Um algoritmo *kernel* baseado na ortogonalização de Gram-Schmidt para redes de difusão adaptativas

André A. Bueno, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva

***Resumo*— Redes de difusão adaptativas têm atraído atenção na comunidade científica como soluções eficientes para estimação distribuída de sinais, sendo também utilizadas para estimação não linear com algoritmos adaptativos baseados em núcleo (*kernel*). Essas soluções apresentam um elevado custo computacional devido ao dicionário dos algoritmos *kernel*. Neste artigo, é proposto um algoritmo *kernel* de custo computacional reduzido para redes de difusão adaptativas. Ele utiliza a ortogonalização de Gram-Schmidt para calcular a base do espaço vetorial dos elementos mapeados do dicionário, gerando um dicionário com cardinalidade reduzida. Resultados de simulação mostram uma economia de custo computacional em comparação com técnicas clássicas.**

***Palavras-Chave*— Redes de difusão adaptativas, filtragem adaptativa baseada em núcleo, esparsificação de dicionário, processamento não linear de sinais.**

***Abstract*— Adaptive diffusion networks have attracted attention in the scientific community as an efficient solution for distributed estimation of signals. They also have been employed in nonlinear estimation problems with distributed kernel adaptive algorithms. These solutions present a high computational cost due to the dictionary of kernel algorithms. In this paper, we propose a reduced-cost kernel algorithm for adaptive diffusion networks. It is based on the Gram-Schmidt orthogonalization process, used to span the vector space of the mapped vectors contained in the dictionary, which leads to a dictionary with a reduced cardinality. By means of simulations, we observe an advantageous computational savings in comparison to classical techniques.**

***Keywords*— Adaptive diffusion networks, kernel adaptive filtering, dictionary sparsification, nonlinear signal processing.**

I. Introdução e formulação do problema

Métodos baseados em núcleo (*kernel*) são capazes de resolver problemas não lineares, projetando o vetor de entrada em um espaço de dimensão mais alta, onde um método linear é usado [1]. Nesses métodos, o vetor coluna $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^M$ é mapeado em um espaço de alta dimensão $\mathcal{H}$ como $\boldsymbol{\varphi}(\mathbf{u})$, usando um *kernel* de Mercer $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, tal que [2], [3]

$$\kappa(\mathbf{u},\mathbf{v}) = \langle \boldsymbol{\varphi}(\mathbf{u}), \boldsymbol{\varphi}(\mathbf{v}) \rangle_{\mathcal{H}} \triangleq \boldsymbol{\varphi}(\mathbf{u})^{\mathrm{T}} \boldsymbol{\varphi}(\mathbf{v}), \quad (1)$$

$\forall \mathbf{u}, \mathbf{v} \in \mathcal{U}$, em que $(\cdot)^{\mathrm{T}}$ representa o operador de transposição de uma matriz. A Eq. (1) é conhecida como truque do *kernel* já que o produto interno dos vetores mapeados $\boldsymbol{\varphi}(\mathbf{u})$ e $\boldsymbol{\varphi}(\mathbf{v})$ pode ser calculado no espaço $\mathcal{H}$ sem se conhecer explicitamente a função $\boldsymbol{\varphi}(\cdot)$ [1]. O *kernel* Gaussiano, $\kappa(\mathbf{u},\mathbf{v}) = e^{-\zeta \|\mathbf{u}-\mathbf{v}\|^2}$, é o mais utilizado na literatura, em que $\|\cdot\|$ denota a norma euclidiana, $\zeta = 1/(2\sigma^2)$ sendo σ a largura do *kernel*. Esse *kernel* induz um espaço de dimensão infinita, é numericamente estável e apresenta a capacidade de aproximação universal, por isso, costuma ser a escolha padrão [1].

André A. Bueno, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, e-mails: {aabueno, dtiglea, renatocan, magno}@lps.usp.br. Este trabalho foi financiado pela CAPES (88887.603151/2021-00, 88887.512247/2020-00 e 001) e FAPESP (2021/02063-6).

A Eq. (1) possibilita gerar versões *kernel* de filtros adaptativos, como é o caso do algoritmo *kernel least-mean-squares* (KLMS) [3]. No espaço $\mathcal{H}$, o algoritmo LMS é usado para atualizar o vetor coluna de coeficientes a fim de estimar o sinal desejado $d(n)$ [3], ou seja,

$$\boldsymbol{\omega}(n) = \boldsymbol{\omega}(n-1) + \mu \Big[d(n) - \boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{u}(n)) \boldsymbol{\omega}(n-1) \Big] \boldsymbol{\varphi}(\mathbf{u}(n)), \quad (2)$$

em que $\boldsymbol{\omega}(0) = \mathbf{0}$, μ é um passo de adaptação e $\mathbf{u}(n)$ o vetor de entrada do filtro. Como não se conhece explicitamente a função $\boldsymbol{\varphi}(\cdot)$ e o espaço $\mathcal{H}$ pode ser de dimensão infinita, não é possível implementar o algoritmo da forma (2). Utilizando a Eq. (1), é possível implementar o KLMS da forma [4]

$$\boldsymbol{\omega}(n) = \boldsymbol{\omega}(n-1) + \mu [d(n) - \boldsymbol{\kappa}_N^{\mathrm{T}}(n) \boldsymbol{\omega}(n-1)] \boldsymbol{\kappa}_N(n), \quad (3)$$

em que

$$\boldsymbol{\kappa}_N(n) = [\kappa(\mathbf{u}(n),\mathbf{u}_{\mathcal{D}_1})\ \kappa(\mathbf{u}(n),\mathbf{u}_{\mathcal{D}_2}) \cdots \kappa(\mathbf{u}(n),\mathbf{u}_{\mathcal{D}_N})]^{\mathrm{T}}. \quad (4)$$

A cada iteração n, o KLMS utiliza no cálculo de $\boldsymbol{\kappa}_N(n)$ os vetores do dicionário $\mathcal{D} = \{\mathbf{u}_{\mathcal{D}_1}, \mathbf{u}_{\mathcal{D}_2}, \cdots, \mathbf{u}_{\mathcal{D}_N}\}$. No KLMS convencional sem esparsificação, $\mathbf{u}_{\mathcal{D}_1} = \mathbf{u}(1)$, $\mathbf{u}_{\mathcal{D}_2} = \mathbf{u}(2)$, $\cdots$, $\mathbf{u}_{\mathcal{D}_N} = \mathbf{u}(n-1)$. Neste caso, a cardinalidade do dicionário e a dimensão do vetor de coeficientes $\boldsymbol{\omega}(n)$ vale $N = n-1$ e cresce linearmente com n [3]. O subscrito que aparece no vetor $\boldsymbol{\kappa}$ indica sua dimensão.

Devido ao crescimento linear do dicionário com o tempo, métodos *kernel* apresentam elevado custo computacional, o que pode inviabilizá-los. Diante disso, algumas técnicas de esparsificação foram propostas na literatura para redução do dicionário [1], [5], [6]. Dentre essas técnicas, o critério da coerência (CC) [5] é um dos mais utilizados. Para decidir se $\mathbf{u}(n)$ deve fazer parte do dicionário, o CC calcula o produto interno entre $\boldsymbol{\varphi}(\mathbf{u}(n))$ e $\boldsymbol{\varphi}(\mathbf{u}_{\mathcal{D}_i})$, $i = 1,2,\cdots,N$, usando a Eq. (1). Se $\max_i |\kappa(\mathbf{u}(n), \mathbf{u}_{\mathcal{D}_i})| \leqslant \varepsilon_{CC}$, sendo ε_{CC} um limiar escolhido pelo projetista do filtro, o vetor $\mathbf{u}(n)$ é incluído no dicionário como o elemento $\mathbf{u}_{N+1}$ e o vetor de coeficientes $\boldsymbol{\omega}(n)$ tem sua dimensão aumentada para $N+1$, o que é feito concatenando esse vetor com zero. Caso contrário, o dicionário e a dimensão de $\boldsymbol{\omega}(n)$ permanecem como antes.

Uma outra técnica de esparsificação que merece destaque foi proposta recentemente em [7]. Ela utiliza a ortogonalização de Gram-Schmidt (GS) para calcular a base do espaço vetorial dos vetores mapeados do dicionário. A cada novo vetor de entrada $\mathbf{u}(n)$, a técnica verifica se a projeção do vetor $\boldsymbol{\varphi}(\mathbf{u}(n))$ nesse espaço é menor que um limiar predefinido. Se isso ocorrer, o vetor $\mathbf{u}(n)$ é inserido no dicionário e um novo vetor

da base desse espaço é calculado. Caso contrário, o dicionário e a base permanecem inalterados. Essa técnica foi aplicada ao KLMS, dando origem ao algoritmo GS-KLMS [7]. Por meio de simulações, verificou-se que o GS-KLMS apresenta um erro quadrático médio (*mean square error* – MSE) semelhante ao do KLMS implementado em conjunto com técnicas de esparsificação, mas com um custo computacional reduzido. Isso ocorre, pois o procedimento de GS para o cálculo da base leva a um dicionário com cardinalidade muito inferior ao do KLMS com CC, por exemplo.

Os algoritmos adaptativos *kernel* também têm sido utilizados recentemente em problemas não lineares de estimação distribuída com redes de difusão. Uma rede de difusão consiste em um conjunto de V nós conectados a partir de uma topologia predefinida. Cada nó é capaz de coletar dados locais, realizar cálculos e se comunicar com nós vizinhos. O objetivo da rede é estimar um sinal de interesse sem uma unidade central de processamento [8]. Aplicações em potencial para redes de difusão baseadas em *kernel* incluem, por exemplo, o monitoramento de fenômenos naturais e de infraestrutura [9]. A vizinhança do nó k, denotada por $\mathcal{N}_k$, é dada pelo conjunto de seus vizinhos, incluindo o próprio nó k, como ilustrado na Fig. 1. A cada iteração, cada nó k tem acesso a um vetor de entrada $\mathbf{u}_k(n)$ e a um sinal desejado $d_k(n) = \phi^{\mathrm{o}}[\mathbf{u}_k(n)]+v_k(n)$, sendo ϕ^{o} uma função não linear e $v_k(n)$ o ruído de medição no nó k. Esse ruído é considerado independente de qualquer outra variável e estacionário no sentido amplo com média nula e variância $\sigma^2_{v_k}$.

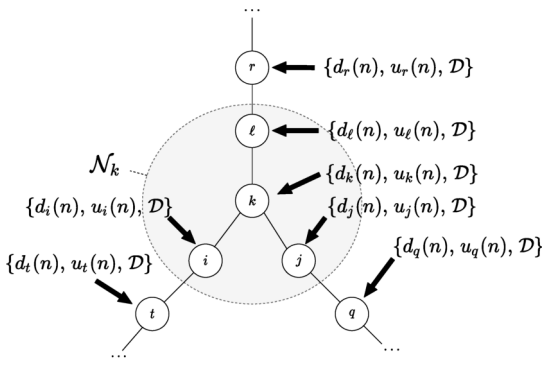

Fig. 1: Exemplo de uma rede de difusão. Neste caso, o conjunto $\mathcal{N}_k$ é formado pelos nós i, j, k e ℓ.

A extensão do KLMS para estimação distribuída leva a um custo computacional elevado, principalmente se cada nó tiver seu próprio dicionário, mesmo considerando técnicas de esparsificação. Para reduzir o custo computacional, considera-se um dicionário $\mathcal{D}$ fixo com N elementos, comum a todos os nós. Assim, seguindo uma configuração do tipo ATC (*adapt-then-combine*)[1] [8], as etapas de adaptação e de combinação do KLMS distribuído (dKLMS) são respectivamente dadas por [9], [10]

$$\boldsymbol{\delta}_k(n) = \boldsymbol{\omega}_k(n\text{–}1)+\mu_k[d_k(n)-\boldsymbol{\kappa}_N^{(k)\mathrm{T}}(n)\boldsymbol{\omega}_k(n\text{–}1)]\boldsymbol{\kappa}_N^{(k)}(n) \quad (5a)$$

$$\boldsymbol{\omega}_k(n) = \textstyle\sum_{j\in\mathcal{N}_k} c_{jk}\boldsymbol{\delta}_j(n), \quad (5b)$$

[1]Neste trabalho, apenas a abordagem ATC será considerada, mas os resultados podem ser estendidos para uma estratégia do tipo CTA (*combine-then-adapt*), em que a ordem das Eqs. (5a) and (5b) é trocada [8].

em que $\boldsymbol{\delta}_k$ e $\boldsymbol{\omega}_k$ são, respectivamente, as estimativas local e combinada de $\boldsymbol{\omega}$ no nó k, μ_k um passo de adaptação e

$$\boldsymbol{\kappa}_N^{(k)}(n)=[\kappa(\mathbf{u}_k(n),\mathbf{u}_{\mathcal{D}_1}),\cdots,\kappa(\mathbf{u}_k(n),\mathbf{u}_{\mathcal{D}_N})]^{\mathrm{T}}. \quad (6)$$

Por fim, os $\{c_{jk}\}$ são pesos de combinação que satisfazem $c_{jk} \geqslant 0$, $\sum_{j\in\mathcal{N}_k} c_{jk} = 1$ e $c_{jk} = 0$ se $j \notin \mathcal{N}_k$ [11], [12]. Possíveis escolhas para os $\{c_{jk}\}$ incluem as regras Uniforme, Metrópolis, do Grau Relativo e do Grau Máximo [8]. Neste trabalho, utilizou-se apenas a regra do Grau Máximo, cujos pesos são dados por [8]

$$c_{jk} = \begin{cases} \dfrac{1}{\max_{i=1,\cdots,V}|\mathcal{N}_i|}, \text{ se } j\in\mathcal{N}_k,\ j\neq k, \\ 1-\dfrac{|\mathcal{N}_k|-1}{\max_{i=1,\cdots,V}|\mathcal{N}_i|}, \text{ se } j=k, \\ 0, \text{ caso contrário.} \end{cases} \quad (7)$$

Em muitas aplicações, para que o dKLMS atinja um nível aceitável de MSE em regime permanente, é necessário que o dicionário contenha muitos elementos. Mesmo considerando um dicionário fixo e comum a todos os nós, o custo computacional do dKLMS com CC, por exemplo, é muito elevado. Neste contexto, o uso do GS-KLMS é vantajoso devido ao seu dicionário reduzido. Neste artigo, o GS-KLMS de [7] é estendido para o caso distribuído e comparado com o dKLMS implementado com o CC. O dicionário de ambos algoritmos são considerados fixos, mas é possível verificar, por meio de simulações, que o fato do GS-KLMS ter um dicionário reduzido leva a uma vantajosa economia de custo computacional. O artigo está organizado da seguinte forma. Na Seção II, o algoritmo GS-KLMS é revisitado e estendido para o caso distribuído. Na Seção III, são apresentados resultados de simulação. Por fim, na Seção IV são apresentadas as principais conclusões do artigo.

II. O ALGORITMO GS-KLMS DISTRIBUÍDO

Seja $\mathcal{D} \triangleq \{\mathbf{u}_{\mathcal{D}_1}, \mathbf{u}_{\mathcal{D}_2}, \cdots, \mathbf{u}_{\mathcal{D}_N}\}$ um dicionário cujos vetores mapeados $\boldsymbol{\varphi}(\mathcal{D}) \triangleq \{\boldsymbol{\varphi}(\mathbf{u}_{\mathcal{D}_1}),\boldsymbol{\varphi}(\mathbf{u}_{\mathcal{D}_2}),\cdots,\boldsymbol{\varphi}(\mathbf{u}_{\mathcal{D}_N})\}$ formam um conjunto linearmente independente na iteração n. Usando $\boldsymbol{\varphi}(\mathcal{D})$, uma base ortonormal $\mathbf{B}_N \triangleq \{\boldsymbol{\beta}_1,\boldsymbol{\beta}_2,\cdots,\boldsymbol{\beta}_N\}$ do espaço vetorial $\mathcal{B} \subset \mathcal{H}$ pode ser calculada. A projeção ortogonal de $\boldsymbol{\varphi}(\mathbf{u}(n))$ em $\mathcal{B}$ é dada por $\mathbf{p}(n) \triangleq \mathrm{proj}_{\mathcal{B}}\boldsymbol{\varphi}(\mathbf{u}(n)) = \sum_{i=1}^{N} p_i(n)\boldsymbol{\beta}_i$, em que $p_i(n) \triangleq \langle\boldsymbol{\varphi}(\mathbf{u}(n)),\boldsymbol{\beta}_i\rangle_{\mathcal{H}}$. Se $\mathbf{u}(n)$ coincide com um dos elementos de $\mathcal{D}$, a projeção de $\boldsymbol{\varphi}(\mathbf{u}(n))$ em $\mathcal{B}$ é o próprio vetor $\boldsymbol{\varphi}(\mathbf{u}(n))$, ou seja, $\boldsymbol{\varphi}(\mathbf{u}(n)) = \mathbf{p}(n)$ já que $\boldsymbol{\varphi}(\mathbf{u}(n))$ foi usado para criar a base $\mathbf{B}_N$. Caso contrário, $\boldsymbol{\varphi}(\mathbf{u}(n)) = \mathbf{p}(n) + \mathbf{p}^{\perp}(n)$, em que $\mathbf{p}^{\perp}(n) \neq \mathbf{0}$ é o componente de $\boldsymbol{\varphi}(\mathbf{u}(n))$ ortogonal a $\mathcal{B}$. Os escalares $p_i(n)$, $i = 1,2,\cdots,N$, podem ser agrupados no vetor $\tilde{\mathbf{p}}(n) = [\,p_1(n)\ p_2(n)\ \cdots\ p_N(n)\,]^{\mathrm{T}}$, que é usado como uma representação do vetor $\boldsymbol{\varphi}(\mathbf{u}(n))$. O fato de se usar uma base ortonormal possibilita calcular $\langle\mathbf{p}(\ell),\mathbf{p}(m)\rangle_{\mathcal{H}}$ como $\tilde{\mathbf{p}}^{\mathrm{T}}(\ell)\tilde{\mathbf{p}}(m)$, para todos os vetores $\mathbf{p}(\ell),\mathbf{p}(m) \in \mathcal{B}$. Essa representação pode ser usada para substituir $\boldsymbol{\varphi}(\mathbf{u}(n))$ na Eq. (2).

Como demonstrado em [7], utilizando o processo de ortogonalização de GS e a Eq. (1), é possível calcular o vetor $\tilde{\mathbf{p}}(n)$ de forma eficiente sem se conhecer explicitamente $\boldsymbol{\varphi}(\cdot)$. Esse vetor é calculado como $\tilde{\mathbf{p}}(n) = \mathbf{H}_N\boldsymbol{\kappa}_N(n)$ com $\boldsymbol{\kappa}_N(n)$ definido em (4). A matriz quadrada $\mathbf{H}_N$ é calculada de

forma recorrente a partir dos vetores do dicionário $\mathcal{D}$. Assim, supondo que $\mathbf{u}_{\mathcal{D}_N}$ foi o último elemento inserido no dicionário e definindo o vetor

$$\widetilde{\boldsymbol{\kappa}}_{N-1} \triangleq [\kappa(\mathbf{u}_{\mathcal{D}_N},\mathbf{u}_{\mathcal{D}_1})\ \kappa(\mathbf{u}_{\mathcal{D}_N},\mathbf{u}_{\mathcal{D}_2}) \cdots \kappa(\mathbf{u}_{\mathcal{D}_N},\mathbf{u}_{\mathcal{D}_{N-1}})]^{\mathrm{T}},$$

a matriz $\mathbf{H}_N$ é calculada como

$$\mathbf{H}_N = \begin{bmatrix} \mathbf{H}_{N-1} & \mathbf{0}_{N-1\times 1} \\ \overline{\mathbf{h}}/\sqrt{\overline{\mathbf{h}}\mathbf{G}_N\overline{\mathbf{h}}^{\mathrm{T}}} \end{bmatrix},$$

em que

$$\mathbf{G}_N = \begin{bmatrix} \mathbf{G}_{N-1} & \widetilde{\boldsymbol{\kappa}}_{N-1} \\ \widetilde{\boldsymbol{\kappa}}_{N-1}^{\mathrm{T}} & \kappa(\mathbf{u}_{\mathcal{D}_N},\mathbf{u}_{\mathcal{D}_N}) \end{bmatrix}$$

é a matriz Gramiano, $\overline{\mathbf{h}} = [-\widetilde{\boldsymbol{\kappa}}_{N-1}^{\mathrm{T}}\mathbf{H}_{N-1}^{\mathrm{T}}\mathbf{H}_{N-1} \quad 1]$ é um vetor linha e $\mathbf{0}_{N-1\times 1}$ representa um vetor de zeros com dimensão $N-1\times 1$. A forma recorrente de calcular as matrizes $\mathbf{H}_N$ e $\mathbf{G}_N$ exige uma inicialização, ou seja, $\mathbf{G}_1 = \mathbf{H}_1 = 1$ e $\mathbf{u}_{\mathcal{D}_1} = \mathbf{u}(1)$. Uma vez calculado o vetor $\widetilde{\mathbf{p}}(n)$, o vetor mapeado $\boldsymbol{\varphi}(\mathbf{u}(n))$ da Eq. (2) pode ser substituído por essa representação, o que leva à seguinte equação de atualização para o GS-KLMS:

$$\boldsymbol{\omega}(n) = \boldsymbol{\omega}(n-1) + \mu[d(n) - \widetilde{\mathbf{p}}^{\mathrm{T}}(n)\boldsymbol{\omega}(n-1)]\widetilde{\mathbf{p}}(n).$$

O procedimento para calcular $\widetilde{\mathbf{p}}(n)$ é usado no GS-KLMS como técnica de esparsificação [7]. A ideia é decidir se o vetor $\mathbf{u}(n) \notin \mathcal{D}$ deve ser incluído ou não no dicionário. Para isso, verifica-se se os vetores do conjunto aumentado $\boldsymbol{\varphi}(\mathcal{D}) \cup \boldsymbol{\varphi}(\mathbf{u}(n))$ são próximos de serem linearmente dependentes. Isso pode ser medido pelo determinante da matriz Gramiano. Assim, escolhendo-se o limiar ε_{GS}, se

$$\|\mathbf{p}^{\perp}(n)\|^2 = \kappa(\mathbf{u}(n),\mathbf{u}(n)) - \widetilde{\mathbf{p}}^{\mathrm{T}}(n)\widetilde{\mathbf{p}}(n) < \varepsilon_{GS} \qquad (8)$$

for satisfeito, conjectura-se que o conjunto aumentado $\boldsymbol{\varphi}(\mathcal{D}) \cup \boldsymbol{\varphi}(\mathbf{u}(n))$ está próximo de ser linearmente dependente e $\mathbf{u}(n)$ não é incluído no dicionário [7]. Caso contrário, $\mathbf{u}(n)$ é incluído no dicionário, as matrizes $\mathbf{G}_{N+1}$ e $\mathbf{H}_{N+1}$ são calculadas e o vetor de coeficientes $\boldsymbol{\omega}(n)$ tem sua dimensão aumentada para $N+1$, o que é feito concatenando esse vetor com zero. Para o kernel Gaussiano, $\kappa(\mathbf{u}(n),\mathbf{u}(n)) = 1$ e (8) se reduz a $\widetilde{\mathbf{p}}^{\mathrm{T}}(n)\widetilde{\mathbf{p}}(n) > 1 - \varepsilon_{GS}$. Como $0 < \widetilde{\mathbf{p}}^{\mathrm{T}}(n)\widetilde{\mathbf{p}}(n) \leqslant 1$, o limiar deve ser escolhido no intervalo $0 \leqslant \varepsilon_{GS} \leqslant 1$, o que simplifica o ajuste desse parâmetro. Neste caso, com $\varepsilon_{GS} = 1$ nenhum vetor é incluído no dicionário e com $\varepsilon_{GS} = 0$ o dicionário conterá todos os vetores de entrada. Na Fig. 2(a), é mostrado um vetor $\boldsymbol{\varphi}(\mathbf{u}(n))$, cuja representação a partir dos componentes de sua projeção na base $\mathbf{B}_2$ não é apropriada, ou seja, $\|\mathbf{p}^{\perp}(n)\| \geqslant \varepsilon_{GS}$. Na Fig. 2(b), é mostrado um caso com representação apropriada, ou seja, com $\|\mathbf{p}^{\perp}(n)\| < \varepsilon_{GS}$. Os vetores com projeção não apropriada devem ser incluídos no dicionário.

É importante observar que a maior parte do custo computacional do GS-KLMS se deve ao cálculo das matrizes $\mathbf{G}_{N+1}$ e $\mathbf{H}_{N+1}$ para a inclusão do vetor $\mathbf{u}(n)$ no dicionário. Considerando um dicionário fixo, o custo computacional tanto do GS-KLMS quanto do KLMS convencional depende do número de vetores contidos em $\mathcal{D}$, tendo a mesma ordem de grandeza para um dicionário com cardinalidade N. Em geral, o

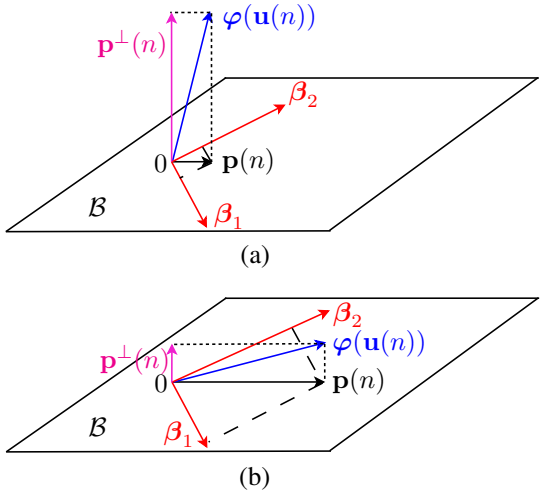

Fig. 2: Representação (a) não apropriada e (b) apropriada do vetor $\boldsymbol{\varphi}(\mathbf{u}(n))$ a partir dos componentes de sua projeção.

GS-KLMS necessita de menos elementos no dicionário para atingir o mesmo desempenho do KLMS implementado com uma técnica de esparsificação. Assim, o GS-KLMS consegue uma vantajosa economia de custo computacional quando se considera um dicionário fixo, como ocorre no caso distribuído.

Para um dicionário fixo de cardinalidade N, a extensão do GS-KLMS para redes de difusão adaptativas segue os mesmos passos da do KLMS [9], [10]. Para uma configuração ATC, as equações do GS-KLMS distribuído (dGS-KLMS) são dadas para o nó k por

$$\boldsymbol{\delta}_k(n) = \boldsymbol{\omega}_k(n-1) + \mu_k[d_k(n) - \widetilde{\mathbf{p}}_k^{\mathrm{T}}(n)\boldsymbol{\omega}_k(n-1)]\widetilde{\mathbf{p}}_k^{\mathrm{T}}(n) \qquad (9a)$$

$$\boldsymbol{\omega}_k(n) = \sum_{j\in\mathcal{N}_k} c_{jk}\boldsymbol{\delta}_j(n). \qquad (9b)$$

O vetor $\widetilde{\mathbf{p}}_k^{\mathrm{T}}(n)$ deve ser calculado como $\widetilde{\mathbf{p}}_k^{\mathrm{T}}(n) = \mathbf{H}_N\boldsymbol{\kappa}_N^{(k)}$, com $\boldsymbol{\kappa}_N^{(k)}$ definido em (6). Como o dicionário é fixo, a matriz $\mathbf{H}_N$ também é fixa e é comum a todos os nós da rede.

III. Resultados de Simulação

Nesta seção, são apresentados resultados de simulações a fim de ilustrar o comportamento do dGS-KLMS. Para efeito de comparação, também são apresentados os resultados obtidos com o algoritmo dKLMS e com as versões não-cooperativas de ambos os algoritmos, denotadas aqui por GS-KLMS e KLMS. Nas simulações, consideraram-se a rede mostrada na Fig. 3 com $V = 9$ nós, pesos $\{c_{jk}\}$ dados pela regra do Grau Máximo e kernel Gaussiano com parâmetro $\zeta = 1$. Os resultados apresentados foram obtidos a partir de uma média de 100 realizações para cada simulação. Como métrica de desempenho, adotou-se o erro quadrático médio da rede (*network* MSE – NMSE), dado por

$$\mathrm{NMSE}(n) = \frac{1}{V}\sum_{k=1}^{V} \mathrm{E}\{[d_k(n) - \widetilde{\mathbf{u}}^{\mathrm{T}}(n)\boldsymbol{\omega}_k(n)]^2\}, \qquad (10)$$

em que $\widetilde{\mathbf{u}}(n) = \boldsymbol{\kappa}_N^{(k)}(n)$ para algoritmos do tipo KLMS e $\widetilde{\mathbf{u}}(n) = \widetilde{\mathbf{p}}(n)$ para algoritmos do tipo GS-KLMS. Os algoritmos foram ajustados de forma a se obter o melhor desempenho em termos de NMSE em regime permanente.

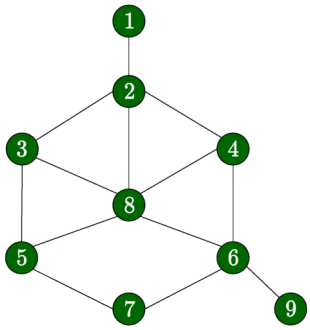

Fig. 3: Rede considerada nas simulações.

Nas simulações das Figuras 4, 5 e 6, considerou-se um problema de identificação de sistemas, em que o sistema a ser identificado é dado por [4], [13]

$$x(n) = \frac{x(n-1)}{1+x^2(n-1)} + u^3(n-1) \tag{11}$$

$$d(n) = x(n) + \nu(n), \tag{12}$$

sendo $u(n)$ um ruído branco Gaussiano com média nula e desvio padrão $\sigma_u = 0{,}35$ e ν um ruído de medição também branco, de média nula e desvio padrão $\sigma_\nu = 10^{-2}$. Os vetores de entrada dos filtros adaptativos são da forma $\mathbf{u}(n) = [u(n)\ u(n-1)\ u(n-2)]^{\mathrm{T}}$. Para gerar o dicionário fixo do dGS-KLMS e do GS-KLMS, foi utilizado o parâmetro $\varepsilon_{GS} = 1 - 0{,}99^2 = 0{,}0199$, enquanto que para o dicionário fixo dos algoritmos KLMS e dKLMS, foi usado o critério da coerência com limiar $\varepsilon_{CC} = 0{,}98$. Adotaram-se $\mu_k = 1$ para todos os nós no caso do dGS-KMLMS e GS-KLMS e $\mu_k = 0{,}005$ para todos os nós no caso do dKLMS e KLMS. O tamanho médio do dicionário para o dGS-KLMS e GS-KLMS foi de 107 elementos, enquanto para os algoritmos KLMS e dKLMS foi de 1503. A partir da Fig. 4, é possível verificar que as versões cooperativas, dKLMS e dGS-KLMS, apresentam um NMSE aproximadamente 1 dB menor do que as não cooperativas, KLMS e GS-KLMS. Nas Figuras 5 e 6, mostra-se o número de multiplicações e exponenciais acumuladas, respectivamente, para os algoritmos d-KLMS e d-GS-KLMS. O custo computacional das versões centralizadas não é mostrado por ser inerentemente menor, uma vez que tais soluções contam com um único nó. Entretanto, é interessante ressaltar que o custo computacional dos algoritmos KLMS e GS-KLMS apresenta comportamento análogo aos das Figuras 5 e 6 [7]. Pode-se observar que algoritmo dGS-KLMS requer aproximadamente $6{,}2 \times 10^8$ multiplicações e exponenciais a menos na iteração $n = 5\times10^4$ em comparação com o dKLMS, o que representa um custo cerca de 15 vezes menor. Apesar do dGS-KLMS apresentar um NMSE ligeiramente superior ao do dKLMS, essa diferença de cerca de 0,2 dB é compensada pela economia expressiva de custo computacional.

Nas simulações das Figuras 7, 8 e 9, considerou-se novamente um problema de identificação de sistemas, em que o sistema a ser identificado é dado por [4]

$$x(n) = u(n) + 0{,}5u(n-1) - 0{,}2x(n-1) + 0{,}35x(n-2), \tag{13}$$

$$\phi(n) = \begin{cases} \dfrac{x(n)}{3\sqrt{0{,}1+0{,}9x^2(n)}}, & \text{se } x(n) \geqslant 0 \\ \dfrac{-x^2(n)(1-e^{0{,}7x(n)})}{3}, & \text{se } x(n) < 0, \end{cases} \tag{14}$$

$$d(n) = \phi(n) + \nu(n), \tag{15}$$

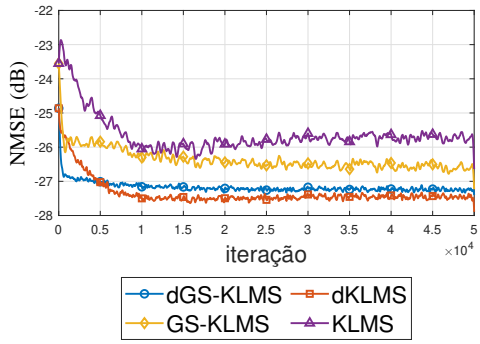

Fig. 4: NMSE apresentado pelos algoritmos dGS-KLMS, dKLMS, GS-KLMS e KLMS para a identificação do sistema dado pelas Equações (11) e (12).

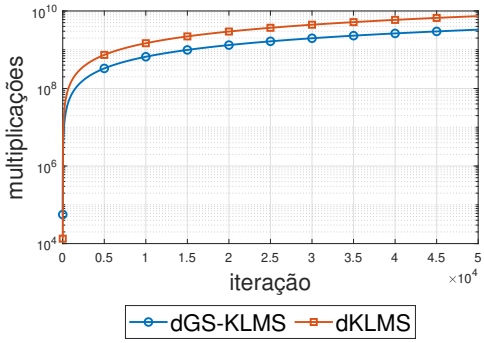

Fig. 5: Número de multiplicações acumuladas por iteração dos algoritmos dGS-KLMS, dKLMS, GS-KLMS e KLMS para a identificação do sistema dado pelas Equações (11) e (12).

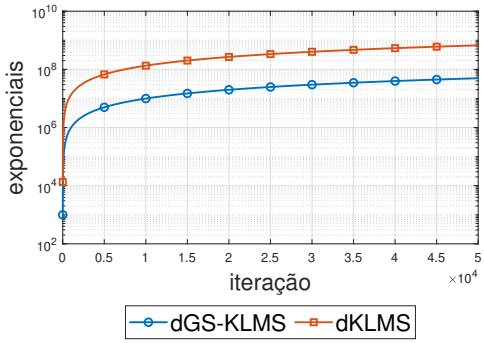

Fig. 6: Número de exponenciais acumuladas por iteração dos algoritmos dGS-KLMS, dKLMS, GS-KLMS e KLMS para a identificação do sistema dado pelas Equações (11) e (12).

em que $u(n)$ é um ruído branco Gaussiano com média nula e desvio padrão $\sigma_u = 0{,}15$ e ν é um ruído de medição também branco, de média nula e desvio padrão $\sigma_\nu = 10^{-3}$. Os vetores de entrada dos filtros adaptativos são da forma $\mathbf{u}(n) = [u(n)\ u(n-1)\ u(n-2)\ u(n-3)\ u(n-4)]^{\mathrm{T}}$. Para gerar o dicionário fixo dos algoritmos dGS-KLMS e GS-KLMS, foi utilizado o parâmetro $\varepsilon_{GS} = 1 - 0{,}99^2 = 0{,}0199$, enquanto para o dicionário fixo dos algoritmos KLMS e dKLMS, foi usado o critério da coerência com limiar $\varepsilon_{CC} = 0{,}99$. Para todos os nós, adotaram-se $\mu_k = 0{,}5$ no caso do GS-KLMS e dGS-KLMS e $\mu_k = 0{,}00005$ para o KLMS e o dKLMS. Cabe observar que passos maiores para KLMS e dKLMS levam a um desempenho em regime permanente pior do que o observado na Fig. 7. O tamanho médio do dicionário para o dGS-KLMS e GS-KLMS foi de 79 elementos enquanto que para os

algoritmos KLMS e dKLMS foi de 7401. A partir da Fig. 7, verifica-se que os algoritmos do tipo GS-KLMS atingiram um NMSE cerca de 5 dB mais baixo que os algoritmos do tipo KLMS. Além disso, a versão cooperativa do GS-KLMS atinge um NMSE cerca de 1 dB menor que a não cooperativa. Nas Figuras 8 e 9, são mostrados o número de multiplicações e exponenciais acumuladas, respectivamente. Pode-se observar que algoritmo dGS-KLMS requer aproximadamente $3{,}3 \times 10^9$ multiplicações e exponenciais a menos na iteração $n = 5 \times 10^4$ em comparação com o dKLMS, o que representa um custo cerca de 100 vezes menor. Neste cenário, além de atingir um nível de NMSE em regime mais baixo, o dGS-KNLMS apresenta novamente uma economia expressiva em termos de custo computacional em comparação ao dKLMS.

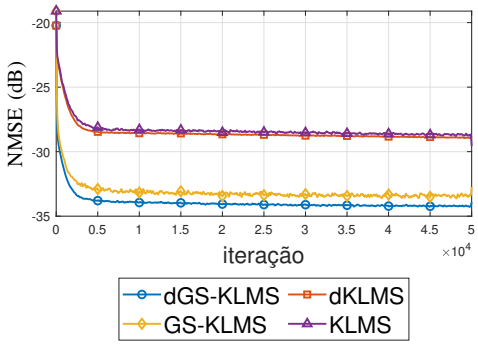

Fig. 7: NMSE apresentado pelos algoritmos dGS-KLMS, dKLMS, GS-KLMS e KLMS para a identificação do sistema dado pelas Equações (13)–(15).

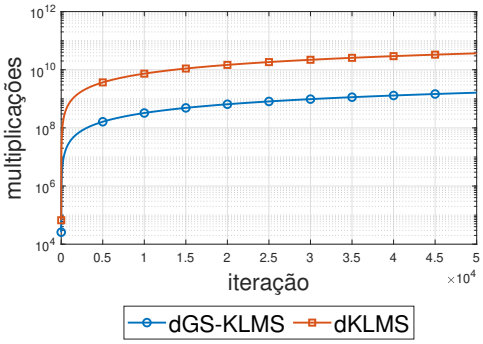

Fig. 8: Número de multiplicações acumuladas por iteração dos algoritmos dGS-KLMS, dKLMS, GS-KLMS e KLMS para a identificação do sistema dado pelas Equações (13)–(15).

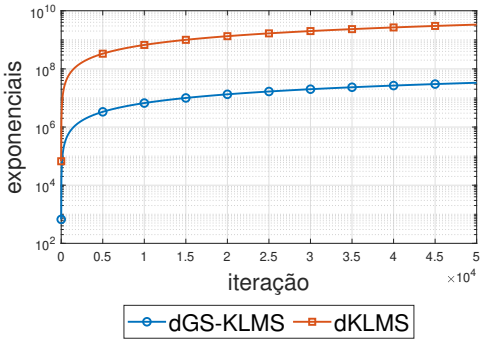

Fig. 9: Número de exponenciais acumuladas por iteração dos algoritmos dGS-KLMS, dKLMS, GS-KLMS e KLMS para a identificação do sistema dado pelas Equações (13)–(15).

IV. Conclusões

Soluções distribuídas baseadas em algoritmos *kernel* para redes de difusão adaptativas são ferramentas interessantes para estimação não linear de sinais. O maior problema dessas soluções é o elevado custo computacional devido ao dicionário dos métodos *kernel*. Neste contexto, o algoritmo dGS-KLMS apresenta vantagens em relação às soluções clássicas, como o dKLMS. Com um dicionário reduzido, verificou-se por simulações que o dGS-KLMS apresenta um custo computacional menor, atingindo um nível de NMSE em regime permanente igual ou menor que o do dKLMS. Em trabalhos futuros, pretende-se considerar redes adaptativas heterogêneas, em que os passos de adaptação e a potências do ruído em cada nó são diferentes, situação em que a cooperação apresenta mais vantagens do que o caso não cooperativo.

Referências

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

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

[3] W. Liu, P. P. Pokharel, and J. C. Príncipe, "The kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, pp. 543–554, Feb. 2008.

[4] 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 Trans. Signal Process.*, vol. 60, pp. 2208–2222, May 2012.

[5] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, pp. 1058–1067, Mar. 2009.

[6] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. Signal Process.*, vol. 52, pp. 2275–2285, Aug. 2004.

[7] A. A. Bueno and M. T. M. Silva, "Gram-Schmidt-based sparsification for kernel dictionary," *IEEE Signal Process. Lett.*, vol. 27, pp. 1130–1134, 2020.

[8] A. H. Sayed, *Adaptation, Learning, and Optimization over Networks*, vol. 7, Foundations and Trends in Machine Learning, now Publishers Inc., Hanover, MA, 2014.

[9] W. Gao, J. Chen, C. Richard, and J. Huang, "Diffusion adaptation over networks with kernel least-mean-square," in *Proc. IEEE Int. Workshop Comput. Adv. Multi-Sensor Adaptive Process. (CAMSAP)*, 2015, pp. 217–220.

[10] S. Scardapane, J. Chen, and C. Richard, "Adaptation and learning over networks for nonlinear system modeling," in *Adaptive Learning Methods for Nonlinear System Modeling*, pp. 223–242. Elsevier, 2018.

[11] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Process.*, vol. 56, pp. 3122–3136, 2008.

[12] F. S. Cattivelli and A. H. Sayed, "Diffusion LMS strategies for distributed estimation," *IEEE Trans. Signal Process.*, vol. 58, pp. 1035–1048, 2009.

[13] W. Gao, "Gaussian kernel least-mean-square: design, analysis, and applications," *PhD. Thesis*, Université Nice-Sophia Antipolis, 2015.