

UNIVERSIDADE FEDERAL DO MARANHÃO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Aleksandro Costa Nogueira

*ALGORITMO RECURSIVO BASEADO EM UMA  
FUNÇÃO NÃO QUADRÁTICA USANDO KERNEL*

São Luís

2014

Aleksandro Costa Nogueira

*ALGORITMO RECURSIVO BASEADO EM UMA  
FUNÇÃO NÃO QUADRÁTICA USANDO KERNEL*

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA, como requisito parcial para a obtenção do grau de MESTRE em Engenharia Elétrica.

**Orientador: Prof. Dr. Ewaldo Eder Carvalho Santana**

**Universidade Federal do Maranhão**

**Co-orientador: Prof. Dr. João Viana da Fonseca Neto**

**Universidade Federal do Maranhão**

São Luís

2014

Nogueira, Aleksandro Costa

ALGORITMO RECURSIVO BASEADO EM UMA FUNÇÃO  
NÃO QUADRÁTICA USANDO KERNEL / Aleksandro Costa No-  
gueira - 2014

68.p

1. Filtros adaptativos 2. Metodo kernel 3. Algoritmo RNQ 4.  
Algoritmo KRNQ 5. Redução de ruído. I.Título.

CDU 621:004.021

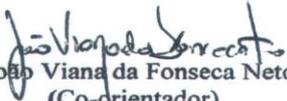
**ALGORITMO RECURSIVO BASEADO EM UMA FUNÇÃO NÃO  
QUADRÁTICA USANDO KERNEL**

**Aleksandro Costa Nogueira**

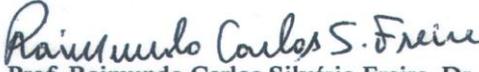
Dissertação aprovada em 28 de fevereiro de 2014.



Prof. Ewandro Eder Carvalho Santana, Dr.  
(Orientador)



Prof. João Viana da Fonseca Neto, Dr.  
(Co-orientador)



Prof. Raimundo Carlos Silvério Freire, Dr.  
(Membro da Banca Examinadora)



Prof. Allan Kardec Duailibe Barros Filho, Ph.D.  
(Membro da Banca Examinadora)

# Agradecimentos

Em primeiro lugar, sou grato a **Deus** por tudo que ele tem me dado. Sem sua misericórdia não teria chegado até aqui.

À minha mãe **Zelieide Costa Nogueira**, meu pai **Salvador Rios Nogueira**, **Zelia Lago** minha madrinha e meu padrinho **João Batista**, pelo amor incondicional.

Aos meus irmãos **Wandro Costa Nogueira** e **Wendre Costa Nogueira**, pela enorme compreensão.

À minha esposa e companheira, **Ana Paula Léda Moura**, por estar ao meu lado em todas as dificuldades que nós vivemos até agora, e por sua generosidade e amor imenso.

Aos meus filhos **Arthur Aleksandro Moura Nogueira** e **Davi Aleksandro Moura Nogueira**, meu grande tesouro.

Ao **Paulo Henrique de Oliveira Léda**, pelo apoio durante toda essa jornada.

Aos pais da minha esposa, a senhora **Maria Cleofas Léda** e o senhor **Paulo Sergio** e o seu filho **Carlos Henrique**, pela confiança e apoio total nessa conquista.

Ao meu orientador Professor **Dr. Ewaldo Santana**, pela responsabilidade em que conduziu esse trabalho e pelas horas de conversas sobre as novas descobertas.

Ao meu co-orientador Professor, **Dr. João Viana da Fonseca Neto**, por toda a confiança creditada e apoio total nessa caminhada difícil, sendo objetivo e mostrando empenho e convicção no trabalho que estava sendo executado naquele momento.

Ao Professor **Dr. Allan Kardec**, pela confiança e oportunidade concedida nos desafios lançados a qualquer momento.

À Professora **Dra. Maria da Guia da Silva**, pelo incentivo e crédito.

Ao Professor **Msc. José Mairton**, pelo ensinamento e dedicação total nesse trabalho

ajudando-me a vencer os desafios pelo qual passei.

Ao Professor **Dr. Marcos Araújo**, pelo seus ensinamentos desde à época da graduação em Matemática.

Ao Professor **Dr. Carlos Alberto**, pelos conselhos no que diz respeito à ciência e a Pós Graduação e também pela credibilidade como profissional e por sua amizade.

Ao Professor **Dr. Antônio Pinto**, pelo seu apoio no início do mestrado e também as longas conversas sobre superação.

Ao Professor **Dr. Jaldyr Varela**, pela amizade desde do periodo como aluno de graduação.

Expresso minha gratidão aos colegas do Programa de Pós-Graduação em Engenharia de Eletricidade (Laboratório de Processamento da Informação Biológica - **PIB** e Laboratório de Automação e Controle - **LAC**), pelas conversas sobre ciência, vida cotidiana e outras coisas: André Cavalcante, Marcus Vinicius, Luis Claudio, Cristiane, Áurea, Fausto Lucena, Eder Santana Jr, Marcos Roberto, Fabiano, Daniel, Gutemberg, Alexandre Pereira Sousa, Mailson da Silva, Gustavo Araújo, Madson Cruz, Jonathan Araujo, Watson Robert Macedo Santos, Marcio Eduardo Gonçalves Silva, Ernesto Ferreira, Marcus André Ramos Berger, Patricia Helena Moraes Rêgo, Charles Molney, Moisés Santos e Santos, Denis Fabrício Sousa de Sá, Sarah Mesquita, Eduardo da Cunha e Cleydson Moraes.

Aos funcionários da Pós-Graduação, em especial a **Alcides Neto** e **José Neto** pela ajuda nos processos, obrigado!

À **FAPEMA** pelo financiamento deste trabalho, pois sem a mesma não seria possível concluir.

À todos o meu muito obrigado!

# Resumo

Este trabalho tem como objetivo desenvolver um modelo analítico que faça a previsão do comportamento do algoritmo RLS como uma função dos parâmetros de projeto (passo de adaptação, função kernel e seus parâmetros). Utiliza-se uma função não quadrática baseado em kernel, realizando uma transformação não linear do espaço de entrada aplicada à filtragem. Foi desenvolvido e implementado na redução de ruídos para a filtragem adaptativa baseada em Kernel, que fornece uma análise do comportamento do algoritmo KRLS, bem como de suas propriedades de convergência. Aplica-se uma função kernel na função de custo a partir da função recursiva não quadrática de quarta potência, que minimiza o erro, definido como a expectativa do custo cumulativo de ações tomadas ao longo de uma sequência de passos. Verifica-se que essa abordagem possibilita a determinação dos parâmetros do problema com uma maior confiabilidade e robustez e o menor custo, quando comparado com algoritmos tradicionais (RLS, KRLS, RNQ).

**Palavras-chave:** Filtros adaptativos, Método kernel, Algoritmo KRNQ, Algoritmo KRNQ, Redução de ruído.

# Abstract

This work has the objective to develop an analytical model that makes prediction of the behavior of the algorithm as a function of the design parameters (step adaptation, kernel function and its parameters). We use a non-quadratic function based on kernel, performing a nonlinear transformation of the input space filtering applied on line. Was developed and implemented in the system for adaptive filtering based on Kernel, which provides an analysis of the behavior of KRLS algorithm as well as its properties of convergence. It applies a kernel function in the cost function from the non-recursive quadratic function of an even power, which minimizes the error, defined as the expectation of the cumulative cost of actions taken along a sequence of steps. It appears that this approach allows the determination of the parameters of the problem with greater reliability and robustness and lower cost compared with traditional algorithms (RLS, KRLS, RNQ) .

**Keywords:** Adaptive filters, Method kernel, RNQ algorithm, KRNQ Algorithm , Noise reduction.

# Sumário

<b>Lista de Símbolos</b>	<b>vii</b>
<b>Lista de Abreviaturas</b>	<b>x</b>
<b>Lista de Figuras</b>	<b>xi</b>
<b>Lista de Tabelas</b>	<b>xiii</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Objetivos . . . . .	2
1.1.1 Objetivos específicos . . . . .	2
1.2 Contribuições . . . . .	2
1.3 Publicação . . . . .	3
1.4 Organização do Trabalho . . . . .	3
<b>2 Filtro Adaptativo</b>	<b>4</b>
2.1 O Combinador Linear Adaptativo . . . . .	5
2.2 A Função de Performance . . . . .	8
2.3 O Vetor Gradiente e o Erro Médio Quadrático . . . . .	9
2.4 Aplicações . . . . .	10
<b>3 Método Kernel Usando o Espaço de Hilbert</b>	<b>11</b>
3.1 Definições e propriedades do Kernel . . . . .	11
3.1.1 A Origem do kernel . . . . .	13
3.1.2 Produto interno aplicado no Kernel . . . . .	14

---

3.2	O "truque" do kernel . . . . .	16
3.2.1	Kernel Polinomial . . . . .	17
3.2.2	Kernel Gaussiano . . . . .	17
3.2.3	Kernel Exponencial . . . . .	18
3.2.4	Kernel Laplaciano . . . . .	18
<b>4</b>	<b>Algoritmos Adaptativos</b>	<b>19</b>
4.1	O algoritmo Recursivo dos Mínimos Quadrados (Recursive Least Square- RLS) . . . . .	19
4.1.1	O algoritmo RLS ponderado exponencialmente . . . . .	21
4.1.2	Resumo do algoritmo RLS . . . . .	22
4.2	Algoritmo Kernel dos Mínimos Quadrados Recursivo (Kernel Recursive Least Square-KRLS) . . . . .	22
4.2.1	O algoritmo KRLS com bloco de inversão de matriz . . . . .	24
4.2.2	Resumo do algoritmo KRLS . . . . .	26
4.3	Algoritmo RNQ baseado em uma única potência par do erro . . . . .	26
4.3.1	O algoritmo RNQ ponderado exponencialmente . . . . .	29
4.3.2	Atualização do vetor peso . . . . .	31
4.3.3	Resumo do algoritmo RNQ . . . . .	32
<b>5</b>	<b>Algoritmo KRNQ</b>	<b>33</b>
5.1	Introdução . . . . .	33
5.2	Desenvolvimento do Método . . . . .	34
5.2.1	Identificação de Sistema KRNQ . . . . .	38
5.2.2	Implementação no Sinal Eletrocardiograma(ECG) Não Artificial e Artificial . . . . .	40
<b>6</b>	<b>Conclusões</b>	<b>46</b>
<b>A</b>	<b>Propriedades de Matrizes</b>	<b>48</b>
	<b>Referências Bibliográficas</b>	<b>53</b>

# Lista de Símbolos

$(.)^T$	Vetor ou Matriz Transposta
$A^{-1}$	Matriz Inversa de $A$
$E(.)$	Valor Esperado de uma Variável Aleatória
$m(.)^T$	A Média de uma Variável Aleatória
$\langle ., . \rangle$	Produto Interno
$\ .\ $	Norma de um Vetor
$ \cdot $	Valores Absolutos de um Número Real ou Determinante de uma Matriz
$\nabla$	Gradiente
$\mathbf{0}$	Vetor Nulo ou Matriz Nula
$\beta$	Regularizador de Parâmetro
$C_n$	Centro de Conjuntos de Interações de $n$
$d_n$	Saída Desejada em $n$ Interações
$A$	Conjunto de Dados
$e_n$	Saída de Erro Estimado no Tempo de $n$ Interações
$\mathbf{F}$	Vetor ou Matriz Transposta
$\mathcal{H}$	Reprodução do Kernel no Espaço de Hilbert
$I$	Matriz Identidade
$J_n$	Função de Custo do Erro em $n$ Interações
$\kappa(u, u')$	Kernel (ou Covariância) em Função da Evolução de $u$ em $u'$
$L$	Dimensão dos Dados de Entrada
$\lambda$	Fator de Esquecimento
$M$	Dimensão do Espaço Futuro
$N$	Número de Treinamento dos Dados

---

$K_{n \times n}$	Matriz Complexa é dita Definida Positiva
$s_i, s_j$	Números Reais
$\bar{s}_j$	Denota o Complexo Conjugado
$\mathbb{R}$	Conjunto dos Números Reais
$\mathbb{C}$	Conjunto dos Números Complexos
$U$	Conjunto de Dados de Entrada
$\mathbf{U}$	Entrada de Vetor $U$
$d$	Grau do Polinômio
$c$	Termo Constante
$h$	Filtro FIR
$m$	Inteiro Positivo que Define a potência Par do Erro
$\alpha$	Parâmetro Ajustável que é dado pela Inclinação
$Z^{-1}$	Operador de Unidade de Atraso
$K_{n \times n}$	Matriz Complexa
$K$	Matriz Gram
$f$	Função Qualquer
$\Phi$	Dados da Matriz de Entrada Transformada
$\varphi(\cdot)$	Mapeamento Induzido pela Reprodução do Kernel
$\varphi_n$	Transformação do dado do Filtro no Tempo ou $n$ Interações
$\mathbf{w}_n$	Estimativa do Peso em $n$ Interações (Vetor Coluna no Espaço Euclidiano)
$\omega_n$	Estimativa do Peso em $n$ Interações (Vetor Coluna no Espaço Característico)
$O(\cdot)$	A Ordem de um Número
$\zeta$	Erro Médio Quadrático
$\xi$	Estimação a Priori do Erro
$\xi^*$	Diferencia da Média Quadrática Percentual
$E_{orig(i)}$	Sinal do ECG Original
$E_{rec(i)}$	Sinal Recuperado Após a Filtragem do Sinal Ruidoso
$\alpha_j = 2j - 2$	Expressão que Simplifica os Cálculos na Potência do Erro
$a_j = k^{n-i}$	Fator de ponderação para Simplificar os Cálculos

<b>U</b>	Matriz de Dados de Entrada
<b>u</b>	Dados de Entrada
$\hat{\mathbf{w}}_n$	Valor Ótimo do Vetor Peso
$\Phi_{n-1}$	Valor Anterior da Matriz de Autocorrelação
$\mathbf{Z}_{n-1}$	Valor Anterior do Vetor de Correlação Cruzada
<b>A</b> e <b>B</b>	Matrizes Definidas Positivas de Dimensão $L \times L$
<b>D</b>	Matriz Positiva Definida de Dimensão $N \times L$
<b>C</b>	Matriz $L \times N$
$g_n$	Vetor Ganho de Dimensão $L - 1$

# Lista de Abreviaturas

LMS	Mínimo Quadrado Médio ( <i>Least Mean Squares</i> )
RLS	Mínimo Quadrado Recursivo ( <i>Recursive Least Squares</i> )
RNQ	Recursivo Não Quadrado ( <i>Recursive Non Squares</i> )
KRLS	Kernel Mínimo Quadrado Recursivo ( <i>Kernel Recursive Least Squares</i> )
KRNQ	Kernel Recursivo Não Quadrado ( <i>Kernel Recursive Non-Quadratic</i> )
PCA	Análise de Componentes Principais ( <i>Principal Components Analysis</i> )
RKHS	Espaço de Hilbert de Kernel Reprodutivo ( <i>Reproducing Kernel Hilbert Space</i> )
LS	Mínimos Quadrados ( <i>Least Squares</i> )
SVM	Máquina de Vetor de Suporte ( <i>Support Vector Machine</i> )
LDA	Análise Discriminante Linear ( <i>Linear Discriminant Analysis</i> )

# Lista de Figuras

2.1	Filtragem adaptativa para identificação de sistemas . . . . .	5
2.2	Combinador Linear Adaptativo-Forma Transversal . . . . .	6
2.3	Superfície de uma porção quadrática tridimensional, juntamente com alguns contornos. O erro quadrático médio está plotado na vertical, $w_0$ e $w_1$ variam de -1 a 1. . . . .	7
3.1	Ilustração da construção do Espaço kernel com característica. . . . .	12
3.2	Definições e propriedades da estrutura do Kernel . . . . .	13
4.1	Superfície RNQ tridimensional, juntamente com alguns contornos. O erro quadrático médio está plotado na vertical, $w_0$ e $w_1$ variam de -1 a 1. . . . .	27
5.1	Gráfico comparando o RNQ co RLS em relação a sua inclinação que é maior na primeira em peso de $w_0$ e $w_1$ variam de $[-1, 1]$ . . . . .	34
5.2	Estrutura do filtro adaptativo na configuração de identificação do sistema, onde $u_i$ é vetor de entrada, $\mathbf{h}$ é o filtro FIR, $d_i$ é o sinal desejado, $\eta_i$ é sinal de ruído, $e_i$ é erro sinal e $y_i$ é saída do filtro. . . . .	38
5.3	Curva de aprendizagem dos algoritmos KRNQ e KRLS quando foi usada uma distribuição com erro Gaussiano. . . . .	38
5.4	Gráfico do erro em tempo real antes dos coeficientes de estimação do filtro para ambos os algoritmos. . . . .	39
5.5	Curva de aprendizagem dos algoritmos KRNQ e KRLS quando usada uma distribuição de erro Laplaciano. Usamos $\beta = 0,98$ para ambos os algoritmos é $k = 2$ para o KRNQ. . . . .	39

---

5.6	Gráfico do erro no tempo real antes dos coeficientes de estimação do filtro para ambos os algoritmos. . . . .	39
5.7	Diagrama de bloco do filtro adaptativo $s$ é o sinal determinística, $n$ é o ruído descorrelacionado com $s$ e $[u_1, u_2, \dots, u_n]^T$ é o vetor de entrada. . . . .	41
5.8	Gráfico do sinal que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo RNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro. . . . .	41
5.9	Gráfico do sinal ECG que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo KRNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro. . . . .	42
5.10	Gráfico do sinal que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo RNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro . . . . .	42
5.11	Gráfico do sinal ECG que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo KRNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro. . . . .	43
5.12	Gráfico comparando as curvas de aprendizagem dos algoritmos RNQ e KRNQ em um sinal cardíaco o ECG não artificial. . . . .	43
5.13	Gráfico do sinal artificial que foi adicionado um ruído gaussiano de 15% e logo temos o algoritmo RNQ gera o gráfico do sinal ECG sem o erro depois do filtro. . . . .	44
5.14	Gráfico do sinal ECG artificial que foi adicionado um ruído gaussiano de 15% e logo temos o algoritmo KRNQ gera o gráfico do sinal ECG sem o erro depois do filtro. . . . .	44
5.15	Gráfico comparando as curvas de aprendizagem dos algoritmos RNQ e KRNQ em um sinal cardíaco o ECG artificial. . . . .	45

# Lista de Tabelas

5.1	Estimação das médias quadráticas percentual do sinal ECG não artificial de 1% até 25% com sinal ruidoso . . . . .	43
5.2	Estimação das médias quadráticas percentual do sinal ECG artificial de 1% até 25% com sinal ruidoso . . . . .	44

# Capítulo 1

## Introdução

O problema de filtragem teve as primeiras contribuições com Wiener para a teoria de filtragem ótima. Esta contribuição consistiu na formulação do problema de predição linear, em que foi obtido um sistema de equações (chamadas equações de Wiener-Hopf) [1], as quais resultam, após sua resolução, nos coeficientes do filtro ótimo. O caso mais geral é em termos de duas funções de correlação: a função de autocorrelação do sinal de entrada e a função de correlação cruzada entre o sinal de entrada e o sinal desejado.

A filtragem adaptativa tem sido largamente utilizada nas últimas décadas em diversas áreas de aplicação. Sua principal característica é a possibilidade de ajuste ótimo dos parâmetros de filtragem na ausência de uma informação estatística prévia dos sinais envolvidos [2] e [3]. Modelos lineares são ainda rotineiramente usados devido a sua simplicidade, do ponto de vista conceitual e de implementação. Entretanto, em muitas situações práticas, tais como, identificação, predição e controle aplicados à áreas de comunicações e engenharia biomédica, um processamento não linear do sinal se faz necessário.

Diferentemente do caso de sistemas lineares e invariantes no tempo, os quais são completamente caracterizados por suas respostas ao impulso, não existe um contexto matemático simples para descrever os sistemas não lineares. Filtros polinomiais, comumente chamados de séries de Volterra [4] que tem como dificuldade prática em identificação de sistema é que, mesmo para sistemas pouco não lineares, o número de parâmetros a determinar é muito grande e a necessidade de parâmetros é utilizar bases ortogonormais como forma de parametrizar os núcleos. A rede neural é o mais popular e estudado em modelos não lineares para filtragem adaptativa [5].

De maneira alternativa, foram verificados progressos em métodos de aproximação de funções baseados em Espaço de Hilbert de Kernel Reprodutivo (RKHS) [6] [7] incluindo, por exemplo, máquina de vetor de suporte [8]. Uma característica comum dos métodos

baseados em kernel, entretanto, é a utilização de matrizes e modelos cujas dimensões são iguais ao número de dados avaliados. Os métodos baseados em kernel, em particular as Máquinas de Vetor de Suporte (SVMs), são aplicados em muitos problemas do mundo real, sendo considerados estados da arte em vários domínios [9]. Recentemente, tais métodos tornaram-se uma tendência em aprendizagem de máquina e inferência bayesiana.

O trabalho proposto sobre o RLS e suas aplicações em Kernel para filtragem tem como principal foco apresentar um levantamento e método para desenvolver um algoritmo de aproximação baseado em Kernel de mínimos quadrados recursivos para solucionar os problemas com adaptação da equação algébrica de Riccati discreta e para regulador linear quadrático discreto usado também no algoritmo recursivo não quadrático. O algoritmo dos Mínimos Quadrados Recursivos (RLS) podem ser vistos como um caso especial do algoritmo RNQ que é uma forma dos mínimos quadrados que utiliza a aprendizagem por reforço, isto é, o aprendizado de um mapeamento de entrada e saída é realizado através da interação contínua com o ambiente, visando a minimizar um índice escalar de desempenho. Tais algoritmos têm como vantagem a baixa sensibilidade à natureza do sinal de entrada e uma maior velocidade de convergência quando comparado aos algoritmos de gradiente estocástico [4]. O algoritmo mais popular desta família é o RLS.

## 1.1 Objetivos

### 1.1.1 Objetivos específicos

- Propor uma versão baseada em kernel do algoritmo recursivo não quadrático (RNQ);
- Permitir a filtragem não linear em um espaço de entrada de dados lineares, que transforma com o produto vetorial desses dados gravados em uma alta dimensão de característica que está relacionada ao espaço não linear de entrada;
- Explorar as estatísticas de ordem superior dos sinais gravados;
- Desenvolver o algoritmo que apresente melhor desempenho em um problema de identificação de sistema quando comparado ao algoritmo dos mínimos quadrados recursivo baseado em Kernel (KRLS).

## 1.2 Contribuições

As principais contribuições desse trabalho são as seguintes:

- Identificação de sistemas não lineares, no entanto o algoritmo kernel recursivo não quadrático pode ser adequado para aplicações em tempo real, e apresentam um comportamento estável de operação com grandes conjuntos de dados.
- Aplicação em tempo real do algoritmo KRNQ em sinais de Eletrocardiograma (ECG), que tiveram excelentes resultados na filtragem desse sinal.

### 1.3 Publicação

- Recursive Algorithm Based on Non-Quadratic Functions Using Kernel, IEEE - MEMEA, 2014.

### 1.4 Organização do Trabalho

Essa dissertação está organizada em capítulos, os quais descrevem os problemas e sua formulação desse novo método, teste de validação e análise de resultados.

O **Capítulo 2** trata de conceitos básicos sobre filtragem adaptativa que são abordados nesse trabalho e assim introduzir os conceitos de filtro.

No **Capítulo 3** é definido o conceito do método kernel no espaço de Hilbert no qual aborda a metodologia utilizada em sua análise que estão interligados ao novo algoritmo.

No **Capítulo 4** mostram-se os algoritmos RLS e KRLS que influenciam no cálculo do novo algoritmo e também discorre-se sobre o algoritmo RNQ e seu desenvolvimento e suas propriedades para que possamos interligá-los com os conceitos que envolvem o kernel.

No **Capítulo 5** é descrito o algoritmo KRNQ além das técnicas baseadas em kernels e métodos matemáticos propostos para a realização dessa dissertação.

No **Capítulo 6** são as conclusões do trabalho.

## Capítulo 2

# Filtro Adaptativo

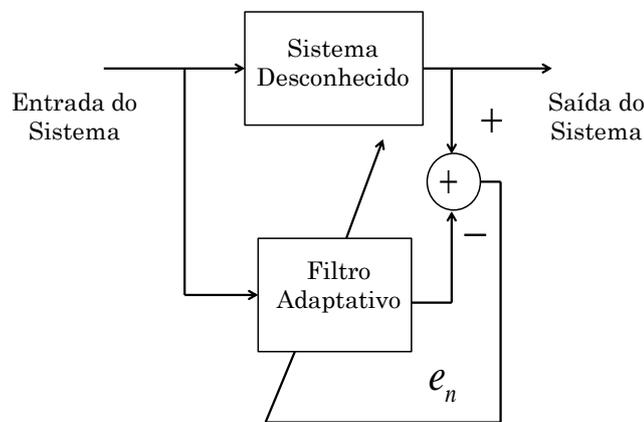
O objetivo da filtragem de sinais é retirada do ruído e melhorar a qualidade do sinal de acordo com um critério de desempenho. Os sinais podem ser considerados tanto no domínio do tempo como no domínio da frequência. A diferença dos filtros adaptativos com relação aos demais é o seu desempenho auto ajustável e variante no tempo. Contudo muitos algoritmos adaptativos se utilizam do erro quadrático médio 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. Para isso pode-se proceder de tal forma a encontrar esse mínimo, o mais rapidamente possível e com o menor erro final.

Na filtragem convencional do tipo Finite-Impulse-Response (FIR) ou do tipo Finite-Impulse-Response (IIR), geralmente assume-se que os parâmetros do processo estocástico de entrada são conhecidos, tais parâmetros determinam as características do sinal de entrada [4]. No entanto, em muitos problemas práticos, pelo menos alguns parâmetros de entrada podem mudar, ou ainda, o sistema pode não conhecer exatamente o comportamento destes parâmetros. Nestes casos é altamente desejável um filtro que possua o atributo de auto aprendizado, de maneira que este possa ajustar-se automaticamente de acordo com cada situação.

Filtros adaptativos são interessantes nos casos onde algumas das características da aplicação não são conhecidas. Desta maneira, pode-se dizer que os filtros adaptativos são mais adequados para aplicações em que as condições do sinal de entrada ou os parâmetros do sistema variam lentamente e o filtro seja capaz de se auto ajustar para compensar essas mudanças. O algoritmo clássico LMS (Least Mean Squares) constitui uma das técnicas mais conhecidas para a implementação de um filtro adaptativo [3]. Esta adaptação consiste no ajuste contínuo (e adaptativo) dos valores dos coeficientes do filtro, tendo como métrica a minimização do erro no sentido da média dos quadrados[10]. Como exemplos de aplicação

que requerem filtragem adaptativa, pode-se citar: decodificação de canal de transmissão, detecção de sinais em presença de ruídos aleatórios, cancelamento de eco, reconhecimento de padrões de imagens e reconhecimento de padrões de voz, dentre outros.

Os mínimos quadrados em filtro converge solução para o filtro de Wiener, assumindo que o sistema desconhecido é linear invariante no tempo e o ruído é estacionário [11]. Ambos os filtros podem ser usados para identificar a resposta de impulso de um sistema desconhecido, sabendo-se apenas o sinal original de entrada e a saída do sistema desconhecido [3]. Ao relaxar o critério de erro para reduzir o erro de amostragem corrente, em vez de minimizar o erro total sobre todos os  $n$ , o algoritmo LMS pode ser derivado a partir do filtro de Wiener[10].



**Figura 2.1:** Filtragem adaptativa para identificação de sistemas

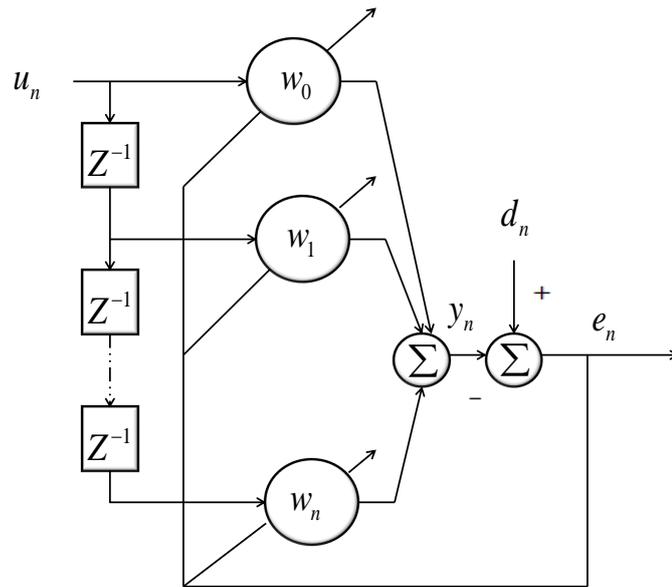
## 2.1 O Combinador Linear Adaptativo

A estrutura mais usada na implementação de filtros adaptativos é o combinador linear adaptativo, mostrado na Figure 2.2 onde pode-se observar que o filtro adaptativo possui uma única entrada  $u_n$  (no tempo  $n$ ), definida como [12];

$$\mathbf{u}_n = [u_n, u_{n-1}, \dots, u_{n-(L-1)}]^T \quad (2.1)$$

sendo  $L$  a ordem do filtro  $\mathbf{w}_n$  é o vetor peso no tempo  $n$ , definido por;

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



**Figura 2.2:** *Combinador Linear Adaptativo-Forma Transversal*

e a saída,  $y_n$ , é igual ao produto interno de  $u_n$  por  $\mathbf{w}_n$ :

$$y_n = \mathbf{u}_n^T \mathbf{w}_n = \mathbf{w}_n^T \mathbf{u}_n \quad (2.3)$$

Conforme Figura 2.2, o sinal de erro no instante  $n$  é dado por;

$$e_n = d_n - y_n \quad (2.4)$$

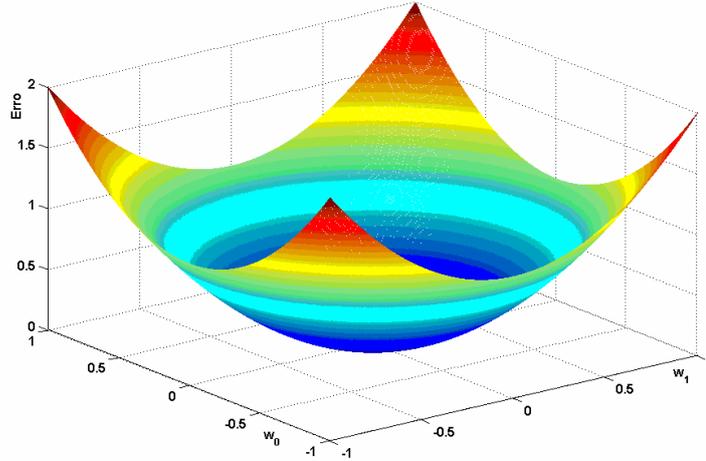
Substituindo nesta Equação 2.4, temos:

$$e_n = d_n - \mathbf{w}_n^T \mathbf{u}_n \quad (2.5)$$

Ao relaxar a soma infinita do filtro de Wiener que representa a solução ótima do erro no tempo  $n$ , pode ser encontrada pelo método de otimização, denominado "método de decida 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-Holf, em forma matricial, esta solução ótima de Wiener e o algoritmo LMS pode ser derivada [4].

O erro quadrado médio pode ser expresso como;

$$\zeta = E [e_n^2] = [d_n - y_n]^2 = [d_n - \mathbf{w}_n^T \mathbf{u}_n]^2 \quad (2.6)$$



**Figura 2.3:** Superfície de uma porção quadrática tridimensional, juntamente com alguns contornos. O erro quadrático médio está plotado na vertical,  $w_0$  e  $w_1$  variam de -1 a 1.

$$\frac{\partial E}{\partial \mathbf{w}_n} = \frac{\partial}{\partial \mathbf{w}_n} (d_n - y_n)^2 \quad (2.7)$$

Aplicar regra da cadeia e definição de substituto  $y_n$ ;

$$\frac{\partial E}{\partial \mathbf{w}_n} = 2(d_n - y_n) \frac{\partial}{\partial \mathbf{w}_n} \left( d_n - \sum_{n=0}^{N-1} \mathbf{w}_n \mathbf{u}_n \right) \quad (2.8)$$

$$\frac{\partial E}{\partial \mathbf{w}_n} = -2e_n \mathbf{u}_n \quad (2.9)$$

A partir desta equação, é possível derivar uma equação para cada atualização  $\mathbf{w}_n$  em cada novo  $n$  usando gradiente descendente e um tamanho de passo  $\mu$ ;

$$\hat{\mathbf{w}}_n = \mathbf{w}_n - \mu \frac{\partial E}{\partial \mathbf{w}_n} \quad (2.10)$$

que se transforma, para  $i = 0, 1, \dots, N - 1$  ;

$$\hat{\mathbf{w}}_n = \mathbf{w}_n + 2\mu e_n \mathbf{u}_n. \quad (2.11)$$

Tal parâmetro é uma constante que comanda a velocidade de convergência do algoritmo que é a equação de atualização do LMS.

## 2.2 A Função de Performance

O sinal erro no instante  $n$  é dado por;

$$\zeta_n = d_n - y_n \quad (2.12)$$

Substituindo o valor de  $y_n$  dado na Equação 2.13:

$$\zeta_n = d_n - u_n^T W_n = d_n - W_n u_n^T \quad (2.13)$$

Na discussão que segue, supõe-se que as entradas do vetor peso não estarão sendo ajustados, ou seja,  $n$  é fixo. Tomando o produto escalar  $\langle \zeta_n, \zeta_n \rangle$ , obtém-se o erro quadrático  $\zeta_n^2$  dado por;

$$\zeta_n^2 = \langle \zeta_n, \zeta_n \rangle = d_n^2 + W_n^T u_n u_n^T W_n - 2d_n u_n^T W_n \quad (2.14)$$

Assumindo a hipótese adicional de que  $\zeta_n$ ,  $d_n$  e  $u_n$  são estatisticamente estacionários e tomando o valor esperado desta última expressão obtém-se;

$$E[\zeta_n^2] = E[d_n^2] + W_n^T E[u_n u_n^T] W_n - 2E[d_n u_n^T] W_n \quad (2.15)$$

Lembramos que o valor esperado de um produto não é em geral o produto dos valores esperados. Entretanto, tal resultado é verdadeiro quando as variáveis são estatisticamente independentes [1]. Entretanto,  $u_n$  e  $d_n$  não são independente. Seja  $R$  a matriz quadrada de ordem  $L$ , dada por;

$$R = E[u_n u_n^T] = E \begin{bmatrix} u_{0,n}^2 & u_{0,n} & \cdots & u_{0,n} \\ u_{1,n} & u_{1,n} & \cdots & u_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ u_{L-1,n} & u_{L-1,n} & \cdots & u_{L-1,n}^2 \end{bmatrix} \quad (2.16)$$

A matriz 2.16 é chamada matriz de correlação das entradas. Os termos da diagonal principal são as médias quadradas dos componentes das entradas e as correlações cruzadas são as correlações dos componentes das entradas. Seja  $P$  definido como o vetor coluna.

$$P = E[d_n u_n] = E[d_n u_{0,n}, d_n u_{1,n}, \dots, d_n u_{(L-1)n}]^T \quad (2.17)$$

Esse vetor é a correlação cruzada entre o sinal resposta e os componentes das entradas. Os elementos tanto de  $R$  quanto de  $P$  são estatísticas de segunda ordem quando  $U_n$  e  $d_n$  são estacionários.

Agora tomemos o erro médio quadrático dado na Equação 2.15 e o designemos por  $\zeta$ , ou seja:

$$\zeta = E [\zeta_n^2] = E [d_n^2] + W_n^T R W_n - 2P^T W_n \quad (2.18)$$

Segue-se dessa expressão que o erro médio quadrático  $\zeta$  é uma função das componentes do vetor peso  $W$  quando as componentes da entrada desejada são variáveis estocásticas estacionárias.

Uma porção de uma típica função erro médio quadrático está ilustrada na Figura 2.3 abaixo. Essa superfície é chamada superfície de performance. Note que ela é concava para cima. As curvas de nível dessa superfície são elipses. A projeção do ponto onde a superfície atinge seu valor mínimo é justamente o ponto de ótimo  $W$  do erro médio quadrático.

## 2.3 O Vetor Gradiente e o Erro Médio Quadrático

Muitos dos processos adaptativos usuais que buscam o vetor peso que determina o número da superfície se utiliza os métodos do gradiente. O gradiente da superfície de performance do erro médio quadrático, denotado por  $\nabla(\zeta)$ , é um vetor coluna que pode ser obtido derivando parcialmente a Equação 2.18 com relação ao vetor peso. Portanto,

$$\nabla\zeta = \frac{\partial\zeta}{\partial W} = \left[ \frac{\partial\zeta}{\partial W_0}, \frac{\partial\zeta}{\partial W_1}, \dots, \frac{\partial\zeta}{\partial W_{L-1}} \right]^T \quad (2.19)$$

$$= 2RW - 2P \quad (2.20)$$

As matrizes  $R$  e  $P$  são como definidas na seção anterior. A condição necessária para a obtenção do mínimo é que  $\nabla(\zeta) = [0, 0, \dots, 0] = 0$ . Essa condição é também suficiente porque a função é uma forma quadrática, e portanto convexa. Assim, chega-se à equação:

$$2RW - 2P = 0 \quad (2.21)$$

Assim, se designarmos por  $W^*$  o vetor ótimo, segue-se que;

$$W^* = R^{-1}P \quad (2.22)$$

A equação definida pela Equação 2.22 são conhecidas como equações de Wiener-Hopf. Agora, substituindo o valor da solução  $W^*$  na Equação 2.18, conclui-se que o erro médio quadrático mínimo é dado pela expressão:

$$\begin{aligned} \zeta_{\min} &= E [d_n^2] + (W^*)^T R W^* - 2P^T W^* \\ &= E [d_n^2] + (R^{-1}P)^T R R^{-1}P - 2P^T R^{-1}P \end{aligned} \quad (2.23)$$

## 2.4 Aplicações

Cancelamento de ruído adaptativo é quando medidas de certos sinais e processos estão sendo coletadas, limitações físicas frequentemente não permitem que medidas precisas das quantidades de interesse sejam obtidas. Tipicamente, o sinal de interesse é linearmente somado com outros ruídos estranhos no processo de medida, introduzindo erros inaceitáveis nas medidas. Porém, caso alguma medida deste ruído possa ser obtida em alguma parte do sistema, um filtro adaptativo poderá ser utilizado para determinar a relação entre o ruído cujo a componente deste ruído contida no sinal de interesse. Pois, o filtro adaptativo processa o sinal produzindo o sinal de saída, que por sua vez, acompanha as características do ruído contido. Desta maneira, o sinal de erro gerado é o próprio sinal de interesse sem o indesejável ruído.

## Capítulo 3

# Método Kernel Usando o Espaço de Hilbert

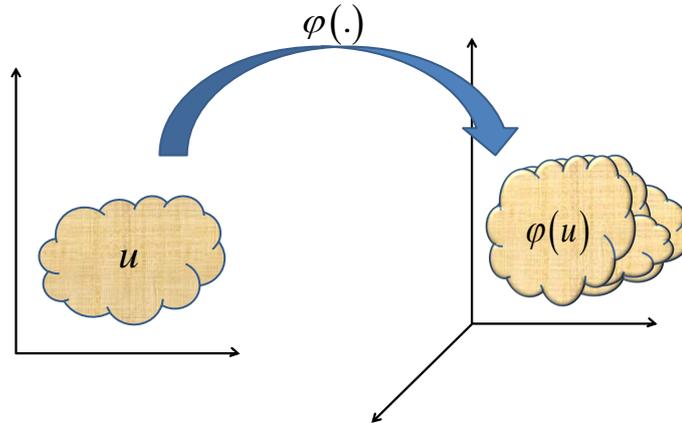
### 3.1 Definições e propriedades do Kernel

O método de kernel aborda o mapeamento dos dados em uma alta dimensionalidade no espaço de característica, em que cada coordenada corresponde a uma característica dos itens de dados, transformando os dados para um conjunto de pontos num espaço Euclidiano. Ao usar funções kernel, tem-se a possibilidade de trabalhar em um espaço de características, que contenham funções não lineares de aproximação e os produtos internos entre os dados no espaço de características que serve para extrair métricas [7]. Esta operação tem-se o custo computacional menor e mais rápido. Essa abordagem é chamada "truque" do kernel [13].

As funções de kernel têm a finalidade de projetar os vetores de características de entrada em um espaço de características de alta dimensão para classificação de problemas que se encontram em espaços não linearmente separáveis. Isso é feito, pois à medida que se aumenta o espaço da dimensão do problema, aumenta também a probabilidade desse problema se tornar linearmente separável em relação a um espaço de baixa dimensão. Entretanto, para obter uma boa distribuição para esse tipo de problema é necessário um conjunto de treinamento com um elevado número de instâncias [14] .

A Fig. 3.1 mostra o processo de transformação de um domínio não linearmente separável, em um problema linearmente separável através do aumento da dimensão, onde é feito um mapeamento por uma função de kernel  $\mathbb{F}(u)$ .

Algoritmos capazes de operar com kernel incluem máquina de vetor de suporte (SVM) [15], processos de Gauss, análise discriminante linear (LDA), análise de componentes prin-



**Figura 3.1:** Ilustração da construção do Espaço kernel com característica.

cipais (PCA), análise de correlação canônica, agrupamento espectral, filtros adaptativos lineares e muitos outros [6].

A maioria dos métodos baseados em kernel apenas processa matrizes quadradas, as quais são simétricas e definidas positivas. Iniciamos esta seção com algumas definições básicas para compreendermos o conceito de kernel definido positivo [14] [16].

**Definição 1** (Matriz Gram). Dada uma função  $\kappa : U^2 \rightarrow \mathbb{K}$  ( $\mathbb{K} = \mathbb{C}$  ou  $\mathbb{K} = \mathbb{R}$ ) é um conjunto de padrões  $u_1, u_2, \dots, u_n \in U$ , a matriz  $[K]_{n \times n}$  com elementos [17];

$$[K]_{ij} = \kappa(u_i, u_j) \quad (3.1)$$

é denominada matriz Gram ou (matriz kernel) para  $\kappa$  com relação aos elementos  $u_1, u_2, \dots, u_n$ .

**Definição 2** (Matriz definida positiva). Dada uma matriz complexa  $K_{n \times n}$  simétrica (hermitiana, rep.) a forma quadrática é definida como  $Q_R(u) := u^T R u$ , para  $u \in \mathbb{R}^n$  (resp.  $\mathbb{C}$ ). Se  $\forall u \neq 0, Q_R(u) > 0$ , então a forma quadrática é chamada positiva definida.

$$\sum_{j=1}^n \sum_{i=1}^n s_i \bar{s}_j [K]_{n \times n} \geq 0 \quad (3.2)$$

seja  $s_i, s_j \in \mathbb{R}(\mathbb{C})$  e  $\bar{s}_j$  denota o complexo conjugado de  $s_j$  no caso de  $s_i$  e  $s_j$  reais isso não tem efeito [17]. Note que a matriz simétrica é definida positiva se e somente se todos os seus autovalores são positivos [18].

**Definição 3** (Kernel definido positivo). Uma função  $\kappa : U \times U \rightarrow \mathbb{R}$  é chamada kernel definido positivo se para todo  $n > 0$  e todo  $u_1, u_2, \dots, u_n \in U$  a matriz Gram  $K$  gerada a

partir do par de medidas  $[K]_{ij} = \kappa(u_i, u_j)$  é definida positiva [14].

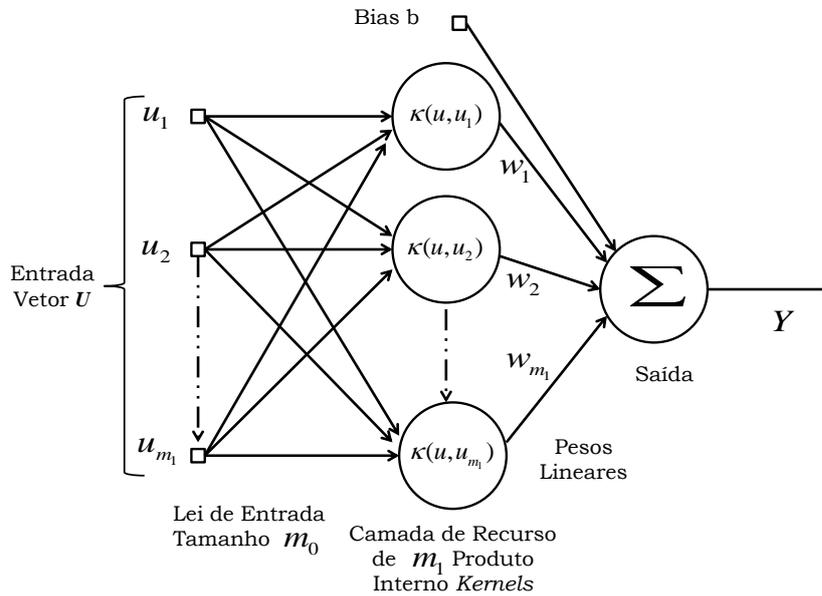


Figura 3.2: Definições e propriedades da estrutura do Kernel

### 3.1.1 A Origem do kernel

Segundo Schölkopf em [7], o termo kernel vem de um primeiro uso para um tipo de função em um corpo de operadores integrais tais como os estudados por Hilbert. Uma função  $\kappa$  é definida a partir do operador  $T_\kappa$  dado por [17]:

$$(T_\kappa f)(u) = \int_U \kappa(u, u') f(u') du' \quad (3.3)$$

e é denominado kernel de  $T_\kappa$ .

Diferentes termos são utilizados para denominar os kernels definidos positivos, tais como kernel reprodutivo, kernel de Mercer, kernel admissível, kernel de suporte vetorial, kernel definido não negativo e função de covariância [7].

**Teorema 3.1.1** (Mercer). *Seja  $K$  uma função simétrica contínua definida sobre um subconjunto compacto  $U$ . Se para toda função  $g(u)$  definida sobre  $U$ , tal que [7]:*

$$\int_U g(u)^2 du < \infty \quad (3.4)$$

é válido que,

$$\int_{U \times U} K(u, u') g(u) g(u') du du' \geq 0 \quad (3.5)$$

então há uma expansão de  $K$  numa série uniformemente convergente:

$$K(u, u') = \sum_{n=1}^{\infty} \lambda_n \Phi(u) \Phi(u'). \quad (3.6)$$

Uma propriedade interessante do kernel definido positivo é: como se está livre para escolher pontos nos quais o kernel é avaliado, temos que, para toda escolha de pontos, o kernel definido positivo sempre induz a uma matriz Gram definida positiva [7].

Os kernels podem ser considerados como produto interno. Certamente, todo produto interno é um kernel, mas a linearidade nos argumentos, que é uma propriedade padrão de produto interno, não se aplica sobre os kernels gerais. Entretanto, uma outra propriedade de produtos internos, a desigualdade de Cauchy-Schwarz [19], tem uma generalização para kernels [7]:

**Proposição 3.1.2** (Desigualdade de Cauchy-Schwarz para kernels). *Se  $\kappa$  é um kernel definido positivo, dados  $u, u' \in U$ , então:*

$$|\kappa(u, u')|^2 \leq \kappa(u, u) \kappa(u', u') \quad (3.7)$$

A demonstração da Proposição pode ser vista em [7].

### 3.1.2 Produto interno aplicado no Kernel

Dado  $\varphi$  um mapeamento no espaço das funções definidas de  $U$  em  $\mathbb{R}$ . Sendo que este espaço de funções é denotado por  $\mathbb{R}^U := \{f : U \rightarrow \mathbb{R}\}$  [7] [18]:

$$\begin{aligned} \varphi : U &\longrightarrow \mathbb{R}^U \\ u &\longmapsto \kappa(\cdot, u) \end{aligned} \quad (3.8)$$

em que  $\varphi(u)$  denota a função que assume o valor  $\kappa(u, u')$  para um dado  $u' \in U$  fixo, isto é,  $\varphi(u)(\cdot) = \kappa(\cdot, u)$  [7]. Para cada padrão de  $u$  é transformado em uma função do domínio  $U$  sendo a imagem em  $\mathbb{R}$ . Seguem então as etapas para a construção do espaço das características associado a  $\varphi$ :

- Transformar a imagem de  $\varphi(\cdot)$  em um espaço vetorial;

- Definir um produto interno, isto é, uma forma bilinear estritamente definida positiva neste espaço vetorial ;
- Mostrar que o produto interno satisfaz  $\kappa(u, u') = \langle \varphi(u), \varphi(u') \rangle$ .

Pode-se definir o produto interno das imagens de todos os padrões  $u$  em  $\varphi(\cdot)$ , e também deve-se definir o espaço vetorial, mas não significa que  $\varphi(\cdot)$  seja um vetor no sentido comum de elementos do um  $\mathbb{R}^p$ , mas sim no sentido mais amplo, como elemento de um espaço vetorial como apresentado em [20] . Isso pode ser feito tomando as combinações lineares dos elementos de  $\mathbb{R}^U$  [7]:

$$f(\cdot) = \sum_{i=1}^n \alpha_i \kappa(\cdot, u_i) \quad (3.9)$$

em que  $n \in N, \alpha_i \in \mathbb{R}$  e  $u_1, u_2, \dots, u_n \in U$ , quaisquer. Agora, é necessário definir o produto interno entre  $f$  e uma outra função qualquer do espaço,

$$g(\cdot) = \sum_{j=1}^{n'} \beta_j \kappa(\cdot, u'_j) \quad (3.10)$$

em que,  $n' \in N, \beta_j \in \mathbb{R}$  e  $u'_1, u'_2, \dots, u'_{n'} \in U$ . Note que para elementos  $f, g \in \mathbb{R}^U$  e  $\alpha_i, \beta_j \in \mathbb{R}$  o espaço  $U$  é fechado para soma e para o produto por escalar  $\alpha \in \mathbb{R}$ . Logo, é fácil ver que o espaço gerado pelas combinações lineares de elementos de  $\mathbb{R}^U$  é um espaço vetorial. Agora pode-se definir o produto no espaço das combinações lineares de  $\mathbb{R}^U$  como [7]:

$$\langle f, g \rangle := \sum_{i=1}^n \sum_{j=1}^{n'} \alpha_i \beta_j \kappa(u_i, u'_j) \quad (3.11)$$

Esta expressão contém explicitamente os coeficientes da expansão, a qual não necessita ser única. Isto pode ser visto fazendo:

$$\begin{aligned} \langle f, g \rangle &= \sum_{j=1}^{n'} \beta_j \sum_{i=1}^n \alpha_i \kappa(u_i, u'_j) = \\ \langle f, g \rangle &= \sum_{j=1}^{n'} \beta_j \sum_{i=1}^n \alpha_i \kappa(u'_j, u_i) = \\ \langle f, g \rangle &= \sum_{j=1}^{n'} \beta_j f(u'_j) \end{aligned} \quad (3.12)$$

Assim, pode-se dizer que a operação resulta em elementos do próprio espaço vetorial das funções  $\mathbb{R}^U$ , já que pode ser reescrita como combinação linear de elementos deste espaço. Verificar-se então se o produto interno no espaço  $\mathbb{R}^U$ . Note que as igualdades são respectivamente da simetria do kernel. Sendo assim, não depende de uma expansão particular de  $f$ , pois similarmente pode-se escrever [5] [7]:

$$\langle f, g \rangle = \sum_{i=1}^n \alpha_i g(u_i)$$

Serão mostrados os itens que qualificam a operação  $\langle \cdot, \cdot \rangle$ , definida pela Equação 3.9 como produto interno de um espaço de funções, dadas algumas propriedades de  $\varphi$  a seguir [7]:

- Definição dada pela Equação 3.9 tem-se que;

$$\begin{aligned} f(u) &= \sum_{i=1}^n \alpha \kappa(u, u_i) \\ &= \sum_{j=1}^1 \sum_{i=1}^n 1 \cdot \alpha \kappa(u, u_i) \\ &= \langle \kappa(\cdot, u), f \rangle \end{aligned} \tag{3.13}$$

Portanto,  $\kappa$  pode ser analisado como um representante de  $f$  em  $u$ .

- Constituindo uma particularidade da função  $f(\cdot, u') = \kappa(\cdot, u')$ , tem-se que;

$$\begin{aligned} f(\cdot) &= \langle \kappa(\cdot, u), \kappa(\cdot, u') \rangle \\ &= \sum_{i=1}^1 \sum_{j=1}^n 1 \cdot 1 \cdot \kappa(u, u') \\ &= \kappa(u, u') \end{aligned} \tag{3.14}$$

## 3.2 O "truque" do kernel

O truque do Kernel é uma ferramenta matemática que pode ser aplicada a qualquer algoritmo que depende apenas do produto escalar entre os dois vetores. Quando um produto escalar é usado, ele é substituído por uma função kernel. Se aplicado corretamente,

esses algoritmos lineares são transformados em algoritmos não lineares (com pouco esforço). No entanto, os termos são utilizados, na função  $\varphi$ , cuja natureza dos vetores de entrada não tem restrições e pode ser definido entre qualquer tipo de estrutura [7].

Dado um algoritmo que seja formulado em termos de um kernel definido positivo, podemos construir um algoritmo alternativo trocando  $\kappa$  por outro kernel definido positivo  $\tilde{\kappa}$ , com aplicação do truque do kernel será quando  $\kappa(.,.)$  é um produto interno no domínio da entrada de dados. Nesse caso o algoritmo original opera como o produto interno entre  $\{\varphi(u_1), \varphi(u_2), \dots, \varphi(u_n)\}$  para  $n \in N$ . Com isso, o algoritmo obtido pela troca de  $\kappa(.,.)$  por  $\tilde{\kappa}(.,.)$  será o mesmo, só que este passará a operar vetores  $\{\tilde{\varphi}(u_1), \tilde{\varphi}(u_2), \dots, \tilde{\varphi}(u_n)\}$  para algum  $n \in N$ . Entretanto, a aplicação do truque do kernel não se limita a esse caso, pois  $\kappa(.,.)$  e  $\tilde{\kappa}(.,.)$  pode ser ambos kernels não lineares [7].

### 3.2.1 Kernel Polinomial

O kernel polinomial é um kernel não estacionário para os problema que tenham os dados de treinamento normalizado.

$$\kappa(u, u') = (\alpha u^T u' + c)^d \quad (3.15)$$

Analizamos os parâmetros ajustáveis que é dado pela inclinação de  $\alpha$ , com o termo constante  $c$  e  $d$  grau do polinômio .

### 3.2.2 Kernel Gaussiano

O kernel gaussiano é um exemplo de função de base radial do kernel

$$\kappa(u, u') = \exp\left(-\frac{\|u - u'\|^2}{2\sigma^2}\right). \quad (3.16)$$

Alternativamente, poderia também ser implementada usando.

$$\kappa(u, u') = \exp\left(-\gamma\|u - u'\|^2\right) \quad (3.17)$$

O parâmetro de  $\gamma$  tem um papel importante na performance do kernel, e deve ser cuidadosamente analisado. Se superestimada, a exponencial vai se comportar quase linearmente e a projeção de dimensão superior vai começar a perder o seu poder não linear.

Por outro lado, se subestimada, a função faltará regularização e a fronteira de decisão será altamente sensível ao ruído nos dados de treino.

### 3.2.3 Kernel Exponencial

O kernel exponencial é estreitamente relacionado com o núcleo gaussiano, com apenas o quadrado da norma deixado de fora. É também uma base radial.

$$\kappa(u, u') = \exp\left(-\frac{\|u - u'\|^2}{2\sigma^2}\right). \quad (3.18)$$

### 3.2.4 Kernel Laplaciano

É completamente equivalente ao kernel exponencial, exceto por ser menos sensível a mudanças no parâmetro  $\sigma$ , equivalente, também é um kernel da função de base radial.

$$\kappa(u, u') = \exp\left(-\frac{\|u - u'\|}{\sigma}\right). \quad (3.19)$$

# Capítulo 4

## Algoritmos Adaptativos

Inicia-se com o algoritmo dos mínimos quadrados recursivo (RLS), em seguida usa-se o mapeamento do kernel no RLS. Contudo será explorada a relação em álgebra matricial conhecido como o lema da matriz inversa cujo desenvolvimento do algoritmo KRLS foi aplicado[21]. O objetivo de rever os principais passos do desenvolvimento do algoritmo baseado em uma potência par do erro [22], inspirado no algoritmo RLS padrão. Chama-se esse algoritmo de recursivo não quadrático (RNQ), para o qual escolhe-se uma função baseada em uma única potência par do erro com critério a ser minimizado. O objetivo é determinar um algoritmo que ajuste os pesos do combinador linear adaptativo.

### 4.1 O algoritmo Recursivo dos Mínimos Quadrados (Recursive Least Square-RLS)

Dado uma sequência de treinamento de dados que inclui o momento  $i - 1$  no algoritmo recursivo dos mínimos quadrados sendo que a estimativa dos pesos  $\mathbf{w}_{i-1}$  é a minimização da função de custo encontrado em;

$$\min_{\mathbf{w}} \sum_{j=1}^i |d_j - \mathbf{u}_j^T \mathbf{w}|^2 \quad (4.1)$$

onde,  $\mathbf{u}_j$  é o regressor de entrada  $L \times L$  e  $d_j$  é a resposta desejada [21]. Denotada;

$$\begin{aligned} \mathbf{d}_{i-1} &= [d_1, \dots, d_{i-1}]^T \\ \mathbf{U}_{i-1} &= [\mathbf{u}_1, \dots, \mathbf{u}_{i-1}]_{L \times (i-1)} \end{aligned} \quad (4.2)$$

A solução da equação é dado por;

$$\mathbf{w}_{i-1} = (\mathbf{U}_{i-1} \mathbf{U}_{i-1}^T)^{-1} \mathbf{U}_{i-1} \mathbf{d}_{i-1} \quad (4.3)$$

Quando um novo par de entradas e saída  $\mathbf{u}_i, d_i$  antes de validar a estimativa de peso  $\mathbf{w}_i$ , com minimização de; [23]

$$\min_{\mathbf{w}} \sum_{j=1}^i |d_j - \mathbf{u}_j^T \mathbf{w}|^2 \quad (4.4)$$

é calculado como;

$$\mathbf{w}_i = (\mathbf{U}_i \mathbf{U}_i^T)^{-1} \mathbf{U}_i \mathbf{d}_i \quad (4.5)$$

onde;

$$\mathbf{U}_i = [\mathbf{u}_{i-1}, \mathbf{u}_i]_{L \times (i-1)} \quad (4.6)$$

e

$$\mathbf{d}_i = [\mathbf{d}_{i-1}^T, d_i]^T \quad (4.7)$$

Define as matrizes;

$$\mathbf{P}_i = (\mathbf{U}_i \mathbf{U}_i^T)^{-1} \quad (4.8)$$

e

$$\mathbf{P}_{i-1} = (\mathbf{U}_{i-1} \mathbf{U}_{i-1}^T)^{-1} \quad (4.9)$$

Observa-se;

$$\mathbf{P}_i^{-1} = \mathbf{P}_{i-1}^{-1} + \mathbf{u}_i \mathbf{u}_i^T \quad (4.10)$$

### 4.1.1 O algoritmo RLS ponderado exponencialmente

Quando, usa-se o lema da matriz inversa [23];

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} - \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1} \quad (4.11)$$

com a identificação;

$$\begin{aligned} \mathbf{P}_{i-1} &\rightarrow \mathbf{A}, \\ \mathbf{u}_i &\rightarrow \mathbf{B}, \\ 1 &\rightarrow \mathbf{C}, \\ \mathbf{u}_i^T &\rightarrow \mathbf{D} \end{aligned} \quad (4.12)$$

É necessário escolher a atualização de  $\mathbf{P}_i$  diretamente de  $\mathbf{P}_{i-1}$  [23]:

$$\mathbf{P}_i = \left[ \mathbf{P}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{P}_{i-1}}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} \right] \quad (4.13)$$

$$\mathbf{w}_i = \mathbf{P}_i\mathbf{U}_i\mathbf{d}_i \quad (4.14)$$

$$\begin{aligned} &= \left[ \mathbf{P}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{P}_{i-1}}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} \right] [\mathbf{U}_{i-1}\mathbf{d}_{i-1} + \mathbf{u}_i d_i] \\ &= \mathbf{P}_{i-1}\mathbf{U}_{i-1}\mathbf{d}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{P}_{i-1}}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i}\mathbf{P}_{i-1}\mathbf{U}_{i-1}\mathbf{d}_{i-1} + \mathbf{P}_{i-1}\mathbf{u}_i d_i - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i d_i}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} \\ &= \mathbf{w}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{w}_{i-1}}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} + \left[ \mathbf{P}_{i-1}\mathbf{u}_i d_i - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i d_i}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} \right] \\ &= \mathbf{w}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{w}_{i-1}}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} + \frac{\mathbf{P}_{i-1}\mathbf{u}_i d_i}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} \\ \mathbf{w}_i &= \mathbf{w}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{u}_i\mathbf{u}_i^T\mathbf{w}_{i-1}}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} [d_i - \mathbf{u}_i^T\mathbf{w}_{i-1}] \end{aligned}$$

Que é;

$$\mathbf{w}_i = \mathbf{w}_{i-1} - \frac{\mathbf{P}_{i-1}\mathbf{u}_i}{1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i} [d_i - \mathbf{u}_i^T\mathbf{w}_{i-1}] \quad (4.15)$$

Definido;

$$r_i = 1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i \quad (4.16)$$

$$\mathbf{k}_i = \mathbf{P}_{i-1}\mathbf{u}_i/r_i \quad (4.17)$$

### 4.1.2 Resumo do algoritmo RLS

Quando a apresentação RLS é resumida no Algoritmo 1 [23].

---

**Algoritmo 1** Algoritmo dos Mínimos Quadrados Recursivo (RLS)

---

- 1: Inicializar  
 $\mathbf{w}_0 = 0, \mathbf{P}_0$
- 2: Calculando
- 3: Interações de  $i \geq 1$

$$\begin{aligned} r_i &= 1 + \mathbf{u}_i^T\mathbf{P}_{i-1}\mathbf{u}_i \\ \mathbf{k}_i &= \mathbf{P}_{i-1}\mathbf{u}_i / r_i \\ e_i &= d_i - \mathbf{u}_i^T\mathbf{w}_{i-1} \\ \mathbf{w}_i &= \mathbf{w}_{i-1} + \mathbf{k}_i e_i \\ \mathbf{P}_i &= [\mathbf{P}_{i-1} - \mathbf{k}_i\mathbf{k}_i^T r_i] \end{aligned}$$


---

## 4.2 Algoritmo Kernel dos Mínimos Quadrados Recursivo (Kernel Recursive Least Square-KRLS)

Inicialmente deriva-se o RLS reproduzindo o kernel no espaço de Hilbert (RKHS) [21] [24] que é resumido o algoritmo KRLS no Algoritmo 2. Para isso, precisamos usar o teorema de Mercer para transformar o  $u(i)$ , que são os dados de entrada para a função de espaço  $\mathbb{F}$   $\varphi(\mathbf{u}(i))$  [denotado como  $\varphi(i)$ ] [23]. Organizamos e formulamos o algoritmo dos mínimos quadrados recursivo no exemplo na sequência  $\{d(1), d(2), \dots\}$  e  $\{\varphi(1), \varphi(2), \dots\}$  [25]. Para cada iteração, os vetores de peso  $\boldsymbol{\omega}(i)$  [23].

$$\min_{\boldsymbol{\omega}} \sum_{j=1}^i |d(j) - \boldsymbol{\omega}^T \boldsymbol{\varphi}(j)|^2 + \lambda \|\boldsymbol{\omega}\|^2 \quad (4.18)$$

Que precisa ser solucionado de maneira que seja na forma recursiva igual a Equação 4.17. O RKHS é um espaço com alta dimensionalidade, com isso a regularização do espaço é necessária para atualizar os pesos [24] [25]. Assim, que a dimensionalidade de  $\boldsymbol{\varphi}(j)$  é tão grande que pode ser adicionado no mesmo espaço), com isso a Equação 4.17, não pode ser aplicado diretamente nesse caso [23].

Ao inicializar as definições tem-se que;

$$\begin{aligned} \mathbf{d}(i) &= [d(1), \dots, d(i)]^T \\ \boldsymbol{\Phi}(i) &= [\boldsymbol{\varphi}(1), \dots, \boldsymbol{\varphi}(i)] \end{aligned} \quad (4.19)$$

assim;

$$\boldsymbol{\omega}(i) = [\lambda \mathbf{I} + \boldsymbol{\Phi}(i) \boldsymbol{\Phi}(i)^T]^{-1} \boldsymbol{\Phi}(i) \mathbf{d}(i). \quad (4.20)$$

Usando o lema da matriz inversar na Equação 4.11 com as identificações;

$$\begin{aligned} \lambda \mathbf{I} &\rightarrow \mathbf{A}, \\ \boldsymbol{\Phi}(i) &\rightarrow \mathbf{B}, \\ \mathbf{I} &\rightarrow \mathbf{C}, \\ \boldsymbol{\Phi}(i)^T &\rightarrow \mathbf{D} \end{aligned} \quad (4.21)$$

é fácil observar que;

$$[\lambda \mathbf{I} + \boldsymbol{\Phi}(i) \boldsymbol{\Phi}(i)^T]^{-1} \boldsymbol{\Phi}(i) = \boldsymbol{\Phi}(i) [\lambda \mathbf{I} + \boldsymbol{\Phi}(i)^T \boldsymbol{\Phi}(i)]^{-1}$$

Substituindo este resultado na Equação 4.20, obtém-se;

$$\boldsymbol{\omega}(i) = \boldsymbol{\Phi}(i) [\lambda \mathbf{I} + \boldsymbol{\Phi}(i)^T \boldsymbol{\Phi}(i)]^{-1} \mathbf{d}(i) \quad (4.22)$$

Agora apresenta-se a mudança da Equação 4.20 em 4.22. Seja,  $\boldsymbol{\Phi}(i)^T \boldsymbol{\Phi}(i)$  que é calculado pelo truque do kernel e em segundo lugar, o peso é mostrado explicitamente como

uma combinação linear dos dados de entrada [21]:

$$\boldsymbol{\omega}(i) = \boldsymbol{\Phi}(i) \mathbf{a}(i)$$

com;

$$\mathbf{a}(i) = \left[ \lambda \mathbf{I} + \boldsymbol{\Phi}(i)^T \boldsymbol{\Phi}(i) \right]^{-1} \mathbf{d}(i)$$

Pode-se substituir por;

$$\mathbf{Q}(i) = \left[ \lambda \mathbf{I} + \boldsymbol{\Phi}(i)^T \boldsymbol{\Phi}(i) \right]^{-1} \quad (4.23)$$

Também pode-se observar que;

$$\mathbf{Q}(i)^{-1} = \begin{bmatrix} \mathbf{Q}(i-1)^T & \mathbf{h}(i) \\ \mathbf{h}(i)^T & \lambda + \boldsymbol{\varphi}(i)^T \boldsymbol{\varphi}(i) \end{bmatrix} \quad (4.24)$$

Em que  $\mathbf{h}(i) = \boldsymbol{\Phi}(i-1)^T \boldsymbol{\varphi}(i)$ .

$$\mathbf{Q}(i) = \begin{bmatrix} \mathbf{Q}(i-1) r(i) + \mathbf{z}(i) \mathbf{z}(i)^T & -\mathbf{z}(i) \\ -\mathbf{z}(i)^T & 1 \end{bmatrix} \quad (4.25)$$

onde;

$$\begin{aligned} \mathbf{z}(i) &= \mathbf{Q}(i-1) \mathbf{h}(i) \\ r(i) &= \lambda + \boldsymbol{\varphi}(i)^T \boldsymbol{\varphi}(i) - \mathbf{z}(i)^T \mathbf{h}(i) \end{aligned} \quad (4.26)$$

#### 4.2.1 O algoritmo KRLS com bloco de inversão de matriz

Na Equação 4.25 pode ser estabelecida usando o bloco de matriz de inversão [23];

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix} \quad (4.27)$$

em que  $\mathbf{A}$  e  $\mathbf{D}$  são matrizes quadradas.

Portanto, os coeficientes de expansão do peso na Equação 4.22;

$$\begin{aligned} \mathbf{a}(i) &= \mathbf{Q}(i) \mathbf{d}(i) \\ \mathbf{a}(i) &= \begin{bmatrix} \mathbf{Q}(i-1) + \mathbf{z}(i) \mathbf{z}(i)^T r(i)^{-1} & -\mathbf{z}(i) r(i)^{-1} \\ -\mathbf{z}(i)^T r(i)^{-1} & r(i)^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{d}(i-1) \\ d(i) \end{bmatrix} \\ \mathbf{a}(i) &= \begin{bmatrix} \mathbf{a}(i-1) + \mathbf{z}(i) r(i)^{-1} e(i) \\ r(i)^{-1} e(i) \end{bmatrix} \end{aligned} \quad (4.28)$$

em que  $e(i)$  é o sinal de erro cuja predição é calculado pela diferença entre o sinal desejado de predição com  $f_{i-1}(\mathbf{u}(i))$ :

$$f_{i-1}(\mathbf{u}(i)) = \mathbf{h}(i)^T \mathbf{a}(i-1) = \sum_{j=1}^{i-1} \mathbf{a}(i-1) \kappa(\mathbf{u}(j), \mathbf{u}(i)) \quad (4.29)$$

$$e(i) = d(i) - f_{i-1}(\mathbf{u}(i)) \quad (4.30)$$

Resumindo, o KRLS assume uma base radial com estrutura de rede em função em cada iteração. O  $a_j(i-1)$  é o componente de  $a(i-1)$  na Equação 4.29. A partir da Equação 4.28, observa-se que o processo de aprendizagem dos KRLS em todos os coeficientes anteriores dado em  $-\mathbf{z}(i) r(i)^{-1} e(i)$ .

Denota-se  $f_i$  como a estimativa da entrada com o mapeamento de saída  $i$  iterações, com isso ocorre uma aprendizagem sequencial após seguir o KRLS:

$$f_i = f_{i-1} + r(i)^{-1} \left[ \kappa(\mathbf{u}(i), \cdot) - \sum_{j=1}^{i-1} \mathbf{z}_j(i) \kappa(\mathbf{u}(j), \cdot) \right] e(i) \quad (4.31)$$

Os coeficientes  $a(i)$ , com os centros  $C(i)$  devem ser armazenados em um computador durante a inicialização do processo. As atualizações necessárias para o KRLS com iteração  $i$  são;

$$\mathbf{a}_i(i) = r(i)^{-1} e(i) \quad (4.32)$$

$$\mathbf{a}_j(i) = \mathbf{a}_j(i-1) - r(i)^{-1} e(i) \mathbf{z}_j(i), j = 1, \dots, i-1 \quad (4.33)$$

$$C(i) = \{C(i-1), u(i)\} \quad (4.34)$$

## 4.2.2 Resumo do algoritmo KRLS

O tempo e as complexidades de espaço são ambos  $O(i^2)$ , cuja complexidade também é reduzida para  $O(m_i^2)$ , em que  $m_i$  é o número efetivos de centros na rede no momento  $i$ .

---

### Algoritmo 2 Algoritmo Kernel Mínimos Quadrados Recursivo (KRLS)

---

1: Inicializar

$$\mathbf{Q}(1) = (\lambda + \kappa(\mathbf{u}(1), \mathbf{u}(1)))^{-1}, \mathbf{a}(1) = \mathbf{Q}(1) d(1)$$

2: Calculando

3: Interações  $i > 1$

$$\begin{aligned} \mathbf{h}(i) &= [\kappa(\mathbf{u}(i), \mathbf{u}(1)), \dots, \kappa(\mathbf{u}(i), \mathbf{u}(i-1))]^T \\ \mathbf{z}(i) &= \mathbf{Q}(i-1) \mathbf{h}(i) \\ r(i) &= \lambda + \kappa(\mathbf{u}(i), \mathbf{u}(i)) - \mathbf{z}(i) \mathbf{z}(i)^T \mathbf{h}(i) \\ \mathbf{Q}(i) &= r(i)^{-1} \begin{bmatrix} \mathbf{Q}(i-1) r(i) + \mathbf{z}(i) \mathbf{z}(i)^T & -\mathbf{z}(i) \\ -\mathbf{z}(i)^T & 1 \end{bmatrix} \\ e(i) &= d(i) - \mathbf{h}(i)^T \mathbf{a}(i-1) \\ \mathbf{a}(i) &= \begin{bmatrix} \mathbf{a}(i-1) + \mathbf{z}(i) r(i)^{-1} e(i) \\ r(i)^{-1} e(i) \end{bmatrix} \end{aligned}$$


---

## 4.3 Algoritmo RNQ baseado em uma única potência par do erro

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  $\mathbf{w}_n = [w_{0,n}, w_{1,n}, \dots, w_{L-1,n}]^T$ . Deve-se recuperar  $d_i$  estimando o sinal de saída  $y_i = \mathbf{w}_n^T \mathbf{u}_i$ , depois de calcular o erro  $e_i = d_i - y_i$  [22]. Sendo  $n$  o número de interação,  $1 \leq i \leq n$  e  $L$  a ordem do filtro. Para desenvolver o algoritmo RNQ baseado em uma potência par de erros, utiliza-se como função de custo uma função par, contínua e simétrica, representada por;

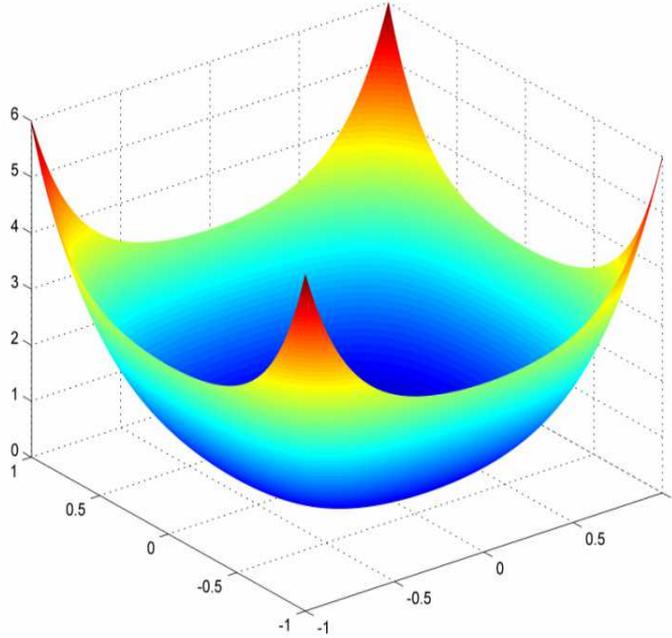
$$J_n = \sum_{i=1}^n \left[ \lambda^{n-i} (e_i)^{2j} \right] \quad (4.35)$$

sendo que  $j$  e  $n$  são inteiros positivos, com  $j > 1$  e  $0 \ll \lambda < 1$ ,  $\lambda$  é o fator peso exponencial ou fator de esquecimento. Pode-se observar que, se  $j = 1$  obtemos o mesmo critério (função de custo) utilizado no algoritmo RLS padrão, deduzido no capítulo anterior [22].

Objetivando encontrar o peso ótimo, deve-se calcular o gradiente da função  $J_n$ . Assim, derivando a equação em relação a  $\mathbf{w}$ , obtém-se;

$$J_n = \sum_{i=1}^n \lambda^{n-i} (e_i)^{2j} \quad (4.36)$$

onde  $e_i = d_i - \mathbf{w}_n^T \mathbf{u}_i$ , cujo  $\mathbf{w}_n^T \mathbf{u}_i = y_i$ .



**Figura 4.1:** Superfície RNQ tridimensional, juntamente com alguns contornos. O erro quadrático médio está plotado na vertical,  $w_0$  e  $w_1$  variam de -1 a 1.

$$\nabla J_n = \sum_{i=1}^n \lambda^{n-i} (-\mathbf{u}_i) 2j e_i^{2j-1} \quad (4.37)$$

$$\nabla J_n = -2j \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i e_i^{2j-1} \quad (4.38)$$

$$\nabla J_n = -2j \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i e_i^{2j-2} \cdot \underbrace{e_i}_{d_i - \mathbf{w}_n^T \mathbf{u}_i} \quad (4.39)$$

$$\nabla J_n = -2j \sum_{i=1}^n \lambda^{n-i} (d_i \mathbf{u}_i - \mathbf{u}_i \mathbf{w}_n^T \mathbf{u}_i) e_i^{2j-2} \quad (4.40)$$

$$\nabla J_n = -2j \sum_{i=1}^n \lambda^{n-i} (d_i \mathbf{u}_i - \mathbf{u}_i \mathbf{u}_i^T \mathbf{w}_n) e_i^{2j-2} \quad (4.41)$$

$$\nabla J_n = -2j \sum_{i=1}^n \lambda^{n-i} d_i \mathbf{u}_i e_i^{2j-2} + 2j \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i \mathbf{u}_i^T \mathbf{w}_n e_i^{2j-2} \quad (4.42)$$

Igualando  $\nabla J_n$  a zero, com o intuito de encontrar o valor mínimo, o valor ótimo do vetor peso  $\hat{\mathbf{w}}_n$ , tem-se em forma matricial;

$$\nabla J_n = 0$$

fazendo,

$$\Phi_n = \sum_{i=1}^n \lambda^{n-i} \mathbf{u}_i \mathbf{u}_i^T e_i^{2j-2} \quad (4.43)$$

$$\mathbf{Z}_n = \sum_{i=1}^n \lambda^{n-i} d_i \mathbf{u}_i e_i^{2j-2} \quad (4.44)$$

pode-se reescrever a Equação 4.42 como:

$$\Phi_n \hat{\mathbf{w}}_n = \mathbf{Z}_n \quad (4.45)$$

Multiplicando ambos os membros da Equação (6.6) por  $\Phi_n^{-1}$ , tem-se:

$$\Phi_n^{-1} \Phi_n \hat{\mathbf{w}}_n = \Phi_n^{-1} \mathbf{Z}_n \quad (4.46)$$

Assim, visto que  $\mathbf{I} = \Phi_n^{-1} \Phi_n$ , tem o vetor de correlação cruzada  $\mathbf{Z}_n$ , entre as entradas e a resposta desejada de dimensão  $L \times 1$ . E a matriz de correlação do vetor de entrada de dimensão  $L \times L$  definida por  $\Phi_n$  [22]. Então dessa forma, pode-se reescrever a Equação 4.46 como,

$$\hat{\mathbf{w}}_n = \Phi_n^{-1} \mathbf{Z}_n. \quad (4.47)$$

Assim, o valor ótimo do vetor peso,  $\hat{\mathbf{w}}_n$ , para que a função na Equação 4.35 atinja o valor mínimo é definido pela equação normal escrita em forma matricial. Agora, para encontrar uma regra de atualização para o peso  $\mathbf{w}$ , adotou-se o seguinte desenvolvimento [22]:

Primeiro, isola-se para obter o termo correspondente a  $i = n$  pode-se reescrever;

$$\Phi_n = \sum_{i=1}^n \lambda^{n-i} e_i^{2j-2} \mathbf{u}_i \mathbf{u}_i^T \quad (4.48)$$

Analogamente pode-se reescrever a Equação 4.48 como;

$$\mathbf{Z}_n = \sum_{i=1}^n \lambda^{n-i} e_i^{2j-2} d_i \mathbf{u}_i \quad (4.49)$$

sendo  $a_j = k^{n-i}$  e  $\alpha_j = 2j - 2$ , cujo  $k = 1$ .

Logo,

$$\Phi_n = \lambda \Phi_{n-1} + (a_j e_n^{\alpha_j}) \mathbf{u}_n \mathbf{u}_n^T \quad (4.50)$$

e

$$\mathbf{Z}_n = \lambda \mathbf{Z}_{n-1} + (a_j e_n^{\alpha_j}) d_n \mathbf{u}_n, \quad (4.51)$$

onde  $\Phi_{n-1}$  o valor anterior da matriz de autocorrelação e  $\mathbf{Z}_{n-1}$  o valor anterior do vetor de correlação cruzada [22]. Pode-se 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 [26]. Isto significa que ganha-se 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 [22].

### 4.3.1 O algoritmo RNQ ponderado exponencialmente

As deduções das Equações 4.50 e 4.51 foram feitas com o intuito de facilitar o cálculo da inversa da matriz de correlação  $\Phi_{n-1}$ , que é necessária para determinar o valor estimado,  $\hat{\mathbf{w}}_n$ , do vetor ótimo de acordo com a Equação 4.47. Como a matriz de autocorrelação

é positiva definida e não singular pode-se aplicar o lema de inversão de matrizes para Equação recursiva em 4.51. Inicialmente, fazem-se as seguintes identificações [22].

$$\begin{aligned}\mathbf{A} &= \Phi_n \\ \mathbf{B}^{-1} &= \lambda \Phi_{n-1} \\ \mathbf{C} &= \mathbf{u}_n \\ \mathbf{D}^{-1} &= a_j e_n^{\alpha_j}\end{aligned}$$

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{C}^{-1} - \mathbf{D}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{D}\mathbf{A}^{-1}$$

Sendo  $\mathbf{A}$  e  $\mathbf{B}$  matrizes definidas positivas de dimensão  $L \times L$ , tem-se;

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T$$

onde  $\mathbf{D}$  uma matriz positivamente definida de dimensão  $N \times L$  e  $\mathbf{C}$  uma matriz  $L \times N$  [26].

Assim,

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{B} \quad (4.52)$$

Substituindo a Equação 4.52 em 4.52, teremos:

$$\Phi_n^{-1} = \lambda^{-1}\Phi_{n-1}^{-1} - \frac{\lambda^{-1}\Phi_{n-1}^{-1}\mathbf{u}_n\mathbf{u}_n^T\Phi_{n-1}^{-1}}{\lambda(a_j e_n^{\alpha_j})^{-1} + \mathbf{u}_n^T\Phi_{n-1}^{-1}\mathbf{u}_n} \quad (4.53)$$

Por conveniência de notação, usa-se;

$$\mathbf{P}_n = \Phi_n^{-1} \quad (4.54)$$

e

$$g_n = \frac{\mathbf{P}_{n-1}\mathbf{u}_n}{\lambda(a_j e_n^{\alpha_j})^{-1} + \mathbf{u}_n^T\mathbf{P}_{n-1}\mathbf{u}_n} \quad (4.55)$$

sendo  $g_n$  o vetor ganho de dimensão  $L - 1$ .

Ao comparar este resultado com RLS, observa-se que o erro, que aparece no termo par, afeta diretamente o fator de esquecimento [22]. Pode-se considerar que a abordagem do fator de esquecimento com a informação de erro instantâneo, sendo que o ganho não depende apenas da dinâmica do sinal de entrada, mas a resposta desejada através do erro [22].

Assim,

$$\mathbf{P}_n = \lambda^{-1} (\mathbf{P}_{n-1} - g_n \mathbf{u}_n^T \mathbf{P}_{n-1}). \quad (4.56)$$

Para facilitar futuras manipulações matemáticas, reescrevemos a Equação 4.53 como segue-se;

$$g_n = \mathbf{P}_n \mathbf{u}_n (a_j e_n^{\alpha_j}) \quad (4.57)$$

### 4.3.2 Atualização do vetor peso

No desenvolvimento da equação recursiva utilizam-se as Equações 4.46, 4.48 e 4.52 para determinar a regra de atualização do vetor peso  $\hat{\mathbf{w}}_n$  no instante  $n$  como segue-se [22]:

$$\hat{\mathbf{w}}_n = \Phi_n^{-1} \mathbf{Z}_n$$

$$\hat{\mathbf{w}}_n = \lambda \mathbf{P}_n \mathbf{Z}_{n-1} + \mathbf{P}_n (a_j e_n^{\alpha_j}) d_n \mathbf{u}_n \quad (4.58)$$

$$\begin{aligned} \hat{\mathbf{w}}_n &= \mathbf{P}_n \mathbf{Z}_{n-1} - g_n \mathbf{u}_n^T \mathbf{P}_{n-1} \mathbf{Z}_{n-1} + (a_j e_n^{\alpha_j}) \mathbf{P}_n \mathbf{u}_n d_n \\ &= \hat{\mathbf{w}}_{n-1} - g_n \mathbf{u}_n^T \mathbf{P}_{n-1} \mathbf{Z}_{n-1} + (a_j e_n^{\alpha_j}) \mathbf{P}_n \mathbf{u}_n d_n \end{aligned} \quad (4.59)$$

Das Equações 4.56 e 4.57, obtemos a estrutura de atualização do algoritmo 3 RNQ dada;

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_{n-1} + g_n \xi_n \quad (4.60)$$

onde  $\xi_n$  é chamado de estimativa a priori do erro, ou seja;

$$\xi_n = d_n + \hat{\mathbf{w}}_{n-1}^T \mathbf{u}_n \quad (4.61)$$

As Equações 4.55, 4.56, 4.61 e 4.60 respectivamente, constituem o algoritmo RNQ baseado em uma potência par do erro .

### 4.3.3 Resumo do algoritmo RNQ

---

**Algoritmo 3** Algoritmo de Recursivo Não Quadrático (RNQ)

---

1: Inicializar

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

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

2: Calculando

3: Para cada instante de tempo  $n \geq 1$ ;

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

$$g_n = \frac{\mathbf{P}_{n-1} \mathbf{u}_n}{\lambda (a_j e_n^{\alpha_j})^{-1} + \mathbf{u}_n^T \mathbf{P}_{n-1} \mathbf{u}_n}$$

$$\hat{\mathbf{w}}_n = \hat{\mathbf{w}}_{n-1} + g_n \xi_n$$

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


---

# Capítulo 5

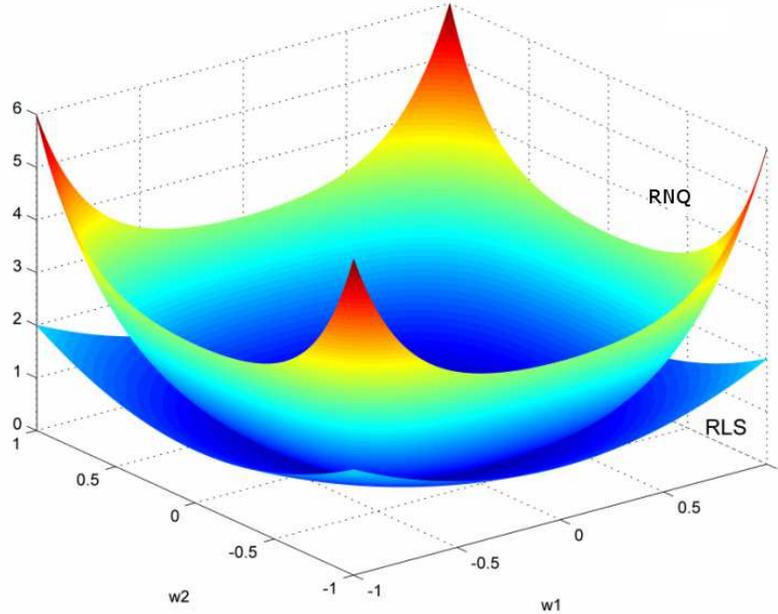
## Algoritmo KRNQ

### 5.1 Introdução

É apresentada uma versão não linear do algoritmo recursivo não quadrático (RNQ) [22]. Este executa a regressão linear em um espaço de características de alta dimensão, induzida por um kernel de Mercer e pode, portanto ser utilizado para construir recursivamente soluções dos erros de mínimos quadrados em problemas não lineares, que são encontrados frequentemente em aplicações de processamento de sinais, a fim de regularizar soluções e manter a complexidade do algoritmo delimitada. Usa-se um processo sequencial de separação. Este procedimento permite que o algoritmo de separação opere em tempo real.

Analisar o comportamento do algoritmo e sua convergência [27], comparando suas propriedades de escala na (identificação de sistema) na resolução de problemas de processamentos de sinais on line. O algoritmo de adaptação KRNQ é um dispositivo computacional que tem por objetivo modelar a relação entre dois sinais em tempo real de maneira interativa sendo que os sinais são processados por um filtro e os parâmetros inerentes à estrutura que pode ser iterativamente utilizada para alterar a relação entrada - saída do filtro sendo o algoritmo adaptativo que descreve como os parâmetros são ajustados de um determinado instante para o seguinte. O algoritmo kernel de funções não quadráticas são utilizados para atualizar os valores dos parâmetros de sistema, podendo assumir uma forma a ser deduzida em um procedimento de otimização que minimiza um critério de erro. Nesse trabalho propõe o uso do algoritmo de uma função de quarta ordem usando kernel apresentado em um problema geral da filtragem adaptativa ou problemas não lineares, introduzimos a notação matemática para apresentar a forma e o modo de operar de um filtro adaptativo. Em seguida vamos discutir a diferentes estruturas de algoritmos como (KRLS, KRNQ) propostas para serem usadas nas aplicações práticas. Finalmente,

damos uma dedução simples do algoritmo KRNQ, como um dos métodos mais simples e de rápida convergência de dados para o ajuste dos coeficientes de um filtro adaptativo.



**Figura 5.1:** Gráfico comparando o RNQ co RLS em relação a sua inclinação que é maior na primeira em peso de  $w_0$  e  $w_1$  variam de  $[-1, 1]$ .

## 5.2 Desenvolvimento do Método

O filtro adaptativo FIR é composto pelos sinais desejados  $d_i$ , vetor de entrada  $\mathbf{u}_i = [u_i, u_{i-1}, \dots, u_{i-L+1}]^T$ , e o erro  $e_i$ , onde são usados para atualizar o vetor peso do filtro  $\mathbf{w}_n = [w_{0,n}, w_{1,n}, \dots, w_{L-1,n}]^T$ . O objetivo é recuperar uma aproximação  $d_i$  estimando o sinal de saída  $y_i = \mathbf{w}_n^T \mathbf{u}_i$ , depois de calcular o erro  $e_i = d_i - y_i$ , onde  $n$  representa a amostra atual, com o mesmo  $1 \leq i \leq n$  e  $L$  é o comprimento do filtro.

Neste trabalho, iremos derivar o kernel dos mínimos quadrados recursivos (KRLS) em [21], modificando a superfície de desempenho do erro com base no algoritmo RNQ, cujo nome é kernel recursivo não quadrático (KRNQ). Para calcular o algoritmo KRNQ, usa-se o teorema de Mercer para transformar os dados de  $\mathbf{u}_i$  para a função espaço  $\mathbb{F}$  tal que  $\varphi(\mathbf{u}_i)$  [denota-se  $\varphi_i$ ] como mostrado em [23].

Definir a função de custo como,

$$J_n = \min_{\omega} \sum_{i=1}^n \beta^{n-i} |d_i - \mathbf{w}^T \varphi_i|^{2k} + \beta^n \lambda \|\mathbf{w}\|^2, \quad (5.1)$$

onde  $i$ ,  $n$  e  $k$  são inteiros positivos e  $\lambda$ , é o fator de esquecimento convencional ( $\lambda \in [0, 1)$ ).

Finalmente, calculamos o gradiente de  $J_n$  como,

$$\begin{aligned}\nabla J_n &= \sum_{i=1}^n \left[ -2k\beta^{n-i} (d_i - \mathbf{w}^T \varphi_i)^{2k-1} \right] \cdot \varphi_i + 2\beta^n \lambda \mathbf{w} \\ &= \sum_{i=1}^n \left[ -2k\beta^{n-i} (e_i)^{\alpha_k} \cdot (e_i) \right] \cdot \varphi_i + 2\beta^n \lambda \mathbf{w},\end{aligned}\quad (5.2)$$

onde  $e_i = (d_i - \mathbf{w}^T \varphi_i)$  e  $\alpha_k = 2k - 2$ .

Agora, igualando  $\nabla J_n$  a zero e depois de algumas manipulações matemática, obtém-se,

$$\sum_{i=1}^n \beta^{n-i} e_n^{2k-2} d_n \varphi_n = \left( \sum_{i=1}^n \beta^{n-i} e_n^{2k-2} \varphi_n \varphi_n^T \right) \mathbf{w}_n + \beta^n \lambda \mathbf{w}_n$$

temos

$$\begin{aligned}\mathbf{w}_n &= \sum_{i=1}^n \beta^{n-i} e_n^{\alpha_k} d_n \varphi_n \cdot \left( \sum_{i=1}^n \beta^{n-i} e_n^{\alpha_k} \varphi_n \varphi_n^T + \beta^n \lambda I \right)^{-1} \\ \mathbf{w}_n &= \left( \sum_{i=1}^n \beta^{n-i} e_n^{\alpha_k} \varphi_n \varphi_n^T + \beta^n \lambda I \right)^{-1} \sum_{i=1}^n \beta^{n-i} e_n^{\alpha_k} d_n \varphi_n\end{aligned}\quad (5.3)$$

onde,

$$\begin{aligned}\mathbf{d}_n &= [e_1^{\alpha_k} d_1, e_2^{\alpha_k} d_2, \dots, e_n^{\alpha_k} d_n]^T \\ \Phi_n &= [e_1^{\alpha_k} \varphi_1, e_2^{\alpha_k} \varphi_2, \dots, e_n^{\alpha_k} \varphi_n]\end{aligned}\quad (5.4)$$

Em seguida, o valor ótimo do vetor peso é definido pela equação da matriz,

$$\mathbf{w}_n = [\lambda \beta^n \mathbf{I} + \Phi_n \mathbf{B}_n \Phi_n^T]^{-1} \Phi_n \mathbf{B}_n \mathbf{d}_n \quad (5.5)$$

Sendo  $\beta$  o fator de esquecimento cujo  $\mathbf{B}_n = \text{diag}\{\beta^{n-1}, \beta^{n-2}, \dots, 1\}$  é a matriz diagonal.

Além disso, utilizando uma sequência de base de álgebra de matrizes conhecida como *lema da matriz inversa* na Equação (5.5), com as identificações;

$$\lambda\beta^n \mathbf{I} \rightarrow \mathbf{A}, \Phi_n \rightarrow \mathbf{B}, \mathbf{B}_n \rightarrow \mathbf{C}, \Phi_n^T \rightarrow \mathbf{D}, \quad (5.6)$$

obtemos,

$$\begin{aligned} \mathbf{w}_n &= \Phi_n [\lambda\beta^n \mathbf{B}_n^{-1} + \Phi_n^T \Phi_n]^{-1} \mathbf{d}_n \\ \mathbf{w}_n &= \Phi_n \mathbf{a}_n, \end{aligned} \quad (5.7)$$

sendo que.

$$\mathbf{a}_n = [\lambda\beta^n \mathbf{B}_n^{-1} + \Phi_n^T \Phi_n]^{-1} \mathbf{d}_n \quad (5.8)$$

Onde  $\mathbf{a}_n$  é o vetor coeficiente e  $\Phi_n^T \Phi_n$  é calculado pelo "truque" do kernel.

Denotamos como,  $\mathbf{Q}_n = [\lambda\beta^n \mathbf{B}_n^{-1} + \Phi_n^T \Phi_n]^{-1}$  pode-se verificar que,

$$\mathbf{Q}_n^{-1} = \begin{bmatrix} \mathbf{Q}_{n-1}^{-1} & \mathbf{h}_n \\ \mathbf{h}_n^T & \lambda\beta^n + e_n^{\alpha_k} \varphi_n^T \varphi_n \end{bmatrix}, \quad (5.9)$$

onde  $\mathbf{h}_n = e_n^{\alpha_k} \Phi_{n-1}^T \varphi_n$ . Reescrevendo a Equação (5.9) usando o bloco de inversão de matriz (conforme é feito no KRLS), tem-se;

$$\begin{aligned} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} &= \begin{bmatrix} (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} & (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \end{bmatrix} \\ \mathbf{Q}_n &= \begin{bmatrix} \mathbf{Q}_{n-1} + \mathbf{z}_n \mathbf{z}_n^T r_n^{-1} & -\mathbf{z}_n r_n^{-1} \\ -\mathbf{z}_n^T r_n^{-1} & r_n^{-1} \end{bmatrix}, \end{aligned} \quad (5.10)$$

onde

$$\begin{aligned} \mathbf{z}_n &= \mathbf{Q}_{i-1} \mathbf{h}_n \\ r_n &= \lambda \beta^n + \varphi_n^T \varphi_n - \mathbf{z}_n^T \mathbf{h}_n \end{aligned} \quad (5.11)$$

Pode-se reescrever a Equação (5.8) como,

$$\begin{aligned} \mathbf{a}_n &= \mathbf{Q}_n \mathbf{d}_n \\ \mathbf{a}_n &= \begin{bmatrix} \mathbf{Q}_{n-1} + \mathbf{z}_n \mathbf{z}_n^T r_n^{-1} & -\mathbf{z}_n r_n^{-1} \\ -\mathbf{z}_n^T r_n^{-1} & r_n^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{d}_{n-1} \\ d_n \end{bmatrix} \\ \mathbf{a}_n &= \begin{bmatrix} \mathbf{a}_{n-1} + \mathbf{z}_n r_n^{-1} e_n \\ r_n^{-1} e_n \end{bmatrix} \end{aligned} \quad (5.12)$$

onde  $\mathbf{a}_n$  são as expansões dos coeficientes da Equação 5.7, onde é o mesmo em **KRLS** e [21] [23] [24].

O Algoritmo 4, que foi proposto pode ser resumido o Algoritmo Recursivo Não Quadrático baseado em Kernel (KRNQ)

---

**Algoritmo 4** Algoritmo Kernel Recursivo Não Quadrático (KRNQ)

---

1: Inicializar

$$\begin{aligned} \mathbf{Q}_1 &= (\lambda \beta + \kappa(\mathbf{u}_1, \mathbf{u}_1))^{-1}, \\ \mathbf{a}_1 &= \mathbf{Q}_1 \mathbf{d}_1 \end{aligned}$$

2: Calculando

3: Para o instante de tempo,  $n = 2, 3, \dots$ ,

$$\mathbf{h}_n = [e_1^{\alpha_k} \kappa(\mathbf{u}_n, \mathbf{u}_1), \dots, e_{n-1}^{\alpha_k} \kappa(\mathbf{u}_n, \mathbf{u}_{n-1})]^T,$$

$$\mathbf{z}_n = \mathbf{Q}_{n-1} \mathbf{h}_n,$$

$$r_n = \lambda \beta^n + \kappa(\mathbf{u}_n, \mathbf{u}_n) - \mathbf{z}_n^T \mathbf{h}_n,$$

$$\mathbf{Q}_n = r_n^{-1} \begin{bmatrix} \mathbf{Q}_{n-1} r_n + \mathbf{z}_n \mathbf{z}_n^T & -\mathbf{z}_n \\ -\mathbf{z}_n^T & 1 \end{bmatrix},$$

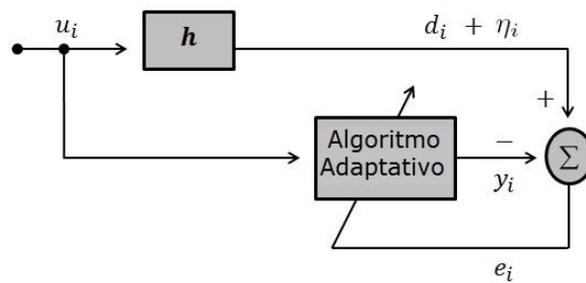
$$e_n = d_n - \mathbf{h}_n^T \mathbf{a}_{n-1},$$

$$\mathbf{a}_n = \begin{bmatrix} \mathbf{a}_{n-1} + \mathbf{z}_n r_n^{-1} e_n \\ r_n^{-1} e_n \end{bmatrix}.$$


---

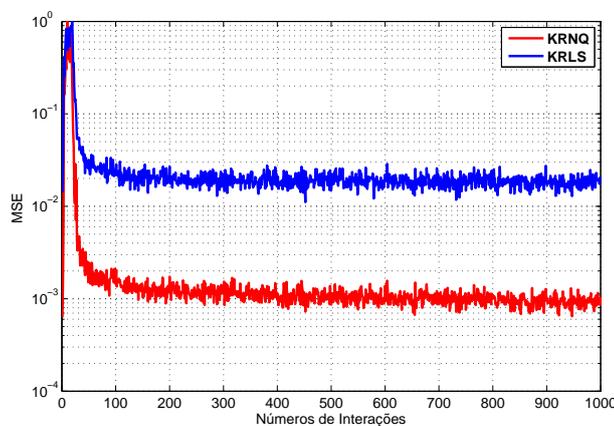
### 5.2.1 Identificação de Sistema KRNQ

O objetivo da implementação do algoritmo em uma estrutura cujo problema é não linear em um filtro adaptativo FIR que será feita a comparação dos desempenhos nos algoritmos KRLS e KRNQ baseado em uma potência par do erro. As simulações foram realizadas em dois tipos de sistemas: um ruidoso com dois tipos diferentes de ruído. Sendo que aplicamos o algoritmo proposto em uma estrutura de identificação de sistema, onde podemos analisar na figura 1, onde usamos um filtro FIR com resposta ao impulso  $h$  e a ordem  $L = 15$ , com o sinal de entrada,  $u_i$ , é um sinal branco uniformemente distribuído, limitado no intervalo  $[-1, 1]$ .

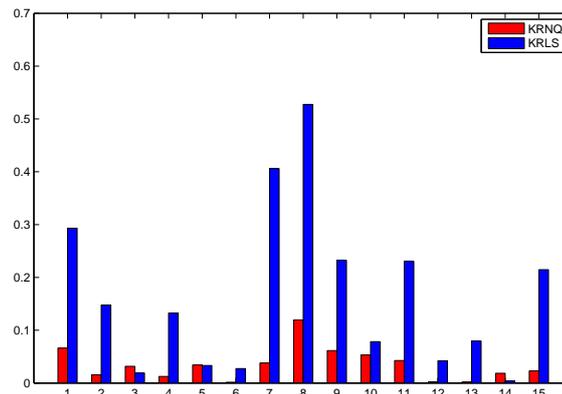


**Figura 5.2:** Estrutura do filtro adaptativo na configuração de identificação do sistema, onde  $u_i$  é vetor de entrada,  $h$  é o filtro FIR,  $d_i$  é o sinal desejado,  $\eta_i$  é sinal de ruído,  $e_i$  é erro sinal e  $y_i$  é saída do filtro.

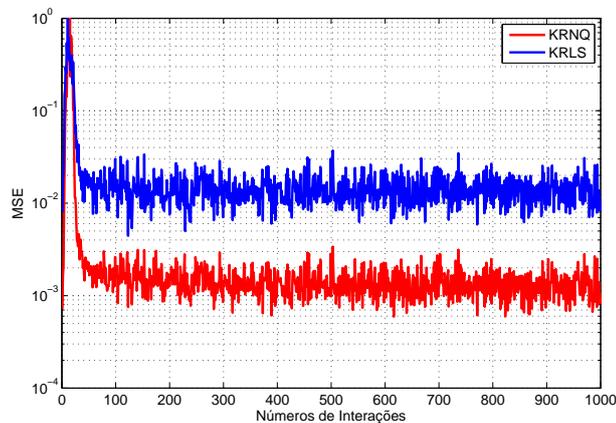
Nossa simulação onde descrevemos um sinal corrompido pelo ruído adaptativo,  $\eta_i$ . Usamos dois sinais ruidosos, o primeiro uma distribuição Gaussiana e o outro uma distribuição Laplaciana, ambos gerados com média zero e variância unitária. Para ambos os algoritmos usamos  $\beta = 0,98$  e o kernel polinomial [23], com parâmetro  $a = 1$ , e conjunto de parâmetro de regularização  $0,001$ .



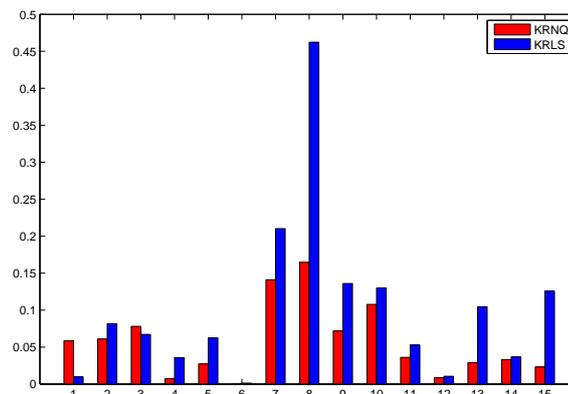
**Figura 5.3:** Curva de aprendizagem dos algoritmos KRNQ e KRLS quando foi usada uma distribuição com erro Gaussiano.



**Figura 5.4:** Gráfico do erro em tempo real antes dos coeficientes de estimação do filtro para ambos os algoritmos.



**Figura 5.5:** Curva de aprendizagem dos algoritmos KRNQ e KRLS quando usada uma distribuição de erro Laplaciano. Usamos  $\beta = 0,98$  para ambos os algoritmos e  $k = 2$  para o KRNQ.



**Figura 5.6:** Gráfico do erro no tempo real antes dos coeficientes de estimação do filtro para ambos os algoritmos.

Na Figura 5.3 o primeiro podemos observar a curvas de aprendizagem de ambos algoritmos no experimento com distribuição de sinais com ruído Gaussiano. Pode-se observar

que o algoritmo KRNQ tem o melhor desempenho que o algoritmo KRLS. Na Figura 5.4 pode-se analisar que no gráfico mostra erros menores em relação aos coeficientes do filtro sendo verificado o estimador do coeficientes dos algoritmos KRNQ e KRLS no mesmo experimento.

Na Figuras 5.5 e 5.6 pode-se ver os resultados quando um ruído com distribuição Laplaciana foi usado. Podemos também analisar que o algoritmo KRNQ tem o melhor desempenho que o algoritmo KRLS.

### 5.2.2 Implementação no Sinal Eletrocardiograma(ECG) Não Artificial e Artificial

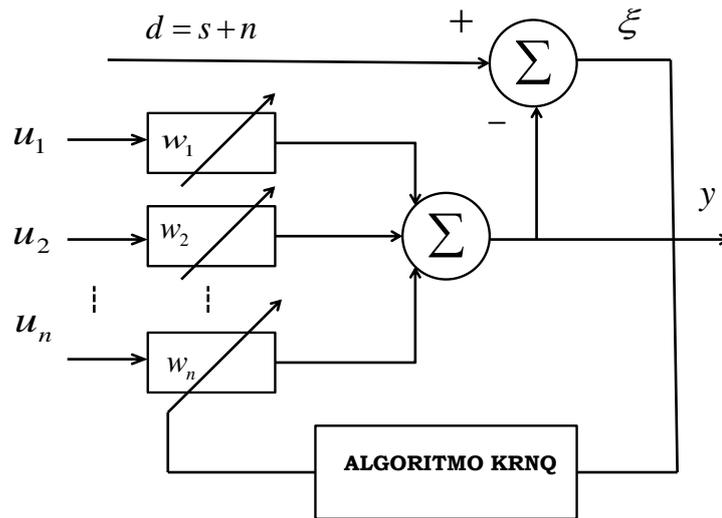
O eletrocardiograma (ECG) é um teste não invasivo usado para determinar as condições do coração através das medidas de sua atividade elétrica. O ECG é construído utilizando as medidas dos potenciais elétricos presentes nos tecidos cardíacos. Estas medidas são obtidas por meio de derivações, circuitos formados por dois eletrodos ligados aos polos positivo e negativo de um galvanômetro [28].

Através do ECG, é possível diagnosticar: aumento do coração; defeitos cardíacos congênitos; arritmias; posição anormal do coração; inflações (pericardite e miocardite) entre outras.

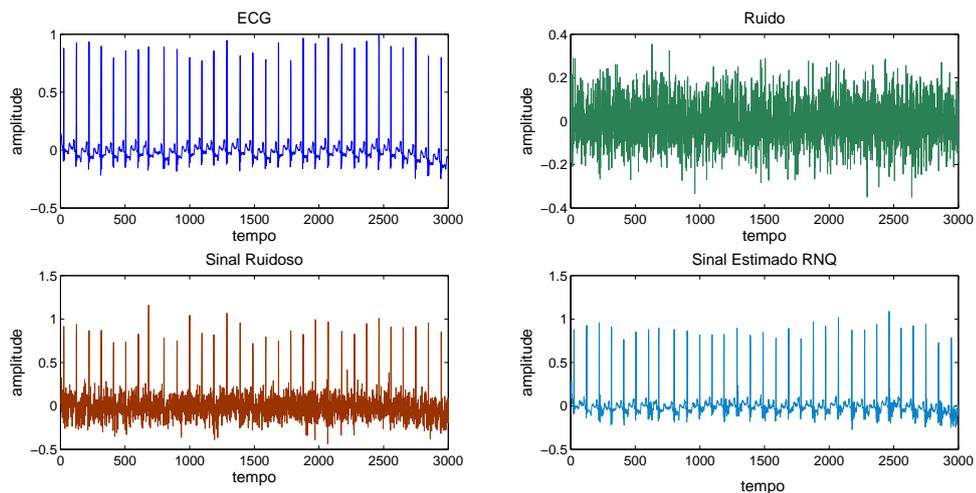
Embora os sinais de ECG [28] possam ser corrompidos por vários tipos de ruído, existem sete que se destacam devido a distorção introduzida nos sinais de ECG [29].

- Interferência da rede elétrica durante a obtenção das leituras de ECG, pois o sinal é degradado pela soma das componentes de frequência (50 ou 60 Hz) oriundas da interferência da rede elétrica. [30].
- Ruído de contato do eletrodo é a perda do contato do eletrodo com o peito do paciente ou voluntário em análise. A perda de contato pode ter duração em torno de 1 segundo [28].
- Artefatos de movimento é a alteração da impedância no contato do eletrodo com o peito [28].
- Contrações musculares é o ruído gaussiano em um sinal de banda limitada com média zero [29].
- Alterações na linha de base e modulação da amplitude do ECG com a respiração [28].

- Ruído gerado por dispositivos elétricos usados em processamento de sinais que são gerados por dispositivos eletrônicos nos sistemas de instrumentação [28].
- Ruído eletrocirúrgico é a interferência dos sinais de frequência da rádio gerados por uma unidade eletrocirúrgica, os quais alteram completamente o sinal de ECG [28].

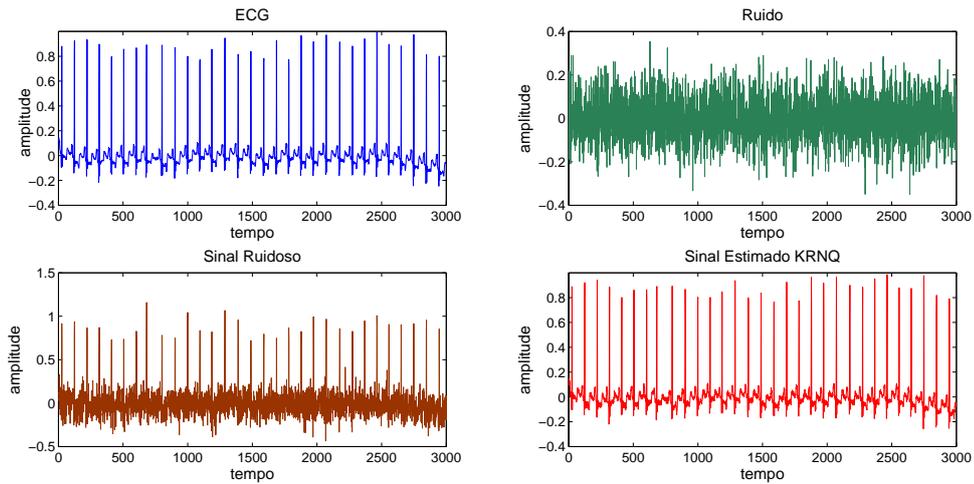


**Figura 5.7:** Diagrama de bloco do filtro adaptativo  $s$  é o sinal determinística,  $n$  é o ruído decorrelacionado com  $s$  e  $[u_1, u_2, \dots, u_n]^T$  é o vetor de entrada.

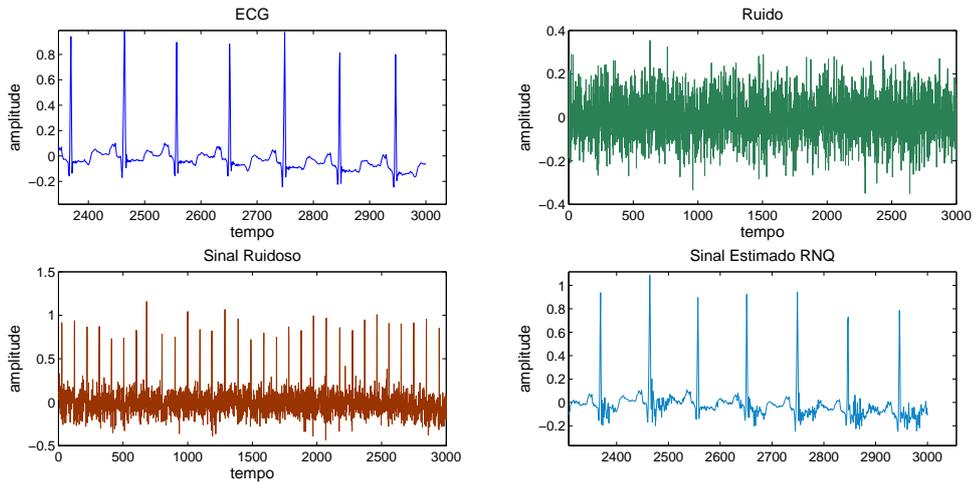


**Figura 5.8:** Gráfico do sinal que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo RNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro.

O objetivo principal da filtragem de eletrocardiograma, assim como de qualquer outra técnica de filtragem, é a eliminação do ruído e a recuperação do sinal original [28], mantendo as características morfológicas significativas do sinal após sua reconstrução [9].



**Figura 5.9:** Gráfico do sinal ECG que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo KRNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro.

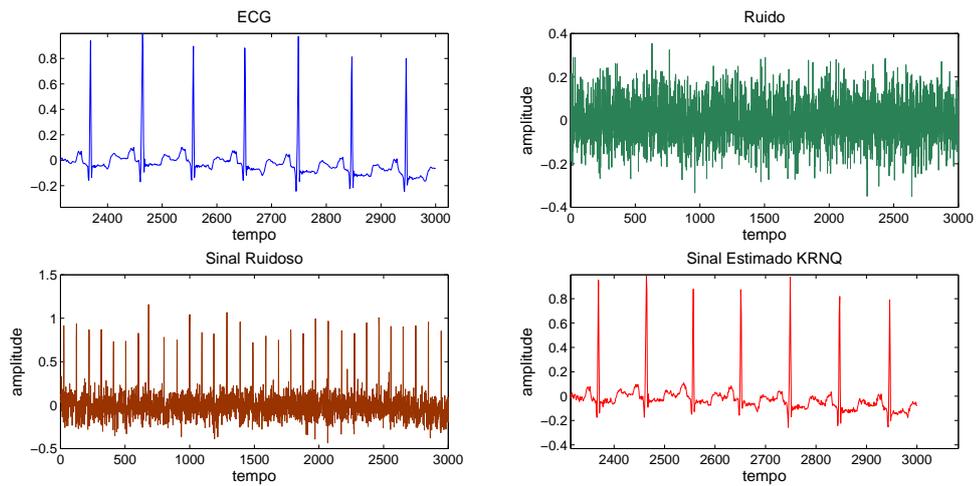


**Figura 5.10:** Gráfico do sinal que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo RNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro

Além da comparação visual, a maioria dos métodos de filtragem adaptativa tem utilizado a diferença média quadrática percentual  $\xi^*$  como forma de avaliação do sinal reconstruído.

$$\xi^* = \sqrt{\frac{\sum_{i=1}^n [E_{orig(i)} - E_{rec(i)}]^2}{\sum_{i=1}^n E_{orig(i)}^2}} * 100 \quad (5.13)$$

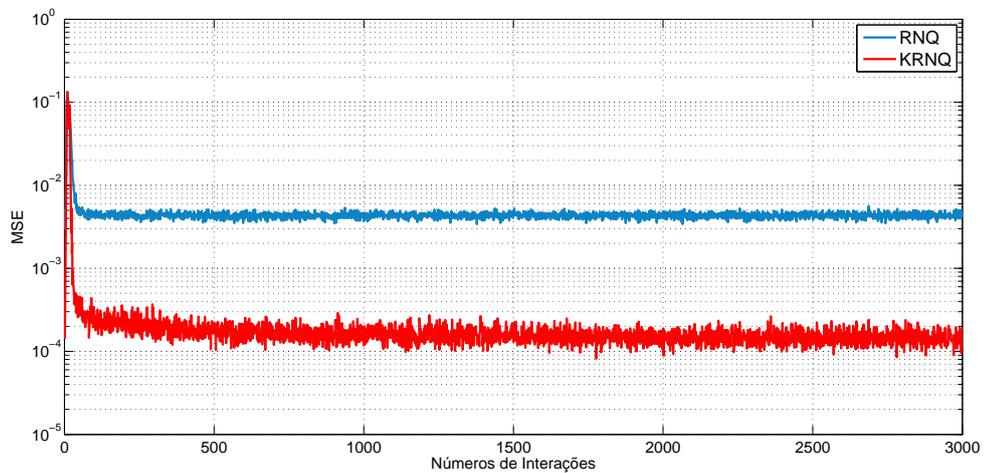
O  $\xi^*$  é definido por, onde  $E_{orig(i)}$  é o sinal do ECG original, enquanto  $E_{rec(i)}$  é o sinal recuperado após a filtragem do sinal ruidoso. Na Tabela 5.1 mostra a comparação de alguns métodos de filtragem adaptativa do ECG.



**Figura 5.11:** Gráfico do sinal ECG que foi adicionado um ruído gaussiano de 25% e logo temos o algoritmo KRNQ gera o gráfico do sinal ECG não artificial sem o erro depois do filtro.

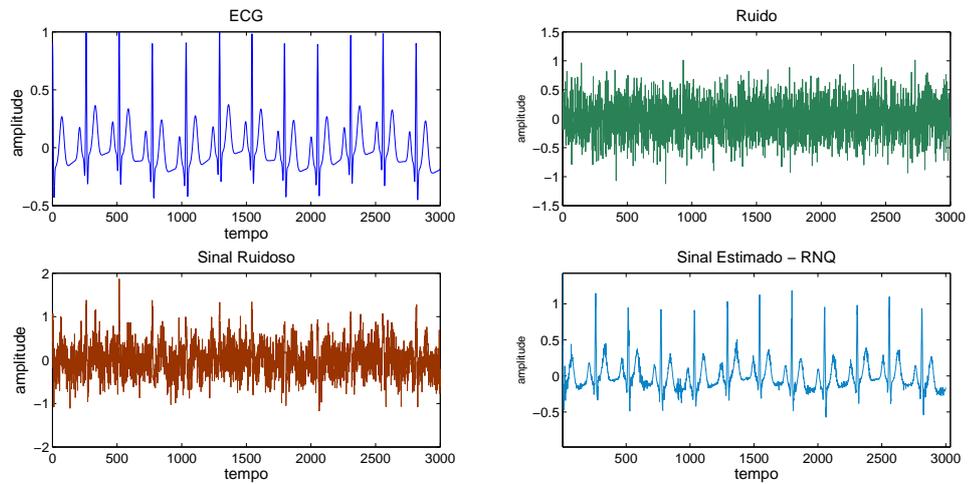
**Tabela 5.1:** Estimação das médias quadráticas percentual do sinal ECG não artificial de 1% até 25% com sinal ruidoso

	1%	5%	10%	15%	20%	25%
RNQ	10,08	15,39	16,10	19,10	18,69	20,57
KRNQ	0,14	0,43	1,44	2,16	2,87	3,59

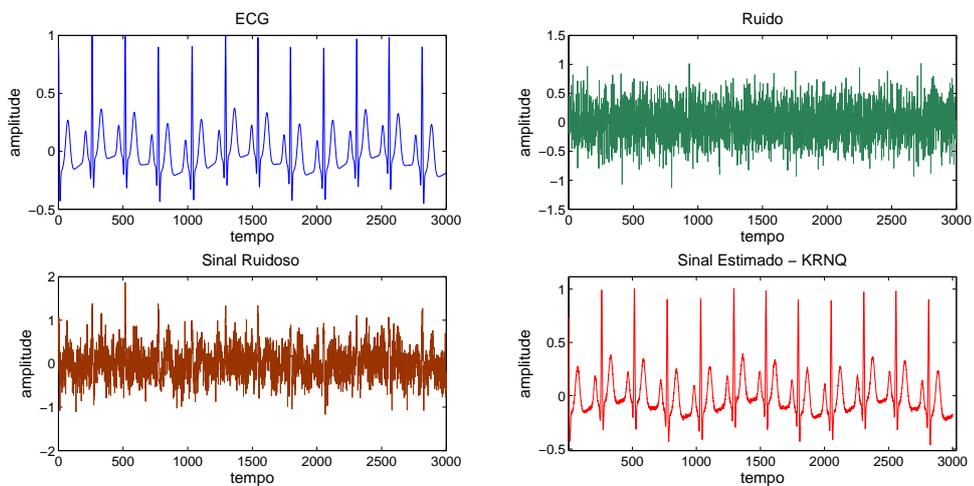


**Figura 5.12:** Gráfico comparando as curvas de aprendizagem dos algoritmos RNQ e KRNQ em um sinal cardíaco o ECG não artificial.

Agora iremos fazer a simulação do sinal ECG artificial [28] em seguida analisaremos os resultados obtidos para esse sinal específico com sinal de erro de 25%.



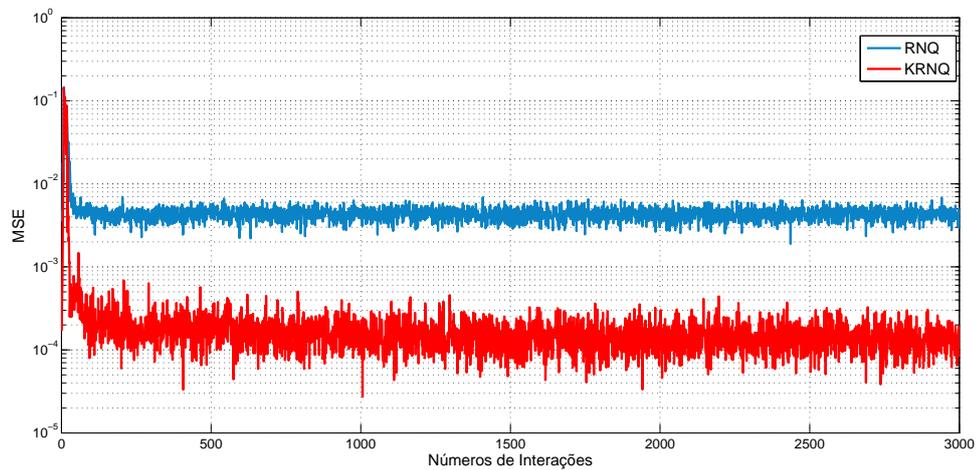
**Figura 5.13:** Gráfico do sinal artificial que foi adicionado um ruído gaussiano de 15% e logo temos o algoritmo RNQ gera o gráfico do sinal ECG sem o erro depois do filtro.



**Figura 5.14:** Gráfico do sinal ECG artificial que foi adicionado um ruído gaussiano de 15% e logo temos o algoritmo KRNQ gera o gráfico do sinal ECG sem o erro depois do filtro.

**Tabela 5.2:** Estimação das médias quadráticas percentual do sinal ECG artificial de 1% até 25% com sinal ruidoso

	1%	5%	10%	15%	20%	25%
RNQ	12,19	16,95	19,06	22,65	25,29	30,13
KRNQ	0,25	1,55	3,09	4,64	6,87	7,73



**Figura 5.15:** Gráfico comparando as curvas de aprendizagem dos algoritmos RNQ e KRNQ em um sinal cardíaco o ECG artificial.

# Capítulo 6

## Conclusões

Nessa dissertação propomos o desenvolvimento de um algoritmo recursivo inspirado no algoritmo RNQ baseado em kernel para ajustar os pesos em um filtro adaptativo. Analisamos também a construção do novo algoritmo e para isso devemos reproduzir o kernel no espaço de Hilbert (RKHS) que resultará no desenvolvimento que chamamos de algoritmo recursivo não quadrático baseado em kernel (KRNQ). O vetor obtido do vetor peso ótimo,  $\mathbf{w}_n$ , retirado a posteriori da função de custo (5.1) tem-se como finalidade reduzir os valores mínimos que serão definidos pela Equação (5.7). No algoritmo 4 observa-se o algoritmo em etapas graduais, como o objetivo de esclarecer o desenvolvimento passo a passo do algoritmo KRNQ.

Simulamos um problema de identificação de sistema, para ilustrar e verificar os resultados teóricos. Usamos dois tipos de ruído, um Gaussiano e um Laplaciano. Na Figura 5.3 pode-se ver as curvas de aprendizagem de um experimento com o ruído Gaussiano, onde observou-se que os dois algoritmos tem aproximadamente o mesmo tempo de convergência, mas o algoritmo KRNQ atinge um valor de erro mínimo, pelo menos, dez vezes menos que aquele atingido pelo algoritmo KRLS.

Na Figura 5.4 é possível analisar o gráfico dos erros entre os coeficientes de filtros reais e os coeficientes estimados por ambos os algoritmos. Pode-se ver que os algoritmos KRNQ tem um desempenho melhor do que o algoritmo KRLS.

Os resultados experimentais com ruído Laplaciano são mostrados na Figura 5.5. Na Figura 5.6 pode-se ver as curvas de aprendizagem e o gráfico dos erros entre os coeficientes de filtros reais e os coeficientes estimados por ambos os algoritmos. Tal como no experimento anterior, o algoritmo KRNQ possui melhor desempenho que o algoritmo KRLS.

Portanto, o objetivo maior foi apresentar um algoritmo kernelizado para filtragem

---

adaptativa recursiva que utilize as estatísticas de ordem superior dos sinais gravados para estimar os coeficientes do filtro em um problema de identificação do sistema. Os resultados aqui apresentados, mostraram que o algoritmo KRNQ proposto é uma opção alternativa para o algoritmo KRLS, uma vez que tem melhor desempenho no erro médio quadrático.

Simula-se também um problema de filtragem de eletrocardiograma, com contrações musculares que é o ruído gaussiano em um sinal de banda limitada com média zero. Fiz a eliminação do ruído e a recuperação do sinal original, mantendo as características morfológicas significativas do sinal após sua reconstrução, além da comparação visual. A maioria dos métodos de filtragem adaptativa tem utilizado a diferença média quadrática percentual  $\xi^*$  como forma de avaliação do sinal reconstruído que obteve bons resultados mostrados na Tabela 5.1.

Analisa-se em seguida uma simulação do sinal ECG artificial. Os resultados obtidos para esse sinal específico com sinal de erro, teve como resultado a curva de aprendizagem do algoritmo RNQ do sinal ECG artificial que foi adicionado um ruído gaussiano de 15%. Sendo assim, o algoritmo RNQ gera o gráfico do sinal ECG sem o erro depois do filtro, que tem 80,94% de recuperação do sinal artificial. Já o gráfico do sinal ECG artificial que foi adicionado um ruído gaussiano de 15% obtemos o algoritmo KRNQ gerando o gráfico do sinal ECG sem o erro depois do filtro, com 96,91% de recuperação do sinal artificial atingindo bons resultados observados na Tabela 5.2.

# Apêndice A

## Propriedades de Matrizes

Neste apêndice, reunimos algumas propriedades úteis e identidades envolvendo matrizes e determinantes. Isto não tem a intenção de ser um tutorial introdutório, e presume-se que o leitor esteja familiarizado com álgebra linear. Para alguns resultados, indicar como prová-los, ao passo que em casos mais complexos deixamos o leitor interessado para se referir a livros de texto padrão sobre o assunto. Em todos os casos, assumir que existem inversos e que as dimensões da matriz são tais, que as fórmulas estão corretamente definidas.

**Definição 4 (Identidades Básicas de Matrizes).** *Uma matriz  $\mathbf{A}$  tem elementos  $A_{ij}$  onde  $i$  índice de linhas,  $j$  índice de colunas. Usamos  $\mathbf{I}_N$  para denotar a matriz identidade  $N \times N$  (Podemos chamar de matriz unitária), onde não há ambiguidade sobre dimensionalidade usamos simplesmente  $\mathbf{I}$ . A matriz transposta  $\mathbf{A}^T$  tem elemento  $(\mathbf{A}^T)_{ij} = A_{ij}$ . A partir da definição de transposição, temos;*

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T \tag{A.1}$$

*Que pode ser escrito os índices. O inverso de  $\mathbf{A}$ , denotado  $\mathbf{A}^{-1}$ , satisfaz;*

$$\mathbf{AA}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{A.2}$$

*Porque  $\mathbf{ABB}^{-1}\mathbf{A}^{-1} = \mathbf{I}$ , temos;*

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \tag{A.3}$$

*Também temos;*

$$(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T \quad (\text{A.4})$$

Que é facilmente ser comprovado pelas transposta de A.2 e aplicado A.1. A utilização da identidade na matriz inversa, temos;

$$(\mathbf{P}^{-1} + \mathbf{B}^T \mathbf{R}^{-1} \mathbf{B}) \mathbf{B}^T \mathbf{R}^{-1} = \mathbf{P} \mathbf{B}^T (\mathbf{B} \mathbf{P} \mathbf{B}^T + \mathbf{R})^{-1} \quad (\text{A.5})$$

Fácil verificar a multiplicação a direita dos mesmo lado por  $(\mathbf{B} \mathbf{P} \mathbf{B}^T + \mathbf{R})$ . Supondo que  $\mathbf{P}$  tem  $N \times N$  dimensionalidade e  $\mathbf{R}$  tem  $M \times M$  dimencionalidade, de modo que  $\mathbf{B}$  é  $M \times N$ .

Se  $M \ll N$ , verificamos A.5 do lado esquerdo. E concluímos que;

$$(\mathbf{I} + \mathbf{A} \mathbf{B})^{-1} \mathbf{A} = \mathbf{A} (\mathbf{I} + \mathbf{B} \mathbf{A})^{-1} \quad (\text{A.6})$$

Outra forma

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

que é conhecido como a identidade Woodbury e que pode ser verificado pela multiplicação de ambos os lados por  $(\mathbf{A} + \mathbf{B} \mathbf{C} \mathbf{D})$ . Isto é útil, por exemplo, quando se tem a matriz  $\mathbf{A}$  com uma grande diagonalidade portanto é fácil de se ter a inverta, enquanto a matriz  $\mathbf{B}$  tem muitas linhas e poucas colunas e (é inversamente de  $\mathbf{C}$ ) de modo que do lado á direita é mais fácil de avaliar do que o lado á esquerda.

Um conjunto de vetores  $\{a_1, \dots, a_N\}$  é linearmente independente se a relação  $\sum_n \alpha_n a_n$  detém apenas se todos  $\alpha_n = 0$ . Isto implica que nenhum dos vetores pode ser expresso como combinação linear do restante. O posto ou (rank) da matriz é o número máximo de linhas linearmente independente (ou equivalente ao número máximo de colunas linearmente independente)

*Demonstração.* Seja  $\mathbf{A}$ ,  $\mathbf{B}$  e  $\mathbf{C}^{-1} + \mathbf{D} \mathbf{A}^{-1} \mathbf{B}$  seja não singular da matriz quadrática. Quando  $\mathbf{A} + \mathbf{B} \mathbf{C} \mathbf{D}$  seja inversível e;

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

Pela multiplicação à direta nós encontramos que:

$$\begin{aligned}
& (\mathbf{A} + \mathbf{BCD}) \left( \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B} (\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1} \mathbf{DA}^{-1} \right) \\
&= \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{B} (\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}) \mathbf{DA}^{-1} - \mathbf{BCDA}^{-1}\mathbf{B} (\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}) \mathbf{DA}^{-1} \\
&= \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{BC} (\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B}) (\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})^{-1} \mathbf{DA}^{-1} \\
&= \mathbf{I} + \mathbf{BCDA}^{-1} - \frac{\mathbf{BCDA}^{-1}\mathbf{B} (\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})}{(\mathbf{C}^{-1} + \mathbf{DA}^{-1}\mathbf{B})} \\
&= \mathbf{I} + \mathbf{BCDA}^{-1} - \mathbf{BCDA}^{-1} \\
&= \mathbf{I}
\end{aligned}$$

■

**Definição 5 (Traço e Determinantes).** *São aplicados em espaços métricos. O traço  $Tr(\mathbf{A})$  na matriz  $\mathbf{A}$  é definida como somatório dos elementos da diagonal. Podemos escrever, como;*

$$Tr(\mathbf{AB}) = Tr(\mathbf{BA}) \quad (\text{A.7})$$

*Podemos aplicar essa forma com o produto de três matrizes, teremos que;*

$$Tr(\mathbf{ABC}) = Tr(\mathbf{CBA}) = Tr(\mathbf{BCA}) \quad (\text{A.8})$$

*O determinante  $|\mathbf{A}|$  de uma matriz de ordem  $N \times N$  de  $\mathbf{A}$  é definida por;*

$$|\mathbf{A}| = \sum (\pm 1) A_{1i_2} A_{2i_2} \dots A_{Ni_N} \quad (\text{A.9})$$

*em que a soma é tomada sobre os produtos que consistem em examinar um elemento de cada linha e um elemento de cada coluna, com um coeficiente de +1 ou -1 para saber se a permutação  $i_1 i_2 \dots i_N$  é par ímpar, respectivamente. Note que  $|\mathbf{I}| = 1$ . Assim, para uma*

matriz de  $2 \times 2$ , o determinante tem a forma;

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21} \quad (\text{A.10})$$

O determinante do produto de duas matrizes é dado por;

$$|\mathbf{AB}| = |\mathbf{A}| |\mathbf{B}| \quad (\text{A.11})$$

podemos mostrar em A.9. Além disso, o determinante de uma matriz inversa é dada pela;

$$|\mathbf{A}^{-1}| = \frac{1}{|\mathbf{A}|} \quad (\text{A.12})$$

que pode ser mostrado, tomando o determinante de A.2 e aplicando em A.11.

Se  $\mathbf{A}$  e  $\mathbf{B}$  são matrizes de dimensão  $N \times M$ , em seguida.

$$|\mathbf{I}_N + \mathbf{AB}|^T = |\mathbf{I}_N + \mathbf{A}^T \mathbf{B}| \quad (\text{A.13})$$

Caso especial,

$$|\mathbf{I}_N + \mathbf{ab}|^T = |\mathbf{I}_N + \mathbf{a}^T \mathbf{b}| \quad (\text{A.14})$$

em que  $\mathbf{a}$  e  $\mathbf{b}$  são vetores de coluna  $N$  dimensionalidade.

**Definição 6 (Lema do bloco da matrix inversa).** Seja  $\mathbf{X}$  uma matrix simétrica ( $N \times N$ ) dado com forma de bloco,

$$\mathbf{X} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & \mathbf{C} \end{bmatrix}, \quad (\text{A.15})$$

onde  $\mathbf{A}$  é  $(N - 1) \times (N - 1)$  matrix inversível,  $\mathbf{b}$  é  $(N - 1) \times 1$  vetor coluna e  $\mathbf{c}$  é um escalar. Onde a inversa de  $\mathbf{X}^{-1}$  existe e admite a forma.

$$\mathbf{X}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \frac{1}{r} \mathbf{A}^{-1} \mathbf{b} \mathbf{b}^T \mathbf{A}^{-1} & -\frac{1}{r} \mathbf{A}^{-1} \mathbf{b} \\ -\frac{1}{r} \mathbf{b}^T \mathbf{A}^{-1} & \frac{1}{r} \end{bmatrix},$$

Se o complementar de Schur  $r = c - b^T \mathbf{A}^{-1} b$  seja o bloco  $\mathbf{A}$  é não nulo.

*Demonstração.* Vamos escrever a fatoração de  $\mathbf{X}$  pela menor diagonal para o maior.

$$\mathbf{X} = \begin{bmatrix} \mathbf{A} & b \\ b^T & c \end{bmatrix} = \mathbf{LDU} = \begin{bmatrix} \mathbf{I} & 0 \\ b^T \mathbf{A}^{-1} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A} & 0 \\ 0 & r \end{bmatrix} \begin{bmatrix} \mathbf{I} & b^T \mathbf{A}^{-1} \\ 0 & 1 \end{bmatrix}, \quad (\text{A.16})$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{A} & b \\ b^T & c \end{bmatrix} = \begin{bmatrix} \mathbf{I} & 0 \\ b^T \mathbf{A}^{-1} \mathbf{A} & r \end{bmatrix} \begin{bmatrix} \mathbf{I} & b^T \mathbf{A}^{-1} \\ 0 & 1 \end{bmatrix}, \quad (\text{A.17})$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{A} & b \\ b^T & c \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{A} b^T \mathbf{A}^{-1} \\ b^T & b^T b^T \mathbf{A}^{-1} + r \end{bmatrix} \quad (\text{A.18})$$

$$\mathbf{X} = \begin{bmatrix} \mathbf{A} & b \\ b^T & c \end{bmatrix} = \begin{bmatrix} \mathbf{A} & b \\ b^T & b^T \mathbf{A}^{-1} b + r \end{bmatrix} \quad (\text{A.19})$$

Temos que  $c = b^T \mathbf{A}^{-1} b + r$ ,

$$\mathbf{X} = \begin{bmatrix} \mathbf{A} & b \\ b^T & c \end{bmatrix} = \begin{bmatrix} \mathbf{A} & b \\ b^T & c \end{bmatrix} \quad (\text{A.20})$$

onde a inversa de  $\mathbf{X}$  podemos encontrar,

$$\mathbf{X}^{-1} = \mathbf{U}^{-1} \mathbf{D}^{-1} \mathbf{L}^{-1} = \begin{bmatrix} \mathbf{I} & b^T \mathbf{A}^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} & 0 \\ 0 & \frac{1}{r} \end{bmatrix} \begin{bmatrix} \mathbf{I} & 0 \\ b^T \mathbf{A}^{-1} & 1 \end{bmatrix}, \quad (\text{A.21})$$

$$\mathbf{X}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \frac{1}{r} \mathbf{A}^{-1} b b^T \mathbf{A}^{-1} & -\frac{1}{r} \mathbf{A}^{-1} b \\ -\frac{1}{r} b^T \mathbf{A}^{-1} & \frac{1}{r} \end{bmatrix}.$$

■

# Referências Bibliográficas

- [1] VOROS, J. Modeling and identification of wiener systems with two-segment nonlinearities. *Control Systems Technology, IEEE Transactions on*, IEEE, v. 11, n. 2, p. 253–257, 2003. 1, 8
- [2] SAYED, A. H. *Fundamentals of adaptive filtering*. [S.l.]: John Wiley & Sons, 2003. 1
- [3] HAYKIN, S. S. *Adaptive Filter Theory, 4/e*. [S.l.]: Pearson Education India, 2005. 1, 4, 5
- [4] HAYKIN, S. *Adaptive filter theory, 1996*. [S.l.]: Prentice-Hall, Upper Saddle River, New Jersey, 2000. 1, 2, 4, 6
- [5] HAYKIN, S. *Neural networks: a comprehensive foundation*. [S.l.]: Prentice Hall PTR, 1994. 1, 16
- [6] HERBRICH, R. *Learning kernel classifiers: theory and algorithms*. [S.l.]: The MIT press, 2002. 1, 12
- [7] SCHÖLKOPF, B.; SMOLA, A. J. Learning with kernels: support vector machines, regularization, optimization, and beyond (adaptive computation and machine learning). The MIT Press, 2001. 1, 11, 13, 14, 15, 16, 17
- [8] SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and computing*, Springer, v. 14, n. 3, p. 199–222, 2004. 1
- [9] SCHÖLKOPF, B.; TSUDA, K.; VERT, J.-P. *Kernel methods in computational biology*. [S.l.]: The MIT press, 2004. 2, 41
- [10] SCHETZEN, M. The volterra and wiener theories of nonlinear systems. {John Wiley & Sons}, 1980. 4, 5
- [11] WANG, J.-S.; HSU, Y.-L. Dynamic nonlinear system identification using a wiener-type recurrent network with okid algorithm. *J. Inf. Sci. Eng.*, v. 24, n. 3, p. 891–905, 2008. 5
- [12] WALACH, E.; WIDROW, B. The least mean fourth (lmf) adaptive algorithm and its family. *Information Theory, IEEE Transactions on*, IEEE, v. 30, n. 2, p. 275–283, 1984. 5
- [13] BOUBOULIS, P.; THEODORIDIS, S. Extension of wirtinger’s calculus to reproducing kernel hilbert spaces and the complex kernel lms. *Signal Processing, IEEE Transactions on*, IEEE, v. 59, n. 3, p. 964–978, 2011. 11

- 
- [14] RICHARD, C.; BERMUDEZ, J. C. M.; HONEINE, P. Online prediction of time series data with kernels. *Signal Processing, IEEE Transactions on*, IEEE, v. 57, n. 3, p. 1058–1067, 2009. 11, 12, 13
- [15] TSOCHANTARIDIS, I. et al. Support vector machine learning for interdependent and structured output spaces. In: ACM. *Proceedings of the twenty-first international conference on Machine learning*. [S.l.], 2004. p. 104. 11
- [16] GIROLAMI, M. Mercer kernel-based clustering in feature space. *Neural Networks, IEEE Transactions on*, IEEE, v. 13, n. 3, p. 780–784, 2002. 12
- [17] WATKINS, C. Dynamic alignment kernels. *Advances in Neural Information Processing Systems*, MIT; 1998, p. 39–50, 1999. 12, 13
- [18] HAUSSLER, D. *Convolution kernels on discrete structures*. [S.l.], 1999. 12, 14
- [19] LIMA, E. L. Espaços métricos, 2a edição. *Instituto de Matemática Pura e Aplicada-IMPA*, 1977. 14
- [20] KREYSZIG, E. *Introductory functional analysis with applications*. [S.l.]: Wiley. com, 2007. 15
- [21] ENGEL, Y.; MANNOR, S.; MEIR, R. The kernel recursive least-squares algorithm. *Signal Processing, IEEE Transactions on*, IEEE, v. 52, n. 8, p. 2275–2285, 2004. 19, 22, 24, 34, 37
- [22] SILVA, C. D. et al. An adaptive recursive algorithm based on non-quadratic function of the error. *Signal Processing*, Elsevier, v. 92, n. 4, p. 853–856, 2012. 19, 26, 27, 28, 29, 30, 31, 33
- [23] PRÍNCIPE, J. C.; LIU, W.; HAYKIN, S. *Kernel Adaptive Filtering: A Comprehensive Introduction*. [S.l.]: John Wiley & Sons, 2011. 20, 21, 22, 23, 24, 34, 37, 38
- [24] SINDHWANI, V.; ROSENBERG, D. S. An rkhs for multi-view learning and manifold co-regularization. In: ACM. *Proceedings of the 25th international conference on Machine learning*. [S.l.], 2008. p. 976–983. 22, 23, 37
- [25] SLAVAKIS, K.; THEODORIDIS, S. Sliding window generalized kernel affine projection algorithm using projection mappings. *EURASIP Journal on Advances in Signal Processing*, Springer, v. 2008, n. 1, p. 735351, 2008. 22, 23
- [26] LOAN, C. V.; GOLUB, G. *Matrix computations*. [S.l.]: Baltimore, MD: Johns Hopkins University Press, 1996. 29, 30
- [27] TROPP, J. A. Greed is good: Algorithmic results for sparse approximation. *Information Theory, IEEE Transactions on*, IEEE, v. 50, n. 10, p. 2231–2242, 2004. 33
- [28] CLIFFORD, G. D.; MCSHARRY, P. E. A realistic coupled nonlinear artificial ecg, bp, and respiratory signal generator for assessing noise performance of biomedical signal processing algorithms. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Second International Symposium on Fluctuations and Noise*. [S.l.], 2004. p. 290–301. 40, 41, 43

- [29] DAVIDSON, R. J. et al. Approach-withdrawal and cerebral asymmetry: Emotional expression and brain physiology: I. *Journal of personality and social psychology*, American Psychological Association, v. 58, n. 2, p. 330, 1990. 40
- [30] SAHAMBI, J.; TANDON, S.; BHATT, R. Using wavelet transforms for ecg characterization. an on-line digital signal processing system. *Engineering in Medicine and Biology Magazine, IEEE*, IEEE, v. 16, n. 1, p. 77–83, 1997. 40