Samplets: Construction and scattered data compression

Helmut Harbrecht, Michael Multerer

Departement Mathematik und Informatik
Fachbereich Mathematik
Universität Basel
CH-4051 Basel

Preprint No. 2022-05
February 2022

dmi.unibas.ch

SAMPLETS: CONSTRUCTION AND SCATTERED DATA COMPRESSION

HELMUT HARBRECHT AND MICHAEL MULTERER

Abstract. We introduce the concept of samplets by transferring the construction of Tausch-White wavelets to scattered data. This way, we obtain a multiresolution analysis tailored to discrete data which directly enables data compression, feature detection and adaptivity. The cost for constructing the samplet basis and for the fast samplet transform, respectively, is $\mathcal{O}(N)$, where N is the number of data points. Samplets with vanishing moments can be used to compress kernel matrices, arising, for instance, kernel based learning and scattered data approximation. The result are sparse matrices with only $\mathcal{O}(N \log N)$ remaining entries. We provide estimates for the compression error and present an algorithm that computes the compressed kernel matrix with computational cost $\mathcal{O}(N \log N)$. The accuracy of the approximation is controlled by the number of vanishing moments. Besides the cost efficient storage of kernel matrices, the sparse representation enables the application of sparse direct solvers for the numerical solution of corresponding linear systems. In addition to a comprehensive introduction to samplets and their properties, we present numerical studies to benchmark the approach. Our results demonstrate that samplets mark a considerable step in the direction of making large scattered data sets accessible for multiresolution analysis.

1. Introduction

Multiresolution methods and wavelet techniques in particular have a long standing tradition and are a versatile tool in many different fields. Applications comprise nonlinear approximation, image analysis, signal processing and machine learning, see for instance [6,10,15,16,31,32] and the references therein. Starting from a signal, the pivotal idea of wavelet techniques is the splitting of this signal into its contributions relative to a hierarchy of scales. Such a multiresolution ansatz starts from an approximation on a coarse scale and successively resolves details, that have not been captured so far, at finer scales. Therefore, multiresolution methods naturally accommodate data compression and adaptivity. In particular, the transformation of a signal into its wavelet representation and the backward transformation can be performed with linear cost in terms of the size of the wavelet basis, see for instance [8]. The classical construction of wavelets is based on dilations and translations of a given mother wavelet. This way, a nested sequence of approximation spaces is obtained, where the elements of this sequence are scaled copies of each other. As a consequence, the classical construction of wavelets is limited to structured data, such as uniform subdivisions of the real line. Adaptions to deal with intervals have been suggested in [2, 7, 12], while wavelet constructions on manifolds are the topic of [14, 26, 41]. An extension to (surface) triangulations, has been suggested in [39], where (multi-)wavelets are constructed as linear combinations of functions at a fixed fine scale. In particular, the stability of the resulting basis, which is known as Tausch-White wavelets, is guaranteed by its orthonormality. An approach to obtain a multiresolution analysis on unstructured data, for example on graphs, are *diffusion wavelets*, see [9]. However, there is no linear cost bound for the computation of a diffusion wavelet basis.

In this article, generalize the concept of Tausch-White wavelets towards scattered data. To this end, we modify the construction from [1, 39] and construct a multiresolution analysis which consists of localized and discrete signed measures. Inspired by the term wavelet, we call such signed measures *samplets*. Samplets are tailored towards the underlying data set and can be constructed such that their associated measure integrals vanish for polynomial integrands. If this is the case for all polynomials of total degree less or equal than q, we say that the samplets have *vanishing moments* of order $q+1$. Lowest order samplets, i.e. $q=0$, have been considered earlier for data compression in [35]. The construction of samplets is, however, not limited to the use of polynomial vanishing moments. Indeed, it is easily be possible to adapt the construction to other primitives with different desired properties. We present a general construction template for samplets with an arbitrary number of vanishing moments. This construction can always be performed with linear cost for a balanced cluster tree, even for non-quasi-uniform data. Moreover, the obtained basis is always orthonormal and hence stable. Representing scattered data by samplets, there is a fast decay of the samplet coefficients with respect to the support size if the data are smooth, due to the vanishing moments. This straightforwardly enables data compression. In contrast, non-smooth regions in the data are indicated by large samplet coefficients. This, in turn, enables feature detection and extraction. As examples we shall consider time-series data, images and unstructured point clouds in three spatial dimensions. Furthermore, we provide rigorous estimates for the decay of the samplet coefficients based on the local regularity of the underlying signal.

In addition to the construction of samplets and signal compression, we consider the compression of kernel matrices, as they arise in kernel based learning and scattered data approximation, compare [17,27,36,42,43,44]. Kernel matrices are typically densely populated, since the underlying kernels are nonlocal. Nonetheless, these kernels are usually *asymptotically smooth*, meaning that they behave like smooth functions apart from the diagonal. Cluster methods, such the fast multipole method, see [21,33,45], or hierarchical matrices, cp. [5,22], exploit this asymptotical smoothness to obtain a data-sparse representation of the kernel matrix by means of blockwise low-rank approximations. In turn, the discretization of asymptotical smooth kernels employing a samplet or a wavelet basis with vanishing moments results in quasi-sparse kernel matrices, i.e. they can be compressed such that only a sparse matrix remains, compare [4, 11, 13, 38, 40], where this has been shown for the wavelet case. We derive corresponding compression error estimates for samplets and present an algorithm with almost linear runtime to compute the compressed matrix. In [25], it has been numerically demonstrated that nested dissection, see [18, 30], is applicable to obtain a fill-in reducing reordering of such compressed matrices in the standard form. This reordering in turn allows for the rapid factorization of the compressed matrix by the Cholesky factorization without introducing additional errors. A reordering approach based on operator adapted wavelets, cf. [34], is discussed in [37]. The latter is, however, only proven to work for Green's functions with homogenous boundary conditions on Lipschitz domains. The approximate Cholesky factorization was also computed for matrices given in wavelet coordinates by means of the non-standard form in [20] and with the aid of hierarchical matrices, see for instance [22].

The rest of this article is organized as follows. In Section 2, the concept of samplets is introduced. The subsequent Section 3 is devoted to the construction of samplets and to their properties. The change of basis by means of the fast samplet transform is the topic of Section 4. Section 5 deals with the samplet compression of kernel matrices. Especially, we recapitulate certain $\mathcal{H}^2$-matrix techniques and leverage them to efficiently compute the compressed kernel matrix. In Section 6, we numerically demonstrate the capabilities of samplets for data compression and the compression of kernel matrices. Numerical results in up to four dimensions are provided. Finally, in Section 7, we state concluding remarks.

Throughout this article, in order to avoid the repeated use of generic but unspecified constants, by $C \lesssim D$ we indicate that C can be bounded by a multiple of D, independently of parameters which C and D may depend on. Moreover, $C \gtrsim D$ is defined as $D \lesssim C$ and $C \sim D$ as $C \lesssim D$ and $D \lesssim C$.

2. Samplets

Let $X := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$ denote a set of points within some bounded or unbounded region $\Omega \subset \mathbb{R}^d$. Associated to each point $\boldsymbol{x}_i$, we introduce the Dirac measure

$$\delta_{\boldsymbol{x}_i}(\boldsymbol{x}) := \begin{cases} 1, & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \\ 0, & \text{otherwise.} \end{cases}$$

With a slight abuse of notation, we define the point evaluation functional according to

$$(f, \delta_{\boldsymbol{x}_i})_\Omega = \int_\Omega f(\boldsymbol{x}) \delta_{\boldsymbol{x}_i}(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} := \int_\Omega f(\boldsymbol{x}) \delta_{\boldsymbol{x}_i}(\mathrm{d}\boldsymbol{x}) = f(\boldsymbol{x}_i),$$

where $f \in C(\Omega)$ is a continuous function.

Next, we define the space $\mathcal{X} := \operatorname{span}\{\delta_{\boldsymbol{x}_1}, \ldots, \delta_{\boldsymbol{x}_N}\}$ as the N-dimensional vector space of all discrete and finite signed measures supported at the points in X. An inner product on $\mathcal{X}$ is given by

$$\langle u, v \rangle_{\mathcal{X}} := \sum_{i=1}^N u_i v_i, \quad \text{where } u = \sum_{i=1}^N u_i \delta_{\boldsymbol{x}_i},\ v = \sum_{i=1}^N v_i \delta_{\boldsymbol{x}_i}.$$

Indeed, the space $\mathcal{X}$ is isometrically isomorphic to $\mathbb{R}^N$ endowed with the canonical inner product. To construct a multiresolution analysis, we introduce the spaces $\mathcal{X}_j := \operatorname{span} \boldsymbol{\Phi}_j$, where

$$\boldsymbol{\Phi}_j := \{\varphi_{j,k} : k \in I_j\}.$$

Herein, I_j denotes a suitable index set with cardinality $|I_j| = \dim \mathcal{X}_j$ and $j \in \mathbb{N}$ is referred to as *level*. Moreover, each basis element $\varphi_{j,k}$ is a linear combination of Dirac measures such that

$$\langle \varphi_{j,k}, \varphi_{j,k'} \rangle_{\mathcal{X}} = 0 \quad \text{for } k \neq k'.$$

In what follows, we shall identify bases by row vectors, such that, for $\boldsymbol{v}_j = [v_{j,k}]_{k \in I_j}$, the corresponding measure can simply be written as a dot product according to

$$v_j = \boldsymbol{\Phi}_j \boldsymbol{v}_j = \sum_{k \in I_j} v_{j,k} \varphi_{j,k}.$$

Rather than using the multiresolution analysis corresponding to the hierarchy

$$\mathcal{X}_0 \subset \mathcal{X}_1 \subset \cdots \subset \mathcal{X},$$

the idea of samplets is to keep track of the increment of information between two consecutive levels j and $j+1$. Since we have $\mathcal{X}_j \subset \mathcal{X}_{j+1}$, we may decompose

$$\mathcal{X}_{j+1} = \mathcal{X}_j \overset{\perp}{\oplus} \mathcal{S}_j \tag{1}$$

by using the *detail space* $\mathcal{S}_j$. Of practical interest is the choice of the basis of the detail space $\mathcal{S}_j$ in $\mathcal{X}_{j+1}$. This basis is assumed to be orthonormal as well and will be denoted by

$$\boldsymbol{\Sigma}_j = \big\{\sigma_{j,k} : k \in I_j^\Sigma := I_{j+1} \setminus I_j\big\}.$$

Recursively applying decomposition (1), we notice that the set

$$\boldsymbol{\Sigma}_J = \boldsymbol{\Phi}_0 \cup \bigcup_{j=0}^{J-1} \boldsymbol{\Sigma}_j$$

forms a basis of $\mathcal{X}_J := \mathcal{X}$, which we call a *samplet basis*. In view of data compression, an essential ingredient is the vanishing moment condition, meaning that

$$(p, \sigma_{j,k})_\Omega = 0 \quad \text{for all } p \in \mathcal{P}_q(\Omega), \tag{2}$$

where $\mathcal{P}_q(\Omega)$ denotes the space of all polynomials with total degree at most q. We say then that the samplets have $q+1$ *vanishing moments*.

Remark 2.1. *For quasi-uniform points, i.e. if the separation radius* $q_X := \frac{1}{2}\min_{i\neq j}\|\boldsymbol{x}_i-\boldsymbol{x}_j\|_2$ *of* X *is similar to the fill distance* $h_{X,\Omega} := \sup_{\boldsymbol{x}\in\Omega}\min_{\boldsymbol{x}_i\in X}\|\boldsymbol{x}-\boldsymbol{x}_i\|_2$ *in the sense that* $q_X \sim h_{X,\Omega}$, *we obtain bases which satisfy* $\operatorname{diam}(\operatorname{supp}\varphi_{j,k}) := \operatorname{diam}(\{\boldsymbol{x}_{i_1},\ldots,\boldsymbol{x}_{i_p}\}) \sim 2^{-j/d}$ *and, likewise,*

$$\operatorname{diam}(\operatorname{supp}\sigma_{j,k}) \sim 2^{-j/d}. \tag{3}$$

These properties are favorable with regard to the compression of data and the compression of kernel matrices. However, we stress that this is not a requirement in our construction.

Remark 2.2. *The concept of samplets has a very natural interpretation in the context of reproducing kernel Hilbert spaces, compare [3]. If* $(\mathcal{H},\langle\cdot,\cdot\rangle_{\mathcal{H}})$ *is a reproducing kernel Hilbert space with reproducing kernel* $\mathcal{K}$, *then there holds* $(f,\delta_{\boldsymbol{x}_i})_\Omega = \langle\mathcal{K}(\boldsymbol{x}_i,\cdot),f\rangle_{\mathcal{H}}$. *Hence, the samplet* $\sigma_{j,k} = \sum_{\ell=1}^p \beta_\ell\delta_{\boldsymbol{x}_{i_\ell}}$ *can be identified with the function*

$$\hat{\sigma}_{j,k} := \sum_{\ell=1}^{p}\beta_\ell\mathcal{K}(\boldsymbol{x}_{i_\ell},\cdot)\in\mathcal{H}.$$

Especially, it holds $\langle\hat{\sigma}_{j,k},h\rangle_{\mathcal{H}} = 0$ *for any* $h\in\mathcal{H}$ *which satisfies* $h|_{\operatorname{supp}\sigma_{j,k}}\in\mathcal{P}_q(\operatorname{supp}\sigma_{j,k})$.

3. Construction and properties of samplets

3.1. **Cluster tree.** To construct samplets with the desired properties, especially vanishing moments, cp. (2), we shall transfer the wavelet construction from [39] into our setting. The first step is to construct a hierarchy subspaces of signed measures. To this end, we perform a hierarchical clustering of the set X.

Definition 3.1. *Let* $\mathcal{T} = (V,E)$ *be a tree with vertices* V *and edges* E. *We define its set of leaves as* $\mathcal{L}(\mathcal{T}) := \{\nu\in V\colon \nu$ *has no sons*$\}$. *The tree* $\mathcal{T}$ *is a* cluster tree *for the set* $X = \{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N\}$, *iff* X *is the root of* $\mathcal{T}$ *and all* $\nu\in P\setminus\mathcal{L}(\mathcal{T})$ *are disjoint unions of their sons.*

The level j_ν *of* $\nu\in\mathcal{T}$ *is its distance from the root, i.e. the number of edges that are required for traveling from* X *to* ν. *The* depth J *of* $\mathcal{T}$ *is the maximum level of all clusters. We define the set of clusters on level* j *as* $\mathcal{T}_j := \{\nu\in\mathcal{T}\colon \nu$ *has level* $j\}$. *Finally, the* bounding box B_ν *of* ν *is the smallest axis-parallel cuboid that contains all its points.*

There exist several choices for the construction of a cluster tree for the set X. Within this article, we will exclusively consider binary trees and remark that other options, such as 2^d-trees, are possible, with the obvious modifications. Definition 3.1 provides a hierarchical cluster structure on the set X. Even so, it does not provide guarantees for the cardinalities of the clusters. Therefore, we introduce the concept of a *balanced binary tree*.

Definition 3.2. *Let* $\mathcal{T}$ *be a cluster tree for* X *with depth* J. $\mathcal{T}$ *is called a* balanced binary tree, *if all clusters* ν *satisfy the following conditions:*

(*i*) *The cluster* ν *has exactly two sons if* $j_\nu < J$. *It has no sons if* $j_\nu = J$.
(*ii*) *It holds* $|\nu| \sim 2^{J-j_\nu}$.

A balanced binary tree can be constructed by *cardinality balanced clustering*. This means that the root cluster is split into two son clusters of identical (or similar) cardinality. This process is repeated recursively for the resulting son clusters until their cardinality falls below a certain threshold. For the subdivision, the cluster's bounding box is split along its longest edge such that the resulting two boxes both contain an equal number of points. Thus, as the cluster cardinality halves with each level, we obtain $\mathcal{O}(\log N)$ levels in total. The total cost for constructing the cluster tree is therefore $\mathcal{O}(N\log N)$. Finally, we remark that a balanced tree is only required to guarantee the cost bounds for the presented algorithms. The error and compression estimates we shall present later on are robust in the sense that they are formulated directly in terms of the actual cluster sizes rather than the introduced cluster level.

3.2. **Construction of samplet bases.** Having a cluster tree at hand, we shall now construct a samplet bases on the resulting hierarchical structure. We begin by introducing a *two-scale* transform between basis elements on a cluster ν of level j. To this end, we create *scaling distributions* $\boldsymbol{\Phi}_j^\nu = \{\varphi_{j,k}^\nu\}$ and *samplets* $\boldsymbol{\Sigma}_j^\nu = \{\sigma_{j,k}^\nu\}$ as linear combinations of the scaling distributions $\boldsymbol{\Phi}_{j+1}^\nu$ of ν's son clusters. This results in the *refinement relation*

$$[\boldsymbol{\Phi}_j^\nu, \boldsymbol{\Sigma}_j^\nu] := \boldsymbol{\Phi}_{j+1}^\nu \boldsymbol{Q}_j^\nu = \boldsymbol{\Phi}_{j+1}^\nu \big[\boldsymbol{Q}_{j,\Phi}^\nu, \boldsymbol{Q}_{j,\Sigma}^\nu\big]. \tag{4}$$

In order to provide both, vanishing moments and orthonormality, the transformation $\boldsymbol{Q}_j^\nu$ has to be appropriately constructed. For this purpose, we consider an orthogonal decomposition of the *moment matrix*

$$\boldsymbol{M}_{j+1}^\nu := \begin{bmatrix} (\boldsymbol{x}^{\boldsymbol{0}}, \varphi_{j+1,1})_\Omega & \cdots & (\boldsymbol{x}^{\boldsymbol{0}}, \varphi_{j+1,|\nu|})_\Omega \\ \vdots & & \vdots \\ (\boldsymbol{x}^{\boldsymbol{\alpha}}, \varphi_{j+1,1})_\Omega & \cdots & (\boldsymbol{x}^{\boldsymbol{\alpha}}, \varphi_{j+1,|\nu|})_\Omega \end{bmatrix} = [(\boldsymbol{x}^{\boldsymbol{\alpha}}, \boldsymbol{\Phi}_{j+1}^\nu)_\Omega]_{|\boldsymbol{\alpha}|\leq q} \in \mathbb{R}^{m_q\times|\nu|},$$

where

$$m_q := \sum_{\ell=0}^{q} \binom{\ell+d-1}{d-1} = \binom{q+d}{d} \leq (q+1)^d \tag{5}$$

denotes the dimension of $\mathcal{P}_q(\Omega)$.

In the original construction by Tausch and White, the matrix $\boldsymbol{Q}_j^\nu$ is obtained from a singular value decomposition of $\boldsymbol{M}_{j+1}^\nu$. For the construction of samplets, we follow the idea from [1] and rather employ the QR decomposition, which has the advantage of generating samplets with an increasing number of vanishing moments. It holds

$$(\boldsymbol{M}_{j+1}^\nu)^\intercal = \boldsymbol{Q}_j^\nu \boldsymbol{R} =: \big[\boldsymbol{Q}_{j,\Phi}^\nu, \boldsymbol{Q}_{j,\Sigma}^\nu\big]\boldsymbol{R} \tag{6}$$

Consequently, the moment matrix for the cluster's own scaling distributions and samplets is given by

$$\begin{aligned} \big[\boldsymbol{M}_{j,\Phi}^\nu, \boldsymbol{M}_{j,\Sigma}^\nu\big] &= \big[(\boldsymbol{x}^{\boldsymbol{\alpha}}, [\boldsymbol{\Phi}_j^\nu, \boldsymbol{\Sigma}_j^\nu])_\Omega\big]_{|\boldsymbol{\alpha}|\leq q} = \big[(\boldsymbol{x}^{\boldsymbol{\alpha}}, \boldsymbol{\Phi}_{j+1}^\nu[\boldsymbol{Q}_{j,\Phi}^\nu, \boldsymbol{Q}_{j,\Sigma}^\nu])_\Omega\big]_{|\boldsymbol{\alpha}|\leq q} \\ &= \boldsymbol{M}_{j+1}^\nu[\boldsymbol{Q}_{j,\Phi}^\nu, \boldsymbol{Q}_{j,\Sigma}^\nu] = \boldsymbol{R}^\intercal. \end{aligned} \tag{7}$$

As $\boldsymbol{R}^\intercal$ is a lower triangular matrix, the first $k-1$ entries in its k-th column are zero. This corresponds to $k-1$ vanishing moments for the k-th distribution generated by the transformation $\boldsymbol{Q}_j^\nu = [\boldsymbol{Q}_{j,\Phi}^\nu, \boldsymbol{Q}_{j,\Sigma}^\nu]$. By defining the first m_q distributions as scaling distributions and the remaining ones as samplets, we obtain samplets with vanishing moments at least up to order $q+1$. By increasing the polynomial degree to $\hat{q} > q$ at the leaf clusters such that $m_{\hat{q}} \geq 2m_q$, we can even construct samplets with an increased number of vanishing moments up to order $\hat{q}+1$ without any additional cost.

Remark 3.3. *The samplet construction using vanishing moments is inspired by the classical wavelet theory. However, it is easily possible to adapt the construction to other primitives than polynomials.*

Remark 3.4. *Each cluster has at most a constant number of scaling distributions and samplets. For a given cluster ν, their number is identical to the cardinality of $\boldsymbol{\Phi}_{j+1}^\nu$. For leaf clusters, this number is bounded by the leaf size. For non-leaf clusters, it is bounded by the number of scaling distributions from its son clusters. As there are at most two son clusters with a maximum of m_q scaling distributions each, we obtain the bound $2m_q$ for non-leaf clusters. If $\boldsymbol{\Phi}_{j+1}^\nu$ has less than m_q+1 elements, there are no samplets and all distributions are considered as scaling distributions.*

For leaf clusters, we define the scaling distributions by the Dirac measures supported at the points $\boldsymbol{x}_i$, i.e. $\boldsymbol{\Phi}_J^\nu := \{\delta_{\boldsymbol{x}_i} : \boldsymbol{x}_i \in \nu\}$. The scaling distributions of all clusters on a specific level j then generate the spaces

$$\mathcal{X}_j := \operatorname{span}\{\varphi_{j,k}^\nu : k \in I_j^\nu,\ \nu \in \mathcal{T}_j\}, \tag{8}$$

while the samplets span the detail spaces

$$\mathcal{S}_j := \operatorname{span}\left\{\sigma_{j,k}^\nu : k \in I_j^{\Sigma,\nu},\ \nu \in \mathcal{T}_j\right\} = \mathcal{X}_{j+1} \overset{\perp}{\ominus} \mathcal{X}_j. \tag{9}$$

Combining the scaling distributions of the root cluster with all clusters' samplets gives rise to the samplet basis

$$\boldsymbol{\Sigma}_N := \boldsymbol{\Phi}_0^X \cup \bigcup_{\nu\in\mathcal{T}} \boldsymbol{\Sigma}_j^\nu. \tag{10}$$

Writing $\boldsymbol{\Sigma}_N = \{\sigma_k : 1 \leq k \leq N\}$, where σ_k is either a samplet or a scaling distribution at the root cluster, we obtain a unique indexing of all the signed measures comprising the samplet basis. The indexing induces an order for the set $\boldsymbol{\Sigma}_N$. We choose this order to be level-dependent, i.e. the samplets of a cluster are grouped together, with those on finer levels having larger indices.

Remark 3.5. *The present construction of samplet bases on a balanced cluster tree can always be performed with linear cost $\mathcal{O}(N)$, we refer to [1] for a proof of this statement.*

3.3. Properties of samplets.

By construction, samplets satisfy the following properties, which can be inferred by adapting the corresponding results from [24, 39].

Theorem 3.6. *The spaces $\mathcal{X}_j$ defined in equation (8) form a multiresolution analysis*

$$\mathcal{X}_0 \subset \mathcal{X}_1 \subset \cdots \subset \mathcal{X}_J = \mathcal{X},$$

where the corresponding complement spaces $\mathcal{S}_j$ from (9) satisfy $\mathcal{S}_{j+1} = \mathcal{X}_j \overset{\perp}{\oplus} \mathcal{S}_j$ for all $j = 0, 1, \ldots, J-1$. The associated samplet basis $\boldsymbol{\Sigma}_N$ defined in (10) is an orthonormal basis in $\mathcal{X}$. In particular, there holds:

(i) *The number of all samplets on level j behaves like 2^j.*
(ii) *The samplets have $q+1$ vanishing moments.*
(iii) *Each samplet is supported in a specific cluster ν.*

Remark 3.7. *In the situation of Theorem 3.6, if the points in X are even quasi-uniform, then the diameter of the cluster satisfies $\operatorname{diam}(\nu) \sim 2^{-j_\nu/d}$ and it holds (3).*

Remark 3.8. *Due to $\mathcal{S}_j \subset \mathcal{X}$ and $\mathcal{X}_0 \subset \mathcal{X}$, we conclude that each samplet is a linear combination of the Dirac measures supported at the points in X. Especially, the related coefficient vectors $\boldsymbol{\omega}_{j,k}$ in the representations*

$$\sigma_{j,k} = \sum_{i=1}^N \omega_{j,k,i}\delta_{\boldsymbol{x}_i} \quad \text{and} \quad \varphi_{0,k} = \sum_{i=1}^N \omega_{0,k,i}\delta_{\boldsymbol{x}_i} \tag{11}$$

are pairwise orthonormal with respect to the inner product on $\mathbb{R}^N$.

Later on, the following bound on the samplets' coefficients $\|\cdot\|_1$-norm will be essential:

Lemma 3.9. *The coefficient vector $\boldsymbol{\omega}_{j,k} = \left[\omega_{j,k,i}\right]_i$ of the samplet $\sigma_{j,k}$ on the cluster ν fulfills*

$$\|\boldsymbol{\omega}_{j,k}\|_1 \leq \sqrt{|\nu|}. \tag{12}$$

The same bound holds for the coefficient vectors of the scaling distributions $\varphi_{j,k}$.

Proof. It holds $\|\boldsymbol{\omega}_{j,k}\|_{\ell^2} = 1$. Hence, the assertion follows immediately from the Cauchy-Schwarz inequality

$$\|\boldsymbol{\omega}_{j,k}\|_1 \le \sqrt{|\nu|}\|\boldsymbol{\omega}_{j,k}\|_2 = \sqrt{|\nu|}.$$

□

The key for data compression and feature detection is the following estimate which shows that the samplet coefficients decay with respect to the samplet's support size provided that the data result from the evaluation of a smooth function. Hence, in case of smooth data, the samplet coefficients are small and can be set to zero without compromising the accuracy. Vice versa, a large samplet coefficient indicates that the data are singular in the region of the samplet's support.

Lemma 3.10. *Let $f \in C^{q+1}(\Omega)$. Then, it holds for a samplet $\sigma_{j,k}$ supported on the cluster ν that*

$$|(f, \sigma_{j,k})_\Omega| \le \left(\frac{d}{2}\right)^{q+1} \frac{\operatorname{diam}(\nu)^{q+1}}{(q+1)!} \|f\|_{C^{q+1}(\Omega)} \|\boldsymbol{\omega}_{j,k}\|_1. \tag{13}$$

Proof. For $\boldsymbol{x}_0 \in \nu$, a Taylor expansion of f yields

$$f(\boldsymbol{x}) = \sum_{|\boldsymbol{\alpha}|\le q} \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial \boldsymbol{x}^{\boldsymbol{\alpha}}} f(\boldsymbol{x}_0) \frac{(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} + R_{\boldsymbol{x}_0}(\boldsymbol{x}).$$

Herein, the remainder $R_{\boldsymbol{x}_0}(\boldsymbol{x})$ reads

$$R_{\boldsymbol{x}_0}(\boldsymbol{x}) = (q+1) \sum_{|\boldsymbol{\alpha}|=q+1} \frac{(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!} \int_0^1 \frac{\partial^{q+1}}{\partial \boldsymbol{x}^{\boldsymbol{\alpha}}} f\big(\boldsymbol{x}_0 + s(\boldsymbol{x}-\boldsymbol{x}_0)\big)(1-s)^q \,\mathrm{d}s.$$

In view of the vanishing moments, we conclude

$$\begin{aligned}|(f, \sigma_{j,k})_\Omega| = |(R_{\boldsymbol{x}_0}, \sigma_{j,k})_\Omega| &\le \sum_{|\boldsymbol{\alpha}|=q+1} \max_{\boldsymbol{x}\in\nu} \frac{\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^{|\boldsymbol{\alpha}|}}{\boldsymbol{\alpha}!} \max_{\boldsymbol{x}\in\nu} \left|\frac{\partial^{q+1}}{\partial \boldsymbol{x}^{\boldsymbol{\alpha}}} f(\boldsymbol{x})\right| \|\boldsymbol{\omega}_{j,k}\|_1 \\ &\le \left(\frac{d}{2}\right)^{q+1} \frac{\operatorname{diam}(\nu)^{q+1}}{(q+1)!} \|f\|_{C^{q+1}(\Omega)} \|\boldsymbol{\omega}_{j,k}\|_1.\end{aligned}$$

Here, we used the identity

$$\sum_{|\boldsymbol{\alpha}|=q+1} \frac{2^{-(q+1)}}{\boldsymbol{\alpha}!} = \frac{2^{-(q+1)}}{(q+1)!} \sum_{|\boldsymbol{\alpha}|=q+1} \frac{(q+1)!}{\boldsymbol{\alpha}!} = \frac{1}{(q+1)!}\left(\frac{d}{2}\right)^{q+1},$$

which is obtained by choosing $\boldsymbol{x}_0$ as the cluster's midpoint and the multinomial theorem. □

4. Fast samplet transform

In order to transform between the samplet basis and the basis of Dirac measures, we introduce the *fast samplet transform* and its inverse. To this end, we assume that the data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ result from the evaluation of some (unknown) function $f\colon \Omega \to \mathbb{R}$, i.e.

$$y_i = f_i^\Delta = (f, \delta_{\boldsymbol{x}_i})_\Omega.$$

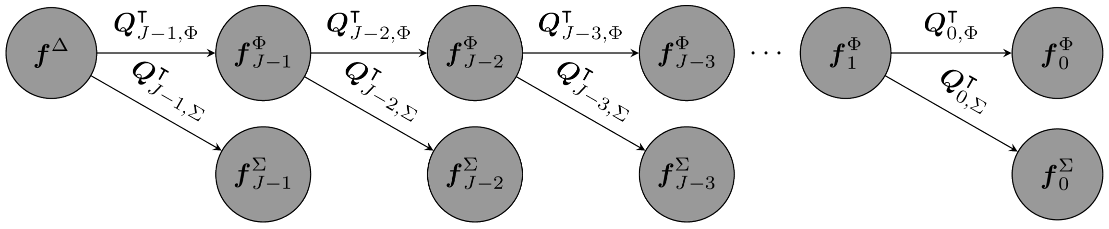
Hence, we may represent the function f on X according to

$$f = \sum_{i=1}^N f_i^\Delta \delta_{\boldsymbol{x}_i}.$$

Our goal is now to compute the representation

$$f = \sum_{i=1}^{N} f_k^{\Sigma} \sigma_k$$

with respect to a samplet basis. For the sake of a simpler notation, let $\boldsymbol{f}^{\Delta} := [f_i^{\Delta}]_{i=1}^N$ and $\boldsymbol{f}^{\Sigma} := [f_i^{\Sigma}]_{i=1}^N$ denote the associated coefficient vectors. Then, the samplet transform amounts to a change of basis $\boldsymbol{f}^{\Sigma} = \boldsymbol{T}\boldsymbol{f}^{\Delta}$ with an orthogonal matrix $\boldsymbol{T} \in \mathbb{R}^{N\times N}$. The actual implementation of this change of basis is, however, recursive.



$$\begin{tikzcd}
\boldsymbol{f}^{\Delta} \arrow[r, "\boldsymbol{Q}_{J-1,\Phi}^{\intercal}"] \arrow[dr, "\boldsymbol{Q}_{J-1,\Sigma}^{\intercal}"'] & \boldsymbol{f}_{J-1}^{\Phi} \arrow[r, "\boldsymbol{Q}_{J-2,\Phi}^{\intercal}"] \arrow[dr, "\boldsymbol{Q}_{J-2,\Sigma}^{\intercal}"'] & \boldsymbol{f}_{J-2}^{\Phi} \arrow[r, "\boldsymbol{Q}_{J-3,\Phi}^{\intercal}"] \arrow[dr, "\boldsymbol{Q}_{J-3,\Sigma}^{\intercal}"'] & \boldsymbol{f}_{J-3}^{\Phi} & \cdots & \boldsymbol{f}_{1}^{\Phi} \arrow[r, "\boldsymbol{Q}_{0,\Phi}^{\intercal}"] \arrow[dr, "\boldsymbol{Q}_{0,\Sigma}^{\intercal}"'] & \boldsymbol{f}_{0}^{\Phi} \\
& \boldsymbol{f}_{J-1}^{\Sigma} & \boldsymbol{f}_{J-2}^{\Sigma} & \boldsymbol{f}_{J-3}^{\Sigma} & & & \boldsymbol{f}_{0}^{\Sigma}
\end{tikzcd}$$

FIGURE 1. Visualization of the fast samplet transform.

To implement the fast samplet transform, we recursively apply the refinement relation (4) to the point evaluations

$$(f, [\boldsymbol{\Phi}_j^{\nu}, \boldsymbol{\Sigma}_j^{\nu}])_{\Omega} = (f, \boldsymbol{\Phi}_{j+1}^{\nu}[\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}])_{\Omega} = (f, \boldsymbol{\Phi}_{j+1}^{\nu})_{\Omega}[\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]. \tag{14}$$

On the finest level, the entries of the vector $(f, \boldsymbol{\Phi}_J^{\nu})_{\Omega}$ are exactly those of $\boldsymbol{f}^{\Delta}$. Recursively applying Equation (14) therefore yields all the coefficients $(f, \boldsymbol{\Sigma}_j^{\nu})_{\Omega}$, including $(f, \boldsymbol{\Phi}_0^{X})_{\Omega}$, required for the representation of f in the samplet basis, see Figure 1 for a visualization of the resulting fish bone scheme. The complete procedure is formulated in Algorithm 1.

Algorithm 1: Fast samplet transform

```
Data: Data $\boldsymbol{f}^{\Delta}$, cluster tree $\mathcal{T}$ and transformations $[\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]$.
Result: Coefficients $\boldsymbol{f}^{\Sigma}$ stored as $[(f, \boldsymbol{\Phi}_0^{X})_{\Omega}]^{\intercal}$ and $[(f, \boldsymbol{\Sigma}_j^{\nu})_{\Omega}]^{\intercal}$.
begin
    store $[(f, \boldsymbol{\Phi}_0^{X})_{\Omega}]^{\intercal}$ := transformForCluster($X$)
end
```

Function transformForCluster(ν)

```
begin
    if $\nu = \{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{|\nu|}}\}$ is a leaf of $\mathcal{T}$ then
        set $\boldsymbol{f}_{j+1}^{\nu} := [f_{i_k}^{\Delta}]_{k=1}^{|\nu|}$
    else
        for all sons $\nu'$ of $\nu$ do
            execute transformForCluster($\nu'$)
            append the result to $\boldsymbol{f}_{j+1}^{\nu}$
        end
    end
    set $[(f, \boldsymbol{\Sigma}_j^{\nu})_{\Omega}]^{\intercal} := (\boldsymbol{Q}_{j,\Sigma}^{\nu})^{\intercal}\boldsymbol{f}_{j+1}^{\nu}$
    return $(\boldsymbol{Q}_{j,\Phi}^{\nu})^{\intercal}\boldsymbol{f}_{j+1}^{\nu}$
end
```

The inverse transformation is obtained by reversing the steps of the fast samplet transform: For each cluster, we compute

$$(f, \boldsymbol{\Phi}_{j+1}^{\nu})_\Omega = (f, [\boldsymbol{\Phi}_j^{\nu}, \boldsymbol{\Sigma}_j^{\nu}])_\Omega [\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]^\intercal$$

to either obtain the coefficients of the son clusters' scaling distributions or, for leaf clusters, the coefficients $\boldsymbol{f}^{\Delta}$. The procedure is summarized in Algorithm 2.

```
Algorithm 2: Inverse samplet transform
  Data: Coefficients $\boldsymbol{f}^{\Sigma}$, cluster tree $\mathcal{T}$ and transformations $[\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]$.
  Result: Coefficients $\boldsymbol{f}^{\Delta}$ stored as $[(f, \boldsymbol{\Phi}_j^{\nu})_\Omega]^\intercal$.
  begin
    inverseTransformForCluster($X$, $[(f, \boldsymbol{\Phi}_0^{X})_\Omega]^\intercal$)
  end
```

```
Function inverseTransformForCluster($\nu$, $[(f, \boldsymbol{\Phi}_j^{\nu})_\Omega]^\intercal$)
  begin
    $[(f, \boldsymbol{\Phi}_{j+1}^{\nu})_\Omega]^\intercal := [\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}] \begin{bmatrix} [(f, \boldsymbol{\Phi}_j^{\nu})_\Omega]^\intercal \\ [(f, \boldsymbol{\Sigma}_j^{\nu})_\Omega]^\intercal \end{bmatrix}$
    if $\nu = \{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_{|\nu|}}\}$ is a leaf of $\mathcal{T}$ then
      set $[f_{i_k}^{\Delta}]_{k=1}^{|\nu|} := [(f, \boldsymbol{\Phi}_{j_\nu+1}^{\nu})_\Omega]^\intercal$
    else
      for all sons $\nu'$ of $\nu$ do
        assign the part of $[(f, \boldsymbol{\Phi}_{j+1}^{\nu})_\Omega]^\intercal$ belonging to $\nu'$ to $[(f, \boldsymbol{\Phi}_{j'}^{\nu'})_\Omega]^\intercal$
        execute inverseTransformForCluster($\nu'$, $[(f, \boldsymbol{\Phi}_{j'}^{\nu'})_\Omega]^\intercal$)
      end
    end
  end
```

The fast samplet transform and its inverse can be performed in linear cost. This result is well known in case of wavelets and was crucial for their rapid development.

Theorem 4.1. *The runtime of the fast samplet transform and the inverse samplet transform are $\mathcal{O}(N)$, each.*

Proof. As the samplet construction follows the construction of Tausch and White, we refer to [39] for the details of the proof. □

5. Compression of kernel matrices

5.1. **Kernel matrices.** The second application of samplets we consider is the compression of matrices arising from positive (semi-) definite kernels, as they emerge in scattered data approximation kernel based learning or Gaussian process regression, see for example [27, 36, 42, 44] and the references therein. We start by recalling the concept of a positive kernel.

Definition 5.1. *A symmetric kernel $\mathcal{K}\colon \Omega \times \Omega \to \mathbb{R}$ is called positive (semi-)definite on $\Omega \subset \mathbb{R}^d$, iff $[\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j=1}^N$ is a symmetric and positive (semi-)definite matrix for all $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \Omega$ and all $N \in \mathbb{N}$.*

Given the set of points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, many applications require the assembly and the inversion of the *kernel matrix*

$$\boldsymbol{K} := [\mathcal{K}(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j=1}^N \in \mathbb{R}^{N \times N}$$

or an appropriately regularized version $\boldsymbol{K}+\rho\boldsymbol{I},\ \rho>0$, thereof. In case that N is a large number, already the assembly and storage of $\boldsymbol{K}$ can easily become prohibitive. For the solution of an associated linear system, the situation is even worse. Fortunately, the kernel matrix can be compressed by employing samplets. To this end, the evaluation of the kernel function at the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ will be denoted by

$$(\mathcal{K},\delta_{\boldsymbol{x}_i}\otimes\delta_{\boldsymbol{x}_j})_{\Omega\times\Omega}:=\mathcal{K}(\boldsymbol{x}_i,\boldsymbol{x}_j).$$

Hence, in view of $V=\{\delta_{\boldsymbol{x}_1},\ldots,\delta_{\boldsymbol{x}_N}\}$, we may write the kernel matrix as

$$\boldsymbol{K}=\big[(\mathcal{K},\delta_{\boldsymbol{x}_i}\otimes\delta_{\boldsymbol{x}_j})_{\Omega\times\Omega}\big]_{i,j=1}^N.$$

5.2. **Asymptotically smooth kernels.** The essential ingredient for the samplet compression of kernel matrices is the *asymptotical smoothness* property of the kernel $\mathcal{K}$, that is

$$\frac{\partial^{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}}{\partial\boldsymbol{x}^{\boldsymbol{\alpha}}\partial\boldsymbol{y}^{\boldsymbol{\beta}}}\mathcal{K}(\boldsymbol{x},\boldsymbol{y})\le c_{\mathcal{K}}\frac{(|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|)!}{r^{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}\|\boldsymbol{x}-\boldsymbol{y}\|_2^{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}},\quad c_{\mathcal{K}},r>0. \tag{15}$$

Remark 5.2. *A particular class of positive definite kernels which are asymptotically smooth are the* Matérn kernels *given by*

$$k_\nu(r):=\frac{2^{1-\nu}}{\Gamma(\nu)}\bigg(\frac{\sqrt{2\nu}r}{\ell}\bigg)^\nu K_\nu\bigg(\frac{\sqrt{2\nu}r}{\ell}\bigg),\quad r\ge0,\ \ell>0.$$

Herein, K_ν is the modified Bessel function of the second kind of order ν and Γ is the gamma function. The parameter ν steers for the smoothness of the kernel function. In particular, we have

$$k_{1/2}(r)=\exp\bigg(-\frac{r}{\ell}\bigg),\quad k_\infty(r)=\exp\bigg(-\frac{r^2}{2\ell^2}\bigg).$$

A positive definite kernel in the sense of Definition 5.1 is obtained by $\mathcal{K}(\boldsymbol{x},\boldsymbol{y}):=k_\nu(\|\boldsymbol{x}-\boldsymbol{y}\|_2)$.

Based on the asymptotical smoothness property (15), we obtain the following result, which is the basis for the matrix compression introduced thereafter.

Lemma 5.3. *Consider two samplets $\sigma_{j,k}$ and $\sigma_{j',k'}$, exhibiting $q+1$ vanishing moments with supporting clusters ν and ν', respectively. Assume that* $\operatorname{dist}(\nu,\nu')>0$. *Then, for kernels satisfying* (15), *it holds that*

$$(\mathcal{K},\sigma_{j,k}\otimes\sigma_{j',k'})_{\Omega\times\Omega}\le c_{\mathcal{K}}\frac{\operatorname{diam}(\nu)^{q+1}\operatorname{diam}(\nu')^{q+1}}{(dr\operatorname{dist}(\nu_{j,k},\nu_{j',k'}))^{2(q+1)}}\|\boldsymbol{\omega}_{j,k}\|_1\|\boldsymbol{\omega}_{j',k'}\|_1. \tag{16}$$

Proof. Let $\boldsymbol{x}_0\in\nu$ and $\boldsymbol{y}_0\in\nu'$. A Taylor expansion of the kernel with respect to $\boldsymbol{x}$ yields

$$\mathcal{K}(\boldsymbol{x},\boldsymbol{y})=\sum_{|\boldsymbol{\alpha}|\le q}\frac{\partial^{|\boldsymbol{\alpha}|}}{\partial\boldsymbol{x}^{\boldsymbol{\alpha}}\mathcal{K}(\boldsymbol{x}_0,\boldsymbol{y})}\frac{(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}+R_{\boldsymbol{x}_0}(\boldsymbol{x},\boldsymbol{y}),$$

where the remainder $R_{\boldsymbol{x}_0}(\boldsymbol{x},\boldsymbol{y})$ is given by

$$R_{\boldsymbol{x}_0}(\boldsymbol{x},\boldsymbol{y})=(q+1)\sum_{|\boldsymbol{\alpha}|=q+1}\frac{(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\int_0^1\frac{\partial^{q+1}}{\partial\boldsymbol{x}^{\boldsymbol{\alpha}}}\mathcal{K}\big(\boldsymbol{x}_0+s(\boldsymbol{x}-\boldsymbol{x}_0),\boldsymbol{y}\big)(1-s)^q\,\mathrm{d}s.$$

Next, we expand the remainder $R_{\boldsymbol{x}_0}(\boldsymbol{x},\boldsymbol{y})$ with respect to $\boldsymbol{y}$ and derive

$$\begin{aligned}R_{\boldsymbol{x}_0}(\boldsymbol{x},\boldsymbol{y})&=(q+1)\sum_{|\boldsymbol{\alpha}|=q+1}\frac{(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\sum_{|\boldsymbol{\beta}|\le q}\frac{(\boldsymbol{y}-\boldsymbol{y}_0)^{\boldsymbol{\beta}}}{\boldsymbol{\beta}!}\\&\quad\times\int_0^1\frac{\partial^{q+1}}{\partial\boldsymbol{x}^{\boldsymbol{\alpha}}}\frac{\partial^{|\boldsymbol{\beta}|}}{\partial\boldsymbol{y}^{\boldsymbol{\beta}}}\mathcal{K}\big(\boldsymbol{x}_0+s(\boldsymbol{x}-\boldsymbol{x}_0),\boldsymbol{y}_0\big)(1-s)^q\,\mathrm{d}s+R_{\boldsymbol{x}_0,\boldsymbol{y}_0}(\boldsymbol{x},\boldsymbol{y}).\end{aligned}$$

Here, the remainder $R_{\boldsymbol{x}_0,\boldsymbol{y}_0}(\boldsymbol{x},\boldsymbol{y})$ is given by

$$R_{\boldsymbol{x}_0,\boldsymbol{y}_0}(\boldsymbol{x},\boldsymbol{y}) = (q+1)^2 \sum_{|\boldsymbol{\alpha}|,|\boldsymbol{\beta}|=q+1} \frac{(\boldsymbol{x}-\boldsymbol{x}_0)^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\frac{(\boldsymbol{y}-\boldsymbol{y}_0)^{\boldsymbol{\beta}}}{\boldsymbol{\beta}!}$$
$$\times \int_0^1\int_0^1 \frac{\partial^{2(q+1)}}{\partial\boldsymbol{x}^{\boldsymbol{\alpha}}\partial\boldsymbol{y}^{\boldsymbol{\beta}}}\mathcal{K}\big(\boldsymbol{x}_0+s(\boldsymbol{x}-\boldsymbol{x}_0),\boldsymbol{y}_0+t(\boldsymbol{y}-\boldsymbol{y}_0)\big)(1-s)^q(1-t)^q\,\mathrm{d}t\,\mathrm{d}s.$$

We thus arrive at the decomposition

$$\mathcal{K}(\boldsymbol{x},\boldsymbol{y}) = p_{\boldsymbol{y}}(\boldsymbol{x}) + p_{\boldsymbol{x}}(\boldsymbol{y}) + R_{\boldsymbol{x}_0,\boldsymbol{y}_0}(\boldsymbol{x},\boldsymbol{y}),$$

where $p_{\boldsymbol{y}}(\boldsymbol{x})$ is a polynomial of degree q in $\boldsymbol{x}$, with coefficients depending on $\boldsymbol{y}$, while $p_{\boldsymbol{x}}(\boldsymbol{y})$ is a polynomial of degree q in $\boldsymbol{y}$, with coefficients depending on $\boldsymbol{x}$. Due to the vanishing moments, we obtain

$$(\mathcal{K},\sigma_{j,k}\otimes\sigma_{j',k'})_{\Omega\times\Omega} = (R_{\boldsymbol{x}_0,\boldsymbol{y}_0},\sigma_{j,k}\otimes\sigma_{j',k'})_{\Omega\times\Omega}.$$

In view of (15), we thus find

$$|(\mathcal{K},\sigma_{j,k}\otimes\sigma_{j',k'})_{\Omega\times\Omega}| = |(R_{\boldsymbol{x}_0,\boldsymbol{y}_0},\sigma_{j,k}\otimes\sigma_{j',k'})_{\Omega\times\Omega}|$$
$$\leq c_{\mathcal{K}}\Bigg(\sum_{|\boldsymbol{\alpha}|,|\boldsymbol{\beta}|=q+1}\frac{(|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|)!}{\boldsymbol{\alpha}!\boldsymbol{\beta}!}\Bigg)\frac{(\|\cdot-\boldsymbol{x}_0\|_2^{q+1},|\sigma_{j,k}|)_\Omega(\|\cdot-\boldsymbol{y}_0\|_2^{q+1},|\sigma_{j',k'}|)_\Omega}{r^{2(q+1)}\operatorname{dist}(\nu,\nu')^{2(q+1)}}.$$

Now, we have by means of multinomial coefficients that

$$(|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|)! = \binom{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}{|\boldsymbol{\beta}|}\binom{|\boldsymbol{\alpha}|}{\boldsymbol{\alpha}}\binom{|\boldsymbol{\beta}|}{\boldsymbol{\beta}}\boldsymbol{\alpha}!\boldsymbol{\beta}!,$$

which in turn implies that

$$\sum_{|\boldsymbol{\alpha}|,|\boldsymbol{\beta}|=q+1}\frac{(|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|)!}{\boldsymbol{\alpha}!\boldsymbol{\beta}!} = \binom{2(q+1)}{q+1}\sum_{|\boldsymbol{\alpha}|,|\boldsymbol{\beta}|=q+1}\binom{|\boldsymbol{\alpha}|}{\boldsymbol{\alpha}}\binom{|\boldsymbol{\beta}|}{\boldsymbol{\beta}}$$
$$= \binom{2(q+1)}{q+1}d^{2(q+1)} \leq d^{2(q+1)}2^{2(q+1)}.$$

Moreover, we use

$$(\|\cdot-\boldsymbol{x}_0\|_2^{q+1},|\sigma_{j,k}|)_\Omega \leq \bigg(\frac{\operatorname{diam}(\nu)}{2}\bigg)^{q+1}\|\boldsymbol{\omega}_{j,k}\|_1,$$

and likewise

$$(\|\cdot-\boldsymbol{y}_0\|_2^{q+1},|\sigma_{j',k'}|)_\Omega \leq \bigg(\frac{\operatorname{diam}(\nu')}{2}\bigg)^{q+1}\|\boldsymbol{\omega}_{j',k'}\|_1.$$

Combining all the estimates, we arrive at the desired result (16). □

5.3. **Matrix compression.** Lemma 5.3 immediately suggests a compression strategy for kernel matrices in samplet representation. This compression differs from the wavelet matrix compression introduced in [11], since we do not exploit the decay of the samplet coefficients with respect to the level in case of smooth data. This enables us to also cover the case of non-quasi-uniform data. Consequently, we use on all levels the same accuracy, which is similar to the setting in [4].

Theorem 5.4. *Set all coefficients of the kernel matrix*

$$\boldsymbol{K}^\Sigma := \big[(\mathcal{K},\sigma_{j,k}\otimes\sigma_{j',k'})_{\Omega\times\Omega}\big]_{j,j',k,k'}$$

to zero which satisfy the admissibility condition

$$\operatorname{dist}(\nu,\nu') \geq \eta\max\{\operatorname{diam}(\nu),\operatorname{diam}(\nu')\},\quad \eta>0, \tag{17}$$

where ν is the cluster supporting $\sigma_{j,k}$ and ν' is the cluster supporting $\sigma_{j',k'}$, respectively. Then, it holds

$$\big\|\boldsymbol{K}^\Sigma - \boldsymbol{K}^\Sigma_\varepsilon\big\|_F \leq c_{\mathcal{K}}c_{\mathrm{sum}}(\eta dr)^{-2(q+1)}m_qN\sqrt{\log(N)}.$$

for some constant $c_{\text{sum}} > 0$*, where* m_q *is given by* (5)*.*

Proof. Fix the levels j and j'. In view (16), we can estimate any coefficient which satisfies (17) by

$$|(\mathcal{K}, \sigma_{j,k} \otimes \sigma_{j',k'})_{\Omega\times\Omega}| \le c_{\mathcal{K}} \left(\frac{\min\{\operatorname{diam}(\nu), \operatorname{diam}(\nu')\}}{\max\{\operatorname{diam}(\nu), \operatorname{diam}(\nu')\}} \right)^{q+1} (\eta d r)^{-2(q+1)} \|\boldsymbol{\omega}_{j,k}\|_1 \|\boldsymbol{\omega}_{j',k'}\|_1.$$

If we next set

$$\theta_{j,j'} := \max_{\nu\in\mathcal{T}_j, \nu'\in\mathcal{T}_{j'}} \left\{ \frac{\min\{\operatorname{diam}(\nu), \operatorname{diam}(\nu')\}}{\max\{\operatorname{diam}(\nu), \operatorname{diam}(\nu')\}} \right\},$$

then we obtain

$$|(\mathcal{K}, \sigma_{j,k} \otimes \sigma_{j',k'})_{\Omega\times\Omega}| \le c_{\mathcal{K}} \theta_{j,j'}^{q+1} (\eta d r)^{-2(q+1)} \|\boldsymbol{\omega}_{j,k}\|_1 \|\boldsymbol{\omega}_{j',k'}\|_1$$

for all coefficients such that (17) holds. In view of (12) and the fact that there are at most m_q samplets per cluster, we arrive at

$$\sum_{k,k'} \|\boldsymbol{\omega}_{j,k}\|_1^2 \|\boldsymbol{\omega}_{j',k'}\|_1^2 \le \sum_{k,k'} |\nu| \cdot |\nu'| = m_q^2 N^2.$$

Thus, for a fixed level-level block, we arrive at the estimate

$$\big\|\boldsymbol{K}_{j,j'}^{\Sigma} - \boldsymbol{K}_{\varepsilon,j,j'}^{\Sigma}\big\|_F^2 \le \sum_{\substack{k,k':\ \operatorname{dist}(\nu,\nu') \\ \ge \eta \max\{\operatorname{diam}(\nu), \operatorname{diam}(\nu')\}}} |(\mathcal{K}, \sigma_{j,k} \otimes \sigma_{j',k'})_{\Omega\times\Omega}|^2 \le c_{\mathcal{K}}^2 \theta_{j,j'}^{2(q+1)} (\eta d r)^{-4(q+1)} m_q^2 N^2.$$

Finally, summation over all levels yields

$$\begin{aligned} \big\|\boldsymbol{K}^{\Sigma} - \boldsymbol{K}_{\varepsilon}^{\Sigma}\big\|_F^2 &= \sum_{j,j'} \big\|\boldsymbol{K}_{j,j'}^{\Sigma} - \boldsymbol{K}_{\varepsilon,j,j'}^{\Sigma}\big\|_F^2 \le c_{\mathcal{K}}^2 (\eta d r)^{-4(q+1)} m_q^2 N^2 \sum_{j,j'} \theta_{j,j'}^{2(q+1)} \\ &\le c_{\mathcal{K}}^2 c_{\text{sum}} (\eta d r)^{-4(q+1)} m_q^2 N^2 \log N, \end{aligned}$$

which is the desired claim. □

Corollary 5.5. *In case of quasi-uniform points* $\boldsymbol{x}_i \in X$*, we have* $\big\|\boldsymbol{K}^{\Sigma}\big\|_F \sim N$*. Thus, we immediately obtain*

$$\frac{\big\|\boldsymbol{K}^{\Sigma} - \boldsymbol{K}_{\varepsilon}^{\Sigma}\big\|_F}{\big\|\boldsymbol{K}^{\Sigma}\big\|_F} \le c_{\mathcal{K}} \sqrt{c_{\text{sum}}} (\eta d r)^{-2(q+1)} m_q \sqrt{\log N},$$

where the compressed matrix has $\mathcal{O}(m_q^2 N \log N)$ *remaining coefficients.*

Proof. We fix j, j' and assume $j \ge j'$. In case of quasi-uniform points, it holds $\operatorname{diam}(v) \sim 2^{-j_\nu/d}$. Hence, for the cluster $\nu_{j',k'}$, there exist only $\mathcal{O}([2^{j-j'}]^d)$ clusters $\nu_{j,k}$ from level j, which do not satisfy the admissibility condition (17). Since each cluster contains at most m_q samplets, we arrive at

$$\sum_{j=0}^{J} \sum_{j'\le j} m_q^2 (2^{j'} 2^{(j-j')})^d = m_q^2 \sum_{j=0}^{J} j 2^{jd} \sim m_q^2 N \log N,$$

which implies the assertion. □

Remark 5.6. *The admissibility condition* (17) *is a multilevel version of the admissibility condition used by hierarchical matrices, see e.g. [22].*

5.4. **Compressed matrix assembly.** For a given pair of clusters, we can now determine whether the corresponding entries need to be calculated. As there are $\mathcal{O}(N)$ clusters, naively checking the cut-off criterion for all pairs would still take $\mathcal{O}(N^2)$ operations, however. Hence, we require smarter means to determine the non-negligible cluster pairs. For this purpose, we first state the transferability of the cut-off criterion to son clusters, compare [11] for a proof.

Lemma 5.7. *Let ν and ν' be clusters satisfying the admissibility condition* (17). *Then, for the son clusters ν_{son} of ν and ν'_{son} of ν', we have*

$$\begin{aligned}\operatorname{dist}(\nu,\nu'_{\mathrm{son}}) &\geq \eta\max\{\operatorname{diam}(\nu),\operatorname{diam}(\nu'_{\mathrm{son}})\},\\ \operatorname{dist}(\nu_{\mathrm{son}},\nu') &\geq \eta\max\{\operatorname{diam}(\nu_{\mathrm{son}}),\operatorname{diam}(\nu')\},\\ \operatorname{dist}(\nu_{\mathrm{son}},\nu'_{\mathrm{son}}) &\geq \eta\max\{\operatorname{diam}(\nu_{\mathrm{son}}),\operatorname{diam}(\nu'_{\mathrm{son}})\}.\end{aligned}$$

The lemma tells us that we may omit cluster pairs whose father clusters already satisfy the admissibility condition. This will be essential for the assembly of the compressed matrix. The computation of the compressed kernel matrix can be sped up further by using $\mathcal{H}^2$-matrix techniques, see [19, 23]. This idea was used earlier in [1, 24, 28] in case of Tausch-White wavelets. $\mathcal{H}^2$-matrices approximate the kernel interaction for sufficiently distant clusters ν and ν' in the sense of the admissibility condition (17) by means of a polynomial interpolant, see [5]. More precisely, given a suitable set of interpolation points $\{\boldsymbol{\xi}_t^\nu\}_t$ for each cluster ν with associated Lagrange polynomials $\{\mathcal{L}_t^\nu(\boldsymbol{x})\}_t$, we introduce the interpolation operator

$$\mathcal{I}^{\nu,\nu'}[\mathcal{K}](\boldsymbol{x},\boldsymbol{y}) = \sum_{s,t}\mathcal{K}(\boldsymbol{\xi}_s^\nu,\boldsymbol{\xi}_t^{\nu'})\mathcal{L}_s^\nu(\boldsymbol{x})\mathcal{L}_t^{\nu'}(\boldsymbol{y})$$

and approximate an admissible matrix block via

$$\begin{aligned}\boldsymbol{K}^\Delta_{\nu,\nu'} &= [(\mathcal{K},\delta_{\boldsymbol{x}}\otimes\delta_{\boldsymbol{y}})_{\Omega\times\Omega}]_{\boldsymbol{x}\in\nu,\boldsymbol{y}\in\nu'}\\ &\approx \sum_{s,t}\mathcal{K}(\boldsymbol{\xi}_s^\nu,\boldsymbol{\xi}_t^{\nu'})[(\mathcal{L}_s^\nu,\delta_{\boldsymbol{x}})_\Omega]_{\boldsymbol{x}\in\nu}[(\mathcal{L}_t^{\nu'},\delta_{\boldsymbol{y}})_\Omega]_{\boldsymbol{y}\in\nu'} =: \boldsymbol{V}_\Delta^\nu\boldsymbol{S}^{\nu,\nu'}(\boldsymbol{V}_\Delta^{\nu'})^\intercal.\end{aligned}\tag{18}$$

Herein, the *cluster bases* are given according to

$$\boldsymbol{V}_\Delta^\nu := [(\mathcal{L}_s^\nu,\delta_{\boldsymbol{x}})_\Omega]_{\boldsymbol{x}\in\nu},\quad \boldsymbol{V}_\Delta^{\nu'} := [(\mathcal{L}_t^{\nu'},\delta_{\boldsymbol{y}})_\Omega]_{\boldsymbol{y}\in\nu'},\tag{19}$$

while the *coupling matrix* is given by $\boldsymbol{S}^{\nu,\nu'} := [\mathcal{K}(\boldsymbol{\xi}_s^\nu,\boldsymbol{\xi}_t^{\nu'})]_{s,t}$.

Remark 5.8. *Different from the $\mathcal{H}^2$-matrix setting, we shall consider the expansion* (18) *also when the clusters ν and ν' are located on different levels of the cluster tree.*

Directly transforming the cluster bases into their corresponding samplet representation results in a log-linear cost. This can be avoided by the use of nested cluster bases, as they have been introduced for $\mathcal{H}^2$-matrices. For the sake of simplicity, we assume from now on that a fixed polynomial degree is p used for the kernel interpolation at all different cluster combinations. Therefore, the Lagrange polynomials of a father cluster can exactly be represented by those of the son clusters. Introducing the *transfer matrices* $\boldsymbol{T}^{\nu_{\mathrm{son}}} := [\mathcal{L}_s^\nu(\boldsymbol{\xi}_t^{\nu_{\mathrm{son}}})]_{s,t}$, it holds

$$\mathcal{L}_s^\nu(\boldsymbol{x}) = \sum_t \boldsymbol{T}_{s,t}^{\nu_{\mathrm{son}}}\mathcal{L}_t^{\nu_{\mathrm{son}}}(\boldsymbol{x}),\quad \boldsymbol{x}\in B_{\nu_{\mathrm{son}}}.$$

Exploiting this relation in the construction of the cluster bases (19) leads to the recursive refinement relation

$$\boldsymbol{V}_\Delta^\nu = \begin{bmatrix}\boldsymbol{V}_\Delta^{\nu_{\mathrm{son}_1}}\boldsymbol{T}^{\nu_{\mathrm{son}_1}}\\ \boldsymbol{V}_\Delta^{\nu_{\mathrm{son}_2}}\boldsymbol{T}^{\nu_{\mathrm{son}_2}}\end{bmatrix}.$$

Combining this refinement relation with the recursive nature of the samplet basis, results in the variant of the fast samplet transform summarized in Algorithm 3.

Algorithm 3: Recursive computation of the multiscale cluster basis

```
Data: Cluster tree $\mathcal{T}$, transformations $[\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]$, nested cluster bases $\boldsymbol{V}_{\Delta}^{\nu}$ for leaf clusters and
      transformation matrices $\boldsymbol{T}^{\nu_{\mathrm{son}_1}}$, $\boldsymbol{T}^{\nu_{\mathrm{son}_2}}$ for non-leaf clusters.
Result: Multiscale cluster basis matrices $\boldsymbol{V}_{\Phi}^{\nu}$, $\boldsymbol{V}_{\Sigma}^{\nu}$ for all clusters $\nu \in \mathcal{T}$.
begin
    computeMultiscaleClusterBasis($X$);
end
```

Function computeMultiscaleClusterBasis(ν)

```
begin
    if $\nu$ is a leaf cluster then
        store $\begin{bmatrix}\boldsymbol{V}_{\Phi}^{\nu}\\ \boldsymbol{V}_{\Sigma}^{\nu}\end{bmatrix} := [\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]^{\intercal}\boldsymbol{V}_{\Delta}^{\nu}$
    else
        for all sons $\nu'$ of $\nu$ do
            computeMultiscaleClusterBasis($\nu'$)
        end
        store $\begin{bmatrix}\boldsymbol{V}_{\Phi}^{\nu}\\ \boldsymbol{V}_{\Sigma}^{\nu}\end{bmatrix} := [\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]^{\intercal}\begin{bmatrix}\boldsymbol{V}_{\Phi}^{\nu_{\mathrm{son}_1}}\boldsymbol{T}^{\nu_{\mathrm{son}_1}}\\ \boldsymbol{V}_{\Phi}^{\nu_{\mathrm{son}_2}}\boldsymbol{T}^{\nu_{\mathrm{son}_2}}\end{bmatrix}$
    end
end
```

Having the multiscale cluster bases at our disposal, the next step is the actual assembly of the compressed kernel matrix. The computation of the required matrix blocks is exclusively based on the two refinement relations

$$\begin{bmatrix}\boldsymbol{K}_{\nu,\nu'}^{\Phi,\Phi} & \boldsymbol{K}_{\nu,\nu'}^{\Phi,\Sigma}\\ \boldsymbol{K}_{\nu,\nu'}^{\Sigma,\Phi} & \boldsymbol{K}_{\nu,\nu'}^{\Sigma,\Sigma}\end{bmatrix} = \begin{bmatrix}\boldsymbol{K}_{\nu,\nu'_{\mathrm{son}_1}}^{\Phi,\Phi} & \boldsymbol{K}_{\nu,\nu'_{\mathrm{son}_2}}^{\Phi,\Phi}\\ \boldsymbol{K}_{\nu,\nu'_{\mathrm{son}_1}}^{\Sigma,\Phi} & \boldsymbol{K}_{\nu,\nu'_{\mathrm{son}_2}}^{\Sigma,\Phi}\end{bmatrix}[\boldsymbol{Q}_{j,\Phi}^{\nu'}, \boldsymbol{Q}_{j,\Sigma}^{\nu'}]$$

and

$$\begin{bmatrix}\boldsymbol{K}_{\nu,\nu'}^{\Phi,\Phi} & \boldsymbol{K}_{\nu,\nu'}^{\Phi,\Sigma}\\ \boldsymbol{K}_{\nu,\nu'}^{\Sigma,\Phi} & \boldsymbol{K}_{\nu,\nu'}^{\Sigma,\Sigma}\end{bmatrix} = [\boldsymbol{Q}_{j,\Phi}^{\nu}, \boldsymbol{Q}_{j,\Sigma}^{\nu}]^{\intercal}\begin{bmatrix}\boldsymbol{K}_{\nu_{\mathrm{son}_1},\nu'}^{\Phi,\Phi} & \boldsymbol{K}_{\nu_{\mathrm{son}_1},\nu'}^{\Phi,\Phi}\\ \boldsymbol{K}_{\nu_{\mathrm{son}_2},\nu'}^{\Sigma,\Phi} & \boldsymbol{K}_{\nu_{\mathrm{son}_2},\nu'}^{\Sigma,\Phi}\end{bmatrix},$$

where we set

$$\begin{bmatrix}\boldsymbol{K}_{\nu,\nu'}^{\Phi,\Phi} & \boldsymbol{K}_{\nu,\nu'}^{\Phi,\Sigma}\\ \boldsymbol{K}_{\nu,\nu'}^{\Sigma,\Phi} & \boldsymbol{K}_{\nu,\nu'}^{\Sigma,\Sigma}\end{bmatrix} := \begin{bmatrix}(\mathcal{K}, \boldsymbol{\Phi}^{\nu}\otimes\boldsymbol{\Phi}^{\nu'})_{\Omega\times\Omega} & (\mathcal{K}, \boldsymbol{\Phi}^{\nu}\otimes\boldsymbol{\Sigma}^{\nu'})_{\Omega\times\Omega}\\ (\mathcal{K}, \boldsymbol{\Sigma}^{\nu}\otimes\boldsymbol{\Phi}^{\nu'})_{\Omega\times\Omega} & (\mathcal{K}, \boldsymbol{\Sigma}^{\nu}\otimes\boldsymbol{\Sigma}^{\nu'})_{\Omega\times\Omega}\end{bmatrix}.$$

Based on these relations, we introduce the function `recursivelyDetermineBlock`, which is the key ingredient for the computation of the compressed kernel matrix. Note that this function never requires the formation of the actual $\mathcal{H}^2$-matrix, as it only embeds the multilevel interpolation procedure to rapidly evaluate admissible blocks. Especially, the evaluation of the coupling matrices can be performed on the fly, significantly reducing the memory requirements of the method.

Next, to assemble the compressed kernel matrix in standard form, we have to traverse the tensor product $\mathcal{T}\otimes\mathcal{T}$ of the cluster tree. To this end, we employ two nested recursive calls of the cluster tree, which is traversed in a depth first search way. Algorithm 4 first computes the lower right matrix block and advances from bottom to top and from right to left. It relies on the two recursive functions `setupColumn` and `setupRow`. The purpose of the function `setupColumn` is to recursively traverse the column cluster tree, i.e. the cluster tree associated to the columns of the matrix. Before returning, each instance of `setupColumn` calls the function `setupRow`, which performs the actual assembly of the compressed matrix. For a given column cluster ν', the

Function recursivelyDetermineBlock(ν, ν')

```
Result: Approximation of the block $\begin{bmatrix} \boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu,\nu'} \\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'} \end{bmatrix}$.
begin
    if $(\nu,\nu')$ is admissible then
        return $\begin{bmatrix} \boldsymbol{V}^{\nu}_{\Phi} \\ \boldsymbol{V}^{\nu}_{\Sigma} \end{bmatrix} \boldsymbol{S}^{\nu,\nu'} \big[(\boldsymbol{V}^{\nu'}_{\Phi})^{\intercal}, (\boldsymbol{V}^{\nu'}_{\Sigma})^{\intercal}\big]$
    else if $\nu$ and $\nu'$ are leaf clusters then
        return $\big[\boldsymbol{Q}^{\nu}_{j,\Phi}, \boldsymbol{Q}^{\nu}_{j,\Sigma}\big]^{\intercal} \boldsymbol{K}^{\Delta}_{\nu,\nu'} \big[\boldsymbol{Q}^{\nu'}_{j,\Phi}, \boldsymbol{Q}^{\nu'}_{j,\Sigma}\big]$
    else if $\nu'$ is not a leaf cluster and $\nu$ is a leaf cluster then
        for all sons $\nu'_{\mathrm{son}}$ of $\nu'$ do
            $\begin{bmatrix} \boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'_{\mathrm{son}}} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu,\nu'_{\mathrm{son}}} \\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'_{\mathrm{son}}} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'_{\mathrm{son}}} \end{bmatrix}$ := recursivelyDetermineBlock$(\nu, \nu_{\mathrm{son}'})$
        end
        return $\begin{bmatrix} \boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'_{\mathrm{son}_1}} & \boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'_{\mathrm{son}_2}} \\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'_{\mathrm{son}_1}} & \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'_{\mathrm{son}_2}} \end{bmatrix} \big[\boldsymbol{Q}^{\nu'}_{j,\Phi}, \boldsymbol{Q}^{\nu'}_{j,\Sigma}\big]$
    else if $\nu$ is not a leaf cluster and $\nu'$ is a leaf cluster then
        for all sons $\nu_{\mathrm{son}}$ of $\nu$ do
            $\begin{bmatrix} \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}},\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu_{\mathrm{son}},\nu'} \\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\mathrm{son}},\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu_{\mathrm{son}},\nu'} \end{bmatrix}$ := recursivelyDetermineBlock$(\nu_{\mathrm{son}}, \nu')$
        end
        return $\big[\boldsymbol{Q}^{\nu}_{j,\Phi}, \boldsymbol{Q}^{\nu}_{j,\Sigma}\big]^{\intercal} \begin{bmatrix} \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}_1},\nu'} & \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}_1},\nu'} \\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\mathrm{son}_2},\nu'} & \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\mathrm{son}_2},\nu'} \end{bmatrix}$.
    else
        for all sons $\nu_{\mathrm{son}}$ of $\nu$ and all sons $\nu'_{\mathrm{son}}$ of $\nu'$ do
            $\begin{bmatrix} \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}},\nu'_{\mathrm{son}}} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu_{\mathrm{son}},\nu'_{\mathrm{son}}} \\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\mathrm{son}},\nu'_{\mathrm{son}}} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu_{\mathrm{son}},\nu'_{\mathrm{son}}} \end{bmatrix}$ := recursivelyDetermineBlock$(\nu_{\mathrm{son}}, \nu_{\mathrm{son}'})$
        end
        return $\big[\boldsymbol{Q}^{\nu}_{j,\Phi}, \boldsymbol{Q}^{\nu}_{j,\Sigma}\big]^{\intercal} \begin{bmatrix} \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}_1},\nu'_{\mathrm{son}_1}} & \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}_1},\nu'_{\mathrm{son}_2}} \\ \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}_2},\nu'_{\mathrm{son}_1}} & \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\mathrm{son}_2},\nu'_{\mathrm{son}_2}} \end{bmatrix} \big[\boldsymbol{Q}^{\nu'}_{j,\Phi}, \boldsymbol{Q}^{\nu'}_{j,\Sigma}\big]$
    end
end
```

Algorithm 4: Computation of the compressed kernel matrix

```
Data: Cluster tree $\mathcal{T}$, multiscale cluster bases $\boldsymbol{V}^{\nu}_{\Phi}$, $\boldsymbol{V}^{\nu}_{\Sigma}$ and transformations $[\boldsymbol{Q}^{\nu}_{j,\Phi}, \boldsymbol{Q}^{\nu}_{j,\Sigma}]$.
Result: Sparse matrix $\boldsymbol{K}^{\Sigma}_{\varepsilon}$
begin
    setupColumn$(X)$
    store the blocks the remaining blocks $\boldsymbol{K}^{\Sigma}_{\varepsilon,\nu,X}$ for $\nu \in \mathcal{T} \setminus \{X\}$ in $\boldsymbol{K}^{\Sigma}_{\varepsilon}$ (they have already been
      computed by earlier calls to recursivelyDetermineBlock)
end
```

function `setupRow` recursively traverses the row cluster tree, i.e. the cluster tree associated to the rows of the matrix, and assembles the corresponding column of the compressed matrix. The function reuses the already computed blocks to the right of the column under consideration and blocks at the bottom of the very same column.

Function setupColumn(ν')

```
begin
    for all sons $\nu'_{\text{son}}$ of $\nu'$ do
        setupColumn($\nu'_{\text{son}}$)
    end
    store $\boldsymbol{K}^{\Sigma}_{\varepsilon,X,\nu'}$ := setupRow($X,\nu'$) in $\boldsymbol{K}^{\Sigma}_{\varepsilon}$
end
```

Function setupRow(ν, ν')

```
begin
    if $\nu$ is not a leaf then
        for all sons $\nu_{\text{son}}$ of $\nu$ do
            if $\nu_{\text{son}}$ and $\nu'$ are not admissible then
                $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu_{\text{son}},\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu_{\text{son}},\nu'}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\text{son}},\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu_{\text{son}},\nu'}\end{bmatrix}$ := setupRow($\nu_{\text{son}},\nu'$)
            else
                $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu_{\text{son}},\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu_{\text{son}},\nu'}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\text{son}},\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu_{\text{son}},\nu'}\end{bmatrix}$ := recursivelyDetermineBlock($\nu_{\text{son}},\nu'$)
            end
        end
        $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu,\nu'}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'}\end{bmatrix} := [\boldsymbol{Q}^{\nu}_{\Phi},\boldsymbol{Q}^{\nu}_{\Sigma}]^{\intercal}\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu_{\text{son}_1},\nu'} & \boldsymbol{K}^{\Phi,\Phi}_{\nu_{\text{son}_1},\nu'}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\text{son}_2},\nu'} & \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\text{son}_2},\nu'}\end{bmatrix}$
    else
        if $\nu'$ is a leaf cluster then
            $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu_{\text{son}},\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu_{\text{son}},\nu'}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu_{\text{son}},\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu_{\text{son}},\nu'}\end{bmatrix}$ := recursivelyDetermineBlock($\nu_{\text{son}},\nu'$)
        else
            for all sons $\nu'_{\text{son}}$ of $\nu$' do
                if $\nu$ and $\nu'_{\text{son}}$ are not admissible then
                    load already computed block $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'_{\text{son}}} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu,\nu'_{\text{son}}}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'_{\text{son}}} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'_{\text{son}}}\end{bmatrix}$
                else
                    $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'_{\text{son}}} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu,\nu'_{\text{son}}}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'_{\text{son}}} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'_{\text{son}}}\end{bmatrix}$ := recursivelyDetermineBlock($\nu,\nu_{\text{son}'}$)
                end
            end
        end
        $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu,\nu'}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'}\end{bmatrix} := \begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'_{\text{son}_1}} & \boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'_{\text{son}_2}}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'_{\text{son}_1}} & \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'_{\text{son}_2}}\end{bmatrix}[\boldsymbol{Q}^{\nu'}_{\Phi},\boldsymbol{Q}^{\nu'}_{\Sigma}]$
    end
    store $\boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'}$ as part of $\boldsymbol{K}^{\Sigma}_{\varepsilon}$ return $\begin{bmatrix}\boldsymbol{K}^{\Phi,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Phi,\Sigma}_{\nu,\nu'}\\ \boldsymbol{K}^{\Sigma,\Phi}_{\nu,\nu'} & \boldsymbol{K}^{\Sigma,\Sigma}_{\nu,\nu'}\end{bmatrix}$
end
```

Remark 5.9. *Algorithm 4 has a cost of $\mathcal{O}(N\log N)$ and requires an additional storage of $\mathcal{O}(N\log N)$ if all stored blocks are directly released when they are not required anymore. We refer to [1] for all the details.*

6. Numerical results

6.1. **Data compression.** To demonstrate the efficacy of the samplet analysis, we compress data in one, two and three spatial dimensions. For each example, we use samplets with $q+1=3$ vanishing moments.

One dimension. We start with two one-dimensional examples. On the one hand, we consider the function

$$f(x) = \frac{3}{2}e^{-40|x-\frac{1}{4}|} + 2e^{-40|x|} - e^{-40|x+\frac{1}{2}|},$$

sampled at 8192 uniformly distributed random points on $[-1,1]$. On the other hand, we consider a sample path of the Brownian motion sampled at the same points. The coefficients of the samplet transformed data are thresholded with $10^{-i}\|\boldsymbol{f}^{\Sigma}\|_\infty$, $i=1,2,3$, respectively. The resulting compression ratios and the reconstructions can be found in Figure 2 and Figure 3, respectively. One readily infers that in both cases high compression rates are achieved at high accuracy. In case of the Brownian motion, the smoothing of the sample data can be realized by increasing the compression rate, corresponding to truncating more and more detail information. Due to the orthonormality of the samplet basis, this procedure amounts to a least squares fit of the data.

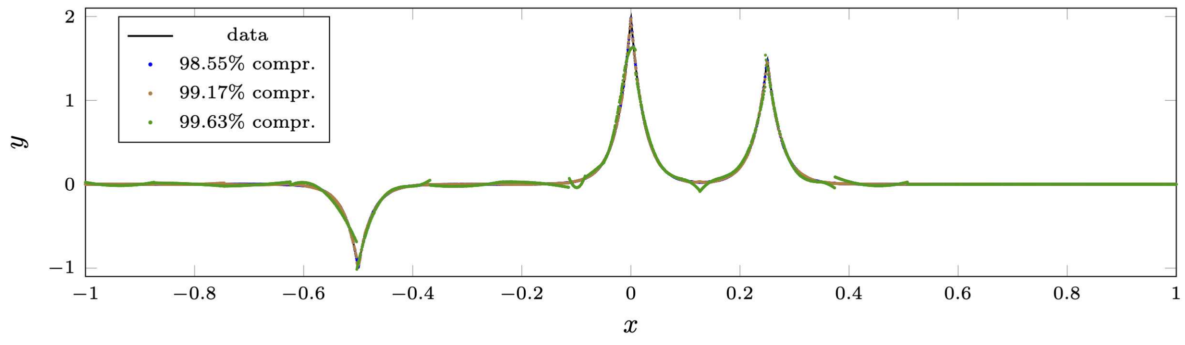


Figure 2. Sampled function approximated with different compression ratios.

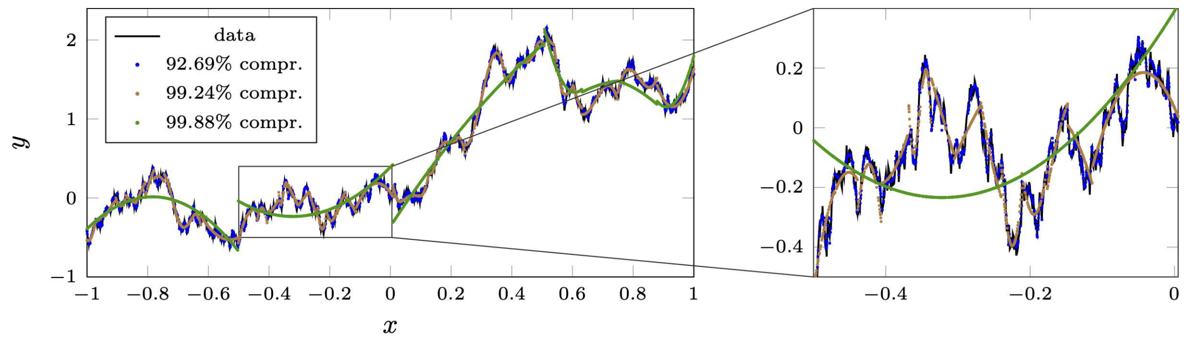


Figure 3. Sampled Brownian motion approximated with different compression ratios.

Two dimensions. As a second application for samplets, we consider image compression. We use a 2000×2000 pixel grayscale landscape image depicted in Figure 4. The coefficients of the samplet transformed image are thresholded with $10^{-i}\|\boldsymbol{f}^{\Sigma}\|_\infty$, $i=2,3,4$, respectively. The corresponding results and compression rates can be found on the left hand side of the figure. A visualization of the samplet coefficients in case of the respective low compression can be found on the right hand side of the figure. As can be seen, the samplets localize at the sharp features of the image.

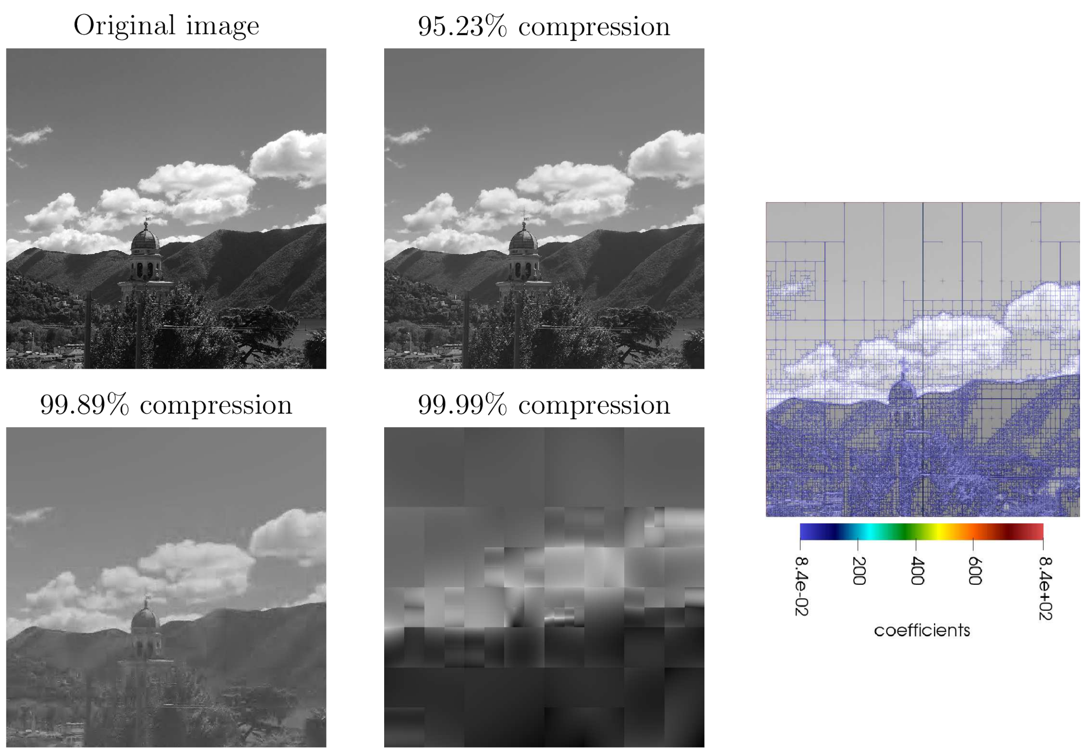


FIGURE 4. Different compression rates of the test image (left) and dominant samplet coefficients for the low compression (right).

Three dimensions. Finally, we show a data compression result in three dimensions. Here, the data are generated for a uniform subsample of a surface triangulation of the Stanford bunny. We consider data resulting from the evaluation of the function

$$f(\boldsymbol{x}) = e^{-20\|\boldsymbol{x}-\boldsymbol{p}_0\|_2} + e^{-20\|\boldsymbol{x}-\boldsymbol{p}_1\|_2},$$

where the points $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are located at the tips of the bunny's ears. The plot on the left hand side of Figure 5 visualizes the sample data, while the plot on the right hand side shows the dominant coefficients in case of a threshold parameter of $10^{-2}\|\boldsymbol{f}^\Sigma\|_\infty$. The samplets perfectly refine towards the points of interest $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$.

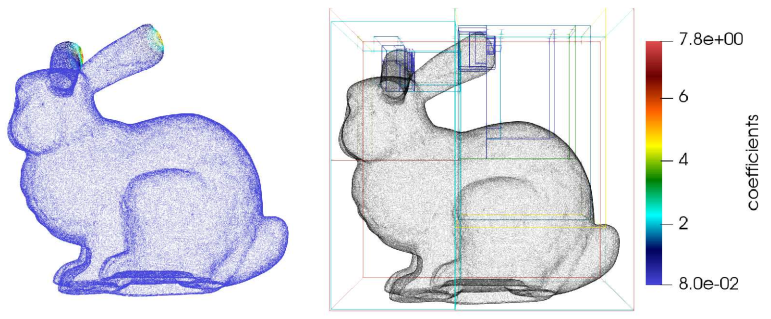


FIGURE 5. Data on the Stanford bunny (left) and dominant samplet coefficients (right).

6.2. **Compression of kernel matrices.** All computations in this section have been performed on a single node with two Intel Xeon E5-2650 v3 @2.30GHz CPUs and up to 512GB of main memory[1]. To achieve consistent timings, only a single core was used for all computations. The samplet compression is implemented in `C++11` and relies on the `Eigen` template library[2] for linear algebra operations. To benchmark the compression, we consider two different kernel functions, namely the exponential kernel k_{exp} and the rational quadratic kernel k_{RQ} given by

$$k_{\text{exp}}(\boldsymbol{x},\boldsymbol{y}) = e^{-\|\boldsymbol{x}-\boldsymbol{y}\|_2}, \quad k_{\text{RQ}}(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{\sqrt{1+\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}},$$

see e.g. [43]. The exponential kernel decays exponentially for $\|\boldsymbol{x}-\boldsymbol{y}\|_2 \to \infty$ and exhibits a kink for $\boldsymbol{x} = \boldsymbol{y}$. On the other hand, the rational quadratic kernel only decays linearly for $\|\boldsymbol{x}-\boldsymbol{y}\|_2 \to \infty$, while the kernel itself is smooth. In what follows, we consider the compression of these kernel functions, for data sets based on uniformly distributed points and on exponentially distributed points.

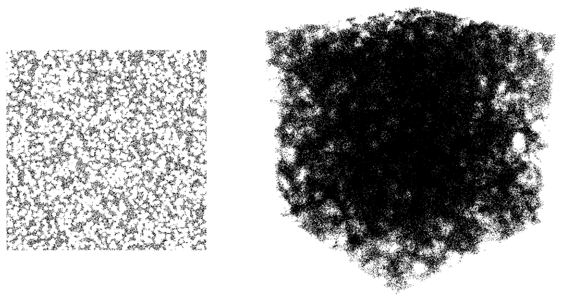


FIGURE 6. Test data sets for $d = 2$ and $d = 3$. We consider uniformly distributed random points on the hypercube with randomly cut out circular holes.

Uniformly distributed points. In this benchmark problem, the data set is selected from the hypercube $[0,1]^d$ with randomly distributed cut out circular holes. The radii of the holes are exponentially distributed, while their position is uniformly distributed. The points themselves are uniformly distributed, see Figure 6 for a visualization of two data sets for $d = 2,3$. The convergence of the samplet compression is steered by the parameter η in the admissibility condition and the number of vanishing moments. In the experiments, we shall keep η fixed and increase the number of vanishing moments. In addition, we introduce an a-posteriori thresholding of small matrix entries using the parameter τ, i.e. all entries whose modulus is smaller than τ are neglected. Finally, to keep the consistency error issuing from the kernel approximation (18) in the admissible blocks of the order of the compression error, we have to increase the polynomial degree p of the kernel approximation when increasing the number of vanishing moments. The respective parameter values can be found in Table 1. As a measure of sparsity, we introduce the

	$q=0$	$q=1$	$q=2$	$q=3$
m_q	$1,1,1$	$2,3,4$	$3,6,10$	$4,10,20$
p	2	3	4	6
η	1.25	1.25	1.25	1.25
τ	10^{-2}	10^{-3}	10^{-4}	10^{-5}

TABLE 1. Parameters chosen for the different numbers of vanishing moments. The three numbers for m_q correspond to $d = 1,2,3$.

[1]The full specifications can be found on https://www.euler.usi.ch/en/research/resources.

[2]`https://eigen.tuxfamily.org/`

average number of nonzeros per row

$$\operatorname{anz}(\boldsymbol{A}) := \frac{\operatorname{nnz}(\boldsymbol{A})}{N}, \qquad \boldsymbol{A} \in \mathbb{R}^{N\times N},$$

where $\operatorname{nnz}(\boldsymbol{A})$ is the number of nonzero entries of $\boldsymbol{A}$.

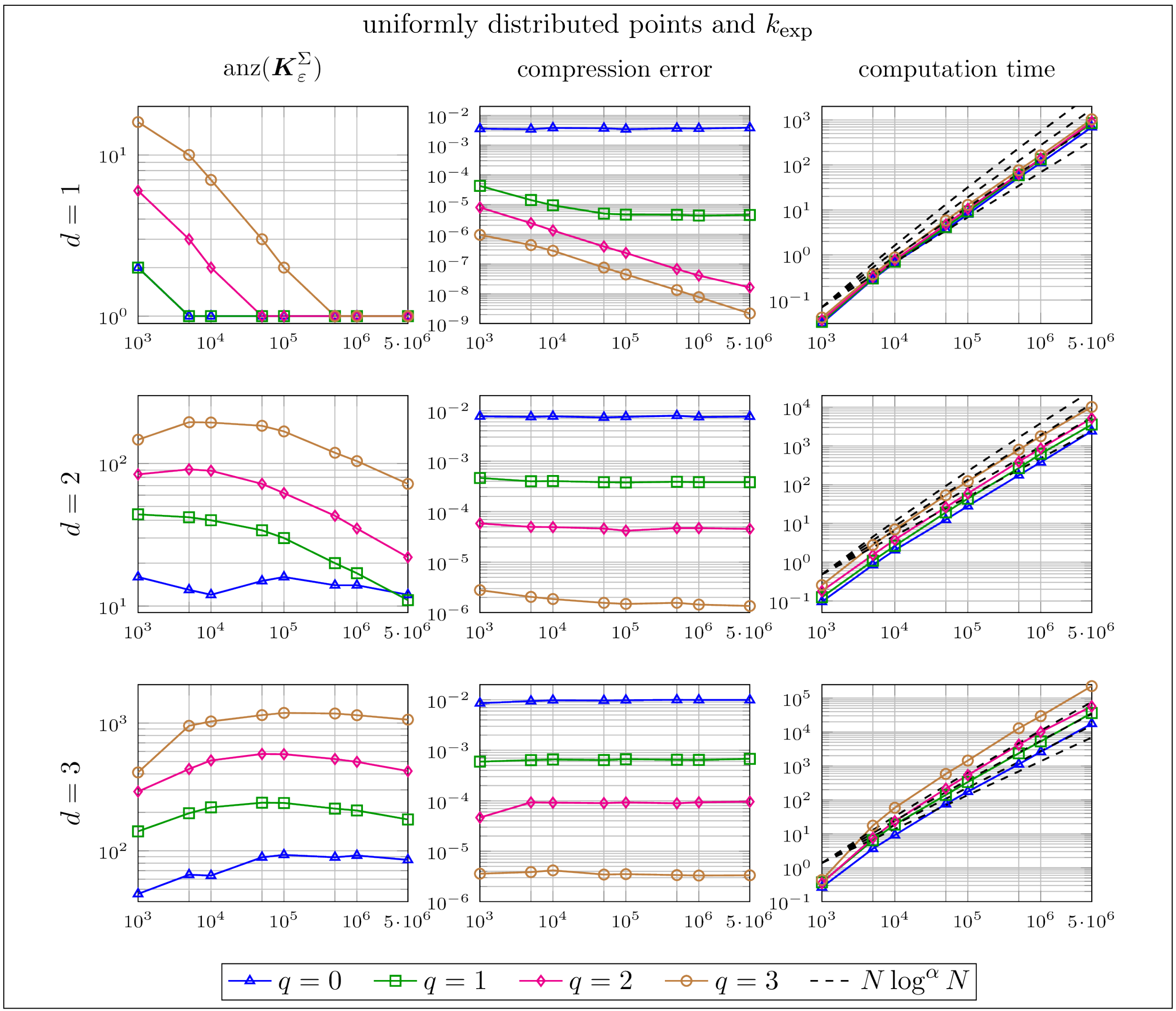


FIGURE 7. Number of entries per row (left), relative compression error (middle) and computation time (right) for $d = 1, 2, 3$ for the exponential kernel k_{exp} and uniformly distributed points.

Figure 7 shows the numerical results in case of the exponential kernel k_{exp} and uniformly distributed points. The left column shows the values for $\operatorname{anz}(\boldsymbol{K}_\varepsilon^\Sigma)$. The middle column shows the relative compression errors, where we have approximated the Frobenius norm by randomly sampling 100 columns of the original kernel matrix and the respective columns of the compressed one. The right column of Figure 7 shows the computation times for the matrix compression. Here, the dashed lines correspond to the asymptotics $N \log^\alpha N$ for $\alpha = 0, 1, 2, 3$. As we keep the precision fixed, the average number of matrix entries per row decreases for $d = 1, 2, 3$ and increasing N. Moreover, we see that the compression error reduces approximately by one order of magnitude if the number of vanishing moments is increased by one. For $d = 1$, we retrieve a log-linear rate for all numbers of vanishing moments. For $d = 2, 3$, we observe $\alpha > 1$ for higher

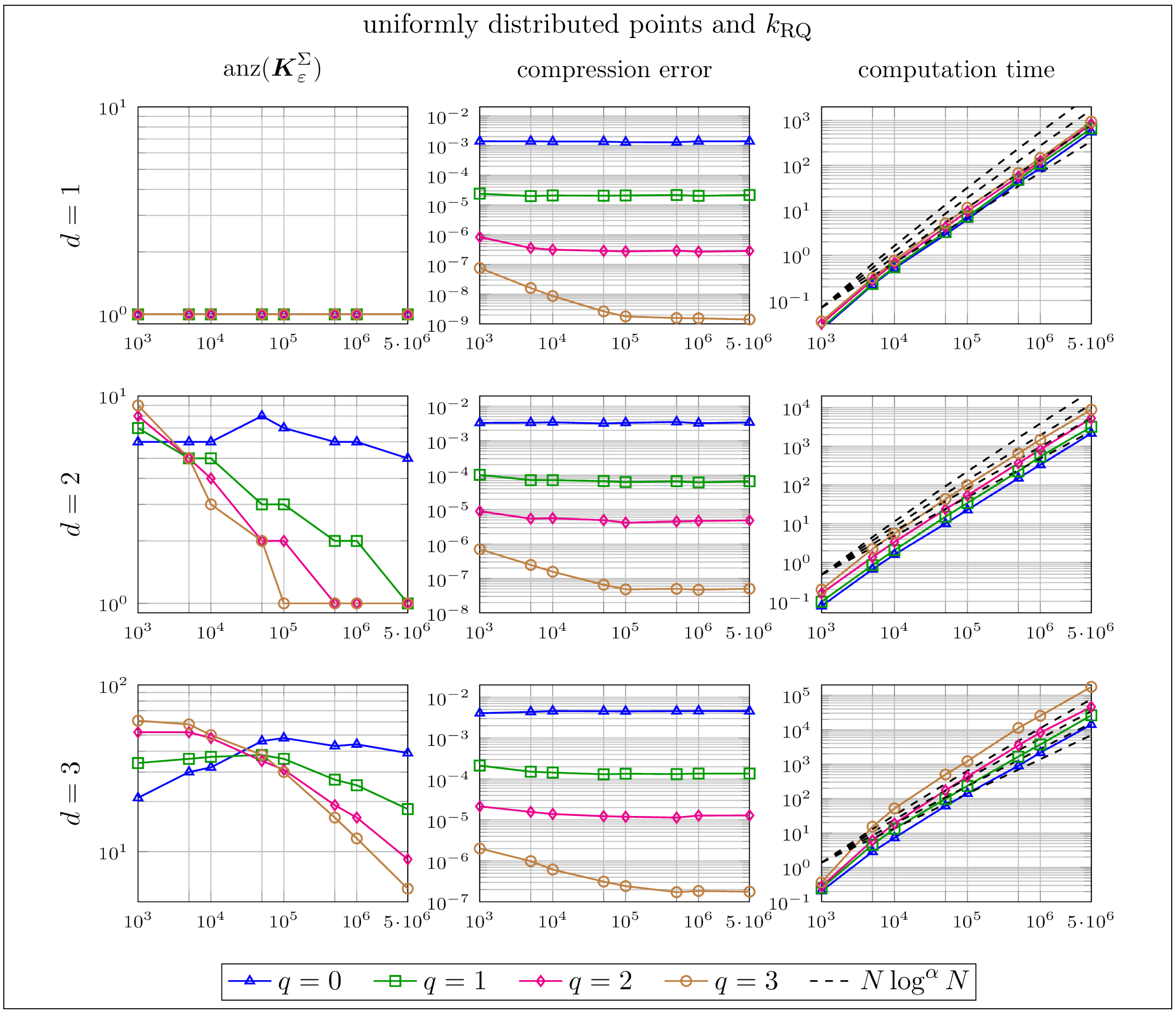


FIGURE 8. Number of entries per row (left), relative compression error (middle) and computation time (right) for $d = 1, 2, 3$ for the rational quadratic kernel k_{RQ} and uniformly distributed points.

numbers of vanishing moments. Even so, it seems that the power is reduced for larger values of N, indicating a preasymptotical behavior.

Figure 8 shows the same metrics as before in case of the rational quadratic kernel k_{RQ}. As the kernel is analytic, we observe a very low average number of entries per row, while obtaining a very high accuracy, particularly in case $q = 3$. The computation times are similar to the case of the exponential kernel.

Exponentially distributed points. To demonstrate that samplets also work on non-quasi-uniform data sets, we consider the hypercube $[0, 1]^d$ with randomly distributed cut out circular holes from before. This time, the points are exponentially distributed with respect to their distance from the origin. Figure 9 shows a visualization of the data set for $d = 2, 3$. Figure 10 shows the respective results for the exponential kernel. The average numbers of entries per row and the computation times are very similar to the case of uniformly distributed points, while the approximation error is slightly larger.

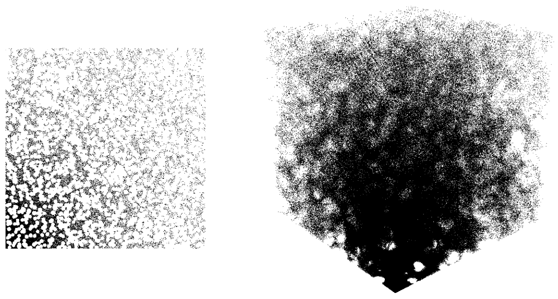


FIGURE 9. Test data sets for $d = 2$ and $d = 3$. We consider exponentially distributed random points, with respect to the origin, on the hypercube with randomly cut out circular holes.

The results for the rational quadratic kernel are found in Figure 11. As for the exponential kernel, there are no significant qualitative differences between the two point distributions, except that the compression error is slightly larger.

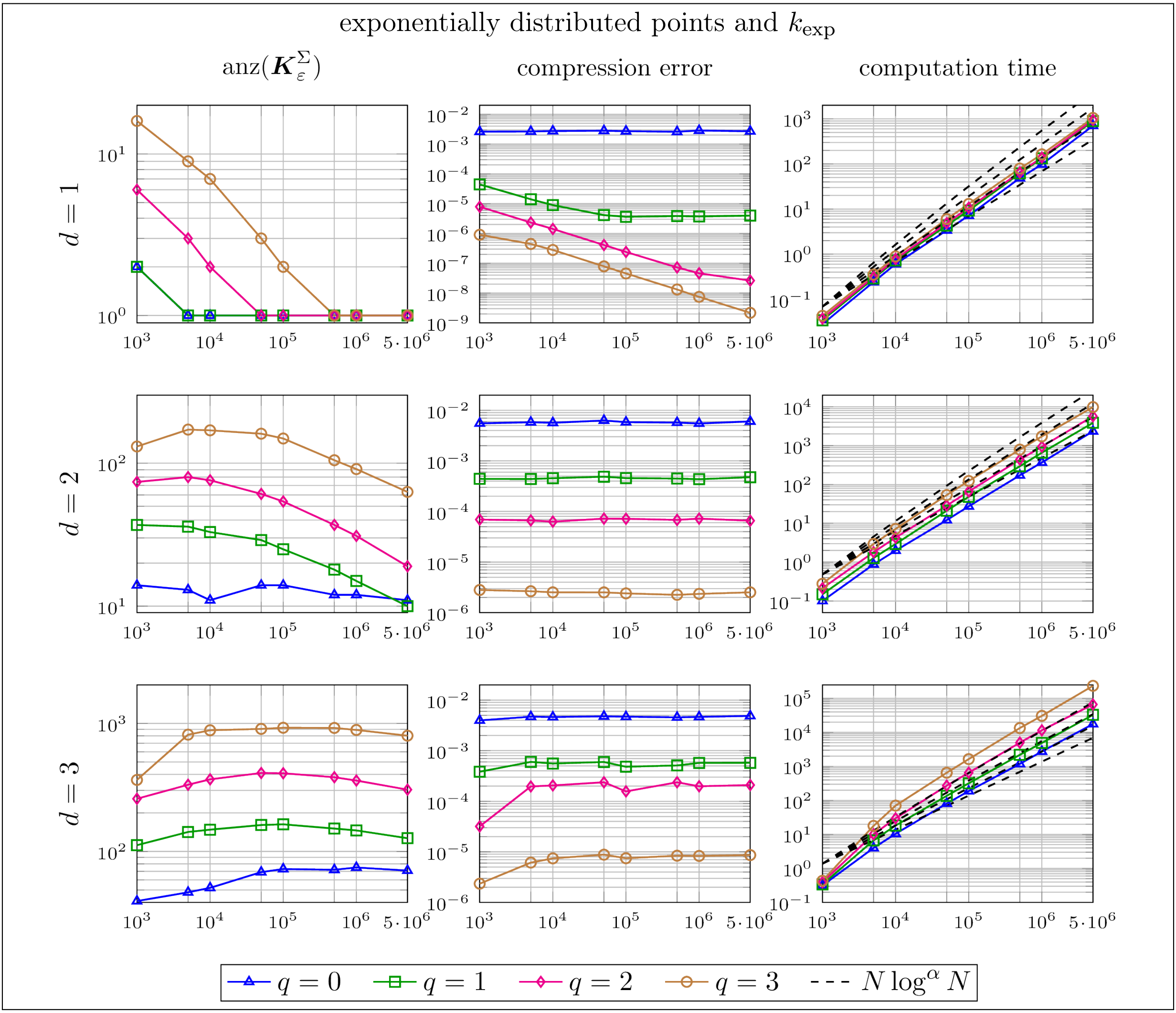


FIGURE 10. Number of entries per row (left), relative compression error (middle) and computation time (right) for $d = 1, 2, 3$ for the exponential kernel k_{exp} and exponentially distributed points.

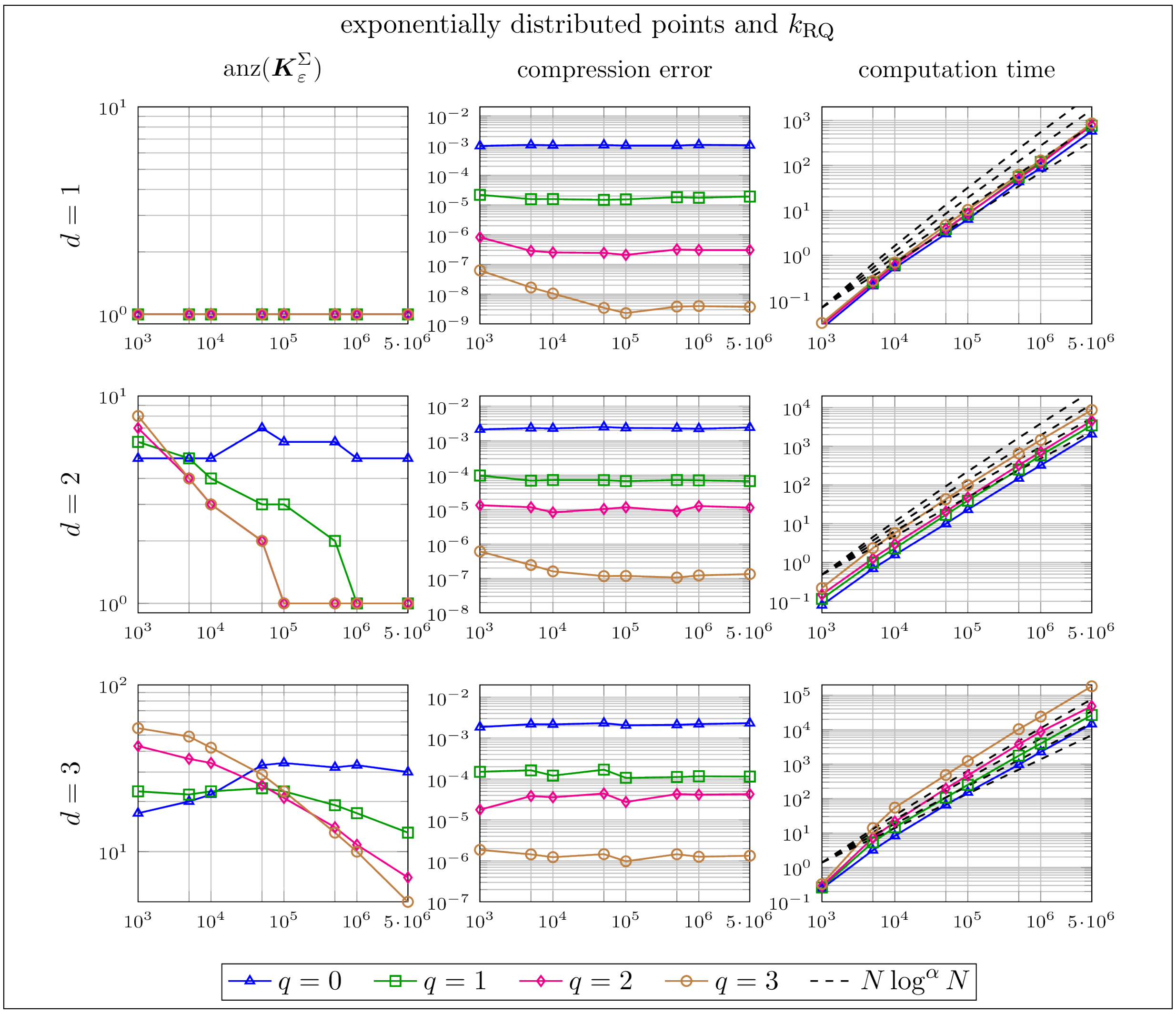


FIGURE 11. Number of entries per row (left), relative compression error (middle) and computation time (right) for $d = 1, 2, 3$ for the rational quadratic kernel k_{RQ} and exponentially distributed points.

Simulation of a Gaussian random field. As final example, we consider a Gaussian random field evaluated at randomly chosen points at the surface of the Stanford bunny. The bounding box of the Stanford bunny is given by $[-1.89, 1.22] \times [-0.34, 2.75] \times [-1.24, 1.176]$ and we consider the exponential kernel k_{exp} as covariance function, while we set the mean to zero. In order to demonstrate that our approach works also for dimensions larger than 3, the Stanford bunny has been embedded into $\mathbb{R}^4$ and randomly rotated to prevent axis-aligned bounding boxes. The parameters are set to $q = 2, p = 4, \eta = 1.25, \tau = 10^{-3}$, which results in a relative compression error of about $3 \cdot 10^{-4}$ for $N = 1\,000\,000$ points. For the simulation of the Gaussian random field, we compute the Cholesky decomposition of the compressed covariance matrix. To this end, we have added a ridge parameter of $\rho = 10^{-6}$ relative to the trace of the covariance matrix. The Cholesky decomposition is performed using the nested dissection ordering implemented in the `METIS` library, cp. [29]. The graph on the left of Figure 12 shows the computation times for the Cholesky decomposition including the ordering of the matrix. As can be seen, the computation times are even better than the expected rate of $\mathcal{O}(N^{3/2})$ for graphs that exhibit a $\sqrt{N}$-separator. The associated number of entries per row is about 1000. The sparsity pattern of

$\boldsymbol{K}_\varepsilon^\Sigma$ for $N = 100\,000$ can be found in the middle of Figure 12, while the corresponding sparsity pattern of the Cholesky factor is found on the right. Each dot represents a matrix block of size 100×100, lighter blocks have less entries. Performing an a-posteriori thresholding of the Cholesky factor with 10^{-6} reduces this number by about 30%, while a thresholding with 10^{-3} even reduces this number by about 80%. Four different realizations of the corresponding Gaussian random field field projected to $\mathbb{R}^3$ are shown in Figure 13.

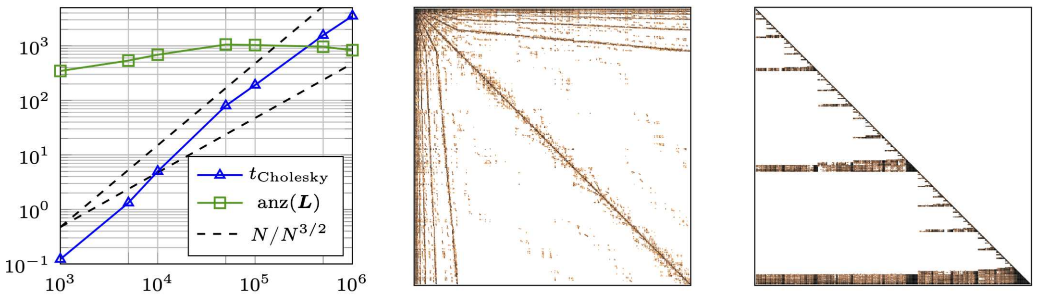


FIGURE 12. Computation times for the Cholesky decomposition for k_{exp} and corresponding values for anz($\boldsymbol{L}$) (left), sparsity pattern of $\boldsymbol{K}_\varepsilon^\Sigma$ (middle) and sparsity pattern of $\boldsymbol{L}$ (right) for the four dimensional Stanford bunny. Each dot represents a 100×100 matrix block, lighter blocks have less entries.

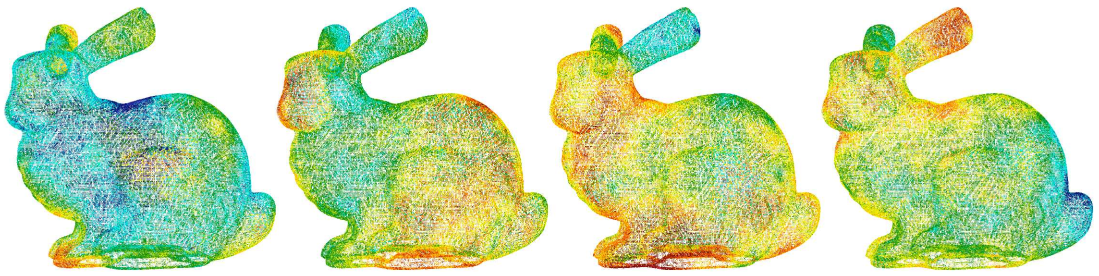


FIGURE 13. Four different realizations of a Gaussian random field with covariance k_{exp} on a Stanford bunny embedded into four dimensions for $N = 100\,000$ (three dimensional projection shown).

7. CONCLUSION

Samplets are multiresolution approach for the analysis of large data sets. They are easy to construct and scattered data can be transformed into a samplet basis with linear cost. In our construction, we deliberately let out the discussion of a level dependent compression of the given data, as it is known from wavelet analysis, in favor of a robust error analysis. We emphasize however that, under the assumption of uniformly distributed points, different norms can be incorporated, allowing for the construction of band-pass filters and level dependent thresholding. In this situation, also an improved samplet matrix compression is possible such that a fixed number of vanishing moments would be sufficient to achieve a precision proportional to the fill distance with log-linear cost.

Besides data compression, detection of singularities and adaptivity, we have demonstrated how samplets can be employed for the compression of kernel matrices to obtain sparse representations. Having a sparse representation of the kernel matrix, algebraic operations, such as matrix vector multiplications can considerably be sped up. Moreover, in combination with a fill-in reducing reordering, the factorization of the compressed kernel matrix becomes computationally feasible. This, in turn, allows for the fast application of the inverse kernel matrix on the one hand and the efficient solution of linear systems involving the kernel matrix on the other hand. The numerical results, featuring about $5 \cdot 10^6$ data points in up to four dimensions, demonstrate the capabilities of samplets.

Future research will be directed to the extension of samplets towards high-dimensional data. This extension requires the incorporation of different clustering strategies, such as locality sensitive hashing, to obtain a manifold-aware cluster tree and the careful construction for the vanishing moments, for example by anisotropic polynomials.

References

[1] D. Alm, H. Harbrecht, and U. Krämer. The $\mathcal{H}^2$-wavelet method. *J. Comput. Appl. Math.*, 267:131–159, 2014.

[2] B.K. Alpert. A class of bases in L^2 for the sparse representation of integral operators. *SIAM J. Math. Anal.*, 24(1):246–262, 1993.

[3] N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68(3):337–404, 1950.

[4] G. Beylkin, R. Coifman, and V. Rokhlin. The fast wavelet transform and numerical algorithm. *Comm. Pure Appl. Math*, 44:141–183, 1991.

[5] S. Börm. *Efficient numerical methods for non-local operators: $\mathcal{H}^2$-matrix compression, algorithms and analysis.* European Mathematical Society, Zürich, 2010.

[6] C.K. Chui. *An introduction to wavelets.* Academic Press, San Diego, 1992.

[7] C.K. Chui and E. Quak. Wavelets on a bounded interval. *Numer. Meth. Approx. Theory*, 9:53–75, 1992.

[8] A. Cohen. *Numerical Analysis of Wavelet Methods.* Elsevier, Amsterdam, 2003.

[9] R.R. Coifman and M. Maggioni. Diffusion wavelets. *Appl. Comput. Harmon. Anal.*, 21(1):53–94, 2006.

[10] W. Dahmen. Wavelet and multiscale methods for operator equations. *Acta Numer.*, 6:55–228, 1997.

[11] W. Dahmen, H. Harbrecht, and R. Schneider. Compression techniques for boundary integral equations. Optimal complexity estimates. *SIAM J. Numer. Anal.*, 43(6):2251–2271, 2006.

[12] W. Dahmen, A. Kunoth, and K. Urban. Biorthogonal spline wavelets on the interval – stability and moment conditions. *Appl. Comp. Harm. Anal.*, 6(2):132–196, 1999.

[13] W. Dahmen, S. Prößdorf, and R. Schneider. Wavelet approximation methods for pseudodifferential equations ii: Matrix compression and fast solution. *Adv. Comput. Math.*, 1(3):259–335, 1993.

[14] W. Dahmen and R. Stevenson. Element-by-element construction of wavelets satisfying stability and moment conditions. *SIAM J. Numer. Anal.*, 37(1):319–352, 1999.

[15] I. Daubechies. *Ten lectures on wavelets.* Society of Industrial and Applied Mathematics, Philadelphia, 1992.

[16] R. A. DeVore. Nonlinear approximation. *Acta Numer.*, 7:51–150, 1998.

[17] G.E. Fasshauer. *Meshfree approximation methods with MATLAB.* World Scientific, River Edge, 2007.

[18] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10(2):345–363, 1973.

[19] K. Giebermann. Multilevel approximation of boundary integral operators. *Computing*, 67:183–207, 2001.

[20] D. Gines, G. Beylkin, and J. Dunn. LU factorization of non-standard forms and direct multiresolution solvers. *Appl. Comput. Harmon. Anal*, 5(2):156–201, 1998.

[21] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73(2):325–348, 1987.

[22] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis.* Springer, Berlin Heidelberg, 2015.

[23] W. Hackbusch and S. Börm. $\mathcal{H}^2$-matrix approximation of integral operators by interpolation. *Appl. Numer. Math.*, 43(1–2):129–143, 2002.

[24] H. Harbrecht, U. Kähler, and R. Schneider. Wavelet Galerkin BEM on unstructured meshes. *Comput. Vis. Sci.*, 8(3–4):189–199, 2005.

[25] H. Harbrecht and M. Multerer. A fast direct solver for nonlocal operators in wavelet coordinates. *J. Comput. Phys.*, 428:110056, 2021.

[26] H. Harbrecht and R. Schneider. Biorthogonal wavelet bases for the boundary element method. *Math. Nachr.*, 269(1):167–188, 2004.

[27] T. Hofmann, B. Schölkopf, and A.J. Smola. Kernel methods in machine learning. *Ann. Stat.*, 36(3):1171–1220, 2008.

[28] U. Kähler. *$\mathcal{H}^2$-wavelet Galerkin BEM and its application to the radiosity equation.* Dissertation TU Chemnitz, Chemnitz, 2007.

[29] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[30] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized nested dissection. *SIAM J. Numer. Anal.*, 16(2):346–358, 1979.

[31] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, 1999.

[32] S. Mallat. Understanding deep convolutional networks. *Philos. Trans. R. Soc. A*, 374(2065):20150203, 2016.

[33] W.B. March, B. Xiao, S. Tharakan, D.Y. Chenhan, and G. Biros. A kernel-independent FMM in general dimensions. In *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12, 2015.

[34] H. Owhadi and C. Scovel. *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*, volume 35. Cambridge University Press, Cambridge, 2019.

[35] I. Ram, M. Elad, and I. Cohen. Generalized tree-based wavelet transform. *IEEE Trans. Signal Process.*, 59(9):4199–4209, 2011.

[36] R. Schaback and H. Wendland. Kernel techniques: from machine learning to meshless methods. *Acta Numer.*, 15:543–639, 2006.

[37] F. Schäfer, T.J. Sullivan, and H. Owhadi. Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity. *SIAM Multiscale Model. Simul.*, 19(2):688–730, 2021.

[38] R. Schneider. *Multiskalen- und Wavelet-Matrixkompression: Analysisbasierte Methoden zur Lösung großer vollbesetzter Gleichungssysteme*. B.G. Teubner, Stuttgart, 1998.

[39] J. Tausch and J. White. Multiscale bases for the sparse representation of boundary integral operators on complex geometry. *SIAM J. Sci. Comput.*, 24(5):1610–1629, 2003.

[40] T. von Petersdorff and C. Schwab. Fully discrete multiscale Galerkin BEM. In W. Dahmen, A. Kurdila, and P. Oswald, editors, *Multiscale wavelet methods for PDEs*, pages 287–346. Academic Press, San Diego, 1997.

[41] T. von Petersdorff, C. Schwab, and R. Schneider. Multiwavelets for second-kind integral equations. *SIAM J. Numer. Anal.*, 34(6):2212–2227, 1997.

[42] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, Cambridge, 2004.

[43] C.K. Williams and C.E. Rasmussen. *Gaussian processes for machine learning*. MIT Press, Cambridge, 2006.

[44] C.K.I. Williams. Prediction with Gaussian processes. From linear regression to linear prediction and beyond. In M.I. Jordan, editor, *Learning in Graphical Models*, volume 89 of *NATO ASI Series (Series D: Behavioural and Social Sciences)*. Springer, Dordrecht, 1998.

[45] L. Ying, G. Biros, and D. Zorin. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Computat. Phys.*, 196(2):591–626, 2004.

Helmut Harbrecht, Departement für Mathematik und Informatik, Universität Basel, Spiegelgasse 1, 4051 Basel, Switzerland.

Email address: helmut.harbrecht@unibas.ch

Michael Multerer, Euler Institute, USI Lugano, Via la Santa 1, 6962 Lugano, Svizzera.

Email address: michael.multerer@usi.ch