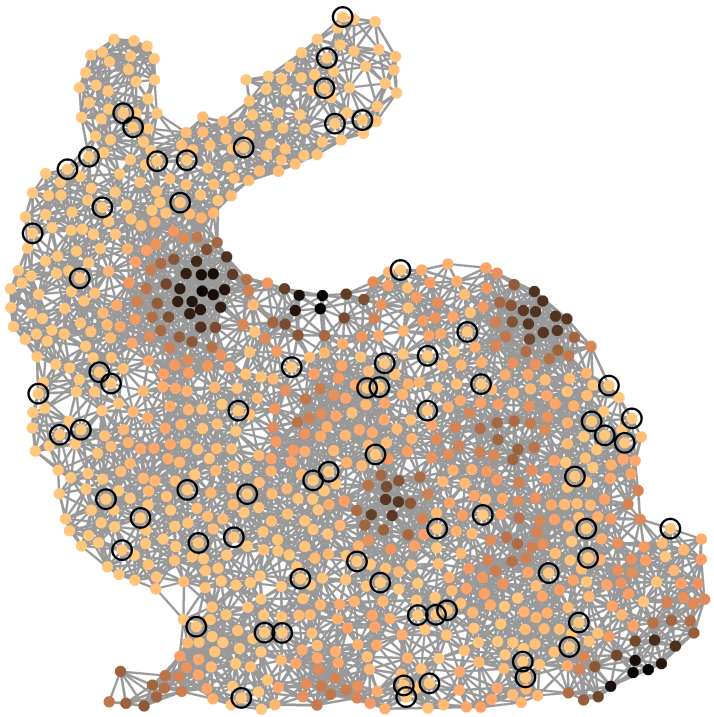
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università degli Studi di Padova

Modern Computational Harmonic Analysis on Graphs and Networks

2. Kernel-based methods on graphs

Wolfgang Erb
University of Padova

C.I.M.E. Summer School
"Modern Perspectives in Approximation Theory: Graphs, Networks, quasi-interpolation and Sampling Theory"
July 21-24, 2025, Cetraro (CS), Italy



wolfgang.erb@unipd.it

Outline of this talk

1. Kernel-based interpolation on graphs
   - The Graph Fourier Transform, convolution and generalized translation on graphs
   - How to set up kernel-based interpolation on graphs
   - Approximation and classification with Graph basis functions (GBFs)
2. Krylov subspace methods for the approximation of kernels ;methods
   - Block Krylov methods for the approximation of matrix functions
   - How to guarantee positive definiteness in Krylov space approximations
3. Partition of Unity Methods (PUMs)
   - How to generate partition of unities on graphs
   - How to combine PUMs with GBF approximation

Recapitulation: the Graph Fourier Transform

We write the eigendecomposition of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ as

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*,$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$ contains the eigenvalues of $\mathbf{L}$ (increasingly ordered) and the unitary matrix $\mathbf{U} = (u_1, u_2, \ldots, u_n)$ the eigenvectors.

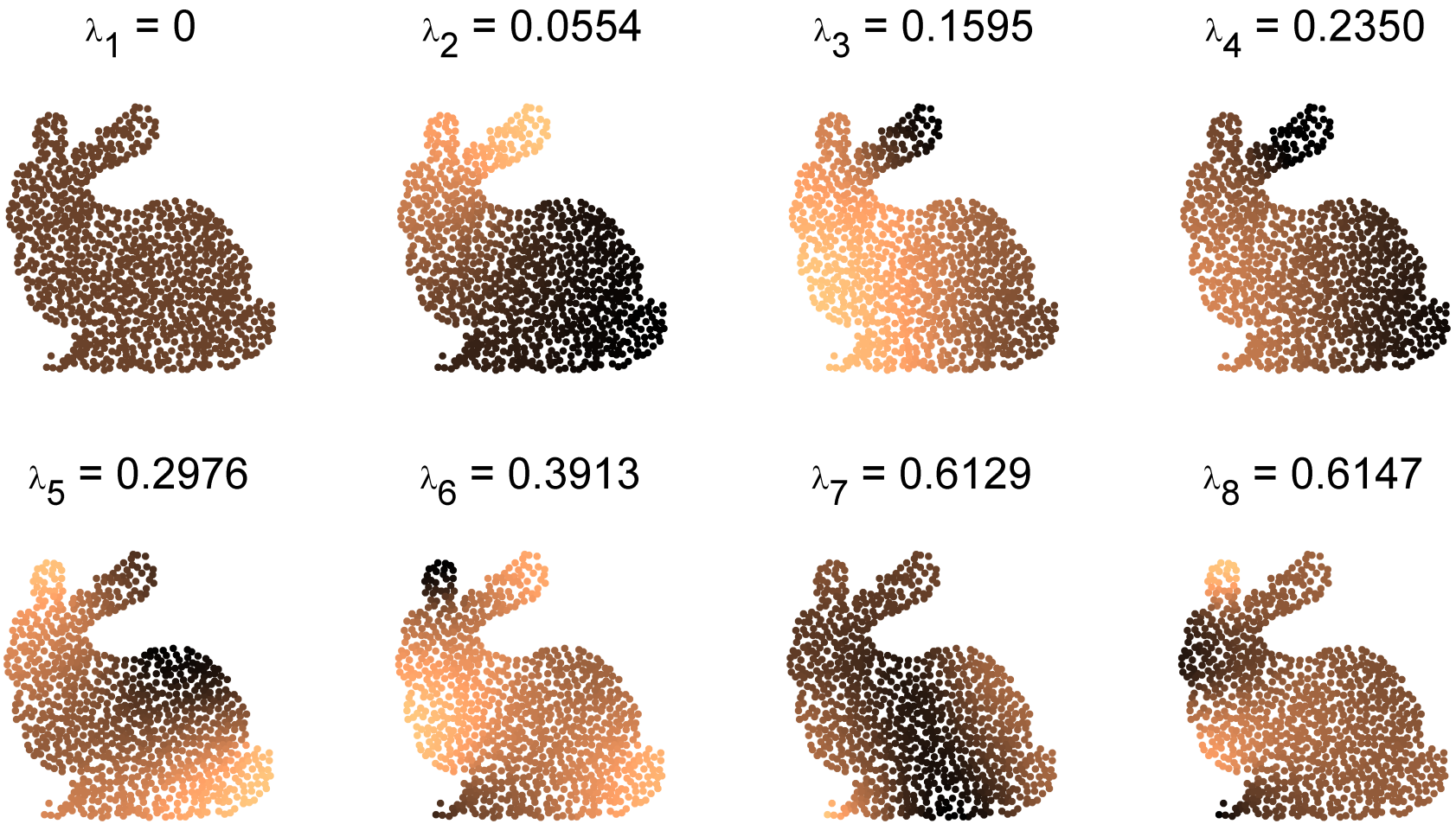
Then, the Graph Fourier transform of x is defined as

$$\hat{x} = \mathbf{U}^* x, \quad \text{with } k\text{-th. entry} \quad \hat{x}_k = u_k^* x = \langle x, u_k \rangle.$$

The inverse Fourier transform is correspondingly given as

$$x = \mathbf{U}\hat{x}.$$

Example: the bunny graph



The first 8 eigenfunctions of the graph Laplacian **L** on the bunny graph.

Convolution on graphs

Convolution in $\mathbb{R}$

$$(x * y)(s) = \int_{\mathbb{R}} x(t)y(s-t)\mathrm{d}t.$$

In the Fourier domain:

$$\widehat{(x * y)}(\omega) = \hat{x}(\omega)\hat{y}(\omega)$$

Graph convolution

No translation available
Idea: define convolution
via graph Fourier transform

$$\widehat{(x * y)}_k = \hat{x}_k \hat{y}_k$$

We define the graph convolution as

$$y * x := \mathbf{U}\mathbf{M}_{\hat{y}}\hat{x} = \mathbf{U}\mathbf{M}_{\hat{y}}\mathbf{U}^* x, \quad \text{where } \mathbf{M}_{\hat{y}} = \mathrm{diag}(\hat{y}_1, \dots \hat{y}_n).$$

Further, we define the convolution matrix $\mathbf{C}_y \in \mathbb{R}^{n \times n}$ as

$$\mathbf{C}_y = \mathbf{U}\mathbf{M}_{\hat{y}}\mathbf{U}^*.$$

Note: the graph convolution depends on the choice of the basis u_k.

Generalized translation on graphs

Translation in $\mathbb{R}$

In the weak sense, we have

$$x(t+s) = (x * \delta_s)(t)$$
$$= \int_{\mathbb{R}} \hat{x}(\omega) e^{-2\pi\imath\omega s} e^{-2\pi\imath\omega t} \mathrm{d}\omega$$

Graph translation

We can define a generalized translation as

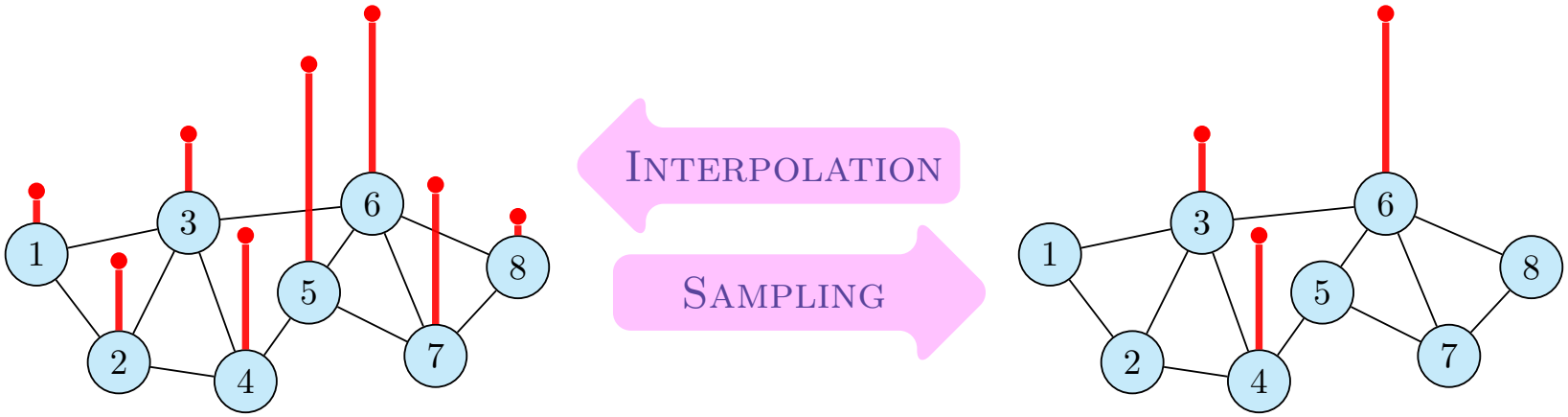
$$\mathbf{C}_{\delta_v} x = (\delta_v * x)$$
$$= \mathbf{U}\mathbf{M}_{\hat{\delta}_v} \mathbf{U}^* x$$

Warnings:

- In general, no group structure for the generalized translation
- In general, the matrix $\mathbf{C}_{\delta_v}$ is not unitary.
- Depends on the choice of the basis elements u_k.

Interpolation on graphs

Graph signals are functions $x : V \to \mathbb{R}$ on the node set of G, we can represent them as vectors $x = (x(v_1), \ldots, x(v_n))^* \in \mathbb{R}^n$.



Interpolation problem: from the knowledge of the signal x on a subset

$$W = \{w_1, \ldots, w_N\} \subset V,$$

reconstruct the signal x on V.

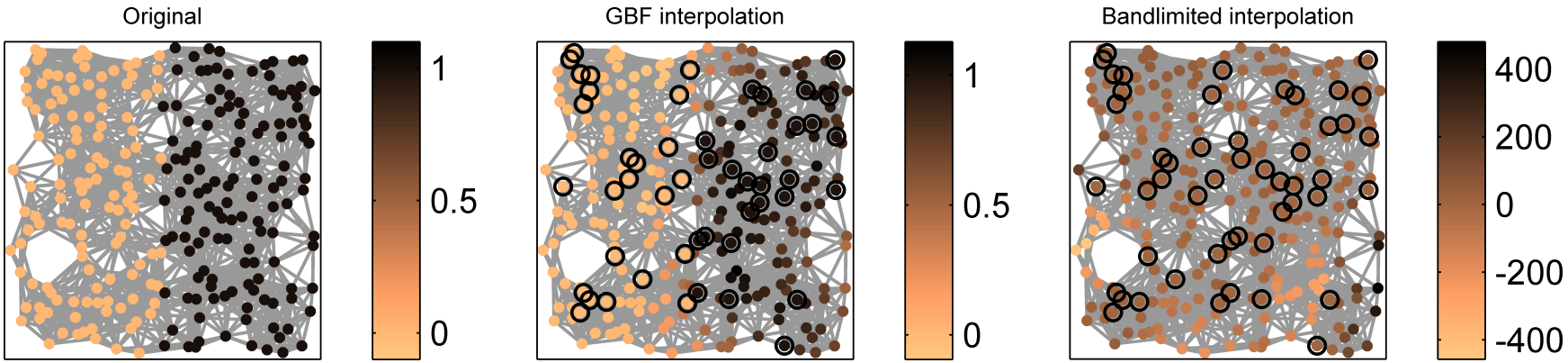
1. Idea: Bandlimited signals

First approach: interpolate the given data $x(\mathrm{w})$, $\mathrm{w} \in W$ in the space of bandlimited signals

$$\mathcal{B}_M = \mathrm{span}\{u_1, \ldots, u_M\}.$$

Drawbacks:

- If $N = M$, uniqueness of interpolant is not guaranteed. W is not necessarily a norming set for $\mathcal{B}_M$.
- Also if interpolant can be calculated, bad-conditioning occurs (Runge-type artifacts).



2. Idea: kernels for interpolation

Consider a kernel $K : V \times V \to \mathbb{R}$ on the graph G. A kernel K allows to introduce a linear operator $\mathbf{K} : \mathcal{L}(G) \to \mathcal{L}(G)$ as

$$\mathbf{K}x(\mathrm{v}_i) = \sum_{j=1}^{n} K(\mathrm{v}_i, \mathrm{v}_j)x(\mathrm{v}_j).$$

We can as well write it as a matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ given by

$$\mathbf{K} = \begin{pmatrix} K(\mathrm{v}_1, \mathrm{v}_1) & K(\mathrm{v}_1, \mathrm{v}_2) & \dots & K(\mathrm{v}_1, \mathrm{v}_n) \\ K(\mathrm{v}_2, \mathrm{v}_1) & K(\mathrm{v}_2, \mathrm{v}_2) & \dots & K(\mathrm{v}_2, \mathrm{v}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathrm{v}_n, \mathrm{v}_1) & K(\mathrm{v}_n, \mathrm{v}_2) & \dots & K(\mathrm{v}_n, \mathrm{v}_n) \end{pmatrix}.$$

The kernel K is called positive definite, if the matrix $\mathbf{K}$ is symmetric and positive definite.

2. Idea: kernels for interpolation

Assume that the kernel K is positive definite. Then, in the space

$$\mathcal{N}_{K,W} = \left\{ x \in \mathcal{L}(G) \mid x(\mathrm{v}) = \sum_{k=1}^{N} c_k K(\mathrm{v}, \mathrm{w}_k) \right\}$$

we can find a unique interpolant $\mathrm{I}_W x$ such that

$$\mathrm{I}_W x(\mathrm{w}) = x(\mathrm{w}), \quad \text{for all } \mathrm{w} \in W.$$

The coefficients c_k of the interpolant can be calculated as

$$\underbrace{\begin{pmatrix} K(\mathrm{w}_1, \mathrm{w}_1) & K(\mathrm{w}_1, \mathrm{w}_2) & \dots & K(\mathrm{w}_1, \mathrm{w}_N) \\ K(\mathrm{w}_2, \mathrm{w}_1) & K(\mathrm{w}_2, \mathrm{w}_2) & \dots & K(\mathrm{w}_2, \mathrm{w}_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathrm{w}_N, \mathrm{w}_1) & K(\mathrm{w}_N, \mathrm{w}_2) & \dots & K(\mathrm{w}_N, \mathrm{w}_N) \end{pmatrix}}_{\mathbf{K}_W = \mathrm{E}_W^* \mathbf{K} \mathrm{E}_W} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} x(\mathrm{w}_1) \\ x(\mathrm{w}_2) \\ \vdots \\ x(\mathrm{w}_N) \end{pmatrix}.$$

How to obtain meaningful kernels on graphs?

Idea: link kernels with spectral graph structure.

RBF-kernel in $\mathbb{R}$

Use $K_f(t,s) = f(t-s)$ with a positive definite radial basis function f.

$$\mathrm{I}_W x(t) = \sum_{k=1}^{N} c_k f(t - \mathrm{w}_k)$$

GBF-kernel on graphs [1]

Use $K_f(\mathrm{v},\mathrm{w}) = \mathbf{C}_{\delta_\mathrm{w}} f(\mathrm{v})$ with a positive definite graph basis function f.

$$\mathrm{I}_W x(\mathrm{v}) = \sum_{k=1}^{N} c_k \mathbf{C}_{\delta_{\mathrm{w}_k}} f(\mathrm{v})$$

Definition: we call a graph signal f positive definite (p.d.) if the kernel $K_f(\mathrm{v},\mathrm{w}) = \mathbf{C}_{\delta_\mathrm{w}} f(\mathrm{v})$ is positive definite.

Characterization of p.d. functions

Bochner's Theorem on $\mathbb{R}$

A continuous function f is p.s.d. if and only if it is the inverse Fourier transform of a non-negative Borel measure.

Bochner's result on graphs

A graph signal f is p.s.d. if and only if $\hat{f}_k \geq 0$ for all k.
A graph signal f is p.d. if and only if $\hat{f}_k > 0$ for all k.

Note: for a p.d. GBF f the Mercer decomposition of K_f is given as

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

Examples of GBF kernels

1. Diffusion kernels [5] are based on the eigendecomposition $\mathbf{L} = \sum_{k=1}^{n} \lambda_k u_k u_k^*$ of the graph Laplacian and given as

$$\mathbf{K} = e^{-t\mathbf{L}} = \sum_{k=1}^{n} e^{-t\lambda_k} u_k u_k^*.$$

2. Variational splines [6,9] are based on the decomposition

$$\mathbf{K} = (\epsilon \mathbf{I}_n + \mathbf{L})^{-s} = \sum_{k=1}^{n} \frac{1}{(\epsilon + \lambda_k)^s} u_k u_k^*.$$

3. Kernels with a polynomial decay in the eigenbasis:

$$\mathbf{K} = \sum_{k=1}^{n} \frac{1}{k^s} u_k u_k^*.$$

Example of GBF kernels

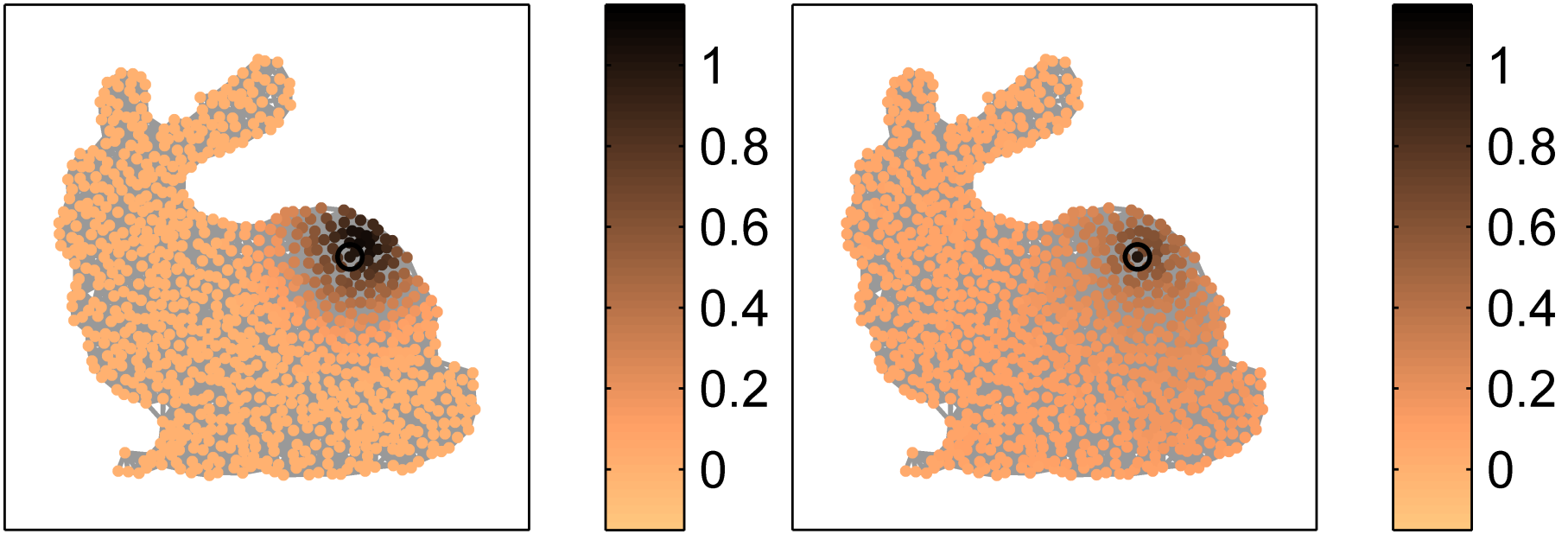


Figure 1: Illustration of columns $K(\cdot, \mathrm{w})$ of the kernels matrices in Example (1) and (3). The ringed node corresponds to w. These columns can be interpreted as generalized translates of a graph basis function f (see [1] for more details).

2. Example of GBF kernels

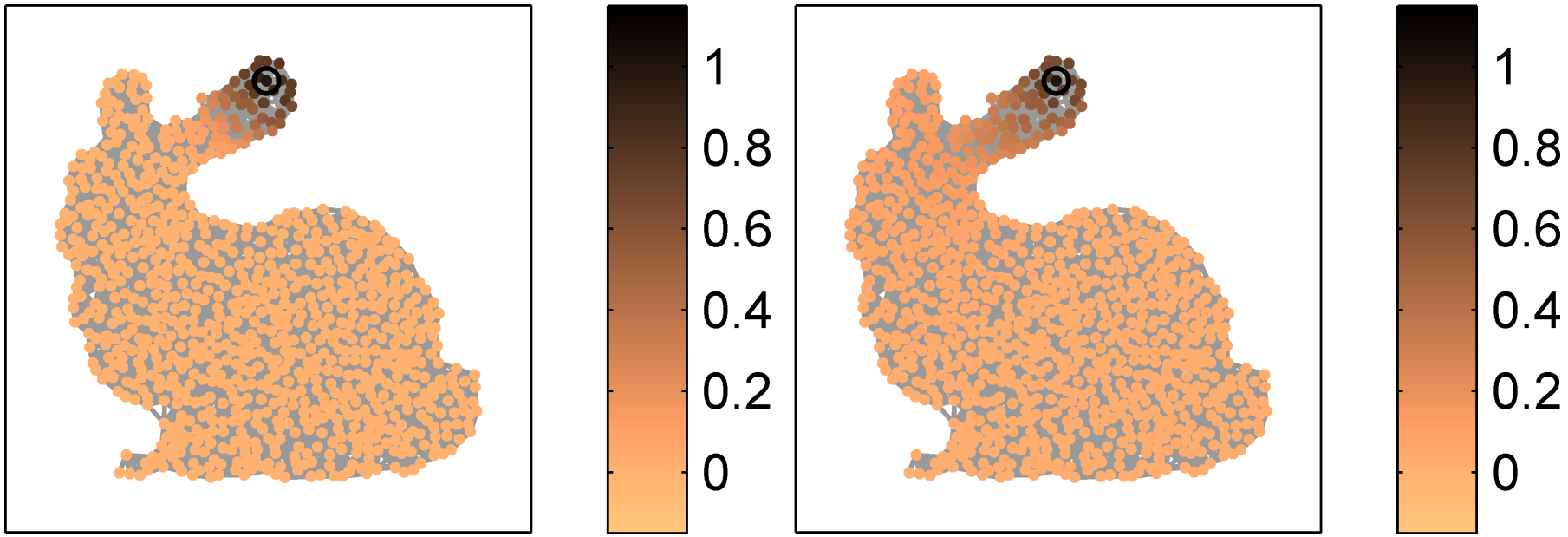


Figure 2: Illustration of columns $K(\cdot, \mathrm{w})$ of the kernels matrices in Example (1) and (3) for another center node w. These columns can be interpreted as generalized translates of a graph basis function f (see [1] for more details)

.

Algorithm 1: Kernel-based interpolation on graphs

Input: Samples $x(\mathrm{w}_1), \ldots, x(\mathrm{w}_N)$, and a positive definite kernel K.

Calculate the N kernel columns $K(\cdot, \mathrm{w}_1), \ldots, K(\cdot, \mathrm{w}_N)$.

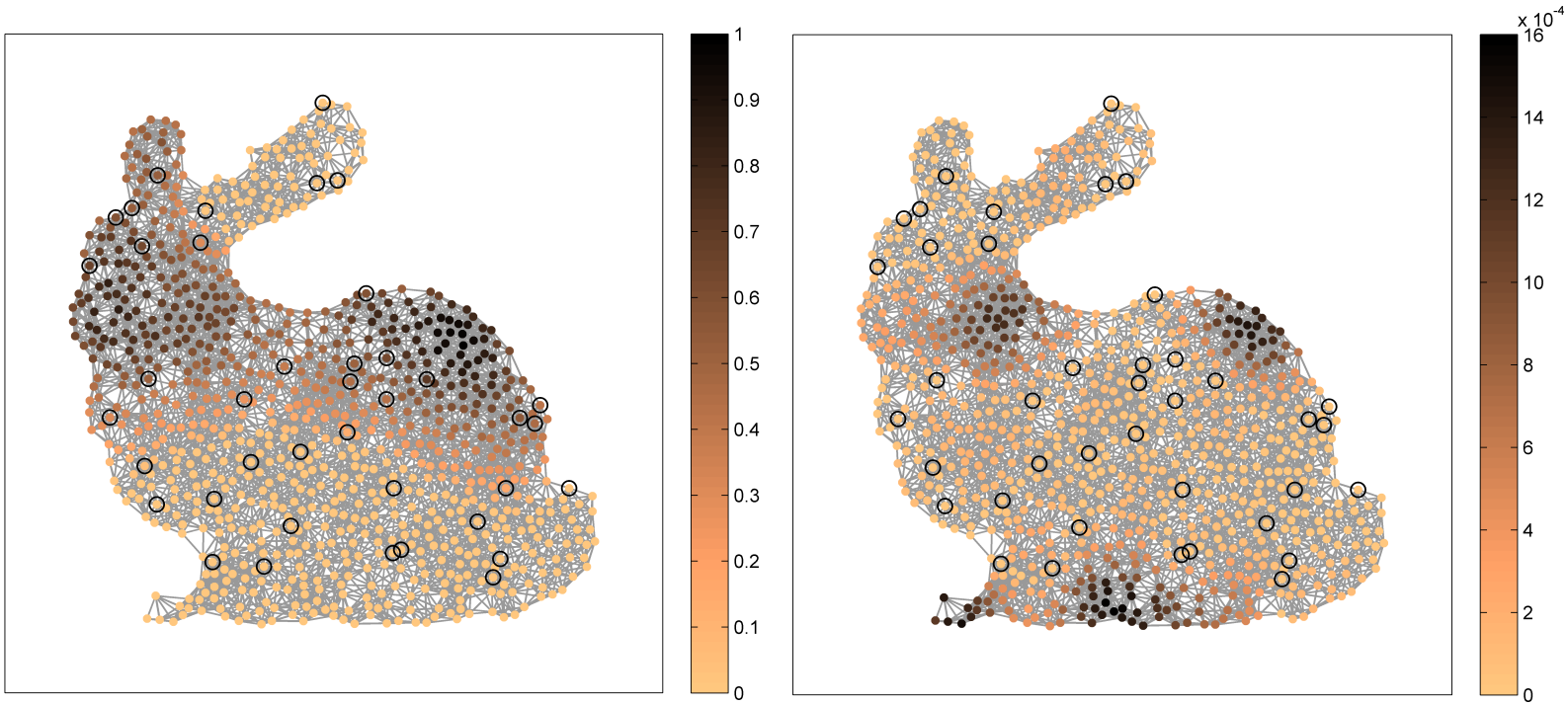
Solve the linear system of equations

$$\underbrace{\begin{pmatrix} K(\mathrm{w}_1, \mathrm{w}_1) & K(\mathrm{w}_1, \mathrm{w}_2) & \ldots & K(\mathrm{w}_1, \mathrm{w}_N) \\ K(\mathrm{w}_2, \mathrm{w}_1) & K(\mathrm{w}_2, \mathrm{w}_2) & \ldots & K(\mathrm{w}_2, \mathrm{w}_N) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathrm{w}_N, \mathrm{w}_1) & K(\mathrm{w}_N, \mathrm{w}_2) & \ldots & K(\mathrm{w}_N, \mathrm{w}_N) \end{pmatrix}}_{\mathbf{K}_W = \mathrm{E}_W^* \mathbf{K} \mathrm{E}_W} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} x(\mathrm{w}_1) \\ x(\mathrm{w}_2) \\ \vdots \\ x(\mathrm{w}_N) \end{pmatrix}.$$

Calculate the kernel interpolant

$$\mathrm{I}_W x(\mathrm{v}) = \sum_{k=1}^{N} c_k K(\mathrm{v}, \mathrm{w}_k).$$

Examples: interpolation on bunny graph



GBF interpolation for the input signal $x = u_4$.
Left: GBF interpolant for 40 given nodes (out of 900 nodes) and the GBF $f_{\mathrm{pol},4}$.

Right: interpolation error with respect to the original signal.

Time-critical parts in the calculation of the interpolant

Time-critical parts in the calculation of $\mathrm{I}_W x$

1. For graphs (with an irregular structure) the N kernel columns $K(\cdot, \mathrm{w}_k) = \mathbf{K} e_{\mathrm{w}_k}$, $k \in \{1, \ldots, N\}$ have to be calculated in a first step. This might be cost-intensive for large graphs.
2. The resolution of the interpolation problem $\mathbf{K}_W c = x(W)$ might get computationally expensive if N is large.

We will consider two strategies two overcome these issues.

1. Krylov subspace techniques to generate the kernel columns $K(\cdot, \mathrm{w}_k)$ efficiently [2].
2. Partition of unity methods (PUMs) to reduce cost-intensive calculations by splitting the graph into smaller subgraphs [3].

Software for efficient kernel methods on graphs

A software for efficient interpolation on graphs can be found as MATLAB package `GBFlearn` at

`https://github.com/WolfgangErb/GBFlearn`

or as Python version (by G. Santin) at

`https://github.com/GabrieleSantin/GraphBasisFunctions`

- GBF stands for graph basis function. Important kernels defined upon the graph Fourier transform can be described through GBFs [1].
- `GBFlearn` contains simple scripts for the generation and application of GBFs on graphs as the solution of interpolation and classification problems [1,4]. Efficient schemes based an Krylov subspace approximations and PUMs have been incorporated [2,3].

Krylov subspace methods

GBF kernels on graphs are usually given as matrix functions of the symmetric graph Laplacian $\mathbf{L}$ such that the kernel functions read as

$$K(\cdot, \mathrm{w}) = \phi(\mathbf{L}) e_{\mathrm{w}},$$

where ϕ is a positive function on the spectrum of $\mathbf{L}$.

Idea: use Krylov subspace methods to approximate the matrix-vector product $\phi(\mathbf{L})e_{\mathrm{w}}$ with a polynomial $p_{\phi,m}(\mathbf{L})e_{\mathrm{w}}$. For the interpolation algorithm we need N of these, i.e. we want to obtain a polynomial approximation of the form

$$p_{\phi,m}(\mathbf{L})\mathrm{E}_W \approx \phi(\mathbf{L})\mathrm{E}_W, \tag{1}$$

where $\mathrm{E}_W \in \mathbb{R}^{n \times N}$ is the block

$$\mathrm{E}_W = [e_{\mathrm{w}_1}, \ldots, e_{\mathrm{w}_N}] \in \mathbb{R}^{n \times N}.$$

Classical block Lanczos methods

Consider the block Krylov space

$$\mathcal{K}_m^{\text{cbl}}(\mathbf{L}, \mathrm{E}_W) = \left\{ \sum_{k=0}^{m-1} \mathbf{L}^k \mathrm{E}_W \mathrm{C}_k \ : \ \mathrm{C}_k \in \mathbb{R}^{N \times N} \right\}.$$

An orthonormal system $\{q_1, \ldots, q_{mN}\} \subset \mathbb{R}^n$ of vectors related to the classical Krylov space $\mathcal{K}_m^{(\text{cbl})}(\mathbf{L}, \mathrm{E}_W)$ can be obtained by applying $m-1$ steps of a block Lanczos algorithm to the initial block $\mathrm{Q}_1 = [q_1, \ldots, q_N] = \mathrm{E}_W$. We store also the remaining basis elements in $n \times N$-blocks Q_k by setting

$$\mathrm{Q}_k = [q_{(k-1)N+1}, \ldots, q_{kN}], \quad k \in \{1, \ldots, m\}.$$

The blocks $\mathrm{Q}_1, \ldots, \mathrm{Q}_m$ in $\mathcal{K}_m^{\mathrm{cbl}}(\mathbf{L}, \mathrm{E}_W)$ are determined in such a way that after $m-1$ steps the block Lanczos relation

$$\mathbf{L}[\mathrm{Q}_1, \ldots \mathrm{Q}_m] = [\mathrm{Q}_1, \ldots, \mathrm{Q}_{m+1}]\tilde{\mathbf{H}}_m, \tag{2}$$

is satisfied with a block tridiagonal matrix $\tilde{\mathbf{H}}_m \in \mathbb{R}^{(m+1)N \times mN}$ of the form

$$\tilde{\mathbf{H}}_m = \begin{bmatrix} \mathrm{H}_{1,1} & \mathrm{H}_{1,2} & & & \\ \mathrm{H}_{2,1} & \mathrm{H}_{2,2} & \ddots & & \\ & \ddots & \ddots & \ddots & \mathrm{H}_{m-1,m} \\ & & & \mathrm{H}_{m,m-1} & \mathrm{H}_{m,m} \\ & & & & \mathrm{H}_{m+1,m} \end{bmatrix} \Bigg\} \mathbf{H}_m.$$

The blocks $\mathrm{H}_{k+1,k} \in \mathbb{R}^{N \times N}$ are upper triangular and invertible and satisfy $\mathrm{H}_{k+1,k} = \mathrm{H}_{k,k+1}^*$, while $\mathrm{H}_{k,k} = \mathrm{H}_{k,k}^*$. In addition, the classical block Lanczos method enforces the system $\{q_1, \ldots, q_{mN}\}$ to be orthonormal. An approximation of $\phi(\mathbf{L})\mathrm{E}_W$ is given by

$$p_{\phi,m}^{(\mathrm{cbl})}(\mathbf{L})\mathrm{E}_W := [\mathrm{Q}_1, \ldots, \mathrm{Q}_{m+1}]\phi(\mathbf{H}_{m+1})\mathrm{F}_1.$$

Algorithm 2: Classical Block Lanczos to approximate $\phi(\mathbf{L})\mathrm{E}_W$

1: $\mathrm{Q}_1 = \mathrm{E}_W$, $\mathrm{Q}_0 = 0$, $\mathrm{H}_{0,1} = 0$;

2: **for** $k = 1$ to m **do**

3: $\quad \mathrm{X} = \mathbf{L}\mathrm{Q}_k - \mathrm{Q}_{k-1}\mathrm{H}_{k-1,k}$;

4: $\quad \mathrm{H}_{k,k} = \mathrm{Q}_k^* \mathrm{X}$;

5: $\quad \mathrm{X} = \mathrm{X} - \mathrm{Q}_k\mathrm{H}_{k,k}$;

6: $\quad$ Compute reduced QR decomposition of X such that

$$\mathrm{Q}_{k+1}\mathrm{H}_{k+1,k} = \mathrm{X},$$

$\quad$ with $\mathrm{Q}_{k+1} \in \mathbb{R}^{n \times N}$ containing N orthonormal columns $q_{kN+1}, \ldots, q_{(k+1)N}$ and $\mathrm{H}_{k+1,k} \in \mathbb{R}^{N \times N}$ is upper triangular;

7: $\quad$ Set $\mathrm{H}_{k,k+1} = \mathrm{H}_{k+1,k}$;

8: **end for**

9: Set up $\mathbf{H}_m$ from the blocks $\mathrm{H}_{k+1,k}$, $\mathrm{H}_{k,k}$, $k \in \{1, \ldots, m\}$, and calculate

$$\mathrm{U} = \phi(\mathbf{H}_m)\mathrm{F}_1,$$

where $\mathrm{F}_1 \in \mathbb{R}^{mN \times N}$ contains the identity matrix as first $N \times N$-block and all the remaining blocks of F_1 are zero.

10: **Return** $p^{(\mathrm{cbl})}_{\phi,m-1}(\mathbf{L})\mathrm{E}_W := [\mathrm{Q}_1, \ldots, \mathrm{Q}_m]\mathrm{U}$ as an approximation to $\phi(\mathbf{L})\mathrm{E}_W$.

Theoretical advantages of classical block Lanczos methods

Lemma 1

Let the spectrum of $\mathbf{L}$ *be contained in* $[0, \Lambda]$ *and the function* ϕ *be positive on* $[0, \Lambda]$. *Then the matrix* $\mathbf{K}_W^{(\mathrm{cbl})} = \mathrm{E}_W^* p_{\phi,m}^{(\mathrm{cbl})}(\mathbf{L})\mathrm{E}_W$ *is symmetric and positive definite.*

Note: this statement in general not true for other block Krylov methods. For sequentially applied Lanczos the matrix $\mathbf{K}_W^{(\mathrm{sbl})} = \mathrm{E}_W^* p_{\phi,m}^{(\mathrm{sbl})}(\mathbf{L})\mathrm{E}_W$ can even have imaginary eigenvalues.

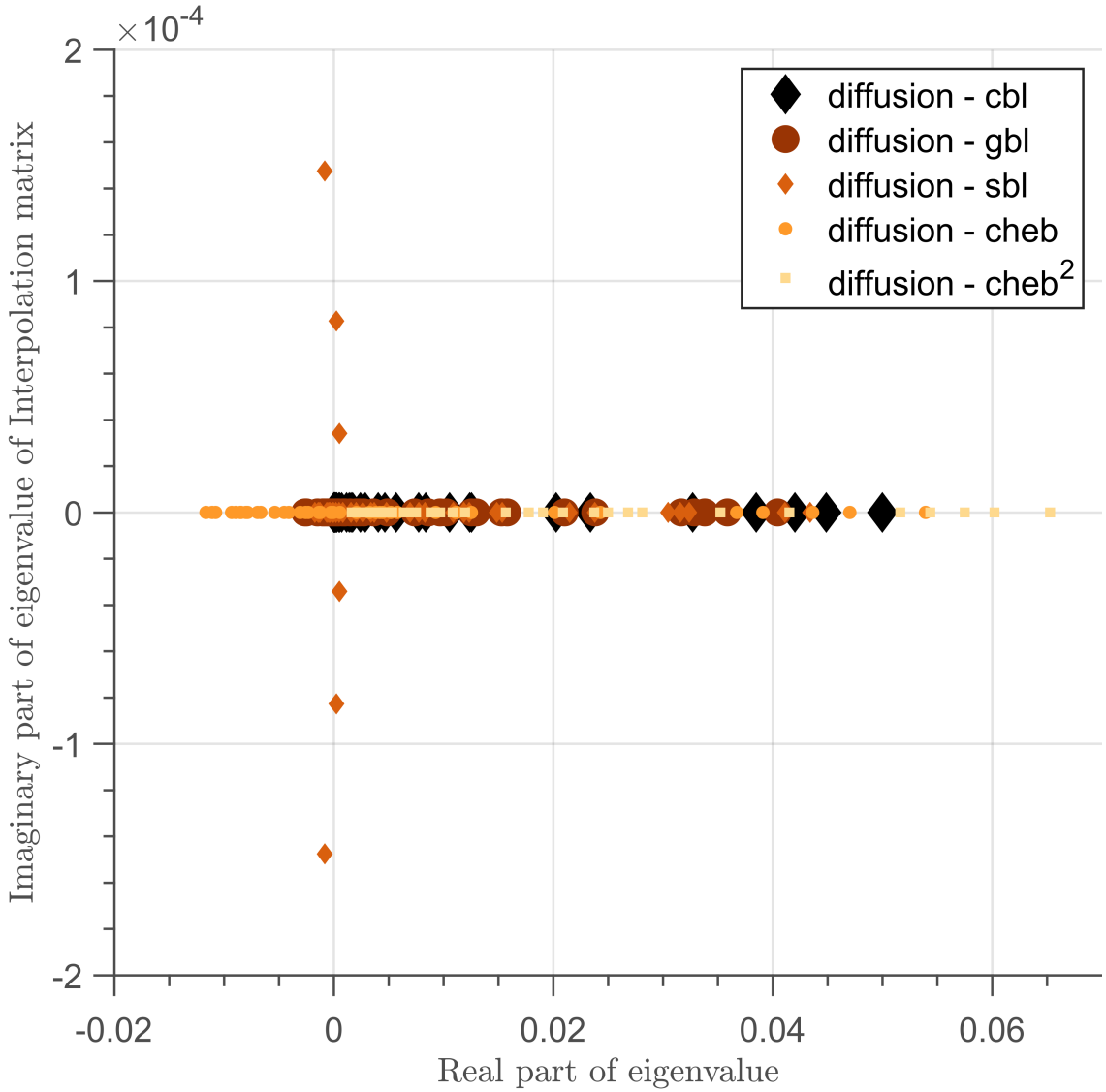


Figure 3: Eigenvalues of $\mathrm{E}_W^* p_{\phi,5}^{(\mathrm{kr})}(\mathbf{L})\mathrm{E}_W$ for five different Krylov subspace methods $\mathrm{kr} \in \{\mathrm{cbl}, \mathrm{gbl}, \mathrm{sbl}, \mathrm{cheb}, \mathrm{cheb}^2\}$, with kernel $\phi(\mathbf{L}) = e^{-t\mathbf{L}}$, $t = 20$.

Proof

Using the same notation as in the description of Algorithm 2, we can rewrite the block vector $p^{(\mathrm{cbl})}_{\phi,m-1}(\mathbf{L})\mathrm{E}_W$ as

$$\mathrm{E}_W^* p^{(\mathrm{cbl})}_{\phi,m-1}(\mathbf{L})\mathrm{E}_W = \mathrm{E}_W^*[\mathrm{Q}_1,\ldots,\mathrm{Q}_m]\phi(\mathbf{H}_m)\mathrm{F}_1 = \mathrm{F}_1^*\phi(\mathbf{H}_m)\mathrm{F}_1.$$

In particular, $\mathrm{E}_W^* p^{(\mathrm{cbl})}_{\phi,m-1}(\mathbf{L})\mathrm{E}_W$ corresponds to the first $N \times N$ principal submatrix of the matrix $\phi(\mathbf{H}_m)$. As $\mathbf{L}$ is symmetric, also the block Lanczos matrix $\mathbf{H}_m$ is symmetric. Further, the block Lanczos relation (2) implies the identity

$$[q_1,\ldots,q_{mN}]^*\mathbf{L}[q_1,\ldots,q_{mN}] = \mathbf{H}_m.$$

These two properties in combination with the Cauchy interlacing theorem guarantee that the spectrum of $\mathbf{H}_m$ is contained in the same interval $[0,\Lambda]$ as the spectrum of $\mathbf{L}$. Thus, if ϕ is positive on $[0,\Lambda]$, the matrix $\phi(\mathbf{H}_m)$ is symmetric and positive definite. Therefore, also the principal submatrix $\mathrm{E}_W^* p^{(\mathrm{cbl})}_{\phi,m-1}(\mathbf{L})\mathrm{E}_W = \mathrm{F}_1^*\phi(\mathbf{H}_m)\mathrm{F}_1$ is symmetric and positive definite.

Error estimates

Theorem 2

Let $\mathbf{L} \in \mathbb{R}^{n \times n}$ *be symmetric with spectrum in* $[0, \Lambda]$. *Then, we get*

$$\left\| \phi(\mathbf{L}) \mathrm{E}_W - p_{\phi,m}^{(\mathrm{cbl})}(\mathbf{L}) \mathrm{E}_W \right\|_F \leq 2\sqrt{N} E_m(\phi),$$

where

$$E_m(\phi) = \min_{p \in \Pi_m} \max_{\lambda \in [0,\Lambda]} |\phi(\lambda) - p(\lambda)|$$

denotes the best approximation error for the function ϕ *in the space of polynomials* Π_m *of degree less or equal to* m *on the interval* $[0, \Lambda]$.

Note: it is also possible to use Chebyshev interpolation for the approximation of $\phi(\mathbf{L})$. In this case, one gets the bound

$$\left\| \phi(\mathbf{L}) \mathrm{E}_W - p_{\phi,m}^{(\mathrm{cheb})}(\mathbf{L}) \mathrm{E}_W \right\|_F \leq \sqrt{N} \left(2 + \tfrac{2}{\pi} \log(m+1)\right) E_m(\phi).$$

Error estimates

Theorem 3

Let ϕ be continuous and positive on $[0, \Lambda]$ with

$$\phi_{\min} = \min_{\lambda \in [0,\Lambda]} |\phi(\lambda)| \quad \textit{and} \quad \phi_{\max} = \max_{\lambda \in [0,\Lambda]} |\phi(\lambda)|.$$

Then, for $m \to \infty$, we have the asymptotic bound

$$\|\mathrm{I}_W x - \mathrm{I}_W x^{(\mathrm{cbl})}\|_2 \leq \frac{\|x\|_2}{\phi_{\min}} \left(1 + \frac{\phi_{\max}}{\phi_{\min}}\right) \left\|\phi(\mathbf{L})\mathrm{E}_W - p_{\phi,m}^{(\mathrm{cbl})}(\mathbf{L})\mathrm{E}_W\right\|_F.$$

Lanczos approximation

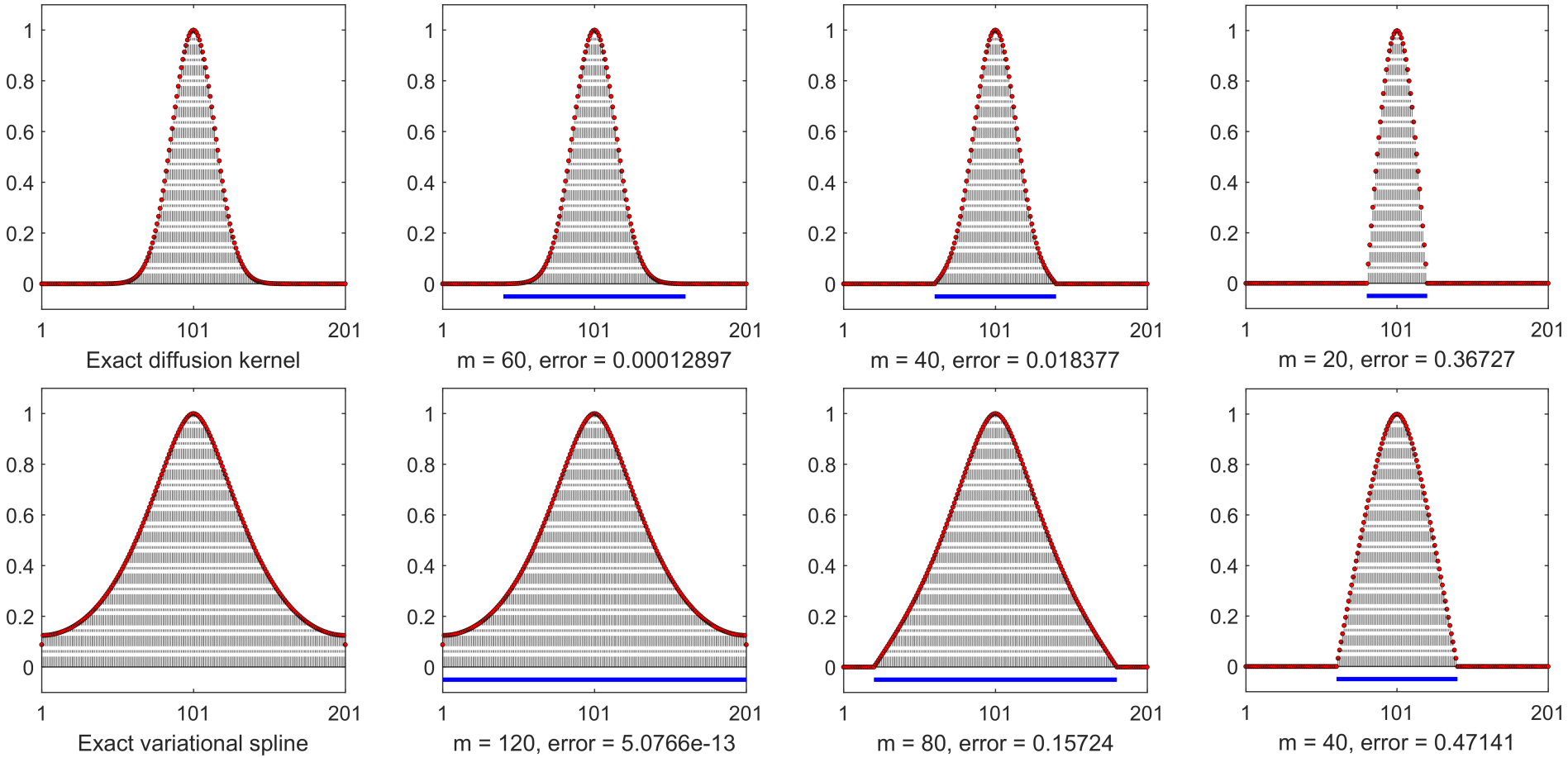


Figure 4: Lanczos approximation of a diffusion and a variation spline kernel on the path graph. The blue line indicates the support of the approximant, the error with respect to the exact kernel column is measured in the uniform norm.

Chebyshev approximation

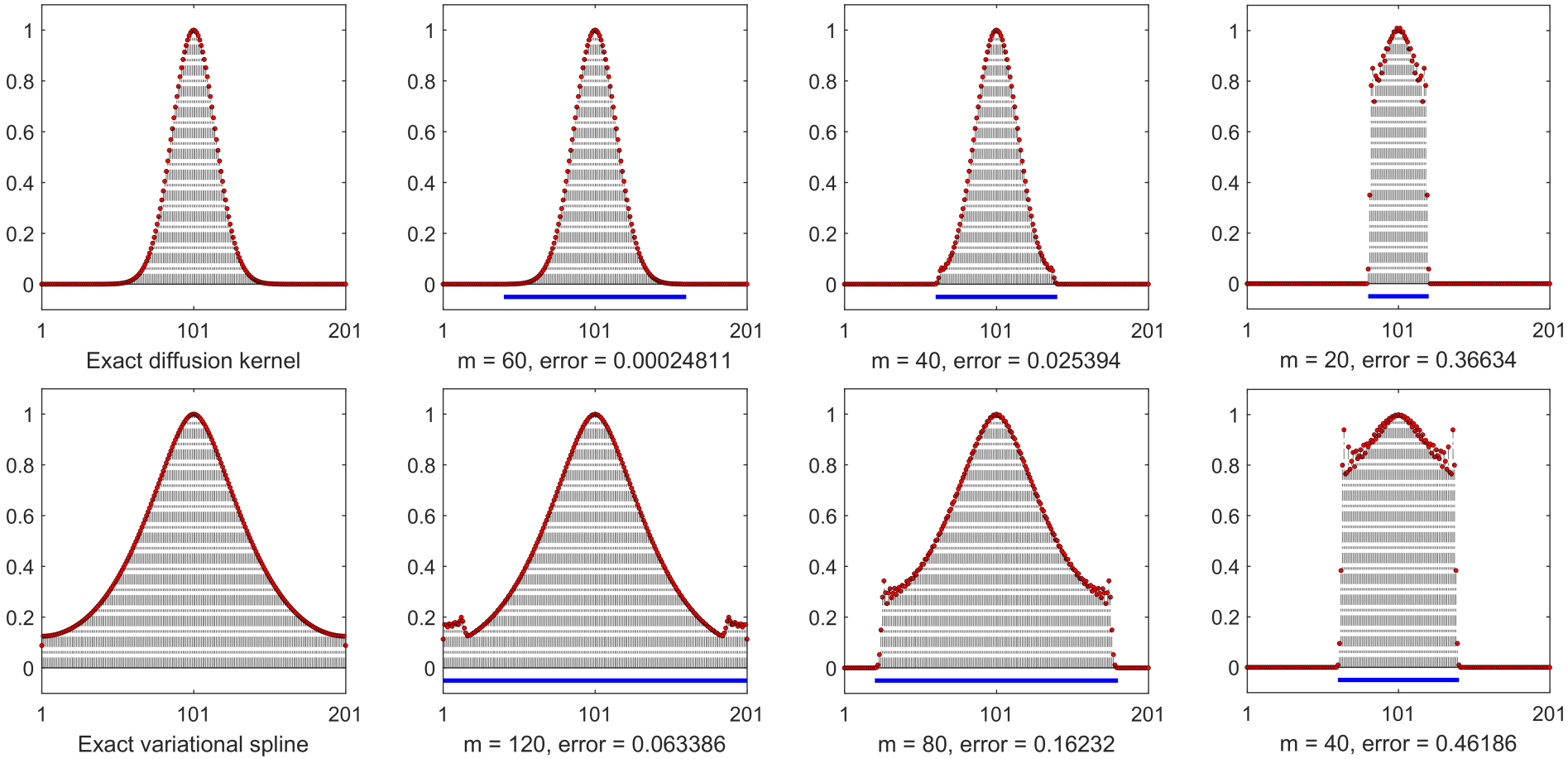


Figure 5: Chebyshev approximation of a diffusion and a variation spline kernel on the path graph. The blue line indicates the support of the approximant, the error with respect to the exact kernel column is measured in the uniform norm.

Experiment

On the bunny graph we calculate two kernel interpolants $\mathrm{I}_W x$ iteratively with five different block Krylov methods.

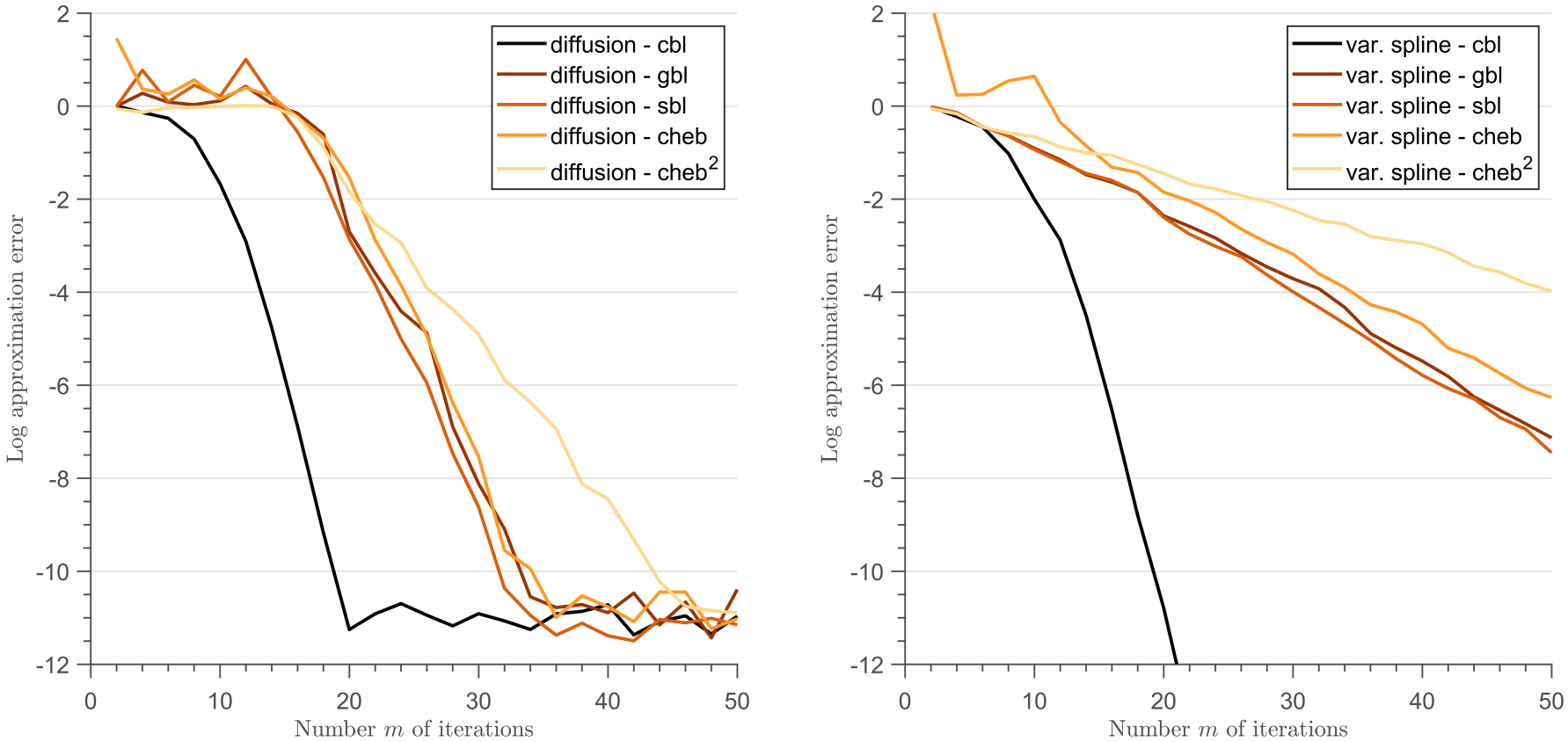


Figure 6: Uniform error $\|\mathrm{I}_W x - \mathrm{I}_W x^{(\mathrm{kr})}\|_\infty$ for five block Krylov methods $\mathrm{kr} \in \{\mathrm{cbl}, \mathrm{gbl}, \mathrm{sbl}, \mathrm{cheb}, \mathrm{cheb}^2\}$ in terms of the iteration numbers m.

Warning

The last figure shows only one side of the medal. Classical block Krylov methods can perform poorly in practice if the block size N gets large.

Operations	cbl	gbl	sbl	cheb
MVs	mN	mN	mN	mN
DOTs	$\mathcal{O}(mN^2)$	$\mathcal{O}(mN)$	$\mathcal{O}(mN)$	-
AXPYs	$\mathcal{O}(mN^2)$	$\mathcal{O}(mN)$	$\mathcal{O}(mN)$	$\mathcal{O}(mN)$
$\phi(\mathbf{H}_m)/c_k(\phi)$	$\mathcal{O}(mN^3)+\mathcal{O}(m^2N^2)$	$\mathcal{O}(m^2)$	$\mathcal{O}(m^2N)$	$\mathcal{O}(m\log m)$

Table 1: Required operations to calculate $p^{(\mathrm{kr})}_{\phi,m}(\mathbf{L})\mathrm{E}_W$ for the Krylov space methods $\mathrm{kr} \in \{\mathrm{cbl}, \mathrm{gbl}, \mathrm{sbl}, \mathrm{cheb}\}$.

Warning

Classical block Krylov methods perform also poorly in terms of memory requirements.

Storage	cbl	gbl	sbl	cheb
$\mathrm{Q}_k / T_k(\mathbf{L})\mathrm{E}_W$	mnN	mnN	mn	$2n$
$\mathbf{H}_m / c_k(\phi)$	$\mathcal{O}(mN^2)$	$\mathcal{O}(m)$	$\mathcal{O}(m)$	m

Table 2: Memory requirements for the calculation of the matrix polynomial $p_{\phi,m}^{(\mathrm{kr})}(\mathbf{L})\mathrm{E}_W$ for the Krylov space methods $\mathrm{kr} \in \{\mathrm{cbl}, \mathrm{gbl}, \mathrm{sbl}, \mathrm{cheb}\}$.

Partition of Unity Methods (PUMs)

To reduce the computational load, a second strategy consists in the usage of a Partition of Unity Method.

Observation:

- On smaller segments of the graph, the computational load to solve the interpolation problem is much smaller.
- In Euclidean setting, the combination of RBFs with PUMs yields significantly sparser system matrices in collocation or interpolation problems, and, therefore, a considerable speed-up of calculations

Idea: calculate local interpolants on small subgraphs of G instead of a global one and merge them via a Partition of Unity.

Advantage for graphs: segmentation algorithms on graphs allow to generate partition of unities in a simple way.

Partition of Unities on a graph

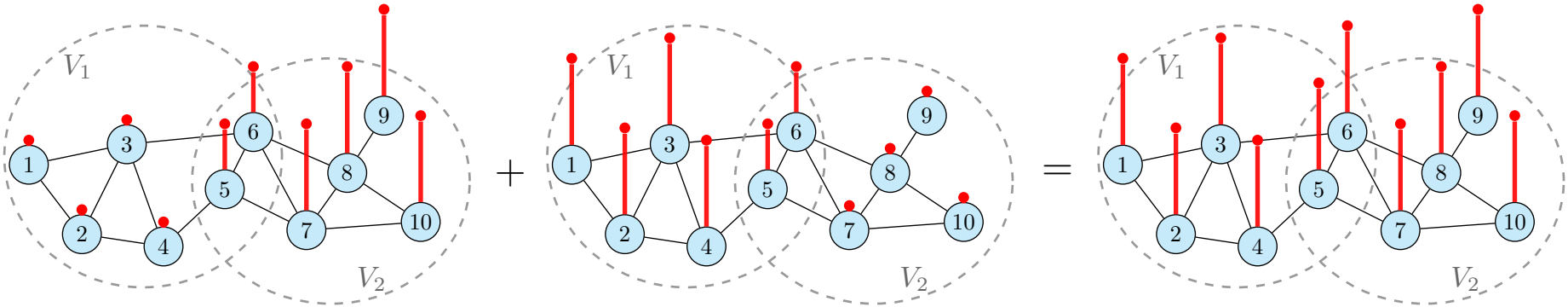


Figure 7: Schematic sketch of a partition of unity on a graph.

Definition 4

Let $\{V_j\}_{j=1}^J$ be a cover of the vertex set V. Then, a partition of unity $\{\varphi^{(j)}\}_{j=1}^J$ subordinate to this cover is a set of functions $\varphi^{(j)} \in \mathcal{L}(G)$, $j \in \{1, \ldots, J\}$, such that

$$\operatorname{supp}(\varphi^{(j)}) \subseteq V_j, \quad \varphi^{(j)} \geq 0, \quad \text{and} \quad \sum_{j=1}^{J} \varphi^{(j)}(\mathrm{v}) = 1 \quad \text{for all } \mathrm{v} \in V.$$

Construction of the partition of unity

We construct a cover $\{V_j\}_{j=1}^J$ of the node set V and a partition of unity $\{\varphi^{(j)}\}_{j=1}^J$ subordinate to $\{V_j\}_{j=1}^J$ by the following three operations:

a) We use a (modified) J-center clustering algorithm to decompose V into J disjoint clusters C_j. For this we use the graph geodesic as a distance between two nodes.

b) We augment the clusters C_j with neighboring nodes and obtain a cover of overlapping subdomains V_j, $j \in \{1, \ldots, J\}$.

c) We use Shepard weight functions on the cover $\{V_j\}_{j=1}^J$ to obtain a desired partition of unity $\{\varphi^{(j)}\}_{j=1}^J$.

a) J-center clustering for graphs

Algorithm 3: Greedy J-center clustering based on interpolation nodes

Input: The interpolation nodes $W = \{\mathrm{w}_1, \ldots, \mathrm{w}_N\}$, a starting center $\mathrm{q}_1 \in W$ and the number J of partitions.

for $j = 2$ *to* J **do**

select a center $\mathrm{q}_j = \operatorname{argmax}_{\mathrm{w} \in W} \min_{\mathrm{q} \in \{\mathrm{q}_1, \ldots, \mathrm{q}_{j-1}\}} \mathrm{d}(\mathrm{q}, \mathrm{w})$ farthest away from $Q_{j-1} = \{\mathrm{q}_1, \ldots, \mathrm{q}_{j-1}\}$.

Calculate the J clusters

$$\mathrm{C}_j = \left\{ \mathrm{v} \in V \ : \ \mathrm{d}(\mathrm{q}_j, \mathrm{v}) = \min_{\mathrm{q} \in \{\mathrm{q}_1, \ldots, \mathrm{q}_J\}} \mathrm{d}(\mathrm{q}, \mathrm{v}) \right\}, \quad j \in \{1, \ldots, J\}.$$

Return centers $\mathrm{Q}_J = \{\mathrm{q}_1, \ldots, \mathrm{q}_J\}$ and clusters $\{\mathrm{C}_1, \ldots, \mathrm{C}_J\}$.

b) Augmentation of the clusters

We enlarge the clusters C_j to obtain overlapping subdomains V_j.

Method: add all vertices to C_j that have a graph distance to C_j less than or equal to $r \geq 0$.

Algorithm 4: Augmentation of clusters with neighbors of distance r

---

Input: A cluster C_j and an enlargement distance $r \geq 0$.

Add all nodes $\mathrm{v} \in V$ with a graph distance less than r to C_j, i.e.,

$$V_j = \mathrm{C}_j \cup \{\mathrm{v} \in V \mid \mathrm{d}(\mathrm{v}, \mathrm{w}) \leq r,\ \mathrm{w} \in \mathrm{C}_j\}.$$

Return subdomain V_j.

Note: although possible, it is not recommendable to use the disjoint clusters C_j, $j \in \{1, \ldots, J\}$, directly as subdomains for the PUM, as the topological information of the graph G gets lost.

c) Creation of Partition of Unity with Shepard weights

A simple construction principle to generate a partition of unity $\{\varphi^{(j)}\}_{j=1}^{J}$ is based on Shepard weights:

If $\psi^{(j)}$, $j \in \{1, \ldots, J\}$, denote nonnegative weight functions on V with $\mathrm{supp}(\psi^{(j)}) \subseteq V_j$ and $\sum_{j=1}^{J} \psi^{(j)} > 0$, then the functions

$$\varphi^{(j)}(\mathrm{v}) = \frac{\psi^{(j)}(\mathrm{v})}{\sum_{j=1}^{J} \psi^{(j)}(\mathrm{v})}, \quad \mathrm{v} \in V,$$

form a partition of unity subordinate to $\{V_j\}_{j=1}^{J}$.

Example: We consider the disjoint clusters C_j as subsets of the augmented domains V_j. Then, choosing $\psi^{(j)}(\mathrm{v}) = \mathbf{1}_{\mathrm{C}_j}(\mathrm{v})$, the corresponding partition of unity is given as

$$\varphi^{(j)}(\mathrm{v}) = \psi^{(j)}(\mathrm{v}) = \mathbf{1}_{\mathrm{C}_j}(\mathrm{v}).$$

Example

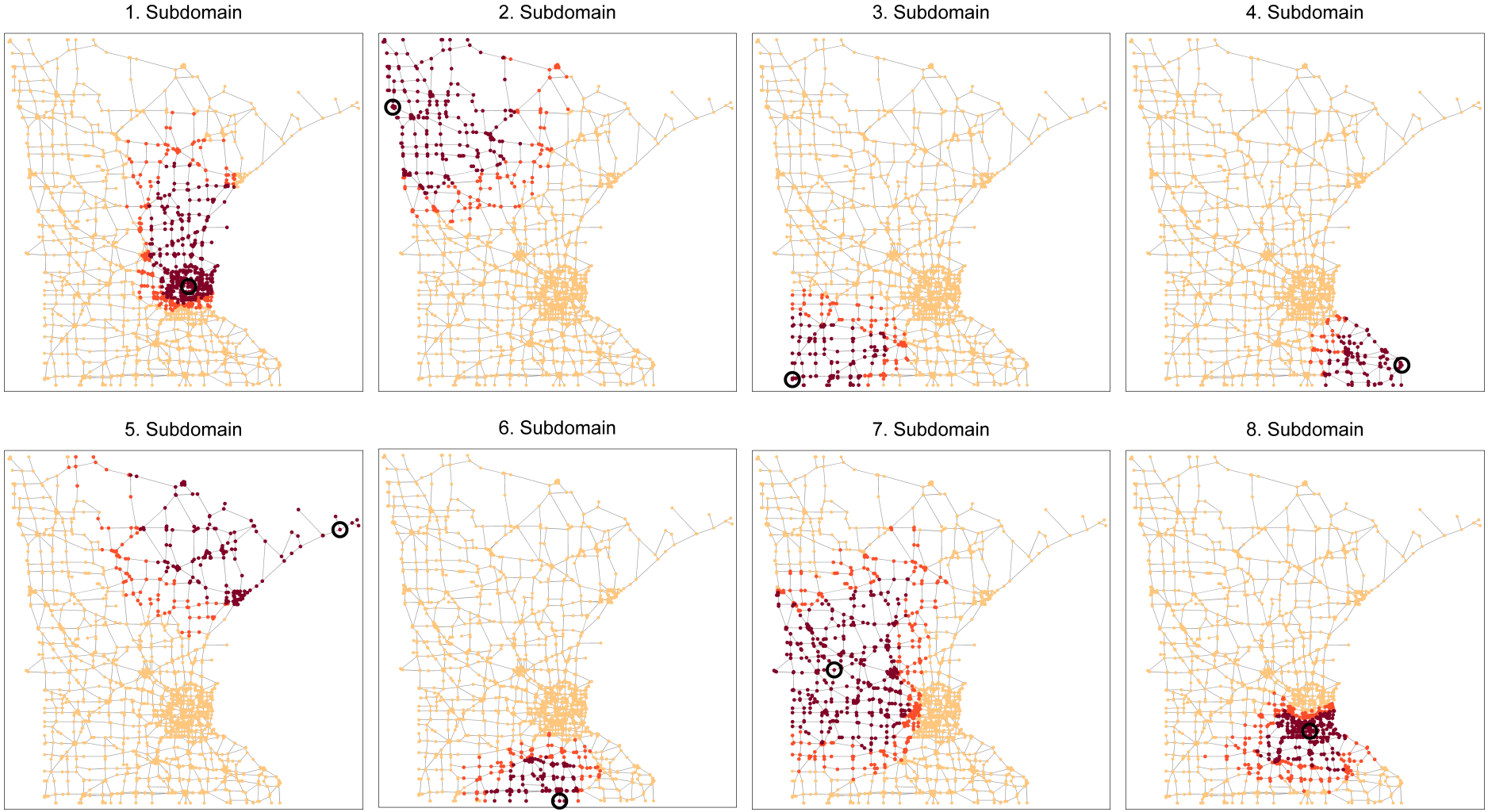


Figure 8: Partition of the Minnesota graph in $J = 8$ subdomains via reduced J-center clustering (dark domains) and domain augmentation (red nodes, $r = 8$). The ringed nodes indicate the centers of the subdomains.

PUM for Kernel-based interpolation

Algorithm 5: Kernel-PUM interpolation on graphs

Input: (i) Samples $x(\mathrm{w}_1), \ldots, x(\mathrm{w}_N) \in \mathbb{R}$ at the interpolation nodes $W = \{\mathrm{w}_1, \ldots, \mathrm{w}_N\}$,
(ii) Number J of subdomains.

1.a) Use J-center clustering to decompose V into J disjoint clusters C_j.

1.b) Create J subdomains V_j by augmenting the clusters C_j into J overlapping subdomains $V_j = \{\mathrm{v}_{j_1}, \ldots, \mathrm{v}_{j_{n_j}}\}$ with n_j elements.

1.c) Generate Partition of Unity $\{\varphi^{(j)}\}_{j=1}^J$ subordinate to the cover $\{V_j\}_{j=1}^J$ such that

$$\operatorname{supp}(\varphi^{(j)}) \subseteq V_j, \ \varphi^{(j)} \geq 0, \text{ and } \sum_{j=1}^{J} \varphi^{(j)}(\mathrm{v}) = 1 \text{ for all } \mathrm{v} \in V.$$

PUM for Kernel-based interpolation

Algorithm 4: Kernel-PUM interpolation on graphs

2. Extract local graphs and local Laplacians $\mathbf{L}^{(j)}$ For all subdomains V_j, $j \in \{1, \dots, J\}$, generate the subgraphs $G_j = (V_j, E_j)$, with the node sets V_j, the edges $E_j = \{(\mathrm{v}, \mathrm{w}) \in E \mid \mathrm{v}, \mathrm{w} \in V_j\}$ and the local Laplacian

$$\mathbf{L}^{(j)} = \begin{pmatrix} \mathbf{L}_{j_1, j_1} & \cdots & \mathbf{L}_{j_1, j_{n_j}} \\ \vdots & \ddots & \vdots \\ \mathbf{L}_{j_{n_j}, j_1} & \cdots & \mathbf{L}_{j_{n_j}, j_{n_j}} \end{pmatrix}.$$

If required, calculate the spectral decomposition $\mathbf{L}^{(j)} = \mathbf{U}^{(j)} \mathbf{M}_{\lambda^{(j)}} \mathbf{U}^{(j)*}$ to define a local graph Fourier transform on the subgraphs G_j.

3.a) Construct a local kernel $K^{(j)}(\mathrm{v}, \mathrm{w})$ on the subgraph G_j. One possibility is to use the variational spline kernel

$$\mathbf{K}^{(j)} = (\epsilon \mathbf{I}_{n_j} + \mathbf{L}^{(j)})^{-s} = \sum_{k=1}^{n_j} \frac{1}{(\epsilon + \lambda_k^{(j)})^s} u_k^{(j)} u_k^{(j)*}, \quad \epsilon > -\lambda_1^{(j)}, \ s > 0.$$

PUM for Kernel-based interpolation

Algorithm 4: Kernel-PUM interpolation on graphs

3.b) For the sampling sets $W_j = W \cap V_j$ with $N_j \geq 1$ elements **solve**

$$\mathbf{K}_{W_j}^{(j)} c^{(j)} = x(W_j).$$

3.c) Calculate the local kernel interpolants $\mathrm{I}_{W_j} x$ on G_j:

$$\mathrm{I}_{W_j} x(\mathrm{v}) = \sum_{i=1}^{N_j} c_i^{(j)} K^{(j)}(\mathrm{v}, \mathrm{w}_{j_i}).$$

4. A **global kernel-PUM interpolant** on G is then given as

$$\mathrm{I}_W x(\mathrm{v}) = \sum_{j=1}^{J} \varphi^{(j)}(\mathrm{v}) \mathrm{I}_{W_j} x(\mathrm{v}).$$

Example

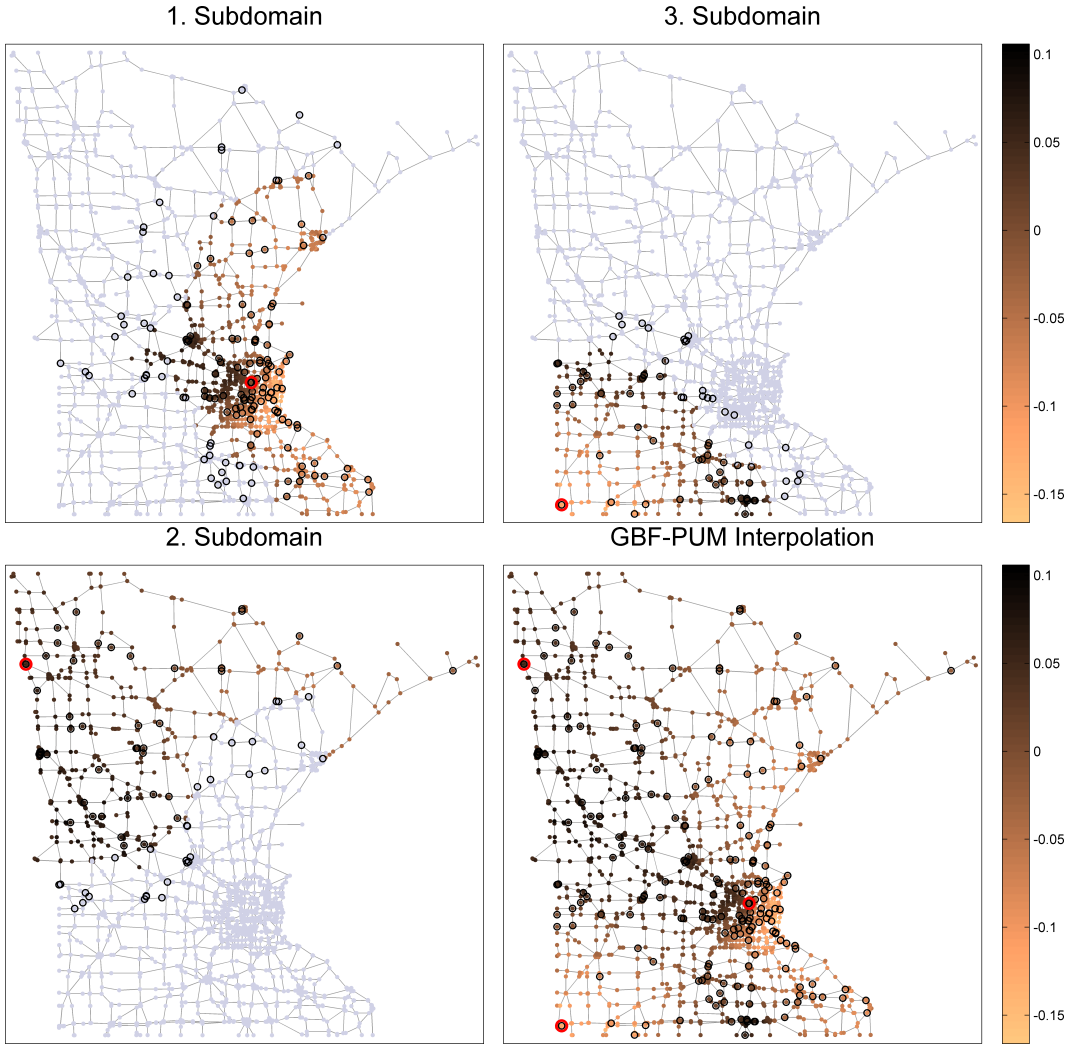


Figure 9: PUM interpolation based on local interpolants with the variational spline kernel ($\epsilon = 0.001$, $s = 2$) on $J = 3$ subdomains and 200 samples (black ringed nodes). The red ringed nodes denote the centers q_j of the subdomains.

Important parameters for PUM interpolation

- J: the number of domains in the partition of unity. The applied greedy J-center clustering generates a quasi-uniform clustering of the given graph (can be verified theoretically).
- r: augmentation distance for the initial clusters. It determines how strong the domains of the PU overlap.
- ϵ, s: parameters of the variational spline kernel, s describes the smoothness of the kernel, ϵ the localization on the graph domain.

Setting for numerical tests:

- Minnesota graph with $n = 2642$ vertices and 3304 edges.
- Variational spline kernel with $s = 2$ and $\epsilon = 0.001$.
- Test function for interpolation is very smooth, a linear combination of the first 10 eigenvectors of the normalized graph Laplacian.

Some experiments

1. Observation: Compared to a global kernel interpolation a PUM interpolant is less accurate for smooth functions but requires less time.

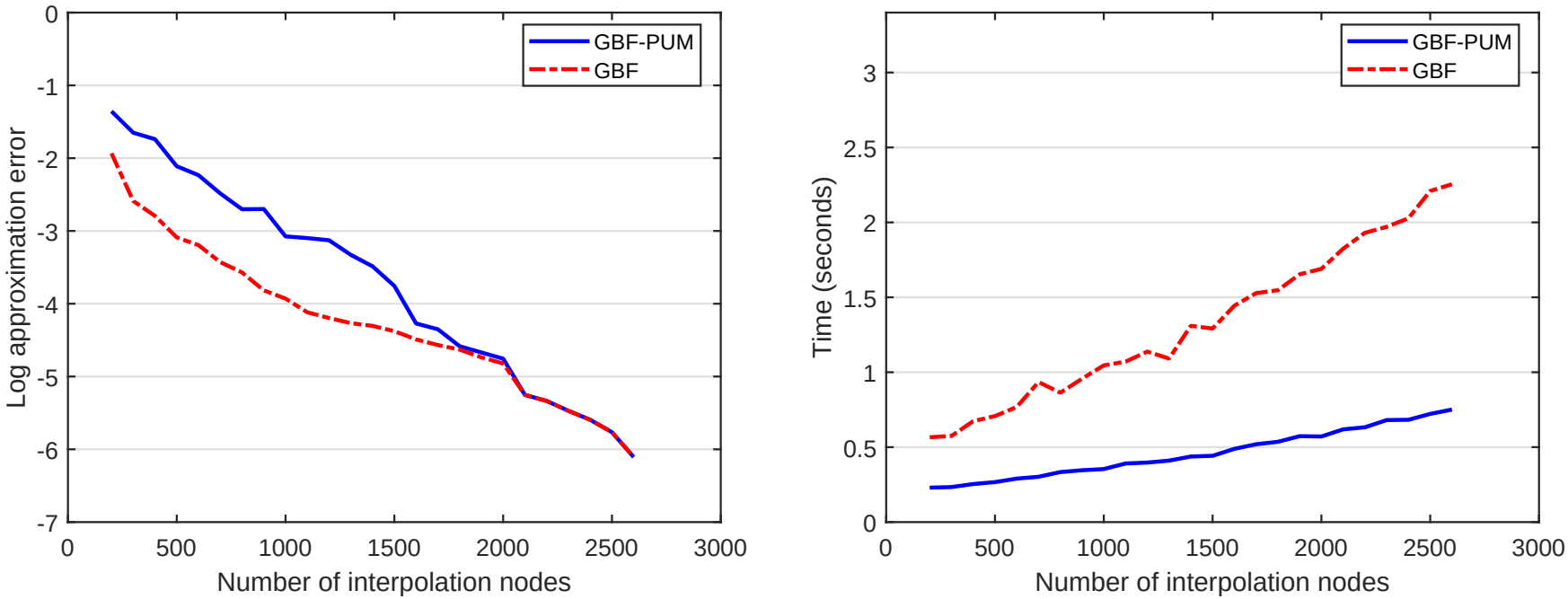


Figure 10: RRMSEs and CPU times (in seconds) obtained for GBF-PUM (with $J = 8$ and $r = 8$) and the global GBF scheme in terms of N interpolation nodes. We used the variational spline kernel with $\epsilon = 0.001$ and $s = 2$.

Some experiments

2. Observation: for the interpolation of smooth signals, a larger overlap parameter r is useful (but the computational cost gets larger).

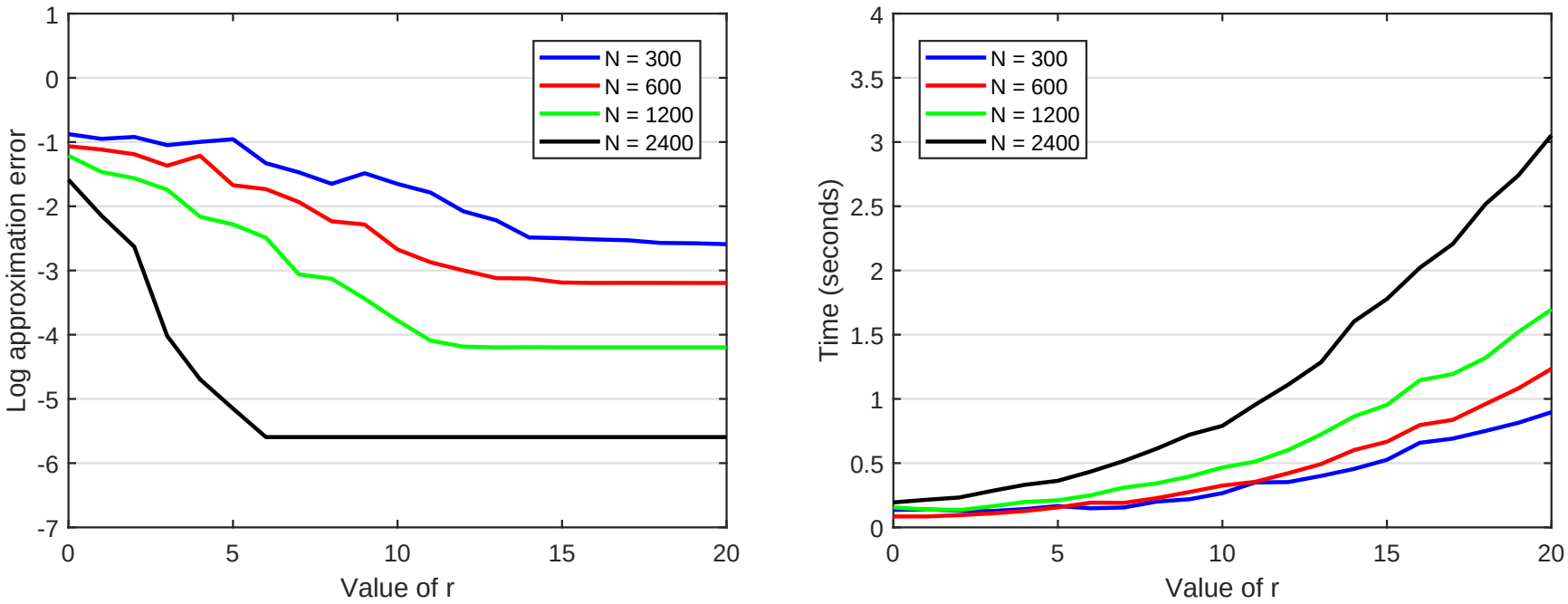


Figure 11: Analysis of the impact of the parameter r on errors (left) and times (right) obtained for the GBF-PUM with $J = 8$ in terms of N interpolation nodes. We use the variational spline kernel with $\epsilon = 0.001$ and $s = 2$.

Literature

Some of my works related to this talk:

[1] Erb, W. Graph signal interpolation with Positive Definite Graph Basis Functions. *Appl. Comput. Harmon. Anal. 60* (2022), 368-395.

[2] Erb, W. Krylov subspace methods to accelerate kernel machines on graphs. *Advances in Computational Science and Engineering 1*, 1 (2023), 59-81.

[3] Cavoretto, R., De Rossi, A., and Erb, W. Partition of Unity Methods for Signal Processing on Graphs. *J. Fourier Anal. Appl. 27* (2021), Art. 66.

[4] Cuomo, S., Erb, W., Santin, G. Kernel-Based Models for Influence Maximization on Graphs based on Gaussian Process Variance Minimization. *J. Comput. Appl. Math. 423* (2023), 114951.

Further references related to graph signal processing and kernel approximation:

[5] Kondor, R.I., Lafferty, J. Diffusion kernels on graphs and other discrete input spaces. in *Proc. of the 19th. Intern. Conf. on Machine Learning ICML02* (2002), 315-322.

[6] Pesenson, I.Z. Variational Splines and Paley-Wiener Spaces on Combinatorial Graphs. *Constr. Approx. 29*, 1 (2009), 1–21.

[7] Shuman, D.I., Ricaud, B., and Vandergheynst, P. Vertex-frequency analysis on graphs. *Appl. Comput. Harm. Anal. 40*, 2 (2016), 260–291.

[8] Stanković, L., Daković, L., and Sejdić, E. Introduction to Graph Signal Processing. In *Vertex-Frequency Analysis of Graph Signals*, Springer, (2019), 3–108.

[9] Ward, J.P., Narcowich, F.J., and Ward, J.D., Interpolating splines on graphs for data science applications. *ACHA 49*, 2 (2020), 540–557.

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università degli Studi di Padova

Thanks a lot for your attention!