GLS Kernel Regression for Network-Structured Data

Edward Antonian, Gareth W. Peters, Mike Chantler, and Hongxuan Yan

***Abstract*—A common problem encountered in network applications is learning a mapping between a set of explanatory variables $\{\mathbf{x}\}_t$ and graph signals $\{\mathbf{y}\}_t$. Kernel Graph Regression (KGR) and Gaussian Processes over Graph (GPoG) have emerged as promising methods for such scenarios. This paper broadens the literature on KGR and GPoG by approaching the problem from the perspective of graph signal reconstruction, utilising insights from both areas. This allows for training set observations to be made on a subset of nodes only, which is a common feature of real-world network data. Next, we examine the statistical effect of correlated prediction error and propose an $O(N^3 + T^3)$ method for Generalized Least Squares (GLS) on graphs. In particular we examine AR(1) vector autoregressive processes, which are commonly found in time-series applications. Finally we use the Laplace approximation to determine a lower bound for the out-of-sample prediction error, and derive a scalable expression for the marginal variance of each prediction. These methods are tested and compared on real data taken from a network of air quality monitoring stations across California. We find that KGR and GLS KGR out-perform standard techniques demonstrating their effectiveness in specific use cases.**

***Index Terms*—Graph, Network, Regression, Kernel, GLS, autocorrelation**

I. Introduction

A. Motivation and related work

GRAPHS have proven to be a useful way of describing complex datasets due to their ability to capture general relational information between entities [1]. Knowledge about pairwise relationships between data points can, for example, help enhance regularization, improve computational efficiency and robustness, or better exploit unlabelled data [2], [3]. Successful applications have included social networks [4], brain imaging [5], biomolecular systems [6], sensor networks [7], image and point cloud processing [8] and chemical structure [9] to name a few.

The Graph Signal Processing (GSP) community focuses in particular on generalisations of traditional signal processing techniques on regular domains. A toolbox of analogous graph-based techniques has been built up by translating concepts such as convolution and spectral decomposition into the graph domain. These have been applied to a diverse range of problems including signal reconstruction [10], graph learning [11], denoising [12] and regression [13].

Graph-based regression specifically has received some attention recently in the form of kernel regression and Gaussian processes [13], [14], [15], [16]. In this context, we are interested in constructing a function that maps input features to an output space, which is interpreted as a graph signal. This opens interesting avenues for enhanced regularisation using GSP tools and offers novel ways of defining kernel functions.

In this paper, we examine Kernel Graph Regression (KGR) and the closely related topic of Gaussian Processes over Graph (GPoG) from the perspective of graph signal reconstruction. This framework allows signal prediction at unobserved test nodes as well as test times. Next, extend this by developing a method for Generalized Least Squares (GLS) regression on graphs. This has particular relevance for the statistical analysis of time-series data where a graph signal is sampled at regular intervals, as it allows, for example, node-level error autocorrelation to be incorporated. Such scenarios are encountered in the analysis of many common graph applications such as sensor networks and financial time series. Finally we use the Laplace approximation to calculate a lower bound for the out-of-sample prediction uncertainty and derive a scalable expression for the marginal uncertainty at each node.

B. Problem Overview and Scope

The goal of supervised learning and regression is to estimate an optimal function that maps an input space to an output space given a set of training examples $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T'}$. In this paper we are concerned specifically with real-valued vector inputs and outputs, that is, $\mathbf{x}_t \in \mathbb{R}^M$ and $\mathbf{y}_t \in \mathbb{R}^{N'}$. Furthermore, it is assumed that $\mathbf{y}_t$ is a partially observed graph signal, meaning the elements can be interpreted as existing on a subset of the nodes of a graph. This graph is assumed to be static in time with known adjacency matrix $A \in \mathbb{R}^{N \times N}$.

In this formulation, the input features $\mathbf{x}_t$ are 'global' in the sense that individual elements are not necessarily associated with any particular node in the output space. For example, if the graph represents a network of corporations, and the graph signal is their quarterly revenue growth, the features could represent global factors such as inflation, GDP etc, *as well as* firm-specific data such as the number of employees at a particular company. This is the most general formulation and allows for maximum flexibility in problem specification. Additional constraints explicitly linking elements from one space to the other are possible but not considered in this paper. This approach is shared by other recent work such as [13], [17].

Throughout this paper we take a Bayesian perspective, using graph filters to construct spectral priors that can be used to make statements about the expected graph signal. Using graph filters in this way shares aesthetic and practical features with the graph kernel framework described in [18], [19] and elsewhere. Graph filters as opposed to kernels, however, provide a more directly Bayesian interpretation which is useful in this context.

As mentioned, in this work, the graph adjacency matrix A is assumed to be known a priori. In many applications such as social or sensor networks this is a reasonable assumption, as a graph may be self-evident or easy to construct. For other applications, such as financial or biological networks, the dependency structure may be more opaque. In such scenarios, we refer the reader to the large body of work on graph learning [20], [21], [22], [23].

C. Contributions

This paper contributes a statistical exploration of Kernel Graph Regression and Gaussian Processes over Graphs in the context of graph signal reconstruction. A primary aim is to incorporate the possibility of error correlation, which is of practical concern in many applications. We pay particular attention to time series modelling and node-level autocorrelation, and describe an algorithm for determining function and model parameters in the presence of autocorrelated noise. We also focus throughout on deriving mathematical expressions which lend themselves to practical computation for large scale problems, and use Bayesian reasoning to provide a lower bound for the model parameter uncertainty. These contributions are clarified below.

1) Incorporating unobserved nodes via graph signal reconstruction: Prior work on KGR and GPoG has assumed that the signal from all nodes under consideration is observed throughout training. We introduce a small modification incorporating ideas from graph signal reconstruction to allow a constant subset of nodes to remain unobserved at train time, and use the topology of the graph to make smooth predictions at these points.

2) A generalisation of kernel graph regression by incorporating error correlation, with specific applications to time series modelling: We relax the statistical assumption that all prediction errors should be independent and identically distributed, allowing for the possibility of cross-correlation amongst nodes, and autocorrelation over time. This creates a broader class of model with rich statistical properties. In particular, we consider the scenario where AR(1) autocorrelation may exist at each node which has relevance in many time-series applications.

3) Strict enforcement of worst-case $O(N^3+T^3)$ complexity: Graph regression problems are naturally expressed in terms of $(N, N) \otimes (T, T)$ Kronecker products. This is problematic for all but the smallest of problems as memory and computational costs escalate quickly. This work contributes an analysis of the complexity of the methods and ensures that all final mathematical expressions are given in terms of worst-case $O(N^3 + T^3)$ operations, ensuring medium to large problems remain tractable.

4) A Laplace approximation for parameter uncertainty: We use Laplace approximation to estimate the full posterior, and derive a tractable expression for the marginal variance. This can be used to provide a lower bound for the uncertainty over out-of-sample model predictions.

D. Paper Organisation

Section II gives a brief introduction to some fundamental GSP concepts such as the Graph Fourier Transform (GFT) and graph filters. In section III we revisit the problem of Kernel Graph Regression (KGR) from a Bayesian perspective and analyse it in the context of graph signal reconstruction. Section IV builds on this, introducing the GLS model and outlining an iterative algorithm for estimating the prediction error, and focusing particularly on an autoregressive model. Section V uses the Laplace approximation to derive a tractable lower bound for the marginal prediction uncertainty. Finally, section VI analyses these methods on a real dataset concerning the prediction of pollutant levels across a network on monitoring stations in California.

E. Notation conventions

Effort is made to adhere to the following variable naming conventions throughout this paper.

TABLE I
THE NOTATIONAL CONVENTIONS USED IN THIS PAPER

Object	Description	Example
Integer variable	Lower case Latin	n, t, m
Integer constant	Upper case Latin	N, T, M
Scalar variable	Lower case Greek	α, β, ξ
Vector	Lower case bold Latin	$\mathbf{a}, \mathbf{x}, \mathbf{y}$
Vector element	Lower case sub lower case Latin	λ_i
Matrix	Upper case Latin or Greek	L, Σ_N, Λ_K
Matrix element	Matrix sub double lower case Latin	J_{tn}, G_{mn}
Matrix column	Matrix sub lower case Latin	J_n
Transpose	Symbol raised to $\top$	$X^\top$
Identity matrix	Capital I sub dimension	I_N
Optimal value	Symbol raised to $\star$	$W^\star$

II. PRELIMINARIES

A. Graphs and the GFT

Consider a weighted, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ made up of N vertices and edge set $\mathcal{E}$ connecting vertex pairs. $A \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix, where $A_{ij} = A_{ji}$ represents the strength of interaction between nodes i and j. $(i, j) \in \mathcal{E}$ implies $A_{ij} > 0$ and $(i, j) \notin \mathcal{E}$ implies $A_{ij} = 0$. The graph may equivalently be defined in terms of the Laplacian matrix $L = D - A$ where D is the diagonal degree matrix; D_{ii} being the ith row sum (or column sum) of A. One key property of the graph Laplacian is that it can be used to measure the smoothness of a signal $\mathbf{f} \in \mathbb{R}^N$ with respect to the graph.

$$\mathbf{f}^\top L\mathbf{f} = \sum_{(i,j)\in\mathcal{E}} A_{ij}\big(f_i - f_j\big)^2 \geq 0 \tag{1}$$

As visible, the quadratic product $\mathbf{f}^\top L\mathbf{f}$ measures the square difference in signal value between connected nodes, weighted

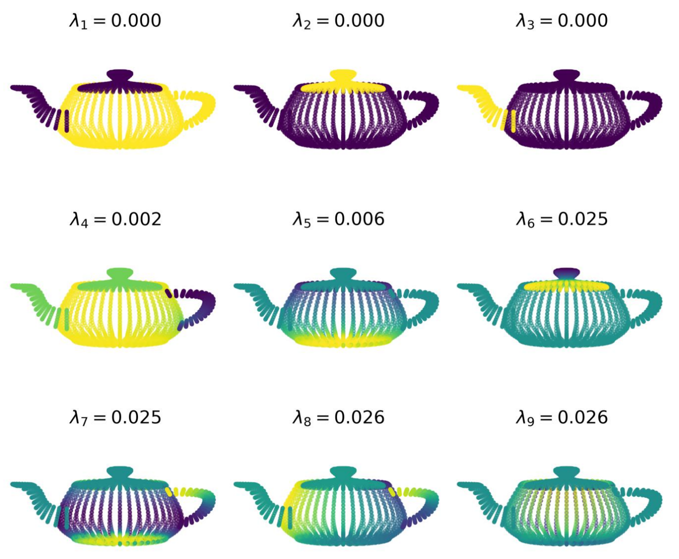

Fig. 1. Colour representation of the first nine eigenvectors of a 3D mesh graph constructed from the Utah Teapot, along with their associated eigenvalue. Here, the main body, lid and spout are all mutually disconnected, giving three zero eigenvalues.

by the respective adjacency matrix entry. In this way, it can be considered a measure of 'roughness' of the signal $\mathbf{f}$, with smaller values indicating a smoother signal, achieving a minimum of zero when the function is constant on each disconnected subgraph [24]. It can be useful to view the Laplacian in terms of its eigendecomposition, $L = U \Lambda_L U^\top$, where the columns of the unitary matrix U are the normalised eigenvectors of L, forming an orthonormal spanning set. (The eigenvalues are typically ordered such that $\lambda_1^L \leq \lambda_2^L \leq ... \leq \lambda_N^L$). Any graph signal can be expressed as a linear combination of the eigenvectors of the Laplacian as $\mathbf{f} = \sum_i a_i \mathbf{u}_i$, where $\mathbf{a} = U^\top \mathbf{f}$ is known as the Graph Fourier Transform of $\mathbf{f}$ [25]. This gives an alternative way to express the quadratic form of equation 1.

$$\mathbf{f}^\top L \mathbf{f} = \mathbf{a}^\top \Lambda_L \mathbf{a} = \sum_{i=1}^{N} a_i^2 \lambda_i^L \tag{2}$$

These two facts provide the interpretation of the eigenvectors of the Laplacian as having varying degrees of smoothness, ordered by the size of the associated eigenvalue. The number of eigenvalues equal to zero is the number of disconnected subgraphs. Beyond that, larger eigenvalues, analogous to frequency in classical signal processing, are associated with increasingly 'rough' eigenvectors. Figure 1 shows an example of this behaviour for a 3D mesh graph.

B. *Graph filters*

The concept of a graph filter can be constructed by extending the analogy of the GFT [26]. A low-pass filter, H, which operates on a graph signal $\mathbf{f}$, attenuates the high-frequency graph Fourier components and can be used to smooth or de-noise a signal [27]. It can be constructed by defining a decreasing function $\eta(\lambda)$ for which $\eta(0) = 1$ and $\lim_{\lambda \to \infty} \eta(\lambda) = 0$. The filter, H, is then defined by

$$H = U \eta(\Lambda_L) \, U^\top \tag{3}$$

where $\eta(\Lambda_L)$ denotes the application of $\eta(\cdot)$ element wise to the diagonal. The filtered signal $\mathbf{f}'$ is then calculated as $H\mathbf{f}$. Note that the operator H is necessarily symmetric and semi-positive definite. Numerous possibilities exist for $\eta(\lambda)$, some of which are outlined in table II.

TABLE II
POSSIBLE GRAPH FILTER FUNCTIONS

Filter	$\eta(\lambda;\, \beta)$	**Parameter**
Inverse	$(1 + \beta\lambda)^{-1}$	$\beta \in \mathbb{R}^+$
Exponential	$\exp(-\beta\lambda)$	$\beta \in \mathbb{R}^+$
ReLu	$\max(1 - \beta\lambda, 0)$	$\beta \in \mathbb{R}^+$
Sigmoid	$2\big(1 + \exp(\beta\lambda)\big)^{-1}$	$\beta \in \mathbb{R}^+$
Cosine	$\cos^{\beta}(\lambda\pi/2\lambda_{\text{max}})$	$\beta \in \mathbb{R}^+$
Cut-off	1, if $\lambda \leq \beta$ else 0	$\beta \in \mathbb{R}^+$

Graph filters constructed in this way bear a close resemblance to graph kernels as described in [18], where instead the authors define an increasing function $r(\lambda)$. In practical terms, a graph filter can be considered the inverse of a graph kernel as defined in this paper or, more accurately, $\eta(\lambda) = \lim_{c \to 0} 1/(c + r(\lambda))$. "Graph kernel" in this context should not be confused with another concept baring the same name, which concerns distance metrics between pairs of graphs [28].

C. *Graph spectral priors*

Graph filters can also be used to construct covariance matrices, for example in [14], [29]. Consider a Gaussian white noise graph signal $\mathbf{f}_\text{w}$ with precision γ. If a filter H is applied to give $\mathbf{f}_\text{g} = H\mathbf{f}_\text{w}$, the resultant signal will be smoother with respect to the graph, and will be drawn from a distribution

$$\mathbf{f}_\text{G} \sim \mathcal{N}(\mathbf{0}, \gamma^{-1} H^2) \tag{4}$$

In a Bayesian setting, H^2 can act as a covariance matrix which encodes the assumption that signals observed over a graph are likely to be smooth with respect to the underlying topology.

The selection of an appropriate graph filter in this context can be guided by prior knowledge about the expected power spectrum. Adaptive techniques have also been proposed for jointly learning a graph filter such as [15], [30] but are not considered in this paper.

III. KERNEL GRAPH REGRESSION

A. *Data and Definitions*

This problem runs over a set of T discrete "times" $\mathcal{T} = \{1, 2, ... T\}$. (These need not be regularly-sampled time intervals but in many practical problems will be). At each time, there exists a fixed graph with N nodes $\mathcal{N} = \{1, 2, \dots N\}$ related by a static adjacency matrix $A \in \mathbb{R}^{N \times N}$.

At a distinct subset of times $\mathcal{T}' \subset \mathcal{T}$ where $|\mathcal{T}'| = T'$ we observe supervised learning pairs $\{\mathbf{x}_t, \mathbf{y}_t\}_{t \in \mathcal{T}'}$. Here, $\mathbf{x}_t \in \mathbb{R}^M$ is a vector of explanatory variables, and $\mathbf{y}_t \in \mathbb{R}^{N'}$ is a partially observed graph signal. This signal is measured on a subset of the nodes $\mathcal{N}' \subset \mathcal{N}$ where $|\mathcal{N}'| = N'$ which remains constant over all t. In a sensor network, this may be the case if only a subset of the sensors are functioning correctly, or we wish to estimate the reading at locations where no pysical sensor is placed. The goal is to predict the graph signal at the unobserved nodes for $t \in \mathcal{T}'$ and at all nodes for $t \in \mathcal{T} - \mathcal{T}'$.

It is useful to define two binary selection matrices S_N and S_T which help in mapping between observed and unobserved node sets. S_N has entries (n, n') equal to one for $n = 1$ to $n = N'$ and $n' \in \mathcal{N}'$, with the rest set to zero. Similarly S_T has entries (t, t') equal to one for $t = 1$ to $t = T'$ and $t' \in \mathcal{T}'$, with the rest set to zero. For simplicity, and without loss of generality, we can assume that

$$\begin{aligned} S_N &= \begin{bmatrix} I_{N'} & 0_{N' \times (N-N')} \end{bmatrix} \in \mathbb{R}^{N' \times N} \\ S_T &= \begin{bmatrix} I_{T'} & 0_{T' \times (T-T')} \end{bmatrix} \in \mathbb{R}^{T' \times T} \end{aligned} \tag{5}$$

which implies that the first N' nodes are observed, and the first T' times contain the labeled examples. This is always possible under a reordering of nodes/times.

B. KGR via Signal Reconstruction

We will now derive a Kernel Graph Regression model from the perspective of graph signal smoothing and reconstruction. Let us assume that there exists an underlying noiseless graph signal $\mathbf{f}_t \in \mathbb{R}^N$ at each time that we wish to estimate. The observation $\mathbf{y}_t$ is modelled as a partial noisy observation of this true signal. This statement is summarised by the following model.

$$\mathbf{y}_t = S_N \mathbf{f}_t + \mathbf{e}_t, \quad \forall t \in \mathcal{T}' \tag{6}$$

where $\mathbf{e} \in \mathbb{R}^{N'}$ is a vector of standard normal iid noise. Or, more compactly,

$$Y = S_N F S_T^\top + E \tag{7}$$

Here $Y \in \mathbb{R}^{N' \times T'}$ is a horizontal stacking of each partial graph observation, so that the tth column is the partial graph signal observed at time t, and the nth row is the time series observed at node n. $F \in \mathbb{R}^{N \times T}$ is similar, but represents the true underlying function at the full, length-N, node set at all times. $E \in \mathbb{R}^{N' \times T'}$ is an error matrix of standard normal iid noise. The probability distribution of Y can therefore be expressed as

$$\text{vec}(Y) | F \sim \mathcal{N}\big(\text{vec}(S_N F S_T^\top),\, I_{T'} \otimes I_{N'}\big) \tag{8}$$

where $\text{vec}(\cdot)$ has the usual meaning of a vertical stacking of matrix columns, and $\otimes$ is the Kronecker product. In order to estimate the latent signal F, we must specify a prior indicating our belief about the likelihood of different signals. This provides the necessary Tikhonov regularization term and avoids under-specification.

To create this prior we can use two facts: 1) that graph signals at times t_1 and t_2 should be similar when $|\mathbf{x}_{t_1} - \mathbf{x}_{t_2}|$ is small. 2) The graph signal at each time should be smooth with respect to the graph topology. This information can be captured in the following prior:

$$\text{vec}(F) | X \sim \mathcal{N}\big(\mathbf{0},\, \gamma^{-1} K \otimes H^2\big) \tag{9}$$

where $K \in \mathbb{R}^{T \times T}$ is a kernel matrix defined by the relation $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j; \sigma)$, where $\kappa(\cdot, \cdot; \sigma)$ is an appropriate Mercer kernel with parameter(s) σ. γ is a parameter which controls the regularisation strength. Applying Bayes rule to equations 8 and 9 results in the following Maximum a Priori (MAP) optimisation problem for F.

$$F^\star = \underset{F}{\text{argmin}} \Big[- \ln \pi(F | Y, X) \Big] = \underset{F}{\text{argmin}} \Big[\xi(F) \Big]$$

where

$$\begin{aligned} \xi(F) = \text{tr}\big((Y - S_N F S_T^\top)^\top (Y - S_N F S_T^\top) \big) \\ + \gamma \, \text{tr}\big(K^{-1} F^\top H^{-2} F \big) \end{aligned} \tag{10}$$

Taking the derivative of this expression with respect to F and setting the result to zero yields a solution for $F^\star$. We refer to this as the KGR solution, but can also equivalently be considered the mean of a GPoG solution.

$$\begin{aligned} \text{vec}(F^\star) = \Big(S_T^\top S_T \otimes S_N^\top S_N + \gamma K^{-1} \otimes H^{-2} \Big)^{-1} \\ \big(S_T^\top \otimes S_N^\top \big) \text{vec}(Y) \end{aligned} \tag{11}$$

Applying a well-known matrix identity ([31], eqn. 162) allows the dimension of the inverse to be reduced from $NT \times NT$ to $N'T' \times N'T'$.

$$\begin{aligned} \text{vec}(F^\star) = \big(K S_T^\top \otimes H^2 S_N^\top \big) \times \\ \Big(\gamma I_{T'} \otimes I_{N'} + \bar{K} \otimes \bar{H}^2 \Big)^{-1} \text{vec}(Y) \end{aligned} \tag{12}$$

where $\bar{K} = S_T K S_T^\top$ and $\bar{H}^2 = S_N H^2 S_N^\top$. However, in this form, computation is impractical for large problems due to the high complexity involved in inverting the Kronecker-structured matrix. Thankfully this can be overcome by performing eigen-decomposition on $\bar{K}$ and $\bar{H}^2$ separately and leveraging well-known properties of the Kronecker product. For a detailed derivation, consult the supplementary materials. The result is

$$F^\star = H^2 S_N^\top \bar{U} \big(J \circ (\bar{U}^\top Y \bar{V}) \big) \bar{V}^\top S_T K \tag{13}$$

where

$$\bar{K} = \bar{V} \bar{\Lambda}_K \bar{V}^\top, \quad \text{and} \quad \bar{H}^2 = \bar{U} \bar{\Lambda}_H \bar{U}^\top \tag{14}$$

and $J \in \mathbb{R}^{N' \times T'}$ has the elements given by

$$J_{ij} = \frac{1}{\bar{\lambda}_j^K \bar{\lambda}_i^H + \gamma} \tag{15}$$

A key benefit of the solution in this form is that the necessary dense eigendecompositions, which are typically the most computationally taxing step, are performed on $\bar{K}$ and $\bar{H}^2$, of size $T' \times T'$ and $N' \times N'$ respectively, whereas a naive implementation would require this to be $T \times T$ and $N \times N$. This can be a significant speed-up for large problems, especially when a meaningful portion of the data is unlabeled. (Decomposition of $L \in \mathbb{R}^{N \times N}$ is also required, but this is typically sparse so can often be accelerated using sparsity-specific linear algebra tools).

A full outline of all the steps for graph kernel regression is highlighted in **Algorithm 1**.

Algorithm 1 KGR via Signal Reconstruction

```
Input: Covariate matrix $X \in \mathbb{R}^{M \times T}$
Input: Signal observations $Y \in R^{N' \times T'}$
Input: Selection matrices $S_N \in \mathbb{R}^{N' \times N}$ and $S_T \in \mathbb{R}^{T' \times T}$
Input: Graph Laplacian $L \in \mathbb{R}^{N \times N}$
Input: Regularisation parameter $\gamma \in \mathbb{R}$
Input: Graph filter function $\eta(\,\cdot\,; \beta \in \mathbb{R}^+)$
Input: Kernel function $\kappa(\,\cdot\,,\,\cdot\,; \sigma)$
    Compute $K \in \mathbb{R}^{T \times T}$ where $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j; \sigma)$
    Decompose $L$ into $U \Lambda_L U^\top$
    Compute $H^2$ as $H^2 = U \eta^2(\Lambda_L) U^\top$
    Decompose $S_T K S_T^\top$ into $\bar{V} \bar{\Lambda}_K \bar{V}^\top$
    Decompose $S_N H^2 S_N^\top$ into $\bar{U} \bar{\Lambda}_H \bar{U}^\top$
    Compute $J$ as $J_{ij} = 1/(\gamma + \lambda_j^K \lambda_i^H)$
Output: $H^2 S_N^\top \bar{U} \big(J \circ (\bar{U}^\top Y \bar{V})\big) \bar{V}^\top S_T K$
```

C. KGR with Graph Features via Product Graphs

In this short section we highlight an alternative way of posing KGR that unites it fully with literature on graph signal reconstruction. In section III-B, equation 9, we used the covariance matrix $K \otimes H^2$ to encode the belief that signals should be smooth over the graph, and smooth in feature space. A natural alternative to using the kernel matrix K is instead to use another graph filter H_T so that the total covariance matrix is $H_T^2 \otimes H_N^2$, by defining a time-like "graph" that is in some way derived from the features $\{\mathbf{x}_t\}$. Under this formulation, the regression task becomes a pure signal reconstruction problem for a partially observed signal over an extended product graph.

This is closely related to the concept of "Doubly-selective space-time kernels" outlined in [32]. As described by the authors, one can consider an "extended" graph (also known as a Cartesian product graph), of dimension $NT \times NT$, made up of T slices. Each slice is internally connected via adjacency matrix A_N and connected to other slices according to adjacency matrix A_T. A visual representation is depicted in Figure 2.

The resultant extended graph has adjacency matrix A_E and Laplacian L_E, which are given by

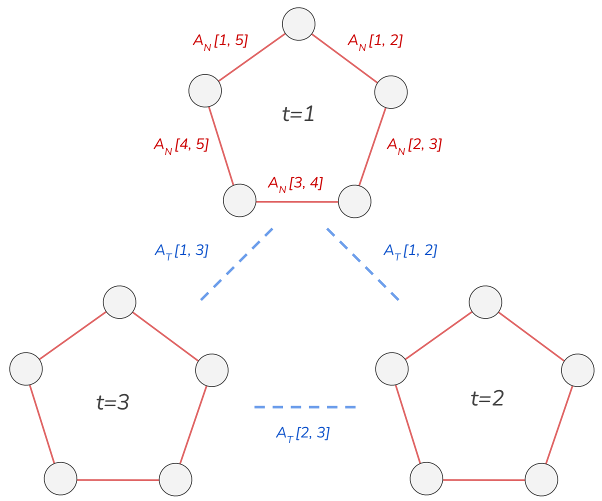

Fig. 2. A visual representation of an extended graph. Here, there are three slices of a graph with adjacency matrix $A_N \in \mathbb{R}^{5 \times 5}$ in red. Each of these slices is mutually connected according to $A_T \in \mathbb{R}^{3 \times 3}$.

$$A_E = A_T \oplus A_N, \quad L_E = L_T \oplus L_N$$

where $\oplus$ is the Kronecker sum defined by $A \oplus B = A \otimes I + I \otimes B$. We can then proceed to construct a graph filter for the extended graph as described in section II-B, by applying a decreasing function $\eta(\cdot)$ to the eigenvalues of L_E.

$$\begin{aligned} H_E &= U_E \, \eta(\Lambda_E) U_E^\top \\ &= (U_T \otimes U_N) \, \eta(\Lambda_T \oplus \Lambda_N)(U_T^\top \otimes U_N^\top) \end{aligned} \tag{16}$$

It follows that when $\eta(\Lambda_T \oplus \Lambda_N)$ can be expressed as $\eta(\Lambda_T) \otimes \eta(\Lambda_N)$, H_E can be expressed as $H_T \otimes H_N$. This is equivalent to the statement that the function $\eta(\cdot)$ is separable in the following way: $\eta(\lambda_1 + \lambda_2) = \eta(\lambda_1)\eta(\lambda_2)$. This is not true in general or indeed for the majority of the filter functions defined in table II, but is the case with, for example, the exponential filter. This is significant because, if this condition holds, then we can justifiably use $H_T^2 \otimes H_N^2$ as the global signal covariance matrix on the grounds that it is the square of a valid filter defined on the extended graph.

The following questions then arise: 1. How should we construct the time-like adjacency matrix from the features $\{\mathbf{x}_t\}$? 2. Under what circumstances is this likely outperform a more common kernel? The first question comes under the topic of data-driven graph construction. Typically, simple sparsity-inducing methods are chosen such as weighted k-nearest neighbour or ϵ-neighborhood, however more sophisticated approaches have been explored recently such as [33]. The answer to the second question may depend on specifics of the data. If there is significantly more unlabelled data than labelled data, a graph-based kernel may help improve prediction accuracy by exploiting the inherent structure of $\{\mathbf{x}_t\}$. This concept forms the basis of semi-supervised learning [34]. However, it can be challenging to select the best graph construction method and tune its relevant hyperparameters, a detailed discussion of which can be found in [35].

The parameters which characterise the behaviour of filters H_T and H_N need not be the same. To see this, take the case of defining an exponential filter with parameter $\beta = 1$ for the graph given by $A_E = \beta_T A_T \oplus \beta_N A_N$. This achieves the same result as defining our filter with two different parameters β_T and β_N, and applying it to the graph given by $A_E = A_T \oplus A_N$. Since the former is a valid way to construct the extended graph, the latter is also justified.

The method, which in practice very similar to KGR as outlined in section III-B, is shown in in **Algorithm 2**. We refer to it as Kernel Graph Regression with Graph Features (KGR GF)

Algorithm 2 KGR with Graph Features

```
Input: Signal observations $Y \in R^{N' \times T'}$
Input: Selection matrices $S_N \in \mathbb{R}^{N' \times N}$ and $S_T \in \mathbb{R}^{T' \times T}$
Input: Graph Laplacian $L_N \in \mathbb{R}^{N \times N}$
Input: Time-like graph Laplacian $L_T \in \mathbb{R}^{T \times T}$
Input: Regularisation parameter $\gamma \in \mathbb{R}$
Input: *Separable* graph filter functions $\eta(\,\cdot\,; \beta_N), \eta(\,\cdot\,; \beta_T)$
    Decompose $L_N$ into $U_N \Lambda_N U_N^\top$
    Decompose $L_T$ into $U_T \Lambda_T U_T^\top$
    Compute $H_N^2 = U_N \eta^2(\Lambda_N\,; \beta_N) U_N^\top$
    Compute $H_T^2 = U_T \eta^2(\Lambda_T\,; \beta_T) U_T^\top$
    Decompose $S_N H_N^2 S_N^\top$ into $\bar{U}_N \bar{\Lambda}_N \bar{U}_N{}^\top$
    Decompose $S_T H_T^2 S_T^\top$ into $\bar{U}_T \bar{\Lambda}_T \bar{U}_T^\top$
    Compute $J$ as $J_{ij} = 1/(\gamma + \lambda_j^T \lambda_i^N)$
Output: $H_N^2 S_N^\top \bar{U}_N \big(J \circ (\bar{U}_N^\top Y \bar{U}_T)\big)\bar{U}_T^\top S_T H_T^2$
```

IV. GLS Kernel Graph Regression

In many real-world applications of regression modelling, the assumption that the error terms of equation 6 are independent and identically distributed is unlikely to hold. In this section, we consider the situation where the differences between the observed and underlying signal are instead correlated according to a matrix normal distribution. Under this model, the elements of Y are distributed as

$$\mathrm{vec}(Y)|F \sim \mathcal{N}\big(\mathrm{vec}(S_N F S_T^\top),\, \Sigma_T \otimes \Sigma_N\big) \tag{17}$$

Although alternative covariance models exist and have been studied, the matrix normal distribution is a relatively natural way to describe the characteristics of spatio-temporal data as it has a simple and meaningful interpretation [36].

A. KGR with Gauss–Markov Estimator

To begin, we derive a Gauss–Markov estimator for F, that is, the Best Linear Unbiased Estimator (BLUE) assuming that the covariance matrix $\Sigma_T \otimes \Sigma_N$ is known. (This restriction is relaxed in the subsequent section). In this case, the log-likelihood of making an observation Y is altered such that the cost function of equation 10 becomes

$$\begin{aligned}\xi(F) = \mathrm{tr}\big((Y - S_N F S_T^\top)^\top \Sigma_N^{-1} (Y - S_N F S_T^\top)\, \Sigma_T^{-1}\big) \\ + \gamma\, \mathrm{tr}\big(K^{-1} F^\top H^{-2} F\big)\end{aligned} \tag{18}$$

The BLUE estimator for a given Σ_N and Σ_T is defined to be the value of F which minimises this expression, and can be found by differentiating it with respect to F and setting the result equal to zero. By again following the same steps as in section III, this procedure results in

$$\begin{aligned}\mathrm{vec}(F^\star) = \Big(S_T^\top \Sigma_T^{-1} S_T \otimes S_N^\top \Sigma_N^{-1} S_N + \gamma K^{-1} \otimes H^{-2}\Big)^{-1} \\ \big(S_T^\top \Sigma_T^{-1} \otimes S_N^\top \Sigma_N^{-1}\big)\, \mathrm{vec}(Y)\end{aligned} \tag{19}$$

Once again, in this form, the solution remains prohibitively expensive to compute for large N' and T'. However, it can be expressed alternatively with $O(N'^3 + T'^3)$ complexity by making the following definitions. First eigen-decompose Σ_N and Σ_T.

$$\Sigma_T = \Psi \Lambda_{\Sigma_T} \Psi^\top, \quad \Sigma_N = \Phi \Lambda_{\Sigma_N} \Phi^\top$$

Then perform eigendecomposition on the following matrices:

$$\begin{aligned}\Lambda_{\Sigma_T}^{-1/2} \Psi^\top S_T K S_T^\top \Psi \Lambda_{\Sigma_T}^{-1/2} &= \bar{V} \bar{\Lambda}_K \bar{V}^\top \\ \Lambda_{\Sigma_N}^{-1/2} \Phi^\top S_N H^2 S_N^\top \Phi \Lambda_{\Sigma_N}^{-1/2} &= \bar{U} \bar{\Lambda}_H \bar{U}^\top\end{aligned}$$

Note that both these matrices are guaranteed to be symmetric with positive, real eigenvalues [37]. Finally, make the following definitions:

$$\begin{aligned}J_{ij} &= \frac{1}{\gamma + \bar{\lambda}_j^K \bar{\lambda}_i^H} \\ B &= H^2 S_N^\top \Phi \Lambda_{\Sigma_N}^{-1/2} \bar{U} \\ C &= K S_T^\top \Psi \Lambda_{\Sigma_T}^{-1/2} \bar{V} \\ \bar{Y} &= \bar{U}^\top \Lambda_{\Sigma_N}^{-1/2} \Phi^\top Y \Psi \Lambda_{\Sigma_T}^{-1/2} \bar{V}\end{aligned}$$

The GLS estimator for $F^\star$ is given by

$$F^\star = B\,(J \circ \bar{Y})\,C^\top \tag{20}$$

While a simpler $O(N^3 + T^3)$ solution is possible, the above method ensures that the eigendecompositions are performed on reduced size matrices, effectively scaling as $O(N'^3 + T'^3)$. For a detailed derivation with intermediate steps we refer the reader to the supplementary materials.

Algorithm 3 KGR Gauss–Markov Estimator

```
function BLUEESTIMATOR($\Sigma_N, \Sigma_T$)
    Decompose $\Sigma_T$ into $\Psi \Lambda_{\Sigma_T} \Psi^\top$
    Decompose $\Sigma_N$ into $\Phi \Lambda_{\Sigma_N} \Phi^\top$
    Decompose $\Lambda_{\Sigma_T}^{-1/2} \Psi^\top S_T K S_T^\top \Psi \Lambda_{\Sigma_T}^{-1/2}$ into $\bar{V} \Lambda_K \bar{V}^\top$
    Decompose $\Lambda_{\Sigma_N}^{-1/2} \Phi^\top S_N H^2 S_N^\top \Phi \Lambda_{\Sigma_N}^{-1/2}$ into $\bar{U} \Lambda_H \bar{U}^\top$
    Compute $J$ as $J_{ij} = 1/(\gamma + \lambda_j^K \lambda_i^H)$
    $B \leftarrow H^2 S_N^\top \Phi \Lambda_{\Sigma_N}^{-1/2} \bar{U}$
    $C \leftarrow K S_T^\top \Psi \Lambda_{\Sigma_T}^{-1/2} \bar{V}$
    $\bar{Y} \leftarrow \bar{U}^\top \Lambda_{\Sigma_N}^{-1/2} \Phi^\top Y \Psi \Lambda_{\Sigma_T}^{-1/2} \bar{V}$
    return $B\,(J \circ \bar{Y})\,C^\top$
end function
```

B. Estimating Σ_N and Σ_T given F

In this section, we construct estimators for the covariance matrices Σ_N and Σ_T, given that the latent signal F, and therefore the prediction error $E = Y - S_N F S_T^\top$, is known. It follows from equation 17, that E has a matrix normal distribution of the following form.

$$\text{vec}(E) \sim \mathcal{N}(\mathbf{0},\ \Sigma_T \otimes \Sigma_N) \tag{21}$$

Past literature on parameter estimation in matrix-normal models has mostly focused on the task of estimating the covariance given multiple realisations of the error matrix E [38], [39], [40], [41]. In order to estimate both Σ_N and Σ_T in full, the number of observations of E must be greater than $[\max(N'/T', T'/N') + 1]$ [42]. However, in the present case, only a single observation comprising $N'T'$ values is available. It is therefore necessary to assume a simplified, parametrised structure for at least one of these matrices to reduce the number of degrees of freedom.

For this reason, we assume that the matrix $\Sigma_T(\theta)$ describes the autocorrelation between different time measurements and is a function of a parameter or set of parameters θ. The elements of $\Sigma_T(\theta)$ are constrained to be dimensionless Pearson correlation coefficients, with 1 along the diagonal. Conversely, Σ_N is assumed to be an arbitrary covariance matrix that can take on any value satisfying semi-positive definiteness. By explicitly parametrising one as a correlation matrix, while letting the other float freely as a covariance matrix, the problem of the solution being under-determined (since $a\Sigma_T \otimes \frac{1}{a}\Sigma_N$ give the same matrix for any a) is avoided.

The likelihood of observing an error matrix E, given covariance matrices Σ_N and $\Sigma_T(\theta)$, is

$$\begin{aligned}\pi(E|\theta, \Sigma_N) = 2\pi^{\frac{NT}{2}} \big(|\Sigma_T(\theta)| \otimes |\Sigma_N|\big)^{-1/2} \times \\ \exp\Big(-\frac{1}{2}\text{tr}\big(E^\top \Sigma_N^{-1} E\, \Sigma_T^{-1}(\theta)\big)\Big).\end{aligned} \tag{22}$$

By specifying two independent prior distributions $\pi(\theta)$ and $\pi(\Sigma_N)$, the posterior density over θ and Σ_N can be expressed via Bayes rule as

$$\pi(\theta, \Sigma_N|E) = \frac{\pi(E|\theta, \Sigma_N)\,\pi(\theta)\,\pi(\Sigma_N)}{\pi(E)},$$

Depending on the size of the problem, this could be used along with a Monte Carlo algorithm to sample from the whole posterior or, more likely, be used to calculate single MAP estimates for θ and Σ_N. In either case, the negative log posterior, up to a scaling and additive constant, is equal to

$$\begin{aligned}\xi(\theta, \Sigma_N) \;&=\; N \ln|\Sigma_T(\theta)| + T\ln|\Sigma_N| \\ &\quad + \text{tr}\big(E^\top \Sigma_N^{-1} E\, \Sigma_T^{-1}(\theta)\big) \\ &\quad - 2\ln\pi(\theta) - 2\ln\pi(\Sigma_N)\end{aligned} \tag{23}$$

A MAP estimate can be found by minimising this quantity jointly with respect to the parameters. There are a number of possible strategies for solving this, such as simple gradient descent. However, since an analytical solution for θ is often possible given Σ_N and vice versa, an efficient algorithm using temporary approximate solutions can often be found. This so-called ‘flip-flop’ strategy is widely used in the literature on Kronecker-product covariance estimation [42].

C. Example: AR(1) process

The previous section stated the general formulation of the problem without any particular parametrisation of the matrix $\Sigma_T(\theta)$. In this section we give a concrete example by considering the widely used AR(1) model, which assumes serial simple correlation between prediction errors. Denoting the t-th column of E as E_t, the general AR(1) model assumes that

$$E_t = \Theta\, E_{t-1} + \boldsymbol{\varepsilon}_t, \quad \boldsymbol{\varepsilon}_t \sim \mathcal{N}(\mathbf{0}, \Sigma_N), \tag{24}$$

where $\Theta \in \mathbb{R}^{N \times N}$ is a matrix of autoregression coefficients [43]. In the following we assume a simplified version, where the n-th element of E_t is assumed only to be serially correlated with the n-th element of $_{t-1}$, in a way that is both stationary and uniform across all n. In effect, this requires that $\Theta = \theta I_N$. This is, in essence, the multivariate extension of the AR(1) model considered in the seminal paper by Cochrane and Orcutt [44]. In the limit of large T, the correlation matrix Σ_T, and its inverse Σ_T^{-1}, have a well-known form:

$$\Sigma_T(\theta) = \begin{bmatrix} 1 & \theta & \theta^2 & \cdots & \theta^{T-1} \\ \theta & 1 & \theta & \cdots & \theta^{T-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \theta^{T-1} & \theta^{T-2} & \theta^{T-3} & \cdots & 1 \end{bmatrix},$$

$$\Sigma_T^{-1}(\theta) = \frac{1}{1-\theta^2} \begin{bmatrix} 1 & -\theta & & & 0 \\ -\theta & 1+\theta^2 & -\theta & & \\ & \ddots & \ddots & \ddots & \\ & & -\theta & 1+\theta^2 & -\theta \\ 0 & & & -\theta & 1 \end{bmatrix}$$
$$= \frac{1}{1-\theta^2}\Big(I_T - \theta B_1 + \theta^2 B_2\Big), \tag{25}$$

where B_1 is a matrix with 0 on the diagonal and with 1 on the first upper and lower diagonal, and B_2 is the identity matrix with zero on the first and last diagonal entries [45]. Note also that the determinant of Σ_T is given by $(1-\theta^2)^{T-1}$ [46]. To be a valid stationary process, the parameter θ must lie on the interval $[-1, 1]$. An appropriate prior distribution for θ could therefore be

$$\pi(\theta|\alpha) \propto \exp\Big(-\frac{NT\alpha/2}{1-\theta^2}\Big), \quad \text{for} \quad \theta \in [-1, 1] \tag{26}$$

for some scalar parameter α. Depending on the value of α, this prior is roughly uniform across the majority of the interval, whilst rapidly decreasing in likelihood towards -1 and 1. This effectively encodes a belief that the time series is stationary across all nodes, with α controlling the strength of that stationarity assumption. The resultant θ-dependant part of the cost function of equation 23 therefore becomes

$$\xi(\theta, \Sigma_N) = N(T-1)\ln(1-\theta^2) + \text{tr}\big(E\Sigma_T^{-1}(\theta)E^\top \Sigma_N\big) + \frac{NT\alpha}{1-\theta^2} \tag{27}$$

Whilst this is not a quadratic function of θ, the unique maximum likelihood estimator $\hat{\theta}$ can be found by differentiating this expression and setting the result to zero. This results in a MAP estimator which is the real root of a cubic polynomial.

$$\hat{\theta}(\Sigma_N) = \text{real root}\Big[\,\frac{b}{2} + \big(N(T-1) - a - c - NT\alpha\big)\,\theta + \frac{b}{2}\theta^2 - N(T-1)\,\theta^3\,\Big], \tag{28}$$

where

$$a = \text{tr}\big(E^\top\Sigma_N^{-1}E\big), \quad b = \text{tr}\big(E^\top\Sigma_N^{-1}EB_1\big), \quad \text{and} \quad c = \text{tr}\big(E^\top\Sigma_N^{-1}EB_2\big). \tag{29}$$

A derivation of this expression can be found in the supplementary materials. Note that this estimator is a function of Σ_N.

In terms of the cross-covariance matrix Σ_N, we choose to implement a modified form of the estimator proposed in [47] and further developed in [48]. In essence, these papers considered a weighted combination of the high-variance and often ill-posed maximum likelihood estimate, $\Sigma_{\text{ML}} = \frac{1}{T}EE^\top$, and the heavily-biased but well-conditioned estimate, $\frac{1}{N}\text{tr}\big(\Sigma_{\text{ML}}\big)I_N$. The only modification presented here is the introduction of Σ_T^{-1} into the ML estimate, that is, $\Sigma_{\text{ML}} = \frac{1}{T}E\Sigma_T^{-1}(\theta)E^\top$, which is necessary to account for the effect of autocorrelation.

$$\hat{\Sigma}_N(\theta) = (1-\rho)\Sigma_{\text{ML}} + \frac{\rho}{N}\text{tr}\big(\Sigma_{\text{ML}}\big)I_N \tag{30}$$

[47] and [48] also investigate the optimal setting of the shrinkage coefficient ρ, which is a parameter between 0 and 1. Here, we choose to implement the Rao-Blackwell Ledoit-Wolf (RBLW) estimator described in [48]. This is given by

$$\hat{\rho}_{\text{RBLW}} = \min\left[\frac{\frac{T-2}{T}\text{tr}\big(\Sigma_{\text{ML}}^2\big) + \text{tr}^2\big(\Sigma_{\text{ML}}\big)}{(T+2)\Big(\text{tr}\big(\Sigma_{\text{ML}}^2\big) - \frac{1}{N}\text{tr}^2\big(\Sigma_{\text{ML}}\big)\Big)},\ 1\right]$$

Algorithm 4 AR(1) Estimator for Σ_N, Σ_T given F

```
Initialise $\Sigma_N$
function EstimateCovar($F$)
    $E \leftarrow Y - S_N F S_T^\top$
    while $\theta, \Sigma_N$ not converged do
        $a \leftarrow \text{tr}\big(E^\top\Sigma_N^{-1}E\big)$
        $b \leftarrow \text{tr}\big(E^\top\Sigma_N^{-1}EB_1\big)$
        $c \leftarrow \text{tr}\big(E^\top\Sigma_N^{-1}EB_2\big)$
        $\theta \leftarrow \text{real root}\big(\text{cubic}(\theta; a, b, c, \alpha)\big)$        [eqn. 28]
        $\Sigma_{\text{ML}} \leftarrow \frac{1}{T}E\Sigma_T^{-1}(\theta)E^\top$
        $\rho \leftarrow \hat{\rho}_{\text{RBLW}}(\Sigma_{\text{ML}})$
        $\Sigma_N \leftarrow (1-\rho)\Sigma_{\text{ML}} + \frac{\rho}{N}\text{tr}\big(\Sigma_{\text{ML}}\big)I_N$
    end while
    return $\Sigma_T(\theta), \Sigma_N$
end function
```

D. GLS Kernel Graph Regression

To complete the GLS kernel regression algorithm, estimation of both F and $\Sigma_T \otimes \Sigma_N$ must be performed simultaneously. Since both rely on each other to be estimated, it is necessary to implement an alternating iterative algorithm, where first F is estimated for some reasonable initial guess of Σ_N and Σ_T, and then Σ_N and Σ_T are estimated for this value of F. This process then continues until some convergence criterion is met.

These steps are outlined in **algorithm 5**.

V. Prediction Uncertainty via the Laplace Approximation

In this final methodology section we outline the steps for estimating the uncertainty over the latent signal F via the

Algorithm 5 AR(1) GLS Kernel Graph Regression

```
Input: Covariate matrix $X \in \mathbb{R}^{M \times T}$
Input: Target signal $Y \in R^{N' \times T'}$
Input: Graph Laplacian $L \in \mathbb{R}^{N \times N}$
Input: Regularisation parameter $\gamma \in \mathbb{R}$
Input: Graph filter function $\eta(\,\cdot\,; \beta \in \mathbb{R}^+)$
Input: Kernel function $\kappa(\,\cdot\,,\,\cdot\,; \sigma)$
Input: Autocorrelation regularisation parameter $\alpha$
  Decompose $L$ into $U \Lambda_L U^\top$
  $H \leftarrow U \eta(\Lambda_L) U^\top$
  $\theta \leftarrow 0$
  $\Sigma_N \leftarrow I_{N'}$
  while $F$ not converged do
      $F \leftarrow$ EstimateFGLS($\Sigma_T(\theta), \Sigma_N$)        [Algorithm 3]
      $\Sigma_T, \Sigma_N \leftarrow$ EstimateCovar($F$)          [Algorithm 4]
  end while
Output: $F, \Sigma_T, \Sigma_N$
```

Laplace approximation. In the case of KGR, this is in fact not an approximation, but the exact Gaussian process prior. On the other hand, in the GLS case it is indeed an approximation as point estimates for the uncertain covariance matrices Σ_N and Σ_T are used. Here, we derive the approximate posterior for the GLS case only, but note that the exact posterior for simple KGR can be restored by setting Σ_T and Σ_N equal to the identity matrix of an appropriate size.

Since the latent signal F has size $N \times T$, the true covariance matrix specifying its uncertainty has size $NT \times NT$. Deriving a mathematical expression for this matrix is simple, however it is generally impractical to compute and store in memory for all but the smallest of problems. For this reason we take steps to derive an efficient expression for the $N \times T$ matrix representing the marginal variance of each element of F, which is generally the most useful information and allows scaling to relatively large problems.

Consider the negative log-likelihood given in equation 18. Applying Laplcace's approximation gives that the covariance matrix for the uncertainty over the vectorized latent signal.

$$\Sigma_F = \left[- \frac{\partial^2 \ln \pi(F|Y,X)}{\partial \text{vec}(F)_i \partial \text{vec}(F)_j} \Big|_{F=F^\star} \right]^{-1} \tag{31}$$

$$= \left(S_T^\top \Sigma_T^{-1} S_T \otimes S_N^\top \Sigma_N^{-1} S_N + \gamma K^{-1} \otimes H^{-2} \right)^{-1} \tag{32}$$

Holding this dense matrix in memory is often impractical, and inverting it generally out of the question. Instead, define the matrix $\Omega_F \in \mathbb{R}^{N \times T}$ defined by $\Omega_F = \text{mat}(\text{diag}^{-1}(\Sigma_F))$. That is, Ω_F is the diagonal of Σ_F of length NT stacked into a $N \times T$ matrix of marginal variances, such that element (t, n) is the square uncertainty for the n-th node at time t. Here we present an expression for Ω_F followed by a short proof.

$$\Omega_F = \left(\tilde{U}^{-\top} \circ (H^2 \tilde{U}) \right) J \left(\tilde{V}^{-1} \circ (\tilde{V}^\top K) \right) \tag{33}$$

where $\circ$ is the Hadamard product,

$$S_T^\top \Sigma_T^{-1} S_T K = \tilde{V} \tilde{\Lambda}_K \tilde{V}^{-1}, \quad S_N^\top \Sigma_N^{-1} S_N H^2 = \tilde{U} \tilde{\Lambda}_H \tilde{U}^{-1}$$

and

$$J_{ij} = \frac{1}{\gamma + \tilde{\lambda}_i^K \tilde{\lambda}_j^H}$$

In order to prove this expression, first note that Σ_F can be factorized into the following:

$$\Sigma_F = \left(K \otimes H^2 \right) \times \left(S_T^\top \Sigma_T^{-1} S_T K \otimes S_N^\top \Sigma_N^{-1} S_N H^2 + \gamma I_T \otimes I_N \right)^{-1}$$

From here, substitute in the eigen-decomposition definitions.

$$\Sigma_F = (K\tilde{V} \otimes H^2 \tilde{U}) \, \text{vec}(\text{diag}(J)) \, (\tilde{V}^{-1} \otimes \tilde{V}^{-1})$$

This can be split into the following sum.

$$\Sigma_F = \sum_{t=1}^{T} K \tilde{V} \delta_t \tilde{V}^{-1} \otimes H^2 \tilde{U} \, \text{diag}(J_t) \, \tilde{U}^{-1}$$

where δ_t is defined to be a $T \times T$ matrix of zeros with element (t, t) equal to one. From this, it can be seen that the re-stacked diagonal is given by the following outer product sum.

$$\Omega_F = \sum_{t=1}^{T} \mathbf{a}_t \mathbf{b}_t^\top$$

where

$$\mathbf{a}_t = \left(\tilde{U}^{-\top} \circ (H^2 \tilde{U}) \right) J_t, \quad \mathbf{b}_t = (\tilde{V}^{-1})_t \circ (K\tilde{V})_t$$

Or more, compactly

$$\Omega_F = AB^\top = \left(\tilde{U}^{-\top} \circ (H^2 \tilde{U}) \right) J \left(\tilde{V}^{-1} \circ (\tilde{V}^\top K) \right)$$

Further detail concerning this derivation can be found in the supplementary materials.

VI. Experimental Results

A. Spatio-temporal pollutant analysis

In this section, we consider the problem of predicting the concentration of various airborne pollutants, measured across a network of air quality monitoring stations in and around California. Each pollutant type is measured at a unique set of locations and is therefore treated as an independent graph regression task. The goal is to make accurate predictions using weather and environmental features from the previous day. The methods of Kernel Graph Regression and AR(1) GLS Kernel Graph Regression are analysed and compared to some standard baseline algorithms. We also investigate the effect of using an extended product graph outlined in section III-C.

Pollutant concentration data was taken from the US Environmental Protection Agency's air quality monitoring program [49]. Specifically, daily measurements of Ozone, Carbon

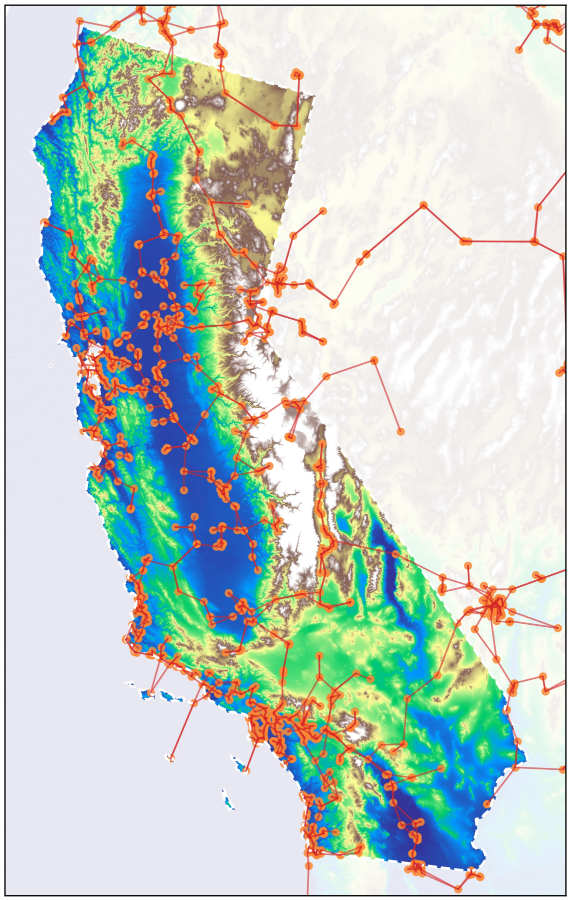

Fig. 3. A graphical depiction of the graph connecting the available monitoring stations, with the map colour representing the elevation profile.

Monoxide (CO), , Nitrogen Dioxide (NO_2), $PM_{2.5}$ and PM_{10} were taken from January 2017 to April 2021. This data set also contains daily measurements of humidity, pressure, wind speed and temperature at various locations. In addition, data concerning historical wildfires in California was sourced from the Department of Forestry and Fire Protection in California [50].

B. Graph Construction

A key decision which can significantly impact the effectiveness of graph a signal processing method is how to construct the underlying graph. In certain applications, such as social networks, a sparse graph may be self-evident or relatively simple to construct. However for a large portion of practical problems, including the current case of geographically placed sensors, it is required to either propose a sensible construction method or learn a graph from available signal data. In this paper we opt for the former, and omit a full discussion of the available techniques, which can be found in, for example, [51].

The graph construction method we use makes use of both pairwise geodesic distances between monitors and the intermediate elevation profile. This is important as environmental processes can be strongly influenced by topography, especially in mountainous regions. Elevation data is sourced from the GLOBE30 project [52]. This dataset gives the approximate height above sea level over a 30 arc-second latitude/longitude grid. While more complex models incorporating land use, prevailing wind direction etc. are possible, we opt for a simpler model for the sake of brevity.

Our first step is to create an $N \times N$ symmetric distance matrix D. We define the "distance" between two monitors D_{ij} to be weighted combination of their geodesic distance and the vertical relief along the intermediate path. This introduces a hyperparameter defining the relative importance of each component which we learn later via cross-validation. We then use the perturbed minimum spanning tree algorithm outlined in [53] to construct a sparse, fully connected graph. A representation can be found in figure VI-A.

C. Data pre-processing

Several steps were taken to pre-process the raw sensor data before feeding it into the models. Firstly, any stations with more than a total of 15 null readings due to equipment failure, over the three year period, were discarded. Any remaining missing values were interpolated linearly between the available readings. Secondly, meter readings for each of the pollutant variables, which generally have units of mass or particle count per unit volume, were transformed via $\log(1+x)$. This rectified the left skew of the readings, generally producing more symmetric aggregate histograms. The same transformation was also applied to wind speed, however pressure, humidity and temperature, which already had largely symmetric data distributions, were not transformed in this way.

In order to construct a daily set of features for wildfires, a historical list of all recorded wildfire events in California was used. This dataset includes information about the start and end date of each fire, along with the central location and the total ground area burned. For each fire we assumed that the burn rate rose and fell with a Gaussian shape, peaking at the midpoint of the burn window, such that 95% of the area burns within the specified time range. From this we estimate the total area burning on any given day in eleven different regions in California.

The final step before estimation was to transform all variables so that they were scaled and translated to achieve a unit marginal variance and zero mean. PCA dimensionality reduction was then performed on each of the feature groups individually keeping enough dimensions to preserve 90% of the total variance for that group. The resultant input data for X and Y had 1570 columns representing the dates from 2017-01-02 to 2021-04-20. Each of the N rows of Y held a time series of readings for a unique monitoring sensor with a well-defined location. The columns of X contain first the PCA components of the weather conditions, second the approximate number of acres burning in each of the eleven regions. Regression was then performed for each of the five different pollutants.

D. Results

In order to test the performance of the KGR and GLS KGR methods, we compared them against several other regression

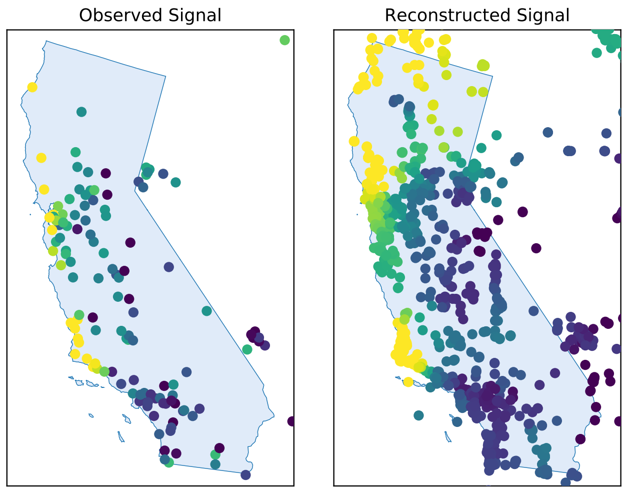

Fig. 4. Left: the observed graph signal for Ozone on a particular day represented via a colour map. Right: prediction for the reconstructed latent graph signal across the entire network on the same day made by the GLS KGR method.

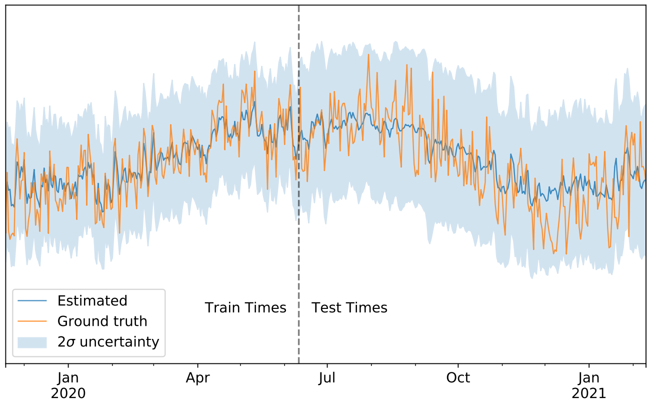

Fig. 5. A section of the Ozone time-series signal predicted by GLS KGR at a particular unobserved node is shown along with the ground truth. The blue shading depicts two standard deviations of prediction error arising from latent signal uncertainty as calculated by the Laplace approximation.

algorithms, namely Ordinary Lest Squares (OLS) regression, Ridge regression, Lasso regression. For both KGR and GLS KGR we compared both the standard feature kernel method, and the Graph Feature (GF) method described in section III-C.

For each model tested, all relevant hyperparameters were tuned using cross-validation. First, the input data was split such that the first 80% of the days served as a training/validation set and the final 20% of the days as a test set. The train/validation set was then split into four folds of 219 days each. Hyperparameters were set by attempting to minimise the mean square error averaged across each of these four folds, using three to train and one to validate accordingly. Final results were then reported on the test set.

Table III shows the mean square error as reported on the test set after hyperparameter tuning. As is visible, either GLS KGR or KGR performs the best by this metric across all pollutants. Using graph features as described in section III-C seems to

TABLE III
MEAN SQUARE ERROR ON ALL UNSEEN DATA.

	Ozone	CO	NO_2	PM2.5	PM10
GLS KGR	0.6444	0.8207	0.6912	0.7512	0.6899
KGR	0.6539	0.8438	0.7201	0.7809	0.6728
GLS KGR (GF)	0.6835	0.8602	0.7392	0.7859	0.7008
KGR (GF)	0.6706	0.8685	0.7703	0.8003	0.6957
Ridge	0.7298	0.9704	0.7316	0.7819	0.7195
Lasso	0.7283	0.9881	0.7296	0.7763	0.7222
OLS	1.1276	2.4638	1.0311	1.1715	1.1784

result in a small decrease in performance.

VII. DISCUSSION

A. Hyperparameter tuning

One weakness with KGR and GLS KGR in practice is that several hyperparameters must be tuned in order to produce an accurate model, namely α, β, γ and σ, as well as any used in the graph construction period. Each is important and can have a significant effect on the output produced. For example, insufficient regularisation via γ can cause severe over-fitting and result in unduly small entries within the estimated cross-covariance matrix Σ_N. Similarly, setting α to high can effectively drive the estimated value of θ to zero, while setting it too low can result in singular matrices if the data is at all non-stationary. This creates a large, non-convex search space which can be both costly to explore and full of local minima. The degree to which this is an issue depends largely on the size of the problem at hand. For relatively small problems, such as the one considered here with ~ 1000 nodes and ~ 1000 time steps, well-known scalar minimization algorithms such as Nelder-Mead can help automate this process. However, with significantly larger problems hyperparameters would have to be carefully selected based on domain knowledge.

B. The normalised graph Laplacian

One model choice not discussed so far is the question of whether to use the normalised or un-normalised version of the graph Laplacian. Until now, we have assumed the regular Laplacian, L, is being used, however the normalised version, defined by $\tilde{L} = D^{-1/2}LD^{-1/2}$, has properties that may make it preferable in some circumstances. For example, graph penalties constructed using $\tilde{L}$ give equal weight to all nodes, whereas penalties constructed using L implicitly favour high-degree nodes because they appear more frequently in the sum of equation 1 [24]. This may or may not be desirable depending on the problem at hand. Another potential benefit of $\tilde{L}$ is that its eigenvalues are guaranteed to fall in the interval $[0, 2]$ which makes the graph-regularisation parameter β easier to set and interpret, as well as making sensible comparison across problems possible. In the case of environmental monitoring networks our initial experiments indicate that the normalized Laplacian may result in slightly better performance, however further investigation is necessary.

C. Options for increasing scalability

In the previous analysis we assumed that, while the graph Laplacian L is often sparse, the filter and kernel matrices H and K were dense. This typically means that the primary bottle-neck for KGR and GLS KGR is eigen-decomposition of these matrices, even in their down-sampled form. For large problems it may be sufficient to only calculate the first k eigen-vectors and eigenvalues, which can be performed efficiently for sparse matrices. Therefore finding sparse representations for H and K can be highly beneficial for large applications.

One way to create a sparse graph filter is to simply use a low-degree polynomial for the filter function $\eta(\lambda)$. In this way, H can be calculated efficiently as $\eta(L)$ and decomposed faster. This strategy can also be used for K if using the graph features variant outlined in section III-C. If not, a large literature also exists on compactly supported kernels, for example, the Wendland kernel [54]. Here, an integral operator is applied recursively to Askey's truncated power functions to create a set of compactly supported radial basis functions, which are guaranteed to be smooth, continuous and positive definite. Fast algorithms for the computation of these basis functions has been outlined in [55].

VIII. Conclusions and Future Work

This paper has contributed a statistical analysis of regression methods for signals defined over networks. Drawing upon recent work on Kernel Graph Regression, Gaussian Processes over Graphs and graph signal reconstruction, we proposed method of GLS KGR and demonstrated its effectiveness on a relevant task. In particular, we addressed the situation where a partial observation of an N-node graph signal is made at a subset of T evenly spaced time-points, deriving the steps for an AR(1) autocorrelation regression model, with general cross-correlation. By assuming a matrix-normal error distribution, an algorithm was designed with $O(N^3 + T^3)$ at ecah iteration. Finally, the Laplace approximation was used to derive a lower bound for the prediction error arising from latent signal uncertainty. This was tested and shown to be effective on real data taken from a network of pollutant monitoring stations.

References

[1] 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 Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[2] X. Zhu, J. Lafferty, and Z. Ghahramani, "Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions," *AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[3] X. Dong, D. Thanou, L. Toni, M. Bronstein, and P. Frossard, "Graph signal processing for machine learning: A review and new perspectives," pp. 1–22, 2020. [Online]. Available: http://arxiv.org/abs/2007.16061

[4] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 701–710. [Online]. Available: https://doi.org/10.1145/2623330.2623732

[5] W. Huang, T. A. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, "A Graph Signal Processing Perspective on Functional Brain Imaging," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 868–885, 2018.

[6] A. Pirayre, C. Couprie, L. Duval, and J. Pesquet, "Brane clust: Cluster-assisted gene regulatory network inference refinement," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 3, pp. 850–860, 2018.

[7] R. Wagner, V. Delouille, and R. Baraniuk, "Distributed wavelet denoising for sensor networks," 01 2007, pp. 373 – 379.

[8] W. Hu, S. Chen, and D. Tian, *Graph Spectral Point Cloud Processing*. John Wiley and Sons, Ltd, 2021, ch. 7, pp. 181–219. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119850830.ch7

[9] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, p. 595–608, Aug 2016. [Online]. Available: http://dx.doi.org/10.1007/s10822-016-9938-8

[10] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Transactions on Signal Processing*, vol. 64, no. 18, pp. 4845–4860, 2016.

[11] X. Pu, S. L. Chau, X. Dong, and D. Sejdinovic, "Kernel-Based Graph Learning from Smooth Signals: A Functional Viewpoint," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 7, pp. 192–207, 2021.

[12] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," *2014 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2014*, no. 3, pp. 872–876, 2014.

[13] A. Venkitaraman, S. Chatterjee, and P. Handel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 4, pp. 698–710, 2019.

[14] a. Venkitaraman, S. Chatterjee, and P. Handel, "Gaussian Processes Over Graphs," pp. 5640–5644, 2020.

[15] Y.-C. Zhi, Y. C. Ng, and X. Dong, "Gaussian Processes on Graphs via Spectral Kernel Learning," 2020. [Online]. Available: http://arxiv.org/abs/2006.07361

[16] D. B. Dunson, H.-T. Wu, and N. Wu, "Graph based gaussian processes on restricted domains," 2021.

[17] T. Li, E. Levina, and J. Zhu, "Prediction models for network-linked data," *The Annals ofApplied Statistics 2019*, vol. 13, no. 1, pp. 132–164, 2019.

[18] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-Based Reconstruction of Graph Signals," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 764–778, 2017.

[19] Z. Xiaojin, K. Jaz, L. John, and G. Zoubin, "Graph Kernels by Spectral Transforms," *Semi-Supervised Learning*, pp. 276–291, 2013.

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

[21] A. Venkitaraman, H. P. Maretic, S. Chatterjee, and P. Frossard, "Supervised Linear Regression for Graph Learning from Graph Signals," pp. 1–19, 2018. [Online]. Available: http://arxiv.org/abs/1811.01586

[22] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology Identification and Learning over Graphs: Accounting for Nonlinearities and Dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[23] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning Graphs From Data: A Signal Representation Perspective," *IEEE Signal Processing Magazine*, vol. 36, no. May, pp. 44–63, 2019.

[24] F. Fouss, M. Saerens, and M. Shimbo, *Algorithms and Models for Network Data and Link Analysis*, 2016.

[25] J. M. Moura, "Graph Signal Processing," in *Cooperative and Graph Signal Processing: Principles and Applications*, P. M. Djurić and C. Richard, Eds., 2018, pp. 239–259.

[26] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[27] N. Tremblay, P. Gonçalves, and P. Borgnat, "Design of Graph Filters and Filterbanks," in *Cooperative and Graph Signal Processing: Principles and Applications*, P. M. Djurić and C. Richard, Eds., 2018, pp. 299–324.

[28] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, Jan 2020. [Online]. Available: http://dx.doi.org/10.1007/s41109-019-0195-3

[29] S. P. Chepuri and G. Leus, "Graph sampling for covariance estimation," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 451–466, 2017.

[30] V. N. Ioannidis, D. Romero, and G. B. Giannakis, "Inference of Spatio-Temporal Functions over Graphs via Multikernel Kriged Kalman Filtering," *IEEE Transactions on Signal Processing*, vol. 66, no. 12, pp. 3228–3239, 2018.

[31] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*, 2012.

[32] D. Romero, V. N. Ioannidis, and G. B. Giannakis, “Kernel-Based Reconstruction of Space-Time Functions on Dynamic Graphs,” *IEEE Journal on Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 856–869, 2017.

[33] S. Shekkizhar and A. Ortega, “Graph Construction from Data by Non-Negative Kernel Regression,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, no. 1, pp. 3892–3896, 2020.

[34] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2010. [Online]. Available: https://books.google.co.uk/books?id=A3ISEAAAQBAJ

[35] C. A. R. de Sousa, S. O. Rezende, and G. E. A. P. A. Batista, “Influence of graph construction on semi-supervised learning,” in *ECML/PKDD*, 2013.

[36] D. N. Naik and S. S. Rao, “Analysis of multivariate repeated measures data with a Kronecker product structured covariance matrix,” *Journal of Applied Statistics*, vol. 28, no. 1, pp. 91–105, 2001.

[37] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed., 2013, ch. 7, pp. 486–487.

[38] P. Dutilleuel, “The MLE algorithm for the matrix normal distribution,” *Journal of Statistical Computation and Simulation*, vol. 64, no. 2, pp. 105–123, 1999.

[39] A. Roy and R. Khattree, “On implementation of a test for Kronecker product covariance structure for multivariate repeated measures data,” *Statistical Methodology*, vol. 2, no. 4, pp. 297–306, 2005.

[40] ——, “Testing the Hypothesis of a Kroneckar Product Covariance Matrix in Multivariate Repeated Measures Data,” *Sugi 30*, no. 199-30, pp. 1–11, 2005.

[41] M. S. Srivastava, T. von Rosen, and D. von Rosen, “Models with a Kronecker product covariance structure: Estimation and testing,” *Mathematical Methods of Statistics*, vol. 17, no. 4, pp. 357–370, 2008.

[42] I. Soloveychik and D. Trushin, “Gaussian and robust Kronecker product covariance estimation: Existence and uniqueness,” *Journal of Multivariate Analysis*, vol. 149, no. 1, pp. 92–113, 2016.

[43] J. Campbell, J. Champbell, J. Campbell, A. LO, P. Lo, A. Lo, A. MacKinlay, A. MacKinlay, and M. C, *The Econometrics of Financial Markets*, ser. Book collections on Project MUSE. Princeton University Press, 1997.

[44] D. Cochrane and G. H. Orcutt, “Application of Least Squares Regression to Relationships Containing Auto- Correlated Error Terms,” *American Statistic Association*, vol. 44, no. 245, pp. 32–61, 1949.

[45] T. Kariya and H. Kurata, *Generalized Least Squares*. Wiley, 2004, vol. 7.

[46] D. Peña and J. Rodríguez, “The log of the determinant of the autocorrelation matrix for testing goodness of fit in time series,” *Journal of Statistical Planning and Inference*, vol. 136, no. 8, pp. 2706–2718, 2006.

[47] O. Ledoit and M. Wolf, “A well-conditioned estimator for large-dimensional covariance matrices,” *Journal of Multivariate Analysis*, vol. 88, no. 2, pp. 365–411, 2004.

[48] Y. Chen, A. Wiesel, Y. C. Eldar, and A. O. Hero, “Shrinkage algorithms for MMSE covariance estimation,” *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5016–5029, 2010.

[49] “Us environmental protection agency. air quality system data mart,” https://www.epa.gov/airdata, accessed November 20, 2020.

[50] “The department of forestry and fire protection,” https://www.fire.ca.gov/incidents/, accessed November 20, 2020.

[51] L. Qiao, L. Zhang, S. Chen, and D. Shen, “Data-driven graph construction and graph learning: A review,” *Neurocomputing*, vol. 312, pp. 336–351, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218306696

[52] D. A. Hastings, P. K. Dunbar, G. M. Elphingstone, M. Bootz, H. Murakami, H. Maruyama, H. Masaharu, P. Holland, J. Payne, N. A. Bryant, T. L. Logan, J. P. Muller, G. Schreier, and J. S., “The global land one-kilometer base elevation (globe) digital elevation model, version 1.0,” http://www.ngdc.noaa.gov/mgg/topo/globe.html, 1999, accessed November 20, 2020.

[53] R. Zemel and M. Carreira-Perpiñán, “Proximity graphs for clustering and manifold learning,” in *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., vol. 17. MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper/2004/file/dcda54e29207294d8e7e1b537338b1c0-Paper.pdf

[54] H. Wendland, “Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree,” *Advances in Computational Mathematics*, vol. 4, no. 1, pp. 389–396, 1995.

[55] S. Zhu, “Compactly supported radial basis functions: How and why?” Tech. Rep., 2012.