Kernel Regression over Graphs using Random Fourier Features

Vitor R. M. Elias, Vinay Chakravarthi Gogineni, *Member, IEEE,* Wallace A. Martins, *Senior Member, IEEE* and Stefan Werner, *Senior Member, IEEE*

***Abstract*—This paper proposes efficient batch-based and online strategies for kernel regression over graphs (KRG). The proposed algorithms do not require the input signal to be a graph signal, whereas the target signal is defined over the graph. We first use random Fourier features (RFF) to tackle the complexity issues associated with kernel methods employed in the conventional KRG. For batch-based approaches, we also propose an implementation that reduces complexity by avoiding the inversion of large matrices. Then, we derive two distinct online strategies using RFF, namely, the mini-batch gradient KRG (MGKRG) and the recursive least squares KRG (RLSKRG). The stochastic-gradient KRG (SGKRG) is introduced as a particular case of the MGKRG. The MGKRG and the SGKRG are low-complexity algorithms that employ stochastic gradient approximations in the regression-parameter update. The RLSKRG is a recursive implementation of the RFF-based batch KRG. A detailed stability analysis is provided for the proposed online algorithms, including convergence conditions in both mean and mean-squared senses. A discussion on complexity is also provided. Numerical simulations include a synthesized-data experiment and real-data experiments on temperature prediction, brain activity estimation, and image reconstruction. Results show that the RFF-based batch implementation offers competitive performance with a reduced computational burden when compared to the conventional KRG. The MGKRG offers a convenient trade-off between performance and complexity by varying the number of mini-batch samples. The RLSKRG has a faster convergence than the MGKRG and matches the performance of the batch implementation.**

***Index Terms*—kernel regression on graphs, random Fourier features, stochastic gradient, recursive least squares**

I. Introduction

Graph signal processing (GSP) employs graph-structural information to model, process, and analyze signals defined over graph nodes [1]–[4]. The growing importance of GSP is due to its applicability to networked data processing, as connectivity between real-world elements progressively increases with the advent of the internet-of-things, sensor networks, and better communication technologies [5]–[7]. By associating real-world network elements with graph nodes and encoding their interrelations through graph edges, GSP leverages the graph structure to process or analyze the network data, modeled as a graph signal. Like conventional signal processing, the literature consists of various GSP techniques and approaches that address different needs associated with real-world networks.

Here, we are particularly interested in GSP approaches for modeling relations between a reference signal and a target signal, usually referred to as an input-output pair [8]–[18]. Typically, state-of-the-art techniques address the case where both reference and target signals share the same graph. In the context of linear system modeling, different learning problems have been studied within the GSP framework, e.g., classification on graphs [19], autoregressive models for graph signal prediction [15], [16], [20], dictionary learning [21], and distributed adaptive filtering [8]–[14]. Several learning strategies have been proposed for the nonlinear setting as well. In particular, kernel regression has been extensively employed for a range of nonlinear learning tasks, such as reconstruction [22]–[24] and prediction of graph signals [15].

In contrast to previous works, [15] proposes a batch-based kernel regression method that maps a general signal, not necessarily a graph signal, to an output signal that resides on a given graph. This means that the input signal can be agnostic to the graph or that the relation between the input signal and the graph structure is unavailable. A penalty term, added to the loss function, achieves this mapping and enforces the graph signal at the output, whose smoothness across the graph is defined by the graph Laplacian. The batch implementation in [15] requires that all samples are available before computing the solution, which induces a delay when dealing with streaming data. Moreover, obtaining the regression parameters through the batch-based KRG for large amounts of data may be computationally prohibitive. Finally, the approach in [15] inherits the well-known scaling issue of kernel methods [25], [26] since the model dimension increases with the number of training samples, which increases with the network size and with time. Other batch-based learning methods that take the graph structure into account, with different objective functions, are available in the literature, such as geometric deep learning [27], methods that link GSP and graph neural networks [28]–[30], and other GSP-based machine-learning methods [31], [32].

This work proposes an approach for kernel regression on graphs using random Fourier features (RFF) [33], [34],

This work was partly supported by: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) –- Grant number: 88887.310189/2018-00, CNPq, ERC project AGNOSTIC (Actively Enhanced Cognition based Framework for Design of Complex Systems – Grant number: 742648), FAPERJ, and the Research Council of Norway.

Vitor R. M. Elias, Vinay Chakravarthi Gogineni and Stefan Werner are with the Norwegian University of Science and Technology, Norway (e-mail: vitor.elias@ntnu.no, vinay.gogineni@ntnu.no and stefan.werner@ntnu.no).

Wallace A. Martins is with the University of Luxembourg, Luxembourg (e-mail: wallace.alvesmartins@uni.lu).

Vitor R. M. Elias and Wallace A. Martins are also with the Federal University of Rio de Janeiro, Brazil.

This article has been accepted for publication in a future issue of the IEEE Transactions on Signal Processing, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2022.3149134.

which enjoys a reduced model complexity compared to the batch-based KRG. Also, we derive and analyze two online strategies, namely, the mini-batch gradient KRG (MGKRG), with the particular case of the stochastic-gradient descent KRG (SGKRG), and the recursive least squares KRG (RLSKRG). The proposed RFF-based algorithms approximate the kernel evaluations by inner-products in a fixed-dimensional space, avoiding the model dimension dependency on the number of training samples encountered in the conventional KRG. Additionally, we propose an efficient implementation applicable to the conventional and RFF-based KRG that avoids large-scale matrix inversions. Similar to the approach in KRG [15], the proposed algorithms produce signals that vary smoothly over the graph, while input signals need not reside on a graph.

Among the proposed online algorithms, the stochastic gradient implementations, SGKRG and MGKRG, offer low-complexity alternatives. While the SGKRG requires the least computational effort, the MGKRG can improve the performance at a small additional cost by incorporating more samples in the stochastic gradient approximation. The RLSKRG, being the most complex, has faster convergence and higher accuracy than the other online implementations.

This paper is organized as follows. In Section II, we present some basic GSP concepts, formulate the problem of learning over graphs, and briefly describe the KRG methodology proposed in [15]. Section III presents the proposed methodology for batch-based KRG using RFF, along with an efficient implementation for large networks. The proposed online algorithms, namely the MGKRG, the RLSKRG, and its efficient implementation, are presented in Section IV. Section V provides a convergence analysis of the proposed online algorithms, and Section VI provides a brief discussion on the complexity of the algorithms. Numerical experiments to validate the performance of the proposed algorithms on both synthesized and real data are presented in Section VII. In the real-data experiments, we tackle the problems of predicting temperature on a weather-station network, estimating brain activity, and reconstructing corrupted video frames. Finally, concluding remarks for this work are presented in Section IX.

Mathematical notation: scalars are denoted by lowercase letters, column vectors by bold lowercase, and matrices by bold uppercase. Superscripts $(\cdot)^{\mathrm{T}}$ and $(\cdot)^{-1}$ denote the transpose and inverse operators, respectively. Given a matrix $\mathbf{A} = [\mathbf{a}_1\, \mathbf{a}_2\, \ldots\, \mathbf{a}_N]$, the column-stacking operation is denoted by $\mathrm{vec}(\mathbf{A}) = [\mathbf{a}_1^{\mathrm{T}}\, \mathbf{a}_2^{\mathrm{T}}\, \ldots\, \mathbf{a}_N^{\mathrm{T}}]^{\mathrm{T}}$ and the reverse operation that reshapes a column vector back to its appropriate matrix form is $\mathbf{A} = \mathrm{mat}(\mathrm{vec}(\mathbf{A}))$. The (i,j)th element of matrix $\mathbf{A}$ is denoted by $A_{i,j}$. Symbol $\otimes$ denotes the Kronecker product. $\mathbf{1}_N$ denotes the $N \times 1$ vector with all entries equal to unity and $\mathbf{I}_N$ denotes the $N \times N$ identity matrix. The $M \times N$ matrix with all entries equal to zero is denoted by $\mathbf{0}_{M\times N}$. $\|\cdot\|_2$ denotes the 2-norm of the argument vector or the spectral norm of the argument matrix. The Frobenius norm of the argument matrix is denoted by$\|\cdot\|_{\mathrm{F}}$. $\mathrm{E}[\cdot]$ denotes the expected value of the argument.

II. Background and Problem Formulation

Consider an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{1, 2, \ldots, K\}$ is the set of nodes and $\mathcal{E}$ is the set of edges such that $(k,l) \in \mathcal{E}$ if nodes k and l are connected. To each edge $(k,l) \in \mathcal{E}$, a weight $w_{k,l} \in \mathbb{R}_+$ can be assigned, which represents the strength of the relation between nodes k and l [1]–[3]. The set of edges is usually represented by the adjacency matrix $\mathbf{A} \in \mathbb{R}_+^{K\times K}$, such that the entry $A_{k,l} = A_{l,k} = w_{k,l}$ if $(k,l) \in \mathcal{E}$ and $A_{k,l} = 0$ otherwise. At time instant n, the graph signal is defined by a vector $\mathbf{y}_n = [y_{1,n}\, y_{2,n}\, \ldots\, y_{K,n}]^{\mathrm{T}}$, with $y_{k,n} \in \mathbb{R}$ being the signal value at node k.

Let $\mathcal{N}_k$ denote the neighborhood of node k, which is the set of nodes connected to k including itself. The graph Laplacian matrix is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix of $\mathcal{G}$, with $D_{k,k} = \sum_{l=1}^{K} w_{k,l}$. The graph Laplacian is associated with the total-variation metric $\nu(\mathbf{y})$ of a graph signal $\mathbf{y}$ as follows:[1]

$$\begin{aligned}\nu(\mathbf{y}) &= \mathbf{y}^{\mathrm{T}}\mathbf{L}\mathbf{y} \\ &= \sum_{k<l} A_{k,l}\big(y_k - y_l\big)^2. \end{aligned} \tag{1}$$

The metric (1) represents how much a signal varies across the graph, taking into account the edge weights [2], [15].

Considering a graph-based system, which takes an input vector $\mathbf{x} \in \mathbb{R}^M$ and outputs a graph signal $\mathbf{t} \in \mathbb{R}^K$, we are interested in estimating the corresponding mapping $\mathcal{M} : \mathbb{R}^M \to \mathbb{R}^K$. Given a set of training (available) data pairs $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^N$, regression methods can estimate $\mathcal{M}$. Regression methods that leverage the graph structure to improve the estimation are proposed in [15]. These methods were shown to outperform other approaches that do not use graph information.

A. Kernel Regression on Graphs

In [15], the model is estimated in terms of a matrix $\mathbf{W} \in \mathbb{R}^{M\times K}$ such that

$$\mathbf{y}_n = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \tag{2}$$

where $\mathbf{y}_n$ is an estimate of the target graph signal $\mathbf{t}_n$ and $\boldsymbol{\phi} : \mathbb{R}^M \to \mathbb{R}^M$ is an unknown function of the input signal. The optimal parameter matrix $\mathbf{W}$ is found by minimizing the cost function

$$C(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + \beta \sum_{n=1}^{N} \nu(\mathbf{y}_n), \tag{3}$$

where $N \geq M$. The cost function $C(\mathbf{W})$ augments traditional regression methods by incorporating the penalty $\sum_{n=1}^{N} \nu(\mathbf{y}_n)$, which enforces smoothness of the output signal with respect to the graph. Defining the matrices

$$\mathbf{T} = [\mathbf{t}_1\, \mathbf{t}_2\, \ldots\, \mathbf{t}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times K}, \tag{4}$$

$$\mathbf{Y} = [\mathbf{y}_1\, \mathbf{y}_2\, \ldots\, \mathbf{y}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times K}, \tag{5}$$

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1)\, \boldsymbol{\phi}(\mathbf{x}_2)\, \ldots\, \boldsymbol{\phi}(\mathbf{x}_N)]^{\mathrm{T}} \in \mathbb{R}^{N\times M}, \tag{6}$$

[1]The term "total variation" has been used to denote different smoothness metrics in the GSP literature [2], [35], [36]. We follow the notation from [36].

and assuming $\mathbf{\Phi}$ is full rank, we can make the substitution $\mathbf{W} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Psi}$, so that the optimization is now conducted in terms of $\mathbf{\Psi} \in \mathbb{R}^{N\times K}$. The predicted output of the kernel regression is given by [15]

$$\begin{aligned}\mathbf{y} &= \mathbf{\Psi}^{\mathrm{T}}\mathbf{\Phi}\boldsymbol{\phi}(\mathbf{x}) = \mathbf{\Psi}^{\mathrm{T}}\boldsymbol{\kappa}(\mathbf{x}) \\ &= \left(\mathrm{mat}\left((\mathbf{B}+\mathbf{C})^{-1}\mathrm{vec}(\mathbf{T})\right)\right)^{\mathrm{T}}\boldsymbol{\kappa}(\mathbf{x}),\end{aligned} \tag{7}$$

where $\boldsymbol{\kappa}(\mathbf{x}) = [\kappa_1(\mathbf{x})\ \kappa_2(\mathbf{x})\ \ldots\ \kappa_N(\mathbf{x})]^{\mathrm{T}}$, with $\kappa_n(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$. Also,

$$\mathbf{B} = (\mathbf{I}_K \otimes (\mathbf{K} + \alpha\mathbf{I}_N)), \tag{8}$$

$$\mathbf{C} = (\beta\mathbf{L}\otimes\mathbf{K}), \tag{9}$$

with $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} \in \mathbb{R}^{N\times N}$. Here, the kernel trick is employed to avoid the explicit knowledge of $\boldsymbol{\phi}(\cdot)$, by replacing the inner product $\kappa_n(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)$ with a kernel $\kappa(\mathbf{x}_i, \mathbf{x}_n)$ [38], [39]. The method described in (7), which outputs an estimate $\mathbf{y}$ for an input $\mathbf{x}$, is referred to as kernel regression on graphs.

The regression in (7) is performed in a batch-based fashion, assuming that all training samples are available *a priori*. A significant drawback of this implementation is the inherent delay of batch-based implementations, as the computation of the parameter matrix $\mathbf{\Psi}$ must wait for all training samples $\{\mathbf{x}_n, \mathbf{t}_n\}_{n=1}^N$ to be available. The increase in the computational burden of the KRG with the number of training samples is twofold. First, computing $\mathbf{\Psi}$ becomes more complex as the dimensions of $\mathbf{K}$ increase with N. Second, the regression dimension increases as the size of $\boldsymbol{\kappa}(\mathbf{x})$ increases with N, and each additional training sample requires a kernel evaluation. The model complexity also depends on the number of training samples N, requiring N kernel evaluations for each new input signal, which is an issue if an online implementation is derived. In the following section, we treat the growing complexity by proposing a batch-based approach using random Fourier features.

III. Batch KRG Using Random Fourier Features

Random Fourier features is a widely used technique to circumvent the scaling problems of kernel methods [33]. This methodology presumes that the evaluation of a shift-invariant kernel, which satisfies $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \kappa(\mathbf{x}_m - \mathbf{x}_n, 0)$, can be approximated as an inner product in the D-dimensional RFF space. This turns the problem into a finite-dimension linear filtering problem while avoiding the evaluation of kernel functions. Let $\mathbf{z}_n$ be the mapping of $\mathbf{x}_n$ into the RFF space $\mathbb{R}^D$, given by

$$\mathbf{z}_n = (D/2)^{-\frac{1}{2}}\left[\cos(\mathbf{v}_1^{\mathrm{T}}\mathbf{x}_n + b_1)\ \ldots\ \cos(\mathbf{v}_D^{\mathrm{T}}\mathbf{x}_n + b_D)\right]^{\mathrm{T}}, \tag{10}$$

where the phase terms $\{b_i\}_{i=1}^D$ are drawn from a uniform distribution on the interval $[0,\ 2\pi]$. Vectors $\{\mathbf{v}_i\}_{i=1}^D$ are realizations of a random variable with probability density function (pdf) $p(\mathbf{v})$ such that

$$\kappa(\mathbf{x}_m, \mathbf{x}_n) = \int p(\mathbf{v})\exp\left(\mathrm{j}\mathbf{v}^{\mathrm{T}}(\mathbf{x}_m - \mathbf{x}_n)\right)\mathrm{d}\mathbf{v}, \tag{11}$$

where $\mathrm{j}^2 = -1$. In other words, the Fourier transform of $\kappa(\mathbf{x}_m, \mathbf{x}_n)$ is given by $p(\mathbf{v})$. From (10) and (11), it can be verified that $\mathrm{E}[\mathbf{z}_n^{\mathrm{T}}\mathbf{z}_m] = \kappa(\mathbf{x}_m, \mathbf{x}_n)$ [33].

A. RFF-based KRG

To employ RFF in the KRG methodology, we first consider the kth entry of the estimate $\mathbf{y}$ as

$$y_k = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \tag{12}$$

where $\mathbf{w}_k$ denotes the kth column of the parameter matrix $\mathbf{W}$. Using the substitution $\mathbf{W} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Psi}$, and the kernel trick $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \boldsymbol{\phi}(\mathbf{x}_m)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)$, (12) can be rewritten as

$$y_k = \left(\sum_{n=1}^N \Psi_{n,k}\boldsymbol{\phi}(\mathbf{x}_n)\right)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) = \left(\sum_{n=1}^N \Psi_{n,k}\kappa(\mathbf{x}_n, \mathbf{x})\right). \tag{13}$$

Using RFF, we can approximate (13) as

$$y_k \approx \sum_{n=1}^N \Psi_{n,k}\mathbf{z}_n^{\mathrm{T}}\mathbf{z} = \mathbf{h}_k^{\mathrm{T}}\mathbf{z}. \tag{14}$$

Finally, the RFF-based regression estimate for the entire graph signal is written as

$$\mathbf{y} = \mathbf{H}^{\mathrm{T}}\mathbf{z}, \tag{15}$$

where $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \ldots\ \mathbf{h}_K] \in \mathbb{R}^{D\times K}$ is the representation of the regression coefficient matrix in the RFF space. Letting the matrix

$$\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \ldots\ \mathbf{z}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times D} \tag{16}$$

represent the RFF mapping of all training input vectors $\{\mathbf{x}_n\}_{n=1}^N$, and using $\mathbf{T}$ and $\mathbf{Y}$ as respectively defined in (4) and (5), the cost function (3) can be rewritten in terms of $\mathbf{H}$ as

$$\begin{aligned}C(\mathbf{H}) = \sum_{n=1}^N \|\mathbf{t}_n\|_2^2 - 2\mathrm{tr}(\mathbf{T}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) + \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) \\ + \alpha(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \beta\mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}).\end{aligned} \tag{17}$$

The gradient of $C(\mathbf{H})$ with respect to $\mathbf{H}$ is given by

$$\nabla C(\mathbf{H}) = -2\mathbf{Z}^{\mathrm{T}}\mathbf{T} + 2\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H} + 2\alpha\mathbf{H} + 2\beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}. \tag{18}$$

Setting $\nabla C(\mathbf{H}) = \mathbf{0}$, we obtain

$$(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)\mathbf{H}_{\mathrm{opt}} + \beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}_{\mathrm{opt}}\mathbf{L} = \mathbf{Z}^{\mathrm{T}}\mathbf{T}. \tag{19}$$

Then, vectorizing both sides of (19) and using the relation $\mathrm{vec}(\mathbf{AXB}) = (\mathbf{B}^{\mathrm{T}}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X})$, the regression coefficients in the RFF space can be obtained as

$$\mathrm{vec}(\mathbf{H}_{\mathrm{opt}}) = (\mathbf{B}_{\mathrm{RFF}} + \mathbf{C}_{\mathrm{RFF}})^{-1}\mathrm{vec}(\mathbf{Z}^{\mathrm{T}}\mathbf{T}), \tag{20}$$

where

$$\mathbf{B}_{\mathrm{RFF}} = (\mathbf{I}_K \otimes (\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)), \tag{21}$$

$$\mathbf{C}_{\mathrm{RFF}} = (\beta\mathbf{L}\otimes\mathbf{Z}^{\mathrm{T}}\mathbf{Z}). \tag{22}$$

Once the regression coefficients are trained, the target estimate $\mathbf{y}$ given an input signal $\mathbf{x}$ corresponding to $\mathbf{z}$ in the RFF space is given by

$$\mathbf{y} = \mathbf{H}_{\mathrm{opt}}^{\mathrm{T}}\mathbf{z}. \tag{23}$$

From (21) and (22), it can be observed that the computational burden of obtaining the regression parameters is

drastically reduced when compared to the conventional KRG, as the size of the $\mathbf{B}_{\mathrm{RFF}}$ and $\mathbf{C}_{\mathrm{RFF}}$ is now $KD \times KD$, with D possibly much smaller than N. From (23), we see that the estimation does not depend on the number of training samples and the model has a fixed size D, requiring only the mapping of each new input sample into the RFF space.

B. Efficient Computation For Large Networks

For large networks, computing the inverses in (7) and (20) may be prohibitively complex. We propose an efficient way to compute the parameters in these cases. We adopt the notation of the conventional KRG, but the same reasoning applies directly to the RFF-based implementation. We rewrite $(\mathbf{B}+\mathbf{C})^{-1}$ as

$$\begin{aligned}(\mathbf{B}+\mathbf{C})^{-1} &= (\mathbf{I}_K \otimes (\mathbf{K}+\alpha\mathbf{I}_N) + \beta\mathbf{L}\otimes\mathbf{K})^{-1} \\ &= (\mathbf{I}_K\otimes\alpha\mathbf{I}_N + (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{K})^{-1} \\ &= (\alpha\mathbf{I}_{KN} + (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{K})^{-1}. \end{aligned} \quad (24)$$

Consider the eigendecompositions $(\mathbf{I}_K+\beta\mathbf{L}) = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$ and $\mathbf{K} = \mathbf{V}\mathbf{\Omega}\mathbf{V}^{\mathrm{T}}$. We use the mixed-product property $(\mathbf{AB})\otimes(\mathbf{CD}) = (\mathbf{A}\otimes\mathbf{C})(\mathbf{B}\otimes\mathbf{D})$ to rewrite the Kronecker product. Note also that matrices $\alpha\mathbf{I}_{KN}$ and $(\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{K}$ are simultaneously diagonalizable. Then, (24) can be written as

$$(\mathbf{B}+\mathbf{C})^{-1} = (\mathbf{U}\otimes\mathbf{V})(\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}(\mathbf{U}^{\mathrm{T}}\otimes\mathbf{V}^{\mathrm{T}}), \quad (25)$$

and

$$\begin{aligned}&(\mathbf{B}+\mathbf{C})^{-1}\mathrm{vec}(\mathbf{T}) \\ &\quad= (\mathbf{U}\otimes\mathbf{V})(\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}(\mathbf{U}^{\mathrm{T}}\otimes\mathbf{V}^{\mathrm{T}})\mathrm{vec}(\mathbf{T}) \\ &\quad= (\mathbf{U}\otimes\mathbf{V})(\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}\mathrm{vec}(\mathbf{V}^{\mathrm{T}}\mathbf{T}\mathbf{U}). \end{aligned} \quad (26)$$

Letting

$$\mathbf{\Gamma} = \mathrm{mat}\left((\alpha\mathbf{I}_{KN}+\mathbf{\Sigma}\otimes\mathbf{\Omega})^{-1}\mathrm{vec}(\mathbf{V}^{\mathrm{T}}\mathbf{T}\mathbf{U})\right) \quad (27)$$

and using the relation $(\mathbf{B}^{\mathrm{T}}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{AXB})$, we have

$$\mathbf{\Psi} = \mathbf{V}\mathbf{\Gamma}\mathbf{U}^{\mathrm{T}}. \quad (28)$$

Note that (27) requires the trivial inversion of a diagonal matrix. Hence, the dominating complexity is reduced from $(KN)^3$ operations due to matrix inversion to approximately K^3 and N^3 operations required for the eigendecompositions of $(\mathbf{I}_K+\beta\mathbf{L})$ and $\mathbf{K}$, respectively.

IV. Online Kernel Regression on Graphs

In what follows, we consider online implementations of the KRG. To bypass the dimensionality problem associated with the kernel dictionary, we resort to online RFF-based KRG implementations.

Algorithm 1: MGKRG

Initialization:
$\mathbf{H}_1 = \mathbf{0}_{D\times K}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
Learning:
for each time instant n **do**
  map $\mathbf{x}_n$ into $\mathbf{z}_n$;
  if $(n \bmod \delta) = 0$ **then**
    $\mathbf{Z}_n = [\mathbf{z}_{(n-N_\mathrm{b}+1)} \ \cdots \ \mathbf{z}_n]^{\mathrm{T}}$;
    $\mathbf{T}_n = [\mathbf{t}_{(n-N_\mathrm{b}+1)} \ \cdots \ \mathbf{t}_n]^{\mathrm{T}}$;
    $\mathbf{Y}_n = \mathbf{Z}_n\mathbf{H}_n$;
    $\mathbf{E}_n = \mathbf{T}_n - \mathbf{Y}_n$;
    $\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^{\mathrm{T}}(\mathbf{E}_n - \beta\mathbf{Y}_n\mathbf{L})$;
  end
  store $\mathbf{z}_n$;
  release $\mathbf{z}_{(n-N_\mathrm{b}+1)}$;
end

A. Mini-batch Stochastic-Gradient KRG

Consider the following minimization problem:

$$\min_{\mathbf{H}} \mathrm{E}\left[\|\mathbf{t}-\mathbf{y}\|_2^2\right] + \alpha\mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \mathrm{E}[\beta\nu(\mathbf{y})]. \quad (29)$$

Similar to the batch-based approach, which conducts the optimization over N batch samples, problem (29) considers the expectation of the regularized regression problem. The gradient of the cost function in (29) is

$$\nabla C(\mathbf{H}) = -2\mathbf{R}_{zt} + 2\mathbf{R}_z\mathbf{H} + 2\alpha\mathbf{H} + 2\beta\mathbf{R}_z\mathbf{H}\mathbf{L}, \quad (30)$$

where $\mathbf{R}_{zt} = \mathrm{E}[\mathbf{z}\mathbf{t}^{\mathrm{T}}]$ and $\mathbf{R}_z = \mathrm{E}[\mathbf{z}\mathbf{z}^{\mathrm{T}}]$. In practice, the statistics $\mathbf{R}_{zt}$ and $\mathbf{R}_z$ can be unavailable.

In the proposed approach, we use mini-batch averages to approximate $\mathbf{R}_{zt}$ and $\mathbf{R}_z$. We define the matrices composed by the signals corresponding to each individual mini-batch as

$$\mathbf{Z}_n = [\mathbf{z}_{(n\delta-N_\mathrm{b}+1)} \ \cdots \ \mathbf{z}_{n\delta}]^{\mathrm{T}} \in \mathbb{R}^{N_\mathrm{b}\times D}$$

and

$$\mathbf{T}_n = [\mathbf{t}_{(n\delta-N_\mathrm{b}+1)} \ \cdots \ \mathbf{t}_{n\delta}]^{\mathrm{T}} \in \mathbb{R}^{N_\mathrm{b}\times K},$$

where $1 \le \delta \le N_\mathrm{b}$ is the batch displacement parameter. For the nth batch, we can compute the approximations $\hat{\mathbf{R}}_{zt,n} = (\mathbf{Z}_n^{\mathrm{T}}\mathbf{T}_n)/N_\mathrm{b}$ and $\hat{\mathbf{R}}_{z,n} = (\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n)/N_\mathrm{b}$. We implement the sliding-window MGKRG, with $\delta = 1$, such that consecutive mini-batches have maximum overlap of $N_\mathrm{b}-1$ samples. A particular case of the MGKRG is defined by making $N_\mathrm{b} = \delta = 1$. In this case, only the current sample is used to compute the approximation of the gradient. This corresponds to a stochastic-gradient approach and will be referred to as stochastic gradient KRG (SGKRG).

The regression parameters are updated at the nth batch by taking a step in the negative direction of the corresponding approximate gradient, i.e.,

$$\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^{\mathrm{T}}(\mathbf{T}_n - \mathbf{Z}_n\mathbf{H}_n - \beta\mathbf{Z}_n\mathbf{H}_n\mathbf{L}), \quad (31)$$

where $\mu > 0$ is the gradient-descent step size. Letting $\mathbf{Y}_n = \mathbf{Z}_n\mathbf{H}_n$ be the mini-batch estimate and $\mathbf{E}_n = \mathbf{T}_n - \mathbf{Y}_n$ be the corresponding *a priori* error matrix, the update equation for the mini-batch gradient KRG is written as

$$\mathbf{H}_{n+1} = (1-\mu\alpha)\mathbf{H}_n + \frac{\mu}{N_\mathrm{b}}\mathbf{Z}_n^\mathrm{T}\left(\mathbf{E}_n - \beta\mathbf{Y}_n\mathbf{L}\right). \tag{32}$$

B. Recursive Least-Squares KRG

We now explore the principles of the recursive least squares algorithms [40] to derive a recursive learning of the regression coefficients of the RFFKRG. That is, we part from the same optimization problem (17) but, instead of solving (20) directly, we solve it recursively. First, we rewrite (20) as

$$\begin{aligned}\mathrm{vec}(\mathbf{H}_n) &= \left((\mathbf{I}_K \otimes (\mathbf{Z}^\mathrm{T}\mathbf{Z} + \alpha\mathbf{I}_D)) + (\beta\mathbf{L}\otimes\mathbf{Z}^\mathrm{T}\mathbf{Z})\right)^{-1}\mathrm{vec}(\mathbf{Z}^\mathrm{T}\mathbf{T}) \\ &= \mathbf{R}_n^{-1}\mathbf{r}_n,\end{aligned} \tag{33}$$

where

$$\mathbf{R}_n = \alpha\mathbf{I}_K\otimes\mathbf{I}_D + (\mathbf{I}_K + \beta\mathbf{L})\otimes\mathbf{Z}^\mathrm{T}\mathbf{Z} \tag{34}$$

$$\mathbf{r}_n = \mathrm{vec}(\mathbf{Z}^\mathrm{T}\mathbf{T}). \tag{35}$$

Note that these terms are obtained at time n, i.e., once n training samples are available. We aim to write both $\mathbf{R}_n^{-1}$ and $\mathbf{r}_n$ in terms of $\mathbf{R}_{n-1}^{-1}$ and $\mathbf{r}_{n-1}$, respectively, to derive a recursive algorithm. First, we rewrite (34) as

$$\mathbf{R}_n = \alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L})\otimes\sum_{i=0}^{n}\mathbf{z}_i\mathbf{z}_i^\mathrm{T} \tag{36}$$

$$\begin{aligned}&= \alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L})\otimes\sum_{i=0}^{n-1}\mathbf{z}_i\mathbf{z}_i^\mathrm{T} + (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{z}_n\mathbf{z}_n^\mathrm{T} \\ &= \mathbf{R}_{n-1} + (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{z}_n\mathbf{z}_n^\mathrm{T}.\end{aligned} \tag{37}$$

We rewrite the second term on the right-hand side (RHS) of (37) using the mixed-product property and the fact that the resulting matrix is symmetric, as

$$\begin{aligned}(\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{z}_n\mathbf{z}_n^\mathrm{T} &= ((\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{z}_n)(\mathbf{I}_K\otimes\mathbf{z}_n^\mathrm{T}) \\ &= (\mathbf{I}_K\otimes\mathbf{z}_n)((\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{z}_n^\mathrm{T})\end{aligned}$$

Now, letting $\mathbf{P}_n = (\mathbf{I}_K\otimes\mathbf{z}_n)$ and $\mathbf{Q}_n = ((\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{z}_n^\mathrm{T})$, we can use the matrix inversion lemma to derive a recursive equation for $\mathbf{R}_n^{-1}$ as

$$\mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - \underbrace{\mathbf{R}_{n-1}^{-1}\mathbf{P}_n\left(\mathbf{I}_K + \mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\mathbf{P}_n\right)^{-1}}_{\mathbf{G}_n\in\mathbb{R}^{KD\times K}}\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}. \tag{38}$$

where the gain matrix $\mathbf{G}_n$ may be simplified as follows:

$$\mathbf{G}_n = \left(\mathbf{R}_{n-1}^{-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\right)\mathbf{P}_n = \mathbf{R}_n^{-1}\mathbf{P}_n. \tag{39}$$

We now write (35) in a recursive manner as

$$\mathbf{r}_n = \mathrm{vec}\left(\sum_{i=0}^{n}\mathbf{z}_i\mathbf{t}_i^\mathrm{T}\right) = \mathbf{r}_{n-1} + \mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^\mathrm{T}). \tag{40}$$

Substituting (40) into (33), we obtain

$$\mathrm{vec}(\mathbf{H}_n) = \mathbf{R}_n^{-1}\mathbf{r}_{n-1} + \mathbf{R}_n^{-1}\mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^\mathrm{T}). \tag{41}$$

Algorithm 2: RFF-based RLSKRG

Initialization:
$\mathbf{R}_{-1}^{-1} = \frac{1}{\alpha}\mathbf{I}_{KD}$;
$\mathbf{H}_{-1} = \mathbf{0}_{D\times K}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^D$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^D$ from $[0, 2\pi]$;
Learning:
for each time instant *n* **do**
  map $\mathbf{x}_n$ into $\mathbf{z}_n$;
  $\mathbf{P}_n = \mathbf{I}_K\otimes\mathbf{z}_n$;
  $\mathbf{Q}_n = (\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{z}_n^\mathrm{T}$;
  $\mathbf{G}_n = \mathbf{R}_{n-1}^{-1}\mathbf{P}_n\left(\mathbf{I}_K + \mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\mathbf{P}_n\right)^{-1}$;
  $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^\mathrm{T}\mathbf{z}_n$;
  $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$;
  $\mathbf{H}_n = \mathbf{H}_{n-1} + \mathrm{mat}(\mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n))$;
  $\mathbf{R}_n^{-1} = \mathbf{R}_{n-1}^{-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}$;
end

Using the relation $\mathrm{vec}(\mathbf{AXB}) = (\mathbf{B}^\mathrm{T}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X})$ and the mixed-product property, $\mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^\mathrm{T})$ can be written as

$$\mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^\mathrm{T}) = \mathbf{t}_n\otimes\mathbf{z}_n = (\mathbf{I}_K\otimes\mathbf{z}_n)\mathbf{t}_n.$$

and (41) becomes

$$\begin{aligned}\mathrm{vec}(\mathbf{H}_n) &= \mathbf{R}_n^{-1}\mathbf{r}_{n-1} + \mathbf{R}_n^{-1}(\mathbf{I}_K\otimes\mathbf{z}_n)\mathbf{t}_n \\ &= \mathbf{R}_n^{-1}\mathbf{r}_{n-1} + \mathbf{G}_n\mathbf{t}_n\end{aligned} \tag{42}$$

Substituting (38) into (42)

$$\begin{aligned}\mathrm{vec}(\mathbf{H}_n) &= \mathbf{R}_{n-1}^{-1}\mathbf{r}_{n-1} - \mathbf{G}_n\mathbf{Q}_n\mathbf{R}_{n-1}^{-1}\mathbf{r}_{n-1} + \mathbf{G}_n\mathbf{t}_n \\ &= \mathrm{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n(\mathbf{t}_n - \mathbf{Q}_n\mathrm{vec}(\mathbf{H}_{n-1})) \\ &= \mathrm{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n\Big(\mathbf{t}_n - (\mathbf{I}_K\otimes\mathbf{z}_n^\mathrm{T})\mathrm{vec}(\mathbf{H}_{n-1}) \\ &\qquad - (\beta\mathbf{L}\otimes\mathbf{z}_n^\mathrm{T})\mathrm{vec}(\mathbf{H}_{n-1})\Big) \\ &= \mathrm{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n(\mathbf{t}_n - \mathbf{H}_{n-1}^\mathrm{T}\mathbf{z}_n - \beta\mathbf{L}\mathbf{H}_{n-1}^\mathrm{T}\mathbf{z}_n) \\ &= \mathrm{vec}(\mathbf{H}_{n-1}) + \mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n),\end{aligned} \tag{43}$$

or, equivalently,

$$\mathbf{H}_n = \mathbf{H}_{n-1} + \mathrm{mat}(\mathbf{G}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n)), \tag{44}$$

where $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^\mathrm{T}\mathbf{z}_n$ is the *a priori* target estimate and $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$ is the *a priori* error. Equation (43) is the recursive update equation for the proposed recursive least squares KRG (RLSKRG) algorithm. The steps for the implementation of the RLSKRG algorithm are summarized in **Algorithm 2**.

Due to its recursive nature, the RLSKRG algorithm considers past samples when computing the update matrix at each iteration. Thus, its performance is expected to match that of the batch-based approach.

C. Efficient RLSKRG Implementation

The complexity associated with large matrix multiplications or inversions can render the RLSKRG impractical for large networks. For instance, the computations of $\mathbf{G}_n$ and $\mathbf{R}_n^{-1}$ in **Algorithm 2** require multiplications of matrices with dimension $KD\times KD$ and $KD\times K$. This implies at least K^3D^2

Algorithm 3: Efficient RLSKRG

Initialization:
$\mathbf{R}_{z,-1} = \mathbf{0}_{D\times D}$;
$\mathbf{H}_{-1} = \mathbf{0}_{D\times K}$;
get $\mathbf{U}$ and $\mathbf{\Sigma}$;
draw vectors $\{\mathbf{v}_i\}_{i=1}^{D}$ from $p(\mathbf{v})$;
draw phase terms $\{b_i\}_{i=1}^{D}$ from $[0, 2\pi]$;
Learning:
for each time instant n **do**
  map $\mathbf{x}_n$ into $\mathbf{z}_n$;
  $\mathbf{R}_{z,n} = \mathbf{R}_{z,n-1} + \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}$;
  Get $\mathbf{V}_n$ and $\mathbf{\Omega}_n$;
  $\mathbf{P}_n = \mathbf{I}_K \otimes \mathbf{z}_n$;
  $\hat{\mathbf{y}}_n = \mathbf{H}_{n-1}^{\mathrm{T}}\mathbf{z}_n$;
  $\mathbf{e}_n = \mathbf{t}_n - \hat{\mathbf{y}}_n$;
  $\mathbf{\Xi} = \mathrm{mat}(\mathbf{P}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n))$;
  $\mathbf{\Gamma}_n = \mathrm{mat}((\alpha\mathbf{I}_{KD} + \mathbf{\Sigma}\otimes\mathbf{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\mathbf{\Xi}_n\mathbf{U}))$;
  $\mathbf{H}_n = \mathbf{H}_{n-1} + \mathbf{V}_n\mathbf{\Gamma}_n\mathbf{U}^{\mathrm{T}}$;
end

multiplication operations for each computation. We now derive an alternative implementation with reduced complexity.

Substituting (36) into (39), and substituting the result into (44), we obtain

$$\begin{aligned}\mathbf{H}_n &= \mathbf{H}_{n-1} \\ &+ \mathrm{mat}\left(\left(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L})\otimes\mathbf{R}_{z,n}\right)^{-1}\boldsymbol{\xi}_n\right),\end{aligned} \tag{45}$$

where $\mathbf{R}_{z,n} = \sum_{i=0}^{n}\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}$ and $\boldsymbol{\xi}_n = \mathbf{P}_n(\mathbf{e}_n - \beta\mathbf{L}\hat{\mathbf{y}}_n)$. We now use the eigendecompositions $(\mathbf{I}_K + \beta\mathbf{L}) = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}$ and $\mathbf{R}_{z,n} = \mathbf{V}_n\mathbf{\Omega}_n\mathbf{V}_n^{\mathrm{T}}$. Using the mixed-product property of the Kronecker product, and considering that $\alpha\mathbf{I}_{KD}$ and $(\mathbf{I}_K + \beta\mathbf{L})\otimes\mathbf{R}_{z,n}$ share the same set of eigenvectors, (45) can be rewritten as

$$\begin{aligned}\mathbf{H}_n &= \mathbf{H}_{n-1} \\ &+ \mathrm{mat}\left((\mathbf{U}\otimes\mathbf{V}_n)(\alpha\mathbf{I}_{KD} + \mathbf{\Sigma}\otimes\mathbf{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\mathbf{\Xi}_n\mathbf{U})\right),\end{aligned} \tag{46}$$

where $\mathbf{\Xi}_n = \mathrm{mat}(\boldsymbol{\xi}_n)$. Letting $\mathbf{\Gamma}_n = \mathrm{mat}((\alpha\mathbf{I}_{KD} + \mathbf{\Sigma}\otimes\mathbf{\Omega}_n)^{-1}\mathrm{vec}(\mathbf{V}_n^{\mathrm{T}}\mathbf{\Xi}_n\mathbf{U}))$, and using the relation $(\mathbf{B}^{\mathrm{T}}\otimes\mathbf{A})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{AXB})$, the update equation for the efficient RLSKRG algorithm is given by

$$\mathbf{H}_n = \mathbf{H}_{n-1} + \mathbf{V}_n\mathbf{\Gamma}_n\mathbf{U}^{\mathrm{T}}. \tag{47}$$

All steps for the implementation of the efficient RLSKRG are presented in **Algorithm 3**.

V. Convergence Analysis

This section examines the convergence of the proposed online algorithms; in particular, we study their first- and second-order stability conditions. In the following analysis, $\mathbf{H}_{\mathrm{o}}$ denotes the optimal linear estimator in the least mean squares sense of $\mathbf{T}_n$ in the RFF domain. In this case, $\mathbf{T}_n = \mathbf{Z}_n\mathbf{H}_{\mathrm{o}} + \mathbf{\Upsilon}_n$, where $\mathbf{\Upsilon}_n = [\boldsymbol{\upsilon}_{(n\delta-N_{\mathrm{b}}+1)}\ \ldots\ \boldsymbol{\upsilon}_{n\delta}]^{\mathrm{T}} \in \mathbb{R}^{N_{\mathrm{b}}\times K}$ denotes the corresponding optimum-error matrix, which satisfies the orthogonality condition $\mathrm{E}[\mathbf{Z}_n^{\mathrm{T}}\mathbf{\Upsilon}_n] = \mathbf{0}_{D\times K} \Leftrightarrow \mathrm{E}[(\mathbf{I}_K\otimes\mathbf{Z}_n^{\mathrm{T}})\mathrm{vec}(\mathbf{\Upsilon}_n)] = \mathbf{0}_{KD\times 1}$ [40], [41].

For the derivations that follow, let $\lambda_{\max}(\cdot)$ denote the maximum eigenvalue of the argument matrix and let $\rho(\cdot)$ denote the spectral radius of the argument matrix, i.e., the largest absolute value of its eigenvalues. Additionally, we use the following property of the Kronecker product: let the eigenvalues of a matrix $\mathbf{A}$ be $\{\lambda_1, \lambda_2, \ldots, \lambda_M\}$ and of a matrix $\mathbf{B}$ be $\{\sigma_1, \sigma_2, \ldots, \sigma_N\}$. Then, the eigenvalues of $\mathbf{A}\otimes\mathbf{B}$ and $\mathbf{B}\otimes\mathbf{A}$ are given by $\{\lambda_i\sigma_j\}_{i=1,j=1}^{M,N}$ [41].

A. First-Order Analysis of the MGKRG

Making the substitution in (31) and subtracting both sides from $\mathbf{H}_{\mathrm{o}}$ yields

$$\begin{aligned}\widetilde{\mathbf{H}}_{n+1} = \widetilde{\mathbf{H}}_n - \mu\alpha\widetilde{\mathbf{H}}_n - \frac{\mu}{N_{\mathrm{b}}}\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n\widetilde{\mathbf{H}}_n - \frac{\mu}{N_{\mathrm{b}}}\beta\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n\widetilde{\mathbf{H}}_n\mathbf{L} \\ + \frac{\mu}{N_{\mathrm{b}}}\beta\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n\mathbf{H}_{\mathrm{o}}\mathbf{L} + \mu\alpha\mathbf{H}_{\mathrm{o}} - \frac{\mu}{N_{\mathrm{b}}}\mathbf{Z}_n^{\mathrm{T}}\mathbf{\Upsilon}_n,\end{aligned} \tag{48}$$

where $\widetilde{\mathbf{H}}_n = \mathbf{H}_{\mathrm{o}} - \mathbf{H}_n$ is the parameter-deviation matrix. Defining $\tilde{\mathbf{h}}_n = \mathrm{vec}(\widetilde{\mathbf{H}}_n)$, $\mathbf{h}_{\mathrm{o}} = \mathrm{vec}(\mathbf{H}_{\mathrm{o}})$, and $\boldsymbol{\gamma}_n = \mathrm{vec}(\mathbf{\Upsilon}_n)$, the above recursion can be alternatively expressed as

$$\begin{aligned}\tilde{\mathbf{h}}_{n+1} = \left(\mathbf{I}_{KD} - \mu\left(\alpha\mathbf{I}_{KD} + \frac{1}{N_{\mathrm{b}}}(\mathbf{I}_K + \beta\mathbf{L})\otimes(\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n)\right)\right)\tilde{\mathbf{h}}_n \\ + \mu\left(\alpha\mathbf{I}_{KD} + \frac{\beta}{N_{\mathrm{b}}}\mathbf{L}\otimes(\mathbf{Z}_n^{\mathrm{T}}\mathbf{Z}_n)\right)\mathbf{h}_{\mathrm{o}} \\ - \frac{\mu}{N_{\mathrm{b}}}(\mathbf{I}_K\otimes\mathbf{Z}_n^{\mathrm{T}})\boldsymbol{\gamma}_n.\end{aligned} \tag{49}$$

To study the convergence behavior of the proposed MGKRG governed by the form (49), we make the following assumptions:

- **A1**: The RFF-mapped data signal $\mathbf{z}_n$ is drawn from a wide-sense stationary multivariate random sequence with correlation matrix $\mathbf{R}_z = \mathrm{E}[\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}]$.
- **A2**: For n large enough, the contribution of the batch $\mathbf{Z}_n$ to $\mathbf{H}_n$ is negligible, such that $\mathbf{H}_n$ is considered to be independent of $\mathbf{Z}_n$.
- **A3**: The graph topology is assumed to be static, meaning that the graph Laplacian $\mathbf{L}$ is fixed throughout the process.

Theorem 1. A sufficient condition on the step size μ for the convergence of the proposed MGKRG algorithm governed by (32), is given by

$$0 < \mu < \frac{2}{\lambda_{\max}(\mathbf{R}_z) + \alpha + \beta\lambda_{\max}(\mathbf{L})\lambda_{\max}(\mathbf{R}_z)}. \tag{50}$$

Proof. Taking the expectation $\mathrm{E}[\cdot]$ on both sides of (49), using **A1**-**A2**, and using the orthogonality condition such that the error-related term can be set to zero, we obtain

$$\mathrm{E}[\tilde{\mathbf{h}}_{n+1}] = \boldsymbol{\mathcal{A}}\mathrm{E}[\tilde{\mathbf{h}}_n] + \boldsymbol{\mathcal{B}}\mathbf{h}_{\mathrm{o}}, \tag{51}$$

where

$$\begin{aligned}\boldsymbol{\mathcal{A}} &= \mathbf{I}_{KD} - \mu\left(\alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L})\otimes\mathbf{R}_z\right) \\ \boldsymbol{\mathcal{B}} &= \mu\left(\alpha\mathbf{I}_{KD} + \beta\mathbf{L}\otimes\mathbf{R}_z\right).\end{aligned} \tag{52}$$

Iterating the above recursion back down to zero, we obtain

$$\mathrm{E}[\tilde{\mathbf{h}}_n] = \boldsymbol{\mathcal{A}}^n\mathrm{E}[\tilde{\mathbf{h}}_0] + \sum_{j=0}^{n-1}\boldsymbol{\mathcal{A}}^{n-1-j}\boldsymbol{\mathcal{B}}\mathbf{h}_{\mathrm{o}}. \tag{53}$$

Therefore, we see that convergence is guaranteed if $\rho(\boldsymbol{\mathcal{A}}) < 1$. We note that a scalar matrix $a\mathbf{I}$, with $a \in \mathbb{R}$, is simultaneously diagonalizable with any arbitrary matrix with adequate dimensions. Using the properties of the Kronecker product, and recalling that the eigenvalues of $\mathbf{L}$ and $\mathbf{R}_z$ are non-negative, the above condition reduces to $0 < \mu(\alpha + (1 + \beta\lambda_{\max}(\mathbf{L}))\lambda_{\max}(\mathbf{R}_z)) < 2$. The result in (50) follows from here. □

Remark 1. Under the convergence condition (50), (53) converges asymptotically to $(\mathbf{I}_{KD} - \boldsymbol{\mathcal{A}})^{-1}\boldsymbol{\mathcal{B}}\mathbf{h}_{\mathrm{o}}$, which reduces to $(\alpha\mathbf{I}_{KD}+(\mathbf{I}_K+\beta\mathbf{L})\otimes\mathbf{R}_z)^{-1}(\alpha\mathbf{I}_{KD}+\beta\mathbf{L}\otimes\mathbf{R}_z)\mathbf{h}_{\mathrm{o}}$. This means that $\lim_{n\to\infty}\mathbf{H}_n$ is a biased estimate of $\mathbf{H}_{\mathrm{o}}$. Also, the bias is introduced by the regularization coefficients α and β, such that a non-regularized problem leads to an unbiased solution.

B. Second-Order Analysis of the MGKRG

For the second-order analysis of the MGKRG, we consider the following additional assumption:

A4: The step size μ is sufficiently small so that the terms involving higher order powers of μ can be ignored.

Using **A1**-**A4**, the covariance matrix of the parameter deviation vector $\tilde{\mathbf{h}}_{n+1}$ is given by

$$\begin{aligned}\mathrm{E}[\tilde{\mathbf{h}}_{n+1}\tilde{\mathbf{h}}_{n+1}^{\mathrm{T}}] = \mathrm{E}[\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\mathrm{T}}] - \mu\mathrm{E}[\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\mathrm{T}}]\boldsymbol{\mathcal{C}} - \mu\boldsymbol{\mathcal{C}}\mathrm{E}[\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\mathrm{T}}] \\ - \mu\mathrm{E}[\tilde{\mathbf{h}}_n\mathbf{h}_{\mathrm{o}}^{\mathrm{T}}]\boldsymbol{\mathcal{D}} - \mu\boldsymbol{\mathcal{D}}\mathrm{E}[\mathbf{h}_{\mathrm{o}}\tilde{\mathbf{h}}_n^{\mathrm{T}}],\end{aligned} \tag{54}$$

where

$$\begin{aligned}\boldsymbol{\mathcal{C}} &= \alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes \mathbf{R}_z \\ \boldsymbol{\mathcal{D}} &= \alpha\mathbf{I}_{KD} + \beta\mathbf{L} \otimes \mathbf{R}_z.\end{aligned} \tag{55}$$

The cross terms involving $\frac{\mu}{N_{\mathrm{b}}}(\mathbf{I}_K \otimes \mathbf{Z}_n^{\mathrm{T}})\boldsymbol{\gamma}_n$ are zero due to the orthogonality condition. By vectorizing both sides of (54) and defining $\boldsymbol{\eta}_n = \mathrm{vec}(\tilde{\mathbf{h}}_n\tilde{\mathbf{h}}_n^{\mathrm{T}})$, we can now write

$$\mathrm{E}[\boldsymbol{\eta}_{n+1}] = \boldsymbol{\Delta}\mathrm{E}[\boldsymbol{\eta}_n] + \boldsymbol{\Theta}_n, \tag{56}$$

where

$$\boldsymbol{\Delta} = \mathbf{I}_{K^2D^2} - \mu(\boldsymbol{\mathcal{C}} \otimes \mathbf{I}_{KD}) - \mu(\mathbf{I}_{KD} \otimes \boldsymbol{\mathcal{C}}) \tag{57}$$

and

$$\begin{aligned}&\boldsymbol{\Theta}_n = \\ &- \mu(\boldsymbol{\mathcal{D}} \otimes \mathbf{I}_{KD})\mathrm{vec}(\mathrm{E}[\tilde{\mathbf{h}}_n]\mathbf{h}_{\mathrm{o}}^{\mathrm{T}}) - \mu(\mathbf{I}_{KD} \otimes \boldsymbol{\mathcal{D}})\mathrm{vec}(\mathbf{h}_{\mathrm{o}}\mathrm{E}[\tilde{\mathbf{h}}_n]^{\mathrm{T}}).\end{aligned} \tag{58}$$

Theorem 2. Assume **A1**-**A4** hold. Then, the second-order convergence of the proposed gradient-based algorithms, namely the MGKRG and the SGKRG, is guaranteed under

$$0 < \mu < \frac{1}{\lambda_{\max}(\mathbf{R}_z) + \alpha + \beta\lambda_{\max}(\mathbf{L})\lambda_{\max}(\mathbf{R}_z)}. \tag{59}$$

Proof. Iterating the recursion (56) back down to zero, we obtain

$$\mathrm{E}[\boldsymbol{\eta}_n] = \boldsymbol{\Delta}^n\mathrm{E}[\boldsymbol{\eta}_o] + \sum_{j=0}^{n-1}\boldsymbol{\Delta}^{n-1-j}\boldsymbol{\Theta}_j. \tag{60}$$

Recalling that $\mathrm{E}[\tilde{\mathbf{h}}_n]$ is finite under (50), so $\boldsymbol{\Theta}_n$ converges asymptotically with n. Therefore, equation (60) is stable iff $\rho(\boldsymbol{\Delta}) < 1$. Since matrices $\boldsymbol{\mathcal{C}} \otimes \mathbf{I}_{KD}$ and $\mathbf{I}_{KD} \otimes \boldsymbol{\mathcal{C}}$ commute and are both diagonalizable, the eigenvalues of their sum equal the sum of their eigenvalues. Moreover, these matrices share the same eigenvalues under the properties of the Kronecker product. Then, the condition for $\rho(\boldsymbol{\Delta}) < 1$ reduces to

$$\rho(\mathbf{I}_{K^2D^2} - 2\mu(\boldsymbol{\mathcal{C}} \otimes \mathbf{I}_{KD})) < 1, \tag{61}$$

which can be written as $|1 - 2\mu\lambda_{\max}(\boldsymbol{\mathcal{C}})| < 1$. Substituting $\boldsymbol{\mathcal{C}}$ as in (55), the second-order convergence condition reduces to

$$0 < 2\mu\big(\alpha + (1 + \beta\lambda_{\max}(\mathbf{L}))\lambda_{\max}(\mathbf{R}_z)\big) < 2, \tag{62}$$

from which (59) follows. □

Theorem 2 shows that the condition for second-order stability of the MGKRG is more strict than that of the first-order stability. The upper-bound imposed on the step-sizes for second-order stability is half of the upper-bound established in Theorem 1.

C. First-Order Analysis of the RLSKRG

In the analysis of the RLSKRG, the following additional assumption is considered:

A5: The random sequence that governs signals $\mathbf{z}_n$ is ergodic. Then, for sufficiently large n, $\mathbf{R}_n$ behaves as a deterministic matrix given by $\mathbf{R}_n = \alpha\mathbf{I}_{KD} + (\mathbf{I}_K + \beta\mathbf{L}) \otimes (n+1)\mathbf{R}_z$.

Assumption **A5** is commonly employed in the analysis of RLS-based algorithms [42]. It considers that, given ergodicity, the time average of rank-one covariance matrices $\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}$ can be replaced by the expected value for large enough n.

Multiplying both sides of (33) from the left by $\mathbf{R}_n$, and using (40) in conjunction with (33) we can write

$$\begin{aligned}&\mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) = \mathbf{r}_n \\ &\mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) = \mathbf{r}_{n-1} + \mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^{\mathrm{T}}) \\ &\mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) = \mathbf{R}_{n-1}\mathrm{vec}(\mathbf{H}_{n-1}) + \mathrm{vec}(\mathbf{z}_n\mathbf{t}_n^{\mathrm{T}})\end{aligned} \tag{63}$$

Substituting the model $\mathbf{t}_n = \mathbf{H}_{\mathrm{o}}^{\mathrm{T}}\mathbf{z}_n + \boldsymbol{\upsilon}_n$ into (63), we have

$$\mathbf{R}_n\mathrm{vec}(\mathbf{H}_n) = \mathbf{R}_{n-1}\mathrm{vec}(\mathbf{H}_{n-1}) + \mathrm{vec}(\mathbf{z}_n\boldsymbol{\upsilon}_n^{\mathrm{T}}) + \mathrm{vec}(\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_{\mathrm{o}}) \tag{64}$$

We now subtract both sides from $\mathbf{R}_n\mathrm{vec}(\mathbf{H}_{\mathrm{o}})$. By recalling that $\tilde{\mathbf{h}}_n = \mathrm{vec}(\mathbf{H}_{\mathrm{o}} - \mathbf{H}_n)$, we obtain

$$\begin{aligned}\mathbf{R}_n\tilde{\mathbf{h}}_n =& \mathbf{R}_n\mathrm{vec}(\mathbf{H}_{\mathrm{o}}) - \mathbf{R}_{n-1}\mathrm{vec}(\mathbf{H}_{n-1}) \\ &- \mathrm{vec}(\mathbf{z}_n\boldsymbol{\upsilon}_n^{\mathrm{T}}) - \mathrm{vec}(\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}\mathbf{H}_{\mathrm{o}}).\end{aligned} \tag{65}$$

Substituting (37) into the first term on the RHS, we rewrite (65) as

$$\mathbf{R}_n\tilde{\mathbf{h}}_n = \mathbf{R}_{n-1}\tilde{\mathbf{h}}_{n-1} - \mathrm{vec}(\mathbf{z}_n\boldsymbol{\upsilon}_n^{\mathrm{T}}) + (\beta\mathbf{L} \otimes \mathbf{z}_n\mathbf{z}_n^{\mathrm{T}})\mathrm{vec}(\mathbf{H}_{\mathrm{o}}). \tag{66}$$

Taking the recursion down to $n = 0$ and solving for $\tilde{\mathbf{h}}_n$, we obtain

$$\tilde{\mathbf{h}}_n = \mathbf{R}_n^{-1}\mathbf{R}_0\tilde{\mathbf{h}}_0 + \mathbf{R}_n^{-1}\boldsymbol{\pi}_n, \tag{67}$$

where

$$\boldsymbol{\pi}_n = \sum_{i=0}^{n}\mathrm{vec}(\beta\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}\mathbf{H}_{\mathrm{o}}\mathbf{L} - \mathbf{z}_i\boldsymbol{\upsilon}_i^{\mathrm{T}}). \tag{68}$$

Using $\mathrm{vec}(\mathbf{AXB}) = (\mathbf{B}^{\mathrm{T}} \otimes \mathbf{A})\mathrm{vec}(\mathbf{X})$ and **A5**, we get

$$\boldsymbol{\pi}_n = (\beta\mathbf{L} \otimes (n+1)\mathbf{R}_z)\,\mathbf{h}_{\mathrm{o}} - \sum_{i=0}^{n} \mathrm{vec}(\mathbf{z}_i \boldsymbol{\upsilon}_i^{\mathrm{T}}). \tag{69}$$

Theorem 3. The RLSKRG described in **Algorithm 2** is stable in the mean sense and converges to a steady state.

Proof. The expected value of the parameter deviation in (67) is given by

$$\mathrm{E}[\tilde{\mathbf{h}}_n] = \mathrm{E}[\mathbf{R}_n^{-1}\mathbf{R}_0\tilde{\mathbf{h}}_0] + \mathrm{E}[\mathbf{R}_n^{-1}\boldsymbol{\pi}_n]. \tag{70}$$

For sufficiently large n, we can apply **A5** so that $\mathbf{R}_n^{-1}$ can be regarded as a deterministic matrix for which $\lim_{n\to\infty}\mathbf{R}_n^{-1} = \mathbf{0}_{KD\times KD}$, since $\mathbf{R}_n$ is dominated by the term $(\mathbf{I}_K + \beta\mathbf{L}) \otimes (n+1)\mathbf{R}_z$. Thus, the first term on the RHS of (70) tends to zero. As for the second term, under the same conditions we have that

$$\begin{aligned}\mathrm{E}[\mathbf{R}_n^{-1}\boldsymbol{\pi}_n] =& \mathbf{R}_n^{-1}\,(\beta\mathbf{L} \otimes (n+1)\mathbf{R}_z)\,\mathbf{h}_{\mathrm{o}} \\ &- \mathbf{R}_n^{-1}\sum_{i=0}^{n} \underbrace{\mathrm{vec}\left(\mathrm{E}[\mathbf{z}_i\boldsymbol{\upsilon}_i^{\mathrm{T}}]\right)}_{=\mathbf{0}_{KD\times 1}},\end{aligned} \tag{71}$$

where the second term on the RHS is zero due to the orthogonality condition. Regarding the first term, as $\mathbf{R}_n$ is dominated by the term $(\mathbf{I}_K + \beta\mathbf{L}) \otimes (n+1)\mathbf{R}_z$, then we can write $\lim_{n\to\infty}\mathbf{R}_n^{-1}\,(\beta\mathbf{L} \otimes (n+1)\mathbf{R}_z)\,\mathbf{h}_{\mathrm{o}} = \boldsymbol{\mathcal{E}}\mathbf{h}_{\mathrm{o}}$, in which

$$\begin{aligned}\boldsymbol{\mathcal{E}} &= \lim_{n\to\infty}\left[(\mathbf{I}_K + \beta\mathbf{L}) \otimes (n+1)\mathbf{R}_z\right]^{-1}(\beta\mathbf{L} \otimes (n+1)\mathbf{R}_z) \\ &= \left[(\mathbf{I}_K + \beta\mathbf{L})^{-1}\beta\mathbf{L}\right] \otimes \mathbf{I}_D\,,\end{aligned} \tag{72}$$

where we used the Kronecker product property $(\mathbf{A}\otimes\mathbf{B})^{-1} = (\mathbf{A}^{-1} \otimes \mathbf{B}^{-1})$ and the mixed-product property. Hence, we have that $\mathbf{H}_n$ is an asymptotically biased estimate of $\mathbf{H}_{\mathrm{o}}$, and by using the relation $(\mathbf{B}^{\mathrm{T}} \otimes \mathbf{A})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{AXB})$, we can rewrite the bias term $\boldsymbol{\mathcal{E}}\mathbf{h}_{\mathrm{o}}$ as follows: $\lim_{n\to\infty}\mathrm{E}[\widetilde{\mathbf{H}}_n] = \beta\mathbf{H}_{\mathrm{o}}\mathbf{L}(\mathbf{I}_K + \beta\mathbf{L})^{-1}$. □

Remark 2. Under the convergence condition (50), the bias of the MGRKG tends to the bias of the RLSKRG when $\alpha \to 0^{+}$. In addition, the bias in the RLSKRG is introduced solely by the regularization coefficient β, since the regularization coefficient α contributes only with an initial condition for the matrix $\mathbf{R}_n$, which plays no role in the algorithm's average behavior as n grows to infinity.

D. Second-Order Analysis of the RLSKRG

For the second order analysis, we assume further that

A6: Variables $\mathbf{z}_n$ and $\mathbf{t}_n$ are jointly ergodic, so that, for sufficiently large n, $\sum_{i=0}^{n}\mathbf{z}_i\mathbf{t}_i^{\mathrm{T}} \approx (n+1)\mathrm{E}[\mathbf{z}_i\mathbf{t}_i^{\mathrm{T}}]$.

Assumptions **A5** and **A6** imply that, for sufficiently large n, matrix $\sum_{i=0}^{n}\mathbf{z}_i(\mathbf{t}_i^{\mathrm{T}} - \mathbf{z}_i^{\mathrm{T}}\mathbf{H}_{\mathrm{o}})$ can be approximated as $(n+1)\mathrm{E}[\mathbf{z}_i\boldsymbol{\upsilon}_i^{\mathrm{T}}]$, which is equal to $\mathbf{0}_{K\times D}$ due to the orthogonality condition.

Theorem 4. The RLSKRG described in **Algorithm 2** is stable in the mean-squared sense and converges to a steady state.

Proof. From (67), we have

$$\begin{aligned}\mathrm{E}[\|\tilde{\mathbf{h}}_n\|_2^2] =& \mathrm{E}[\|\mathbf{R}_n^{-1}\mathbf{R}_0\tilde{\mathbf{h}}_0\|_2^2] \\ &+ 2\mathrm{E}[\tilde{\mathbf{h}}_0^{\mathrm{T}}\mathbf{R}_0\mathbf{R}_n^{-2}\boldsymbol{\pi}_n] + \mathrm{E}[\|\mathbf{R}_n^{-1}\boldsymbol{\pi}_n\|_2^2].\end{aligned} \tag{73}$$

For sufficiently large n, we can apply **A5** so that the first non-negative term on the RHS of (73) is upper bounded by $\|\mathbf{R}^{-1}\|_2^2 \cdot \mathrm{E}[\|\mathbf{R}_0\tilde{\mathbf{h}}_0\|_2^2]$, which tends to zero since $\mathrm{E}[\|\mathbf{R}_0\tilde{\mathbf{h}}_0\|_2^2]$ is bounded and $\lim_{n\to\infty}\mathbf{R}_n^{-1} = \mathbf{0}_{KD\times KD}$. Under **A5** and **A6**, we can write $\mathbf{R}_n^{-1}\boldsymbol{\pi}_n = \mathrm{vec}\left(\beta\mathbf{H}_{\mathrm{o}}\mathbf{L}(\mathbf{I}_K + \beta\mathbf{L})^{-1}\right)$ for sufficiently large n. This implies both that the middle term on the RHS of (73) can be written as $2\mathrm{E}[\tilde{\mathbf{h}}_0^{\mathrm{T}}\mathbf{R}_0]\mathbf{R}_n^{-1}\mathrm{vec}\left(\beta\mathbf{H}_{\mathrm{o}}\mathbf{L}(\mathbf{I}_K + \beta\mathbf{L})^{-1}\right)$, which tends to zero as n grows to infinity, and that the last term on the RHS of (73) is finite. Therefore, the RLSKRG converges in the mean-squared sense to $\lim_{n\to\infty}\mathrm{E}[\|\tilde{\mathbf{h}}_n\|_2^2] = \|\mathrm{vec}\left(\beta\mathbf{H}_{\mathrm{o}}\mathbf{L}(\mathbf{I}_K + \beta\mathbf{L})^{-1}\right)\|_2^2$. □

VI. Discussion on Complexity

For the MGKRG algorithm, the update (32) requires $DK + N_{\mathrm{b}}(K^2 + 2DK + K)$ multiplication operations. That is, the complexity of the MGKRG increases linearly with N_b with a slope equal to $K^2 + 2DK + K$. Additionally, the MGKRG requires a memory to store $N_{\mathrm{b}} > 1$ samples. Hence, the batch-size translates into a trade-off between complexity and performance since the gradient approximation using more samples yields a better update direction than those using a reduced number of samples. In this sense, the SGKRG yields the lowest computational burden of the proposed online KRG implementations.

The proposed efficient implementation of the RLSKRG in (47) requires $D^3 + D^2 + 2D^2K + 5DK + 2DK^2 + K^2$ multiplication operations to update $\mathbf{H}_n$. The terms D^2 and D^3 correspond to the complexity of updating the matrix $\mathbf{R}_{z,n}$ and computing its eigendecomposition, respectively. Since $\mathbf{R}_{z,n}$ is only updated with $\mathbf{z}_n\mathbf{z}_n^{\mathrm{T}}$ at time n, and we only need its eigensystem, the complexity can be reduced using efficient techniques for rank-one updates of the singular value decomposition [43]. Other techniques for reducing the complexity of the RLSKRG can be considered. For instance, dichotomous-coordinate descent (DCD) iterations, which uses only additions and bit-shifts with no multiplications, have been considered for reduced-complexity RLS implementations [44]. Under a reasonable assumption that N_{b} has the same order of magnitude as D and K, we observe that the RLSKRG has a slightly heavier computational burden per iteration when compared to the MGKRG.

The efficient implementation (28) of the offline batch KRG using RFF requires $D^3 + D^2N + 2D^2K + 3DK + 2DK^2 + KDN$ multiplications. We highlight that this complexity is considerably smaller than that of the conventional implementation (20), which requires the inversion of a $DK \times DK$ matrix, leading to complexity equivalent to D^3K^3 multiplications for the inversion operation only. Moreover, we note that the computations of $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ and $\mathbf{Z}^{\mathrm{T}}\mathbf{T}$ depend on N and yield the terms D^2N and KND, respectively. This implies that the complexity of the offline RFF-based KRG is not constant with time. The batch-based KRG can be considered in an online

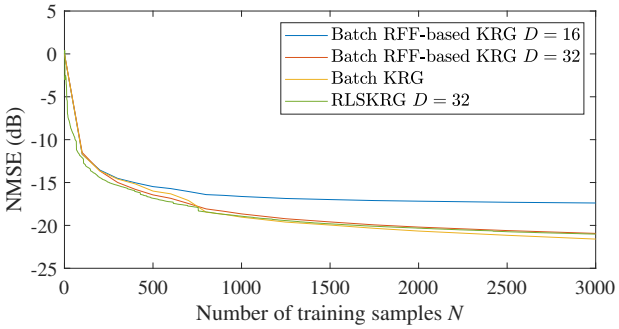


Fig. 1. NMSE achieved by the Batch-based and RLSKRG implementations versus number of training samples using synthesized data.

fashion, such that matrices $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ and $\mathbf{Z}^{\mathrm{T}}\mathbf{T}$ are stored and only rank-one updates are required at each time instant, reducing the complexity of these terms to D^2 and KD multiplications per iteration, respectively.

The complexity of the proposed algorithms is further discussed in Section VII-A, where we evaluate the time taken for the algorithms to learn the regression coefficients.

VII. Numerical Results

In this section, we validate the proposed methodology and the theoretical results with numerical experiments using both synthesized and real datasets. To assess the performance of the proposed algorithms in terms of learning accuracy, we evaluate the normalized mean squared error

$$\mathrm{NMSE} = 10\log_{10}\left(\mathrm{E}\left[\frac{\|\mathbf{Y}-\mathbf{T}_0\|_{\mathrm{F}}^2}{\|\mathbf{T}_0\|_{\mathrm{F}}^2}\right]\right), \tag{74}$$

where $\mathbf{T}$ denotes the true target matrix and $\mathbf{Y}$ denotes the estimated matrix. The NMSE is also used for the online algorithms, instead of the commonly used learning error, to allow comparison with the batch-based algorithms. We compare the results against the conventional KRG proposed in [15] using different hyperparameters, such as the dimension D of the RFF space, the step size μ for online algorithms, and the number N_{b} of samples in the mini-batches. In the experiments, we use the Gaussian kernel $\kappa(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i-\mathbf{x}_j\|_2^2/(2\sigma^2)\right)$, with σ^2 obtained via grid search.

A. Synthesized Data

Similar to the setup in [15], we consider an Erdös Rényi graph with $K = 50$ nodes and edge-probability equal to 0.1. A total of $S = 20000$ K-dimensional i.i.d. samples, $\{\mathbf{x}_n\}_{n=1}^{S}$, are generated, where $\mathbf{x}_n \sim \mathcal{N}(\mathbf{0},\mathbf{C}_S)$. The S-dimensional covariance matrix $\mathbf{C}_S \in \mathbb{R}^{S\times S}$ is drawn from the inverse Wishart distribution with an identity scale matrix. We generate the target graph signals $\{\mathbf{t}_n\}_{n=1}^{S}$ as in [15], i.e., by solving $\mathbf{t}_n = \arg\min_{\boldsymbol{\tau}}\left\{\|\mathbf{x}_n-\boldsymbol{\tau}\|_2^2 + \boldsymbol{\tau}^{\mathrm{T}}\mathbf{L}\boldsymbol{\tau}\right\}$. The generated signals are divided into a training set and a test set, containing N_{ts} and N samples, respectively, with $N_{\mathrm{ts}}+N \leq S$. The target signals in the training dataset are perturbed by white Gaussian noise (AWGN). The SNR is fixed across all nodes, with noise variance on the kth node $\sigma_{\mathrm{n},k}^2 = \frac{\sigma_{\mathrm{s},k}^2}{\sqrt{10}}$, where $\sigma_{\mathrm{s},k}^2$ denotes the signal variance on the kth node. In our simulations, we fix $N_{\mathrm{ts}} = 1000$ and let N vary. Finally, α and β were obtained from the training set, via grid search and 5-fold cross-validation, by minimizing the NMSE.

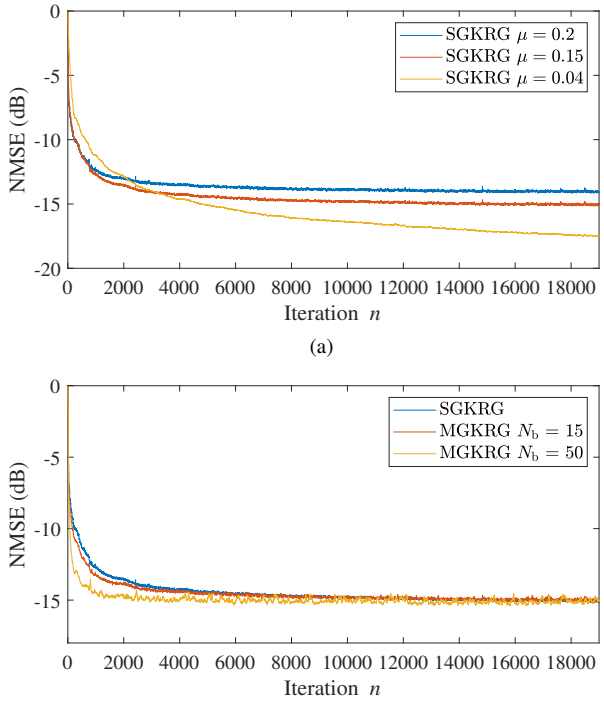


Fig. 2. NMSE achieved by the MGKRG implementations versus number of training samples for different step sizes and mini-batch sizes.

We evaluate the NMSE over the entire test dataset for the proposed online algorithms at each iteration n. That is, for every n, we obtain $\mathbf{H}_n$, calculate the estimates of all N_{ts} test signals, and we compute the NMSE using (74). The expected value is obtained as the ensemble average over 500 independent runs.

Fig. 1 presents the results of the batch-based implementations and the RLSKRG. We see that the RFF implementation approximates well the conventional KRG even for relatively small $D = 32$. The performance of the RLSKRG closely matches the performance of the batch-based implementation. Results in Fig. 2a show that online algorithms can effectively learn the regression parameters. We analyze different step sizes and we show that the NMSE level achieved by the SGKRG approximates that of the batch RFF-based KRG as μ decreases. Fig. 2b shows the performance of the MGKRG for different mini-batch sizes. Plots show an increase in convergence speed as N_{b} increases to 15 and then to 50 samples.

B. Runtime and Complexity Analysis

Here we evaluate the time that the proposed algorithms, along with their different implementations, take to learn the regression parameters, starting from the mapping of the input signals into the RFF space. We use the same simulation setup as in Section VII-A, with ensemble averages over 50 inde-

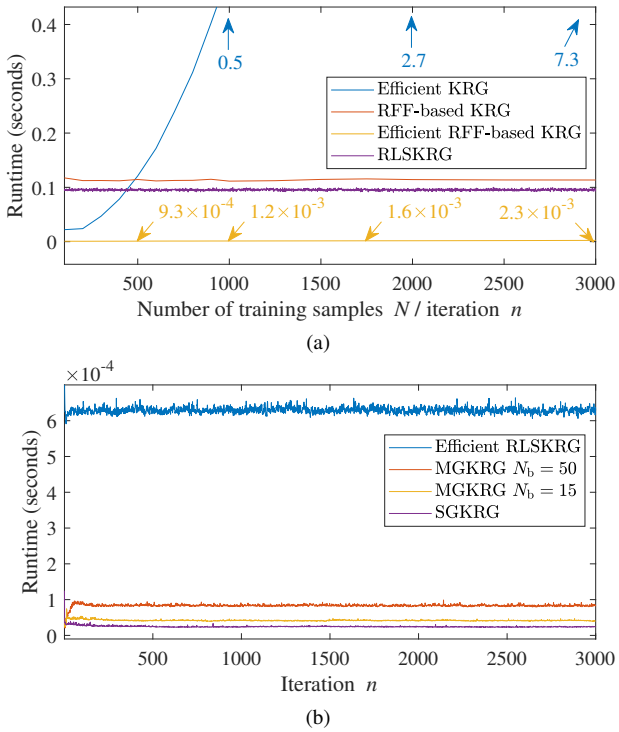


Fig. 3. Runtime of the algorithms versus number of training samples (for batch-based algorithms) or iteration (for online algorithms).

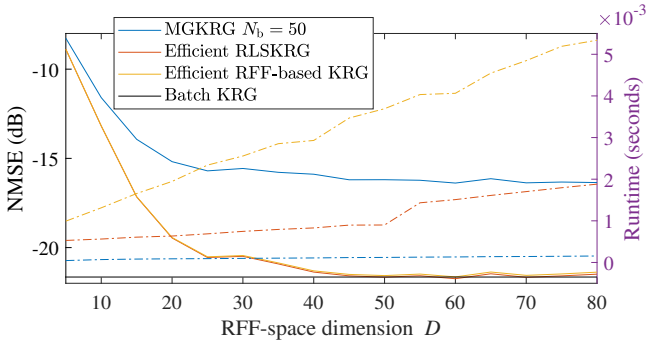


Fig. 4. Trade-off between NMSE (solid lines) and runtime (dash-dot lines) versus the RFF-space dimension D.

pendent runs. The algorithms are implemented in MATLAB® running on an i7-6700HQ CPU @2.60 GHz with 16 GB RAM DDR4 @2133 MHz.

Fig. 3a shows the results for the efficient implementation of the conventional KRG, the RFF-based KRG and its efficient implementation, and the RLSKRG from **Algorithm 2**. We do not show the results for the conventional implementation of the KRG since the runtimes escalate too quickly to be represented in the plots. For example, for 100, 200, and 300 training samples, the KRG takes approximately 3 seconds, 23 seconds, and 82 seconds, respectively. These values become considerably larger than the runtimes depicted in Fig. 3. Simulation results show that the efficient implementation considerably reduces runtimes. The dependence on N of the conventional KRG can be observed in Fig. 3a, illustrating the well-known complexity issue of kernel methods, whereas the runtimes of the RFF-based KRG are mostly stable with N. As discussed in Section VI, we can observe a relatively slight increase in the runtime of the efficient RFF-based KRG as the number of samples increases, even though the RFF-based model does not depend on N anymore. This is due to the computations of $\mathbf{Z}^{\mathrm{T}}\mathbf{Z}$ and $\mathbf{Z}^{\mathrm{T}}\mathbf{T}$ in an offline fashion. In the conventional RFF-based implementation, this effect is masked by the increased complexity of the inversion operation.

Fig. 3b shows the results for the online implementations only. We can observe that the runtime per iteration is mostly constant for all online algorithms. As presented in Section VI, the RLSKRG in its efficient implementation is computationally heavier than the stochastic-gradient approaches. The efficient RLSKRG runs considerably faster than its non-efficient counterpart. It can also be observed that increasing N_{b}, the number of samples used to compute the gradient approximations, increases the computational burden.

Fig. 4 shows the trade-off between complexity and accuracy of the proposed algorithms as a function of the dimension of the RFF-space, D, at $N = 3000$ samples. For the online algorithms, we show the average runtime per iteration, which is expected to be constant with N. The value of D controls how well the RFF approximate the kernel function [33]. Thus, theoretically, increasing D leads to accuracy closer to that achieved by the conventional KRG, while increasing the computational complexity. In this particular simulation, approximately $D > 30$ is enough to approximate the kernel function with considerable accuracy and we observe that the runtime increases in a linear fashion.

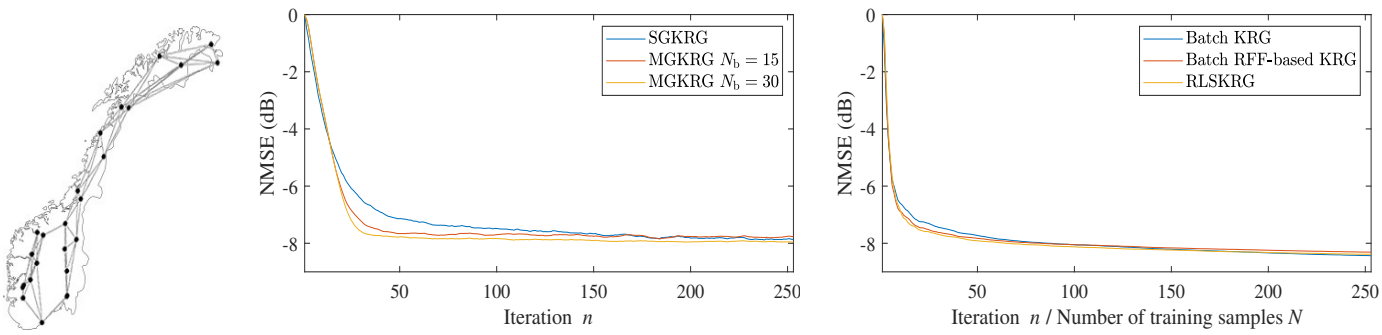
C. Real Data - Temperature Prediction

In this experiment, we use temperature data from 30 weather stations distributed across Norway's mainland, collected by the Norwegian Meteorological Institute [45]. Data from 2019 are used for the final experiment, while data from 2018 are used for training hyperparameters σ, α, and β.

We construct a nearest-neighbor graph with $K = 30$ nodes using the GSPBOX toolbox for MATLAB, such that each station is connected to its five nearest neighbors. The latitude and longitude coordinates of the stations are available in [45] and are used for computing the distance between stations. Fig. 5a shows the graph and the approximate positions of the weather stations.

We employ the KRG for a 4-days ahead temperature prediction on the 2019 data, with a 70% and 30% split between training and test data, respectively. For the RFF-based implementations, we use $D = 128$. The results are obtained as an ensemble average over 100 independent experiments, with different permutations of the data to generate the corresponding training and test datasets.

Results are presented in Fig. 5. In this example, we observe that the SG-based approaches, shown in Fig. 5b, achieve performance similar to the performance achieved by the batch-based approaches and the RLSKRG, in Fig. 5c. Comparing the SG implementations among themselves, we control the step size to achieve a similar steady-state NMSE. We observe that increasing the mini-batch size increases convergence speed. Plots in Fig. 5c show that the RLSKRG closely matches the



(a) (b) (c)

Fig. 5. Setup and results for temperature-prediction simulation: (a) illustration of the map of Norway and the approximate position of the stations, as represented by the graph used in the simulations; (b) results for SGKRG algorithms; and (c) results for batch-based algorithms and RLSKRG.

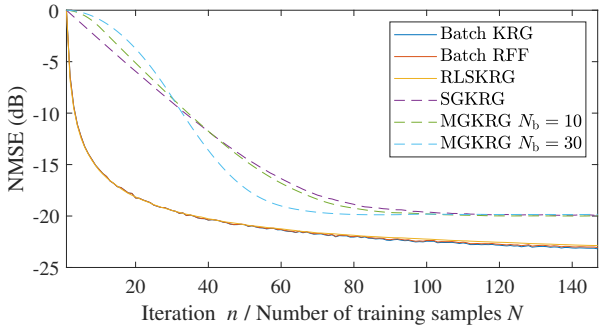


Fig. 6. NMSE achieved by the KRG implementations versus number of training samples for the fMRI signal simulation.

batch-based KRG using RFF. Also, for $D = 128$ used in this example, the RFF offer an approximation that allows the RFF-based KRG to match the conventional KRG in performance.

D. Real Data - fMRI Signal Extrapolation

This section reproduces the example from [15], which employs the conventional KRG to estimate the intensities of voxels in a functional magnetic resonance imaging (fMRI) dataset. The data and graph used are available in [46].

In the fMRI context, a voxel is a volumetric unit that constitutes a 3-dimensional image of the brain, analogous to pixels in 2-dimensional digital images. Each voxel is associated with a small cubic portion of the brain. The fMRI measures the changes in blood flow on each of these voxels. The blood flow is, in turn, associated with brain activity and, thus, by collecting measurements on all voxels, one can obtain a mapping of the brain activity. Regions of the brain relate to each other anatomically and, by considering these relations, a graph can be constructed where voxels are the nodes, and edges represent relations between them. For more details on this dataset and graph construction, see [15].

The regression experiment consists of estimating the signal on 90 of the voxels using the signal from other 10 voxels. In other words, we consider an input signal $\mathbf{x} \in \mathbb{R}^{10}$ to estimate a graph signal $\mathbf{t} \in \mathbb{R}^{90}$. The graph corresponds to the pairwise relations of the 90 voxels. A total of 292 snapshots is available. We consider training and test datasets of same size equal to 146 input-target pairs. The training signals are corrupted by an AWGN with covariance matrix $0.1 \cdot \mathbf{I}_K$. RFF-based implementations use $D = 32$.

Fig. 6 shows the results for all algorithms. We can observe that $D = 32$ is enough for the RFF-based KRG to closely match the conventional KRG, converging to approximately -23 dB. Again, the RLSKRG mostly coincides with the batch-based implementations. Results also show that the SG-based implementations can achieve low NMSE, around -20 dB, while increasing the number of samples when computing the stochastic approximation for the gradient increases the convergence speed.

E. Real Data - Image Reconstruction

We now consider the application of KRG in the image and video processing scenario. This simulation showcases the performance and the capability of the online algorithms to deal with large datasets. In particular, we tackle the reconstruction of a corrupted video frame. Each frame is divided into blocks of 4×4 pixels. Each block of pixels is represented as a graph with $K = 16$ nodes, where each node corresponds to a pixel, and nodes are connected to their nearest neighbors inside a fixed radius equal to the minimum distance between two pixels. Frames are black and white, and pixels are treated in double format, such that a zero corresponds to black and a one corresponds to white. In this setup, corrupted frames have up to one random pixel per block that is set to unity, simulating a saturated pixel. An example of a corrupted frame, along with a block of 4×4 pixels with one corrupted pixel, and the corresponding graph are illustrated in Fig. 7a.

The video recording used in this simulation corresponds to a sequence of objects being captured against a generic wooden background as the camera pans from left to right, moving along the objects, at 30 frames per second. The video frames have a resolution of 480×480 pixels, which results in 14400 non-overlapping blocks per frame. We utilize six full frames taken with a distance of 50 frames between them to train the regression parameters, and two frames are used as the test dataset. We highlight that, in a real application, six frames correspond to approximately 0.2 seconds of video. Consistent

(a) (b)

Fig. 7. Example of the image reconstruction process using KRG: (a) shows an example frame and how a 4×4 block of pixels is treated as a graph; (b) shows the original frame, the corrupted frame, and the reconstructed frame, from left to right.

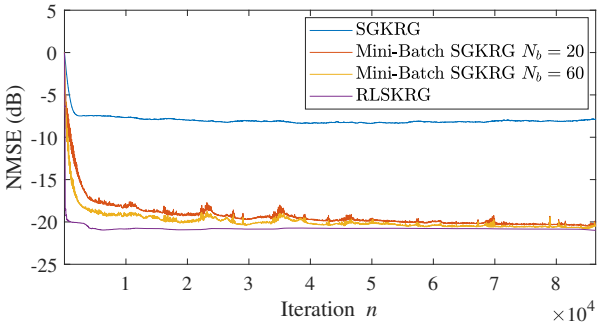


Fig. 8. NMSE achieved by the KRG implementations versus number of training samples in the image reconstruction simulation.

learning during a video sequence becomes quickly impractical for the conventional KRG due to the large dataset.

Fig. 8 shows the NMSE versus iterations for the proposed algorithms. These results are consistent with previous simulations and show that online KRG strategies can successfully learn the target model. We observe that the RLSKRG exhibits the best performance while the single-sample SGKRG exhibits the worst performance, as expected, given the complexity-performance trade-off. In this simulation, increasing the number of blocks to $N_\mathrm{b} = 20$ and $N_\mathrm{b} = 60$ (which corresponds to half the number of blocks on a single line in an image) considerably increases the performance of the MGKRG. A depiction of the frame reconstruction using the RLSKRG is presented in Fig. 7b, which showcases the capabilities of the proposed algorithm.

Other dedicated methods for image processing are available in the literature [47]. We provide online algorithms with reduced-complexity. Better accuracy can be achieved with other methods, at the cost of increased complexity or batch-based implementations. Moreover, the performance of KRG methods can be further improved as discussed in the following section.

VIII. Discussion on Applicability

According to the cost function in (3), the proposed methodology builds upon the assumption that the target signal is smooth with respect to the graph, by learning models that enforce a small variation metric (1). This suggests that KRG algorithms perform better when the signal variation is small with respect to the network topology, whereas modeling accuracy should not benefit from the proposed regularization when the graph signals are not smooth with respect to the topology.

The definition of total variation in (1) depends both on the graph topology, as a function of the Laplacian $\mathbf{L}$, and the graph signal. Thus, the performance of the KRG algorithms is subject to proper topology identification [36]. In the simulations presented in Section VII, we have resorted to simple methods for constructing the graphs, such as probability-based approaches in the synthesized-data example, and nearest-neighbor graphs for temperature prediction and image reconstruction. The nearest-neighbor topology is reasonable for many real-world applications based on physical structures, since one can expect that elements physically close to each other have similar behavior. However, given the availability of data, other approaches can be considered for constructing a graph to further benefit the proposed KRG algorithms.

In [15], a joint learning approach is proposed to learn both the graph Laplacian $\mathbf{L}$ and the regression parameters in matrix $\mathbf{W}$, by minimizing the cost function

$$C(\mathbf{W},\mathbf{L}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{W}^\mathrm{T}\mathbf{W}) + \beta \sum_{n=1}^{N} \nu(\mathbf{y}_n) + \gamma \mathrm{tr}(\mathbf{L}^\mathrm{T}\mathbf{L}). \tag{75}$$

The minimization problem (75) can be solved using an alternating minimization approach. A similar approach can be considered for the batch-based algorithm proposed here, and an adaptation of this method can be developed for the online algorithms. In fact, many other approaches for graph-topology learning can be considered to benefit the KRG algorithms. In practice, we are interested in learning a topology that induce smoothness on the graph signal. For instance, in [48], an approach based on factor analysis is proposed to estimate the graph Laplacian under the smoothness assumption. Other approaches including edge sparsity can be found in the litera-

ture [49], [50]. These approaches can be used to improve the performance of the proposed KRG algorithms.

IX. Conclusion

This paper proposed efficient batch-based implementations for kernel regression on graphs (KRG). The proposed implementations use random Fourier features (RFF) to overcome the growing complexity of kernel methods. Additionally, we showed that we could leverage the properties of the matrices involved in the regression process to formulate a less computationally-demanding derivation of the regression parameters. Furthermore, online strategies for RFF-based KRG were proposed, namely the mini-batch gradient KRG, the stochastic-gradient KRG, and the recursive least squares KRG. We showed that the RLSKRG also enjoys an alternative reduced-complexity implementation leveraging matrices' properties. For all online algorithms, conditions for convergence in the mean and the mean-squared sense were derived. We also presented a brief discussion on the trade-off between complexity and performance of the proposed algorithms. Finally, the performance of all algorithms was validated with numerical experiments using synthesized and real data simulations. Results confirmed that both the proposed KRG using RFF and the RLSKRG have accuracy close to that of the conventional KRG, with a considerable reduction in complexity. Additionally, simulations showed that the MGKRG can effectively learn the regression parameters and that its performance can be improved at a small increase in computational cost by increasing the number of samples in the mini-batch.

References

[1] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.

[2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, pp. 83–98, May 2013.

[3] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, pp. 808–828, May 2018.

[4] V. R. M. Elias, W. A. Martins and S. Werner, "Extended adjacency and scale-dependent graph Fourier transform via diffusion distances," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 592-604, Aug. 2020.

[5] A. Sandryhaila and J. M. F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, pp. 80–90, Sep. 2014.

[6] I. Jablonski, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors J.*, vol. 17, pp. 7659–7666, Dec. 2017.

[7] A. Gavili and X. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, pp. 6303–6318, Dec. 2017.

[8] M. J. M. Spelta and W. A. Martins, "Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation," *Signal Process.*, vol. 167, 107326, Feb. 2020.

[9] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, "Diffusion LMS with communication delays: Stability and performance analysis," *IEEE Signal Process. Lett.*, vol. 27, pp. 730-734, 2020.

[10] P. Di Lorenzo, P. Banelli, S. Barbarossa, and S. Sardellitti, "Distributed adaptive learning of graph signals," *IEEE Trans. Signal Process.*, vol. 65, pp. 4193–4208, Aug. 2017.

[11] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "A graph diffusion LMS strategy for adaptive graph signal processing," in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pp. 1973–1976, Oct. 2017.

[12] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 4129–4133.

[13] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, "Online distributed learning over graphs with multitask graph-filter models," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 63–77, Jan. 2020.

[14] R. Nassif, S. Vlaski, C. Richard, J. Chen and A. H. Sayed, "Multitask learning over graphs: An approach for distributed, streaming machine learning," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 14-25, May 2020.

[15] A. Venkitaraman, S. Chatterjee and P. Händel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 698–710, Dec. 2019.

[16] V. R. M. Elias, V. C. Gogineni, W. A. Martins and S. Werner, "Adaptive graph filters in reproducing kernel Hilbert spaces: Design and performance analysis," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 7, pp. 62-74, Jan. 2021.

[17] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1920–1932, 2018.

[18] Y. Shen, G. Leus, and G. B. Giannakis, "Online graph-adaptive learning with scalability and privacy," *IEEE Trans. Signal Process.*, vol. 67, pp. 2471–2483, May 2019.

[19] A. Sandryhaila and J. M. F. Moura, "Classification via regularization on graphs," in *Proc. IEEE Global Conf. Signal Inf. Process.*, 2013, pp. 495-498.

[20] E. Isufi, A. Loukas, N. Perraudin and G. Leus, "Forecasting time series with VARMA recursions on graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 18, pp. 4870-4885, Sep. 2019.

[21] D. Thanou, D. I. Shuman and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849-3862, Aug 2014.

[22] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[23] D. Romero, V. N. Ioannidis and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 856–869, Sep. 2017.

[24] V. N. Ioannidis, D. Romero and G. B. Giannakis, "Inference of Spatio-Temporal Functions Over Graphs via Multikernel Kriged Kalman Filtering," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3228-3239, June 2018.

[25] A. Singh, N. Ahuja, and P. Moulin, "Online learning with kernels: Overcoming the growing sum problem," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process.*, 2012, pp. 1–6.

[26] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[27] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.,* vol. 34, no. 4, pp. 18-42, Jul. 2017.

[28] F. Gama, E. Isufi, G. Leus and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Process. Mag.,* vol. 37, no. 6, pp. 128-138, Nov. 2020.

[29] V. C. Gogineni, G. S. R. E. Langberg, V. Naumova, J. Nygård, M. Nygård, M. Grasmair and S. Werner, "Data-driven personalized cervical cancer risk prediction: A graph-perspective," in *Proc. IEEE Int. Workshop Stat. Signal Process.*, Rio de Janeiro, 2021, pp. 46-50

[30] V. C. Gogineni, G. S. R. E. Langberg, V. Naumova, J. Nygård, M. Nygård, M. Grasmair and S. Werner, "Recurrent time-varying multi-graph convolutional neural networks for personalized cervical cancer risk prediction," in *Proc. Asilomar Conf. on Signals, Systems, and Computers*, 2021.

[31] X. Dong, D. Thanou, L. Toni, M. Bronstein and P. Frossard, "Graph signal processing for machine learning: A review and new perspectives," *IEEE Signal Process. Mag.* vol. 37, no. 6, pp. 117-127, Nov. 2020.

[32] G. Lewenfus, W. A. Martins, S. Chatzinotas and B. Ottersten, "Joint forecasting and interpolation of time-varying graph signals using deep learning," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 761-773, 2020.

[33] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1177–1184, 2007.

[34] P. Bouboulis, S. Pougkakiotis, and S. Theodoridis, "Efficient KLMS and KRLS algorithms: A random Fourier feature perspective," in *Proc. IEEE Stat. Signal Process. Workshop*, 2016, pp. 1–5.

[35] A. Sandryhaila and J. M. F. Moura, "Discrete Signal Processing on Graphs: Frequency Analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042-3054, June, 2014

[36] G. Mateos, S. Segarra, A. G. Marques and A. Ribeiro, "Connecting the Dots: Identifying Network Structure via Graph Signal Processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16-43, May 2019.

[37] J. Kivinen, A. J. Smola and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165-2176, Aug. 2004.

[38] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines. Lecture Notes in Computer Science*, vol. 2777, pp. 144-158, Springer, 2003.

[39] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. Wiley, Feb. 2010.

[40] P. S. R. Diniz, *Adaptive Filtering*. Springer, 2013.

[41] A. H. Sayed, *Adaptive Filters*. Wiley, Jan. 2008.

[42] R. Arablouei, K. Doğançay, S. Werner and Y. Huang, "Adaptive distributed estimation based on recursive least-squares and partial diffusion," *IEEE Trans. Signal Process.*, vol. 62, no. 14, pp. 3510-3522, July 2014.

[43] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear Algebra Appl.*, vol. 415, no 1, pp. 20–30, May 2006.

[44] R. Arablouei, K. Doğançay and S. Werner, "Recursive total least-squares algorithm based on inverse power method and dichotomous coordinate-descent iterations," *IEEE Trans. Signal Process.*, vol. 63, no. 8, pp. 1941–1949, Apr. 2015.

[45] Norwegian Meteorological Institute, *eKlima*, Norway, 2021. [Online] Available: http://sharki.oslo.dnmi.no/. [Accessed: 2021-01-08].

[46] KTH Royal Institute of Technology, Division of Information Science and Engineering, *Reproducible Research*, Sweden, 2020. [Online] Available: https://www.kth.se/ise/research/reproducibleresearch-1.433797. Accessed: 2020-08-18.

[47] G. Wang, J. C. Ye, K. Mueller and J. A. Fessler, "Image Reconstruction is a New Frontier of Machine Learning," *IEEE Trans. Med. Imaging* , vol. 37, no. 6, pp. 1289-1296, June 2018.

[48] X. Dong, D. Thanou, P. Frossard and P. Vandergheynst, "Learning Laplacian Matrix in Smooth Graph Signal Representations," *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6160-6173, Dec. 2016.

[49] S. P. Chepuri, S. Liu, G. Leus and A. O. Hero, "Learning sparse graphs under smoothness prior," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2017, pp. 6508-6512.

[50] V. Kalofolias, "How to learn a graph from smooth signals," *Proc. Int. Conf. Artif. Intell. Stat.*, 2016, pp. 920–929.