Um Estudo Comparativo de Algoritmos de Filtragem Adaptativa Não Linear

Relatório Final de Iniciação Científica

Daniel Gilio Tiglea

Orientador: Prof. Dr. Magno Teófilo Madeira da Silva

Coorientador: Dr. Renato Candido

Escola Politécnica

Universidade de São Paulo

8 de Agosto de 2016

Conteúdo

Resumo

Filtros adaptativos podem ser empregados em diversas aplicações, tais como radar, sonar, comunicações, engenharia biomédica, predição de séries temporais, controle ativo de ruído, cancelamento de eco acústico, identificação de sistemas, entre outras. Em várias dessas aplicações, soluções lineares podem apresentar um desempenho inferior em comparação com as não lineares. Neste trabalho, serão estudados os algoritmos adaptativos do tipo LMS (*least-mean-squares*) baseados em núcleo. Como exemplo de aplicação serão consideradas a identificação de sistemas e a equalização de canais de comunicações não lineares.

Este relatório analisa criticamente a literatura encontrada acerca dos algoritmos denominados KLMS (*kernel least-mean-squares*) e KNLMS (*kernel normalized least-mean-squares*) e investiga algumas de suas propriedades, estando dividido em cinco seções e um apêndice. Na Seção 1 são apresentados os conceitos relacionados ao tema do trabalho, como filtros adaptativos e núcleos, bem como os algoritmos que serão analisados. Na Seção 2, expõe-se a motivação do trabalho. Em seguida, na Seção 3, são explicados os métodos adotados para a elaboração do trabalho e para a obtenção dos dados mostrados. Os resultados são apresentados na Seção 4, que conta com três subseções. Na Subseção 4.1 são apresentadas as diferenças entre duas versões dos algoritmos KLMS e KNLMS. Em seguida, esses algoritmos são aplicados à identificação de um sistema não linear na Subseção 4.2 e à equalização de um canal com distorções não lineares na Subseção 4.3. As constatações mais importantes estão sintetizadas na Seção 5. No Apêndice A é abordada a questão da equalização de canais em sistemas de comunicação baseados em caos. Ali são apresentadas novas versões dos algoritmos estudados ao longo do relatório, voltadas especificamente a esse tipo de aplicação, e são mostrados resultados obtidos por meio de simulações computacionais empregando as versões propostas.

1 Introdução

Filtros adaptativos são empregados em situações em que o ambiente está constantemente mudando, de maneira que mesmo que projetássemos e construíssemos um sistema fixo ótimo para uma dada aplicação em um certo instante de tempo, em instantes posteriores o desempenho desse filtro poderia se mostrar insatisfatório em decorrência das alterações nas características do ambiente. Devido à sua capacidade de se ajustar a diferentes ambientes, filtros adaptativos são bastante versáteis, encontrando muitas aplicações em processamento de sinais e controle [1–4]. Apesar das particularidades de cada aplicação, é possível estabelecer uma formulação comum para os problemas de filtragem adaptativa, mostrada na Figura 1 [1],

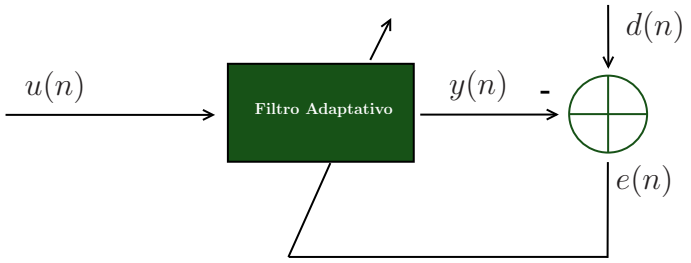


Figura 1: Entradas e saídas do filtro adaptativo.

em que $u(n)$ representa a entrada do filtro e $d(n)$ é denominado *sinal desejado*, que se pretende estimar por meio da saída $y(n)$. O sinal $e(n)$, denominado *sinal de erro* ou *erro de estimação*, é calculado por

$$e(n) = d(n) - y(n). \tag{1}$$

De modo geral, o ajuste dos filtros adaptativos é realizado por meio da medição de $e(n)$, no sentido de minimizar a função custo

$$J(n) \triangleq E\{e^2(n)\}, \tag{2}$$

em que $E\{\cdot\}$ denota o operador *esperança matemática*. A função $J(n)$ é também denominada *erro quadrático médio*, ou *"MSE"* (sigla em inglês para *mean square error*) [1], sendo um dos principais meios de se avaliar o desempenho de algoritmos adaptativos.

1.1 O algoritmo LMS

Proposto no começo da década de 1960 [5], o algoritmo LMS (*least-mean-squares*) é até hoje um dos mais populares algoritmos de filtragem adaptativa, em grande parte devido à sua simplicidade e facilidade de implementação. Além disso, por ter sido proposto há tanto tempo, suas propriedades já foram extensivamente estudadas e hoje são bem conhecidas [1–4]. Esse algoritmo visa a minimizar uma versão instantânea de $J(n)$, dada por

$$\hat{J}(n) \triangleq e^2(n) = [d(n) - y(n)]^2. \tag{3}$$

A saída $y(n)$ é calculada por meio de

$$y(n) = \mathbf{w}^{\mathrm{T}}(n-1)\mathbf{u}(n), \tag{4}$$

em que

$$\mathbf{w}(n-1) = [w_0(n-1) \quad w_1(n-1) \quad \cdots \quad w_{M-1}(n-1)]^{\mathrm{T}}$$

é o vetor de coeficientes de um filtro de resposta ao pulso finita (mais conhecido por FIR – *finite impulse response*) de ordem M, sendo M um parâmetro que o projetista deve escolher, e

$$\mathbf{u}(n) = [u(n) \quad u(n-1) \quad \cdots \quad u(n-M+1)]^{\mathrm{T}}$$

é denominado *vetor regressor de entrada*, constituído por amostras do sinal $u(n)$.

O algoritmo LMS é obtido por meio do método do gradiente, o que faz com que o ajuste do vetor de coeficientes seja feito no sentido contrário ao do gradiente de $\hat{J}(n)$ em relação a $\mathbf{w}(n-1)$. Substituindo (4) em (3), o cálculo desse gradiente é dado por

$$\boldsymbol{\nabla}_{\mathbf{w}}\hat{J}(n) = -2\mathbf{u}(n)d(n) + 2\mathbf{u}(n)\mathbf{u}^{\mathrm{T}}(n)\mathbf{w}(n-1) = -2\mathbf{u}(n)e(n). \tag{5}$$

E consequentemente a atualização de $\mathbf{w}(n-1)$ se torna

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu e(n)\mathbf{u}(n), \tag{6}$$

em que μ é chamado de "passo de adaptação", uma outra constante que o projetista também deve escolher. A influência da escolha de μ no comportamento do LMS será comentada mais adiante. A Tabela 1 contém as operações do algoritmo.

Tabela 1: O algoritmo LMS

Inicialização
$\mathbf{w}(0) = \mathbf{0}$
Cálculos
Para $n = 1,\ 2,\ \cdots,\ N$, calcule:
$y(n) = \mathbf{w}^{\mathrm{T}}(n-1)\mathbf{u}(n)$
$e(n) = d(n) - y(n)$
$\mathbf{w}(n) = \mathbf{w}(n-1) + \mu\mathbf{u}(n)e(n)$
Fim

O comportamento do LMS é altamente dependente do valor do passo de adaptação [1], tanto durante o transitório como em regime. Se μ for muito pequeno, o algoritmo atinge um erro quadrático pequeno em regime estacionário, mas demora a convergir. Conforme se aumenta μ, o algoritmo vai convergindo cada vez mais rapidamente. Entretanto, o patamar de erro quadrático atingido em regime permanente vai se tornando cada vez mais alto, até o ponto em que o algoritmo passa a divergir.

O valor de μ que faz o algoritmo divergir é função do número de elementos do vetor regressor e da potência do sinal de entrada. É possível mostrar [1, 3] que o intervalo de μ para que o algoritmo não divirja na média é

$$0 < \mu < \frac{2}{M\sigma_u^2}. \tag{7}$$

Este é um resultado teórico que é obtido quando se fazem algumas hipóteses simplificadoras acerca dos sinais envolvidos. Como nem sempre essas hipóteses valem na prática, é possível que o algoritmo divirja mesmo se escolhendo um valor de μ dentro desse intervalo.

Para facilitar a escolha do passo de adaptação, foi proposto o algoritmo LMS normalizado (NLMS – *Normalized LMS*), que utiliza um passo variante no tempo, dado por

$$\mu(n) = \frac{\widetilde{\mu}}{\delta + \|\mathbf{u}(n)\|^2}, \tag{8}$$

em que δ é uma constante positiva usada para evitar divisão por zero. Com esse passo

variante no tempo, a atualização de $\mathbf{w}(n-1)$ passa a ser dada por

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \frac{\tilde{\mu}}{\delta + ||\mathbf{u}(n)||^2}\mathbf{u}(n)e(n). \tag{9}$$

É possível mostrar que, para evitar a divergência, $\widetilde{\mu}$ deve ser escolhido no intervalo

$$0 < \widetilde{\mu} < 2. \tag{10}$$

Diferentemente do LMS, o intervalo do passo de adaptação do NLMS independe da potência do sinal de entrada. Isso torna mais simples a escolha desse parâmetro, já que em algumas aplicações, a potência do sinal de entrada pode variar com o tempo. A seguir, é mostrado um exemplo comparando os dois algoritmos.

Ambos os algoritmos foram empregados na identificação de um sistema ou *planta*, na terminologia de Controle, como mostrado na Figura 2. Nesse tipo de aplicação, alimenta-se o filtro adaptativo com a mesma entrada que o sistema que se quer identificar. O filtro tenta então reproduzir a saída desse sistema. O sinal $z(n)$ representa um ruído de medição, geralmente não correlacionado com a entrada $u(n)$. Espera-se que com o tempo $y(n)$ se aproxime de $x(n)$, e que consequentemente $e(n)$ se aproxime de $z(n)$.

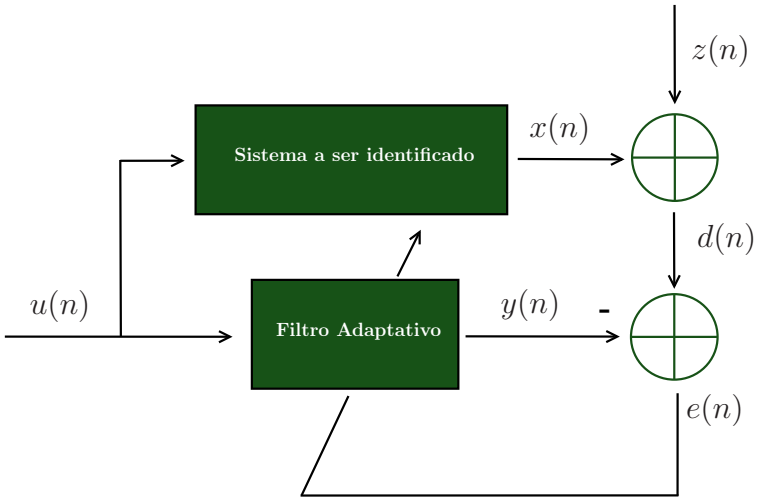


Figura 2: Aplicação de um filtro adaptativo na identificação de um sistema.

Foi feita uma simulação em que o sinal $u(n)$ é um ruído gaussiano branco, inicialmente com potência $\sigma_u^2 = 0{,}25$. Após 2500 iterações, a sua potência é bruscamente alterada para $\sigma_u^2 = 0{,}7656$. Para o cálculo do MSE, foram utilizadas 500 realizações. O sinal $u(n)$ é então

aplicado à entrada de um sistema linear cuja saída $x(n)$ é dada por

$$x(n) = \mathbf{w}_{\mathrm{o}}^{\mathrm{T}}\mathbf{u}(n),$$

em que se fez

$$\mathbf{w}_{\mathrm{o}} = [0{,}25 \quad 1{,}25 \quad 0{,}75 \quad 1 \quad 0{,}75 \quad 1{,}25 \quad 0{,}25]^{\mathrm{T}}.$$

O sinal $d(n)$ é dado por $d(n) = x(n) + z(n)$, em que $z(n)$ é um ruído gaussiano branco com variância $\sigma_z^2 = 0{,}0001$ não correlacionado com $u(n)$.

Na Figura 3, são mostradas curvas do erro quadrático médio dos algoritmos LMS e NLMS ao longo das iterações. Pode-se ver que, decorrido um certo tempo a partir da inicialização dos algoritmos, ambos atingem o mesmo valor de MSE em regime estacionário. A partir do instante em que a potência do sinal de entrada aumenta, o desempenho do algoritmo LMS piora consideravelmente. Os picos no gráfico são indícios de que, em algumas das realizações, o algoritmo se aproximou da divergência. Em contrapartida, pode-se ver que o MSE do algoritmo NLMS praticamente não se alterou.

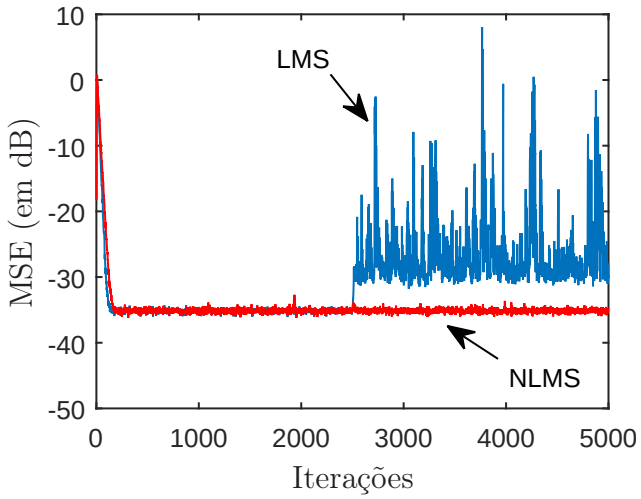


Figura 3: Curvas de erro quadrático médio dos algoritmos LMS e NLMS aplicados na identificação de um sistema linear cujo sinal de entrada $u(n)$ tem a sua potência bruscamente alterada de $\sigma_u^2 = 0{,}25$ para $\sigma_u^2 = 0{,}7656$.

Os algoritmos LMS e NLMS são capazes de atingir o ponto de mínimo global da função custo para aplicações de natureza linear. No entanto, em aplicações de natureza não linear, o desempenho dos algoritmos LMS e NLMS pode ser comprometido. Nesses casos é reco-

mendada a utilização de algoritmos não lineares, como os baseados em núcleo, alguns dos quais serão apresentados na Subseção 1.3.

Por fim, cabe notar que caso se tenha $\mathbf{w}(0) = \mathbf{0}$, a Equação (6) pode ser reescrita da seguinte forma:

$$\begin{aligned}
\mathbf{w}(n) &= \mathbf{w}(n-1) + \mu e(n)\mathbf{u}(n) \\
&= [\mathbf{w}(n-2) + \mu e(n-1)\mathbf{u}(n-1)] + \mu e(n)\mathbf{u}(n) \\
&= \mathbf{w}(n-2) + \mu[e(n-1)\mathbf{u}(n-1) + e(n)\mathbf{u}(n)] \\
&\vdots \\
&= \mathbf{w}(0) + \mu \sum_{j=1}^{n} e(j)\mathbf{u}(j) \\
&= \mu \sum_{j=1}^{n} e(j)\mathbf{u}(j).
\end{aligned} \tag{11}$$

Analogamente, para o algoritmo NLMS se obtém

$$\mathbf{w}(n) = \tilde{\mu} \sum_{j=1}^{n} e(j) \frac{\mathbf{u}(j)}{\delta + ||\mathbf{u}(j)||^2}. \tag{12}$$

Embora as Equações (11) e (12) representem modos menos eficientes de se atualizar o vetor $\mathbf{w}$ do que as Equações (6) e (9), elas podem ajudar a compreender o funcionamento do algoritmo KLMS, que será exposto mais adiante.

1.2 Núcleos

Esta seção começa com uma breve definição do que são os *kernels*, ou núcleos. Na Subseção 1.2.2, introduz-se o conceito de *kernels positivos definidos*. A Subseção 1.2.3 enuncia o *truque do kernel* e, por fim, a Subseção 1.2.4 apresenta o *kernel gaussiano*, que será usado em todo o restante do relatório.

Neste trabalho, optou-se por abordar apenas as propriedades dos *kernels* que serão utilizadas mais adiante ou que são fundamentais para tratar de outros temas posteriormente abordados. Contudo, a literatura sobre núcleos é vasta. Caso se deseje estudar esse tópico mais a fundo, as referências [4], [6] e [7] tratam de *kernels* com mais profundidade.

1.2.1 Definição

Seja uma função de mapeamento $\mathbf{\Phi}(\cdot)$, que leva elementos $\mathbf{x}$ de um espaço de entrada $\mathbb{U}$ a $\mathbf{\Phi}(\mathbf{x}) \in \mathbb{F}$, em que $\mathbb{F}$ é denominado "espaço das características" ou *feature space*. Define-se núcleo ou *kernel* como [7]

$$k(\mathbf{x},\mathbf{x}') = \mathbf{\Phi}^{\mathrm{T}}(\mathbf{x})\mathbf{\Phi}(\mathbf{x}'). \tag{13}$$

Assim, pode-se dizer que um *kernel* consiste em um produto interno realizado entre os seus vetores de entrada mapeados em $\mathbb{F}$ por meio da função $\mathbf{\Phi}(\cdot)$. Geralmente a dimensão de $\mathbb{F}$ é maior do que a de $\mathbb{U}$, podendo inclusive ser infinita.

É interessante notar que todo produto interno pode ser entendido como um *kernel*. A recíproca é sempre verdadeira no espaço $\mathbb{F}$, mas no espaço $\mathbb{U}$ isso não necessariamente ocorre, pois $k(\mathbf{x},\mathbf{x}')$ pode não satisfazer às propriedades necessárias para que um operador seja considerado um produto interno, uma vez que a função de mapeamento $\mathbf{\Phi}(\cdot)$ pode ser não linear. No contexto deste trabalho (e na maior parte da literatura citada), usam-se apenas funções de mapeamento desse tipo, pois pretende-se realizar filtragem adaptativa não linear.

1.2.2 *Kernel* Positivo Definido

Dada uma função $k : \mathbb{U}^2 \to \mathbb{K}$ (em que $\mathbb{K} = \mathbb{R}$ ou $\mathbb{K} = \mathbb{C}$) e dados $\mathbf{x}_1, \cdots, \mathbf{x}_m \in \mathbb{U}$, a matriz $\mathbf{K}$ $m \times m$ com elementos

$$\mathbf{K}_{ij} := k(\mathbf{x}_i,\mathbf{x}_j)$$

é definida como a Matriz de Gram de k com respeito a $\mathbf{x}_1,...,\mathbf{x}_m$ [7]. Se um dado *kernel* der origem a uma matriz de Gram positiva definida, ele é chamado de *kernel positivo definido*. Um *kernel* positivo definido também pode ser chamado de *kernel de Mercer* ou de *kernel reprodutor*[1] [7].

[1]Existe uma certa divergência na bibliografia levantada em relação a essa questão: em [4], os autores consideram que para que um *kernel* positivo definido possa ser considerado um *kernel* de Mercer ou *kernel* reprodutor, é necessário também que ele seja contínuo. Em [7], em contrapartida, os autores não fazem menção à continuidade da função *kernel*. Neste trabalho, exceto pelo exemplo de aplicação do Truque do *Kernel* na Subseção 1.2.3, só se utilizará o *kernel* gaussiano, que é contínuo.

1.2.3 Truque do *kernel*

O **método do *kernel*** é uma ferramenta de modelagem não paramétrica, cuja ideia principal consiste em calcular produtos internos em um espaço $\mathbb{F}$ de alta dimensão por meio de um *kernel* positivo definido, sem realizar explicitamente o mapeamento dos dados de entrada $\mathbf{x}$ por meio de $\mathbf{\Phi}(\mathbf{x})$ (ver Equação (13)). Contanto que uma operação possa ser formulada em termos de produtos internos ou de uma função de *kernel* equivalente, não há necessidade de se calcular as coordenadas de $\mathbf{\Phi}(\mathbf{x})$ [4]. A isso se dá o nome de **Truque do *Kernel***. Uma consequência do truque do *kernel* é que, dado um algoritmo formulado em termos de um *kernel* positivo definido k, pode-se construir um algoritmo alternativo substituindo-se k por outro *kernel* positivo definido $\tilde{k}$ [7].

Consequentemente, substituindo-se o produto interno convencional no espaço de entrada $\mathbb{U}$ por um *kernel* relacionado a uma função de mapeamento $\mathbf{\Phi}(\cdot)$ não linear, pode-se realizar uma filtragem que é não linear em $\mathbb{U}$ adotando procedimentos que são lineares em $\mathbb{F}$. É possível fazer isso porque, como mencionado anteriormente, todo produto interno pode ser entendido como um *kernel*. A Figura 4 mostra um exemplo, extraído de [7].

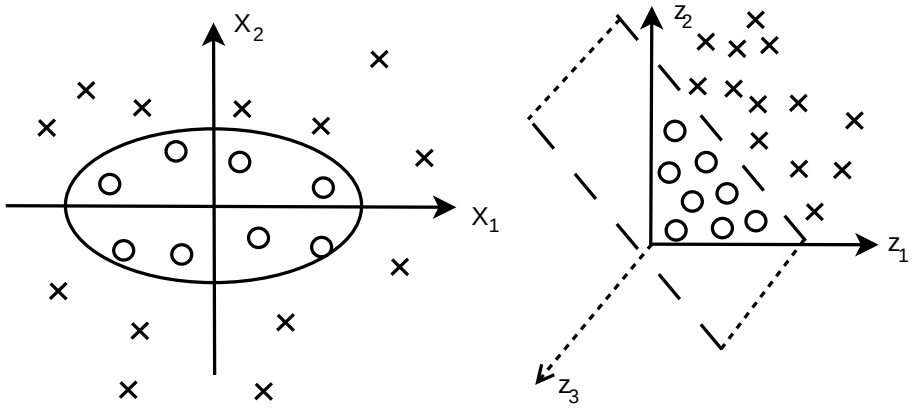


Figura 4: Exemplo de aplicação do truque do *kernel* extraído de [7].

À esquerda, tem-se o espaço de entrada $\mathbb{R}^2$ com alguns de seus elementos representados. Deseja-se separar os elementos representados por "$\circ$" daqueles representados por "$\times$". Como se pode ver, não é possível separá-los linearmente; a maneira mais fácil de realizar essa separação no caso é com uma elipse. À direita, tem-se o espaço das características $\mathbb{F}$ (que no caso equivale a $\mathbb{R}^3$) onde os elementos do espaço $\mathbb{U}$ são mapeados através da função

$\mathbf{\Phi} : \mathbb{U} = \mathbb{R}^2 \rightarrow \mathbb{F} = \mathbb{R}^3, (x_1,x_2) \mapsto (x_1^2,x_2^2,\sqrt{2}x_1x_2)$. Nesse espaço, existe uma solução linear para o problema em questão: a elipse passa a ser representada pelo plano desenhado na figura. Isso ocorre porque é possível descrever elipses como equações lineares em função de x_1^2, x_2^2 e x_1x_2, ou seja, em função dos elementos da função de mapeamento $\mathbf{\Phi}(\mathbf{x})$.

Outro fato interessante ilustrado por esse exemplo é que, para a função $\mathbf{\Phi}(\cdot)$ apresentada, tem-se:

$$k(\mathbf{x},\mathbf{x}') = \mathbf{\Phi}^{\mathrm{T}}(\mathbf{x})\mathbf{\Phi}(\mathbf{x}') = x_1^2x_1'^2 + x_2^2x_2'^2 + 2x_1x_1'x_2x_2' = (\mathbf{x}^{\mathrm{T}}\mathbf{x}')^2. \tag{14}$$

Em outras palavras, escolhendo-se essa função de mapeamento, o *kernel* fica sendo simplesmente o quadrado do produto interno convencional calculado em $\mathbb{U}$. Trata-se, portanto, de um exemplo de aplicação do truque do *kernel*: o produto interno em $\mathbb{F}$ pode ser calculado sem que seja necessário realizar o mapeamento explicitamente, bastando para isso calcular o produto interno convencional no espaço de entrada e elevá-lo ao quadrado. Isso representa uma grande vantagem, pois geralmente $\mathbb{F}$ terá uma dimensionalidade elevada (podendo ser infinita), e caso fosse necessário trabalhar nesse espaço, o custo computacional poderia ser muito elevado. Na prática, só são utilizados *kernels* que podem ser calculados sem que se tenha que realizar o mapeamento explicitamente.

O truque do *kernel* é a base dos algoritmos adaptativos baseados em núcleo. Observando a Equação (4), vê-se que a saída do filtro gerado pelo algoritmo LMS foi definida em termos de um produto interno. Isso abre espaço para a existência de uma versão "*kernelizada*" do LMS e, consequentemente, do NLMS. Esses algoritmos são denotados, respectivamente, pelas siglas KLMS e KNLMS.

1.2.4 O *kernel* gaussiano

O ***kernel* gaussiano** pode ser escrito da seguinte forma:

$$k(\mathbf{x},\mathbf{x}') = e^{(-a||\mathbf{x}-\mathbf{x}'||^2)}, \tag{15}$$

também podendo ser escrito como

$$k(\mathbf{x},\mathbf{x}') = e^{\frac{(-||\mathbf{x}-\mathbf{x}'||^2)}{2h^2}}, \tag{16}$$

em que $h = \frac{1}{\sqrt{2a}}$ é denominado **largura do *kernel***.

É interessante notar que a dimensionalidade do espaço $\mathbb{F}$ gerado por um *kernel* gaussiano é **infinita**. Além do núcleo gaussiano, há vários outros, como o sigmoidal, o polinomial não homogêneo e o polinomial homogêneo. Cabe notar que este último foi utilizado no exemplo de aplicação da Subseção 1.2.3. No entanto, o *kernel* gaussiano é um dos mais utilizados e mais estudados na literatura, e por esse motivo, será o único utilizado neste relatório.

1.3 Os algoritmos KLMS e KLMS2

A seguir, mostra-se como obter o algoritmo KLMS a partir do LMS. Os passos seguidos são os mesmos de [4]. Pode-se demonstrar [8] que, no caso do *kernel* gaussiano, dada qualquer função de mapeamento $f : \mathbb{U} \to \mathbb{R}$ contínua, sendo $\mathbb{U}$ o espaço de entrada, existem parâmetros $\{\boldsymbol{\alpha}_i\}_{i=1}^m \in \mathbb{U}$ e números reais $\{a_i\}_{i=1}^m$ tais que, para qualquer $\zeta > 0$,

$$||f(\cdot) - \sum_{i=1}^{m} a_i k(\cdot, \boldsymbol{\alpha}_i)||_2 < \zeta. \tag{17}$$

Seja o vetor $\boldsymbol{\omega} \in \mathbb{F}$ dado por

$$\boldsymbol{\omega} = \sum_{i=1}^{m} a_i \boldsymbol{\Phi}(\boldsymbol{\alpha}_i). \tag{18}$$

Então, pelas equações (13) e (17), tem-se

$$||f(\mathbf{u}) - \boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{\Phi}(\mathbf{u})||_2 < \zeta. \tag{19}$$

De modo geral, a ideia do algoritmo KLMS é que $\boldsymbol{\omega}^{\mathrm{T}} \boldsymbol{\Phi}(\mathbf{u})$ pode ser um modelo mais poderoso do que o produto interno $\mathbf{w}^{\mathrm{T}}\mathbf{u}$ do LMS devido à diferença de dimensão entre $\boldsymbol{\Phi}(\mathbf{u}(n))$ e $\mathbf{u}(n)$.

Abusando ligeiramente da notação e denotando, por simplicidade, $\boldsymbol{\Phi}(\mathbf{u}(n))$ por $\boldsymbol{\Phi}(n)$, a aplicação do LMS utilizando $\{\boldsymbol{\Phi}(n), d(n)\}$ fornece

$$\begin{aligned}
&\boldsymbol{\omega}(0) = \mathbf{0} \\
&e(n) = d(n) - \boldsymbol{\omega}^{\mathrm{T}}(n-1)\boldsymbol{\Phi}(n) \\
&\boldsymbol{\omega}(n) = \boldsymbol{\omega}(n-1) + \mu e(n)\boldsymbol{\Phi}(n).
\end{aligned}$$

Repetindo o procedimento acima recursivamente, assim como foi feito para o algoritmo LMS na Subseção 1.1, obtém-se

$$\begin{aligned}
\boldsymbol{\omega}(n) &= \boldsymbol{\omega}(n-1) + \mu e(n)\boldsymbol{\Phi}(n) \\
&= [\boldsymbol{\omega}(n-2) + \mu e(n-1)\boldsymbol{\Phi}(n-1)] + \mu e(n)\boldsymbol{\Phi}(n) \\
&= \boldsymbol{\omega}(n-2) + \mu[e(n-1)\boldsymbol{\Phi}(n-1) + e(n)\boldsymbol{\Phi}(n)] \\
&\vdots \\
&= \boldsymbol{\omega}(0) + \mu \sum_{j=1}^{n} e(j)\boldsymbol{\Phi}(j) \\
&= \mu \sum_{j=1}^{n} e(j)\boldsymbol{\Phi}(j),
\end{aligned} \tag{20}$$

assumindo $\boldsymbol{\omega}(0) = \mathbf{0}$. A saída do filtro, dada uma nova entrada $\mathbf{u}'$, pode ser expressa com base em produtos internos por meio de

$$\begin{aligned}
\boldsymbol{\omega}^{\mathrm{T}}(n-1)\boldsymbol{\Phi}(\mathbf{u}') &= [\mu \sum_{j=1}^{n-1} e(j)\boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{u}(j))]\boldsymbol{\Phi}(\mathbf{u}') \\
&= \mu \sum_{j=1}^{n-1} e(j)[\boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{u}(j))\boldsymbol{\Phi}(\mathbf{u}')].
\end{aligned} \tag{21}$$

Agora, aplicando o truque do *kernel*, é possível efetuar o cálculo acima eficientemente por meio de

$$\boldsymbol{\omega}^{\mathrm{T}}(n-1)\boldsymbol{\Phi}(\mathbf{u}') = \mu \sum_{j=1}^{n-1} e(j)k(\mathbf{u}(j), \mathbf{u}'). \tag{22}$$

A Equação (22) representa uma estimativa da função de mapeamento f mencionada anteriormente. Se a estimativa no instante n for denotada por f_n, tem-se

$$f_{n-1}(\cdot) = \mu \sum_{j=1}^{n-1} e(j)k(\mathbf{u}(j), \cdot)$$

$$f_{n-1}(\mathbf{u}(n)) = \mu \sum_{j=1}^{n-1} e(j)k(\mathbf{u}(j), \mathbf{u}(n))$$

$$e(n) = d(n) - f_{n-1}(\mathbf{u}(n)).$$

Estas equações formam a base do algoritmo KLMS, cujas operações são mostradas na Tabela 2.

Tabela 2: O algoritmo KLMS [4]

Inicialização

$\mathbf{a}_1 = \mu d(1)$, $C = \{\mathbf{u}(1)\}$, $y_1 = \mathbf{a}_1(1)k(\mathbf{u}(1),\cdot)$

Cálculos

Para $n = 2, 3, \cdots, N$, calcule:

% Cálculo da saída:

$f_{n-1}(\mathbf{u}(n)) = \sum_{j=1}^{n-1} \mathbf{a}_j(n-1)k(\mathbf{u}(n), \mathbf{u}(j))$

% Cálculo do erro:

$e(n) = d(n) - f_{n-1}(\mathbf{u}(n))$

% Armazenamento do novo elemento:

$C = \{C(n-1), \mathbf{u}(n)\}$

% Atualização do vetor de coeficientes:

$\mathbf{a}_n(n) = \mu e(n)$

Fim

O vetor $\mathbf{a}$ é o vetor de pesos do algoritmo KLMS, sendo que $a_j(n)$ denota o j-ésimo elemento do vetor na n-ésima iteração. C é denominado o *dicionário* do algoritmo. Como se pode ver, o termo $f_{n-1}(\mathbf{u}(n))$ é equivalente à saída $y(n)$ do LMS. Embora no início da seção tenhamos restringido o *kernel* ao caso gaussiano, o KLMS também pode ser deduzido utilizando outros núcleos.

Tal como originalmente proposto, o KLMS inclui a cada iteração o *kernel* referente ao vetor atual de entrada $k(\cdot,\mathbf{u}(n))$ no dicionário, e a cada iteração todos os elementos anteriores de C são utilizados para calcular a saída. Isso significa que a cada iteração o tempo de execução do algoritmo aumenta. Por esse motivo, o KLMS tal como definido acima dificilmente poderia ser utilizado em uma aplicação em tempo real. Existem métodos para a restrição do tamanho do dicionário que podem viabilizar a implementação do algoritmo nesse tipo de aplicação. Esses métodos serão discutidos mais adiante neste relatório.

Existe um outro algoritmo, também chamado de KLMS na literatura consultada [9–12],

cuja saída é definida como

$$y(n) = \boldsymbol{\alpha}^{\mathrm{T}}(n-1)\mathbf{k}(n), \tag{23}$$

em que $\boldsymbol{\alpha}$ é um vetor de coeficientes e $\mathbf{k}(n)$ é o vetor dado por

$$\mathbf{k}(n) = [k(\mathbf{u}(n),\, \mathbf{u}(c_1)),\; k(\mathbf{u}(n),\mathbf{u}(c_2)),\; \cdots,\; k(\mathbf{u}(n),\mathbf{u}(c_L))]^{\mathrm{T}}, \tag{24}$$

sendo $\mathbf{u}(n)$ o vetor de entrada no instante n e $k(\cdot,\mathbf{u}(c_1))$, $k(\cdot,\mathbf{u}(c_2))$, $\ldots$, $k(\cdot,\mathbf{u}(c_L))$ os L elementos atualmente armazenados no dicionário. Note-se que o vetor $\mathbf{k}(n)$ já foi escrito admitindo-se a possibilidade de que nem todas as entradas sejam adicionadas ao dicionário, restringindo o seu crescimento.

Mantendo as definições de $e(n) = d(n) - y(n)$ e de $\hat{J}(n) = e^2(n)$ utilizadas anteriormente e calculando o gradiente de $\hat{J}(n)$ em relação a $\boldsymbol{\alpha}(n-1)$, de modo similar ao que foi feito na Seção 1.1, obtém-se

$$\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\hat{J}(n) = -2\mathbf{k}(n)d(n) + 2\mathbf{k}(n)\mathbf{k}^{\mathrm{T}}(n)\boldsymbol{\alpha}(n-1) = -2\mathbf{k}(n)e(n). \tag{25}$$

Portanto, a atualização do vetor $\boldsymbol{\alpha}$ é dada por

$$\boldsymbol{\alpha}(n) = \boldsymbol{\alpha}(n-1) + \mu e(n)\mathbf{k}(n). \tag{26}$$

Note que, caso se deseje aumentar o tamanho do dicionário em uma dada iteração, é necessário incrementar as dimensões de $\boldsymbol{\alpha}(n)$ e de $\mathbf{k}(n)$.

Neste trabalho, por simplicidade, o algoritmo deduzido em [4] será denotado por KLMS, ao passo que aquele empregado em [9–12] será denotado por KLMS2.

Pode-se notar que as equações (6) e (26) são muito semelhantes, com a diferença de que, em vez de $\mathbf{w}(n-1)$ e $\mathbf{u}(n)$, são utilizados $\boldsymbol{\alpha}(n-1)$ e $\mathbf{k}(n)$, respectivamente. As operações do algoritmo KLMS2 estão listadas na Tabela 3.

1.4 As versões normalizadas

Assim como no caso do algoritmo LMS, os algoritmos KLMS e KLMS2 possuem versões com o passo de adaptação normalizado em $\mathbb{F}$, as quais foram chamadas indistintamente de

Tabela 3: O algoritmo KLMS2 [9–12]

Inicialização

$\boldsymbol{\alpha}(0) = \mathbf{0}$

Cálculos

Para $n = 1, 2, \cdots, N$, calcule:

$$\mathbf{k}(n) = [k(\mathbf{u}(n), \mathbf{u}(c_1)), \; k(\mathbf{u}(n),\mathbf{u}(c_2)), \; \vdots \; k(\mathbf{u}(n),\mathbf{u}(c_L))]^{\mathrm{T}}$$

$$y(n) = \boldsymbol{\alpha}^{\mathrm{T}}(n-1)\mathbf{k}(n)$$

$$e(n) = d(n) - y(n)$$

$$\boldsymbol{\alpha}(n) = \boldsymbol{\alpha}(n-1) + \mu\mathbf{k}(n)e(n)$$

Fim

KNLMS na literatura consultada. A seguir, mostra-se como as versões normalizadas de ambos os algoritmos foram obtidas.

Em [4], por analogia com o caso linear, propõe-se que o cálculo da saída leve em conta a norma do vetor $\boldsymbol{\Phi}(\mathbf{u}(n))$, ou seja,

$$f_{n-1}(\mathbf{u}(n)) = \sum_{j=1}^{n-1} \mathbf{a}_j(n-1)\frac{k(\mathbf{u}(n), \mathbf{u}(j))}{\boldsymbol{\Phi}^{\mathrm{T}}(\mathbf{u}(j))\boldsymbol{\Phi}(\mathbf{u}(j)}.$$

Pela Equação (13), tem-se que

$$f_{n-1}(\mathbf{u}(n)) = \sum_{j=1}^{n-1} \mathbf{a}_j(n-1)\frac{k(\mathbf{u}(n), \mathbf{u}(j))}{k(\mathbf{u}(j), \mathbf{u}(j))}. \tag{27}$$

Note que para o *kernel* gaussiano $k(\mathbf{x},\mathbf{x}) = 1$, qualquer que seja $\mathbf{x}$. Isso implica que, ao se escolher esse núcleo, o algoritmo KLMS está automaticamente normalizado [4].

Por sua vez, a versão normalizada do algoritmo KLMS2 adota um passo variante no tempo na equação de atualização do vetor $\boldsymbol{\alpha}$, dado por [9]

$$\mu(n) = \frac{\tilde{\mu}}{\delta + \mathbf{k}^{\mathrm{T}}(n)\mathbf{k}(n)}, \tag{28}$$

em que δ é uma constante positiva usada para evitar divisão por zero.

Novamente, a equação obtida é similar àquela deduzida para o caso linear (ver Equação (8)), com a única diferença de que é a norma do vetor $\mathbf{k}$ que está sendo utilizada para normalizar o passo de adaptação, em vez da norma do vetor regressor $\mathbf{u}(n)$.

Uma vez que o algoritmo KNLMS coincide com o KLMS quando se utiliza o *kernel* gaussiano, e neste relatório só se trabalhará com esse núcleo, optou-se por denotá-lo também por "KLMS" deste ponto em diante. A versão normalizada do KLMS2, por sua vez, será denotada no restante do relatório por KNLMS2.

2 Objetivos

As soluções baseadas em núcleo foram propostas recentemente na literatura, e portanto ainda têm muitas características a serem estudadas. Identificação de sistemas não lineares e equalização em sistemas de comunicação com distorções não lineares são exemplos de aplicações em que soluções desse tipo tendem a apresentar um desempenho mais satisfatório do que as lineares.

A literatura contém duas versões distintas dos algoritmos KLMS, aqui denominadas de KLMS e KLMS2. Essas versões apresentam diferenças em termos de desempenho sobre as quais não se encontrou nenhuma menção na bibliografia levantada. Este trabalho tem como objetivo expor essas discrepâncias e em seguida empregar esses algoritmos nas aplicações citadas, investigando algumas de suas propriedades como velocidade de convergência e desempenho em regime e tecendo comparações entre eles.

3 Metodologia

Este relatório busca proporcionar uma melhor compreensão acerca do comportamento dos algoritmos de filtragem adaptativa baseados em núcleo, a fim de facilitar tanto as suas aplicações em situações práticas como estudos posteriores.

Dessa forma, buscou-se investigar teórica e empiricamente propriedades desses algoritmos. Como eles são relativamente recentes (o KLMS data de 2008), muitas de suas caracte-

rísticas ainda estão sendo estudadas. Embora já haja uma literatura significativa a respeito do assunto, esses algoritmos ainda estão em processo de difusão e são pouco conhecidos em comparação com algoritmos de filtragem adaptativa linear, como o LMS, ou com alguns outros métodos de resolução de problemas não lineares, como as redes neurais.

O trabalho realizado consistiu no levantamento bibliográfico relativo ao assunto, no desenvolvimento teórico e na realização de simulações computacionais utilizando um microcomputador e o *software* Matlab com os pacotes específicos de processamento de sinais. Cabe observar que os algoritmos baseados em núcleo sofrem influência de muitos parâmetros, como inicialização, passo de adaptação, largura do *kernel*, entre outros. Diante disso, definiu-se um roteiro de simulações para estudar a influência de cada parâmetro da maneira mais independente possível da dos demais.

4 Resultados Finais

Esta seção foi dividida em três subseções. Na Subseção 4.1 são mostradas diferenças entre os algoritmos KLMS e KLMS2. Já as outras duas subseções focam em aplicações práticas, mostrando alguns resultados obtidos por meio de simulações. Na Subseção 4.2 o KLMS, o KLMS2 e o KNLMS2 são empregados à identificação de um sistema não linear e se investigam algumas de suas propriedades, tais como tempo de convergência e erro quadrático médio em regime e como os seus comportamentos são influenciados pelos valores dos seus parâmetros. Na Subseção 4.3 o KLMS e o KNLMS2 são utilizados para equalizar um canal de comunicações com distorções não lineares, e seus desempenhos são comparados. Por fim, o Apêndice A apresenta versões dos algoritmos KLMS e KLMS2 adaptadas à equalização de um canal de um sistema de comunicação baseado em caos e os resultados de algumas simulações computacionais empregando essas versões propostas.

4.1 Diferenças entre os Algoritmos KLMS e KLMS2

Apesar de ambos os algoritmos baseados em núcleo expostos anteriormente terem sido chamados indistintamente de KLMS na literatura, eles não são iguais, mesmo quando seus

dicionários são idênticos. A seguir será mostrado que, de fato, são algoritmos distintos.

Note que a equação de atualização do vetor $\boldsymbol{\omega}(n)$ do algoritmo KLMS é dada por

$$\boldsymbol{\omega}(n) = \boldsymbol{\omega}(n-1) + \mu e(n)\boldsymbol{\Phi}(n).$$

Transpondo os vetores e multiplicando os dois lados da equação por $\boldsymbol{\Phi}(n)$ à direita, obtém-se

$$\boldsymbol{\omega}^{\mathrm{T}}(n)\boldsymbol{\Phi}(n) = \boldsymbol{\omega}^{\mathrm{T}}(n-1)\boldsymbol{\Phi}(n) + \mu e(n)\boldsymbol{\Phi}^{\mathrm{T}}(n)\boldsymbol{\Phi}(n). \tag{29}$$

Aplicando o Truque do Kernel, tem-se

$$\boldsymbol{\omega}^{\mathrm{T}}(n)\boldsymbol{\Phi}(n) = \boldsymbol{\omega}^{\mathrm{T}}(n-1)\boldsymbol{\Phi}(n) + \mu e(n)k(\mathbf{u}(n),\mathbf{u}(n)). \tag{30}$$

Assumindo que as saídas dos algoritmos sejam iguais em todo instante de tempo (ou seja, que $\boldsymbol{\alpha}^{\mathrm{T}}(n-1)\mathbf{k}(n) = \boldsymbol{\omega}^{\mathrm{T}}(n-1)\boldsymbol{\Phi}(n)$) e que a aproximação $\boldsymbol{\alpha}^{\mathrm{T}}(n)\mathbf{k}(n) \approx \boldsymbol{\omega}^{\mathrm{T}}(n)\boldsymbol{\Phi}(n)$ seja válida, a Equação (30) fornece

$$\boldsymbol{\alpha}^{\mathrm{T}}(n)\mathbf{k}(n) = \boldsymbol{\alpha}^{\mathrm{T}}(n-1)\mathbf{k}(n) + \mu e(n)k(\mathbf{u}(n),\mathbf{u}(n)). \tag{31}$$

Em contrapartida, transpondo vetores dos dois lados da Equação (26) e multiplicando à direita por $\mathbf{k}(n)$, obtém-se

$$\boldsymbol{\alpha}^{\mathrm{T}}(n)\mathbf{k}(n) = \boldsymbol{\alpha}^{\mathrm{T}}(n-1)\mathbf{k}(n) + \mu e(n)\mathbf{k}^{\mathrm{T}}(n)\mathbf{k}(n). \tag{32}$$

Contudo, nota-se que

$$k(\mathbf{u}(n),\mathbf{u}(n)) \neq \mathbf{k}^{\mathrm{T}}(n)\mathbf{k}(n) = k(\mathbf{u}(n),\mathbf{u}(1))^2 + k(\mathbf{u}(n),\mathbf{u}(2))^2 + \cdots + k(\mathbf{u}(n),\mathbf{u}(n-1))^2. \tag{33}$$

Isto mostra que os algoritmos são diferentes entre si.

Uma outra forma de mostrar que os algoritmos são diferentes consiste em reescrever o somatório do cálculo da saída do algoritmo KLMS como o produto interno entre dois vetores, assim como no algoritmo KLMS2. Fazendo isso, obtém-se

$$f_{n-1}(\mathbf{u}(n)) = \mathbf{a}^{\mathrm{T}}(n-1)\mathbf{k}(n), \tag{34}$$

em que o vetor $\mathbf{k}(n)$ é dado pela Equação (24) no caso em que os novos elementos são sempre adicionados ao dicionário (ou seja, $\mathbf{u}(c_1) = \mathbf{u}(1),\ \mathbf{u}(c_2) = \mathbf{u}(2),\ \cdots,\ \mathbf{u}(c_L) = \mathbf{u}(n-1)$), e

$$\mathbf{a}(n) = [\mu e(1),\ \mu e(2),\ \cdots,\ \mu e(n-1)]^{\mathrm{T}}.$$

Assumindo que as saídas dos dois algoritmos sejam iguais quando estes são aplicados ao mesmo problema utilizando os mesmos parâmetros μ e h e sem nenhum método de restrição do dicionário, e comparando-se as Equações (34) e (23), vê-se que é necessário que

$$\mathbf{a}(n) = \boldsymbol{\alpha}(n). \tag{35}$$

O vetor $\boldsymbol{\alpha}(n)$ é atualizado de acordo com a Equação (26). Assim, assumindo que a cada iteração a entrada atual é incluída no dicionário e que $\boldsymbol{\alpha}(0) = 0$, obtém-se

$$\begin{aligned}
\boldsymbol{\alpha}(n) &= \left[\mu \sum_{i=1}^{n-1} e(i)k(\mathbf{u}(n),\mathbf{u}(i)),\ \mu \sum_{i=2}^{n-1} e(i)k(\mathbf{u}(n),\mathbf{u}(i)),\ \cdots,\ \mu e(n-1)k(\mathbf{u}(n),\mathbf{u}(n-1))\right]^{\mathrm{T}} \\
&= \Bigg[\mu\left(e(1)k(\mathbf{u}(n),\mathbf{u}(1)) + \sum_{i=2}^{n-1} e(i)k(\mathbf{u}(n),\mathbf{u}(i))\right), \\
&\quad \mu\left(e(2)k(\mathbf{u}(n),\mathbf{u}(2)) + \sum_{i=3}^{n-1} e(i)k(\mathbf{u}(n),\mathbf{u}(i))\right), \\
&\quad \cdots, \\
&\quad \mu e(n-1)k(\mathbf{u}(n),\mathbf{u}(n-1))\Bigg]^{\mathrm{T}}.
\end{aligned} \tag{36}$$

Portanto, para que a Igualdade (34) seja verdadeira, é preciso que

$$\begin{aligned}
e(1) &= e(1)k(\mathbf{u}(n),\mathbf{u}(1)) + \sum_{i=2}^{n-1} e(i)k(\mathbf{u}(n),\mathbf{u}(i)), \\
e(2) &= e(2)k(\mathbf{u}(n),\mathbf{u}(2)) + \sum_{i=3}^{n-1} e(i)k(\mathbf{u}(n),\mathbf{u}(i)), \\
&\vdots \\
e(n-2) &= e(n-2)k(\mathbf{u}(n),\mathbf{u}(n-2) + e(n-1)k(\mathbf{u}(n),\mathbf{u}(n-1)), \\
e(n-1) &= e(n-1)k(\mathbf{u}(n),\mathbf{u}(n-1)).
\end{aligned} \tag{37}$$

Note que, como foi assumido que os dois algoritmos fornecem saídas iguais quando aplicados a situações idênticas, os sinais de erros devem ser iguais tanto para o KLMS como para o KLMS2. Assim, comparando-se os dois lados de cada igualdade do Sistema de Equações (37), tem-se que é preciso que

$$
\begin{aligned}
&k(\mathbf{u}(n),\mathbf{u}(n-1)) = 1,\\
&e(n-2) = k(\mathbf{u}(n),\mathbf{u}(n-2))e(n-2) + e(n-1)\\
&\vdots\\
&e(n-i) = k(\mathbf{u}(n),\mathbf{u}(n-i))e(n-i) + e(n-i+1),\ 1 < i < n,\\
&\vdots\\
&e(1) = k(\mathbf{u}(n),\mathbf{u}(1))e(1) + e(2).
\end{aligned}
\tag{38}
$$

Pode-se constatar que o Sistema de Equações (38) representa uma restrição severa, pois exige relações bastante específicas entre os valores de $\mathbf{k}(n)$ e $e(n)$ em todas as iterações de 1 até n. Cabe notar que não se tem controle sobre esses dois sinais, pois $e(n)$ depende do sinal desejado $d(n)$, que não pode ser alterado pelo filtro, e $\mathbf{k}(n)$ depende da entrada $\mathbf{u}(n)$. Pode-se argumentar que seria possível escolher um valor de largura de *kernel* que forçasse alguma das equações do sistema a ser válida (por exemplo, escolhendo um valor de h suficientemente elevado a equação $k(\mathbf{u}(n),\mathbf{u}(n-1)) = 1$ poderia ser aproximadamente satisfeita para quaisquer valores de $\mathbf{u}(n)$ e $\mathbf{u}(n-1)$), mas em qualquer situação prática seria impossível encontrar um valor de largura do *kernel* que tornasse todas as equações do Sistema (38) verdadeiras.

Uma vez que o Sistema (38) não vale no caso geral (ou em qualquer situação realista), pode-se concluir que o KLMS e o KLMS2 são algoritmos distintos. Consequentemente, suas versões normalizadas também apresentam diferenças entre si, muito embora, como mencionado anteriormente, tenham ambas sido chamadas indistintamente de KNLMS na literatura.

Por fim, cabe ressaltar que tanto o KLMS como o KLMS2 podem apresentar desempenhos satisfatórios em aplicações que exigem filtragem adaptativa não linear [4, 9–12].

4.2 Identificação de Sistemas

Optou-se por dividir esta subseção em quatro subseções menores, cada uma estudando o impacto de um dos parâmetros dos algoritmos KLMS, KLMS2 e KNLMS2 utilizando o

kernel gaussiano.

Na Subseção 4.2.1, o passo de adaptação e a largura do *kernel* foram escolhidos de maneira mais ou menos arbitrária, e investigou-se o impacto da inicialização do vetor de pesos no desempenho do algoritmo como um todo em diferentes situações. Identificada a melhor inicialização para o algoritmo na situação de maior interesse, ela foi adotada na subseção seguinte, que visava a estudar a influência do passo de adaptação e da escolha entre as versões normalizadas e não normalizadas, ainda mantendo a largura do *kernel* fixa. Na Subseção 4.2.3, que investiga a influência da largura do *kernel* gaussiano, foi necessário alterar o passo de adaptação, além da própria largura, para mostrar um aspecto importante dos algoritmos estudados. Por fim, a Subseção 4.2.4 versa sobre métodos de restrição do dicionário a fim de viabilizar a utilização dos algoritmos em aplicações em tempo real. Para agilizar a execução das simulações, o crescimento do dicionário foi restringido nas subseções 4.2.1, 4.2.2 e 4.2.3 utilizando a técnica adotada em [10], que consiste em manter um número fixo de elementos para o dicionário, descartando-se a cada iteração o elemento mais antigo e incluindo a entrada atual. Foi utilizado um vetor regressor com um único elemento em todas as simulações (ou seja, adotou-se $M = 1$).

4.2.1 Inicialização

Esta subseção investiga a influência da inicialização do vetor de pesos $\boldsymbol{\alpha}$ no desempenho do algoritmo KLMS2. Muitos algoritmos de filtragem não linear apresentam pontos de mínimos locais que prejudicam o seu desempenho, pois dependendo da inicialização do vetor de pesos, o erro quadrático pode convergir para um desses pontos de mínimos locais em vez de convergir para o mínimo global. De acordo com [4], o algoritmo KLMS não apresenta pontos de mínimos locais, e portanto seu desempenho em regime não seria afetado pela inicialização.

Nos testes, aplicou-se o KLMS2 para tentar identificar o seguinte sistema (ver Figura 2 na Subseção 1.1), extraído de [10]:

$$x(n) = \frac{x(n-1)}{1 + x^2(n-1)} + u^3(n-1), \tag{39}$$

em que $u(n)$ é um sinal gaussiano branco de média zero e desvio padrão 0,15, e com $x(0) = 0$.

O sinal $d(n)$ é obtido pela soma de $x(n)$ com um ruído gaussiano branco de média nula e $\sigma = 0{,}01$.

Testaram-se inicializações com um vetor de zeros, com um vetor de uns, com um vetor aleatório (com distribuição normal, média nula e $\sigma = 0{,}1$), e utilizando um *script* escrito para o Matlab e incluso em [10]. O gráfico com os erros quadráticos médios em função das iterações pode ser visto a seguir, e a Tabela 4 resume os resultados obtidos. Foram realizadas 500 experiências com 5000 iterações cada. Para facilitar a visualização, as curvas de MSE foram filtradas usando a função "filtfilt" do Matlab, considerando filtfilt(ones(1,8),8,MSE). Os valores da largura do *kernel* h, do passo de adaptação μ e do tamanho do dicionário foram mantidos constantes em 0,0225, 0,0393 e 6, respectivamente (valores extraídos de [10]).

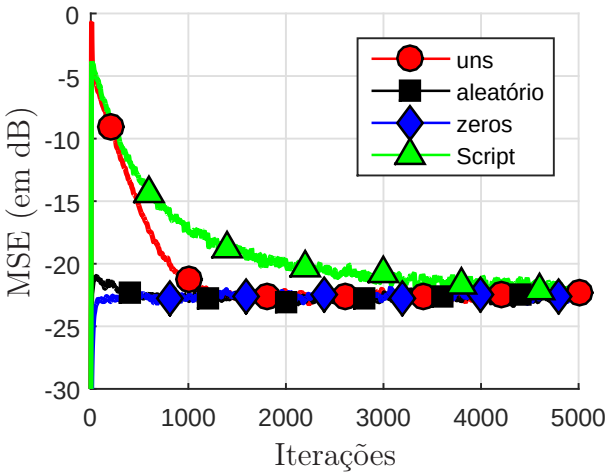


Figura 5: Curvas de erro quadrático médio do algoritmo KLMS2 com diferentes inicializações aplicado à identificação de um sistema regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Na legenda, *Script* se refere ao programa em Matlab para inicialização usado em [10].

Observando a figura, vê-se claramente que a inicialização afeta a velocidade com que se atinge o regime permanente. A inicialização com vetor de zeros levou à convergência praticamente instantânea do algoritmo, o que não aconteceu quando se inicializou o vetor de pesos de outras maneiras. A inicialização aleatória apresentou uma convergência ligeiramente mais tardia, ao passo que as inicializações com vetor de uns e com a do *script* de [10] fizeram com que o algoritmo demorasse mais para convergir, sendo que esta última foi a que apresentou o pior resultado. Contudo, a forma como se inicializou o algoritmo não parece ter

influenciado o MSE atingido em regime estacionário, o que indica que o algoritmo KLMS2 também apresenta a convexidade mencionada em [4]. Para verificar esse fato, a Tabela 4 mostra os valores de MSE calculados utilizando as últimas 500 iterações de cada realização. A ligeira discrepância entre o resultado obtido com a inicialização utilizando o *Script* de [10] e os demais se deve ao fato de que essa inicialização levou a uma convergência mais lenta, de modo que o algoritmo ainda estava terminando de convergir durante as 500 últimas iterações, período em que se calculou o MSE.

Tabela 4: MSE estacionário (em dB, calculado usando as 500 últimas iterações de 500 realizações) do algoritmo KLMS2 aplicado na identificação de um sistema não linear dado pela Equação (39), sendo a entrada $u(n)$ um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

Inicialização	MSE (em dB)
Com zeros	-22,5859
Com uns	-22,4505
Aleatória	-22,6446
Extraída do *Script* de [10]	-21,8974

A fim de averiguar a convexidade do KLMS alegada em [4], optou-se por também realizar simulações com esse algoritmo inicializado de diferentes formas. Como o *Script* de [10] foi escrito tendo-se em mente o algoritmo KLMS2, essa inicialização não foi incluída na simulação.

Foram testadas inicializações com zero ($\mathbf{a}(1) = 0$), com um ($\mathbf{a}(1) = 1$), com uma variável aleatória de distribuição gaussiana e desvio padrão 0,15, e com $\mu e(1)$, que é aquela mostrada originalmente na Tabela 2. A Figura 6 mostra as curvas de MSE resultantes.

A influência da inicialização sobre o algoritmo KLMS foi visivelmente menor do que sobre o KLMS2, cuja velocidade de convergência foi mais fortemente impactada por algumas das inicializações testadas. Assim como no caso anterior, não se visualizam diferenças claras no comportamento em regime. Isto pode ser atestado pelos resultados mostrados na Tabela 5. Nela são exibidos os valores médios de MSE em dB calculados usando as 500 últimas iterações de cada experiência. Como se pode ver, os resultados obtidos foram muito semelhantes para

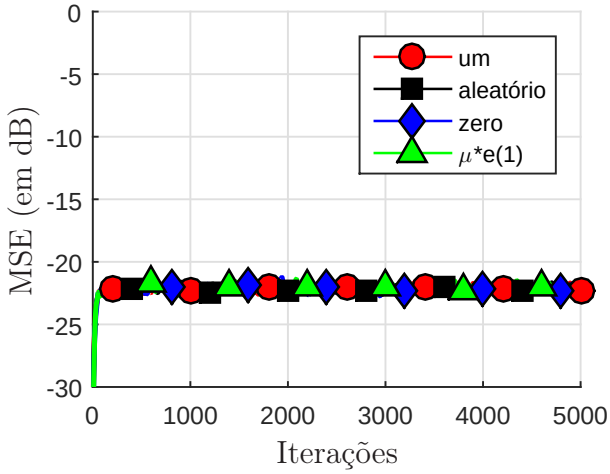


Figura 6: Curvas de erro quadrático médio do algoritmo KLMS com diferentes inicializações aplicado à identificação de um sistema regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

todas as inicializações (a diferença entre o maior e o menor valor de MSE em regime foi de menos de um décimo de decibel), o que indica que de fato o algoritmo KLMS não tem o seu desempenho em regime influenciado pela inicialização.

Tabela 5: MSE estacionário (em dB, calculado usando as 500 últimas iterações de 500 realizações) do algoritmo KLMS aplicado na identificação de um sistema não linear dado pela Equação (39), sendo a entrada $u(n)$ um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

Inicialização	MSE (em dB)
Com zero	-22,0045
Com um	-21,9820
Aleatória	-22,0581
$\mu e(1)$	-21,9801

4.2.2 Passo de adaptação

Nesta subseção, compara-se o algoritmo KLMS com o KLMS2 e com a sua versão normalizada, o KNLMS2.Também se estuda a influência do passo de adaptação no desempenho desses algoritmos. Como foi mostrado na Seção 1.1 comparando o algoritmo LMS com o NLMS, a utilização de um passo de adaptação normalizado é vantajosa porque a faixa de

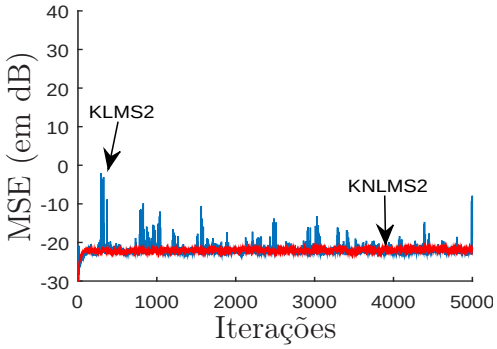
valores de passos $\tilde{\mu}$ que evitam a divergência do algoritmo não depende da potência do sinal de entrada, o que facilita a escolha desse parâmetro.

Emprega-se ainda nesta subseção o sistema da Equação (39). Optou-se por inicializar o vetor $\boldsymbol{\alpha}$ dos algoritmos KLMS2 e KNLMS2 com zeros, pois esta foi a inicialização que fez com que o algoritmo apresentasse melhor desempenho na Subseção 4.2.1. Como todas as inicializações testadas para o KLMS produziram resultados muito semelhantes, optou-se por inicializar o coeficiente $\boldsymbol{a}(1)$ por meio de $\boldsymbol{a}(1) = \mu e(1)$, uma vez que esta foi a inicialização originalmente proposta para o algoritmo [4].

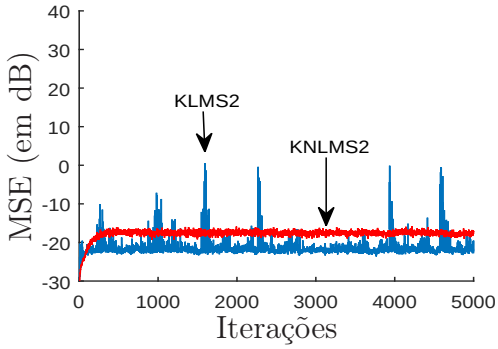
Na Figura 7 é mostrado o que acontece quando se aumenta o passo de adaptação dos algoritmos KLMS2 e KNLMS2. Em cada um dos gráficos, os valores de μ e $\tilde{\mu}$ foram ajustados de modo que o MSE calculado nas 500 últimas iterações fosse numericamente igual para os dois algoritmos. Pode-se notar a convergência praticamente instantânea observada na Subseção 4.2.1, devida à inicialização do vetor de coeficientes com zeros. Por esta razão, o número de iterações necessárias até a convergência praticamente não se alterou com a modificação do passo de adaptação. Pode-se ver que execuções dos algoritmos com passos de adaptação menores levaram o KNLMS2 a atingir patamares de erro quadrático médio mais baixos, assim como ocorre com o LMS e o NLMS.

Comparando o desempenho dos dois algoritmos em cada caso, é possível ver que as curvas de MSE do KNLMS2 não contêm os picos pronunciados das curvas referentes ao KLMS2. Como mencionado na Seção 1.1, esses picos abruptos são indícios de que, em algumas realizações, o algoritmo esteve perto de divergir. Pode-se ver que mesmo com o aumento de $\tilde{\mu}$, o KNLMS2 continuou a convergir, embora o seu desempenho tenha piorado. Em contrapartida, conforme se aumenta o passo não normalizado μ do KLMS2, o algoritmo passa a mostrar um comportamento cada vez mais instável.

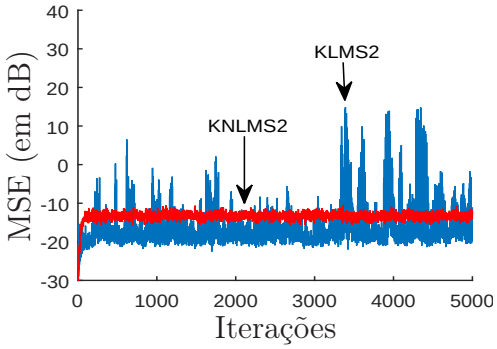
Os resultados mostram que, embora o KLMS2 não normalizado possa atingir patamares de MSE mais baixos em certas condições, o algoritmo KNLMS2 é mais estável, sendo portanto uma escolha mais segura. Por esse motivo, no restante deste relatório não será mais utilizado o algoritmo KLMS2, e sim a sua versão normalizada. Na Subseção 4.2.3 a seguir será apresentado mais um argumento a favor da sua utilização.



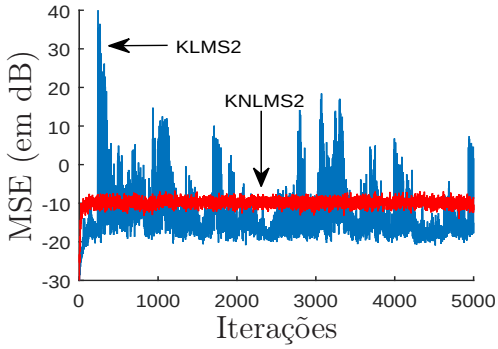
(a) MSE de -21 dB nas últimas 500 iterações. Utilizou-se $\mu = 1{,}35$ para o KLMS2 e $\tilde{\mu} = 0{,}004$ para o KNLMS2.



(b) MSE de -17 dB nas últimas 500 iterações. Utilizou-se $\mu = 1{,}4$ para o KLMS2 e $\tilde{\mu} = 0{,}0393$ para o KNLMS2.



(c) MSE de -13 dB nas últimas 500 iterações. Utilizou-se $\mu = 1{,}6$ para o KLMS2 e $\tilde{\mu} = 0{,}393$ para o KNLMS2.

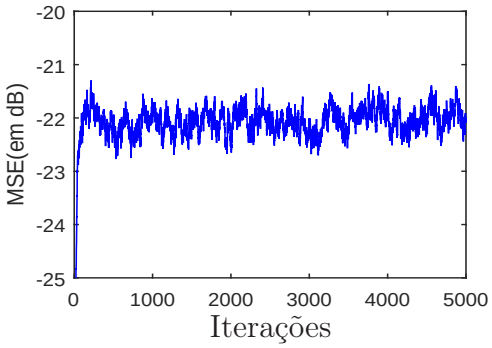


(d) MSE de -9 dB nas últimas 500 iterações. Utilizou-se $\mu = 1{,}672$ para o KLMS2 e $\tilde{\mu} = 1{,}0$ para o KNLMS2.

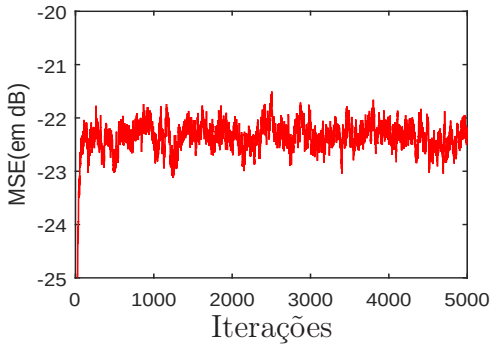
Figura 7: Curvas de erro quadrático médio dos algoritmos KLMS2 e KNLMS2 com $h = 0{,}0225$ e diferentes passos de adaptação (indicados nas legendas). Ambos os algoritmos foram aplicados na identificação do Sistema (39). A entrada é dada por um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Para o algoritmo KNLMS2, utilizou-se $\delta = 10^{-5}$.

A fim de investigar a influência do passo de adaptação no algoritmo KLMS, este foi aplicado também ao sistema da Equação (39) com largura de *kernel* $h = 0{,}0225$ e diferentes valores de μ. Foram realizadas 500 experiências com 5000 iterações cada. Na Figura 8 são mostrados os resultados obtidos para $\mu = 0{,}00393$, $\mu = 0{,}393$, $\mu = 1{,}0$ e $\mu = 1{,}9$.

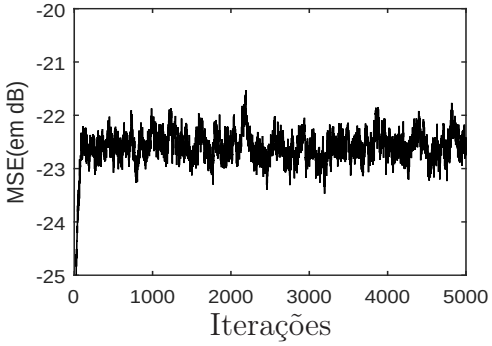
Cabe lembrar que, como mencionado na Subseção 1.4, o algoritmo KLMS é automaticamente normalizado quando se utiliza o *kernel* gaussiano. De fato, pode-se ver que o desempenho do algoritmo variou muito pouco com o passo, mesmo com este assumindo va-



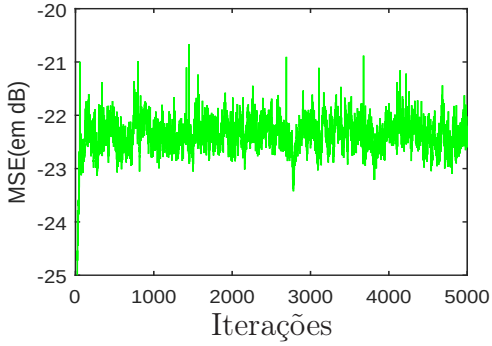
(a) Algoritmo KLMS com $\mu = 0{,}004$. O valor médio do MSE calculado com base nas 500 últimas interações é -21,9432.



(b) Algoritmo KLMS com $\mu = 0{,}393$. O valor médio do MSE calculado com base nas 500 últimas interações é -22,4603.



(c) Algoritmo KLMS com $\mu = 1{,}0$. O valor médio do MSE calculado com base nas 500 últimas interações é -22,5701.



(d) Algoritmo KLMS com $\mu = 1{,}9$. O valor médio do MSE calculado com base nas 500 últimas interações é -22,4174.

Figura 8: Curvas de erro quadrático médio do KLMS com $h = 0{,}0225$ e diferentes passos de adaptação (indicados nas legendas). O algoritmo foi aplicado na identificação do Sistema (39). A entrada consiste em ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

lores de diferentes ordens de grandezas. Não foram observados picos nas curvas de MSE, ao contrário do que ocorreu com o KLMS2 na Figura 7. Isso indica que o KLMS, assim como o KNLMS2, apresenta uma maior estabilidade em comparação com esse algoritmo. Também é possível ver que, apesar das curvas adquirirem um aspecto mais ruidoso com passos maiores, o valor médio de MSE nas últimas iterações praticamente não se alterou. Esse comportamento é diferente daquele exibido pelos algoritmos LMS e NLMS [1–3], conforme mencionado na Subseção 1.1, e também daquele exibido pelos algoritmos KLMS2 e KNLMS2, como se constatar comparando as Figuras 7 e 8.

Apesar da diferença entre os valores de MSE ter sido muito pequena, como o valor de passo que levou ao menor erro em regime foi $\mu = 1{,}0$, esse foi o valor adotado na Seção 4.2.3.

4.2.3 Largura do *kernel* gaussiano

Agora será investigada a influência da largura do *kernel* gaussiano h no desempenho dos algoritmos KLMS e KNLMS2. Como já mencionado, o *kernel* gaussiano é dado pela Equação (16), aqui reproduzida:

$$k(\mathbf{x},\mathbf{x}') = e^{\frac{(-||\mathbf{x}-\mathbf{x}'||^2)}{2h^2}}.$$

Analisando essa equação, é possível verificar que

$$\lim_{h\to 0} k(\mathbf{x},\mathbf{x}') = 0. \tag{40}$$

Isso sugere que, quanto menor h, mais ortogonais $\mathbf{\Phi}(\mathbf{u})$ e $\mathbf{\Phi}(\mathbf{u}')$ serão entre si para $\mathbf{u} \neq \mathbf{u}'$ em $\mathbb{F}$, pois pela Equação (13) tem-se $\mathbf{\Phi}(\mathbf{u})^{\mathrm{T}}\mathbf{\Phi}(\mathbf{u}') \to 0$. Assim, deve ser mais difícil escrever a transformada da entrada atual $\mathbf{\Phi}(\mathbf{u}(n))$ como combinação linear dos elementos do dicionário. Portanto, se h for muito pequeno, isso deve prejudicar o desempenho do algoritmo. Em contrapartida,

$$\lim_{h\to \infty} k(\mathbf{x},\mathbf{x}') = 1. \tag{41}$$

Ou seja, se h for muito elevado, $\mathbf{\Phi}(\mathbf{u})^{\mathrm{T}}\mathbf{\Phi}(\mathbf{u}') \to 1$, e, consequentemente, $\mathbf{k}^{\mathrm{T}}(n)\mathbf{k}(n) \to L$, sendo que L denota a dimensão do vetor $\mathbf{k}(n)$. Isso quer dizer que ao se trabalhar com a versão não normalizada do algoritmo KLMS2, para um valor fixo de μ e valores crescentes de h, o algoritmo tende cada vez mais a divergir. Para evitar isso, deve-se trabalhar com o KLMS ou o KNLMS2, ou então reduzir o passo μ toda vez que se aumentar substancialmente h. Como discutido na Seção 4.2.2, a normalização do passo de adaptação traz vantagens em termos de estabilidade do algoritmo de maneira geral, de modo que a primeira estratégia foi a adotada nesta subseção e no restante do relatório.

Para ilustrar o que foi dito no parágrafo anterior, o histograma da Figura 9 mostra a porcentagem de realizações em que o algoritmo KLMS2 com $\mu = 0{,}393$ divergiu em função de h. Foram realizadas 500 realizações com 5000 iterações cada, ainda utilizando o mesmo

sistema das subseções anteriores, com o sinal $u(n)$ com média nula e $\sigma_u = 0{,}15$. Pode-se ver que, quanto maior esse parâmetro, maior a taxa de divergência.

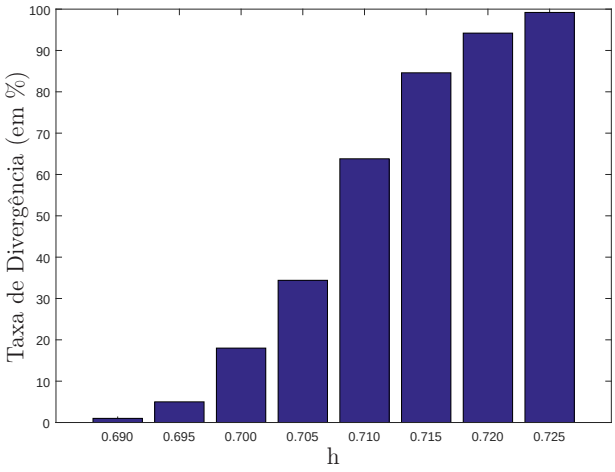
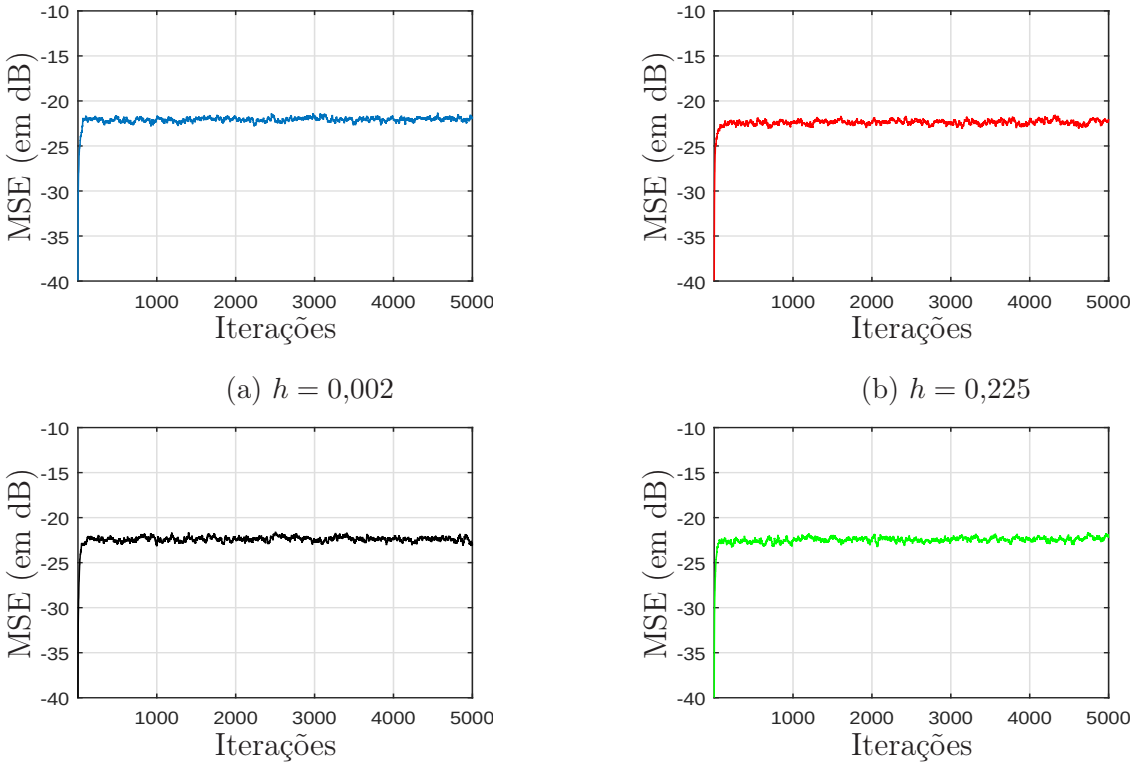


Figura 9: Taxa de divergência do algoritmo KLMS2 em função de h, com μ e L fixados em 0,393 e 6, respectivamente. O algoritmo foi aplicado à identificação do Sistema (39), e o sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

Enquanto se estudava a influência da largura do *kernel* gaussiano no desempenho do algoritmo, percebeu-se que, mesmo utilizando o KNLMS2, existe uma relação entre $\tilde{\mu}$ e h. Mesmo após a realização de várias simulações diferentes, esta relação ainda não está clara e precisa ser melhor investigada, ficando como sugestão de tema para estudos posteriores. Os gráficos das Figuras 10 e 11 mostram as curvas de MSE do KNLMS2 aplicado ao Sistema (39) com dois passos de adaptação diferentes.

Observando a Figura 10, vê-se que, para $\tilde{\mu} = 0{,}001$, o MSE praticamente não variou com as mudanças na largura do *kernel*. Mesmo com o valor de h variando entre 0,002 e 100, o erro quadrático médio em regime praticamente não se alterou. Também não é possível identificar grandes diferenças em termos de velocidade de convergência.

Em contrapartida, na Figura 11, nota-se algo diferente. Pode-se ver que o valor da largura do *kernel* influenciou fortemente o valor do erro quadrático médio atingido em regime. Para $h = 1{,}0$ e $h = 100{,}0$, o algoritmo mostrou um desempenho consideravelmente melhor do que para os outros valores de h. É interessante notar que os dois valores mencionados ($h = 1{,}0$ e $h = 100{,}0$) levaram a desempenhos muito semelhantes, embora um dos valores seja cem vezes maior do que o outro.



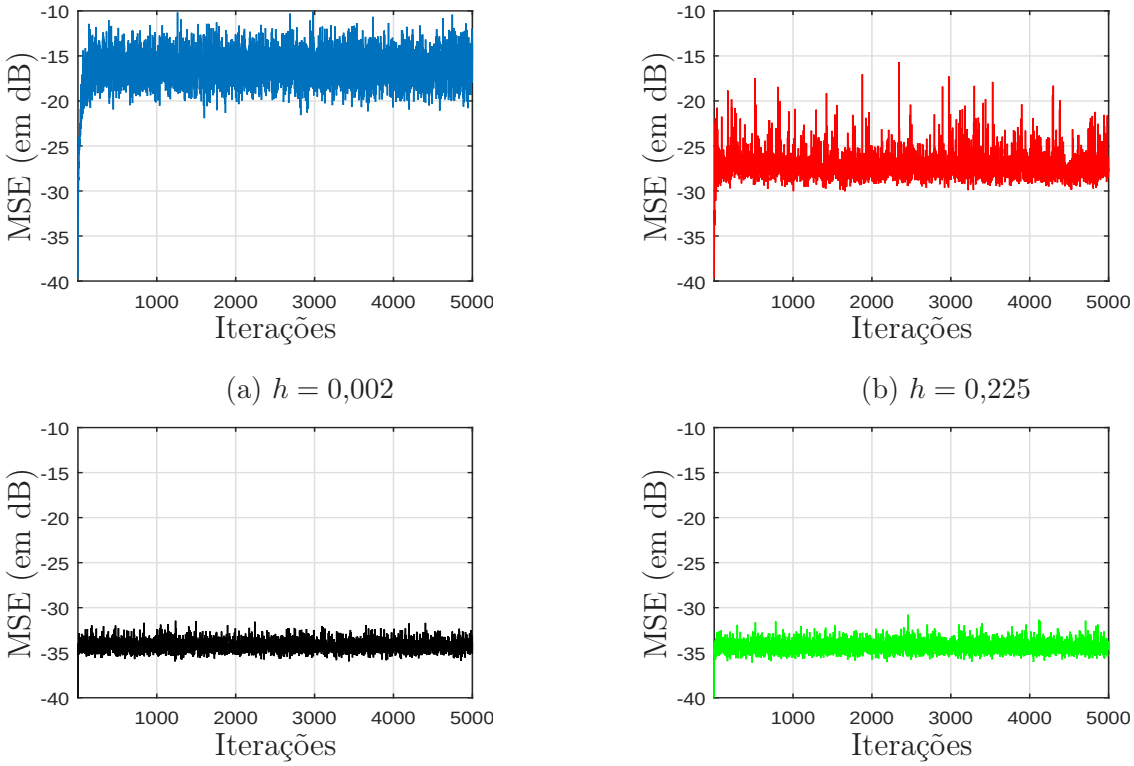
(a) $h = 0{,}002$ (b) $h = 0{,}225$

(c) $h = 1{,}0$ (d) $h = 100{,}0$

Figura 10: Curvas de erro quadrático médio apresentadas pelo algoritmo KNLMS2 com diferentes valores de h, $\tilde{\mu} = 0{,}001$ e $\delta = 10^{-5}$. O algoritmo foi empregado na identificação do sistema regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

Surpreendentemente, pode-se ver que, para um mesmo valor de h, a diminuição do passo de adaptação não necessariamente leva a um MSE mais baixo em regime. Comparando-se os gráficos da Figura 10 com os da Figura 11, vê-se que o desempenho do KNLMS2 com $h = 1{,}0$ e $\tilde{\mu} = 1{,}0$ foi superior ao do mesmo algoritmo com o mesmo valor de h, mas com $\tilde{\mu} = 0{,}001$. Consequentemente, a afirmação feita na Subseção 4.2.2 de que o aumento de $\tilde{\mu}$ leva a uma piora no desempenho do algoritmo deve ser vista com cautela, pois embora tenha se mostrado válida naquele exemplo, não é necessariamente verdadeira em todos os casos.

Como mencionado, essa relação entre a largura do *kernel* e o passo de adaptação ainda precisa ser investigada mais a fundo, a fim de que se descubra a sua causa e o tipo de relação existente. Por ora, os resultados sugerem que as escolhas de $\tilde{\mu}$ (ou μ) e h para os algoritmos KLMS2 e KNLMS2 devem ser feitas testando-se diferentes combinações de valores para ambos os parâmetros.



(a) $h = 0{,}002$ (b) $h = 0{,}225$

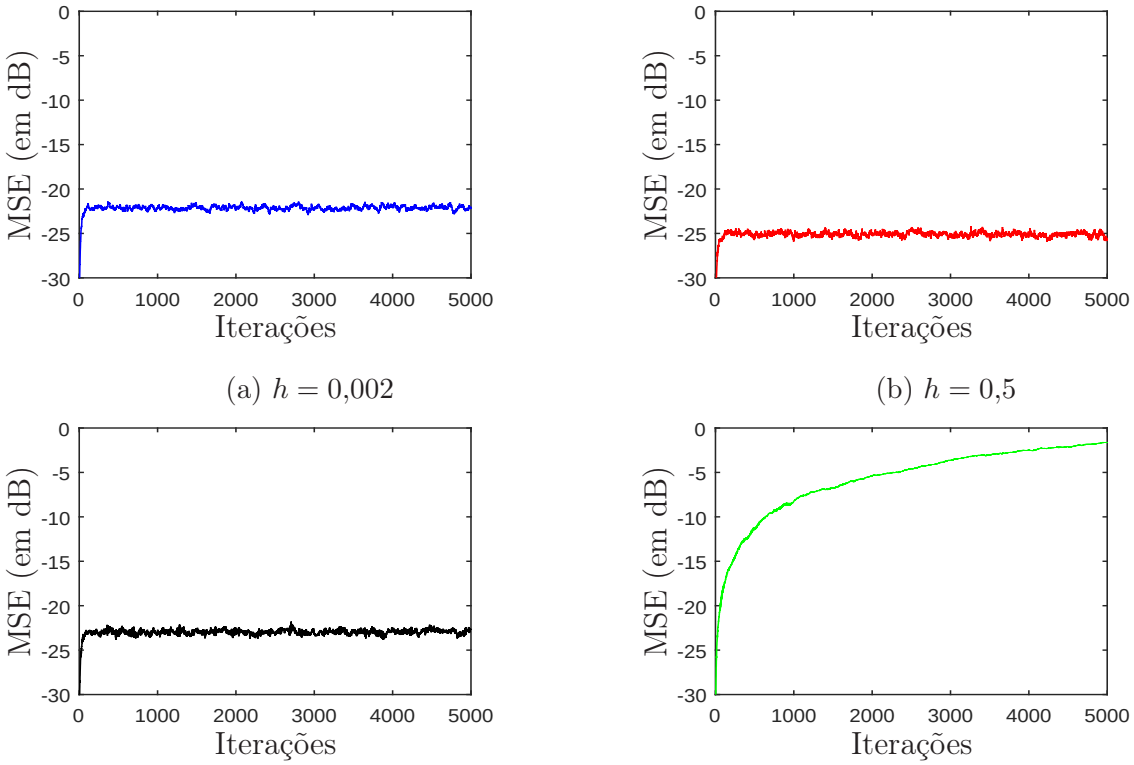
(c) $h = 1{,}0$ (d) $h = 100{,}0$

Figura 11: Curvas de erro quadrático médio apresentada pelo algoritmo KNLMS2 com diferentes valores de h, $\tilde{\mu} = 1{,}0$ e $\delta = 10^{-5}$. O algoritmo foi empregado na identificação do sistema regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

Com o passo de adaptação fixo em $\mu = 1{,}0$, também foram executadas simulações aplicando o algoritmo KLMS à identificação do Sistema (39) e diferentes valores de largura do *kernel* visando a estudar a influência desse parâmetro no desempenho do algoritmo. Na Figura 12 são mostrados os resultados obtidos. Mais uma vez, foram realizadas 500 experiências com 5000 iterações cada.

Pode-se ver que, ao contrário do que foi observado com o algoritmo KNLMS2, a adoção de uma largura de *kernel* muito elevada comprometeu seriamente o desempenho do algoritmo, levando a uma deterioramento ao longo do tempo no seu comportamento. Os outros valores de h produziram resultados semelhantes entre si, com o valor de $h = 0{,}5$ gerando um patamar de MSE ligeiramente menor em regime.

É importante ressaltar que não se observou uma relação clara entre a largura do *kernel* e o passo de adaptação do KLMS com a restrição do dicionário aplicada a ele, ao contrário



(a) $h = 0{,}002$ (b) $h = 0{,}5$

(c) $h = 1{,}0$ (d) $h = 100{,}0$

Figura 12: Curvas de erro quadrático médio apresentadas pelo algoritmo KLMS com diferentes valores de h e $\mu = 1{,}0$. O algoritmo foi empregado na identificação do sistema regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$.

do que ocorreu com o KNLMS2. Para todos os valores de h testados, o algoritmo continuou a mostrar a insensibilidade ao passo vista na Subseção 4.2.2, com diferenças muito pequenas entre os resultados. Contudo, na Subseção 4.2.4 a seguir será mostrado que o algoritmo KLMS sem restrições impostas ao seu dicionário também apresenta uma certa relação entre h e μ.

4.2.4 Métodos de restrição do dicionário

O KLMS em sua forma original apresenta um dicionário que cresce a cada iteração, o que acarreta um aumento no custo computacional e inviabiliza o seu uso em aplicações em tempo real. Para torná-lo viável, diversas técnicas já foram propostas. A mais simples é manter a quantidade de elementos do dicionário fixa em um número pré-determinado, utilizada por exemplo em [10]. Outros métodos incluem o *critério da novidade* [4], o *critério*

da coerência [10, 11], e o *critério da surpresa* [13].

Nas Subseções 4.2.1, 4.2.2 e 4.2.3, fixou-se o número de elementos do dicionário em seis. Esse número foi extraído de [10], mas é importante ter em mente que muitas vezes pode ser difícil saber *a priori* a quantidade de elementos que levará, estatisticamente falando, a um desempenho bom do algoritmo. Se o tamanho do dicionário for fixado em um número elevado, pode ser que o algoritmo mostre um bom desempenho, mas que suas execuções sejam lentas. Em contrapartida, caso se fixe o número de elementos em um número baixo, ganha-se em velocidade, mas não se pode ter certeza de que o desempenho será satisfatório. Embora seja um método simples de restringir o crescimento do dicionário, ele exige a escolha de um parâmetro que pode se mostrar difícil dependendo da aplicação. Daí a importância dos critérios citados no parágrafo anterior, em que o dicionário é inicializado com um único elemento, e os vetores de entrada são incluídos se satisfizerem ao critério usado.

Neste trabalho, optou-se por focar no critério da coerência. De acordo com esse critério, o *kernel* da entrada atual $k(\cdot,\mathbf{u}(n))$ é inserido no dicionário se [10, 11]

$$max_j|k(\mathbf{u}(n),\mathbf{u}(c_j))| \leq \epsilon, \tag{42}$$

em que $\mathbf{u}(c_j)$ é o j-ésimo elemento do dicionário e ϵ é um parâmetro que o usuário deve escolher, denominado *nível de coerência*. A ideia desse critério é que se $max_j|k(\mathbf{u}(n),\mathbf{u}(c_j))|$ for muito pequeno, isso quer dizer que $\Phi(\mathbf{u}(n))$ é aproximadamente ortogonal a $\Phi(\mathbf{u}(c_1)$, $\Phi(\mathbf{u}(c_2)$, $\cdots$, $\Phi(\mathbf{u}(c_L)$, não podendo ser bem escrita como combinação linear destas. Assim, a nova entrada deve ser incluída para que o dicionário possa conter uma base de vetores em $\mathbb{F}$.

As Expressões (16) e (42) indicam que o número de elementos a serem inseridos no dicionário de acordo com esse critério depende dos valores de ϵ e de h e do sinal $\mathbf{u}(n)$. Isso que implica que os algoritmos KLMS, KLMS2 e KNLMS2 devem incluir o mesmo número de elementos aos seus respectivos dicionários se aplicados a situações idênticas com os mesmos valores de largura de *kernel* e de nível de coerência. Ademais, os valores dos passos de adaptação μ e $\tilde{\mu}$ não devem influenciar na quantidade de elementos no dicionário e, consequentemente, no tempo de execução dos algoritmos.

Quanto maior ϵ, mais elementos serão introduzidos no dicionário. Como

$$k(\mathbf{x},\mathbf{x}') = e^{\frac{(-||\mathbf{x}-\mathbf{x}'||^2)}{2h^2}} \leq 1 \ \forall \ \mathbf{x}, \ \mathbf{x}', \tag{43}$$

se usarmos o *kernel* gaussiano e escolhermos $\epsilon \geq 1$, a cada iteração todos os novos elementos serão incorporados ao dicionário, assim como ocorre no algoritmo KLMS originalmente proposto (ver Seção 1.3). Além disso, em decorrência dos Limites (40) e (41), nota-se que a tendência é que valores baixos de h levem a dicionários maiores, enquanto que valores elevados devem levar a dicionários mais enxutos.

As Tabelas 6 e 7 a seguir comparam o desempenho do algoritmo KNLMS2 com um número variável de elementos no dicionário. Utilizou-se o critério da coerência com $\epsilon = 0{,}001$ e com diferentes valores de h e de $\tilde{\mu}$ (0,001 no caso da primeira e 1,0 no caso da segunda). A comparação é feita em termos do número médio de elementos contidos no dicionário ao fim das realizações, do tempo médio de execução de cada realização e do MSE em regime. Foram executadas 100 realizações com 4000 iterações cada. Cabe notar que o tempo de execução do algoritmo pode variar dependendo da máquina em que as simulações são feitas. Os dados apresentados foram coletados em um microcomputador com um processador "Intel Core i5" e 6 Giga Bytes de memória RAM instalada.

Na Tabela 6 é possível observar que, como era esperado, valores menores de h levaram a dicionários maiores. Com $h = 1{,}0$ e $h = 0{,}5$, apenas um elemento foi adicionado ao dicionário em todas as iterações. Além disso, pode-se ver claramente que quanto mais elementos incorporados ao dicionário na média, maior o tempo médio de execução do algoritmo. Contudo, observa-se que, para esse valor de $\tilde{\mu}$, o MSE em regime praticamente não se alterou com as mudanças no valor de h, assim como na Subseção 4.2.3.

Na Tabela 7, é mostrada uma situação semelhante à anterior. Novamente é possível ver que quanto menor h, mais elementos são incorporados ao dicionário e que isso, por sua vez, leva a um tempo maior de execução do algoritmo. Vê-se também que, assim como na Subseção 4.2.3, para $\tilde{\mu} = 1{,}0$, o MSE em regime variou mais fortemente com h. A última linha da tabela mostra um resultado surpreendente: mesmo com um único elemento no dicionário, o algoritmo KNLMS2 mostra um desempenho muito satisfatório com essa combinação de

Tabela 6: KNLMS2 com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). Utilizou-se $\tilde{\mu} = 0{,}001$, $\epsilon = 0{,}001$ e $\delta = 10^{-5}$. O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Os dados são apresentados na forma média ± desvio padrão.

Algoritmo KNLMS2 com $\tilde{\mu} = 0{,}001$ e $\epsilon = 0{,}001$			
h	Número médio de elementos	Tempo médio de execução (em segundos)	MSE (em dB)
0,001	$169{,}9000 \pm 4{,}1157$	$3{,}3986 \pm 0{,}4532$	-21,9942
0,01	$21{,}7100 \pm 1{,}4723$	$0{,}9009 \pm 0{,}1265$	-22,2681
0,1	$3{,}0000 \pm 0{,}0000$	$0{,}5051 \pm 0{,}1025$	-22,0671
0,5	$1{,}0000 \pm 0{,}0000$	$0{,}4668 \pm 0{,}0325$	-22,4277
1,0	$1{,}0000 \pm 0{,}0000$	$0{,}4969 \pm 0{,}06179$	-22,1760

h e $\tilde{\mu}$. Essa foi a combinação que levou ao menor tempo médio de execução e, ao mesmo tempo, ao erro quadrático médio mais baixo em regime.

De modo geral, os resultados experimentais corroboram a afirmação de que o passo de adaptação não influencia no número de elementos adicionados ao dicionário. Comparando as Tabelas 6 e 7, vê-se que as diferenças, para os mesmos valores de largura do *kernel*, são pequenas e provavelmente se devem às aleatoriedades dos processos envolvidos. Ainda comparando essas duas tabelas pode-se ver que, assim como na Subseção 4.2.3, os resultados obtidos com $\tilde{\mu} = 1{,}0$ foram melhores do que aqueles obtidos com $\tilde{\mu} = 0{,}001$ para $h = 0{,}5$ e $h = 1{,}0$, mas não para valores menores de largura do *kernel*.

Para ilustrar a afirmação de que, escolhendo-se $\epsilon = 1{,}0$, a cada iteração o novo vetor regressor de entrada será incluído no dicionário, a Tabela 8 compara execuções do algoritmo KNLMS2 com $h = 1{,}0$, $\tilde{\mu} = 1{,}0$ e com dois valores de ϵ: 0,001 (o mesmo das simulações anteriores) e 1,0. Como se pode ver, com $\epsilon = 1{,}0$, ao final de todas as realizações o algoritmo havia incorporado 4000 elementos ao dicionário, que é igual ao número de iterações. Isso significa que, de fato, a cada iteração um novo elemento foi adicionado. Comparando-se o tempo médio de execução, fica clara a necessidade de se restringir o crescimento do dicionário

Tabela 7: KNLMS2 com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Utilizou-se $\tilde{\mu} = 1{,}0$, $\epsilon = 0{,}001$ e $\delta = 10^{-5}$. Os dados são apresentados na forma média ± desvio padrão.

Algoritmo KNLMS2 com $\tilde{\mu} = 1{,}0$ e $\epsilon = 0{,}001$			
h	Número médio de elementos	Tempo médio de execução (em segundos)	MSE (em dB)
0,001	$170{,}0400 \pm 4{,}4720$	$3{,}2293 \pm 0{,}3055$	-10,8655
0,01	$21{,}6900 \pm 1{,}3758$	$0{,}8530 \pm 0{,}1107$	-11,6920
0,1	$3{,}0 \pm 0{,}0$	$0{,}4695 \pm 0{,}04360$	-19,4326
0,5	$1{,}0 \pm 0{,}0$	$0{,}4985 \pm 0{,}0668$	-33,7085
1,0	$1{,}0 \pm 0{,}0$	$0{,}5277 \pm 0{,}0511$	-34,3119

caso se queira empregar os algoritmos KLMS2 ou KNLMS2 em aplicações em tempo real. Incorporando todos os novos vetores de entrada, as execuções do KNLMS2 demoraram muito mais do que as mostradas nas tabelas anteriores. É interessante notar que o MSE em regime praticamente não se alterou com as mudanças em ϵ.

Tabela 8: KNLMS2 com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Utilizou-se $\tilde{\mu} = 1{,}0$, $h = 1{,}0$ e $\delta = 10^{-5}$. Os dados são apresentados na forma média ± desvio padrão.

Algoritmo KNLMS2 com $\tilde{\mu} = 1{,}0$ e $h = 1{,}0$			
ϵ	Número médio de elementos	Tempo médio de execução (em segundos)	MSE (em dB)
0,001	$1{,}0 \pm 0{,}0$	$0{,}5277 \pm 0{,}0511$	-34,3119
1,0	$4000{,}0 \pm 0{,}0$	$36{,}1734 \pm 4{,}1464$	-34,1735

Também foi realizada uma simulação com $\epsilon = 1{,}0$, $h = 1{,}0$ e $\tilde{\mu} = 0{,}001$, a fim de investigar se a relação entre o passo de adaptação e a largura do *kernel* constatada na Subseção 4.2.3

para o algoritmo KNLMS2 havia sido causada de alguma forma pela restrição aplicada ao dicionário. O resultado dessa simulação pode ser visto na Tabela 9, em que se reproduziram os resultados obtidos com $\epsilon = 1{,}0$, $h = 1{,}0$ e $\tilde{\mu} = 1{,}0$ para efeito de comparação. Pode-se constatar que, assim como na Subseção 4.2.3, o desempenho obtido com $\tilde{\mu} = 1{,}0$ foi bastante superior àquele obtido com um valor menor de passo de adaptação, o que indica que a restrição do dicionário não influenciou o resultado observado anteriormente.

Tabela 9: KNLMS2 com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Utilizou-se $\epsilon = 1{,}0$, $h = 1{,}0$ e $\delta = 10^{-5}$. Os dados são apresentados na forma média $\pm$ desvio padrão.

Algoritmo KNLMS2 com $\epsilon = 1{,}0$ e $h = 1{,}0$	
$\tilde{\mu}$	MSE (em dB)
0,001	-22,4789
1,0	-34,1735

O critério da coerência também foi aplicado ao algoritmo KLMS. Os resultados para $\mu = 0{,}001$, $\epsilon = 0{,}001$ e diferentes valores de h podem ser vistos na Tabela 10. Os resultados obtidos com $\mu = 1{,}0$ e os mesmos valores de ϵ e h podem ser vistos na Tabela 11. Comparando-se essas duas tabelas, pode-se ver que, assim como ocorre com o KNLMS2, a largura do *kernel* influencia fortemente no número médio de elementos no dicionário, mas o passo de adaptação não. Além disso, comparando-se a Tabela 6 com a Tabela 10 e a Tabela 7 com a Tabela 11, nota-se que não há diferença significativa entre o comportamento do KNLMS2 e do KLMS no que se refere ao número médio de elementos incorporados ao dicionário, o que também era esperado. As pequenas discrepâncias entre os números médios de elementos se devem novamente às aleatoriedades envolvidas nas simulações.

É importante ressaltar que as diferenças entre os tempos de execução desses dois algoritmos são devidas ao fato de a saída do KLMS tal como originalmente proposto é calculada em cada instante de tempo por meio de um *loop* do tipo *for*, enquanto que a saída do KNLMS2 é dada por um produto interno entre dois vetores. Essa última operação é implementada de forma otimizada no *software* Matlab em relação ao *for*, o que faz com que a execução do

KNLMS2 demande menos tempo nesse programa. Cabe notar que pode ser possível implementar o algoritmo KLMS de forma mais eficiente em um *script* para o Matlab, mas isso não foi investigado neste relatório.

Devido a essa diferença entre as implementações, não se pretende aqui comparar os algoritmos em termos do tempo de execução, uma vez que essa comparação pode ser considerada injusta. Os tempos de execução foram incluídos nas tabelas 6 a 8 e 10 a 12 a fim de tecer comparações não entre o KLMS e o KNLMS2, mas sim entre implementações dos mesmos algoritmos com diferentes valores de μ, $\tilde{\mu}$, h e ϵ. No entanto, é possível compará-los em termos de número médio de elementos no dicionário, uma vez que esse parâmetro independe de eventuais otimizações do *software*.

Tabela 10: KLMS com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). Utilizou-se $\mu = 0{,}001$ e $\epsilon = 0{,}001$. O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Os dados são apresentados na forma média $\pm$ desvio padrão.

Algoritmo KLMS com $\mu = 0{,}001$ e $\epsilon = 0{,}001$			
h	Número médio de elementos	Tempo médio de execução (em segundos)	MSE (em dB)
0,001	$171{,}1100 \pm 4{,}1607$	$9{,}7139 \pm 0{,}7900$	-21,8711
0,01	$21{,}500 \pm 1{,}2978$	$1{,}5533 \pm 0{,}1698$	-21,9126
0,1	$2{,}7700 \pm 0{,}4894$	$0{,}2690 \pm 0{,}0453$	-21,9863
0,5	$1{,}0000 \pm 0{,}0000$	$0{,}1432 \pm 0{,}0219$	-22,0213
1,0	$1{,}0000 \pm 0{,}0000$	$0{,}1530 \pm 0{,}0237$	-22,0253

Em seguida se empregou o algoritmo KLMS com $\epsilon = 1{,}0$, o que já foi mostrado que equivale a não restringir o tamanho do dicionário. Utilizou-se $\mu = 1{,}0$ e $h = 0{,}5$. Na Tabela 12 é mostrada uma comparação entre os resultados obtidos com essa simulação e aqueles obtidos com os mesmos valores de passo de adaptação e de largura de *kernel*, mas com nível de coerência de 0,001. Como se pode ver, a diferença entre os desempenhos é muito grande, com a versão sem restrição de dicionário alcançando um patamar muito mais baixo de erro

Tabela 11: KLMS com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). Utilizou-se $\mu = 1{,}0$ e $\epsilon = 0{,}001$. O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Os dados são apresentados na forma média $\pm$ desvio padrão.

Algoritmo KLMS com $\mu = 1{,}0$ e $\epsilon = 0{,}001$			
h	Número médio de elementos	Tempo médio de execução (em segundos)	MSE (em dB)
0,001	$170{,}9900 \pm 4{,}4665$	$8{,}7214 \pm 0{,}3644$	-20,7591
0,01	$21{,}7600 \pm 1{,}3190$	$1{,}2820 \pm 0{,}0668$	-21,5735
0,1	$2{,}8000 \pm 0{,}4264$	$0{,}2464 \pm 0{,}0373$	-21,7610
0,5	$1{,}0000 \pm 0{,}0000$	$0{,}1241 \pm 0{,}0109$	-21,8321
1,0	$1{,}0000 \pm 0{,}0000$	$0{,}1185 \pm 0{,}0078$	-22,1790

quadrático médio em regime. Comparando-se a Tabela 8 com a Tabela 12, pode-se ver que apenas quando nenhuma restrição foi aplicada ao seu dicionário o algoritmo KLMS conseguiu mostrar desempenho comparável ao melhor comportamento exibido pelo KNLMS2, o qual, em contrapartida, se mostrou capaz de atingir patamares consideravelmente baixos de MSE mesmo com restrições impostas ao seu dicionário. Além disso, comparando-se os resultados das Tabelas 10, 11 e 12 com aqueles apresentados pelo algoritmo KLMS nas subseções anteriores, pode-se constatar que o critério da coerência não apresentou vantagem significativa sobre a manutenção de um número fixo de elementos no dicionário.

Por fim, decidiu-se realizar uma simulação empregando o algoritmo KLMS com $\epsilon = 1{,}0$, $h = 0{,}5$ e $\mu = 0{,}001$. Na Tabela 13 é mostrado o resultado obtido com esses valores de nível de coerência, largura do *kernel* e passo de adaptação. Além disso, para facilitar a comparação, reproduziu-se o resultado mostrado na Tabela 12 com $\epsilon = 1{,}0$, $h = 0{,}5$ e $\mu = 1{,}0$.

Embora não se tenha constatado a existência clara de uma relação entre μ e h para o algoritmo KLMS com o crescimento do seu dicionário restringido (ver último parágrafo da Subseção 4.2.3, assim como as Tabelas 10 e 11 desta mesma subseção), os resultados indicam que sem essa restrição o KLMS apresenta, sim, uma relação entre esses parâmetros. Pode-se

Tabela 12: KLMS com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Utilizou-se $\mu = 1{,}0$ e $h = 1{,}0$. Os dados são apresentados na forma média ± desvio padrão.

Algoritmo KLMS com $\mu = 1{,}0$ e $h = 0{,}5$			
ϵ	Número médio de elementos	Tempo médio de execução (em segundos)	MSE (em dB)
0,001	$1{,}0 \pm 0{,}0$	$0{,}1241 \pm 0{,}0109$	-21,8321
1,0	$4000{,}0 \pm 0{,}0$	$127{,}9742 \pm 16{,}1826$	-33,8035

ver que o desempenho exibido pelo algoritmo com $\mu = 1{,}0$ foi significativamente superior ao resultado obtido com o mesmo valor de h, mas com passo de adaptação menor. Isso mostra que a relação que se constatou entre a largura do *kernel* e o passo de adaptação do KNLMS2 também se manifesta no caso do algoritmo KLMS, porém apenas quando nenhuma restrição é aplicada ao seu dicionário.

Tabela 13: KLMS com dicionário variável empregado na identificação de um sistema não linear regido pela Equação (39). O sinal de entrada $u(n)$ consiste em um ruído gaussiano branco de média nula e $\sigma_u = 0{,}15$. Utilizou-se $\epsilon = 1{,}0$ e $h = 0{,}5$. Os dados são apresentados na forma média ± desvio padrão.

Algoritmo KLMS com $\epsilon = 1{,}0$ e $h = 0{,}5$	
μ	MSE (em dB)
0,001	-22,6706
1,0	-33,8035

4.3 Equalização de Canais

Nesta seção os algoritmos KLMS e KNLMS2 são aplicados à equalização de um canal com distorção não linear. Uma mensagem binária $m(n) \in \{-1{,}1\}$ é transmitida por um canal com função de transferência $H(z)$, cuja saída $y_h(n)$ sofre uma distorção caracterizada

pela equação

$$s(n) = f(y_h(n)) + z(n), \tag{44}$$

em que $s(n)$ denota o sinal que chega ao receptor, f representa uma função não linear e $z(n)$ é um ruído gaussiano branco de média nula e desvio padrão σ_z.

Nesse tipo de aplicação, o filtro adaptativo geralmente passa por um período de treinamento, em que o transmissor envia uma determinada sequência de *bits*, a qual deve ter sido combinada previamente com o receptor. Essa sequência é utilizada para compor o sinal $d(n)$ a ser aplicado ao filtro, enquanto que o sinal recebido $s(n)$ é a entrada a partir da qual o filtro tentará recompor a mensagem original. Passado o período de treinamento, o filtro adaptativo deixa de atualizar os seus coeficientes e opera como um filtro fixo.

Nesta seção, optou-se por simular apenas o período de treinamento, em que se conhece perfeitamente *a priori* o sinal desejado $d(n)$. A mensagem $m(n)$ foi gerada de modo que a cada iteração ambos os bits fossem equiprováveis, ou seja, para cada instante de tempo n tem-se $P(m(n) = -1) = P(m(n) = 1) = 0{,}5$. Modelou-se o canal como um filtro FIR com função de transferência

$$H(z) = 0{,}3 + z^{-1} + 0{,}3z^{-2}, \tag{45}$$

enquanto que a função f foi escolhida de modo a se obter

$$s(n) = y_h(n) + 0{,}5y_h^2(n) + 0{,}25y_h^3(n) + z(n), \tag{46}$$

sendo que se adotou σ_z de modo a se obter uma relação sinal-ruído de 30 dB. O sinal $d(n)$ é igual à mensagem $m(n)$, a menos de um atraso de três instantes de tempo.

Testaram-se diferentes combinações de μ, $\tilde{\mu}$, h e M a fim de encontrar os conjuntos de valores que levavam os algoritmos KLMS e KNLMS2 aos seus melhores desempenhos quando aplicados sem restrição de dicionário. Em seguida, mantendo-se fixos esses valores, os algoritmos foram empregados com o critério da coerência. As Figuras 13, 14 e 15 mostram os resultados obtidos para três situações distintas: sem nenhuma restrição ao crescimento dos dicionários, com o critério da coerência e $\epsilon = 0{,}9$ e com o critério da coerência e $\epsilon = 0{,}8$, respectivamente. Foram realizadas 100 experiências com 10000 iterações cada.

É importante ressaltar que nas últimas iterações das Figuras 13, 14 e 15 os algoritmos (principalmente o KNLMS2) dão sinais de que ainda estão convergindo. Contudo, seria inviável aumentar o número de iterações além do valor adotado de 10000 por experiência, pois as execuções se tornam proibitivamente demoradas conforme cresce o número de pontos. Pode-se argumentar que a equalização de canais é uma aplicação em que os algoritmos devem ser capazes de operar em tempo real, de modo que seria impraticável empregá-los sem nenhuma restrição ao crescimento de seus dicionários de qualquer forma. No entanto, o objetivo desta seção consiste em investigar os comportamentos dos algoritmos KLMS e KNLMS2 e obter comparações entre eles quando aplicados a esse tipo de situação do que em construir um cenário realista propriamente dito, de modo que se optou por realizar simulações sem restrições ao dicionário mesmo assim .

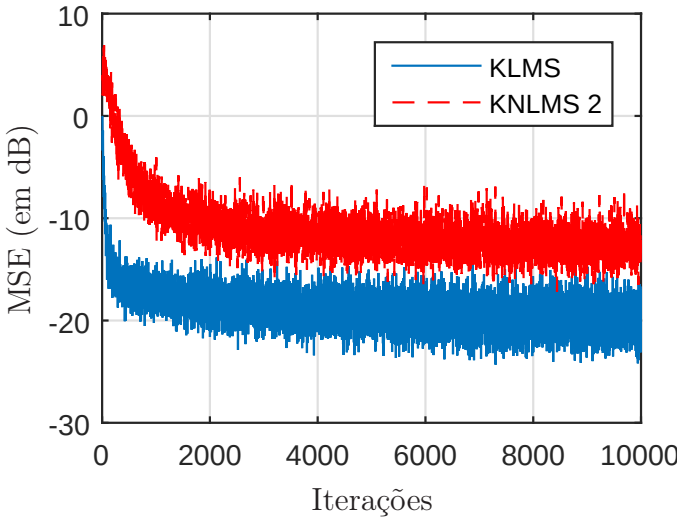


Figura 13: Curvas de erro quadrático médio dos algoritmos KLMS e KNLMS2 aplicados à equalização de um canal dado por (45) na presença de uma distorção não linear regida pela Equação (46). Para o KLMS adotou-se $\mu = 0{,}5$ e $h = 1{,}0$, enquanto que para o KNLMS2 foram utilizados $\tilde{\mu} = 0{,}05$, $h = 1$ e $\delta = 10^{-5}$. Em ambos os casos, adotou-se um vetor regressor composto por três amostras do sinal $s(n)$ (ou seja, escolheu-se $M = 3$). Não se restringiu o crescimento do dicionário dos algoritmos.

Nota-se que quanto mais se restringiu o dicionário, maior foi o erro quadrático médio atingido em regime pelos dois algoritmos, diferentemente do que ocorreu na Subseção 4.2.4, em que o KNLMS2 se mostrou capaz de fornecer resultados tão bons tanto com restrição ao crescimento do seu dicionário como sem. Contudo, assim como na Subseção 4.2.4, pode-

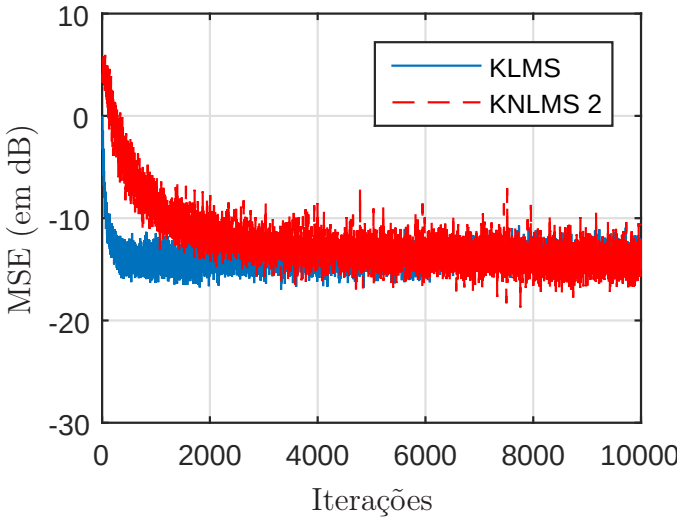


Figura 14: Curvas de erro quadrático médio dos algoritmos KLMS e KNLMS2 aplicados à equalização de um canal dado por (45) na presença de uma distorção não linear regida pela Equação (46). Para o KLMS adotou-se $\mu = 0{,}5$ e $h = 1{,}0$, enquanto que para o KNLMS2 foram utilizados $\tilde{\mu} = 0{,}05$, $h = 1$ e $\delta = 10^{-5}$. Em ambos os casos, adotou-se um vetor regressor composto por três amostras do sinal $s(n)$ (ou seja, escolheu-se $M = 3$). Aplicou-se o critério da coerência com $\epsilon = 0{,}9$.

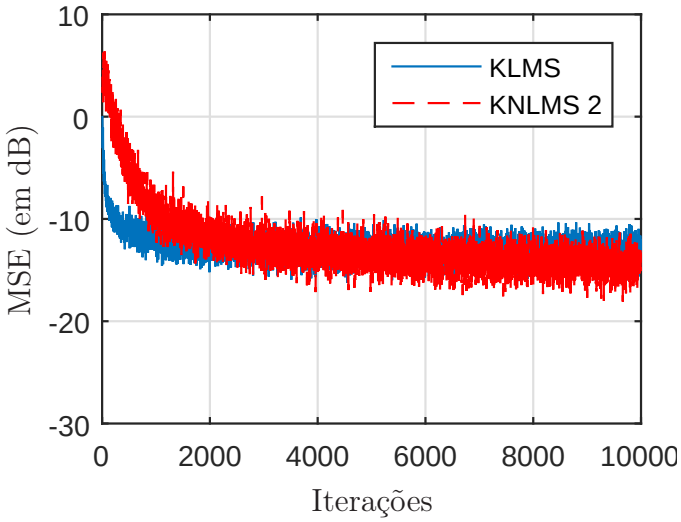


Figura 15: Curvas de erro quadrático médio dos algoritmos KLMS e KNLMS2 aplicados à equalização de um canal dado por (45) na presença de uma distorção não linear regida pela Equação (46). Para o KLMS adotou-se $\mu = 0{,}5$ e $h = 1{,}0$, enquanto que para o KNLMS2 foram utilizados $\tilde{\mu} = 0{,}05$, $h = 1$ e $\delta = 10^{-5}$. Em ambos os casos, adotou-se um vetor regressor composto por três amostras do sinal $s(n)$ (ou seja, escolheu-se $M = 3$). Aplicou-se o critério da coerência com $\epsilon = 0{,}8$.

se notar que o KLMS foi mais afetado pela restrição do dicionário do que o KNLMS2: escolhendo-se $\epsilon = 1{,}0$, o desempenho do primeiro algoritmo foi visivelmente superior, ao

passo que para $\epsilon = 0{,}9$ as curvas de MSE se tornam muito semelhantes após a convergência inicial, e para $\epsilon = 0{,}8$ o KNLMS2 passa a apresentar um desempenho melhor nas últimas iterações.

Para efeito de comparação, a Figura 16 mostra os algoritmos LMS e NLMS aplicados a essa mesma situação, com passos de adaptação $\mu = 0{,}01$ e $\tilde{\mu} = 0{,}05$, respectivamente, e $M = 10$ e $\delta = 10^{-5}$. Esses são os valores que os levaram ao melhor desempenho que pôde ser observado nas simulações.

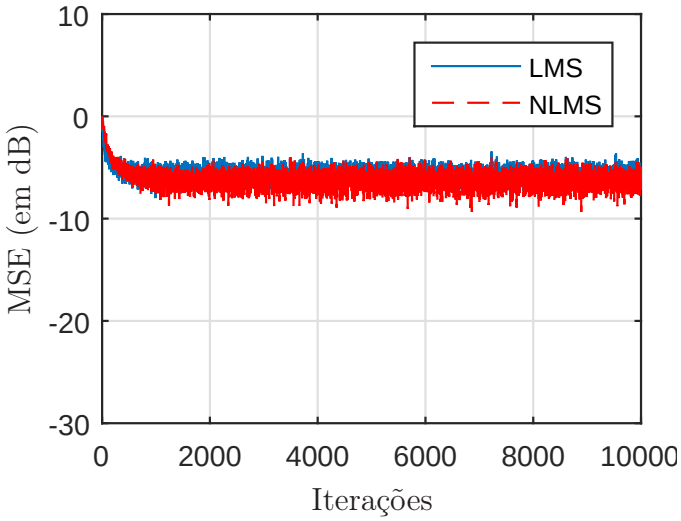


Figura 16: Curvas de erro quadrático médio dos algoritmos LMS e NLMS aplicados à equalização de um canal dado por (45) na presença de uma distorção não linear regida pela Equação (46). Adotou-se $\mu = 0{,}01$, $\tilde{\mu} = 0{,}05$, $\delta = 10^{-5}$ e $M = 10$.

Pode-se ver que o desempenho alcançado pelos algoritmos KLMS e KNLMS2 com os valores escolhidos de μ, $\tilde{\mu}$, h, M e ϵ foi superior àquele mostrado pelos algoritmos lineares. Contudo, cabe ressaltar que a redução do nível de coerência a valores menores do que os indicados nas figuras anteriores levou o KLMS e o KNLMS2 a níveis de MSE proibitivos em regime, inclusive mais elevados do que aqueles atingidos pelo LMS e pelo NLMS.

5 Análises e Conclusões Finais

Esta seção sintetiza os resultados apresentados na Seção 4, expondo as principais observações que podem ser extraídas das análises teóricas realizadas e dos dados experimentais.

Neste relatório foi mostrado de forma teórica na Seção 4.1 que dois algoritmos que vinham sendo tratados como iguais na literatura consultada são na verdade diferentes entre si. Em seguida, nas Seções 4.2 e 4.3 se mostrou que esses algoritmos não apenas são diferentes como também apresentam propriedades diferentes. Ao longo de toda essa seção se mostrou que variações no passo de adaptação, na largura do *kernel* e na restrição ao crescimento do dicionário impactam esses algoritmos de formas distintas.

Dos resultados obtidos na Seção 4.2.1, foi possível perceber que, assim como ocorre com o LMS e o NLMS, a inicialização do vetor de coeficientes dos algoritmos KLMS2 e KNLMS2 e do coeficiente $\mathbf{a}(1)$ do algoritmo KLMS pode alterar o tempo necessário para a convergência, mas na prática não influencia o valor final do MSE. Isso significa que, embora possa ser difícil prever qual inicialização se mostrará melhor em cada situação, isso não afetará o desempenho desses algoritmos em regime permanente, o que é bastante desejável. Trata-se de uma constatação importante, porque na prática é raro sabermos *a priori* qual é a melhor inicialização para o vetor de coeficientes em cada aplicação. É importante ter em mente que uma inicialização pode levar a uma convergência rápida em uma dada aplicação e não se mostrar tão boa em outra aplicação. Isso pode ser facilmente constatado comparando-se os resultados obtidos na Subseção 4.2 com aqueles mostrados na Subseção 4.3: as mesmas inicializações que levaram os algoritmos a convergir praticamente instantaneamente na primeira produziram um efeito diferente na segunda, em que as curvas de MSE mostram claramente um período de convergência mais longo.

Na Subseção 4.2.2, mostrou-se que os algoritmos KLMS e KNLMS2 são vantajosos em relação ao KLMS2 por serem mais estáveis, apresentando uma tendência menor a divergir, mesmo com o aumento do passo de adaptação. O algoritmo KLMS com restrição ao dicionário se mostrou especialmente insensível a mudanças no valor de μ, apresentando comportamentos muito semelhantes mesmo para passos de adaptação de diferentes ordens de grandeza.

Os resultados da Subseção 4.2.3 indicam que há uma relação entre o passo de adaptação e a largura do *kernel* nos algoritmos KLMS2 e KNLMS2. Nessa subseção se mostrou que o menor valor de passo de adaptação não necessariamente leva ao menor erro em regime. Pela

analogia com o caso linear, seria de se esperar que, para um valor fixo de h, a redução do passo de adaptação diminuísse o erro quadrático médio em regime e tornasse a convergência mais lenta. Contudo, não é isso que se constata nas simulações. Na literatura consultada, não se encontrou nenhuma menção a esse fato. Um estudo mais profundo acerca dessa relação entre a largura do *kernel* e o passo de adaptação se mostra necessário.

Ainda na Seção 4.2.3, foi mostrado que escolher valores mais elevados de h podem levar o KLMS2 à divergência, dependendo do valor do passo de adaptação, o que não ocorre com o KNLMS2 ou com o KLMS. Trata-se de mais uma razão pela qual esses algoritmos se mostram mais interessantes, principalmente quando se quiser testar diferentes valores para a largura do *kernel*.

Na Seção 4.2.4, mostrou-se o impacto de se restringir o crescimento do dicionário. Com $h = 1{,}0$ e $\tilde{\mu} = 1{,}0$, o tempo médio de execução do algoritmo KNLMS2 sem restrição do dicionário foram cerca de 68 vezes maiores que os da versão com um nível de coerência de 0,001, sendo que o MSE em regime praticamente não se alterou com essa mudança. Ou seja, nessa aplicação a restrição ao dicionário não afetou o desempenho desse algoritmo e tornou a sua execução muito mais rápida. No caso do algoritmo KLMS, utilizando-se $h = 0{,}5$ e $\mu = 1{,}0$, a diferença entre os tempos de execução foi ainda maior: sem restrição, o algoritmo demorou em média mais de mil vezes do que com o critério da coerência e $\epsilon = 0{,}001$. Em contrapartida, o desempenho sem a restrição foi significativamente melhor do que qualquer resultado obtido pelo mesmo algoritmo implementado com o critério da coerência e $\epsilon = 0{,}001$ ou com um número fixo de elementos no dicionário.

Ainda na Subseção 4.2.4, mostrou-se que o algoritmo KLMS sem restrição do dicionário apresenta a mesma relação entre h e μ observada no caso do algoritmo KNLMS2, o que não ocorreu quando se fixou o número de elementos no seu dicionário, nem quando se utilizou o critério da coerência. Isso mostra que a restringir o crescimento do dicionário pode alterar não apenas o desempenho do algoritmo KLMS, mas também a influência do passo de adaptação no seu comportamento.

Essa relação encontrada nas simulações entre h e μ, no caso do KLMS sem restrição do dicionário e do KLMS2, ou entre h e $\tilde{\mu}$, no caso do KNLMS2, ainda não é bem compreendida

e torna mais complicada a escolha dos parâmetros dos algoritmos estudados. Por enquanto, a melhor abordagem parece ser testar diferentes valores de largura do *kernel* e de passo de adaptação, e fazer combinações desses valores até se encontrar uma combinação que leve a um desempenho satisfatório.

Além disso, utilizou-se a teoria para mostrar que, utilizando o critério da coerência, o tamanho do dicionário depende da largura do *kernel* (com valores menores de h levando a dicionários maiores, em concordância com os Limites (41) e (42)), mas independe do valor do passo de adaptação ou da escolha de se utilizar o KLMS ou o KNLMS2. Esse resultado teórico foi corroborado pelos dados experimentais obtidos.

Comparando-se os resultados da Subseção 4.2.4 com os das Subseções 4.2.1 a 4.2.3, vê-se que tanto o critério da coerência como a manutenção de um número fixo de elementos no dicionário se mostraram métodos válidos para restringir o seu crescimento, mas pode-se argumentar que o primeiro é mais confiável, uma vez que só são incorporados os vetores de entrada que podem formar uma base em $\mathbb{F}$ (além do fato de que a escolha do número de elementos do dicionário tende a ser mais difícil do que a escolha do nível de coerência).

É importante ressaltar que as conclusões obtidas sobre os efeitos da restrição ao crescimento do dicionário dizem respeito à manutenção de um número fixo de elementos no dicionário e ao critério da coerência. Como mencionado na Subseção 4.2.4, existem outros métodos que não foram testados neste relatório. Pode ser interessante em estudos posteriores investigar se esses critérios impactam os algoritmos da mesma forma que os métodos utilizados neste trabalho.

Já na Seção 4.3, os algoritmos KLMS e KNLMS2 foram aplicados à equalização de um canal na presença de distorção não linear. Nessa aplicação, a restrição ao crescimento do dicionário levou a uma piora no desempenho dos dois algoritmos. Em conjunto com o resultado da Seção 4.2.4, isso indica que a restrição ao dicionário do KNLMS2 pode impactar ou não o seu desempenho, dependendo da aplicação. Nessa seção também se repetiu a conclusão a que se tinha chegado na Seção 4.2.4 de que o algoritmo KLMS é mais afetado pela restrição ao dicionário do que o KNLMS2. Por fim, nessa seção também se pôde verificar que os algoritmos LMS e NLMS apresentaram desempenho bastante inferior em comparação

com as soluções não lineares, exceto quando estas foram aplicadas com um nível de coerência suficientemente baixo.

A fim de resumir as principais informações extraídas das simulações realizadas, as características dos algoritmos KLMS e KNLMS2 foram sintetizadas na forma de tópicos a seguir. Além disso, na Tabela 14 são descritos os impactos dos parâmetros μ (ou $\tilde{\mu}$, h e ϵ sobre os algoritmos quando estes são utilizados aplicando-se restrições ao crescimento de seus dicionários. Como se constatou que a inicialização dos algoritmos não influencia os seus desempenhos em regime, seus efeitos não foram incluídos na tabela.

- Algoritmo KLMS
  - Seu desempenho em regime não é afetado por sua inicialização;
  - Com restrições impostas ao seu dicionário, mostra-se pouco sensível a variações no passo;
  - Sem essas restrições, apresenta relação entre o passo de adaptação e a largura do *kernel*;
  - Seu desempenho é significativamente afetado pela aplicação de restrições ao seu dicionário.
- Algoritmo KNLMS2
  - Seu desempenho em regime também não é afetado por sua inicialização;
  - Apresenta relação entre $\tilde{\mu}$ e h, independentemente de restrições impostas ao dicionário;
  - A sua versão não normalizada (o algoritmo KLMS2) apresenta uma tendência cada vez maior a divergir conforme se aumenta a largura do *kernel*;
  - Relativamente menos afetado pela restrição ao dicionário.

Tabela 14: Comparação entre os algoritmos KLMS e KNLMS2 com restrições ao crescimento de seus dicionários (tamanho fixo e critério da coerência).

Algoritmos KLMS e KNLMS2 Com Restrições ao Dicionário			
Algoritmo	Passo de Adaptação	Largura do *Kernel*	Nível de Coerência
KLMS	Pouco sensível a variações no passo. Não apresenta relação clara com a largura do *kernel*.	É mais prejudicado por valores muito elevados de h do que o KNLMS2. Usando-se o critério da coerência, quanto menor a largura, mais elementos são incorporados ao dicionário.	Apresenta maior sensibilidade a esse parâmetro do que o KNLMS2. Quanto menor ϵ, menos elementos são incorporados ao seu dicionário, e pior o seu desempenho.
KNLMS2	Apresenta relação com a largura do *kernel*.	Usando-se o critério da coerência, quanto menor a largura, mais elementos são incorporados ao dicionário.	Comparativamente menos sensível, podendo não ter seu desempenho significativamente afetado dependendo da aplicação e do valor de $\tilde{\mu}$ e h. Quanto menor o seu valor, menos elementos são incorporados ao dicionário.

Referências

[1] S. Haykin, *Adaptive Filter Theory*, Prentice Hall, Upper Saddle River, 4th edition, 2002.

[2] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, NJ, 2008.

[3] V. H. Nascimento and M. T. M. Silva, "Adaptive filters," in *Academic Press Library in Signal Processing: Signal Processing Theory and Machine Learning*, R. Chellapa and S. Theodoridis, Eds., vol. 1, chapter 12, pp. 619–761. Academic Press, Chennai, 2014.

[4] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, Wiley, 2010.

[5] B. Widrow, "Thinking about thinking: the discovery of the LMS algorithm," *IEEE Signal Processing Magazine*, vol. 22, no. 1, pp. 100–106, Jan. 2005.

[6] S. Y. Kung, *Kernel Methods and Machine Learning*, Cambridge University Press, 2014.

[7] B. Scholkopf and A. J. Smola, *Learning with Kernels: support vector machines, regularization, optimization, and beyond*, MIT Press, Cambridge, 2002.

[8] I. Steinwart, "On the influence of the kernel on the consistency of support vector machines," *The Journal of Machine Learning Research*, vol. 2, pp. 67–93, 2002.

[9] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2009.

[10] W. D. Parreira, *Comportamento estocástico do algoritmo kernel least-mean-square*, Tese de Doutorado em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2012.

[11] W. D. Parreira, J. C. M. Bermudez, C. Richard, and J.Y. Tourneret, "Stochastic behavior analysis of the gaussian kernel least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2208–2222, 2012.

[12] P. Honeine, C. Richard, J.C.M. Bermudez, and H. Snoussi, "Distributed prediction of time series data with kernels and adaptive filtering techniques in sensor networks," in *2008 42nd Asilomar Conference on Signals, Systems and Computers*. IEEE, 2008, pp. 246–250.

[13] W. Liu, I. Park, and J. C. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters," *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1950–1961, Dec 2009.

[14] R. Candido, *A questão da equalização em sistemas de comunicação que utilizam sinais caóticos.*, Tese de Doutorado em Engenharia Elétrica, Universidade de São Paulo, São Paulo, 2015.

[15] M. Hénon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics*, vol. 50, no. 1, pp. 69–77, 1976.

[16] G. A. Abib and M. Eisencraft, "Sobre o desempenho em canal com ruıdo de um sistema de comunicaçao baseado em caos," *Anais do XXXI Simpósio Brasileiro de Telecomunicaçoes (SBrT'13)*, 2013.

[17] S. Boccaletti, J. Kurths, G. Osipov, D.L. Valladares, and C.S. Zhou, "The synchronization of chaotic systems," *Physics reports*, vol. 366, no. 1, pp. 1–101, 2002.

Apêndice A

Equalização de Canais de Comunicação Baseados em Caos

Originalmente, o objetivo deste projeto de Iniciação Científica consistia em empregar algoritmos baseados em núcleo na equalização de canais de comunicação baseados em caos e na predição de séries temporais caóticas. No entanto, no decorrer do trabalho se constatou que a aplicação desses algoritmos era mais complicada do que se pressupunha a princípio, e que a literatura existente não respondia a várias dúvidas que vieram à tona durante a realização das atividades programadas. Assim, considerou-se que seria importante investigar melhor algumas de suas propriedades antes de partir para aplicações mais complexas – como as que envolvem caos. Posteriormente se percebeu que havia dois algoritmos que eram tratados como iguais em muitas das referências consultadas, mas que eram na verdade diferentes entre si. A combinação desses fatores levou a equipe a alterar o foco deste relatório, o qual passou a versar mais sobre as propriedades dos algoritmos estudados e a compará-los entre si do que sobre a aplicação desses algoritmos nas situações inicialmente pretendidas.

Contudo, também foi realizado um desenvolvimento teórico acerca de sistemas de comunicação baseados em caos e se executaram algumas simulações relacionadas ao assunto. Este apêndice visa a mostrar alguns dos resultados obtidos, mais a fim de guiar estudos posteriores do que de tirar conclusões definitivas acerca do assunto. Por brevidade, a formulação completa do problema não será dada aqui, podendo ser encontrada em detalhes em [14]. Em vez disso, serão apresentadas aqui apenas as informações essenciais para o entendimento do assunto.

Deseja-se enviar uma mensagem binária $m(n) \in \{-1,1\}$ codificada. Para isso, gera-se um sinal [14]

$$s(n) = \gamma_1 x_1(n) - \gamma_2[m(n)+1]\text{sign}[\gamma_1 x_1(n)], \tag{47}$$

em que γ_1 e γ_2 são constantes,

$$\text{sign}[x] = \begin{cases} -1, & x < 0 \\ \\ 1, & x \geq 0 \end{cases}, \tag{48}$$

e $x_1(n)$ é um dos componentes do sistema dinâmico bidimensional de tempo discreto $\mathbf{x}(n) = [x_1(n)\ x_2(n)]^{\mathrm{T}}$ caracterizado pela expressão

$$\mathbf{x}(n+1) = \begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \begin{bmatrix} x_2(n) + 1 - \eta s^2(n) \\ \theta x_1(n) \end{bmatrix}, \tag{49}$$

em que η e θ são constantes. A Equação (49) representa uma versão modificada do chamado *mapa de Hénon* [14, 15], podendo ser reescrita como

$$\mathbf{x}(n+1) = \mathbf{A}\mathbf{x}(n) + \mathbf{b} + \mathbf{f}(s(n)), \tag{50}$$

com

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ \theta & 0 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{f}(s(n)) = \begin{bmatrix} -\eta s^2(n) \\ 0 \end{bmatrix}. \tag{51}$$

A Equação (47) representa uma forma de codificar a mensagem proposta em [14] com base na chamada *codificação com soma* [16]. É importante frisar que há outras formas de codificar a mensagem, mas apenas essa forma de codificação será abordada neste relatório. Da mesma forma, diversos mapas podem ser utilizados na codificação, mas neste relatório só se usará o mapa de Hénon.

Por sua vez, seja $\hat{\mathbf{x}}(n)$ um sistema regido por

$$\hat{\mathbf{x}}(n+1) = \mathbf{A}\hat{\mathbf{x}}(n) + \mathbf{b} + \mathbf{f}(s(n)), \tag{52}$$

e seja $\mathbf{e}_{\mathbf{x}}(n) \triangleq \mathbf{x}(n) - \hat{\mathbf{x}}(n)$. Das Equações (50) e (52) tem-se

$$\mathbf{e}_{\mathbf{x}}(n+1) = \mathbf{A}e_{\mathbf{x}}(n). \tag{53}$$

Se $\lim_{n\to\infty} \mathbf{e}_{\mathbf{x}}(n) = \mathbf{0}$, diz-se que os sistemas $\mathbf{x}(n)$ e $\hat{\mathbf{x}}(n)$ são *perfeitamente sincronizados* [17]. Um condição suficiente para isso é que todos os autovalores de $\mathbf{A}$ estejam dentro do círculo unitário no plano complexo [14].

A ideia dos sistemas de comunicação baseados em caos consiste em empregar dois geradores de sinais caóticos: um no transmissor e o outro no receptor. O primeiro gera o sinal $\mathbf{x}(n)$, que é então utilizado para codificar a mensagem, enquanto que o segundo gera o sinal $\hat{\mathbf{x}}(n)$ a fim de decodificá-la. A Equação (53) indica que se os dois geradores utilizarem os mesmos

valores de $\mathbf{A}$ e $\mathbf{b}$ e a mesma função $\mathbf{f}(\cdot)$, esses sistemas estarão perfeitamente sincronizados e consequentemente $\hat{\mathbf{x}}(n)$ será uma estimativa cada vez mais próxima de $\mathbf{x}(n)$.

Note que as Equações (50) e (52) exigem que o sinal $s(n)$ enviado pelo transmissor chegue inalterado ao receptor. No entanto, em qualquer situação prática o sinal recebido $r(n)$ será diferente do enviado, pois este está sujeito a interferência intersimbólica e a diversas fontes de ruído. Assim sendo, é necessário primeiramente se obter uma estimativa $\hat{s}(n)$ do sinal original para depois decodificá-lo e obter a mensagem original.

Tal estimativa pode ser obtida empregando-se um equalizador. De posse de $\hat{s}(n)$ e de $\hat{\mathbf{x}}(n)$, é possível estimar então a mensagem original. Para o caso da função de codificação regida pela Equação (47) empregando-se o mapa de Hénon, a equação de decodificação é dada por [14]

$$\hat{m}(n) = \frac{\gamma_1 \hat{x}_1(n) - \hat{s}(n)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(n)]} - 1. \tag{54}$$

Note que analisando-se a Equação (47) vê-se que se $\gamma_1 > 0$, $\gamma_2 > 0$, $m(n) = 1$ e $2\gamma_2 > \text{sign}[\gamma_1 x_1(n)]$, $s(n)$ ficará com o sinal trocado em relação a $\gamma_1 x_1(n)$. O efeito disso na decodificação é que se obtém $\hat{m}(n) = -3$ quando na realidade se deveria obter $\hat{m}(n) = 1$ [14]. Contudo, é possível corrigir esse erro considerando-se $\hat{m}(n) \leftarrow \hat{m}(n) + 4$ quando $-3{,}5 < \hat{m}(n) < -2{,}5$.

Em [14] foram propostos algoritmos voltados à equalização de canais de comunicação com codificação regida pela Equação (47), os quais foram denominados cLMS$_+$ e cNLMS$_+$. Esse algoritmo utilizam a estrutura de um filtro adaptativo FIR, assim como os algoritmos LMS e NLMS convencionais. Suas saídas são dadas por

$$\hat{s}(n) = \mathbf{w}_+^{\text{T}}(n-1)\mathbf{r}(n), \tag{55}$$

em que $\mathbf{w}_+^{\text{T}}(n)$ é um vetor de coeficientes variantes no tempo e r(n) é um vetor regressor composto por amostras do sinal $r(n)$. Definindo-se $e_+(n) = m(n-\Delta) - \hat{m}(n)$, em que Δ denota um atraso no tempo, e $\hat{J}_+(n) = e_+^2(n)$, tem-se [14]

$$\boldsymbol{\nabla}_{\mathbf{w}_+} \hat{J}(n) = 2e_+(n)\frac{\partial e_+(n)}{\partial \mathbf{w}_+(n-1)} = -2e_+(n)\frac{\partial \hat{m}(n)}{\partial \mathbf{w}_+(n-1)}. \tag{56}$$

Levando-se em conta as Equações (54) e (55), pode-se reescrever a expressão acima como

$$\boldsymbol{\nabla}_{\mathbf{w}_+}\hat{J}(n) = -2\frac{e_+(n)}{\gamma_2 \text{sign}[\gamma_1\hat{x}_1(n)]}\left[-\frac{\partial\hat{s}(n)}{\partial\mathbf{w}_+(n-1)}\right] = 2\frac{e_+(n)}{\gamma_2 \text{sign}[\gamma_1\hat{x}_1(n)]}\mathbf{r}(n). \tag{57}$$

Note que se assumiu que $x_1(n)$ não depende de $\mathbf{w}_+(n-1)$. Na verdade existe, sim, uma dependência, pois o cálculo de $\hat{\mathbf{x}}(n)$ utiliza o valor atual do sinal $\hat{s}(n)$, que por sua vez depende de $\mathbf{w}(n-1)$. No entanto, sem essa hipótese, as expressões se tornam muito difíceis de serem resolvidas analiticamente, e chega-se a um algoritmo mais complicado.

Com base na Equação (57), a equação de atualização do vetor de coeficientes do algoritmo cLMS$_+$ fica sendo [14]

$$\mathbf{w}_+(n) = \mathbf{w}_+(n-1) - \mu\frac{e_+(n)}{\gamma_2 \text{sign}[\gamma_1\hat{x}_1(n)]}\mathbf{r}(n). \tag{58}$$

Para se obter o algoritmo cNLMS$_+$, define-se um erro *a posteriori*

$$e_p(n) = m(n-\Delta) - \frac{\gamma_1\hat{x}_1(n) - \mathbf{w}_+(n)\mathbf{r}(n)}{\gamma_2 \text{sign}[\gamma_1\hat{x}_1(n)]} + 1. \tag{59}$$

Usando-se $\mu(n)$ em vez de μ e substituindo-se o valor de $\mathbf{w}_+(n)$ da Equação (58) na Equação (59), obtém-se [14]

$$e_p(n) = e_+(n)\left[1 - \mu(n)\frac{||\mathbf{r}(n)||^2}{\gamma_2^2 \text{sign}^2[\gamma_1\hat{x}_1(n)]}\right]. \tag{60}$$

Portanto, para garantir que o erro *a posteriori* seja nulo em todos os instantes de tempo n, deve-se fazer [14]

$$\mu(n) = \frac{\gamma_2^2 \text{sign}^2[\gamma_1\hat{x}_1(n)]}{||\mathbf{r}(n)||^2}. \tag{61}$$

Introduzindo-se um passo fixo $\tilde{\mu}$ para controlar a velocidade de convergência e uma parcela δ para evitar divisão por zero no denominador, obtém-se a equação de atualização do vetor de coeficientes do algoritmo cNLMS$_+$ [14]

$$\mathbf{w}_+(n) = \mathbf{w}_+(n-1) - \frac{\tilde{\mu}}{\delta + ||\mathbf{r}(n)||^2}\gamma_2 \text{sign}[\gamma_1\hat{x}_1(n)]e_+(n)\mathbf{r}(n). \tag{62}$$

Com base nisso, buscou-se desenvolver algoritmos análogos baseados em núcleo, os quais serão chamados a partir daqui de cKLMS$_+$, cKLMS2$_+$ e cKNLMS2$_+$, os quais serão apresentados a seguir.

Os algoritmos cKLMS2$_+$ e cKNLMS2$_+$ foram obtidos definindo-se as suas saídas $\hat{s}(n)$ por meio de

$$\hat{s}(n) = \boldsymbol{\alpha}^{\mathrm{T}}(n-1)\mathbf{k}(n). \tag{63}$$

Mantendo-se as definições de $\hat{J}_+(n)$, $e_+(n)$ e $\hat{m}(n)$, tem-se que o gradiente de $\hat{J}_+(n)$ pode ser calculado por meio de

$$\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\hat{J}(n) = 2e_+(n)\frac{\partial e_+(n)}{\partial \boldsymbol{\alpha}(n-1)} = -2e_+(n)\frac{\partial \hat{m}(n)}{\partial \boldsymbol{\alpha}(n-1)}. \tag{64}$$

Substituindo a Equação (54) na expressão acima e considerando-se a Equação (63), tem-se

$$\boldsymbol{\nabla}_{\boldsymbol{\alpha}}\hat{J}(n) = -2\frac{e_+(n)}{\gamma_2 \mathrm{sign}[\gamma_1 \hat{x}_1(n)]}\left[-\frac{\partial \hat{s}(n)}{\partial \boldsymbol{\alpha}(n-1)}\right] = 2\frac{e_+(n)}{\gamma_2 \mathrm{sign}[\gamma_1 \hat{x}_1(n)]}\mathbf{k}(n). \tag{65}$$

Dessa forma, a atualização do vetor $\boldsymbol{\alpha}(n)$ no caso do algoritmo cKLMS2$_+$ é dada por

$$\boldsymbol{\alpha}(n) = \boldsymbol{\alpha}(n-1) - \mu\frac{e_+(n)}{\gamma_2 \mathrm{sign}[\gamma_1 \hat{x}_1(n)]}\mathbf{k}(n). \tag{66}$$

De forma análoga à obtenção do cNLMS$_+$ a partir do cNLMS$_+$, a definição de um erro *a posteriori* e a manipulação algébrica das Equações (63) a (66) leva à equação de atualização do algoritmo cKNLMS2$_+$, dada por

$$\boldsymbol{\alpha}(n) = \boldsymbol{\alpha}(n-1) - \frac{\tilde{\mu}}{\delta + ||\mathbf{k}(n)||^2}\gamma_2 \mathrm{sign}[\gamma_1 \hat{x}_1(n)]e_+(n)\mathbf{k}(n). \tag{67}$$

Por fim, o algoritmo cKLMS$_+$ foi desenvolvido considerando-se

$$\hat{s}(n) = \boldsymbol{\omega}^{\mathrm{T}}(n-1)\boldsymbol{\Phi}(n). \tag{68}$$

Tem-se

$$\boldsymbol{\nabla}_{\boldsymbol{\omega}}\hat{J}(n) = 2e_+(n)\frac{\partial e_+(n)}{\partial \boldsymbol{\omega}(n-1)} = -2e_+(n)\frac{\partial \hat{m}(n)}{\partial \boldsymbol{\omega}(n-1)}, \tag{69}$$

que leva a

$$\boldsymbol{\nabla}_{\boldsymbol{\omega}}\hat{J}(n) = -2\frac{e_+(n)}{\gamma_2 \mathrm{sign}[\gamma_1 \hat{x}_1(n)]}\left[-\frac{\partial \hat{s}(n)}{\partial \boldsymbol{\omega}(n-1)}\right] = 2\frac{e_+(n)}{\gamma_2 \mathrm{sign}[\gamma_1 \hat{x}_1(n)]}\boldsymbol{\Phi}(n). \tag{70}$$

Portanto, o cálculo da atualização do vetor de coeficientes será dado por

$$\boldsymbol{\omega}(n) = \boldsymbol{\omega}(n-1) - \mu\frac{e_+(n)}{\gamma_2 \mathrm{sign}[\gamma_1 \hat{x}_1(n)]}\boldsymbol{\Phi}(n). \tag{71}$$

Reescrevendo a equação acima recursivamente e admitindo-se $\boldsymbol{\omega}(0) = \mathbf{0}$, chega-se a

$$\begin{aligned}
\boldsymbol{\omega}(n) &= \boldsymbol{\omega}(n-1) - \mu \frac{e_+(n)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(n)]} \boldsymbol{\Phi}(n) \\
&= \left[\boldsymbol{\omega}(n-2) - \mu \frac{e_+(n-1)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(n-1)]} \boldsymbol{\Phi}(n-1) \right] - \mu \frac{e_+(n)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(n)]} \boldsymbol{\Phi}(n) \\
&= \boldsymbol{\omega}(n-2) - \mu \left[\frac{e_+(n-1)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(n-1)]} \boldsymbol{\Phi}(n-1) + \frac{e_+(n)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(n)]} \boldsymbol{\Phi}(n) \right] \\
&\vdots \\
&= \boldsymbol{\omega}(0) - \mu \sum_{j=1}^{n} \frac{e_+(j)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(j)]} \boldsymbol{\Phi}(j) \\
&= -\mu \sum_{j=1}^{n} \frac{e_+(j)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(j)]} \boldsymbol{\Phi}(j).
\end{aligned} \tag{72}$$

Com base no último somatório e utilizando o Truque do *Kernel*, a saída $\hat{s}(n)$ pode então ser calculada por meio de

$$\hat{s}(n) = -\mu \sum_{j=1}^{n-1} \frac{e_+(j)}{\gamma_2 \text{sign}[\gamma_1 \hat{x}_1(j)]} k(\mathbf{r}(n), \mathbf{r}(j)). \tag{73}$$

A seguir são apresentados os resultados de algumas simulações empregando o cKLMS$_+$ e o cKNLMS2$_+$ na equalização de um canal modelado por um filtro FIR com função de transferência

$$H(z) = 0{,}3 + z^{-1} + 0{,}3z^{-2} \tag{74}$$

e na ausência de ruído. Adotaram-se nas simulações $\eta = 1{,}4$, $\theta = 0{,}3$, $\gamma_1 = 0{,}9$ e $\gamma_2 = 0{,}3$. Em cada caso foram realizadas 100 experiências com 5000 iterações cada. Não se aplicou nenhuma técnica de restrição ao crescimento dos dicionários.

Pode-se ver que o algoritmo cKLMS$_+$ mostrou um desempenho superior ao exibido pelo cKNLMS2$_+$, cujo nível de MSE em regime se mostrou proibitivamente elevado para essa dada aplicação.

Como mencionado no início deste Apêndice, não se pretende aqui tirar conclusões definitivas acerca do cKLMS$_+$ e do cKNLMS2$_+$. Como o foco do trabalho mudou durante a sua elaboração, ainda há muitas investigações necessárias a se fazer em relação aos algoritmos propostos ou mesmo em relação à possibilidade de implementação de outros algoritmos similares (utilizando outras funções de codificação ou outros mapas, por exemplo). No entanto,

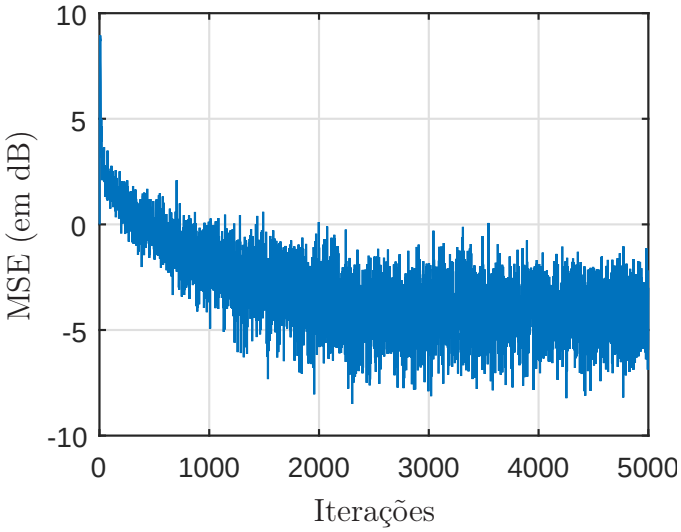


Figura 17: Erro quadrático médio apresentado ao longo do tempo pelo algoritmo cKLMS$_+$ aplicado à equalização de um canal de coeficientes dados pela Equação (74). Utilizou-se $\mu = 0{,}01$, $h = 1$, $M = 5$ e $\Delta = 3$.

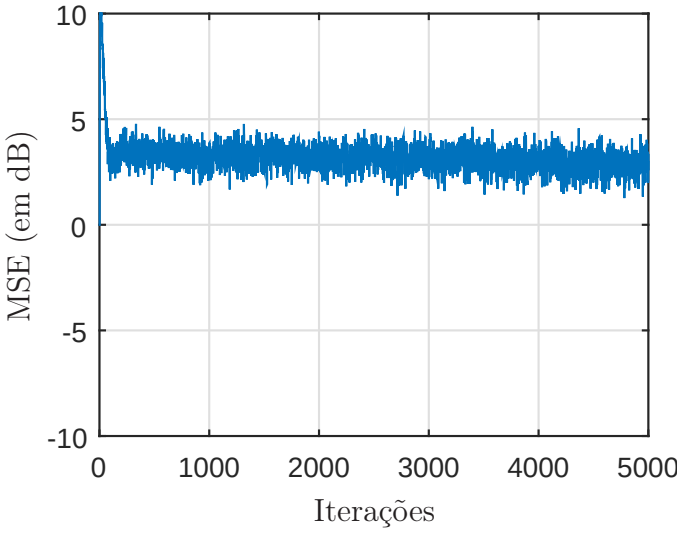


Figura 18: Erro quadrático médio apresentado ao longo do tempo pelo algoritmo cKNLMS2$_+$ aplicado à equalização de um canal de coeficientes dados pela Equação (74). Utilizou-se $\tilde{\mu} = 0{,}01$, $h = 1$, $M = 5$, $\delta = 10^{-5}$ e $\Delta = 3$.

o trabalho realizado nesse campo pode servir de base para estudos posteriores, e por isso se considerou que era importante documentá-lo e deixá-lo registrado.