaptativo para a estrutura cascata. Considere o vetor de coeficientes $\mathbf{p}(n)$:

$$\mathbf{p}(n) = \Big[b_{00}(n) \quad b_{10}(n) \quad b_{20}(n) \quad a_{10}(n) \quad a_{20}(n) \quad \cdots \quad b_{0K}(n) \quad b_{1K}(n) \quad b_{2K}(n) \quad a_{1K}(n) \quad a_{2K}(n) \Big]^T \tag{2.37}$$

O vetor de derivadas parciais $\frac{\partial y(n)}{\partial \mathbf{p}(n)}$ tem a forma:

$$\frac{\partial y(n)}{\partial \mathbf{p}(n)} = \Big[\frac{\partial y(n)}{\partial b_{00}(n)} \quad \frac{\partial y(n)}{\partial b_{10}(n)} \quad \frac{\partial y(n)}{\partial b_{20}(n)} \quad \frac{\partial y(n)}{\partial a_{10}(n)} \quad \frac{\partial y(n)}{\partial a_{20}(n)} \quad \cdots \quad \frac{\partial y(n)}{\partial b_{0K}(n)} \quad \frac{\partial y(n)}{\partial b_{1K}(n)} \quad \frac{\partial y(n)}{\partial b_{2K}(n)} \quad \frac{\partial y(n)}{\partial a_{1K}(n)} \quad \frac{\partial y(n)}{\partial a_{2K}(n)} \Big]^T \tag{2.38}$$

Utilizando o desenvolvimento realizado na seção 2.3, tem-se que, para a k-ésima seção de 2ª ordem:

$$\frac{\partial y_k(n)}{\partial a_{jk}(n)} = -y_k(n-j) - \sum_{i=1}^{M-1} a_{ik}(n) \frac{\partial y_k(n)}{\partial a_{jk}(n)} \qquad 1 \le j \le 2 \tag{2.39}$$

$$\frac{\partial y_k(n)}{\partial b_{jk}(n)} = y_{k-1}(n-j) - \sum_{i=1}^{M-1} a_{ik}(n) \frac{\partial y_k(n)}{\partial b_{jk}(n)} \qquad 0 \le j \le 2 \tag{2.40}$$

onde os sinais $y_k(n)$ representam a saída da k-ésima seção, sendo $y_0(n) = x(n)$ e $y_K(n) = y(n)$, como indicado na figura 2.3.

Sejam $S_{a_{jk}(n)}$ e $S_{b_{jk}(n)}$ as transformadas Z das derivadas parciais $\frac{\partial y(n)}{\partial a_{jk}(n)}$ e $\frac{\partial y(n)}{\partial b_{jk}(n)}$, respectivamente. Utilizando o resultado acima:

$$S_{a_{jk}(n)} = -z^{-j} \frac{X(z)}{D_k(z)} \prod_{i=k}^{K} H_i(z) \tag{2.41}$$

$$S_{b_{jk}(n)} = z^{-j} \frac{X(z)}{D_k(z)} \prod_{i=k}^{K} H_i(z) \tag{2.42}$$

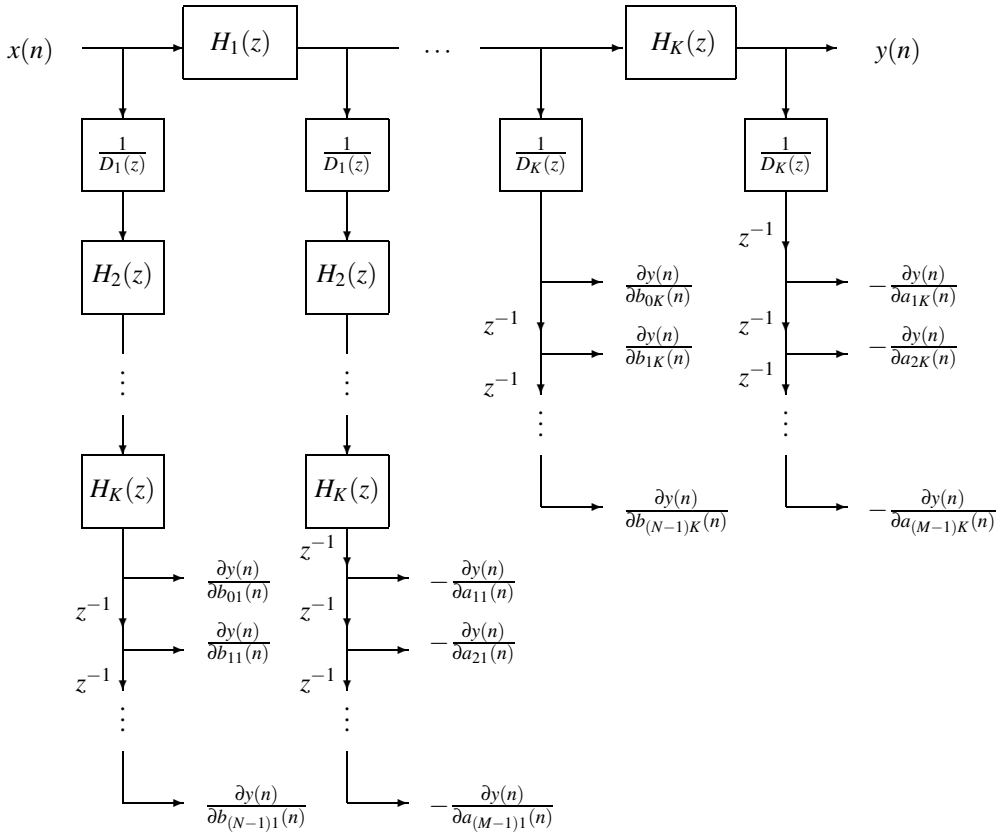


Figura 2.3: Cálculo das derivadas parciais em uma estrutura IIR cascata

2.3.2 Estabilidade de filtros IIR

A recursividade apresentada pelas estruturas IIR introduz um problema inexistente nas estruturas FIR: a possibilidade de instabilidade. Uma função de transferência $H(z)$ é considerada BIBO (*bounded input, bounded output*) estável se os coeficientes de sua resposta impulsional forem absolutamente somáveis:

$$\mathcal{S} = \sum_{n=-\infty}^{\infty} |h(n)| < \infty \tag{2.43}$$

Pode ser demonstrado que uma função de transferência $H(z)$ será estável se todos os seus pólos estiverem contidos dentro do círculo unitário do plano complexo z.

Considerando um polinômio em z de segunda ordem $D(z) = 1 + d_1 z^{-1} + d_2 z^{-2}$, é possível utilizar o critério de Jury [4] para determinar se as raízes de $D(z)$ estão contidas no interior do círculo unitário do plano complexo z:

$$|d_2| < 1 \tag{2.44}$$

$$|d_1| < 1 + d_2 \tag{2.45}$$

Se d_1 e d_2 estiverem localizados no interior da região delimitada pelas equações 2.44 e 2.45 (figura 2.4), as raízes do polinômio $D(z)$ possuirão módulo menor que 1. Estas duas inequações serão úteis mais adiante na detecção de pólos instáveis durante o processo de adaptação.

2.4 Superfície de desempenho

Filtros com estruturas IIR possuem vantagens sobre filtros FIR no que diz respeito ao número de coeficientes necessários para a realização de determinadas funções de transferência. A recursividade de estruturas IIR permite que estes filtros sejam capazes de realizar, com um número significativamente menor de coeficientes, a mesma função de transferência que seu equivalente FIR.

No entanto, se por um lado a simplicidade das funções de transferência de filtros IIR é atraente, por outro sua natureza recursiva introduz um problema que a categoria FIR

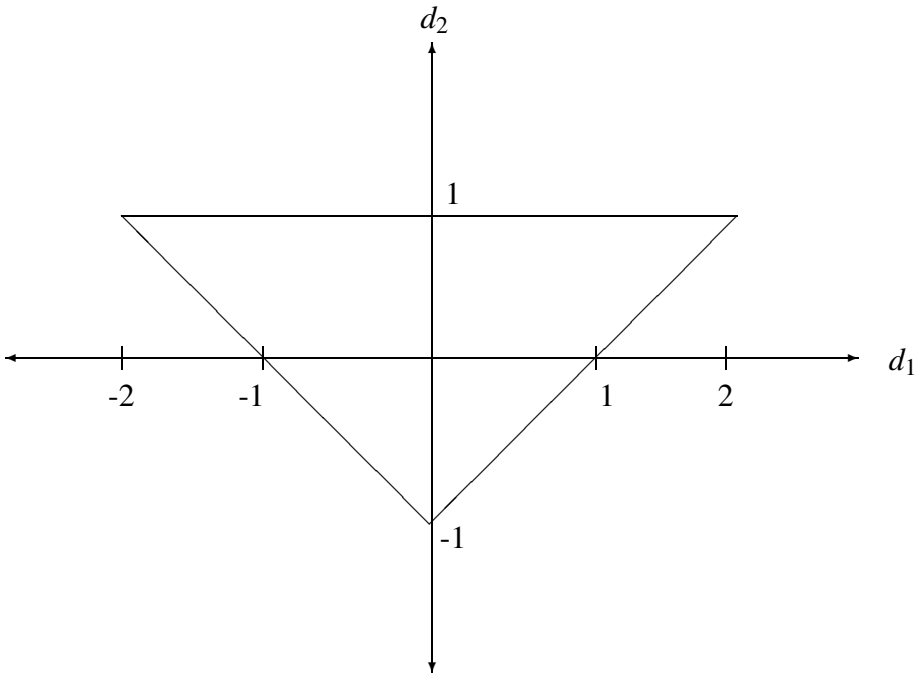


Figura 2.4: Triângulo de estabilidade

não possui: a possibilidade de instabilidade da função de transferência. Como visto na Seção 2.3.2, pode ser demonstrado que, se o módulo dos pólos de uma estrutura IIR não estiverem contidos dentro do círculo unitário do plano complexo z, o filtro será instável.

Este fato representa um sério risco ao funcionamento de algoritmos adaptativos atuando em estruturas IIR por modificarem constantemente o valor dos coeficientes do numerador e do denominador da função de transferência. Neste sentido, é necessário um estudo mais detalhado da característica das superfícies de desempenho de cada estrutura IIR de modo a prever e detectar problemas potenciais de convergência e de estabilidade em tais estruturas.

Estudos anteriores ([3], [5] e [6]) abordam propriedades das superfíceis de convergência de filtros IIR, além de se preocuparem-se com a implementação de funções de transferência nas formas cascata e paralela.

Considerando um algoritmo adaptativo baseado na minimização do erro médio quadrático, $J(n) = E[e^2(n)]$, pode-se derivar a forma da superfície de erro. Considerando um sistema adaptativo $G(z)$ cujo objetivo seja o de modelar uma estrutura IIR $H(z)$, o sinal

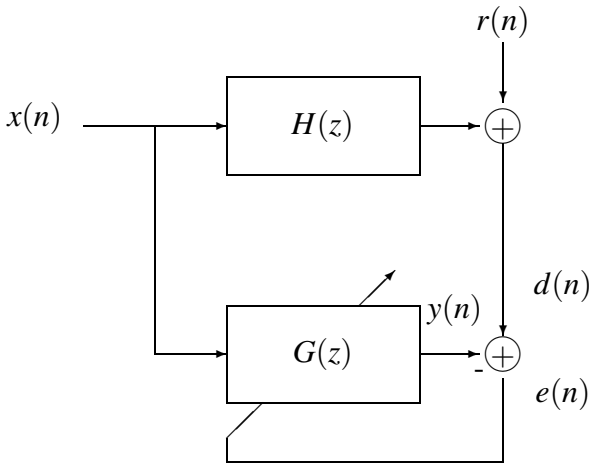


Figura 2.5: Identificação de sistema

desejado será modelado como:

$$D(z) = H(z)X(z) + R(z) \tag{2.46}$$

onde $D(z)$, $X(z)$ e $R(z)$ são as transformadas Z dos sinais desejado, de entrada e de ruído, respectivamente.

A superfície de desempenho é obtida através da expansão de $E[e^2(n)]$:

$$\begin{aligned} E[e^2(n)] &= E[(d(n) - y(n))^2] \\ &= E[d^2(n) - 2d(n)y(n) + y^2(n)] \end{aligned} \tag{2.47}$$

Considerando que para dois processos $p(n)$ e $q(n)$ estacionários com função de correlação cruzada $\Phi_{pq}(z)$:

$$E[p(n)q(n)] = \frac{1}{2\pi j} \oint_{\gamma} \Phi_{pq}(z) \frac{dz}{z} \tag{2.48}$$

onde γ é o círculo unitário do plano z no sentido anti-horário, a equação 2.47 pode ser reescrita como [3]:

$$E[e^2(n)] = \frac{1}{2\pi j} \oint_{\gamma} \left[|H(z) - G(z)|^2 \Phi_{xx}(z) + \Phi_{rr}(z) \right] \frac{dz}{z} \tag{2.49}$$

onde $\Phi_{xx}(z)$ e $\Phi_{rr}(z)$ são os espectros de potência dos sinais de entrada, $x(n)$, e de ruído, $r(n)$, respectivamente. No caso de $r(n) = 0$, a superfície reduz-se a [6]:

$$E[e^2(n)] = \frac{1}{2\pi j} \oint_{\gamma} |H(z) - G(z)|^2 \Phi_{xx}(z) \frac{dz}{z} \tag{2.50}$$

As integrais das equações 2.49 e 2.50, e conseqüentemente as equações da superfície de desempenho, podem ser resolvidas através do teorema dos resíduos de Cauchy [7].

2.5 Efeito da estrutura na superfície de desempenho

Uma função de transferência $H(z)$ pode ser realizada de diversas formas (direta, cascata, paralela, etc.). No entanto, a forma com a qual $H(z)$ é realizada interfere diretamente na forma da superfície de desempenho do filtro, com a introdução de pontos de sela (*saddle points*) [3].

Se duas estruturas forem equivalentes, suas funções custo serão iguais se

$$F_1(\mathbf{w}_1) = F_2(\mathbf{w}_2) = F_2(\mathbf{f}_3(\mathbf{w}_1)) \tag{2.51}$$

onde F_1 e F_2 são as funções custo de duas realizações equivalentes, $\mathbf{w}_1$ e $\mathbf{w}_2$ são os vetores de coeficientes destas duas realizações, na forma $\mathbf{w} = \begin{bmatrix} w_0 & w_1 & \cdots & w_{M-1} \end{bmatrix}^T$ e $\mathbf{f}_3(\mathbf{w}_1)$ é o conjunto de funções que realiza o mapeamento entre os coeficientes da realização 1 para a 2.

Desta forma, se $\mathbf{f}_3$ for diferenciável, pela regra da cadeia obtém-se:

$$\frac{\partial F_1(\mathbf{w}_1)}{\partial \mathbf{w}_1} = \frac{\partial F_2(\mathbf{f}_3(\mathbf{w}_1))}{\partial \mathbf{w}_1} = \frac{\partial F_2(\mathbf{f}_3(\mathbf{w}_1))}{\partial \mathbf{f}_3(\mathbf{w}_1)} \frac{\partial \mathbf{f}_3(\mathbf{w}_1)}{\partial \mathbf{w}_1} \tag{2.52}$$

Se $\mathbf{w}_2'$ for um ponto estacionário de $F_2(\mathbf{w}_2)$ então:

$$\left.\frac{\partial F_2(\mathbf{w}_2)}{\partial \mathbf{w}_2}\right|_{\mathbf{w}_2=\mathbf{w}_2'} = \mathbf{0} = \left.\frac{\partial F_1(\mathbf{w}_1)}{\partial \mathbf{w}_1}\right|_{\mathbf{w}_1=\mathbf{w}_1'} \tag{2.53}$$

onde $\mathbf{w}_2' = \mathbf{f}_3(\mathbf{w}_1')$.

No entanto, pode ocorrer o caso em que

$$\left.\frac{\partial F_2(\mathbf{f}_3(\mathbf{w}_1))}{\partial \mathbf{f}_3(\mathbf{w}_1)}\right|_{\mathbf{w}_1=\mathbf{w}_1''} = \mathbf{0} \tag{2.54}$$

mas

$$\left.\frac{\partial F_1(\mathbf{w}_1)}{\partial \mathbf{w}_1}\right|_{\mathbf{w}_1=\mathbf{w}_1''} \neq \mathbf{0} \tag{2.55}$$

O caso descrito nas equações 2.54 e 2.55 só será possível se $\mathbf{f}_3$ não for diferenciável em $\mathbf{w}_1 = \mathbf{w}_1''$, caso este no qual a regra da cadeia da equação 2.52 não se aplica. Pode ser demonstrado que os pontos estacionários $\mathbf{w}_2'' = \mathbf{f}_3(\mathbf{w}_1'')$ serão pontos de sela [3].

A detecção de pontos de sela pode ser realizada verificando se a matriz $\frac{\partial \mathbf{f}_3(\mathbf{w}_1)}{\partial \mathbf{w}_1}^{-1}$ é não-singular.

Capítulo 3

Casos de teste

O processo de escolha do melhor algoritmo adaptativo para o caso em estudo passou por etapas preliminares, nas quais foram estudadas metodologias que pudessem possibilitar um melhor funcionamento. Nestas etapas, buscou-se realizar o processo de adaptação na forma cascata, de modo a ajustar os coeficientes de cada seção de segunda ordem diretamente, ao invés de realizar a adaptação na forma direta. Alguns problemas foram detectados durante estes testes preliminares, onde as superfícies de desempenho obtidas foram insatisfatórias.

No decorrer deste capítulo, serão explicitados os motivos pelos quais resolveu-se adotar a abordagem de adaptação na forma direta, ao invés de na forma cascata (forma original do filtro).

Neste capítulo serão testadas algumas estruturas de filtros IIR semelhantes à estrutura do filtro alvo (Seção 4.1). O objetivo é verificar a estabilidade e a superfície de convergência de cada uma das estruturas testadas. Neste caso, portanto, não se lançará mão de modificações no algoritmo para adaptação em taxa reduzida.

Inicialmente será abordada a estrutura de um filtro IIR na forma direta e sua superfície de convergência. Em seguida serão adicionadas seções de 2^a ordem até que não seja mais possível a adaptação. O objetivo é demonstrar as razões pelas quais optou-se pela adaptação na forma direta ao invés da forma cascata.

Nos casos de testes que se seguem o objetivo é identificar uma planta de estrutura

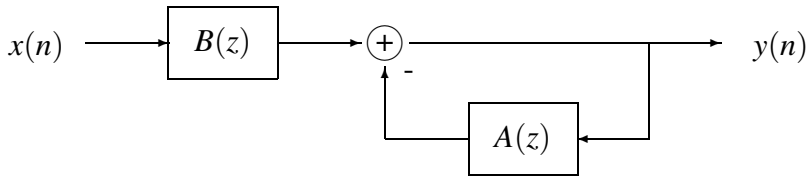


Figura 3.1: Filtro na forma direta

idêntica à do filtro adaptativo, utilizando-se como sinal de entrada ruído gaussiano de média zero e variância unitária.

3.1 Forma direta

Como primeiro caso de teste, será utilizada a forma direta apresentada na figura 3.1:

$$y(n) = \sum_{i=0}^{N-1} b_i x(n-i) - \sum_{i=1}^{M-1} a_i y(n-i) \tag{3.1}$$

ou, utilizando a transformada Z:

$$H(z) = \frac{B(z)}{1+A(z)} = \frac{\sum_{i=0}^{N-1} b_i z^{-i}}{1+\sum_{i=1}^{M-1} a_i z^{-i}} \tag{3.2}$$

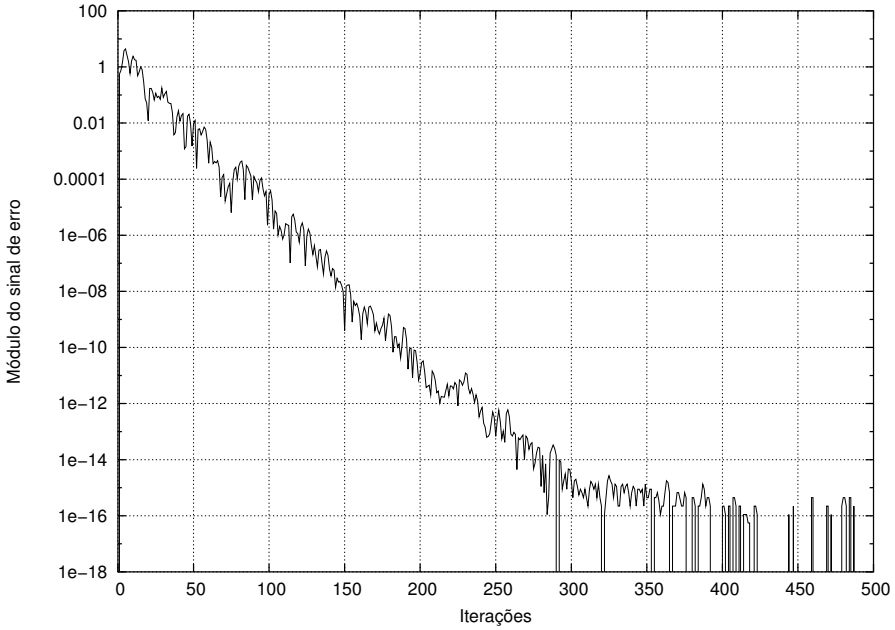
Fazendo $M = N = 3$, a estrutura reduz-se a uma seção FIR em cascata com uma seção só pólos IIR, ambas de 2ª ordem. A teoria da Seção 2.3 aplica-se diretamente a este caso. Os códigos dos programas de simulação podem ser vistos na Seção A.1.

3.1.1 Adaptação somente dos zeros

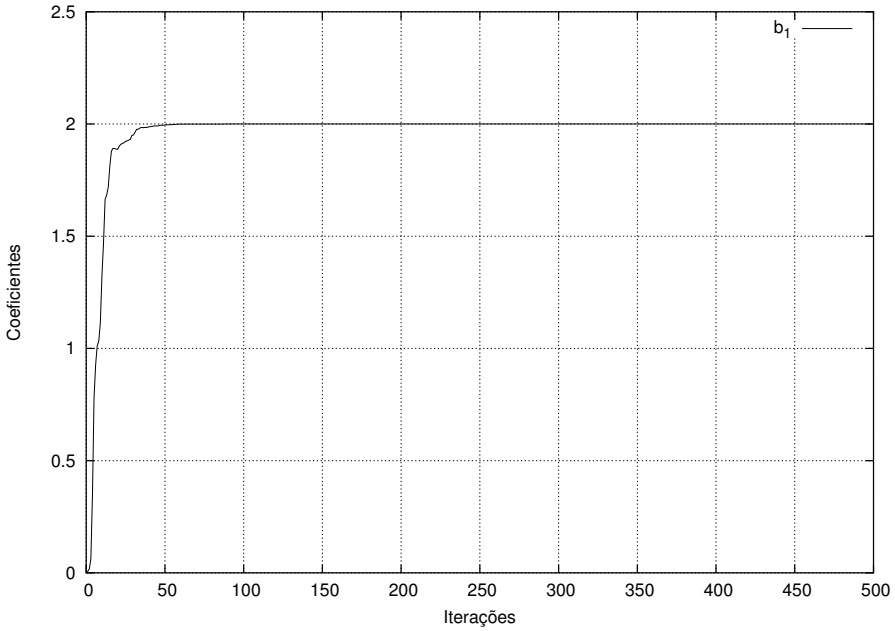
Inicialmente a adaptação será efetuada somente no coeficiente b_1, mantendo os pólos fixos. A função de transferência do filtro adaptativo resultante é, portanto:

$$G(z) = \frac{D(z)}{1+A(z)} = \frac{1+d_1 z^{-1}+z^{-2}}{1+a_1 z^{-1}+a_2 z^{-2}} \tag{3.3}$$

O sinal de erro e a evolução do processo de adaptação do coeficiente d_1 são exibidos nas figuras 3.2 (LMS) e 3.3 (RLS). Nota-se que o processo de adaptação ocorre sem problemas, com o sinal de erro quadrático atingindo um valor inferior a 10^{-8} a partir

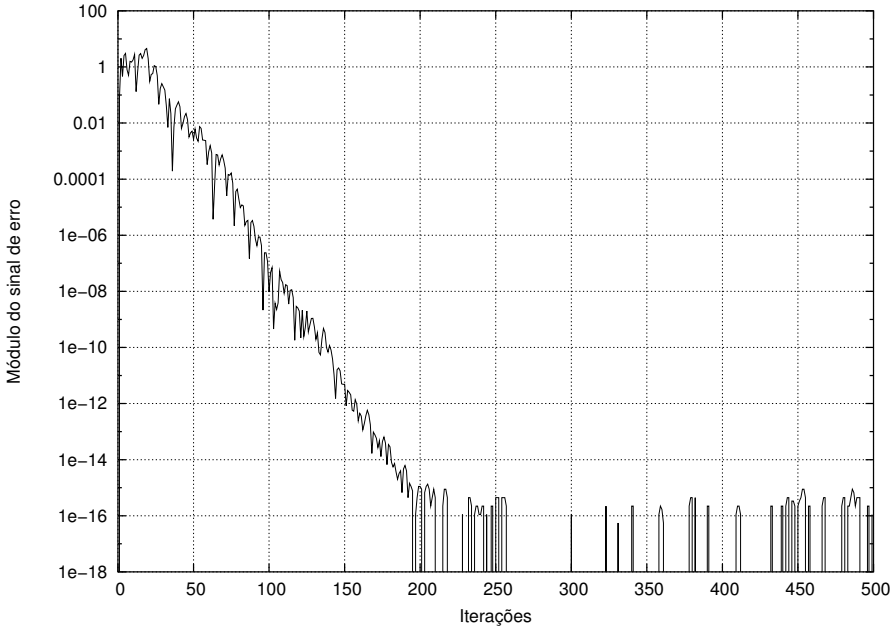


(a) sinal de erro

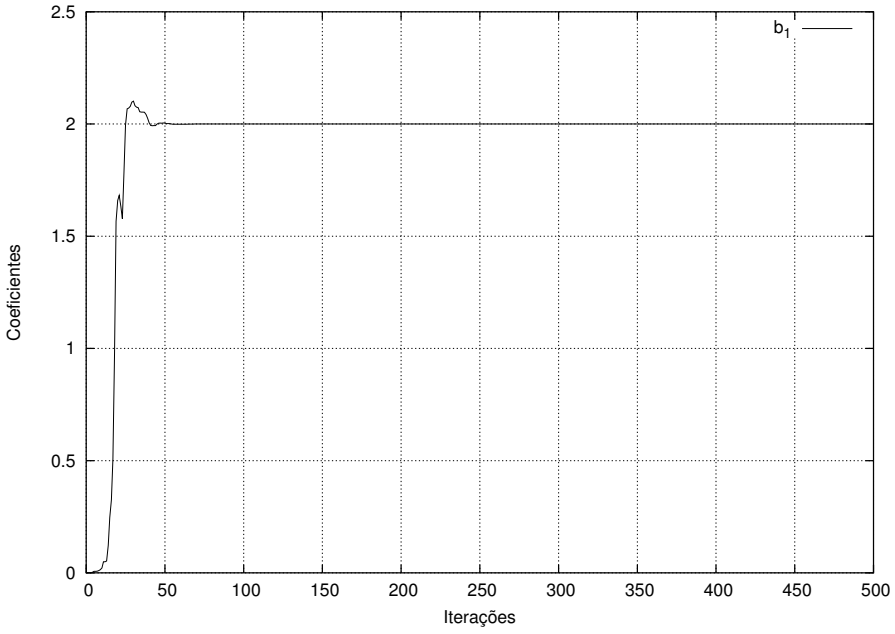


(b) coeficientes

Figura 3.2: Adaptação na forma direta (somente zeros) – LMS



(a) sinal de erro



(b) coeficientes

Figura 3.3: Adaptação na forma direta (somente zeros) – RLS

da 150ª iteração (LMS). Em comparação, o algoritmo RLS realiza a convergência do coeficiente b_1 com um erro inferior a 10^{-8} após 125 iterações.

Como o único coeficiente sendo adaptado é d_1, a função custo sendo minimizada é uma função de d_1. O cálculo da expressão da curva de desempenho é apresentado na próxima seção.

3.1.2 Adaptação dos pólos e zeros

Utilizando a mesma estrutura adaptativa da seção anterior, desta vez os pólos da função de transferência também serão adaptados. Iniciando a adaptação, verifica-se um erro de menos de 10^{-8} entre o sinal desejado e o calculado pelo filtro adaptativo a partir da 600ª iteração (figura 3.4) com o algoritmo LMS. Com o algoritmo RLS, os coeficientes convergem para os valores corretos com erro inferior a 10^{-8} após 350 iterações (figura 3.5).

Considere que

$$G(z) = \frac{D(z)}{1+C(z)} = \frac{1+d_1 z^{-1}+z^{-2}}{1+c_1 z^{-1}+c_2 z^{-2}} \tag{3.4}$$

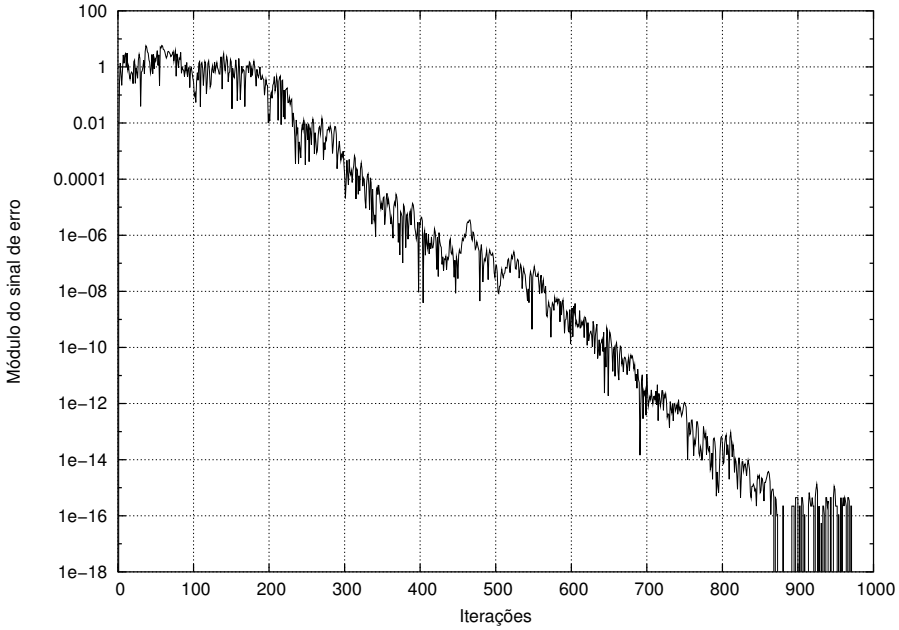
seja a função de transferência do filtro adaptativo tentando identificar os valores corretos dos coeficientes de $H(z)$: b_1, a_1 e a_2. A equação da superfície de desempenho será uma função de três variáveis, a saber: d_1, c_1 e c_2.

Considerando o desenvolvimento apresentado na Seção 2.4, procede-se agora à formulação da equação da superfície de desempenho para o caso da forma direta. Para simplificar o cálculo, será considerado que o sinal de entrada, $x(n)$, possui variância unitária e média zero e que o sinal de ruído é nulo. A tabela 3.1 apresenta o valor dos resíduos da expressão

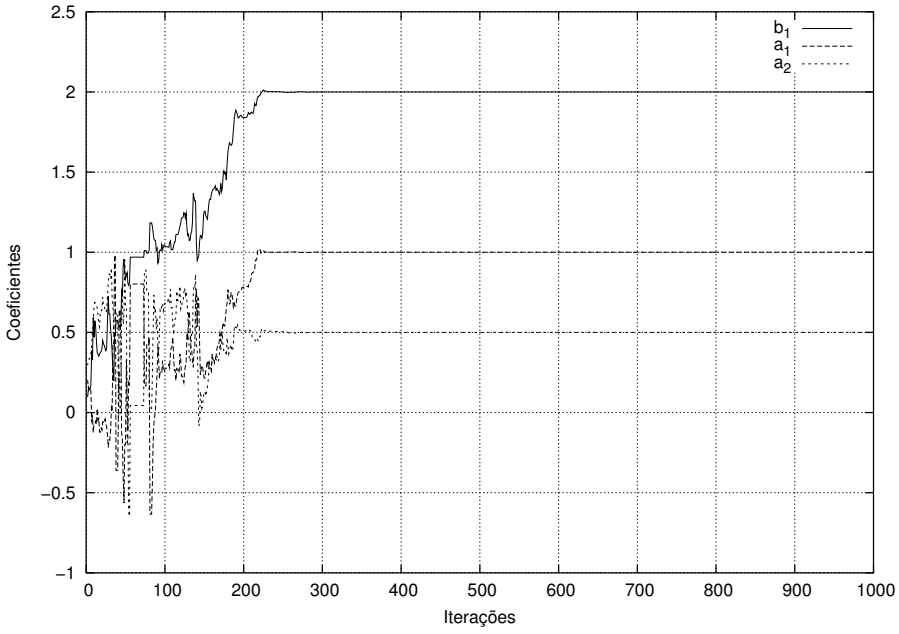
$$\frac{|H(z)-G(z)|^2}{z} = \frac{H(z)H(z^{-1})}{z} - \frac{2H(z)G(z^{-1})}{z} + \frac{G(z)G(z^{-1})}{z} \tag{3.5}$$

onde $\alpha 1$, α_2, β_1 e β_2 são os pólos de $H(z)$ e $G(z)$, respectivamente. A equação da superfície de desempenho é obtida através da soma dos resíduos dos três termos acima.

No caso do filtro da Seção 3.1.1, a superfície de desempenho será função somente do parâmetro d_1. Pelos valores relacionados na tabela 3.1, conclui-se que a superfície

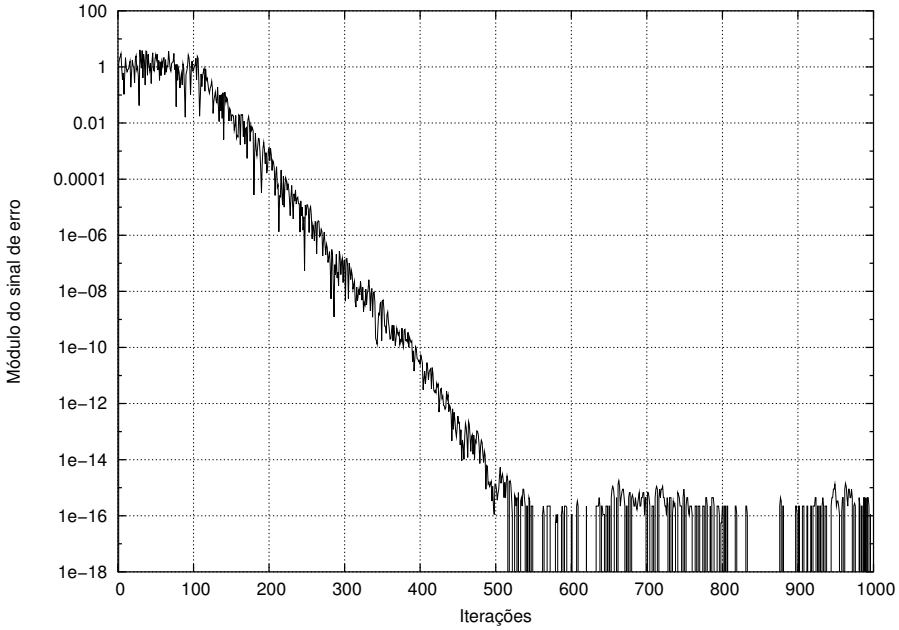


(a) sinal de erro

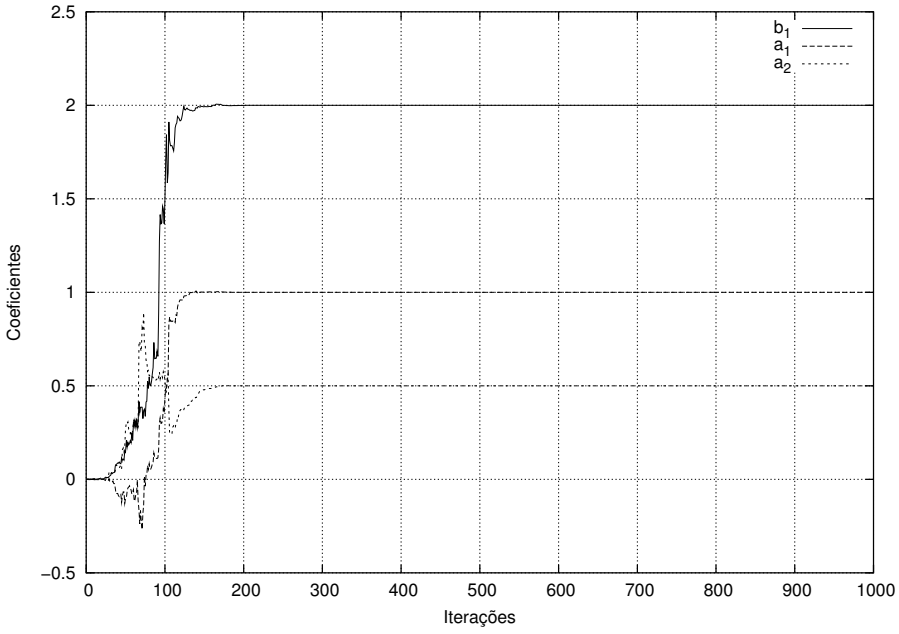


(b) coeficientes

Figura 3.4: Adaptação na forma direta – LMS



(a) sinal de erro



(b) coeficientes

Figura 3.5: Adaptação na forma direta – RLS

será, na verdade, uma função de 2º grau em d_1 possuindo, portanto, um único mínimo global. Desta forma não há possibilidade de incorreções para o processo de adaptação apresentado na seção 3.1.1.

3.2 Duas seções de 2ª ordem

A estrutura apresentada na seção anterior comportou-se bem durante o processo de adaptação. A partir de agora, serão adicionadas sucessivas seções FIR (*all-zeros*) de segunda ordem para avaliar o comportamento da estrutura cascata quando submetida à adaptação.

Neste caso, a função de transferência do filtro sendo adaptado será

$$H(z) = \frac{(1 + b_1 z^{-1} + z^{-2})(1 + b_2 z^{-1} + z^{-2})}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{3.6}$$

e a do filtro adaptativo

$$G(z) = \frac{(1 + d_1 z^{-1} + z^{-2})(1 + d_2 z^{-1} + z^{-2})}{1 + c_1 z^{-1} + c_2 z^{-2}} \tag{3.7}$$

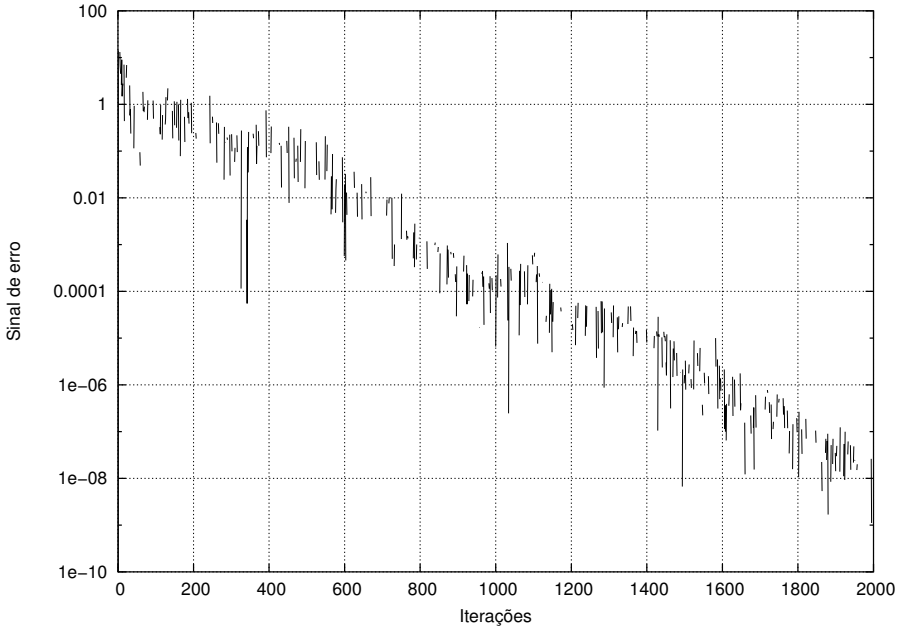
3.2.1 Adaptação dos zeros

O primeiro teste ocorre adaptando-se somente os zeros da função de transferência, ou seja, ajustando os coeficientes do denominador: d_1 e d_2. As figuras 3.6 e 3.7 apresentam a evolução dos sinais de erro e dos coeficientes durante o processo de adaptação utilizando os algoritmos LMS e RLS, respectivamente.

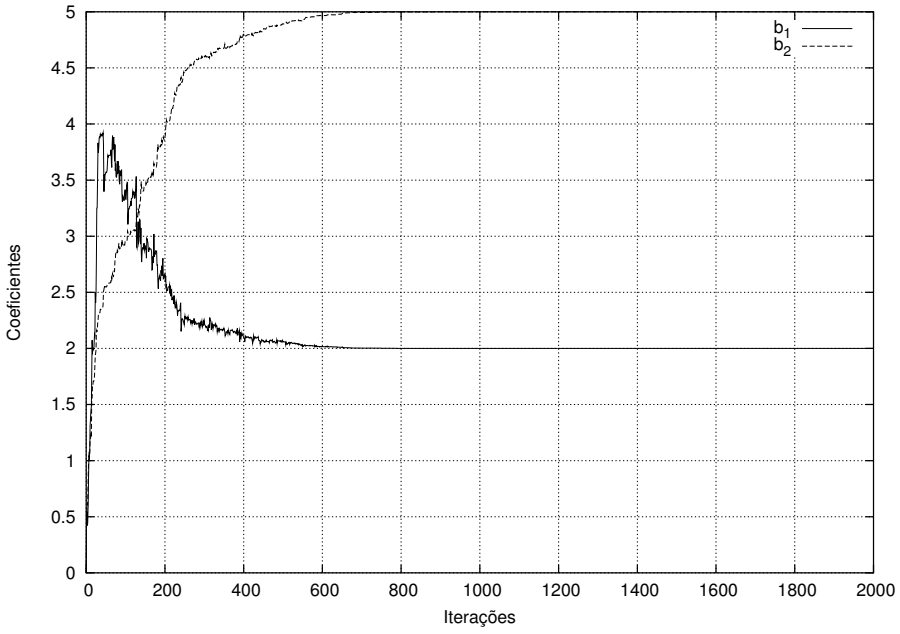
Observa-se que os coeficientes adaptados convergem corretamente para os valores desejados, com um erro menor que 10^{-6} após 1500 iterações para o algoritmo LMS e 650 iterações com o algoritmo RLS. Como esperado, a convergência é mais lenta do que a observada na Seção 3.1.1 devido à introdução de uma nova variável de adaptação, d_2. Na seção seguinte, os pólos também serão adaptados.

Tabela 3.1: Resíduos para composição da superfície de desempenho na forma direta

Termos	**Pólos**	**Resíduos**
$\frac{H(z)H(z^{-1})}{z}$	$z=0$	$\alpha_1^{-1}\alpha_2^{-1}$
	$z=\alpha_1$	$\frac{(1+b_1\alpha_1^{-1}+\alpha_1^{-2})(1+b_1\alpha_1+\alpha_1^2)}{\alpha_1(1-\alpha_2\alpha_1^{-1})(1-\alpha_1^2)(1-\alpha_1\alpha_2)}$
	$z=\alpha_2$	$\frac{(1+b_1\alpha_2^{-1}+\alpha_2^{-2})(1+b_1\alpha_2+\alpha_2^2)}{\alpha_2(1-\alpha_1\alpha_2^{-1})(1-\alpha_2^2)(1-\alpha_1\alpha_2)}$
	$z=\alpha_1^{-1}$	$\frac{(1+b_1\alpha_1^{-1}+\alpha_1^{-2})(1+b_1\alpha_1+\alpha_1^2)}{\alpha_1^{-1}(1-\alpha_1^2)(1-\alpha_2\alpha_1^{-1})(1-\alpha_2\alpha_1^{-1})}$
	$z=\alpha_2^{-1}$	$\frac{(1+b_1\alpha_2^{-1}+\alpha_2^{-2})(1+b_1\alpha_2+\alpha_2^2)}{\alpha_2^{-1}(1-\alpha_2^2)(1-\alpha_1\alpha_2^{-1})(1-\alpha_1\alpha_2^{-1})}$
$\frac{H(z)G(z^{-1})}{z}$	$z=0$	$\alpha_1^{-1}\alpha_2^{-1}$
	$z=\alpha_1$	$\frac{(\alpha_1^2+b_1\alpha_1+1)(\alpha_1^2+d_1\alpha_1+1)}{\alpha_1^2(\alpha_1-\alpha_2)(1-\beta_1\alpha_1)(1-\beta_2\alpha_1)}$
	$z=\alpha_2$	$\frac{(\alpha_2^2+b_1\alpha_2+1)(\alpha_2^2+d_1\alpha_2+1)}{\alpha_2^2(\alpha_2-\alpha_1)(1-\beta_1\alpha_2)(1-\beta_2\alpha_2)}$
	$z=\beta_1^{-1}$	$\frac{(1+b_1\beta_1+\beta_1^2)(\beta_1^{-2}+d_1\beta_1^{-1}+1)}{\beta_1^{-1}(1-\alpha_1\beta_1)(1-\alpha_2\beta_1)(1-\beta_2\beta_1^{-1})}$
	$z=\beta_2^{-1}$	$\frac{(1+b_1\beta_2+\beta_2^2)(\beta_2^{-2}+d_1\beta_2^{-1}+1)}{\beta_2^{-1}(1-\alpha_1\beta_2)(1-\alpha_2\beta_2)(1-\beta_1\beta_2^{-1})}$
$\frac{G(z)G(z^{-1})}{z}$	$z=0$	$\beta_1^{-1}\beta_2^{-1}$
	$z=\beta_1$	$\frac{(1+d_1\beta_1^{-1}+\beta_1^{-2})(1+d_1\beta_1+\beta_1^2)}{\beta_1(1-\beta_2\beta_1^{-1})(1-\beta_1^2)(1-\beta_1\beta_2)}$
	$z=\beta_2$	$\frac{(1+d_1\beta_2^{-1}+\beta_2^{-2})(1+d_1\beta_2+\beta_2^2)}{\beta_2(1-\beta_1\beta_2^{-1})(1-\beta_2^2)(1-\beta_1\beta_2)}$
	$z=\beta_1^{-1}$	$\frac{(1+d_1\beta_1^{-1}+\beta_1^{-2})(1+d_1\beta_1+\beta_1^2)}{\beta_1^{-1}(1-\beta_1^2)(1-\beta_2\beta_1^{-1})(1-\beta_2\beta_1^{-1})}$
	$z=\beta_2^{-1}$	$\frac{(1+d_1\beta_2^{-1}+\beta_2^{-2})(1+d_1\beta_2+\beta_2^2)}{\beta_2^{-1}(1-\beta_2^2)(1-\beta_1\beta_2^{-1})(1-\beta_1\beta_2^{-1})}$

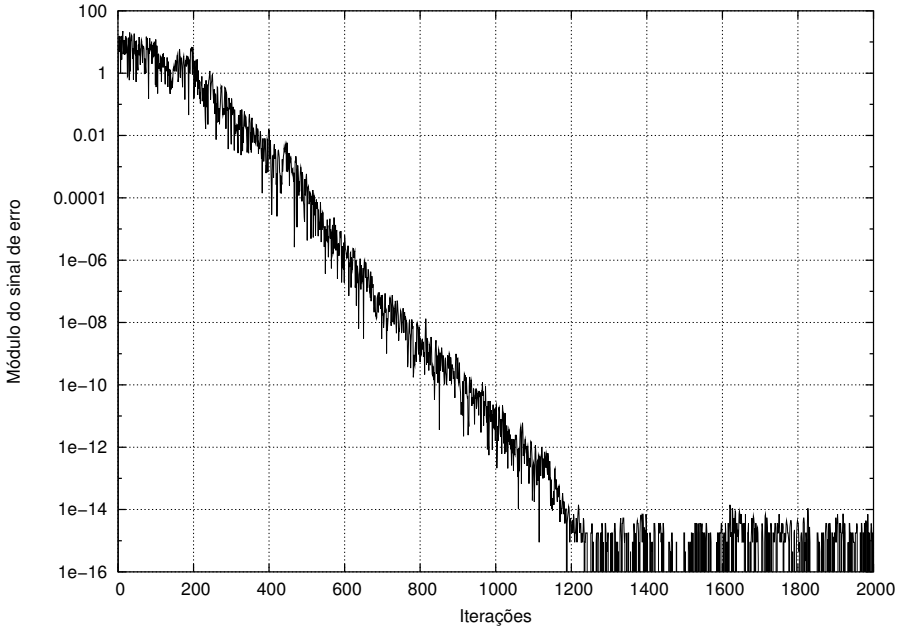


(a) sinal de erro (módulo)

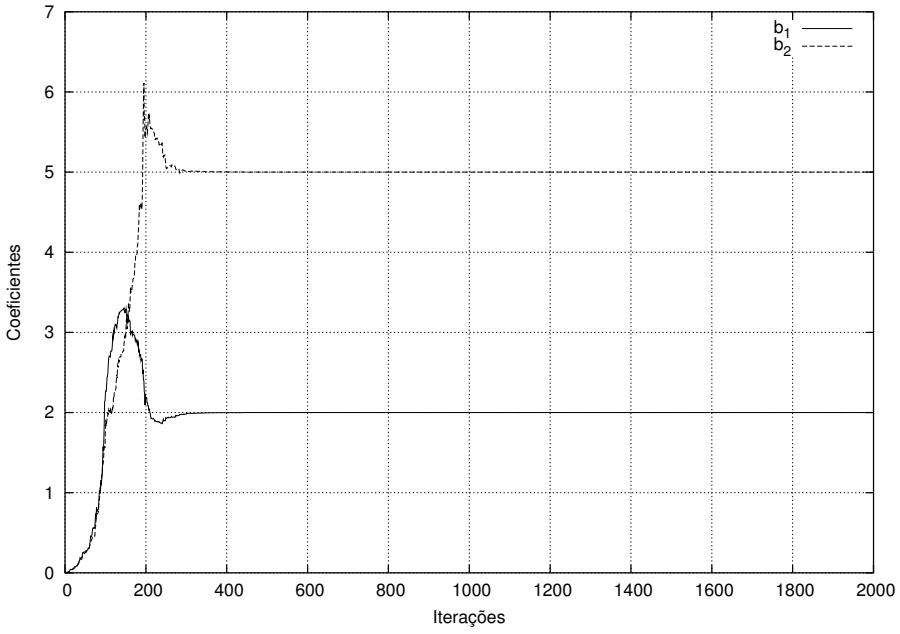


(b) coeficientes

Figura 3.6: Duas seções de segunda ordem (somente zeros) – LMS



(a) sinal de erro



(b) coeficientes

Figura 3.7: Duas seções de segunda ordem (somente zeros) – RLS

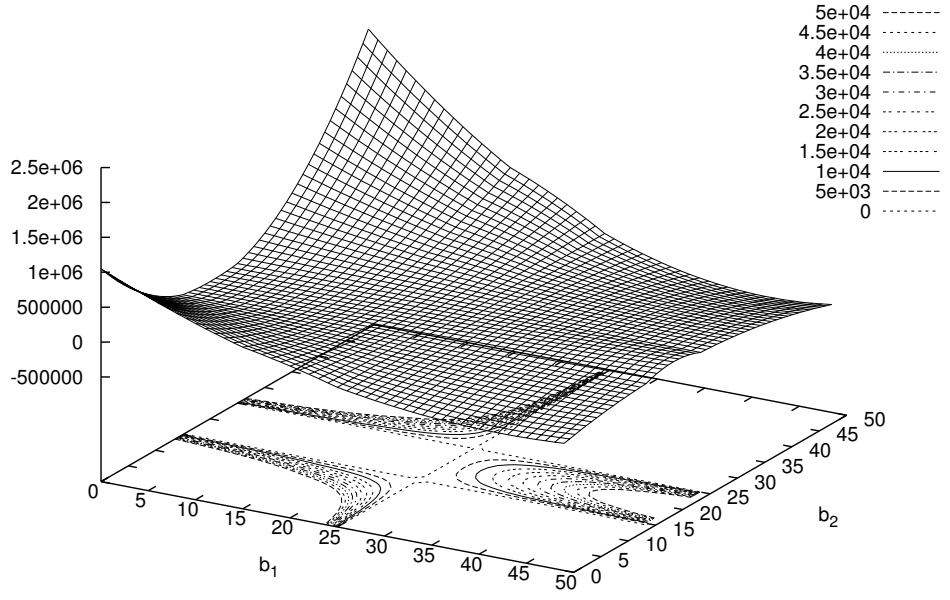


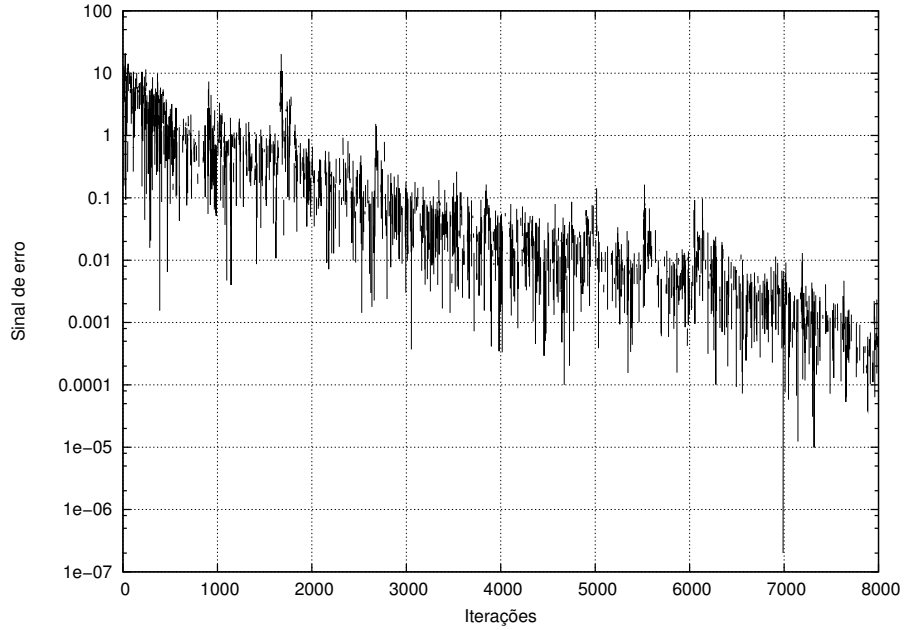
Figura 3.8: Superfície de desempenho para duas seções de segunda ordem

3.2.2 Adaptação de pólos e zeros

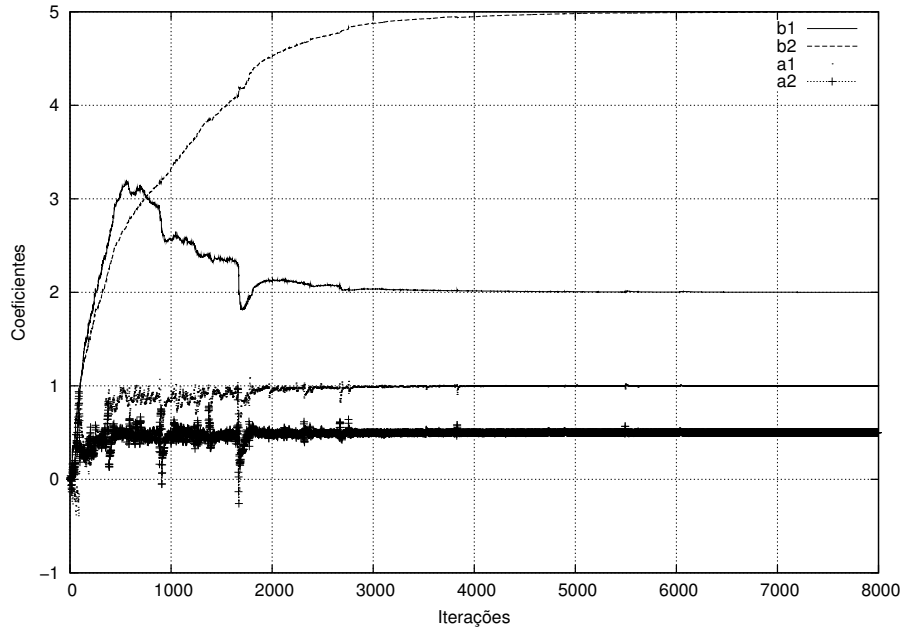
As figuras 3.9 e 3.10 apresentam a evolução dos sinais de erro e dos valores dos coeficientes adaptados através dos algoritmos LMS e RLS, respectivamente. Verifica-se que, em relação à simulação anterior, a velocidade de adaptação torna-se mais lenta. Na verdade, isto é conseqüência da superfície de desempenho não-quadrática (figura 3.8) que obrigou uma variação nos valores dos parâmetro de adaptação μ (LMS) e λ (RLS). Estes valores tiveram que ser modificados para estabilizar o processo adaptativo.

Como conseqüência direta da alteração no valor de μ, o sinal de erro ao fim de 8000 iterações é muitas vezes superior ao obtido na simulação anterior para o algoritmo LMS. O algoritmo RLS também foi afetado, embora menos significativamente: o valor do sinal de erro após 1000 iterações ficou abaixo de 10^{-6}. A Seção A.2 apresenta os códigos utilizados para a geração dos gráficos.

O cálculo da superfície de desempenho para o caso de duas seções de 2ª ordem pode ser derivado do resultado obtido na tabela 3.1, multiplicando cada um dos resíduos pelo polinômio $(1 + d_2 z^{-1} + z^{-2})$ e atribuindo a z o valor dos respectivos pólos.

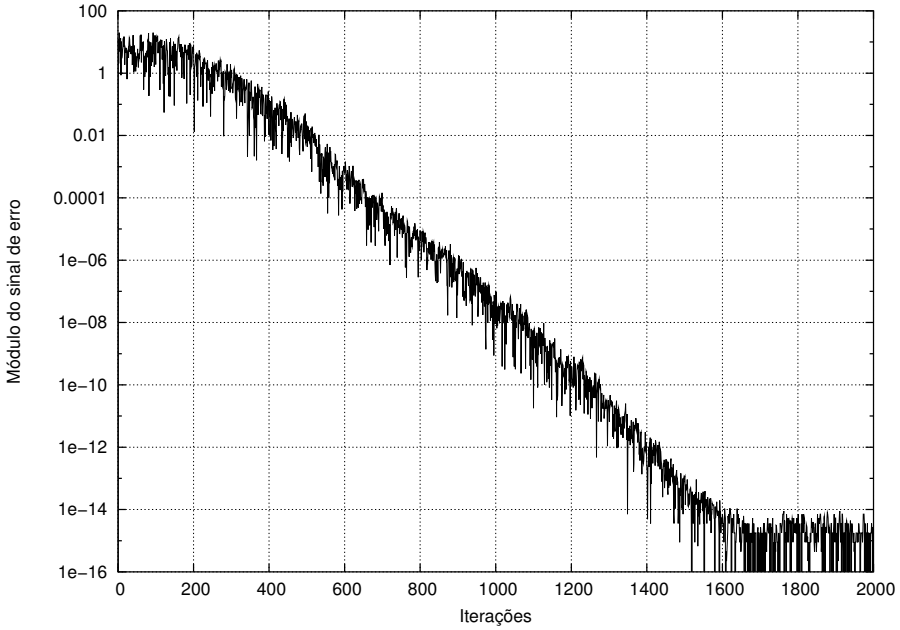


(a) sinal de erro (módulo)

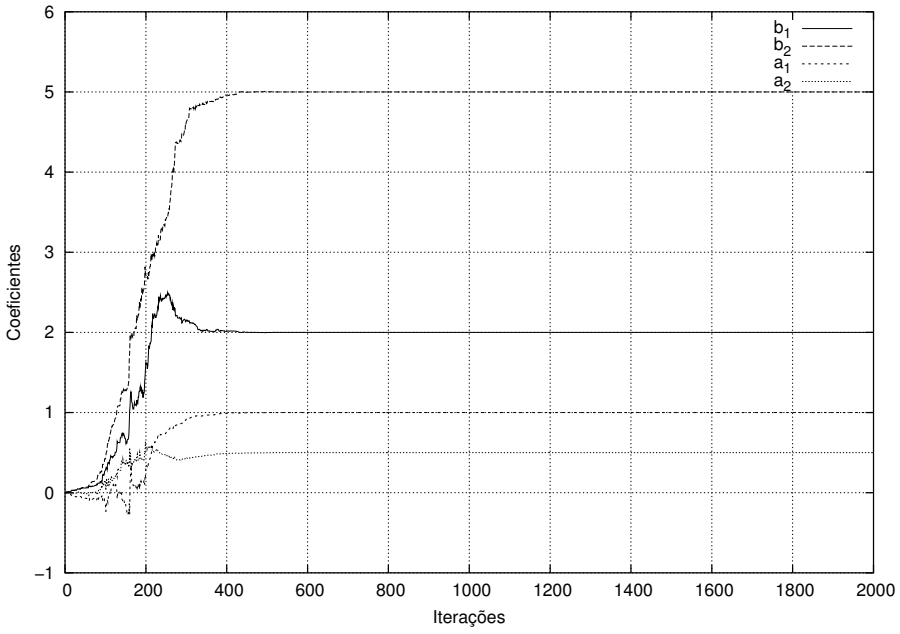


(b) coeficientes

Figura 3.9: Duas seções de segunda ordem – LMS



(a) sinal de erro



(b) coeficientes

Figura 3.10: Duas seções de segunda ordem – RLS

3.3 Três seções de 2ª ordem

Dando continuidade às simulações, nesta seção será testada a estrutura com três seções de segunda ordem, na forma:

$$H(z) = \frac{(1+b_1z^{-1}+z^{-2})(1+b_2z^{-1}+z^{-2})(1+b_3z^{-1}+z^{-2})}{(1+a_1z^{-1}+a_2z^{-2})} \tag{3.8}$$

Do mesmo modo, procede-se à adaptação dos zeros com pólos fixos e, posteriormente, à adaptação de pólos e zeros.

Neste caso as simulações não convergiram para o resultado esperado. O sinal de erro tornou-se instável, assim como os valores dos coeficientes sendo adaptados. A forma da superfície de desempenho, com a adição de mais uma seção FIR de segunda ordem, tornou-se muito complexa para o procedimento de adaptação, impossibilitando a correta identificação dos coeficientes do filtro.

Capítulo 4

Desenvolvimento

Neste capítulo será descrito o filtro para o qual foi desenvolvido o algoritmo apresentado neste trabalho [8], considerando sua função de transferência, topologia e demais características que impossibilitaram a adoção de algoritmos bem conhecidos (LMS, RLS, etc.) para a adaptação de seus coeficientes.

Em seguida, serão citados os requisitos necessários para o funcionamento do algoritmo, levando-se em consideração as restrições decorrentes da estrutura do filtro.

Finalmente, será apresentado o algoritmo desenvolvido para o filtro em questão, detalhando-se seu funcionamento e teoria.

4.1 Estrutura do filtro

O filtro para o qual o algoritmo foi desenvolvido possui características particulares, que impedem a adoção de algoritmos bem conhecidos. Sua estrutura consiste de 4 (quatro) seções FIR de 2ª ordem (figura 4.2), seguidas por 1 (uma) seção IIR de 2ª ordem, como mostrado na figura 4.1, sendo implementada por circuito a capacitores chaveados cuja configuração básica é apresentada na figura 4.2. Cada seção FIR possui função de transferência [8]:

$$H(z) = -\frac{C_0}{C_1}\left(1 + \frac{C_4 - C_5}{C_3}z^{-1} + z^{-2}\right) \tag{4.1}$$

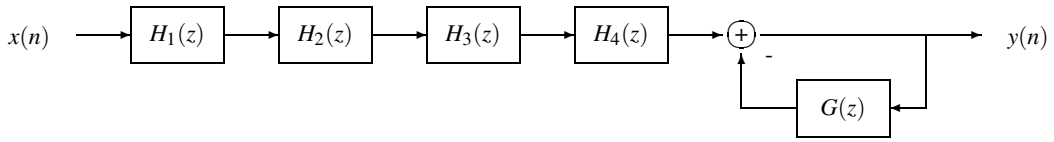


Figura 4.1: Estrutura global do filtro a capacitores chaveados

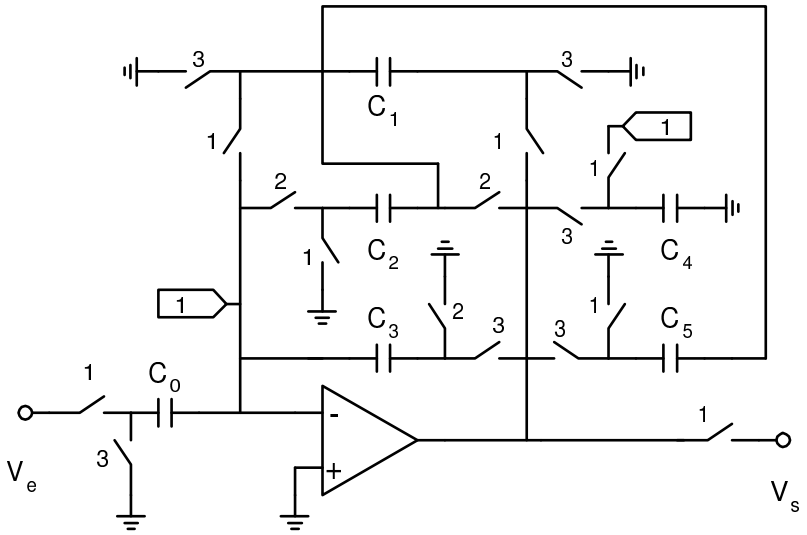


Figura 4.2: Seção FIR de segunda ordem no filtro a capacitores chaveados

Fazendo $C_0 = C_1$, cada uma das seções FIR assume, portanto, a forma:

$$B_i(z) = 1 + b_i z^{-1} + z^{-2} \tag{4.2}$$

Os valores do primeiro e terceiro coeficientes, termos z^0 e z^{-2}, respectivamente, são garantidamente sempre iguais a 1 (um), pois a estrutura do filtro assim os define. O valor do coeficiente b_i pode variar dentro do intervalo $[-2,2]$, fazendo com que todos os zeros da função de transferência residam sobre o círculo unitário.

A seção recursiva do filtro possui a forma abaixo:

$$\frac{1}{A(z)} = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.3}$$

O primeiro coeficiente do denominador (termo z^0) tem seu valor determinado pela estrutura do filtro que, no caso, será sempre igual a 1 (um). O coeficiente a_1 pode variar dentro do intervalo $[-2,2]$ e a_2 dentro de $[0,1]$, garantidas as condições de estabilidade.

Assim sendo, a função de transferência apresentada pelo filtro em questão possui a forma:

$$H(z) = \frac{\prod_{i=1}^{4} B_i(z)}{A(z)} = \frac{\prod_{i=1}^{4}(1 + b_i z^{-1} + z^{-2})}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.4}$$

4.1.1 Equivalência das formas direta e cascata

Como visto na Seção 2.5, algumas precauções devem ser tomadas quando se tenta realizar a função de transferência $H_c(z)$, na forma cascata, como $H_d(z)$, na forma direta:

$$H_c(z) = \frac{\prod_{i=1}^{4}(1 + b_i z^{-1} + z^{-2})}{(1 + a_1 z^{-1} + a_2 z^{-2})} \tag{4.5}$$

$$H_d(z) = \frac{\sum_{i=0}^{8} c_i z^{-i}}{(1 + a_1 z^{-1} + a_2 z^{-2})} \tag{4.6}$$

Realizando o mapeamento entre os coeficientes de $H_c(z)$ e $H_d(z)$, tem-se:

$$c_0 = c_8 = 1 \tag{4.7}$$

$$c_1 = c_7 = b_1 + b_2 + b_3 + b_4 \tag{4.8}$$

$$c_2 = c_6 = b_1 b_2 + b_3 b_4 + (b_1 + b_2)(b_3 + b_4) + 4 \tag{4.9}$$

$$c_3 = c_5 = (b_1 + b_2)(b_3 b_4 + 3) + (b_3 + b_4)(b_1 b_2 + 3) \tag{4.10}$$

$$c_4 = (b_1 + b_2)(b_3 + b_4) + (b_1 b_2 + 2)(b_3 b_4 + 2) + 2 \tag{4.11}$$

A estrutura das seções de segunda ordem na forma cascata faz com que $c_0 = c_8 = 1$ e que $c_1 = c_7$, $c_2 = c_6$ e $c_3 = c_5$. Portanto, para o mapeamento entre as realizações $H_c(z)$ e $H_d(z)$ é suficiente considerar apenas os coeficientes c_1, c_2, c_3 e c_4. Logo:

$$\mathbf{f}(\mathbf{w}_1) = \begin{bmatrix} c_1 & c_2 & c_3 & c_4 \end{bmatrix}^T \tag{4.12}$$

Para que o mapeamento ocorra corretamente e para que não apareçam pontos de sela, é necessário o cálculo do determinante da inversa da matriz $\frac{\partial \mathbf{f}(\mathbf{w}_1)}{\partial \mathbf{w}_1}$:

$$\frac{\partial \mathbf{f}(\mathbf{w}_1)}{\partial \mathbf{w}_1} = \begin{bmatrix} \frac{\partial c_1}{\partial b_1} & \cdots & \frac{\partial c_1}{\partial b_4} \\ \vdots & \ddots & \vdots \\ \frac{\partial c_4}{\partial b_1} & \cdots & \frac{\partial c_4}{\partial b_4} \end{bmatrix} \tag{4.13}$$

onde:

$$\frac{\partial c_1}{\partial b_1} = \frac{\partial c_1}{\partial b_2} = \frac{\partial c_1}{\partial b_3} = \frac{\partial c_1}{\partial b_4} = 1 \tag{4.14}$$

$$\frac{\partial c_2}{\partial b_1} = b_2 + b_3 + b_4 \tag{4.15}$$

$$\frac{\partial c_2}{\partial b_2} = b_1 + b_3 + b_4 \tag{4.16}$$

$$\frac{\partial c_2}{\partial b_3} = b_1 + b_2 + b_4 \tag{4.17}$$

$$\frac{\partial c_2}{\partial b_4} = b_1 + b_2 + b_3 \tag{4.18}$$

$$\frac{\partial c_3}{\partial b_1} = b_2 b_3 + b_2 b_4 + b_3 b_4 + 3 \tag{4.19}$$

$$\frac{\partial c_3}{\partial b_2} = b_1 b_3 + b_4 b_1 + b_3 b_4 + 3 \tag{4.20}$$

$$\frac{\partial c_3}{\partial b_3} = b_1 b_2 + b_1 b_4 + b_2 b_4 + 3 \tag{4.21}$$

$$\frac{\partial c_3}{\partial b_4} = b_1 b_2 + b_1 b_3 + b_2 b_3 + 3 \tag{4.22}$$

$$\frac{\partial c_4}{\partial b_1} = b_2 b_3 b_4 + 2b_2 + b_3 + b_4 \tag{4.23}$$

$$\frac{\partial c_4}{\partial b_2} = b_1 b_3 b_4 + 2b_1 + b_3 + b_4 \tag{4.24}$$

$$\frac{\partial c_4}{\partial b_3} = b_1 b_2 b_4 + 2b_4 + b_1 + b_2 \tag{4.25}$$

$$\frac{\partial c_4}{\partial b_4} = b_1 b_2 b_3 + 2b_3 + b_1 + b_2 \tag{4.26}$$

O determinante da matriz $\frac{\partial \mathbf{f}(\mathbf{w}_1)}{\partial \mathbf{w}_1}^{-1}$ é dado por:

$$\begin{aligned}
\det\left[\frac{\partial \mathbf{f}(\mathbf{w}_1)}{\partial \mathbf{w}_1}^{-1}\right] &= b_1 b_2 (b_1 b_2 + 1)(b_1 - b_2)(b_3 - b_4) \\
&+ b_1^2 b_3^2 (b_1 - b_3)(b_4 - b_2) + b_2^2 b_3^2 (b_2 - b_3)(b_1 - b_4) \\
&+ b_3 b_4 (b_3 b_4 + 1)(b_1 - b_2)(b_3 - b_4) \\
&+ b_1^2 b_4^2 (b_1 - b_4)(b_2 - b_3) + b_2^2 b_4^2 (b_2 - b_4)(b_3 - b_1) \\
&+ (b_1^3 - b_2^3)(b_3 - b_4) + (b_3^3 - b_4^3)(b_1 - b_2) \\
&+ 2(b_1^2 - b_2^2)(b_3^2 - b_4^2)
\end{aligned} \tag{4.27}$$

A matriz $\frac{\partial \mathbf{f}(\mathbf{w}_1)}{\partial \mathbf{w}_1}^{-1}$ será singular quando todos os termos da equação 4.27 forem nulos. Isto somente será possível quando $b_1 = b_2 = b_3 = b_4$. Desta forma, ao se realizar a adaptação de $H_c(z)$ deve-se evitar que os coeficientes b_i possuam valores iguais.

4.2 Requisitos do algoritmo

O filtro digital descrito acima foi implementado em um circuito integrado de 68 pinos. Para operá-lo utiliza-se uma placa de testes especialmente projetada para tal, responsável pela alimentação do circuito integrado e pela geração de sinais de controle e programação dos coeficientes do filtro.

Dada a configuração dos pinos do circuito integrado e da placa de testes, não é possível a obtenção, no mesmo instante n de tempo, de sinais na saída do filtro e de sinais em qualquer estágio intermediário do mesmo.

Esta restrição impede a adoção de algoritmos adaptativos usuais, tais como LMS e RLS, uma vez que estes algoritmos utilizam os sinais de mais de um nó da malha do filtro no mesmo instante de tempo n. Desta forma, a amostragem dos sinais pertinentes ao algoritmo deve ser realizada de maneira mais lenta, de modo a possibilitar a consulta a todos os sinais necessários à adaptação.

Os requisitos básicos de operação do algoritmo adaptativo, portanto, são:

- poder operar em uma taxa inferior à freqüência normal de operação do filtro;
- ser capaz de estimar os sinais intermediários (entre cada seção FIR e IIR) para utilização no processo de adaptação.

4.3 Teoria do algoritmo

O algoritmo desenvolvido que atende aos requisitos acima baseia-se nos métodos LMS-CT (*Least Mean Square-Coordinate Transform*) e LMS-ICT (*Least Mean Square-Inverse Coordinate Transform*) [1], os quais serão descritos a seguir.

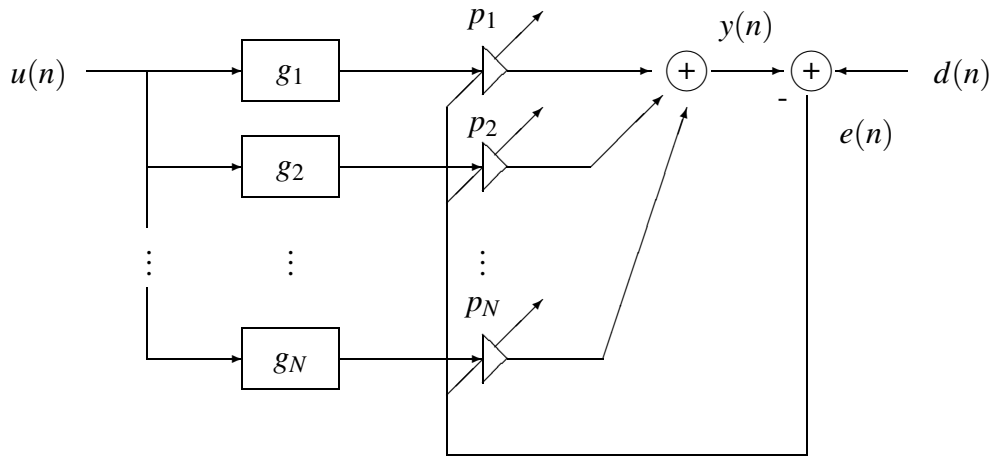


Figura 4.3: Estrutura ALC (*Adaptive Linear Combiner*)

4.3.1 LMS–CT

Um filtro FIR linear pode ter sua estrutura representada por uma topologia ALC (*Adaptive Linear Combiner*), exibida na figura 4.3. Esta estrutura possui uma série de filtros digitais FIR $\mathbf{g}_i$ que, por sua vez, geram os sinais $\mathbf{x}(n) = \begin{bmatrix} x_0(n) & \cdots & x_{N-1}(n) \end{bmatrix}^T$, que representam o estado do filtro. A geração dos sinais $x_i(n)$ é dada por:

$$x_i(n) = \mathbf{g}_i^T(n)\mathbf{u}(n) \tag{4.28}$$

onde $\mathbf{u}(n)$ é o vetor do sinal de entrada do filtro.

Os filtros digitais $\mathbf{g}_i$ são fixos, e refletem a estrutura fundamental do filtro original. Deste modo torna-se possível a adaptação do filtro em questão através de N parâmetros, chamados $p_i(n)$. Definindo-se o vetor $\mathbf{p}(n) = \begin{bmatrix} p_0(n) & \cdots & p_{N-1}(n) \end{bmatrix}^T$, sua equação de adaptação, de acordo com o algoritmo LMS, é dada por:

$$\mathbf{p}(n+1) = \mathbf{p}(n) + 2\mu e(n)\mathbf{x}(n) \tag{4.29}$$

$$\mathbf{p}(n+1) = \mathbf{p}(n) + 2\mu e(n)\mathbf{G}^T\mathbf{u}(n) \tag{4.30}$$

onde $\mathbf{G}^T$ $(M \times N)$ é:

$$\mathbf{G} = \begin{bmatrix} \hat{g}_1(0) & \cdots & \hat{g}_N(0) \\ \vdots & \ddots & \vdots \\ \hat{g}_1(M-1) & \cdots & \hat{g}_N(M-1) \end{bmatrix} \tag{4.31}$$

4.3.2 LMS-ICT

O método LMS-CT, descrito na Seção 4.3.1, permite que o processo de adaptação ocorra sem que seja necessário o conhecimento de sinais internos do filtro, uma vez que o conjunto de sinais $\mathbf{x}(n)$ pode ser gerado externamente. Deste modo, o requisito de funcionamento sem conhecimento (ou acesso) aos sinais internos do filtro é atingido.

Por outro lado, ainda não é possível utilizar o algoritmo LMS-CT devido à segunda restrição: somente um dos sinais está disponível a cada passo de iteração do algoritmo. O objetivo do algoritmo LMS-ICT (*Least Mean Square-Inverse Coordinate Transform*) é o de possibilitar a subamostragem dos sinais de entrada, $u(n)$, de saída, $y(n)$, e consequentemente, o de erro, $e(n)$. Deste modo, a cada passo de iteração do algoritmo será possível selecionar um sinal diferente, respeitando a segunda restrição e permitindo que o processo de adaptação ocorra.

Para ilustrar o funcionamento do algoritmo LMS–ICT, considere um filtro com coeficientes $\mathbf{q}^T(n) = \begin{bmatrix} q_0(n) & \cdots & q_{M-1}(n) \end{bmatrix}$. O procedimento de atualização dos coeficientes para este filtro seria, considerando o algoritmo LMS:

$$\mathbf{q}(n+1) = \mathbf{q}(n) + 2\mu e(n)\mathbf{u}(n) \tag{4.32}$$

onde $\mathbf{u}^T(n) = \begin{bmatrix} u(n) & \cdots & u(n-M+1) \end{bmatrix}$.

Se subamostrarmos os sinais de entrada ($u(n)$) e de erro ($e(n)$) por fatores V e W, a equação de atualização dos parâmetros será dada por:

$$\begin{aligned} \mathbf{q}(VW(n+1)) &= \mathbf{q}(VWn) \\ &\quad + 2\mu \begin{bmatrix} e(VWn+1)u(VWn) \\ e(VWn+V+1)u(VWn+W) \\ \vdots \\ e(VWn+(M-1)V+1)u(VWn+(M-1)W) \end{bmatrix} \\ &\equiv \mathbf{q}(VWn) - \mu\hat{\nabla}_{\varepsilon}\mathbf{q}(VWn) \end{aligned} \tag{4.33}$$

onde

$$\hat{\nabla}_{\varepsilon}\mathbf{q}(VWn) = -2\begin{bmatrix} e(VWn+1)u(VWn) \\ e(VWn+V+1)u(VWn+W) \\ \vdots \\ e(VWn+(M-1)V+1)u(VWn+(M-1)W) \end{bmatrix} \tag{4.34}$$

Agora deve haver um procedimento para mapear o vetor gradiente $\hat{\nabla}_{\varepsilon}\mathbf{q}(n) \in \mathbb{R}^M$ no espaço $\mathbb{R}^N$, de modo a possibilitar a atualização do vetor de coeficientes $\mathbf{p}(n)$. Em [1], o mapeamento é realizado forçando-se a seguinte condição:

$$\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{q} = \mathbf{q} \tag{4.35}$$

$$(\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T - \mathbf{I})\mathbf{q} = \mathbf{0} \tag{4.36}$$

Na regra de atualização dos coeficientes, a condição acima é forçada alterando-se a estimativa do gradiente $\hat{\nabla}_{\varepsilon}\mathbf{q}(n)$ por $\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\hat{\nabla}_{\varepsilon}\mathbf{q}(n)$ [1]:

$$\mathbf{q}(n+1) = \mathbf{q}(n) - \mu\mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\hat{\nabla}_{\varepsilon}\mathbf{q}(n) \tag{4.37}$$

Sabendo que $\mathbf{q} = \mathbf{G}\mathbf{p}$, obtém-se a equação de atualização dos coeficientes $\mathbf{p}(n)$:

$$\mathbf{p}(n+1) = \mathbf{p}(n) - \mu(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\hat{\nabla}_{\varepsilon}\mathbf{q}(n) \tag{4.38}$$

$$= \mathbf{p}(n) - \mu\mathbf{K}\hat{\nabla}_{\varepsilon}\mathbf{q}(n) \tag{4.39}$$

onde $\mathbf{K} = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T$.

Fazendo $\hat{\nabla}_{\varepsilon}\mathbf{q}(n) = -2e(n)\mathbf{u}(n)$:

$$\mathbf{p}(n+1) = \mathbf{p}(n) + 2\mu e(n)\mathbf{K}\mathbf{u}(n) \tag{4.40}$$

4.4 Funcionamento do algoritmo

4.4.1 Adaptação com pólos fixos

Com os pólos fixos, as variáveis de adaptação ficam sendo b_1, b_2, b_3 e b_4. Os coeficientes que regulam a localização dos pólos, a_1 e a_2 assumem-se fixos, determinados

por algum método de estimação *offline*. A função de transferência do filtro adaptativo é, portanto:

$$H(z) = \frac{\prod_{i=1}^{4}(1 + b_i z^{-1} + z^{-2})}{(1 + a_1 z^{-1} + a_2 z^{-2})} \tag{4.41}$$

Neste caso, pelo fato dos pólos serem fixos, não há a necessidade do emprego do teste de estabilidade (equações 2.44 e 2.45). A adaptação ocorre sem se verificar se os pólos estão ou não no interior do círculo unitário do plano z, pois assume-se que isto tenha sido feito inicialmente.

A análise realizada no Capítulo 3 demonstrou ser complexa a adoção do algoritmo na forma cascata. Embora este fosse o objetivo inicial, esta abordagem foi descartada quando verificou-se a instabilidade das adaptações efetuadas em filtros IIR com mais de duas seções de segunda ordem em série.

Para realizar a adaptação do filtro em questão, optou-se, portanto, pela forma direta. Neste caso, a função de transferência realizada torna-se:

$$H(z) = \frac{\sum_{i=0}^{8} c_i z^{-i}}{(1 + a_1 z^{-1} + a_2 z^{-2})} \tag{4.42}$$

onde c_i corresponde ao coeficiente de ordem i da função de transferência citada na equação 4.41.

A opção pela forma direta torna necessário um método para o cálculo dos valores individuais dos coeficientes b_i. Um método [9] para encontrar as raízes do polinômio do numerador da equação 4.42 foi utilizado para a atualização dos coeficientes do filtro sendo adaptado.

Sendo necessária a subamostragem dos sinais de entrada e de saída, define-se inicialmente os fatores V e W com os quais os respectivos sinais serão subamostrados pelo algoritmo adaptativo. A taxa de atualização dos coeficientes adaptados ocorrerá a cada $Q = V \times W$ iterações. O algoritmo funcionará, portanto, Q vezes mais lentamente.

O cálculo do vetor gradiente, relativo aos coeficientes b_i, será efetuado empregando-se a matriz $\mathbf{K}$ citada na Seção 4.3.2, sendo a equação de atualização dos coeficientes dada por:

$$\mathbf{p}(n+1) = \mathbf{p}(n) + 2\mu\mathbf{K}e(n)\mathbf{x}(n) \tag{4.43}$$

O procedimento de fatoração do numerador da função de transferência ocorrerá a cada Q iterações, quando o vetor de coeficientes for atualizado.

Simulações

A figura 4.4 exibe o resultado de uma simulação com o algoritmo, mostrando a evolução do sinal de erro e dos valores dos coeficientes b_i, $1 \leq i \leq 4$.

Observa-se a correta convergência dos coeficientes b_1, b_2, b_3 e b_4, assim como a contínua diminuição do sinal de erro. No caso em questão, o fator de decimação para o sinal de entrada, $x(n)$, é 3. O sinal de saída, $y(n)$, é decimado por um fator de 2.

4.4.2 Adaptação de pólos e zeros

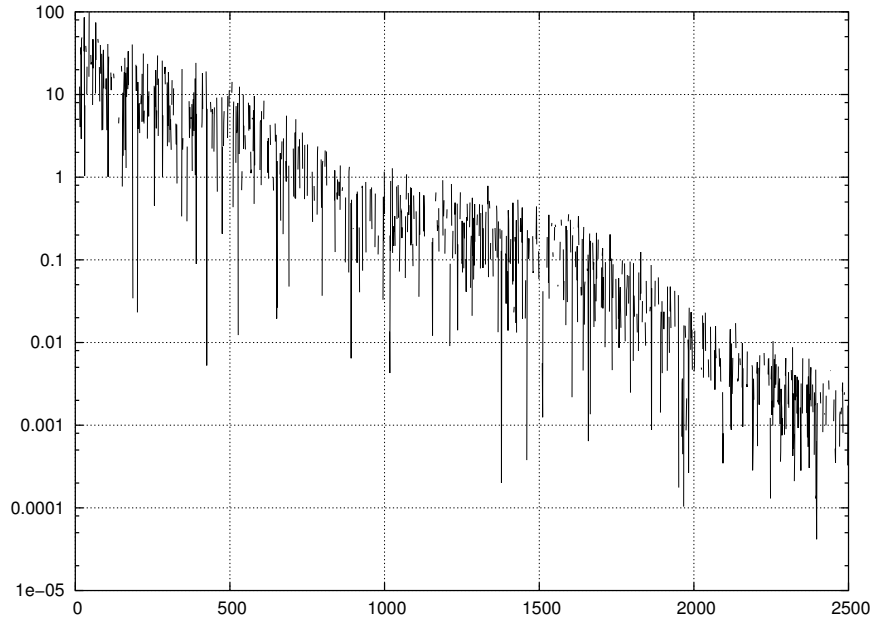
O algoritmo apresentado na seção anterior deve sofrer alterações para que os pólos do filtro também sejam adaptados. A elaboração do algoritmo LMS-ICT levou em consideração somente uma estrutura FIR, o que dificulta sua modificação para o caso mais geral, em que o filtro possui uma função de transferência IIR. Porém, o cálculo dos sinais gradientes relativos aos coeficientes a_1 e a_2, responsáveis pela seção recursiva do filtro, poderia ser efetuado empregando-se o mesmo método apresentado na Seção 2.3.1.

Desta forma, ao vetor gradiente obtido através do método LMS-ICT seriam adicionados os valores relativos aos coeficientes da seção recursiva, a_1 e a_2, obtendo-se assim a adaptação dos pólos e zeros do filtro em questão.

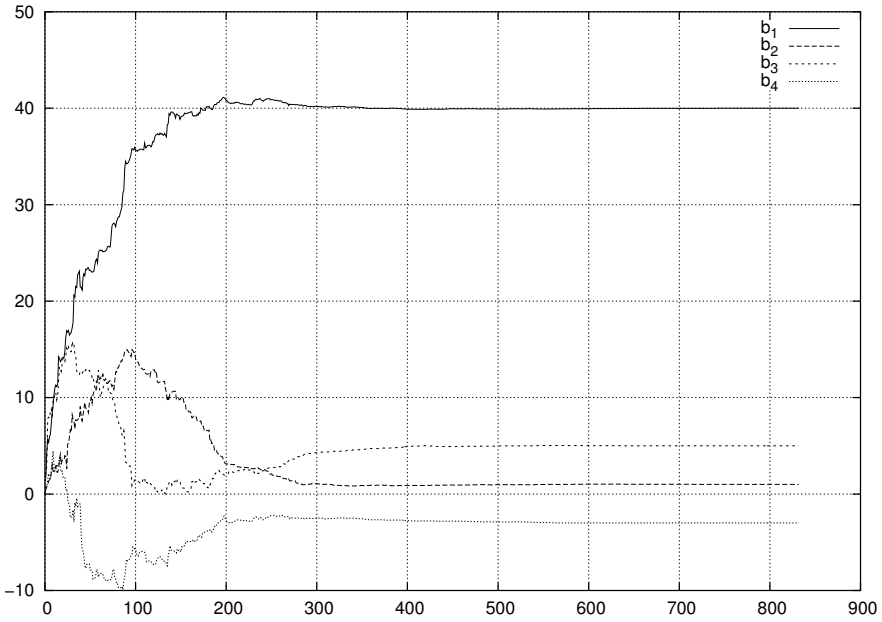
No entanto, testes realizados empregando-se o método descrito acima não produziram resultados satisfatórios. A adaptação dos pólos através do algoritmo LMS-ICT não foi possível.

4.4.3 Fatoração do polinômio do numerador

O algoritmo adaptativo foi realizado na forma direta, apesar do filtro estar implementado na forma cascata. Neste sentido, durante o processo de adaptação é necessária a fatoração do polinômio do numerador da função de transferência, $N_d(z)$, para a obtenção



(a) iterações × módulo do sinal de erro



(b) iterações × coeficientes

Figura 4.4: Adaptação com taxa reduzida: somente zeros

do numerador na forma cascata, $N_c(z)$:

$$N_d(z) = \sum_{i=0}^{8} c_i z^{-i} = \prod_{i=1}^{4}(1 + b_i z^{-1} + z^{-2}) = N_c(z) \tag{4.44}$$

O método utilizado para a obtenção das raízes do polinômio $N_d(z)$ consiste no cálculo dos autovalores da matriz $\mathbf{A}$ (8×8) abaixo [9]:

$$\mathbf{A} = \begin{bmatrix} -c_1 & -c_2 & -c_3 & \cdots & -c_7 & -1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{4.45}$$

Os autovalores de $\mathbf{A}$ serão as raízes não nulas do polinômio $N(z)$. Como citado na Seção 4.1:

$$|b_i| \leq 2, \text{ para } 1 \leq i \leq 4 \tag{4.46}$$

Portanto, o polinômio $N_d(z)$, cujos coeficientes são dados nas equações 4.7 a 4.11, possuirá raizes complexas conjugadas. Os coeficientes na forma cascata, representada por $N_c(z)$, serão obtidos efetuando-se a multiplicação de cada par complexo conjugado, gerando-se assim a forma de cada seção de segunda ordem, $1 + b_i z^{-1} + z^{-2}$.

Entende-se que o procedimento necessário para o cálculo de autovalores seja complexo. Desta forma, outros métodos computacionalmente mais simples para obtenção de zeros de polinômios podem ser empregados [10].

A forma direta adiciona um grau de liberdade à adaptação: ela pode convergir para uma solução em que as restrições da equação 4.46 não são satisfeitas. Neste caso o algoritmo terá convergido para uma solução inválida, devendo esta ser descartada.

Capítulo 5

Conclusão

O funcionamento adequado de um algoritmo adaptativo depende da estrutura sendo utilizada pelo filtro adaptado. O surgimento de irregularidades e, por conseqüência, de mínimos locais na superfície de desempenho pode ser ocasionado pela forma como a função de transferência é implementada, seja na forma direta ou cascata.

Uma análise da forma das superfícies de desempenho para a forma direta e cascata do filtro analógico em questão foram conduzidas com o objetivo de identificar potenciais problemas durante o processo de adaptação. Foi visto que, na forma cascata, o número crescente de seções de segunda ordem inviabiliza a adaptação, por tornar a superfície de desempenho extremamente irregular e suscetível a erros de convergência. Por outro lado, a adaptação na forma direta (com pólos fixos ou adaptados) não demonstrou os mesmos problemas verificados na estrutura cascata.

Outro ponto verificado foi que, devido à forma com a qual os coeficientes do filtro adaptativo são implementados nas duas estruturas, a superfície de desempenho deste é influenciada. A falha na convergência da simulação em forma cascata deveu-se ao desdobramento ocasionado pelo efeito gerado com a colocação em série de diversas seções FIR de segunda ordem. Neste sentido, os zeros das N funções de transferência de cada seção FIR procuraram, cada um, N possíveis soluções. Isto ocasionava uma busca por um universo total de soluções igual a $N!$. Considerando 4 seções de segunda ordem (estrutura do filtro analógico sendo adaptado), com a estrutura cascata o algoritmo adaptativo

poderia, teoricamente, encontrar $4! = 24$ soluções distintas. Nas simulações efetuadas, o algoritmo convergiu com sucesso apenas com 2 seções de segunda ordem ($N! = 2$). A partir de $N = 3$ ($N! = 6$) a convergência não foi possível.

Baseado nos resultados obtidos nas simulações citadas acima, o algoritmo adaptativo foi desenvolvido. Procurando diminuir a taxa de amostragem requerida, dedicou-se um cuidado maior na elaboração de um algoritmo com taxa de adaptação reduzida, em relação à taxa real de amostragem. De fato, a taxa de adaptação do algoritmo pode estar abaixo da freqüência de Nyquist necessária à amostragem do sinal de entrada. Isto é possível porque o algoritmo não reconstrói o sinal, e sim adapta o filtro com amostras dos sinais de entrada, saída e erro subamostrados.

Verificou-se que a adaptação com taxa reduzida na forma direta apresentou resultados satisfatórios somente com pólos fixos. Impossibilitado de utilizar a forma cascata para o filtro em questão, o algoritmo não foi capaz de modificar diretamente os valores dos coeficientes adaptados. Desta forma, um procedimento de fatoração foi aplicado ao denominador da função de transferência do filtro adaptativo para a obtenção dos valores individuais de cada coeficiente adaptado para posterior cópia (programação) do filtro sendo adaptado.

5.1 Trabalhos futuros

Trabalhos futuros podem incluir a pesquisa por um método em que seja possível a adaptação dos pólos pelo algoritmo adaptativo. Também seria interessante a implementação do algoritmo em hardware, onde seriam aplicadas as restrições do filtro quanto aos valores dos coeficientes. Além disso, pode-se melhorar o algoritmo de fatoração utilizado durante o processo de adaptação para a obtenção dos valores dos coeficiente do numerador do filtro analógico. Outra possibilidade é o estudo do efeito causado pela quantização dos coeficientes durante a etapa de programação do filtro analógico.

Referências Bibliográficas

[1] CARUSONE, A. C., JOHNS, D. A., "Digital LMS Adaptation of Analog Filters Without Gradient Information", *IEEE Transactions on Circuits and Systems II: Express Briefs*, v. 50, n. 9, pp. 539–552, 2003.

[2] HAYKIN, S., *Adaptive Filter Theory*. Prentice Hall, 2001.

[3] DINIZ, P. S. R., *Adaptive Filtering: Algorithms and Practical Implementation*, Kluwer Academic Publishers, pp. 361–422, 1997.

[4] MITRA, S. K., *Digital Signal Processing: A Computer Based Approach*, McGraw-Hill, pp. 252–257, 1998.

[5] NAYERI, M., JENKINS, W. K., "Alternate Realizations to Adaptive IIR Filters and Properties of Their Performance Surfaces", *IEEE Transactions on Circuits and Systems*, v. 36, n. 4, pp. 485–496, 1989.

[6] STEARNS, S. D., "Error Surfaces of Recursive Adaptive Filters", *IEEE Transactions on Acoustics, Speech and Signal Processing*, v. 29, n. 3, pp. 763–766, 1981.

[7] KNOPP, K., BAGEMIHL, F., *Theory of Functions Parts I and II, Two Volumes Bound as One, Part I*, Dover, pp. 129–134, 1996.

[8] MONTEIRO, J. B., *Filtro Recursivo a Capacitores Chaveados Digitalmente Programável por Controle de Carga*. Tese de D. Sc., COPPE/UFRJ, Março 2004.

[9] WEISSTEIN, E. W., "Polynomial", From MathWorld–A Wolfram Web Resource, http://mathworld.wolfram.com/Polynomial.html.

[10] RUGGIERO, M. A. G., LOPES, V. L. D. R., *Cálculo Numérico: Aspectos Teóricos e Computacionais*. McGraw-Hill, 1988.

[11] SHYNK, J. J., "Adaptive IIR Filtering", *IEEE ASP Magazine*, v. 6, n. 2, pp. 4–21, Abril 1989.

[12] WEI, Y., GELFAND, S. B., KROGMEIER, J. V., "Noise-Constrained Least Mean Squares Algorithm", *IEEE Transactions on Signal Processing*, v. 49, n. 9, pp. 1961–1970, 2001.

[13] WILLIAMSON, G. A., ASHLEY, J. P., NAYERI, M., "Structural Issues in Cascade-Form Adaptive IIR Filters". In: *International Conference on Acoustics, Speech and Signal Processing*, v. 2, pp. 1436–1439, 1995.

[14] USEVITCH, B. E., JENKINS, W. K., "A Cascade Implementation of a New IIR Adaptive Digital Filter with Global Convergence and Improved Convergence Rates". In: *IEEE International Symposium on Circuits and Systems*, v. 3, pp. 2140–2143, 1989.

[15] GAO, F. X. Y., SNELGROVE, W. M., "An Efficient Adaptive Cascade IIR Filter". In: *IEEE International Symposium on Circuits and Systems*, v. 1, pp. 444–447, 1991.

[16] NAYERI, M., JENKINS, W. K., "Analysis of Alternative Realizations of Adaptive IIR Filters". In: *IEEE International Symposium on Circuits and Systems*, v. 3, pp. 2157–2160, 1988.

[17] JENKINS, W. K., NAYERI, M., "Adaptive Filters Realized with Second Order Sections". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, v. 11, pp. 2103–2106, 1986.

[18] RAO, B. D., "Adaptive IIR Filtering Using Cascade Structures". In: *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, v. 1, pp. 194–198, 1993.

Apêndice A

Programas

Neste apêndice serão listados os programas utilizados em simulações nos programas MatLab e Octave[1].

A.1 Simulação na forma direta

O programa abaixo utiliza o algoritmo LMS para identificar uma função de transferência na forma

$$H(z) = \frac{1 + b_1 z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{A.1}$$

Os valores dos coeficientes de $H(z)$ estão representados nos vetores B e A do código (numerador e denominador, respectivamente). A variável n controla o número de iterações. Opcionalmente, pode-se controlar a duração da adaptação através de um limite mínimo para o sinal de erro, e. O parâmetro de adaptação μ é representado pela variável *mi*, cujo valor $(0,04)$ foi obtido através de sucessivas tentativas.

Ao final do laço *for* principal, o vetor *ve* conterá o valor do sinal de erro ao longo de toda a adaptação. A evolução dos valores dos coeficientes adaptados pode ser acompanhada através da variável *vw*. Os gráficos da figura 3.4 foram gerados através deste programa.

[1] `http://www.octave.org/`

Para realizar a adaptação somente dos zeros da função de transferência, é necessário somente igualar a zero o valor do sinal gradiente relativo aos coeficientes a_1 e a_2, e inicializá-los com seus valores corretos. Além disso, a verificação de pólos estáveis via critério de Jury torna-se dispensável. Os gráficos da figura 3.2 foram obtidos desta forma.

```
n = 1000;
B = [ 1 2 1 ]'; N = length(B);
A = [ 1 1 0.5 ]'; M = length(A);
x = normal_rnd(0, 1, n, 1);
d = filter(B, A, x);
mi=0.04;

w = zeros(N + M, 1);
b = [1 0 1]';
a = [1 0 0]';
Sb = zeros(N, 1);
Sa = zeros(M, 1);
vx = zeros(N, 1);
vy = zeros(M, 1);
vw = [];
ve = [];

for i = 1:n;
  vx = [ x(i); vx(1:N-1) ];
  Sb = [ x(i); Sb(1:N-1) ];
  y = b'*vx - a(2:M)'*vy(1:M-1);
  vy = [ y; vy(1:M-1) ];
  %Sa = [ -vy(1:M-1)'*a(1:M-1); Sa(1:M-1) ];
  Sa = [ y-a(2:M)'*Sa(1:M-1); Sa(1:M-1) ];

  e = d(i) - y;
  ve = [ ve e ];
  nw = w - mi * e * [ -Sb; Sa ];
  if (abs(nw(6)) < 1 && abs(nw(5)) < 1 + nw(6))
    w = [ 1 nw(2) 1 1 nw(5) nw(6) ]';
    vw = [ vw; w' ];
```

```
    b = w(1:N);
    a = w(N+1:M+N);
  end
end
```

Para realizar a adaptação utilizando o algoritmo RLS foi necessário modificar o algoritmo acima. A figura 3.5 foi obtida com o algoritmo abaixo, onde se ajustam os valores de pólos e zeros. A adaptação somente dos zeros requer pequenas alterações no algoritmo, anulando os valores das derivadas parciais $\frac{\partial e(n)}{\partial a_1(n)}$ e $\frac{\partial e(n)}{\partial a_2(n)}$ no vetor $\frac{\partial e(n)}{\partial \mathbf{w}(n)}$, representado no programa pela variável *phi*.

```
n = 1000;
B = [ 1 2 1 ]'; N = length(B);
A = [ 1 1 0.5 ]'; M = length(A);
x = normal_rnd(0, 1, n, 1);
d = filter(B, A, x);
mi=0.04;

w = zeros(N + M -1, 1);
Sb = zeros(N, 1);
Sa = zeros(M, 1);
vx = zeros(N, 1);
vy = zeros(M, 1);
vw = [];
ve = [];
delta = 1e-4;
SD = delta*eye(length(w));
lambda = 0.9;

for i = 1:n;
  vx = [ x(i); vx(1:N-1) ];
  Sb = [ x(i); Sb(1:N-1) ];
  y = w(1:3)'*vx - w(4:5)'*vy(1:M-1);
  vy = [ y; vy(1:M-1) ];
  Sa = [ y-w(4:5)'*Sa(1:M-1); Sa(1:M-1) ];
```

```
  e = d(i) - y;
  ve = [ ve; e ];
  phi = [ 0; -Sb(2); 0; Sa(2:3) ];
  SD = inv(lambda)*(SD - (SD*phi*phi'*SD)/(lambda + phi'*SD*phi));
  nw = w - SD*phi*e;
  if (abs(nw(5)) < 1 && abs(nw(4)) < 1 + nw(5))
    w = [ 1 nw(2) 1 nw(4:5)' ]';
    vw = [ vw; w' ];
  end
end
```

A.2 Simulação com duas seções de 2ª ordem

A mesma convenção utilizada na elaboração da listagem acima foi utilizada no programa desta seção. No algoritmo LMS, a variação do parâmetro μ (de $0,04$ para $0,0018$) foi necessária para possibilitar a convergência dos coeficientes. No caso do algoritmo RLS, o valor do parâmetro λ sofreu uma pequena alteração, $0,95$.

O filtro sendo identificado possui a seguinte função de transferência:

$$H(z) = \frac{(1+b_1z^{-1}+z^{-2})(1+b_2z^{-1}+z^{-2})}{(1+a_1z^{-1}+a_2z^{-2})} \tag{A.2}$$

Nos casos de teste exibidos nas figuras 3.6 e 3.9 os valores utilizados para coeficiente são: $b_1 = 2$, $b_2 = 5$, $a_1 = 1$ e $a_2 = 0,5$. O valor do parâmetro μ foi obtido experimentalmente. Para obter os gráficos das simulações citadas acima, foi utilizado o código apresentado a seguir. A adaptação com pólos fixos requer pequenas modificações, semelhantes às citadas na seção A.1.

```
n = 8000;
B1 = [ 1 2 1 ]';  N1 = length(B1);
B2 = [ 1 5 1 ]';  N2 = length(B2);
A =  [ 1 1 0.5 ]'; M = length(A);
x = normal_rnd(0, 1, n, 1);
d = filter(conv(B1, B2), A, x);
mi=0.0018;
```

```
w = zeros(4, 1); % Somente quatro coeficientes: b1, b2, a1, a2
Sb1 = zeros(N1, 1);
Sb2 = zeros(N2, 1);
Sa  = zeros(M, 1);
vx  = zeros(N1, 1);
vx1 = zeros(N2, 1);

vy  = zeros(M, 1);
vw  = [];
ve  = [];

for i = 1:n;
  vx  = [ x(i); vx(1:N1-1) ];
  Sb1 = [ [ 1 w(1) 1 ]*vx; Sb1(1:N1-1) ];

  vx1 = [ [ 1 w(1) 1 ]*vx; vx1(1:N2-1) ];
  Sb2 = [ vx(1); Sb2(1:N2-1) ];

  vx2 = [ 1 w(2) 1 ]*vx1;
  vy  = [ vx2-w(3:4)'*vy(1:2); vy(1:2) ];
  Sa  = [ vy(1)-w(3:4)'*Sa(1:2); Sa(1:2) ];

  e = d(i) - vy(1);
  ve = [ ve; e ];
  nw = w - mi * e * [ -Sb1(2) -Sb2(2) Sa(2:3)' ]';
  if (abs(nw(4)) < 1 && abs(nw(3)) < 1 + nw(4))
    w = nw;
    vw = [ vw; w' ];
  end
end
```

Utilizando o algoritmo RLS (figuras 3.7 e 3.10) o código do programa acima foi ligeiramente modiicado, permanecendo inalterados os valores dos coeficientes b_1, b_2, a_1 e a_2. A listagem abaixo mostra o programa relativo à adaptação RLS neste caso.

```
n = 2000;
B1 = [ 1 2 1 ]';  N1 = length(B1);
B2 = [ 1 5 1 ]';  N2 = length(B2);
A =  [ 1 1 0.5 ]'; M = length(A);
x = normal_rnd(0, 1, n, 1);
d = filter(conv(B1, B2), A, x);

w = zeros(4, 1); % Somente quatro coeficientes: b1, b2, a1, a2
Sb1 = zeros(N1, 1);
Sb2 = zeros(N2, 1);
Sa  = zeros(M, 1);
vx  = zeros(N1, 1);
vx1 = zeros(N2, 1);

vy  = zeros(M, 1);
vw  = [];
ve  = [];
delta = 1e-4;
SD = delta*eye(length(w));
lambda = 0.95;

for i = 1:n;
  vx  = [ x(i); vx(1:N1-1) ];
  Sb1 = [ [ 1 w(1) 1 ]*vx; Sb1(1:N1-1) ];

  vx1 = [ [ 1 w(1) 1 ]*vx; vx1(1:N2-1) ];
  Sb2 = [ vx(1); Sb2(1:N2-1) ];

  vx2 = [ 1 w(2) 1 ]*vx1;
  vy  = [ vx2-w(3:4)'*vy(1:2); vy(1:2) ];
  Sa  = [ vy(1)-w(3:4)'*Sa(1:2); Sa(1:2) ];

  e = d(i) - vy(1);
  ve = [ ve; e ];
  phi = [ -Sb1(2); -Sb2(2); Sa(2:3) ];
```

```
  SD = inv(lambda)*(SD - (SD*phi*phi'*SD)/(lambda + phi'*SD*phi));
  nw = w - SD*phi*e;
  if (abs(nw(4)) < 1 && abs(nw(3)) < 1 + nw(4))
    w = nw;
    vw = [ vw; w' ];
  end
end
```

A.3 Algoritmo com taxa reduzida

Nesta seção será apresentado o código do programa de simulação contendo o algoritmo de adaptação com taxa reduzida, utilizado na elaboração dos gráficos da figura 4.4.

```
c = [ 40 1 5 -3 ]';  % Coeficientes verdadeiros
den = [ 1 -0.9 0.2 ]'; % Coeficientes verdadeiros (denominador)

IT = 5000; % Número de iterações
m = 0.01; % mi
N = length(c); % Coeficientes do filtro FIR
M = N; % Comprimento da resposta impulsional dos filtros G's

p = zeros(N,1);  % Vetor de coeficientes adaptativos

% Sinais
u = normal_rnd(0, 1, IT, 1); % Sinal de entrada
d = filter(c, den, u); % Sinal desejado

% Filtros G's
G = eye(N,M);
% Matriz K
K = inv(G'*G)*G';
% Fator de decimação para u(k)
V=3;
% Fator de decimação para e(k)
W=2;
```

```
% Entrada FIFO
vu = zeros(N, 1);
vdu = zeros(N, 1);
vde = zeros(N, 1);

% Vetores para formação de gráficos
ve = [];
vy = zeros(3,1);
vp = [];
vd = [];

for i = [1:IT]
  vu = [ u(i); vu(1:N-1)];
  y = p' * vu - den(2:3)'*vy(1:2);
  vy = [ y; vy(1:2) ];

  if (mod(i,V) == 0)
    vdu = [ u(i)-den(2:3)'*vdu(1:2); vdu(1:N-1) ];
  end

  if (mod(i, W) == 0)
    e = d(i) - y;
    ve = [ ve e ];
    vde = [ e; vde(1:N-1) ];
  end

  if (mod(i, V*W) == 0)
    p = p + 2 * m * K * vde .* vdu;
    vp = [ vp p ];
  end
end
```