SAMPLETS: A NEW PARADIGM FOR DATA COMPRESSION

HELMUT HARBRECHT AND MICHAEL MULTERER

Abstract. In this article, we introduce the concept of samplets by transferring the construction of Tausch-White wavelets [41] to the realm of data. This way we obtain a multilevel representation of discrete data which directly enables data compression, detection of singularities and adaptivity. Applying samplets to represent kernel matrices, as they arise in kernel based learning or Gaussian process regression, we end up with quasi-sparse matrices. By thresholding small entries, these matrices are compressible to $O(N \log N)$ relevant entries, where $N$ is the number of data points. This feature allows for the use of fill-in reducing reorderings to obtain a sparse factorization of the compressed matrices. Besides the comprehensive introduction to samplets and their properties, we present extensive numerical studies to benchmark the approach. Our results demonstrate that samplets mark a considerable step in the direction of making large data sets accessible for analysis.

1. Introduction

Wavelet techniques have a long standing history in the field of data science. Applications comprise signal processing, image analysis and machine learning, see for instance [6,9,14,29,30] and the references therein. Assuming a signal generated by some function, the pivotal idea of wavelet techniques is the splitting of this function into its contributions with respect to a hierarchy of scales. Such a multiscale ansatz starts from an approximation on a relatively coarse scale and successively resolves details at finer scales. Hence, compression and adaptive representation are inherently built into this ansatz. The transformation of a given signal into its wavelet representation and the inverse transformation can be performed with linear cost in terms of the degrees of freedom.

Classically, wavelets are constructed by refinement relations and therefore require a sequence of nested approximation spaces which are copies of each other, except for a different scaling. This restricts the concept of wavelets to structured data. Some adaption of the general principle is possible in order to treat intervals, bounded domains and surfaces, compare [2,7,11,13,24,33] for example. The seminal work [41] by Tausch and White overcomes this obstruction by constructing wavelets as suitable linear combinations of functions at a given fine scale. In particular, the stability of the resulting basis, which is essential for numerical algorithms is guaranteed by orthonormality.

In this article, we take the concept of wavelets to the next level and consider discrete, unstructured data. To this end, we modify the construction of Tausch and White and construct a multiscale basis which consists of localized and discrete signed measures. Inspired by the term wavelet, we call such signed measures samplets. Samplets 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 $q+1$.

We remark that lowest order samplets, i.e., $q = 0$, have been considered earlier for data compression in [35]. Another concept for constructing multiscale bases on data sets are diffusion wavelets, which employ a diffusion operator to construct the multiscale hierarchy, see [8]. In contrast to diffusion wavelets, however, the construction of samplets is solely based on discrete structures and can always be performed with linear cost for a balanced cluster tree, even for non-uniformly distributed data.

When representing discrete data by samplets, then, due to the vanishing moments, there is a fast decay of the corresponding samplet coefficients with respect to the support size if the data are smooth. This straightforwardly enables data compression. In contrast, non-smooth regions in the data are indicated by large samplet coefficients. This, in turn, enables singularity detection and
extraction. Furthermore, the construction of samplets is not limited to the use of polynomials. Indeed, it is easily be possible to adapt the construction to other primitives with different desired properties.

The second application of samplets we consider is compression of kernel matrices, as they arise in kernel based machine learning and scattered data approximation, compare [15, 25, 36, 38, 42, 43]. 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. A discretization of an asymptotical smooth kernel with respect to a samplet basis with vanishing moments results in quasi-sparse kernel matrices, which means that they can be compressed such that only a sparse matrix remains, compare [1, 10, 12, 34, 39]. Especially, it has been demonstrated in [23] that nested dissection, see [16, 28], is applicable in order to obtain a fill-in reducing reordering of the matrix in the standard form. This reordering in turn allows for the rapid factorization of the system matrix by the Cholesky factorization without introducing additional errors. This is in contrast to the approximate computation of the Cholesky factorization with respect to the so-called non-standard form of operators or by $H$-matrices which has been proposed earlier, compare [18, 20].

The asymptotic smoothness of the kernels is also exploited by cluster methods, like the fast multipole method, see [19, 37, 44] and particularly [31] for high-dimensional data. However, these methods do not allow for the direct and exact factorization, which is for example advantageous for the simulation of Gaussian random fields. A further approach, which is more in line of the present work, is the use of gamblets, see [32], for the compression of the kernel matrix, cp. [40]. Different from the discrete construction of samplets with vanishing moments, the construction of gamblets is adapted to an underlying pseudo-differential operator and basis functions need to be truncated in order to obtain localized supports, while localized supports are automatically obtained by the samplet construction.

As samplets are directly constructed with respect to a discrete data set, their applications are manifold. Within this article, we particularly consider time-series data, image data, kernel matrix representation and the simulation of Gaussian random fields as examples. We remark, however, that we do not claim to have invented a new method for high-dimensional data approximation. The current construction is based on total degree polynomials and is hence not dimension robust, thus limited to data of moderate dimension. Even so, we believe that samplets provide most of the advantages of other approaches for scattered data, while being easy to implement. Especially, most of the algorithms available for wavelets with vanishing moments are transferable.

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 actual construction of samplets and to their properties. The change of basis by means of the discrete samplet transform is the topic of Section 4. In Section 5, we demonstrate the capabilities of samplets for data compression and smoothing for data in one, two and three dimensions. Section 6 deals with the samplet compression of kernel matrices. Especially, we also employ an interpolation based $H^2$-matrix approach in order to efficiently assemble the compressed kernel matrix. Corresponding numerical results are then presented in Section 7 for up to four dimensions. Finally, in Section 8, we state concluding remarks.

2. Samplets

Let $X := \{x_1, \ldots, x_N\} \subseteq \Omega$ denote a set of points within some region $\Omega \subseteq \mathbb{R}^d$. Associated to each point $x_i$, we introduce the Dirac measure

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

With a slight abuse of notation, we also introduce the point evaluation functional

$$(f, \delta_{x_i})_\Omega = \int_\Omega f(x)\delta_{x_i}(x) \, dx := \int_\Omega f(x)\delta_{x_i}(dx) = f(x_i),$$

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

Next, we define the space \( V := \text{span}\{\delta_{x_1}, \ldots, \delta_{x_N}\} \) as the \( N \)-dimensional vector space of all discrete and finite signed measures supported at the points in \( X \). An inner product on \( V \) is defined by

\[
(u, v)_V := \sum_{i=1}^{N} u_i v_i,
\]

where \( u = \sum_{i=1}^{N} u_i \delta_{x_i} \), \( v = \sum_{i=1}^{N} v_i \delta_{y_i} \).

Indeed, the space \( V \) is isometrically isomorphic to \( \mathbb{R}^N \) endowed with the canonical inner product.

Similar to the idea of a multiresolution analysis in the construction of wavelets, we introduce the spaces \( V_j := \text{span} \Phi_j \), where

\[
\Phi_j := \{ \varphi_{j,k} : k \in \Delta_j \}.
\]

Here, \( \Delta_j \) denotes a suitable index set with cardinality \( |\Delta_j| = \dim V_j \) 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

\[
\langle \varphi_{j,k}, \varphi_{j,k'} \rangle_V = 0 \quad \text{for } k \neq k'.
\]

For the sake of notational convenience, we shall identify bases by row vectors, such that, for \( \mathbf{v}_j = [v_{j,k}] \), the corresponding measure can simply be written as a dot product according to

\[
\mathbf{v}_j = \Phi_j \mathbf{v}_j = \sum_{k \in \Delta_j} v_{j,k} \varphi_{j,k}.
\]

Rather than using the multiresolution analysis corresponding to the hierarchy

\[
V_0 \subset V_1 \subset \cdots \subset V,
\]

the idea of samplets is to keep track of the increment of information between two consecutive levels \( j \) and \( j + 1 \). Since we have \( V_j \subset V_{j+1} \), we may decompose

\[
(1) \quad V_{j+1} = V_j \perp S_j
\]

by using the detail space \( S_j \). Of practical interest is the particular choice of the basis of the detail space \( S_j \) in \( V_{j+1} \). This basis is assumed to be orthonormal as well and will be denoted by

\[
\Sigma_j = \{ \sigma_{j,k} : k \in \nabla_j := \Delta_{j+1} \setminus \Delta_j \}.
\]

Recursively applying the decomposition \((1)\), we see that the set

\[
\Sigma_j = \Phi_0 \cup \bigcup_{j=0}^{J-1} \Sigma_j
\]

forms a basis of \( V_J := V \), which we call a samplet basis. In view of data compression, an essential ingredient is the vanishing moment condition, meaning that

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

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.** In case of uniformly distributed points, we can obtain bases which satisfy

\[
\text{diam(supp } \varphi_{j,k} \text{)} := \text{diam} (\{ x_{i_1}, \ldots, x_{i_p} \}) \sim 2^{-j/d}
\]

and, likewise,

\[
\text{diam(supp } \sigma_{j,k} \text{)} \sim 2^{-j/d}.
\]

These properties are favorable with regard to the compression of data and 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 \((H, \langle \cdot, \cdot \rangle_H)\) is a reproducing kernel Hilbert space with reproducing kernel \(K\), then there holds \(\langle f, \delta_{x_i} \rangle_H = \langle K(x_i, \cdot), f \rangle_H\). Hence, the samplet \(\sigma_{j,k} = \sum_{\ell=1}^p \beta_\ell \delta_{x_{i\ell}}\) can directly be identified with the function
\[
\hat{\sigma}_{j,k} := \sum_{\ell=1}^p \beta_\ell K(x_{i\ell}, \cdot) \in H.
\]
In particular, it holds
\[
\langle \hat{\sigma}_{j,k}, h \rangle_H = 0
\]
for any \(h \in H\) which satisfies \(h|_{\text{supp} \sigma_{j,k}} \in \mathcal{P}_q(\text{supp} \sigma_{j,k})\).

3. Construction of samplets

3.1. Cluster tree. In order to construct samplets with the desired properties, especially vanishing moments, cf. (2), we shall transfer the wavelet construction of Tausch and White from [41] into our setting. The first step is to construct a hierarchy of subspaces of signed measures. To this end, we perform a hierarchical clustering on the set \(X\).

Definition 3.1. Let \(T = (P, E)\) be a tree with vertices \(P\) and edges \(E\). We define its set of leaves as
\[
\mathcal{L}(T) := \{\nu \in P: \nu\ has no sons\}.
\]
The tree \(T\) is a cluster tree for the set \(X = \{x_1, \ldots, x_N\}\), if the set \(X\) is the root of \(T\) and all \(\nu \in P \setminus \mathcal{L}(T)\) are disjoint unions of their sons.

The level \(j_\nu\) of \(\nu \in T\) is its distance from the root, i.e., the number of son relations that are required for traveling from \(X\) to \(\nu\). The depth \(J\) of \(T\) is the maximum level of all clusters. We define the set of clusters on level \(j\) as
\[
T_j := \{\nu \in T: \nu\ has level j\}.
\]
Finally, the bounding box \(B_\nu\) of \(\nu\) is defined as the smallest axis-parallel cuboid that contains all its points.

There exist several possibilities for the choice of a cluster tree for the set \(X\). However, within this article, we will exclusively consider binary trees and remark that it is of course possible to consider other options, such as \(2^d\)-trees, 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 \(T\) be a cluster tree on \(X\) with depth \(J\). \(T\) is called a balanced binary tree, if all clusters \(\nu\) satisfy the following conditions:

1. The cluster \(\nu\) has exactly two sons if \(j_\nu < J\). It has no sons if \(j_\nu = J\).
2. 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 \(O(\log N)\) levels in total. The total cost for constructing the cluster tree is therefore \(O(N \log N)\). 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. Multiscale hierarchy. Having a cluster tree at hand, we shall now construct a samplet basis on the resulting hierarchical structure. We begin by introducing a two-scale transform between basis elements on a cluster $\nu$ of level $j$. To this end, we create scaling functions $\Phi_j = \{\varphi_{j,k}\}$ and samplets $\Sigma_j = \{\sigma_{j,k}\}$ as linear combinations of the scaling functions $\Phi_{j+1}$ of $\nu$'s son clusters. This results in the refinement relation

$$\Phi_j, \Sigma_j := \Phi_{j+1} Q_j = \Phi_{j+1} \left[ Q_{j,\Phi}, Q_{j,\Sigma} \right].$$

In order to provide both, vanishing moments and orthonormality, the transformation $Q_j$ has to be appropriately constructed. For this purpose, we consider an orthogonal decomposition of the moment matrix

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

where

$$m_q := \sum_{k=0}^q \left( \ell + d - 1 \right) = \frac{(q + d)^d}{d - 1} \leq (q + 1)^d$$

denotes the dimension of $P_q(\Omega)$.

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

$$(M_{j+1})^T = Q_j R =: [Q_{j,\Phi}, Q_{j,\Sigma}] R.$$

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

$$M_{j+1} = \begin{bmatrix} (x^\alpha, \Phi_j)_{\nu} \Sigma_j \end{bmatrix}_{|\alpha| \leq q} = \begin{bmatrix} (x^\alpha, \Phi_{j+1})_{\nu} [Q_{j,\Phi}, Q_{j,\Sigma}] \end{bmatrix}_{|\alpha| \leq q} = M_{j+1} [Q_{j,\Phi}, Q_{j,\Sigma}] = R^T.$$

As $R^T$ 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 function generated by the transformation $Q_j = [Q_{j,\Phi}, Q_{j,\Sigma}]$. By defining the first $m_q$ functions as scaling functions 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 $q + 1$ at the leaf clusters such that $m_q \geq 2m_q$, we can even construct samplets with an increased number of vanishing moments up to order $q + 1$ without any additional cost.

Remark 3.3. We remark that 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 of interest.

Remark 3.4. Each cluster has at most a constant number of scaling functions and samplets: For a particular cluster $\nu$, their number is identical to the cardinality of $\Phi_{j+1}$. For leaf clusters, this number is bounded by the leaf size. For non-leaf clusters, it is bounded by the number of scaling functions provided from all its son clusters. As there are at most two son clusters with a maximum of $m_q$ scaling functions each, we obtain the bound $2m_q$ for non-leaf clusters. Note that, if $\Phi_{j+1}$ has at most $m_q$ elements, a cluster will not provide any samplets at all and all functions will be considered as scaling functions.

For leaf clusters, we define the scaling functions by the Dirac measures supported at the points $x_i$, i.e. $\Phi_j := \{\delta_{x_i} : x_i \in \nu\}$. The scaling functions of all clusters on a specific level $j$ then generate the spaces

$$V_j := \text{span}\{\varphi_{j,k} : k \in \Delta_{j}^\nu, \nu \in T_j\},$$

while the samplets span the detail spaces

\[ S_j := \text{span}\{\sigma_{\nu,j,k}^\nu : k \in \nabla_{\nu,j}^\nu, \ \nu \in T_j\} = V_{j+1} \oplus V_j. \]

Combining the scaling functions of the root cluster with all clusters’ samplets gives rise to the samplet basis

\[ \Sigma_N := \Phi_0^X \cup \bigcup_{\nu \in T} \Sigma_j^\nu. \]

Writing \( \Sigma_N = \{\sigma_k : 1 \leq k \leq N\} \), where \( \sigma_k \) is either a samplet or a scaling function at the root cluster, we can establish a unique indexing of all the signed measures comprising the samplet basis. The indexing induces an order on the basis set \( \Sigma_N \), which we choose to be level-dependent: Samplets belonging to a particular cluster are grouped together, with those on finer levels having larger indices.

Remark 3.5. We remark that the samplet basis on a balanced cluster tree can be computed in cost \( O(N) \), we refer to [1] for a proof of this statement.

3.3. Properties of the samplets. By construction, samplets satisfy the following properties, which can directly be inferred from the corresponding results in [22,41].

Theorem 3.6. The spaces \( V_j \) defined in equation \( (8) \) exhibit the desired multiscale hierarchy

\[ V_0 \subset V_1 \subset \cdots \subset V_J = V, \]

where the corresponding complement spaces \( S_j \) from \( (9) \) satisfy \( V_{j+1} = V_j \oplus S_j \) for all \( j = 0, 1, \ldots, J - 1 \). The associated samplet basis \( \Sigma_N \) defined in \( (10) \) forms an orthonormal basis of \( V \). 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 \( \nu \).

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

Remark 3.8. Due to \( S_j \subset V \) and \( V_0 \subset V \), we conclude that each samplet is a linear combination of the Dirac measures supported at the points in \( X \). Especially, the related coefficient vectors \( \omega_{j,k} \) in

\[ \sigma_{j,k} = \sum_{i=1}^{N} \omega_{j,k,i} \delta_{x_i}, \quad \text{and} \quad \varphi_{0,k} = \sum_{i=1}^{N} \omega_{0,k,i} \delta_{x_i}, \]

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 \( \omega_{j,k} = [\omega_{j,k,i}]_1 \) of the samplet \( \sigma_{j,k} \) on the cluster \( \nu \) fulfills

\[ \| \omega_{j,k} \|_1 \leq \sqrt{\nu}. \]

The same holds for the scaling functions \( \varphi_{j,k} \).

Proof. It holds \( \| \omega_{j,k} \|_2 = 1 \). Hence, the assertion follows immediately from the Cauchy-Schwarz inequality

\[ \| \omega_{j,k} \|_1 \leq \sqrt{\nu} \| \omega_{j,k} \|_2 = \sqrt{\nu}. \]

\[ \square \]
Here, we used the estimate which shows that the samplet coefficients decay with respect to the samplet’s level provided that the data result from the evaluation of a smooth function. Therefore, 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 coefficients reflects 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 \( \nu \) that

\[
|f(\nu) - f(\sigma_{j,k})| \leq \text{diam}(\nu)^{q+1}\|f\|_{C^{q+1}(\Omega)}\|\omega_{j,k}\|_1.
\]

**Proof.** For \( x_0 \in \nu \), a Taylor expansion of \( f \) yields

\[
f(x) = \sum_{|\alpha| \leq q} \frac{\partial^{|\alpha|}}{\partial x^\alpha} f(x_0) (x - x_0)^\alpha + R_{x_0}(x).
\]

Herein, the remainder \( R_{x_0}(x) \) reads

\[
R_{x_0}(x) = (q + 1) \sum_{|\alpha| = q+1} \frac{(x - x_0)^\alpha}{\alpha!} \int_0^1 \frac{\partial^q f}{\partial x^\alpha} (x_0 + s(x - x_0))(1-s)^q \, ds.
\]

In view of the vanishing moments, we conclude

\[
|f(\nu) - f(\sigma_{j,k})| = |(R_{x_0}, \sigma_{j,k})| \leq \sum_{|\alpha| = q+1} \left\| \frac{(x - x_0)^\alpha}{\alpha!} \right\| \frac{\partial^q f}{\partial x^\alpha} (x) \left\| \omega_{j,k} \right\|_1 \leq \text{diam}(\nu)^{q+1}\|f\|_{C^{q+1}(\Omega)}\|\omega_{j,k}\|_1.
\]

Here, we used the estimate

\[
\sum_{|\alpha| = q+1} \frac{2^{-|\alpha|}}{\alpha!} \leq 1,
\]

which is obtained by choosing \( x_0 \) as the cluster’s midpoint. \( \square \)

**4. Discrete Samplet Transform**

In order to transform between the samplet basis and the basis of Dirac measures, we introduce the *discrete samplet transform* and its inverse. To this end, we assume that the data \((x_1, y_1), \ldots, (x_N, y_N)\) result from the evaluation of some (unknown) function \( f: \Omega \to \mathbb{R} \), i.e.

\[
y_i = f^{\Delta} = (f, \delta_{x_i})_{\Omega}.
\]

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

\[
f = \sum_{i=1}^N f^{\Delta}_i \delta_{x_i}.
\]

Our goal is now to compute the representation

\[
f = \sum_{i=1}^N f^\Sigma_i \sigma_k
\]

with respect to the samplet basis. For sake of a simpler notation, let \( f^{\Delta} := [f^{\Delta}_i]_{i=1}^N \) and \( f^\Sigma := [f^\Sigma_i]_{i=1}^N \) denote the associated coefficient vectors.

**Figure 1.** Visualization of the discrete samplet transform.
The discrete samplet transform is based on recursively applying the refinement relation \( \Omega \) to the point evaluations

\[
(f, \Phi_{j+1}^\nu) = (f, \Phi_j^\nu) [Q_{j,\Phi}^\nu, Q_{j,\Sigma}^\nu].
\]

On the finest level, the entries of the vector \((f, \Phi_N^\nu)\) are exactly those of \(f^\Delta\). Recursively applying equation (14) therefore yields all the coefficients \((f, \Phi_j^\nu)\), including \((f, \Phi_N^\nu)\), 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 4.1.

**Algorithm 4.1: Discrete samplet transform**

**Input:** Data \( f^\Delta \), cluster tree \( T \) and transformations \([Q_{j,\Phi}^\nu, Q_{j,\Sigma}^\nu]\).

**Result:** Coefficients \( f^\Sigma \) stored as \([f, \Phi_N^\nu]^\top \) and \([f, \Sigma_N^\nu]^\top \).

begin

\[
\text{store } ([f, \Phi_N^\nu]^\top) := \text{transformForCluster}(X)
\]

**Function** transformForCluster(\(\nu\))

begin

if \(\nu = \{x_{i_1}, \ldots, x_{i_m}\}\) is a leaf of \(T\) then

\[
\text{set } f_{j+1}^\nu := [f^\Delta_{k=1}^\nu]
\]

else

for all sons \(\nu'\) of \(\nu\) do

execute transformForCluster(\(\nu'\))

append the result to \(f_{j+1}^\nu\)

\[
\text{set } ([f, \Sigma_j^\nu]^\top)^\top := (Q_{j,\Sigma}^\nu)^\top f_{j+1}^\nu
\]

return \((Q_{f^\Sigma})^\top f_{j+1}^\nu\)

end

**Remark 4.1.** Algorithm 4.1 is based on the transposed version of (14) to preserve the column vector structure of \(f^\Delta\) and \(f^\Sigma\).

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

\[
(f, \Phi_j^\nu) = (f, \Phi_j^\nu) [Q_{j,\Phi}^\nu, Q_{j,\Sigma}^\nu]^\top
\]

to either obtain the coefficients of the son clusters’ scaling functions or, for leaf clusters, the coefficients \(f^\Delta\). The procedure is summarized in Algorithm 4.2.

**Algorithm 4.2: Inverse samplet transform**

**Input:** Coefficients \( f^\Sigma \), cluster tree \( T \) and transformations \([Q_{j,\Phi}^\nu, Q_{j,\Sigma}^\nu]\).

**Result:** Coefficients \( f^\Delta \) stored as \([f, \Phi_N^\nu]^\top \).

begin

inverseTransformForCluster(\(X, ([f, \Phi_N^\nu]^\top)\))

**Function** inverseTransformForCluster(\(\nu, ([f, \Phi_N^\nu]^\top)\))

begin

\[
([f, \Phi_{j+1}^\nu]^\top)^\top := [Q_{j,\Phi}^\nu, Q_{j,\Sigma}^\nu][([f, \Phi_j^\nu]^\top)^\top][([f, \Sigma_j^\nu]^\top)^\top]
\]

if \(\nu = \{x_{i_1}, \ldots, x_{i_m}\}\) is a leaf of \(T\) then

\[
\text{set } [f^\Delta_{k=1}^\nu] := ([f, \Phi_{j+1}^\nu]^\top)^\top
\]

else

for all sons \(\nu'\) of \(\nu\) do

assign the part of \([f, \Phi_{j+1}^\nu]^\top\) belonging to \(\nu'\) to \([f, \Phi_{j+1}^\nu]^\top\)

execute inverseTransformForCluster(\(\nu', ([f, \Phi_N^\nu]^\top)^\top\))

end

The discrete 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 4.2.** The runtime of the discrete samplet transform and the inverse samplet transform are \(O(N)\), each.
**Proof.** As the samplet construction follows the construction of Tausch and White, we refer to [41] for the details of the proof.

5. **Numerical results I**

To demonstrate the efficacy of the samplet analysis, we compress different sample 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

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

sampled at 8192 uniformly distributed points on $[-1, 1]$. On the other hand, we consider a path of a Brownian motion sampled at the same points. The coefficients of the samplet transformed data are thresholded with $10^{-i} \| f \Sigma \|_{\infty}$, $i = 1, 2, 3$, 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 throwing away more and more detail information. Indeed, due to the orthonormality of the samplet basis, this procedure amounts to a least squares fit of the data.
Two dimensions. As a second application for samplets, we consider image compression. To this end, we use a 2000 × 2000 pixel grayscale landscape image. The coefficients of the samplet transformed image are thresholded with $10^{-i}\|f^2\|_\infty$, $i = 2, 3, 4$, respectively. The corresponding results and compression rates can be found in Figure 4. A visualization of the samplet coefficients in case of the respective low compression can be found in Figure 5.

![Original image 95.23% compression](image1.png) ![95.23% compression](image2.png) ![99.89% compression](image3.png) ![99.99% compression](image4.png)

Figure 4. Different compression rates of the test image.

![Figure 5. Visualization of the samplet coefficients for the test image.](image5.png)

Three dimensions. Finally, we show a result in three dimensions. Here, the points are given by a uniform subsample of a triangulation of the Stanford bunny. We consider data on the Stanford bunny generated by the function

$$f(x) = e^{-20\|x - p_0\|^2} + e^{-20\|x - p_1\|^2},$$

where the points $p_0$ and $p_1$ are located at the tips of the bunny’s ears. Moreover, the geometry has been rescaled to a diameter of 2. The plot on the left-hand side of Figure 6 visualizes the sample data, while the plot on the right-hand side shows the dominant coefficients in case of a threshold parameter of $10^{-2}\|f^2\|_\infty$. 

![Plot of the sample data](image6.png) ![Plot of the dominant coefficients](image7.png)
6. Compression of kernel matrices

6.1. Kernel matrices. The second application of samplets we consider is the compression of matrices arising from positive (semi-) definite kernels, as they emerge in kernel methods, such as scattered data analysis, kernel based learning or Gaussian process regression, see for example \cite{25,38,42,43} and the references therein.

We start by recalling the concept of a positive kernel.

**Definition 6.1.** A symmetric kernel \( K : \Omega \times \Omega \rightarrow \mathbb{R} \) is called positive (semi-)definite on \( \Omega \subset \mathbb{R}^d \), iff \( [K(x_i, x_j)]_{i,j=1}^N \) is a symmetric and positive (semi-)definite matrix for all \( \{x_1, \ldots, x_N\} \subset \Omega \) and all \( N \in \mathbb{N} \).

As a particular class of positive definite kernels, we consider the Matérn kernels given by

\[
k_\nu(r) := 2^{1-\nu} \left( \frac{\sqrt{2\nu r}}{\ell} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu r}}{\ell} \right), \quad r \geq 0, \ \ell > 0.
\]

Herein, \( K_\nu \) is the modified Bessel function of the second kind of order \( \nu \) and \( \Gamma \) is the gamma function. The parameter \( \nu \) steers for the smoothness of the kernel function. Especially, the analytic squared-exponential kernel is retrieved for \( \nu \rightarrow \infty \). Especially, we have

\[
k_{1/2}(r) = \exp \left( -\frac{r}{\ell} \right), \quad k_\infty(r) = \exp \left( -\frac{r^2}{2\ell^2} \right).
\]

A positive definite kernel in the sense of Definition 6.1 is obtained by considering

\[
K(x, x') := k_\nu(||x - x'||_2).
\]

Given the set of points \( X = \{x_1, \ldots, x_N\} \), many applications require the assembly and the inversion of the kernel matrix

\[
K := [K(x_i, x_j)]_{i,j=1}^N \in \mathbb{R}^{N \times N}
\]

or an appropriately regularized version

\[
K + \rho I, \quad \rho > 0,
\]

thereof. In case that \( N \) is a large number, already the assembly and storage of \( K \) 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 \( x_i \) and \( x_j \) will be denoted by

\[
(K, \delta_{x_i} \otimes \delta_{x_j})_{\Omega \times \Omega} := K(x_i, x_j).
\]

Hence, in view of \( V = \{\delta_{x_1}, \ldots, \delta_{x_N}\} \), we may write the kernel matrix as

\[
K = \left[ (K, \delta_{x_i} \otimes \delta_{x_j})_{\Omega \times \Omega} \right]_{i,j=1}^N.
\]
6.2. Asymptotically smooth kernels. The essential ingredient for the samplet compression of kernel matrices is the asymptotical smoothness property of the kernel

\[
\frac{\partial^{\lvert \alpha \rvert + |\beta|}}{\partial x^\alpha \partial y^\beta} K(x, y) \leq c_K \frac{(\lvert \alpha \rvert + |\beta|)!}{r^{\lvert \alpha \rvert + |\beta|} \|x - y\|_2^{\lvert \alpha \rvert + |\beta|}}, \quad c_K, r > 0,
\]

which is for example satisfied by the Matérn kernels. Using this estimate, we obtain the following result, which is the basis for the matrix compression introduced thereafter.

**Lemma 6.2.** Consider two sampling kernels \(\sigma_{j,k}\) and \(\sigma_{j',k'}\), exhibiting \(q + 1\) vanishing moments with supporting clusters \(\nu\) and \(\nu'\), respectively. Assume that \(\text{dist}(\nu, \nu') > 0\). Then, for kernels satisfying \([17]\), it holds that

\[
(K, \sigma_{j,k} \otimes \sigma_{j',k'})_{\Omega \times \Omega} \leq c_K \frac{\text{diam}(\nu)^q \text{diam}(\nu')^{q + 1}}{(dr \text{ dist}(\nu_{j,k}, \nu'_{j',k'}))^{2(q + 1)} \|\omega_{j,k}\|_1 \|\omega_{j',k'}\|_1}.
\]

**Proof.** Let \(x_0 \in \nu\) and \(y_0 \in \nu'\). A Taylor expansion of the kernel with respect to \(x\) yields

\[
K(x, y) = \sum_{\lvert \alpha \rvert \leq q} \frac{\partial^{\lvert \alpha \rvert}}{\partial x^\alpha} K(x_0, y) \frac{(x - x_0)^\alpha}{\alpha!} + R_{x_0}(x, y),
\]

where the remainder \(R_{x_0}(x, y)\) is given by

\[
R_{x_0}(x, y) = (q + 1) \sum_{\lvert \alpha \rvert = q + 1} \frac{(x - x_0)^\alpha}{\alpha!} \int_0^1 \frac{\partial^{q + 1}}{\partial x^\alpha} K(x_0 + s(x - x_0), y) (1 - s)^q \, ds.
\]

Next, we expand the remainder \(R_{x_0}(x, y)\) with respect to \(y\) and derive

\[
R_{x_0}(x, y) = (q + 1) \sum_{\lvert \alpha \rvert = q + 1} \frac{(x - x_0)^\alpha}{\alpha!} \sum_{\lvert \beta \rvert \leq q} \frac{(y - y_0)^\beta}{\beta!} \int_0^1 \frac{\partial^{q + 1}}{\partial x^\alpha} \frac{\partial^{q + 1}}{\partial y^\beta} K(x_0 + s(x - x_0), y_0) (1 - s)^q \, ds + R_{x_0,y_0}(x, y).
\]

Here, the remainder \(R_{x_0,y_0}(x, y)\) is given by

\[
R_{x_0,y_0}(x, y) = (q + 1)^2 \sum_{\lvert \alpha \rvert, \lvert \beta \rvert = q + 1} \frac{(x - x_0)^\alpha (y - y_0)^\beta}{\alpha! \beta!} \int_0^1 \int_0^1 \frac{\partial^{2(q + 1)}}{\partial x^\alpha \partial y^\beta} K(x_0 + s(x - x_0), y_0 + t(y - y_0)) (1 - s)^q (1 - t)^q \, ds \, dt.
\]

We thus arrive at the decomposition

\[
K(x, y) = p_y(x) + p_x(y) + R_{x_0,y_0}(x, y),
\]

where \(p_y(x)\) is a polynomial of degree \(q\) in \(x\), with coefficients depending on \(y\), while \(p_x(y)\) is a polynomial of degree \(q\) in \(y\), with coefficients depending on \(x\). Due to the vanishing moments, we obtain

\[
(K, \sigma_{j,k} \otimes \sigma_{j',k'})_{\Omega \times \Omega} = (R_{x_0,y_0}, \sigma_{j,k} \otimes \sigma_{j',k'})_{\Omega \times \Omega}.
\]

In view of \([17]\), we thus find

\[
\|K, \sigma_{j,k} \otimes \sigma_{j',k'}\|_{\Omega \times \Omega} = \|R_{x_0,y_0}\|_{\Omega \times \Omega} \leq c_K \left( \sum_{\lvert \alpha \rvert, \lvert \beta \rvert = q + 1} \frac{\lvert \alpha \rvert + |\beta|)!}{\alpha! \beta!} \right) \left( \frac{\|x - y\|_2^{q + 1} + |\sigma_{j,k}|_{\Omega} (\|x - y\|_2^{q + 1} + |\sigma_{j',k'}|_{\Omega})}{r^{2(q + 1)} \text{ dist}(\nu, \nu')^{2(q + 1)}} \right).
\]

Next, we have by means of multinomial coefficients

\[
\|\alpha \rvert + |\beta|)! = \left( \frac{\lvert \alpha \rvert + |\beta|)!}{|\beta| \alpha!} \right) \left( \frac{|\beta|)!}{\alpha! \beta!}.
\]
which in turn implies that
\[
\sum_{|\alpha|,|\beta|=q+1} \frac{(|\alpha| + |\beta|)!}{\alpha!\beta!} = \left(\frac{2(q+1)}{q+1}\right) \sum_{|\alpha|,|\beta|=q+1} \left(\frac{|\alpha|}{\alpha}\right)^{\alpha} \left(\frac{|\beta|}{\beta}\right)^{\beta} = \left(\frac{2(q+1)}{q+1}\right)^d 2^{(q+1)} 2^{(q+1)}.
\]
Moreover, we use
\[
\|\cdot - x_0\|_{\Omega} \leq \left(\frac{\text{diam}(\nu)}{2}\right)^{q+1} \|\omega_{j,k}\|_1,
\]
and likewise
\[
\|\cdot - y_0\|_{\Omega} \leq \left(\frac{\text{diam}(\nu')}{2}\right)^{q+1} \|\omega_{j',k'}\|_1.
\]
Combining all the estimates, we arrive at the desired result (18). $\square$

6.3. Matrix compression. Lemma 6.2 immediately suggests a compression strategy for kernel matrices in samplet representation. We mention that this compression differs from the wavelet matrix compression introduced in [10], 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 consider a non-uniform distribution of the points in $V$. Consequently, we use on all levels the same accuracy, what is more similar to the setting in [4].

**Theorem 6.3.** Set all coefficients of the kernel matrix
\[
K^\Sigma := [(K, \sigma_{j,k} \odot \sigma_{j',k'})_{\Omega \times \Omega}]_{j,j',k,k'}
\]
to zero which satisfy
\[
\text{dist}(\nu, \nu') \geq \eta \max\{\text{diam}(\nu), \text{diam}(\nu')\}, \quad \eta > 0,
\]
where $\nu$ is the cluster supporting $\sigma_{j,k}$ and $\nu'$ is the cluster supporting $\sigma_{j',k'}$, respectively. Then, it holds
\[
\|K^\Sigma - K^\Sigma_c\|_F \leq c_K c_{\text{sum}} (\eta dr)^{-2(q+1)} m_q N \sqrt{\log(N)}.
\]
for some constant $c_{\text{sum}} > 0$, where $m_q$ is given by [5].

**Proof.** We first fix the levels $j$ and $j'$. In view of (18), we can estimate any coefficient which satisfies (19) by
\[
\|(K, \sigma_{j,k} \odot \sigma_{j',k'})_{\Omega \times \Omega}\| \leq c_K \left(\min\{\text{diam}(\nu), \text{diam}(\nu')\}\right)^{q+1} (\eta dr)^{-2(q+1)} \|\omega_{j,k}\|_1 \|\omega_{j',k'}\|_1.
\]
If we next set
\[
\theta_{j,j'} := \max_{\nu \in T_j, \nu' \in T_{j'}} \left\{ \frac{\min\{\text{diam}(\nu), \text{diam}(\nu')\}}{\max\{\text{diam}(\nu), \text{diam}(\nu')\}} \right\},
\]
then we obtain
\[
\|(K, \sigma_{j,k} \odot \sigma_{j',k'})_{\Omega \times \Omega}\| \leq c_K \theta_{j,j'}^{q+1} (\eta dr)^{-2(q+1)} \|\omega_{j,k}\|_1 \|\omega_{j',k'}\|_1
\]
for all coefficients such that (19) holds. In view of (12) and the fact that there are at most $m_q$ samplers per cluster, we arrive at
\[
\sum_{k,k'} \|\omega_{j,k}\|^2_1 \|\omega_{j',k'}\|^2_1 \leq \sum_{k,k'} |\nu| \cdot |\nu'| = m_q^2 N^2.
\]
Thus, for a fixed level-level block, we arrive at the estimate
\[
\|K^\Sigma_{j,j'} - K^\Sigma_{c,j,j'}\|^2_2 \leq c^2 K \theta_{j,j'}^{2(q+1)} (\eta dr)^{-4(q+1)} m_q^2 N^2.
\]
Finally, summation over all levels yields
\[
\| K^\Sigma - K^\Sigma_{\xi} \|_F^2 = \sum_{j \leq j'} \| K^\Sigma_{j,j'} - K^\Sigma_{\xi,j,j'} \|_F^2 \\
\leq c_k^2 (\eta d r)^{-4(q+1)} m_q^2 N^2 \sum_{j,j'} t^{2(q+1)} \\
\leq c_k^2 c_{\text{sum}} (\eta d r)^{-4(q+1)} m_q^2 N^2 \log N,
\]
which is the desired claim. \(\square\)

**Corollary 6.4.** In case of uniformly distributed points \(x_i \in X\), we have \(\| K^\Sigma \|_F \sim N\). Thus, we immediately obtain
\[
\frac{\| K^\Sigma - K^\Sigma_{\xi} \|_F}{\| K^\Sigma \|_F} \leq c_k \sqrt{c_{\text{sum}} (\eta d r)^{-2(q+1)} m_q \sqrt{\log N}}.
\]
In particular, the matrix can be compressed to \(O(m_q^2 N \log N)\) remaining coefficients without compromising the overall accuracy.

**Proof.** We fix \(j, j'\) and assume \(j \geq j'\). In case of uniformly distributed points, it holds \(\text{diam}(\nu) \sim 2^{-j} d / q\). Hence, for the cluster \(\nu_{j,j'}\), there exist only \(O(2^{j-j'})\) clusters \(\nu_{j,k}\) from level \(j\), which do not satisfy the cut-off criterion \(\text{19}\). Since each cluster contains at most \(m_q\) sampels, we hence arrive at
\[
\sum_{j=0}^{j} \sum_{j' \leq j} m_q^2 (2^{j-2(j'-1)} d)^d = m_q^2 \sum_{j=0}^{j} 2^{j^2} d^2 \sim m_q^2 N \log N,
\]
which implies the assertion. \(\square\)

**Remark 6.5.** The chosen cut-off criterion \(\text{19}\) coincides with the so called admissibility condition used by hierarchical matrices. We particularly refer here to \(\text{17}\), as we will later on rely the \(H^2\)-matrix method presented there for the fast assembly of the compressed kernel matrix.

### 6.4. Compressed matrix assembly

For a given pair of clusters, we can now determine whether the corresponding entries need to be calculated. As there are \(O(N)\) clusters, naively checking the cut-off criterion for all pairs would still take \(O(N^2)\) 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 \(\text{10}\) for a proof.

**Lemma 6.6.** Let \(\nu\) and \(\nu'\) be clusters satisfying the cut-off criterion \(\text{19}\). Then, for the son clusters \(\nu_{\text{son}}\) of \(\nu\) and \(\nu'_{\text{son}}\) of \(\nu'\), we have
\[
\text{dist}(\nu, \nu'_{\text{son}}) \geq \eta \max\{\text{diam}(\nu), \text{diam}(\nu'_{\text{son}})\},
\]
\[
\text{dist}(\nu_{\text{son}}, \nu') \geq \eta \max\{\text{diam}(\nu_{\text{son}}), \text{diam}(\nu')\},
\]
\[
\text{dist}(\nu_{\text{son}}, \nu'_{\text{son}}) \geq \eta \max\{\text{diam}(\nu_{\text{son}}), \text{diam}(\nu'_{\text{son}})\}.
\]

The lemma tells us that we may omit cluster pairs whose father clusters already satisfy the cut-off criterion. 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 \(H^2\)-matrix techniques, see \(\text{17,21}\). Similarly to \(\text{1,22,26}\), we shall rely here on \(H^2\)-matrices for this purpose. The idea of \(H^2\)-matrices is to approximate the kernel interaction for sufficiently distant clusters \(\nu\) and \(\nu'\) in the sense of the admissibility condition \(\text{19}\) by means of the interpolation based \(H^2\)-matrix approach. More precisely, given a suitable set of interpolation points \(\{\xi_t^\nu\}_t\) for each cluster \(\nu\) with associated Lagrange polynomials \(\{L_t^\nu(x)\}_t\), we introduce the interpolation operator
\[
T^{\nu,\nu'}[K](x, y) = \sum_{s,t} K(\xi_s^{\nu}, \xi_t^{\nu'}) L_s^\nu(x) L_t^\nu(y).
\]
and approximate an admissible matrix block via
\[
K_{\nu',\nu}^{\Delta} = [K, \delta_x \otimes \delta_y]_{x \in V, y \in V'} \approx \sum_{s,t} K(\xi_s^\nu, \xi_t^\nu) [L_s^\nu, \delta_x]_{x \in V'} [L_t^\nu, \delta_y]_{y \in V'} =: V_{\Delta}^{\nu} S_{\nu',\nu}^{\nu'} (V_{\Delta}^{\nu})^\top.
\]

Herein, the cluster bases are given according to
\[
V_{\Delta}^{\nu} := [L_s^\nu, \delta_x]_{x \in V}, \quad V_{\Delta}^{\nu'} := [L_t^\nu, \delta_y]_{y \in V'},
\]
while the coupling matrix is given by \( S_{\nu',\nu}^{\nu'} := [K(\xi_s^\nu, \xi_t^\nu)]_{s,t} \).

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 \( H^2 \)-matrices. For the sake of simplicity, we assume from now on that tensor product polynomials of degree \( p \) are used for the kernel interpolation at all different cluster combinations. As a consequence, the Lagrange polynomials of a father cluster can exactly be represented by those of the son clusters. Introducing the transfer matrices \( T_{\nu',\nu}^{\nu} := [L_s^\nu(\xi_t^\nu)]_{s,t} \), there holds
\[
L_s^\nu(x) = \sum_t T_{\nu',\nu}^{\nu}(x), \quad x \in B_{\nu,\nu'}.
\]

Exploiting this relation in the construction of the cluster bases (20) finally leads to
\[
V_{\nu}^{\nu'} = \begin{bmatrix} V_{\Delta}^{\nu_{\text{son}1}} T_{\nu,\nu_{\text{son}1}}^{\nu'} \\ V_{\Delta}^{\nu_{\text{son}2}} T_{\nu,\nu_{\text{son}2}}^{\nu'} \end{bmatrix}.
\]

Combining this refinement relation with the recursive nature of the samplet basis, results in the variant of the discrete samplet transform summarized in Algorithm 6.1.

---

**Algorithm 6.1:** Recursive computation of the multiscale cluster basis

**Data:** Cluster tree \( T \), transformations \( [Q_{s,\nu}, Q_{t,\nu}^\top] \); nested cluster bases \( V_{\Delta}^{\nu} \) for leaf clusters and transformation matrices \( T_{\nu',\nu}^{\nu} \) for non-leaf clusters.

**Result:** Multiscale cluster basis matrices \( V_{\nu}^{\nu} \), \( V_{\Delta}^{\nu} \) for all clusters \( \nu \in T \).

begin
\[
\text{computeMultiscaleClusterBasis}(X);
\]

Function computeMultiscaleClusterBasis(\( \nu \))

begin
\[
\text{if } \nu \text{ is a leaf cluster then store } V_{\nu}^{\nu} := [Q_{s,\nu}, Q_{t,\nu}^\top]^\top V_{\Delta}^{\nu};
\]
\[
\text{else for all sons } \nu' \text{ of } \nu \text{ do computeMultiscaleClusterBasis}(\nu')
\]
\[
\text{store } V_{\nu}^{\nu} := [Q_{s,\nu}, Q_{t,\nu}^\top]^\top V_{\Delta}^{\nu_{\text{son}1}} T_{\nu,\nu_{\text{son}1}}^{\nu'} V_{\nu_{\text{son}2}}^{\nu_{\text{son}2}} T_{\nu,\nu_{\text{son}2}}^{\nu'};
\]

end

Having the multiscale cluster bases at our disposal, the next step is the assembly of the compressed kernel matrix. The computation of the required matrix blocks is exclusively based on the two refinement relations
\[
\begin{bmatrix}
K_{\nu',\nu}^{\nu} \\ K_{\nu',\nu}^{\nu} 
\end{bmatrix} =
\begin{bmatrix}
K_{\nu,\nu_{\text{son}1}}^{\nu} & K_{\nu,\nu_{\text{son}2}}^{\nu} \\ K_{\nu,\nu_{\text{son}1}}^{\nu} & K_{\nu,\nu_{\text{son}2}}^{\nu}
\end{bmatrix} \cdot \begin{bmatrix}
Q_{s,\nu}^{\nu} & Q_{t,\nu}^{\nu} \\ Q_{s,\nu}^{\nu} & Q_{t,\nu}^{\nu}
\end{bmatrix},
\]

and
\[
\begin{bmatrix}
K_{\nu',\nu}^{\nu} \\ K_{\nu',\nu}^{\nu} 
\end{bmatrix} =
\begin{bmatrix}
K_{\nu,\nu_{\text{son}1}}^{\nu} & K_{\nu,\nu_{\text{son}2}}^{\nu} \\ K_{\nu,\nu_{\text{son}1}}^{\nu} & K_{\nu,\nu_{\text{son}2}}^{\nu}
\end{bmatrix} \cdot \begin{bmatrix}
Q_{s,\nu}^{\nu} & Q_{t,\nu}^{\nu} \\ Q_{s,\nu}^{\nu} & Q_{t,\nu}^{\nu}
\end{bmatrix},
\]

where we set
\[
\begin{bmatrix}
K_{\nu',\nu}^{\nu} \\ K_{\nu',\nu}^{\nu} 
\end{bmatrix} := \begin{bmatrix}
(K, \Phi^{\nu} \otimes \Phi^{\nu})_{\Omega \times \Omega} & (K, \Phi^{\nu} \otimes \Sigma^{\nu})_{\Omega \times \Omega} \\
(K, \Sigma^{\nu} \otimes \Phi^{\nu})_{\Omega \times \Omega} & (K, \Sigma^{\nu} \otimes \Sigma^{\nu})_{\Omega \times \Omega}
\end{bmatrix}.
\]
We obtain the following function, which is the key ingredient for the computation of the compressed kernel matrix.

\begin{verbatim}
Function recursivelyDetermineBlock(ν, ν')
Result: Approximation of the block \(\begin{bmatrix} K^{\phi,\phi}_{\nu',\nu} & K^{\phi,\Sigma}_{\nu',\nu} \\ K^{\Sigma,\phi}_{\nu,\nu'} & K^{\Sigma,\Sigma}_{\nu,\nu'} \end{bmatrix}\).
begin
  if (ν, ν') is admissible then
    return \(V^e_{\nu'} S^{\nu',\nu} (V^e_{\nu})^T, (V^e_{\nu})^T\)
  else if ν and ν' are leaf clusters then
    return \(Q'_{\nu',\nu} \Sigma_{\nu,\nu'} Q'_{\nu,\nu'}^T\)
  else if ν' is not a leaf cluster and ν is a leaf cluster then
    for all sons ν'' of ν' do
      \[\begin{bmatrix} K^{\phi,\phi}_{\nu'',\nu} & K^{\phi,\Sigma}_{\nu'',\nu} \\ K^{\Sigma,\phi}_{\nu,\nu''} & K^{\Sigma,\Sigma}_{\nu,\nu''} \end{bmatrix} := recursivelyDetermineBlock(ν, ν'')\]
    return \(K^{\phi,\phi}_{\nu''},\nu' \Sigma_{\nu,\nu''} K^{\phi,\Sigma}_{\nu'',\nu} \)
  else if ν is not a leaf cluster and ν' is a leaf cluster then
    for all sons ν'' of ν do
      \[\begin{bmatrix} K^{\phi,\phi}_{\nu,\nu''} & K^{\phi,\Sigma}_{\nu,\nu''} \\ K^{\Sigma,\phi}_{\nu'',\nu} & K^{\Sigma,\Sigma}_{\nu'',\nu} \end{bmatrix} := recursivelyDetermineBlock(ν'', ν')\]
    return \(Q'_{\nu'',\nu} K^{\phi,\phi}_{\nu,\nu''},\nu' \Sigma_{\nu'',\nu} K^{\phi,\Sigma}_{\nu,\nu''}\)
  else
    for all sons ν'' of ν and all sons ν''' of ν'
      \[\begin{bmatrix} K^{\phi,\phi}_{\nu,\nu''} & K^{\phi,\Sigma}_{\nu,\nu''} \\ K^{\Sigma,\phi}_{\nu'',\nu} & K^{\Sigma,\Sigma}_{\nu'',\nu} \end{bmatrix} := recursivelyDetermineBlock(ν'', ν''', ν''')\]
    return \(Q'_{\nu,\nu''} K^{\phi,\phi}_{\nu'',\nu''},\nu' \Sigma_{\nu'',\nu''} K^{\phi,\Sigma}_{\nu',\nu''}\)
end
\end{verbatim}

We remark that the algorithm never requires the formation of the entire \(H^2\)-matrix, as it only embeds the multilevel interpolation procedure to rapidly evaluate admissible blocks. In particular, the evaluation of the coupling matrices can be performed on the fly.

Now, in order to assemble the compressed kernel matrix, we require two nested recursive calls of the cluster tree, which is traversed in a depth first search way. Algorithm 6.2 first computes the lower right matrix block and advances from bottom to top and from right to left. To this end, the two recursive functions setupColumn and setupRow are introduced.

\begin{verbatim}
Algorithm 6.2: Computation of the compressed kernel matrix
Data: Cluster tree \(\mathcal{T}\), multiscale cluster bases \(V^e_{\phi}, V^e_{\Sigma}\) and transformations \(\{Q'_{\phi,\nu}, Q'_{\Sigma,\nu}\}\).
Result: Sparse matrix \(K^\Sigma_{\nu}\)
begin
  setupColumn(X)
    store the blocks remaining blocks \(K^\Sigma_{\nu,X}\) for \(\nu \in \mathcal{T} \setminus \{X\}\) in \(K^\Sigma_{\nu}\) (they have already been computed by earlier calls to recursivelyDetermineBlock)
end
\end{verbatim}

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.

\begin{verbatim}
Function setupColumn(ν')
begin
  for all sons ν'' of ν' do
    setupColumn(ν''')
  store \(K^\Sigma_{\nu,X,\nu'} := setupRow(X, ν')\) in \(K^\Sigma_{\nu}\)
end
\end{verbatim}
For a given column cluster \( \nu' \), the function \texttt{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.

\begin{algorithm}
\textbf{Function} \texttt{setupRow}(\nu, \nu')
\begin{algorithmic}
\Procedure{}{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}
\Comment{\( K_{\nu_{\text{son}},\nu'}^{\Phi,\Sigma} \) := \texttt{setupRow}(\( \nu_{\text{son}}, \nu' \))}
\Else
\Comment{\( K_{\nu_{\text{son}},\nu'}^{\Phi,\Sigma} := \texttt{recursivelyDetermineBlock}(\nu_{\text{son}}, \nu') \)}
\EndIf
\EndFor
\Elsif {\( \nu' \) is a leaf cluster then}
\Comment{\( K_{\nu_{\text{son}},\nu'}^{\Phi,\Sigma} := \texttt{recursivelyDetermineBlock}(\nu_{\text{son}}, \nu') \)}
\Else
\For {all sons \( \nu'_{\text{son}} \) of \( \nu' \) do}
\If {\( \nu \) and \( \nu'_{\text{son}} \) are not admissible then}
\Comment{\( K_{\nu_{\text{son}},\nu'}^{\Phi,\Sigma} := \texttt{recursivelyDetermineBlock}(\nu_{\text{son}}, \nu') \)}
\Else
\Comment{\( K_{\nu_{\text{son}},\nu'}^{\Phi,\Sigma} := \texttt{recursivelyDetermineBlock}(\nu, \nu'_{\text{son}}) \)}
\EndIf
\EndFor
\EndIf
\EndIf
\Comment{\( K_{\nu,\nu'}^{\Phi,\Sigma} \) as part of \( \Sigma_{\nu} \)}
\Return {\( K_{\nu,\nu'}^{\Phi,\Sigma} \)}
\Comment{\( \Sigma_{\nu} \)}
\EndProcedure
\end{algorithmic}
\end{algorithm}

\textbf{Remark 6.7.} Algorithm [6.3] has a cost of \( O(N \log N) \) and requires an additional storage of \( O(N \log N) \) if all stored blocks are directly released when they are not required anymore. We refer to [7] for all the details.

7. Numerical results II

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\footnote{The full specifications can be found on https://www.euler.usi.ch/en/research/resources.}. In order to obtain consistent timings, only a single core was used for all computations.

\textbf{Benchmark problem.} To benchmark the compression of kernel matrices, we consider the exponential kernel
\[ k(x, y) = e^{-100||x-y||_2}, \]
evaluated at an increasing number of non-uniformly distributed cloud of point samples. Namely, in \( d = 1 \) dimension, we consider standard normally distributed points. In \( d > 1 \) dimensions, the random sample points are drawn from the mixture of two multivariate Gaussian distributions with
zero expectation covariances
\[
\begin{bmatrix}
1 & -1/2 & 0 \\
-1/2 & 29/100 & 0 \\
0 & 0 & 1
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
1 & 1/2 & 0 \\
1/2 & 29/100 & 0 \\
0 & 0 & 1
\end{bmatrix}.
\]

Note that the last coordinate is dropped if \( d = 2 \). The resulting data sets are visualized in Figure 7 with the corresponding bounding boxes of the domain and of the tree leaves. For \( d = 2 \), the bounding box is given by \([-5.21, 4.56] \times \[-2.50, 2.48]\), while it is given by \([-5.21, 4.56] \times \[-2.50, 2.48] \times \[-4.81, 4.84]\) for \( d = 3 \). As can be seen, the points have a much higher density at the center of the point cloud, which results in an adaptively refined cluster tree.

As a measure of sparsity, we introduce the average number of nonzeros per row
\[
\text{anz}(A) := \frac{\text{nnz}(A)}{N}, \quad A \in \mathbb{R}^{N \times N},
\]
where \( \text{nnz}(A) \) is the number of nonzero entries of \( A \). Besides the compression, we also report the fill-in generated by the Cholesky factorization in combination with the nested dissection reordering from [27]. For the reordering and the Cholesky factorization, we rely on MATLABR2020a\(^2\), while the samplet compression is implemented in C++11 using the Eigen template library\(^3\) for linear algebra operations. For the computations, we consider a polynomial degree of 3 for the kernel interpolation and \( q + 1 = 3 \) vanishing moments for the samplets. We set \( \eta = 2 \) for \( d = 1 \), \( \eta = 1.25 \) for \( d = 2 \) and \( \eta = 0.5 \) for \( d = 3 \). In addition, we have performed a thresholding of the computed matrix coefficients that were smaller than \( \varepsilon = 10^{-5} \).

Figure 8 shows the resulting relative compression errors, which have been computed by estimating the Frobenius norm from 20 randomly chosen columns of \( K^\Sigma \) and \( K^\Sigma_{\varepsilon} \), respectively. As can be seen, for all dimensions under consideration, the compression errors roughly follow the theoretical rate of \( \varepsilon \log N \).

\(^2\)Version 9.8.0.1396136, The MathWorks Inc., Natick, Massachusetts, 2020.
\(^3\)https://eigen.tuxfamily.org/
The left-hand side of Figure 9 shows the wall time for the assembly of the compressed kernel matrices. The different dashed lines indicate the asymptotics $N \log^\alpha N$ for $\alpha = 0, 1, 2, 3$. For increasing number $N$ of points and the dimensions $d = 1, 2, 3$ under consideration, all computation times approach the expected rate of $N \log N$. The right-hand side of Figure 9 shows the average number of nonzeros per row for an increasing number $N$ of points. This number becomes constant or even slightly decreases, as expected.

Next, we examine the Cholesky factorization of the compressed kernel matrix. As the largest eigenvalue of the kernel matrix grows proportionally to the number $N$ of points, while the smallest eigenvalue is given by the ridge parameter, the condition number grows with $N$ as well. Hence, to obtain a constant condition number for increasing $N$, the ridge parameter needs to be adjusted accordingly. However, as we are only interested in the generated fill-in and the computation times, we neglect this fact and just fix the ridge parameter to $\rho = 1$ for all considered $N$ and $d = 1, 2, 3$. The obtained results are found in Figure 10. Herein, on the left-hand side, the wall times for the Cholesky factorization of the reordered matrix are found. For $d = 1$, the average number of nonzeros per row becomes constant when the number $N$ of points increases. This indicates that the kernel function is already fully resolved up to the threshold parameter on the coarser levels. For $d = 2$, the observed rate is slightly worse than the expected one of $N^{3/2}$ for the Cholesky factorization, which is caused by the high connectivity of the associated graph. Asymptotically, the expected rate seems to be achieved. Likewise, for $d = 3$, one figures out the rate $N^{2-3}$ in contrast to the expected rate $N^2$. This is again caused by the high connectivity of the associated graph. On the right-hand side of the same figure, it can be seen that the fill-in remains rather moderate. A visualization of the matrix patterns for the matrix $K_\Sigma + \rho I$, the reordered matrix and the Cholesky factor for $N = 131,072$ points is shown in Figure 11. Each dot corresponds to a
block of $256 \times 256$ matrix entries and its intensity indicates the number of nonzero entries, where
darker blocks contain more entries than lighter blocks.

![Sparsity pattern](image)

**Figure 11.** Sparsity pattern of $K^\Sigma$, reordered matrices (middle) and the Cholesky factors $L$ (right) for $d = 1, 2, 3$ and $N = 131072$.

**Simulation of a Gaussian random field.** As our last example, we consider a Gaussian random
field evaluated at 100,000 randomly chosen points at the surface of the Stanford bunny. As before,
the Stanford bunny has been rescaled to have a diameter of 2. In order to demonstrate that our
approach works also for larger dimensions, the Stanford bunny has been embedded into $\mathbb{R}^4$ and
randomly rotated to prevent axis-aligned bounding boxes. The polynomial degree for the $H^2$-
matrix representation is set to 3 as before and likewise we consider $q + 1 = 3$ vanishing moments.
The covariance function is given by the exponential kernel

$$k(x, y) = e^{-25 \|x-y\|^2}.$$  

Moreover, we discard all computed matrix entries which are below the threshold of $\varepsilon = 10^{-6}$.
The ridge parameter is set to $\rho = 10^{-2}$. The compressed covariance matrix exhibits $\text{anz}(K^\Sigma) = 6457$ nonzero matrix entries per row on average, while the corresponding Cholesky factor exhibits $\text{anz}(L) = 14,898$ nonzero matrix entries per row on average. Having the Cholesky factor $L$ at hand,
the computation of a realization of the Gaussian random field is extremely fast, as it only requires
a simple sparse matrix-vector multiplication of $L$ by a Gaussian random vector and an inverse
samplet transform. Four different realizations of the random field projected to $\mathbb{R}^3$ are shown in
Figure 12.

8. **Conclusion**

Samplets provide a new methodology for the analysis of large data sets. They are easy to
construct and discrete data can be transformed into the samplet basis in 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 is 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 kernel matrices to obtain an essentially sparse matrix. 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 matrices becomes computationally feasible, which 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 $10^6$ 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 $H^2$-wavelet method. J. Comput. Appl. Math., 267:131–159 (2014).
[2] B. Alpert. A class of bases in $L^2$ for the sparse representation of integral operators. SIAM J. Math. Anal., 24(1), 247–262 (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 algorithms. Comm. Pure Appl. Math., 44:141–183 (1991).
[5] S. Börm. Efficient numerical methods for non-local operators: $H^2$-matrix compression, algorithms and analysis. European Mathematical Society, Zürich, 2010.
[6] C.K. Chui. An Introduction to Wavelets. Academic Press, San Diego (CA), 1992.
[7] C.K. Chui and E. Quak. Wavelets on a bounded interval. Numer. Meth. Approx. Theory, 9:53–75 (1992).
[8] R.R. Coifman and M. Maggioni. Diffusion wavelets. Appl. Comput. Harmon. Anal., 21:53–94 (2006).
[9] W. Dahmen. Wavelet and multiscale methods for operator equations. Acta Numerica, 6:55–228 (1997).
[10] W. Dahmen, H. Harbrecht, and R. Schneider. Compression techniques for boundary integral equations. Optimal complexity estimates. SIAM J. Numer. Anal., 43:2251–2271 (2006).
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