A class of Laplacian multiwavelets bases for high-dimensional data

Nir Sharon[a,*], Yoel Shkolnisky[a]

[a]*School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel*

Abstract

We introduce a framework for representing functions defined on high-dimensional data. In this framework, we propose to use the eigenvectors of the graph Laplacian to construct a multiresolution analysis on the data. We assume the dataset to have an associated hierarchical tree partition, together with a function that measures the similarity between pairs of points in the dataset. The construction results in a one parameter family of orthonormal bases, which includes both the Haar basis as well as the eigenvectors of the graph Laplacian, as its two extremes. We describe a fast discrete transform for the expansion in any of the bases in this family, and estimate the decay rate of the expansion coefficients. We also bound the error of non-linear approximation of functions in our bases. The properties of our construction are demonstrated using various numerical examples.

Keywords: High dimensional data, graph Laplacian, multiwavelets, multiresolution analysis

1. Introduction

Expanding a function in an orthonormal basis is one of the most basic tools in mathematics. Such an expansion transforms a given function, given either as a mathematical object or as a set of samples stored in a computer, into a set of coefficients. Then, instead of analyzing the original function, one analyzes the resulting expansion coefficients. This procedure arises in many areas such as harmonic analysis, numerical analysis, partial differential equations, and signal processing, to name a few. Examples for tasks that are easily implemented using this approach include denoising, compression, and extrapolation. This approach goes back to classical harmonic analysis, where the properties of a function are analyzed by inspecting its expansion coefficients into a Fourier series. Other more modern bases include wavelet bases, splines, and orthogonal polynomials.

Although this approach turned out to be very powerful, most tools and theory are only

[*]Corresponding author
Email address: `nirsharo@post.tau.ac.il` (Nir Sharon)

available in one dimension. Extensions to higher dimensions are commonly derived by tensor products of one-dimensional elements, which usually do not exploit any special structure of the underlying domain. Moreover, even the one-dimensional theory cannot be applied to arbitrary sets on the line, and is typically restricted to an interval or the entire line. For example, it cannot be applied when our domain is simply a finite scattered set of points on the real line.

In recent years, the need to analyze data in high dimensions has grown rapidly. Moreover, most often the data is simply some finite set of high-dimensional vectors, without any of the rich Euclidean structure inherent into the classical tools. Analyzing functions defined on such general datasets arises naturally, for example, in meteorology [17], gene research [32], medical imaging [31], etc.

This practical need for analyzing large high-dimensional datasets has motivated extensive research in past years. These studies resulted in many methods for data representation and dimensionality reduction such as [6, 39]. In other cases, these studies gave rise to generalizations of the Euclidean space construction of wavelets and wavelet packets. Such a wavelets analogue on manifolds and graphs, based on the diffusion operator, is suggested in [10, 15, 42]. These diffusion based constructions exploit the decay of the spectrum of the diffusion operator and its powers. A different approach for graphs was introduced in [27], where the wavelet operator is defined using the localization and scaling of the graph Laplacian. Another construction for non-Euclidean setting is presented in [3] for general spaces of homogeneous type.

The problem of basis construction is also known as dictionary learning in the machine learning community, where one often uses a given hierarchy tree on the data to construct a multiresolution analysis [8, 30]. Thus, constructing a basis is related to finding optimal trees [9], and is also known as data-adaptive signal representation [22].

Two other important constructions available today for high-dimensional data in almost arbitrary domains, modelled as graphs, are the graph Laplacian (e.g., [16, 38]) and the Haar basis on graphs [24]. The graph Laplacian approach uses as basis functions the eigenvectors of the graph Laplacian over the data. If the data are uniformly sampled from some Euclidean domain, then, under certain conditions, as the number of data points goes to infinity, these eigenvectors converge to the eigenfunctions of the Laplacian over the underlying domain [41]. For example, if the data are sampled from the unit circle, then the eigenvectors would converge to sine and cosine functions. Thus, this approach results in an analogue of Fourier basis for general domains [37]. This approach is described in details in Section 2.1.

The second basis available for general data is the Haar basis on graphs. The continuous Haar basis is considered as the simplest wavelet basis, and is defined using dilation and translation of a piecewise constant function. To extend the Haar basis to general datasets, one is required to replace the standard dyadic partition of the interval with some hierarchical partition of the data (see Subsection 2.3). Once such a partition is given, the basis is constructed by taking the characteristic functions over the elements of the partition, and applying an orthogonalization procedure to these functions. Such a construction is described in [24].

On one hand we have the eigenvectors of the graph Laplacian whose support cannot be controlled and can be shown to be “smooth” under an appropriate definition [4]. On the other hand we have the Haar basis whose basis elements are localized piecewise constant functions. An open question is whether there is any basis “in between”.

In this paper, we suggest a family of bases, named Laplacian multiwavelets, which can be constructed for any set of N data points. We require the dataset to have an associated hierarchical tree partition, together with a function that measures the similarity between pairs of points in the dataset. These requirements will be made precise in Section 3. The family of bases is parameterized by one integer parameter $1 \leq k \leq N$, where $k = 1$ corresponds to the Haar basis defined in [24], and $k = N$ corresponds to the eigenvectors of the graph Laplacian (Fourier basis). Intermediate values of k correspond to various degrees of “generalized vanishing moments” as explained in Section 5 and Subsection 6.1. In particular, we show that generalized vanishing moments are related to the decay rate of the expansion coefficients in our basis. We also show through numerical examples that by tuning the value of k, our basis is capable of efficiently representing both slowly varying functions, just like the graph Laplacian basis [37], as well as highly oscillatory functions, just like the Haar basis [24]. We also present an algorithm that evaluates all N basis functions in $\mathcal{O}(k^2 N \log N + T(N,k) \log N)$ operations, where N is the number of data points and $T(N,k)$ is the runtime of extracting the first k eigenvectors of the graph Laplacian over the data points.

The paper is organized as follows. Section 2 introduces the required mathematical background. Section 3 defines formally the basis construction problem, as well as the requirements from such a basis. Section 4 provides an informal description of the Laplacian multiwavelets, followed by a detailed description in Section 5. In Section 6 we discuss the notion of generalized vanishing moments and study the relation between this notion and the decay of the expansion coefficients. In addition, we define the smoothness of a function with respect to our basis and deduce its approximation rate. In Section 7 we present a few numerical examples to illustrate the Laplacian multiwavelet bases. Finally, we give some concluding remarks in Section 8.

2. Mathematical preliminaries

In this section we briefly introduce the mathematical tools used in the construction of the Laplacian multiwavelets. These tools are the graph Laplacian, multiresolution analysis, and hierarchical data partitioning.

2.1. The graph Laplacian

As a main tool for our construction of the Laplacian multiwavelets we use the eigenvectors of the graph Laplacian, which are widely used in machine learning for dimensionality reduction, semi-supervised learning and spectral clustering [5].

Let $\mathcal{X} = \{x_i\}_{i=1}^{N}$ be a set of N data points. For simplicity, we assume that $x_i \in \mathbb{R}^D$, although this assumption can be easily removed to fit more general spaces. We also assume that we are given a symmetric and non-negative function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_+$. In the machine learning community, $K(x, y)$ is known as a kernel function, which is used to measure the similarity between pairs of data points x and y. A popular choice is, for example, $K(x, y) = e^{-x-y^2/2\varepsilon}$, where ε is a parameter that determines the width of the kernel.

To construct the graph Lapacian, we start by defining the symmetric matrix W of size $N \times N$ given by

$$W_{ij} = K(x_j, x_j).$$

The matrix W defines a complete undirected weighted graph on the points $\mathcal{X}$, where the weight on the edge between x_i and x_j is W_{ij}. This matrix is known as the adjacency matrix of the graph. Thus, whenever a function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ is associated with the set $\mathcal{X}$, we say that it induces a graph structure on $\mathcal{X}$.

Next, we define the $N \times N$ diagonal matrix B given by

$$B_{ii} = \sum_{j=1}^{N} W_{ij}.$$

The graph Laplacian L is then defined by

$$L = B^{-1}W - I, \tag{1}$$

where I is the $N \times N$ identity matrix.

If the points in $\mathcal{X}$ are independent identically distributed samples from the uniform distribution over a Riemannian manifold $\mathcal{M}$, then, for any smooth function $f : \mathcal{M} \to \mathbb{R}$ it holds that [41],

$$\frac{1}{\varepsilon} \sum_{j=1}^{N} L_{i,j} f(x_j) = \Delta_{\mathcal{M}} f(x_i) + \mathcal{O}\left(\frac{1}{N^{\frac{1}{2}} \varepsilon^{\frac{1}{2}+\frac{d}{4}}}, \varepsilon \right),$$

where $\Delta_{\mathcal{M}}$ is the Laplace-Beltrami operator on the manifold $\mathcal{M}$. This theorem justifies the name graph Laplacian given to the matrix L in (1). Moreover, we deduce that the graph Laplacian depends only on the intrinsic dimension of the manifold, and not on the dimension of the ambient space. In [7] Belkin and Niyogi proved that the first few eigenvectors of the graph Laplacian are discrete approximations of the eigenfunctions of the Laplace-Beltrami operator for data uniformly sampled from $\mathcal{M}$. The pseudo-code for computing the first k eigenvectors of the graph Laplacian is summarized in Algorithm 1.

Algorithm 1 GLE (Graph Laplacian Eigenvectors)

```
Input: A set $\mathcal{X} = \{x_i\}_{i=1}^N$ of data points. An integer $1 \leq k \leq N$.
Output: First $k$ eigenvectors of the graph Laplacian over $\mathcal{X}$.
 1: for $i, j = 1$ to $N$ do
 2:     $W_{i,j} \leftarrow K(x_i, x_j)$
 3: end for
 4: for $i = 1$ to $N$ do
 5:     $B_{i,i} \leftarrow \sum_{j=1}^N W_{i,j}$.
 6: end for
 7: $A \leftarrow B^{-\frac{1}{2}} W B^{-\frac{1}{2}}$.
 8: $[U, \Lambda] \leftarrow \mathrm{eig}(A)$ {Diagonalize $A$ s.t. $A = U \Lambda U^T$}
 9: $U \leftarrow B^{-1/2} U(:, 1:k)$.
10: return $U$
```

There are two implementation issues that need to be considered when implementing Algorithm 1. First, as described in Algorithm 1, the graph Laplacian is a dense matrix. This is computationally prohibitive if N is very large. To reduce the time and memory complexity of the construction, one possible solution is to set W_{ij} to be nonzero only for some small predetermined number of nearest neighbours of x_i. These nearest neighbours can be found using a fast k-NN algorithm [28]. Choosing a fixed neighbourhood size results in a number of nonzero entries in W which is linear in N, and thus constructing W in Algorithm 1 would be linear in N. Second, the matrix L as defined in (1) is not symmetric. Numerically, it is better to work with symmetric matrices, and thus, instead of computing the eigenvectors corresponding to the smallest eigenvalues of the matrix L, we compute the eigenvectors corresponding to the largest eigenvalues of the symmetric matrix $B^{-\frac{1}{2}} W B^{-\frac{1}{2}}$ (B is the diagonal matrix defined in line 5 of Algorithm 1). This matrix is similar to $B^{-1}W$, and the eigenvectors of the matrix L can be computed from the eigenvectors of $B^{-\frac{1}{2}} W B^{-\frac{1}{2}}$ by multiplying the letter ones by the diagonal matrix $B^{-\frac{1}{2}}$.

2.2. Multiresolution analysis (MRA)

The classical multiresolution analysis in signal processing is a framework for constructing orthogonal wavelets. In this classical setting one uses a translation operator to define an approximation space and a dilation operator to define details spaces. Obviously, this setting is not applicable for general datasets, as it relies on the special geometry of the real line. Thus, for high-dimensional or complicated domains, generalized definitions are needed.

Recent works [24, 30, 36] use a discrete generalization of the classical MRA, which fits our construction as well. To use the generalized MRA method to derive an orthogonal basis for a vector space V, we start by defining nested approximation spaces

$$V_0 \subset V_1 \subset \cdots \subset V_m = V \ . \tag{2}$$

As in the classical MRA, approximating a function at resolution j means projecting it on V_j. Thus, the approximation, also called the resolution of the function, is more accurate as j increases.

Next, we construct the orthogonal complement spaces W_j that are given by

$$V_j \oplus W_j = V_{j+1}, \quad V_j \perp W_j, \tag{3}$$

and thus satisfy $W_j \subset V_{j+1}$ and $W_j \perp W_{j+1}$. Hence, one gets inductively,

$$V_j = V_0 \oplus W_0 \oplus W_1 \oplus \cdots \oplus W_{j-1} \ . \tag{4}$$

The latter suggests a basis for any prescribed resolution. In particular, for a full reconstruction in V we use $j = m$.

The term multiwavelets refers to the case where V_0 is defined using several functions, and not just one function as in classical wavelets. In our construction we use the eigenvectors of the graph Laplacian to define V_0. Thus, we refer to our construction as a class of Laplacian multiwavelets bases.

In the algorithm presented in the next sections, with a slight abuse of notation, we use the notation V_j to denote a matrix whose columns span the approximation space V_j. Therefore, the complement space W_j is calculated via an orthogonalization procedure applied to V_j. It is worth mentioning here that the setting discussed in this subsection is also used to define the so-called induced MRA when a hierarchy tree (see next subsection) is available [22, 30, 33].

2.3. Hierarchy tree partition

Hierarchical clustering is widely used to cluster images, biological data, networks data, etc. into groups based on their similarity [44]. Such a hierarchy among data points is naturally described using trees, and is defined as follows.

Given a set $\mathcal{X}$ of data points, a corresponding hierarchy tree $T_{\mathcal{X}} = \{\mathcal{X}_{l,n}\}_{l=0,\dots,m}^{n\in\mathcal{I}_l}$ is a connected acyclic graph, where each node $\mathcal{X}_{l,n} \in T$ is a subset of $\mathcal{X}$, $\mathcal{I}_l$ is a finite set of indices, and $\mathcal{X}_{l,n}$ stands for the n-th set at the l-th level of the tree. The index of the last level m is called the depth of the tree, and is denoted by $\mathrm{depth}(T_{\mathcal{X}})$. Any fixed level l of the tree $T_{\mathcal{X}}$ is a partition of $\mathcal{X}$,

$$\mathcal{X} = \bigcup_{n \in \mathcal{I}_l} \mathcal{X}_{l,n} \ .$$

In addition, we have the following conditions on $T_{\mathcal{X}}$:

1. The only node at the first level in the tree is the root, namely, $\mathcal{X}_{0,0} = \mathcal{X}$.
2. Nodes of the same level are disjoint, namely,

$$\mathcal{X}_{l,n_1} \cap \mathcal{X}_{l,n_2} = \emptyset \ , \quad n_1 \neq n_2 \ , \quad n_1, n_2 \in \mathcal{I}_l \ . \tag{5}$$

3. Edges in the tree exist only between nodes of adjacent levels. This connection between levels defines the hierarchy of the data: a node $\mathcal{X}_{l+1,i'}$ of the $l+1$-th level that is connected with an l-th level node $\mathcal{X}_{l,i}$ satisfies the inclusion relation $\mathcal{X}_{l+1,i'} \subset \mathcal{X}_{l,i}$. The nodes of level $l+1$ that are connected to $\mathcal{X}_{l,i}$ are termed the children of $\mathcal{X}_{l,i}$. We denote the set of all children of a certain node $\mathcal{X}_{l,i}$ by

$$\mathrm{child}(\mathcal{X}_{l,i}) = \{\mathcal{X}_{l+1,i'} : \mathcal{X}_{l+1,i'} \subset \mathcal{X}_{l,i}\} \ . \tag{6}$$

An hierarchy tree for data points on the real line is naturally given by the dyadic partition. For domains of higher dimension, the task of partitioning the data into clusters becomes more complicated. However, clustering algorithms became very popular and many recent studies of data clustering can be applied to set up a hierarchy tree. For a survey see [19, 23] and references therein.

There are a few properties which characterize a good tree for our algorithm. One desirable property is that the partitions that are defined by the tree distinguish between different features of the data. Namely, cluster the data in a reasonable way. Naturally, for general data this property is hard to define rigorously, and so we avoid such a definition here; for more details see, e.g., [13, 40]. Another desirable property is a balanced tree. Denote by $|X|$ the number of elements in a finite set X. Then, a tree $T_{\mathcal{X}} = \{\mathcal{X}_{l,n}\}_{l=0,\dots,m}^{n \in \mathcal{I}_l}$ is balanced if there exist positive constants $1 < c \leq C$ such that

$$\begin{aligned}
&0 \leq |\,\mathrm{child}(\mathcal{X}_{0,0})| \leq C, \\
&c \leq |\,\mathrm{child}(\mathcal{X}_{l,n})| \leq C, \quad 0 < l < m, \quad n \in \mathcal{I}_l, \\
&\mathrm{child}(\mathcal{X}_{m,n}) = \emptyset, \quad \mathcal{X}_{m,n} \neq \emptyset, \quad n \in \mathcal{I}_m.
\end{aligned} \tag{7}$$

The latter conditions (7) ensure that a dataset of N points leads to a tree whose depth is

$O(\log N)$. This tree depth is reflected in our runtime analysis (see Subsection 5.3).

Moreover, we require that the number of points in each set $\mathcal{X}_{l,n}$ is at least k, for some fixed constant k (to be defined in Section 5), and at most γk for some small constant $\gamma > 1$. We denote such a tree by $T_{\mathcal{X}} = T_{\mathcal{X}}(k)$.

Throughout the paper we use the term "support of a vector" (or support of a discrete function). This term stands for the set of data points outside of which the vector is zero. In the context of a hierarchy tree, we often refer to vectors that are defined on a node $\mathcal{X}_{l,n}$ of the tree. For the context of this paper, such vectors are understood as discrete functions on the entire dataset with $\mathcal{X}_{l,n}$ as their support.

3. Problem setup

Let $\mathcal{X} = \{x_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}^D$, be a set of N points. In machine learning applications it is typical to assume that x_i are sampled from some manifold $\mathcal{M} \subset \mathbb{R}^D$ such that $\dim(\mathcal{M}) \ll D$. In our construction we assume no such structure. Yet, in order to introduce some structure into the space $\mathcal{X}$, we pose two requirements. First, we require that the set $\mathcal{X}$ is associated with a kernel function $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}_+$, and thus with the graph structure induced by K (see Subsection 2.1). This would allow us to define the graph Laplacian over subsets of $\mathcal{X}$. The second requirement is that $\mathcal{X}$ has an associated tree structure (see Subsection 2.3).

Under these assumptions, we are looking for an orthonormal basis defined on the set $\mathcal{X}$, that is we are looking for N functions

$$\{\phi_n\}_{n=1}^{N}, \quad \phi_n : \mathcal{X} \to \mathbb{R},$$

such that

$$\langle \phi_n, \phi_m \rangle = \delta_{n,m},$$

where $\delta_{n,m}$ is the Kronecker delta. We use the standard Euclidean inner product

$$\langle f, g \rangle = \sum_{x \in \mathcal{X}} f(x) g(x), \quad \forall f, g : \mathcal{X} \to \mathbb{R}. \tag{8}$$

Obviously, a trivial solution to this problem is to set ϕ_i to be the indicator function on the point x_i. This basis is excellent for representing very localized functions, that is, functions that are nonzero on only small subsets of $\mathcal{X}$, but requires many coefficients for representing any other function.

We have already mentioned in Section 1 two other possible bases, namely the eigenvectors of the graph Laplacian and the Haar basis. These two bases represent two extremes. The eigenvectors of the graph Laplacian, under appropriate assumptions, can be shown to give rise to sparse representations of smooth functions. On the other hand, the Haar basis, whose most elements have compact support (they are nonzero on small subsets of the data points), is not well suited for representing smooth functions, but represents sparsely functions with local phenomena such as jumps or oscillations (as we show in Section 7). In addition, unlike the eigenvectors of the graph Laplacian, the Haar basis has a fast transform with complexity of $O(N \log N)$. These two solutions illustrate the well known trade-off from the classical theory of wavelets, between smooth representation and compact support, e.g., [18, Chapter 7]. We aim to design a basis where the trade-off between localization and smoothness of its elements can be controlled.

Thus, when designing a basis for the set $\mathcal{X}$, we require the following. First, the construction must be applicable in cases where D (the dimension of each point in $\mathcal{X}$) is very large. Second, it should allow for a sparse representation of a large family of functions. And third, it must have a fast and numerically stable algorithm.

We propose a family of bases parameterized by a single parameter $1 \leq k \leq N$, which controls the localization of the elements of the basis, that is, how many elements of the basis have compact support. In our family of bases, the Haar basis and the basis of the graph Laplacian's eigenvectors are the two extreme members, corresponding to the values $k = 1$ and $k = N$, respectively. In other words, our construction defines a transition between the Haar and the Laplacian bases.

4. Informal description of the construction

In this section we give an informal description of our construction to explain the intuition behind it. The full details will be given in Section 5.

Let us consider a toy example, with a dataset $\mathcal{X}$ consisting of $N = 17$ equally spaced points over the interval $[0, 1]$, that is $x_j = (j-1)/N$, $j = 1, \ldots, N$. Points sampled from the real line are well-ordered and we can use, for example, the dyadic partition over our interval to define a tree partition. We start by choosing the parameter $k = 4$. This choice determines that each node of the tree partition consists of at least 4 data points. Thus, we get the tree partition as appears in Figure 1.

Our construction consists of two phases: defining the sequence of approximation spaces (2) and orthogonalizing them to satisfy (4).

The first phase begins by defining V_0 as the span of the first $k = 4$ eigenvectors of the graph Laplacian of the data points $x_1, \ldots, x_{17}$ (see Algorithm 1). Recall that the first eigenvector is always the constant vector, which in our case is $1/\sqrt{17}(1, \ldots, 1)^T$, and therefore is always part of V_0, regardless of the value of k. We denote the eigenvectors that comprise V_0 by $v_1, \ldots, v_4$ and present them in Figure 2. Clearly, $\dim(V_0) = 4$. The MRA structure (4)

$$\begin{tikzcd}
 & & & \mathcal{X}_{0,0} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}\} \arrow[dll] \arrow[drr] & & & \\
 & \mathcal{X}_{1,0} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\} \arrow[dl] \arrow[dr] & & & & \mathcal{X}_{1,1} = \{x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}\} \arrow[dl] \arrow[dr] & \\
\mathcal{X}_{2,0} = \{x_1, x_2, x_3, x_4, x_5\} & & \mathcal{X}_{2,1} = \{x_6, x_7, x_8, x_9\} & & \mathcal{X}_{2,2} = \{x_{14}, x_{15}, x_{16}, x_{17}\} & & \mathcal{X}_{2,3} = \{x_{10}, x_{11}, x_{12}, x_{13}\}
\end{tikzcd}$$

Figure 1: Tree partition for our $N = 17$ example, based on the dyadic partition.

suggests that the basis of V_0 is part of the orthogonal basis. However, these vectors are not orthogonal and thus we will orthogonalize them later.

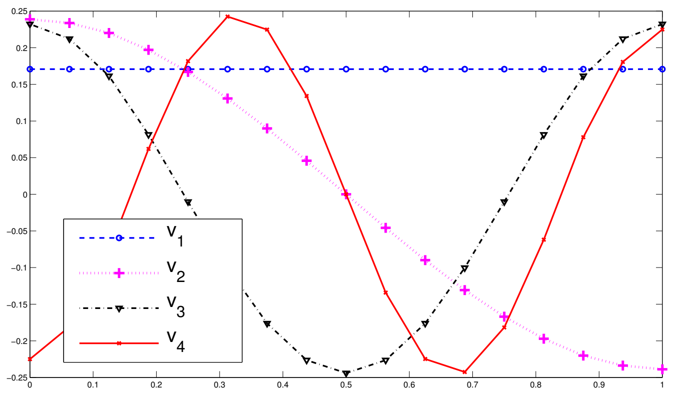
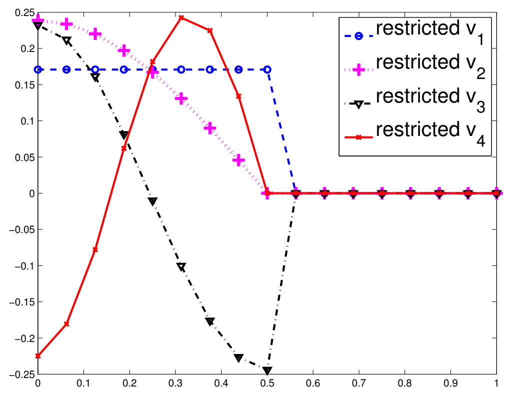
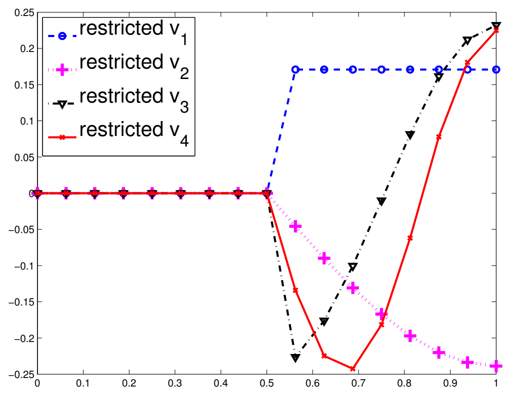


Figure 2: The basis of V_0.

To define the next approximation space, V_1, we use a restriction operator. This operator takes as an input a vector and a set, and zeros the coordinates of the vector outside the set. At this point we use the partition of the data points given by the first level of the hierarchy tree, that is, the sets $\mathcal{X}_{1,0}$ and $\mathcal{X}_{1,1}$, and restrict each of the vectors of V_0 to these two sets. Restricting V_0 to $\mathcal{X}_{1,0}$ results in 4 vectors, which are depicted in Figure 3(a), and restricting it to $\mathcal{X}_{1,1}$ results in another 4 vectors, shown in Figure 3(b). The two sets of 4 vectors are mutually orthogonal, as the sets $\mathcal{X}_{1,0}$ and $\mathcal{X}_{1,1}$ are disjoint. This gives a total of 8 vectors spanning V_1. It is clear that the space V_1 contains the space V_0. In our example, one can easily verify numerically that the restricted vectors, on each set, are linearly independent, and consequently the space V_1 satisfies $\dim(V_1) = 8$.



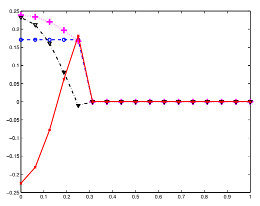
(a) V_0 after restriction to $\mathcal{X}_{1,0}$

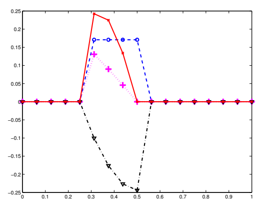


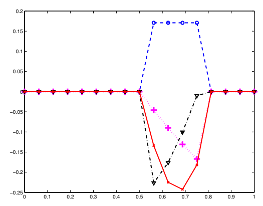
(b) V_0 after restriction to $\mathcal{X}_{1,1}$

Figure 3: The basis of V_1.

To construct the next approximation space V_2, we observe that each of the subsets $\mathcal{X}_{1,0}$ and $\mathcal{X}_{1,1}$ has two children nodes in the given tree partition (see Figure 1); the children of $\mathcal{X}_{1,0}$ are $\mathcal{X}_{2,0}$ and $\mathcal{X}_{2,1}$ and the children of $\mathcal{X}_{1,1}$ are $\mathcal{X}_{2,2}$ and $\mathcal{X}_{2,3}$. Therefore, we can repeat the procedure used to construct V_1 for V_2 by restricting the vectors of V_1 to the sets $\mathcal{X}_{2,0}, \ldots, \mathcal{X}_{2,3}$. Restricting a given vector on a set and then again on its subset is equivalent to restricting the original vector to the subset. Therefore, to avoid introducing excess notation in this example, we consider the second procedure of restriction as restricting the vectors of V_0 to the sets $\mathcal{X}_{2,0}, \ldots, \mathcal{X}_{2,3}$. The restriction of the 4 vectors of V_0 to the 4 sets $\mathcal{X}_{2,0}, \ldots, \mathcal{X}_{2,3}$ results in 16 vectors whose union is defined as V_2. Clearly, the relation $V_0 \subset V_1 \subset V_2$ holds. The vectors of V_2 are given in Figure 4.


(a) Restriction to $\mathcal{X}_{2,0}$


(b) Restriction to $\mathcal{X}_{2,1}$


(c) Restriction to $\mathcal{X}_{2,2}$

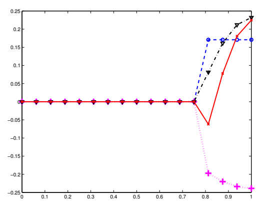

(d) Restriction to $\mathcal{X}_{2,3}$

Figure 4: The vectors of V_2 that are generated by restriction to $\mathcal{X}_{2,0}, \ldots, \mathcal{X}_{2,3}$.

Constructing the nested approximation spaces (2) solely by restriction leads to two problems. First, the last approximation space, defined on the leafs of the tree, namely V_2 in our case, may not have enough vectors to define a basis. In our example, the restriction results in only 16 vectors in V_2, defined on $\mathcal{X}_{2,0}, \ldots, \mathcal{X}_{2,3}$, while the last approximation space must form a basis, due to (2) and as described in Subsection 2.2. In other words, it must have $N = 17$ independent vectors. Second, repeating the restriction may lead to linearly dependent vectors or nearly dependent vectors (for more details on numerical dependency see e.g., [25, Chapter 5]).

To overcome these problems, generally caused by an insufficient number of independent vectors in the approximation spaces, we construct local graph Laplacians on each of the sets defined by the tree and compute their first eigenvectors. These eigenvectors are added to the sets of restricted vectors to assure two conditions. The first is that for any set in the hierarchy tree, which is not a leaf, there are exactly $k = 4$ independent vectors that have this set as their support. In our example, the sets $\mathcal{X}_{0,0}, \mathcal{X}_{1,0}$ and $\mathcal{X}_{1,1}$ satisfy this condition using only their restricted vectors, and thus, no local graph Laplacian is needed. The second condition is that the last approximation space forms a basis, or equivalently, the number of independent vectors in each leaf is equal to the number of data points in the leaf. In our example, the leaf nodes $\mathcal{X}_{2,1}, \mathcal{X}_{2,2}$ and $\mathcal{X}_{2,3}$ satisfy this condition as the vectors restricted to each of these sets are linearly independent. However, $\mathcal{X}_{2,0}$ does not satisfy this second condition. For $\mathcal{X}_{2,0}$ an additional independent vector is required since restriction results in 4 vectors while $\mathcal{X}_{2,0}$ has 5 data points and so requires 5 independent vectors. Therefore, we construct the graph Laplacian on $\mathcal{X}_{2,0}$ and add its first eigenvector that is independent with respect to the restricted vectors, that is, the second eigenvector. The restricted vectors and the additional local eigenvector are presented in Figure 5. These vectors ensure that $\dim(V_2) = N = 17$ as required.

The fact that we add to $\mathcal{X}_{2,0}$ the second eigenvector and not the first is not a coincidence, and it is because the first eigenvector (over any set of points) is always the constant vector. As a result, on one hand, the constant vector is an element of V_0, regardless the value of k. On the other hand, for any subset $\mathcal{X}_{j,n}$ of the hierarchy tree, the restriction of the constant vector (which is the characteristic function of $\mathcal{X}_{j,n}$) is always an element of V_j. The meaning of this observation is that adding the first eigenvector of the local graph Laplacian, defined on $\mathcal{X}_{j,n}$, to V_j does not increase $\dim(V_j)$. That is to say, the relevant set of eigenvectors to add to V_j always starts with the second eigenvector of the local graph Laplacian.

Once the approximation spaces that satisfy (2) have been constructed, we need to obtain the complement spaces W_j from (3). This is the second phase of our algorithm. In our example, we have to calculate the complement spaces W_0 and W_1 that satisfy $V_0 \oplus W_0 = V_1$, where $V_0 \perp W_0$, and $V_1 \oplus W_1 = V_2$, where $V_1 \perp W_1$.

The structure of the MRA, given in (4), dramatically reduces the complexity of calculating the complement spaces W_j due to two important principles. First, as described in Subsection 2.2, there is no need to directly orthogonalize two different complement spaces W_{j_1} and W_{j_2}

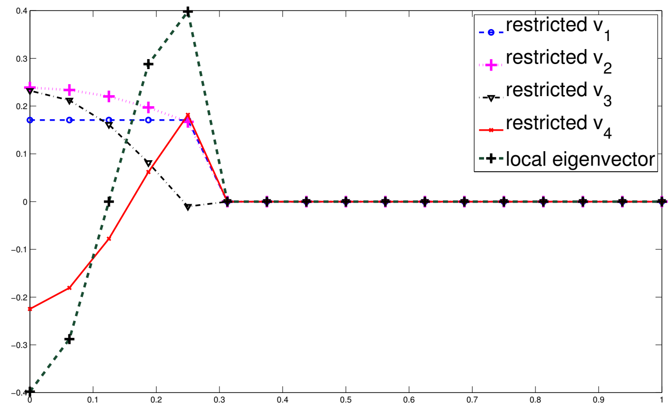


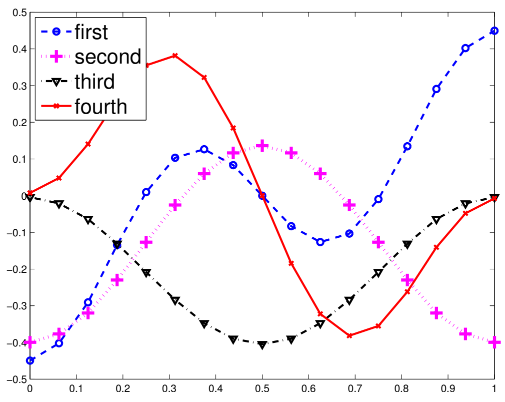
Figure 5: The vectors of V_2 that are nonzero on $\mathcal{X}_{2,0} = \{x_1, \ldots, x_5\}$. These vectors are linearly independent and derived from the restriction of the four vectors of V_0, and supplemented by the (second) eigenvector of the local graph Laplacian, defined on $\mathcal{X}_{2,0}$.

with $j_1 \neq j_2$. For instance, in our example, we calculate W_0 using V_0 and V_1 without having to consider W_1. In other words, it is guaranteed that W_0 will automatically be orthogonal to W_1. The second principle is local orthogonalization. Namely, due to the disjoint sets, defined by each level of the hierarchy tree partition, we only need to orthogonalize vectors that are defined on the same set, as the other ones are automatically orthogonal.

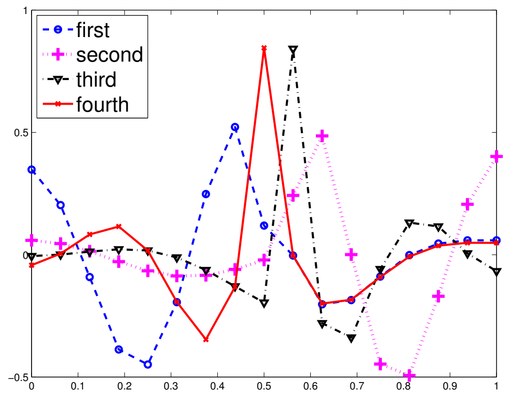
An orthogonalization process applied to sparse vectors typically does not preserve their sparsity. Consequently, the supports of the vectors (the subsets of the data where outside of these subsets the vectors are zero) of the complement spaces are determined as following. The vectors of W_1 have the same support as vectors of V_1, that is $\mathcal{X}_{1,0}$ or $\mathcal{X}_{1,1}$. Moreover, vectors from W_0 are always supported on the entire dataset $\mathcal{X}_{0,0}$, meaning they are global. These observations are true in the general case as well, that is, vectors of W_0 are always globally supported and the vectors of W_j have the same support as the vectors of V_j.

In our example, the basis we get is given by $V_0 \oplus W_0 \oplus W_1$. It consist of 17 vectors – 4 vectors of V_0 (after orthogonalization), shown in Figure 6(a), 4 vectors of W_0, shown in Figure 6(b), and 9 vectors of W_1, shown in Figure 7. The support of the vectors in W_1, which is roughly half of the data, is clearly noticed.

In general, the resulting basis for a given (small enough) k consists of N orthogonal vectors, where $N - k$ of them are orthogonal to the first k eigenvectors of the global graph Laplacian and $N - 2k$ of them have support that is smaller than the entire data, which depends on the hierarchy tree.

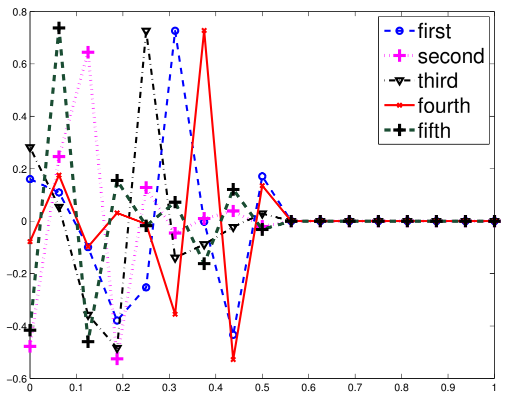


(a) V_0 after orthogonalization

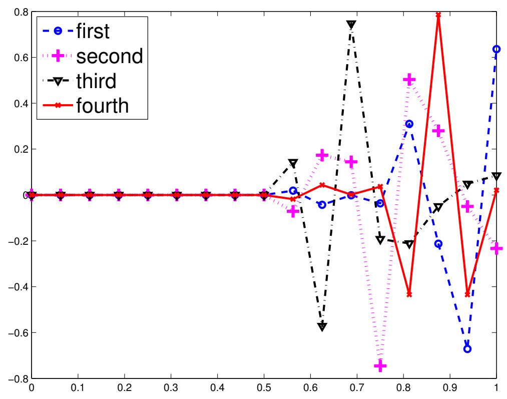


(b) W_0

Figure 6: The vectors of V_0 and W_0



(a) Vectors of W_1 having support $\mathcal{X}_{1,0}$



(b) Vectors of W_1 having support $\mathcal{X}_{1,1}$

Figure 7: The vectors of W_1. The supports correspond to $\mathcal{X}_{1,0}$ and $\mathcal{X}_{1,1}$ of the hierarchy tree (see Figure 1).

5. Detailed description of the construction

The input to the construction of the Laplacian multiwavelets (LMW) consists of N distinct points $\mathcal{X} = \{x_i\}_{i=1}^{N}$, a positive integer $1 \leq k \leq N$, their associated hierarchy tree $T_{\mathcal{X}}(k)$, and an affinity kernel or metric. We will construct an orthonormal basis for the N-dimensional space of functions defined on $\mathcal{X}$. Inspired by [2, 15], we require the basis to satisfy two fundamental properties:

1. All but k basis vectors have k "generalized vanishing moments", that is, they are orthogonal to the first k eigenvectors of the graph Laplacian on $\mathcal{X}$. The notion of generalized vanishing moments will be defined and explained in Subsection 6.1.
2. All but $\mathcal{O}(k)$ basis vectors have small support (that is, they are nonzero on a small set

of data points). The exact number of such vectors and their support size are determined by $T_{\mathcal{X}}(k)$, as will be explained in Subsections 5.1–5.2.

As described in Section 4, the construction consists of two phases, described in the following subsections.

5.1. Phase one - nested approximation spaces

We follow the framework of MRA (see Subsection 2.2). Accordingly, in the first phase we construct the nested approximation spaces (2).

The first step is to define the first approximation space V_0. This space is important since it determines the generalized vanishing moments property; the MRA conditions (3) and (4) imply that every other vector in the basis is orthogonal to the elements of V_0.

Henceforth, in this section, with a slight abuse of notation, we denote by V_j a matrix whose columns span the approximation space V_j (see also Subsection 2.2). Now, based on Algorithm 1, that extracts the first eigenvectors of the graph Laplacian (GLE), we define $V_0 = \mathrm{GLE}(\mathcal{X}, k)$, namely, V_0 is an $N \times k$ matrix, whose columns are the eigenvectors computed by Algorithm 1. Those eigenvectors are assumed to be ordered in increasing order of the eigenvalues of the graph Laplacian, starting with the smallest zero eigenvalue. Therefore, the first column of V_0 is always the constant vector $\frac{1}{\sqrt{N}}(1, \ldots, 1)^T \in \mathbb{R}^N$. Note that the columns of V_0 are not orthogonal with respect to (8), thereby we will later orthogonalize them.

Next, we recursively define the approximation space V_j, $j \geq 1$, based on V_{j-1} and the hierarchy tree $T_{\mathcal{X}}(k)$ (see subsection 2.3). The space V_{j-1} is defined to be the matrix whose columns are the union of the columns of $V_{j-1,n}$, $n \in \mathcal{I}_{j-1}$. The columns of each matrix $V_{j-1,n}$ are vectors of length N that are zero outside $\mathcal{X}_{j-1,n}$. This structure of V_{j-1} and V_j enables us to reduce the problem of defining V_j to the following one: given a node in the tree (or equivalently a set of data points) $\mathcal{X}_{j-1,n^*}$, how to define $V_{j,n}$ for j, n such that $\mathcal{X}_{j,n} \in \mathrm{child}(\mathcal{X}_{j-1,n^*})$, based on the vectors of V_{j-1,n^*}.

The recurrence considered in the previous paragraph starts with the space V_0, whose columns are globally supported, to wit, they have the entire dataset $\mathcal{X}_{0,0}$ as their support. The set $\mathcal{X}_{0,0}$ is considered as the 0-level of the hierarchy tree.

Before proceeding to the general case, consider the special case where $\mathcal{X}_{j-1,n^*}$ is a leaf (has no children in the hierarchy tree). In such a case, all the vectors of V_{j-1,n^*} are automatically inherited to V_j. In addition, for consistency with our recursive definition, we regard $\mathcal{X}_{j-1,n^*}$ as part of any j^*-level of the tree for $j^* \geq j$.

For the general case, where $\mathcal{X}_{j-1,n^*}$ is not a leaf, the basic requirement on the dimension of each $V_{j,n}$, where $\mathcal{X}_{j,n} \in \text{child}(\mathcal{X}_{j-1,n^*})$, is

$$dim(V_{j,n}) = \begin{cases} |\mathcal{X}_{j,n}| & \text{if } \mathcal{X}_{j,n} \text{ is a leaf ,} \\ k & \text{otherwise.} \end{cases} \tag{9}$$

Now, we define $V_{j,n}$ using a restriction operator, that restricts functions to the subset $\mathcal{X}_{j,n}$. For any $u \in \mathbb{R}^N$ and $\mathcal{X}_{l,n} \in T_{\mathcal{X}}(k)$, we define the restriction of u to $\mathcal{X}_{l,n}$ as $R_{\mathcal{X}_{l,n}} u$ where $R_{\mathcal{X}_{l,n}}$ is an $N \times N$ diagonal matrix given by

$$\left(R_{\mathcal{X}_{l,n}}\right)_{p,q} = \begin{cases} 1 & \text{if } p = q \text{ and } x_p \in \mathcal{X}_{l,n}, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

For example, to define V_1, we apply to all the vectors of V_0 the operators (10), with $l = 1$ and $n \in \mathcal{I}_1$. In other words, we restrict each of the first k eigenvectors of the graph Laplacian on $\mathcal{X}_{0,0}$ to the sets $\mathcal{X}_{1,n}$, $n \in \mathcal{I}_1$, of the hierarchy tree $T_{\mathcal{X}}(k)$. The result of applying $R_{\mathcal{X}_{1,n}}$ on V_0 is a sparse matrix $V_{1,n}$ of size $N \times k$ with at most $|\mathcal{X}_{1,n}| \times k$ nonzeros.

There are two main advantages to using restriction. First, it guarantees that $V_{j-1} \subset V_j$, as required in (2). Second, it generates sparse matrices, with vectors that have known support. Nevertheless, the restriction by itself is not sufficient to construct V_j since the requirement (9) may be violated. The reason for this is two-folded; first, restriction of linearly independent vectors may result in vectors are dependent. Second, if $\mathcal{X}_{j,n} \in \text{child}(\mathcal{X}_{j-1,n^*})$ is a leaf, we may need to have $\dim(V_{j,n}) > \dim(V_{j-1,n^*})$, due to (9).

In cases where the requirement (9) is violated, we construct a local graph Laplacian on $\mathcal{X}_{j,n}$. The first eigenvectors of the local graph Laplacian are extracted and added to $V_{j,n}$. The number of those eigenvectors added to $V_{j,n}$ is such that (9) is satisfied. Obviously, we use only eigenvectors of the local graph Laplacian that are linearly independent of the vectors already in V_j. Note that adding vectors to V_j preserves the relation $V_{j-1} \subset V_j$. For example, the characteristic function of $\mathcal{X}_{j,n}$ is always an element of $V_{j,n}$, generated from repeatedly restricting the constant function of V_0. Therefore, the first eigenvector of the local graph Laplacian is never added to V_j. See also the example of Section 4.

We remark that the eigenvectors of the local graph Laplacian are defined only on $\mathcal{X}_{j,n}$, and thus, when considering them as vectors of $V_{j,n}$, we define any component outside $\mathcal{X}_{j,n}$ to be zero. By doing so, the sparsity of the matrix $V_{j,n}$ is preserved. In addition, observe that since the sets $\mathcal{X}_{j,n}$ of the j-level are disjoint, any two vectors of two different matrices $V_{j_1,n}$ and $V_{j_2,n}$, with $j_1 \neq j_2$ are orthogonal. Thus, we have $\dim(V_j) = \sum_n \dim(V_{j,n})$.

To summarize, the resulting spaces from this phase clearly satisfy (2), and for the deepest (maximal) level m in the tree $T_{\mathcal{X}}(k)$ we have

$$\dim(V_m) = \sum_{n\in\mathcal{I}_m} \dim(V_{m,n}) = \sum_{n\in\mathcal{L}_T} |\mathcal{X}_{m,n}| = N, \tag{11}$$

where $\mathcal{L}_T$ denotes the indices of all leafs in $T_{\mathcal{X}}(k)$. Furthermore, by construction, $V_{j,n}$ has at most $|\mathcal{X}_{j,n}| \times k$ nonzero elements for every $\mathcal{X}_{j,n}$ which is not a leaf, and at most $|\mathcal{X}_{j,n}| \times \gamma k$ nonzeros when $\mathcal{X}_{j,n}$ is a leaf, with a small constant $\gamma > 1$ (see Subsection 2.3). Thus, we have $\mathcal{O}(Nk)$ nonzeros elements in each V_j. Note that for the case of a balanced hierarchy tree we have $m = \mathcal{O}\big(\log(N/k)\big)$. Thus, in such a case we can pack the resulting vectors of this phase in a sparse $N \times N$ matrix containing $\mathcal{O}\big(N \log N\big)$ nonzeros.

Algorithm 2 provides the pseudo-code for constructing the approximation spaces V_j. We use Matlab notation for matrix concatenation and indexing. For example $[A, B]$ is the concatenation of A and B, and $A(:, 1)$ stands for the first column in A. To orthogonalize the columns of a given matrix, and to check for numerical linear dependencies of vectors, we use an SVD procedure [25, Chapter 2]. In addition, to make all vectors of appropriate length, we use two variants of the $R_{\mathcal{X}_{l,n}}$ matrix of (10). The first, $R^e_{\mathcal{X}_{l,n}}$ is an $N \times |\mathcal{X}_{l,n}|$ matrix derived from $R_{\mathcal{X}_{l,n}}$ by using its $|\mathcal{X}_{l,n}|$ columns corresponding to the set of indices of the data points x_i such that $x_i \in \mathcal{X}_{l,n}$. This matrix is used to zero pad vectors defined on $\mathcal{X}_{l,n}$ to be of length N. The second, $R^s_{\mathcal{X}_{l,n}}$ is a $|\mathcal{X}_{l,n}| \times N$ matrix derived from $R_{\mathcal{X}_{l,n}}$ by using its $|\mathcal{X}_{l,n}|$ rows corresponding to the set of indices of the data points x_i such that $x_i \in \mathcal{X}_{l,n}$. This matrix is used to truncate vectors to length $|\mathcal{X}_{l,n}|$.

5.2. Phase two - fast orthogonalization

In this phase we construct the complement spaces W_j that satisfy (4) using fast orthogonaliztion. This fast orthogonaliztion follows from the MRA structure using two observations. The first is that the computation of any W_j requires only V_j and the next approximation space V_{j+1}, and it is independent on any other space. In other words, the MRA structure ensures that W_j is automatically orthogonal to W_{j^*}, $j^* \neq j$. This is true since the nesting property (2) implies that if $V_{j^*+1} \subseteq V_j$ and $W_j \perp V_j$ then $V_{j^*+1} \perp W_j$, which together with $W_{j^*} \subseteq V_{j^*+1}$ results in $W_j \perp W_{j^*}$ for all $0 \leq j^* < j$. The second observation is that the calculation of any W_j can be done locally, since any two subsets at the same level of the hierarchy tree are disjoint. Specifically, the columns of W_j span the complement space of the columns of V_j with respect to the span of the columns of V_{j+1}. However, we actually calculate separately the complement space for each V_{j,n^*} with respect to all $V_{j+1,n}$ such that $\mathcal{X}_{j+1,n} \in \text{child}(\mathcal{X}_{j,n^*})$. By doing so, we not only decrease the overall complexity (next to follow), but also guarantee that any column of W_j has at the most $\mathcal{X}_{j,n^*}$ nonzeros, just like

Algorithm 2 Approximation Spaces

```
Input: A set $\mathcal{X} = \{x_i\}_{i=1}^N$ of data points, an integer $1 \le k \le N$, and its corresponding
    hierarchy tree $T_{\mathcal{X}}(k)$.
Output: $\{V_{l,n}\}_{l=0,n\in\mathcal{I}_l}^m$.
 1: $V_0 \leftarrow \mathrm{GLE}(\mathcal{X},\ k)$
 2: $V_{0,0}\Sigma Y \leftarrow \mathrm{svd}(V_0)$
 3: $m \leftarrow \mathrm{depth}(T_{\mathcal{X}})$
 4: for $l = 1$ to $m$ do
 5:   for $n \in \mathcal{I}_l$ do {$\mathcal{I}_l$ is the set of indices of the $l$-th level of the tree.}
 6:     $V_{l,n} \leftarrow R_{\mathcal{X}_{l,n}} V_{l-1}$ {Restriction phase, see (10).}
 7:     if $\mathrm{isleaf}(\mathcal{X}_{l,n})$ then
 8:       $\tilde{k} \leftarrow |\mathcal{X}_{l,n}|$
 9:     else
10:       $\tilde{k} \leftarrow k$
11:     end if
12:     $\tilde{U}_{l,n} \leftarrow \mathrm{GLE}(\mathcal{X}_{l,n}, \tilde{k})$ {Local Laplacian eigenvectors.}
13:     $X\Sigma Y \leftarrow \mathrm{svd}(\left[R^s_{\mathcal{X}_{l,n}} V_{l,n}, \tilde{U}_{l,n}\right])$
14:     $V_{l,n} \leftarrow R^e_{\mathcal{X}_{l,n}} X(:, 1:\tilde{k})$
15:   end for
16: end for
```

any column of V_{j,n^*}. In other words, we preserve the sparsity of the matrix V_j in the matrix W_j.

The orthogonalization process of the second phase is done iteratively using the nested approximation spaces of the first phase. The resulting basis is extracted from the orthogonalized vectors of V_0 together with the orthogonal bases of W_j, $j = 0, \ldots, m-1$, where m is the index of the last approximation space V_m (see (11)). As mentioned above, the sparsity of V_j is also preserved in W_j. Thus, the resulting basis can also be packed in a sparse $N \times N$ matrix containing $\mathcal{O}\big(N \log N\big)$ nonzeros.

The entire fast orthogonalization process is described in Algorithm 3.

5.3. Runtime analysis

One essential assumption for the following complexity analysis is that the given hierarchy tree is balanced, as defined in (7). This property implies that the number of iterations in the outer loops of Algorithms 2 and 3 (lines 4 and 2, respectively), or equivalently the tree depth is $m = \mathcal{O}(\log N)$. Note that C in (7) is the bound for the maximal degree of a node in the tree and that a full binary tree satisfies $c = C = 2$.

We analyze each of the phases of our construction separately. We start with the analysis of Algorithm 2 line by line. In line 1 we extract the first k eigenvectors of the graph Laplacian on the data. The graph Laplacian is an $N \times N$ matrix, yet we can set up its nonzero

Algorithm 3 Fast Orthogonalization

```
Input: ${V_{l,n}}_{l=0,n\in\mathcal{I}_l}^{m}$ ($V_{l,n}$ is the output of Algorithm 2), $T_{\mathcal{X}}(k)$.
Output: ${W_l}_{l=0}^{m-1}$
1: $m \leftarrow \text{depth}(T_{\mathcal{X}}(k))$
2: for $l = 0$ to $m-1$ do
3:   for $n \in \mathcal{I}_l$ do {$\mathcal{I}_l$ is the set of indices of the $l$-th level of $T_{\mathcal{X}}$.}
4:     if not(isleaf($\mathcal{X}_{l,n}$)) then
5:       $X\Sigma Y \leftarrow \text{svd}(R^s_{\mathcal{X}_{l,n}} V_{l,n})$
6:       for $\mathcal{X}_{l+1,\tilde{n}} \in \text{child}(\mathcal{X}_{l,n})$ do
7:         $\tilde{W}_{l,n} \leftarrow \left[\tilde{W}_{l,n}, \left(I - XX^T\right) R^s_{\mathcal{X}_{l,n}} V_{l+1,\tilde{n}}\right]$
8:       end for
9:       $\tilde{W}_{l,n}\Sigma Y \leftarrow \text{svd}(\tilde{W}_{l,n})$ {Inner orthogonalization of columns.}
10:      $W_{l,n} \leftarrow R^e_{\mathcal{X}_{l,n}} \tilde{W}_{l,n}$
11:    end if
12:  end for
13: end for
```

entries in $\mathcal{O}(N)$ operations, using only local neighbourhoods (see Subsection 2.1). Computing the eigenvectors of the Laplacian corresponding to the first k eigenvalues is related to the problem of sparse symmetric eigenvalues extraction [25, chapter 8]. This issue is still an ongoing research. Generally, the time complexity of extracting the first k eigenvectors depends on many factors, such as the separation of the eigenvalues (which is also related to the ε parameter of the graph Laplacian), the geometry of the data points, and the desired accuracy. A linear $\mathcal{O}(Nk)$ implementation of this procedure is suggested in [29] using multigrid approaches. Other algorithms are variations on the Lanczos iterations [25, chapter 9]. We denote by $T(N,k)$ the upper bound for the running time of extracting the bottom k eigenvectors of the graph Laplacian constructed from N data points. Clearly, the complexity of our algorithm depends on $T(N,k)$ and hence we include it in our overall bound.

The bottom k eigenvectors of the graph Laplacian comprise V_0. These vectors are independent but not orthogonal and thus need to be orthogonalized. Line 2 computes an orthonormal basis for V_0 by applying an SVD procedure on the $N \times k$ matrix V_0. This line requires $\mathcal{O}(Nk^2)$ operations.

Next we analyze the complexity of lines 6-14 in Algorithm 2 for fixed n and l, that is, for a single set $\mathcal{X}_{l,n}$. Line 6 (the restriction) requires a total of $\mathcal{O}(|\mathcal{X}_{l,n}|k)$ operations, since we apply a matrix with $|\mathcal{X}_{l,n}|$ nonzero entries to k columns of V_{l-1}. In line 12 we extract the eigenvectors corresponding to the first k eigenvalues of the local graph Laplacian, which requires $T(|\mathcal{X}_{l,n}|,k)$ operations. Lines 13 and 14 determine the vectors that span the columns of V_l by applying an SVD procedure on a $|\mathcal{X}_{l,n}| \times 2k$ matrix, which requires

$\mathcal{O}(|\mathcal{X}_{l,n}|k^2)$ operations. We note that a sophisticated implementation can improve this up to $\mathcal{O}(|\mathcal{X}_{l,n}|k \log k)$ using random projections based procedures [26].

Now, given (5) and (7), we have that for a given level l the total runtime of lines 4-15 is bounded by

$$\sum_{n \in \mathcal{I}_l} T(|\mathcal{X}_{l,n}|, k) + \mathcal{O}(k^2 |\mathcal{X}_{l,n}|) \leq \mathcal{O}(k^2 N + T(N,k)),$$

and we conclude that an overall bound for the complexity of Algorithm 2 is

$$\mathcal{O}(k^2 N + T(N,k)) \times m = \mathcal{O}(k^2 N \log N + T(N,k) \log N), \tag{12}$$

where usually $N \gg k$.

In the second phase, of fast orthogonalization, given in Algorithm 3, the complement spaces (4) are calculated. We analyze the complexity of lines 5-9 in Algorithm 3 for fixed n and l. In line 5 we use an SVD to compute an orthonormal basis for $V_{l,n}$, which requires $\mathcal{O}(|\mathcal{X}_{l,n}|k^2)$ operations. In line 7 we project the vectors from V_{l+1} with support in $\mathcal{X}_{l,n}$ onto $V_{l,n}^{\perp}$. The matrix X, calculated in line 5, is of size $|\mathcal{X}_{l,n}| \times k$ and the matrix $V_{l+1,\tilde{n}}$ is of size $|\mathcal{X}_{l+1,\tilde{n}}| \times k$, with $|\mathcal{X}_{l+1,\tilde{n}}| < |\mathcal{X}_{l,n}|$. We omit the analysis of the case of a leaf, which is essentially the same. Thus, the task of evaluating $X^T V_{l+1,\tilde{n}}$ is bounded by $\mathcal{O}(|\mathcal{X}_{l,n}|k^2)$ operations. The following (left) multiplication by X in line 7 involves the product of a $|\mathcal{X}_{l,n}| \times k$ matrix with a $k \times k$ matrix. Again, this requires $\mathcal{O}(|\mathcal{X}_{l,n}|k^2)$ operations. The time complexity for subtracting two matrices is linear in the number of their elements. This computation is repeated for at most C iterations (C is given in (7)). Line 9 has the same complexity as line 5.

To conclude, under the assumption of a balanced hierarchy tree, the total complexity of Algorithm 3 is $\mathcal{O}(k^2 N \log N)$ operations, which together with (12) concludes the runtime analysis.

To further examine the runtime of the fast orthogonalization procedure, we measure the actual time in seconds needed to orthogonalize the nested approximation spaces for datasets of various sizes. The data points for this experiment are distributed uniformly on the unit cube in $\mathbb{R}^3$ and then normalized back to the unit sphere. We use a full binary tree based on spectral clustering (more details in Subsection 7.1) and a fixed parameter of $k = 5$. Then, we compare the running time with our theoretical bound of $\mathcal{O}(k^2 N \log N)$ and present the results in Figure 8. Indeed, it can be seen that the actual runtime agrees with our analysis.

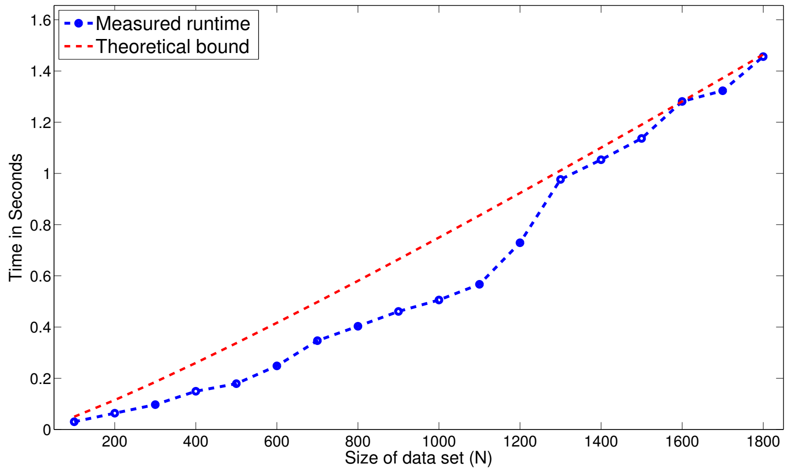


Figure 8: Runtime measures of Algorithm 3 versus its theoretical bound of $\mathcal{O}(k^2 N \log N)$ for different values of N and a fixed $k = 5$.

6. Approximation analysis

We analyze the properties of our bases in two steps. First we recall the notion of generalized vanishing moments, discuss it, and derive decay rates of the expansion coefficients with respect to elements of our basis that possess such a property. Then, we use the analogues of Besov spaces to define the smoothness of a function on a graph, with respect to our basis. That being so, we derive the connection between that definition of smoothness and the corresponding rate of approximation.

6.1. Generalized vanishing moments

The classical vanishing moments property of a function f over a domain Ω is defined as the largest $k \in \mathbb{N}$ such that

$$\int_\Omega t^p f(t) dt = 0, \quad p = 0, 1, \ldots, k-1.$$

Namely, f is orthogonal to all polynomials up to degree k. This definition is very natural for several reasons. For instance, approximations with polynomials have many well-known bounds (e.g., Taylor approximation) and the space of polynomials is shift and scale invariant. This invariance is useful, for example, when constructing wavelets, and indeed, most classical wavelets are defined using the scale and shift operators.

For the setting of discrete high-dimensional data, there is no natural interpretation for the scale and shift operators. Moreover, the approximation with high-dimensional polynomials is more complicated [20], and for data sampled from a smooth manifold, even the meaning of high degree polynomials is not trivial [11]. Thus, there is a need to try to extend the concept of vanishing moments to our setting.

Inspired by other papers such as [15], we propose to use the eigenfunctions of the Laplace-Beltrami operator, corresponding to the smallest eigenvalues, to define a generalization of the notion of vanishing moments. This approach has an advantage of being amenable to discretization using the graph Laplacian.

Let $\mathcal{X}$ be a finite set of points, and denote by u_p, $p = 0, 1, \dots, |\mathcal{X}| - 1$, the $p-$th eigenvector (increasing eigenvalues order) of the graph Laplacian defined on $\mathcal{X}$. We say that $f : \mathcal{X} \mapsto \mathbb{R}$ has k (discrete) generalized vanishing moments if

$$\sum_{x \in \mathcal{X}} f(x) u_p(x) = 0, \quad p = 0, 1, \dots, k-1. \tag{13}$$

We define the best approximation of a function $f : \mathcal{X} \mapsto \mathbb{R}$ using the first k eigenvectors of the graph Laplacian by

$$E_k(f) = \inf_{g \in \operatorname{span}\{u_0, u_1, \dots, u_{k-1}\}} \|f - g\|. \tag{14}$$

Equipped with (14) we have the following bound on the expansion coefficients in a basis having k generalized vanishing moments.

Theorem 6.1. *Let ϕ be an element in an orthonormal basis defined on $\mathcal{X}$. Furthermore, suppose that the basis has k generalized vanishing moments* (13). *Denote by $\mathcal{J} \subset \mathcal{X}$ the support of ϕ. Then, for f defined over $\mathcal{X}$ we have*

$$|\langle f, \phi \rangle| \leq E_k(f) \sqrt{|\mathcal{J}|},$$

where $|\mathcal{J}|$ is the number of elements in $\mathcal{J}$.

Proof. Denote $r_k(f) = f - \sum_{i=0}^{k-1} \langle f, u_i \rangle u_i = f - f_k$, that is $E_k(f) = \|r_k\|$. Then, since the basis has k generalized vanishing moments, we have $\langle f, \phi \rangle = \langle f_k + r_k, \phi \rangle = \langle r_k, \phi \rangle$. Therefore,

$$|\langle f, \phi \rangle| = \left| \sum_{x \in \mathcal{X}} r_k(x) \phi(x) \right| \leq \sum_{x \in \mathcal{J}} |r_k(x) \phi(x)| \leq \max_{x \in \mathcal{J}} |r_k(x)| \sum_{x \in \mathcal{J}} |\phi(x)|.$$

By Hölder's inequality we have

$$\sum_{x \in \mathcal{J}} |\phi(x)| \leq \sqrt{\sum_{x \in \mathcal{J}} \phi(x)^2} \sqrt{|\mathcal{J}|} = \|\phi\| \sqrt{|\mathcal{J}|}.$$

Since $\|\phi\| = 1$ and $\max_{x\in\mathcal{J}} |r_k(x)| \le \|r_k\|$ the proof follows. □

Theorem 6.1 shows that the decay rate depends on two factors: the size of the support of ϕ and the rate of best approximation. For the rate of best approximation we use the result in [1], where the relation

$$\|\nabla f\|^2 = \langle \Delta f, f\rangle \tag{15}$$

is used to derive

$$E_k(f) \le \frac{\|\nabla f\|}{\sqrt{\lambda_{k+1}}}. \tag{16}$$

The bound (16) is also valid for the discrete Laplacian on graphs. The gradient ∇f of a discrete function f can be defined as in [13, Chapter 13] such that (15) holds with respect to the graph Laplacian.

Taking into account (16) and Theorem 6.1, we get an explicit decay rate for the expansion coefficients of a function with a bounded gradient.

Corollary 6.1. *Let ϕ and f be as in Theorem 6.1, where f satisfies $\|\nabla f\| < M$, $M > 0$. Then,*

$$|\langle f,\phi\rangle| \le \frac{M}{\sqrt{\lambda_{k+1}}}\sqrt{|\mathcal{J}|}.$$

The bound of Corollary 6.1 is composed of three elements: M which is related to the regularity of f on the graph, the size of the support of ϕ, and λ_{k+1}^{-1} where λ_{k+1} grows (slowly) as k increases, e.g., [14, Chapter 1]. Nevertheless, λ_k is ultimately bounded, and so the decay of the coefficients depends solely on the support.

The supports of the elements of the LMW basis are defined using a hierarchy tree. Assuming that the tree is balanced (see (7)), we have that the support $|\mathcal{J}|$ decays exponentially with the levels of the tree. In this sense, this result is an analogue to the classical coefficients decay of wavelets.

6.2. Smoothness of functions on discrete datasets

We presented the construction of orthonormal bases for the representation of functions defined on discrete datasets. Next, we propose to use such a basis to measure the smoothness of a given function. Using the theory of non-linear analysis [21], we suggest a discrete definition of regularity and derive the relation between the function's regularity and its approximation rates.

For the clarity of this section, we focus on the L_2 norm. However, most of the results can be generalized to L_p, $1 < p < \infty$ using Minkowski's inequality and the fact that (due to

Jensen's inequality) for a function f defined on the domain I

$$\|f\|_p = |I|^{\frac{1}{q}-\frac{1}{p}}\|f\|_q, \quad \frac{1}{q}+\frac{1}{p}=1, \quad p,q\in[1,\infty].$$

Ideally, given a function to analyze, we hope to use only a small number of coefficients in regions where the function is “smooth”. On other regions, we would have to use more coefficients. The natural questions that arise are: how do we measure this smoothness for our bases and how good is the approximation using our bases with only a partial set of the coefficients?

Given an orthonormal basis $\Phi = \{\phi_j\}_{j=1}^N$, we recall the definition of its non-linear best approximation (also known as n-term approximation)

$$\sigma_n(f) = \min_{\phi_{j_1},\dots,\phi_{j_n}\in\Phi} \|f - \sum_{i=1}^{n} \langle f, \phi_{j_i}\rangle \phi_{j_i}\| \ . \tag{17}$$

We later use the non-linear approximation (17) in our numerical examples as well. The characterization of this approximation can be done by considering the classical smoothness spaces, namely Besov spaces [35].

Denote by

$$|f|_\tau = \left(\sum_j |\langle f, \phi_j\rangle|^\tau\right)^{\frac{1}{\tau}} \tag{18}$$

a measure of smoothness of f in terms of a parameter $\tau > 0$ and with respect to Φ. We term this the τ-measure of f. It turns out [21] that for the case of orthonormal bases, the classical Besov space can be reduced to the space

$$B_\tau = \{f \mid |f|_\tau < \infty\}, \quad 0<\tau<2.$$

In the classical smoothness analysis, we are especially interested in exploring how rich is the space B_τ as τ decreases. However, this formal definition looses most of its meaning in the discrete setting, since for N data points and a fixed τ we always have that

$$\|f\| \le |f|_\tau \le N^{\frac{1}{\tau}-\frac{1}{2}}\|f\|,$$

which implies that any function belongs to B_τ. Thus, to adapt the definition of B_τ to the discrete setting we define

$$B_{\tau,M} = \{f \mid |f|_\tau < M \text{ and } \|f\|=1\}, \quad 0<\tau<2, \quad 1\le M\le N^{\frac{1}{\tau}-\frac{1}{2}}. \tag{19}$$

Now, for a given function f, defined on our data points, and a fixed M, we look for the smallest τ such that $f \in B_{\tau,M}$. This value of τ determines the smoothness of f.

In the next theorem, we summarize the basic properties of the τ-measure, which we use later to gain a better understanding of the discrete smoothness defined above.

Theorem 6.2. *Let f be a real-valued function, defined over N data points, and assume $\|f\| = 1$. Given an orthonormal basis $\Phi = \{\phi_j\}_{j=1}^N$, the associated τ-measure* (18) *satisfies*

1. *Monotonicity,*
$$|f|_{\tau_1} \geq |f|_{\tau_2}, \quad \tau_1 \leq \tau_2.$$
2. $|f|_\tau \geq 1$, *where equality holds if and only if $f \in \Phi$ (that is, f is a basis function up to a sign).*
3. *Given an integer $s \leq N$, the maximal τ-measure for f of the form*
$$f = \sum_{j=1}^{s} \langle f, \phi_{I_j} \rangle \phi_{I_j}, \quad \phi_{I_1}, \ldots, \phi_{I_s} \in \Phi,$$
where $\|f\| = 1$ is obtained with $|\langle f, \phi_{I_j} \rangle| = \frac{1}{\sqrt{s}}$. In this case $|f|_\tau = s^{\frac{1}{\tau}-\frac{1}{2}}$.

The proof of Theorem 6.2 is given in Appendix A.

In the classical regularity of functions, one usually gets a hierarchy of smoothness classes. For example, a twice-differentiable function is also differentiable, and a differentiable function is continuous, so the class of twice-differentiable functions is contained in the class of differentiable functions which is contained in the class of continuous functions. The monotonicity property of Theorem 6.2 guarantees that the definition of $B_{\tau,M}$ also yields a hierarchy of smoothness classes. Namely, if $f \in B_{\tau_1,M}$ then $f \in B_{\tau_2,M}$ for $\tau_1 \leq \tau_2$.

The third claim of Theorem 6.2 indicates that as the number s of nonzero coefficients in the expansion of f in the basis Φ increases, the maximal value of the τ-measure may increase as well, since it is bounded by $s^{\frac{1}{\tau}-\frac{1}{2}}$, with $\frac{1}{\tau} - \frac{1}{2} > 0$, which grows with s. This assertion also suggests that as we take functions whose energy spreads over more and more expansion coefficients in the given basis, these functions may become less smooth according to our definition. In particular, for a fixed M in (19), there exists at least one function for which we may need to increase τ to satisfy $s^{\frac{1}{\tau}-\frac{1}{2}} < M$, namely, this function becomes less smooth as τ decreases.

To further illustrate the latter claim, we consider an LMW basis, defined over data points that are equally spaced on $[0,1]$, with parameters $N = 150$ and $k = 8$. Such a basis is discussed in detail in Subsection 7.1 (a similar basis is also given in the informal description of Section 4). Using this basis, we compare the τ-measure, $|f|_\tau$ of (18), for three functions having a different distribution of energy over their coefficients. This is presented in Figure 9. The example is composed of the following functions. f_1 is an element of the basis. Thus, by the second claim of Theorem 6.2 it has a τ-measure of 1, which is the smallest possible. The next two functions f_2 and f_3 are sampled from $\cos(x)$ and $\sin(10x)$, respectively. Recall that the first $k = 8$ eigenvectors of the graph Laplacian (and thus the first k vectors of the basis) are related to $\cos(2\pi nx)$, $n = 0, \ldots, k-1 = 7$, see e.g., [16], which results in a sparser representation of f_2 compared to that of f_3. The comparison between these two representations is given in Figure 10, where the (sorted) absolute values of the most significant coefficients are presented. Consequently, the τ-measure of f_3 is larger than that of f_2. This relation also demonstrates another interesting interpretation of the discrete smoothness; while $\cos(x)$ and $\sin(10x)$ have essentially the same smoothness in the classical sense, their samples may define discrete functions with different smoothness with respect to a given basis.

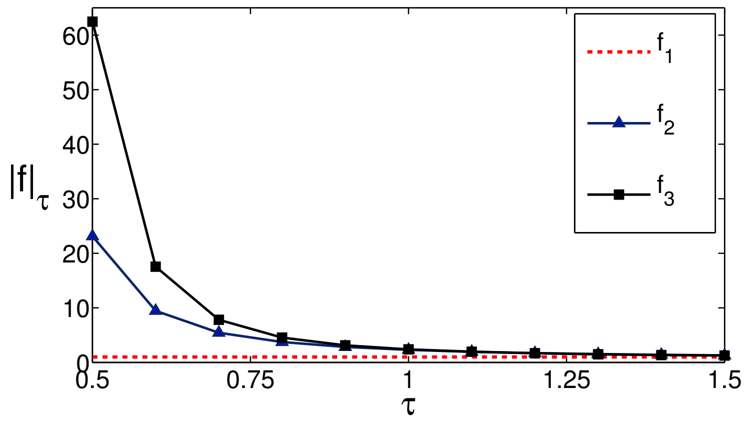


Figure 9: The τ-measure (18) of three different functions, with respect to an LMW basis with parameters $N = 150$ and $k = 8$, for data points equally spaced on $[0,1]$. The functions f_1, f_2 and f_3 are an element from the LMW basis, $\cos(x)$, and $\sin(10x)$, respectively.

Similar to the classical case, we can also obtain a Jackson type inequality for the non-linear approximation in terms of the discrete smoothness spaces $B_{\tau,M}$.

Theorem 6.3. *Let* $f \in B_{\tau,M}$, $0 < \tau < 2$. *Then,*

$$\sigma_n(f) \leq \frac{|f|_\tau}{n^\alpha} \leq \frac{M}{n^\alpha},$$

where $\alpha = \frac{1}{\tau} - \frac{1}{2}$.

Proof. Let the expansion coefficients of f be ordered in a non-increasing order, that is

$$|\langle f, \phi_{I_{j+1}} \rangle| \leq |\langle f, \phi_{I_j} \rangle|, \quad 1 \leq j \leq N-1.$$

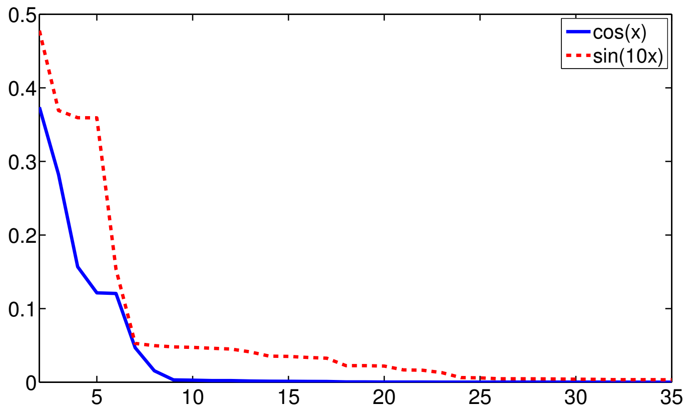


Figure 10: The absolute values of the most significant expansion coefficients of $\cos(x)$ and $\sin(10x)$, with respect to an LMW basis with parameters $N = 150$ and $k = 8$, for data points equally spaced on $[0, 1]$.

Then,

$$\begin{aligned}
\sigma_n(f) &= \left(\sum_{j=n+1}^{N} \left\langle f, \phi_{I_j} \right\rangle^2 \right)^{\frac{1}{2}} \\
&\leq \left(|\left\langle f, \phi_{I_n} \right\rangle|^{2-\tau} \sum_{j=n+1}^{N} |\left\langle f, \phi_{I_j} \right\rangle|^{\tau} \right)^{\frac{1}{2}} \\
&= |\left\langle f, \phi_{I_n} \right\rangle|^{1-\frac{\tau}{2}} \left(\sum_{j=n+1}^{N} |\left\langle f, \phi_{I_j} \right\rangle|^{\tau} \right)^{\frac{1}{2}} \\
&= n^{-\alpha} \left(n |\left\langle f, \phi_{I_n} \right\rangle|^{\tau} \right)^{\frac{1}{\tau}-\frac{1}{2}} \left(\sum_{j=n+1}^{N} |\left\langle f, \phi_{I_j} \right\rangle|^{\tau} \right)^{\frac{1}{2}}.
\end{aligned}$$

From [34], for any $0 < \alpha \leq s$ and $a, b > 0$ we have $a^{\alpha} b^{s-\alpha} \leq (a + b)^s$. Thus, using the monotonicity of the expansion coefficients,

$$\sigma_n(f) \leq n^{-\alpha} \left(n |\left\langle f, \phi_{I_n} \right\rangle|^{\tau} + \sum_{j=n+1}^{N} |\left\langle f, \phi_{I_j} \right\rangle|^{\tau} \right)^{\frac{1}{\tau}} \leq n^{-\alpha} |f|_{\tau}.$$

□

Theorem 6.3 directly links the regularity of a function, given in terms of the basis, to the best n-term approximation. The opposite direction of Theorem 6.3 is a Bernstein type estimation, given by

$$|f|_{\tau} \leq n^{\alpha} \|f\|.$$

The proof is merely an application of Hölder's inequality, since $\|fg\|_r \leq \|f\|_p \|g\|_q$ where $\frac{1}{p} + \frac{1}{q} = \frac{1}{r}$, $0 < r < 2$, $0 < p, q$ and $r < p, q$. This completes our discrete analogue for the classical characterization of the non-linear n-term approximation.

One aspect of Theorem 6.3 is that knowing a bound on the coefficients of a given function, immediately provides a bound on its best n-term approximation. For example, assume $|\langle f, \phi_j \rangle| \leq \varepsilon$, for all $1 \leq j \leq N$. Then

$$|f|_\tau^\tau = \sum_{j=1}^{N} |\langle f, \phi_j \rangle|^\tau \leq \varepsilon^\tau N.$$

In other words, $f \in B_{\tau, \varepsilon N^{\frac{1}{\tau}}}$ and thus Theorem 6.3 guarantees that

$$\sigma_n(f) \leq \frac{\varepsilon N^{\frac{1}{\tau}}}{n^\alpha}.$$

7. Numerical examples

In this section we demonstrate numerically our construction and its properties using various examples. For these examples we use the kernel $K(x, y) = e^{-x-y^2/2\varepsilon}$, where ε is chosen by the criterion described in [16].

7.1. Approximating functions

Our class of bases includes two important cases with extreme values of k. The first is the Haar basis where $k = 1$. In this case, all nested approximation spaces (2) consist only of the constant vector. The second case is the basis that corresponds to $k = N$, where the nested approximation spaces (2) include only V_0. In this case V_0 comprises of all the eigenvectors of the graph Laplacian on the data. We call the latter the graph Laplacian basis.

In this subsection we explore the representation of functions using four different bases from our family of bases - the Haar basis, the graph Laplacian basis, and two intermediate bases where $1 < k < N$. The Haar basis represents well functions with highly localized changes (e.g., the delta function). However, it typically doesn't efficiently represent other signals having more global behaviour. On the other hand, the graph Laplacian basis, which is a generalization of the Fourier basis for general domains, is known to represent poorly functions with local oscillations or "jumps".

We study the different representations using three synthetic datasets. The first comprises of equally spaced points on $[0, 1]$, the second consists of points randomly distributed over the unit sphere $S^2 \subset \mathbb{R}^3$, and the third consists of points in $\mathbb{R}^{1000}$, which lie on an intrinsically low-dimensional manifold.

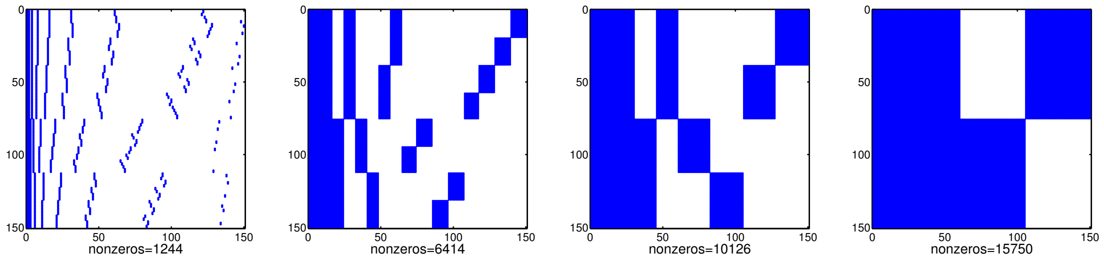
Case 1 - equally spaced points on $[0,1]$

For the first set of examples, we use 150 equally spaced points on $[0,1]$, that is the set

$$\mathcal{X} = \{x_j\}_{j=1}^{N}, \quad x_j = (j-1)/N, \quad N = 150. \tag{20}$$

We choose three functions with different properties: a smooth function, a function with large local oscillations, and a smooth function with a simple jump discontinuity. In each case we present the function and the relative error (L_2 norm) associated with the best non-linear approximation (17) (where we use the largest coefficients of the representation) versus the number of coefficients in use.

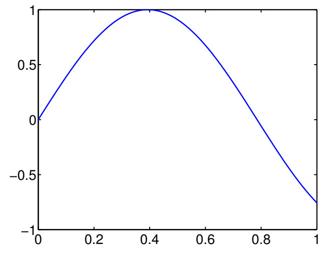
Our intermediate cases in this example are $k = 8$ and $k = 15$, corresponding to 5% and 10% of the dataset size N, respectively. As the tree partition we simply use the dyadic partition over $[0,1]$. The result is a full binary tree $T_{\mathcal{X}}(k)$. Recall that the value of the parameter k determines the depth of the tree $T_{\mathcal{X}}(k)$ (see Subsection 2.3). Thus, higher values of k lead to bases with many non-localized elements, which become inefficient for practical use with large datasets. To demonstrate the last claim, we present in Figure 11 the nonzero entries of 4 matrices corresponding to 4 bases with different values of k. In these matrices, each column corresponds to one basis vector. Since the tree $T_{\mathcal{X}}(k)$ is binary, all but the first $2k$ columns are zero on at least half of the data. We can clearly see that the sparsity of these matrices decreases as k grows. Note that for $k = N$ the matrix is full, that is $22,500$ nonzero entries, and for values just above $k = 15$, which is 10% of N, the matrix has more nonzero elements than zeros.



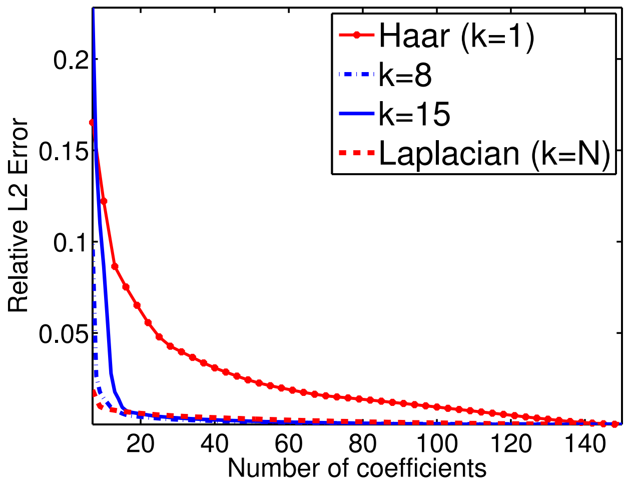
(a) $k = 1$ (Haar) (b) $k = 8$ (c) $k = 15$ (d) $k = 30$

Figure 11: The sparsity of the matrices corresponding to different LMW bases defined on $\mathcal{X}$ of (20). The elements of each basis are given as columns.

Our first numerical example is of the representation of the smooth function $\sin(4x)$. This function is presented in Figure 12(a). The relative approximation errors using a varying number of expansion coefficients are displayed in Figure 12(b). As expected, for a smooth function, the graph Laplacian basis gives the fastest decay of the error. However, despite the small support of most of the elements in the bases corresponding to the intermediate cases $k = 8$ and $k = 15$, the representation using these bases is almost as sparse as in the graph Laplacian basis.



(a) The function $\sin(4x)$



(b) L_2 relative error in different bases

Figure 12: Representing a smooth function.

In the second one-dimensional example, we use samples from the function

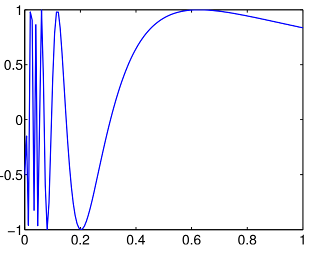
$$g(x) = \sin\left(\frac{1}{0.01 + 2x}\right), \tag{21}$$

shown in Figure 13(a). This function oscillates rapidly near zero. Thus, we expect the local bases to give better results in terms of the relative approximation error. As illustrated in Figure 13(b), the graph Laplacian basis provides the poorest approximation. As expected, the more localized bases exhibit better results. The Haar basis starts well with only a few coefficients in use, but then the intermediate case of $k = 8$ achieves the lowest approximation error.

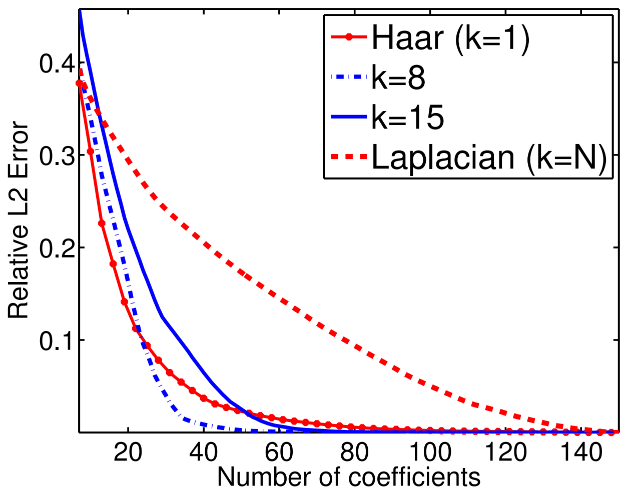
The last one-dimensional example is the piecewise smooth function

$$h(x) = \begin{cases} \sin(4x) & \text{if } x \in [0, \frac{1}{2}], \\ -\sin(4x) & \text{if } x \in (\frac{1}{2}, 1], \end{cases} \tag{22}$$

presented in Figure 14(a). The relative errors are illustrated in Figure 14(b). Here, both the $k = 8$ and $k = 15$ bases preform significantly better than the others. Note that in this example the jump discontinuity is aligned with the dyadic partition. In the next scenario



(a) The function $g(x)$ from (21)



(b) L_2 relative error in different bases

Figure 13: Representing an oscillatory function.

we will examine a case where the partition of the data is misaligned with the jump in the function.

Case 2 - data points randomly distributed on S^2

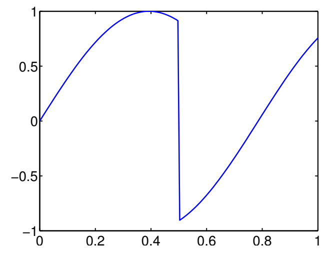
We choose 1000 data points in $\mathbb{R}^3$, uniformly distributed on the unit sphere S^2. As in the one-dimensional case, we compare the graph Laplacian and Haar bases, defined on S^2, with other LMW bases. As shown in the previous examples, lower values of k provide better results than higher values of k. Thus, in the following set of experiments, we set k to be 1% and 5% of N, namely, $k = 10$ and $k = 50$.

Henceforth, unless otherwise stated, for each dataset we construct a binary tree by using a recursive spectral partition – at each node of the tree we construct the graph Laplacian on the points in that node, compute the second eigenvector of the graph Laplacian, and partition the points into the left and right children at the median value of the eigenvector [40, 43].

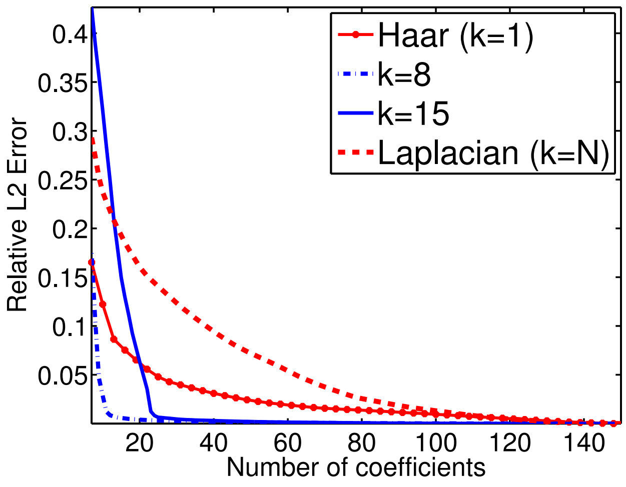
We study four functions defined on the sphere. Two smooth functions, a function with a region of high oscillations, and a quadratic function restricted to two sub-domains on the sphere. The latter is of special interest because we choose one sub-domain not to coincide with any sub-domain of the hierarchy tree. This allows to demonstrate the effect of the hierarchy tree on our representation.

Consider the Gaussian function,

$$G(x) = \exp(-\|x - x_0\|^2) \ , \tag{23}$$



(a) The function $h(x)$ from (22)



(b) L_2 relative error in different bases

Figure 14: Representing a piecewise smooth function.

with a fixed $x_0 \in S^2$. This function is presented in Figure 15(a) with its relative approximation errors in Figure 15(b). As $G(x)$ is smooth and has global support, the Haar basis performs poorly, while the graph Laplacian basis gives the best results. The intermediate cases tend to behave as the graph Laplacian in this example. Another example for the representation of a smooth function on the sphere is given by the function

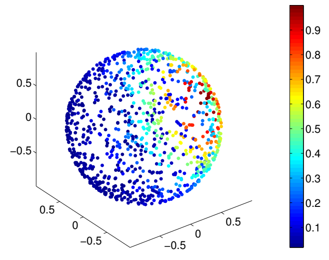
$$C(x) = \| \cos(2x) \|, \tag{24}$$

where the cosine is calculated componentwise. The function and its approximation errors are shown in Figures 16(a) and 16(b), respectively. The intermediate cases clearly show the significance of the generalized vanishing moments in this case, as the errors decrease as k grows.

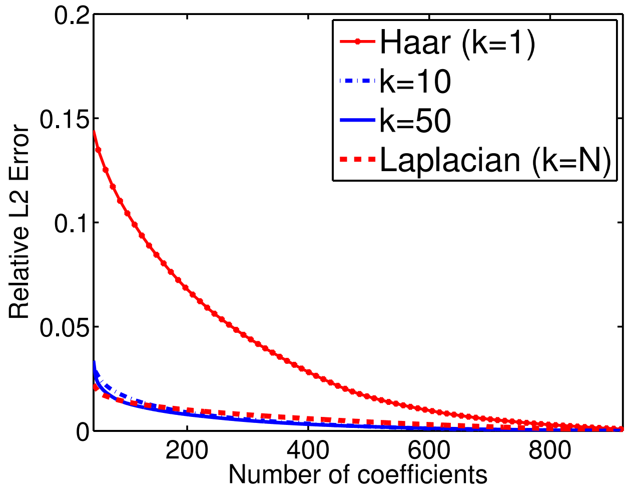
In the next examples, we use functions with some local behaviour. In Figures 17(a) and 17(b) we present the results for the function

$$R(x) = \sin\left((x^T x_0 + 0.2)^{-1}\right), \tag{25}$$

where x_0 is a fixed point on the sphere. This function has global support, however, it oscillates rapidly in regions on the sphere where x is close to being orthogonal to x_0. One can observe the advantage of bases having elements with local support over the basis of the graph Laplacian. For this example, the best choice for the parameter k is the intermediate

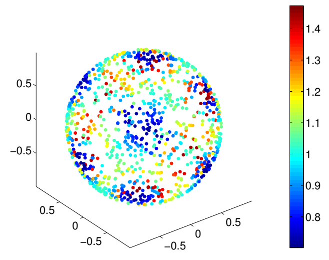


(a) The function $G(x)$ from (23)

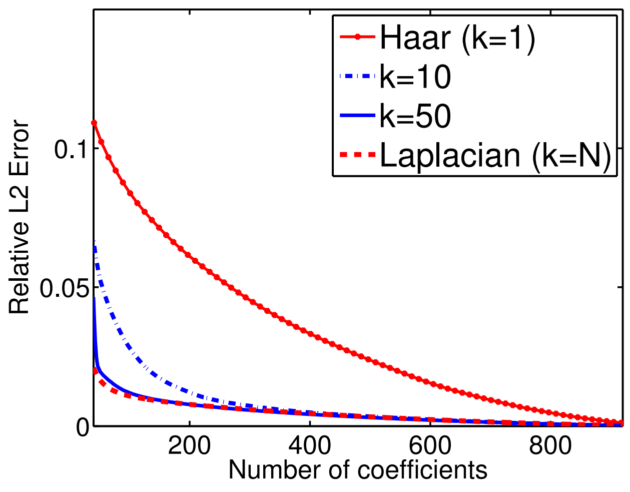


(b) L_2 relative error in different bases

Figure 15: Representing a Gaussian function on the sphere.

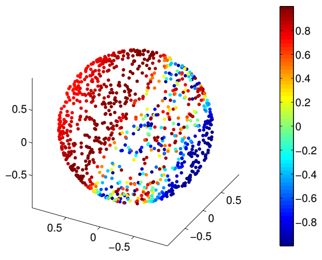


(a) The function $C(x)$ from (24)

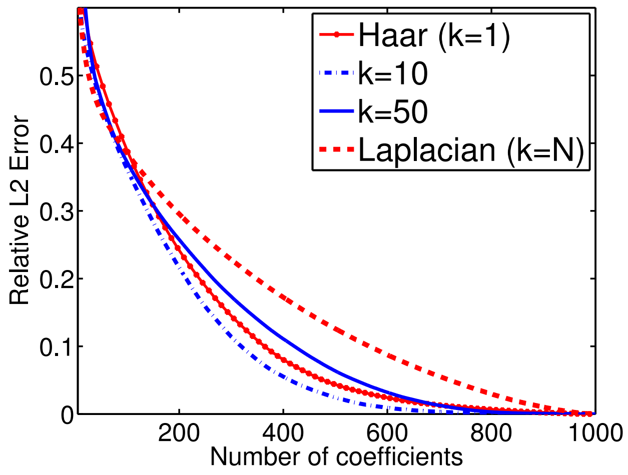


(b) L_2 relative error in different bases

Figure 16: Representing a smooth function on the sphere.



(a) The function $R(x)$ from (25)



(b) L_2 relative error in different bases

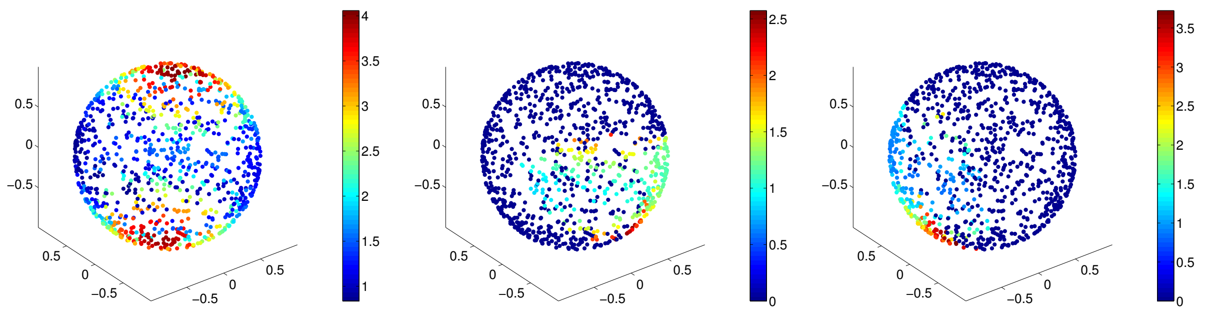
Figure 17: Representing a rapidly changing function on the sphere.

value of $k = 10$.

A natural question is what is the effect of using a "bad" hierarchy tree. In some sense, wrong partitions are the Achilles heel of wavelets theory. The next simple example illustrates this phenomenon in our setting. We use a quadratic function of the form $x^TAx + x^Tb$ restricted to two different subsets of 250 data points (25% of the data). The first subset, denoted by A_1, coincides with one of the subsets in the hierarchy tree. The second subset, denoted by A_2, consists of the 250 data points that are closest to a fixed $x_0 \in S^2$. This subset does not coincide with any subset in the hierarchy tree. The function and its two restrictions are presented in Figure 18. The approximation results are presented in Figure 19. Obviously, the graph Laplacian is not affected by the hierarchy tree, and thus produces the same results when restricting the function to either A_1 or A_2. For the subset A_1, the three bases with $k = 1, 10, 50$ provide almost zero error when using about 200 coefficients, where the $k = 10$ basis exhibits the lowest approximation error. The Haar basis also exhibits low approximation errors but reveals again its difficulties in representing smooth functions (even on a local scale). For the subset A_2, we get error rates which seem to grow as k increases. Both $k = 10$ and $k = 1$ bases show almost identical best results.

Case 3 - large dataset on $\mathbb{R}^{1000}$

We next apply the algorithm to a large dataset consisting of $N = 100000$ data points in $\mathbb{R}^{1000}$. To introduce structure into this high-dimensional example, we design the samples to have low intrinsic dimension. Specifically, we take N points uniformly distributed on S^2,



(a) Original function (b) Restriction to A_1 (c) Restriction to A_2

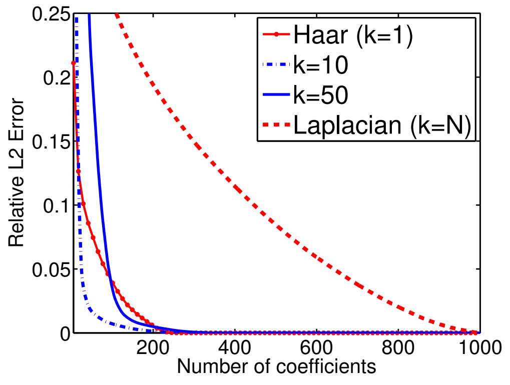
Figure 18: A quadratic function on the sphere (a) and its two restrictions to the subsets (b) A_1, which coincides with one of the subsets of the tree, and (c) A_2, which is not compatible with the tree partition.

and use them as coordinates in a coordinate system whose axes are three random orthogonal vectors in $\mathbb{R}^{1000}$.

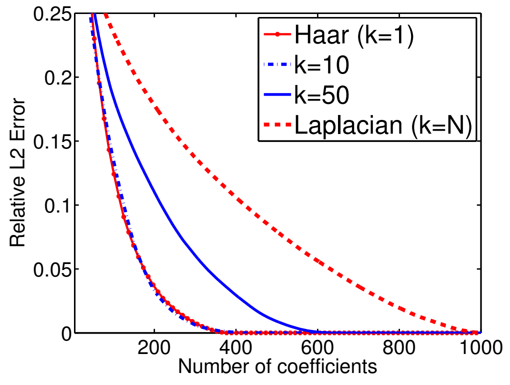
We compare the Haar basis with an intermediate one. As shown in previous examples, the value of k can be relatively small compared to N and still have an advantage over the Haar basis in representing smooth functions. Therefore, we choose $k = 15$. The graph Laplacian is constructed based on 30 nearest neighbours using a fast k-NN algorithm [28]. The kernel is $K(x, y) = e^{-x-y^2/2\varepsilon}$, where ε is chosen by the criterion described in [16].

We consider two test functions on this data. The first is a smooth function where the value at each point equals the point's first intrinsic coordinate (1 out of 3). The error comparison is presented in Figure 20(a), where the $k = 15$ basis clearly outperforms the Haar basis. Note that using the $k = 15$ basis for that smooth function yields about 1% relative error using only 50 largest coefficients, that are merely 0.05% of the coefficients. The second test function is the function $R(x)$ of (25), applied over the coordinates of the data, which contains a region of high oscillations (see above). The results are shown in Figure 20(b). The Haar basis exhibits better results up to 20 coefficients, after which the $k = 15$ basis outperforms it quickly, reaching an error of less than 2%.

Note that a few computational considerations become significant for such a large dataset. On one hand, a large value of k leads to a less sparse basis matrix (see Figure 11) and thus it increases the storage and the complexity of any calculation done with this basis. Furthermore, the runtime also grows due to the many eigenvectors calculated during the construction. On the other hand, a small value of the parameter k results in a deeper and larger tree (almost twice as large for Haar than for $k = 15$). This requires more storage space and also increases the runtime as the number of approximation levels grows.



(a) Results on restriction A_1



(b) Results on restriction A_2

Figure 19: The relative approximation error in representing a quadratic function on the sphere using two different trees: (a) the support of the function corresponds to one of the subsets of the tree partition (subset A_1) (b) the support of the function is not compatible with the tree partition (subset A_2). The results show the effect of the tree partition for bases with $k < N$.

7.2. Compression of hyperspectral images

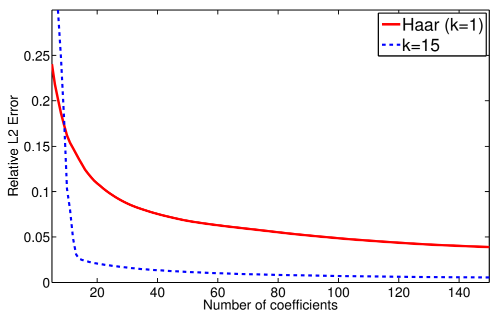
Next, we demonstrate our algorithm on a real dataset of a hyperspectral image. Hyperspectral data analysis has become increasingly popular in the last few years due to the growing number of applications that produce or use such data, such as earth monitoring applications.

A hyperspectral image is an image where each pixel is associated with a vector of measurements at various wavelengths [12]. The rationale behind measuring many wavelengths is that different materials have different signatures at different wavelengths. Typically, this type of data is collected by a remote sensing platform such as a satellite or an aircraft.

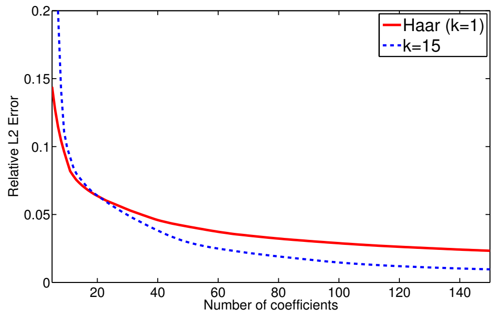
The data which we use to construct our bases consist of a hyperspectral image of the visible spectral region. The function defined on the image is the surface temperature at each pixel. Our goal is to compress the image of the surface temperature which was derived from sensors in the non-visible LWIR (long-wave infrared) spectral region.

The image data for our example consists of 400 rows and 400 columns, with 12 different wavelengths per pixel. In other words, the input dataset for our algorithm includes $160,000$ vectors in $\mathbb{R}^{12}$. The 12 different wavelength measurements are presented as images in Figure 21.

As a conclusion from the large dataset example in the previous subsection we choose the parameter $k = 20$. We use a hierarchy tree partition based on spectral clustering, and the graph Laplacian is constructed based on 30 nearest neighbours using a fast k-NN algorithm [28] and with the kernel $K(x, y) = e^{-x-y^2/2\varepsilon}$, where ε is chosen by the criterion



(a) Representing a smooth function



(b) Representing a rapidly changing function

Figure 20: The relative approximation error in representing two functions defined on a low-dimensional manifold embedded in $\mathbb{R}^{1000}$.

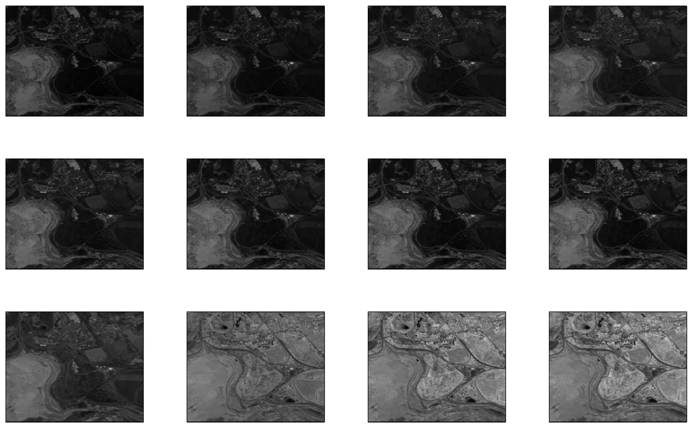


Figure 21: The 12 wavelength bands of the input hyperspectral image.

described in [16]. The algorithm was implemented in Matlab and was executed on a dual Intel Xeon(R) CPU X5560 @ 2.8GHz with 96GB of memory.

We compare our compression results with two (non-adaptive) image compression schemes. The first is the discrete cosine transform (DCT) and the second is the JPEG2000 compression. For compression using our basis we apply the best non-linear approximation (17).

The full image of surface temperature appears in Figure 22. The temperature was rescaled linearly to the range $[0, 255]$ for ease of manipulation. One can easily spot the urban islands which appear as high valued points. Other areas consist of a variety of surfaces including forests, roads and open fields.

The compression results are presented in Figures 23–25. In Figure 23 we show the compression using 0.125% of the coefficients, that is, 200 coefficients. The LMW based compres-

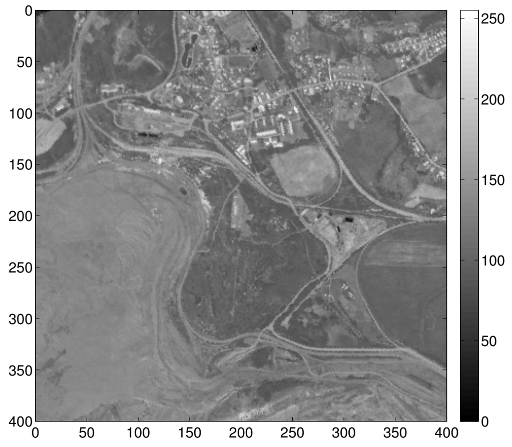


Figure 22: The image of surface temperature, acquired by long-wave infrared sensors. High values indicate warm locations and lower values indicate colder locations.

sion is shown in Figure 23(a), the DCT in Figure 23(b), and the JPEG2000 in Figure 23(c). As can be seen from these figures, almost none of the original image features are preserved when using the non-adaptive compressions.

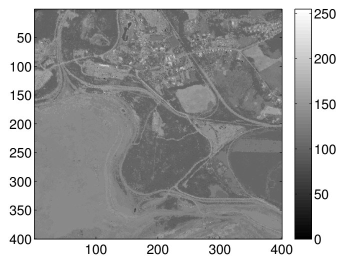
Another comparison is shown in Figure 24 for the LMW and JPEG2000 compressions, where we use 0.5% of coefficients, that is, 800 expansion coefficients. To highlight the differences, we present the results for a small part of the image.

To further quantify the results of the LMW based compression, we present the accumulated energy (L_2 norm) of the non-linear approximation, that is, the ratio between the energy of the compressed image and the original image, as a function of the number of coefficients we use. This measure complements the relative error measure of our approximation. These two measures are displayed in Figure 25 and indicate that compression using less than 1% of the coefficients retains up to 90% of the total energy with a relative error of about 8%.

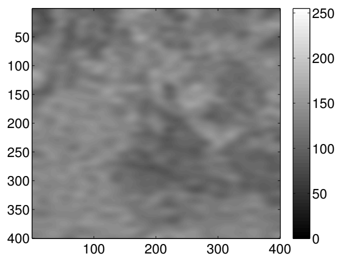
The code of all examples, including the full implementation of the LMW algorithm and the scripts for producing the examples, is available at the author's website `www.tau.ac.il/~nirsharo`.

8. Conclusions and discussion

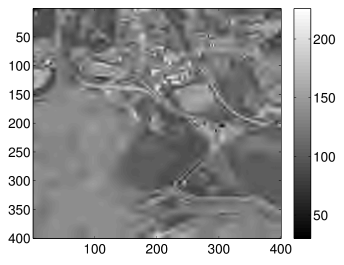
We presented a construction of a family of bases for representing functions defined on discrete scattered data. The bases of this family are characterized by a single integer parameter, which corresponds to the number of generalized vanishing moments on one hand, and determines the locality of the elements of the bases on the other hand. This parameter provides the flexibility of choosing the most appropriate basis to use, depending on the application.



(a) LMW based compression



(b) DCT



(c) JPEG2000

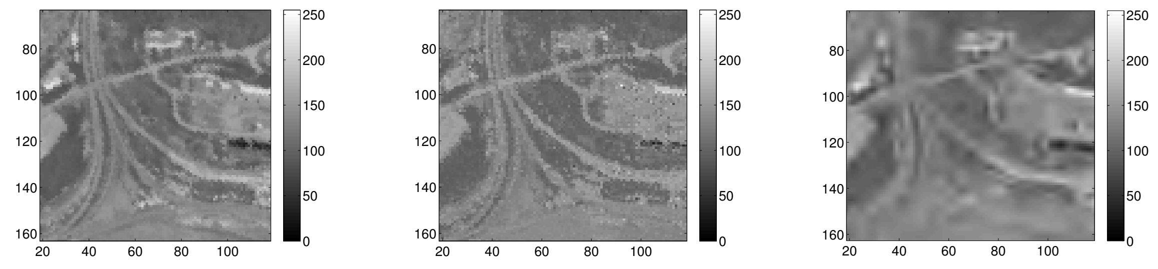
Figure 23: Compression using 0.125% of the coefficients.

We proved the connection between generalized vanishing moments and the decay of the expansion coefficients, and demonstrated it with numerical examples. In addition, we showed that multiscale elements (having varying support sizes) lead to a fast transform and enable to apply our construction to large datasets. Specifically, we illustrated our construction using a hyperspectral image. The resulting basis gives rise to sparse representations of a function on the data (an image of new modality) and enables the compression of such a function. We believe that this example reveals some of the potential of the LMW bases, for example, for compressing and denoising of large scale data, for calculating the inverses of operators, and for prediction tasks in machine learning.

During our study we encountered several theoretical issues to be addressed in future work. One such issue is related to the partition trees used. In particular, how to characterize or construct an optimal tree to be used as an input to the LMW algorithm. Another issue is to draw general guidelines for fine tuning the parameter k for a given a dataset.

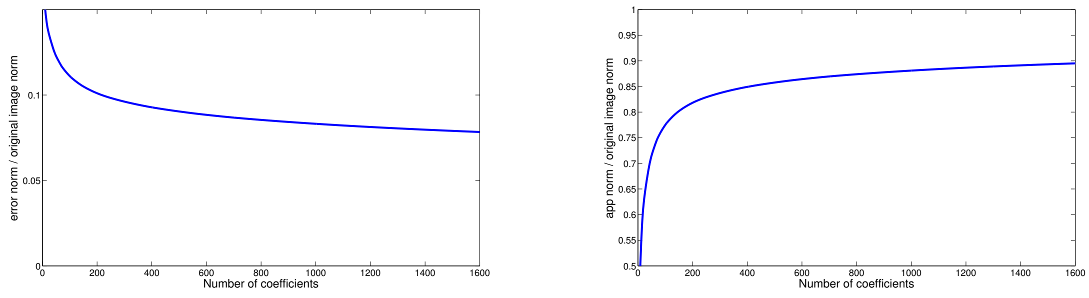
Acknowledgement

We wish to express our gratitude to Prof. Nira Dyn, Prof. Raphy Coifman, and Prof. Pencho Petrushev who shared their vast knowledge during many fruitful discussions. We also thank the anonymous reviewers for their valuable comments which helped improving



(a) Original subimage (b) LMW based compression (c) JPEG2000

Figure 24: Compression using 0.5% of the coefficients. Only a small region of the image is shown.



(a) Relative error (b) Accumulate energy

Figure 25: Evaluating LMW based compression using up to 1% of the expansion coefficients.

the paper.

The data for this study (Subsection 7.2) was supported by the FP7-project EO-Miners, grant agreement no. 2442242 and EUFAR under Transnational Access program - DeMinTIR project.

Appendix A. Proof of Theorem 6.2

Proof. First, one observes that the exponential function c^x, $x > 0$ is monotonically increasing for $c > 1$, a constant function for $c = 1$, and monotonically decreasing for $c < 1$, since its derivative is $\ln(c)c^x$.

Denote the expansion coefficients of f in the orthonormal basis $\{\phi_j\}$ by $\alpha_j = \langle f, \phi_j \rangle$. Then, since $\|f\| = 1$, the orthogonality of the basis functions $\{\phi_j\}$ yields $\sum_{j=1}^{N} |\alpha_j|^2 = 1$, and thus $0 \leq |\alpha_j| \leq 1$. By the monotonicity of c^x we get that $\tau_1 \leq \tau_2 < 2$ leads to

$|\alpha_j|^2 \le |\alpha_j|^{\tau_2} \le |\alpha_j|^{\tau_1}$. Therefore,

$$1 = \sum_{j=1}^{N} |\alpha_j|^2 \le \sum_{j=1}^{N} |\alpha_j|^{\tau_2} \le \sum_{j=1}^{N} |\alpha_j|^{\tau_1}.$$

Similarly, $0 < \frac{1}{\tau_2} \le \frac{1}{\tau_1}$ results in $1 \le |f|_{\tau_2} \le |f|_{\tau_1}$, which proves the first claim and first part of the second claim.

For the second claim, if $f = \pm\phi_{j^*}$ for some index j^*, then $|\alpha_{j^*}| = 1$ and $\alpha_j = 0$ for $j \ne j^*$. In this case, $|f|_\tau = 1$ for any $0 < \tau < 2$. Conversely, if $f \ne \pm\phi_j$ for all j, then, there exist $2 \le s \le N$ nonzero coefficients $0 < |\alpha_{j_k}| < 1$ such that

$$f = \sum_{k=1}^{s} \alpha_{j_k} \phi_{j_k}.$$

By using again the monotonicity of c^x, we get that for $\tau < 2$ it holds that $|\alpha_{j_k}|^2 < |\alpha_{j_k}|^\tau$ for $k = 1, \ldots, s$. Summing over k gives

$$1 = \sum_{k=1}^{s} |\alpha_{j_k}|^2 < \sum_{k=1}^{s} |\alpha_{j_k}|^\tau = |f|_\tau,$$

which concludes the proof of the second claim.

For the third claim, we look for $\max(\alpha_1^\tau + \cdots + \alpha_s^\tau)$ such that $\alpha_1, \ldots, \alpha_s > 0$ and $\alpha_1^2 + \cdots + \alpha_s^2 = 1$. To resolve this problem, we define the Lagrangian,

$$L(\alpha_1, \ldots, \alpha_s, \lambda) = \alpha_1^\tau + \cdots + \alpha_s^\tau + \lambda(\sum_{j=1}^{s} \alpha_j^2 - 1)$$

and deduce that the extremum is achieved for $\lambda = -\frac{\tau}{2}\alpha_j^{\tau-2}$. Since we have the lower bound of the second claim, the extremum is a maximum, and is obtained for $\alpha_1 = \alpha_2 = \cdots = \alpha_s = \frac{1}{\sqrt{s}}$. Given a function f with such expansion coefficients, a direct calculation gives $|f|_\tau = s^{\frac{1}{\tau}-\frac{1}{2}}$ where $\frac{1}{\tau} - \frac{1}{2} > 0$. $\square$

[1] Aflalo, Y., Kimmel, R., Brezis, H., 2013. On the optimality of the Laplace Beltrami eigenfunction to represent function of H^1. Tech. rep., Technion University, Haifa 3200, Israel.

[2] Alpert, B., Beylkin, G., Coifman, R., Rokhlin, V., 1993. Wavelet-like bases for the fast solution of second-kind integral equations. SIAM J. Sci. Comput. 14 (1), 159–184.

[3] Auscher, P., Hytönen, T., 2013. Orthonormal bases of regular wavelets in spaces of homogeneous type. Applied and Computational Harmonic Analysis 34 (2), 266–296.

[4] Belkin, M., Niyogi, P., 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in neural information processing systems (NIPS) 14, 585–591.

[5] Belkin, M., Niyogi, P., 2002. Using manifold structure for partially labeled classification. Advances in neural information processing systems (NIPS) 15, 929–936.

[6] Belkin, M., Niyogi, P., 2003. Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation 15 (6), 1373–1396.

[7] Belkin, M., Niyogi, P., 2008. Convergence of Laplacian eigenmaps. preprint.

[8] Binev, P., Cohen, A., Dahmen, W., DeVore, R., Temlyakov, V., 2005. Universal algorithms for learning theory part I: piecewise constant functions. Journal of Machine Learning Research 6 (2), 1297–1321.

[9] Binev, P., DeVore, R., 2004. Fast computation in adaptive tree approximation. Numerische Mathematik 97 (2), 193–217.

[10] Bremer, J. C., Coifman, R. R., Maggioni, M., Szlam, A. D., 2006. Diffusion wavelet packets. Applied and Computational Harmonic Analysis 21 (1), 95–112.

[11] Camarinha, M., Silva Leite, F., Crouch, P., 2001. On the geometry of Riemannian cubic polynomials. Differential Geometry and its Applications 15 (2), 107–135.

[12] Chang, C.-I., 2007. Hyperspectral data exploitation: theory and applications. Wiley-Interscience.

[13] Chapelle, O., Schölkopf, B., Zien, A., et al. (Eds.), 2006. Semi-Supervised Learning. MIT Press, Cambridge, MA.

[14] Chung, F. R., 1997. Spectral graph theory. Vol. 92. AMS Bookstore.

[15] Coifman, R., Maggioni, M., 2006. Diffusion wavelets. Applied and Computational Harmonic Analysis 21 (1), 53–94.

[16] Coifman, R., Shkolnisky, Y., Sigworth, F., Singer, A., oct. 2008. Graph Laplacian tomography from unknown random projections. IEEE Transactions on Image Processing 17 (10), 1891 –1899.

[17] Daley, R., 1994. Atmospheric data analysis. Cambridge Atmospheric and Space Science Series. Cambridge University Press.

[18] Daubechies, I., 1992. Ten lectures on wavelets. CBMS-NSF Regional Conference Series in Applied Mathematics (Book 61). SIAM: Society for Industrial and Applied Mathematics.

[19] David, G., Averbuch, A., 2012. Hierarchical data organization, clustering and denoising via localized diffusion folders. Applied and Computational Harmonic Analysis 33 (1), 1–23.

[20] De Boor, C., Ron, A., 1990. On multivariate polynomial interpolation. Constructive Approximation 6 (3), 287–302.

[21] DeVore, R. A., 1998. Nonlinear approximation. Acta numerica 7, 51–150.

[22] Donoho, D. L., 1997. CART and best-ortho-basis: a connection. Ann. Statist. 25 (5), 1870–1911.

[23] Gan, G., Ma, C., Wu, J., 2007. Data clustering. SIAM: Society for Industrial and Applied Mathematics.

[24] Gavish, M., Nadler, B., Coifman, R. R., 2010. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In: Fürnkranz, J., Joachims, T. (Eds.), Proceedings of the 27th International Conference on Machine Learning (ICML-10). Omnipress, pp. 367–374.

[25] Golub, G. H., Van Loan, C. F., 1996. Matrix Computations, 3rd Edition. Johns Hopkins University Press, Baltimore, MD, USA.

[26] Halko, N., Martinsson, P., Tropp, J., 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. SIAM review 53 (2), 217–288.

[27] Hammond, D., Vandergheynst, P., Gribonval, R., 2011. Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis 30 (2), 129–150.

[28] Jones, P. W., Osipov, A., Rokhlin, V., 2013. A randomized approximate nearest neighbors algorithm. Applied and Computational Harmonic Analysis 34 (3), 415 – 444.

[29] Kushnir, D., Galun, M., Brandt, A., 2010. Efficient multilevel eigensolvers with applications to data analysis tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (8), 1377–1391.

[30] Lee, A. B., Nadler, B., Wasserman, L., 06 2008. Rejoinder of: Treelets - an adaptive multi-scale basis for spare unordered data. The Annals of Applied Statistics 2 (2), 494–500.

[31] Lenglet, C., Rousson, M., Deriche, R., Faugeras, O., 2006. Statistics on the manifold of multivariate normal distributions: Theory and application to diffusion tensor MRI processing. Journal of Mathematical Imaging and Vision 25 (3), 423–444.

[32] Michalewicz, Z., 1996. Genetic algorithms+ data structures. Springer.

[33] Murtagh, F., 2007. The Haar wavelet transform of a dendrogram. J. Classification 24 (1), 3–32.

[34] Narcowich, F., Petrushev, P., Ward, J., 2006. Decomposition of Besov and Triebel–Lizorkin spaces on the sphere. Journal of Functional Analysis 238 (2), 530–564.

[35] Petrushev, P. P., 1988. Direct and converse theorems for spline and rational approximation and Besov spaces. In: Function spaces and applications. Springer, pp. 363–377.

[36] Ram, I., Elad, M., Cohen, I., 2011. Generalized tree-based wavelet transform. IEEE Transactions on Signal Processing 59 (9), 4199–4209.

[37] Saito, N., 2008. Data analysis and representation on a general domain using eigenfunctions of Laplacian. Applied and Computational Harmonic Analysis 25 (1), 68–97.

[38] Saito, N., Woei, E., 2009. Analysis of neuronal dendrite patterns using eigenvalues of graph Laplacians. Japan SIAM Letters 1, 13–16.

[39] Saul, L., Roweis, S., 2003. Think globally, fit locally: unsupervised learning of low dimensional manifolds. The Journal of Machine Learning Research 4, 119–155.

[40] Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8), 888–905.

[41] Singer, A., 2006. From graph to manifold Laplacian: The convergence rate. Applied and Computational Harmonic Analysis 21 (1), 128–134.

[42] Szlam, A. D., Maggioni, M., Coifman, R. R., BremerJr, J. C., 2005. Diffusion-driven multiscale analysis on manifolds and graphs: top-down and bottom-up constructions. In: Optics & Photonics 2005. International Society for Optics and Photonics, pp. 59141D–59141D.

[43] Von Luxburg, U., 2007. A tutorial on spectral clustering. Statistics and computing 17 (4), 395–416.

[44] Xu, R., Wunsch, D., 2009. Clustering. Wiley-IEEE Press.