

UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE
ELETRICIDADE

*Um algoritmo tipo RLS baseado em superfícies
não quadráticas*

CRISTIANE CRISTINA SOUSA DA SILVA

São Luis - MA, Brasil

6 de setembro de 2013

UM ALGORITMO TIPO RLS BASEADO EM SUPERFÍCIES NÃO QUADRÁTICAS

Tese de doutorado submetida à coordenação do Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como parte dos requisitos necessários para obtenção do grau de doutora em engenharia de eletricidade na área de automação e controle.

CRISTIANE CRISTINA SOUSA DA SILVA

Julho, 2013

Silva, Cristiane Cristina Sousa da.

Um algoritmo tipo RLS baseado em superfícies não quadráticas/ Cristiane Cristina Sousa da Silva – São Luís, 2013.

77 f.

Impresso por computador (fotocópia).

Orientador: Prof. Allan Kardec Duailibe Barros Filho, Ph.D..

Tese (Doutorado) – Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade, 2013.

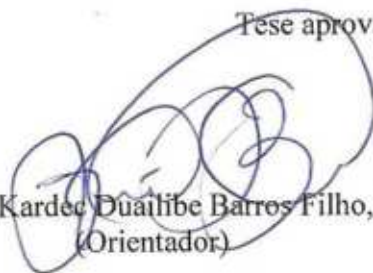
1. Filtragem adaptativa. 2. Função não quadrática. 3. Velocidade de convergência I. Título.

CDU 621.372.544

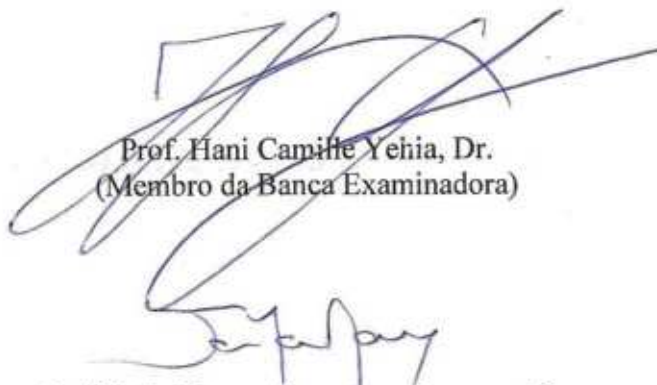
UM ALGORITMO TIPO RLS BASEADO EM SUPERFÍCIES NÃO QUADRÁTICAS

Cristiane Cristina Sousa da Silva.

Tese aprovada em 19 de julho de 2013.



Prof. Allan Kardec Duailibe Barros Filho, Ph.D.
(Orientador)



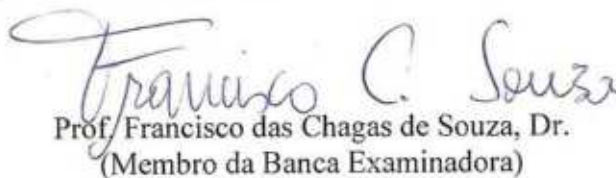
Prof. Hani Camille Yehia, Dr.
(Membro da Banca Examinadora)



Prof. João Marcos Travassos Romano, Dr.
(Membro da Banca Examinadora)



Prof. Ewaldo Eder Carvalho Santana, Dr.
(Membro da Banca Examinadora)



Prof. Francisco das Chagas de Souza, Dr.
(Membro da Banca Examinadora)

AGRADECIMENTOS

Ao professor Allan Kardec Duailibe Barros Filho pela motivação, apoio, carinho e dedicação, fundamentais para a orientação deste trabalho. Pela oportunidade de crescimento e aprendizado. Professor e orientador no mestrado e no doutorado, agradeço por sua cumplicidade e responsabilidade direta na construção desta Tese.

Ao professor Jose Carlos Principe pelo incentivo e amizade. Gostaria de ratificar a sua competência, participação com discussões, correções e sugestões que fizeram com que concluíssemos este trabalho.

Aos professores Ewaldo Eder Carvalho Santana e João Viana Fonseca Neto, Co-Orientadores deste trabalho, pelo incentivo, orientação, e crédito durante todas as fases do curso de doutorado.

Aos professores João Marcos Travassos Romano e Hani Camile Yehia, pela orientação nas correções e sugestões para a conclusão desse trabalho.

Ao professor Marcos Antonio F. de Araújo, pela atenção, carinho, amizade e dedicação a mim dispensados em todas as fases do curso de doutorado.

A todos os meus amigos do PIB pelo companherismo e amizades sinceras.

A Deus pelas oportunidades que me foram dadas.

À toda a minha família pelo carinho, apoio e compreensão ao longo do ano em que o presente trabalho foi desenvolvido. Em especial ao meu marido Marcio John Moreira e meus filhos Gustavo Henrique Sousa S. Moreira e Marcio John Moreira Júnior.

À CAPES pela bolsa a mim concedida.

Resumo

Em filtragem adaptativa, vários filtros são baseados no método do erro quadrático médio (do inglês, *MSE-mean squared error*) e muitos desses foram desenvolvidos para obter uma convergência rápida com um menor desajuste. Os algoritmos mínimo quadrático médio (do inglês, *LMS-least mean square*) e mínimos quadrados recursivos (do inglês, *RLS- recursive least square*) foram um marco em filtragem adaptativa.

Nesse trabalho apresentamos o desenvolvimento de uma família de algoritmos adaptativos baseados nas potências pares do erro, inspirado na dedução do algoritmo RLS padrão. Chamaremos esses novos algoritmos de **recursivo não-quadrático** (RNQ). A idéia básica é baseada na função de custo apresentada por Widrow no algoritmo mínimo quarto médio (do inglês, *LMF-least mean square fourth*).

Inicialmente derivamos equações baseadas em uma potência par do erro para obter critérios que garantam a convergência. Determinamos também, equações que definem o desajuste e o tempo de aprendizagem do processo de adaptação do algoritmo RNQ baseado em uma potência par arbitrária.

Trabalhamos também, no sentido de tornar o algoritmo menos sensível ao tamanho do erro numa direção alternativa, propondo uma função de custo baseada na soma das potências pares do erro. Essa segunda abordagem torna explícito o papel do erro na formulação do RLS ao propor uma nova função de custo que preserve a solução MSE, mas permite a utilização dos momentos de alta ordem do erro para aumentar a velocidade de convergência do algoritmo.

O principal objetivo do nosso trabalho é criar a partir dos primeiros princípios (novas funções de custo) um mecanismo para incluir informações de erro instantâneo no algoritmo RLS e torná-lo um seguidor melhor. Assim, o aspecto-chave dessa nova abordagem é incluir o erro no "ganho de Kalman" que controla efetivamente a velocidade de adaptação do algoritmo de RLS.

Palavras-chaves: Filtragem adaptativa, função não-quadrática e velocidade de convergência.

Abstract

In adaptive filtering many adaptive filter are based on the mean square error method (MSE). These filters were developed to improve convergence speed with a lower misadjustment. The least mean square (LMS) and the recursive least square (RLS) algorithms have been the hallmark of adaptive filtering.

In this work we develop adaptive algorithms based on the even powers of the error inspired in the recursive least square (RLS) algorithm. Namely **recursive non quadratic (RNQ)** algorithm. The idea is based on Widrow's least mean square fourth (LMF) algorithm.

First we derive equations based on a single even power of the error in order to obtain criterions that guarantee convergence. We also determine equations that measure the misadjustment and the time constant of the adaptive process of the RNQ algorithm.

We work also, toward making the algorithm less sensitive to the size of the error in an alternative direction, by proposing a cost function which is a sum of the even powers of the error. This second approach bring the error explicitly to the RLS algorithm formulation by proposing a new cost function that preserves the mean-square-error (MSE) solution, but allows for the exploitation of higher order moments of the error to speedup the convergence of the algorithm.

The main goal this work is to create from first principles (new cost functions) a mechanism to include instantaneous error information in the RLS algorithm, make it track better, and allow for the design of the forgetting factor. As we will see the key aspect of our approach is to include the error in the "Kalman gain" that effectively controls the speed of adaptation of the RLS algorithm.

Keywords: Adaptive filtering, non-quadratic function and convergence speed.

Lista de Figuras

| | | |
|-----|--|----|
| 2.1 | Combinador Linear Adaptativo - forma transversal | 8 |
| 2.2 | Porção de uma superfície quadrática tridimensional, juntamente com alguns contornos. O erro quadrático médio está plotado na vertical, w_0 e w_1 variam de -1 a 1 | 10 |
| 3.1 | Estrutura de um filtro adaptativo no contexto de identificação de sistema. \mathbf{u} é o vetor de entrada, h é o filtro FIR, \mathbf{w} é o vetor peso, e é o sinal erro, d é o sinal desejado, e y é a saída do filtro | 34 |
| 3.2 | Curvas de aprendizagem. As simulações foram realizadas com e sem ruídos. (a) Sem ruído. (b), (c) e (d) adicionando ruídos com distribuição uniforme, gaussiana e laplaciana, respectivamente. Foi utilizado $\lambda = 0,95$ para todas as simulações, e utilizamos também três valores para j ($j = 1, 2$ e 3). | 35 |
| 3.3 | Os decaimentos exponenciais para os dois primeiros estágios do tempo de convergência associados ao algoritmo RNQ. As linhas retas denotam a curva de aprendizagem; as linhas tracejadas representam o primeiro estágio do tempo de convergência e as linhas pontilhadas representam o segundo estágio, a qual é aproximadamente o mesmo decaimento exponencial da constante de tempo associada ao algoritmo RLS. (a) A curva de aprendizagem do algoritmo RNQ para $j = 2$ e (b) a curva de aprendizagem do algoritmo RNQ para $j = 3$ | 36 |
| 4.1 | Gráficos das funções $(e^2 + e^4 + e^6)$ e e^2 , onde podemos ver a maior inclinação da primeira, no intervalo $[-1; 1]$ | 40 |

| | | |
|-----|--|----|
| 4.2 | Estrutura de um filtro adaptativo no contexto de identificação de sistema, sendo u_i o vetor de entrada, \mathbf{h} um filtro FIR, d_i o sinal desejado, η_i o sinal ruído, e_i o sinal erro e y_i a saída do filtro. | 48 |
| 4.3 | Desempenhos (a) do tempo de adaptação e (b) do MSE pelo número m de termos da soma e pelos valores do parâmetro k apresentados no termo de ponderação a_j com $\lambda = 0.96$, sendo $m = 1, 2, 3, \dots, 14, 15$ e $k = 1, 2, 3, \dots, 49, 50$ para ambos os gráficos. | 50 |
| 4.4 | Desempenhos (a) do tempo de adaptação e (b) do MSE pelo número m de termos da soma e por dez diferentes valores para o fator de esquecimento λ com $k = 10$, sendo $m = 1, 2, 3, \dots, 14, 15$ e $\lambda = 0, 90; 0, 91; 0, 92, \dots; 0, 98; 0, 99$ para ambos os gráficos. | 51 |
| 4.5 | Desempenhos (a) do tempo de adaptação e (b) do MSE pelos valores do parâmetro k apresentados no termo de ponderação a_j e por dez diferentes valores para o fator de esquecimento λ com $m = 15$, sendo $m = 1, 2, 3, \dots, 14, 15$ e $\lambda = 0, 90; 0, 91; 0, 92, \dots; 0, 98; 0, 99$ para ambos os gráficos. | 52 |
| 4.6 | Curvas de aprendizagem dos algoritmos RLS, RNQ de Silva et al. [13] com $j = 3$ e o algoritmo RNQ com $m = 5, 10$ e 15 . Para ambos os algoritmos utilizamos o fator de esquecimento $\lambda = 0, 96$. As simulações foram feitas sem ruído. | 53 |
| 4.7 | Desempenho do vetor ganho e do fator $\frac{\epsilon^2}{k}$, sendo $k = 5$ com $m = 5$ e $\lambda = 0, 96$ pelo número de iterações. | 53 |
| 4.8 | Curvas de aprendizagem dos algoritmos RLS e RNQ com $m = 5$ e $m = 10$. Para ambos os algoritmos utilizamos $\lambda = 0, 96$. As simulações foram feitas com ruído, (a) Gaussiano e (b) Laplaciano. A relação sinal ruído é de $30dB$ para ambos os casos. | 54 |
| 4.9 | Desempenho do vetor ganho e do fator $\frac{\epsilon^2}{k}$, sendo $k = 10$ com $m = 10$, $\lambda = 0, 96$ e com ruído Laplaciano pelo número de iterações. | 55 |

Lista de Tabelas

| | | |
|-----|---|----|
| 3.1 | Resumo do algoritmo RNQ para uma potência par do erro | 26 |
| 3.2 | Melhora no desempenho do algoritmo RNQ para uma potência par do erro em termos de ξ_{mim} em relação ao RLS padrão para três diferentes interferências de ruídos. | 36 |
| 4.1 | Algoritmo RNQ baseado na soma das potências pares do erro | 45 |

Lista de Abreviaturas

MSE *Mean squared error* Erro quadrático médio.

CLA Combinador linear adaptativo.

FIR *Finite impulse response* Resposta ao impulso finita.

LMS *Least mean square* Mínimo quadrado médio.

RLS *Recursive least square* Mínimo quadrado recursivo.

RNQ Recursivo não-quadrático.

LMF *Least mean fourth* Mínimo quarto médio.

FTF *Fast transversal filter* Filtro transversal rápido.

fastRLS *Fast recursive least square* Mínimo quadrado recursivo rápido.

Lista de Símbolos

\mathbf{u} vetor de entrada.

\mathbf{w} vetor peso.

L ordem do filtro.

\mathbf{y} vetor de saída.

d sinal desejado.

e sinal de erro.

∇ operador gradiente.

ϵ função de custo no RLS.

φ matriz de auto-correlação do sinal de entrada no RLS.

\mathbf{z} vetor de correlação cruzada do sinal de entrada com o sinal desejado.

μ passo de adaptação.

ρ fator de ponderação.

λ fator de esquecimento.

tr traço.

J função de custo no RNQ.

ϕ matriz de auto-correlação do vetor de entrada no RNQ.

\mathbf{g} vetor ganho.

\mathbf{P} matriz inversa da matriz de auto-correlação do vetor de entrada.

ξ estimativa a priori do erro.

\mathbf{v} vetor desvio.

\mathbf{K} matriz de auto-correlação do vetor desvio.

$\mathbf{\Lambda}$ matriz diagonal.

\mathbf{U} matriz dos dados de entrada.

τ_{RNQ_1} tempo de convergência do RNQ nos instantes iniciais.

τ_{RNQ_2} tempo de convergência do RNQ após os instantes iniciais.

\mathcal{M}_{RLS} o desajuste do RLS.

\mathcal{M}_{RNQ} o desajuste do RNQ.

h filtro FIR.

m inteiro positivo que define a quantidade de termos no somatório do RNQ.

j inteiro positivo que define a potência par do erro.

k^{m-j} fator de ponderação no somatório do RNQ.

$a_j = j.k^{m-j}$ fator de ponderação para simplificar os cálculos.

$\alpha_j = 2j - 2$ expressão que simplifica os cálculos na potência do erro.

Sumário

| | | |
|----------|--|-----------|
| 1 | Introdução | 2 |
| 1.1 | Motivações | 3 |
| 1.2 | Organização do texto | 5 |
| 2 | Filtragem adaptativa | 7 |
| 2.1 | Introdução | 7 |
| 2.2 | O combinador linear adaptativo | 8 |
| 2.3 | Algoritmos de gradiente estocástico | 9 |
| 2.4 | Algoritmos de mínimos quadrados | 11 |
| 2.5 | O algoritmo mínimos quadrados recursivo (<i>recursive least squares-RLS</i>) | 12 |
| 2.5.1 | Dedução do algoritmo RLS | 12 |
| 2.5.2 | Convergência do algoritmo <i>RLS</i> | 15 |
| 2.6 | Conclusão | 21 |
| 3 | Algoritmo RNQ baseado em uma única potência par do erro | 22 |
| 3.1 | Introdução | 22 |
| 3.2 | Dedução do algoritmo | 22 |
| 3.2.1 | Atualização do vetor peso | 25 |
| 3.2.2 | Resumo do algoritmo | 26 |
| 3.3 | Convergência do algoritmo | 27 |
| 3.3.1 | O comportamento médio do vetor peso no algoritmo | 27 |
| 3.3.2 | Matriz de correlação do vetor desvio | 28 |
| 3.3.3 | Curva de aprendizagem | 29 |
| 3.3.4 | Análise do tempo de aprendizagem | 31 |
| 3.3.5 | Desajuste | 31 |

| | | |
|----------|--|-----------|
| 3.3.6 | Análise comparativa | 32 |
| 3.3.7 | Complexidade computacional | 33 |
| 3.4 | Resultados | 33 |
| 3.5 | Discussões e conclusões | 35 |
| 4 | Algoritmo RNQ baseado na soma de potências pares do erro | 39 |
| 4.1 | Introdução | 39 |
| 4.2 | Dedução do algoritmo | 39 |
| 4.2.1 | Atualização do vetor peso | 44 |
| 4.2.2 | Resumo do algoritmo RNQ baseado na soma das potências pares do erro | 45 |
| 4.2.3 | Análise do vetor ganho do algoritmo RNQ | 45 |
| 4.3 | Resultados | 47 |
| 4.3.1 | Identificação perfeita de sistema usando um filtro FIR | 47 |
| 4.3.2 | Identificação de sistema ruidoso usando um filtro FIR | 49 |
| 4.4 | Conclusão | 55 |
| 5 | Conclusão e trabalhos futuros | 57 |
| | Referências Bibliográficas | 60 |

Capítulo 1

Introdução

Os sinais envolvidos em processamentos de filtragem, predição e estimação são sinais aleatórios, os quais são caracterizados por suas propriedades estatísticas. O projeto de filtro para o processamento de sinais aleatórios requer o conhecimento prévio de algumas informações sobre as propriedades estatísticas dos sinais envolvidos. Quando isso é possível, trata-se o problema no âmbito do processamento estatístico de sinais. Nos casos em que tais informações são desconhecidas e não podem ser estimadas em tempo real, a melhor solução é o emprego de filtros adaptativos [1].

A filtragem adaptativa constitui uma ferramenta fundamental no processamento de sinais digitais. Essa técnica tem sido explorada com sucesso em um grande número de problemas de economia, engenharia biomédica, equalização de canais, sistemas de controle e em telecomunicações. Em especial, na área biomédica, diversas aplicações podem ser encontradas, tais como cancelamento de interferências do coração no eletrocardiograma (ECG) e cancelamento da influencia materna em ECG fetal.

Trabalhos em filtragem adaptativa envolvem o estudo de algoritmos e de estruturas de filtragem de forma a melhorar o desempenho dos sistemas adaptativos existentes, cuja a estrutura é alterável (através do ajuste dos seus coeficientes) de tal forma que seu comportamento melhore de acordo com algum critério de desempenho através da exposição ao ambiente no qual será inserido. O ajuste

dos coeficientes do filtro adaptativo é realizado através da implementação de um algoritmo, devidamente escolhido, cujo objetivo é atender a requisitos dos sistemas. Estes algoritmos são definidos como algoritmos adaptativos. Muito do sucesso dos filtros adaptativos é devido ao desenvolvimento do popular e robusto algoritmo mínimo quadrado médio (do inglês, *LMS-least mean square*) [2](p.99 - 111), o qual determina os parâmetros do filtro minimizando o critério do erro quadrático médio (do inglês, *MSE-mean squared-error*). Além disso, o algoritmo convergirá para a solução de Wiener, na teoria de estimação de mínimos quadrados [3].

Os algoritmos LMS e mínimo quadrado recursivo (do inglês, *RLS-recursive least square*) foram um marco em filtragem adaptativa, mas com desenvolvimento baseado em duas abordagens muito diferentes. O primeiro sendo uma aproximação estocástica de gradiente descendente e o último uma formulação matemática recursiva da solução de Wiener. Normalmente, pode-se destacar como a maior diferença entre os dois algoritmos, o uso da aproximação de segunda ordem no RLS, isto é, que o RLS faz uma busca online de maneira similar ao método de Newton, usando passos diferente na busca do mínimo. Mas, o erro instantâneo nunca aparece explicitamente no algoritmo RLS, e isso é ruim, diferente do LMS que usa como estimativa do erro o próprio erro instantâneo, pois o erro fornece informação instantânea sobre estimação paramétrica do modelo para cada amostra da série temporal. Essa é a razão pela qual o LMS é um seguidor rápido e porque o RLS tem dificuldades em seguir o sinal desejado, como bem conhecido na literatura [1].

1.1 Motivações

Motivado por aplicações em tempo real de algoritmos de processamento de sinais adaptativos, diversas modificações e novos algoritmos têm sido propostos em filtragem adaptativa. Trabalhos anteriores sobre combinações convexas de filtros adaptativos restringiram-se principalmente a combinações de filtros das mesmas famílias, ou seja, dois LMS [4, 5, 6], ou dois RLS [4, 7]. Uma combinação de dois filtros baseados em diferentes funções de custo foi proposto em [8]. Usando uma regra de combinação diferente, foram propostas as combinações de filtros de

Kalman ou RLS em [9, 10]. Em [11] o desenvolvimento dos filtros são baseados em uma combinação convexa de filtros de famílias diferentes, objetivando alcançar um filtro geral com o desempenho de rastreamento superior.

Modificações na dedução do algoritmo RLS padrão foram feitas no intuito de reduzir sua carga computacional, por exemplo, o algoritmo *RLS approximate* [12] utiliza uma abordagem de substituição da matriz de correlação de dados do algoritmo RLS por um escalar. Os algoritmos *fast RLS* [14, 15] e o *fast transversal filter* (FTF) [16] usam propriedades dos dados para reduzir o custo computacional do algoritmo RLS. No entanto esses algoritmos têm códigos bastante complexos, que podem não ser adequados para aplicações em tempo real, e apresentam um comportamento instável de operação com grandes conjuntos de dados [17].

Identificação de sistemas não-lineares e problemas de controle em [18, 19, 20, 21, 22, 23, 24, 25] demonstram aplicações do algoritmo RLS modificado para problemas práticos e complexos. Em outros trabalhos, o uso da estatística de alta ordem como forma de extrair mais informações dos sinais envolvidos no processo de adaptação tem se mostrado útil em sistemas adaptativos [26, 27, 28, 29, 30].

Trabalhos propostos anteriormente, como em [31, 32] utilizam o fator de esquecimento variável na dedução do algoritmo RLS. Novos resultados de convergência para os algoritmos RLS, com modificações no fator de esquecimento, têm sido apresentados em [33, 34, 35, 36]. Outros tentaram incluir o erro instantâneo no algoritmo de RLS, mas os métodos têm sido heurísticos. Em [37] uma técnica para melhorar a taxa de convergência do algoritmo RLS é apresentada, onde o vetor de ganho leva em consideração o valor de $\left| \frac{1}{\xi(n)} \right|$, sendo $\xi(n)$ uma estimativa a priori do erro.

Em nosso trabalho é explícito o papel do erro na formulação do RLS ao propor uma nova função de custo que preserve a solução MSE, permitindo a utilização dos momentos de alta ordem do erro para aumentar a velocidade de convergência do algoritmo RLS. A idéia é baseada no algoritmo mínimo quarto médio (do inglês, LMF *least mean square fourth*) [28], onde a velocidade da dinâmica de aprendizagem pode ser aumentada para que se atinja a mesma solução MSE. Isso pode ser compreendido geometricamente, pois as funções de custo dos erros quadrático e de

quarta potência têm o mesmo mínimo global, mas na maior parte do espaço de busca, a quarta potência tem uma inclinação maior, exceto na vizinhança do mínimo, onde a velocidade de adaptação é maior. Essa idéia básica foi utilizada em [13], onde foi pela primeira vez ligado ao algoritmo RLS, isto é, este artigo mostrou que é possível desenvolver um algoritmo similar ao RLS para a superfície de desempenho do erro de quarta potência, que de fato gerou melhores tempos de convergência. Entretanto o artigo analisou o porquê disto e nem caracterizou totalmente os parâmetros livres contidos na nova formulação. Além disso, o problema da pequena curvatura da superfície de desempenho perto do ponto ótimo não foi estudado.

Neste trabalho apresentamos o desenvolvimento de uma família de algoritmos adaptativos baseados nas potências pares do erro, inspirado na dedução do algoritmo RLS [1](p.562 - 569). Chamaremos esses novos algoritmos de **recursivo não-quadrático** (RNQ). O principal objetivo do nosso trabalho é criar a partir dos primeiros princípios (novas funções de custo) um mecanismo para incluir informações de erro instantâneo no algoritmo RLS e torná-lo um seguidor melhor. Dessa forma, como iremos apresentar, o aspecto-chave dessa nova abordagem é incluir o erro no “ganho de Kalman” [1] que controla efetivamente a velocidade de adaptação do algoritmo de RLS.

1.2 Organização do texto

Este trabalho está organizado em cinco capítulos. No **Capítulo 2**, apresentamos o combinador linear adaptativo, fazemos uma revisão da superfície quadrática e um breve comentário sobre algoritmos de gradiente estocástico e de mínimos quadrados. Revisamos também, o algoritmo *RLS* mostrando a sua dedução, convergência do vetor peso, tempo de aprendizagem e desajuste final.

No **Capítulo 3**, desenvolvemos um novo algoritmo baseado em uma única potência par do erro. Obtemos expressão para garantir a convergência e determinamos o desajuste. Simulações são realizadas para comparar o desempenho do algoritmo proposto com o *RLS* padrão. Apresentamos também discussões e conclusões sobre o desenvolvimento desse novo algoritmo.

No **Capítulo 4**, desenvolvemos o algoritmo RNQ baseado na soma ponderada

das potências pares do erro. Analisamos os parâmetros livres presentes e a curva de aprendizagem do algoritmo proposto através de simulações. Mostramos que a inclusão do erro instantâneo no vetor ganho controla efetivamente a velocidade de adaptação do algoritmo. Discussões e conclusões sobre o algoritmo RNQ também são apresentadas.

Finalmente no **Capítulo 5**, apresentamos a conclusão e os trabalhos futuros.

Capítulo 2

Filtragem adaptativa

2.1 Introdução

Os filtros adaptativos representam uma parte significativa no processamento de sinais digitais. Historicamente, a abordagem paramétrica de sinais tem sido explorada com sucesso em problemas de comunicações, controle robótica, radar, sismologia e engenharia biomédica. Os chamados problemas de filtragem podem ser identificados e caracterizados mais especificamente pelos termos de filtragem, suavização, predição [1](p.03).

O principal objetivo da filtragem de sinais é melhorar a qualidade do sinal de acordo com um critério de desempenho. Os sinais podem ser considerados tanto no domínio do tempo com no domínio da frequência. A diferença dos filtros adaptativos com relação aos demais é o seu desempenho auto-ajustável e variante no tempo.

Muitos algoritmos adaptativos se utilizam do erro quadrático médio (do inglês, *MSE-mean squared-error*) como função de custo que se deseja minimizar. O erro quadrático médio é uma função convexa dos componentes do vetor peso e gera uma superfície hiperparabolóide que garante a existência de um mínimo global. O problema consiste em determinar procedimentos de tal forma a encontrar esse mínimo, o mais rapidamente possível e com o menor erro final.

2.2 O combinador linear adaptativo

A estrutura mais usada na implementação de filtros adaptativos é o combinador linear adaptativo (CLA), mostrado na Figura 2.1[1](p.15). Pode-se observar que o filtro adaptativo possui uma entrada única, \mathbf{u}_i (no tempo i), definida como

$$\mathbf{u}_i = [u_i, u_{i-1}, \dots, u_{i-(L-1)}]^T \quad (2.1)$$

sendo L a ordem do filtro.

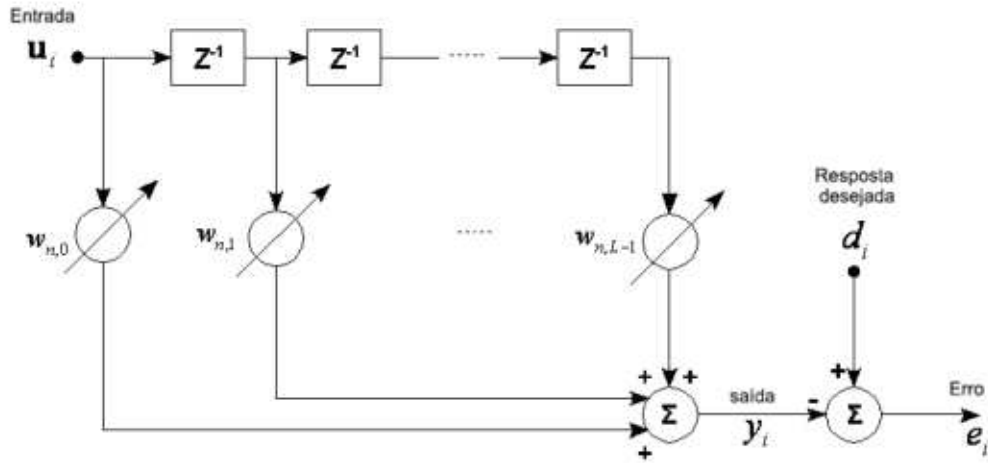


Figura 2.1: Combinador Linear Adaptativo - forma transversal

\mathbf{w}_n , é o vetor peso no tempo n , definido por

$$\mathbf{w}_n = [w_{n0} \ w_{n1} \ \dots \ w_{n(L-1)}]^T, \quad (2.2)$$

e a saída, y_i , é igual ao produto interno de \mathbf{u}_i por \mathbf{w}_n :

$$y_i = \mathbf{u}_i^T \mathbf{w}_n = \mathbf{w}_n^T \mathbf{u}_i. \quad (2.3)$$

Conforme visto na Figura 2.1, o sinal de erro no instante i é dado por

$$e_i = d_i - y_i. \quad (2.4)$$

Substituindo (2.3) nesta expressão, temos:

$$e_i = d_i - \mathbf{u}_i^T \mathbf{w}_n = d_i - \mathbf{w}_n^T \mathbf{u}_i. \quad (2.5)$$

Existem vários algoritmos e abordagens que podem ser utilizados em filtragem adaptativa, dependendo dos requisitos do problema. No entanto, existem duas abordagens principais para o desenvolvimento de algoritmos de filtros adaptativos [1]. Discutiremos essas duas abordagens nas próximas seções.

2.3 Algoritmos de gradiente estocástico

O MSE, como já foi dito, é uma função convexa dos componentes do vetor peso e gera uma superfície hiperparabolóide que garante a existência de um mínimo global, o que representa a solução ótima de Wiener [1](p.203-205). Esta solução pode ser encontrada pelo bem conhecido método de otimização, denominado “método de descida mais íngreme”, que utiliza o vetor gradiente para descer gradualmente passo a passo para o mínimo da função de erro. As chamadas equações de Wiener-Hopf, em forma matricial, definem esta solução ótima de Wiener. Vamos, agora determinar a solução ótima a partir do critério do erro quadrático médio dado por [2](p.54-60).

$$\xi = E[e_i^2] = E[(d_i - \mathbf{w}_n^T \mathbf{u}_i)^2]. \quad (2.6)$$

Considerando um ambiente estacionário, podemos desenvolver a Equação (2.6) como

$$\xi = d_i + \mathbf{w}^T \boldsymbol{\varphi} \mathbf{w} - 2\mathbf{z}^T \mathbf{w}, \quad (2.7)$$

sendo $\boldsymbol{\varphi}$ a matriz de auto-correlação do sinal de entrada \mathbf{u}_i e \mathbf{z} a vetor de correlação cruzada do sinal de entrada \mathbf{u}_i com o sinal desejado d_i .

Pode-se observar que o MSE é uma função quadrática dos pesos, cujo gráfico é uma superfície côncava hiperparabólica, conforme vemos na Figura 2.2, onde consideramos apenas dois pesos. Esta função, obviamente, nunca pode ser negativa.

O gradiente da superfície de desempenho do erro quadrático médio, designado por ∇ , pode ser obtido derivando a Equação (2.7) para obter o erro quadrático médio mínimo, isto é:

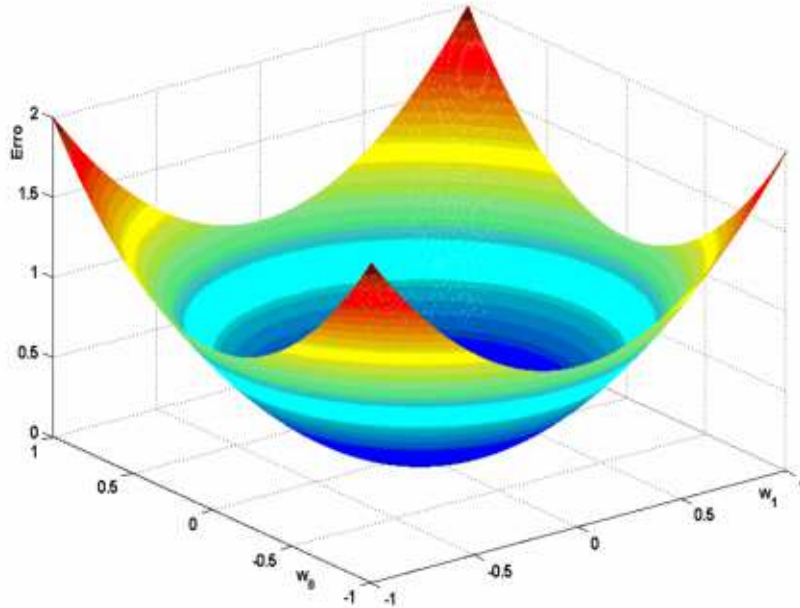


Figura 2.2: Porção de uma superfície quadrática tridimensional, juntamente com alguns contornos. O erro quadrático médio está plotado na vertical, w_0 e w_1 variam de -1 a 1

$$\hat{\nabla}_n = \begin{bmatrix} \frac{\partial e_n^2}{\partial w_0} \\ \vdots \\ \frac{\partial e_n^2}{\partial w_L} \end{bmatrix} = 2\varphi\mathbf{w} - 2\mathbf{z}. \quad (2.8)$$

O vetor peso é ajustado para seu valor ótimo, \mathbf{w}_* , onde o gradiente é zero, ou seja:

$$\mathbf{w}_* = \varphi^{-1}\mathbf{z}. \quad (2.9)$$

Esta equação é a forma matricial da equação de Wiener-Hopf [1](p.203-205).

Um sistema modificado das equações de Wiener-Hopf, usado para adaptar os pesos do filtro em direção ao mínimo é o algoritmo chamado mínimo quadrado médio (do inglês, *LMS-least mean squares*), inventado por B. Widrow e M.E.Hoff Jr em 1959. Este algoritmo calcula o gradiente da função de erro de valores instantâneos da matriz de correlação das entradas e do vetor de correlação cruzada entre as entradas e a resposta desejada. Nesta seção, faremos uma breve discussão deste algoritmo.

Para desenvolver o algoritmo *LMS*, usamos o próprio e_i^2 como uma estimativa de ξ_i . Então, a cada iteração, no processo adaptativo, nós teremos uma estimação do gradiente da forma

$$\hat{\nabla}_n = \begin{bmatrix} \frac{\partial e_n^2}{\partial w_0} \\ \vdots \\ \frac{\partial e_n^2}{\partial w_L} \end{bmatrix} = 2e_n \begin{bmatrix} \frac{\partial e_n}{\partial w_0} \\ \vdots \\ \frac{\partial e_n}{\partial w_L} \end{bmatrix} = -2e_n \mathbf{u}_n. \quad (2.10)$$

As derivadas de e_i , em relação aos pesos, seguem, diretamente da Equação (2.4).

A partir da Equação (2.10) temos o algoritmo *LMS* [28]

$$\mathbf{W}_n = \mathbf{W}_{n-1} + \mu e_n \mathbf{u}_n, \quad (2.11)$$

sendo μ o passo de adaptação. Tal parâmetro é uma constante que comanda a velocidade de convergência do algoritmo.

2.4 Algoritmos de mínimos quadrados

Na dedução do algoritmo *LMS* o objetivo é minimizar o quadrado médio da estimação do erro. No método dos mínimos quadrados, por outro lado, no instante $n > 0$, os pesos do filtro adaptativo são calculados tal que a quantidade

$$\zeta_n = \sum_{i=1}^n \rho_i \cdot |e_i|^2, \quad (2.12)$$

é minimizada, sendo ρ_i um fator de ponderação, daí o nome mínimos quadrados

Um algoritmo muito popular da família dos mínimos quadrados é o algoritmo mínimo quadrado recursivo (do inglês, *RLS-recursive least squares*). Tal algoritmo tem como vantagem a baixa sensibilidade à natureza do sinal de entrada e uma maior velocidade de convergência quando comparado com os algoritmos de gradiente estocástico [1](p.580-583). Na próxima seção faremos a dedução deste algoritmo.

2.5 O algoritmo mínimos quadrados recursivo (*recursive least squares-RLS*)

Em implementações recursivas do método dos mínimos quadrados, começamos com condições iniciais conhecidas e utilizamos a informação contida em novas amostras de dados para atualizar as estimativas passadas. Assim, encontramos que o tamanho do dado observável é variável. Desta forma, expressamos a função de custo para ser minimizada como ε_n , sendo n o tamanho variável do dado observável. Portanto é comum introduzir um fator peso na definição da função de custo.

2.5.1 Dedução do algoritmo RLS

No algoritmo *RLS* o fator ponderação ρ_i é escolhido como sendo

$$\rho_i = \lambda^{n-i}, \quad (2.13)$$

sendo $0 \ll \lambda < 1$ uma constante positiva a ser escolhida.

O método dos mínimos quadrados padrão visto anteriormente corresponde ao caso em que $\lambda = 1$. O parâmetro λ é denominado fator de esquecimento. Claramente quando $\lambda < 1$, os fatores de ponderação definidos (2.13) dão um peso maior às amostras recentes das estimativas do erro (e assim, às amostras recentes do dado observado) comparado com as amostras mais antigas.

Substituindo (2.13) em (2.12) obtem-se a função de custo a ser minimizada na dedução do algoritmo *RLS*:

$$\varepsilon_n = \sum_{i=1}^n \lambda^{n-i} \cdot |e_i|^2. \quad (2.14)$$

Supondo que φ seja uma matriz não singular, o valor ótimo do vetor peso, $\hat{\mathbf{w}}_n$, para que a função de custo atinja este valor mínimo é definido pela equação normal escrita em forma matricial

$$\hat{\mathbf{w}}_n = \varphi_n^{-1} \mathbf{z}_n, \quad (2.15)$$

sendo

$$\varphi_n = \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i \mathbf{u}_i^T, \quad (2.16)$$

a matriz de auto-correlação do sinal de entrada \mathbf{u}_i e

$$\mathbf{z}_n = \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i d_i, \quad (2.17)$$

o vetor de correlação cruzada do sinal de entrada \mathbf{u}_i com o sinal desejado d_i .

Expandindo a Equação (2.16) e isolando o termo $i = n$, temos

$$\begin{aligned} \varphi_n &= \lambda \left[\sum_{i=1}^{n-1} \lambda^{n-1-i} \mathbf{u}_i \mathbf{u}_i^T \right] + \mathbf{u}_n \mathbf{u}_n^T \\ \varphi_n &= \lambda [\varphi_{n-1}] + \mathbf{u}_n \mathbf{u}_n^T. \end{aligned} \quad (2.18)$$

Analogamente podemos escrever a Equação (2.17) da seguinte forma

$$\mathbf{z}_n = \lambda [\mathbf{z}_{n-1}] + \mathbf{u}_n d_n. \quad (2.19)$$

O lema de inversão de matrizes

Lema 1: Sejam \mathbf{A} e \mathbf{B} matrizes definidas positivas de dimensão $L \times L$, definimos

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C} \mathbf{D}^{-1} \mathbf{C}^T \quad (2.20)$$

De 2.20 obtemos

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B} \mathbf{C} (\mathbf{D} + \mathbf{C}^T \mathbf{B} \mathbf{C})^{-1} \mathbf{C}^T \mathbf{B} \quad (2.21)$$

Sendo \mathbf{D} matriz positivamente definida de dimensão $N \times N$ e \mathbf{C} uma matriz $L \times N$.

A demonstração desse Lema é estabelecida pela multiplicação da Equação (2.20) por (2.21). Na próxima seção mostraremos como o lema de inversão de matrizes pode ser aplicado para obter uma equação recursiva.

O algoritmo *RLS* ponderado exponencialmente

Considerando que a matriz de auto-correlação é positivamente definida e não singular, podemos aplicar o lema de inversão de matrizes para equação recursiva (2.16). Primeiramente faremos as seguintes identificações:

$$\begin{cases} \mathbf{A} = \varphi_n \\ \mathbf{B}^{-1} = \lambda\varphi_{n-1} \rightarrow \mathbf{B} = \lambda^{-1}\varphi_{n-1}^{-1} \\ \mathbf{C} = \mathbf{u}_n \\ \mathbf{D} = \mathbf{I} \end{cases} \quad (2.22)$$

Substituindo as definições (2.22) na Equação (2.21), obtemos a relação de recursividade da matriz φ_n , ou seja:

$$\varphi_n^{-1} = \lambda^{-1}\varphi_{n-1}^{-1} - \frac{\lambda^{-2}\varphi_{n-1}^{-1}\mathbf{u}_n\mathbf{u}_n^T\varphi_{n-1}^{-1}}{1 + \lambda^{-1}\mathbf{u}_n^T\varphi_{n-1}^{-1}\mathbf{u}_n}. \quad (2.23)$$

Por conveniência computacional podemos escrever as seguintes igualdades:

$$\mathbf{P}_n = \varphi_n^{-1} \quad (2.24)$$

e

$$\mathbf{k}_n = \frac{\lambda^{-1}\mathbf{P}_{n-1}\mathbf{u}_n}{1 + \lambda^{-1}\mathbf{u}_n^T\mathbf{P}_{n-1}\mathbf{u}_n}. \quad (2.25)$$

Podemos reescrever a Equação (2.23) como segue:

$$\mathbf{P}_n = \lambda^{-1}\mathbf{P}_{n-1} - \lambda^{-1}\mathbf{k}_n\mathbf{u}_n^T\mathbf{P}_{n-1}, \quad (2.26)$$

sendo \mathbf{P}_n a inversa da matriz de auto-correlação de dimensão $L \times L$ e \mathbf{k}_n o vetor ganho de dimensão $L \times 1$. Reorganizando a Equação (2.25), temos:

$$\begin{aligned} \mathbf{k}_n + \mathbf{k}_n\lambda^{-1}\mathbf{u}_n^T\mathbf{P}_{n-1}\mathbf{u}_n &= \lambda^{-1}\mathbf{P}_{n-1}\mathbf{u}_n, \\ \mathbf{k}_n &= [\lambda^{-1}\mathbf{P}_{n-1} - \mathbf{k}_n\lambda^{-1}\mathbf{u}_n^T\mathbf{P}_{n-1}] \mathbf{u}_n. \end{aligned} \quad (2.27)$$

Substituindo a Equação (2.26) em (2.27) seque-se:

$$\mathbf{k}_n = \mathbf{P}_n \mathbf{u}_n \quad (2.28)$$

ou

$$\mathbf{k}_n = \varphi_n^{-1} \mathbf{u}_n. \quad (2.29)$$

Atualização do vetor peso

No desenvolvimento de uma equação recursiva, utilizaremos as Equações (2.15), (2.19) e (2.24) para expressar a estimativa do mínimo quadrado do vetor peso, $\hat{\mathbf{w}}_n$, no instante n como segue:

$$\begin{aligned} \hat{\mathbf{w}}_n &= \mathbf{P}_n [\lambda \mathbf{z}_{n-1} + \mathbf{u}_n d_n] \\ \hat{\mathbf{w}}_n &= \lambda \mathbf{P}_n \mathbf{z}_{n-1} + \mathbf{P}_n \mathbf{u}_n d_n. \end{aligned} \quad (2.30)$$

Substituindo a Equação (2.26) na Equação (2.30), temos:

$$\hat{\mathbf{w}}_n = \mathbf{P}_{n-1} \mathbf{z}_{n-1} - \mathbf{k}_n \mathbf{u}_n^T \mathbf{P}_{n-1} \mathbf{z}_{n-1} + \mathbf{P}_n \mathbf{u}_n d_n \quad (2.31)$$

Das Equações (2.29) e (2.31), segue-se:

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_{n-1} - \mathbf{k}_n \epsilon_n, \quad (2.32)$$

sendo ϵ_n a estimativa do erro a priori definida por:

$$\epsilon_n = d_n - \mathbf{u}_n^T \hat{\mathbf{w}}_{n-1}. \quad (2.33)$$

2.5.2 Convergência do algoritmo *RLS*

Estudaremos nesta seção a convergência do algoritmo *RLS* no contexto de um problema de modelagem de sistema. Como planta consideramos um regressor múltiplo linear caracterizado pela equação

$$d_n = \mathbf{w}_*^T \mathbf{u}_n + e_n \quad (2.34)$$

sendo \mathbf{w}_* o vetor pesos do regressor, \mathbf{u}_n o vetor entrada, e_n o ruído da planta e d_n a saída da planta. O ruído do processo é branco com média zero e variância σ^2

O comportamento médio do vetor peso no algoritmo *RLS*

De (2.15) e (2.17) obtemos

$$\hat{\mathbf{w}}_n = \varphi_n^{-1} \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i d_i. \quad (2.35)$$

Substituindo (2.34) em (2.35) e usando (2.16), temos

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_* + \varphi_n^{-1} \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i e_i. \quad (2.36)$$

Aplicando o operador esperança em ambos os membros da Equação (2.36) e reconhecendo do princípio de ortogonalidade que todos os elementos do vetor \mathbf{u}_n são ortogonais ao erro e_n , obtemos

$$E[\hat{\mathbf{w}}_n] = \hat{\mathbf{w}}_*. \quad (2.37)$$

Matriz de correlação do vetor desvio

Definimos o vetor de desvio como

$$\mathbf{v}_n = \hat{\mathbf{w}}_n - \mathbf{w}_*. \quad (2.38)$$

De (2.36), temos,

$$\mathbf{v}_n = \varphi_n^{-1} \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i e_i. \quad (2.39)$$

Definimos a matriz de correlação do vetor desvio da seguinte forma,

$$\mathbf{K}_n = E[\mathbf{v}_n \mathbf{v}_n^T]. \quad (2.40)$$

Substituindo (2.39) em (2.40) e notando que $[\varphi_n^{-1}]^T = \varphi_n^{-1}$ e $[\lambda^{n-i}]^T = \lambda^{n-i}$, obtemos,

$$\mathbf{K}_n = E \left[\left(\varphi_n^{-1} \sum_{i=1}^n \lambda^{n-i} \cdot \mathbf{u}_i e_i \right) \left(\sum_{i=1}^n \lambda^{n-i} e_i^T \mathbf{u}_i^T \right) \varphi_n^{-1} \right]. \quad (2.41)$$

Podemos observar em [38](p.427) que para o rigoroso cálculo da Equação (2.41) devemos fazer as seguintes hipóteses:

1. O vetor de entrada \mathbf{u}_i possui amostras de um processo ergótico. Assim, podemos usar as médias do tempo ao invés do conjunto de médias.
2. O fator de esquecimento λ é muito próximo de 1.
3. O tempo n o qual \mathbf{K}_n é calculado é grande.

Notamos, de (2.16) que φ_n é uma soma ponderada dos produtos externos,

$$\mathbf{u}_n \mathbf{u}_n^T, \mathbf{u}_{n-1} \mathbf{u}_{n-1}^T, \mathbf{u}_{n-2} \mathbf{u}_{n-2}^T, \dots$$

Assim, considerando as hipóteses acima, encontramos,

$$\varphi_n \approx \frac{1 - \lambda^n}{1 - \lambda} \mathbf{R}, \quad (2.42)$$

sendo $\mathbf{R} = E[\mathbf{u}_n \mathbf{u}_n^T]$ a matriz de correlação do vetor de entrada.

Substituindo (2.42) em (2.41) e notando que $E[e_n e_n^T] = \sigma^2$, obtemos,

$$\begin{aligned} \mathbf{K}_n &= \sigma^2 \left(\frac{1 - \lambda^n}{1 - \lambda} \right)^2 \cdot \left(\frac{1 - \lambda^{2n}}{1 - \lambda^2} \right) \mathbf{R}^{-1} \\ &= \sigma^2 \left(\frac{1 - \lambda}{1 + \lambda} \right) \cdot \left(\frac{1 + \lambda^n}{1 - \lambda^n} \right) \mathbf{R}^{-1}. \end{aligned} \quad (2.43)$$

Curva de aprendizagem do algoritmo *RLS*

Para algoritmo *RLS*, é conveniente usar o erro ϵ_n para definir o MSE, assim podemos expressar a curva de aprendizagem do algoritmo *RLS* em termos do erro a priori como,

$$\xi_n = E[\epsilon_n^2]. \quad (2.44)$$

De (2.38) e (2.34) podemos escrever o erro a priori da seguinte forma:

$$\epsilon_n = e_n - \mathbf{v}_{n-1}^T \mathbf{u}_n. \quad (2.45)$$

Substituindo (2.45) em (2.44), obtemos,

$$\begin{aligned} \xi_n &= E[e_n^2] + E[\mathbf{u}_n^T \mathbf{v}_{n-1} \mathbf{v}_{n-1}^T \mathbf{u}_n] \\ &\quad - E[\mathbf{v}_{n-1}^T \mathbf{u}_n e_n] - E[e_n \mathbf{u}_n^T \mathbf{v}_{n-1}]. \end{aligned} \quad (2.46)$$

O primeiro valor esperado do lado direito da Equação (2.46) é simplesmente a variância de e_n . Para os demais valores esperados, podemos fazer as seguintes observações:

1. A estimativa $\hat{\mathbf{w}}_{n-1}$, e portanto o vetor desvio \mathbf{v}_{n-1} , é independente do vetor de entrada \mathbf{u}_n ; o último é assumido como sendo derivado de um processo estacionário no sentido amplo de média zero. Conseqüentemente, podemos usar esta independência estatística junto com os resultados conhecidos de álgebra matricial para expressar o segundo valor esperado do lado direito da Equação (2.46), como se segue,

$$\begin{aligned} E[\mathbf{u}_n^T \mathbf{v}_{n-1} \mathbf{v}_{n-1}^T \mathbf{u}_n] &= E[\text{tr}\{\mathbf{u}_n^T \mathbf{v}_{n-1} \mathbf{v}_{n-1}^T \mathbf{u}_n\}] \\ &= E[\text{tr}\{\mathbf{u}_n^T \mathbf{u}_n \mathbf{v}_{n-1} \mathbf{v}_{n-1}^T\}] \\ &= \text{tr}\{E[\mathbf{u}_n^T \mathbf{u}_n] E[\mathbf{v}_{n-1} \mathbf{v}_{n-1}^T]\} \\ &= \text{tr}\{\mathbf{R}\mathbf{K}_{n-1}\}, \end{aligned} \quad (2.47)$$

sendo que na última linha utilizamos as definições de médias da matriz de correlação $\mathbf{R} = E[\mathbf{u}_n \mathbf{u}_n^T]$ e da matriz de correlação do vetor desvio $\mathbf{K}_n = E[\mathbf{v}_n \mathbf{v}_n^T]$.

2. O erro de medição e_n depende do vetor de entrada \mathbf{u}_n ; isto segue de uma simples manipulação da Equação (2.34). O vetor desvio \mathbf{v}_{n-1}^T é portanto independente de \mathbf{u}_n e e_n . Entretanto, podemos mostrar que o terceiro valor esperado do lado direito da Equação (2.46) é zero. Assim,

$$E[\mathbf{v}_{n-1}^T \mathbf{u}_n e_n] = E[\mathbf{v}_{n-1}^T] \cdot E[\mathbf{u}_n e_n]. \quad (2.48)$$

Reconhecemos do princípio de ortogonalidade que todos os elementos do vetor \mathbf{u}_n são ortogonais ao erro de medição e_n , isto é,

$$E[\mathbf{v}_{n-1}^T \mathbf{u}_n e_n] = 0. \quad (2.49)$$

3. O quarto valor esperado do lado direito da Equação (2.46) tem a mesma forma matemática considerada no item 2. Entretanto podemos dizer que este valor esperado é igual a zero,

$$E[e_n \mathbf{u}_n^T \mathbf{v}_{n-1}] = 0. \quad (2.50)$$

Assim, reconhecendo que $E[e_n^2] = \xi_{min}$, e usando os resultados das Equações (2.47) até (2.50) em (2.46), obtemos a seguinte fórmula para o erro quadrático médio no algoritmo *RLS*:

$$\xi_n = \xi_{min} + tr\{\mathbf{R}\mathbf{K}_{n-1}\}. \quad (2.51)$$

sendo ξ_{min} o mínimo *MSE* do filtro encontrado quando uma estimativa perfeita de \mathbf{w}_* é calculada. Substituindo (2.43) em 2.51 obtemos

$$\xi_n = \xi_{min} + \left(\frac{1-\lambda}{1+\lambda}\right) \cdot \left(\frac{1+\lambda^{n-1}}{1-\lambda^{n-1}}\right) \cdot L\xi_{min}. \quad (2.52)$$

Este resultado descreve a curva de aprendizagem do algoritmo *RLS*.

Tempo de aprendizagem

Façamos, agora, uma análise do comportamento do algoritmo *RLS*. O segundo termo do lado direito da Equação (2.52) é um valor positivo que indica o desvio de ξ_n de ξ_{min} . Notamos também que a velocidade com que estes termos convergem é determinada pelo termo exponencial λ^{n-1} , ou equivalentemente λ^n . Desta forma, definimos o tempo de aprendizagem $\tau_{\mathcal{RLS}}$ associado com o algoritmo *RLS* usando a seguinte equação,

$$\lambda^n = e^{\frac{-n}{\tau_{\mathcal{RLS}}}}. \quad (2.53)$$

Resolvendo (2.53) para $\tau_{\mathcal{RLS}}$ obtemos,

$$\tau_{\mathcal{RLS}} = -\frac{1}{\ln \lambda}. \quad (2.54)$$

Para simplificar (2.54) usaremos a seguinte aproximação,

$$\ln \lambda = \ln(1 - (1 - \lambda)) \approx -(1 - \lambda). \quad (2.55)$$

Substituindo (2.55) em (2.54) temos,

$$\tau_{\mathcal{RLS}} \approx \frac{1}{(1 - \lambda)}. \quad (2.56)$$

Deste resultado, notamos que o comportamento da convergência do algoritmo *RLS* é independente dos autovalores da matriz de correlação das entradas.

Excesso do erro quadrático médio e desajuste

Sabendo que o erro quadrático médio (ξ_n) é maior que o erro quadrático médio mínimo (ξ_{min}) temos, então, um excesso no erro final e podemos definir o excesso do erro quadrático médio, *ExcessoMSE*, como a diferença entre o erro quadrático médio atual e o erro quadrático médio mínimo [38].

Usando 2.52 obtemos,

$$ExcessoMSE = \left(\frac{1 - \lambda}{1 + \lambda} \right) \cdot L\xi_{min}. \quad (2.57)$$

Definimos também o ExcessoMSE normalizado pelo erro quadrático médio mínimo, como o desajuste $\mathcal{M}_{\mathcal{RLS}}$.

$$\mathcal{M}_{\mathcal{RLS}} = \frac{ExcessoMSE}{\xi_{min}}. \quad (2.58)$$

Substituído 2.57 em 2.58 temos

$$\mathcal{M}_{\mathcal{RLS}} = \left(\frac{1 - \lambda}{1 + \lambda} \right) \cdot L, \quad (2.59)$$

sendo L a ordem do filtro.

2.6 Conclusão

Neste capítulo, realizamos uma revisão da superfície quadrática gerada quando se utiliza o erro quadrático médio como critério aplicado sobre o erro em um filtro adaptativo. Mostramos a dedução do algoritmo *LMS*. Realizamos também, a dedução do algoritmo *RLS* e descrevemos as equações que determinam sua condição de convergência. O tempo de aprendizagem e o desajuste também foram enfatizados, pois os mesmos são utilizados como referência comparativa de outros algoritmos adaptativos.

Capítulo 3

Algoritmo RNQ baseado em uma única potência par do erro

3.1 Introdução

Neste capítulo desenvolveremos um algoritmo adaptativo baseado em uma potência par do erro, inspirado no algoritmo RLS padrão. Chamaremos esse novo algoritmo de **recursivo não-quadrático** (RNQ), para o qual escolhemos uma função baseada em uma única potência par do erro como critério a ser minimizado. O nosso objetivo é determinar um algoritmo que ajuste os pesos do combinador linear adaptativo de forma tal a minimizar esta função. Mostraremos, também, que a superfície de desempenho obtida por este critério oferece maior velocidade de convergência, bem como um menor desajuste na busca do peso ótimo quando comparado com o algoritmo RLS.

3.2 Dedução do algoritmo

A estrutura básica de um filtro adaptativo é composta por um sinal desejado d_i , um vetor de entrada $\mathbf{u}_i = [u_i, u_{i-1}, \dots, u_{i-L+1}]^T$, e o erro e_i , usado para atualizar o vetor peso $\mathbf{w}_n = [w_{0,n}, w_{1,n}, \dots, w_{L-1,n}]^T$. Devemos recuperar d_i estimando o sinal de saída $y_i = \mathbf{w}_n^T \mathbf{u}_i$, depois de calcular o erro $e_i = d_i - y_i$, sendo n o número de iteração, $1 \leq i \leq n$ e L a ordem do filtro.

Para desenvolver o algoritmo RNQ baseado em uma potência par do erro, utilizamos como função de custo uma função par, contínua e simétrica, representada por

$$J_n = \sum_{i=1}^n \left\{ \lambda^{n-i} [e_i]^{2j} \right\}, \quad (3.1)$$

sendo j e n inteiros positivos, com $j > 1$ e $0 \ll \lambda < 1$ fator peso exponencial ou fator de esquecimento. Podemos observar que para $j = 1$ obtemos o mesmo critério (função de custo) utilizado no algoritmo RLS padrão, deduzido no capítulo anterior.

Objetivando encontrar o peso ótimo, devemos calcular o gradiente da função J_n . Assim, derivando a Equação (3.1) em relação a \mathbf{w} , obtemos,

$$\begin{aligned} \nabla J_n &= - \sum_{i=1}^n \left\{ \lambda^{n-i} \mathbf{u}_i 2j [e_i]^{2j-1} \right\} \\ &= - \sum_{i=1}^n \left\{ \lambda^{n-i} \mathbf{u}_i 2j [d_i - \mathbf{w}_n^T \mathbf{u}_i]^{2j-1} \right\}. \end{aligned} \quad (3.2)$$

Desenvolvendo o binômio $[d_i - \mathbf{w}_n^T \mathbf{u}_i]^{2j-1}$, e negligenciando as potências de alta ordem, podemos reescrever a Equação (3.2) como,

$$\begin{aligned} \nabla J_n &\approx - \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i 2j \left\{ d_i^{2j-1} - (2j-1) d_i^{2j-2} (\mathbf{w}_n^T \mathbf{u}_i) \right\} \\ &\approx -2j \left[\sum_{i=1}^n [\lambda^{n-i} d_i^{2j-1} \mathbf{u}_i] - (2j-1) \sum_{i=1}^n [\lambda^{n-i} d_i^{2j-2} \mathbf{u}_i \mathbf{u}_i^T] \mathbf{w}_n \right], \end{aligned} \quad (3.3)$$

sendo o vetor de correlação cruzada, \mathbf{z}_n , entre as entradas e a resposta desejada de dimensão $L \times 1$, definido pelo primeiro termo do lado direito da Equação (3.3), isto é,

$$\mathbf{z}_n = \sum_{i=1}^n [\lambda^{n-i} d_i^{2j-1} \cdot \mathbf{u}_i]. \quad (3.4)$$

E a matriz de correlação do vetor de entrada de dimensão $L \times L$ definida pelo segundo termo da Equação (3.3),

$$\Phi_n = (2j-1) \sum_{i=1}^n [\lambda^{n-i} d_i^{2j-2} \mathbf{u}_i \mathbf{u}_i^T]. \quad (3.5)$$

Dessa forma, podemos reescrever a Equação (3.3) como,

$$\nabla J_n \approx -2j [\mathbf{z}_n - \Phi_n \cdot \mathbf{w}_n]. \quad (3.6)$$

Assim, o valor ótimo do vetor peso, $\hat{\mathbf{w}}_n$, para que a função da Equação (3.1) atinja o valor mínimo é definido pela equação normal escrita em forma matricial

$$\hat{\mathbf{w}}_n = \Phi_n^{-1} \mathbf{z}_n. \quad (3.7)$$

Agora, para encontrarmos uma regra de atualização para o peso \mathbf{w} , faremos o seguinte desenvolvimento:

Primeiramente, isolando o termo correspondente a $i = n$ da Equação (3.5), podemos escrever

$$\Phi_n = \lambda \Phi_{n-1} + [(2j - 1) \cdot d_n^{2j-2}] \mathbf{u}_n \cdot \mathbf{u}_n^T. \quad (3.8)$$

Analogamente podemos reescrever a Equação (3.4) como,

$$\mathbf{z}_n = \lambda \mathbf{z}_{n-1} + [d_n^{2j-1}] \mathbf{u}_n. \quad (3.9)$$

A dedução das Equações (3.8) e (3.9) foi feita com o intuito de facilitar o cálculo da inversa da matriz de correlação Φ_n , que é necessário de acordo com a Equação (3.7).

Considerando a matriz de correlação Φ_n positivamente definida e não singular, podemos aplicar o lema de inversão de matrizes para equação recursiva 3.5, onde faremos as seguintes identificações,

$$\begin{aligned} \mathbf{A} &= \Phi_n \\ \mathbf{B}^{-1} &= \lambda \Phi_{n-1} \\ \mathbf{C} &= \mathbf{u}_n \\ \mathbf{D}^{-1} &= (2j - 1) d_n^{2j-2}. \end{aligned} \quad (3.10)$$

Substituindo as definições 3.10 na Equação 2.21 do lema de inversão de matrizes, obtemos a relação de recursividade da matriz Φ_n , ou seja,

$$\Phi_n^{-1} = \lambda^{-1}\Phi_{n-1}^{-1} - \frac{\lambda^{-1}\Phi_{n-1}^{-1}\mathbf{u}_n\mathbf{u}_n^T\lambda^{-1}\Phi_{n-1}^{-1}}{((2j-1)d_n^{2j-2})^{-1} + \lambda^{-1}\mathbf{u}_n^T\Phi_{n-1}^{-1}\mathbf{u}_n}. \quad (3.11)$$

Por conveniência computacional podemos escrever as seguintes igualdades,

$$\mathbf{P}_n = \Phi_n^{-1} \quad (3.12)$$

e

$$\mathbf{g}_n = \frac{\lambda^{-1}\mathbf{P}_{n-1}\mathbf{u}_n}{((2j-1)d_n^{2j-2})^{-1} + \lambda^{-1}\mathbf{u}_n^T\mathbf{P}_{n-1}\mathbf{u}_n}. \quad (3.13)$$

Assim, podemos reescrever a Equação 3.11 como,

$$\mathbf{P}_n = \lambda^{-1}\mathbf{P}_{n-1} - \lambda^{-1}\mathbf{g}_n\mathbf{u}_n^T\mathbf{P}_{n-1}, \quad (3.14)$$

sendo \mathbf{P}_n a inversa da matriz de auto-correlação de dimensão $L \times L$ e \mathbf{g}_n o vetor ganho de dimensão $L \times 1$.

Para facilitar futuras manipulações matemáticas, nós reorganizamos a Equação (3.13) como,

$$((2j-1)d_n^{2j-2})^{-1}\mathbf{g}_n = \mathbf{P}_n\mathbf{u}_n. \quad (3.15)$$

3.2.1 Atualização do vetor peso

No desenvolvimento de uma equação recursiva utilizaremos as Equações 3.7, 3.9 e 3.12 para determinar a regra de atualização do vetor peso \mathbf{w}_n . Assim,

$$\begin{aligned} \mathbf{w}_n &= \Phi_n^{-1}\mathbf{z}_n \\ &= \lambda\mathbf{P}_n\mathbf{z}_{n-1} + d_n^{2j-1}\mathbf{P}_n\mathbf{u}_n. \end{aligned} \quad (3.16)$$

Substituindo a Equação 3.14 no primeiro termo do lado direito da Equação (3.16) obtemos,

$$\begin{aligned}
\mathbf{w}_n &= \mathbf{P}_{n-1}\mathbf{z}_{n-1} - \mathbf{g}_n \mathbf{u}_n^T \mathbf{P}_{n-1}\mathbf{z}_{n-1} + d_n^{2j-1} \mathbf{P}_n \mathbf{u}_n \\
&= \mathbf{w}_{n-1} - \mathbf{g}_n^T \mathbf{u}_n^T \mathbf{w}_{n-1} + d_n^{2j-1} \mathbf{P}_n \mathbf{u}_n.
\end{aligned} \tag{3.17}$$

Usando o fato de que $\mathbf{P}_n \mathbf{u}_n$ é uma fatoração de \mathbf{g}_n , como mostrado na Equação (3.15), obtemos a seguinte equação,

$$\begin{aligned}
\mathbf{w}_n &= \mathbf{w}_{n-1} - \mathbf{g}_n^T \mathbf{u}_n^T \mathbf{w}_{n-1} + \frac{d_n}{(2j-1)} \mathbf{g}_n \\
&= \mathbf{w}_{n-1} + \mathbf{g}_n \left[\frac{d_n}{(2j-1)} - \mathbf{u}_n^T \mathbf{w}_{n-1} \right].
\end{aligned} \tag{3.18}$$

Podemos ver que o termo entre cochetes na Equação (3.18) é a *estimação a priori do erro*, o qual denotamos por,

$$\epsilon_n = \frac{d_n}{(2j-1)} - \mathbf{u}_n^T \hat{\mathbf{w}}_{n-1} \tag{3.19}$$

Assim, a estrutura de atualização do algoritmo proposto é dada por,

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_{n-1} + \mathbf{g}_n \cdot \epsilon_n. \tag{3.20}$$

3.2.2 Resumo do algoritmo

Na Tabela 3.1 podemos observar o resumo do algoritmo proposto.

Tabela 3.1: Resumo do algoritmo RNQ para uma potência par do erro

Inicializar o algoritmo, definindo
 $\mathbf{P}_0 = \delta^{-1} \mathbf{I}$, δ é uma constante positiva pequena
 $\hat{\mathbf{w}}_0 = \mathbf{0}$
Para cada instante de tempo, $n = 1, 2, \dots$, computar

$$\begin{aligned}
\mathbf{g}_n &= \frac{\lambda^{-1} \mathbf{P}_{n-1} \mathbf{u}_n}{\left[(2j-1) d_n^{2j-2} \right]^{-1} + \lambda^{-1} \mathbf{u}_n^T \mathbf{P}_{n-1} \mathbf{u}_n} \\
\epsilon_n &= \frac{d_n}{(2j-1)} - \hat{\mathbf{w}}_{n-1}^T \mathbf{u}_n \\
\hat{\mathbf{w}}_n &= \hat{\mathbf{w}}_{n-1} + \mathbf{g}_n \epsilon_n \\
\mathbf{P}_n &= \lambda^{-1} \mathbf{P}_{n-1} - \lambda^{-1} \mathbf{g}_n \mathbf{u}_n^T \mathbf{P}_{n-1}
\end{aligned}$$

3.3 Convergência do algoritmo

Estudaremos nesta seção a convergência do algoritmo RNQ para uma potência par do erro no contexto de um problema de identificação de sistemas adaptativos, da mesma forma que estudamos a convergência do algoritmo *RLS* no capítulo anterior. A primeira análise será encontrar as condições de convergência do algoritmo e descrever como ele se comporta até atingir o estado estacionário. Isso pode ser realizado através do estudo do comportamento dos pesos. Em seguida analisaremos o desajuste e o tempo de aprendizagem do algoritmo proposto.

Como planta, consideramos um regressor múltiplo linear caracterizado pela equação,

$$d_n = e_n + \mathbf{w}_*^T \mathbf{u}_n, \quad (3.21)$$

sendo \mathbf{w}_* o vetor peso do regressor, \mathbf{u}_n o vetor entrada, e_n é o ruído da planta e d_n a saída da planta.

3.3.1 O comportamento médio do vetor peso no algoritmo

De (3.4), (3.7) e (3.21) obtemos,

$$\hat{\mathbf{w}}_n = \Phi_n^{-1} \cdot \left\{ \sum_{i=1}^n \left[\lambda^{n-i} \cdot \mathbf{u}_i \cdot (\mathbf{w}_*^T \mathbf{u}_i + e_i)^{2j-1} \right] \right\}. \quad (3.22)$$

Utilizando desenvolvimento binomial do termo $(\mathbf{w}_*^T \mathbf{u}_i + e_i)^{2j-1}$, e desprezando os termos de alta potência (assumindo esses termos próximo de zero), podemos reescrever a Equação (3.22) como,

$$\hat{\mathbf{w}}_n \cong \mathbf{w}_* + \Phi_n^{-1} \cdot \sum_{i=1}^n \left[\lambda^{n-i} \cdot \mathbf{u}_i \cdot e_i^{2j-1} \right]. \quad (3.23)$$

Aplicando o operador esperança em ambos os membros da Equação (3.23) e lembrando que \mathbf{u}_i e e_i são independentes para todo i , essa e que e_i tem média zero, obtemos,

$$E[\hat{\mathbf{w}}_n] = \mathbf{w}_*. \quad (3.24)$$

Desse resultado temos que $\hat{\mathbf{w}}_n$ é uma estimação não enviesada de \mathbf{w}_* . Em outras palavras, podemos afirmar que o algoritmo proposto converge em torno da média.

3.3.2 Matriz de correlação do vetor desvio

Para descrever a curva de aprendizagem do algoritmo nós estudamos a matriz de correlação do vetor desvio. Assim, iremos fazer uma mudança de variável para definir o vetor desvio como,

$$\mathbf{v}_n = \hat{\mathbf{w}}_n - \mathbf{w}_*, \quad (3.25)$$

sendo \mathbf{w}_* a solução ótima.

De (3.23) e (3.25), obtemos,

$$\mathbf{v}_n = \Phi_n^{-1} \cdot \sum_{i=1}^n [\lambda^{n-i} \cdot \mathbf{u}_i \cdot e_i^{2j-1}]. \quad (3.26)$$

Agora, definimos a matriz de correlação do vetor desvio como,

$$\mathbf{K}_n = E[\mathbf{v}_n \mathbf{v}_n^T]. \quad (3.27)$$

Substituindo (3.26) em (3.27) e notando que $(\Phi_n^{-1})^T = \Phi_n^{-1}$, sabendo que o erro de medição neste processo é um ruído branco com variância σ^2 e que o vetor de entrada e o vetor desvio são independentes, essa última afirmação tem sido uma prática comum na análise de algoritmos em filtragem adaptativa [39, 40, 41, 42, 43, 44, 45, 46, 47, 48], obtemos,

$$\mathbf{K}_n = \sigma^2 \cdot E[\Phi_n^{-1} (\mathbf{U}_n \mathbf{\Lambda}_n \mathbf{\Lambda}_n \mathbf{U}_n^T) \Phi_n^{-1}], \quad (3.28)$$

sendo $\mathbf{\Lambda}_n$ uma matriz diagonal formadas pelos fatores exponenciais

$\lambda^{n-1}, \lambda^{n-2}, \dots$, e \mathbf{U}_n é a matriz dos dados de entrada, isto é:

$$\mathbf{U}_n = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \dots \quad \mathbf{u}_n]. \quad (3.29)$$

Para o cálculo rigoroso da Equação (3.28) devemos fazer as seguintes hipóteses [38](p.427):

1. O vetor de entrada \mathbf{u}_i constitui amostras de um processo ergódico. Assim, podemos usar as médias do tempo ao invés do conjunto de médias.
2. O fator de esquecimento λ é muito próximo de 1.
3. O tempo n o qual \mathbf{K}_n é calculado é grande.

Podemos notar da Equação (3.5) que Φ é uma soma ponderada dos produtos internos $[\mathbf{u}_n \cdot \mathbf{u}_n^T, \mathbf{u}_{n-1} \cdot \mathbf{u}_{n-1}^T, \dots]$. Assim, considerando as hipóteses acima, temos que,

$$\Phi_n \approx (2j - 1) d_n^{2j-2} \frac{1 - \lambda^n}{1 - \lambda} \mathbf{R}, \quad (3.30)$$

sendo $\mathbf{R} = E[\mathbf{u}_n \mathbf{u}_n^T]$ a matriz de correlação de entrada.

Analogamente, obtemos,

$$\mathbf{U}_n \Lambda_n \Lambda_n \mathbf{U}_n^T \approx \frac{1 - \lambda^{2n}}{1 - \lambda^2} \mathbf{R}, \quad (3.31)$$

Substituindo (3.31) e (3.30) em (3.28) temos,

$$\mathbf{K}_n = \sigma^2 \cdot \left\{ \frac{1}{[(2j - 1) d_n^{2j-2}]^2} \cdot \frac{1 - \lambda}{1 + \lambda} \cdot \frac{1 + \lambda^n}{1 - \lambda^n} \mathbf{R}^{-1} \right\}. \quad (3.32)$$

No estado estacionário, isto é, quando $n \rightarrow \infty$, e usando a Equação (3.32), obtemos,

$$\mathbf{K}_\infty = \sigma^2 \cdot \left\{ \frac{1}{[(2j - 1) d^{2j-2}]^2} \cdot \frac{1 - \lambda}{1 + \lambda} \cdot \mathbf{R}^{-1} \right\}. \quad (3.33)$$

3.3.3 Curva de aprendizagem

De (3.19), (3.21) e (3.25) obtemos

$$\epsilon_n = \frac{1}{(2j - 1)} \cdot (e_n - \mathbf{v}_{n-1}^T \mathbf{u}_n), \quad (3.34)$$

sendo \mathbf{v}_{n-1} o vetor desvio no tempo $n - 1$.

Como índice de desempenho estatístico para o algoritmo proposto, é conveniente usar um erro de estimativa a priori ϵ_n para definir o valor esperado do erro de grau $2j$,

$$\xi_n = E \left[|\epsilon_n|^{2j} \right], \quad (3.35)$$

sendo j um inteiro positivo. Substituindo (3.34) em (3.35), obtemos,

$$\xi_n = \frac{1}{(2j-1)^{2j}} \cdot E \left[\left| -\mathbf{v}_{n-1}^T \mathbf{u}_n + e_n \right|^{2j} \right]. \quad (3.36)$$

Utilizando desenvolvimento binomial e notando que \mathbf{v}_{n-1} é próximo de zero, após o estado transitório, podemos desconsiderar alguns termos do lado direito da Equação (3.36) que incluem potências altas de \mathbf{v}_{n-1} . Entretanto, supondo que a distribuição de densidade do sinal erro, e_n , é simétrica e que $E \left[|e_n|^{2j} \right]$ é o $2j$ -ésima potência de e_n , temos,

$$\begin{aligned} \xi_n &\approx \frac{1}{(2j-1)^{2j}} \cdot E \left[e_n^{2j} \right] + \frac{1}{(2j-1)^{2j}} \cdot j \cdot (2j-1) \cdot E \left[e_n^{2j-2} \right] \text{tr} \{ \mathbf{R} \mathbf{K}_{n-1} \} \\ &+ \frac{1}{(2j-1)^{2j}} \cdot \text{tr} \{ (\mathbf{R} \mathbf{K}_{n-1})^j \}. \end{aligned} \quad (3.37)$$

Substituindo (3.32) em (3.37) e definindo $\xi_{min} = \frac{1}{(2j-1)^{2j}} \cdot E \left[|e_n|^{2j} \right]$ como o mínimo de e_n , quando $\mathbf{w} = \mathbf{w}_*$, obtemos a seguinte expressão para o algoritmo proposto,

$$\begin{aligned} \xi_n &\approx \xi_{min} + \frac{j}{(2j-1)^{2j+1} \cdot d_n^{4j-4}} \cdot \frac{1-\lambda}{1+\lambda} \cdot \frac{1+\lambda^{n-1}}{1-\lambda^{n-1}} \cdot L \cdot \sigma^2 \cdot E \left[|e_n|^{2j-2} \right] \\ &+ \frac{1}{(2j-1)^{4j} \cdot d_n^{4j-4}} \cdot \frac{(1-\lambda)^j}{(1+\lambda)^j} \cdot \frac{(1+\lambda^{n-1})^j}{(1-\lambda^{n-1})^j} \cdot L \cdot (\sigma^2)^j, \end{aligned} \quad (3.38)$$

sendo L a ordem do filtro.

A Equação (3.38) descreve a curva de aprendizagem do algoritmo RNQ para uma potência par do erro.

3.3.4 Análise do tempo de aprendizagem

Analogamente à análise feita na dedução da constante de tempo do RLS, no capítulo anterior, podemos verificar que a velocidade na qual o segundo e o terceiro termo do lado direito da Equação 3.38 convergem é determinada pelos termos exponenciais λ^n e λ^{nj} , respectivamente. Assim, podemos obter o tempo de convergência associado ao algoritmo proposto definido por $\tau_{\mathcal{RNQ}_1}$ e $\tau_{\mathcal{RNQ}_2}$, da seguinte forma: primeiro, para o segundo termo no lado direito Equação (3.38) temos,

$$e^{\frac{-n}{\tau_{\mathcal{RNQ}_1}}} = \lambda^n, \quad (3.39)$$

que nos dá,

$$\tau_{\mathcal{RNQ}_1} = \frac{-1}{\ln \lambda} = \tau_{\mathcal{RLS}}. \quad (3.40)$$

Analogamente obtemos para o último termo no lado direito da Equação (3.38),

$$e^{\frac{-n}{\tau_{\mathcal{RNQ}_2}}} = \lambda^{nj}, \quad (3.41)$$

que nos dá,

$$\tau_{\mathcal{RNQ}_2} = \frac{-1}{j \cdot \ln \lambda}. \quad (3.42)$$

3.3.5 Desajuste

O desajuste, $\mathcal{M}_{\mathcal{RNQ}}$, é definido como uma diferença dimensional entre o erro quadrático médio (*MSE*), e o *MSE* mínimo. Em outras palavras, este é uma medida normalizada do custo de adaptação [28].

O desajuste do algoritmo RLS é dado por [38],

$$\mathcal{M}_{\mathcal{RLS}} = \left(\frac{1 - \lambda}{1 + \lambda} \right) \cdot L. \quad (3.43)$$

Para o algoritmo proposto, encontramos,

$$\mathcal{M}_{\mathcal{RNQ}} = \frac{\lim_{n \rightarrow \infty} (\xi_n - \xi_{min})}{\xi_{min}} \cong \left(\frac{1 - \lambda}{1 + \lambda} \right) \frac{j}{(2j - 1)^{2j+1} \cdot d_n^{4j-4}} L. \quad (3.44)$$

3.3.6 Análise comparativa

Nesta seção faremos uma análise comparativa da velocidade de convergência e dos desajustes associados aos algoritmos RLS e RNQ para uma potência par do erro.

A curva de aprendizagem do algoritmo RLS é dada por [1],

$$\xi_{nRLS} = \sigma^2 + \frac{1 - \lambda}{1 + \lambda} \cdot \frac{1 + \lambda^{n-1}}{1 - \lambda^{n-1}} \cdot L \cdot (\sigma^2). \quad (3.45)$$

Podemos observar da Equação (3.45) que o erro mínimo do algoritmo RLS é igual a σ^2 e que o estado transitório é governado pelo segundo termo da Equação (3.45).

Por outro lado, da Equação (3.38), observamos que a curva de aprendizagem do algoritmo proposto é composta por três estágios durante o aprendizado. O estado estacionário é dado por ξ_{min} . Os outros dois estão associados a diferentes estatísticas do erro. Um está associado com σ^{2j} e o outro está associado com $|e_n|^{2j-2}$. A soma desses dois termos tende para zero mais rápido que o segundo termo na Equação (3.45) quando n tende para o infinito.

Para comparar a velocidade de adaptação dos algoritmos dividimos $\tau_{\mathcal{RLS}}$ por $\tau_{\mathcal{RNQ}_2}$,

$$\frac{\tau_{\mathcal{RLS}}}{\tau_{\mathcal{RNQ}_2}} = j, \quad (3.46)$$

sendo j um inteiro positivo.

Da Equação (3.46), podemos observar que $\tau_{\mathcal{RNQ}_2} = \tau_{\mathcal{RLS}}$ for $j = 1$ e, $\tau_{\mathcal{RNQ}_2} < \tau_{\mathcal{RLS}}$ for $j > 1$.

Analisando as Equações (3.40) e (3.42), podemos observar claramente que a velocidade de convergência do algoritmo RNQ para uma potência par do erro é mais rápida que a do algoritmo RLS padrão.

Das Equações (3.43) e (3.44), temos que o desajuste $\mathcal{M}_{\mathcal{RNQ}} = \mathcal{M}_{\mathcal{RLS}}$ para $j = 1$ e $\mathcal{M}_{\mathcal{RNQ}} < \mathcal{M}_{\mathcal{RLS}}$, para $j > 1$.

3.3.7 Complexidade computacional

Podemos extrair da Equação (3.20) a complexidade computacional do algoritmo proposto. O algoritmo RNQ para uma potência par do erro requer duas divisões, $L^2 + 5L + 2j$ multiplicações, $L^2 + 3L$ adições por iteração, para um filtro de ordem L . Assim, esse algoritmo requer $2j - 1$ multiplicações e uma divisão escalar extras por iteração quando comparado com algoritmo RLS padrão [49]. A complexidade computacional é da ordem de $O(N^2)$ e é comparável com a do algoritmo RLS.

3.4 Resultados

Objetivando verificar a exatidão das equações deduzidas neste capítulo, fizemos simulações para estimar os coeficientes de um filtro FIR e comparamos os desempenhos dos algoritmos RLS e RNQ baseado em uma potência par do erro.

As simulações foram realizadas em dois tipos de sistemas: um sem ruído e outro com sinal ruidoso.

Para o sistema sem ruído, aplicamos o algoritmo proposto em uma estrutura de identificação de sistema, onde usamos um filtro FIR com resposta ao impulso h e ordem $L = 15$, como podemos ver na Figura 3.1. O sinal de entrada, u_i , é um sinal branco uniformemente distribuído, limitado no intervalo $[-1, 1]$. Filtramos este sinal u_i por h e usamos este sinal como sinal desejado d_i . O sinal de saída do filtro é denotado por y_i . O sinal erro é definido como a diferença entre d_i e y_i . Esse erro é utilizado para ajustar os pesos do filtro adaptativo usando o algoritmo dado em (3.20).

Para cada $j = 1, 2$ e 3 fizemos 100 simulações de Monte Carlo. Utilizamos o fator de esquecimento, $\lambda = 0,95$ para ambos os algoritmos. As curvas de aprendizagem resultantes, para o sistema sem ruído, são mostradas na Figura 3.2a.

No caso do sistema ruidoso, nós usamos a estrutura do sistema dado na primeira simulação, mas depois da filtragem do sinal de entrada nós adicionamos um sinal ruído n_i . Fizemos simulações para três diferentes distribuições de sinais ruidosos:

distribuições uniforme, gaussiana e laplaciana. Todos os sinais ruidosos foram gerados com média zero e variância unitária. O fator de esquecimento $\lambda = 0,95$, foi aplicado para ambos os algoritmos.

Nas Figuras 3.2b, 3.2c e 3.2d podemos visualizar o desempenho do algoritmo RNQ baseado em uma potência par do erro para diferentes valores de j ($j = 2$ e $j = 3$) e o desempenho do algoritmo RLS ($j = 1$), através das curvas de aprendizagem correspondentes para cada simulação. Na Figura 3.2b temos o resultado com o ruído distribuído uniformemente, enquanto que nas Figuras 3.2c e 3.2d temos o resultado para os ruídos com distribuição gaussiana e laplaciana respectivamente.

Na tabela 3.2 podemos ver a melhora, em decibéis, do desempenho em termos de ξ_{mim} do algoritmo proposto em relação ao RLS padrão, para três diferentes interferências de ruídos.

Para reforçar nossos resultados, mostramos na Figura 3.3 o decaimento exponencial, na qual podemos ver os três estágios do tempo de convergência como descrito na subseção da análise do tempo de aprendizagem desse capítulo.

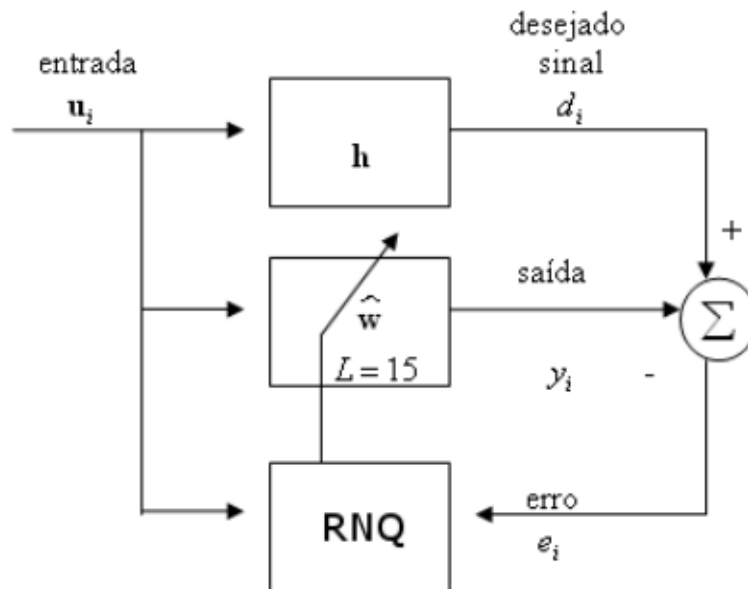


Figura 3.1: Estrutura de um filtro adaptativo no contexto de identificação de sistema. \mathbf{u} é o vetor de entrada, h é o filtro FIR, \mathbf{w} é o vetor peso, e é o sinal erro, d é o sinal desejado, e y é a saída do filtro

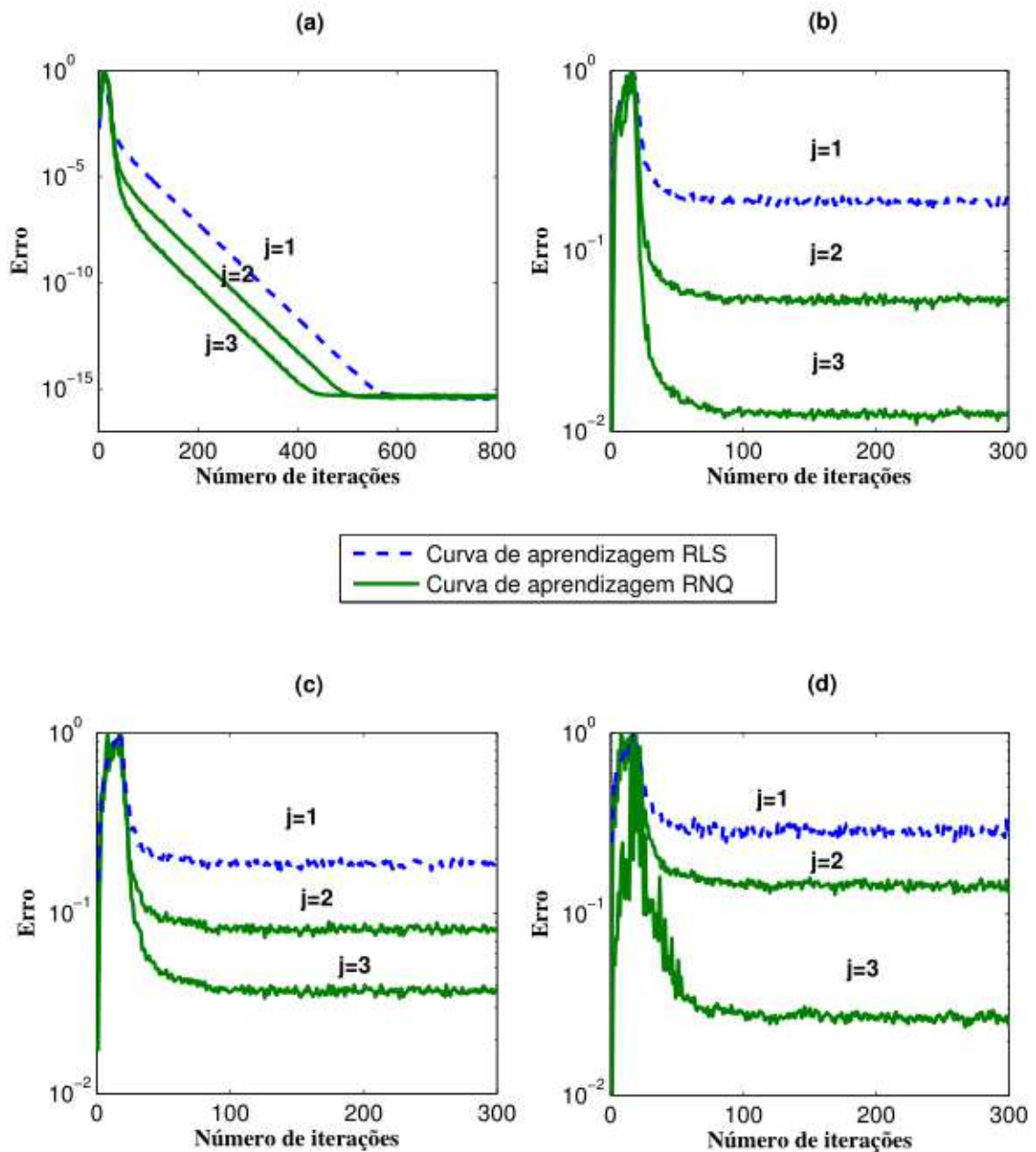


Figura 3.2: Curvas de aprendizagem. As simulações foram realizadas com e sem ruídos. (a) Sem ruído. (b), (c) e (d) adicionando ruídos com distribuição uniforme, gaussiana e laplaciana, respectivamente. Foi utilizado $\lambda = 0,95$ para todas as simulações, e utilizamos também três valores para j ($j = 1, 2$ e 3).

3.5 Discussões e conclusões

Nesse trabalho nós propomos uma função de custo baseada na ponderação exponencial da potência par do erro, inspirado na função de densidade de

Tabela 3.2: Melhora no desempenho do algoritmo RNQ para uma potência par do erro em termos de ξ_{min} em relação ao RLS padrão para três diferentes interferências de ruídos.

| Ruído | j=2 | j=3 |
|------------|-------|-------|
| Uniforme | 11 dB | 22 dB |
| Gaussiana | 8 dB | 19 dB |
| Laplaciana | 4 dB | 18 dB |

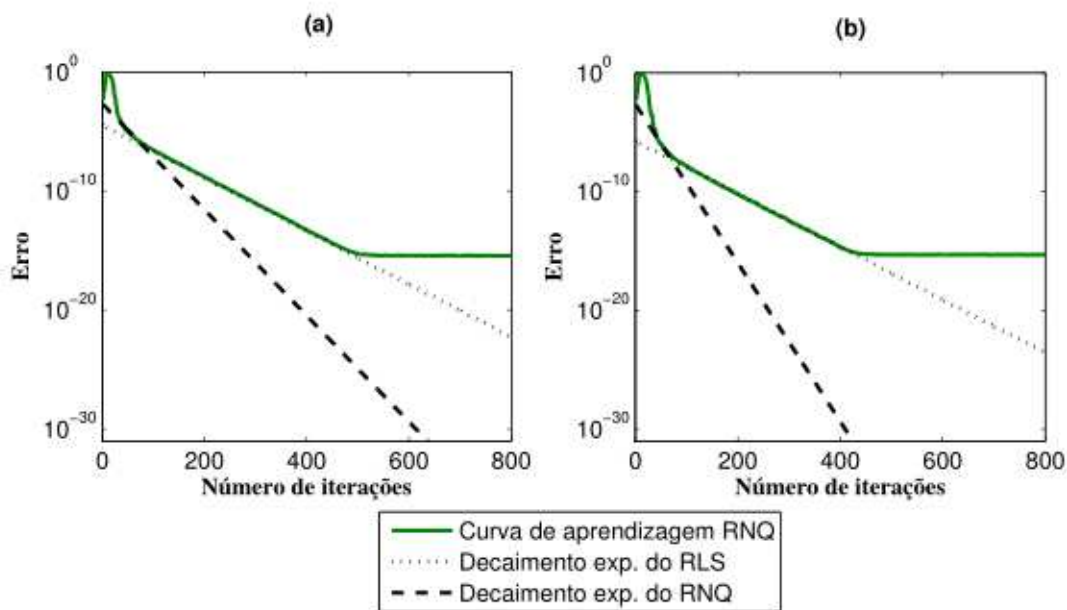


Figura 3.3: Os decaimentos exponenciais para os dois primeiros estágios do tempo de convergência associados ao algoritmo RNQ. As linhas retas denotam a curva de aprendizagem; as linhas tracejadas representam o primeiro estágio do tempo de convergência e as linhas pontilhadas representam o segundo estágio, a qual é aproximadamente o mesmo decaimento exponencial da constante de tempo associada ao algoritmo RLS. (a) A curva de aprendizagem do algoritmo RNQ para $j = 2$ e (b) a curva de aprendizagem do algoritmo RNQ para $j = 3$.

probabilidade (*probability density function*) de uma variável aleatória, que pode ser escrita como uma combinação linear dessas potências. O resultado é um algoritmo dado pela Equação (3.20), similar ao algoritmo RLS, mas com um desempenho melhor em termos de velocidade de convergência, como estabelecido na Equação (3.46) e visualizado na Figura 3.2. A vantagem das superfícies de alta ordem quando comparadas com as quadráticas é que elas naturalmente apresentam uma rápida

convergência com menor desajuste.

Na análise da relação entre velocidade e estabilidade, que é causada pela alta ordem da função de custo do erro, demonstramos que nosso algoritmo é um estimador não enviesado e consistente de \mathbf{w}_* , como podemos ver das Equações (3.24) e (3.33).

Comparando a Equação (3.38) com (3.45) podemos ver que o algoritmo RNQ para uma potência par do erro tem um melhor desempenho que o algoritmo RLS para maiores valores de j tanto no sistema sem ruído como no sistema com ruído. No caso do sistema sem ruído, temos $\xi_{min} \approx 0$ para ambos os algoritmos, mas ressaltando que o algoritmo proposto tem um menor tempo de convergência, como podemos observar da Equação (3.46) e da Figura 3.2a.

Além disso, sendo $\xi_{min} = E[e_n]^2$ para o algoritmo RLS, e para o RNQ para uma potência par do erro temos $\xi_{min} = E[e_n]^{2j}$, obviamente, em um cenário ruidoso o algoritmo proposto atinja um ξ_{min} menor, pois no estado estacionário $|e_n| \leq 1$. Esses fatos, previstos pela Equação (3.38), podem ser vistos na Figura 3.2 (a, b, c and d). Na Figura 3.2a, para um sistema sem ruído, a principal vantagem notada com o aumento de j é na velocidade de convergência, atingindo o mesmo ξ_{min} . No entanto, na presença de ruído, além de uma rápida convergência, o algoritmo proposto atinge um menor erro mínimo para ruídos uniforme, gaussiano e laplaciano como mostrado nas Figuras 3.2b, 3.2c e 3.2d, respectivamente. Essa redução do ξ_{min} também pode ser vista em decibéis na Tabela 3.2. Para $j = 2$, podemos ver que a melhora de em média de 11 dB, 8 dB e 4 dB quando usado distribuições uniforme, gaussiana e laplaciana, respectivamente. Analogamente, para $j = 3$, a melhora foi de 2 dB, 19 dB e 18 dB, respectivamente.

Nota-se que, para $j = 2$, o ruído Uniforme apresentou um melhor desempenho que os outros dois ruídos. Interessantemente, para $j = 3$, os desempenhos para os três ruídos são muito mais próximos o que implica em um maior desempenho para o ruído laplaciano.

O estudo do comportamento dos pesos foi realizado através da análise do desajuste e do tempo de convergência. Comparando a Equação (3.43) com a Equação (3.44) podemos notar que o desajuste diminui com o aumento de j .

Para a análise do tempo de convergência associado ao algoritmo RNQ baseado em uma potência par do erro, usamos as aproximações exponenciais das Equações (3.39) e (3.41) para os dois primeiros estágios da curva de aprendizagem dada pela Equação (3.38), sendo o estágio final dado pelo ξ_{min} . O tempo de convergência resultante foi expressado pelas Equações (3.40) e (3.42), sendo cada uma representando seu estágio. Da Equação (3.46) podemos ver que τ_{RLS} é j vezes maior que τ_{RNQ_2} . Na verdade, mostramos que o primeiro estágio tem uma convergência mais rápida do que o segundo, observamos também que o segundo estágio tem aproximadamente a mesma velocidade de convergência do algoritmo RLS, dessa forma tornando todo o processo de aprendizagem do algoritmo proposto mais rápido que o do algoritmo RLS padrão.

Este é um importante resultado, que mostra, que com o aumento de j é possível alcançar melhores desempenhos, em termos de velocidade de aprendizagem, sem aumento significativo na ordem da complexidade computacional. Mesmo para valores maiores de j pode ser notado que a complexidade computacional continua na ordem $O(N^2)$, como podemos observar da análise feita na subseção 3.3.7.

O resultado apresentado mostra o algoritmo RNQ proposto é uma escolha alternativa para o algoritmo RLS, uma vez que tem um melhor desempenho na velocidade de convergência e erro médio mínimo sem aumento considerável na complexidade computacional. Entretanto, outros algoritmos baseados nessa abordagem poderiam ser propostos, por exemplo, em vez de usar apenas uma potência par do erro, a função de custo pode ser baseada na soma ponderada exponencialmente de todas as potências pares do erro. Essa nova superfície de desempenho, que usa todas as potências pares do erro, poderá possivelmente atingir melhores resultados através da captura informações completas de baixa e alta ordem, que inclui uma ampla variedade de sinais. Este é o ponto abordado no próximo capítulo.

Capítulo 4

Algoritmo RNQ baseado na soma de potências pares do erro

4.1 Introdução

Neste capítulo faremos uma generalização do algoritmo RNQ desenvolvido no capítulo anterior. Para isso, em vez de usarmos apenas uma potência par, utilizaremos como função de custo a soma ponderada das potências pares do erro. Dessa forma escolhemos a função baseada na soma das potências pares do erro como critério a ser minimizado na dedução do algoritmo RNQ proposto. Mostraremos, também que o aspecto chave do uso dessa nova abordagem é incluir o erro instantâneo na determinação do vetor ganho, \mathbf{g} , que controla efetivamente a velocidade de adaptação do algoritmo.

4.2 Dedução do algoritmo

Seja J_n uma função par, contínua e simétrica baseada na soma das potências pares do erro, e , definida por

$$J = \sum_{j=1}^m k^{m-j} E[\lambda e^{2j}], \quad (4.1)$$

sendo m , j e k inteiros positivos. Outra característica dos elementos deste conjunto de funções é que, para um valor fixo de m , podemos determinar intervalos $[-\delta, \delta]$ onde as curvas destas funções apresentem inclinação maior do que a curva da função quadrática, neste mesmo intervalo. Podemos observar esta característica na Figura

4.1, onde plotamos os gráficos das funções $(e^2 + e^4 + e^6)$ e e^2 . Observamos também que estas funções, não possuem mínimo local, apenas o mínimo global.

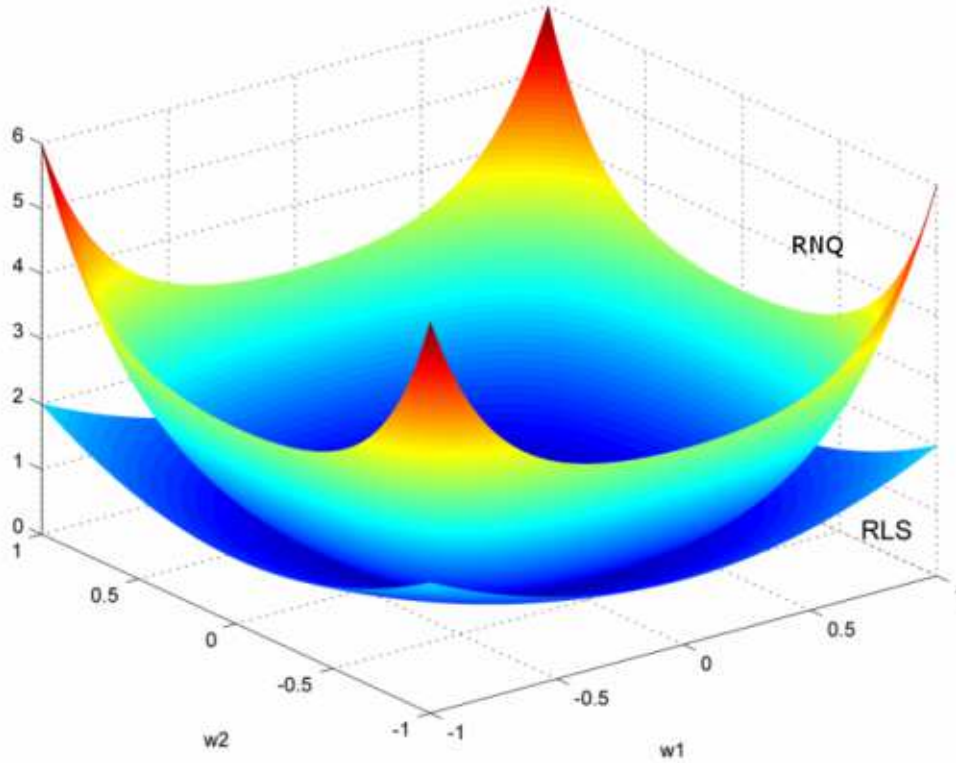


Figura 4.1: Gráficos das funções $(e^2 + e^4 + e^6)$ e e^2 , onde podemos ver a maior inclinação da primeira, no intervalo $[-1; 1]$

Para desenvolver uma generalização do algoritmo RNQ, definimos da Equação (4.1) o seguinte risco empírico,

$$J_n = \sum_{j=1}^m k^{m-j} \sum_{i=1}^n \left\{ \lambda^{n-i} [e_i]^{2j} \right\}, \quad (4.2)$$

sendo m , j e k inteiros positivos. Podemos notar que quando $j = 1$, $k = 1$ e $m = 1$ obtemos a função de custo do MSE. O fator peso exponencial, λ , é o fator de esquecimento convencional que normalmente é selecionado maior que zero e próximo de um, isto é, $0 << \lambda < 1$. A ponderação k^{m-j} é apenas um termo multiplicativo que desempenha o papel de acelerar a convergência de adaptação do algoritmo. Os termos mais interessantes são os que definem o primeiro somatório, j e m , ou seja, podemos observar que $j > 1$ define a potência par do erro e que m adiciona diferentes

potências do erro determinando a função de custo como uma soma das potências pares do erro. Quanto maior o valor de j , mais íngreme é a função de custo e maior o gradiente, exceto perto do mínimo cuja inclinação se torna basicamente plana. Portanto um algoritmo de gradiente descendente utilizando um único valor de j terá uma diminuição muito rápida do erro, mas em seguida é feito uma busca do valor mínimo.

Esta intuição pode ser traduzida matematicamente da seguinte forma: à medida que a potência do erro aumenta, o espalhamento dos autovalores da matriz de autocorrelação das entradas aumenta proporcionalmente. Portanto para $j = 2$ e $m = 1$ essa função de custo se iguala à função do algoritmo *least mean fourth* (LMF) [28]. A função de custo do algoritmo LMF apresenta, basicamente, uma grande inclinação quando comparada com a função de custo do MSE em quase todas as áreas da curva, com exceção de uma vizinhança do mínimo.

Em [13], nós mostramos que a função de custo com $k = 1$, $m = 1$ e $j > 1$ de fato pode ser vista como um algoritmo do tipo RLS que apresenta uma velocidade de convergência mais rápida quando comparado com o algoritmo RLS padrão, mas podemos observar que esse algoritmo utilizando uma única potência par do erro não é ótimo, pelo fato de essa função apresentar na vizinhança do mínimo uma inclinação basicamente plana, como discutido acima.

Agora nos iremos mostrar como a Equação (4.2) pode ser usada para derivar uma nova família de algoritmos do tipo RLS ainda mais rápida em relação a velocidade de convergência.

Objetivando encontrar o vetor peso ótimo, $\hat{\mathbf{w}}_n$, devemos calcular o gradiente da função J_n . Assim, derivando a Equação (4.2) em relação a \mathbf{w} , obtemos,

$$\begin{aligned} \nabla J_n &= \sum_{j=1}^m \left\{ -2.j.k^{m-j} \sum_{i=1}^n [\lambda^{n-i}.e_i^{2j-1}.\mathbf{u}_i] \right\} \\ &= \sum_{j=1}^m \left\{ -2.a_j \sum_{i=1}^n [\lambda^{n-i}.e_i^{\alpha_j}.\mathbf{u}_i] \right\}, \end{aligned} \quad (4.3)$$

sendo $a_j = j.k^{m-j}$ e $\alpha_j = 2j - 2$.

Assim, igualando ∇J_n a zero para que a função da Equação (4.2) atinja o valor

mínimo, definimos o valor ótimo do vetor peso, $\hat{\mathbf{w}}_n$, pela equação escrita em forma matricial,

$$\hat{\mathbf{w}}_n = \Phi_n^{-1} \mathbf{z}_n, \quad (4.4)$$

sendo Φ_n a matriz de autocorrelação do vetor de entrada de dimensão $L \times L$ definida por,

$$\Phi_n = \sum_{j=1}^m \left\{ a_j \sum_{i=1}^n [\lambda^{n-i} \cdot e_i^{\alpha_j} \cdot \mathbf{u}_i \mathbf{u}_i^T] \right\} \quad (4.5)$$

e o vetor de correlação cruzada, \mathbf{z}_n , entre as entradas e a resposta desejada de dimensão $L \times 1$ definido por,

$$\mathbf{z}_n = \sum_{j=1}^m \left\{ a_j \sum_{i=1}^n [\lambda^{n-i} \cdot e_i^{\alpha_j} \cdot d_i \cdot \mathbf{u}_i] \right\}. \quad (4.6)$$

Podemos notar que a matriz de autocorrelação e o vetor de correlação cruzada são agora funções explícitas do erro instantâneo, diferente dos algoritmos convencionais em filtragem adaptativa. Isto significa que nós ganhamos mais controle sobre a dinâmica de aprendizagem, modificando a forma da superfície de desempenho sem afetar a solução final, continuando dessa forma uma solução de Wiener, como iremos demonstrar no final dessa seção.

Isolando o termo correspondente a $i = n$ da Equação (4.5), podemos escrever

$$\Phi_n = \lambda \Phi_{n-1} + \left[\sum_{j=1}^m \{ a_j \cdot e_n^{\alpha_j} \} \right] \mathbf{u}_n \cdot \mathbf{u}_n^T, \quad (4.7)$$

e

$$\mathbf{z}_n = \lambda \mathbf{z}_{n-1} + \left[\sum_{j=1}^m \{ a_j \cdot e_n^{\alpha_j} \} \right] d_n \cdot \mathbf{u}_n, \quad (4.8)$$

sendo Φ_{n-1} o valor anterior da matriz de autocorrelação e \mathbf{z}_{n-1} o valor anterior do vetor de correlação cruzada.

As deduções das Equações (4.7) e (4.8) foram feitas com o intuito de facilitar o cálculo da inversa da matriz de autocorrelação Φ_n , que é necessária para determinar o valor estimado, $\hat{\mathbf{w}}_n$, do vetor ótimo de acordo com a Equação (4.4).

Como a matriz de autocorrelação é positivamente definida e não singular, podemos aplicar o lema de inversão de matrizes para equação recursiva (4.7). Primeiramente faremos as seguintes identificações,

$$\begin{cases} \mathbf{A} = \Phi_n \\ \mathbf{B}^{-1} = \lambda \Phi_{n-1} \\ \mathbf{C} = \mathbf{u}_n \\ \mathbf{D}^{-1} = \sum_{j=1}^m \{a_j \cdot e_n^{\alpha_j}\} \end{cases} \quad (4.9)$$

Substituindo as definições (4.9) na Equação (2.21), obtemos a relação de recursividade para a inversa da matriz de autocorrelação Φ_n , ou seja,

$$\Phi_n^{-1} = \lambda^{-1} \Phi_{n-1}^{-1} - \frac{\lambda^{-1} \Phi_{n-1}^{-1} \mathbf{u}_n \mathbf{u}_n^T \Phi_{n-1}^{-1}}{\lambda \cdot \left[\sum_{j=1}^m \{a_j \cdot e_n^{\alpha_j}\} \right]^{-1} + \mathbf{u}_n^T \Phi_{n-1}^{-1} \mathbf{u}_n}. \quad (4.10)$$

Por conveniência de notação, temos

$$\mathbf{P}_n = \Phi_n^{-1} \quad (4.11)$$

e

$$\mathbf{g}_n = \frac{\mathbf{P}_{n-1} \mathbf{u}_n}{\lambda \cdot \left[\sum_{j=1}^m \{a_j \cdot e_n^{\alpha_j}\} \right]^{-1} + \mathbf{u}_n^T \mathbf{P}_{n-1} \mathbf{u}_n}, \quad (4.12)$$

sendo \mathbf{g}_n o vetor ganho de dimensão $L \times 1$.

Ao compararmos este resultado com o ganho convencional do algoritmo RLS, observamos que o erro, que aparece no termo do somatório, afeta diretamente o fator de esquecimento. Portanto podemos considerar que a nossa abordagem modula efetivamente o fator de esquecimento com a informação de erro instantâneo, o que faz com que o ganho seja não dependente apenas da dinâmica do sinal de entrada, mas também da resposta desejada através do erro.

Assim,

$$\mathbf{P}_n = \lambda^{-1} (\mathbf{P}_{n-1} - \mathbf{g}_n \mathbf{u}_n^T \mathbf{P}_{n-1}). \quad (4.13)$$

Para facilitar futuras manipulações matemáticas, nós reescrevemos a Equação (4.12) como segue:

$$\mathbf{g}_n = \mathbf{P}_n \mathbf{u}_n \left[\sum_{j=1}^m (a_j \cdot e_n^{\alpha_j}) \right]. \quad (4.14)$$

4.2.1 Atualização do vetor peso

No desenvolvimento da estrutura de adaptação do algoritmo RNQ proposto utilizaremos as Equações (4.4), (4.8) e (4.11) para expressar a estimativa $\hat{\mathbf{w}}_n$ do vetor peso no instante n como segue:

$$\begin{aligned} \hat{\mathbf{w}}_n &= \Phi_n^{-1} \mathbf{z}_n \\ \hat{\mathbf{w}}_n &= \lambda \mathbf{P}_n \mathbf{z}_{n-1} + \mathbf{P}_n \left[\sum_{j=1}^m \{a_j \cdot e_n^{\alpha_j}\} \right] d_n \mathbf{u}_n. \end{aligned} \quad (4.15)$$

Substituindo a Equação (4.13) na Equação (4.15), temos:

$$\begin{aligned} \hat{\mathbf{w}}_n &= \mathbf{P}_{n-1} \mathbf{z}_{n-1} - \mathbf{g}_n \mathbf{u}_n^T \cdot \mathbf{P}_{n-1} \mathbf{z}_{n-1} + \left[\sum_{j=1}^m \{a_j \cdot e_n^{\alpha_j}\} \right] \mathbf{P}_n \mathbf{u}_n d_n \\ &= \hat{\mathbf{w}}_{n-1} - \mathbf{g}_n \mathbf{u}_n^T \hat{\mathbf{w}}_{n-1} + \left[\sum_{j=1}^m \{a_j \cdot e_n^{\alpha_j}\} \right] \mathbf{P}_n \mathbf{u}_n d_n. \end{aligned} \quad (4.16)$$

Das Equações (4.14) e (4.16), obtemos a estrutura de atualização do algoritmo RNQ dada pela seguinte equação,

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_{n-1} + \mathbf{g}_n \cdot \xi_n, \quad (4.17)$$

sendo ξ_n uma estimação a priori do erro, dada por,

$$\xi_n = d_n - \hat{\mathbf{w}}_{n-1}^T \mathbf{u}_n. \quad (4.18)$$

As Equações (4.12), (4.13), (4.18) e (4.17) respectivamente, constituem o algoritmo RNQ baseado na soma das potências pares do erro, resumido na tabela 4.1.

4.2.2 Resumo do algoritmo RNQ baseado na soma das potências pares do erro

Na tabela 4.1 podemos observar o resumo do algoritmo RNQ baseado na soma das potências pares do erro.

Tabela 4.1: Algoritmo RNQ baseado na soma das potências pares do erro

Inicializar o algoritmo, definindo

$$\mathbf{P}_0 = \delta^{-1} \mathbf{I}, \quad \delta \text{ é uma constante positiva pequena}$$

$$\hat{\mathbf{w}}_0 = \mathbf{0}$$

Para cada instante de tempo, $n = 1, 2, \dots$, computar

$$\xi_n = d_n - \hat{\mathbf{w}}_{n-1}^T \mathbf{u}_n,$$

$$\mathbf{g}_n = \frac{\mathbf{P}_{n-1} \mathbf{u}_n}{\lambda \left[\sum_{j=1}^m (a_j \cdot \xi_n^{2j}) \right]^{-1} + \mathbf{u}_n^T \mathbf{P}_{n-1} \mathbf{u}_n},$$

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_{n-1} + \mathbf{g}_n \cdot \xi_n,$$

$$\mathbf{P}_n = \lambda^{-1} (\mathbf{P}_{n-1} - \mathbf{g}_n \mathbf{u}_n^T \mathbf{P}_{n-1}).$$

4.2.3 Análise do vetor ganho do algoritmo RNQ

Nessa subseção, iremos analisar com mais detalhe o vetor ganho do algoritmo RNQ da Equação (4.12) e comparar com o vetor ganho do algoritmo RLS padrão. Como visto no Capítulo 2, é sabido que o vetor ganho do algoritmo RLS depende somente do sinal de entrada, enquanto que na Equação (4.12) podemos observar que o vetor ganho do algoritmo proposto é eficientemente modulado não apenas pela dinâmica do sinal de entrada, mas também pelo erro instantâneo. Para melhor análise, iremos reescrever o somatório da Equação (4.14) como

$$\sum_{j=1}^m \left[\frac{k^m}{e_n^2} \cdot \left(\frac{e_n^2}{k} \right)^j \right]. \quad (4.19)$$

Quando o erro quadrático sobre k é maior que um, como normalmente é o caso no momento inicial da aprendizagem, o ganho torna-se maior, pois a base é elevada a α_j . Dessa forma, espera-se que quanto maior a potência na função de custo mais rápida a velocidade de convergência. Por outro lado quando o erro quadrado sobre k é menor que um, os termos do somatório passam a ter significância, e k é sempre elevado a potência de m , que equilibra o fato de o erro ser exponencialmente pequeno. Portanto o ganho será sempre grande para ambos os casos em que o erro é pequeno e grande.

Em outras palavras, a estrutura de atualização do algoritmo RNQ proposto é muito similar ao do algoritmo RLS padrão, mas com mais flexibilidade em termos de velocidade de convergência, ou seja, o vetor ganho \mathbf{g}_n do algoritmo RNQ é controlado pela soma ponderada das potências pares do erro, fornecendo assim uma rápida velocidade de convergência ao algoritmo. Esse fato é muito importante para entendermos sobre quais condições o algoritmo RNQ atingirá a solução ótima do MSE. Dessa forma, podemos reescrever as Equações (4.5) e (4.6) em termos de valor esperado como segue:

$$\Phi_n = \sum_{j=1}^m \{a_j \lambda E[e_n^{\alpha_j} \cdot \mathbf{u}_n \mathbf{u}_n^T]\} \quad (4.20)$$

e

$$\mathbf{z}_n = \sum_{j=1}^m \{a_j \lambda E[e_n^{\alpha_j} \cdot d_n \cdot \mathbf{u}_n]\}. \quad (4.21)$$

Se assumirmos que o erro é estatisticamente independente do sinal de entrada e do sinal desejado, temos

$$\Phi_n = \sum_{j=1}^m \{a_j \lambda E[e_n^{\alpha_j}] \cdot E[\mathbf{u}_n \mathbf{u}_n^T]\}, \quad (4.22)$$

e

$$\mathbf{z}_n = \sum_{j=1}^m \{a_j \lambda E[e_n^{\alpha_j}] \cdot E[d_n \cdot \mathbf{u}_n]\}. \quad (4.23)$$

A solução ótima torna-se

$$\begin{aligned} \Phi_n^{-1} \mathbf{z}_n &= E[\mathbf{u}_n \mathbf{u}_n^T]^{-1} \cdot \frac{E[d_n \cdot \mathbf{u}_n] \cdot \sum_{j=1}^m \{a_j \lambda E[e_n^{\alpha_j}]\}}{\sum_{j=1}^m \{a_j \lambda E[e_n^{\alpha_j}]\}} \\ &= \mathbf{R}^{-1} \cdot \mathbf{P}, \end{aligned} \quad (4.24)$$

que, de fato, é a solução de MSE. Entretanto, o algoritmo RNQ seguirá, na maioria das vezes, a solução de MSE durante sua adaptação.

4.3 Resultados

O sinal de entrada usado nas simulações é um sinal de ruído rosa com distribuição uniforme. Esse sinal de entrada gerado pelo software MatLab apresenta espectro de potência com redução de aproximadamente 3dB por oitava. Fizemos 100 simulações de Monte Carlo para comparar o desempenho dos algoritmos RNQ e RLS.

4.3.1 Identificação perfeita de sistema usando um filtro FIR

Para os primeiros testes criamos uma configuração onde o sistema desconhecido pode ser perfeitamente identificado (i.e. com erro muito próximo de zero). Utilizamos um filtro FIR com resposta ao impulso \mathbf{h} , tamanho $L = 10$ e $\eta_i = 0$, como podemos ver na Figura 4.2. O sinal desejado d_i é obtido pelo sinal de entrada, u_i , filtrado por \mathbf{h} . A saída do filtro adaptativo é denotada por y_i . O sinal erro é definido como a diferença entre d_i e y_i .

Da Equação (4.2), podemos observar que o algoritmo proposto apresenta em sua formulação três parâmetros livres: m , a maior ordem da função de custo, que também controla o número de termos da soma; o parâmetro k que funciona como um termo de ganho adicional, bem como um parâmetro de escala para o erro; e o

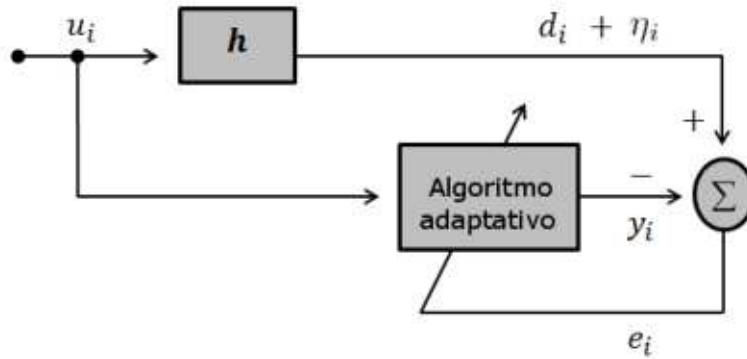


Figura 4.2: Estrutura de um filtro adaptativo no contexto de identificação de sistema, sendo u_i o vetor de entrada, \mathbf{h} um filtro FIR, d_i o sinal desejado, η_i o sinal ruído, e_i o sinal erro e y_i a saída do filtro.

fator de esquecimento λ . Chamamos atenção para esses três parâmetros livres com o objetivo de fazer um estudo sistemático do comportamento do algoritmo RNQ. Para esta análise, mostraremos a dinâmica de aprendizagem (tempo de adaptação e o MSE no estado estacionário) do algoritmo RNQ em função dos três parâmetros, m , k e λ . Definimos o tempo de adaptação como o número de iterações necessárias para que o erro permaneça dentro da faixa de 5% do valor de estado estacionário. Nas Figuras 4.3a, 4.4a e 4.5a podemos visualizar o número de iterações requeridas para que o algoritmo RNQ atinja a convergência e a nas Figuras 4.3b, 4.4b e 4.5b o MSE no estado estacionário. Esses resultados foram obtidos através da média de 100 simulações de Monte Carlo.

Na Figura 4.3 podemos visualizar o tempo de adaptação e MSE final em função da maior potência do erro, m , e o parâmetro k com $\lambda = 0.96$ do algoritmo RNQ em seu estado estacionário, respectivamente. Este estudo é interessante para notarmos que a velocidade de convergência aumenta lentamente com m e com k até $k = 10$ e então satura. Além disso, o MSE é basicamente independente desses dois parâmetros. Podemos observar na Figura 4.4 o tempo de adaptação e o MSE final em função da potência do erro, m , e de dez diferentes valores para o fator de esquecimento, λ , com $k = 10$. Como podemos ver a velocidade de convergência melhora com m , como o esperado, enquanto o erro é basicamente constante com m e variável com o fator de esquecimento (nota-se que os valores de MSE são muito pequenos, de modo que o pico não pode ser significativo).

Finalmente, na Figura 4.5 visualizamos o tempo de adaptação e o MSE final em função do parâmetro k e dez diferentes valores para o fator de esquecimento, λ , com $m = 5$. Observamos que a velocidade de convergência é muito rápida para $k > 10$ e que não existe uma interação notável com o fator de esquecimento, enquanto que o erro não apresenta uma mudança substancialmente independente desses dois parâmetros. Para essas simulações nós utilizamos $m = 1, 2, 3, \dots, 14, 15$; $k = 1, 2, 3, \dots, 49, 50$ e $\lambda = 0, 90; 0, 91; 0, 92, \dots; 0, 98; 0, 99$.

Em conclusão, os resultados apresentados nessas três simulações mostram que o MSE no estado estacionário é basicamente o mesmo, como esperado, enquanto que a velocidade de convergência é controlada, efetivamente pelos parâmetros $m \geq 5$, $k \geq 10$ e λ que normalmente é selecionado como $0 \ll \lambda < 1$. Finalmente, na Figura 4.6 podemos observar a comparação entre os algoritmos RLS, o algoritmo não quadrático de [13] com $j = 3$ e o algoritmo RNQ com $m = 5, 10$ e 15 , e $k = 10$. O fator de esquecimento $\lambda = 0, 96$ para ambos os algoritmos. Este resultado mostra um aumento na velocidade de adaptação do algoritmo RNQ proposto quando comparado com a abordagem proposta em [13].

Nota-se que esses resultados são obtidos quando o sistema faz uma identificação perfeita da planta, o que explica os pequenos erros no estado estacionário. Na Figura 4.7 podemos visualizar a evolução do ganho em (4.14) com a proporção e^2/k da soma (4.19) e o número de interações para $k = 5$. Observamos que a dinâmica do erro no começo da adaptação pode ser controlada por essa proporção. Quando a soma em (4.19) estabiliza, o ganho também é estabilizado, o que mostra que utilizar o ganho em função do erro, ajuda o algoritmo RNQ obter um aumento em sua velocidade de convergência.

4.3.2 Identificação de sistema ruidoso usando um filtro FIR

No caso do sistema com ruído, usamos a estrutura da na Figura 4.1 modificada com a não linearidade $d = x^2$ aplicada para o resultado garantir que o erro será grande, pois a planta é agora uma função não linear da entrada, sendo x a nova saída da planta. Além disso, o sinal desejado d_i é corrompido por um ruído. Nós fizemos simulações para dois diferentes sinais ruidosos: um com distribuição Gaussiana e

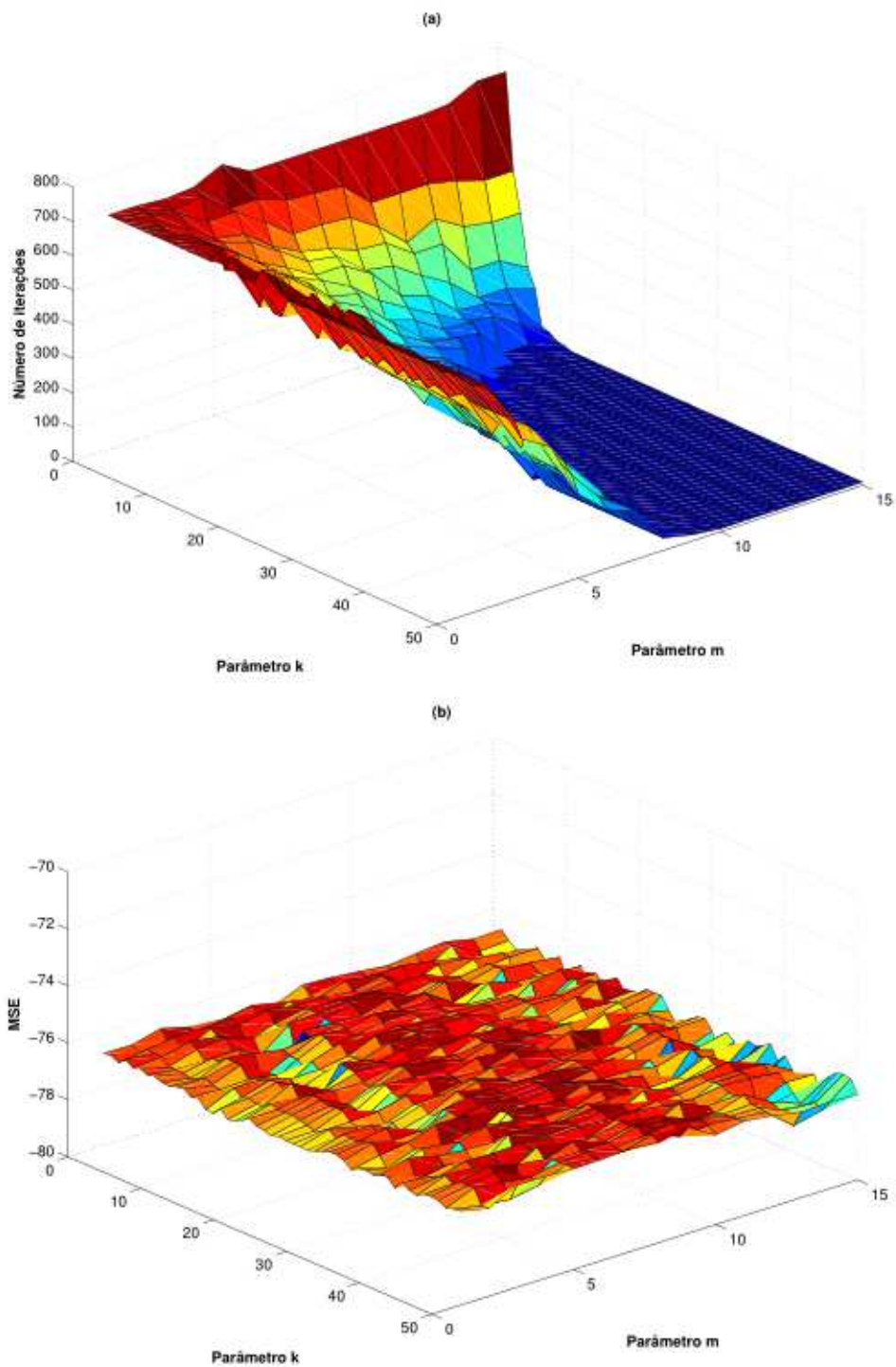


Figura 4.3: Desempenhos (a) do tempo de adaptação e (b) do MSE pelo número m de termos da soma e pelos valores do parâmetro k apresentados no termo de ponderação a_j com $\lambda = 0.96$, sendo $m = 1, 2, 3, \dots, 14, 15$ e $k = 1, 2, 3, \dots, 49, 50$ para ambos os gráficos.

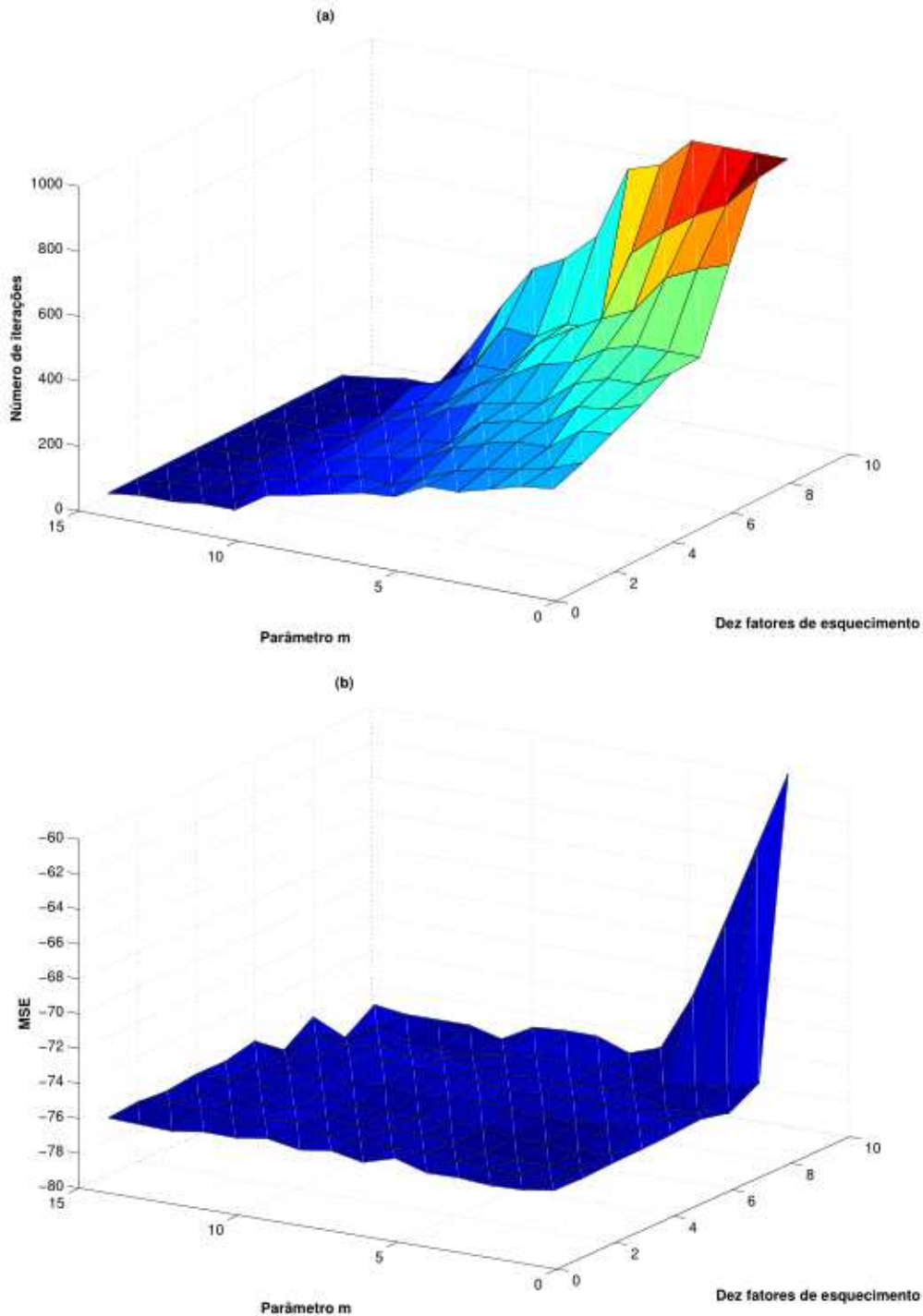


Figura 4.4: Desempenhos (a) do tempo de adaptação e (b) do MSE pelo número m de termos da soma e por dez diferentes valores para o fator de esquecimento λ com $k = 10$, sendo $m = 1, 2, 3, \dots, 14, 15$ e $\lambda = 0,90; 0,91; 0,92, \dots; 0,98; 0,99$ para ambos os gráficos.

outro Laplaciana, sendo que esses sinais foram gerados com média zero e variância unitária. A relação sinal ruído é de $30dB$ para ambos os casos.

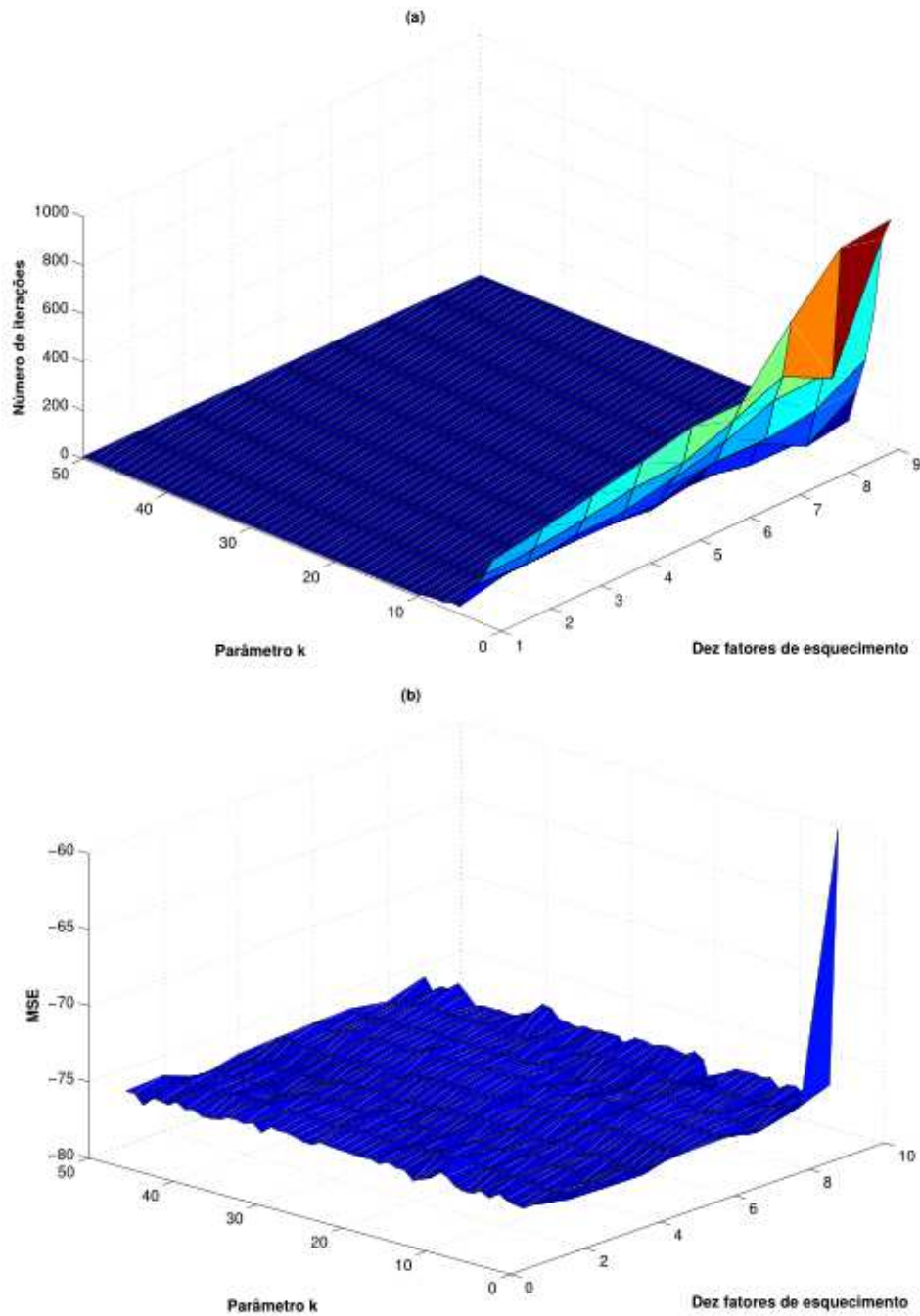


Figura 4.5: Desempenhos (a) do tempo de adaptação e (b) do MSE pelos valores do parâmetro k apresentados no termo de ponderação a_j e por dez diferentes valores para o fator de esquecimento λ com $m = 15$, sendo $m = 1, 2, 3, \dots, 14, 15$ e $\lambda = 0,90; 0,91; 0,92 \dots; 0,98; 0,99$ para ambos os gráficos.

Os resultados das simulações com sinais ruidosos são mostrados na Figura 4.8, onde podemos visualizar as curvas de aprendizagem do algoritmo RLS com $\lambda = 0,96$ e do algoritmo RNQ com $m = 5$ e $m = 10$, $k = 10$ e, $\lambda = 0,96$. Na Figura 4.8a

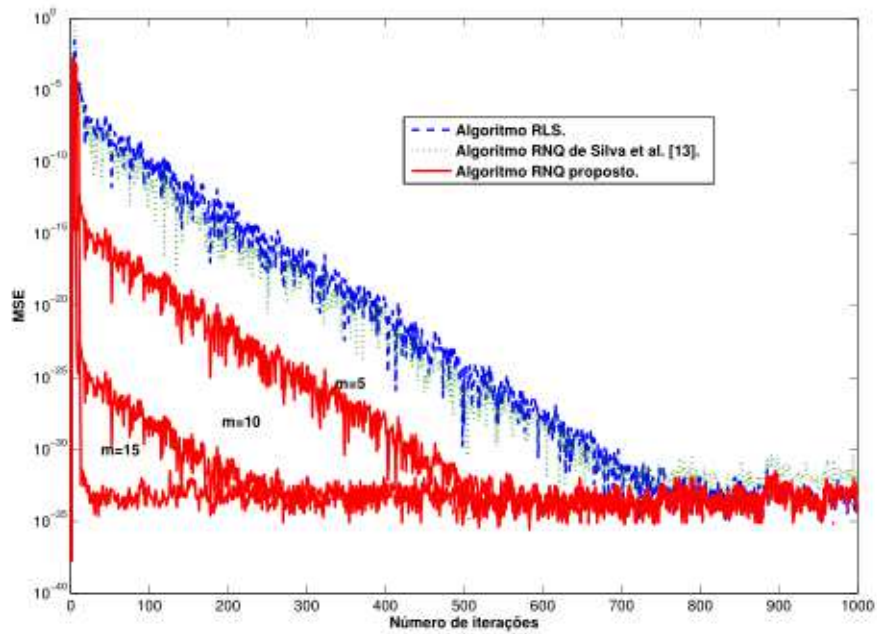


Figura 4.6: Curvas de aprendizagem dos algoritmos RLS, RNQ de Silva et al. [13] com $j = 3$ e o algoritmo RNQ com $m = 5, 10$ e 15 . Para ambos os algoritmos utilizamos o fator de esquecimento $\lambda = 0,96$. As simulações foram feitas sem ruído.

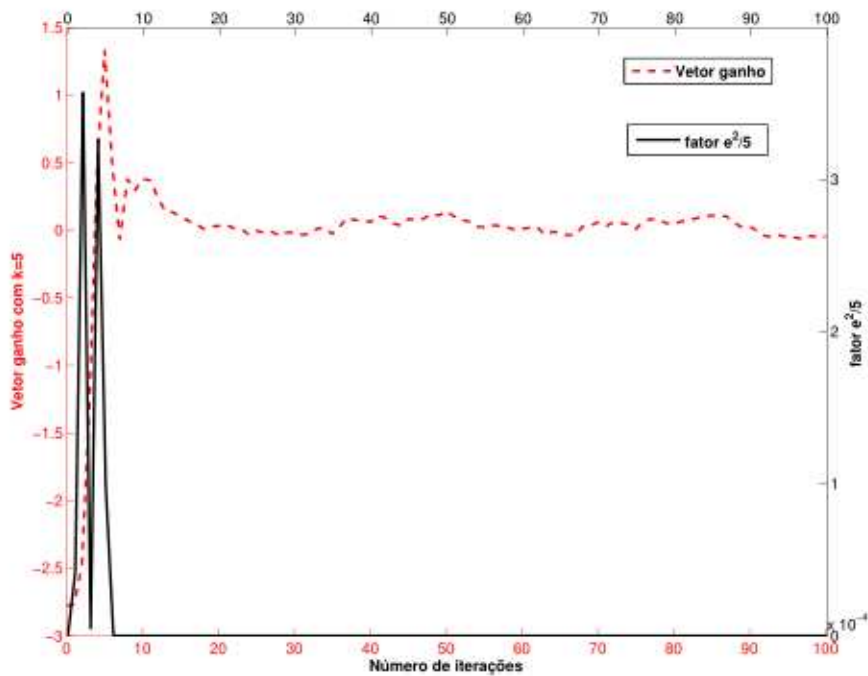


Figura 4.7: Desempenho do vetor ganho e do fator $\frac{e^2}{k}$, sendo $k = 5$ com $m = 5$ e $\lambda = 0,96$ pelo número de iterações.

visualizamos o resultado para o sinal ruído com distribuição Gaussiana e na Figura 4.8b para o ruído com distribuição Laplaciana. Nota-se que para o caso onde a

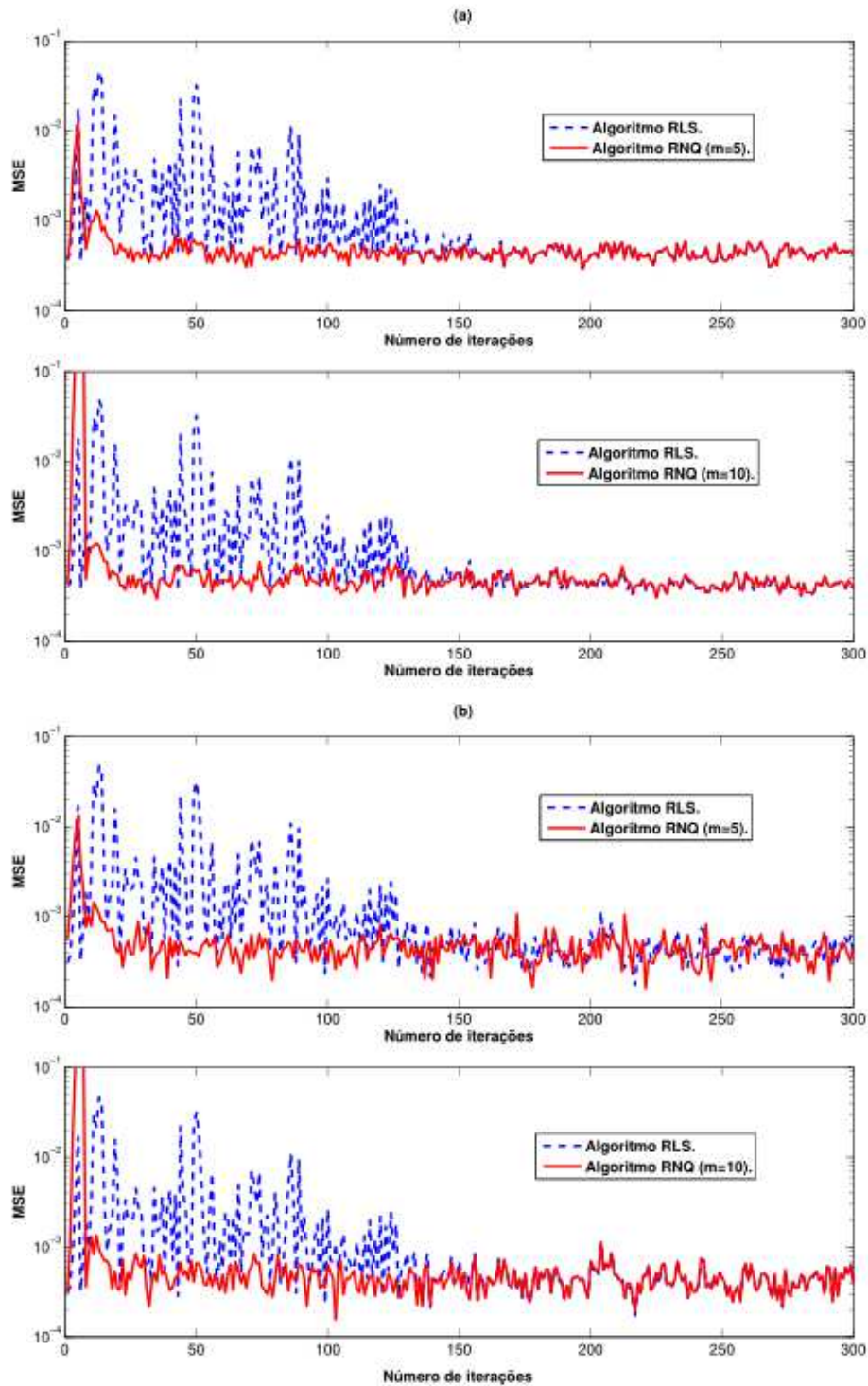


Figura 4.8: Curvas de aprendizagem dos algoritmos RLS e RNQ com $m = 5$ e $m = 10$. Para ambos os algoritmos utilizamos $\lambda = 0,96$. As simulações foram feitas com ruído, (a) Gaussiano e (b) Laplaciano. A relação sinal ruído é de $30dB$ para ambos os casos.

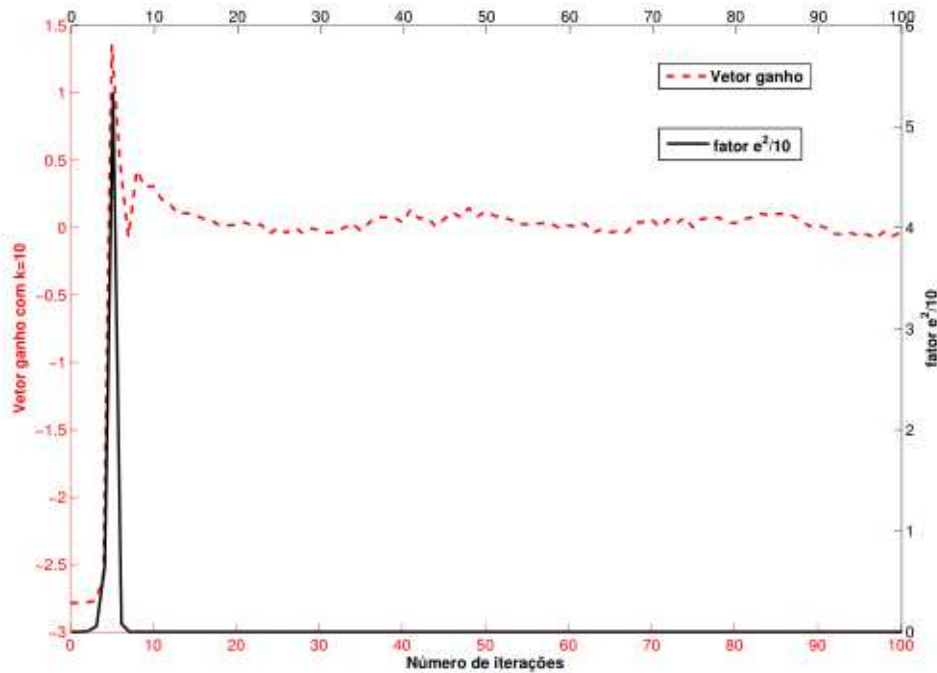


Figura 4.9: Desempenho do vetor ganho e do fator $\frac{e^2}{k}$, sendo $k = 10$ com $m = 10$, $\lambda = 0,96$ e com ruído Laplaciano pelo número de iterações.

planta não pode ser perfeitamente identificada, o mesmo comportamento para o caso da subseção anterior é obtido, isto é o algoritmo RNQ converge mais rápido que o RLS, mas ambos os algoritmos apresentam erros muito próximos no estado estacionário significando que eles convergem para o mesmo valor mínimo no espaço de peso.

A dinâmica do ganho do algoritmo RNQ e da proporção e^2/k , para o caso onde o erro é grande, podem ser visualizadas na Figura 4.9. Como podemos observar as dinâmicas são muito similares às mostradas na Figura 4.7 (caso em que o erro é pequeno), dessa forma podemos afirmar que o algoritmo RNQ apresenta uma dinâmica estável.

4.4 Conclusão

Nesse capítulo, nós apresentamos o desempenho de um novo algoritmo, RNQ, na atualização dos pesos de um filtro adaptativo usando uma função de custo baseada na soma das potências pares do erro. Na dedução desse algoritmo observamos que o erro instantâneo aparece na atualização do vetor ganho que controla efetivamente

a velocidade de adaptação do algoritmo.

Os testes foram realizados tanto para o caso em que o algoritmo é capaz de identificar o sistema com um erro desprezível (identificação perfeita) quanto quando o erro é muito grande. Através dos resultados apresentados mostramos que, com a soma ponderada das potências pares do erro, a nossa função de custo, J_n , teve um desempenho melhor do que o algoritmo RLS padrão em ambas as condições, em termos de velocidade de convergência.

Capítulo 5

Conclusão e trabalhos futuros

Nesse trabalho apresentamos duas abordagens similares à apresentada para o método dos mínimos quadrados recursivo (RLS - *recursive least squares*). A primeira consiste na utilização de uma potência par arbitrária como função de erro a ser minimizada, em contraste com a potência quadrática utilizada no RLS padrão. Nossa motivação foi a generalização apresentada por no algoritmo mínimo quarto médio (LMF - *least mean fourth*) [28].

No estudo da primeira abordagem foi feita uma análise comparativa com o RLS padrão no que se diz respeito à velocidade de convergência e o desajuste mínimo para os casos de sinais puros e corrompidos por ruídos brancos com distribuições uniforme, guassiana e laplaciana.

A segunda generalização, inédita na literatura, consiste na utilização da soma das potências pares ponderadas exponencialmente em lugar das potências pares simples usadas na primeira abordagem que propomos em [13]. Aqui, trabalhamos para tornar o algoritmo menos sensível ao tamanho do erro numa abordagem alternativa. Observamos também, que o erro instantâneo aparece na atualização do vetor ganho controlando efetivamente a velocidade de adaptação do novo algoritmo RNQ. Os testes foram realizados tanto para o caso em que o algoritmo é capaz de identificar o sistema com um erro desprezível (identificação perfeita) quanto quando o erro é muito grande.

Essa nova abordagem traz o efeito do erro instantâneo no algoritmo RLS. De fato, o fator de esquecimento, λ , do RLS se torna uma função da potência do erro, utiliza no RNQ, mas ambos os algoritmos apresentam a mesma solução MSE. Portanto o

algoritmo RNQ afeta apenas a dinâmica de aprendizagem. Uma vez que o algoritmo RLS é conhecido por ser um rastreador fraco para parâmetros variantes no tempo quando comparado com o algoritmo LMS [50], nós esperamos que o algoritmo RNQ seja muito útil na condição de aprendizagem, onde a estatística do sinal muda com o tempo.

Os resultados apresentados mostram que o algoritmo RNQ proposto é uma escolha alternativa para o algoritmo RLS, uma vez que tem um melhor desempenho na velocidade de convergência sem aumento considerável na complexidade computacional.

Em relação a trabalhos futuros, algumas possibilidades merecem ser investigadas. A primeira é um estudo mais detalhados do fator de ponderação apresentado no somatório do algoritmo RNQ, considerando a base desse fator como um número real qualquer. Uma possibilidade de extensão do estudo é a realização da análise de convergência do algoritmo proposto na segunda abordagem, fazendo uma comparação com o tempo de aprendizagem e o desajuste do algoritmo RLS padrão.

Publicações

Estudos descritos nesse trabalho geraram os seguintes artigos:

- Congresso:

A recursive algorithm based on non-linear functions of the error.

Cristiane Silva, Ewaldo Santana, Marcos Antonio Araújo, A. K. Barros.

International Instrumentation and Measurement Technology Conference, Austin, TX., 2010.

- Periódico:

An adaptive recursive algorithm based on non-quadratic function of the error.

Cristiane Silva, Ewaldo Santana, Enio Aguiar, Marcos Antonio Araújo e A. K. Barros.

Signal Processing, volume 92, number 4 (2012) 853 - 856.

- Enviado a periódico:

A RLS Algorithm for non Quadratic Performance Surfaces.

Cristiane Silva, C. Principe, Ewaldo Santana, João V. Fonseca, Marcos Antonio Araújo e A. K. Barros.

Enviado à revista *Signal Processing-Elsevier*. Em fase de revisão.

Referências Bibliográficas

- [1] S. Haykin, Adaptive filter theory, Prentice Hall, (1991).
- [2] B. Widrow and Samuel D. Stearns. Adaptive Signal Processing, Prentice-Hall signal processing series, (1985).
- [3] B. N. Wiener. Extrapolation, Interpolation and smoothing of Stationary Time Series, with Engineering Applications, New York: Wiley, (1949).
- [4] J. Arenas-García, M. Martínez-Ramón, A. Navia-Vázquez, and A. R. Figueiras-Vidal, Plant identification via adaptive combination of transversal filters, Signal Processing, vol. 86, pp. 2430-2438, Sep. (2006).
- [5] J. Arenas-García, A. R. Figueiras-Vidal, and A. H. Sayed, Mean-square performance of a convex combination of two adaptive filters, IEEE Transactions on Signal Processing, vol. 54, pp. 1078-1090, Mar. (2006).
- [6] J. Arenas-García, A. R. Figueiras-Vidal, and A. H. Sayed, Tracking properties of a convex combination of two adaptive filters, in Proc. Of IEEE 13th Workshop on Statistical Signal Processing (SSP 05), pp. 109-114 (2005).
- [7] P. Andersson, Adaptive forgetting in recursive identification through multiple models, International Journal of Control, vol. 42, pp. 1175-1193, Nov. (1985).
- [8] J. Arenas-García and A. R. Figueiras-Vidal, Adaptive combination of normalized filters for robust system identification, Electronics Letters, vol. 41, no. 15, pp. 874-875, Jul. (2005).
- [9] M. Niedzwiecki, Identification of nonstationary stochastic systems using parallel estimation schemes, IEEE Transactions on Automatic Control, vol. 35, pp. 329-334, Mar. (1990).

- [10] M. Niedzwiecki, Multiple-model approach to finite memory adaptive filtering, *IEEE Transactions on Signal Processing*, vol. 40, pp. 470-473, Feb. (1992).
- [11] M. T. M. Silva and V. H. Nascimento, Improving the tracking capability of adaptive filters via convex combination, *IEEE Transactions on signal processing*, (2008).
- [12] M. Chansarkar, U. Desai, B. Rao, A comparative study of the approximate RLS with LMS and RLS algorithms, *Proceedings of IEEE Region 10 Conference TENCEN-89, Bombay*, (1989), 255 - 258.
- [13] C. C. S. Silva, E. C. Santana, E. Aguiar, M. Araújo and A. K. Barros, An adaptive recursive algorithm based on non-quadratic function of the error, *Signal Processing*, volume 92, número 4 (2012) 853 - 856.
- [14] L. Ljung, M. Morf, D. Falconer, Fast calculation of gain matrices for recursive estimation schemes, *International Journal of Control* 27, (1978), 1 - 19.
- [15] G. Carayannis, D. Manolakis, N. Kalouptsidis, A fast sequential algorithm for least-squares filtering and prediction, *Acoustics, Speech and Signal Processing, IEEE Transactions on* 31, (1983), 1394 - 1402.
- [16] J. Cioffi, T. Kailath, Fast, recursive-least-squares transversal filters for adaptive filtering, *Acoustics, Speech and Signal Processing, IEEE Transactions on* 32, (1984), 304 - 337.
- [17] M. M. Chansarkar and U. B. Desai, A fast approximate RLS algorithm, *IEEE Tencon* (1993).
- [18] A.P. Gokhale and P.M. Nawghare, Modified recursive least squares (RLS) algorithm for neural networks using piecewise linear function Circuits, *IEE Proceedings Devices and Systems*, Vol.151, no.6, 15 Dec. (2004).
- [19] Levin, A.U., and Narendra, K.S., Control of non-linear dynamical systems using neural networks: controllability and stabilisation, *IEEE Trans. Neural Netw.*, 4, (2), pp. 192-206, (1993).

- [20] Narendra, K.S., and Parthasarathy, K., Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Netw.*, 1, (1), pp. 4-27, (1990).
- [21] Levin, A.U., and Narendra, K.S., Control of non-linear dynamical systems using neural networks-part II: observability, identification and control, *IEEE Trans. Neural Netw.*, 7, (1), pp. 30-42, (1996).
- [22] Sontag, E.D., Feedback stabilisation using two hidden layer nets, *IEEE Trans. Neural Netw.*, 3, pp. 981-990, (1992).
- [23] Ljung, L., *System identification: theory for the user* (Prentice-Hall, Englewood Cliffs, NJ, USA, 1987).
- [24] Ljung, L., and Soderstrom, T., *Theory and practice of recursive identification* (MIT Press, Cambridge, MA, USA, 1983).
- [25] Ljung, L., and Gunnarsson, S., Adaptation and tracking in system identification-a survey, *Automatica*, 26, pp. 7-21, (1990).
- [26] A. Barros, J. Principe, Y. Takeuchi, C. Sales, N. Ohnishi, An algorithm based on the even moments of the error, in *Proc. 8th Workshop on Neural Networks for Signal Processing*, Toulouse, France, (2003), 879-885
- [27] E. E. C. Santana, Y. Yasuda, Y. Takeuchi, A. Barros, Adaptive Estimation of Impedance Cardiographic Signal by the Sigmoidal Algorithm, *Proceedings of the Fifth International Workshop on Biosignal Interpretation*, September 6-8, Tokyo Japão, (2005).
- [28] E. Walach, B. Widrow, The least mean fourth (LMF) adaptive algorithm and its family, *Information Theory, IEEE Transactions on* 30, (1984), 275 - 283.
- [29] M. Fliess, C. Join and H. S.-Ramirez, Non-linear estimation is easy, *Int. J. Modelling Identification Control*, 4, 1 (2008) 12-27.
- [30] J.F.Weng and S.H.Leung, Nonlinear RLS algorithm for amplitude estimation in class A noise, *Iee proceedings communications*, Vol 147, no 02, April (2000).

- [31] Junfeng Wang, A Variable Forgetting Factor RLS Adaptive Filtering Algorithm IEEE 978-1-4244-4076-4/09,(2009).
- [32] D. Childers, J. C. Principe, Y. Ting, K. Lee, Adaptive WRLS-VFF for speech analysis, Trans. on Signal Processing, 3, 3, 209-213, (1995).
- [33] Eweda, E. ; Macchi, O., Convergence of the RLS and LMS adaptive filters, IEEE Transactions on Circuits and Systems, vol 34, no.7, pp.799 - 803, Jul (1987).
- [34] J. M. Cioffi and T. Kailath, Fast, fixed-order, least-squares algorithms for adaptive filtering, Proc. ICASSP83, pp.679 -683.
- [35] P. Fabre and C. Gueguen, Improvement of the fast recursive least-squares algorithms via normalization: A comparative study, IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-34, no. 2, pp.296 -308 (1986).
- [36] H. J. Kushner and H. Huang, Asymptotic properties of stochastic approximations with constant coefficients, SIAM J. Contr., vol. 19, no. 1, pp.87 -105 (1981).
- [37] Anum Ali, Anis-ur-Rehman, R. Liaqat Ali, An improved gain vector to enhance convergence characteristics of recursive least squared algorithm, International Journal of Hybrid Information Technology, 4, 2, April (2011).
- [38] B. Farhang-Boroujeny, Adaptive Filter Theory and Application, John Wiley & Sons, (1998).
- [39] T. Adali and S.H. Ardalan, On the effect of input signal correlation on weight misadjustment in the RLS algorithm, IEEE Transactions on Signal Processing, vol. 43, no. 4, pp. 988 - 991, Apr (1995).
- [40] S. H. Ardalan, Floating point roundoff error analysis of the RLS and LMS algorithms, IEEE Trans. Circuits Syst., vol. CAS-33, no. 12, pp.1192 -1208 (1986).

- [41] E. Eleftheriou and D. D. Falconer, Tracking properties and steady-state performance of RLS adaptive filter algorithms, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 5, pp.1097 -1109 (1986).
- [42] N. J. Bershad, Analysis of the normalized LMS algorithm with Gaussian inputs, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, no. 4, pp.793 -806 (1986).
- [43] N. J. Bershad, On the probability density function of the LMS adaptive filter weights, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 1, pp.43 -56 (1989).
- [44] V. J. Mathews and Z. Xie, Fixed-point error analysis of stochastic gradient adaptive lattice filters, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 1, pp.70 -80 (1990).
- [45] G. Ungerboeck, Theory on the speed of convergence in adaptive equalizers for digital communication, *IBM J. Res. Develop.*, pp.546 -555 (1972).
- [46] S. T. Alexander, Transient weight misadjustment properties for the finite precision LMS algorithm, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, no. 9, pp.1250 -1258.
- [47] S. H. Ardalan and S. T. Alexander, Fixed point roundoff error analysis of the exponentially windowed RLS algorithm for time varying systems, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, no. 6, pp.770 -783 (1987).
- [48] C. Caraiscos and B. Liu, A roundoff error analysis of the LMS adaptive algorithm, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 1, pp.34 -41 (1984).
- [49] A. H. Sayed, *Adaptive filters*, John Wiley & Sons, (2008).
- [50] E. Eweda, Comparison of RLS, LMS and sign algorithms for tracking randomly time-varying channels,. *IEEE Transactions on Signal Processing*, vol. 42, pp. 2937.2944, Nov. (1994).