Compressing the Combination Technique for continuous Functions

Antonia Judit Etelka Krendelsberger

Born 24th July 1998 in Neuss, Germany

15.12.2021

Master's Thesis Mathematics

Advisor: Prof. Dr. Griebel

Second Advisor: Dr. Bohn

Institut für Numerische Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Contents

Introduction

Approximating functions in finite dimensional subspaces, whether directly or as a solution to a differential equation, is an important problem in the field of numerics. It is necessary for being able to use a computer to solve a given problem. One main approach is the *finite element method* [3] which naturally leads to a so-called full grid approach on a tensor product domain. Another is the low-rank approximation method.

To mitigate the curse of dimensionality, which appears when trying to use the full grid approach, the sparse grid method (first introduced by Sergey Smolyak in 1963, introductions can be found in [9, 38, 12]) has been formulated. To further optimize, since the for the sparse grid approach needed multilevel systems can be quite involved in praxis, the *combination technique* [23] was introduced. It recombines the sparse grid into a sum of anisotropic full grids of smaller size, on which the solution to the problem has to be computed.

In this work we wish to further optimize the combination technique by compressing it with the help of low-rank approximation, in this context the *singular value decomposition* [37, 36, 18]. For higher dimensional problems the singular value decomposition does not generalize, so a tensor decomposition has to be used, in this case the tensor train format [35, 11, 24] which is also known as *matrix product states* (MPS). This format, which is newly introduced in this work, is named *Combi-format* in reference to both the combination technique and the use of the tensor train format.

Furthermore we formulate (for two dimensions only) a convergence analysis to prove that we can reach the same order of convergence with the Combi-format as we could with the combination technique. This is in fact also the order of convergence of the straightforward (Galerkin-) projection onto the sparse grid [17] and therefore optimal in this context.

This work is structured as following: in the first part the necessary basics are introduced. This includes the approximation spaces (both full and sparse grids) and the considered function spaces in section 1. Also the projections needed are introduced in section 2, we restrict ourselves to $L^2(\Omega)-$orthogonal projections to solve approximation problems and Galerkin-projections in tensor product form to solve elliptic differential operator equations in tensor product form.

The second part introduces the Combi-format in two dimensions. First the necessary definitions are detailed in section 3 for the combination technique and section 4 for the singular value decomposition. Then the Combi-format can be considered in section 6, including a convergence analysis which is illustrated in praxis in section 7. The code used for the praxis examples is also part of this work.

Finally the third part generalizes the Combi-format to higher dimensions. For this tensors in general and the tensor train format in particular are introduced in section 8. Then the Combi-format can be defined for higher dimensions, and a praxis test shows convergence also in higher dimensions. There is also a convergence analysis for higher dimensions, although we cannot reach the same results as for two dimensions, mostly due to the non-existence of the Eckart-Young-Mirsky theorem for more than two dimensions.

In section 10 we conclude the usefulness of this new Combi-format.

Part I
Preliminaries

Before a definition of our new format is possible, a few basics need to be introduced. We wish to approximate functions $f \in \mathcal{H}$ in finite dimensional subspaces of $L^2(\Omega)$. All of these concepts need to be defined. As such in this part we wish to introduce the necessary basics for the following work.

This includes the function spaces $\mathcal{H} \subset L^2(\Omega)$ in section 1.1 and the approximation spaces of the full and sparse grids in section 1.2. Then we introduce the approximation problem we wish to solve, which is the computation of a (Galerkin-) projection, in section 2.

Throughout this part we consider a domain $\Omega = \Omega_1 \times \cdots \times \Omega_m \subset \mathbb{R}^m$. The Ω_i for $i = 1, \ldots, m$ are closed intervals and can without loss of generality be considered to be $[0, 1]$.

1 Spaces

In the following section we want to characterize the function spaces which will appear in this work. First, the subsection 1.1 defines the possible function spaces $\mathcal{H}$. These are the spaces the functions $f \in \mathcal{H}$, which we wish to approximate, are from. They are based on the concept of Sobolev spaces, which we will also quickly define. Then in subsection 1.2 we define the approximation spaces $V_{\mathbf{l}}$, which the approximated functions are a part of. These spaces are based on a tensor product basis approach, which will lead us into a quick review of full and sparse grids.

1.1 Function Spaces

We consider in this work functions $f \in \mathcal{H} \subset L^2(\Omega)$, where $\mathcal{H}$ is both a Hilbert and a Sobolev space. Everything in the following subsection is only a short definition, for further details one can consult most introductory works for functional analysis (only as an example: [4]). The Sobolev spaces are especially known as the classical solution space for the solving of PDE problems. Needed in the following are an understanding of weak derivatives, Schwartz spaces and tempered distributions as well as the Fourier transform [2, 4]. Here and in the following work we always consider Ω to have a Lipschitz boundary.

As a foundation we consider *Sobolev spaces*, specifically the so-called Bessel potential Sobolev spaces. Since $\mathcal{H} \subset L^2(\Omega)$, we restrict to those spaces where $p = 2$, although generalization to other L^p spaces is of course possible.

Definition 1 (Sobolev space)**.** Let $s \geq 0$, then the *(Bessel potential) Sobolev space* of order s on $\mathbb{R}^n$ is defined as

$$H^s(\mathbb{R}^n) := \left\{ f \in L^2(\mathbb{R}^n) : \|f\|_{H^s(\mathbb{R}^n)} < \infty \right\} \tag{1.1}$$

and on Ω as

$$H^s(\Omega) := \left\{ f \in L^2(\Omega) : f = g|_{\Omega} \, , \, g \in H^s(\mathbb{R}^n) \right\} . \tag{1.2}$$

The norm is given by

$$\|f\|_{H^s(\mathbb{R}^n)} := \left(\sum_{\|\mathbf{l}\|_1 \leq s} \left\| D^{\mathbf{l}} f \right\|^2_{L^2(\mathbb{R}^n)} \right)^{\frac{1}{2}} \tag{1.3}$$

for $s \in \mathbb{N}$ and for $s \in \mathbb{R}$ by

$$\|f\|_{H^s(\mathbb{R}^n)} := \left\| \mathcal{F}^{-1} \left(1 + \|\xi\|_2^2 \right)^{\frac{s}{2}} \mathcal{F} f \right\|_{L^2(\mathbb{R}^n)} . \tag{1.4}$$

On Ω the norm is given by the restriction

$$\|f\|_{H^s(\Omega)} := \inf_{g \in H^s(\mathbb{R}^n), \, f = g|_{\Omega}} \|g\|_{H^s(\mathbb{R}^n)} . \tag{1.5}$$

Here the differential operator $D^{\mathbf{l}}$ refers to the weak derivative, $\mathcal{F}$ is the Fourier transform and ξ is its variable.

It can be shown [4] that the above definitions are well defined and the so-defined spaces H^s are indeed Hilbert spaces. It is generally enough to consider $s \in \mathbb{N}$, but we can use the fractional Sobolev spaces (where $s \notin \mathbb{N}$) for more precise estimates.

To be able to use the concept of a dual space we define for $s \leq 0$ the space

$$H^s(\Omega) = \left(H^{-s}(\Omega) \right)^* .$$

This definition is not necessarily compatible with others in literature, which define the dual over the whole of $\mathbb{R}^d$. Since we never need to consider any more than the bounded domain Ω, we use the for us straightforward approach and just define it as above.

Due to the estimates we later want to employ to describe convergence behaviour, we need to define some function spaces that are defined specifically for a tensor product space [13, 18]:

Definition 2 (Tensor Product Sobolev Spaces)**.** Let $0 \leq s_1, s_2$, then we define the *Sobolev space of dominating mixed derivatives* or *-dominating mixed smoothness* for a tensor product domain $\Omega_1 \times \Omega_2$ by

$$H^{(s_1,s_2)}_{\text{mix}}(\Omega_1 \times \Omega_2) = H^{s_1}(\Omega_1) \otimes H^{s_2}(\Omega_2). \tag{1.6}$$

Analogously the *isotropic Sobolev space* is defined as

$$H^{(s_1,s_2)}_{\text{iso}}(\Omega_1 \times \Omega_2) = H^{(s_1,0)}_{\text{mix}}(\Omega_1 \times \Omega_2) \cap H^{(0,s_2)}_{\text{mix}}(\Omega_1 \times \Omega_2). \tag{1.7}$$

We also use the shorter name *mixed Sobolev space*[1] for (1.6), also known is the name *anisotropic Sobolev space* or simply *H-mix space.*

The definitions above do generalize for more than two subdomains. Also the following identities hold:

$$H^{(s,s)}_{\text{iso}}(\Omega_1 \times \Omega_2) = H^s(\Omega_1 \times \Omega_2);$$
$$H^s(\Omega_1 \times \Omega_2) \subset H^{(s,0)}_{\text{mix}}(\Omega_1 \times \Omega_2); \; H^s(\Omega_1 \times \Omega_2) \subset H^{(0,s)}_{\text{mix}}(\Omega_1 \times \Omega_2)$$

We can generalize the mixed Sobolev spaces to more specific underlying spaces $\mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$ by defining

$$\mathcal{H}^{(s_1,s_2)}_{mix} := \left\{ f \in \mathcal{H} \middle| \left\| \frac{\partial^{\alpha+\beta}}{\partial^{\alpha}_{x_1} \partial^{\beta}_{x_2}} f \right\|_{\mathcal{H}} < \infty \text{ for } |\alpha| \leq s_1, \; |\beta| \leq s_2 \right\} \tag{1.8}$$

for $0 < s_1, s_2$. This definition is compatible to the one in Definition 2 above. Some examples for different $\mathcal{H}$ can be found in [17].

1.2 Approximation spaces

In this subsection the approximation spaces are defined, which define the characteristics of the approximated functions. We will consider tensor product grid spaces, namely full grids and sparse grids. Introductions to sparse grids can be found in [12, 7, 38, 9].

There are many different ways to approach defining the necessary approximation spaces, the one chosen in this work is based on bases. In the first step of approximation we chose a basis, for the sake of the coming sparse grid approach a *hierarchical* and dyadic one. The most commonly known one is probably the nodal basis, also called Lagrangian basis. But many others like wavelets or pre-wavelets can also be considered.

Without loss of generality we assume the domain to be a hypercube, specifically $\Omega = [0, 1]^m$. As a first step we also assume that our function vanishes on the boundary, we will later give a brief outlook on including boundary data.

[1]It is important to note here that the definitions in [20], [29] and [19] are not the same as in [13] or [18]. In this work we therefore write the superscript in brackets $(\cdot, \cdot)$ in the hope of unifying the notation and not causing more confusion.

Example: Hierarchical hat-basis in 1D Before defining the approximation spaces in generality, the following paragraphs introduce the example of the hierarchical hat-basis in one dimension, most likely the most well known example for a chosen basis. In this example we consider $\Omega = [0, 1]$.

The univariate hat function $\phi : \mathbb{R} \longrightarrow \mathbb{R}$ is defined as

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

and its translations and dilations $\phi_{j,k} : [0, 1] \longrightarrow \mathbb{R}$ are given as

$$\phi_{j,k}(x) := \phi(2^j \cdot x - k)\,\big|_{[0,1]} \tag{1.9}$$

for level $j \in \mathbb{N}$ and index $k \in \left\{0, 1, \ldots, 2^j - 1, 2^j\right\}$.

So for each level $j \in \mathbb{N}$ we now have a basis $\Phi_j = \{\phi_{j,k} : k \in \varDelta_j\}$ which defines the approximation space

$$V_j = \operatorname{span}\{\phi_{j,k} : k \in \varDelta_j\}$$

where $\varDelta_j = \left\{1, \ldots, 2^j - 1\right\}$ denotes the index set of cardinality $\#\varDelta_j \sim 2^{-j}$. (Both 0 and 2^j are not part of the index set, since they index basis functions which are non-zero on the boundary. These are treated separately whenever we have to deal with functions not vanishing on the boundary.)

Now we define *hierarchical increment spaces* by $V_j = V_{j-1} \oplus W_j$, resulting in

$$W_j = \operatorname{span}\{\phi_{j,k} : k \in \nabla_j\} = \operatorname{span}\{\phi_{j,k} : k \in \varDelta_j, k \text{ odd}\} \tag{1.10}$$

where $\nabla_j = \{k \in \varDelta_j : k \text{ odd}\}$ is the index set for the increment. It also holds that $V_{j-1} \cap W_j = \{0\}$ and we define $\Psi_j = \{\phi_{j,k} : k \in \nabla_j\}$.

Finally we arrive at the hierarchical splitting of the space

$$V_j = \bigoplus_{l=1}^{j} W_l \tag{1.11}$$

which was the goal of the entire preparations above.

In Figure 1 the hierarchical hat basis of level 3 is illustrated. It shows the basis-functions of W_1, W_2 and W_3; in the third row the functions of level 3 which are not in W_3 are shown in light grey. The final row shows $V_3 = \bigoplus_{l=1}^{3} W_l$.

It can also be noted here that one of the advantages of using this basis (apart from just its simplicity) is that there exist an efficient algorithm for calculating hierarchical coefficients from non-hierarchical ones and back. Therefore one can also compute the coefficients for all basis functions of one level and then transform them, or especially the other way around, which makes evaluations easier. Especially for interpolation, which we do not actually want to consider in this work, this significantly reduces computational complexity.

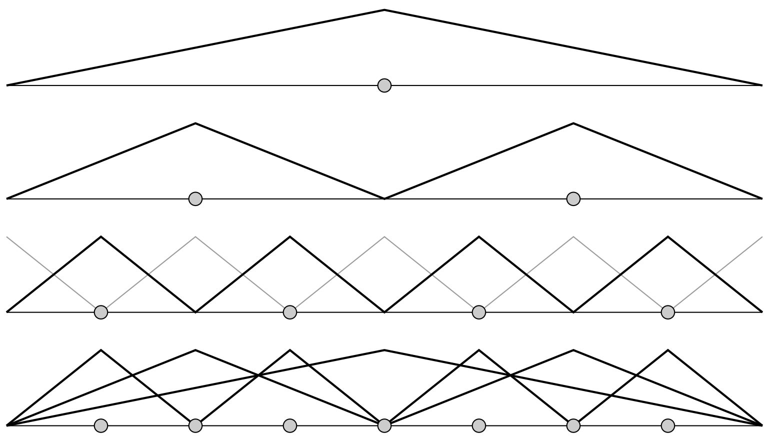


Figure 1: An illustration of the hierarchical hat basis of level 3.

Generalization After this introductory example we consider the generalization of the above concepts to multiple dimensions and general hierarchical bases. For this we now consider tensor products of the bases and therefore the associated spaces. The one-dimensional bases are now arbitrary hierarchical bases, the above considered hat-functions or (pre-)wavelets for example (more bases can be found for example in [7, 2]).

We define the index set of basis functions of level $\mathbf{j}$ as

$$\Delta_{\mathbf{j}} = \{\mathbf{k} \in \mathbb{N}^m : k_i \in \Delta_{j_i}, i = 1, \dots, m\}$$
$$\nabla_{\mathbf{j}} = \{\mathbf{k} \in \mathbb{N}^m : k_i \in \nabla_{j_i}, i = 1, \dots, m\},$$

also the bases

$$\Phi_{\mathbf{j}} = \{\phi_{\mathbf{j},\mathbf{k}} : \mathbf{k} \in \Delta_{\mathbf{j}}\} \tag{1.12}$$
$$\Psi_{\mathbf{j}} = \{\phi_{\mathbf{j},\mathbf{k}} : \mathbf{k} \in \nabla_{\mathbf{j}}\} \tag{1.13}$$

of basis functions

$$\phi_{\mathbf{j},\mathbf{k}} = \phi_{j_1,k_1} \otimes \cdots \otimes \phi_{j_m,k_m} \tag{1.14}$$

defining the spaces

$$V_{\mathbf{j}} = \bigotimes_{l=1}^{m} V_{j_l} = \text{span } \Phi_{\mathbf{j}} \tag{1.15}$$

and hierarchical increment spaces

$$W_{\mathbf{j}} = \bigotimes_{l=1}^{m} W_{j_l} = \text{span } \Psi_{\mathbf{j}}. \tag{1.16}$$

Here, all boldfaced indices are multi-indices of the form $\mathbf{j} = (j_1, \dots, j_m)$. Also the following relations hold:

$$V_{\mathbf{j}} = W_{\mathbf{j}} \oplus V_{\mathbf{j-1}}, \quad W_{\mathbf{j}} \cap V_{\mathbf{j-1}} = \{\mathbf{0}\}, \quad V_{\mathbf{0}} = W_{\mathbf{0}} \tag{1.17}$$

From this follows the representation

$$V_{\mathbf{j}} = \bigoplus_{\mathbf{i} \leq \mathbf{j}} W_{\mathbf{i}}. \tag{1.18}$$

This means that every function $u \in V_{\mathbf{j}}$ can be uniquely expressed as

$$u(\mathbf{x}) = \sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{j}} \sum_{\mathbf{k} \in \nabla_{\mathbf{i}}} u_{\mathbf{i},\mathbf{k}} \, \phi_{\mathbf{i},\mathbf{k}}(\mathbf{x}) \tag{1.19}$$

with *hierarchical coefficients* $u_{\mathbf{i},\mathbf{k}} \in \mathbb{R}$.

There is one more concept and the associated definitions to consider, and that is the assumed approximation property

$$\inf_{f_{j_i} \in V_{j_i}} \|f - f_{j_i}\|_{H^q(\Omega_i)} \lesssim h_{j_i}^{s-q} \|f\|_{H^s(\Omega_i)} \quad f \in H^s(\Omega_i), \tag{1.20}$$

where $q < \gamma_i$ and $q \leq s \leq r_i$ uniformly in j.

We therefore define the *mesh width* $h_{j_i} = 2^{-j_i}$ for the space V_{j_i} over Ω_i; the *regularity* of the functions contained in said space defines

$$\gamma_i = \sup\{s \in \mathbb{R} : V_{j_i} \subset H^s(\Omega_i)\} \tag{1.21}$$

and $r_i \in \mathbb{N}^+$ refers to the *polynomial exactness*, which is the highest degree of polynomials that is contained locally in V_{j_i}.

Any general basis we may wish to consider has to fulfil this property, otherwise our approximation estimates (which will follow in later sections) do not hold. This would invalidate all results.

Since in the following we want to consider various function spaces in the context of grids we formally write:

Definition 3 (Full Grid)**.** The *(regular) full grid* of level n is the space

$$\mathbf{V}_n := V_{\mathbf{n}} = \bigoplus_{|\mathbf{l}|_\infty \leq n} W_{\mathbf{l}}.$$

1.2.1 Regular Sparse Grids

Up to this point all considered spaces were full tensor product spaces. Now we proceed to introduce the regular *sparse tensor product spaces* often just called sparse grids. We again orient ourselves at [12] to define *regular sparse grids.* Further introduction can be found in [38, 9].

In the following the regular sparse tensor product space will be denoted by a hat over the space giving us:

Definition 4 (Sparse Grid)**.** The *regular sparse grid* of level n is the space defined by

$$\hat{V}_n = \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}}. \tag{1.22}$$

This gives us again a unique basis representation for each function $\hat{u} \in \hat{V}_n$:

$$\hat{u}(\mathbf{x}) = \sum_{|\mathbf{l}|_1 \leq n+d-1} \sum_{\mathbf{k} \in \nabla_{\mathbf{l}}} u_{\mathbf{l},\mathbf{k}} \, \phi_{\mathbf{l},\mathbf{k}}(\mathbf{x}) \tag{1.23}$$

with hierarchical coefficients $u_{\mathbf{l},\mathbf{k}} \in \mathbb{R}$.

With this information it is possible to calculate the degrees of freedom of these two spaces. For the regular sparse grid (always without boundaries) we have

$$\begin{aligned}\left|\hat{V}_n\right| &= (-1)^d + 2^n \sum_{i=0}^{d-1} \binom{n+d-1}{i} (-2)^{d-1-i} \\ &= 2^n \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2}) \right) \\ &= \mathcal{O}\left(2^n n^{d-1}\right) \end{aligned} \tag{1.24}$$

degrees of freedom. Comparatively, for the full grid we have $|\mathbf{V}_n| = (2^n - 1)^d = \mathcal{O}\left(2^{nd}\right)$, meaning the sparse grid reduces the degrees of freedom and therefore the necessary storage space and computations significantly. Even more details can be found in [7, 20].

Motivation [12] To understand why these spaces are defined as they are, one has to consider functions with bounded mixed second derivatives as considered in subsection 1.1. Regular sparse grids are optimized for the approximation of these functions $f \in H^{(2,2)}_{\text{mix}}(\Omega)$.

It holds for functions $f \in H^{(2,2)}_{\text{mix}}(\Omega)$ that their hierarchical coefficients $f_{\mathbf{l},\mathbf{i}}$ (as defined in (1.19)) decay as $|f_{\mathbf{l},\mathbf{i}}| = \mathcal{O}\left(2^{-2|\mathbf{k}|_1}\right)$. This is due to Lemma 1.

On the other hand the hierarchical increment spaces have $|W_{\mathbf{k}}| = \mathcal{O}\left(2^{|\mathbf{k}|_1}\right)$ many degrees of freedom. Optimizing these two quantities leads to the spaces defined above.

That the definition is reasonable can be determined by comparing the degrees of freedom calculated above and comparing this to the approximation accuracy for functions $f \in H^{(2,2)}_{\text{mix}}(\Omega)$. We get for the sparse grid an estimate of

$$\left\| f - \hat{f}_n \right\|_{L^2(\Omega)} \lesssim 2^{-2n} \cdot n^{d-1}$$

and by comparison for the full grid

$$\|f - f_n\|_{L^2(\Omega)} \lesssim 2^{-2n}.$$

Here $\hat{f}_n, f_n$ are the approximations of f in $\hat{V}_n, \mathbf{V}_n$ respectively. It is important to note that these estimates use the fact that $f \in H^{(2,2)}_{\text{mix}}(\Omega)$, for other function spaces there would be different rates. The proof for the rate of convergence is formulated later on in Theorem 8.

The above concludes that the definition of the regular sparse grid justifies itself by the significant reduction of degrees of freedom and therefore storage space and calculations, with only a minor worsening of the approximation accuracy.

1.2.2 Specialized Sparse Grids

There are many ways of specializing the sparse grid depending on the kind of problem one wishes to solve. Generally speaking this is achieved by changing the index-set which determines which basis functions of the full grid space are chosen for the sparse grid space.

To get more detailed, consider

$$\hat{V}_n = \bigoplus_{\mathbf{l}\in\mathcal{I}_n^S} W_{\mathbf{l}} \tag{1.25}$$

where

$$\mathcal{I}_n^S = \left\{\mathbf{l}\in\mathbb{N}^d \,|\, |\mathbf{l}|_1 \leq n+d-1\right\}, \tag{1.26}$$

comparing with (1.22) leads to the conclusion that this is indeed consistent with Definition 4.

$\mathcal{I}_n^S$ is the index-set for the regular sparse grid of level n. It defines the regular sparse grids we will mostly be using. As discussed above this index-set is chosen in such a way to include the "more important" (or "more significant") basis functions and exclude the less "important" or "significant" ones in order to optimize the cost-benefit quotient. As already discussed in the section above this is done mostly for functions $f \in H_{mix}^{(2,2)}$.

But, as said above, this index-set can be modified for settings in which the importance (or significance) of the basis functions is weighted differently, for example for $f \in \mathcal{H}$ where $\mathcal{H} \subset L^2(\Omega)$ is some special space. Depending on the setting, optimizing cost-benefit can therefore lead to different index-sets.

First, we can define the generalized sparse grid [22, 10] (for sparse quadrature instead of approximation also [25]), which is just

$$\hat{V}_n^g = \bigoplus_{\mathbf{l}\in\mathcal{I}_n^g} W_{\mathbf{l}}$$

with the general index-set $\mathcal{I}_n^g$ of level n, which has to fulfil an admissibility condition:

$$\mathbf{k}\in\mathcal{I}_n^g \text{ and } \mathbf{l}\leq\mathbf{k} \Rightarrow \mathbf{l}\in\mathcal{I}_n^g. \tag{1.27}$$

This condition guarantees that the whole idea of the telescoping sum construction still works. Multiple examples for different potential choices of this index-set are described in [25, Chapter 4] for quadrature instead of projection. There also a short introduction to dimension-adaptive sparse grids can be found.

Both the regular sparse grid and the one we introduce below satisfy the condition and are therefore sparse grids.

The second one is found in much of the literature cited in this work [16, 17, 18]: the mixed sparse grid space $\hat{V}_J^\sigma$. It is defined over the domain $\Omega_1 \times \Omega_2$, where, although these could be higher-dimensional, we set $\Omega_1 \subset \mathbb{R}$ and $\Omega_2 \subset \mathbb{R}$. Then we have for a $\sigma > 0$:

$$\hat{V}_J^\sigma = \bigoplus_{l_1\sigma + \frac{l_2}{\sigma} \leq J} W_{\mathbf{l}}. \tag{1.28}$$

Since σ does not have to be a natural number, the relevant summation limits do not have to be either. For sake of readability the, for strict formality necessary, floor- and ceiling-brackets are omitted here.

We have that

$$\hat{V}_J^1 = \hat{V}_J$$

by definition, and the index-set for the mixed sparse grid space is

$$\mathcal{I}_{\sigma,J}^{Smix} = \left\{ \mathbf{l} \in \mathbb{N}^2 \middle| l_1\sigma + \frac{l_2}{\sigma} \leq J \right\}.$$

There are different sensible choices for the parameter σ, depending on ones particular setting. These mixed sparse grid spaces where first introduced in [16], where they are called sparse tensor product spaces.

Although we do not utilize these specialized sparse grids in this work, they show a possibility of further optimization, depending on setting. Together with potential dimension-adaptivity the specialized sparse grids are a possible future topic of study.

2 Projection

In the field of Numerics, almost every problem posed needs to be approximated in an appropriate finite dimensional subspace to be practically solvable. Now that we have defined the spaces we need, the next step is to consider the projections onto these spaces and their properties.

2.1 L^2-Projection

The first kind of projection we consider is the L^2-orthogonal projection onto a finite dimensional space. It is a standard tool in the field of mathematics. We will in the following first define it in one dimension and make note of some of its properties. Then we use a tensor product approach to generalize it to higher dimensions. Special attention will be paid to the question of how to compute the projection in praxis, as this is one of the major calculation steps of the format introduced in this work.

2.1.1 Definitions in one dimension

First we define all necessary concepts in one dimension. As such we consider the domain $\Omega \subset \mathbb{R}$, which is $\Omega = [0,1]$ without loss of generality.

Definition 5. The $L^2(\Omega)$-orthogonal projection onto the space V_l is given by

$$Q_l : L^2(\Omega) \longrightarrow V_l. \tag{2.1}$$

It is defined by the so-called *projection identity*:

$$\langle f - Q_l f, g\rangle_{L^2(\Omega)} = 0 \quad \forall g \in V_l \tag{2.2}$$

for an $f \in L^2(\Omega)$.

Q_l is a projection with all the characteristics of one, meaning $(Q_l)^2 = Q_l$. Additionally, since it is also an orthogonal projection it holds: $(Q_l)^\star = Q_l$.

We know that Q_l is a bounded operator, i.e.

$$\|Q_l f\|_{H^s(\Omega)} \lesssim \|f\|_{H^s(\Omega)} \tag{2.3}$$

holds for $|s| \leq \gamma$ where γ is the maximum regularity defined in (1.21).

The projections also give rise to a multilevel decomposition of the function f:

$$f = \sum_{j=0}^{\infty}(Q_j - Q_{j-1})f = \sum_{j=0}^{\infty} f_j. \tag{2.4}$$

Here $Q_{-1} := 0$ and we know $f_j := (Q_j - Q_{j-1})f \in W_j$.

Furthermore we can formulate a L^2-orthogonality as a direct consequence of (2.2):

$$\langle (Q_j - Q_{j-1})\, f, v\rangle_{L^2(\Omega)} = 0 \quad \forall v \in V_{j-1}. \tag{2.5}$$

Due to this L^2-orthogonality, we also have the equality

$$\|f\|^2_{L^2(\Omega)} = \left\| \sum_{j=0}^{\infty}(Q_j - Q_{j-1})f \right\|^2_{L^2(\Omega)} = \sum_{j=0}^{\infty} \|(Q_j - Q_{j-1})f\|^2_{L^2(\Omega)}. \tag{2.6}$$

The final simple estimate needed is induced by the approximation property (1.20) and will later give us our rate of convergence:

$$\|(Q_j - Q_{j-1})f\|_{H^s(\Omega)} \lesssim 2^{-j(t-s)} \|f\|_{H^t(\Omega)} \tag{2.7}$$

where $s < t \leq r$ and r is the polynomial exactness of the approximation space.

2.1.2 Generalization to higher Dimensions

All this generalizes into multiple dimensions by setting $\Omega = \Omega_1 \times \cdots \times \Omega_m$ and considering a tensor product approach.

Definition 6 (L^2-projection)**.** We denote the $L^2(\Omega)$-orthogonal projection onto the subspace $V_{\mathbf{l}} \subset L^2(\Omega)$ as

$$Q_{\mathbf{l}} : L^2(\Omega) \longrightarrow V_{\mathbf{l}}. \tag{2.8}$$

It is defined by

$$Q_{\mathbf{l}} = Q_{l_1} \otimes \ldots \otimes Q_{l_m}. \tag{2.9}$$

Due to the tensor product approach the properties of the projection, namely being idempotent, self-adjoint and bounded translate directly.

It is very important to note that the tensor product approach makes the different directions (the Q_{l_i}) completely independent of each other. Most operations one wishes to perform on the projection can therefore be calculated for each (one-dimensional) direction separately from the others and then the results can be combined together by the rules of the tensor product.

The above is the way we generalize the estimate (2.7). Here it is important to note that the space $H^s(\Omega)$ does not in fact contain the space $H^s(\Omega_1) \otimes \cdots \otimes H^s(\Omega_m)$, as such we need to consider the H-mix spaces introduced in the section 1.1 before. For simplicity of writing we consider two dimensions, more generalize analogously.

Lemma 1 (Approximation property)**.** *For a function* $f \in L^2(\Omega_1 \times \Omega_2)$ *and* $0 < t_1 \leq r_1$ *and* $0 < t_2 \leq r_2$ *the following* approximation property *holds*

$$\|(Q_{\mathbf{j}} - Q_{\mathbf{j-1}})f\|_{L^2(\Omega_1\times\Omega_2)} \lesssim 2^{-(j_1t_1+j_2t_2)} \|f\|_{H^{(t_1,t_2)}_{mix}(\Omega_1\times\Omega_2)}. \tag{2.10}$$

Here r_1, r_2 are the polynomial exactness of the spaces V_{l_1}, V_{l_2} respectively.

Proof. We follow the procedure described above and formulate

$$Q_{\mathbf{j}} - Q_{\mathbf{j-1}} = (Q_{j_1} - Q_{j_1-1}) \otimes (Q_{j_2} - Q_{j_2-1}).$$

Then, since w.l.o.g. f does not exist in tensor product form, we formulate

$$(Q_{j_1} - Q_{j_1-1}) \otimes (Q_{j_2} - Q_{j_2-1}) = (\mathrm{Id} \otimes (Q_{j_2} - Q_{j_2-1}))((Q_{j_1} - Q_{j_1-1}) \otimes \mathrm{Id})$$

to use (2.7):

$$\begin{aligned}
\|(Q_{\mathbf{j}} - Q_{\mathbf{j-1}})f\|_{L^2(\Omega_1\times\Omega_2)} &= \|(\mathrm{Id} \otimes (Q_{j_2} - Q_{j_2-1}))((Q_{j_1} - Q_{j_1-1}) \otimes \mathrm{Id})f\|_{L^2(\Omega_1\times\Omega_2)} \\
&\overset{(2.7)}{\lesssim} 2^{-j_2t_2} \|((Q_{j_1} - Q_{j_1-1}) \otimes \mathrm{Id})f\|_{H^{(0,t_2)}_{\mathrm{mix}}} \qquad (2.11) \\
&\overset{(2.7)}{\lesssim} 2^{-(j_1t_1+j_2t_2)} \|f\|_{H^{(t_1,t_2)}_{\mathrm{mix}}} \qquad (2.12)
\end{aligned}$$

where we twice used the one-dimensional estimate (2.7) and the definition of $H^{(t_1,t_2)}_{\mathrm{mix}}$ (see section 1.1) as a tensor product space. □

The proof above is meant as an example for one of the main concepts in this work: working with tensor products. Later we will also consider more general tensors. One other concept introduced above is the usefulness of the following:

Definition 7 (Detail Projection)**.** For the L^2-projection $Q_{\mathbf{l}} : L^2 \to V_{\mathbf{l}}$ we define the *detail projection* by

$$\Delta_{\mathbf{l}}^{Q} := (Q_{l_1} - Q_{l_1-1}) \otimes (Q_{l_2} - Q_{l_2-1}) = (Q_{\mathbf{l}} - Q_{\mathbf{l-1}}) \tag{2.13}$$

for two dimensions. Higher dimensions generalize.

It is important to remember that despite the name, the detail projection is in general not actually a projection. Only when we have chosen a basis in such a way that the $W_{\mathbf{l}}$ are orthogonal to each other it is indeed a projection.

With this we can generalize (2.4) to the multilevel decomposition of the function $f \in L^2(\Omega_1 \times \Omega_2)$ given by

$$f = \sum_{\mathbf{l}=\mathbf{0}}^{\infty} \Delta_{\mathbf{l}}^{Q} f \tag{2.14}$$

with $\Delta_{\mathbf{l}}^{Q} f \in W_{l_1} \otimes W_{l_2}$. Furthermore we can generalize (2.5) to formulate the important L^2-orthogonality property:

$$\left\langle \Delta_{\mathbf{l}}^{Q} f, v \right\rangle_{L^2(\Omega)} = 0 \quad \forall v \in V_{\mathbf{l-1}}. \tag{2.15}$$

Then we can use the above to formulate an important estimate which will be used in convergence proofs later:

Theorem 8. *For a function $f \in H_{mix}^{(s_1,s_2)}$ with $0 < s_1 \leq r_1$ and $0 < s_2 \leq r_2$ the approximation error can be bounded by*

$$\|f - Q_{\mathbf{l}} f\|_{L^2(\Omega)}^2 \lesssim 2^{-2(l_1 s_1 + l_2 s_2)} \|f\|_{H_{mix}^{(s_1,s_2)}}^2 . \tag{2.16}$$

The proof of it follows directly from Lemma 1, using the following estimates:

$$\begin{aligned} \|f - Q_{\mathbf{l}} f\|_{L^2(\Omega)}^2 = \left\| \sum_{\mathbf{j}>\mathbf{l}} \Delta_{\mathbf{j}}^{Q} f \right\|_{L^2(\Omega)}^2 &\overset{\text{orth}}{=} \sum_{\mathbf{j}>\mathbf{l}} \left\| \Delta_{\mathbf{j}}^{Q} f \right\|_{L^2(\Omega)}^2 \\ &\lesssim \sum_{\mathbf{j}>\mathbf{l}} 2^{-2(j_1 s_1 + j_2 s_2)} \|f\|_{H_{mix}^{(s_1,s_2)}}^2 \\ &\lesssim 2^{-2(l_1 s_1 + l_2 s_2)} \|f\|_{H_{mix}^{(s_1,s_2)}}^2 . \end{aligned}$$

rank-r functions Since we will need it later, in the following paragraphs we will consider the projections of rank-r functions. We will arrive at the result that the projection of a rank-r function is itself again at most of rank r.

First we consider the rank-1 function $f = \phi \otimes \psi \in L^2(\Omega_1 \times \Omega_2)$ with which we can formulate

$$Q_{\mathbf{l}} f = \left(Q_{l_1} \otimes Q_{l_2}\right)(\phi \otimes \psi) = Q_{l_1}\phi \otimes Q_{l_2}\psi.$$

This is again a rank-1 function (here a rank-1 function is simply one that can be written in tensor product form).

A rank-r function is one that can be written as a sum of at least linearly independent rank-1 functions with r summands. This means that in a second step we consider $f = \sum_{\alpha=1}^{r} \phi_\alpha \otimes \psi_\alpha \in L^2(\Omega_1 \times \Omega_2)$, which is indeed a rank-r function. Since $Q_{\mathbf{l}}$ is linear it holds

$$Q_{\mathbf{l}} f = \sum_{\alpha=1}^{r} Q_{l_1}\phi_\alpha \otimes Q_{l_2}\psi_\alpha. \tag{2.17}$$

This gives us the wanted result: The projection of a rank-r function has at most rank r.

(We can note here that it might have lower rank, depending on the function and the space to be projected onto. It is in general not apparent whether one or more of the $Q_{l_1}\phi_\alpha \otimes Q_{l_2}\psi_\alpha$ are linearly dependent on those before them, which would result in a lower rank.)

2.1.3 Projection onto Sparse Grid Space

Above we have considered the projection onto the full grid space $V_{\mathbf{l}}$. There is however no reason why we cannot project onto the sparse grid space $\hat{V}_n$ as well. We again use a L^2-orthogonal projection, it is only the projected space that we change. In the following we will briefly define this projection we call $\hat{Q}$, and detail some of its properties and differences from Q.

Definition 9. For a level $n \in \mathbb{N}$, the L^2-*projection onto the regular sparse grid space* is the operator

$$\hat{Q}_n : L^2(\Omega) \longrightarrow \hat{V}_n. \tag{2.18}$$

It is defined analogously to (2.2) by the *projection identity*:

$$\left\langle f - \hat{Q}_n f, g \right\rangle_{L^2(\Omega)} = 0 \quad \forall g \in \hat{V}_n$$

for an $f \in L^2(\Omega)$.

In contrast to the full grid projection Q, the sparse grid version does not have a simple product structure (recall (2.9)). This is because the underlying space, in this case $\hat{V}_n$, does not have a simple product structure. However, we know by definition

$$\hat{V}_n = \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}} \overset{(1.17)}{=} \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} \left(V_{\mathbf{l}} \ominus V_{\mathbf{l}-\mathbf{1}}\right).$$

Similarly to Theorem 8 we can formulate an estimate for the remainder term:

Theorem 10. *For a function $f \in H_{mix}^{(s_1,s_2)}$ with $0 < s_1 \leq r_1$ and $0 < s_2 \leq r_2$ the remainder term can be bounded by*

$$\sum_{j_1+j_2>J} \left\| \Delta_{\mathbf{j}}^Q f \right\|_{L^2(\Omega_1\times\Omega_2)}^2 \lesssim \begin{cases} 2^{-2J\min(s_1,s_2)} \left\| f \right\|_{H_{mix}^{(s_1,s_2)}}^2, & s_1 \neq s_2 \\ 2^{-2Js_1} J \left\| f \right\|_{H_{mix}^{(s_1,s_2)}}^2, & s_1 = s_2 \end{cases}. \tag{2.19}$$

Here and in the estimates building on this one it can be further generalized for a Hilbert space $\mathcal{H}(\Omega_1 \times \Omega_2) \subset L^2(\Omega_1 \times \Omega_2)$ (on the left side) and the associated $\mathcal{H}_{mix}^{(t_1,t_2)}(\Omega_1 \times \Omega_2)$ on the right hand side. However, in that case we need to set $0 < t_1 \leq p_1$ and $0 < t_2 \leq p_2$, where the $0 < p_1, p_2$ are the largest numbers for which $\mathcal{H}_{mix}^{(p_1,0)} \subset H_{mix}^{(r_1,r_2)}(\Omega_1\times\Omega_2)$ and $\mathcal{H}_{mix}^{(0,p_2)} \subset H_{mix}^{(r_1,r_2)}(\Omega_1\times\Omega_2)$ holds. This is especially useful in the case of Galerkin-projections as defined in section 2.2. More details in [17].

To proof this we again use Lemma 1:

$$\sum_{j_1+j_2>J} \left\| \Delta_{\mathbf{j}}^Q f \right\|_{L^2(\Omega_1\times\Omega_2)}^2 \lesssim \sum_{j_1+j_2>J} 2^{-2(j_1 s_1 + j_2 s_2)} \left\| f \right\|_{H_{mix}^{(s_1,s_2)}}^2.$$

The rest of the proof for this is rather technical, since the two directions (dimensions) of the sum are not independent of each other as before, and can be found in [16, Theorem 4.3].[2]

2.1.4 Practical Computation

In the section above, more specifically Definition 5 and 6, we have defined the L^2-Projection. However, from the point of practical computation, we notice that the projection is not defined directly, but rather via the projection identity (2.2).

This creates the problem of how to actually compute the L^2-Projection, and since we need to do it fairly often, how to do it efficiently. Here it is

[2]In [16] what we denote by $\Delta_{\mathbf{j}}^Q$ is instead written as $Q_{j_1}^{(1)} \otimes Q_{j_2}^{(2)}$ and not to be confused with our projection Q.

very important to note the tensor product structure of the projection shown in (2.9). As already noted above, due to this structure we can calculate the one-dimensional projections in all directions and then simply multiplicate them. In the following we will therefore show how to calculate the projection in one dimension.

If we wish to calculate the L^2-projection of f onto $V_l(\Omega)$, we consider the basis of this space. In the following we will denote it by $\left\{\phi_i, i = 1, \dots, 2^l - 1\right\}$. This basis is in general arbitrary (as long as it is a basis of the space $V_l(\Omega)$), although a natural choice is a hierarchical basis as defined in the previous sections.

Then we know that an equation holds for all functions in a given space, if it holds for all basis functions of said space. Also we know that every function in $V_l(\Omega)$ has a unique basis representation, meaning $Q_l f = \sum_{i=1}^{2^l-1} x_i \phi_i$ with $\mathbf{x} = (x_i)_{i=1}^{2^l-1}$ dependent on f.

As such we can reformulate (2.2) as

$$\sum_{j=1}^{2^l-1} x_j \langle \phi_j, \phi_i \rangle = \langle f, \phi_i \rangle \quad \forall i \in \left\{1, \dots, 2^l - 1\right\} \tag{2.20}$$

or equally as

$$\mathbf{M}\mathbf{x} = \mathbf{b} \tag{2.21}$$

where $\mathbf{M} = \left(\langle \phi_i, \phi_j \rangle_{L^2(\Omega)}\right)_{i,j=1}^{2^l-1}$ is the so-called mass matrix and $\mathbf{b} = \left(\langle f, \phi_i \rangle_{L^2(\Omega)}\right)_{i=1}^{2^l-1}$ the right side vector.

As $\mathbf{M}$ and $\mathbf{b}$ are given and we want to determine $\mathbf{x}$, the problem of determining the L^2-Projection has been reformulated as solving a linear system. This is a well-known problem in Numerics, how exactly one solves this is out of the scope of this work.

For our purposes it is important to note that to calculate a $m-$dimensional L^2-Projection onto $V_{\mathbf{l}}(\Omega_1 \times \cdots \times \Omega_m)$, one needs to solve m many linear systems with $2^{l_i} - 1,\ i = 1, \dots, m$ variables each. If the basis is chosen well, the cost of solving the system (2.21) (or analogously inverting the matrices $\mathbf{M}$) can be greatly reduced compared to an arbitrary system. There are bases (for example those whose mass matrix is a tridiagonal one), where one linear system can be solved in $\mathcal{O}(2^l)$ operations, with l being the level of the space in in the considered direction.

If we denote the cost of solving the linear system (2.21) by $K(l)$, dependent on the level l, then we can formulate a rough estimate for the calculation of the L^2-Projection. For an arbitrary $\mathbf{l}$ we can make no general statements about the value of the l_i, therefore we will estimate with $\max_{1 \le i \le m} l_i$, which can make the estimate quite rough for extreme cases (for example if $\mathbf{l} = (L, 1, 1, 1)$ with L very large).

The cost of calculating the L^2-Projection of a function f onto the space $V_{\mathbf{l}}(\Omega_1 \times \cdots \times \Omega_m)$ can be estimated by

$$\mathcal{O}\left(m \cdot \max_{1\leq i\leq m} K(l_i)\right). \tag{2.22}$$

As such the calculation does not suffer from the curse of dimensionality, since m does not appear in the exponent. (It is however important to note that the term $K(l_i)$ contains the term 2^{l_i} in some form, since this is (asymptotically) the number of variables of the system.) The importance of the tensor product structure can not be understated, without exploiting this structure we would arrive at a cost of $\mathcal{O}\left(\max_{1\leq i\leq m} K(m \cdot l_i)\right)$, which does have the dimension in the exponent and is therefore subject to the curse of dimensionality.

2.2 Galerkin Projection

This linear system above is the discretized form of the operator equation $Ax = f$ where A is the identity operator. Now we can of course consider operators A (mostly differential operators) that are not the identity. Then we arrive at the subject of Galerkin Projections. More detailed discussions of this very complex topic can be found in most introductory literature for the subject Numerics [3].

For simplification purposes we consider a differential form $A : \mathcal{H} \longrightarrow \mathcal{H}^*$ over a Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \ldots \otimes \mathcal{H}_m$[3], which is continuous and $\mathcal{H}-$elliptic. We will also restrict ourselves to operators of the form $A = A_1 \otimes \ldots \otimes A_m$, where for $i = 1\ldots, m$, $A_i : \mathcal{H}_i \longrightarrow \mathcal{H}_i^*$. The reason for this are analogous to the complexity estimates from the section before and will be made clear in section 3.2.

Following from that we can define a bilinear form

$$a(u,v) = (Au, v)_{L^2(\Omega)} : \mathcal{H} \times \mathcal{H} \longrightarrow \mathbb{R} \tag{2.23}$$

which is also continuous and elliptic.

We now solve $Au = f$ for a given $f \in \mathcal{H}^*$ by considering the variational formulation:

$$\text{“ Find } u \in \mathcal{H} \text{ s.t. } a(u,v) = f(v) \quad \forall v \in \mathcal{H}. \text{ ”} \tag{2.24}$$

To actually practically solve this all functions in $\mathcal{H}$ (that would be u and v) are to be approximated in the finite dimensional space $V_{\mathbf{l}}$.

This now characterizes the *Galerkin Projection*

$$P_{\mathbf{l}} : \mathcal{H} \longrightarrow V_{\mathbf{l}} \tag{2.25}$$

[3]This very naturally leads to $\mathcal{H}$ being a H-mix space.

which is orthogonal in the sense of *Galerkin orthogonality*

$$a(u - P_{\mathbf{l}}\, u, v) = 0 \quad \forall v \in V_{\mathbf{l}} \tag{2.26}$$

which in comparison to the L^2-orthogonality replaces the standard scalar product by the bilinear form $a(\cdot,\cdot)$. All this is well defined due to the characteristics of said bilinear form. More details can be found in most introductions to Numerics, for example [3].

In analogy with the definition of $\Delta_{\mathbf{l}}^{Q}$ we now define (again for two dimensions)

$$\Delta_{\mathbf{l}}^{P} = P_{l_1,l_2} - P_{l_1-1,l_2} - P_{l_1,l_2-1} + P_{l_1-1,l_2-1}. \tag{2.27}$$

Again, it generalizes for higher dimensions and is in fact equivalent to the tensor product form we used to define (2.13), since A also exists in tensor product form. As such we can also set

$$\Delta_{\mathbf{l}}^{P} = P_{\mathbf{l}} - P_{\mathbf{l}-\mathbf{1}} = (P_{l_1} - P_{l_1-1}) \otimes (P_{l_2} - P_{l_2-1}),$$

but only for the discussed special case.

Also an orthogonal property exists, generalizing (2.15), but in the sense of Galerkin orthogonality:

$$a\left(\Delta_{\mathbf{l}}^{P} u, v\right) = 0 \quad \forall v \in V_{\mathbf{l}-\mathbf{1}}. \tag{2.28}$$

2.3 Linear vs. Standard Information

Both of the projections discussed above have in common that they are indeed projections (and orthogonal at that). Projecting onto a subspace $V_{\mathbf{l}}$ is of course not the only method to approximate a function $f \in \mathcal{H}$, another well known method is interpolation.

If we were to consider interpolation, we would define (for two dimensions, more generalizes):

$$f_{\mathbf{j}}^{\text{int}} = \sum_{\mathbf{k} \in \varDelta_{\mathbf{j}}} c_{\mathbf{k}}\ \phi_{\mathbf{j},\mathbf{k}}$$

where $\Phi_{\mathbf{j}} = \{\phi_{\mathbf{j},\mathbf{k}} : \mathbf{k} \in \varDelta_{\mathbf{j}}\}$ is a basis of $V_{\mathbf{j}}$ as in section 1.2. Here,

$$c_{\mathbf{k}} = f(x_{k_1}, y_{k_2})$$

are the function evaluations at points (x_{k_1}, y_{k_2}) for each $\mathbf{k} \in \varDelta_{\mathbf{j}}$.

There is an expansive theory on which are the best points to choose depending on your setting, it is to this day an area of research. If we consider the hat-basis from our example in section 1.2, the chosen points of evaluation are (almost always) the hat-points of the hats, meaning the grid points on a uniform mesh.

More generally, projection is referred to as *linear information* (consider that we calculate it via solving a linear system) and interpolation is called *standard information.* These two things are very much not the same and in which ways they relate to each other is still a topic of research. Since the L^2-projection is the best-approximation with respect to the L^2-norm, we know that the interpolation has to be an upper bound on the approximation error.

Since this is not a work about the choice of interpolation points we will in the following not consider interpolation and remain with the linear information as defined above and below, since approximation via singular-value-decomposition is also linear information. The main takeaway from this section should therefore be to not confuse the projections with interpolation, since results for one are different from results for the other.

Part II
Results in two Dimensions

Having now defined the underlying basics, we can proceed to introduce our new Combi-format for the approximation of functions $f \in \mathcal{H}$ in finite dimensional subspaces. The idea is to compress the combination technique [23] using singular value decomposition.

Therefore both of these concepts will be introduced in the following sections 3 and 4 respectively. Then we are finally ready to define the Combi-format in section 6, show that it has good approximation rates in section 6.3 and then demonstrate this in praxis in section 7.

As the name of this second part suggests, we start by defining the Combi-format (and all the underlying concepts necessary) in only two dimensions. This means that we set $d = 2$ for everything already defined in the previous part. We consider bivariate functions $f \in \mathcal{H}(\Omega_1 \times \Omega_2) \subset L^2(\Omega_1 \times \Omega_2)$, where without loss of generality $\Omega_1 \times \Omega_2 = [0,1]^2$. In general we mostly consider $P \equiv Q$, although all results generalize to the setting of Galerkin-projections except those where we specifically note that this is not the case.

3 Combination Technique

The sparse grid approach is an effective method for solving higher dimensional operator equations, which the problem of approximation can be rewritten as. However the needed multilevel basis can be cumbersome in praxis. To circumvent this, the *combination technique* [23] was introduced. It recombines the sum of the basis functions of the sparse grid into a sum of small, anisotropic full grids. These full grids are easier to handle in practical applications. Also a convergence theory has been established especially for the combination technique, see [17].

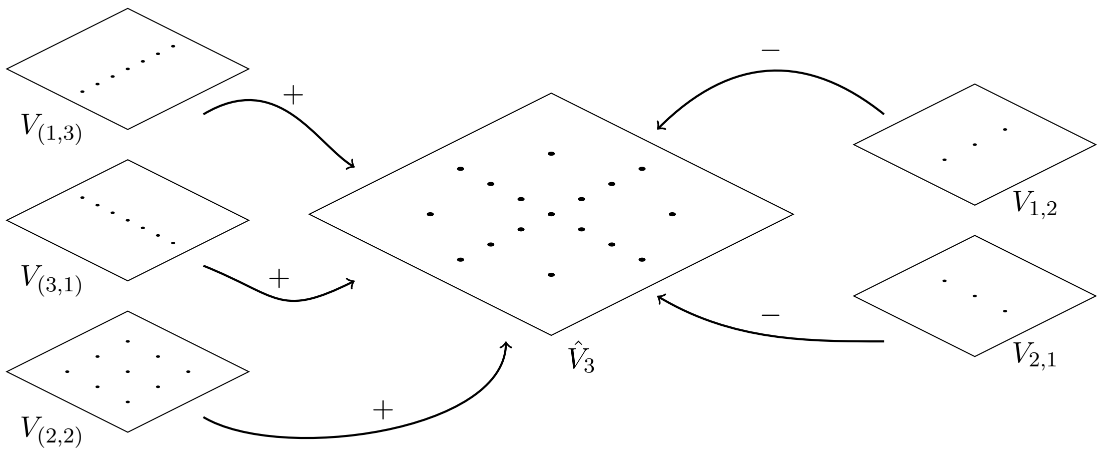


Figure 2: A visualization of the combination technique for the space $\hat{V}_3$.

The Figure 2 illustrates the idea of the combination technique. We add together the anisotropic full grids needed to accumulate all points of the sparse grid and then have to subtract those points that we now have multiple of. More mathematically expressed: Definition 4 gives us the formal definition of a sparse grid of level m as a sum of hierarchical increment spaces, meaning (this is (1.22)):

$$\hat{V}_m = \bigoplus_{|\mathbf{l}|_1 \le m+1} W_{\mathbf{l}}.$$

This can be reformulated as

$$\hat{V}_m = \bigcup_{|\mathbf{l}|_1 = m+1} V_{\mathbf{l}},$$

with the full grid spaces $V_{\mathbf{l}}$. Of course this is now a union of spaces which are not disjoint and therefore not a direct sum, which is what we want (since a direct sum gives us a unique representation, which we need).

Considering the possible level-indices $\mathbf{l} \in \left\{\mathbf{k} \in \mathbb{N}^2 | \, |\mathbf{k}|_1 = m+1\right\}$, to make the spaces disjoint we need to remove

$$\hat{V}_{m-1} = \bigcup_{|\mathbf{l}|_1 = m} V_{\mathbf{l}}.$$

This is exactly what is shown in Figure 2, and how the combination technique is defined in the following.

In only two dimensions the formula for the *combination technique* is given by

$$\hat{u}_m = \mathcal{CT}_m u = \sum_{|\mathbf{l}|_1 = m+1} P_{l_1,l_2} u - \sum_{|\mathbf{l}|_1 = m} P_{l_1,l_2} u. \tag{3.1}$$

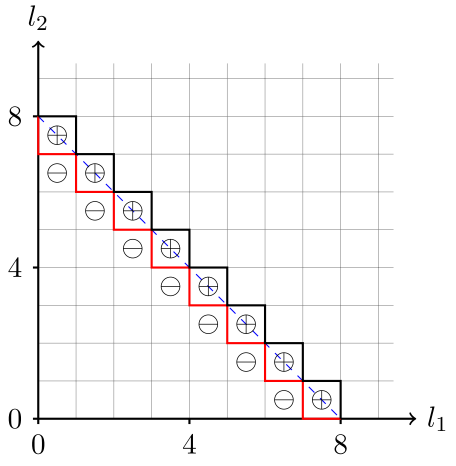


Figure 3: A visualization of the combination technique of level 8. Each box of the grid is associated with its upper right corner, the pluses and minuses show the appearance of the $P_{\mathbf{l}}$ in the combination technique.

The same method is shown in a different visualization in Figure 3, which shows the combination technique of level 8. The dashed blue line shows us $|\mathbf{l}|_1 = m$, it is the line where the sign of the projections changes. Since the level-indices are of course natural numbers in each direction we instead consider the grid and associate each box with its right upper corner. As such the red line actually delimits the area for which $|\mathbf{l}|_1 < m + 1$ holds, whereas the black one delimits $|\mathbf{l}|_1 \leq m + 1$.

The pluses and minuses show the indices $\mathbf{l}$ which appear in the format; if there is a plus in the box associated with $\mathbf{k}$, then $P_{\mathbf{k}}$ appears in the format with positive factor, analogously for the minuses.

This can also be considered as

$$\mathcal{CT}_m u = \sum_{\mathbf{l} \in \mathcal{I}_m^C} (-1)^{m+1-|\mathbf{l}|_1} P_{l_1,l_2} u \tag{3.2}$$

with

$$\mathcal{I}_m^C = \left\{ \mathbf{l} \in \mathbb{N}^2 | m \leq |\mathbf{l}|_1 \leq m + 1 \right\} \tag{3.3}$$

being the index set of the grids of the combination format of level m.

3.1 Proof of Concept

In a previous section we have already defined $\hat{u}_m = \hat{P}_m u$, with the projection onto the sparse grid of level m, $\hat{P}_m$, being defined as in Definition 9. The following provides a proof of concept for the combination technique, by proofing that indeed

$$\hat{u}_m = \mathcal{CT}_m u = \hat{P}_m u$$

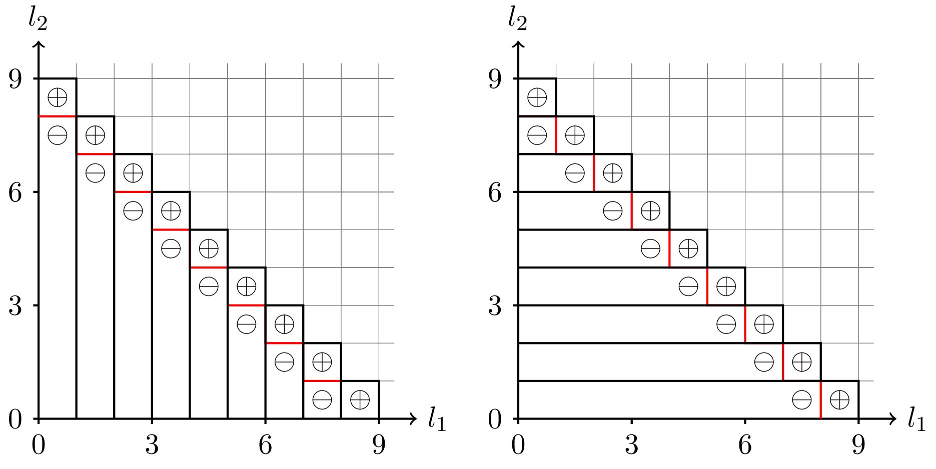


Figure 4: Two different ways to reformulate the formula for the combination technique as described in (3.4). Here the thick black lines are for the sake of visualizing the two possible directions of the subtraction.

holds in a specific setting. (That being $P \equiv Q$ or P being in tensor product form.)

First we rearrange the two-dimensional combination technique into other formulations, which we will need in the following work. A closer look at (3.1), especially the possible level-indices $\mathbf{l}$, leads to the rearrangements in the first line. They are also illustrated in Figure 4, showing the two possible directions to rearrange the sum. Using the detail projection

$$\Delta^P_{l_1,l_2} = P_{l_1,l_2} - P_{l_1,l_2-1} - P_{l_1-1,l_2} + P_{l_1-1,l_2-1}$$

first defined in (2.27) for the Galerkin-projection or (2.13) for the L^2-projection we can rearrange using a telescoping sum which gives us the second line. The third line is based on $u = \sum_{|\mathbf{l}|_1=0}^{\infty} \Delta^P_{l_1,l_2} u$:

$$\mathcal{CT}_m u = \sum_{|\mathbf{l}|_1=m+1} (P_{l_1,l_2} - P_{l_1,l_2-1})u = \sum_{|\mathbf{l}|_1=m+1} (P_{l_1,l_2} - P_{l_1-1,l_2})u \tag{3.4}$$

$$= \sum_{|\mathbf{l}|_1\le m+1} \Delta^P_{l_1,l_2} u \tag{3.5}$$

$$= u - \sum_{|\mathbf{l}|_1> m+1} \Delta^P_{l_1,l_2} u. \tag{3.6}$$

For the following it is very important to recall that we consider either $P \equiv Q$ or P is a Galerkin-projection in tensor product form. In other cases the following theorem does not hold.

Theorem 11. *For a level $m \in \mathbb{N}$ and a function $u \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$ it holds that the combination technique $\hat{u}_m = \mathcal{CT}_m u$ as defined in* (3.1) *is well-defined.*

Proof. This proof is only for the case $P \equiv Q$. We start with what we already know, letting $u \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$ be arbitrary and setting $\hat{u}_m = \hat{P}_m u$ where $\hat{P}$ is the orthogonal projection onto $\hat{V}_m$ as defined in Definition 9.

The first step is to recognize that both $\hat{P}_m : \mathcal{H} \longrightarrow \hat{V}_m$ and $\mathcal{CT}_m : \mathcal{H} \longrightarrow \hat{V}_m$ are operators mapping onto the same space $\hat{V}_m$. Until now we have not formally proven that $\mathcal{CT}_m$ does actually map to $\hat{V}_m$. We prove this now.

The definition of $\hat{V}_m$ (see (1.22)) states that

$$\hat{V}_m = \bigoplus_{|\mathbf{l}|_1 \le m+1} W_{\mathbf{l}}$$

and the detail projection $\Delta_{\mathbf{l}}^P$ maps onto the space $W_{\mathbf{l}}$. (Recall that despite the name the detail projection is not generally an orthogonal projection, it maps onto $W_{\mathbf{l}}$, it does not (generally) project.)

Using the formulation of the combination technique stated in (3.5), $\mathcal{CT}_m = \sum_{|\mathbf{l}|_1 \le m+1} \Delta_{\mathbf{l}}^P$, we conclude the proof that $\mathcal{CT}_m$ maps onto $\hat{V}_m$.

Now, to show that indeed $\hat{u}_m = \mathcal{CT}_m u$, it needs to be proven that $\mathcal{CT}_m u$ is the orthogonal projection of u onto $\hat{V}_m$. Using the projection identity (2.2) and the rearrangement (3.6) we can formulate for all $v \in \hat{V}_m$

$$\begin{aligned}
\langle u - \mathcal{CT}_m u, v \rangle &= \left\langle u - \sum_{|\mathbf{l}|_1 \le m+1} \Delta_{\mathbf{l}}^P u, v \right\rangle \\
&= \left\langle \sum_{|\mathbf{l}|_1 > m+1} \Delta_{\mathbf{l}}^P u, v \right\rangle \\
&= \sum_{|\mathbf{l}|_1 > m+1} \left\langle \Delta_{\mathbf{l}}^P u, v \right\rangle = 0,
\end{aligned}$$

since the detail projections with level-index $|\mathbf{l}|_1 > m+1$ are orthogonal to all functions $v \in \hat{V}_m$.

Since the orthogonal projection is unique and we chose u to be arbitrary, we have now proven that $\hat{u}_m = \mathcal{CT}_m$ and the combination technique is well-defined. $\square$

Again, it is important to note here that we only consider projections $P_{l_1,l_2} = P_{l_1}^{(1)} \otimes P_{l_2}^{(2)}$. There are of course Galerkin-projections which do not have such a tensor product form. For them the above theorem does not generally hold; Galerkin-projection onto the sparse grid space and the combination technique with Galerkin-projections do not generally reach the same result. However, [17] shows that for $\mathcal{H} \subset L^2(\Omega_1 \times \Omega_2) \subset \mathcal{H}^*$ being a

Gelfand triple and the differential operator (which defines the bilinear form which defines the Galerkin-projection) $A : \mathcal{H} \longrightarrow \mathcal{H}^*$ being elliptic, the same rate of convergence of the combination technique can be achieved.

3.2 Properties

To actually use the combination technique, both in praxis and for the estimates later on, we need to consider its properties. Hence we investigate linearity, storage requirements and computational complexity.

The combination technique is linear, that is

$$\mathcal{CT}_m(u+v) = \mathcal{CT}_m u + \mathcal{CT}_m v \tag{3.7}$$

holds. This is easy to proof, since every operation in the definition (3.1) is linear. (Of course $\mathcal{CT}_m(\lambda u) = \lambda \mathcal{CT}_m u$ for $\lambda \in \mathbb{R}$ also holds for the same reason.)

The Theorem 11 justifies the definition of the combination technique, since it is only a rewriting of the sparse grid into sums of full grids. We do this (as previously stated) since we can work with full grids in a practical setting on the computer, where working with a sparse grid would be difficult to impossible depending on the problem.

Storage cost On the downside is also increases the storage space needed. (Although in almost all cases dense matrices are easier to store and address than sparse ones, so a slight increase in storage space does not necessarily make dense storage less efficient. Details depend on implementation.) However we can also note that the storage space needed for the combination technique in two dimensions of level m (so for $\mathcal{CT}_m$) computes to:

$$2 \sum_{l=1}^{\lfloor \frac{m+1}{2} \rfloor} (2^l - 1)(2^{m+1-l} - 1) + 2 \sum_{l=1}^{\lfloor \frac{m}{2} \rfloor} (2^l - 1)(2^{m+1-l} - 1) \sim \mathcal{O}(2^{m+1} m). \tag{3.8}$$

For comparison: The storage requirement of the sparse grid of level m in two dimensions is $\mathcal{O}(2^m m)$, recall section 1.2, especially (1.24).

Since sparse storage comes with a question of efficient addressing of said storage, it can generally be assumed that the additional factor of two in the storage requirements for the combination technique is balanced out by its far more efficient format (in this case meaning dense storage). Therefore the combination technique can be considered a very efficient format, justifying its use not only in the context of this work.

Complexity There is another factor to consider when it comes to practical implementation: complexity. More precisely the question of how computationally difficult it is to compute the combination technique. For the answer of this question it is important to keep the tensor product form of the projection P in mind (which exists for $P \equiv Q$ in the case of L^2-Projection, see also (2.9), and for P being the Galerkin-projection in the case where $A = A_1 \otimes \ldots \otimes A_m$ with the A_i being one-dimensional operators).

We already discussed how to compute $P_\mathbf{l}$ efficiently in section 2.1.4. Now we note that the combination technique in its form as an operator defined by (3.1) is merely a sum of tensor products.

All of these tensor products can be calculated by combining the one-dimensional projections as discussed. Since the projections in each summand project onto different spaces and therefore different bases, we do not actually perform the sum immediately, but rather keep all the grids in storage. Since a lot of operations we want to perform on the combination technique (at least in this work) are linear or bilinear, they can be performed on each grid independently of each other. Only when we actually want to evaluate the function do we perform the outer sum.

This leaves us the the following components of the complexity estimate for the combination technique of level n: First, the projections themselves. Here we have the results from section 2.1.4. In the worst case this means the cost for one projection is $\mathcal{O}(2 \cdot K(n))$ where $K(n)$ is the worst case complexity of inverting the mass matrix of level n. Secondly, there is the sum around the projections. In two dimensions it has $\mathcal{O}(2n)$ summands.

Finally we arrive at a rough estimate for the complexity of computing the combination technique of level n as $\mathcal{O}(4 \cdot nK(n))$. It is a rough estimate since estimating the complexity of the projections uniformly by $K(n)$ ignores the structure of the combination technique which ensures that most projections are cheaper to calculate than this worst case.

The important part of this estimate is to consider how the dimension factors into it: $d = 2$ only appears once in each term, as a factor. Again, section 2.1.4 explains in which way the dimension appears in the estimate for the individual projections. Most importantly, $d = 2$ does not appear in the exponent, as such the curse of dimensionality is broken.

(All of this explains why we only wish to consider $P \equiv Q$ or P is Galerkin-Projection for a differential operator in tensor product form, otherwise the calculation of the projection could have a complexity with d in the exponent and therefore be subject to the curse of dimensionality.)

3.3 Error Estimate

We have an error estimate especially for the combination technique, which takes into account its structure. Above in this section we have shown in Theorem 11 that in our setting the combination technique is equivalent to the projection onto the sparse grid space. As such there already exists a vast convergence theory as that of the convergence of said projections onto sparse grids.

In the following however, we will present another convergence proof, which was first formulated in [17, Theorem 2]. The reason for this is twofold, first we will need the structure of this proof in one of our own proofs in a later section and secondly this proof also generalizes for Projections $P_{\mathbf{l}}$ which do not have a tensor product form. This shows that our theory (although it would become much more cumbersome in praxis) is not limited to the limits we have laid down in this work.

The error estimate below is based on the remainder term, recall Theorem 10.

Theorem 12 (Convergence of the Combination Technique). *For a level $m \in \mathbb{N}$ and a function $u \in H_{mix}^{(s_1,s_2)}(\Omega_1 \times \Omega_2)$ with $0 < s_1 \leq r_1$ and $0 < s_2 \leq r_2$, the combination technique satisfies the error bounds*

$$\|u - \mathcal{CT}_m u\|^2_{L^2(\Omega_1\times\Omega_2)} \lesssim \begin{cases} 2^{-2(m+1)\min(s_1,s_2)} \|f\|^2_{H_{mix}^{(s_1,s_2)}}, & s_1 \neq s_2 \\ 2^{-2(m+1)s_1}(m+1) \|f\|^2_{H_{mix}^{(s_1,s_2)}}, & s_1 = s_2. \end{cases} \tag{3.9}$$

Here r_1, r_2 are the polynomial exactness of the spaces V_{l_1}, V_{l_2} respectively.

Proof. Following (3.6), we have

$$\|u - \mathcal{CT}_m u\|^2_{L^2(\Omega_1\times\Omega_2)} = \left\| \sum_{|\mathbf{l}|_1 > m+1} \Delta^P_{l_1,l_2} u \right\|^2_{L^2(\Omega_1\times\Omega_2)}$$

and following Galerkin orthogonality

$$\left\| \sum_{|\mathbf{l}|_1 > m+1} \Delta^P_{l_1,l_2} u \right\|^2_{L^2(\Omega_1\times\Omega_2)} \sim \sum_{|\mathbf{l}|_1 > m+1} \left\|\Delta^P_{l_1,l_2} u\right\|^2_{L^2(\Omega_1\times\Omega_2)}.$$

Now, if $P \equiv Q$, the rest of the proof is using the estimate for the remainder term (2.19) from Theorem 10.

If however P is a Galerkin-projection, we instead consider

$$\begin{aligned} \|u - \mathcal{CT}_m u\|^2_{L^2(\Omega_1\times\Omega_2)} \lesssim & \sum_{|\mathbf{l}|_1 > m+1} \left\|\Delta^Q_{l_1,l_2} u\right\|^2_{L^2(\Omega_1\times\Omega_2)} \\ & + \sum_{|\mathbf{l}|_1 > m+1} \left\|\left(\Delta^P_{l_1,l_2} - \Delta^Q_{l_1,l_2}\right) u\right\|^2_{L^2(\Omega_1\times\Omega_2)}. \end{aligned} \tag{3.10}$$

The first sum we bound as above. The second one requires a bit of preparation and is very technical. The details can be found in [17]. We arrive at the final bound for the second term also being

$$\sum_{|\mathbf{l}|_1 > m+1} \left\|\left(\Delta^P_{l_1,l_2} - \Delta^Q_{l_1,l_2}\right) u\right\|^2_{L^2(\Omega_1\times\Omega_2)} \lesssim \begin{cases} 2^{-2(m+1)\min(s_1,s_2)} \|f\|^2_{H_{mix}^{(s_1,s_2)}}, & s_1 \neq s_2 \\ 2^{-2(m+1)s_1}(m+1) \|f\|^2_{H_{mix}^{(s_1,s_2)}}, & s_1 = s_2 \end{cases}$$

which finishes the proof. □

4 Low-rank Approximation

Approximating on a sparse grid is only one possible method for efficient approximation. Another one is the low-rank approximation, by which the considered function is approximated by a function with a given, lower rank. In the following section we give meaning to these technical terms.

First, we will consider the *singular value decomposition for bivariate functions* in section 4.1. It is defined by using the Hilbert-Schmidt theorem; the idea for it was first formulated in [36]. Then we will in section 4.2 consider the analogue in a finite dimensional setting (i.e for matrices), which is the well-known *singular value decomposition.* A general overview of the different uses of the singular value decomposition over time can be found in [37].

4.1 Singular Value Decomposition for Bivariate Functions

For sufficiently smooth domains $\Omega_1 \subset \mathbb{R}$ and $\Omega_2 \subset \mathbb{R}$ we want to consider *low-rank decompositions* of functions $f \in L^2(\Omega_1 \times \Omega_2)$, using the ansatz

$$f(\mathbf{x}, \mathbf{y}) \approx f_r(\mathbf{x}, \mathbf{y}) = \sum_{\alpha=1}^{r} \sigma_\alpha \, \phi_\alpha(\mathbf{x}) \, \psi_\alpha(\mathbf{y}),$$

separating $\mathbf{x} \in \Omega_1$ and $\mathbf{y} \in \Omega_2$.

Since most functions cannot be decomposed into a tensor product structure, this ansatz leads us to the best we can do in general: A sum of tensor products. Furthermore, fixing the number of summands r, the so-called *rank* of the function f_r, makes this ansatz practically computable.

Additionally, it is a well known fact [36] that with respect to the rank r of the decomposition, the best possible representation of f in the L^2-sense is the *singular value decomposition*, also sometimes known in this infinite dimensional spaces context as Hilbert-Schmidt decomposition. The singular value decomposition coincides with the ansatz above, how exactly it is calculated is shown below. This makes the σ_α singular values and the $\phi_\alpha(\mathbf{x}), \psi_\alpha(\mathbf{y})$ normalized left- respective right-eigenfunctions.

For a formal definition we need to take a few steps. In preparation we recall the Hilbert-Schmidt theorem [36, 37]. It states that for a Hilbert-space $\mathcal{H}$ and a bounded, compact and self-adjoint operator $A : \mathcal{H} \longrightarrow \mathcal{H}$ there exist a sequence of eigenvalues $|\lambda_\alpha| > 0$ with $\alpha = 1, \ldots, r$ where r is the rank of the operator A. We order the eigenvalues in such a way that the sequence of the absolute values is non-increasing (if necessary repeating eigenvalues if they have a multiplicity greater one), then we also have $\lim_{\alpha \to \infty} \lambda_\alpha = 0$ for $r = \infty$.

Also there exist corresponding eigenfunctions $\phi_\alpha \in \mathcal{H}$ for $\alpha = 1, \ldots, r$ which form an orthonormal basis of the range of A. This means that $A(\phi_\alpha) = \lambda_\alpha \phi_\alpha$ holds for $\alpha = 1, \ldots, r$.

To use this theorem, we now need to define an appropriate operator.

In a first step we define the integral operator

$$\mathcal{S} : L^2(\Omega_1) \longrightarrow L^2(\Omega_2), \quad u \mapsto (\mathcal{S}u)(\mathbf{y}) := \int_{\Omega_1} f(\mathbf{x}, \mathbf{y})\, u(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{4.1}$$

and its adjoint operator

$$\mathcal{S}^\star : L^2(\Omega_2) \longrightarrow L^2(\Omega_1), \quad u \mapsto (\mathcal{S}^\star u)(\mathbf{x}) := \int_{\Omega_2} f(\mathbf{x}, \mathbf{y})\, u(\mathbf{y})\, \mathrm{d}\mathbf{y}. \tag{4.2}$$

These operators are both Hilbert-Schmidt-operators (since $f \in L^2(\Omega_1 \times \Omega_2)$) and therefore well-defined. We use these operators to formulate the relations we wish to achieve:

$$\sigma_\alpha\, \psi_\alpha(\mathbf{y}) = (\mathcal{S}\phi_\alpha)(\mathbf{y}) \text{ and } \sigma_\alpha\, \phi_\alpha(\mathbf{x}) = (\mathcal{S}^\star \psi_\alpha)(\mathbf{x}) \tag{4.3}$$

for the left- and right-eigenfunctions $\left\{\phi_\alpha \in L^2(\Omega_1)\right\}_{\alpha=1}^{\infty}$ and $\left\{\psi_\alpha \in L^2(\Omega_2)\right\}_{\alpha=1}^{\infty}$. Also we point out that the eigenfunctions form orthonormal bases in $L^2(\Omega_1)$ respectively $L^2(\Omega_2)$.

To actually calculate the singular values and proof that they and the eigenfunctions mentioned above exist we however need a self-adjoint integral operator for the Hilbert-Schmidt theorem, which is why we define

$$\mathcal{K} = \mathcal{S}^\star\mathcal{S} : L^2(\Omega_1) \longrightarrow L^2(\Omega_1), \quad u \mapsto (\mathcal{K}u)(\mathbf{x}) := \int_{\Omega_1} k(\mathbf{x}, \mathbf{x}')\, u(\mathbf{x}')\mathrm{d}\mathbf{x}' \tag{4.4}$$

with the kernel (function) k, defined as

$$k(\mathbf{x}, \mathbf{x}') = \int_{\Omega_2} f(\mathbf{x}, \mathbf{y})\, f(\mathbf{x}', \mathbf{y})\mathrm{d}\mathbf{y} \in L^2(\Omega_1 \times \Omega_1). \tag{4.5}$$

It can be proven that the kernel is a symmetric Hilbert-Schmidt kernel (this is equivalent to showing that indeed $k(\mathbf{x}, \mathbf{x}') \in L^2(\Omega_1 \times \Omega_1)$ and $k(\mathbf{x}, \mathbf{x}') = \overline{k(\mathbf{x}', \mathbf{x})}$). Following from this is the fact that $\mathcal{K}$ is again a Hilbert-Schmidt-(integral-)operator, and in contrast to $\mathcal{S}$ a self-adjoint one. This can also be proven the usual way by considering

$$\langle \mathcal{K}u, v \rangle = \langle \mathcal{S}^\star\mathcal{S}u, v \rangle = \langle \mathcal{S}u, \mathcal{S}v \rangle = \langle u, \mathcal{S}^\star\mathcal{S}v \rangle = \langle u, \mathcal{K}v \rangle$$

for arbitrary $u, v \in L^2(\Omega_1)$. Thus the operator $\mathcal{K}$ fulfils the prerequisites for the Hilbert-Schmidt-theorem.

As a consequence of the Hilbert-Schmidt-theorem we can conclude that there exist countably many eigenvalues (of $\mathcal{K}$) $\{\lambda_\alpha\}_{\alpha=1}^{\infty}$ and the associated eigenfunctions $\{\phi_\alpha\}_{\alpha=1}^{\infty}$ form an orthonormal basis in $L^2(\Omega_1)$.

For the analogue for $L^2(\Omega_2)$ we have to change the order of $\mathcal{S}$ and $\mathcal{S}^\star$:

$$\tilde{\mathcal{K}} = \mathcal{S}\mathcal{S}^\star : L^2(\Omega_2) \longrightarrow L^2(\Omega_2), \quad u \mapsto \left(\tilde{\mathcal{K}}u\right)(\mathbf{y}) := \int_{\Omega_2} \tilde{k}(\mathbf{y},\mathbf{y}')\, u(\mathbf{y}')\mathrm{d}\mathbf{y}' \tag{4.6}$$

with symmetric Hilbert-Schmidt kernel

$$\tilde{k}(\mathbf{y},\mathbf{y}') = \int_{\Omega_1} f(\mathbf{x},\mathbf{y})\, f(\mathbf{x},\mathbf{y}')\mathrm{d}\mathbf{x} \in L^2(\Omega_2 \times \Omega_2). \tag{4.7}$$

Analogous by the Hilbert-Schmidt-theorem this gives us countably many eigenvalues $\left\{\tilde{\lambda}_\alpha\right\}_{\alpha=1}^\infty$ and eigenfunctions $\{\psi_\alpha\}_{\alpha=1}^\infty$ which form an orthonormal basis in $L^2(\Omega_2)$.

One can relatively easily prove that the eigenfunctions obtained from the operators $\mathcal{K}$ and $\tilde{\mathcal{K}}$ are indeed the functions in (4.3) and that $\tilde{\lambda}_\alpha = \lambda_\alpha = \sigma_\alpha^2$. We consider the eigenvalues (and therefore also the singular-values) numbered in decreasing order.

After all the above work we can now formally define the low-rank approximation of rank r of a function $f \in L^2(\Omega_1 \times \Omega_2)$ as

$$f_r(x,y) = \sum_{\alpha=1}^{r} \sigma_\alpha\, \phi_\alpha(x)\, \psi_\alpha(y). \tag{4.8}$$

4.2 Singular Value Decomposition for Matrices

Depending on ones background in numerics the most known definition for the singular value decomposition is actually the one of a matrix. That matrices can be considered to be function representations in a particular basis is a well known fact in linear algebra, and explains the connection between this section and the previous. All of the following can be found in most introductory books for numerics (also [37]).

Let $\mathbf{A} \in \mathbb{R}^{n\times m}$ be a real matrix. Then the *singular value decomposition*, SVD for short, of $\mathbf{A}$ is defined as $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. Here, $\mathbf{U} \in \mathbb{R}^{n\times n}, \mathbf{V} \in \mathbb{R}^{m\times m}$ are orthogonal matrices and $\mathbf{\Sigma} \in \mathbb{R}^{n\times m}$ is a rectangular diagonal matrix with non-negative entries. These entries are the singular values of the matrix, again in decreasing order. Following from this the columns of the matrix $\mathbf{U}$ contain the left-eigenvectors, and the columns of $\mathbf{V}$ the right-eigenvectors.

In the matrix case the SVD is only unique if $m = n$ and all the singular values are non-zero and distinct from each other and only up to multiplication with (-1). In all other cases the SVD is not unique, or rather only unique up to orthogonal transformations.

To verify the definition above one can check the relation defining singular values and -vectors, meaning

$$\mathbf{A}\mathbf{v}_i = \sigma_i\mathbf{u}_i, \qquad \mathbf{A}^T\mathbf{u}_i = \sigma_i\mathbf{v}_i \quad i = 1,\dots,\min(n,m), \tag{4.9}$$

where $\mathbf{u}_i, \mathbf{v}_i$ are the i-th left- respective right-singular vector and σ_i the accompanying singular value.

Also we can formulate

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \, \mathbf{u}_i \otimes \mathbf{v}_i \tag{4.10}$$

where r is the *rank* of the matrix, i.e. the number of non-zero eigenvalues.

And because we want to consider approximations, especially the low-rank ones, we define

$$\mathbf{A}_{\tilde{r}} = \sum_{i=1}^{\tilde{r}} \sigma_i \, \mathbf{u}_i \otimes \mathbf{v}_i \tag{4.11}$$

to be an approximation of $\mathbf{A}$ with rank $\tilde{r}$.

It is easy to see the similarity of (4.9) to (4.3), which can be explained by considering $\mathbf{A}$ to be a function in a specific basis representation. This gives us $\mathbf{A} \hat{=} f$, but also due to the definition of the matrix-vector-multiplication $\mathbf{A} \hat{=} \mathcal{S}$.

Now it is easy to conclude from the previous section that to actually calculate the SVD we should solve the eigenvalue problem for the matrix $\mathbf{A}^T\mathbf{A}$ in accordance with (4.4). In theory this produces the correct result, but squaring the matrix also squares the condition number and therefore calculating the eigenvalues like this in praxis is numerically unstable. Instead, the SVD is usually calculated with the help of Householder reflections and the QR-decomposition, although other algorithms exist as well.

Addition Especially because we will define an addition for tensors we do briefly need to consider how to "add" two SVDs. Here we consider an addition of two SVDs, whose original, not decomposed matrices are of the same size. The normal addition of two matrices no longer applies, even if the matrices in the two SVDs are of the same size they are not with regard to the same basis. As such two practical options remain: first, multiplying the SVDs back to the original matrix, adding them together and then performing the SVD again. Secondly, analogously to how we will be adding tensors in section 8. By combining the bases and setting for $R^{n\times m} \ni \mathbf{A}^{(1)} = U^{(1)}\Sigma^{(1)} \left(V^{(1)}\right)^T$ and $R^{n\times m} \ni \mathbf{A}^{(2)} = U^{(2)}\Sigma^{(2)} \left(V^{(2)}\right)^T$ a pseudo-addition in the SVD:

$$\mathbf{A}^{(1)} \dot{+}_{svd} \mathbf{A}^{(2)} = \begin{bmatrix} U^{(1)} & U^{(2)} \end{bmatrix} \begin{bmatrix} \Sigma^{(1)} & \mathbf{0} \\ \mathbf{0} & \Sigma^{(2)} \end{bmatrix} \begin{bmatrix} \left(V^{(1)}\right)^T \\ \left(V^{(2)}\right)^T \end{bmatrix}. \tag{4.12}$$

In our context it is very important to note that if the two original matrices were with regard to the same basis the above defined pseudo-sum of them is with regard to this basis, but duplicated. It is therefore no longer defined with regard to a basis at all and cannot be used together with matrices that are. This is due to the fact that the above is no longer a SVD, it is the representation of the sum

$$\mathbf{A}^{(1)} \dot{+}_{svd} \mathbf{A}^{(2)} = \sum_{i=1}^{r^{(1)}} \sigma_i^{(1)} \mathbf{u}_i^{(1)} \otimes \mathbf{v}_i^{(1)} + \sum_{i=1}^{r^{(2)}} \sigma_i^{(2)} \mathbf{u}_i^{(2)} \otimes \mathbf{v}_i^{(2)} \tag{4.13}$$

where the $\mathbf{u}_i^{(1)}$ and $\mathbf{u}_i^{(2)}$ are not necessarily orthogonal to each other (analogous for the $\mathbf{v}_i^{(k)}, k = 1, 2$). Since it is not an SVD the above sum also does not have the properties of an SVD, it is especially not of lowest rank. Performing a re-orthogonalization step could help in some contexts, but there is no general way to turn the above into an SVD. For more details on this see the section 8 on tensors and the sources cited within.

Concluding we can say that if we wish to perform such a thing as an addition in analogue to the tensor addition we always have to keep the basis-functions behind the matrix-representation in mind.

4.3 Eckart-Young-Mirsky and the Truncation Operator

As already mentioned above the Eckart-Young or sometimes Eckart-Young-Mirsky theorem states the SVD to be the best low-rank approximation of a given matrix or bivariate function in the L^2-sense [36, 37]. Following this we define the truncation operator, which uses the truncated SVD to return the low-rank best-approximation of the given function or matrix.

Theorem 13 (Eckart-Young-Mirsky)**.** *For a given function $f \in L^2(\Omega)$ the solution to the optimization problem*

$$\min_{\tilde{f} \in L^2(\Omega),\ rank(\tilde{f}) \leq r} \left\| f - \tilde{f} \right\|_{L^2(\Omega)} \tag{4.14}$$

is the function f_r defined by (4.8) *and the section above.*
For a given matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ the solution to the optimization problem

$$\min_{\tilde{\mathbf{A}} \in \mathbb{R}^{n \times m},\ rank(\tilde{\mathbf{A}}) \leq r} \left\| \mathbf{A} - \tilde{\mathbf{A}} \right\|_2 \tag{4.15}$$

is the matrix $\mathbf{A}_r$ as defined by (4.11) *above.*

Having established the above we can now define a *truncation operator* which, given a rank r, returns the low-rank best-approximation of a function or matrix as above.

First in the continuous case:

Definition 14. We define for a Hilbert space $\mathcal{H}$ and an integer r the *continuous truncation operator* by

$$\begin{aligned} T_r^{\infty} :& \mathcal{H} \longrightarrow \mathcal{H} \\ & f \mapsto f_r := \arg \min_{\tilde{f} \in \mathcal{H},\ \mathrm{rank}(\tilde{f}) \leq r} \left\| f - \tilde{f} \right\|_{L^2(\Omega)} . \end{aligned}$$

And then in the discretized version:

Definition 15. We define for an integer r the *discrete truncation operator* by

$$\begin{aligned} T_r^{(n,m)} :& \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}^{n\times m} \\ & \mathbf{A} \mapsto \mathbf{A}_r := \arg \min_{\tilde{\mathbf{A}}\in\mathbb{R}^{n\times m},\ \operatorname{rank}(\tilde{\mathbf{A}})\leq r} \left\| \mathbf{A} - \tilde{\mathbf{A}} \right\|_2 . \end{aligned}$$

Should it be clear from context which spaces we are referring to we will omit the superscripts.

A quick note on the discrete truncation operator: We will often use it in combination with the projection $P_{\mathbf{l}} : \mathcal{H} \longrightarrow V_{\mathbf{l}}$ in an expression $T_r^{\mathbf{l}} P_{\mathbf{l}} u$ for some $u \in \mathcal{H}$. For this we define the truncation operator analogous to the Definition 15 but without brackets in the superscript as

$$T_r^{\mathbf{l}} : \mathbb{R}^{(2^{l_1}-1)\times(2^{l_2}-1)} \longrightarrow \mathbb{R}^{(2^{l_1}-1)\times(2^{l_2}-1)} \tag{4.16}$$

in two dimensions. It generalizes for more dimensions.

Now the expression $T_r^{\mathbf{l}} P_{\mathbf{l}} u$ is well-defined, since $V_{\mathbf{l}}$ and $\mathbb{R}^{(2^{l_1}-1)\times(2^{l_2}-1)}$ are isomorph to each other when $V_{\mathbf{l}}$ is given in a fixed basis, which is always the case in this work. For the sake of readability we omit this isomorphism in the notation.

It is important to note that the truncation operator does not posses most properties we find useful in operators. Most importantly it is not linear, $T_r(f+g) \neq T_r f + T_r g$ with $f, g \in \mathcal{H}$. What does hold is a kind of commutativity with itself, it holds $T_{r_1} T_{r_2} f = T_{r_2} T_{r_1} f = T_{\min(r_1,r_2)} f$.

Especially with Part III of this work in mind it is very important to not forget that the Eckart-Young-Mirsky theorem holds for matrices, which are explicitly a two-dimensional object as $\mathbf{A} \in \mathbb{R}^{m\times n}$, and for bivariate functions, meaning $f \in L^2(\Omega_1 \times \Omega_2)$. For more than two dimensions the theorem no longer holds, the SVD is **not** the best-approximation for objects in three or more dimensions. These objects are tensors or functions $f \in L^2(\Omega_1 \times \cdots \times \Omega_m)$, which we will consider in more detail in Part III.

4.4 Error Estimates: Bramble-Hilbert Lemma

Having defined the truncation in the previous section the question now becomes: how to best bound the truncation error? We will see that for the singular value decomposition for matrices only an exact equation in terms of the singular values of the matrix in question exists. For the continuous case a more general estimate using the Bramble-Hilbert Lemma exists and will be shown in the following.

By definition of the L^2-norm,

$$\|f - T_r^{\infty} f\|_{L^2(\Omega_1\times\Omega_2)} = \sqrt{\sum_{\alpha=r+1}^{\infty} \sigma_\alpha^2}$$

exists as a first estimate (strictly speaking this is not an estimate, but rather an equality) of the approximation error for the truncated singular value decomposition.

Since we do not have any general estimates for the singular values themselves, we instead use a trick (as described in [18]) to estimate further: the Bramble-Hilbert lemma [3]. Said Lemma is an estimate for Finite-Element approximations, but can be applied to our context as well.

First we need some setup: We consider approximation in the first coordinate (arbitrarily, we could also choose the second one) and introduce a new finite element space $U_M \subset L^2(\Omega_1)$. The space U_M consists of a quasi-uniform triangulation over Ω_1 with mesh-width $h_M \sim M^{-1}$ and M many discontinuous, piecewise polynomial functions of total degree $\lceil p \rceil$ on said triangulation. Let $P_M : L^2(\Omega_1) \to U_M$ be the $L^2(\Omega_1)$-orthogonal projection onto U_M.

Then the Bramble-Hilbert lemma [3, Theorem 6.3, Theorem 6.4] for any given function $u \in H^p(\Omega_1)$ states

$$\|(\mathrm{Id} - P_M)\, u\|_{L^2(\Omega_1)} \leq c(p, \Omega_1) M^{-p} \, |u|_{H^p(\Omega_1)} \tag{4.17}$$

uniformly in M.

Important to note: we have $p \in \mathbb{R}^+$ depending on the space u is from, but since the degree of polynomials has to be a natural number we set this degree to be $\lceil p \rceil$. If $p \in \mathbb{N}$ we can omit the ceiling-brackets.

If we now set for a function $f \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$

$$f_M(x, y) := \left((P_M \otimes \mathrm{Id})\, f \right)(x, y)\,,$$

then we can formulate the following approximation error estimate in U_M, see also [18, Theorem 3.1]:

Lemma 2. *For $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$ being the eigenvalues of the operator $\mathcal{K}$ as defined in (4.4) and $\lambda_1^M \geq \lambda_2^M \geq \ldots \geq \lambda_M^M \geq 0$ being the eigenvalues of the operator $\mathcal{K}_M := P_M \mathcal{K} P_M$, it holds*

$$\|f - f_M\|^2_{L^2(\Omega_1 \times \Omega_2)} = \mathit{trace}\ \mathcal{K} - \mathit{trace}\ \mathcal{K}_M = \sum_{\alpha=1}^{M} \left(\lambda_\alpha - \lambda_\alpha^M \right) + \sum_{\alpha=M+1}^{\infty} \lambda_\alpha. \tag{4.18}$$

Using the approximation estimate (4.17) of the Bramble-Hilbert lemma leads to the estimate

$$0 \leq \mathrm{trace}\ \mathcal{K} - \mathrm{trace}\ \mathcal{K}_M \lesssim M^{-2p} \, |f|^2_{H^{(p,0)}_{mix}(\Omega_1 \times \Omega_2)}\,. \tag{4.19}$$

This can of course be formulated for approximation in the second coordinate direction analogously. And since $H^{(p,0)}_{mix}(\Omega_1 \times \Omega_2) \cap H^{(0,p)}_{mix}(\Omega_1 \times \Omega_2) =$

$H^{(p,p)}_{iso}(\Omega_1 \times \Omega_2) = H^p(\Omega_1 \times \Omega_2)$ by definition this can be rewritten for the space $H^p(\Omega_1 \times \Omega_2)$.

Now we can reformulate using the notation we have previously used in this work and arrive at the main error estimate for the truncation at rank r therefore being given by:

$$\|f - T^\infty_r f\|_{L^2(\Omega_1\times\Omega_2)} = \sqrt{\sum_{\alpha=r+1}^{\infty} \sigma_\alpha^2} \lesssim r^{-k} \, |f|_{H^k(\Omega_1\times\Omega_2)} \,. \tag{4.20}$$

Comparing this estimate to the ones for the projection P in the previous section 2 shows a problem we will encounter later: The appearance of the semi-norm on the right side makes it difficult to use in combination with our other estimates, which have a norm.

By the above technique [13] and also for more general cases by other methods [21] we can prove a rate of decay for the singular values of a function $f \in H^k(\Omega)$. A specialized, better rate for $f \in H^{(s_1,s_2)}_{mix}(\Omega)$ does not exist (that we know of). This will cause issues for the convergence results in this work.

Based on the previous results we can bound the rate of decay of the eigenvalues and with that of the singular values as well. We achieve a bound of

$$\sigma_\alpha^2 \lesssim \alpha^{-2k-1}, \quad \alpha \to \infty. \tag{4.21}$$

As discussed previously, the discrete matrix case is connected to the above continuous function one. Nonetheless for our consideration:

The error estimate for the discrete truncation of a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ at rank r is given by:

$$\left\|\mathbf{A} - T_r^{(n,m)}\mathbf{A}\right\|_F = \sqrt{\sum_{\alpha=r+1}^{\min(n,m)} \sigma_\alpha^2} = \frac{\sqrt{\sum_{\alpha=r+1}^{\min(n,m)} \sigma_\alpha^2}}{\sqrt{\sum_{\alpha=1}^{\min(n,m)} \sigma_\alpha^2}} \|\mathbf{A}\|_F \tag{4.22}$$

$$\left\|\mathbf{A} - T_r^{(n,m)}\mathbf{A}\right\|_2 = \sigma_{r+1} = \frac{\sigma_{r+1}}{\sigma_1} \|\mathbf{A}\|_2 \tag{4.23}$$

All of the above are equalities, we do not have good estimates for the rate of decay of the singular values of the matrices $\mathbf{A}$ in general. Should such a rate exist for specific matrices $\mathbf{A}$, inserting it in the equations above would give a more general error estimate.

5 Sparse Grids and the Singular Value Decomposition: Versus and Mixing

Having now defined both the sparse grid method and the singular value truncation we can ask ourselves: which is the better approximation method (in which setting)? And how do these two methods interact when used together; how can they be used together? In the following we give a summary of results on the topic, motivating our new Combi-format.

5.1 Versus

After considering the results from the previous section, one question becomes apparent: Why not use the singular value decomposition in all cases, as it realizes the low-rank L^2-best-approximation? Of course in praxis there may be constraints on which method one has to use for approximation, but if this is not the case, why choose the sparse grid approximation?

There has been some work on the topic of comparing the truncated singular value decomposition and the sparse grid method, see for example [18, 14] and also [15]. A short summary of the topic follows.

As we have already briefly hinted at in the above section: for a general function $f \in L^2(\Omega_1 \times \Omega_2)$ there exists no practical algorithm to exactly compute its singular values and (left- respective right-) eigenfunctions. Consequently, these need to be approximated in appropriately chosen spaces, which adds computational complexity and makes the error estimate more complicated.

A more detailed analysis of these facts can be found at [18, Section 3.4]. The result of this can be summarized as follows [18, Theorem 3.4]:

For a function $f \in H_{iso}^{(s_1,s_2)}(\Omega_1 \times \Omega_2)$ and the approximate singular value decomposition approach the degrees of freedom necessary to achieve a prescribed accuracy ε are

$$\mathrm{dof}_{svd}^{iso}(\varepsilon) \sim \varepsilon^{-\min\left(\frac{1}{s_1},\frac{1}{s_2}\right)} \varepsilon^{-\max\left(\frac{1}{\min(s_1,r_1)},\frac{1}{\min(s_2,r_2)}\right)}. \tag{5.1}$$

Here r_1, r_2 are the polynomial exactness of the spaces V_{l_1} and V_{l_2} in which we approximate.

In [15, Corollary 4.3] this rate is improved somewhat for $f \in H_{mix}^{(s_1,s_2)}(\Omega_1 \times \Omega_2)$ and using mixed sparse tensor product spaces for approximation. There it results in

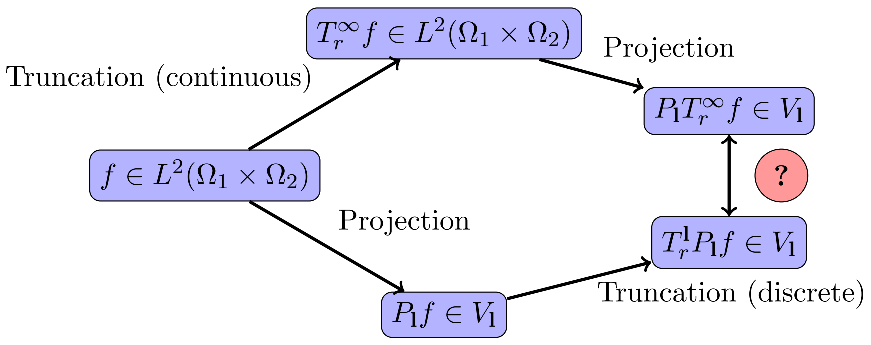
$$\mathrm{dof}_{svd}^{mix}(\varepsilon) \sim \varepsilon^{-\frac{1}{s_1+s_2}} \varepsilon^{-\max\left(\frac{1}{\min(s_1,r_1)},\frac{1}{\min(s_2,r_2)}\right)}. \tag{5.2}$$

In contrast to that we have already briefly considered the efficiency of the sparse grid method in the paragraph “Motivation” in section 1.2.1, which achieves

$$\mathrm{dof}_{sg}^{mix}(\varepsilon) \sim \varepsilon^{-\max\left(\frac{1}{\min(s_1,r_1)},\frac{1}{\min(s_2,r_2)}\right)} \tag{5.3}$$

for a prescribed accuracy ε. For a proof consult for example [18, Theorem 4.1] [4].

[4]The mixed sparse tensor product space $\hat{V}_J^{\sigma}$ used in this proof coincides with our sparse grid space $\hat{V}_J$ since we have set $\dim(\Omega_1) = \dim(\Omega_2) = 1$.



$T_r^\infty f \in L^2(\Omega_1 \times \Omega_2)$
Projection
Truncation (continuous)
$P_1 T_r^\infty f \in V_1$
$f \in L^2(\Omega_1 \times \Omega_2)$
?
Projection
$T_r^1 P_1 f \in V_1$
$P_1 f \in V_1$
Truncation (discrete)

Figure 5: A diagram showing the different possible orders of operations when mixing projections and truncations. The question mark highlights the relation we wish to define in this section.

From these calculation we can conclude that in praxis (when one actually has to compute the singular value decomposition), the sparse grid method requires less degrees of freedom to arrive at the same prescribed accuracy ε than the approximated singular value decomposition. As such the sparse grid method is more efficient.

5.2 Mixing

We have already stated that the Combi-format we wish to define in the following section will combine both sparse grid approximation and truncated singular value decomposition approximation. To be able to understand its properties and formulate an error analysis for it, we need a better understanding of the way approximation via orthogonal projections and via truncated singular value decomposition relate. In contrast to above, this time we wish to consider what happens when we combine the two techniques.

In Figure 5 we see the two different ways we can combine projections and truncations. We start with a function $f \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$; the first way we first apply the continuous truncation to rank r denoted by $T_r^\infty : \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2) \longrightarrow \mathcal{H}$ and then project this onto the space V_1 with an orthogonal projection P_1. The second way consist of first projecting with P_1 and then applying the discrete truncation of rank r denoted by $T_r^1 : V_1 \longrightarrow V_1$.

The question we wish to investigate in this subsection is the following: what relation exists between the two results of the above described paths? In the Figure 5 this relation is highlighted by the question mark in the red circle.

Before we continue we need to underline that although both paths use the same projection $P_1 : \mathcal{H} \longrightarrow V_1$ the two truncations are different from each other. Said difference has already been established earlier, it is the difference between the truncated singular value decomposition for continuous bivariate functions and the SVD for matrices. In a theoretical context the difference might not seem that significant (recall that we can interpret a matrix as a continuous function represented in a chosen basis and as such the matrix SVD is only a special case of the continuous one), but in a practical context it is of utmost significance. There exists no way to efficiently compute the singular value decomposition for an arbitrary function $f \in L^2(\Omega_1 \times \Omega_2)$. In the subsection above we have already briefly mentioned that to compute the eigenfunctions, one needs to approximate them in subspaces. An analytical solution is practically impossible. On the other hand, computing a matrix SVD is very much possible. It is still a relatively costly matrix operation (for a quadratic matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ the cost is $\mathcal{O}(n^3)$), but especially when the matrix has some special form the SVD can be optimized.

Concluding: Only the second path can in praxis be computed as written, the first would need further work and is (as discussed in the subsection above) less efficient.

In contrast to this is the fact that since the projection is linear and the truncation is not, the result of the first path is, for work in theory, easier to handle.

Commutativity The first question we ask ourselves (somewhat naively) is: does the diagram commute?

This would mean that for arbitrary $f \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$ **and** for an arbitrary P_1 (in this case we also mean that the basis of V_1 is arbitrary):

$$P_1 T_r^\infty f \overset{!}{=} T_r^1 P_1 f.$$

It is fairly easy to proof that the equation above does not hold (in general). Due to the fact that the projection of a rank-r function is again of at most rank r (recall for example (2.17)), we can formulate

$$P_1 T_r^\infty f = T_r^1 P_1 T_r^\infty f$$

since truncating a rank-r function at rank r is the identity.

Furthermore, in general

$$P_1 T_r^\infty f \neq P_1 f$$

i.e applying the projection after the rank-r truncation is not the same as just applying the projection. Since the SVD is unique (up to orthogonal transformations), the SVDs of two unequal functions are themselves not equal.

As such we come to the conclusion that in general the two paths do not reach the same result and therefore the diagram does not commute. We cannot put an equal sign in place of the question mark.

However, in the paragraphs above we have often needed to state that we were formulating *in general.* There are special cases in which the diagram does commute.

If indeed, for some (specific) $f \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$ and P_1 it holds

$$P_1 T_r^\infty f = P_1 f$$

then it also follows that

$$P_1 T_r^\infty f = T_r^1 P_1 f$$

and therefore the operations commute.

Now the question remains: for which specific $f \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$ and P_1 does the above hold? There are two possible cases.

1. $T_r^\infty f = f$, which is to say f is already a function of rank r or lower. This holds for arbitrary P_1. In this case it also holds $T_r^1 P_1 f = P_1 f$, since the projection of a rank-r function also has at most rank r, recall (2.17).

2. $T_r^\infty f = P_1 f$, which means that given a function f and a truncation rank r the space V_1 is defined as $V_1 = \operatorname{span}\{\phi_\alpha \otimes \psi_\alpha\}_{\alpha=1}^r \cup \tilde{V}$ where ϕ_α and ψ_α are the (right- and left-)eigenfunctions of f as defined in section 4.1. Furthermore $\tilde{V} \subset (\operatorname{span}\{\phi_\alpha \otimes \psi_\alpha\}_{\alpha=1}^\infty)^\perp$, meaning it is orthogonal to all eigenfunctions of f. The easiest case for this would be to choose the basis underlying V_1 as the eigenfunctions.

While both of these cases are interesting in theory, in praxis they are less important. In the first case the truncation is equal to the identity and we arrive at only an approximation via orthogonal projection. In that case we do not need to consider the truncation at all and most of what we do in this work becomes unnecessary. For the second case we need to be aware of the fact that this involves choosing a basis (and based on that the spaces and projections) dependent on the function f. This removes any sort of generality of the format we will be considering. It would also involve already knowing the eigenfunctions of the considered function (which is in general not feasible), at which point we can just use the truncated singular value decomposition (which our projection would become) without the extra steps we are considering in this work, which would as above become unnecessary.

If we consider our setup without the two special cases above the question remains: what is the relation between the results of the two paths? To formulate a more general answer, we will consider the difference between the two results.

Difference between orders of operations: We have already hinted above that due to its practical computability we wish to make use of the second path in praxis, but due to the linearity of P_1 would like to use the first one in theory. As such we first consider the error for switching the order of operations on a subspace defined by its level $\mathbf{l}$ and with truncation rank given by $r \in \mathbb{N}^+$:

$$\begin{aligned} T_r^{\mathbf{l}} P_{\mathbf{l}} f - P_{\mathbf{l}} T_r^{\infty} f &= T_r^{\mathbf{l}} P_{\mathbf{l}} f - P_{\mathbf{l}} f + P_{\mathbf{l}} f - P_{\mathbf{l}} T_r^{\infty} f \\ &= \left(T_r^{\mathbf{l}} - \mathrm{Id}\right) P_{\mathbf{l}} f + P_{\mathbf{l}} \left(f - T_r^{\infty} f\right) \qquad (5.4) \\ &= - \sum_{\alpha=r+1}^{\infty} \sigma_\alpha^{\mathbf{l}} \, \phi_\alpha^{\mathbf{l}} \psi_\alpha^{\mathbf{l}} + P_{\mathbf{l}} \left(\sum_{\alpha=r+1}^{\infty} \sigma_\alpha \, \phi_\alpha \psi_\alpha \right) \\ &= P_{\mathbf{l}} \left(- \sum_{\alpha=r+1}^{\infty} \sigma_\alpha^{\mathbf{l}} \, \phi_\alpha^{\mathbf{l}} \psi_\alpha^{\mathbf{l}} + \sum_{\alpha=r+1}^{\infty} \sigma_\alpha \, \phi_\alpha \psi_\alpha \right) \qquad (5.5) \\ &=: \mathcal{E}_{\mathbf{l},r}(f) \qquad (5.6) \end{aligned}$$

where $f = \sum_{\alpha=1}^{\infty} \sigma_\alpha \, \phi_\alpha \, \psi_\alpha$ and $P_{\mathbf{l}} f = \sum_{\alpha=1}^{\infty} \sigma_\alpha^{\mathbf{l}} \, \phi_\alpha^{\mathbf{l}} \psi_\alpha^{\mathbf{l}}$. Here we have first added a zero to achieve (5.4), and then used the fact that the projection is idempotent to arrive at (5.5).

(We know (for example [13]) that $\sigma_\alpha - \sigma_\alpha^{\mathbf{l}} \geq 0$ and as such the above is well defined.) An explanation of the result above in words: The error of switching the order of operations is the projection of the difference between the error of the discretized and the continuous truncation onto the subspace in question.

We can follow from (5.4) or (5.5) that in the two cases where the paths commute, which are discussed above, it indeed holds $\mathcal{E}_{\mathbf{l},r} = 0$. In the first case $f - T_r^{\infty} f = 0$ and $\left(T_r^{\mathbf{l}} - \mathrm{Id}\right) P_{\mathbf{l}} f = 0$ which then results in $\mathcal{E}_{\mathbf{l},r} = 0$ by (5.4). The second case arrives at the same result by considering that (in this case only) $P_{\mathbf{l}} f = \sum_{\alpha=1}^{r} \sigma_\alpha \, \phi_\alpha \psi_\alpha$ and therefore $\sum_{\alpha=r+1}^{\infty} \sigma_\alpha^{\mathbf{l}} \, \phi_\alpha^{\mathbf{l}} \psi_\alpha^{\mathbf{l}} = 0$. Using that also the projection projects onto a space that is by definition orthogonal to $\sum_{\alpha=r+1}^{\infty} \sigma_\alpha \, \phi_\alpha \psi_\alpha$ and therefore $P_{\mathbf{l}} \left(\sum_{\alpha=r+1}^{\infty} \sigma_\alpha \, \phi_\alpha \psi_\alpha\right) = 0$, we conclude $\mathcal{E}_{\mathbf{l},r} = 0$ by (5.5).

6 Combination Format

In this section we now define the new format which we have worked towards in this work. We will call this new format *combination format*, emphasising its dependence on the previously referenced combination technique. Often it will be shortened and simply be called *Combi-format*. In the following we will orient ourselves by the definitions from section 3. Specifically we will define this new format in two dimensions, using the SVD.

The main idea of the Combi-format is to compress the combination technique. In the section 5.1 before, we have discussed that for functions $f \in L^2(\Omega_1 \times \Omega_2)$ the sparse grid approximation has many advantages over approximating with the truncated singular value decomposition. As such using the sparse grid approach, and because of the practical problems mentioned in section 3 the combination technique, is a logical step.

On the other hand, the singular value decomposition has the unique property of being the low-rank L^2-best-approximation. Since rank is directly connected to cost, in both the storage and computational complexity sense, this is a desirable property. Furthermore, for a function $u \in V_{\mathbf{l}}$ (which can equivalently be expressed as a matrix with respect to a chosen basis) the SVD would be the classical one: decomposing a matrix. This means that no eigenfunctions would need to be approximated in a finite dimensional space, since the function already exists in such a space.

Therefore it makes sense to consider the Combi-format, where we, after calculating all the projections for the combination technique, compress them via the SVD. Although we loose the linearity of the combination technique, we hope to achieve a good cost-benefit ratio by reducing the rank of the projected functions significantly.

6.1 Definitions and Properties

As described in the introduction above, we define the Combi-format by compressing the combination technique via the SVD. Therefore we get

$$\mathcal{CF}_{m,\mathbf{r}}u = \sum_{|\mathbf{l}|_1=m+1} T^{\mathbf{l}}_{r_{\mathbf{l}}} P_{\mathbf{l}} u - \sum_{|\mathbf{l}|_1=m} T^{\mathbf{l}}_{r_{\mathbf{l}}} P_{\mathbf{l}} u. \tag{6.1}$$

where $\mathbf{r} = (r_{\mathbf{l}})_{\mathbf{l}\in\mathcal{I}^C_m}$ is the vector of truncation ranks indexed by the grid the truncation is applied to.

There are multiple ways to choose this vector $\mathbf{r}$, we will mostly distinguish between a uniform truncation (for each grid the same rank) and specialized truncations. In the first case we will replace the vector $\mathbf{r}$ by the integer r. Choosing the truncation ranks will be discussed later on in section 6.4.

Properties When further considering the above format we can see that its structure resembles the combination technique as defined in (3.1), only the appearing grids have been compressed by the SVD. This unfortunately means that the Combi-format inherits the non-linear property of the SVD. While this guarantees better approximation rates it makes the format harder to handle as well: the Combi-format is not linear, $\mathcal{CF}_{m,\mathbf{r}}(u+v) \neq \mathcal{CF}_{m,\mathbf{r}}u + \mathcal{CF}_{m,\mathbf{r}}v$ in general.

Furthermore, it is not even clear what is meant by the addition of two (or more) Combi-formats: one can not simply add the matrices (if they are even of the same size), since they are as a consequence of the SVD with respect to generally different bases. There are however two ideas to circumvent this, as long as the Combi-formats involved are of the same level and with respect to the same basis.

This has been previously discussed in the paragraph "Addition" in section 4.2.

One idea is to re-multiply the SVD on each grid, add up the (now all in the same basis representation) matrices and then apply the SVD again. The other one is to "add" the SVDs together as described in (4.12). This is analogous to what we will consider for more than two dimensions. In that case we will need to always keep in mind that the sum of two Combi-formats is based on a different basis-function system (strictly speaking not a basis since basis-functions appear twice) than the normal Combi-format.

rank-r functions In section 2.1, recall (2.17), we discussed that the orthogonal projection of a rank-r function also has at most rank r. For the Combi-format this is now very important, since it means that we can truncate at this rank r without actually loosing any information.

For a function $f \in \mathcal{H} \subset L^2(\Omega_1 \times \Omega_2)$ of at most rank r it holds

$$\mathcal{CT}_m f = \mathcal{CF}_{m,r} f \tag{6.2}$$

where $\mathcal{CT}_m$ is the combination technique of level m and $\mathcal{CF}_{m,r}$ is the Combi-format as defined above with universal truncation rank r. This equality follows directly from the properties of the combination technique.

Cost: Computation and Storage To evaluate the usefulness of the Combi-format we need to answer the question of the cost-benefit ratio. For this we need to know the cost of computing the Combi-format, with respect to computational complexity and storage space.

Since (6.1) shows us that to compute the Combi-format we first need to compute the combination technique and then perform an SVD on each grid, we can formulate the computational complexity of the Combi-format based on the one for the combination technique.

Computing the SVD does in general require $\mathcal{O}(n^3)$ operations for a matrix in $\mathbb{R}^{n\times m}$ with $n \leq m$. As such we need to perform for the whole Combi-format additional operations of the size

$$\sum_{\mathbf{l}\in\mathcal{I}_m^C} \mathcal{O}(2^{3\min(l_1,l_2)}).$$

Recalling the complexity estimates in section 3.2, using $\tilde{K}(l)$ as the cost of inverting the mass-matrix in one direction of level l we can formulate a detailed cost formula:

$$\sum_{\mathbf{l}\in\mathcal{I}_m^C} \mathcal{O}(2^{3\min(l_1,l_2)}) + \tilde{K}(l_1) + \tilde{K}(l_2) \sim \sum_{\mathbf{l}\in\mathcal{I}_m^C} \mathcal{O}(2^{3\min(l_1,l_2)3}) + 2\cdot K(\max(l_1,l_2)). \tag{6.3}$$

Unfortunately this does mean that, as long as the mass-matrix has a structure that means it can be inverted in quadratic or even linear complexity, the Combi-format has even asymptotically higher computational complexity. If this is not the case the asymptotic cost is the same for both the Combi-format and the combination technique, as the worst case cost for inverting the matrix is $\mathcal{O}(n^3)$ as well.

In exchange for the additional computations the storage cost is reduced. We recall that for the combination technique we have a storage cost of (this is (3.8)):

$$2\sum_{l=1}^{\lfloor\frac{m+1}{2}\rfloor}(2^l-1)(2^{m+1-l}-1)+2\sum_{l=1}^{\lfloor\frac{m}{2}\rfloor}(2^l-1)(2^{m+1-l}-1)\sim\mathcal{O}(2^{m+1}m).$$

For the Combi-format we need to store for each grid the two (we can always multiply the diagonal matrix Σ onto one of the other two, w.l.o.g. U) matrices of the SVD. As such we get a storage cost of

$$\sum_{\mathbf{l}\in\mathcal{I}_m^C}\left(2^{l_1}-1\right)\cdot r_{\mathbf{l}}+\left(2^{l_2}-1\right)\cdot r_{\mathbf{l}}. \tag{6.4}$$

If we have a particular rule for the $r_{\mathbf{l}}$ we can specify this further. For example if we have a uniform truncation at rank r we achieve:

$$\sum_{\mathbf{l}\in\mathcal{I}_m^C} r\cdot\left(\left(2^{l_1}-1\right)+\left(2^{l_2}-1\right)\right)\sim\mathcal{O}(r\cdot 2^{\lfloor\frac{m+1}{2}\rfloor})$$

if $r<2^{\lfloor\frac{m+1}{2}\rfloor}$ (meaning if a truncation actually took place). Depending on the chosen r this can be a significant reduction. And even in the worst case (no truncation at all) it is asymptotically no worse than the combination technique.

Concluding this we can say that the Combi-format exchanges a (potentially small) increase in computational complexity for a (potentially significant) reduction in storage cost. The storage cost reduction is dependent on the chosen truncation.

6.2 The Truncation

The main difference between the combination technique and the Combi-format is the truncation of each projected function, where we use a truncated SVD as described in 4.3.

Now the question remains: how to choose the truncation parameter(s) $\mathbf{r}=(r_{\mathbf{l}})_{\mathbf{l}\in\mathcal{I}_m^C}$?

Generally speaking we can choose them arbitrarily, there are no conditions on the truncation for which the Combi-format would somehow not be well-defined. As such the real question becomes: what choice of parameters is useful or efficient?

In this work we consider two different options for truncation, chosen because they are practical; practical as in can be efficiently used in praxis. For practicalities sake we set $\mathbf{r} > 0$, meaning no truncation to empty takes place.

Rank-r-truncation The first option is to decide a-priori that the resulting matrix decomposition (and with that the discretized function on that subspace) should have maximum rank r. This is particularly useful in the case that there is a fixed limit to storage space, since fixing the maximum rank also fixes the maximal storage space. Another use is for when the continuous function already has a fixed rank, since in that case the rank of the exactly approximated function cannot be any higher (recall (2.17) and the paragraph above) and any in praxis appearing higher ranks are due to approximation and calculation inaccuracy (meaning the computer precision). These can be discarded without influencing convergence and therefore speed up computation (and reduce the needed storage space).

Singular-value-truncation The second option is to dynamically (during calculation) truncate according to the calculated singular values. Here multiple options present themselves:

- We can truncate by the sum of the squares of the singular values. More precisely we bound $\sum_{\alpha=r+1}^{\min(n,m)} \sigma_\alpha^2$. This term naturally appears in the error analysis, since it is equivalent to the squared Frobenius-norm of the truncation error, as can be seen in (4.22).
- Analogously, we can consider the relative error and bound $\frac{\sum_{\alpha=r+1}^{\min(n,m)} \sigma_\alpha^2}{\sum_{\alpha=1}^{\min(n,m)} \sigma_\alpha^2}$, which again appears in (4.22).
- We can just consider the value of the singular value, bounding σ_{r+1}; this coincides with the error in the 2-norm, as shown in (4.23).
- For the relative error we bound the term $\frac{\sigma_{r+1}}{\sigma_1}$, which again appears in (4.23).

This does not have the advantage that we know the rank of the matrix a-priori, but instead the idea is that it more accurately keeps the "important" singular values dependent on the individual function.

In the case of singular-value-truncation we write $\mathbf{r}(\varepsilon)$ for a chosen bound $\varepsilon > 0$, since the ranks are depending on the bound and not known a-priori.

The above are the options for each sub-grid. For the whole format there are again two main options:

- Uniform truncation, meaning we truncate the same way on each sub-grid. This is the option that appears first in the convergence analysis because it is the easiest one to control for. In this case we can replace the vector of truncation ranks $\mathbf{r}$ by a single $r \in \mathbb{N}$ (and $\mathbf{r}(\varepsilon)$ by $r(\varepsilon) \in \mathbb{N}$).

- Individual truncation, meaning each sub-grid is truncated individually and independently of each other.

- We do need to note that one form of truncation we use in the following examples combines these two concepts: When we use singular-value-truncation, each grid gets truncated individually, although all grids are truncated by one universal rule. As such it is possible to combine the above two options due to the fact that the truncation rank in the singular-value-truncation is not known a-priori.

6.3 Convergence Analysis

Considering the convergence of the Combi-format is not straightforward. Due to the non-linearity of the SVD the convergence theory for the Combi-technique is not directly transferable. As such we need to take a few detours to arrive at a result for the Combi-format

First of all we need to define the setting: In the following we show a convergence analysis for the Combi-format defined in (6.1).

We wish to formulate a result along the lines of Theorem 12, which shows the convergence of the combination technique. For this we consider $f \in H_{mix}^{(s_1,s_2)}(\Omega_1 \times \Omega_2)$ for $0 < s_1 \leq r_1$ and $0 < s_2 \leq r_2$. The proof of this theorem is based on the idea of detail projections $\varDelta_{\mathbf{l}}^{P} f$, which are orthogonal to each other. Due to the nature of the SVD, the terms $(T_r^{\mathbf{l}} - T_{r-1}^{\mathbf{l}})f$ are also orthogonal to each other. Unfortunately what we also need for the proof is an analogy to (3.6), the reformulation of the difference to the function only in terms of the detail projections.

This is not transferable to the combination technique. The telescoping-sum principle does not work if we also perform a truncation after the projection. This is due to what we have already formulated in section 5: the non-commutativity of the projection and the truncation in general. The truncation (whether for the matrix of the continuous case) is not linear in its argument, but the projection is.

Therefore we wish to use the switching of the order of operations we formulated in section 5.2. In the following all norms without index are the $L^2(\Omega)-$norms. Also we consider $P \equiv Q$ for simplicity, generalization is of course possible, analogous to the proof of Theorem 12.

Naive Idea: Triangle Inequality A first idea for estimating the error is by triangle inequality; the combination technique and the Combi-format both have an outer sum, which can be pulled out of the norm by said inequality. We restrict to uniform truncation at a rank r and of course $d = 2$.

Now we wish to use the switching of the orders of operation, recall section 5.2. Following this idea we can formulate for the Combi-format with uniform truncation:

$$\begin{aligned}
\mathcal{CF}_{m,r}(f) &= \sum_{|\mathbf{l}|_1=m+1} T_r^{\mathbf{l}} P_{\mathbf{l}} f - \sum_{|\mathbf{l}|_1=m} T_r^{\mathbf{l}} P_{\mathbf{l}} f \\
&= \sum_{|\mathbf{l}|_1=m+1} (P_{\mathbf{l}} T_r^{\infty} f + \mathcal{E}_{\mathbf{l},r}) - \sum_{|\mathbf{l}|_1=m} (P_{\mathbf{l}} T_r^{\infty} f + \mathcal{E}_{\mathbf{l},r}) \\
&= \mathcal{CT}_m(T_r^{\infty} f) + \sum_{|\mathbf{l}|_1=m+1} \mathcal{E}_{\mathbf{l},r} - \sum_{|\mathbf{l}|_1=m} \mathcal{E}_{\mathbf{l},r}.
\end{aligned}$$

In summary we know that

$$\mathcal{CF}_{m,r}(f) = \mathcal{CT}_m(T_r^{\infty} f) + \sum_{m \le |\mathbf{l}|_1 \le m+1} (-1)^{m+1-|\mathbf{l}|_1} \mathcal{E}_{\mathbf{l},r}(f) \tag{6.5}$$

holds by linearity of the combination technique. Here

$$T_r^{\mathbf{l}} P_{\mathbf{l}} f - P_{\mathbf{l}} T_r^{\infty} f = \mathcal{E}_{\mathbf{l},r}(f)$$

by definition at (5.6).

Given this we can just apply the triangle inequality:

$$\|f - \mathcal{CF}_{m,r}(f)\| \le \|f - \mathcal{CT}_m(T_r^{\infty} f)\| + \left\| \sum_{m \le |\mathbf{l}|_1 \le m+1} (-1)^{m+1-|\mathbf{l}|_1} \mathcal{E}_{\mathbf{l},r}(f) \right\|$$

and then again, consecutively for both terms on the right hand side.

For the first term we formulate:

$$\|f - \mathcal{CT}_m(T_r^{\infty} f)\| \le \|f - T_r^{\infty} f\| + \|T_r^{\infty} f - \mathcal{CT}_m(T_r^{\infty} f)\|.$$

Both of the right hand side terms above we know how to bound already, first

$$\|f - T_r^{\infty} f\|_0 = \sqrt{\sum_{\alpha=r+1}^{\infty} \sigma_\alpha^2} \lesssim r^{-k} |f|_k$$

by the error estimate (4.20) for the truncation operator.

Secondly, by the Theorem 12 about the convergence of the combination technique, we have

$$\|T_r^{\infty} f - \mathcal{CT}_m(T_r^{\infty} f)\| \lesssim \begin{cases} 2^{-J \min\{s_1,s_2\}} \|T_r^{\infty} f\|_{H_{\text{mix}}^{(s_1,s_2)}} & s_1 \neq s_2 \\ 2^{-Js}\sqrt{J} \|T_r^{\infty} f\|_{H_{\text{mix}}^{(s_1,s_2)}} & \text{else} \end{cases}$$

where $J = m + d - 1$.

Finally there remains the term for the switching of operations:

$$\left\| \sum_{m \le |\mathbf{l}|_1 \le m+1} (-1)^{m+1-|\mathbf{l}|_1} \mathcal{E}_{\mathbf{l},r}(f) \right\|,$$

for which we have found no bound yet. We can also apply the triangle inequality to it, but since we also have no bound for the $\mathcal{E}_{\mathbf{l},r}(f)$ separately, this grants no obvious improvement.

To get a cohesive error estimate we would now need to balance the different estimates above. Trying to do so is challenging, especially because of the different (semi-)norms, both in the sense of it being a different norm and a different function in the norm. As such, this estimate is mostly useful for getting a first idea of what kind of convergence rates we can expect (and the difficulty of mixing sparse grid and SVD error estimates.)

Combination Technique Convergence In the following we orient ourselves by the convergence proof given in [17] and start again with (6.5). To use the convergence of the combination technique we try to not pull anything out of the norm before absolutely necessary.

We start by formulating the notation

$$\mathcal{CE}_{m,r}(f) := \sum_{m \le |\mathbf{l}|_1 \le m+1} (-1)^{m+1-|\mathbf{l}|_1} \mathcal{E}_{\mathbf{l},r}(f)$$

to shorten the expressions. Here, $\mathcal{E}_{\mathbf{l},r}(f) = T_r^{\mathbf{l}} P_{\mathbf{l}} f - P_{\mathbf{l}} T_r^{\infty} f$ as defined in (5.6).

Then we consider

$$\begin{aligned}
\|f - \mathcal{CF}_{m,r}(f)\| &= \|f - \mathcal{CT}_m(T_r^{\infty} f) - \mathcal{CE}_{m,r}(f)\| && (6.6)\\
&= \left\| \sum_{|\mathbf{l}|_1=1}^{\infty} \Delta_{\mathbf{l}}^P T_{\infty}^{\infty} f - \sum_{|\mathbf{l}|_1=1}^{m+1} \Delta_{\mathbf{l}}^P T_r^{\infty} f - \mathcal{CE}_{m,r}(f) \right\| && (6.7)\\
&= \left\| \sum_{|\mathbf{l}|_1>m+1}^{\infty} \Delta_{\mathbf{l}}^P T_{\infty}^{\infty} f + \sum_{|\mathbf{l}|_1=1}^{m+1} \Delta_{\mathbf{l}}^P (\mathrm{Id} - T_r^{\infty}) f - \mathcal{CE}_{m,r}(f) \right\| && (6.8)\\
&= \|f - \mathcal{CT}_m(f) + \mathcal{CT}_m((\mathrm{Id} - T_r^{\infty})f) - \mathcal{CE}_{m,r}(f)\| && (6.9)\\
&\le \|f - \mathcal{CT}_m(f)\| + \|\mathcal{CT}_m((\mathrm{Id} - T_r)f) - \mathcal{CE}_{m,r}(f)\|. && (6.10)
\end{aligned}$$

Here we have used a number of known facts. We start with the same idea as (6.5), which gives us (6.6); for (6.7) we use the representations (2.14) of f and (3.5) of $\mathcal{CT}_m(T_r^\infty f)$. We also need the fact that $T_\infty^\infty = \mathrm{Id}$, giving us the ability to just insert this operator wherever we need it without changing anything. This is how the term $(\mathrm{Id} - T_r^\infty)f$ came to be, since both the projection and the detail projection are linear we can simply pull this term into the operator. For (6.9) we again use different representations: (3.6) for $\sum_{|\mathbf{l}|_1>m+1}^{\infty} \Delta_{\mathbf{l}}^P T_\infty^\infty f$, where we also use $T_\infty^\infty = \mathrm{Id}$ again, and (3.5) for $\sum_{|\mathbf{l}|_1=1}^{m+1} \Delta_{\mathbf{l}}^P (\mathrm{Id} - T_r^\infty)f$.

The problem lies in the last line (6.10) where we had to use the triangle inequality. Recalling the method of the proof for the combination technique in Theorem 12, we only wish to use the triangle inequality to pull terms apart once we have pulled the sum of the combination technique out of the norm using (Galerkin-) orthogonality. Unfortunately, while the terms $\Delta_{\mathbf{l}}^P T_\infty^\infty f$ are all orthogonal to each other, they are not orthogonal to either the $\Delta_{\mathbf{l}}^P(\mathrm{Id} - T_r^\infty)f$ (recall that the detail projection does not need to be an actual projection) or the terms of $\mathcal{CE}_{m,r}(f)$. While the orthogonality to the $\Delta_{\mathbf{l}}^P(\mathrm{Id} - T_r^\infty)f$ is fixable by demanding that the basis be chosen in such a way that the $W_{\mathbf{l}}$ are orthogonal to each other, the terms of $\mathcal{CE}_{m,r}(f)$ are by their definition not orthogonal to the other terms.

We have an estimate for the term $\|f - \mathcal{CT}_m(f)\|$ from [17] respective Theorem 12, it is the additional term that needs further consideration.

Bounding the additional Error We now wish to further consider the term

$$\|\mathcal{CT}_m((\mathrm{Id} - T_r^\infty)f) - \mathcal{CE}_{m,r}(f)\| .$$

Writing this term down in a more detailed form leads to

$$\begin{aligned}
&\mathcal{CT}_m((\mathrm{Id} - T_r^\infty)f) - \mathcal{CE}_{m,r}(f)\\
&= \sum_{m\le|\mathbf{l}|_1\le m+1} (-1)^{m+1-|\mathbf{l}|_1} \Big(P_{\mathbf{l}}((\mathrm{Id} - T_r^\infty)f) - P_{\mathbf{l}}(f - T_r^\infty f) - (T_r^{\mathbf{l}} - \mathrm{Id})(P_{\mathbf{l}}f)\Big)\\
&= \sum_{m\le|\mathbf{l}|_1\le m+1} (-1)^{m+1-|\mathbf{l}|_1} (\mathrm{Id} - T_r^{\mathbf{l}})(P_{\mathbf{l}}f)\\
&=: \mathcal{FE}_m \qquad (6.11)
\end{aligned}$$

Comparing this to the above calculations explains why this makes sense, the additional term is essentially only the difference between the combination technique and the Combi-format under our restrictions. Bounding this term is more problematic, to our knowledge a general form has not been found yet.

A very rough estimate is possible via triangle inequality, considering

$\sigma_\alpha - \sigma_\alpha^{\mathbf{l}} \geq 0$, as explained above. This leads to

$$\begin{aligned}
\|\mathcal{F}\mathcal{E}_m\| &= \left\| \sum_{m \leq |\mathbf{l}|_1 \leq m+1} (-1)^{m+1-|\mathbf{l}|_1} (\mathrm{Id} - T_r^{\mathbf{l}})(P_{\mathbf{l}} f) \right\| \\
&\leq \sum_{m \leq |\mathbf{l}|_1 \leq m+1} \left\| (\mathrm{Id} - T_r^{\mathbf{l}})(P_{\mathbf{l}} f) \right\| \\
&\leq \sum_{m \leq |\mathbf{l}|_1 \leq m+1} \|(\mathrm{Id} - T_r^{\infty})(f)\| \\
&\lesssim \sum_{m \leq |\mathbf{l}|_1 \leq m+1} r^{-k} |f|_{H^k(\Omega)} \lesssim (2m+1) \cdot r^{-k} |f|_{H^k(\Omega)}, \qquad (6.12)
\end{aligned}$$

where we also used the error estimate for the truncation operator (4.20) again.

Conclusion All in all we arrive at

$$\|f - \mathcal{CF}_{m,r}(f)\| \leq \|f - \mathcal{CT}_m(f)\| + \|\mathcal{F}\mathcal{E}_m\|. \qquad (6.13)$$

Especially the last error term is highly dependent on the relation between f and the chosen basis. I have not found any general estimates for this term.

It is important to note that the second term is 0 if we choose to project onto the eigenbasis of the function f. Then we have a simple estimate for the Combi-format that coincides with the one for the Combi-Technique. This case should not appear in praxis though, since it presumes that one already knows the eigenbasis (and therefore, recall the theorem of Eckart-Young-Mirsky, the low-rank best-approximation.)

The best general estimate we have at the moment is therefore the following:

$$\begin{aligned}
&\|f - \mathcal{CF}_{m,r}(f)\|_0 \\
&\lesssim \begin{cases} 2^{-(m+1)\min\{s_1,s_2\}} \|f\|_{H_{\text{mix}}^{(s_1,s_2)}} + (2m+1) \cdot r^{-k} |f|_k & s_1 \neq s_2 \\ 2^{-(m+1)s} \sqrt{m+1} \, \|f\|_{H_{\text{mix}}^{(s,s)}} + (2m+1) \cdot r^{-k} |f|_k & s_1 = s_2 = s \end{cases} \qquad (6.14)
\end{aligned}$$

It still requires a balancing of two very different terms, but is a definite step up from the naive approach. We do note that the estimate is not sharp: If we were to truncate at the highest rank a grid in the Combi-format could have (that would be $2^{\lfloor \frac{m+1}{2} \rfloor} - 1$), we essentially would not truncate at all and therefore have the combination technique of the same level. But inserting $r = 2^{\lfloor \frac{m+1}{2} \rfloor}$ (disregarding the -1 for ease of notation) in the above estimate does not achieve the same order of convergence as Theorem 12; the theorem which gives the rate of convergence of the combination technique.

Disregarding the constants the rate of convergence is still significantly slower since we cannot expect a function $f \in H^{(s,s)}_{\text{mix}}$ to be in H^{2s}. (In the worst case f is only in H^s, making the rate of convergence of the combination technique about squared the rate we have above.)

As such we know that the problem is the second term: the estimate for the truncation of the projected function is far from optimal. Unfortunately we do not have a better one, while we do have an estimate for the decay of the eigenvalues of the function f as described in section 4.4, the only estimate we have for the eigenvalues of the projected functions is to bound them by the eigenvalues of f.

All of the above is for truncation after a universal rank r, for individual ranks for each summand we have to go a different way.

Again: Triangle Inequality The conclusion we have reached above has given us a new approach: instead of using the switching of the order of the operators, we will instead consider the term $\mathcal{FE}_m$, which is the difference between Combi-format and combination technique.

Following from that we can disregard the technique of the combination technique convergence proof and instead consider the triangle inequality again directly. Considering the not existent necessary orthogonality in the above approach this should reach about the same result, while making greater generality possible.

And indeed we will find that this reaches the same result as the considerations above, since we did use the triangle inequality to pull exactly the term $\mathcal{FE}_m$ out of the norm. The only way to improve the convergence theory is therefore to find a way to circumvent this step, finding some way to reshape this term (or of course finding a completely different method of proofing convergence). As far as is known such a reshaping has not yet been found and it is questionable whether one actually exists. This reshaped term would depend strongly on the behaviour of the singular values of the approximated function (the $\sigma^{\mathbf{1}}_{\alpha}$), and not much is known of their behaviour (yet).

The following strategy for proving a rate of convergence is similar to the convergence proof in [25, 31], only that instead of the combination technique the sources consider the ANOVA-decomposition and instead of truncation via SVD they consider quadrature (the goal is to approximate an integral, not the function itself). The general strategy still transfers and reinforces the thought that there may not be a better strategy to prove convergence for the Combi-format. In contrast to the method contemplated in [25, 31] we still have the problem of not being able to bound the projected singular values well, as discussed previously. Also in contrast to the ANOVA-decomposition the combination technique does not give us an automatic understanding of which terms are more or less important.

As a first step we consider now this estimate via triangle inequality:

$$\|f - \mathcal{CF}_{m,\mathbf{r}}(f)\| \leq \|f - \mathcal{CT}_m(f)\| + \|\mathcal{CT}_m(f) - \mathcal{CF}_{m,\mathbf{r}}(f)\| . \tag{6.15}$$

Since we no longer need to show orthogonality via the detail projections we can also consider the Combi-format in its generality, with potentially non-uniform truncation ranks.

This leads us to defining

$$\begin{aligned}\mathcal{CT}_m(f) - \mathcal{CF}_{m,\mathbf{r}}(f) &= \sum_{m \leq |\mathbf{l}|_1 \leq m+1} (-1)^{m+1-|\mathbf{l}|_1} (\mathrm{Id} - T^{\mathbf{l}}_{r_\mathbf{l}})(P_\mathbf{l} f) \\ &=: \mathcal{FE}_{m,\mathbf{r}}, \end{aligned} \tag{6.16}$$

compare also to (6.11).

We need a few more definitions; the following is a useful substitution, which will motivate the way we use the singular-value-truncation as we have already defined it. Also worth recalling is (4.22), which shows that

$$\epsilon_\mathbf{l} = \sqrt{\sum_{\alpha = r_\mathbf{l}+1}^{\min(l_1,l_2)} (\sigma^\mathbf{l}_\alpha)^2} \tag{6.17}$$

is also the norm of the difference between a discretized function and its truncated (at $r_\mathbf{l}$) approximation. Considering what we have already formulated, this is exactly the term we need.

We recall the notation

$$P_\mathbf{l} f = \sum_{\alpha=1}^{\infty} \sigma^\mathbf{l}_\alpha \, \phi^\mathbf{l}_\alpha \psi^\mathbf{l}_\alpha, \tag{6.18}$$

especially remembering that the singular values of the projected functions (the $\sigma^\mathbf{l}_\alpha$) are not the same as the singular values of the original function (the σ_α). About the latter we would have more information (again, recall section 4.4) given the smoothness of the function.

Then we reformulate in a very straightforward fashion with the triangle inequality:

$$\begin{aligned}\|\mathcal{FE}_{m,\mathbf{r}}\| &\leq \sum_{m \leq |\mathbf{l}|_1 \leq m+1} \left\| (\mathrm{Id} - T^{\mathbf{l}}_{r_\mathbf{l}})(P_\mathbf{l} f) \right\| && (6.19) \\ &= \sum_{m \leq |\mathbf{l}|_1 \leq m+1} \sqrt{\sum_{\alpha = r_\mathbf{l}+1}^{\min(l_1,l_2)} (\sigma^\mathbf{l}_\alpha)^2} && (6.20) \\ &= \sum_{m \leq |\mathbf{l}|_1 \leq m+1} \epsilon_\mathbf{l}, && (6.21)\end{aligned}$$

knowing that this is only a rough estimate, but the best we can do in this setting.

For an even rougher estimate we can formulate

$$\|\mathcal{FE}_{m,\mathbf{r}}\| \leq (2m+1) \max_{\mathbf{l}\in\mathcal{I}_m^C} \epsilon_{\mathbf{l}}, \tag{6.22}$$

this will help later when choosing the truncation ranks.

Finally we can formulate the convergence result:

Theorem 16 (Convergence of the Combi-format). *For a level $m \in \mathbb{N}$ and a function $f \in H_{mix}^{(s_1,s_2)}(\Omega_1 \times \Omega_2)$ with $0 < s_1 \leq r_1$ and $0 < s_2 \leq r_2$, the Combi-format satisfies the error bounds*

$$\begin{aligned} \|f - \mathcal{CF}_{m,\mathbf{r}}(f)\| &\leq \|f - \mathcal{CT}_m(f)\| + \|\mathcal{FE}_{m,\mathbf{r}}\| \\ &\lesssim \begin{cases} 2^{-(m+1)\min\{s_1,s_2\}} \|f\|_{H_{mix}^{(s_1,s_2)}} + \bar{\varepsilon} & s_1 \neq s_2 \\ 2^{-(m+1)s}\sqrt{m+1}\,\|f\|_{H_{mix}^{(s,s)}} + \bar{\varepsilon} & s_1 = s_2 = s, \end{cases} \end{aligned} \tag{6.23}$$

where

$$\bar{\varepsilon} = \sum_{m\leq|\mathbf{l}|_1\leq m+1} \epsilon_{\mathbf{l}}. \tag{6.24}$$

If we wish to reach more generality (remove the dependence on the singular values of the projected functions) we can consider [21] and $\sigma_\alpha \geq \sigma_\alpha^{\mathbf{l}}$ and $f \in H^k(\Omega)$:

$$\epsilon_{\mathbf{l}} \leq \sqrt{\sum_{\alpha=r_{\mathbf{l}}+1}^{\min(l_1,l_2)} \sigma_\alpha^2} \lesssim r_{\mathbf{l}}^{-k} \tag{6.25}$$

and substitute this in the above result. This however is a rough estimate that does not necessarily make it easier to choose the truncation ranks, since it automatically leads to a uniform truncation.

The above concludes our considerations for a convergence estimate for the Combi-format. It is not a sharp estimate, but it is useable, as will be shown in the following section.

6.4 Truncation-ranks: A Heuristic

Given the convergence technique we have formulated above we now wish to consider the most important remaining question: given the information about convergence that we have, how do we wish to choose the truncation ranks? The following is not formulated as an algorithm, in this work all the choosing of rank was still done per hand. It is however an approach to a possible algorithmic solution of the problem.

The following is already very praxis-oriented, the described procedure takes place in the middle of the algorithm to compute the Combi-format.

First we need to consider the information we have: with only a-priori information we are reduced to the result of (6.25), where we can then uniformly choose the rank of truncation depending on the smoothness of the function. As such we allow ourselves to consider more information.

The projections necessary for the combination technique have to be computed regardless and are completely independent of the chosen truncation. Also regardless of what we choose the SVD has to be calculated. (If one has access to a SVD method which truncates while computing and would stop before calculating the whole decomposition this might cause additional computations. In our computing environment this was however not the case. This motivates the singular-value-truncation, for which all singular values need to be computed any way).

At this point we also have access to the information that is the value of the singular values for each sub-grid. Additionally we know a-priori how fast the combination technique (the first summand of the convergence result above) is expected to converge.

To at least roughly predict the possible convergence (which is the convergence of the combination technique) we also need a starting point. For this it is necessary to compute the error of the first level of the Combi-format to the actual function. The first level of the Combi-format consists of one basis function (we do not truncate to empty, this one point always exists) and is the same as the combination technique of one level. Since we wish to calculate the approximation error in this work, we already have the necessary values computed to calculate this error in our chosen form (analytical error). If this should in other work not be available, one might consider the normalized function.

Following all this we can formulate a convergence rate of the combination technique, which we want to match. (This is not a strict algorithm; adjustments might need to be made, for example if the convergence only sets in after a certain finite number of levels.) Now there are two general approaches:

1. We have a wanted precision ε. Based on this precision we choose our level of the format and the required truncation ranks.

2. We have a given maximal level m of the format. Based on this we wish to achieve the best precision ε possible and choose the required ranks accordingly.

First we consider the first case. Based on the starting precision for level 1 and the expected rate of convergence we estimate for which level this precision is achieved. This is only a rough estimate, recall that there are constants involved in the convergence rate estimate. If these constants are too big it might be necessary to adjust our estimate later. Again, this is only a rough estimate, but given not too big constants we can hope to at least make statements along the line of: for level m our approximation is exact up to i decimal places.

When we have this level m for our wanted precision ε (if we want to be sure we have to actually compute the combination technique approximation error for this level) we can consider truncation. The second case now proceeds similarly from here, only we first need to estimate the possible precision ε for the given level m. We have the value of the singular values of all the projected functions needed available, theoretically we could now formulate some very complicated optimization algorithm.

To actually keep the choice of truncation ranks calculable by hand we instead choose a simpler idea. First we check whether all singular values after a certain fixed number are completely insignificant.

Hereby we keep in mind in which way we calculate the truncation, for example in this work we consider the squared and normalized singular values. Here it is of course possible to consider other options, as long as we stay consistent and consider the resulting truncation error in a compatible norm.

In this context we consider complete insignificance to be a singular value which is either of the size of the precision of the format (which is determined by the precision of the integration necessary for the projection or even the precision of our SVD-algorithm) or compared to our wanted precision ε completely insignificant. In this work we consider this to be the case when the singular value (in our case normed and squared) is smaller than $\varepsilon \cdot 10^{-4}$. The factor 10^{-4} is chosen arbitrarily, as this seemed to be insignificantly small in our case.

If indeed all singular values (of all grids) are of insignificant size after a fixed number we treat the function as a function of said rank and truncate to this rank accordingly.

In the case that this is not the case we have to proceed a bit more generally: the idea is that based on our wanted precision we discard (truncate) all singular values which we judge to be significantly smaller than the wanted precision. All in all the added together, discarded (squared and normalized) singular values of all grids have to be smaller than the wanted precision.

If we do not wish to calculate this (even roughly) we can consider (6.22) and simply truncate uniformly only the minimum amount of completely insignificant singular values. Even this minimal truncation reduces the computational complexity of further calculations with the format and of course storage space.

In conclusion following this procedure of balancing the errors (eventually in multiple iterative steps) leads to a Combi-format with the same precision as the combination technique of the same level. This leads to the conclusion that the Combi-format can show the same rate of convergence as the combination technique, while reducing the size of the format. Unfortunately there are no precise estimates of how strong this reduction is, beyond the very rough (6.25).

7 Praxis

In the following section we consider the concepts necessary to implement the Combi-format in praxis. First we describe the algorithm we implemented for this work, which is the calculation of the approximation error of the Combi-format, measured in the analytical error. For this we need to be able to calculate the Combi-format of a given function (section 7.1.1) and then also the analytical approximation error as described in section 7.2. Finally in section 7.3 we demonstrate the results of this algorithm tested on different functions.

7.1 Algorithmic Considerations

Here we consider the algorithm necessary to create the Combi-format and to calculate its precision, i.e. the approximation error. Furthermore we explain how to implement the concepts discussed in this work in praxis.

7.1.1 Calculating the Combi-format

The first step to being able to calculate the Combi-format is to calculate the combination technique. We already discussed the basics of implementing the combination technique in section 3.2.

In the code accompanying this work we consistently implemented the d-dimensional formula (9.1). This keeps the code general and we avoid programming everything twice. As is noted in the explanation to (9.1), the definition of the d-dimensional formula is compatible with the two-dimensional one.

As a short recall: This means we need to calculate and store the object $P_{\mathbf{l}}f$ for all $\mathbf{l} \in \mathcal{I}_m^C$, where $\mathcal{I}_m^C$ is the index-set of grids in the combination technique (of level m) we wish to calculate. We have already discussed how to calculate $P_{\mathbf{l}}f$ in section 2.1.4.

Again, we wish to solve

$$\mathbf{M}_{\mathbf{l}}\mathbf{X}_{\mathbf{l}} = \mathbf{B}_{\mathbf{l}}$$

where $\mathbf{M}_{\mathbf{l}} = \mathbf{M}_{\mathbf{l}}^{(1)} \otimes \ldots \otimes \mathbf{M}_{\mathbf{l}}^{(m)}$. With $\mathbf{M}_{\mathbf{l}}^{(k)} = \Big(\langle \phi_{l_k,i}, \phi_{l_k,j}\rangle_{L^2(\Omega_k)}\Big)_{i,j\in\Delta_{l_k}}$ for $k = 1,\ldots,m$ and $\mathbf{B}_{\mathbf{l}} = \Big(\langle f, \phi_{\mathbf{l},\mathbf{i}}\rangle_{L^2(\Omega)}\Big)_{\overline{\mathbf{i}\in\Delta_{\mathbf{l}}}}$. The $\overline{\mathbf{i}\in\Delta_{\mathbf{l}}}$ signifies a combined index (a precise definition can be found in section 8), which combines the two indices $\mathbf{i} = (i_1, i_2)$ into one running index, since we need $\mathbf{B}_{\mathbf{l}}$ to be a vector in this two-dimensional case[5].

[5]We do not specify how exactly the bijection underlying the combined index is defined, it only needs to be consistent, meaning compatible with the ordering in $\mathbf{M}_{\mathbf{l}}$. As we see in section 8 the big-endian ordering fulfils this condition.

Fortunately, it holds that $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$. As such we can formulate $\mathbf{X_l} = \left(\left(\mathbf{M_l}^{(1)}\right)^{-1} \otimes \ldots \otimes \left(\mathbf{M_l}^{(m)}\right)^{-1} \right) \mathbf{B_l}$, where the $\mathbf{M_l}^{(k)}$ presumably have a structure that makes them easy to invert.

This is due to the fact that the $\mathbf{M_l}^{(k)}$ are the mass matrices of the chosen one-dimensional basis which, if it is a well-chosen basis, has an exploitable structure. We can note here that there exist an algorithm to transform a hierarchical hat-basis into the non-hierarchical one and back, which is important because the mass-matrix of the non-hierarchical hat-basis is a diagonal one, which is very easy to invert.

In the code attached to this work we do not use that algorithm and instead rely on the so-called finger-structure of the hierarchical hat-basis mass-matrix, which makes the matrix easier to invert than an arbitrary one.

In the above way we again circumvent the curse of dimensionality, since inverting a $d-$tensor naively has a complexity with d in the exponent. We are however not able to consider each direction (dimension) independently of each other, since $\mathbf{B_l}$ is not (in general) in tensor-product form. This is due to the fact that f, the function we wish to approximate, is in general not in tensor product form.

We also need to define $\mathbf{M_l^k}$ for calculating the scalar product later on. It holds

$$\mathbf{M_l^k} = \left(\langle \phi_{\mathbf{l},\mathbf{i}}, \phi_{\mathbf{k},\mathbf{j}} \rangle_{L^2(\Omega)} \right)_{\overline{\mathbf{i} \in \Delta_\mathbf{l}}, \overline{\mathbf{j} \in \Delta_\mathbf{k}}}.$$

The relation between the mass-matrix of level $\mathbf{l}$ and these mass-matrices of mixed levels is

$$\mathbf{M_l^l} = \mathbf{M_l}.$$

In the code attached to this work we define a struct (a kind of class) named `SubspaceD_ID` to store all information we need to now for each subgrid. Since our code is for a general d-dimensional case we also use tensors instead of matrices and vectors, for the code we use the library ITensor [8]. Tensors will be introduced in section 8, for now all we need to know is that tensors include both matrices and vectors.

To compute the projections $P_\mathbf{l} f$, we first need the tensors $\mathbf{M_l}$ and $\mathbf{B_l}$. Because the numerical integration needed for the computation of $\mathbf{B_l}$ can be very costly (in our experience the by far costliest computation step) we only wish to calculate each $\mathbf{B_l}$ once and then store it for later use, preferably in a file. In our code this is accomplished by the function `loop_Btensor`, which computes the $\mathbf{B_l}$ for the full formats of the given levels.

Then we calculate for each subspace (using the function `Combi.Xtensor`) the projection, resulting in a tensor $\mathbf{X_l}$[6]. Afterwards the truncation is calculated, using either rank- or singular-value-truncation

[6]Here the concept of tensors is very useful, since $\mathbf{X_l}$ as we calculate it is a vector, but to truncate it we have to reshape it into a matrix

Since we do not, in the course of this work, wish to store the Combi-format, we do not actually compute the whole format at once. Instead we have an operation we wish to perform on the format (in our case this would be measuring the approximation error). Then we only calculate one sub-grid at once, perform the wanted operation on it and then calculating the next, only storing the result of the operation. Why we chose this method is explained in the following paragraph.

Parallelisation The Combi-format inherits from the combination technique the fact that it is a sum of independent from each other operations. This independence is important in praxis, since it allows for parallelization.

In general, we do not only wish to calculate the Combi-format, but actually perform a calculation on or with it. In this work that is "just" measuring the approximation error (see the next subsection) between the Combi-format and the continuous function but in general we can consider other operations as well.

Let us say that we wish to perform an operation on the Combi-format, the only restriction being that it has to be a linear operation. Then it is the point of the Combi-format that we perform this operation on each of the sub-grids individually and only at the end actually evaluate the expression. This means that the operation and the calculation of the Combi-format before that can be parallelized, by putting each sub-grid on a separate thread.

For example, if we wish to perform an operation $L : \hat{V}_m \longrightarrow \mathbb{R}$, which is linear in its argument, on a function $\tilde{f} = \mathcal{CF}_{m,\mathbf{r}} f$, then we consider

$$L\left(\mathcal{CF}_{m,\mathbf{r}} f\right) = \sum_{m \leq |\mathbf{l}|_1 \leq m+1} (-1)^{m+1-|\mathbf{l}|_1} L\left(T_r^{\mathbf{l}}(P_{\mathbf{l}} f)\right). \tag{7.1}$$

Here we use parallelization by calculating $L\left(T_r^{\mathbf{l}}(P_{\mathbf{l}} f)\right)$ for each $\mathbf{l} \in \mathcal{I}_m^C$ on a separate thread. Afterwards the results from the separate threads only need to be added together with the appropriate weights defined in the formula above.

This procedure cuts down on computation time significantly, even on an (at the time of this writing) average computer without multiple processors the time effort is cut approximately in one fourth of the non-parallel computation. On an actual computer-cluster of sufficient capabilities (it needs to be able to perform $\mathcal{O}(dm)$ threads at once and the communication time for the results of the threads cannot be significant) the computation time is approximately $\mathcal{O}((dm)^{-1})$ as long as the non-parallel computation.

This is a massive reduction in computation time and one of the reasons why the Combi-format and the combination technique are used. The possibility to parallelize is a huge advantage over other non-parallel methods.

7.2 Approximation-error Calculation

The one operation we wish to perform on the Combi-format is the calculation of the approximation error. This is how we justify our concept: We wish to show that the Combi-format converges to the exact solution, and that the rate of convergence signifies practical usability.

To do this we first need to consider how to measure the error. Mainly we will focus on the *analytical error*, this means we calculate for a function $f \in \mathcal{H}$:

$$\|f - \mathcal{CF}_{m,r}(f)\|^2_{L^2(\Omega)} = \langle f, f\rangle_{L^2(\Omega)} - 2\,\langle f, \mathcal{CF}_{m,r}(f)\rangle_{L^2(\Omega)} + \langle \mathcal{CF}_{m,r}(f), \mathcal{CF}_{m,r}(f)\rangle_{L^2(\Omega)}\,.$$

On the right hand side of the equation above there are three terms which need to be computed separately.

The Norm $\langle f, f\rangle_{L^2(\Omega)}$: This is exactly the square of the $L^2(\Omega)-$norm of the function f, meaning we need to calculate the norm of f. If it is a-priori known or can be calculated analytically somehow, the exact term can be used. However, since in general the exact norm of the function is not known, this usually needs to be done via numerical integration. Here it is extremely important to choose the precision of this integration to be much higher than the precision of the numerical integration performed to approximate the projection in the Combi-format. Otherwise the error of this numerical integration would reduce the following calculations to nonsense.

The mixed Term $\langle f, \mathcal{CF}_{m,r}(f)\rangle_{L^2(\Omega)}$: Here we use the procedure of applying an operator to the Combi-format as discussed before, in this case the operator in question is the scalar product with f: $L = \langle f, \cdot\rangle_{L^2(\Omega)}$. As such it holds

$$\langle f, \mathcal{CF}_{m,r}(f)\rangle_{L^2(\Omega)} = \sum_{|\mathbf{l}|_1 = m+1} \left\langle f, T_r^{l_1,l_2} P_{l_1,l_2} f\right\rangle_{L^2(\Omega)} - \sum_{|\mathbf{l}|_1 = m} \left\langle f, T_r^{l_1,l_2} P_{l_1,l_2} f\right\rangle_{L^2(\Omega)}\,.$$

Now we use the fact that each element in a space $V_\mathbf{l}$ has a unique basis-representation, recall (1.19) for the hierarchical representation thereof, which is also unique. Therefore we can write

$$T_r^\mathbf{l} P_\mathbf{l} f \hat{=} \sum_{\mathbf{i}\in\Delta_\mathbf{l}} \tilde{f}_{\mathbf{l},\mathbf{i}} \phi_{\mathbf{l},\mathbf{i}}.$$

Here the matrix[7] $\mathbf{X}_{\mathbf{l},r} = T_r^\mathbf{l}(P_\mathbf{l} f)$ has the coefficients $\tilde{f}_{\mathbf{l},\mathbf{i}}$, meaning

$$\mathbf{X}_{\mathbf{l},r} = \left(\tilde{f}_{\mathbf{l},\mathbf{i}}\right)_{\mathbf{i}\in\Delta_\mathbf{l}}\,.$$

[7]Recall that we had to reshape $\mathbf{X}_\mathbf{l}$, which was originally a vector, for the truncation. To not overload notation we omit the operator indicating this here and everywhere in this work.

Since we wish to compute the summands in the sum of the combination technique in parallel (which we can do as explained above), we can consider these summands independently of each other. This means we consider only one summand identified by its level-index $\mathbf{l} \in \mathcal{I}_m^C$:

$$\begin{aligned}
\left\langle f, T_r^{\mathbf{l}}(P_{\mathbf{l}}f)\right\rangle_{L^2(\Omega)} &= \left\langle f, \sum_{\mathbf{i}\in\Delta_{\mathbf{l}}} \tilde{f}_{\mathbf{l},\mathbf{i}}\phi_{\mathbf{l},\mathbf{i}}\right\rangle_{L^2(\Omega)} \\
&= \sum_{\mathbf{i}\in\Delta_{\mathbf{l}}} \left\langle f, \tilde{f}_{\mathbf{l},\mathbf{i}}\phi_{\mathbf{l},\mathbf{i}}\right\rangle_{L^2(\Omega)} \\
&= \sum_{\mathbf{i}\in\Delta_{\mathbf{l}}} \tilde{f}_{\mathbf{l},\mathbf{i}} \left\langle f, \phi_{\mathbf{l},\mathbf{i}}\right\rangle_{L^2(\Omega)} \\
&= \mathbf{B}_{\mathbf{l}}\mathbf{X}_{\mathbf{l},r}
\end{aligned}$$

where $\mathbf{B}_{\mathbf{l}}$ (reshaped as a matrix) is defined as in section 7.1.1 above.

Concluding, we need to perform one matrix-matrix multiplication per summand, so for level m a total of $2m+1$ matrix-matrix multiplications. The matrices $\mathbf{B}_{\mathbf{l}}$ we have already calculated for the projection as described above, to speed up computation significantly they ought to be stored in memory.

The Format $\langle \mathcal{CF}_{m,r}(f), \mathcal{CF}_{m,r}(f)\rangle_{L^2(\Omega)}$: This is the square of the norm of the function f approximated in the Combi-format, which means we need to compute that norm.

Since we can in general not assume that the bases of the different subgrids are orthogonal to each other, it is easily seen that the computation of this term is no longer linear but quadratic, it involves two nested loops.

$$\langle \mathcal{CF}_{m,r}(f), \mathcal{CF}_{m,r}(f)\rangle_{L^2(\Omega)} = \sum_{\mathbf{l}\in\mathcal{I}_m^C}\sum_{\mathbf{k}\in\mathcal{I}_m^C} (-1)^{m+1-|\mathbf{l}|_1}(-1)^{m+1-|\mathbf{k}|_1}\left\langle T_r^{\mathbf{l}}P_{\mathbf{l}}f, T_r^{\mathbf{k}}P_{\mathbf{k}}f\right\rangle_{L^2(\Omega)}$$

Then, analogously to above, we represent $T_r^{\mathbf{l}}P_{\mathbf{l}}f$ in its unique basis representation, resulting in

$$\begin{aligned}
&\langle \mathcal{CF}_{m,r}(f), \mathcal{CF}_{m,r}(f)\rangle_{L^2(\Omega)} \\
&= \sum_{\mathbf{l}\in\mathcal{I}_m^C}\sum_{\mathbf{k}\in\mathcal{I}_m^C} (-1)^{2(m+1)-|\mathbf{l}|_1-|\mathbf{k}|_1}\left\langle \sum_{\mathbf{i}\in\Delta_{\mathbf{l}}} \tilde{f}_{\mathbf{l},\mathbf{i}}\phi_{\mathbf{l},\mathbf{i}}, \sum_{\mathbf{j}\in\Delta_{\mathbf{k}}} \tilde{f}_{\mathbf{k},\mathbf{j}}\phi_{\mathbf{k},\mathbf{j}}\right\rangle_{L^2(\Omega)} && (7.2) \\
&= \sum_{\mathbf{l}\in\mathcal{I}_m^C}\sum_{\mathbf{k}\in\mathcal{I}_m^C}\sum_{\mathbf{i}\in\Delta_{\mathbf{l}}}\sum_{\mathbf{j}\in\Delta_{\mathbf{k}}} (-1)^{2(m+1)-|\mathbf{l}|_1-|\mathbf{k}|_1}\tilde{f}_{\mathbf{l},\mathbf{i}}\tilde{f}_{\mathbf{k},\mathbf{j}}\left\langle \phi_{\mathbf{l},\mathbf{i}}, \phi_{\mathbf{k},\mathbf{j}}\right\rangle_{L^2(\Omega)} && (7.3) \\
&= \sum_{\mathbf{l}\in\mathcal{I}_m^C}\sum_{\mathbf{k}\in\mathcal{I}_m^C} (-1)^{2(m+1)-|\mathbf{l}|_1-|\mathbf{k}|_1}\mathbf{X}_{\mathbf{l},r}^T\mathbf{M}_{\mathbf{l}}^{\mathbf{k}}\mathbf{X}_{\mathbf{k},r}, && (7.4)
\end{aligned}$$

where $\mathbf{X}_{\mathbf{l},r}$ is defined as above and $\mathbf{M}_{\mathbf{l}}^{\mathbf{k}}$ as in section 7.1.1.

This means that for each grid denoted by $\mathbf{l} \in \mathcal{I}_m^C$ we need to calculate

$$\sum_{\mathbf{k} \in \mathcal{I}_m^C} (-1)^{2(m+1)-|\mathbf{l}|_1-|\mathbf{k}|_1} \mathbf{X}_{\mathbf{l},r} \mathbf{M}_{\mathbf{l}}^{\mathbf{k}} \mathbf{X}_{\mathbf{k},r}^T.$$

As such every sub-grid in the format requires one full loop over the whole format. Should the choice of basis guarantee some sort of orthogonality (that certain sub-grids are orthogonal to others) this can be reduced somewhat.

Our case makes the two nested loops fully necessary. In the code this second loop can be found in the functions `ind_subspace_error_nl` , `ind_subspace_error_sv_nl` and `ind_subspace_error_cut_nl`.

7.2.1 Interpolation Error

In section 2.3 we considered the difference between linear and standard information. Recall that projection, which we have consistently used throughout this work, is linear information. Interpolation on the other hand is considered standard information, and relating it to projection is therefore still an open problem and highly non-trivial.

But we can nonetheless use interpolation in formulating an error analysis: The interpolation error (meaning the error between the exact function and the approximated one calculated by computing the difference of function evaluations) bounds the analytic error from above. It can therefore be used for quickly bounding the error, especially if algorithms for efficient evaluation exist.

Since we use hierarchical hat-bases this means one needs an algorithm for calculating the transformation from hierarchical values to nodal values. This collection of algorithms is an integral part of using sparse grids in practice. For these algorithms consult for example [7, Technical Reference].

7.3 Convergence Analysis in Praxis

Now that we have defined the algorithm we wish to consider in this work, we turn towards the actual praxis. In the previous we have formulated a few properties of the Combi-format, including a convergence analysis and a heuristic for choosing the truncation ranks. These we wish to illustrate with practical examples.

In the following we therefore consider different examples in praxis, to show the behaviour and advantages of our Combi-format. As such we apply the format on functions from different function classes and of different smoothness. This is not an exhaustive study, but does show the versatility of the format.

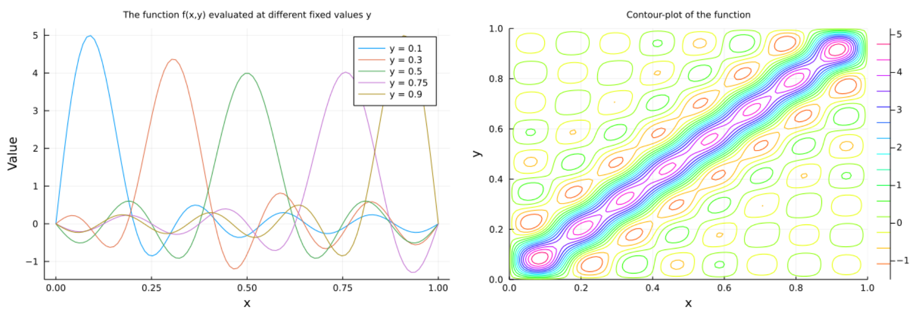


Figure 6: The function (7.5): on the left evaluated for different fixed values in the second coordinate and on the right as a contour-plot.

A rank-8 function The first function we consider is the following

$$f(x, y) = \sum_{i=1}^{8} \sin(i \cdot \pi \cdot x) \cdot \sin(i \cdot \pi \cdot y). \tag{7.5}$$

We show this function in Figure 6, once evaluated for fixed values of the second coordinate and once as contour-plot. It is a wave-function, a sum of products of sinus functions. This means it holds $f \in C^\infty$ and therefore also $f \in H^{(s_1,s_2)}_{mix}([0,1]^2)$ for all $s_1, s_2 > 0$. More colloquially said: f is a very smooth function.

Since we also know that the function $\sin(i \cdot \pi \cdot x)$ is linearly independent of $\sin(j \cdot \pi \cdot x)$ when $i > j$, we can conclude that f is of rank 8. Since $(x = 0 \ \vee \ y = 0) \Rightarrow f(x, y) = 0$ holds, f is zero on the boundary and therefore fulfils all conditions for the Combi-format ($f \in L^2([0,1]^2)$ trivially).

In Figure 7 we can fist see the convergence for the Combi-format without any truncation, which is merely the combination technique. We can also see that for a given level l of the format, the convergence rate of the combination technique is essentially $\mathcal{O}(2^{-2l}) = \mathcal{O}(h^2)$ for $h = 2^{-l}$. This is exactly the result we expect from our theory, also consult Theorem 12. Our function might be in $H^{(s_1,s_2)}_{mix}$ for all strictly positive $\mathbf{s}$, but we have $s_1 \leq r_1$ and $s_2 \leq r_2$ with r_1, r_2 being defined by the polynomial exactness of the underlying spaces. For our particular sparse-grid setup it holds $r_1 = r_2 = 2$, which explains the convergence rate.

This convergence does not set in immediately, which is due to the structure of the function f. Initially, the mesh-width of the grids is not fine enough to catch the high oscillation parts of f. The Figure 6 illustrates this, if one imagines a sparse grid of a lower level ($m < 6$) laid over the function, one can see that some of the smaller contour-circles lie completely between the grid-points. This phenomenon makes the considerations necessary for the choosing of truncation ranks harder.

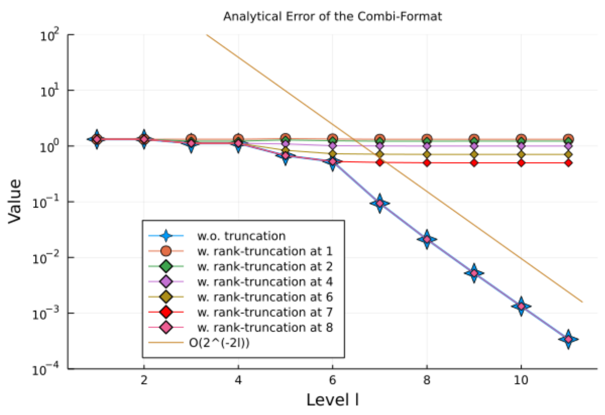


Figure 7: The convergence of a rank-8 function (7.5) approximated in the Combi-format measured in the analytical error.

For this function we do not need to worry about that though, since the first important result found in the figure is that indeed, the Combi-format with rank-truncation after rank 8 shows the same convergence as the combination technique. This means that the approximate function $\mathcal{CT}_m f$ has indeed rank 8 in praxis, as it should as discussed in the theory.

Also visible is the fact that truncating before rank 8 cuts off too much information: the Combi-format converges towards a function which is not the function f.

In Figure 8 we can see the convergence behaviour for the singular-value-truncation. We can compare to the singular values of the different sub-grids (the plot only shows half of the ones with $|\mathbf{l}|_1 = m+1$, since the function is symmetric the plot for the sub-grid of level (i, j) is the same as the one for the sub-grid of level (j, i)) shown in the second, lower plot. The plot shows the normalized and squared value of the singular values of $P_{\mathbf{l}} f$ for the shown $\mathbf{l}$. Normalized and squared in this case means the value of the i-th singular value is

$$\frac{\sigma_i^2}{\sum_{\alpha=1}^{\min(l_1,l_2)} \sigma_\alpha^2}.$$

The second plot also helps us to choose truncation ranks: We can see that the function is only of rank 8 by the fact that all the others are of the size of the calculation error. Since we chose the precision of the integration for the projection to be 10^{-12}, this is the maximum precision we can expect of our method. We can also see that all the significant (squared) singular values are above $\varepsilon = 10^{-3}$.

What can be clearly seen in Figure 8 is the fact that the singular-value-truncation with $\varepsilon = 10^{-3}$ is again just the combination technique, no truncation actually takes place. For $\varepsilon = 10^{-2}$ on the other hand the truncation

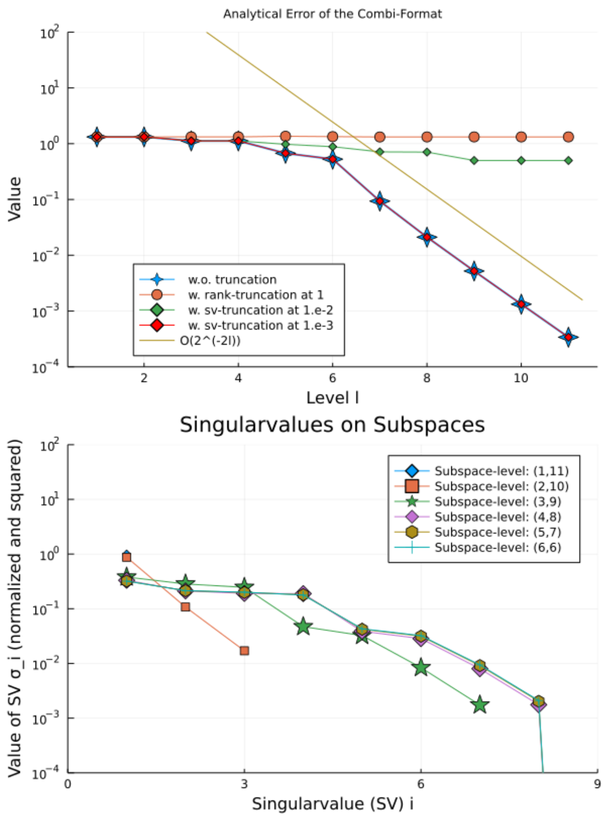


Figure 8: First: The convergence of the function (7.5) in the Combi-format with different truncations; below that the singular values of the projected functions on a number of sub-grids identified by their level-indices.

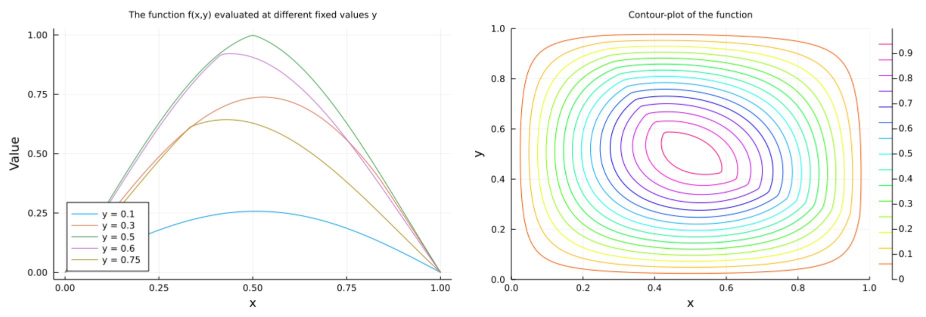


Figure 9: The function (7.6): on the left evaluated for different fixed values in the second coordinate and on the right as a contour-plot.

cuts off too many singular values to still converge in an acceptable matter. This validates our above statements.

As a conclusion we can formulate that it does hold true that a function of rank r is approximated via the Combi-format by a function of at most rank 8. Truncating this function further or applying singular-value-truncation on the other hand makes no practical sense. An explanation for this is given by the singular values of the respective sub-grids; they do not decrease fast enough to make further truncation in any way useful.

This is still an important result: the biggest sub-grid we calculated for the plots is of level $(6,6)$, this results in a matrix of size 63×63 with a total of 3969 entries that need to be stored and are used in further calculations we might want to do with the format. After truncating after rank 8 we need to store (in the SVD) two matrices of size 63×8, or only 1008 entries. That is a reduction by a factor of $\frac{1008}{3969} = \frac{16}{63} \approx 0.254$, so almost a quartering of the data size.

Further abstracted we replace a matrix of size $2^L - 1 \times 2^L - 1$ by two of size $2^L - 1 \times 8$. The higher the level is, the more we gain from using the Combi-format instead of just the combination technique, which is exactly what we wanted as it gives us a justification for its existence.

An absolute-value function The second function we consider is a function with a "kink", a function based on an absolute value function:

$$f(x,y) = sin(\pi x) \cdot sin(\pi y) \frac{1}{|x \cdot y - 0.25| + 1}. \tag{7.6}$$

It is shown in Figure 9.

We need to multiply the fraction with the absolute value function by a product of sinus functions to fulfil the boundary condition. (Of course we could also use other functions which are zero on the boundary, for example polynomials. In this case we chose the sinus function.)

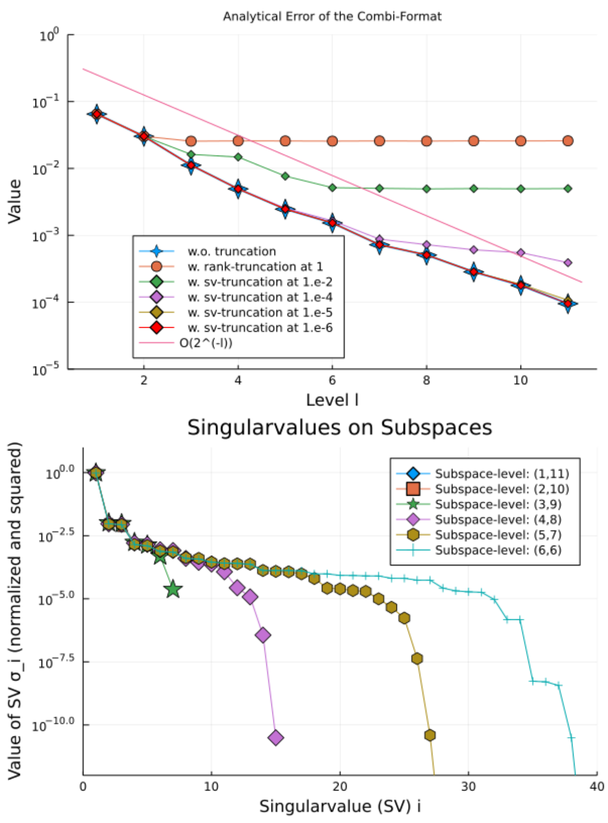


Figure 10: First: The convergence of the function (7.6) in the Combi-format with different truncations; below that the singular values of the projected functions on a number of sub-grids identified by their level-indices.

Since f contains an absolute value function in the shown form it is only in $H_{mix}^{(1,1)}\left([0,1]^2\right)$. As such the rate of convergence we can expect from the combination technique is slower than for the function above, only $\mathcal{O}(h) = \mathcal{O}(2^{-l})$ for level l.

This convergence can indeed be observed in Figure 10.

First we again study the second, lower plot showing us the singular values of the projected functions. This time there is no sudden fall after a fixed rank, we cannot determine the rank of the function off this plot. What we can determine is that only very few (squared and normalized) singular values are smaller than $\varepsilon = 10^{-6}$. We can, even without precisely calculating the sum of the squared and normalized singular values which are smaller than 10^{-6}, estimate that said sum is smaller than 10^{-6}. Considering the rate of convergence the combination technique for our function achieves, we assume that this is not significant for the levels that we can compute (if we were to calculate far higher levels this might of course change). Indeed, we can see in the upper plot that truncating at $\varepsilon = 10^{-6}$ achieves the full convergence up to the level we have computed.

Checking the precise values of the singular values from the output of the program tells us that for $\mathbf{l} = (6, 6)$ we reduced the rank of our approximate function on that sub-grid by 29 compared to the combination technique. This is of course not such a significant reduction as for the function above, but it is still significant when justifying our new format: It reduces computational complexity for all calculations one wishes to perform with the function as approximated by the Combi-format.

When we were choosing to truncate at $\varepsilon = 10^{-6}$ above, we did not use our notion of complete insignificance (recall section 6.4) and instead chose a bigger factor to determine significance. This does show that the factor of 10^{-4} we chose (arbitrarily) for our notion of complete insignificance is chosen to be very much "on the safe side". When we do not need to consider generality but rather a specific function, we can evidently achieve better results with specifically chosen bigger factors (in our case we could estimate the precision of the combination technique to be between the fourth and fifth decimal point, as such the factor was between 10^{-1} and 10^{-2}).

However, the chosen factor cannot be to big, this is shown in the plot for truncation at $\varepsilon = 10^{-5}$. In this case the factor is bigger than 0.1 and as can be seen in Figure 10 in the upper plot the Combi-format with singular-value-truncation at $\varepsilon = 10^{-5}$ does not reach the error of the combination technique or the Combi-format with singular-value-truncation at $\varepsilon = 10^{-6}$. Even though it is only visible for level 11, the Combi-format with singular-value-truncation at $\varepsilon = 10^{-5}$ does not reach the optimal convergence. This is easy to explain: we do not only consider one grid, but rather $2m + 1$ (for $m = 11$ that makes 23) many. Even with some potential error deletion through the combination technique, a factor of more than 0.1 is not enough to balance this out.

Strictly speaking we do not truncate on all of the $2m+1$ grids, some are too small and already have a small enough rank. For $m = 11$ and $\varepsilon = 10^{-5}$ respective $\varepsilon = 10^{-6}$ in the above example we only need to consider 10 grids (recall that we did not show all grids in Figure 10) which even have singular values that are smaller than 10^{-5}. From this we follow that a factor that is smaller than 0.1 should be enough to achieve the wanted convergence. This naturally leads us to truncating at 10^{-6}, which as discussed above achieves the wanted convergence rate.

All of the above considerations reinforce that our heuristic for choosing the truncation ranks works as we had hoped.

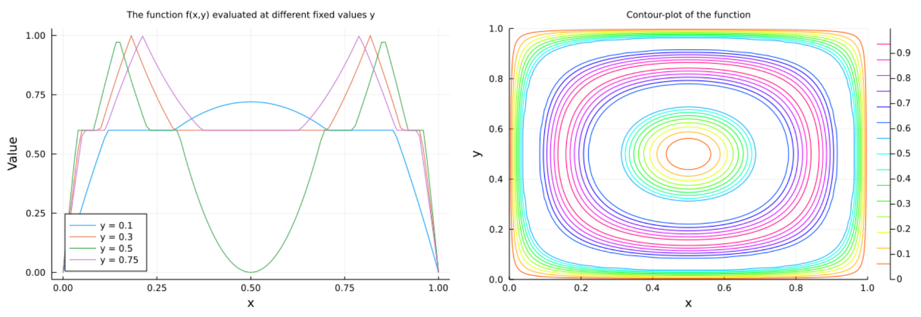


Figure 11: The function (7.7): on the left evaluated for different fixed values in the second coordinate and on the right as a contour-plot.

A piecewise polynomial function The final figure we consider is shown in Figure 11. It is a piecewise polynomial function, defined as

$$f(x,y) = \begin{cases} 16 \cdot x \cdot y \cdot (1-x) \cdot (1-y) & \text{if } r \leq 0.15 \\ 0.6 & \text{if } 0.15 \leq r \leq 0.3 \text{ or } 0.7 \leq r \leq 0.85 \\ 8 \cdot x \cdot y \cdot (1-x) \cdot (1-y) & \text{if } 0.3 \leq r \leq 0.5 \\ -8 \cdot x \cdot y \cdot (1-x) \cdot (1-y) & \text{if } 0.5 \leq r \leq 0.7 \end{cases} \tag{7.7}$$

where for the sake of readability we have set $r := 4 \cdot x \cdot y \cdot (1-x) \cdot (1-y)$.

The function is a continuous piecewise polynomial, as such the distinct pieces might be very smooth, but the overall function (consider especially the “kink” at 0.5) is only in $H_{mix}^{(1,1)}\left([0,1]^2\right)$ which as in the function above sets our expected rate of convergence for the combination technique to $\mathcal{O}(h) = \mathcal{O}(2^{-l})$ for level l.

This is indeed visible in both of the convergence plots, in the upper plot of Figure 12 and the plot of Figure 13.

Considering the lower plot of Figure 12 we can again not determine that the function is of a certain rank. But, since in the maximal level we can compute (based on our computers capabilities; in this case level 9) a singular value with size smaller than 10^{-10} is completely insignificant, we can indeed practically treat the function f as *almost* a rank-15 function. In Figure 13 this statement is proven true: truncating after rank 15 converges as the combination technique and therefore the fastest we can possibly achieve in this setting.

The upper plot shows the same principle, only this time with the singular-value-truncation in mind: considering that all the significant singular values (so all minus those smaller than computational precision and in this case the 16th, which is smaller than 10^{-10}) are bigger than 10^{-4}, if we truncate at 10^{-4}, these singular values will remain and those we deemed insignificant will be discarded. From this we arrive at the same truncation as after rank

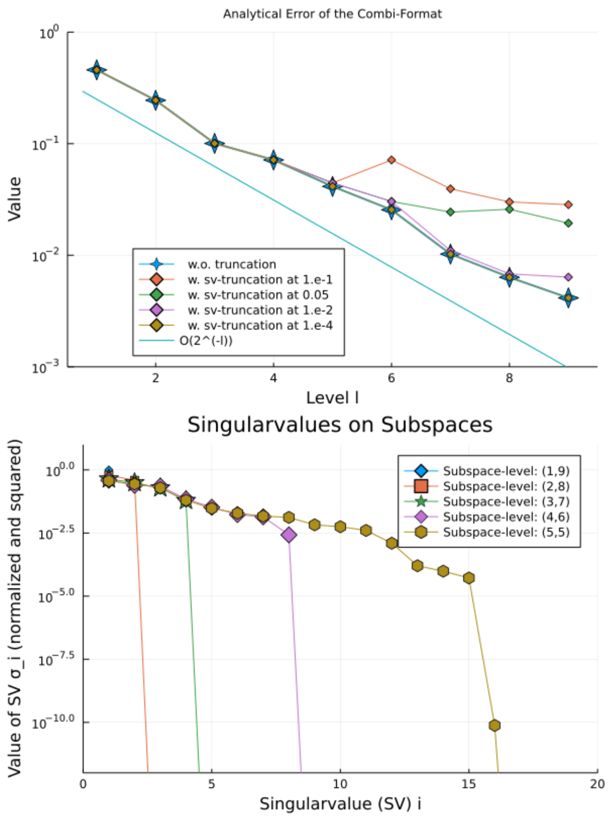


Figure 12: First: The convergence of the function (7.7) in the Combi-format with different truncations; below that the singular values of the projected functions on a number of sub-grids identified by their level-indices.

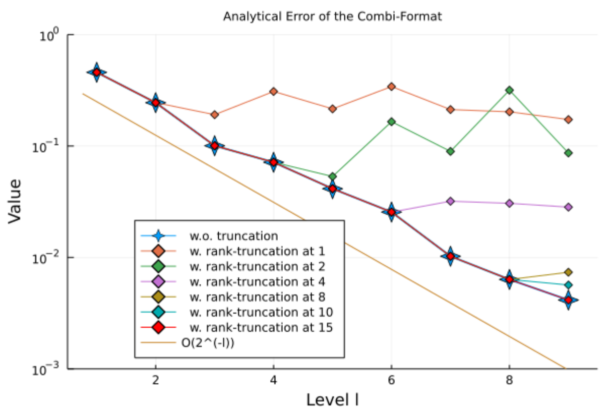


Figure 13: The convergence of the function (7.7) approximated in the Combi-format with different truncations and measured in the analytical error.

15, which achieves the same rate as the combination technique.

In this particular case it is worth noting that the integration necessary for the projections took a long time, over ten hours on a standard laptop. As such we have limited ourselves to at most nine levels of the format.

Conclusion All in all we can say that our theoretical considerations are validated by the practical experiments. We show that the Combi-format can reach the same order of convergence as the combination technique in praxis, while also reducing the cost in a number of variables sense.

This reduction in cost is achieved via a reduction in rank, which is especially significant for functions of a fixed rank (as our first example) or function which are very close to being functions of a fixed rank (as in the third example). But even for other functions such a reduction can be observed, without losing convergence.

The result of the Combi-format achieving the same rate of convergence as the combination technique is only for the case of having a fixed level (the truncation ranks depend on the level, as such there is no simple way to consider a limit to infinity), but since this is necessary in praxis it does not pose a problem, as observable above.

Also observable is that our heuristic for choosing the truncation ranks achieves the wanted results and is therefore useful. This concludes that we have a justification for our new Combi-format: we achieve in context optimal convergence with a reduction in storage cost compared to other methods.

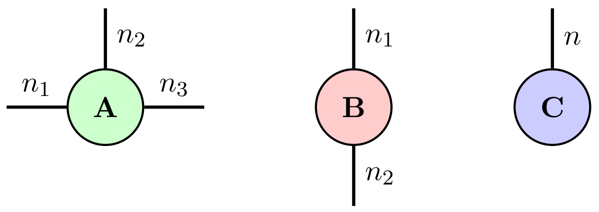


Figure 14: Three different tensors in tensor diagram notation: (1) The 3-tensor $\mathbf{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. (2) The 2-tensor $\mathbf{B} \in \mathbb{R}^{n_1 \times n_2}$, which is a matrix. (3) The 1-tensor $\mathbf{C} \in \mathbb{R}^{n}$, which is a vector.

Part III
Outlook on higher Dimensions

The results of the previous part show that the Combi-format can justify its existence. However, although considerations in two dimensions are not unimportant, the true importance of the combination technique and underlying that the sparse grid method lies in higher dimensions. There are multiple fields in mathematics which rely on great amount of data in many dimensions, for example in finance or machine learning.

As such the true test of the Combi-format is not whether it can be useful in two dimensions, but whether it can show similar results in higher dimensions.

Unfortunately, the singular value decomposition does not easily generalize to more than two dimensions and there is no Eckart-Young-Mirsky theorem for this case. Therefore we generalize matrices to tensors in the following section 8 and then generalize the SVD to tensor decompositions, specifically the tensor train decomposition. This decomposition defines the tensor-train format, often only called TT-format, which we will define in section 8.1.

Following this we can then define the Combi-format for more than two dimensions in section 9 and give an outlook on the results possible. Due to the lack of a Eckart-Young-Mirsky theorem the formulated error estimate is less precise than for the two dimensional case, but we are still able to show convergence in our practical example in section 9.4.

8 Tensors and the TT-format

When we consider more than two dimensions, we need to generalize the concept of matrices to higher dimensions. This is the field of tensors in

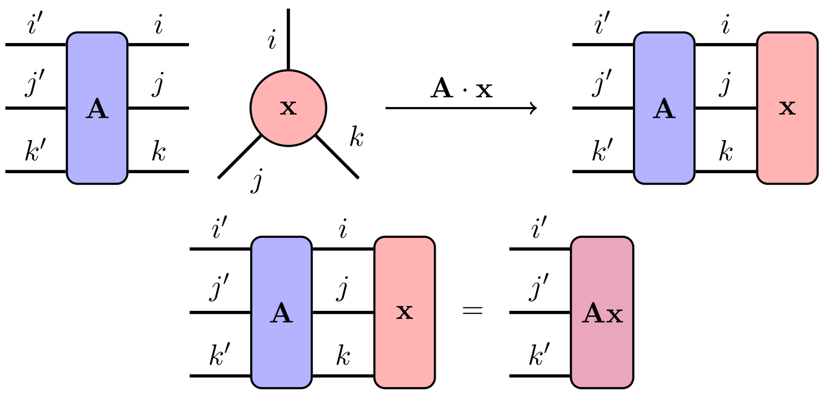


Figure 15: A step-by-step illustration of the tensor-matrix-by-tensor multiplication $\mathbf{A} \cdot \mathbf{x}$ in the tensor diagram notation.

numerics (there are various definitions of what a tensor is in the different sciences, for us it is a generalization of a matrix).

Since especially in practical applications tensors can become unwieldy quickly, there are multiple so-called *tensor formats*, which represent the full tensor in some easier to handle form. The field of tensor representations is a wide one with many applications, mostly in areas where a lot of data is needed. In this context we will consider some concepts from the area of low-rank tensor decompositions. Introductions to this topic can be found for example in [35, 24].

Definition 17 (Tensor)**.** A *$d-$tensor* $\mathbf{x} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ is a d-dimensional array. It is defined over *directions* or *modes* $\{1, \ldots, d\}$, the length in each direction given by the *mode sizes* $n_1, \ldots, n_d$.

A *tensor matrix* $\mathbf{A} \in \mathbb{R}^{n_1 \times \ldots \times n_d \times m_1 \times \ldots \times m_d}$ is also a tensor, which operates on tensors.

In Figure 15 the use of a tensor matrix is illustrated. The mode sizes are also sometimes called the dimension of the direction (or mode) in question. To not add to the potential confusion we will not use this in this work, although it appears in the cited literature.

In Figure 14 some examples of tensors are illustrated in tensor diagram notation [5]. Often we will not write the mode size on the "arms" representing the modes, but rather the name of its index. We use *Einstein notation* to sum over common indices, which gives meaning to expressions like $\mathbf{Ax} \in \mathbb{R}^{m_1 \times \ldots \times m_2}$, as diagrammed in Figure 15. Summing over a common index to essentially make one tensor out of two is also known as a *contraction (over the index)*. Adding two tensors of the same size or multiplying one by a scalar works analogously to matrices.

Matricization Actually using tensors of more than two dimensions (that being matrices) in their form as a multidimensional array can be cumbersome. Also many concepts (like SVD and orthogonalization) are only defined in two dimensions, and as such a method to represent a tensor as a matrix is very helpful and in some situations even necessary. The following is based on [30], some more details can also be found in [35, 11].

First we need to define the concept of a *combined index*, which we have already briefly seen in section 7.1.1. A combined index of dimension d is defined as

$$\overline{i_1 i_2 \dots i_{d-1} i_d} = g(i_1, \dots, i_d)$$

with an in theory arbitrary bijection g. Two examples of possible bijections are the commonly used *little-endian*

$$\overline{i_1 i_2 \dots i_{d-1} i_d} = i_1 + (i_2 - 1)n_1 + \dots + (i_d - 1)n_1 \dots n_{d-1}$$

and *big-endian*,

$$\overline{i_1 i_2 \dots i_{d-1} i_d} = i_d + (i_{d-1} - 1)n_d + \dots + (i_1 - 1)n_2 \dots n_d$$

lexicographical orderings.

In this work we mostly use the latter, because of its compatibility with the *Kronecker (tensor) product.* On the other hand the little-endian index grouping is often found in programming environments. Transitioning from one to the other is unproblematic via another bijection, but it is important to remember to do so, especially in praxis.

Now we can define a *matricization* of a tensor $\mathbf{x}(i_1, \dots, i_d)$ as a matrix defined as

$$X(\overline{i_{j_1} \cdots i_{j_k}}, \overline{i_{j_{k+1}} \cdots i_{j_d}}) = \mathbf{x}(i_1, \dots, i_d)$$

where the $j_n, n = 1, \dots, d$ are a reordering of the set $\{1, \dots, d\}$. In many cases we keep the ordering of the indices and can then formulate

$$X_{i_1,\dots,i_k}^{i_{k+1},\dots,i_d} = X(\overline{i_1 \dots i_k}, \overline{i_{k+1} \dots i_d}) = X_k.$$

This form of matricization is also called an *unfolding.* To simplify the notation we generally write the column-indices as superscript and the row-indices as subscript, understanding that they are combined indices even without the bar over them, in the interest of not overloading the notation.

8.1 Defining the Tensor Train Format

The *Tensor Train Format* (often just TT-format) is a low rank representation format for a tensor. It is based on decomposing the d-tensor into 3-tensors via singular value decomposition. In a numerics context it was introduced in [35], although it has (for example in physics) been known for some time under the name of *matrix product states (MPS).*

The following is strongly based on [30].

Definition 18 (TT format)**.** If $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ is a d-tensor, then $\bar{\mathbf{x}} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(d)})$ is its representation in the TT format, defined by:

$$\sum_{\alpha} \mathbf{x}^{(1)}(\alpha_0, i_1, \alpha_1)\mathbf{x}^{(2)}(\alpha_1, i_2, \alpha_2) \cdots \mathbf{x}^{(d)}(\alpha_{d-1}, i_d, \alpha_d).$$

All tensors are in boldface lettering, and $\bar{\mathbf{x}}$ is the tensor in the TT format. It is itself a vector of TT *cores* $\mathbf{x}^{(p)} \in \mathbb{R}^{r_{p-1} \times n_p \times r_p}, \quad p = 1, \dots, d$ which are 3-tensors. $\alpha_p = 1, \dots, r_p$ are called *rank indices*, their number r_p TT *ranks* (often only ranks, the distinction from matrix ranks clear from the context). The vector of TT ranks is $\mathrm{r}(\bar{\mathbf{x}}) = (r_1, \dots, r_{d-1})$, we can set $r_0 = r_d = 1$ without loss of generality to unify notation.

8.1.1 The TT-decomposition

This section's results are described in the following Theorem [35, Section 2]:

Theorem 19. *For each d-tensor* $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ *whose unfoldings have ranks*

$$rank\, X_k = r_k,$$

there exists a TT-decomposition (**??**) *with ranks no higher than* r_k*, for* $k = 1, \dots, d$*.*

Proof. This proof is a constructive argument, which defines a process by which one can decompose any tensor and which can be used as the proof of concept of the Algorithm 8.1.1. It can be found in more detail at [35, Theorem 2.1].

First, we consider the matricizations of $\mathbf{x}$, specifically X_1. This matrix can be decomposed via the SVD, as

$$X_1 = U \Sigma V^T.$$

Since the TT-format does not contain tensor-cores of singular values we immediately contract Σ and V^T and name the result also V.

This results in a decomposition (this time written down in the index form[8]):

$$X(i_1, \overline{i_2 \dots i_d}) = \sum_{\alpha_1=1}^{r_1} U(i_1, \alpha_1) V(\alpha_1, i_2, \dots, i_d)\,.$$

Here, the tensor U only depends on i_1 and α_1 and is therefore a good candidate for $\mathbf{x}^{(1)}$. Also note that the ranks of U are $(1, r_1)$, as stated in the theorem.

Now we wish to iterate and further decompose V, which we can do by considering it as a $(d-1)$-tensor $\mathbf{v} = V(\overline{\alpha_1 i_2}, i_3, \dots, i_d)$. In order to fulfil

[8]Remember: $r_0 = r_d = 1$; which is why α_0 and α_d can be neglected

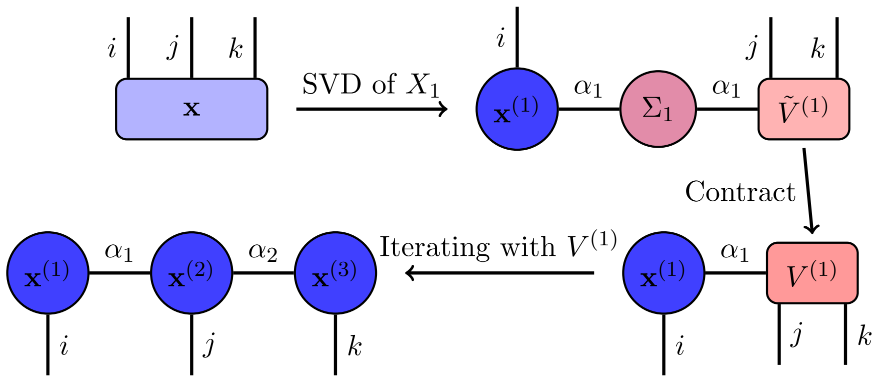


Figure 16: An illustration of the TT-decomposition of a 3-tensor $\mathbf{x}$, according to the TT-SVD-algorithm.

the rank-requirement of the theorem we only need to ensure that the ranks of the matricizations of $\mathbf{v}$ (which determine the TT-rank via the SVD, as above) are also maximum r_k for $k = 2, \dots, d$. Writing V_k in terms of A and U and separating column- and row ranks assures us that the condition holds and we have

$$\text{rank } V_k \leq r_k,$$

where $V_k = V(\overline{\alpha_1 i_2 \dots i_k}, \overline{i_{k+1} \dots i_d})$.

Continuing inductively we now separate the index (α_1, i_2) from the others via V_2 and the SVD and get

$$\mathbf{v}(\overline{\alpha_1 i_2}, i_3, \dots, i_d) = \sum_{\alpha_2} U(\alpha_1, i_2, \alpha_2) V'(\alpha_2, i_3, i_4, \dots, i_d),$$

where U is $\mathbf{x}^{(2)}$ this time. Through repeating this step we receive the entire TT-decomposition. □

Now we have a method for computing the TT-format of a tensor. But analogous to the discrete truncation via SVD defined in Definition 15 we do not only want to compute the format, we also wish to truncate. An idea is of course to just compute the TT-format as described in the proof above and then truncate it, a process called rounding the tensor, which will be defined and described in the section 8.2.

However, this is unnecessarily complicated and also computationally expensive, as we would have to compute the full format of a tensor which might possibly be approximated to a very high precision by a tensor with far smaller ranks. This low-rank approximation tensor does not only need far less storage space, but is also far less expensive to compute.

As such we consider Theorem 19, but instead of setting rank $X_k = r_k$ as our TT-ranks, we want to choose our truncation ranks $\hat{r}_k \leq r_k$. There are different ideas for how to choose them, which we will discuss in the following.

The general idea of the truncated TT-format is detailed in step 4 of the TT-SVD-Algorithm: instead of (as in the proof of the theorem) calculating the full SVD, we instead truncate it after a before chosen rank. In other words, in step 4 we use the operator $\mathcal{T}_{r_k}^{(r_{k-1}n_k, \frac{\text{lenght}(t)}{r_{k-1}n_k})}$.

Algorithm 1 TT-SVD

Require: $d-$tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$, truncation ranks $\mathbf{r} = (r_k)_{k=1}^{d-1}$
Ensure: $d-$tensor $\bar{\mathbf{y}} = \Upsilon_{\mathbf{r}}\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ in the TT-format with TT-ranks $\mathbf{r}$

```
1: Initialization: $\mathbf{t} = \mathbf{x}$, $r_0 = 1$
2: for $k = 1, \dots, d-1$ do
3:     t := reshape(t, $r_{k-1}n_k, \frac{\text{lenght}(t)}{r_{k-1}n_k}$)
4:     Compute the truncated SVD: t = $U\Sigma V^T + E$ where rank($\Sigma$) = $r_k$
5:     Set $\mathbf{y}^{(k)}$ = reshape($U, r_{k-1}, n_k, r_k$)
6:     t := $\Sigma V^T$
7: end for
8: Set $\mathbf{y}^{(d)}$ = t
9: return $\bar{\mathbf{y}} = (\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(d)})$
```

Following from this we can define:

Definition 20 (Truncated-TT-Operator)**.** For a $d-$tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ and truncation ranks $\mathbf{r} = (r_k)_{k=1}^{d-1}$, we define the *truncated TT-format*

$$\bar{\mathbf{y}} = \Upsilon_{\mathbf{r}}\mathbf{x} \tag{8.1}$$

as the output of Algorithm 8.1.1.

Here, the operator $\Upsilon_{\mathbf{r}}$ for a given $\mathbf{r}$ assigns each input-tensor the output of Algorithm 8.1.1.

Error Estimates We can estimate the error between the input of the Algorithm 8.1.1 and its output, as long as we choose the truncation ranks $\mathbf{r}$ in a certain way. These estimates were formulated by I. Oseledets in [35].

Low-ranking tensors do not often occur in praxis, as such we replace the requirement rank $X_k = r_k$ by

$$X_k = R_k + E_k, \quad \text{rank } R_k = r_k, \quad \|E_k\|_F = \varepsilon_k, \quad k = 1, \dots, d-1. \tag{8.2}$$

From this we can formulate an error estimate for the approximation by the algorithm.

Theorem 21. *For each d-tensor* $\mathbf{x} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ *whose unfoldings have ranks fulfilling* (8.2) *the TT-SVD-Algorithm 8.1.1 computes a tensor* $\mathbf{y}$ *in the TT-*

format with TT-ranks r_k and

$$\|\mathbf{x}-\mathbf{y}\|_F \leq \sqrt{\sum_{k=1}^{d-1} \varepsilon_k^2}. \tag{8.3}$$

Proof. We proof this using induction. For $d = 2$ we are considering the SVD and the statement of the theorem follows from (4.22).

Let $d > 2$ be arbitrary and start by considering the first unfolding X_1. It is decomposed as

$$X_1 = U_1 \Sigma_1 V_1^T + E_1 = U_1 Z_1 + E_1$$

where $Z_1 = \Sigma_1 V_1^T$ is the matrix that goes into the iteration in the TT-SVD-Algorithm. Z_1 is also equivalent to a $(d-1)-$tensor $\mathbf{z}_1$ analogous to the proof of Theorem 19. It will be approximated by some tensor $\hat{\mathbf{z}}_1$ in the following. This approximation will in iteration become $\mathbf{y}$, as per the algorithm. As such we need to consider it in our error estimate, and not the original Z_1.

However we first consider the error estimate, knowing $U_1 E_1 = 0$ by properties of the SVD:

$$\|\mathbf{x}-\mathbf{y}\|_F^2 = \left\|A_1 - U_1 \hat{Z}_1\right\|_F^2 = \|A_1 - U_1 Z_1\|_F^2 + \left\|U_1\left(\hat{Z}_1 - Z_1\right)\right\|_F^2$$

and then by definition and the knowledge that U_1 has orthonormal columns, we can formulate by properties of the SVD:

$$\|\mathbf{x}-\mathbf{y}\|_F^2 \leq \varepsilon_1^2 + \left\|\hat{Z}_1 - Z_1\right\|_F^2.$$

Now it remains to confirm that the unfoldings of $\mathbf{z}_1$ are bounded by the property (8.2) to be able to iterate. This is fairly easy to do once we consider $Z_1 = U_1^T X_1$ and since U_1 is orthonormal, this has to hold. Iterating through this induction we arrive at

$$\left\|\hat{Z}_1 - Z_1\right\|_F^2 \leq \sum_{k=2}^{d-1} \varepsilon_k^2.$$

This, together with the estimate above, closes the proof. □

As a direct consequence of this we can formulate the following corollary, which guarantees us quasi-optimality when it comes to tensor-approximation via the TT-SVD-Algorithm:

Corollary 1 (Quasi-optimality)**.** *For each d-tensor $\mathbf{x} \in \mathbb{R}^{n_1 \times \ldots \times n_d}$ and ranks $\mathbf{r}$, a best-approximation of $\mathbf{x}$ in the Frobenius-norm with TT-ranks bound by $\mathbf{r}$ exist, which will be called $\mathbf{x}^{best}$. Also the approximation $\mathbf{y}$ computed by the TT-SVD-Algorithm is quasi-optimal:*

$$\|\mathbf{x}-\mathbf{y}\|_F \leq \sqrt{d-1}\left\|\mathbf{x}-\mathbf{x}^{best}\right\|_F$$

The proof to this can be found in [35, Corollary 2.4].

8.2 Basic Operations

We introduce another notation mostly used for tensor representations of matrices, although it works for such representations of vectors as well [28].

Definition 22. Let $\mathbf{A}$ be a TT core of ranks (p, q) and mode size m. Then we define TT *blocks* of the core $\mathbf{A}$ as a vector in $\mathbb{R}^m$: $U_{\alpha,\beta}$, $\alpha = 1, \dots, p$, $\beta = 1, \dots, q$ with $\mathbf{A}(\alpha, i, \beta) = (U_{\alpha,\beta})_i$. This ensures that $\mathbf{A}$ can be represented by its *core matrix*

$$\begin{bmatrix} U_{1,1} & \cdots & U_{1,q} \\ \vdots & \ddots & \vdots \\ U_{p,1} & \cdots & U_{p,q} \end{bmatrix}$$

This notation is especially useful to describe large tensor matrices (in [28] it is used to define explicit representations), especially when combined with the following definition of the *rank core product* "$\bowtie$".

Definition 23 (rank core product)**.** Let cores $\mathbf{A}^{(1)} \in \mathbb{R}^{r_0 \times m_1 \times r_1}$ and $\mathbf{A}^{(2)} \in \mathbb{R}^{r_1 \times m_2 \times r_1}$ be represented by their TT blocks $U^{(1)}_{\alpha_0,\alpha_1}$ and $U^{(2)}_{\alpha_1,\alpha_2}$, $1 \leq \alpha_k \leq r_k$ for $k \in \{0, 1, 2\}$ of mode sizes m_1 and m_2, respectively. The rank product $\mathbf{A}^{(1)} \bowtie \mathbf{A}^{(2)}$ is defined as a core of ranks (r_0, r_2) consisting of blocks

$$U_{\alpha_0,\alpha_2} = \sum_{\alpha_1=1}^{r_1} U^{(1)}_{\alpha_0,\alpha_1} \otimes U^{(2)}_{\alpha_1,\alpha_2}$$

for all valid values of α_0, α_2 and mode size $m_1 m_2$.

For example,

$$\begin{bmatrix} U_{1,1} & U_{1,2} \\ U_{2,1} & U_{2,2} \end{bmatrix} \bowtie \begin{bmatrix} V_{1,1} & V_{1,2} \\ V_{2,1} & V_{2,2} \end{bmatrix} = \begin{bmatrix} U_{1,1} \otimes V_{1,1} + U_{1,2} \otimes V_{2,1} & U_{1,1} \otimes V_{1,2} + U_{1,2} \otimes V_{2,2} \\ U_{2,1} \otimes V_{1,1} + U_{2,2} \otimes V_{2,1} & U_{2,1} \otimes V_{1,2} + U_{2,2} \otimes V_{2,2} \end{bmatrix}$$

and if we set $U^{(i)}$ as the core matrix of some core $\mathbf{u}^{(i)}$ then it holds that $\mathbf{A} = A^{(1)} \bowtie A^{(2)} \bowtie \cdots \bowtie A^{(d)}$ and we can write:

$$\alpha\mathbf{A} + \beta\mathbf{B} = \begin{bmatrix} A^{(1)} & B^{(1)} \end{bmatrix} \bowtie \begin{bmatrix} A^{(2)} & \\ & B^{(2)} \end{bmatrix} \bowtie \cdots \bowtie \begin{bmatrix} \alpha A^{(d)} \\ \beta B^{(d)} \end{bmatrix}. \tag{8.4}$$

It stands to note that this is not a unique representation of this addition, in fact one with lower ranks might exist, just a simple one in terms of writing and computing. Important to note: to avoid confusion, core matrices will always be in square brackets, while ordinary matrices will be in parentheses.

An important conclusion is the fact that operations on tensors in the TT-format are performed on a core by core principle. This means that the tensors involved are generally of far smaller size than for the same operation on full tensors. This reduction in complexity is an important benefit of using the TT-format instead of full tensors. It is in analogue to the tensor product form of the projections we wish to consider. Only instead of in tensor product form, tensors in the TT-format are in a low-rank form (only a tensor in TT-format with ranks uniformly 1 is in fact in tensor product form).

For more complex operations in the TT-format consider [35, 24]. There is the concept of rounding, by which we compute a truncated SVD for each core to reduce rank, for example after having performed an addition. This is done by computing the QR-decomposition of both the cores adjacent to the considered rank in the direction of said rank. Then the truncated SVD of the product of the two R-matrices (which contain all the non-orthogonal data) is computed and the matrices re-multiplied to get the cores back.

As such it becomes clear why we wish to truncate while computing the TT-format, not only does it stop us from having to compute the full (potentially of high rank) format, truncating while computing also involves less decomposing and reshaping than rounding.

Also important is the concept of matrix-vector multiplication and computing the norm, more details can be found in [35, 24].

An important conclusion is that all these operations are performed on a core by core principle. This means that the tensors involved are generally of far smaller size than for the same operation on full tensors. This reduction in complexity is an important benefit to using the TT-format instead of full tensors.

8.3 Continuous Case

Analogous to section 4, where we consider the SVD in the continuous and discretized case, there is also a continuous version of the TT-format. It is based on the same principle as the tensor format, but applied to continuous functions via the continuous SVD and not tensors. The following results are based on the ones from [13].

As in [13] we consider $f \in H^k(\Omega_1 \times \ldots \times \Omega_d)$. In constructing the continuous TT-decomposition, we proceed as in Theorem 19, using the continuous SVD as defined in section 4.1. This ansatz leads us to separating the variables $x_1 \in \Omega_1$ and $(x_2, \ldots, x_d) \in \Omega_2 \times \ldots \times \Omega_d$ as

$$f(x_1, \ldots, x_d) = \sum_{\alpha_1=1}^{\infty} \sigma_{\alpha_1}^{(1)} \phi_{\alpha_1}^{(1)}(x_1) \psi_{\alpha_1}^{(1)}(x_2, \ldots, x_d).$$

The (1) in the superscript is there to clarify that these objects are defined by the first application of the continuous SVD, which is the splitting off of Ω_1. For example, the in the following appearing $\phi_{\alpha_2}^{(2)}$ is different from $\phi_{\alpha_1}^{(1)}$, not even defined over the same domain.

Now we proceed similarly to the discrete case by separating the combined index $(\alpha_1, x_2) \in \mathbb{N} \times \Omega_2$ from $(x_3, \dots, x_d) \in \Omega_3 \times \dots \times \Omega_d$. Again, we use the continuous SVD to get

$$\left(\sigma_{\alpha_1}^{(1)} \psi_{\alpha_1}^{(1)}(x_2, \dots, x_d)\right)_{\alpha_1}^{\infty} = \sum_{\alpha_2=1}^{\infty} \sigma_{\alpha_2}^{(2)} \left(\phi_{\alpha_1,\alpha_2}^{(2)}(x_2)\right)_{\alpha_1}^{\infty} \psi_{\alpha_2}^{(2)}(x_3, \dots, x_d),$$

here in convenient vector notation, since without loss of generality the indices α_i run to infinity.

Repeating this separation step until all directions are separated provides us the wanted format

$$f(x_1, \dots, x_d) = \sum_{\alpha_1=1}^{\infty} \cdots \sum_{\alpha_d=1}^{\infty} \phi_{\alpha_1}^{(1)}(x_1) \phi_{\alpha_1,\alpha_2}^{(2)}(x_2) \cdots \phi_{\alpha_{d-2},\alpha_{d-1}}^{(d-1)}(x_{d-1}) \phi_{\alpha_{d-1}}^{(d)}(x_d) \tag{8.5}$$

with $\sigma_{\alpha_{d-1}}^{(d-1)} \psi_{\alpha_{d-1}}^{(d-1)}(x_d) = \phi_{\alpha_{d-1}}^{(d)}(x_d)$.

Having d-many nested sums going from 1 to ∞ makes this format only theoretically calculable. As such we of course wish to truncate, which leads to the following definition.

Definition 24 (Continuous TT-Format)**.** For a function $f \in H^k(\Omega_1 \times \dots \times \Omega_d)$ we define the *continuous TT-format* with ranks $\mathbf{r} \in \mathbb{N}^{d-1}$ by

$$f_{\mathbf{r}}^{\mathrm{TT}}(x_1, \dots, x_d) = \sum_{\alpha_1=1}^{r_1} \cdots \sum_{\alpha_d=1}^{r_d} \phi_{\alpha_1}^{(1)}(x_1) \phi_{\alpha_1,\alpha_2}^{(2)}(x_2) \cdots \phi_{\alpha_{d-2},\alpha_{d-1}}^{(d-1)}(x_{d-1}) \phi_{\alpha_{d-1}}^{(d)}(x_d), \tag{8.6}$$

defined using the TT-decomposition from above.

It is very important to note here that the functions $\phi_{\alpha_{i-1},\alpha_i}^{(i)}$ in (8.6) are not the same as in (8.5). This is due to the fact that by truncating to rank r_i in step i we change the function to be approximated in step $i+1$ (which would be $\psi_{\alpha_i}^{(i)}$), which of course iterates down the following steps.

We already encountered similar behaviour in the proof of Theorem 21, which is the analogy to the following theorem for the error estimate. First we need a few definitions: Analogously to (8.2) we define

$$\sqrt{\sum_{\alpha_k=r_j+1}^{\infty} \left(\tilde{\sigma}_{\alpha_j}^{(j)}\right)^2} = \varepsilon_j, \tag{8.7}$$

where $\tilde{\sigma}_{\alpha_j}^{(j)}$ is the singular value computed in the j-th step of the continuous TT-format and is notably not the same as for the not truncated decomposition. And for an estimate independent of the singular values in question we choose an $\epsilon \in \mathbb{R}^+$ and set:

$$r_1 = \epsilon^{-\frac{1}{k}},\ r_2 = \epsilon^{-\frac{2}{k}}, \dots, r_{d-1} = \epsilon^{-\frac{d-1}{k}}. \tag{8.8}$$

Theorem 25. *For a function $f \in H^k(\Omega_1 \times \ldots \times \Omega_d)$ and a rank vector $\mathbf{r} \in \mathbb{N}^{d-1}$ we can formulate the error estimate*

$$\|f - f_{\mathbf{r}}^{TT}\|_{L^2(\Omega_1 \times \ldots \times \Omega_d)} \leq \sqrt{\sum_{j=1}^{d-1} \varepsilon_j^2}$$

where ε_j is defined as in (8.7).

Independently of the singular values we can formulate for a function $f \in H^{k+1}(\Omega_1 \times \ldots \times \Omega_d)$ and a desired accuracy ϵ :

$$\|f - f_{\mathbf{r}}^{TT}\|_{L^2(\Omega_1 \times \ldots \times \Omega_d)} \lesssim \sqrt{d}\epsilon,$$

where the ranks $\mathbf{r}$ *are given by* (8.8).

The proof for both can be found in [13, Section 4.2]. The first statement is also completely analogous to the proof of Theorem 21. The second statement in contrast needs a lot more analytical groundwork, which can be found in the above cited work in all its details.

In the two-dimensional case we now tried to relate the continuous case to the discretized one. Unfortunately for more than two dimensions this is not possible in the same way. This is due to the fact that, as described above, the error of the truncation propagates through the following terms. These accumulating errors make relating the two cases not practically possible.

9 The higher dimensional Combi-format

Having now defined the concept of a tensor and the TT-format, we can generalize the two-dimensional Combi-format to more dimensions.

For this we first need to consider the combination technique in more than two dimensions, which we will do in the following section. Then we can define the generalized Combi-format and consider its properties.

As already mentioned, formulating a convergence theory is more difficult in higher dimensions, a precise estimate even impossible due to the lack of an Eckart-Young-Mirsky theorem. Still we will formulate what is possible in section 9.3 and then briefly show convergence in praxis on one example.

9.1 Combination technique for higher Dimensions

As we wish to proceed as in the two-dimensional case, we need to define the combination technique for higher dimensions. The general idea remains the same; if we recall the visual representation in the coordinate system we are now considering more than two axes. This results in an outer sum over d-many terms.

For a function $u \in \mathcal{H} \subset L^2(\Omega)$ the d-dimensional general combination technique of level m (also compare to [22, Chapter 3.2]) is given by the formula

$$\mathcal{CT}_m u = \sum_{i=0}^{d-1} (-1)^i \binom{d-1}{i} \sum_{|\mathbf{l}|_1 = m+(d-1)-i} P_{\mathbf{l}} u. \tag{9.1}$$

If we compare the formula (9.1) for $d = 2$ to our first definition of the combination technique at (3.1), we see that the two definitions are equivalent.

It is important to remember the combination factor $\binom{d-1}{i}$, which is not necessary in the two-dimensional case.

Analogous to the two-dimensional case (recall (3.2)) we can rewrite

$$\mathcal{CT}_m u = \sum_{\mathbf{l} \in \mathcal{I}_m^C} (-1)^{m+(d-1)-|\mathbf{l}|_1} \binom{d-1}{m+(d-1)-|\mathbf{l}|_1} P_{\mathbf{l}} u \tag{9.2}$$

with the d-dimensional index-set

$$\mathcal{I}_m^C = \left\{ \mathbf{l} \in \mathbb{N}^d | m \leq |\mathbf{l}|_1 \leq m + (d-1) \right\}. \tag{9.3}$$

This is again compatible with the definition (3.3) of the two-dimensional case.

The results of section 3 generalize for this higher-dimensional setting. For storage cost and computational complexity we do need to adjust the number of summands (the size of the index-set $\mathcal{I}_m^C$).

9.2 Combi-format for higher Dimensions

Having done all the preparatory work, we can now define the Combi-format in its general form. As discussed we do this with the help of the TT-format. This also guarantees that the Combi-format for higher dimensions is compatible with the definition for two dimensions.

Analogously to (6.1) we can now define the Combi-format also for higher dimensions, using a low-rank tensor-approximation-scheme instead of the SVD. In general we will consider the TT-format as defined in Definition 18 and therefore the truncated TT-format operator $\Upsilon_{\mathbf{r}}$ as defined in Definition 20.

Therefore we get

$$\mathcal{CF}_{m,\mathbf{r}} u = \sum_{\mathbf{l} \in \mathcal{I}_m^C} (-1)^{m+(d-1)-|\mathbf{l}|_1} \binom{d-1}{m+(d-1)-|\mathbf{l}|_1} \Upsilon_{\mathbf{r}_{\mathbf{l}}} (P_{\mathbf{l}} u) \tag{9.4}$$

where $\mathbf{r} = (r_{\mathbf{l}})_{\mathbf{l} \in \mathcal{I}_m^C}$ is the matrix (vector of vectors) of truncation rank-vectors indexed by the grid it is applied to.

We will in general for each TT-format truncate the whole format with the same rank $r_{\mathbf{r}}$, so the vector of vectors that is the the index $\mathbf{r}$ of the Combi-format is usually only a vector. If all TT-formats are truncated in the same way, we replace this index by an integer r.

The Combi-format inherits its properties from the combination technique and the TT-format. Since the TT-format is based on the SVD, these properties are analogous to the two-dimensional case. Most importantly: the Combi-format is not linear.

What we can do is adding two Combi-formats of the same level, by adding the TT-formats of corresponding level-indices $\mathbf{l}$. Adding two tensors in the TT-format is explained in (8.4). For the sake of efficient computing a re-orthogonalization should be performed after an addition. Since adding two tensors (in the TT-format, by the described method) together results in a tensor having the sum of the ranks of the original tensors as the maximal rank, we might also want to truncate again. Strictly speaking we do not know the rank of the resulting tensor (only the maximum possible one) and can therefore not write down the resulting format in the same way as above. Truncating in the re-orthogonalizing by some chosen ranks would resolve this notational problem.

A point we do wish to highlight is the fact that using the TT-format in contrast to full tensors in the combination technique makes computations easier. We have briefly discussed this in section 8, when we considered the properties of the TT-format. Since computations are generally done on a core-by-core principle and these cores (themselves tensors) are much smaller than the full tensor, the computational complexity is much reduced. More details can be found in [35].

We have chosen the TT-format due to its beneficial properties and its compatibility with the SVD. An idea for further study is to consider other low-rank tensor-decompositions, for example the hierarchical tucker format (also known as HT-format) [11].

9.3 Convergence

In contrast to the two-dimensional case we cannot formulate a precise convergence theory. For more than two dimensions the problems we already had in two dimensions still persist, while we also get more new problems. This is due to one part of what is often also referred to as the Eckart-Young theorem (see for example [26, 37, 36]), which states that there is no such thing as the SVD in more than two dimensions. Or more generally: there is no method which produces a low-rank best-approximation in a least-squares sense for an arbitrary tensor $\mathbf{x}$.

We have already in a way considered this when we formulated Corollary 1. The only result we can provide is that our approximation is quasi-optimal. This does give us hope for the practical application; in the field of higher dimensional tensor formats and algorithms based on them there are many examples where practical computation outperforms the difficult to formulate theory (for example the AMEn-Algorithm for iterative solving of linear systems [6]).

We pursued multiple ideas for our convergence proof in the two dimensional case. The first idea, switching the order of operations, fails to generalize. We could generalize the difference of switching the order of operations to $\varUpsilon_{\mathbf{r}}^{\mathbf{l}} P_{\mathbf{l}} f - P_{\mathbf{l}} f_{\mathbf{r}}^{TT}$ where $f_{\mathbf{r}}^{TT}$ is defined as in Definition 24. However trying to bound this term would not work in even the limited fashion of the two-dimensional case.

All in all this leads to a similar situation as in the two dimensional case: the only estimate we can formulate is based on the triangle inequality and heavily dependent on the function in question.

More precisely formulated we use the same idea as (6.15)

$$\|f - \mathcal{C}\mathcal{F}_{m,\mathbf{r}}(f)\| \leq \|f - \mathcal{C}\mathcal{T}_m(f)\| + \|\mathcal{C}\mathcal{T}_m(f) - \mathcal{C}\mathcal{F}_{m,\mathbf{r}}(f)\|,$$

only this time over more than two dimensions.

The first summand is still the convergence of the combination technique, only now in higher dimensions. We have not found an explicit proof for higher dimensions in the literature, but in [17] it is said that the proof does generalize for higher dimensions, it only becomes a lot more work to write down and is notationally difficult. Also the constant does depend on the dimension.

The second summand is now a sum of differences between discretized functions and their approximation via the truncated TT-format. Here we can again refer to Corollary 1, which at least gives us quasi-optimality of this approximation.

If we want to actually bound the error of this approximation we need to consider Theorem 21. This results in an estimate

$$\begin{aligned} \|\mathcal{C}\mathcal{T}_m(f) - \mathcal{C}\mathcal{F}_{m,\mathbf{r}}(f)\| &\leq \sum_{\mathbf{l} \in \mathcal{I}_m^C} \left\| (\mathrm{Id} - \varUpsilon_{\mathbf{r}_\mathbf{l}}^{\mathbf{l}})(P_{\mathbf{l}} f) \right\| && (9.5) \\ &\leq \sum_{\mathbf{l} \in \mathcal{I}_m^C} \sqrt{\sum_{k=1}^{d-1} \left(\epsilon_{\mathbf{l}}^k\right)^2} && (9.6) \end{aligned}$$

where we have defined the $\epsilon_{\mathbf{l}}^k$ in accordance with Theorem 21 as

$$X_k^{\mathbf{l}} = R_k^{\mathbf{l}} + E_k^{\mathbf{l}}, \quad \operatorname{rank} R_k^{\mathbf{l}} = (r_{\mathbf{l}})_k, \quad \left\| E_k^{\mathbf{l}} \right\|_F = \epsilon_{\mathbf{l}}^k, \quad k = 1, \dots, d-1.$$

with $X_k^{\mathbf{l}}$ being the k-th unfolding of the tensor $\mathbf{X}_{\mathbf{l}} = P_{\mathbf{l}} f$.

This estimate now depends directly on the singular values of the TT-decomposition in each step. These cannot easily be bounded. While we do have Theorem 25, bounding the discretized singular values by the continuous ones is not analogous to the two-dimensional case. This is due to the fact that both the truncation and the discretization error propagate through the format. As such relating the discrete TT-format to the continuous one is not as easily done as for the SVD and requires further study.

Concluding we can say that while we have some results that give us hope that the Combi-format should show usable convergence in praxis, a precise convergence theory cannot be formulated (yet).

Choice of Truncation-ranks Due to the fact that this second term is the one that is determined by the chosen truncation ranks the question on how to choose these ranks is not answerable as in the two-dimensional case.

Theorem 21 and the above give us an idea on how to bound the error using the singular values of unfoldings of the tensor. Due to the already discussed propagation of the truncation error through the format, this is however not a precise bound. What remains is the quasi-optimality of Corollary 1, which gives us the factor $\sqrt{d-1}$ we have to consider additionally.

Generally, we proceed as in the two-dimensional case discussed in section 6.4. Instead of relatively straightforwardly considering one truncation, we need to consider all $d-1$ many. This means instead of adding up all the discarded singular values of all grids as in (6.24) in Theorem 16, we now need to consider (9.6) and the additional factor of quasi-optimality $\sqrt{d-1}$.

How to maybe optimize this has to be studied in future.

9.4 Praxis

In the following we apply the Combi-format for higher dimensions in praxis. As shown above with the rise of dimension the computational complexity also gets higher. We did not in the scope of this work employ a scientific computer cluster to solve our approximation problem and were therefore limited by practicability. As such we do not do any in depth analysis of difficult functions, but rather just show the most important fact to show: Convergence.

This is our proof of concept: the Combi-format converges in higher dimensions as well, and provides a reduction in computational complexity.

This is the function

$$f(x, y, z) = \exp(\pi \cdot x \cdot y \cdot z \cdot (1-x) \cdot (1-y) \cdot (1-z)) - 1 \tag{9.7}$$

defined on $[0,1]^3$. Since $\exp(0) = 1$ this function is indeed zero on the boundary and therefore fulfils the requirements ($f \in L^2\left([0,1]^3\right)$ rather trivially).

Due to the outer exponential function we can expect a very high smoothness, but due to the polynomial exactness of our space still only being two in each direction (recall Theorem 12 and the convergence considerations for

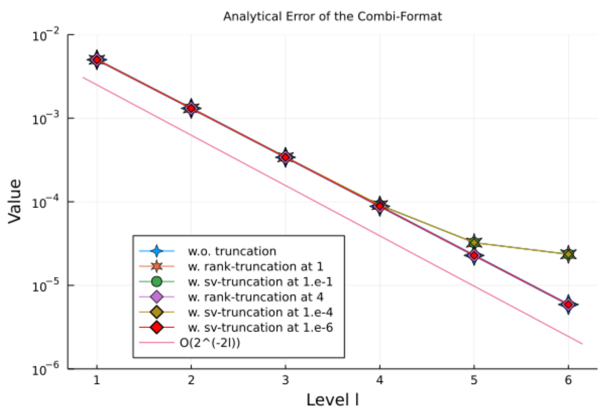


Figure 17: The convergence of the function (9.7) approximated in the Combi-format with different truncations and measured in the analytical error.

the rank-8 function in section 7.3) we can only expect a convergence rate of $\mathcal{O}(2^{-2l}) = \mathcal{O}(h^2)$ for level l. This rate is indeed visible in Figure 17.

Also visible is the convergence behaviour of the Combi-format for different truncations. All truncations are to be understood as being the same throughout the format; when we rank-truncate after rank 4, we truncate for all tensors in the format uniform: all TT-ranks to 4. In this case, for the maximum level of the format we consider, the Combi-format shows the same convergence as the combination technique (as visible in Figure 17). The same holds for singular-value-truncation at 10^{-6}.

At such low levels as we are considering for ease of calculations (maximum level 6 is not a lot for more advanced computers) the full potential of the Combi-format is not realized, but even for such low levels we do see a reduction in size of the considered tensors. Especially in higher dimensions this does have an exponential effect.

Other Environments The code in this work was written in Julia [1], with help of the ITensor [8] module for the tensors and the TT-format. It can of course be transferred into other languages, using other libraries.

The operations that need to exist in the wanted environment are higher-dimensional integration, basic matrix calculations (including inverting) the SVD and/or some way of calculating and storing the TT-format directly. In the following we give a not at all exhaustive overview over different possible programming environments.

The ITensor module exists also for C++; there are also multiple libraries for higher dimensional integration in C++. There also exist a TT-toolbox for MATLAB just called TT-Toolbox [33] and one for Python, called ttpy

[34]. Both are written and maintained by Ivan Oseledets. For both languages libraries for higher dimensional integration also exist.

Finally there is the t3f [32] library for working with tensors in the TT-format in TensorFlow (which is build in Python). The documentation on this library also has a site comparing different libraries for implementing the TT-format, including the above named.

All in all we can see that although the TT-format is not a standard in most programming environments, there are nonetheless multiple options for implementing our code in other environments. Since our code is not highly complex we can assume that porting the code into a different language is not very difficult. As such our Combi-format can with a little effort be used in projects that might for example require a different programming language than Julia.

10 Conclusion

In this work we have introduced the Combi-format, a method for solving operator equations like function approximation or differential equations. It is based on the combination technique for sparse grids and the singular value decomposition and aims to improve the former with use of the latter. Furthermore, to achieve usability for higher dimensions the tensor train format is utilized.

We have succeeded in the goal to improve the combination technique. If we choose the truncation in a balanced way the Combi-format (for two dimensions) achieves the same order of convergence as the combination technique. This is the same order of convergence as straightforward approximation onto the sparse grid. We have also shown this convergence order in praxis on a few examples.

While the cost of computing the format is slightly bigger than computing the combination technique due to having to perform additional singular value decompositions, the Combi-format generally has a reduced number of variables. Therefore the storage cost of the format is less and further computations we may wish to perform with it have less computation cost.

In higher dimensions we have at least shown that achieving the same order of convergence as the combination technique is possible, even if we cannot formulate an efficient algorithm to compute the truncation ranks needed for it. Also the use of the TT-format simplifies calculations in higher dimensions.

Unfortunately the proof for the order of convergence is not optimal, maybe a better analysis can be found. As it is the general rate of convergence is not meaningful for choosing truncation ranks, and the more specific version is not computable a-priori. But since almost all values we need to choose the truncation ranks in a useful matter need to be computed anyway we have formulated an algorithm to do this. In future this might actually be formulated to be done by a computer and not a mathematicians judgement. Additional work may also be done to adapt this format to using further optimizations like dimensionally-adaptive sparse grid methods.

Concluding this work, I have defined the new Combi-format which can justify its higher computation costs by a reduction in storage space and easier computations while still achieving a competitive order of convergence.

A Documentation

The following appendix contains a documentation of the code accompanying this work. The code is written in the Julia programming language [1]. Also used are the modules (basically what libraries are in C-like languages) `HCubature` [27] for the multi-dimensional integration and `ITensor` [8] for the tensors.

A.1 Modules

We define the code for this work within a module, fittingly called `Combi`.

Combi - Module

```
module Combi

    export SubspaceD_ID, f, loop_Btensor,
        loop_CF_error_nl, loop_CF_error_sv_nl,
        loop_CF_error_cut_nl, parallel_svv
    using DelimitedFiles, LinearAlgebra,
        ITensors, HCubature, ITensors.HDF5
```

The module containing all needed functions for the Combi-format.

A.2 Types

Types are similar to the concept of a class in object-oriented programming. The following `struct` is used to collect all necessary data for each sub-grid in the Combi-format.

SubspaceD_ID - Type

```
struct SubspaceD_ID
    dim::Integer
    level::Vector
    str_id::String
    idx_nr::Integer
    tol::Float64
    cutoff::Integer
    index::Vector
end
```

A struct that contains all identifying information for a sub-grid in the Combi-format.

Arguments

- `dim::Integer` : The dimension of the sub-grid (the number of directions)
- `level::Vector` : The level of the grid in all directions
- `str_id::String` : Identifying string used for file in- and output (generally the same for the whole format)
- `idx_nr::Integer` : Index of the sub-grid in the Combi-format
- `tol::Float64` : If singular-value-truncation is used, this is the tolerance
- `cutoff::Integer` : If rank-truncation is used, this is the max. rank
- `index::Vector` : These are the indices of the sub-grid needed for the module "ITensor"

A.3 Constants

These are strings used whenever files need to be accessed. They are assigned the directory of the folders in which these files are stored. When first building the module Combi these strings need to be changed in the source file `Helpfct.jl`.

Combi.bfolder - Constant
: Directory of the folder for the results of `loop_Btensor`

Combi.ifolder - Constant
: Directory of the folder for the results of `fint`

Combi.cfolder - Constant
: Directory of the folder for the results of `loop_CF_error_nl`, `loop_CF_error_cut_nl`, `loop_CF_error_sv_nl`

Combi.sfolder - Constant
: Directory of the folder for the results of `parallel_svv`

A.4 Functions

Finally we define the functions (also called methods) for the Combi-format and the calculation of the approximation error as described in section 7.1.

Exported: We export the following functions.

Combi.f - Function

```
f(x)
```

The function we wish to approximate on the unit cube. `x` has to be of an appropriate type to be numerically integrated.

Combi.loop_Btensor - Function

```
loop_Btensor(a::Integer, b::Integer,
     str_id::String, dim::Integer,
     tol::Float64=1.e-16)
```

Loops `ind_Btensor` of dimension `dim` from level `a` to `b` (both inclusive) and writes results to file identified by `str_id`. Also executes `fint`.

Combi.loop_CF_error_nl - Function

```
loop_CF_error_nl(a::Integer, b::Integer,
     str_id::String, dim::Integer, str_out::String)
```

Loops `parallel_CF_error_nl` of dimension `dim` from level `a` to `b` (both inclusive) and writes results to file identified by `str_out`. This is the combination technique, not truncated.

Combi.loop_CF_error_sv_nl - Function

```
loop_CF_error_sv_nl(a::Integer, b::Integer,
     str_id::String, dim::Integer, str_out::String,
     tol::Float64)
```

Loops `parallel_CF_error_sv_nl` of dimension `dim` from level `a` to `b` (both inclusive) and writes results to file identified by `str_out`. This uses singular-value-truncation with tolerance `tol`.

Combi.loop_CF_error_cut_nl - Function

```
loop_CF_error_cut_nl(a::Integer, b::Integer,
     str_id::String, dim::Integer, str_out::String,
     cut::Integer)
```

Loops `parallel_CF_error_cut_nl` of dimension `dim` from level `a` to `b` (both inclusive) and writes results to file identified by `str_out`. This uses rank-truncation with max rank `cut`.

Combi.parallel_svv - Function

```
parallel_svv(flevel::Integer, str_id::String,
    str_out::String)
```

Calculates the singular values for all sub-grids of level `flevel` using `calc_sv`. Uses function values from the files identified by `str_id` and writes to a file identified by `str_out`.

Implementing the Combi-format: The above functions call on the following ones to actually implement the Combi-format (and perform the approximation error calculation). All functions starting with "parallel_CF" implement the sum of the Combi-format by calling, in parallel, the associated functions starting with "ind_subspace". These functions perform the necessary calculations on a particular sub-grid, creating the necessary `SubspaceD_ID`.

parallel_CF_error_nl - Function

Calls `ind_subspace_error_nl` in parallel and calculates the error for the whole format of level `flevel`, dimension `dim` and identifier `str_id`. This is the combination technique, not truncated. This is not linear!

parallel_CF_error_sv_nl - Function

Calls `ind_subspace_error_sv_nl` in parallel and calculates the error for the whole format of level `flevel`, dimension `dim` and identifier `str_id`. This uses singular-value-truncation with tolerance `tol`. This is not linear!

parallel_CF_error_cut_nl - Function

Calls `ind_subspace_error_cut_nl` in parallel and calculates the error for the whole format of level `flevel`, dimension `dim` and identifier `str_id`. This uses rank-truncation with max rank `cut`. This is not linear!

ind_subspace_error_nl - Function

Calculates the error on an individual sub-grid characterized by the arguments.

This creates a `SubspaceD_ID` and then calls `ind_norm` for the whole Format. This is the combination technique, not truncated. This is not linear!

ind_subspace_error_sv_nl - Function

Calculates the error on an individual sub-grid characterized by the arguments.

This creates a `SubspaceD_ID` and then calls `ind_norm_sv` for the whole Format. This uses singular-value-truncation with tolerance `tol`. This is not linear!

ind_subspace_error_cut_nl - Function

Calculates the error on an individual sub-grid characterized by the arguments.

This creates a `SubspaceD_ID` and then calls `ind_norm_cut` for the whole format. This uses rank-truncation with max rank `cut`. This is not linear!

Basics: There are also the functions returning and/or calculating the basic components of the format.

b - Function

Basic hat function of level 1.

Btensor - Function

Returns the right-hand-side tensor for projection onto the subspace defined by `Ssp`.

write_Btensor - Function

Loads the right-hand-side tensor (result of `Btensor`) from file.

load_Btensor - Function

Writes the right-hand-side tensor (result of `Btensor`) to a file.

Calls `Btensor`.

fint - Function

Writes the result of `fnorm` for `f` and with tolerance `tol` in file identified by `str_id`.

fnorm - Function

Computes the L^2-norm of f with tolerance tol over unit cube of dimension dim.

Mtensor - Function

-Returns the mass-matrix of mixed spaces as a tensor in direction `ind` for the sub-grid defined by `Ssp` and `Sspt`.

-Returns the mass-matrix as a tensor in direction `ind` for the sub-grid defined by `Ssp`.

Mtensor_inv - Function

> Returns the inverted mass-matrix as a tensor in direction `ind` for the subspace defined by `Ssp`.

Xtensor - Function

> Returns the grid-tensor of the subspace defined by `Ssp`.

Xttensor - Function

> Returns the grid-tensor of the subspace defined by `Ssp` in TT-Format, without explicit truncation.

Xttensor_sv - Function

> Returns the grid-tensor of the subspace defined by `Ssp` in TT-Format, with singular-value-truncation

Xttensor_cut - Function

> Returns the grid-tensor of the subspace defined by `Ssp` in TT-Format, with rank-truncation.

calc_sv - Function

> Calculates and returns the normed and squared singular values of the sub-grid identified by `Ssp`. This is ONLY for 2D.

Basics of error calculation: Much the same as above, the following functions calculate parts of the analytical error on the subspaces.

calc_error_m - Function

> Calculates the local error of the subspace defined by `Ssp`.

calc_error_sv_m - Function

> Calculates the local mixed error of the subspace defined by `Ssp`, with singular-value-truncation.

calc_error_cut_m - Function

> Calculates the local mixed error of the subspace defined by `Ssp`, with rank-truncation.

ind_norm - Function

> Calculates the scalar product $X \cdot M \cdot X_2^T$ for X defined by `Ssp` and X_2 by `Sspt` without truncation.

ind_norm_sv - Function

> Calculates the scalar product $X \cdot M \cdot X_2^T$ for X defined by `Ssp` and X_2 by `Sspt` with singular-value-truncation.

ind_norm_cut - Function

> Calculates the scalar product $X \cdot M \cdot X_2^T$ for X defined by `Ssp` and X_2 by `Sspt` with rank-truncation.

Help-functions: The following functions are routines which perform some necessary calculations that are used in the other functions.

There is `ctrafo`, which performs a coordinate transform; `level_order` returns the level of a basis function; `nlev` iterates over level-indices; `smm`, `total_smm` and `it_smm` calculate the number of level-indices of certain parts of the combination technique.

More details can be found in the code.

References

[1] Jeff Bezanson et al. “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM Review* (2017).

[2] Bastian Bohn. “Error analysis of regularized and unregularized least-squares regression on discretized function spaces”. Dissertation. Institut für Numerische Simulation, Universität Bonn, 2017.

[3] Dietrich Braess. *Finite Elemente: Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. 5th ed. Springer Spektrum, 2013. DOI: 10.1017/978-3-642-34797-9.

[4] Haim Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. 2011.

[5] Jacob C Bridgeman and Christopher T Chubb. “Hand-waving and interpretive dance: an introductory course on tensor networks”. In: *Journal of Physics A: Mathematical and Theoretical* 50.22 (May 2017). ISSN: 1751-8121. DOI: 10.1088/1751-8121/aa6dc3. URL: http://dx.doi.org/10.1088/1751-8121/aa6dc3.

[6] Sergey V Dolgov and Dmitry V Savostyanov. “Alternating minimal energy methods for linear systems in higher dimensions”. In: *SIAM Journal on Scientific Computing* 36.5 (2014), A2248–A2271.

[7] Christian Feuersänger. “Sparse Grid Methods for Higher Dimensional Approximation”. Dissertation. Mathematische-Naturwissenschaftliche Fakultät der Universität Bonn, 2010.

[8] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. *The ITensor Software Library for Tensor Network Calculations*. 2020. arXiv: 2007.14822.

[9] J. Garcke. “Sparse Grids in a Nutshell”. In: *Sparse grids and applications*. Ed. by J. Garcke and M. Griebel. Vol. 88. Lecture Notes in Computational Science and Engineering. Springer, 2013, pp. 57–80. DOI: 10.1007/978-3-642-31703-3_3.

[10] Thomas Gerstner and Michael Griebel. “Numerical Integration using Sparse Grids”. In: *Numer. Algorithms* 18 (1998). (also as SFB 256 preprint 553, Univ. Bonn, 1998), pp. 209–232.

[11] Lars Grasedyck and Wolfgang Hackbusch. “An Introduction to Hierarchical (H-) Rank and TT-Rank of Tensors with Examples”. In: *Computational Methods in Applied Mathematics* (2011).

[12] Michael Griebel and Thomas Gerstner. “Sparse Grids”. In: *Encyclopedia of Quantitative Finance* (2008).

[13] Michael Griebel and Helmut Harbrecht. “Analysis of tensor approximation schemes for continuous functions”. Available as INS Preprint No. 1903. Submitted to Foundations of Computational Mathematics. 2019.

[14] Michael Griebel and Helmut Harbrecht. “Approximation of two-variate functions: Singular value decomposition versus sparse grids”. In: *IMA Journal of Numerical Analysis* 34.1 (2014). Also available as INS Preprint No. 1109, pp. 28–54. DOI: `10.1093/imanum/drs047`.

[15] Michael Griebel and Helmut Harbrecht. “Low-rank approximation in Sobolev spaces with dominating mixed smoothness”. Unpublished manuskript provided directly by the authors. With grateful thanks to the authors. 2020.

[16] Michael Griebel and Helmut Harbrecht. “On The Construction Of Sparse Tensor Product Spaces”. In: *Mathematics of Computations* 82.282 (Apr. 2013), pp. 975–994.

[17] Michael Griebel and Helmut Harbrecht. “On the convergence of the combination technique”. In: *Sparse grids and Applications.* Vol. 97. Lecture Notes in Computational Science and Engineering. Springer, 2014, pp. 55–74.

[18] Michael Griebel and Helmut Harbrecht. “Singular value decomposition versus sparse grids: Refined complexity estimates”. In: *IMA Journal of Numerical Analysis* 39.4 (2019), pp. 1652–1671. DOI: `10.1093/imanum/dry039`.

[19] Michael Griebel and Stephan Knapek. *Optimized approximation spaces for operator equations.* Technical Report 568. SFB 256, Universität Bonn, 1999.

[20] Michael Griebel and Stephan Knapek. “Optimized tensor-product approximation spaces”. In: *Constructive Approximation* 16.4 (2000), pp. 525–540.

[21] Michael Griebel and Guanglian Li. “On the decay rate of singular values of bivariate functions”. In: *SIAM Journal on Numeriacal Analysis* (2018). Also available as INS Preprint No. 1702.

[22] Michael Griebel and Alexander Rüttgers. “Multiscale simulation of polymeric fluids using the sparse grid combination technique”. In: *Applied Mathematics and Computation* 319 (2018), pp. 425–443.

[23] Michael Griebel, Michael Schneider, and Christoph Zenger. “A combination technique for the solution of sparse grid problems”. In: *Iterative Methods in Linear Algebra* (1992). Ed. by P. de Groen and R. Beauwens, pp. 263–281.

[24] Wolfgang Hackbusch. *Tensor Spaces and Numerical Tensor Calculus.* 2012.

[25] Markus Holtz. "Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance". Dissertation. 2008.

[26] Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. "The Alternating Linear Scheme for Tensor Optimization in the Tensor Train Format". In: *SIAM J. Sci. Comput.* 34 (2012).

[27] *JuliaMath/HCubature.jl.* URL: `https://github.com/JuliaMath/HCubature.jl` (visited on 11/24/2021).

[28] Vladimir Kazeev and Boris Khoromskij. "Low-Rank Explicit QTT Representation of the Laplace Operator and Its Inverse". In: *SIAM Journal on Matrix Analysis and Applications* 33 (3 July 2012), pp. 742–758. DOI: `10.1137/100820479`.

[29] Stephan Knapek. "Approximation und Kompression mit Tensorprodukt-Multiskalenräumen". Dissertation. Universität Bonn, Apr. 2000.

[30] Antonia Krendelsberger. "Niedrigrang-Tensormethoden zum Lösen linearer Systeme: Das TT-Format und der AMEn-Algorithmus". Bachelorthesis. Institut für Numerische Simulation, Universität Bonn, 2018.

[31] Michael Griebel und Markus Holtz. "Dimension-wise Integration of High-dimensional Functions with Apllications to Finance". In: *Journal of Complexity* 26 (2010), pp. 455–489.

[32] Alexander Novikov et al. "Tensor Train Decomposition on TensorFlow (T3F)". In: *Journal of Machine Learning Research* 21.30 (2020), pp. 1–7. URL: `http://jmlr.org/papers/v21/18-008.html`.

[33] Ivan Oseledets. *TT-Toolbox.* URL: `www.github.com/oseledets/TT-Toolbox` (visited on 12/01/2021).

[34] Ivan Oseledets. *ttpy.* URL: `www.github.com/oseledets/ttpy` (visited on 12/01/2021).

[35] Ivan V. Oseledets. "Tensor-train decomposition". In: *SIAM J. SCI. COMPUT.* 33.5 (2011), pp. 2295–2317.

[36] Erhard Schmidt. "Zur Theorie der linearen und nichtlinearen Integralgleichungen". In: *Math. Ann.* (1907).

[37] Gilbert W. Stewart. "On the Early History of the Singular Value Decomposition". In: *SIAM Review* (1993).

[38] Christoph Zenger and Wolfgang Hackbusch. "Sparse grids". In: *Proceedings of the Research Workshop of the Israel Science Foundation on Multiscale Phenomenon, Modelling and Computation.* 1991, p. 86.