Institut für Numerische Simulation
Rheinische Friedrich-Wilhelms-Universität Bonn

Wegelerstraße 6 • 53115 Bonn • Germany
phone +49 228 73-3427 • fax +49 228 73-7527
www.ins.uni-bonn.de

B. Bohn, M. Griebel

An adaptive sparse grid approach for time series prediction

INS Preprint No. 1201

January 2012

AN ADAPTIVE SPARSE GRID APPROACH FOR TIME SERIES PREDICTION

Bastian Bohn and Michael Griebel

Institute for Numerical Simulation, University of Bonn, 53115 Bonn, Germany
{bohn},{griebel}@ins.uni-bonn.de

Summary. A real valued, deterministic and stationary time series can be embedded in a — sometimes high-dimensional — real vector space. This leads to a one-to-one relationship between the embedded, time dependent vectors in $\mathbb{R}^d$ and the states of the underlying, unknown dynamical system that determines the time series. The embedded data points are located on an m-dimensional manifold (or even fractal) called attractor of the time series. Takens' theorem then states that an upper bound for the embedding dimension d can be given by $d \leq 2m + 1$.
The task of predicting future values thus becomes, together with an estimate on the manifold dimension m, a scattered data regression problem in d dimensions. In contrast to most of the common regression algorithms like support vector machines (SVMs) or neural networks, which follow a data-based approach, we employ in this paper a sparse grid-based discretization technique. This allows us to efficiently handle huge amounts of training data in moderate dimensions. Extensions of the basic method lead to space- and dimension-adaptive sparse grid algorithms. They become useful if the attractor is only located in a small part of the embedding space or if its dimension was chosen too large.
We discuss the basic features of our sparse grid prediction method and give the results of numerical experiments for time series with both, synthetic data and real life data.

Introduction and problem formulation

One of the most important tasks in the field of data analysis is the prediction of future values from a given time series of data. In our setting, a time series $(s_j)_{j=1}^{\infty}$ is an ordered set of real values. The task of forecasting can now be formulated as:

Given the values $s_1, \ldots, s_N$, predict s_{N+1}!

To tackle the forecasting problem, we assume that the values s_j stem from an underlying stationary process which evolves in time. The aim is then to reconstruct the domain of this process as good as possible from the data $s_1, \ldots, s_N$ and to use this reconstruction for the prediction of the value s_{N+1}. To this end, let M_0 represent the phase space of the underlying system, let $\phi : M_0 \to M_0$ denote the corresponding

equations of motion and let $o : M_0 \to \mathbb{R}$ be an observable which defines a time series by

$$(s_j)_{j=1}^{\infty} = \left(o\left(\phi^j\left(\mathbf{x}_0\right)\right)\right)_{j=1}^{\infty} , \tag{1}$$

where $\mathbf{x}_0 \in M_0$ is an arbitrary initial condition of the process and

$$\phi^j = \underbrace{\phi \circ \phi \circ \ldots \circ \phi}_{j \text{ times}} .$$

In practice M_0, ϕ and o are of course not known, but only the values s_j of the time series are given. To establish a situation in which we are able to tackle the forecasting problem, we need to find a connection between the past values of the time series and the next one.

Takens' theorem [2, 23] provides the theoretical background to construct algorithms for this purpose. Assuming that a given equidistant time series is a measurement o of an m-dimensional process which follows some deterministic equation of motion ϕ, there is the possibility to find a regular m-dimensional submanifold U of $\mathbb{R}^{2m+1}$ which is diffeomorphic to the phase space M_0 of the underlying system. The most common construction of such a submanifold works via delay embedding. In this case the time-dependent observations s_j are themselves used as coordinates to represent $U \subset \mathbb{R}^{2m+1}$. Here, since an element $\mathbf{x} \in U$ corresponds to one specific point in phase space, the dynamics of the process and thus the evolution of the time series itself are completely determined by $\mathbf{x}$ and the forecasting problem translates into an ordinary regression-like task in $\mathbb{R}^{2m+1}$.

In this paper we use a regularized least squares approach to find an adequate approximation to the solution of the prediction problem. In combination with a FEM-like grid discretization this leads to a non-data-based approach whose computational costs grow only linearly in the number of elements of the given time series. Then, in contrast to most data-based techniques like support vector machines or standard neural networks using radial basis functions, a discretization-based approach is able to handle huge time series. But, if $d = 2m + 1$ denotes the dimension of the ambient space and 2^t is the number of grid points in one direction, the number of points in a conventionally discretized ambient space would grow like $O\left(2^{td}\right)$. Thus, this naive approach suffers from the curse of dimensionality which restricts the application of a conventional discretization to low-dimensional, i.e. to one-, two- or three-dimensional spaces.

To circumvent this problem the sparse grid discretization [1] is used in this paper. For *regular sparse grids*, the number of grid points increases only like $O(2^t \cdot t^{d-1})$, i.e. the curse of dimensionality is now just present with respect to the term t. This way, we are able to efficiently deal with huge amounts of data in moderate dimensions up to about $d = 10$. Moreover, for most time series the high-dimensional data does not fill the whole space. The process obtained by using the delay embedding method then visits only a small fraction of the whole discretized area. This observation justifies a *space-adaptive sparse grid* [11] discretization which resolves the trajectory of the process. Finally, an ANOVA-like approach leads to *dimension-adaptive sparse grids* [8, 15] that are useful if our a priori choice of d is too large.

Thus, we will introduce two different adaptive algorithms in this paper: The space-adaptive algorithm locally adapts to features of the prediction function whereas the dimension-adaptive algorithm refines by employing subspaces which are relevant for

an efficient representation of the prediction function in its ANOVA-decomposition. In summary, each of our algorithms processes the following steps:

1. Estimation of the dimension m of the underlying process
2. Rewriting the forecasting problem as a regression problem in $\mathbb{R}^d$ with $d = 2m+1$
3. Approximating the solution of the regression problem in a discretized (regular, space-adaptive or dimension-adaptive) sparse grid space
4. Predicting the value s_{N+1} by point evaluation of the computed sparse grid function at $(s_{N-2m}, \ldots, s_N)^T$

Altogether, we obtain a new class of algorithms for the prediction of time series data which scale only linearly with the length of the given time series, i.e. the amount of data points, but still allow us to use reasonably large window sizes for the delay embedding due to our sparse grid approach. The new methods give excellent prediction results with manageable computational costs.
The remainder of this paper is organized as follows: In section 1, we describe the delay embedding scheme and review some crucial issues concerning the application of Takens' theorem. In section 2, we show how the forecasting problem can be rewritten as a regression problem. We also derive the regularized least squares functional which determines our predictor function. In section 3, we deal with the regular sparse grid approximation. We deduce the associated linear system and solve it using a preconditioned CG-algorithm. Then, we introduce and discuss space- and dimension-adaptive sparse grid algorithms. In section 4, we give the results of numerical experiments which illustrate the favorable properties of our new methods.

1 Takens' theorem and the delay embedding scheme

We now provide the essential theory concerning Takens' theorem [23] and give a hint to some modifications from [2].
For an arbitrary $d \in \mathbb{N}$ we can create vectors

$$\mathbf{t}_j := (s_{j-d+1}, s_{j-d+2}, \ldots, s_{j-1}, s_j)^T \in \mathbb{R}^d,\ j \geq d$$

following the so-called delay embedding scheme. Each vector consists of d consecutive past time series values. A connection between these delay vectors and the unknown evolution of the process in the phase space is established by the following theorem:

Theorem 1. *Let M_0 be a compact m-dimensional C^2-manifold, let $\phi : M_0 \to M_0$ denote a C^2-diffeomorphism and let $o \in C^2(M_0, \mathbb{R})$. Then, $\boldsymbol{\rho}_{(\phi,o)} : M_0 \hookrightarrow \mathbb{R}^{2m+1}$ defined by*

$$\boldsymbol{\rho}_{(\phi,o)}(\mathbf{x}) := \left(o(\mathbf{x}), o(\phi(\mathbf{x})), o\left(\phi^2(\mathbf{x})\right), \ldots, o\left(\phi^{2m}(\mathbf{x})\right)\right) \tag{2}$$

is generically[1] *an embedding.*

[1] Here, "generically" means the following:
If $X_l := \left\{\mathbf{x} \in M_0 \mid \phi^l(\mathbf{x}) = \mathbf{x}\right\}$ fulfills $|X_l| < \infty$ for all $l \leq 2m$ and if the Jacobian matrix $\left(\mathrm{D}\phi^l\right)_{\mathbf{x}}$ of ϕ^l at $\mathbf{x}$ has pairwise distinct eigenvalues for all $l \leq 2m, \mathbf{x} \in X_l$, then the set of all $o \in C^2(M_0, \mathbb{R})$ for which the embedding property of Theorem 1 does not hold is a null set. As $C^2(M_0, \mathbb{R})$ is an infinite dimensional vector space, the term "null set" may not be straightforward. It should be understood in the way that every set $Y \supset \left\{o \in C^2(M_0, \mathbb{R}) \mid \boldsymbol{\rho}_{(\phi,o)} \text{ is an embedding } \right\}$ is prevalent.

This is Takens' theorem for discrete time series, see [17, 23]. The embedding $\boldsymbol{\rho}_{(\phi,o)}$ establishes a one-to-one connection between a state in the phase space M_0 and a $(2m+1)$-dimensional delay vector constructed by (2). It can formally be inverted and we obtain

$$\begin{aligned}\phi^{j-2m}(\mathbf{x}_0) &= \boldsymbol{\rho}^{-1}_{(\phi,o)}\left(\left(o\left(\phi^{j-2m}(\mathbf{x}_0)\right), o\left(\phi^{j-2m+1}(\mathbf{x}_0)\right), \ldots, o\left(\phi^{j}(\mathbf{x}_0)\right)\right)\right)\\ &= \boldsymbol{\rho}^{-1}_{(\phi,o)}\left((s_{j-2m}, s_{j-2m+1}, \ldots, s_j)\right)\\ &= \boldsymbol{\rho}^{-1}_{(\phi,o)}(\mathbf{t}_j)\end{aligned}$$

for all $j \geq d$ with $\mathbf{t}_j \in \mathbb{R}^{2m+1}$ and thus $d = 2m+1$. Applying $o \circ \phi^{2m+1}$ on both sides we obtain

$$o\left(\phi^{j+1}(\mathbf{x}_0)\right) = o\left(\phi^{2m+1}\left(\boldsymbol{\rho}^{-1}_{(\phi,o)}(\mathbf{t}_j)\right)\right). \tag{3}$$

This means that the value $s_{j+1} = o\left(\phi^{j+1}(\mathbf{x}_0)\right)$ is completely determined by the previous $2m+1$ values $s_{j-2m}, \ldots, s_j$.[2] Note that not necessarily all of the preceding $2m+1$ values are essential to specify the current one, but Theorem 1 states that $2m+1$ values are always sufficient to do so. Note furthermore that not only the next but all following values are determined by $2m+1$ consecutive time series values. To see this one can just recursively follow the scheme in (3). Thus, if we have for example an equidistant time series with a one minute gap between successive values but are interested in a fifteen minute forecast, we can still use $2m+1$ consecutive values as input in our regression algorithm later on.[3]

Often, the equations of motion are described by a system of time-continuous differential equations instead of a time-discrete mapping ϕ as in Theorem 1. To this end, let V denote a vector field in $C^2(M_0, TM_0)$, let $o \in C^2(M_0, \mathbb{R})$ and let $\mathbf{z} : \mathbb{R}^+ \to M_0$ fulfill the differential equation

$$\frac{\mathrm{d}\mathbf{z}}{\mathrm{d}t} = \mathbf{V}(\mathbf{z}),\ \mathbf{z}(0) = \mathbf{z}_0 \tag{4}$$

for given $\mathbf{z}_0 \in M_0$. We define $\phi_t(\mathbf{z}_0) := \mathbf{z}(t)$ as the flow of the vector field $\mathbf{V}$. Now ϕ_τ can be used in Theorem 1 instead of ϕ for an arbitrary $\tau \in \mathbb{R}^+$ and the time-continuous setting is covered as well. For a thorough treatment of this case we refer to [2, 23].

A main requirement for Takens' theorem is the compactness of the manifold M_0, i.e. the domain of the process which contains all possible states. Sometimes the dynamics tends to form a so-called "strange attractor", which means that the trajectories of the system do not form a manifold anymore but just a point set A of non-integer dimension. In [2] it was shown that it is possible to generalize Theorem 1 also to this case:

[2] All functions on the right hand side of (3) are at least twice differentiable. As M_0 is compact, the concatenation of these functions lies in the standard Sobolev space $H_2(\boldsymbol{\rho}_{(\phi,o)}(M_0))$, where $\boldsymbol{\rho}_{(\phi,o)}(M_0) \subset \mathbb{R}^{2m+1}$ denotes the image of M_0 under $\boldsymbol{\rho}_{(\phi,o)}$.

[3] An alternative would be to simulate a time series with fifteen minute gaps by omitting intermediate values which would lead to a considerable reduction of the number of points. This is however not advantageous, as more points usually lead to better prediction results for the numerical algorithm.

Theorem 2. *Let $A \subset M_0 \subset \mathbb{R}^k$ where M_0 is an open subset of $\mathbb{R}^k$ and A is a compact subset of M_0 which possesses box-counting dimension $\widehat{\dim}(A) = m$. Furthermore, let $\phi : M_0 \to M_0$ be a C^2-diffeomorphism and let $o \in C^2(M_0, \mathbb{R})$. Then, for $\boldsymbol{\rho}_{(\phi,o)} : M_0 \hookrightarrow \mathbb{R}^{\lfloor 2m+1 \rfloor}$ defined as in (2), the properties*

1. *$\boldsymbol{\rho}_{(\phi,o)}$ is one-to-one on A and*
2. *$\boldsymbol{\rho}_{(\phi,o)}$ is an immersion on each compact subset C of a smooth manifold contained in A*

generically[4] *hold.*

In real world applications, the set A is not a priori known. But for the delay embedding scheme to work we only need to know the box-counting dimension $\widehat{\dim}(A)$ of the set A. Its estimation is an elaborate task by its own. To this end, various approaches exist in the literature [20, 21, 24]. Here, we recommend using the Grassberger-Procaccia algorithm [10] to estimate the correlation dimension $\tilde{m}$ as an approximation of the box-counting dimension m since this worked best in our experiments. The delay length is then set to $d = \lfloor 2\tilde{m} + 1 \rfloor$.
In summary we have a theory which provides us with a justification to use delayed vectors like in (2) as input for a learning tool.

2 The regression problem and the regularized least squares approach

In this section, we describe how the task of predicting a time series can be recast into a higher-dimensional regression problem by means of delay embedding. Furthermore, we motivate a specific regularized least squares approach.
We assume that we have an infinite time series $(s_j)_{j=1}^{\infty}$ which is just an observation of a deterministic process on an m-dimensional attractor, compare Sect. 1. From now on, let $d := \lfloor 2m + 1 \rfloor$ denote the embedding dimension used for the delay scheme. We define

$$\mathbf{t}_j := (s_{j-d+1}, s_{j-d+2}, \ldots, s_{j-1}, s_j)^T \in \mathbb{R}^d, \ j \geq d \,, \tag{5}$$

to be the j-th delay vector. Due to Takens' theorem, there exists a $\hat{g} : \mathbb{R}^d \to \mathbb{R}$, with $\hat{g} := o \circ \phi^d \circ \boldsymbol{\rho}_{(\phi,o)}^{-1}$, cf. (3), such that

$$\hat{g}(\mathbf{t}_j) = s_{j+1} \text{ for all } j \geq d \,. \tag{6}$$

If we assume that $(s_j)_{j=1}^{N}$ is known a priori then our goal is to find a good approximation g to $\hat{g}$ with the help of $N - d + 1$ training patterns

$$(\mathbf{t}_j, s_{j+1}) \in \mathbb{R}^d \times \mathbb{R}, \ j = d, \ldots, N-1 \,. \tag{7}$$

Thus, we now have to deal with a regression problem instead of the forecasting problem. Our approach is to choose $g \in X \subset \{f : \mathbb{R}^d \to \mathbb{R}\}$ as

[4] Here "generically" means the following:
If $\tilde{X}_l := \{\mathbf{x} \in A \mid \phi^l(\mathbf{x}) = \mathbf{x}\}$ fulfills $\widehat{\dim}\left(\tilde{X}_l\right) \leq \frac{l}{2}$ for all $l \leq \lfloor 2m+1 \rfloor$ and if $\left(\mathrm{D}\phi^l\right)_{\mathbf{x}}$ has pairwise distinct eigenvalues for all $l \leq \lfloor 2m+1 \rfloor, \mathbf{x} \in \tilde{X}_l$, then the set of all $o \in C^2(M_0, \mathbb{R})$ for which the properties in Theorem 2 do not hold is a null set.

$$g = \arg\min_{f \in X} \mathcal{F}(f)$$

where $\mathcal{F} : X \to \mathbb{R}^+ \cup \{\infty\}$ is a functional that expresses how good functions from X approximate $\hat{g}$. The function space X still has to be specified.
To this end, as we do not know any embedded points on which we want to evaluate g afterwards, it is common to minimize the expectation of some Lebesgue measurable cost function $c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+ \cup \{\infty\}$ with respect to the density $p : \mathbb{R}^d \times \mathbb{R} \to [0, 1]$ of all possible input patterns. This leads to

$$\mathcal{F}(f) = \mathbb{E}_p \left[c\left(y, f(\mathbf{x})\right) \right]$$

and thus gives

$$g = \arg\min_{f \in X} \mathbb{E}_p \left[c\left(y, f(\mathbf{x})\right) \right] = \arg\min_{f \in X} \int_{\mathbb{R}^d \times \mathbb{R}} c\left(y, f(\mathbf{x})\right) p(\mathbf{x}, y) \left(\mathrm{d}\mathbf{x} \otimes \mathrm{d}y\right) \ .$$

Note here that we have to restrict X to contain only Lebesgue measurable functions to make this term well-defined.
Since we have to cope with training patterns and do not know the exact density p, we use the empirical density

$$\hat{p}\left(\mathbf{x}, y\right) = \frac{1}{N-d} \sum_{j=d}^{N-1} \delta_{\mathbf{t}_j}(\mathbf{x}) \delta_{s_{j+1}}(y)$$

instead of p. This results in the problem of finding the argument of the minimum of

$$\mathcal{F}(f) = \int_{\mathbb{R}^d \times \mathbb{R}} c\left(y, f(\mathbf{x})\right) \hat{p}\left(\mathbf{x}, y\right) \left(\mathrm{d}\mathbf{x} \otimes \mathrm{d}y\right) = \frac{1}{N-d} \sum_{j=d}^{N-1} c\left(s_{j+1}, f(\mathbf{t}_j)\right) \ . \quad (8)$$

Note that if we want to calculate the point evaluations $f\left(\mathbf{t}_j\right)$, then the set of admissible functions X has here to be restricted further to contain only functions for which point evaluations are well defined.
We decided to use $c\left(a, b\right) := (a - b)^2$. One can easily show that for this specific cost function a minimizer of $\mathcal{F}$ maximizes the likelihood of the given input data under the assumption of a Gaussian noise term being added to each exact time series value, see section 3.3 in [22].[5]
The minimization of (8) for $f \in X$ still leads to an ill-posed problem and a further restriction of the space of admissible functions is therefore needed. To this end, Tikhonov proposed to add a constraint of the form $\Psi(f) \ \leq \ c$ with an arbitrary positive constant c and a nonnegative functional $\Psi : X \to \mathbb{R}^+$ which is strictly convex on a certain subspace depending on the problem itself, see [25]. Using the method of Lagrange multipliers we then obtain the new minimization problem

$$g = \arg\min_{f \in X} \mathcal{F}(f) := \arg\min_{f \in X} \left(\frac{1}{N-d} \sum_{j=d}^{N-1} c\left(s_{j+1}, f(\mathbf{t}_j)\right) + \lambda \Psi(f) \right) \quad (9)$$

which is well-posed if λ is positive. We will employ the Sobolev semi-norm

[5] Other cost functions can be used as well but these might lead to non-quadratic or even non-convex minimization problems.

$$\Psi(f) := |f|_{H^1_{\text{mix}}} = \sum_{|\mathbf{a}|_\infty = 1} \left|\left| \frac{\mathrm{d}^{a_1}}{\mathrm{d}x_1^{a_1}} \cdots \frac{\mathrm{d}^{a_d}}{\mathrm{d}x_d^{a_d}} f \right|\right|^2_{L_2(\mathbb{R}^d)} \tag{10}$$

since this perfectly fits after discretization to our basis functions as we will see later. Here $\mathbf{a} = (a_1, \ldots, a_d)$ denotes a multi index and $|\mathbf{a}|_\infty := \max_{i=1,\ldots,d} |a_i|$. We will use the function $g \in X$ defined in (9) as continuous approximation to $\hat{g}$ from now on.

Instead of our H^1_{mix}-semi-norm, a method using gradient penalties — which corresponds to the H^1 semi-norm — was presented in [7] and error bounds were provided for a discrete solution achieved by the so-called combination technique. Note that some of these results rely on the assumption of independent and uniformly distributed samples. Nevertheless, similar results can be given for our case under the assumption of independently drawn samples according to the probability distribution on the reconstructed attractor. The resulting errors then refer to the attractor measure instead of the Lebesgue measure.

2.1 Minimization for an arbitrary basis

Now let $\{\gamma_i\}_{i=1}^\infty$ be a basis of $\Gamma := \{f \in X \mid \Psi(f) \leq c\}$. Our task is to find a

$$\mathbf{w} := (w_1, w_2, \ldots)$$

with $w_i \in \mathbb{R}$ for each $i \in \mathbb{N} \setminus \{0\}$, which minimizes

$$\frac{1}{N-d} \sum_{j=d}^{N-1} \left(s_{j+1} - \sum_{i=1}^{\infty} w_i \gamma_i(\mathbf{t}_j) \right)^2 + \lambda \sum_{i=1}^{\infty} \sum_{k=1}^{\infty} \left(w_i w_k \sum_{|\mathbf{a}|_\infty = 1} \langle \mathrm{D}^{\mathbf{a}} \gamma_i, \mathrm{D}^{\mathbf{a}} \gamma_k \rangle_{L_2(\mathbb{R}^d)} \right) \tag{11}$$

where $\mathrm{D}^{\mathbf{a}} = \frac{\mathrm{d}^{a_1}}{\mathrm{d}x_1^{a_1}} \ldots \frac{\mathrm{d}^{a_d}}{\mathrm{d}x_d^{a_d}}$ denotes a multivariate derivative. The corresponding function

$$g(\mathbf{x}) = \sum_{i=1}^{\infty} w_i \gamma_i(\mathbf{x})$$

then would give us the approximate prediction $s_{N+1} \approx g(\mathbf{t}_N)$ by point evaluation at $\mathbf{t}_N$. As (11) is a sum of strictly convex functions, the argument g of the minimum can be found by identifying the zeroes of $\frac{\mathrm{d}}{\mathrm{d}w_l} \mathcal{F}(f)$ for all $l \in \mathbb{N} \setminus \{0\}$. For $\eta := (N-d)\lambda$ this leads to the *infinite* system

$$\sum_{j=d}^{N-1} s_{j+1} \gamma_l(\mathbf{t}_j) = \sum_{i=1}^{\infty} w_i \left(\sum_{j=d}^{N-1} \gamma_l(\mathbf{t}_j) \gamma_i(\mathbf{t}_j) + \eta h(\gamma_i, \gamma_l) \right) \tag{12}$$

for all $l \in \mathbb{N} \setminus \{0\}$, where $h : \Gamma \times \Gamma \to \mathbb{R}$ denotes the semi-definite bilinear form

$$h(s,t) = \sum_{|\mathbf{a}|_\infty = 1} \langle \mathrm{D}^{\mathbf{a}} s, \mathrm{D}^{\mathbf{a}} t \rangle_{L_2(\mathbb{R}^d)} \ .$$

2.2 Minimization for a kernel basis in a reproducing kernel Hilbert space

To derive a finite solution procedure, the following approach is standard in the mathematical learning community. For the case $\Psi(f) = ||f||^2_{\mathcal{H}}$, with $\mathcal{H}$ being a reproducing kernel Hilbert space, we can write g from (9) as a finite linear combination of evaluations of the reproducing kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ in the points corresponding to the training patterns

$$g(\mathbf{x}) = \sum_{j=d}^{N-1} g_j k\left(\mathbf{t}_j, \mathbf{x}\right)$$

with some real-valued weights g_j. This is known as the representer theorem for reproducing kernel Hilbert spaces, see e.g. [22]. Analogous observations as above result with the property $\langle k\left(\mathbf{t}_i, \cdot\right), k\left(\mathbf{t}_j, \cdot\right)\rangle_{\mathcal{H}} = k\left(\mathbf{t}_i, \mathbf{t}_j\right)$ in the finite system

$$\sum_{j=d}^{N-1} s_{j+1} k\left(\mathbf{t}_j, \mathbf{t}_l\right) = \sum_{j=d}^{N-1} g_j \left(\sum_{i=d}^{N-1} k\left(\mathbf{t}_i, \mathbf{t}_j\right) k\left(\mathbf{t}_i, \mathbf{t}_l\right) + \eta k\left(\mathbf{t}_j, \mathbf{t}_l\right) \right)$$

for all $l \in \{d, \ldots, N-1\}$. If the $\left(k\left(\mathbf{t}_j, \mathbf{x}\right)\right)_{j=d}^{N-1}$ are linearly independent[6] this leads to the linear system

$$\mathbf{s} = \left(\mathbf{K} + \eta \mathbf{I}\right) \mathbf{g} \tag{13}$$

where $\mathbf{K} \in \mathbb{R}^{(N-d)\times(N-d)}$ is the kernel matrix with entries $\mathbf{K}_{i,j} = k\left(\mathbf{t}_i, \mathbf{t}_j\right)$, $\mathbf{I} \in \mathbb{R}^{(N-d)\times(N-d)}$ is the identity matrix and $\mathbf{g} = \left(g_d, \ldots, g_{N-1}\right)^T \in \mathbb{R}^{N-d}$, $\mathbf{s} = \left(s_{d+1}, \ldots, s_N\right)^T \in \mathbb{R}^{N-d}$.
Note that for the case (10) we only regularized with a semi-norm of a reproducing kernel Hilbert space but still get the representation

$$g(\mathbf{x}) = \sum_{j=d}^{N-1} g_j k\left(\mathbf{t}_j, \mathbf{x}\right) + g_0(\mathbf{x})$$

with a $g_0 : \mathbb{R}^d \to \mathbb{R}$ from the null space of Ψ and a certain kernel function k, see [22].
One could now try to solve the linear system (13). The major problem of this approach — besides the knowledge of an explicit formulation of the reproducing kernel[7] — is the complexity with respect to the number of input patterns. The direct solution of (13) would involve a number of operations of the order $O\left(N^3\right)$ since we have to deal with a full system matrix here. But even if one does not compute the inverse of $\mathbf{K} + \eta\mathbf{I}$ directly and uses an appropriate iterative scheme instead, the complexity for solving this system is at least $O\left(N^2\right)$ because of the dense kernel matrix $\mathbf{K}$. Therefore, in the next section, we will consider the infinite system (12) in the first place and resort to a further approximation of our prediction problem by discretization.

[6] If this is not the case we can choose a linearly independent subsystem and continue analogously.

[7] See [26] for several reproducing kernels and their corresponding Hilbert spaces.

3 Discretization via sparse grids

To find an approximate solution to (12) we restrict ourselves to a finite dimensional subspace $\Gamma_M := \operatorname{span}\{\gamma_i\}_{i=1}^M \subset \Gamma := \{f \in X \mid \Psi(f) \leq c\}$ for some $M \in \mathbb{N}$. For the naive full grid approach the curse of dimensionality then shows up in the number of necessary grid points which grows exponentially with d. To deal with this issue, we will employ the sparse grid discretization technique and its adaptive enhancements here. To this end, we will assume that the domain of $\hat{g}$ (and thus g) is the d-dimensional hypercube

$$\mathbf{H}_d := [0,1]^d \ .$$

Note that this is not a restriction since the domain of the underlying original process is compact (cf. Theorems 1, 2). By rescaling the resulting domain of the reconstructed process we always can obtain the domain $[0,1]^d$.

3.1 Multilevel hierarchical bases and regular sparse grids

First, we recall the construction of a full grid space using a piecewise linear hierarchical basis and discuss its relation to a sparse grid space. Let the one-dimensional hat function $\phi : \mathbb{R} \to [0,1]$ be defined by

$$\phi(x) := \begin{cases} 1 - |x|, & \text{if } x \in [-1,1] \\ 0 & \text{else} \end{cases}$$

and let

$$\phi_{l,i}(x) := \phi(2^l \cdot x - i)|_{[0,1]}$$

for any $l, i \in \mathbb{N}$ be a dilated and rescaled version of ϕ restricted to the interval $[0,1]$. One can easily see that $\operatorname{supp}(\phi_{l,i}) = \big((i-1)2^{-l}, (i+1)2^{-l}\big) \cap [0,1]$. The construction of a d-dimensional hat function is straightforward via the tensor product

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) := \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j) \ ,$$

where $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ is the multivariate level and $\mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{N}^d$ denotes the multivariate position index. Furthermore, we define $\mathbf{x}_{\mathbf{l},\mathbf{i}} := \mathbf{i} \cdot 2^{-\mathbf{l}}$, where the multiplication has to be understood componentwise, i.e. $\mathbf{x}_{\mathbf{l},\mathbf{i}} = (x_{l_1,i_1}, \ldots, x_{l_d,i_d})^T$ with $x_{l_j,i_j} := i_j \cdot 2^{-l_j}$. For a fixed $\mathbf{l} \in \mathbb{N}^d$, we then have with

$$\Omega_{\mathbf{l}} := \left\{ \mathbf{x}_{\mathbf{l},\mathbf{i}} \mid \mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}} \right\}$$

the full grid of level $\mathbf{l}$. Here, the inequalities are to be understood componentwise and $\mathbf{0} = (0, \ldots, 0)$ is the null index. The space of piecewise d-linear functions on the grid $\Omega_{\mathbf{l}}$ is

$$V_{\mathbf{l}} := \operatorname{span}\{B_{\mathbf{l}}\} \qquad \text{with } B_{\mathbf{l}} = \left\{ \phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}} \right\} \ .$$

$B_{\mathbf{l}}$ is called nodal basis since the value of a function $f_{\mathbf{l}}(\mathbf{x}) = \sum_{\mathbf{0} \leq \mathbf{i} \leq 2^{\mathbf{l}}} f_{\mathbf{l},\mathbf{i}} \cdot \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) \in V_{\mathbf{l}}$ on one of the gridpoints $\mathbf{x}_{\mathbf{l},\mathbf{j}}$ of $\Omega_{\mathbf{l}}$ is given by the coefficient $f_{\mathbf{l},\mathbf{j}} \in \mathbb{R}$ that corresponds to $\phi_{\mathbf{l},\mathbf{j}}$.

Now, let

$$\mathbf{I}_\mathbf{l} := \left\{ \mathbf{i} \in \mathbb{N}^d \,\middle|\, \begin{array}{ll} 0 \leq i_j \leq 1, & \text{if } l_j = 0 \\ 1 \leq i_j \leq 2^{l_j} - 1,\ i_j \text{ odd} & \text{if } l_j > 0 \end{array} \text{ for all } 1 \leq j \leq d \right\} . \tag{14}$$

Then, $W_\mathbf{l} := \operatorname{span}\{\phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \mathbf{I}_\mathbf{l}\}$ is a hierarchical increment space (or detail space) because of the property

$$W_\mathbf{l} = \operatorname{span}\left\{ B_\mathbf{l} \setminus \bigcup_{j=1}^{d} B_{\mathbf{l}-\mathbf{e}_j} \right\}$$

where $\mathbf{e}_j$ denotes the j-th unit vector and $B_\mathbf{k} := \emptyset$ for each $\mathbf{k} = (k_1, \ldots, k_d)$ with $k_j < 0$ for some $j = 1, \ldots, d$. Thus we get

$$V_\mathbf{l} = \bigoplus_{\mathbf{k} \leq \mathbf{l}} W_\mathbf{k} = \operatorname{span}\left\{ \tilde{B}_\mathbf{l} \right\}$$

with the hierarchical basis

$$\tilde{B}_\mathbf{l} := \{\phi_{\mathbf{k},\mathbf{i}} \mid \mathbf{i} \in \mathbf{I}_\mathbf{k}, \mathbf{k} \leq \mathbf{l}\} .$$

Now, we can define the space of piecewise d-linear functions on the regular (isotropic) full grid

$$\Omega_t := \Omega_{(t,\ldots,t)} = \{\mathbf{x}_{\mathbf{k},\mathbf{i}} \mid |\mathbf{k}|_\infty \leq t, \mathbf{i} \in \mathbf{I}_\mathbf{k}\}$$

of level $t \in \mathbb{N}$ by

$$V_t := V_{(t,\ldots,t)} = \bigoplus_{|\mathbf{k}|_\infty \leq t} W_\mathbf{k} .$$

If

$$f_t = \sum_{|\mathbf{k}|_\infty \leq t} \sum_{\mathbf{i} \in \mathbf{I}_\mathbf{k}} f_{\mathbf{k},\mathbf{i}} \phi_{\mathbf{k},\mathbf{i}}$$

is the interpolant of $f \in H^2(\mathbf{H}_d)$ in V_t it holds that

$$||f - f_t||_{L_2(\mathbf{H}_d)} = O\left(2^{-2t}\right) . \tag{15}$$

Next, we define the regular sparse grid of level t by

$$\Omega_t^s := \{\mathbf{x}_{\mathbf{k},\mathbf{i}} \mid n_d(\mathbf{k}) \leq t, \mathbf{i} \in \mathbf{I}_\mathbf{k}\} \tag{16}$$

and the corresponding function space by

$$V_t^s := \bigoplus_{\substack{\mathbf{k} \in \mathbb{N}^d \\ n_d(\mathbf{k}) \leq t}} W_\mathbf{k} ,$$

where $n_d(\mathbf{0}) := 0$ and

$$n_d(\mathbf{k}) := |\mathbf{k}|_1 - d + |\{m \mid \mathbf{k}_m = 0\}| + 1$$

for every other $\mathbf{k} \in \mathbb{N}^d$. Here, $|\mathbf{k}|_1 := \sum_{j=1}^d |\mathbf{k}_j|$ denotes the ℓ^1 norm.
If

$$f_t^s(\mathbf{x}) = \sum_{\substack{\mathbf{k} \in \mathbb{N}^d \\ n_d(\mathbf{k}) \leq t}} \sum_{\mathbf{i} \in \mathbf{I}_\mathbf{k}} \alpha_{\mathbf{k},\mathbf{i}} \phi_{\mathbf{k},\mathbf{i}}(\mathbf{x}) \in V_t^s$$

is the interpolant of $f \in H^2_{\text{mix}}(\mathbf{H}_d)$ in V_t^s, it holds that

$$||f - f_t^s||_{L_2(\mathbf{H}_d)} = O\left(2^{-2t} t^{d-1}\right) .$$

Thus, compared to (15), the accuracy is only slightly worse by a factor t^{d-1}. However, the number of points in the full grid is $|\Omega_t| = O\left(2^{td}\right)$ and suffers from the curse of dimensionality for large d whereas, in the sparse grid case, $M := |\Omega_t^s| = O\left(2^t \cdot t^{d-1}\right)$ holds and the exponential dependence of d now only affects the level t instead of 2^t. For a thorough treatment of sparse grids, approximation results and complexity issues we refer to [1] and the references therein.
By solving (9) in the discrete space $V_t^s \subset \Gamma$ we get (analogously to (12))

$$\sum_{j=d}^{N-1} s_{j+1} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{t}_j) = \sum_{\substack{\mathbf{k} \in \mathbb{N}^d : n_d(\mathbf{k}) \leq t, \\ \mathbf{m} \in \mathbf{I}_\mathbf{k}}} \alpha_{\mathbf{k},\mathbf{m}} \left(\sum_{j=d}^{N-1} \phi_{\mathbf{l},\mathbf{i}}(\mathbf{t}_j)\, \phi_{\mathbf{k},\mathbf{m}}(\mathbf{t}_j) + \eta h\left(\phi_{\mathbf{l},\mathbf{i}}, \phi_{\mathbf{k},\mathbf{m}}\right) \right) \tag{17}$$

for all $\mathbf{l} \in \mathbb{N}^d : n_d(\mathbf{l}) \leq t$ and $\mathbf{i} \in \mathbf{I}_\mathbf{l}$.
A preconditioned multilevel conjugate gradient (pCG) algorithm is used to solve the linear system (17) iteratively. Here, for reasons of simplicity, we employ as preconditioner the inverse of the diagonal of the system matrix of (17) after its transformation to a prewavelet representation, see [13]. As we only need to implement matrix-vector-multiplications for the pCG algorithm, the system matrices are not assembled explicitly. The hierarchical structure and the compact support of our basis functions allow a fast application[8] of the first term in the brackets on the right hand side of (17) in $O\left(N \cdot t^d\right)$ operations. Because of the product structure of H^1_{mix} an efficient implementation of the unidirectional principle can be employed for the on-the-fly multiplication of the term corresponding to the bilinear form h, see e.g. [4]. This needs $O(M)$ operations. Thus, the costs of a single iteration of the pCG algorithm are only $O\left(N \cdot t^d + M\right) = O\left(\left(N \cdot t + 2^t\right) \cdot t^{d-1}\right)$ operations. For a detailed review of computational issues on the implementation of sparse grid methods, grid traversal strategies and linear system solvers, we refer to [4].

3.2 Space-adaptive sparse grids

Since most attractors only fill a sparse pattern of $\mathbf{H}_d$, it is obvious that a regular grid is not necessarily the best structure to approximate a function on such an attractor. On the one hand, there might not be enough grid points in relevant regions to fit the function which leads to bad approximations. On the other hand, there might be too many grid points in irrelevant areas which causes overfitting and results in an unnecessary high cost complexity, see 3.3 in [7] for a thorough treatment of this issue. One would prefer a grid which rather matches the shape of the trajectory than the ambient space $\mathbf{H}_d$. Such a grid (and of course the corresponding function space) can be derived using an iterative algorithm which adaptively creates finer grid resolutions where needed. The main component of such a procedure is an appropriate

[8] Note that the use of the combination technique [14] even allows here for a slight improvement to $O\left(N \cdot t^{d-1}\right)$. In both cases, however, the constant in the O-notation grows exponentially with d.

error indicator which decides if the grid has to be locally refined in a certain region. We here simply use

$$\epsilon_{\mathbf{l},\mathbf{i}} := ||\alpha_{\mathbf{l},\mathbf{i}}\phi_{\mathbf{l},\mathbf{i}}||_{L_\infty(\mathbf{H}_d)} = |\alpha_{\mathbf{l},\mathbf{i}}|$$

as such an indicator. For more elaborate techniques and details on how to choose a reliable *and* efficient indicator $\epsilon_{\mathbf{l},\mathbf{i}}$ for the case of specific norms of the error, we refer to [11].

Our overall algorithm proceeds as follows: First, it starts with a regular sparse grid for some low level $\Omega^s_{\text{adp}} = \tilde{\Omega}^s_{\text{adp}} := \Omega^s_t$ and solves (17) on this grid. Then, it checks for each $\left\{(\mathbf{l},\mathbf{i}) \mid \mathbf{x}_{\mathbf{l},\mathbf{i}} \in \tilde{\Omega}^s_{\text{adp}}\right\}$ if $\epsilon_{\mathbf{l},\mathbf{i}} > \varepsilon$, where $\varepsilon \in \mathbb{R}^+$ is some fix threshold. If this is the case for the pair $(\mathbf{l},\mathbf{i})$ with odd i_j or $i_j = 0$ for each $j \in \{1,\ldots,d\}$, all of its child nodes are inserted into the grid Ω^s_{adp} if they are not already contained.[9] In the one-dimensional case the child nodes are defined as

$$\text{child}\,(x_{l,i}) := \begin{cases} \{x_{l+1,2i\pm1}\} & \text{if } l > 0\,, \\ \{x_{1,1}\} & \text{if } l = 0, i = 1\,, \\ \{x_{0,1}\} & \text{if } l = 0, i = 0\,. \end{cases} \tag{18}$$

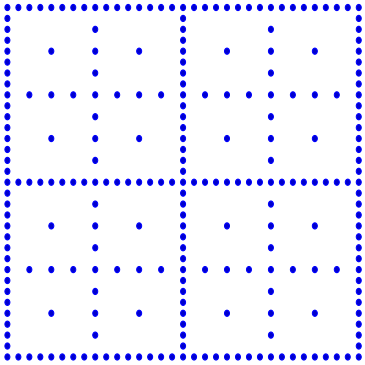
In the multivariate case we define child$(\mathbf{x}_{\mathbf{l},\mathbf{i}})$ as

$$\left\{ \mathbf{x}_{\mathbf{k},\mathbf{m}} \in \Omega_{\mathbf{k}} \;\middle|\; \begin{array}{l} \text{There exists } j \in \{1,\ldots,d\},\ \text{s.t. } x_{k_j,m_j} \in \text{ child}\left(x_{l_j,i_j}\right) \\ \text{and } k_h = l_h, m_h = i_h \text{ for all } h \in \{1,\ldots,d\} \setminus \{j\} \end{array} \right\}. \tag{19}$$

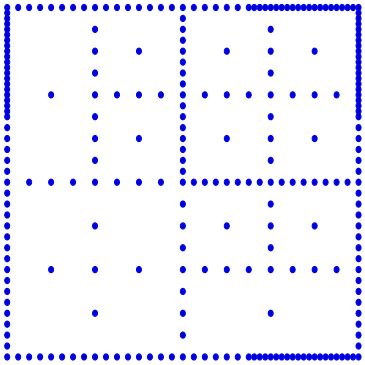
After the insertion it has to be guaranteed — by e.g. inserting further nodes where needed — that all hierarchical ancestors of every inserted point are contained in the resulting grid. Otherwise, an incorrect hierarchical basis representation for the corresponding function space would result and common grid traversal algorithms would run into problems. To achieve this we simply insert each missing direct ancestor and proceed recursively with the inserted points until each direct ancestor to

Fig. 1. Different sparse grid examples in two dimensions

(**a**) Regular sparse grid of level 5



(**b**) Space-adaptive sparse grid



[9] Note here that it is not enough to check the surplus of points which have been inserted in the last iteration. The hierarchical surplus of all other points can change as well when calculating the solution on the refined grid.

every gridpoint has been inserted into Ω^s_{adp}. The direct ancestors of points $\mathbf{x_{l,i}}$ with odd i_j or $i_j = 0$ for each $j = \{1, \ldots, d\}$ are defined by

$$\text{directAnc}\,(\mathbf{x_{l,i}}) := \{\mathbf{x_{k,m}} \in \Omega_{\mathbf{l}} \mid \mathbf{x_{l,i}} \in \text{ child}\,(\mathbf{x_{k,m}})\}\ . \tag{20}$$

When every relevant grid point of $\tilde{\Omega}^s_{\text{adp}}$ has been visited and treated accordingly, we set $\tilde{\Omega}^s_{\text{adp}} = \Omega^s_{\text{adp}}$ and start anew. This iteration runs until either no point needs to be refined or the number of iterations reaches some fixed limit $L \in \mathbb{N}$. A summary of the procedure can be found in Algorithm 1. For details on runtime and technical issues we refer to [4].

Algorithm 1 The space-adaptive sparse grid algorithm

Input: starting level t, threshold ε, #iterations L, error indicators $\epsilon_{\mathbf{l,i}}$, time series $(s_j)_{j=1}^N$, embedding dimension d, regularization parameter λ
Output: space-adaptive sparse grid Ω^s_{adp}

initialize: $\Omega^s_{\text{adp}} \leftarrow \Omega^s_t$, $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_t$, It $\leftarrow 0$
while It $< L$ **do**
  solve (17) on $\tilde{\Omega}^s_{\text{adp}}$
  for all $(\mathbf{k}, \mathbf{m})$ with odd m_j or $m_j = 0$ for each $j \in \{1, \ldots, d\}$ and $\mathbf{x_{k,m}} \in \tilde{\Omega}^s_{\text{adp}}$ **do**
    if $\epsilon_{\mathbf{k,m}} > \varepsilon$ **then**
      $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup \text{ child}\,(\mathbf{x_{k,m}})$
    end if
  end for
  if $\tilde{\Omega}^s_{\text{adp}} = \Omega^s_{\text{adp}}$ **then**
    return Ω^s_{adp}
  end if
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  for all $\mathbf{x_{k,m}}$ with odd m_j or $m_j = 0$ for each $j \in \{1, \ldots, d\}$ and $\mathbf{x_{k,m}} \in \tilde{\Omega}^s_{\text{adp}}$ **do**
    $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup$ AllAncestors($\mathbf{k}$, $\mathbf{m}$, d)
  end for
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  It $\leftarrow$ It+1
end while
return Ω^s_{adp}

Algorithm 2 AllAncestors($\mathbf{l}$, $\mathbf{i}$, d)

Input: multivariate level $\mathbf{l}$, multivariate index $\mathbf{i}$, embedding dimension d
Output: set X of all ancestors of $\mathbf{x_{l,i}}$

initialize: $X \leftarrow \emptyset$
$X \leftarrow X \cup$ directAnc($\mathbf{x_{l,i}}$)
for all $\mathbf{x_{k,m}} \in$ directAnc($\mathbf{x_{l,i}}$) with odd m_j or $m_j = 0$ for each $j \in \{1, \ldots, d\}$ **do**
  $X \leftarrow X \cup$ AllAncestors($\mathbf{k}$, $\mathbf{m}$, d)
end for
return X

3.3 Dimension-adaptive sparse grids

In the case of attractors which fill a highly anisotropic part of the ambient space $\mathbf{H}_d$ or in case the ambient space dimension was overestimated, it is desirable to employ dimension-adaptive refinement instead of pure space-adaptive refinement. There, refinement takes place globally but only in directions which are relevant for the construction of a good forecasting function. Dimension-adaptivity for sparse grids has been introduced in [15]. The application of dimension-adaptive algorithms has been studied for integration in [8] and for approximation in [6]. The approach which we use in the following is a little bit different though, it can be found in [4]. To motivate the idea of dimension-adaptive grids we will shortly review the concept of the ANOVA (Analysis of Variance) decomposition. We introduce a splitting

$$V = \mathbf{1} \oplus \mathcal{C} \tag{21}$$

of a space V of univariate functions with domain $[0,1]$ into the space of constant functions $\mathbf{1}$ and the remainder $\mathcal{C}$. This is done using the identity

$$f = P(f) + (f - P(f))$$

for some projector $P : V \to \mathbf{1}$ with $P|_{\mathbf{1}} = \mathrm{id}$.
For multivariate tensor product function spaces V we apply the splitting in every direction, i.e.

$$\begin{aligned} V &= \bigotimes_{i=1}^{d} V_i = \bigotimes_{i=1}^{d} (\mathbf{1}_i \oplus \mathcal{C}_i) \\ &= \mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_d \\ &\oplus \bigoplus_{i=1}^{d} (\mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_{i-1} \otimes \mathcal{C}_i \otimes \mathbf{1}_{i+1} \otimes \ldots \otimes \mathbf{1}_d) \\ &\oplus \bigoplus_{i=1}^{d} \bigoplus_{j=i+1}^{d} (\mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_{i-1} \otimes \mathcal{C}_i \otimes \mathbf{1}_{i+1} \otimes \ldots \otimes \mathbf{1}_{j-1} \otimes \mathcal{C}_j \otimes \mathbf{1}_{j+1} \otimes \ldots \otimes \mathbf{1}_d) \\ &\vdots \\ &\oplus \mathcal{C}_1 \otimes \ldots \otimes \mathcal{C}_d \,, \end{aligned} \tag{22}$$

and receive a unique splitting of a function $f \in V$ into the sum of a constant function, d univariate functions, $\frac{d(d-1)}{2}$ bivariate functions, and so on, i.e.

$$f(x_1, \ldots, x_d) = f_0 + \sum_{i=1}^{d} f_i(x_i) + \sum_{i=1}^{d} \sum_{j=i+1}^{d} f_{ij}(x_i, x_j) + \ldots + f_{1,\ldots,d}(x_1, \ldots, x_d) \,. \tag{23}$$

We call f_0 the ANOVA component of order 0, the f_i are ANOVA components of order 1, and so on.
The most common choice for P is

$$P(f) := \int_{[0,1]} f(x) \mathrm{d}x$$

for $V \subset L_2([0,1])$, which just gives the classical L_2-ANOVA decomposition. Another choice is

$$P(f) := f(a)$$

which leads to a well-defined decomposition if the point evaluation in a is well-defined for all functions in V. This results in the so-called anchored ANOVA decomposition with anchor a. It is well suited to our piecewise linear basis functions.
Here, to transfer the concept of the multivariate anchored ANOVA decomposition to the piecewise linear hierarchical basis discretization, we have to change the index set introduced in (14). We define

$$\tilde{\mathbf{I}}_{\mathbf{l}} := \left\{ \mathbf{i} \in \mathbb{N}^d \;\middle|\; \begin{array}{ll} i_j = 0, & \text{if } l_j = -1 \\ i_j = 1, & \text{if } l_j = 0 \\ 1 \le i_j \le 2^{l_j} - 1,\ i_j \text{ odd} & \text{if } l_j > 0 \end{array} \quad \text{for all } 1 \le j \le d \right\} \tag{24}$$

and allow the negative level -1. Furthermore, we define the one-dimensional basis function $\phi_{-1,0} := \chi_{[0,1]}$ to be the indicator function of the interval $[0,1]$. With this and the definition

$$\tilde{W}_{\mathbf{l}} := \operatorname{span}\{\phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \tilde{\mathbf{I}}_{\mathbf{l}}\}$$

we see[10] that

$$\begin{aligned} \tilde{V}_{\mathbf{l}} := \bigoplus_{-\mathbf{1} \le \mathbf{k} \le \mathbf{l}} \tilde{W}_{\mathbf{k}} &= \bigoplus_{-\mathbf{1} \le \mathbf{k} \le \mathbf{l}} \operatorname{span}\{\phi_{\mathbf{k},\mathbf{m}} \mid \mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}}\} \\ &= \bigoplus_{\mathbf{0} \le \mathbf{k} \le \mathbf{l}} \operatorname{span}\{\phi_{\mathbf{k},\mathbf{m}} \mid \mathbf{m} \in \mathbf{I}_{\mathbf{k}}\} = \bigoplus_{\mathbf{0} \le \mathbf{k} \le \mathbf{l}} W_{\mathbf{k}} = V_{\mathbf{l}} \end{aligned}$$

for all $\mathbf{l}$ with $l_j \ge 0$ for all $j = 1, \ldots, d$. This way, we just have split the space of linear functions on $[0,1]$, which was previously spanned by the two linear basis functions associated to the two boundary points, further into the sum of one constant (level -1) and one linear function (level 0). If we define the norm of a multivariate level index with possibly negative coordinates as

$$|\mathbf{l}| := |(\max(l_1, 0), \ldots, \max(l_d, 0))|$$

we can maintain our previous definition for sparse grids (16) using

$$\tilde{n}_d(\mathbf{k}) := \begin{cases} 0 & \text{if } k_j \le 0 \text{ for all } 1 \le j \le d \\ |\mathbf{k}|_1 - d + |\{m \mid \mathbf{k}_m \le 0\}| + 1 & \text{else} \end{cases}$$

instead of $n_d(\mathbf{k})$. But we now are able to identify functions which are constant in direction j as they are elements of $\tilde{V}_{(l_1,\ldots,l_{j-1},-1,l_{j+1},\ldots,l_d)}$. This approach fits to a discretized anchored ANOVA decomposition with $a = 0$. To this end, we now define an infinite-dimensional univariate function space

$$V = \tilde{V}_{-1} \oplus \bigoplus_{i=0}^{\infty} \tilde{W}_i \tag{25}$$

and, with the choice $\mathbf{1}_i = \left(\tilde{V}_{-1}\right)_i$ and $\mathcal{C}_i = \left(\bigoplus_{j=0}^{\infty} \tilde{W}_j\right)_i$ in (21), we again obtain the splitting (22) which is now conform to the infinite-dimensional tensor product-hierarchical basis. In other words, if we use the alternative basis that is defined by

[10] Note that $W_{\mathbf{l}}$ and $\tilde{W}_{\mathbf{l}}$ are the same for a multilevel index $\mathbf{l}$ with $l_j \ge 1$ for all $j = 1, \ldots, d$.

the index set $\tilde{\mathbf{I}}_l$, the only univariate basis function ψ for which $P(\psi) \neq 0$ is $\psi = \phi_{-1,0}$ for $P(f) := f(0)$ and the anchored ANOVA decomposition completely fits to the hierarchical tensor product basis.
So far, the subspaces of the ANOVA decomposition are (up to the very first one) still infinite-dimensional and need to be further discretized. To this end, for a regular sparse grid of level t, we truncate each term of the ANOVA-decomposition as follows:

$$\begin{aligned} V_t^s = \; & \mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_d \\ & \oplus \bigoplus_{i=1}^{d} \bigoplus_{\substack{\tilde{n}_1(k_i) \leq t \\ k_i \in \mathbb{N}}} \left(\mathbf{1}_1 \otimes \ldots \otimes \mathbf{1}_{i-1} \otimes \left(\tilde{W}_{k_i} \right)_i \otimes \mathbf{1}_{i+1} \otimes \ldots \otimes \mathbf{1}_d \right) \\ & \oplus \bigoplus_{i=1}^{d} \bigoplus_{j=i+1}^{d} \bigoplus_{\substack{\tilde{n}_2(k_i,k_j) \leq t \\ k_i,k_j \in \mathbb{N}}} \left(\mathbf{1}_1 \otimes \ldots \otimes \left(\tilde{W}_{k_i} \right)_i \otimes \ldots \otimes \left(\tilde{W}_{k_j} \right)_j \otimes \ldots \otimes \mathbf{1}_d \right) \\ & \vdots \\ & \oplus \bigoplus_{\substack{\tilde{n}_d(k_1,\ldots,k_d) \leq t \\ k_1,\ldots,k_d \in \mathbb{N}}} \left(\tilde{W}_{k_1} \right)_1 \otimes \ldots \otimes \left(\tilde{W}_{k_d} \right)_d . \end{aligned}$$

Thus, we discretize every k-variate component function of the ANOVA decomposition (23) with a regular k-dimensional sparse grid, where $k \in \{1, \ldots, d\}$.
A dimension-adaptive procedure can now be defined analogously to the space-adaptive algorithm. To this end, we employ the error indicator

$$\epsilon_{\mathbf{l}} := \max_{\mathbf{i} \in \tilde{\mathbf{I}}_{\mathbf{l}}} \epsilon_{\mathbf{l},\mathbf{i}}$$

which is just defined on the detail spaces $\tilde{W}_{\mathbf{l}}$ and does not rely on a single point anymore. For refinement we now simply insert *all* the points belonging to the basis functions of $\tilde{W}_{\mathbf{l}+\mathbf{e}_j}$ for each direction j with $l_j \neq -1$. Thus, we only insert grid nodes that lie in the same ANOVA component as nodes in $\tilde{W}_{\mathbf{l}}$. By doing this, refinement of the grid affects relevant ANOVA terms of the function but neglects higher-order terms. As in the case of spatial adaptivity, ancestors of new points have to be inserted into the grid.[11] The whole refinement procedure is iterated in the same way as previously.
Additionally, we compress the grid in an initial preprocessing step before starting the dimension-wise refinement procedure. To this end, for a given ε, starting with a regular sparse grid of level t, every detail space $\tilde{W}_{\mathbf{k}} \subset V_t^s$ for which $\epsilon_{\mathbf{k}} \leq \varepsilon$ holds, is marked first. Then, if the points in a marked subspace $\tilde{W}_{\mathbf{k}}$ are not needed as an ancestor to a point in a non-marked subspace, all points belonging to $\tilde{W}_{\mathbf{k}}$ are removed from the grid. This compression is done, since a regular sparse grid of level 0 contains already part of each ANOVA component. Thus, it is not possible to identify relevant ANOVA components of a function just by looking at a current adaptive grid. One would rather want to completely neglect components which do not contribute to the representation of a function and then start the adaptive procedure on a

[11] For the one-dimensional case one simply defines $x_{0,1}$ to be the single child node of $x_{-1,0}$. The generalization to the multi-dimensional case is straightforward.

dimensionally reduced grid. The overall dimension-adaptive process is given in detail in Algorithm 3.

Algorithm 3 The dimension-adaptive sparse grid algorithm using the basis defined by (24)

Input: starting level t, threshold ε, #iterations L, error indicators $\epsilon_{\mathbf{l}}$, time series $(s_j)_{j=1}^{N}$, embedding dimension d, regularization parameter λ
Output: dimension-adaptive sparse grid Ω^s_{adp}

initialize for compression: $Y \leftarrow \emptyset$
for all $\tilde{W}_{\mathbf{k}} \subset V_t^s$ **do**
  if $\epsilon_{\mathbf{k}} > \varepsilon$ **then**
    $Y \leftarrow Y \cup \left\{ \mathbf{x}_{\mathbf{k},\mathbf{m}} \mid \mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}} \right\}$
  end if
end for
$Z \leftarrow Y$
for all $\mathbf{x}_{\mathbf{k},\mathbf{m}} \in Y$ **do**
  $Z \leftarrow Z \cup \text{AllAncestors}(\mathbf{k}, \mathbf{m}, d)$
end for

initialize for adaption: $\Omega^s_{\text{adp}} \leftarrow Z$, $\tilde{\Omega}^s_{\text{adp}} \leftarrow Z$, It $\leftarrow 0$
while It $< L$ **do**
  solve (17) on $\tilde{\Omega}^s_{\text{adp}}$
  for all $\tilde{W}_{\mathbf{k}}$ with $\mathbf{x}_{\mathbf{k},\mathbf{m}} \in \tilde{\Omega}^s_{\text{adp}}$ for each $\mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}}$ **do**
    if $\epsilon_{\mathbf{k}} > \varepsilon$ **then**
      for all $j \in \{1, \ldots, d\}$ **do**
        if $k_j > -1$ **then**
          $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup \left\{ \mathbf{x}_{\mathbf{k}+\mathbf{e}_j,\mathbf{m}} \mid \mathbf{m} \in \tilde{\mathbf{I}}_{\mathbf{k}+\mathbf{e}_j} \right\}$
        end if
      end for
    end if
  end for
  if $\tilde{\Omega}^s_{\text{adp}} = \Omega^s_{\text{adp}}$ **then**
    return Ω^s_{adp}
  end if
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  for all $\mathbf{x}_{\mathbf{k},\mathbf{m}}$ with odd m_j or $m_j = 0$ for each $j \in \{1, \ldots, d\}$ and $\mathbf{x}_{\mathbf{k},\mathbf{m}} \in \tilde{\Omega}^s_{\text{adp}}$ **do**
    $\Omega^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}} \cup \text{AllAncestors}(\mathbf{k}, \mathbf{m}, d)$
  end for
  $\tilde{\Omega}^s_{\text{adp}} \leftarrow \Omega^s_{\text{adp}}$
  It $\leftarrow$ It+1
end while
return Ω^s_{adp}

4 Numerical results

We will now present numerical results for our sparse grid algorithms when applied to both, synthetically constructed time series and series which stem from real world applications. All data has been properly scaled, such that the embedded points are situated in $\mathbf{H}_d$. The preconditioned conjugate gradient algorithm, which is used to solve the linear system (17), is always iterated until the quotient $||r_k||_{D^{-1}}/||r_0||_{D^{-1}}$ is smaller than 10^{-13} where D is the diagonal preconditioning matrix we used[12], r_k denotes the residual of (17) after the k-th iteration and $||r_k||_{D^{-1}} := \sqrt{r_k^T D^{-1} r_k}$.

4.1 Hénon Map in 2*d*

First, we show results concerning the famous Hénon map

$$z_{n+1} := a - z_n^2 + b z_{n-1} \ , \tag{26}$$

see also [16]. Using the notation of Sect. 1 we have

$$\phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} a - x_1^2 + b x_2 \\ x_1 \end{pmatrix}$$

and

$$o\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = x_2 \ ,$$

where ϕ and o are defined on $\mathbb{R}^2$. It is easy to see that

$$\det \mathrm{D}\phi = -b \ ,$$

where $\mathrm{D}\phi$ denotes the Jacobian matrix of ϕ. We will restrict ourselves to the most popular case $a = 1.4$, $b = 0.3$ for which the trajectory of the process approaches an attractor of non-integer box-counting dimension 1.26. As $|\det \mathrm{D}\phi| < 1$, the process is dissipative and the attractor is a compact subset of the ambient space. Therefore[13] we can apply Theorem 2.
A direct application would lead to the embedding dimension $d = \lfloor\, 2 \cdot 1.26 + 1 \,\rfloor = 3$. Nevertheless, we know from equation (26) that two dimensions are sufficient and will use $d = 2$ in our experiments instead of Takens' upper bound $d = 3$. The first $N = 20\,000$ values of the Hénon map are taken into account to construct three different scenarios:

1. The first $T = 50$ points (training data) are used to learn the target function, the remaining $N - T = 19\,950$ points (test data) are used to measure the forecasting error.
2. The first $T = 500$ points (training data) are used to learn the target function, the remaining $N - T = 19\,500$ points (test data) are used to measure the forecasting error.

[12] To this end, the system matrix from (17) is first transformed into the prewavelet basis, see e.g. [4], then, the inverse of its diagonal is taken as preconditioner.

[13] One can easily see that $\tilde{X}_l$ is finite for $l = 1, 2, 3$. Nevertheless, there exist points $\mathbf{x} \in \mathbb{R}^2$ for which $\left(\mathrm{D}\phi^l\right)_{\mathbf{x}}$ has eigenvalues with algebraic multiplicity 2 for $l = 2, 3$.

Table 1. Resulting parameters and errors after three-fold cross-validation for regular sparse grids

T	t	$\log_2(\lambda)$	$\mathrm{RMSE}_{\mathrm{train}}$	$\mathrm{RMSE}_{\mathrm{test}}$
50	3	-17	$5.42 \cdot 10^{-3}$	$\mathbf{1.41 \cdot 10^{-2}}$
500	6	-25	$1.03 \cdot 10^{-4}$	$\mathbf{2.95 \cdot 10^{-4}}$
5 000	7	-22	$9.25 \cdot 10^{-5}$	$\mathbf{1.01 \cdot 10^{-4}}$

Table 2. Resulting parameters and errors after three-fold cross-validation for the support vector machine

T	$\log_2(C)$	$\log_2(\gamma)$	$\log_2(\varepsilon)$	$\mathrm{RMSE}_{\mathrm{train}}$	$\mathrm{RMSE}_{\mathrm{test}}$
50	14	-5	-10	$1.46 \cdot 10^{-3}$	$\mathbf{1.60 \cdot 10^{-3}}$
500	10	1	-15	$2.51 \cdot 10^{-4}$	$\mathbf{2.57 \cdot 10^{-4}}$
5 000	8	3	-17	$2.07 \cdot 10^{-4}$	$\mathbf{2.06 \cdot 10^{-4}}$

3. The first $T = 5\,000$ points (training data) are used to learn the target function, the remaining $N - T = 15\,000$ points (test data) are used to measure the forecasting error.

We compare our regular sparse grid approach to a standard support vector machine regression algorithm using radial basis functions (SVM = RBF ε-SVR). To find appropriate parameters we use three-fold cross-validation. We investigated $t \in \{2, \ldots, 8\}$ and $\lambda \in \{2^{-1}, \ldots, 2^{-25}\}$ for the sparse grid algorithm. For calculations concerning the SVM approach, we used libsvm, see [3] for implementations and parameters. Here, we performed a three-fold cross-validation over the parameters $C \in \{2^0, \ldots, 2^{15}\}$, $\varepsilon \in \{2^{-20}, \ldots, 2^{-1}\}$ and the kernel width $\gamma \in \{2^{-10}, \ldots, 2^5\}$ of the RBF ε-SVR. To measure the forecasting error of any function f on the embedded test data we used the root mean squared error (RMSE). Given the test data $(s_j)_{j=T+1}^{N}$, we can build the embedded vectors $\mathbf{t}_j$ as in (5) for $T + d \leq j \leq N - 1$. Then

$$\mathrm{RMSE}_{\mathrm{test}} := \sqrt{\frac{1}{N - T - d} \sum_{j=T+d}^{N-1} \left(s_{j+1} - f(\mathbf{t}_j)\right)^2} \,.$$

For the training data we define $\mathrm{RMSE}_{\mathrm{train}}$ analogously. The results, i.e. the determined best parameter values and the corresponding errors on training and test data for these parameters, are given in Table 1 for regular sparse grids and in Table 2 for the support vector machine. We observe that the SVM algorithm performs somewhat better than the sparse grid algorithm for the moderate value $T = 50$. But as we increase the size of the training data set, the results for the sparse grid algorithm get successively better. For $T = 5\,000$, the sparse grid algorithm reaches a slightly lower error than the RBF-SVM method while the involved computational costs are substantially less anyway. The sparse grid method is able to use the newly added training data points to discover more structure of the underlying process.
Note here that the number of possible parameter combinations is not the same for the sparse grid method and the SVM. Thus, it is not representative to compare the runtimes of their cross-validation processes. Nevertheless, this comparison gives a

hint of the behavior of the overall runtime when changing T: The cross-validation process for the SVM algorithm was about 20 times faster than that of the sparse grid approach for $T = 50$. For $T = 500$ the runtimes were almost equal and for $T = 5\,000$ the cross-validation process of the sparse grid algorithm was more than three times faster than that of SVM.
In summary, while the SVM algorithm is favorable for few training points, the benefits of the sparse grid algorithm with respect to both, achieved accuracy and necessary computational cost, begin to prevail in situations with more and more training points.

4.2 Jump Map in $5d$

In this experiment we want to show the advantages of the space- and dimension-adaptive sparse grid algorithms compared to the regular sparse grid approach.
Following the rule

$$z_{n+1} := (z_n + z_{n-1}) \mod 1 \,, \tag{27}$$

we get a time series by

$$\phi\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \begin{pmatrix} (x_1 + x_2) \mod 1 \\ x_1 \end{pmatrix}$$

and

$$o\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = x_1 \,,$$

where ϕ and o are defined on $[0,1)^2$. Due to the modulo operation, ϕ has a jump at all points $\{(x_1, x_2)^T \in [0,1)^2 \mid x_1 + x_2 = 1\}$. For each other point of the domain the process is conservative, i.e. $|\det \mathrm{D}\phi| = 1$. Since ϕ is not diffeomorphic, Theorem 2 cannot be invoked. Nevertheless, we will apply the delay embedding scheme and test if the sparse grid solutions are still able to give a suitable predictor for (27).
Again, we use the first $N = 20\,000$ values of the time series and construct three scenarios with the same values of T, i.e. $T = 50, 500, 5\,000$, as for the Hénon map. We now assume that we were just given the time series of length N and do not know anything about the underlying process (27). Thus we have to estimate the dimension m to be able to use the delay embedding scheme with $d = \lfloor 2m + 1 \rfloor$ before applying our sparse grid algorithms. For the small training data set of size $T = 50$ we get an estimate of $m \approx 2.24$ with the Grassberger-Procaccia dimension estimator. For the other two scenarios, the estimated dimension is even closer to 2. Taking $m = 2$ determines our embedding dimension to be $d = 5$ and we therefore build the embedded vectors $\mathbf{t}_j$ in $\mathbb{R}^5$.
We use $\lambda = 10^{-4}$ in the following experiments. The results for a regular grid discretization and for both, a space- and a dimension-adaptive procedure with $\epsilon = 0.1$ and starting level 1, are given in Table 3.
With substantially fewer grid points, both adaptive algorithms achieve the same or even better RMSE values than the regular sparse grid method. The remarkably smaller amount of grid points used in the dimension-adaptive variant is due to the compression step before refinement starts.
Furthermore, note that the dimension-adaptive algorithm is able to reveal the lower-dimensional structure of the embedded process. We can observe from the constructed forecasting function how many grid points have been spent on its different ANOVA

Table 3. RMSE for a regular, a space-, and a dimension-adaptive sparse grid discretization for the jump map in $5d$

(**a**) Regular sparse grid (SG) of level t

T	t	$\lvert\Omega_t^s\rvert$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	3	3 753	$7.67 \cdot 10^{-2}$	$\mathbf{2.34 \cdot 10^{-1}}$
50	4	12 033	$4.91 \cdot 10^{-2}$	$\mathbf{2.20 \cdot 10^{-1}}$
50	5	36 033	$2.57 \cdot 10^{-2}$	$\mathbf{1.47 \cdot 10^{-1}}$
500	3	3 753	$1.12 \cdot 10^{-1}$	$\mathbf{1.41 \cdot 10^{-1}}$
500	4	12 033	$7.12 \cdot 10^{-2}$	$\mathbf{1.11 \cdot 10^{-1}}$
500	5	36 033	$4.39 \cdot 10^{-2}$	$\mathbf{8.53 \cdot 10^{-2}}$
5 000	3	3 753	$1.27 \cdot 10^{-1}$	$\mathbf{1.32 \cdot 10^{-1}}$
5 000	4	12 033	$9.11 \cdot 10^{-2}$	$\mathbf{9.61 \cdot 10^{-2}}$
5 000	5	36 033	$6.43 \cdot 10^{-2}$	$\mathbf{6.99 \cdot 10^{-2}}$

(**b**) Space-adp. SG after $\#It$ iterations

T	$\#It$	$\lvert\Omega_{\text{adp}}^s\rvert$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	3	3 159	$7.68 \cdot 10^{-2}$	$\mathbf{2.34 \cdot 10^{-1}}$
50	4	4 887	$5.03 \cdot 10^{-2}$	$\mathbf{2.14 \cdot 10^{-1}}$
50	5	5 535	$2.75 \cdot 10^{-2}$	$\mathbf{1.33 \cdot 10^{-1}}$
500	3	2 349	$1.12 \cdot 10^{-1}$	$\mathbf{1.40 \cdot 10^{-1}}$
500	4	3 645	$7.21 \cdot 10^{-2}$	$\mathbf{1.10 \cdot 10^{-1}}$
500	5	4 293	$4.66 \cdot 10^{-2}$	$\mathbf{8.32 \cdot 10^{-2}}$
5 000	3	2 079	$1.27 \cdot 10^{-1}$	$\mathbf{1.32 \cdot 10^{-1}}$
5 000	4	2 727	$9.14 \cdot 10^{-2}$	$\mathbf{9.61 \cdot 10^{-2}}$
5 000	5	3 051	$6.47 \cdot 10^{-2}$	$\mathbf{6.96 \cdot 10^{-2}}$

(**c**) Dim.-adp. SG after $\#It$ iterations

T	$\#It$	$\lvert\Omega_{\text{adp}}^s\rvert$	$\text{RMSE}_{\text{train}}$	$\text{RMSE}_{\text{test}}$
50	2	576	$7.90 \cdot 10^{-2}$	$\mathbf{2.43 \cdot 10^{-1}}$
50	3	704	$5.26 \cdot 10^{-2}$	$\mathbf{2.22 \cdot 10^{-1}}$
50	4	832	$2.93 \cdot 10^{-2}$	$\mathbf{1.39 \cdot 10^{-1}}$
500	2	302	$1.15 \cdot 10^{-1}$	$\mathbf{1.40 \cdot 10^{-1}}$
500	3	368	$7.76 \cdot 10^{-2}$	$\mathbf{1.08 \cdot 10^{-1}}$
500	4	424	$5.29 \cdot 10^{-2}$	$\mathbf{8.12 \cdot 10^{-2}}$
5 000	2	310	$1.28 \cdot 10^{-1}$	$\mathbf{1.32 \cdot 10^{-1}}$
5 000	3	326	$9.21 \cdot 10^{-2}$	$\mathbf{9.60 \cdot 10^{-2}}$
5 000	4	342	$6.55 \cdot 10^{-2}$	$\mathbf{6.96 \cdot 10^{-2}}$

components. To this end, counting all grid points with exactly one non-zero coordinate we get the number of points spent on the representation of univariate functions in the ANOVA decomposition. We can continue analogously for bivariate functions (two non-zero coordinates) and so on.

In Fig. 2 we see the distribution of grid points among the ANOVA components of different order for the example of the $5d$ jump map. The dimension-adaptive algorithm successfully detected that there is no term of fifth order and thus grid points are only spent on the boundary of $\mathbf{H}_5$. Terms of third and fourth order are still present, but one observes that most grid points have been used for terms of order 1 and 2. Altogether, the inherent structure of the process was well detected by the algorithm.

As less points are spent by the space- and the dimension-adaptive algorithm there is a significant saving in storage for these methods. In addition, also the absolute runtime of the dimension-adaptive algorithm – especially for the case of few training points and high levels – is better than for the regular sparse grid case, as we see in Table 4. The runtimes of the space-adaptive method and the regular sparse grid algorithm are of the same order for the listed scenarios.

Fig. 2. Distribution of grid points among ANOVA components of different order for the $5d$ jump map and the dimension-adaptive algorithm

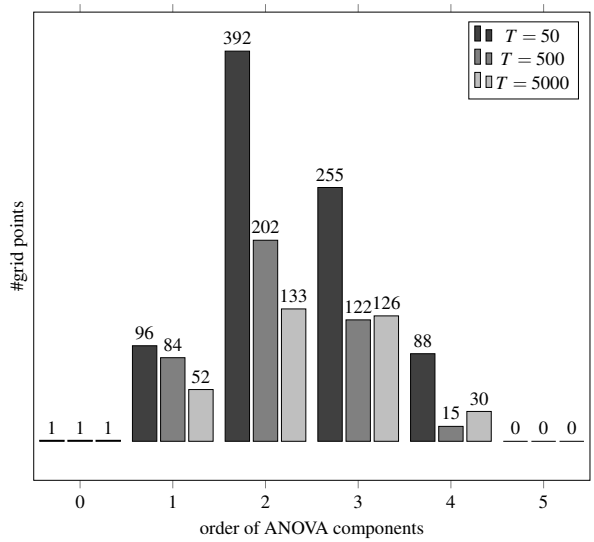


Table 4. Comparison of runtimes for the space- and the dimension-adaptive sparse grid algorithm. $R_{\text{dimadp}}(t, \#It)$ is defined as $\frac{R_{\text{reg}}(t)}{R_{\text{dimadp}}(\#It)}$, where $R_{\text{reg}}(t)$ denotes the runtime of the regular sparse grid algorithm with level t and $R_{\text{dimadp}}(\#It)$ denotes the runtime of $\#It$ iterations of the dimension-adaptive algorithm; $R_{\text{spadp}}(t, \#It)$ is defined analogously for the space-adaptive algorithm

T	t	$\#It$	$R_{\text{spadp}}(t, \#It)$	$R_{\text{dimadp}}(t, \#It)$
50	4	3	0.73	5.05
50	5	4	1.57	13.09
500	4	3	0.70	3.67
500	5	4	1.40	9.51
5 000	4	3	0.61	1.29
5 000	5	4	0.94	1.78

4.3 Small Dataset of the ANN & CI Forecasting Competition 2006/2007

We now consider the performance of the regular and the space-adaptive sparse grid algorithm in a practical application.[14] The reduced dataset of the "Artificial Neu-

[14]Since we restricted ourselves to $d \leq 3$ in this experiment, we did not apply the dimension-adaptive algorithm to this problem.

ral Network and Computational Intelligence Forecasting Competition 2006/2007" consists of eleven empirical business time series.[15] Each of the time series consists of 144 real values where the first 126 values should be used as training data. The goal of the competition is to forecast the last 18 consecutive values. The symmetric mean absolute percent error

$$\text{SMAPE} := \frac{1}{18} \sum_{j=127}^{144} \frac{2|s_j - \hat{s_j}|}{|s_j| + |\hat{s}_j|}$$

determines the quality of the forecast of one particular time series. Here $\hat{s}_j$ denotes the prediction of the j-th test data value. As consecutive values have to be predicted, we cannot simply compute $\hat{s}_j = f(\mathbf{t}_{j-1})$ for a general, computed forecasting function f since the coordinates of the embedded vector $\mathbf{t}_{j-1}$ might not only stem from training data but also from unknown test data. Therefore, we recursively define $\hat{s}_j = f(\hat{\mathbf{t}}_{j-1})$ with $\hat{\mathbf{t}}_{j-1} = (s_{j-d}, s_{j-d+1}, \ldots, \hat{s}_{j-2}, \hat{s}_{j-1})^T$ where a coordinate is set to s_l if $l \leq 126$ and to $\hat{s}_l$ otherwise. Furthermore, we introduce the time step size k which determines which future value is learned. So, by building the vectors $\mathbf{t}_j$ from training data, we are learning s_{j+k}. In the examples introduced earlier we always had set $k = 1$.

In our experiments with the regular sparse grid approach, the first 108 values of a time series are used to learn a prediction model which is then evaluated on the last 18 values of the training data. This way, we determine the best combination of a regularization parameter $\lambda \in \{2^{-15}, \ldots, 2^{-1}\}$, a level $t \in \{2, \ldots, 7\}$, an embedding dimension $d \in \{1, 2, 3\}$ and a future step size $k \in \{1, \ldots, 18\}$. These parameters are then employed to learn a model using the whole training data set. This model is finally taken to predict the 18 values of the test data set. Proceeding in this fashion for every time series we achieve the SMAPEs that can be found in Table 5(**a**).

Alternatively, we fixed $d = 3$ and $t = 1$, started the space-adaptive algorithm with a maximum iteration count of 7. We chose the optimal k and λ in the same fashion as for the regular sparse grid algorithm. The results are shown in table 5(**b**). Even though the space-adaptive algorithm performs better than the non-adaptive method for six of the eleven time series, the average SMAPE of the non-adaptive variant is still smaller. This is mostly due to its bad performance for time series 3. Anyway, with each of the two methods we perform better than 36 of the 44 participants of the competition. Furthermore, we also outperform 8 of the 13 statistical and CI methods that entered the competition as benchmarks.

By combining our two methods such that for every time series the SMAPE on the last 18 values of the training data decides if the space-adaptive or the non-adaptive variant is chosen, we achieve an average SMAPE of 15.3082%. With this result we outscore one more participant and one more statistical benchmark.

Thus, even when dealing with rather short empirical time series where data-based approaches like SVM seem to be a more natural and promising approach, our methods still achieve competitive results.

[15] Further information concerning the setting and the dataset can be found at `http://www.neural-forecasting-competition.com/NN3/index.htm`.

Table 5. SMAPE and average SMAPE for the eleven time series of the reduced dataset of the ANN & CI Forecasting Competition 06/07

	(**a**) Regular sparse grids					(**b**) Space-adaptive sparse grids		
Time Series	t	$\log_2(\lambda)$	d	k	SMAPE in %	$\log_2(\lambda_{\text{adp}})$	k_{adp}	$\text{SMAPE}_{\text{adp}}$ in %
1	7	−13	1	12	2.8398	−11	14	**2.6309**
2	6	−10	2	11	25.6872	−8	11	**23.0448**
3	2	−12	2	11	**30.6692**	−9	10	39.9194
4	4	−9	2	12	**6.3690**	−9	10	8.8058
5	3	−1	1	1	**3.3801**	−10	11	5.1977
6	4	−10	2	10	**4.9186**	−8	9	5.6765
7	4	−13	1	1	6.7220	−3	2	**4.2933**
8	2	−8	2	14	30.3151	−1	12	**27.7111**
9	6	−12	1	4	11.2487	−4	3	**10.2458**
10	7	−6	3	10	**30.3352**	−6	10	30.7120
11	2	−15	2	11	18.5016	−5	11	**16.2794**
Av. SMAPE					**15.5442**			15.8651

5 Concluding remarks

In this article we introduced a sparse grid-based discretization approach to solve the forecasting problem for time series. We gave a short review of Takens' theorem and showed how it can be applied to the field of time series forecasting if the box-counting dimension of the attractor is a priori known or at least properly estimated from the given data. We motivated the use of a regularized quadratic loss-functional and emphasized the difference between kernel-based approaches and arbitrary basis discretizations for the case of reproducing kernel Hilbert spaces. To avoid the curse of dimensionality we introduced regular sparse grids based on piecewise linear B-splines. Space- and dimension-adaptive refinement proved to be useful enhancements, which further reduce costs as e.g. most of the attractors are not uniformly spread across the embedding space. Finally, we have computed numerical results which showed that our algorithms achieve the same or even better results than common SVM-based regression algorithms. The experiments also proved that dimension-adaptive refinement is useful if the embedding dimension of the attractor has been overestimated.

The problem of finding reliable estimates for the box-counting dimension of an attractor has not been discussed in this paper. This is a crucial step for the application of Takens' theorem to real world data. In our further experiments, the Grassberger-Procaccia algorithm proved quite successful to this end.

Another main problem that leads to non-stationarity is noise. Almost all real world data are non-deterministic because of the occurrence of noise as a stochastic component. Several noise-reduction methods and dimension estimators can be found in the literature, see [19] for an overview. These techniques will be incorporated into the sparse grid approach in the future.

Note finally that there are similarities to other existing methods: In [5], a sparse grid approach for manifold learning applications was suggested. There also are links to

a density estimation method with grid-based discretizations, see [12]. An approach that is closely related, but approximates the sparse grid solution by a combination technique, can be found in [6]. A comparison of cost complexity and achieved error between our approach and this technique has still to be done.
Finally, since the curse of dimensionality is still present with respect to the sparse grid level, it is desirable to reduce the dimension of the problem beforehand as good as possible by linear techniques like the well-known SVD [9] or the LT approach of [18].

References

1. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
2. M. Casdagli, T. Sauer, and J. Yorke. Embedology. *Journal of Statistical Physics*, 65:576–616, 1991.
3. C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
4. C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation.* PhD thesis, Institute for Numerical Simulation, University of Bonn, 2010.
5. C. Feuersänger and M. Griebel. Principal manifold learning by sparse grids. *Computing*, 85(4), 2009.
6. J. Garcke. *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern.* PhD thesis, Institute for Numerical Simulation, University of Bonn, 2004.
7. J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1-2):1–25, 2009.
8. T. Gerstner and M. Griebel. Dimension–adaptive tensor–product quadrature. *Computing*, 71(1):65–87, 2003.
9. G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224, 1965.
10. P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica*, D9:189–208, 1983.
11. M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.
12. M. Griebel and M. Hegland. A finite element method for density estimation with Gaussian priors. *SIAM Journal on Numerical Analysis*, 47(6), 2010.
13. M. Griebel and P. Oswald. Tensor product type subspace splitting and multilevel iterative methods for anisotropic problems. *Adv. Comput. Math.*, 4:171–206, 1995.
14. M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.
15. M. Hegland. Adaptive sparse grids. *ANZIAM J.*, 44:C335–C353, 2003.
16. M. Hénon. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50:69–77, 1976.

17. J. Huke. Embedding nonlinear dynamical systems: A guide to Takens' theorem, 2006. Manchester Institute for Mathematical Sciences EPrint: 2006.26.
18. J. Imai and K. Tan. Minimizing effective dimension using linear transformation. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 275–292. Springer, 2004.
19. H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, 2004. 2nd edition.
20. A. Krueger. Implementation of a fast box-counting algorithm. *Computer Physics Communications*, 98:224–234, 1996.
21. L. Liebovitch and T. Toth. A fast algorithm to determine fractal dimensions by box counting. *Physics Letters A*, 141(8,9):386–390, 1989.
22. B. Schölkopf and A. Smola. *Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press – Cambridge, Massachusetts, 2002.
23. F. Takens. Detecting strange attractors in turbulence. *Dynamical Systems and Turbulence, Lecture Notes in Mathematics*, (898):366–381, 1981.
24. J. Theiler. Efficient algorithm for estimating the correlation dimension from a set of discrete points. *Physical Review A*, 36(9):4456–4462, 1987.
25. A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963.
26. G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series In Applied Mathematics*. SIAM: Society for Industrial and Applied Mathematics, 1990.