sciendo

CAIM
ISSN 2038-0909

Commun. Appl. Ind. Math. 15 (1), 2024, 66–85 DOI: 10.2478/caim-2024-0004

Research Article

Exploration of Kernel Parameters in Signal GBF-PUM Approximation on Graphs

R. Cavoretto[1], A. De Rossi[1*], S. Mereu[2]

[1]Department of Mathematics "Giuseppe Peano", University of Torino, Torino, Italy
[2]Italian Institute of Technology, Genova, Italy

[*]Email address for correspondence: alessandra.derossi@unito.it

Communicated by Renato Spigler
Received on 05 31, 2024. Accepted on 06 11, 2024.

Abstract

The application of the Partition of Unity Method (PUM) to signal approximation on graphs represents a recent advancement of this versatile and efficient interpolation technique. Given the novelty of this approach, little is yet known regarding the role of kernel parameters employed in constructing the associated Graph Basis Functions (GBFs). In order to shed light on this aspect, this study proposes several numerical tests obtained using GBFs generated by heat kernels and variational spline kernels.

Keywords: `Graph Interpolation, Kernel Methods, Graph Basis Functions, Partition of Unity Methods, Approximation Algorithms`

AMS subject classification: `65S99, 05C90, 68W25`

1. Introduction

The interest in the field of signal sampling and approximation on graphs has been growing in recent years [1–7] due to the numerous possible applications, ranging from retrieving information on web traffic signals [8] to analyzing road traffic in urban areas of arbitrarily complex shapes [9]. Among the various methods proposed in recent years, the Partition of Unity Method (PUM) stands out for its computational efficiency and versatility [10,11]. Thanks to the approach derived from kernel-based interpolation [12–15] and the efficiency of signal reconstruction from local approximations, computation times and errors produced are extremely limited [10,13]. Moreover, this holds true even in the case of highly intricate networks. In this study, the Minnesota graph is examined as an exemplary instance of formidable complexity and size. This graph boasts 2642 vertices and 3304 edges, and the experiments conducted on it are readily comparable to others found in the literature [10,11]. The choice of kernels used for constructing the Graph Basis Functions (GBFs) [16] is a crucial step in implementing a PUM algorithm. We have opted to experiment with two of the most promising functions in the literature, i.e. the variational spline [17] and heat kernels [18,19]. Still, very little is known about the relationship between the parameters of these kernels and the efficiency of the method. Therefore, after a brief introduction to the fundamental concepts, we present interpolation errors and computation times observed as specific kernel parameters vary, revealing interesting novel insights into the behavior of the interpolant. A comparison between the two kernels is performed as well.

The paper is organized as follows. Section 2 contains the preliminaries about graph theory. Section 3 introduces kernel-based interpolation, while Section 4 focuses on graph-signal interpolation with positive definite GBFs. Section 5 describes the PUM and the algorithms implemented. In Section 6 a large and extensive numerical investigation about the interpolant behaviour depending on the kernel parameters is presented. The paper concludes with Section 7, where we summarize our conclusions and propose directions for future work, setting the stage for subsequent advancements in the field.

© 2024 R. Cavoretto, A. De Rossi, S. Mereu, licensee Sciendo.
This work is licensed under the Creative Commons Attribution 4.0 International Public License (CC BY 4.0).

2. Preliminaries on graph theory

In this section, basic concepts on graph theory will be briefly introduced. Interested readers may refer to [20–22] for further insights.

Definition 2.1 (Graph). *A graph is defined as a set of vertices V (or nodes) and pairs $B \in V \times V$, called edges or links, which represent the connections between vertices.*

Definition 2.2 (Directed and Undirected Graphs). *A graph is termed "undirected" if the presence of the element (m, n) in the set of edges $B \in V \times V$ implies the presence of the element (n, m). Graphically, this means that the segments connecting the vertices are not oriented.*

Definition 2.3 (Adjacency Matrix). *Given a graph G with vertices V and edges B, the adjacency matrix of G is the matrix A with entries:*

$$A_{m,n} = \begin{cases} 1 & \text{if } (m,n) \in B, \\ 0 & \text{if } (m,n) \notin B. \end{cases} \tag{1}$$

For a given set V, a graph with vertices V is completely determined by its adjacency matrix. In the case of an undirected graph, the adjacency matrix is symmetric:

$$A = A^T. \tag{2}$$

If the graph is weighted, additional concepts can be introduced.

Definition 2.4 (Weight Matrix). *Let G be a graph with $N > 0$ vertices. The weight matrix of G is defined as the matrix $W \in M(N, N)$ with entries $W_{m,n}$ corresponding to the weights between the edges m and n.*

In this more general case, an element $W_{m,n} = 0$ in the matrix W indicates that the edge between vertices m and n does not exist, while a non-zero element of the matrix describes the edge between the two vertices and its corresponding weight. Similarly to the matrix A, W is also symmetric for undirected graphs:

$$W = W^T.$$

Thanks to these definitions, it is possible to transform any instance of an unweighted graph into a weighted graph, where every non-zero weight is set to 1.

Now, we can proceed to define the degree matrix and the Laplacian.

Definition 2.5 (Vertex Degree). *Given a vertex $v \in V$, the degree of v is defined as the number of edges connected to it.*

Definition 2.6 (Degree Matrix). *The degree matrix for an undirected graph is defined as the diagonal matrix D with entries:*

$$D_{m,m} = \sum_n W_{m,n}. \tag{3}$$

If the graph is unweighted, $D_{m,m}$ is simply equal to the number of edges connected to the m-th vertex.

Definition 2.7 (Laplacian Matrix). *The Laplacian matrix (or simply Laplacian) L of a graph is defined as the matrix obtained by subtracting the degree matrix from the weight matrix:*

$$L = D - W.$$

For undirected graphs, L is symmetric.

Definition 2.8 (Normalized Laplacian). *The normalized Laplacian matrix for a graph is defined as:*

$$L_N = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}.$$

2.1. *Signals and spectral theory overview*

Given a graph G with the set of vertices V, a signal on G is a real or complex-valued function $x : V \to \mathbb{R}$. Since the vertices V are ordered, the signal can be vectorized as follows:

$$x = [x(v_1), x(v_2), \ldots, x(v_n)]^T.$$

The set of such functions on a graph forms an n-dimensional vector space denoted as $\mathcal{L}(G)$, on which it is natural to define the inner product:

$$y^T x := \sum_{i=1}^{n} x(v_i)y(v_i),$$

where $x, y \in \mathcal{L}(G)$, and $v_i \in V$ for all $i \in \{1, \ldots, n\}$. An orthonormal basis for the space $\mathcal{L}(G)$ is formed by the vectors $\{\delta_{v_1}, \ldots, \delta_{v_n}\}$, where $\delta_{v_j}(v_i) = \delta_{i,j}$ for $i, j \in \{1, \ldots, n\}$. There are two approaches to address the Fourier transform of a signal on a graph: decomposing the signal through the eigenvectors of either the adjacency matrix or the Laplacian.

2.2. *The Fourier transform via the adjacency matrix*

Definition 2.9 (Graph Fourier Transform). *The Fourier transform of a graph signal x is defined as:*

$$\hat{x} = U^{-1}x, \tag{4}$$

where U is the matrix composed of the eigenvectors of the adjacency matrix (arranged in columns).

If $U^{-1} = U^T$, the k-th element of $\hat{x}$, denoted $\hat{x}(k)$, is a projection of the signal onto the k-th eigenvector:

$$\hat{x}(k) = \sum_{n=0}^{N-1} x(n)u_k(n). \tag{5}$$

Thus, the Fourier transform on graphs can be interpreted as a decomposition of the signal onto the set of eigenvectors, which form an orthonormal basis.

Definition 2.10 (Inverse Graph Fourier Transform). *The inverse Fourier transform of a graph signal is defined as the function:*

$$x = U\hat{x}, \tag{6}$$

where U is the matrix composed of the eigenvectors of the adjacency matrix and $\hat{x}$ is the Fourier transform of the signal x.

Analogously to the previous case, we can write:

$$x(n) = \sum_{n=0}^{N-1} \hat{x}(k)u_k(n). \tag{7}$$

2.3. *The Fourier Transform via the Laplacian*

Analogously to what was seen in the previous section, it is also possible to perform the spectral decomposition of a signal on a graph using the Laplacian $L = U\Lambda U^{-1}$.

Definition 2.11 (Fourier Transform via the Laplacian). *The Fourier transform of a signal x on a graph G via the Laplacian is defined as the vector:*

$$\hat{x} = U^{-1}x,$$

where U is the matrix composed of the eigenvectors of the Laplacian (arranged in columns).

The inverse transform can be introduced easily as well.

Definition 2.12 (Inverse Fourier Transform via the Laplacian). *Let x be a signal on a graph G, and let $\hat{x}$ be its Fourier transform computed through the spectral decomposition of the Laplacian. The inverse Fourier transform of $\hat{x}$ is:*

$$x = U\hat{x},$$

where U is the matrix composed of the eigenvectors of the Laplacian.

Further information on graph theory is available in [21–23].

2.4. *Convolution*

The Fourier transform allows us to define the convolution operation between signals on a graph.

Definition 2.13 (Convolution product). *Let $x, y \in \mathcal{L}(G)$ be signals on the graph G. The convolution operation between x and y is defined as follows:*

$$x * y := U(M_{\hat{x}}\hat{y}) = UM_{\hat{x}}U^T y,$$

where $M_{\hat{x}}$ is the diagonal matrix $M_{\hat{x}} = diag(\hat{x})$.

2.4.1. *Properties*

The following properties hold for the convolution product on graphs:

- $x * y = y * x$,
- $(x * y) * z = x * (y * z)$,
- $(x + y) * z = x * z + (y * z)$,
- $(\alpha x) * y = \alpha(y * x), \quad \forall \alpha \in \mathbb{R}$.

The identity element for this operation is the element $f_{\mathbb{1}} = \sum_{i=1}^{n} u_i$. An operator of convolution C_x can be defined as follows:

$$C_x = UM_{\hat{x}}U^T.$$

Remark 2.1. If the eigenvalues of the Laplacian are not simple, the eigenvectors $\hat{G} = \{u_1, \ldots, u_n\}$ are not uniquely determined. This ambiguity is problematic for the definition of convolution. To overcome the problem, two solutions are possible:

i) Assume that the set of eigenvectors $\hat{G}$ is fixed in advance;
ii) Define convolution uniquely on functions x that have an invariant action on the Laplacian's eigenspace of the graph.

3. Kernel-based interpolation on graphs

In order to introduce the PUMs, it is useful to provide a brief introduction to the theory of kernel-based interpolation. A more in-depth review of the topic can be found in [24]. For these techniques, symmetric functions $K : V \times V \to \mathbb{R}$ on a graph G are used, meaning that $K(v, w) = K(w, v)$ for all $v, w \in V$. Such a kernel allows to introduce a linear operator $\mathbf{K} : \mathcal{L}(G) \to \mathcal{L}(G)$ as follows:

$$\mathbf{K}x(v_i) = \sum_{j=1}^{n} K(v_i, v_j)x(v_j),$$

where x is a signal in the space $\mathcal{L}(G)$. The operator $\mathbf{K}$ can be represented by the symmetric matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$:

$$\mathbf{K} = \begin{pmatrix} K(v_1,v_1) & K(v_1,v_2) & \dots & K(v_1,v_n) \\ K(v_2,v_1) & K(v_2,v_2) & \dots & K(v_2,v_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(v_n,v_1) & K(v_n,v_2) & \dots & K(v_n,v_n) \end{pmatrix}. \tag{8}$$

Now, it is necessary to define positive definite, semi-positive definite, and conditionally positive definite kernels.

Definition 3.1 (Positive Definite Kernel). *A symmetric kernel K is positive definite if the matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ is strictly positive definite, i.e., if $x^T\mathbf{K}x > 0$ for all $x \in \mathbb{R}^n, x \neq 0$.*

Definition 3.2 (Semi-Positive Definite Kernel). *A symmetric kernel K is semi-positive definite (p.s.d.) if the matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ is semi-positive definite, i.e., if $x^T\mathbf{K}x \geq 0$ for all $x \in \mathbb{R}^n$.*

Definition 3.3 (Conditionally Positive Definite Kernel). *A symmetric kernel K is called conditionally positive definite with respect to a subspace $\mathcal{Y} \subset \mathcal{L}(G)$ if it is positive definite on $\mathcal{Y}$.*

By means of every positive definite kernel, it is possible to endow the space $\mathcal{L}(G)$ with the following inner product:

$$\langle x, y\rangle_K = y^T\mathbf{K}^{-1}x, \quad x, y \in \mathcal{L}(G).$$

In the literature the resulting space is often called the *native space* and denoted as $\mathcal{N}_K$. This is a Hilbert space with reproducing kernels, where K is precisely the reproducing kernel. It holds the property:

$$\langle x, K(\cdot, v_j)\rangle = x^T\mathbf{K}^{-1}K(\cdot, v_j) = x(v_j), \quad \text{for every } x \in \mathcal{L}(G).$$

A positive definite kernel K can be used to solve interpolation problems in spaces generated by a linear combination of columns of the matrix $\mathbf{K}$ calculated at interpolation nodes.

3.1. *Interpolation via positive definite kernels*

Given signals $x(w_1), \dots, x(w_N)$ on a subset $W = \{w_1, \dots, w_N\} \subset V$, with $N \leq n$, we want to find an interpolating signal $I_W x \in \mathcal{L}(G)$ that satisfies the condition:

$$I_W x(w_k) = x(w_k), \quad \text{for every } k \in \{1, \dots, N\}. \tag{9}$$

at nodes in W. A positive definite kernel can be applied to solve this problem. Considering as a basis for interpolating functions the columns of the matrix $\mathbf{K}$ in (8), the interpolant $I_W x$ has the form:

$$I_W x(v) = \sum_{k=1}^{N} c_k K(v, w_k),$$

where the coefficients are found by solving the linear system

$$\underbrace{\begin{pmatrix} K(w_1,w_1) & K(w_1,w_2) & \dots & K(w_1,w_n) \\ K(w_2,w_1) & K(w_2,w_2) & \dots & K(w_2,w_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(w_N,w_1) & K(w_N,w_2) & \dots & K(w_N,w_n) \end{pmatrix}}_{\mathbf{K}_W} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} x(w_1) \\ x(w_2) \\ \vdots \\ x(w_N) \end{pmatrix}. \tag{10}$$

Since $\mathbf{K}$ is positive definite, the submatrix $\mathbf{K}_W$ is positive definite by the inclusion principle (for further details, see [25] Theorem 4.3.15). Then the system (10) has a unique solution, and the signal $I_W x$ can be uniquely written using the introduced basis. The resulting interpolation space is given by

$$\mathcal{N}_{K,W} = \{x \in \mathcal{L}(G) | x(v) = \sum_{k=1}^{N} c_k K(v, w_k)\}.$$

The following result quantifies the goodness of approximation and justifies the popularity of Hilbert spaces with reproducing kernels in the field of signal approximation.

Theorem 3.1. *The interpolant $I_W x$ of x makes the norm of the native space $||\cdot||_K$ minimum with respect to any possible interpolant of x in $\mathcal{L}(G)$ on the nodes W.*

Additional results justifying the popularity of kernel-based interpolation can be found in [26].

3.2. *Interpolation via conditionally positive definite kernels*

Let K be a conditionally positive definite kernel with respect to a subspace $\mathcal{Y}$. Suppose $Y^{\perp}$, the orthogonal complement, has an orthonormal basis $\{y_1^{\perp}, \ldots, y_M^{\perp}\}$ with M sufficiently small. Then the augmented kernel can be defined as:

$$K^{(\delta)}(v, w) = K(v, w) + \delta \left(\sum_{i=1}^{M} y_i^{\perp}(v) y_i^{\perp}(w) \right).$$

If the relationship $\delta > |\lambda_{min}(\mathbf{K})| \geq 0$ holds, where λ_{min} is the smallest eigenvalue of $\mathbf{K}$, then the kernel $K^{(\delta)}$ is everywhere positive definite and the previously illustrated procedure can be applied. Then it is possible to find an interpolation signal $x \in \mathcal{N}_{K^{(\delta)},W}$ such that (9) is satisfied. The interpolant $I_W x(v)$ has the form:

$$I_W x(v) = \sum_{k=1}^{N} c_k K(v, w_k) + \sum_{i=1}^{M} d_i y_i^{\perp}(v), \quad \text{with } d_i = \delta \sum_{k=1}^{N} c_k y_k^{\perp}(w_k),$$

where the coefficients c_k are the solution of (10) with respect to the kernel $K^{(\delta)}$.

4. Graph signal interpolation with positive definite GBFs

This section will briefly cover GBFs and their application in signal approximation on a graph. These methods represent the next step in defining the PUM.

Definition 4.1 (Positive Definite GBF). *Consider a graph G with the set of vertices $V = \{v_1, \ldots, v_n\}$ and edges $B \subseteq V \times V$. A GBF is defined as positive definite if the matrix*

$$K_f = \begin{pmatrix} C_{\delta_{v_1}} f(v_1) & C_{\delta_{v_2}} f(v_1) & \ldots & C_{\delta_{v_n}} f(v_1) \\ C_{\delta_{v_1}} f(v_2) & C_{\delta_{v_2}} f(v_2) & \ldots & C_{\delta_{v_n}} f(v_2) \\ \vdots & \vdots & \ddots & \vdots \\ C_{\delta_{v_1}} f(v_n) & C_{\delta_{v_2}} f(v_n) & \ldots & C_{\delta_{v_n}} f(v_n) \end{pmatrix} \tag{11}$$

is symmetric and positive definite. Here, $\{\delta_{v_1}, \ldots, \delta_{v_n}\}$ denotes the standard unit basis of $\mathcal{L}(G)$, while the convolution operator C in (11) is the operator defined in Subsection 2.4.

Property 4.1.

(a) A GBF f is positive definite if and only if $\hat{f}_k > 0$ for all $k \in \{1, \ldots, n\}$.

(b) The kernel $K_f(v,w) := C_{\delta_w} f(v)$ can be decomposed by Mercer's theorem into orthonormal eigenvectors:

$$K_f(v,w) = C_{\delta_w} f(v) = \sum_{k=1}^{n} \hat{f}_k u_k(v) u_k(w).$$

(c) The operator K_f can be written as:

$$K_f = U M_{\hat{f}} U^T.$$

(d) A positive definite GBF induces an inner product as follows:

$$\langle x, y \rangle K_f = \sum_{k=1}^{n} \frac{\hat{x}_k \hat{y}_k}{\hat{f}_k} = \hat{y}^T M_{\frac{1}{\hat{f}}} \hat{x}, \tag{12}$$

and a norm:

$$||x||K_f := \sqrt{\sum_{k=1}^{n} \frac{\hat{x}_k^2}{\hat{f}_k}}.$$

With the previously defined inner product, the space $\mathcal{L}(G)$ becomes a Hilbert space with reproducing kernels.

4.1. *Examples of positive definite GBFs*

Two notable examples of positive definite functions on graphs are presented below:

I) *Variational splines* on graphs are based on the following kernel:

$$K_{f_{(\epsilon I_n + L)^{-s}}} = (\epsilon I_n + L)^{-s} = \sum_{k=1}^{n} \frac{1}{(\epsilon + \lambda_k)^s} u_k u_k^T. \tag{13}$$

For $\epsilon > -\lambda_1$ and $s > 0$, (13) is positive definite. Variational spline kernels are presented in [27] as interpolants that minimize the energy functional $x^T\ (\epsilon I_n + L)^s x$. They can be used as GBF interpolants based on the function with Fourier transform:

$$\hat{f}_{(\epsilon I_n + L)^{-s}} = \left(\frac{1}{(\epsilon + \lambda_1)^s}, \ldots, \frac{1}{(\epsilon + \lambda_n)^s} \right).$$

II) *Heat* (or *diffusion*) kernels are defined on a graph through Mercer's decomposition:

$$K_{f_{e^{-tL}}} = e^{-tL} = \sum_{k=1}^{n} e^{-t\lambda_k} u_k u_k^T, \tag{14}$$

where $\lambda_k, k \in \{1, \ldots, n\}$ are the eigenvalues of the Laplacian of the considered graph. This kernel is positive definite for every $t \in \mathbb{R}$. The corresponding GBF is uniquely determined by the Fourier transform

$$\hat{f}_{e^{-tL}} = (e^{-t\lambda_1}, \ldots, e^{-t\lambda_n}).$$

4.2. *Error estimation*

In this paragraph, we will use the notation chosen in [13]: $I_W x$ denotes the GBF interpolant of the signal x at nodes W. The space of interpolating GBFs is generated by the translates $C_{e_{j_k}}$ of the GBFs:

$$\mathcal{N}_{K_f,W} = \left\{ x \in \mathcal{L}(G) | x = \sum_{k=1}^{N} c_k C_{e_{j_k}} f \right\}.$$

The inner product and norm on this space are:

$$\langle x, y\rangle_{K_f} = \sum_{k=1}^{n} \frac{\hat{x}_k \hat{y}_k}{\hat{f}_k} = \hat{y}^T M_{1/\hat{f}} \hat{x} \quad \text{and} \quad ||x||_{K_f} = \sqrt{\sum_{k=1}^{n} \frac{\hat{x}_k^2}{\hat{f}_k}}.$$

To estimate the error $|x(v) - I_W x(v)|$, it is necessary to introduce the *norming sets*.

Let $\mathcal{L}(G)$ be an auxiliary space, consisting of signals $\mathcal{B}_M$ with limited bandwidth on G, and S_W and B_M be the following projection operators on $\mathcal{L}(G)$:

$$S_W x(v) = \begin{cases} x(v) \text{ if } v \in W, \\ 0 \quad \text{otherwise,} \end{cases} \qquad \text{and} \quad B_M x = \sum_{k=1}^{M} (u_k^T x) u_k.$$

The map S_W maps the signal x onto $W \subset V$, while B_M is a projection operator onto the space $\mathcal{B}_\mathcal{M}$ of functions with limited bandwidth.

Definition 4.2 (Norming Set). *A norming set of the subspace $\mathcal{B}_\mathcal{M} \subset \mathcal{L}(G)$ is defined as a subset $W = \{w_1, \ldots, w_N\}$ of V such that the operator $S_W B_M$ is injective on $\mathcal{B}_\mathcal{M}$.*

The requirement of injectivity is due to the necessity that the inverse $(S_W B_M)^{-1}$ be well-defined on the image $S_W(B_M) = S_W B_M(\mathcal{L}(G))$. Moreover, the *norming constant* of the set W is defined as the operator norm:

$$||(S_W B_M)^{-1}|| = \sup_{z \in S_W(B_M), ||z|| \leq 1} ||(S_W B_M)^{-1} z||.$$

There exists a simple criterion to determine whether W is a *norming set* of $\mathcal{B}_\mathcal{M}$.

Theorem 4.2. *The set W is a norming set of $\mathcal{B}_\mathcal{M}$ if and only if the spectral norm of the matrix $B_M(I_n - S_W)B_M$ is strictly less than 1. The norming constant of the set W is bounded as follows:*

$$||(S_W B_M)^{-1}|| \leq \frac{1}{1 - ||B_M(I_n - S_W)B_M||}.$$

Now we can introduce the main error result presented in [13].

Theorem 4.3. *Let $f \in P_+$ (space of positive definite functions) and $W \subset V$ be a norming set for the space $\mathcal{B}_\mathcal{M}$ on graph G. Then the following error estimation holds for the GBF interpolation of a signal x using the interpolant $I_W x \in \mathcal{N}_{W,K_f}$:*

$$\max_{v \in V} |x(v) - I_W x(v)| \leq (1 + ||(S_W B_M)^{-1}||) \left(\sum_{k=M+1}^{n} \hat{f}_k \right)^{\frac{1}{2}} ||x||_{K_f}.$$

The error estimation depends on three factors: the constant $||(S_W B_M)^{-1}||$, the sum $\sum_{k=M+1}^{n} \hat{f}_k$, and the norm of the native space $||x||_{K_f}$. Such factors, in turn, depend on the set W, the bandwidth M, the Fourier coefficients $\hat{f}_k$, and the signal x. In particular, the choice of f and W should take into account the following points:

1. The set W and the bandwidth M determine the constant $||(S_W B_M)^{-1}||$. If the complexity of W is excessively reduced or if the nodes in W do not effectively separate the signals in the space B_M, the norming constant could be high or even infinite. Therefore, the choice of W should take into account the value of $||(S_W B_M)^{-1}||$ for large M.
2. As M increases, the value of $||(S_W B_M)^{-1}||$ increases while $\sum_{k=M+1}^{n} \hat{f}_k$ decreases. Hence, there is a trade-off between these two values in terms of bandwidth M.

3. For Fourier coefficients with rapid decay $\hat{f}_k$, the sum $\sum_{k=M+1}^{n} \hat{f}_k$ exhibits the same behavior. However, if the signal x is not sufficiently regular, Fourier series with rapid decay can lead to rather high norms $||x||_{K_f}$. The choice of the appropriate GBF f must therefore take into account this trade-off.

Then, the following result holds:

Corollary 4.4. *Under the same assumptions as Theorem 4.3, the following estimates hold:*

- *If* $\hat{f}_k \leq C_1 k^{-s}, s > 1$, *then:*

$$\max_{v \in V} |x(v) - I_W x(v)| \leq \sqrt{\frac{C_1}{s-1}} (1 + ||(S_W B_M)^{-1}||) M^{-\frac{s-1}{2}} ||x||_{K_f}.$$

- *If* $\hat{f}_k \leq C_2 e^{-tk}$, $t > 0$, *then:*

$$\max_{v \in V} |x(v) - I_W x(v)| \leq \sqrt{\frac{C_2}{1-e^{-t}}} (1 + ||(S_W B_M)^{-1}||) e^{-\frac{t}{2}(M+1)} ||x||_{K_f}.$$

5. The partition of unity method on graphs

The PUM is a rather effective tool when dealing with extensive datasets. The idea of the method is to partition the closed and bounded domain under examination into specific subdomains or communities in which local interpolants are sought. Through the use of local interpolants and a given partition of unity (a family of functions with particular properties), the global interpolant is obtained [10].

5.1. *Graph partition*

In order to apply the PUM to a connected graph, it is necessary to divide it into a suitable set of subdomains. In this section, an algorithm for finding the appropriate subdomains for the PUM will be presented. The general problem of ideal graph partitioning into subgraphs and community detection is an open problem not addressed in many articles. One interesting solution is provided in [11], while the interested reader may find an overview of the topic in [28]. In this section, we will briefly only present the method proposed in [10] and based on the works [29,30]. The chosen strategy is as follows. Given a connected graph G, we want to obtain J subgraphs C_j, $j = \{1, \ldots, J\}$ such that:

$$V = \bigcup_{j=1}^{J} C_j, \quad C_j \cap C_{j'} = \emptyset, \quad \forall j' \neq j \in \{1, \ldots, J\}.$$

Now, considering a metric d on the graph, we need to find j reference nodes for the partitions C_j, called *centers*. In j-Center Clustering (see [31]), the centers are the points q_j that minimize the fill distance:

$$h(Q_J) = \max_{i \in \{1,\ldots,n\}} \min_{j \in \{1,\ldots,J\}} d(q_j, v_j),$$

where d is the distance between points on the graph. The described approach has considerable computational complexity (finding the nodes Q_J that minimize the fill-distance is an NP-Hard problem [30]). To overcome this problem, a simpler greedy algorithm can be used to identify the nodes.

Given a pre-existing set of nodes $Q_{j-1} = \{q_1, \ldots, q_{j-1}\}$, we find the next value q_j such that the distance between Q_{j-1} and q_j is maximized. There is at least one interpolation node of W in each partition C_j. Algorithm 1 in [10] describes in detail the procedure above, while its MATLAB implementation is available in [32]. Further details to determine necessary conditions for the above algorithm and provide a graph partition sufficiently close to the optimal one can be found in [10, Theorem 5.1].

In order to avoid to lose the information related to the topology of the considered graph, it is not appropriate to directly choose the C_j clusters as subdomains for the PUM. A better strategy is to

incorporate the found partitions into further subdomains $V_j, j \in \{1, \dots, J\}$. Algorithm 2 and Corollary 1 in [10] show a possible procedure for this enlargement of the sets C_j. In particular, given in input a partition C_j and a distance $r > 0$, the algorithm adds each node $v \in V$ with distance on the graph less than r from any element of C_j, giving rise to the subdomain V_j:

$$V_j = C_j \cup \{v \in V | d(v, w) \leq r, w \in C_j\},$$

returning the subdomain V_j.

5.2. *PUM for GBF approximation on graphs*

Here, we report the approximation algorithm using the PUM illustrated in [10]. Given a positive definite GBF f and the kernel

$$K_f(v, w) = C_{\delta_w} f(v),$$

to realize a GBF approximation of a signal x of which the values $x(w_i),\ i \in \{1, \dots, n\}$, are known. We use the following functional, which is the solution of the regularized least squares problem:

$$x_* = \underset{y \in \mathcal{N}_{K_f}}{\operatorname{argmin}} \left(\frac{1}{N} \sum_{i=1}^{N} |x(w_i) - y(w_i)|^2 + \gamma ||y||^2_{K_f}\right), \quad \gamma > 0,$$

where γ is the regularization parameter and $\mathcal{N}_{K_f}$ is the space $\mathcal{L}(G)$ with the inner product defined in equation (12). It is recalled that $\mathcal{N}_{K_f}$ is a Hilbert space with reproducing kernel K_f. The first term of the functional ensures that the approximated values $x_*(w_i)$ are sufficiently close to the known values, while the second forces $||x_*||_{K_f}$ not to grow excessively. Thus, one can write x_* as:

$$x_*(v) = \sum_{i=1}^{N} c_i C_{\delta_{w_i}} f(v),$$

where the coefficients c_i, $i = \{1, \dots, n\}$ are obtained through the linear system:

$$\left(\underbrace{\begin{pmatrix} C_{\delta_{w_1}} f(w_1) & \dots & C_{\delta_{w_N}} f(w_1) \\ \vdots & \ddots & \vdots \\ C_{\delta_{w_N}} f(w_1) & \dots & C_{\delta_{w_N} f(w_N)} \end{pmatrix}}_{K_{f,w}} + \gamma N I_N \right) \begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} x(w_1) \\ \vdots \\ x(w_N) \end{pmatrix}. \tag{15}$$

The main submatrix $K_{f,w}$ is positive definite, hence the matrix K_f is positive definite and the system admits a unique solution.

In summary, the main steps to achieve the GBF-PUM approximation on the graph are as follows:

- The set of nodes V is decomposed into J partitions using Algorithm 1 in [10].
- Algorithm 2 in [10] is applied to enlarge C_j by adding nodes with fixed maximum distance $r > 0$ on the graph. This way, the ideal subdomains $V_j = \{v_{j_1}, \dots, v_{j_{n_j}}\}, j \in \{1, \dots, J\}$, are obtained.
- The local subgraphs $G_j = (V_j, E_j, L^{(j)}, d)$ are extracted with vertices V_j, edges $E_j = \{e_{i,i'} \in E | v_i, v_{i'} \in V_j\}$, and Laplacian

$$L^{(j)} = \begin{pmatrix} L_{j_1,j_1} & \dots & L_{j_1,j_{n_j}} \\ \vdots & \ddots & \vdots \\ L_{j_{n_j},j_1} & \dots & L_{j_{n_j},j_{n_j}} \end{pmatrix},$$

  where the latter is the main submatrix of L corresponding to the nodes of V_j. The metric on G_j is the induced metric. If necessary, the local Fourier transform of the subgraphs G_j can be computed through the decomposition of the Laplacian $L^{(j)} = U^{(j)} M_{\lambda^{(j)}} U^{(j)T}$.

- The local GBF approximants

$$x_*^{(j)}(v) = \sum_{i=1}^{N_j} c_i^{(j)} C_{w_{j_i}} f^{(j)}(v) \tag{16}$$

relative to the subgraphs G_j are computed, using a positive definite GBF approximation $f^{(j)}$ on each subgraph. The coefficients in (16) are obtained by solving the equation system:

$$(K_{f^{(j)},W_j} + \gamma N_j I_{N_j}) \begin{pmatrix} c_1^{(j)} \\ \vdots \\ c_{N_j}^{(j)} \end{pmatrix} = \begin{pmatrix} x(w_{j_1}) \\ \vdots \\ x(w_{j_{N_j}}) \end{pmatrix}$$

for the sets $W_j = W \cap V_j$ with $\gamma \geq 0$ and $N_j \geq 1$ elements. Possible kernel expansions to consider in a local form are given in (13) and (14).
- The final form of the global GBF-PUM interpolant on G is:

$$x_*(v) = \sum_{j=1}^{J} \phi^{(j)}(v) x_*^{(j)}(v), \tag{17}$$

where the functionals $(\phi^{(j)})_{j=1}^J$ are constructed in such a way that:

$$\text{supp}(\phi^{(j)}) \subseteq V_j, \phi^{(j)} \geq 0, \quad \text{and} \quad \sum_{j=1}^{J} \phi^{(j)}(v) = 1 \quad \forall\, v \in V.$$

5.3. *Computational cost and error*

In this subsection, we briefly illustrate the computational cost of the GBF-PUM that mainly depends on the calculation of the functions $C_{\delta_w} f$ and $C_{\delta_w} f^{(j)}$ and on the solution of the system (15).

As for the calculation of the basis functions, the complexity depends on the procedure for finding the kernels K_f and $K_{f^{(j)}}$. It may be necessary to perform the spectral decomposition of the Laplacian over the entire graph G, a step that would require $O(n^3)$ operations. Thus, for the PUM the required number of operations is given by $\sum_{j=1}^{J} O(n_j^3) \leq J \cdot O(\max_j(n_j)^3)$. Notice that the computational cost reduction is significant if all subdomains V_j have the same size $n_j \sim \frac{n}{J}$.

Then, for the calculation of the coefficients $(c_1, \ldots, c_N)^T$, we need to solve the system (15). In the global case, $O(N^3)$ operations are required, N being the number of interpolation nodes. Also for solving this problem, there is a reduction in computational costs with the PUM. Indeed, we have to solve J linear systems of size N_j, each requiring $O(N_j^3)$ arithmetic operations.

We can now define the approximation space naturally induced by the partition of unity [10].

Definition 5.1 (Space $\mathcal{N}^{\text{PUM}}$). *Let $\{V_j\}_{j=1}^J$ be a covering of V, and let $\{\phi^{(j)}\}_{j=1}^J$ be a unit partition based on it. Let $G_j = (V_j, E_j, L^{(j)}, d)$ be the subgraphs of G constructed through the nodes V_j, the edges $E_j = \{(v,w) \in E | v, w \in V_j\}$, and the local Laplacian $L^{(j)}$. Furthermore, let $\mathcal{N}^{(j)} \subset \mathcal{L}(G_j)$ be local approximation spaces referred to the subgraphs G_j. The space $\mathcal{N}^{\text{PUM}}$ related to the global unit partition is defined as follows:*

$$\mathcal{N}^{\text{PUM}} = \left\{ x \in \mathcal{L}(G) \;\middle|\; x = \sum_{j=1}^{J} \phi^{(j)} x^{(j)} \in \mathcal{N}^{(j)} \right\}.$$

We will use, to define the quality of the approximation, the Laplacian norm or the weighted gradient $\nabla_L x$.

Definition 5.2 (Weighted Gradient relative to the Laplacian). *Given a graph G, the weighted gradient $\nabla_L x$ defined on the edges $e_{i,i'} = (v_i, v_{i'}) \in E$ of G as:*

$$\nabla_L x(e_{i,i'}) = \sqrt{A_{i,i'}}(x(v_i) - x(v_{i'})).$$

Through this new definition, it is possible to perform a standard Laplacian decomposition of the graph L_S as follows:

$$x^T L_S x = \frac{1}{2}\sum_{i=1}^{n} \sum_{i':e_{i,i'}\in E} A_{i,i'}(x(v_i) - x(v_{i'}))^2 = (\nabla_L x)^T \nabla_L x.$$

This decomposition shows that the Laplacian L_S is semi-positive definite. In order to state the main result related to the accuracy of the PUM, we will consider the L^p norms for functions in $\mathcal{L}(G)$ and $\mathcal{L}(E)$, where E is the set of edges of G. We also introduce for functions $z \in \mathcal{L}(E)$ the norm $||z||_{L^{\infty,p}(E)}$:

$$||z||_{L^{\infty,p}(E)} := \max_{i\in\{1,\dots,n\}} \left(\sum_{i':e_{i,i'}\in E} |z(e_{i,i'})|^p \right)^{\frac{1}{p}}.$$

We can now state the following theorem [10].

Theorem 5.1. *Let $\{V_j\}_{j=1}^J$ be a covering of V and let $\{\phi^{(j)}\}_{j=1}^J$ be a relative partition of unity. Given a signal $x \in \mathcal{L}(G)$, there exists a local approximation $x_*^{(j)} \in \mathcal{L}(G_j)$ such that, given $1 \le p \le \infty$, the relation (a), or relations (a), (b), or relations (a), (b), (c) hold:*

(a) $||x - x_*^{(j)}||_{\mathcal{L}^p(G_j)} \le \epsilon_{p,0}^{(j)}$,
(b) $||\nabla_{L^{(j)}} x - \nabla_{L^{(j)}} x_*^{(j)}||_{\mathcal{L}^p(E_j)} \le \epsilon_{p,1}^{(j)}$,
(c) $||L^{(j)} x - L^{(j)} x_*^{(j)}||_{\mathcal{L}^p(G_j)} \le \epsilon_{p,2}^{(j)}$.

In particular, if (a) or (b) hold, it is assumed that the partition of unity $\{\phi^{(j)}\}_{j=1}^J$ satisfies the boundary conditions:

$$\text{(BC)}\ \ \nabla_L \phi^{(j)}(e_{i,i'}) = 0, \quad \text{for every edge } e_{i,i'} \in E,\ \text{with } v_i \in V_j \text{ and } v_{i'} \notin V_j.$$

Then the following limitations hold for the errors of $x_* = \sum_{j=1}^J \phi^{(j)} x_*^{(j)} \in \mathcal{N}^{PUM}$:

(a)' $||x - x_*||_{\mathcal{L}^p(G)} \le \sum_{j=1}^J \epsilon_{p,0}^{(j)}$ *if (a) is true,*
(b)' $||\nabla_L x - \nabla_L x_*||_{\mathcal{L}^p(E)} \le \sum_{j=1}^J (\epsilon_{p,1}^{(j)} + ||\nabla_L \phi^{(j)}||_{\mathcal{L}^{\infty,p}(E)} \epsilon_{p,0}^{(j)})$ *if (a), (b) and (BC) hold,*
(c)' $||Lx - Lx_*||_{\mathcal{L}^p(G)} \le \sum_{j=1}^J (\epsilon_{p,2}^{(j)} + ||\nabla_L \phi^{(j)}||_{\mathcal{L}^{\infty,q}(E)} \epsilon_{p,1}^{(j)} + ||L_S \phi^{(j)}||_{\mathcal{L}^\infty(G)} \epsilon_{p,0}^{(j)})$ *if (a), (b), (c) and (BC) hold.*

In (c)', $1 \le q \le \infty$ is the dual of p.

6. Numerical experiments

In this section, we present some results coming from a numerical experimentation on the Minnesota graph [33] by varying the involved parameters in the variational spline and diffusion kernels. The programs were executed on a Huawei Mate D15 Laptop with an *Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 2803 Mhz, 4 Core(s)* processor and 16 GB of RAM.

At first, we perform some experiments on the variational spline kernel similar to the one in [10], in order to find the optimal parameters for approximating the signal. In particular, we aim to investigate the role of the parameter ϵ and its relationship with the obtained errors.

To facilitate easy comparison with [10], we choose an enlargement parameter $r = 8$ and a number of subdomains $J = 8$. Tables 1-2 and Figure 1 show the Relative Root Mean Square Errors (RRMSEs) and

Relative Maximum Absolute Errors (RMAEs) obtained from the interpolation of the same signal used in [10] by varying ϵ in the range $[0.0005, 0.0020]$. The calculation of RRMSE and RMAE is performed respectively by the following rules:

$$\text{RRMSE} = \frac{\|(x - x_*)\|_2}{\max(|x|)| \cdot \sqrt{|V|}},$$

$$\text{RMAE} = \frac{\|x - x_*\|_\infty}{\|x\|_\infty},$$

where x_* represents the approximated signal, x the actual signal and $|V|$ is the number of nodes in the graph.

Number of nodes	$\epsilon = 0.0005$	$\epsilon = 0.0010$	$\epsilon = 0.0015$	$\epsilon = 0.0020$
200	4.3×10^{-2}	4.4×10^{-2}	4.5×10^{-2}	4.6×10^{-2}
600	5.9×10^{-3}	5.8×10^{-3}	5.7×10^{-3}	5.7×10^{-3}
1000	8.4×10^{-4}	8.4×10^{-4}	8.4×10^{-4}	8.4×10^{-4}
1400	3.3×10^{-4}	3.3×10^{-4}	3.3×10^{-4}	3.3×10^{-4}
1800	2.2×10^{-5}	2.6×10^{-5}	3.2×10^{-5}	3.8×10^{-5}
2200	3.6×10^{-6}	4.6×10^{-6}	5.7×10^{-6}	7.0×10^{-6}
2600	6.2×10^{-7}	7.8×10^{-7}	9.7×10^{-7}	1.2×10^{-7}

Number of nodes	$\epsilon = 0.0005$	$\epsilon = 0.0010$	$\epsilon = 0.0015$	$\epsilon = 0.0020$
200	3.8×10^{-1}	3.9×10^{-1}	4.0×10^{-1}	4.1×10^{-1}
600	8.7×10^{-2}	8.6×10^{-2}	8.4×10^{-2}	8.3×10^{-2}
1000	1.0×10^{-2}	1.0×10^{-2}	9.9×10^{-3}	9.8×10^{-3}
1400	5.4×10^{-3}	5.4×10^{-3}	5.3×10^{-3}	5.3×10^{-3}
1800	2.2×10^{-4}	2.6×10^{-4}	3.1×10^{-4}	3.8×10^{-4}
2200	3.2×10^{-5}	4.0×10^{-5}	4.9×10^{-5}	5.9×10^{-5}
2600	1.0×10^{-5}	1.3×10^{-5}	1.6×10^{-5}	1.9×10^{-5}

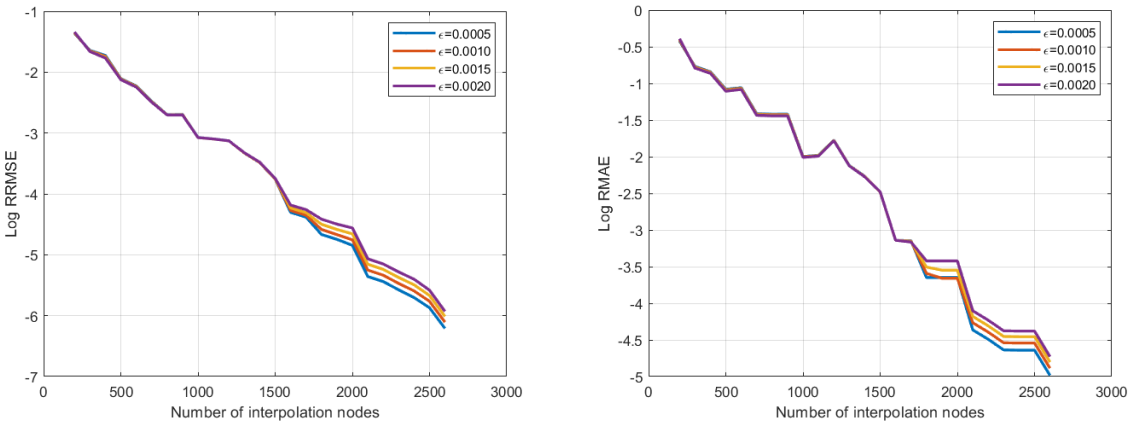

Figure 1. RRMSE (left) and RMAE (right) obtained by using the variational spline kernel in GBF-PUM algorithm with $J = 8$ subdomains, enlargement parameter $r = 8$, kernel parameters $s = 2$ and $\epsilon \in [0.0005, 0.0020]$.

It is immediately noticeable that the precision of the method increases as ϵ decreases, especially for a large number of interpolation nodes. In the range including between 200 and 1400 interpolation nodes,

the numerical results are similar; however, there is a very slight inverse trend compared to that observed with many interpolation nodes: the PUM using the kernel with a larger ϵ appears slightly more accurate. To exclude that this behavior is only local, the experiments were repeated for a wider range of ϵ, i.e. $\epsilon \in [0.0005, 0.01]$. Moreover, from Figure 1, we can clearly note a slightly higher precision for smaller values of ϵ when the number of interpolation nodes grows, i.e. for $N \geq 1800$.

Number of nodes	$\epsilon = 0.0005$	$\epsilon = 0.005$	$\epsilon = 0.007$	$\epsilon = 0.01$
200	4.3×10^{-2}	5.1×10^{-2}	5.6×10^{-2}	6.5×10^{-2}
600	5.9×10^{-3}	5.4×10^{-3}	5.5×10^{-3}	6.6×10^{-3}
1000	8.4×10^{-4}	9.6×10^{-4}	1.2×10^{-3}	1.7×10^{-3}
1400	3.3×10^{-4}	4.1×10^{-4}	5.3×10^{-4}	8.1×10^{-4}
1800	2.2×10^{-5}	1.0×10^{-4}	1.6×10^{-4}	2.8×10^{-4}
2200	3.6×10^{-6}	1.8×10^{-5}	2.9×10^{-5}	5.1×10^{-5}
2600	6.2×10^{-7}	2.9×10^{-6}	4.5×10^{-6}	7.6×10^{-6}

Number of nodes	$\epsilon = 0.0005$	$\epsilon = 0.005$	$\epsilon = 0.007$	$\epsilon = 0.01$
200	3.8×10^{-1}	4.4×10^{-1}	4.5×10^{-1}	4.5×10^{-1}
600	8.7×10^{-2}	7.3×10^{-2}	6.7×10^{-2}	5.6×10^{-2}
1000	1.0×10^{-2}	9.3×10^{-3}	9.0×10^{-3}	1.4×10^{-2}
1400	5.4×10^{-3}	5.2×10^{-3}	5.2×10^{-3}	7.7×10^{-3}
1800	2.2×10^{-4}	1.3×10^{-3}	2.1×10^{-3}	3.8×10^{-3}
2200	3.2×10^{-5}	1.4×10^{-4}	2.3×10^{-4}	4.3×10^{-4}
2600	1.0×10^{-5}	4.0×10^{-5}	5.7×10^{-5}	9.0×10^{-5}

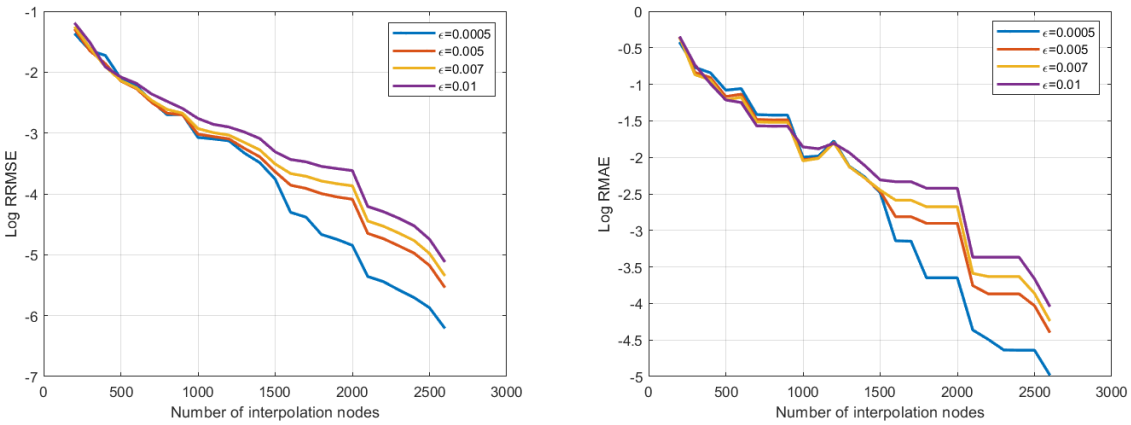

Figure 2. RRMSE (left) and RMAE (right) obtained by using the variational spline kernel in the GBF-PUM algorithm with $J = 8$ subdomains, enlargement parameter $r = 8$, kernel parameters $s = 2$ and $\epsilon \in [0.0005, 0.01]$.

A similar behavior is confirmed by experiments done for different values of $\epsilon \in [0.0005, 0.01]$ and shown in Tables 3-4. In the latter cases, in Table 5 and Figure 3 we also report the computation times (in seconds) of the approximated signals obtained for $\epsilon = 0.0005, 0.005, 0.007, 0.01$. Since these computation times may vary significantly from one routine execution to another, it is challenging to detect particular advantages in the use of one parameter over another. Nevertheless, the repeated iteration of the algorithm seems to indicate that kernels with parameters closer to zero lead to a reduction in computation time, especially in the case of large numbers of interpolation nodes. The general trend is essentially the same as already described in [10]. Obviously, longer computation times are observed for a larger number of nodes.

Number of nodes	$\epsilon = 0.0005$	$\epsilon = 0.001$	$\epsilon = 0.0015$	$\epsilon = 0.0020$
200	7.2×10^{-1}	7.8×10^{-1}	7.9×10^{-1}	$1.1 \times 10^{+0}$
600	7.6×10^{-1}	8.5×10^{-1}	7.9×10^{-1}	$1.2 \times 10^{+0}$
1000	8.2×10^{-1}	8.9×10^{-1}	9.7×10^{-1}	$1.3 \times 10^{+0}$
1400	9.0×10^{-1}	8.8×10^{-1}	$1.0 \times 10^{+0}$	$1.3 \times 10^{+0}$
1800	9.4×10^{-1}	$1.0 \times 10^{+0}$	$1.3 \times 10^{+0}$	$1.3 \times 10^{+0}$
2200	$1.1 \times 10^{+0}$	$1.1 \times 10^{+0}$	$1.4 \times 10^{+0}$	$1.5 \times 10^{+0}$
2600	$1.2 \times 10^{+0}$	$1.1 \times 10^{+0}$	$1.5 \times 10^{+0}$	$1.6 \times 10^{+0}$

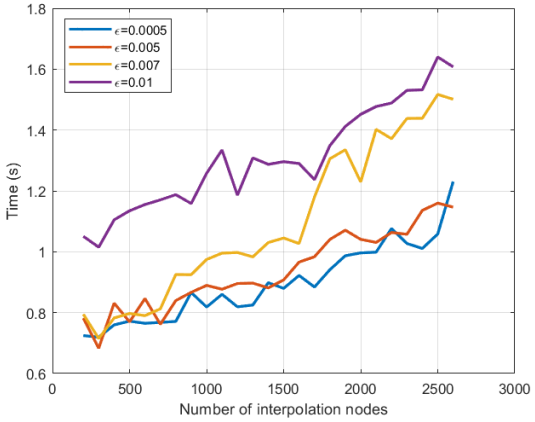

Figure 3. Computation time (in seconds) using the variational spline kernel in GBF-PUM algorithm with $J = 8$ subdomains, enlargement parameter $r = 8$, kernel parameters $s = 2$ and $\epsilon \in [0.0005, 0.01]$.

It is possible to use the same process for calculating an interpolant through the diffusion kernel. However, the results may be largely inaccurate when the kernel parameter is not carefully chosen. Indeed, Figure 4 compares results achieved with the variation spline kernel, pointing out that the use of diffusion kernel with a fixed parameter $t = 10$ underperforms those of the variation spline kernel with $\epsilon = 0.001$ and $s = 2$. In particular, in such figure it can be easily inferred that the GBF-PUM using the diffusion kernel is not particularly accurate with $t = 10$. It is therefore important to suitably determine an appropriate parameter for the diffusion kernel. In Table 6 we thus pertain to the signal approximation obtained by the diffusion kernel with parameter $t = 20$. The sole difference from previous experiments lies in the method of calculating the diffusion kernel. In the current computation, a kernel derived using the complete Laplacian is employed, rather than one generated through combinations of local kernels. The results shown in this table are now consistent with those found in [10], with both errors and computation times significantly reduced. Additionally, in this study the diffusion kernel leads to a more efficient method than the one generated by the variational spline in terms of computational time, especially for a larger number of interpolation nodes (cf. Table 5 and Table 6).

Number of nodes	RMAE	RRMSE	Time (s)
200	5.7×10^{-1}	2.0×10^{-1}	3.5×10^{-1}
600	2.7×10^{-1}	2.7×10^{-2}	3.2×10^{-1}
1000	2.2×10^{-2}	1.2×10^{-3}	3.6×10^{-1}
1400	1.4×10^{-4}	1.4×10^{-5}	4.7×10^{-1}
1800	3.1×10^{-5}	1.4×10^{-6}	4.6×10^{-1}
2200	3.2×10^{-6}	1.6×10^{-7}	6.0×10^{-1}
2600	1.8×10^{-6}	1.2×10^{-7}	6.0×10^{-1}

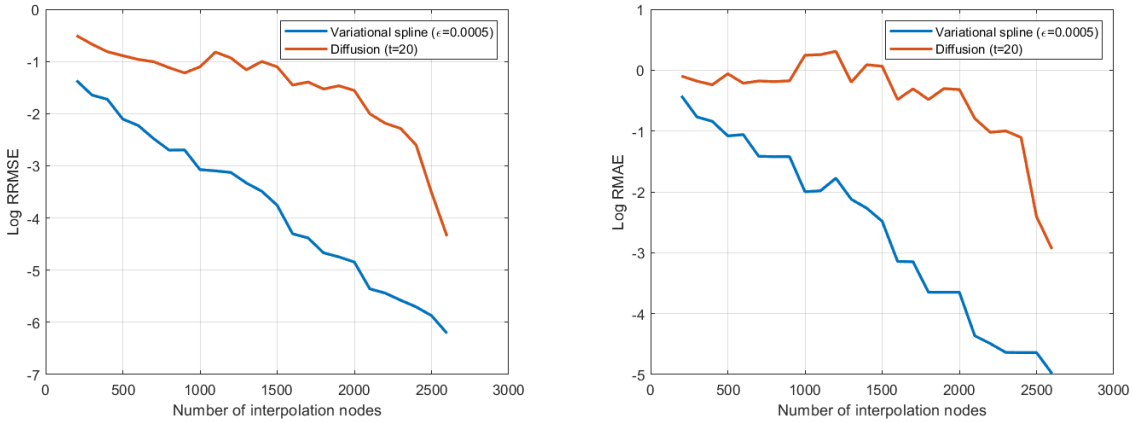

Figure 4. Comparison between errors, RRMSE (left) and RMAE (right), using the diffusion kernel and the variational spline kernel ($s = 2$) with $J = 8$ subdomains and enlargement parameter $r = 8$.

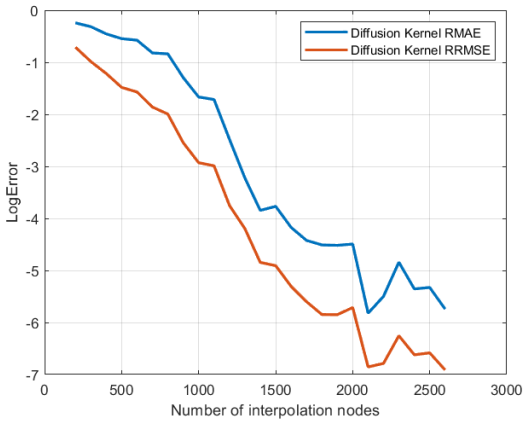

Figure 5. RRMSE and RMAE obtained using the diffusion kernel with parameter $t = 20$. The number of subdomains is $J = 8$ with an enlargement parameter $r = 8$.

Then, we present a direct comparison between the two kernels, as already shown in Figure 4, but using an appropriate parameter for calculating the diffusion kernel. It is easy to observe that the results obtained in Table 7 are much closer to those of variational spline in terms of accuracy. Interestingly, the computation times using the diffusion kernel (reported in Table 8) are lower than those recorded with the variational spline kernel in Table 5.

Number of nodes	Diff. RMAE	Diff. RRMSE	Var. RMAE	Var. RRMSE
200	5.7×10^{-1}	2.0×10^{-1}	3.9×10^{-1}	4.4×10^{-2}
600	2.7×10^{-1}	2.7×10^{-2}	8.9×10^{-2}	5.8×10^{-3}
1000	2.2×10^{-2}	1.2×10^{-3}	1.0×10^{-2}	8.4×10^{-4}
1400	1.4×10^{-4}	1.4×10^{-5}	5.4×10^{-3}	3.3×10^{-4}
1800	3.1×10^{-5}	1.4×10^{-6}	2.6×10^{-4}	2.6×10^{-5}
2200	3.2×10^{-6}	1.6×10^{-7}	4.0×10^{-5}	4.6×10^{-6}
2600	1.8×10^{-6}	1.2×10^{-7}	1.3×10^{-5}	7.8×10^{-7}

Similarly to the variational spline case, we now present some numerical results related to the GBF-PUM interpolation of the same signal used previously on the Minnesota graph. Tables 9-10 and graphs in Figure 7 show the RMAE and RRMSE as the parameter t varies between 10 and 40. We have chosen to always consider an enlargement parameter $r = 8$ with $J = 8$ clusters.

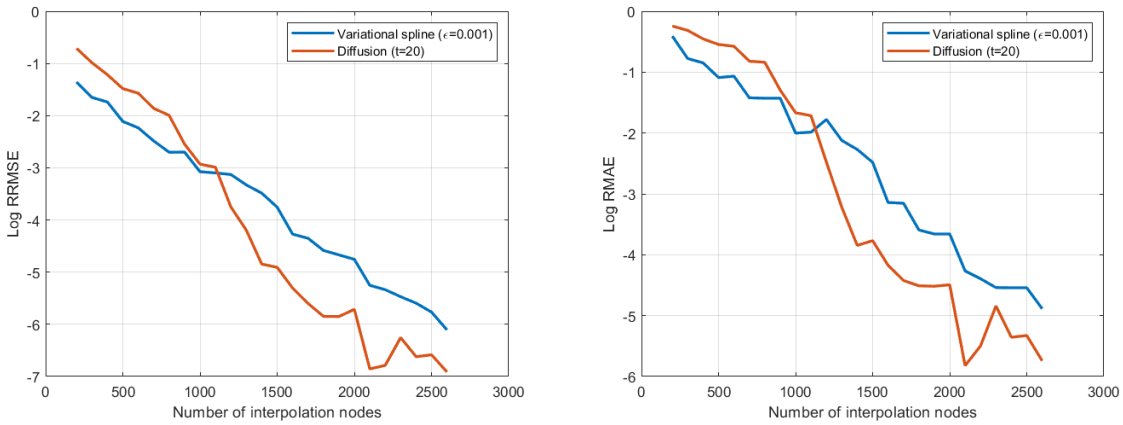

Figure 6. Comparison between errors, RRMSE (left) and RMAE (right), using the diffusion kernel and the variational spline kernel ($s = 2$) with $J = 8$ subdomains and enlargement parameter $r = 8$.

Number of nodes	Time (s) variational spline	Time (s) diffusion kernel
200	7.0×10^{-1}	3.5×10^{-1}
600	7.8×10^{-1}	3.2×10^{-1}
1000	8.8×10^{-1}	3.6×10^{-1}
1400	9.9×10^{-1}	4.7×10^{-1}
1800	9.4×10^{-1}	4.6×10^{-1}
2200	$1.0 \times 10^{+0}$	6.0×10^{-1}
2600	$1.1 \times 10^{+0}$	7.4×10^{-1}

Number of nodes	$t = 10$	$t = 20$	$t = 30$	$t = 40$
200	8.0×10^{-1}	5.7×10^{-1}	4.6×10^{-1}	4.3×10^{-1}
600	6.1×10^{-1}	2.7×10^{-1}	2.0×10^{-1}	1.3×10^{-1}
1000	2.5×10^{-1}	2.2×10^{-2}	1.1×10^{-3}	1.0×10^{-4}
1400	9.6×10^{-2}	1.4×10^{-4}	6.0×10^{-5}	1.9×10^{-4}
1800	4.8×10^{-3}	3.1×10^{-5}	1.1×10^{-5}	3.4×10^{-6}
2200	1.5×10^{-4}	3.2×10^{-6}	1.0×10^{-5}	9.7×10^{-6}
2600	2.7×10^{-6}	1.8×10^{-6}	3.4×10^{-7}	1.8×10^{-6}

Number of nodes	$t = 10$	$t = 20$	$t = 30$	$t = 40$
200	3.1×10^{-1}	1.9×10^{-1}	1.3×10^{-1}	8.9×10^{-2}
600	9.3×10^{-2}	2.7×10^{-2}	1.4×10^{-2}	8.3×10^{-3}
1000	2.3×10^{-2}	1.2×10^{-2}	5.8×10^{-5}	8.1×10^{-6}
1400	6.3×10^{-2}	1.4×10^{-5}	5.1×10^{-6}	1.1×10^{-5}
1800	2.7×10^{-4}	1.4×10^{-6}	8.3×10^{-7}	2.5×10^{-7}
2200	9.5×10^{-6}	1.6×10^{-7}	7.0×10^{-7}	7.7×10^{-7}
2600	1.0×10^{-7}	1.2×10^{-7}	6.5×10^{-8}	2.7×10^{-8}

In this case, the behavior does not appear linear as observed using the previous kernel. However, it can be reasonably assumed that, for a small number of nodes (< 1000), using a higher parameter within the analyzed range leads to more accurate results.

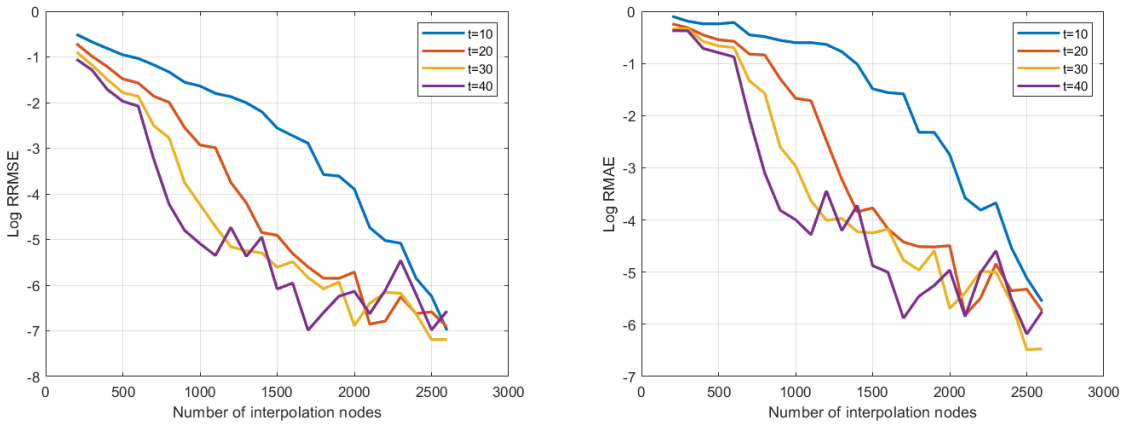

Figure 7. RRMSE (left) and RMAE (right) obtained by using the diffusion kernel in the GBF-PUM algorithm with $J = 8$ subdomains, enlargement parameter $r = 8$ and kernel parameter $t \in [10, 40]$.

7. Conclusions and future work

In this paper we proposed several numerical tests obtained using GBFs generated by the heat kernel and the kernel created through variational splines. From the numerical experiments, as expected, we can note that when the number of sample nodes increases, also the precision of the GBF-PUM improves. Moreover, we remark that in the case of a real dataset, this behaviour is less noticeable but still significant, due to the nature of the data. Future work consists in extending the framework to a multilayer signals using these kernels to recover the temporal dependency and using approximation instead of interpolation for greater applicability to real problems.

Acknowledgements

This work has been supported by the INdAM Research group GNCS, which includes the authors A. De Rossi and R. Cavoretto among its members. It has been also supported by the Spoke 1 "Future HPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 Componente 2 Investimento 1.4: Potenziamento strutture di ricerca e creazione di "campioni nazionali di R&S (M4C2-19)" – Next Generation EU (NGEU). Moreover, the work has been supported by the Fondazione CRT, project 2022 "Modelli matematici e algoritmi predittivi di intelligenza artificiale per la mobilità sostenibile". This research has been accomplished within the RITA "Research ITalian network on Approximation", the UMI Group TAA "Approximation Theory and Applications", and the SIMAI Activity Group ANA&A "Numerical and Analytical Approximation of Data and Functions with Applications".

References

1. A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.
2. I. Z. Pesenson and M. Z. Pesenson, Graph signal sampling and interpolation based on clusters and averages, *Journal of Fourier Analysis and Applications*, vol. 27, no. 39, 2021.
3. I. Z. Pesenson, Sampling in Paley-Wiener spaces on combinatorial graphs, *Transactions of the American Mathematical Society*, vol. 360, no. 10, 2008.
4. R. S. Strichartz, Half sampling on bipartite graphs, *Journal of Fourier Analysis and Applications*, vol. 22, p. 1157–1173, 2016.
5. X. Wang, P. Liu, and Y. Gu, Local-set-based graph signal reconstruction, *IEEE Transactions on Signal Processing*, vol. 63, no. 9, pp. 2432 – 2444, 2015.

6. J. P. Ward, F. J. Narcowich, and J. D. Ward, Interpolating splines on graphs for data science applications, *Applied and Computational Harmonic Analysis*, vol. 49, no. 2, pp. 540—-557, 2020.
7. X. Wang, J. Chen, and Y. Gu, Local measurement and reconstruction for noisy bandlimited graph signals, *Signal Processing*, vol. 129, pp. 119–129, 2016.
8. I. D. Irawati, A. B. Suksmono, and I. J. M. Edward, An interpolation comparative analysis for missing internet traffic data, in *Proceedings of the 3rd International Conference on Electronics, Communications and Control Engineering*, p. 26–30, New York, NY, USA: Association for Computing Machinery, 2020.
9. H. Katayama, S. Yasuda, and T. Fuse, Comparative validation of spatial interpolation methods for traffic density for data-driven travel-time prediction, *International Journal of Intelligent Transportation Systems Research*, vol. 20, pp. 830–837, 2022.
10. R. Cavoretto, A. De Rossi, and W. Erb, Partition of unity methods for signal processing on graphs, *Journal of Fourier Analysis and Applications*, vol. 27, no. 66, pp. 342–346, 2021.
11. R. Cavoretto, A. De Rossi, S. Lancellotti, and F. Romaniello, Node-bound communities for partition of unity interpolation on graphs, *Applied Mathematics and Computation*, vol. 467, p. 128502, 2024.
12. N. Aronszajn, Theory of reproducing kernels, *Transactions of the American Mathematical Society*, vol. 68, pp. 337–404, 1950.
13. W. Erb, Graph signal interpolation with positive definite graph basis functions, *Applied and Computational Harmonic Analysis*, vol. 60, pp. 368–395, 2022.
14. 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, 2016.
15. R. Cavoretto, S. De Marchi, A. De Rossi, E. Perracchione, and G. Santin, Partition of unity interpolation using stable kernel-based techniques, *Applied Numerical Mathematics*, vol. 116, pp. 95–107, 2017.
16. W. Erb, Semi-supervised learning on graphs with feature-augmented graph basis functions, *arXiv:2003.07646*, 2020.
17. J. P. Ward, F. J. Narcowich, and J. D. Ward, Interpolating splines on graphs for data science applications, *Applied and Computational Harmonic Analysis*, vol. 49, no. 2, pp. 540–577, 2020.
18. R. Kondor and J. D. Lafferty, Diffusion kernels on graphs and other discrete input spaces, in *Proc. of the 19th. International Conference on Machine Learning*, ICML02, 2002.
19. R. K. A.J. Smola, Kernels and regularization on graphs, in *Lecture Notes in Computer Science(), vol 2777*, Springer, Berlin, Heidelberg, 2003.
20. L. Stankovic, M. Dakovic, and E. Sejdic, *Introduction to Graph Signal Processing*, pp. 3–108. Cham: Springer International Publishing, 2019.
21. C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer, 2001.
22. F. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.
23. W. Erb, Shapes of uncertainty in spectral graph theory, *IEEE Transactions on Information Theory*, vol. 67, pp. 1291–1307, 2019.
24. G. Fasshauer, Positive definite kernels: Past, present and future, *Dolomites Research Notes on Approximation*, vol. 4, pp. 21–63, 2011.
25. R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.
26. S. De Marchi and R. Schaback, Stability of kernel-based interpolation, *Advances in Computational Mathematics*, vol. 32, no. 2, pp. 155–161, 2010.
27. I. Pesenson, Variational splines and Paley–Wiener spaces on combinatorial graphs, *Constructive Approximation*, vol. 29, pp. 1–21, 2009.
28. M. Newman, *Networks: An Introduction*. Oxford University Press, 2010.
29. T. F. Gonzalez, Clustering to minimize the maximum intercluster distance, *Theoretical Computer Science — Journal*, vol. 38, pp. 293–306, 1985.

30. D. S. Hochbaum and D. B. Shmoys, A best possible heuristic for the k-center problem, *Mathematics of Operations Research*, vol. 10, no. 2, pp. 180–184, 1985.
31. S. Fortunato, Community detection in graphs, *Physics Reports*, vol. 486, pp. 75–174, 2010.
32. R. Cavoretto, A. De Rossi, and W. Erb, GBFPUM - a MATLAB package for partition of unity based signal interpolation and approximation on graphs, *Dolomites Research Notes on Approximation*, vol. 15, no. 2, pp. 25–34, 2022.
33. R. Rossi and N. Ahmed, The network data repository with interactive graph analytics and visualization, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, Mar. 2015.