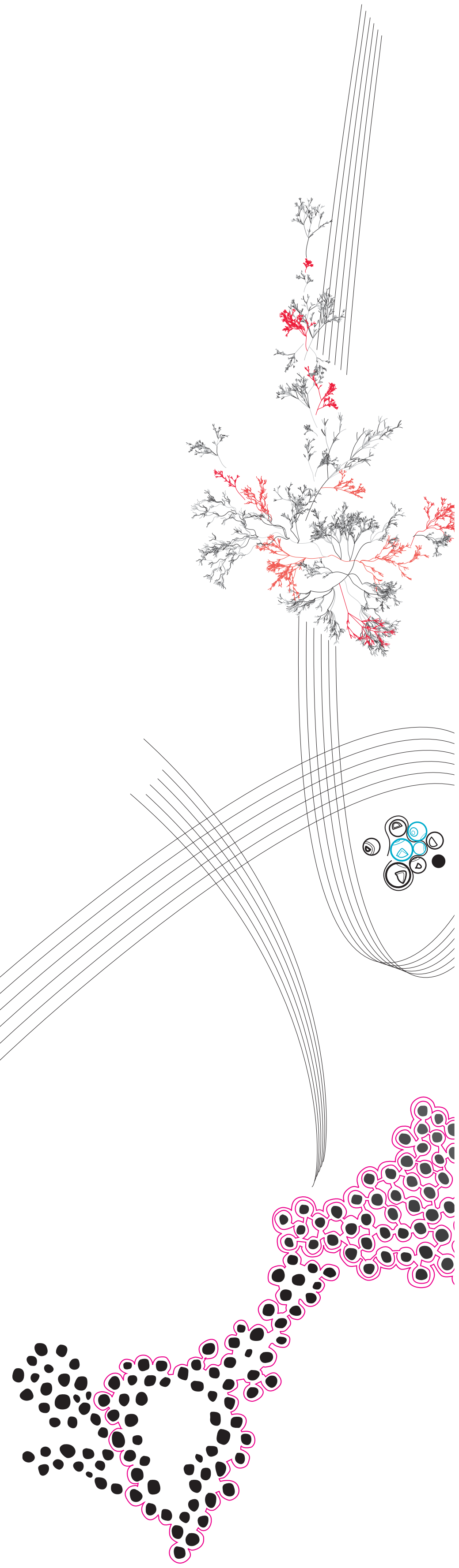
BSc Thesis Applied Mathematics



Numerical Investigation of Bounds for Model Reduction

Hessel Stokman

Supervisors: dr. S.M. Glas, dr. T. van Dijk

July, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Abstract

In Model Order Reduction, the goal is to estimate solutions to high dimensional models of physical systems using a lower dimension reduced order model (ROM) that is faster to compute. A common approach is to build ROMs on a linear subspace of the solution space of the high dimensional model. For ROMs on n-dimensional linear subspaces, the lowest achievable approximation error is given by the Kolmogorov n-width. ROMs on linear subspaces can be extended to ROMs on polynomially mapped manifolds. There is an analogue to the Kolmogorov n-width for ROMs on polynomially mapped manifolds of degree p, called the polynomial Kolmogorov (n, p)-width. In most cases, it is not possible to compute these widths exactly. We propose two methods for estimating both the Kolmogorov n-width and polynomial Kolmogorov $(n, 2)$-width and compare their performance on an example setting. A theoretical approximation bound (lower and upper) can be given for the polynomial Kolmogorov (n, p)-width, formulated in terms of the Kolmogorov n-width. Using our estimation methods for the two widths, we investigate this approximation bound numerically on an example setting.

Contents

1 Introduction

Numerical simulations are often needed to approximate complex physical systems, that are difficult to find exact solutions to. To approximate these systems accurately, the resulting discrete models are possibly of high dimension. These models often have parametric dependencies, requiring separate evaluations for each parameter. For high dimensional models, these repeated evaluations can be computationally expensive and time-consuming. In *model order reduction (MOR)*, the goal is to speed up the computation of the high dimensional model, also known as the *full order model (FOM)*. This is done by building a *reduced order model (ROM)* of a reduced dimension, which is significantly lower than the dimension of the FOM, that approximates the FOM. Instead of repeatedly evaluating the FOM, the ROM is evaluated for an approximate solution.

In classical MOR, a ROM of reduced dimension n, is constructed on an n-dimensional linear subspace of the solution space of the FOM. The approximation quality of such a ROM can be bounded below by the Kolmogorov n-width, which gives the lowest achievable approximation error for ROMs on an n-dimensional linear subspace. In most cases, it is not clear how to construct a ROM that achieves the approximation error given by the Kolmogorov n-width.

A ROM on a linear subspace may not be able to provide satisfactory approximations for problems where the Kolmogorov n-width decays slowly. Other types of ROMs can potentially address this issue. One such class of ROMs are ROMs on polynomially mapped manifolds, which extend ROMs on linear subspaces. In [1], an analogue to the Kolmogorov n-width for ROMs on polynomially mapped manifolds is defined, called the polynomial Kolmogorov (n, p)-width. Additionally, [1] gives approximation bounds for the polynomial Kolmogorov (n, p)-width, in terms of the Kolmogorov n-width. These approximation bounds give a range in which the lowest achievable error by ROMs on polynomially mapped manifolds lie.

In most cases, it is not possible to compute the Kolmogorov n-width and polynomial Kolmogorov (n, p)-width exactly. In this thesis, we aim to numerically investigate the bounds given for the polynomial Kolmogorov (n, p)-width. To this end, we propose two methods for estimating both the Kolmogorov n-width and polynomial Kolmogorov (n, p)-width. We give a comparison of these methods on an example setting. Finally, we use them, to numerically investigate how the polynomial Kolmogorov $(n, 2)$-width behaves within the bounds, on the same example setting.

Our numerical investigation focuses on determining where the polynomial Kolmogorov (n, p)-width lies between the estimations of the theoretical upper and lower bounds, which is not immediately apparent from the theoretical bounds alone. This positioning between the bounds is significant, because it is an indication of the lowest achievable approximation error for ROMs on polynomially mapped manifolds. If our numerical estimates show that the polynomial Kolmogorov (n, p)-width is close to the lower bound, it suggests a lower achievable approximation error, indicating potentially more accurate approximations using ROMs on polynomially mapped manifolds. Conversely, if it is closer to the upper bound, this might indicate a higher lowest achievable error. It is important to note that this numerical investigation is restricted to the methods and test values chosen and does not aim to make any absolute claims about these type of ROMs.

2 Bounds for model reduction

In this section, we give the necessary background to define the Kolmogorov n-width and polynomial Kolmogorov (n, p)-width. These bound the theoretically lowest achievable approximation error for two classes of ROMs.

We begin by defining classical ROMs on linear subspaces. We then introduce the Kolmogorov-n-width to measure the quality of the best possible approximation of this type of ROM. These ROMs on linear subspaces can be extended to ROMs on polynomially mapped manifolds. For this type of ROM there is an analogue to the Kolmogorov n-width called the polynomial Kolmogorov (n, p)-width. Finally, we give an upper- and lower bound for the polynomial Kolmogorov (n, p)-width in terms of the Kolmogorov n-width.

2.1 Reduced order models on linear subspaces

Let $(V, \|\cdot\|_V)$ be an N-dimensional normed vector space over $\mathbb{R}$ with $N \in \mathbb{N}$.

In model order reduction, an element $x \in V$ from the high dimensional space, is mapped to an element $x_n \in \mathbb{R}^n$ called the reduced coordinates of x, that are contained in a low n-dimensional space with $n \ll N$. This x_n is then mapped back to V by some mapping $\Gamma : \mathbb{R}^n \to V$ to obtain an approximation $\tilde{x} \in V$ of x.

The way in which elements of V are mapped to the low n-dimensional subspace $\mathbb{R}^n$, and mapped back to V, depends on the type of ROM. In this section, we consider ROMs on linear subspaces. This type of ROM is based on a low n-dimensional subspace $V_n \subset V$.

Let $\bar{V}_n \in \mathbb{R}^{N\times n}$ be an orthonormal basis matrix of V_n. An element $x \in V$ with reduced coordinates x_n is approximated by a linear combination of basis vectors in $\bar{V}_n$ with

$$\tilde{x} := \bar{V}_n x_n. \tag{1}$$

When the reduced coordinates x_n are chosen as the projection of x onto V_n, with $x_n = \bar{V}_n^\top x$, then the $\tilde{x}$ in (1) is the best possible approximation of x from a linear combination of the basis vectors of V_n ($\tilde{x}$ minimizes $\inf_{v\in V_n} ||x-v||_V$). The reduced coordinates are the scalars of this linear combination.

This optimality of this $\tilde{x}$ can be shown using the Projection Theorem (see Theorem 1) with

$$\inf_{v\in V_n} ||x - v||_V \overset{\substack{\text{Projection}\\(\text{ Theorem })}}{=} ||x - \bar{V}_n\bar{V}_n^\top x||_V = ||x - \bar{V}_n x_n||_V = ||x - \tilde{x}||_V.$$

Theorem 1. ***(Projection Theorem [12])*** *Let $\bar{Y}$ be an orthonormal basis matrix for the subspace Y of the inner product space H. For any $x \in H$, the vector*

$$y_0 = \bar{Y}\bar{Y}^\top x$$

is the unique vector that minimizes $\|x - y\|$ for all $y \in Y$.

We can measure the quality of the linear subspace V_n in approximating a set $S \subset V$ with the worst best-approximation error.

Definition 1. ***(Worst Best-approximation Error [1]).***
Let $(V, \|\cdot\|_V)$ be a normed vector space. For two sets $S, U \subseteq V$, we call

$$dist(S,U) := \sup_{s\in S}\inf_{u\in U}\|s-u\|_V \tag{2}$$

the worst best-approximation error of S in U.

The worst best-approximation error $dist(V_n, U)$ of V_n in $U \subset V$ measures the quality of our approximation of U with the subspace $V_n \subseteq V$. In this thesis, we are interested in how good an approximation can be at best for any n-dimensional linear subspace. This leads us to the Kolmogorov n-width, which measures how good an n-dimensional linear subspace of V is able to perform at best in approximating a set $U \subset V$.

Definition 2. ***(Kolmogorov n-width [10]).*** *Let $(V, \|\cdot\|_V)$ be a normed vector space. For a set $S \subset V$,*

$$d_n(S,V) := \inf_{\substack{V_n\subseteq V\\ dim(V_n)\le n}} dist(S,V_n)$$

is defined as the Kolmogorov-n-width of S in V.

Definition 2, defines the Kolmogorov n-width of S in V as the worst-best approximation error of the "best" (minimal worst best-approximation error) n-dimensional linear subspace V_n of V.

2.2 Reduced order models on polynomially mapped manifolds

For ROMs on linear subspaces, our approximation in (1) only depends linearly on the reduced coordinates. A possibly lower approximation error can be achieved by adding terms that depend differently on the reduced coordinates. We can add higher order terms by combining the elements of the reduced coordinates $x_n := ((x_n)_i)_{i=1}^n$ with the symmetric Kronecker product $K_s(x_n, k)$ of order k, which contains all unique combinations of k elements. It is defined as

$$K_s(x_n,k) := [(x_n)_{i_1}(x_n)_{i_2}...(x_n)_{i_k}|1\le i_1\le i_2\le ...\le i_k\le n] \quad x_n = ((x_n)_i)_{i=1}^n \in \mathbb{R}^n,\ n\in\mathbb{R},\ k\in\mathbb{N}.$$

As an example, the quadratic term, $K_s(x_n, 2)$ consists of all combinations of 2 elements $\{(x_n)_i(x_n)_j|1 \le i \le j \le n\}$.

In ROMs on polynomially mapped manifolds, the reduced coordinates x_n are mapped to the solution space of the FOM through a polynomial mapping with (3), instead of a linear transformation of the reduced coordinates like in (1). Elements $x \in V$ are approximated using higher order terms from the Kronecker product $K_s(x_n, k)$ of x_n in (3), in addition to the linear transformation of the reduced coordinates x_n used in (1).

Let $(V, \|\cdot\|_V)$ be an N-dimensional normed vector space over $\mathbb{R}$ with $N \in \mathbb{N}$. For a polynomial degree $p \in \mathbb{N}$, we estimate $x \in V$ by mapping the reduced coordinates $x_n \in \mathbb{R}^n$ to V with the polynomial map $\Gamma_{n,p} : \mathbb{R}^n \to V$. We obtain an approximation $\tilde{x} \in V$ with

$$\tilde{x} := \Gamma_{n,p}(x_n) \quad \Gamma_{n,p}(x_n) := \sum_{k=0}^{p} A_k K_s(x_n,k) \quad A_k \in R^{N\times|K_s(x_n,k)|} \tag{3}$$

with A_k being called mapping matrices. The quantity $|K_s(x_n, k)|$ is the number of combinations of k elements with replacement $\binom{n+k-1}{n}$. The image of $\Gamma_{n,p}$ is an n-dimensional submanifold $\widetilde{\mathcal{M}}_{n,p} \subseteq V$ that we call *a polynomially mapped submanifold* [1].

There is an analogous measure to the Kolmogorov n-width for measuring the lowest achievable approximation error by ROMs based on polynomially mapped manifolds, called the polynomial Kolmogorov (n, p)-width.

Definition 3 (Polynomial Kolmogorov $(\mathbf{n}, \mathbf{p})$-width [1]). *Let $(V, \|\cdot\|_V)$ be a finite dimensional normed vector space. For a set $S \subseteq V$,*

$$d^{\otimes}_{n,p}(S;V) := \inf_{\substack{\widetilde{\mathcal{M}}_{l,p},\ \textit{poly. mapped submnf.} \\ \dim(\widetilde{\mathcal{M}}_{l,p}) \leq n}} \textit{dist}(\widetilde{\mathcal{M}}_{l,p}, S)$$

is defined as the polynomial Kolmogorov (n, p)-width of S in V.

The polynomial Kolmogorov (n, p)-width of S in V gives a bound for how good an n-dimensional polynomially mapped submanifold of degree p can be at best in approximating a set $S \subseteq V$. It's the worst best-approximation error of the "best" (minimal worst best-approximation error) n-dimensional polynomially mapped submanifold of degree p.

The polynomial Kolmogorov (n, p)-width can be bounded from below and above by the Kolmogorov n-width of dimension n and $t(n, p) \gg n$ respectively, with Theorem 2.

Theorem 2. ***(Approximation Bounds for polynomial Kolmogorov $(\mathbf{n}, \mathbf{p})$-width [1]).*** *Let $(V, \|\cdot\|_V)$ be a normed vector space. For $S \subseteq V$ and $p \geq 1$,*

$$d_{t(n,p)}(S, V) \leq d^{\otimes}_{n,p}(S, V) \leq d_n(S, V) \quad t(n, p) = \sum_{k=0}^{p} \binom{n + k - 1}{n}.$$

The mapping matrices A_k in (3) total $t(n, p)$ columns. The map $\Gamma_{n,p}$ maps an element through a linear combination of these columns, but with a restricted set of scalars. By "restricted" it is meant here that, after choosing the scalars x_n in the linear term $A_1 x_n$ from (3), the other scalars $K_s(x_n, k)$ for $k > 1$ cannot be chosen freely any more.

In Section 4.2.2 we investigate these bounds numerically for an example case and study how $d^{\otimes}_{n,2}$ behaves within the bounds of Theorem 2. For this, we need to be able to estimate the Kolmogorov n-width and polynomial Kolmogorov $(n, 2)$-width. We present methods for this in the next section.

3 Numerically estimating error bounds

The bounds in Theorem 2 give the theoretically lowest achievable approximation error by ROMs on n-dimensional linear subspaces and polynomially mapped manifolds. Although relatively simple to describe, they are not that straightforward to compute. The nested infimums and supremum, from Definitions 2 and 3, combined with (2), cause them to be hard to compute exactly. In this section, we describe methods to estimate both the Kolmogorov n-width and the polynomial Kolmogorov (n, p)-width.

For this section, we let $(V, \langle\cdot,\cdot\rangle)$ be an N-dimensional vector space over $\mathbb{R}$, where $N \in \mathbb{N}$. The inner product $\langle\cdot,\cdot\rangle$ induces the norm $\|\cdot\|_V$. Additionally, let $S \subseteq V$ be a finite subset of V, that is, $|S| < \infty$. In Section 3.1, we give two methods for computing the Kolmogorov n-width $d_n(S, V)$ and in Section 3.2 we give two methods for computing the polynomial Kolmogorov (n, p)-width $d^{\otimes}_{n,p}(S, V)$.

3.1 Kolmogorov n-width

In this section, we present two methods for estimating the Kolmogorov n-widths. The first method, in Section 3.1.1, randomly samples bases for an n-dimensional linear subspace of V. The second method, in Section 3.1.2, uses a well known method to build a linear subspace by taking snapshots of S and minimizing the sum of least squares approximation error of these snapshots.

3.1.1 Randomly sampling bases

In Definition 2, the Kolmogorov n-width is given by the infimum over all n-dimensional subspaces V_n in V. For this method, we randomly sample basis matrices $\bar{V}_n$ of an n-dimensional linear subspace to approximate this infimum.

Part A: Sampling a random orthonormal basis $\mathbf{\bar{V}_n}$
To find such bases matrices, we repeatedly sample a random matrix $U_n \in \mathbb{R}^{N\times n}$ with entries uniform over $[0,1]$. Any other interval $[a,b]$ with $a,b \in \mathbb{R}$ can be used as well. We then perform the singular value decomposition (SVD) [13] on U_n. We choose as basis matrix $\bar{V}_n \in \mathbb{R}^{N\times n}$ the left singular matrix A of the SVD of U_n. The basis matrix $\bar{V}_n$ is an orthonormal basis matrix, because the left singular matrix A is orthonormal [13].

Part B: Computing the worst best-approximation error $\mathbf{dist(V_n, S)}$
We now compute $dist(V_n, S)$ for a given basis matrix V_n. To compute $\sup_{s\in S}\inf_{v\in V_n}\|s-v\|_V$ in (2), we calculate $\inf_{v\in V_n}\|s-v\|_V$ for all $s\in S$ and take the supremum.

Part C: Computing the best-approximation error $\mathbf{\inf_{v\in V_n}||s-v||_V}$
We can compute $\inf_{v\in V_n}||s-v||_V$ using the Projection Theorem (see Theorem 1). We can apply the Projection Theorem for $H = V$ and $Y = V_n$. We get that $\inf_{v\in V_n}|||s-v||_V = ||s - \bar{V}_n\bar{V}_n^\top s||_V$ completing our estimation.

The steps are summarized in Algorithm 1.

Algorithm 1 Kolmorogov n-width: randomly sampling bases

```
1: Input: state space $V \subseteq \mathbb{R}^N$, set to approximate $S \subseteq V$, number of random samples $N_s$
2: Output: Approximation of Kolmogorov $n$-width $d_n(S,V)$
3: $d_n \leftarrow \infty$
4: for $i = 1$ to $N_s$ do
5:     $U_n \leftarrow$ Random matrix in $\mathbb{R}^{N\times n}$ with $U_{ij} \sim$ Uniform([0, 1])
6:     $\bar{V}_n \leftarrow A$ : The left singular matrix of the SVD of $U_n$
7:     $dist \leftarrow \max\left\{\|x - \bar{V}_n\bar{V}_n^T x\|_V \mid x \in S\right\}$
8:     $d_n \leftarrow \min(d_n, dist)$
9: end for
10: return $d_n$
```

3.1.2 Proper orthogonal decomposition

Instead of sampling random bases, in this section we attempt to find one specific basis V_n that is close to optimal in the sense that $dist(V_n, S)$ is close to $d_n(S, V)$. The method we use for this is the proper orthogonal decomposition (POD) [14]. This is a widely used method in the field of model order reduction for reducing the dimensionality of a dataset. We take snapshots as training data and find the basis that minimizes the sum of least squares approximation error of the snapshots.

We start by constructing the snapshot matrix $P \in \mathbb{R}^{N\times|S|}$. The j-th column of P contains a snapshot $p_j := s_j$.

We index the elements in S so that we have a bijective mapping, $f : [1, 2, .., |S|] \to S$ and we define $s_j := f(j)$. For all $s_j \in S$, we create a snapshot p_j. This means there are $N_p := |S|$ snapshots in total and thus $P \in \mathbb{R}^{N\times N_p}$ is given by

$$P = [p_1, p_2, .., p_{N_p}].$$

We now perform the SVD on P to obtain its left singular matrix A. To create the orthonormal basis matrix $\bar{V}_n$, we take the first n left singular values of A.

$$\bar{V}_n = [a_1, a_2, ..., a_n] \in \mathbb{R}^{N\times n}$$

where a_k is the k-th column of A.

We pick the singular vectors with the highest singular values, because they capture the most variance in the data in the least-squares sense [13]. This is the basis that minimizes the sum of least-squares approximation error of the snapshot matrix (4) [6].

$$\bar{V}_n = \underset{\bar{V}_n\in\mathbb{R}^{N\times n}}{\arg\min} \sum_{j=1}^{N_p} ||s_j - \bar{V}_n \hat{s_j}||_V^2 \quad \hat{s_j} := \bar{V}_n^\top s_j \in \mathbb{R}^n. \tag{4}$$

Typically, only a subset of snapshots of S is taken to construct the snapshot matrix. When the elements of S are solutions to the full order model, it can be computationally expensive to use the entire set S, or impossible when $|S|$ is infinite. In the example setting in Section 4.1, the whole set S is available, and so the entire set S is used for the snapshot matrix. However, it could be further explored to use only a subset of S to construct the snapshot matrix.

The Kolmogorov n-width, minimizes $dist(S, V_n)$ for all n-dimensional subspaces V_n of V. Subspaces are evaluated based on their approximation of S with the metric $dist(S, V_n)$. Instead, this method evaluates a subspace based on the sum of least-squares approximation error of S.

Our approximation uses the basis $\bar{V}_n$ that minimizes the sum of least squares error approximation error of S in (4) with the aim that this basis is also close to minimizing $dist(S, V_n)$. These are two different metrics that measure the error of a linear subspace in approximating S.

After obtaining the basis matrix $\bar{V}_n$, the value of $dist(V_n, S)$ is calculated using the method described in part B and C of Section 3.1.1. The steps are summarized in Algorithm 2.

3.2 Polynomial Kolmogorov (n,2)-width

Analogously to the previous section, we give two methods for finding the polynomial Kolmogorov $(n, 2)$-width. By choosing the polynomial degree $p = 2$, we aim to show the

Algorithm 2 Kolmogorov n-width: POD

```
1: Input: state space $V \subseteq \mathbb{R}^N$ , set to approximate $S \subseteq V$ with $|S| = N_p$
2: Output: Approximation Kolmogorov $n$-width $d_n(S, V)$
3: for $j = 1$ to $j = N_p$ do
4:     $p_j \leftarrow s_j$
5: end for
6: $P \leftarrow [p_1, p_2, .., p_{N_p}]$
7: $\bar{V}_n \leftarrow [a_1, a_2, ..., a_n]$ with $A := [a_i]_{i=1}^{N_p}$ the left singular matrix of the SVD of $P$
8: $d_n \leftarrow \max \left\{ ||x - \bar{V}_n \bar{V}_n^T x||_V \mid x \in S \right\}$
9: return $d_n$
```

difference of ROMs on polynomially mapped manifolds, while being more manageable in terms of mathematical and computational complexity. These methods are similar to the methods described for the Kolmogorov n-width, with one method using random bases and another constructing one basis like in Section 3.1.2.

For ROMs on linear subspaces, we used the Projection theorem to calculate $\inf_{v \in V} ||x-v||_V$ for a linear subspace V_n exactly. We cannot apply the Projection Theorem in the same way to calculate $\inf_{v \in \widetilde{\mathcal{M}}_{n,2}} ||x - v||_V$ for quadratically mapped manifolds $\widetilde{\mathcal{M}}_{n,2}$. Therefore, $\inf_{v \in \widetilde{\mathcal{M}}_{n,2}} ||x - v||_V$ must be estimated differently.

For $p = 2$ approximations of $\tilde{x} \in V$ from (3) are of the form

$$\tilde{x} := \Gamma_{n,2}(x_n) \quad \Gamma_{n,2}(x_n) = A_1 x_n + A_2 K_s(x_n, 2) \quad x_n \in \mathbb{R}^n \tag{5}$$
$$A_1 \in \mathbb{R}^{N \times n} A_2 \in \mathbb{R}^{N \times t(n,2)-n-1}.$$

In both methods, we attempt to find basis matrices A_1 and A_2 that are close to minimizing $dist(S, \widetilde{\mathcal{M}}_{n,2})$, where $\widetilde{\mathcal{M}}_{n,2} = \mathrm{img}(\Gamma_{n,2}) \subseteq V$.

3.2.1 Randomly sampling bases

Part A: Sampling random orthonormal bases matrices $\mathbf{A_1}$ and $\mathbf{A_2}$
For this method, we choose A_1 in the same way that we chose the basis matrix in Algorithm 1. We sample a random matrix $U_1 \in \mathbb{R}^{N \times n}$ with $(U_1)_{ij} \sim \mathrm{Uniform}([0, 1])$ and choose A_1 as the left singular matrix of the SVD of U_1.

For A_2, we sample a random matrix $U_2 \in \mathbb{R}^{N \times (t(n,2)-n-1)}$ with $(U_2)_{ij} \sim \mathrm{Uniform}([0, 1])$. We perform the SVD on U_2 to get the left singular matrix X_2 of U_2. We obtain A_2 by projecting X_2 onto the orthogonal complement of the space spanned by A_1 such that $A_1^\top A_2 = 0$. This way, the spaces spanned by A_1 and A_2 are orthogonal to each other. Thus A_2 is given by

$$A_2 = (I - A_1 A_1^\top) X_2.$$

Part B: Estimating the worst best-approximation error $\mathbf{dist}(\mathcal{M}_{\mathbf{n,2}}, \mathbf{S})$
We map an element $x \in S$ to it's reduced coordinates x_n by projecting x onto the subspace spanned by A_1 with

$$x_n = A_1^\top x.$$

We use A_1 and A_2 for the map $\Gamma_{n,2}$ in (5). The quadratically mapped submanifold $\mathcal{M}_{n,2}$ is given by $\mathcal{M}_{n,2} = \mathrm{img}(\Gamma_{n,2}) \subseteq V$.

The value of $dist(S, \widetilde{\mathcal{M}}_{n,2})$ is given by the supremum over $x \in V$ of $\inf_{v \in \widetilde{\mathcal{M}}_{n,2}} ||x - v||_V$. For a particular $x \in V$, we estimate $\inf_{v \in \widetilde{\mathcal{M}}_{n,2}} ||x - v||_V$ with $||x - \tilde{x}||_V$, where $\tilde{x}$ is our approximation of x given by (5).

Thus, our final estimation of $dist(S, \widetilde{\mathcal{M}}_{n,2})$ is the supremum over $x \in V$ of $||x - \tilde{x}||_V$ and our estimation of $d^{\otimes}_{n,2}(S, V)$ is given by the minimum estimation of $dist(S, \widetilde{\mathcal{M}}_{n,2})$ over the realizations of $\widetilde{\mathcal{M}}_{n,2}$. The steps are summarized in Algorithm 3.

Algorithm 3 polynomial Kolmorogov $(n, 2)$-width: randomly sampling bases

```
1: Input: state space $V \subseteq \mathbb{R}^N$, set to approximate $S \subseteq V$, number of random samples $N_s$
2: Output: Approximation polynomial Kolmogorov $(n, 2)$-width $d^{\otimes}_{n,2}(S, V)$
3: $d^{\otimes}_{n,2} \leftarrow \infty$
4: for $i = 1$ to $N_s$ do
5:     $U_1 \leftarrow$ Random matrix in $\mathbb{R}^{N \times n}$ with $(U_1)ij \sim \text{Uniform}([0, 1])$
6:     $A_1 \leftarrow X_1$ : The left singular matrix of the SVD of $U_1$

7:     $U_2 \leftarrow$ Random matrix in $\mathbb{R}^{N \times (t(n,2) - n - 1)}$ with $(U_2)ij \sim \text{Uniform}([0, 1])$
8:     $A_2 = (I_n - A_1 A_1^\top) X_2$ with $X_2$ the left singular matrix of the SVD of $U_2$

9:     $dist \leftarrow 0$
10:    for $x \in S$ do
11:        $x_n = A_1^\top x$
12:        $\tilde{x} = A_1 x_n + A_2 K_s(x_n, 2)$
13:        $dist \leftarrow \max(dist, ||\tilde{x} - x||_V)$
14:    end for

15:    $d^{\otimes}_{n,2} \leftarrow \min(d^{\otimes}_{n,2}, dist)$
16: end for
17: return $d^{\otimes}_{n,2}$
```

3.2.2 Extended proper orthogonal decomposition

Similarly to Section 3.1.2 we construct one set of bases A_1 and A_2 that aim to be close to minimising the worst best-approximation error in (2). The basis matrix A_1 is constructed in the same way as the basis matrix $\bar{V}_n$ in Algorithm 2 using the POD. That is, we create a snapshot matrix $P \in \mathbb{R}^{N \times N_P}$, where $N_P := |S|$, with unique snapshots $p_j := s_j$ for $j \in \{1, ..., N_p\}$, $s_j \in S$. We take the first n left singular vectors of the SVD of P to create the basis matrix A_1.

$$A_1 = [x_1, x_2, ..., x_n] \in \mathbb{R}^{N \times n}$$

where x_i is the i-th left singular vector of P.

From now on, we follow the steps from [6] to construct the quadratic term A_2. The method in [6] extends the traditional POD method for quadratically mapped manifolds. We summarize their method here, but applied to our setting.

The part of P not included by the linear basis matrix A_1 can be found by projecting P onto the orthogonal complement of the space spanned by A_1. We denote this as the residual $\mathcal{E}$ given by

$$\mathcal{E} := (I_N - A_1 A_1^\top)P.$$

Similarly to transforming the reduced coordinates with the Kronecker product in (3), we also transform the snapshots p_j with the symmetric Kronecker product of their reduced coordinates. We get a snapshot matrix W of the reduced coordinates for the quadratic term.

$$W := [K_s(\hat{p_1}, 2), K_s(\hat{p_2}, 2), ..., K_s(\hat{p_{N_p}}, 2)] \in \mathbb{R}^{N \times n(n+1)/2} \qquad \hat{p_j} := A_1^\top p_j \in \mathbb{R}^n.$$

Again, we obtain the reduced coordinates $\hat{p_j}$ by projecting p_j onto the space spanned by A_1. Similarly to (4), we choose A_2 to minimize the sum of least-squares of the approximation error of the snapshot matrix.

$$\begin{aligned} A_2 &= \underset{A_2 \in \mathbb{R}^{N \times n(n+1)/2}}{\arg\min} \sum_{j=1}^{N_p} ||s_j - \Gamma_{n,2}(\hat{s_j})||_V^2 \\ &= \underset{A_2 \in \mathbb{R}^{N \times n(n+1)/2}}{\arg\min} \sum_{j=1}^{N_p} ||s_j - A_1 \hat{s_j} - A_2 K_s(\hat{s_j}, 2)||_V^2. \end{aligned}$$

This can be rewritten in terms of the frobenious norm of a matrix with

$$A_2 = \underset{A_2 \in \mathbb{R}^{N \times n(n+1)/2}}{\arg\min} \frac{1}{2} ||W^\top A_2^\top - \mathcal{E}^\top||_F^2. \tag{6}$$

Large coefficients can lead to overfitting, so we add a regularization parameter γ that penalizes A_2 for having a large norm. This transforms (6) into (7).

$$A_2 = \underset{A_2 \in \mathbb{R}^{N \times n(n+1)/2}}{\arg\min} \left(\frac{1}{2} ||W^\top A_2^\top - \mathcal{E}^\top||_F^2 + \frac{\gamma}{2} ||A_2||_F^2\right). \tag{7}$$

Equation (7) can be solved explicitly. In the end, we have A_2 given by

$$A_2 = \mathcal{E} W^\top (W W^\top + \gamma I_{n(n+1)/2})^{-1}.$$

We use A_1 and A_2 to create the mapping $\Gamma_{n,2}$ in (5). We estimate $dist(S, \widetilde{\mathcal{M}}_{n,2})$ with $\widetilde{\mathcal{M}}_{n,2} = \mathrm{img}(\Gamma_{n,2})$ in the same way as in Section 3.2.2 to get our approximation of the polynomial Kolmorogov $(n, 2)$-width.

The steps are summarized in Algorithm 4.

Algorithm 4 polynomial Kolmorogov $(n, 2)$-width: Extended POD

```
1: Input: state space $V \subseteq \mathbb{R}^N$, set to approximate $S \subseteq V$ with $|S| = N_p$, regularization parameter $\gamma$
2: Output: Approximation polynomial Kolmogorov $(n, 2)$-width $d^{\otimes}_{n,2}(S, V)$

3: for j=1 to j=$N_p$ do
4:     $p_j \leftarrow s_j$
5: end for

6: $P \leftarrow [p_1, p_2, .., p_{N_p}]$
7: $A_1 \leftarrow [x_1, x_2, ..., x_n]$ with $X := [x_i]_{i=1}^{N_p}$ the left singular matrix of the SVD of $P$

8: for j=1 to j=$N_p$ do
9:     $\hat{p}_j \leftarrow A_1^\top p_j$
10: end for

11: $\mathcal{E} \leftarrow (I_N - A_1 A_1^\top)P$
12: $W \leftarrow [K_s(\widehat{p}_1, 2), K_s(\widehat{p}_2, 2), ..., K_s(\widehat{p}_{N_p}, 2)]$
13: $A_2 \leftarrow \mathcal{E}W^\top(WW^\top + \gamma I_{n(n+1)/2})^{-1}$

14: $d^{\otimes}_{n,2} \leftarrow 0$
15: for $x \in S$ do
16:     $x_n = A_1^\top x$
17:     $\tilde{x} = A_1 x_n + A_2 K_s(x_n, 2)$
18:     $d^{\otimes}_{n,2} \leftarrow \max(d^{\otimes}_{n,2}, ||\tilde{x} - x||_V)$
19: end for

20: return $d^{\otimes}_{n,2}$
```

3.3 Time complexity analysis

In this section, we will analyse the time complexity of the four algorithms that approximate the Kolmogorov n-width and polynomial Kolmogorov (n, p)-width. The time complexity analysis is in Big-O notation and in terms of the number of elementary operations. These elementary operations include initialization steps such as setting a variable to a specific value, simple arithmetic operations (e.g., addition, subtraction, multiplication, and division of scalar values), and basic comparisons or assignments. To facilitate the analysis, we first review the time complexities of some well known operations used within the algorithms. These are the time complexities we will assume for these operations.

- **Matrix Multiplication: [M]** The standard algorithm for the multiplication of an $p \times q$ matrix with a $q \times r$ matrix has time complexity $O(p \cdot q \cdot r)$. There are algorithms with a lower bound for the time complexity, but these are usually not used in practice.
- **Singular Value Decomposition (SVD) [S]:** Computing the SVD of an $n \times m$ matrix is commonly done with the Golub-Reinsch algorithm [7], which has a time complexity of $O(n \cdot m \cdot \min(n, m))$ for computing the SVD of an $n \times m$ matrix.
- **Matrix Inversion [I]:** An $n \times n$ square matrix can be inverted in $O(n^3)$ time [2]. Algorithms with a better bound exist, but still remain close to cubic time complexity.

The symbols **[M], [S]** and **[I]** indicate the respective operation(s) used in a step. Additionally, the assumption is made that the norm $\|x\|_V$ for $x \in \mathbb{R}^N$ can be computed in $O(N)$, as is typically the case.

3.3.1 Algorithm 1: Kolmogorov n-width

Input N, n, S, N_s	
Step	**Time Complexity**
$d_n \leftarrow \infty$	
for $i = 1$ to N_s **do**	N_s iterations
$U_n \leftarrow$ Random matrix in $\mathbb{R}^{N\times n}$	$O(N \cdot n)$
$\bar{V}_n \leftarrow$ the left singular matrix of the SVD of U_n	$O(N \cdot n^2)$ **[S]**
$dist \leftarrow \max\left\{\|\mathbf{x} - \bar{V}_n\bar{V}_n^T\mathbf{x}\|_V \mid \mathbf{x} \in S\right\}$	$O(n \cdot N^2 \cdot \|S\|)$ **[M]**
$d_n \leftarrow \min(d_n, dist)$	
end for	
return d_n	

Table 1: Time complexity analysis of Algorithm 1

The time complexity of each iteration is identical. Each iteration in the for loop has time complexity

$$O(N \cdot n) + O(N \cdot n^2) + O(n \cdot N^2 \cdot |S|) = O(n \cdot N^2 \cdot |S|).$$

The time complexity is linear in the number of basis samples N_s with a final time complexity of

$$O(N_s \cdot n \cdot N^2 \cdot |S|)).$$

3.3.2 Algorithm 2: Kolmogorov n-width

Input N, n, S	
Step	**Time Complexity**
for $j = 1$ to $\|S\|$ **do**	$\|S\|$ iterations
$p_j \leftarrow s_j$ in basis V	$O(N)$
end for	
$P \leftarrow [p_1, p_2, \ldots, p_{\|S\|}]$	$O(N \cdot \|S\|)$
$\bar{V}_n \leftarrow [a_1, a_2, \ldots, a_n]$ with $A := [a_i]_{i=1}^{\|S\|}$ the left singular matrix of the SVD of P	$O(N \cdot \|S\| \cdot \min(N, \|S\|))$ **[S]**
$d_n \leftarrow \max\left\{\|\mathbf{x} - \bar{V}_n\bar{V}_n^T\mathbf{x}\|_V \mid \mathbf{x} \in S\right\}$	$O(n \cdot N^2 \cdot \|S\|)$ **[M]**
return d_n	

Table 2: Time complexity of Algorithm 2

The steps in Table 2, come to a combined time complexity of

$$\begin{aligned} &O(|S| \cdot N) + O(N \cdot |S|) + O(N \cdot |S| \cdot \min(N, |S|)) + O(n \cdot N^2 \cdot |S|) \\ =&O(N(|S| + |S| \cdot \min(N, |S|) + n \cdot N \cdot |S|)) \\ =&O(N^2 \cdot n \cdot |S|). \end{aligned}$$

3.3.3 Algorithm 3: polynomial Kolmogorov (n,p)-width

Input N, n, S, N_s	
Step	**Time Complexity**
$d^{\otimes}_{n,2} \leftarrow \infty$	
for $i = 1$ to N_s **do**	N_s iterations
$U_1 \leftarrow$ Random matrix in $\mathbb{R}^{N \times n}$	$O(N \cdot n)$
$A_1 \leftarrow X_1$: The left singular matrix of the SVD of U_1	$O(N \cdot n^2)$ **[S]**
$U_2 \leftarrow$ Random matrix in $\mathbb{R}^{N \times (t(n,2)-n-1)}$	$O(N \cdot t(n,2))$
$A_2 = (I_n - A_1 A_1^\top) X_2$ with X_2 the left singular matrix of the SVD of U_2	$O(N^2 \cdot t(n,2))$**[M]** $+$ $O(N \cdot t(n,2) \cdot \min(N, t(n,2)))$ **[S]**
$dist \leftarrow 0$	
for $x \in S$ **do**	$\lvert S \rvert$ iterations
$x_n = A_1^T x$	$O(n \cdot N)$
$\tilde{x} = A_1 x_n + A_2 K_s(x_n, 2)$	$O(N \cdot n + N \cdot t(n,2))$
$dist \leftarrow \max(dist, \lVert \tilde{x} - x \rVert_V)$	$O(N)$
end for	
$d^{\otimes}_{n,2} \leftarrow \min(d^{\otimes}_{n,2}, dist)$	
end for	
return $d^{\otimes}_{n,2}$	

Table 3: Time complexity analysis of Algorithm 3

Computation of $K_s(x_n, 2) \in \mathbb{R}^{t(n,2)}$ can be done in $O(t(n,2))$ time. In this derivation, we use that $t(n,2) = \frac{n \cdot (n+1)}{2}$ implying that $O(t(n,2)) = O(\frac{n \cdot (n+1)}{2}) = O(n^2)$. Each iteration in the outer for loop has a time complexity of

$$\begin{aligned}
&O(N \cdot n) + O(N \cdot n^2) + O(N \cdot t(n,2)) + O(N^2 \cdot t(n,2)) \\
&+O(N \cdot t(n,2) \cdot \min(N, t(n,2)) + O(|S|(n \cdot N + N \cdot n + N \cdot t(n,2) + N) \\
&=O(N^2 \cdot n^2) + O(|S| \cdot N \cdot n^2) \\
&=O(N \cdot n^2 \cdot (N + |S|)).
\end{aligned}$$

There are N_s identical iterations, totalling a time complexity of

$$O(N_s \cdot N \cdot n^2 \cdot (N + |S|)).$$

3.3.4 Algorithm 4: polynomial Kolmogorov (n,p)-width

Input N, n, S	
Step	**Time Complexity**
for $j = 1$ to $\|S\|$ **do**	$\|S\|$ iterations
$\quad p_j \leftarrow s_j$ in basis $\bar{V}$	$O(N)$
end for	
$P \leftarrow [p_1, p_2, \ldots, p_{\|S\|}]$	$O(N \cdot \|S\|)$
$A_1 \leftarrow [x_1, x_2, \ldots, x_n]$ with $X := [x_i]_{i=1}^{\|S\|}$ the left singular matrix of the SVD of P	$O(N \cdot \|S\| \cdot min(N, \|S\|))$ **[S]**
For j=1 to $\|S\|$ **do**	$\|S\|$ iterations
$\quad \hat{p}_j \leftarrow A_1^\top p_j$	$O(N \cdot n)$
end for	
$\mathcal{E} \leftarrow (I_N - A_1 A_1^T)P$	$O(N^2 \cdot \|S\|)$ **[M]**
$W \leftarrow [K_s(\hat{p}_1, 2), K_s(\hat{p}_2, 2), \ldots, K_s(\hat{p}_{\|S\|}, 2)]$	$O(\|S\| \cdot t(n, 2))$
$A_2 \leftarrow \mathcal{E}W^T(WW^T + \gamma I_{n(n+1)/2})^{-1}$	$O(N \cdot \|S\| \cdot t(n, 2))$ **[M]** + $O(t(n, 2)^2 \cdot \|S\|)$**[M]** + $O(t(n, 2)^3)$**[I]** + $O(N \cdot t(n, 2)^2)$**[M]**
$d_{n,2}^{\otimes} \leftarrow 0$	
for $x \in S$ **do**	$\|S\|$ iterations
$\quad x_n = A_1^T x$	$O(n \cdot N)$
$\quad \tilde{x} = A_1 x_n + A_2 K_s(x_n, 2)$	$O(n \cdot N + N \cdot t(n, 2))$
$\quad d_{n,2}^{\otimes} \leftarrow \max(d_{n,2}^{\otimes}, \|\tilde{x} - x\|_V)$	$O(N)$
end for	
return $d_{n,2}^{\otimes}$	$O(1)$

Table 4: Time complexity analysis of Algorithm 4

We will split the calculation up into three parts. The first part is up to and including the calculation of A_1. The time complexity for this part is

$$O(|S| \cdot N) + O(|S| \cdot N) + O(|S| \cdot N \cdot \min(N, |S|)) = O(|S| \cdot N \cdot min(N, |S|).$$

The second part is up to and including the calculation A_2. The time complexity for this part is given by

$$\begin{aligned}
&O(|S| \cdot N \cdot n) + O(|S| \cdot N^2) + O(|S| \cdot t(n, 2)) + O(N \cdot |S| \cdot t(n, 2)) + \\
&O(|S| \cdot t(n, 2)^2) + O(t(n, 2)^3) + O(t(n, 2)^2 \cdot N) \\
=&O(|S| \cdot N^2) + O(N \cdot |S| \cdot n^2) + O(|S| \cdot n^4) + O(n^6) + O(N \cdot n^4) \\
=&O(|S|(N^2 + N \cdot n^2 + n^4) + n^6 + N \cdot n^4).
\end{aligned}$$

The remaining part has a time complexity of

$$O(|S|(n \cdot N + n \cdot N + N \cdot t(n, 2) + N)) = O(|S| \cdot N \cdot n^2).$$

We can see that the total time complexity is dominated by the second part and thus the final time complexity is

$$O(|S|(N^2 + N \cdot n^2 + n^4) + n^6 + N \cdot n^4)).$$

3.3.5 Time complexity summary

Algorithm	Inputs	Time complexity
1	N, n, S, N_s	$O(N_s \cdot N^2 \cdot n \cdot \lvert S \rvert))$
2	N, n, S	$O(N^2 \cdot n \cdot \lvert S \rvert)$
3	N, n, S, N_s	$O(N_s \cdot N \cdot n^2 \cdot (N + \lvert S \rvert))$
4	N, n, S	$O(\lvert S \rvert(N^2 + N \cdot n^2 + n^4) + n^6 + N \cdot n^4))$

Table 5: Time complexities of the four algorithms

Table 5 summarizes the time complexity of the four algorithms. To estimate the Kolmogorov n-width, Algorithm 1 has a time complexity with an extra linear factor N_s compared to Algorithm 2. It should be noted that the complexity of an algorithm is not definitive for its running time.

The time complexities of Algorithm 3 and 4 cannot be compared easily, without making assumptions about the relations between the inputs. For example, in section 4.1, S is chosen in a way such that $|S|$ grows quadratically in N. Then we can use that $O(S) = O(N^2)$ to reduce the number of inputs that the time complexity depends on. Multiple of these assumptions are needed, to simply the comparison.

4 Numerical investigation

In this section, we implement the four algorithms for an example setting we define in Section 4.1. We evaluate how the two methods of randomly sampling bases and constructing a basis based on the POD compare for both the Kolmogorov n-width and polynomial Kolmogorov $(n, 2)$-width. We then use these methods to examine the relationship given by Theorem 2.

4.1 Example setting

In this section we define our state space V_N and the subset to approximate $S_N \subset V_N$.

This example setting is adapted from [3]. For this setting, the Kolmogorov n-width decays slowly as a function of the reduced dimension n. Thus, there is potential gain over linear subspace ROMs in terms of the approximation error.

Let $(V_N, \|\cdot\|_{V_N})$ be the space of left-continuous piecewise constant functions with a period of 1. As a basis for V_N, we choose $\{f_1, f_2, ..., f_N\}$ where f_i on $[0, 1)$ is defined as

$$f_i(x) := \begin{cases} \sqrt{N}, & \text{if } \frac{i-1}{N} \le x < \frac{i}{N} \\ 0, & \text{else} \end{cases} \qquad x \in [0, 1).$$

The coordinate vector $c_f \in \mathbb{R}^N$ of a function $f \in V_N$ in the basis $\bar{V_N}$ is given by

$$c_f := (c_i)_{i=1}^N \quad c_i \in \mathbb{R} \text{ such that} \quad f(x) = \sum_{i=1}^N c_i f_i(x).$$

The inner product $\langle f, g \rangle_{V_N}$ is defined by the dot product of the coordinate vector of f and g. The definition of the norm $\|\cdot\|_{V_N}$ follows

$$\langle f, g\rangle_{V_N} := \langle c_f, c_g\rangle_{\mathbb{R}^N} \quad ||f||_{V_N} = \sqrt{\langle f, f\rangle_{V_N}} = \sqrt{\langle c_f c_f\rangle_{\mathbb{R}^N}}.$$

We approximate the set $S_N \subset V_N$ of discrete periodic "box functions" with a height of 1 and a period of 1. They are defined on $[0, 1)$ as

$$b_{a,l}(x) = \begin{cases} 1, & \text{if } a \leq x < ((a + l) \mod 1) \\ 0, & \text{else} \end{cases}$$

$$x \in [0, 1), \quad a \in \{0, \frac{1}{N}, ..., \frac{N-1}{N}\}, \quad l \in \{l_{min}, l_{min} + \frac{1}{N}, ..., 1 - l_{min}\}$$

with the constant $l_{min} \in (0, \frac{1}{2})$ a multiple of $\frac{1}{N}$. It can be verified that $S_N \subset V_N$ by writing $b_{a,l}$ as a linear combination of the basis vectors in $\bar{V_N}$

4.2 Results

In this section, we start with a comparison of Algorithm 1 and Algorithm 2 to compute the Kolmogorov n-width. We do the same for Algorithm 3 and 4. Finally, we use these methods to examine Theorem 2 numerically for $p = 2$ and analyse the decay of the polynomial Kolmogorov $(n, 2)$-width within the bounds of Theorem 2.

4.2.1 Random bases sampling versus basis from POD

Figure 1 shows the approximations of $d_n(S_N, V_N)$ by Algorithm 1 and 2 plotted as a function of the reduced dimension n. Increasing the number of bases samples N_s for Algorithm 1 improves the approximation of $d_N(S_N, V_N)$. We can hypothesise that, as $N_s \to \infty$, the approximation should converge to the theoretical value of $d_N(S_N, V_N)$.

In figure 2, the decay of $d_n(S_N, V_N)$, averaged over 200 runs of Algorithm 1, is shown in relation to the number of random basis samples N_s. The decay of $d_n(S_N, V_N)$ can be approximated by a least squares fit of a power law with horizontal asymptote of the form $f(N_s) = a \cdot N_s^{-b} + c$. The asymptote of the power law fit can be used to approximate $d_n(S_N, V_N)$ with $d_n(S_N, V_N) \approx \lim_{N_s \to \infty} f(N_s) = c$. The approximations of $d_n(S_N, V_N)$ in figure 2(a) are plotted in three clusters of N_s values, spread far apart. This is with the aim to improve the fit for large N_s, with less data points. The samples in Figure 2 give an approximation of $d_n(S_N, V_N) \approx 0.292$ this way, while Algorithm 2 gives an approximation of $d_n(S_N, V_N) \approx 0.289$. With more samples of Algorithm 1 over a larger range of N_s, a more accurate approximation of $d_n(S_N, V_N)$ could be found this way.

The number of entries in a random basis matrix U_n in Algorithm 1 is $N \cdot n$. This means that for larger N, more random bases are required to find a good approximation of $d_n(S_N, V_N)$, because there is a larger space of bases matrices to sample.

Table 6 compares the relative performance of Algorithm 1 to Algorithm 2. For $N = 20$, Algorithm 1 already performs significantly worse than for $N = 10$ compared to Algorithm 2, for the same number of bases samples N_s. For a relatively small $N = 10$ and a large number of basis samples $N_s = 10^7$, algorithm 1 still estimates $d_n(S_N, V_N)$ as 1.244 times higher than Algorithm 2. Additionally, Algorithm 2 is much faster to compute than Algorithm 1 for large N_s.

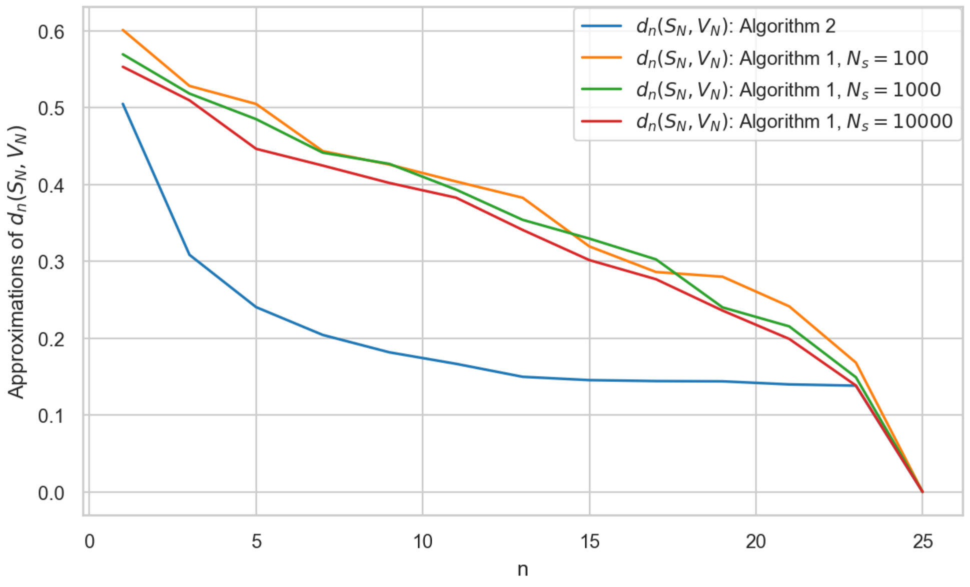
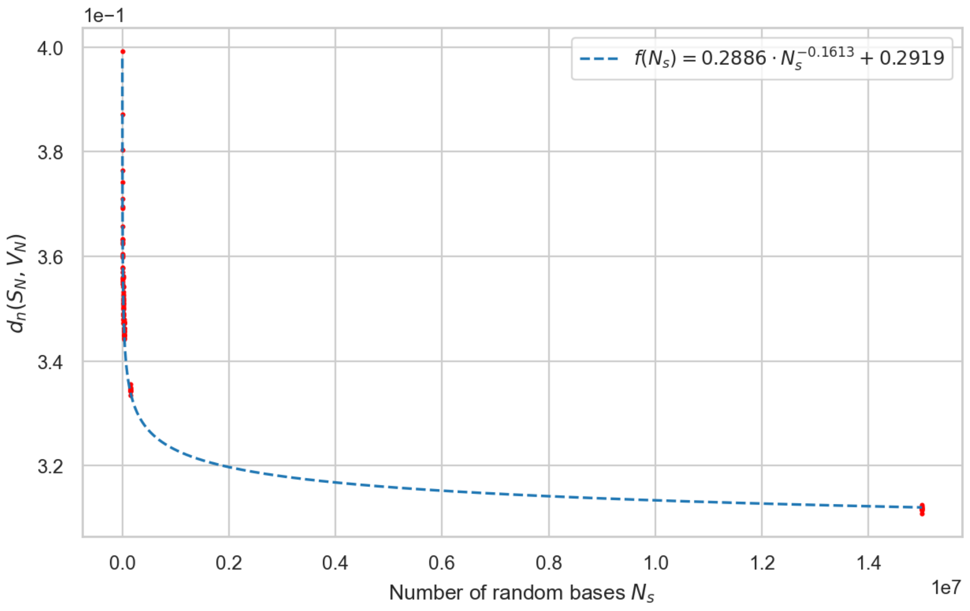


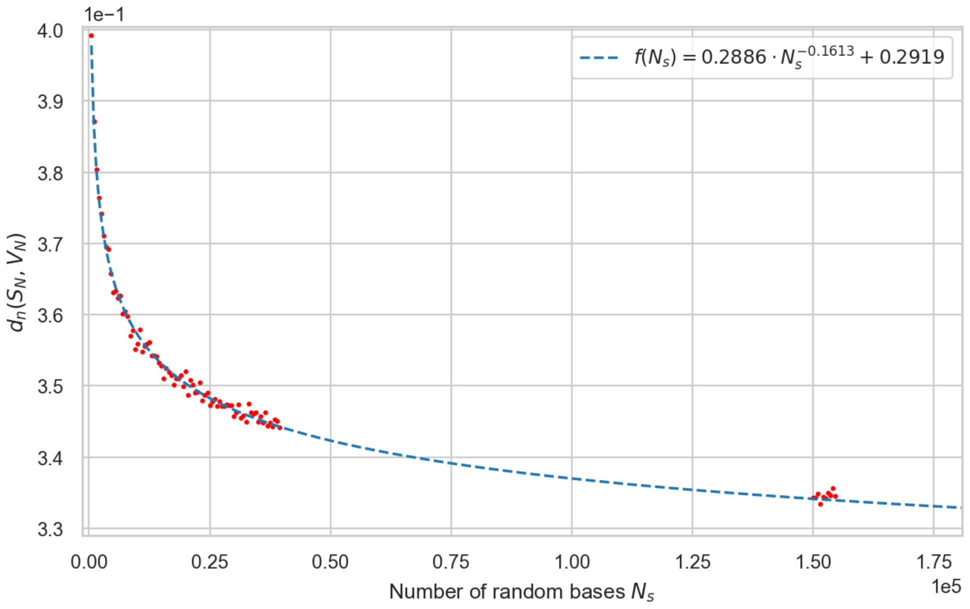
Figure 1: Algorithm 2 vs Algorithm 1 for different values of N_s with full dimension $N = 25$.

N_s	$N = 10, n = 5$	$N = 20, n = 10$
10^4	$1.520\times$	$2.153\times$
10^5	$1.378\times$	$1.981\times$
10^6	$1.262\times$	$1.829\times$
10^7	$1.244\times$	-

Table 6: Ratio of Algorithm 1 and Algorithm 2's approximation of $d_n(S_N, V_N)$ for different values of N_s evaluated for $N = 10$ with $n = 5$ and $N = 20$ with $n = 10$. ($1.000\times$ Meaning that Algorithm 1 and 2's output are equal.)



(a)



(b) Zoomed in version of (a) for clarity.

Figure 2: Decay of $d_n(S_N, V_N)$ as the number of bases samples N_s increases for $N = 6$ and $n = 3$. For each N_s, the value of $d_n(S_N, V_N)$ is calculated by averaging over 200 runs. There are three clusters of samples of N_s, taken far apart from eachother, for a better fit.

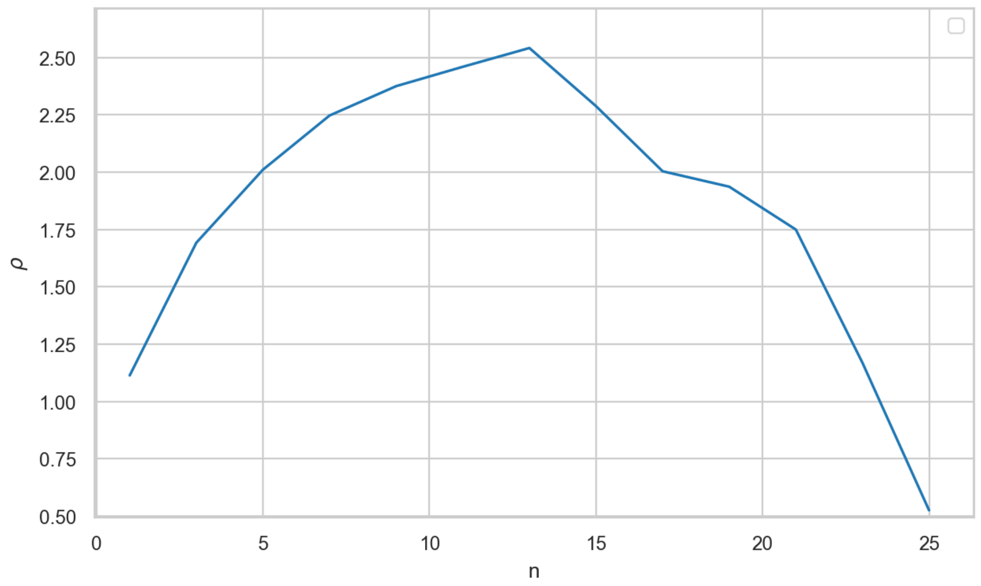
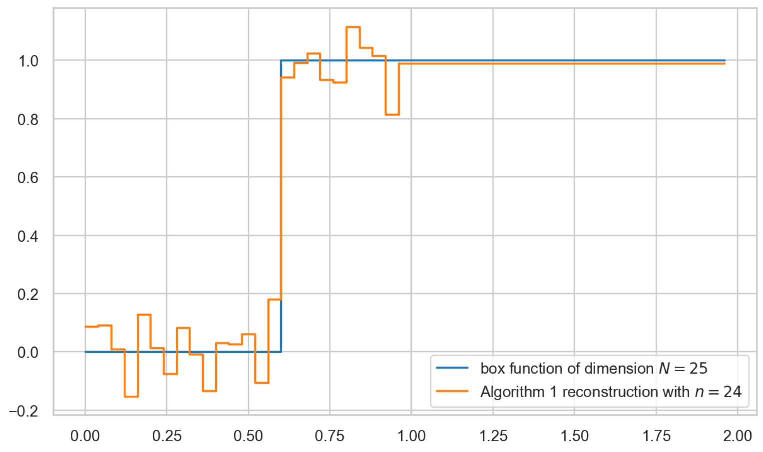


Figure 3: The ratio of approximations of Algorithm of 1 and 2 denoted by $\rho := \frac{d_n^{(1)}(S_N,V_N)}{d_n^{(2)}(S_N,V_N)}$ with (i) indicating the approximation of Algorithm i. ($N = 25$, $N_s = 10^5$)

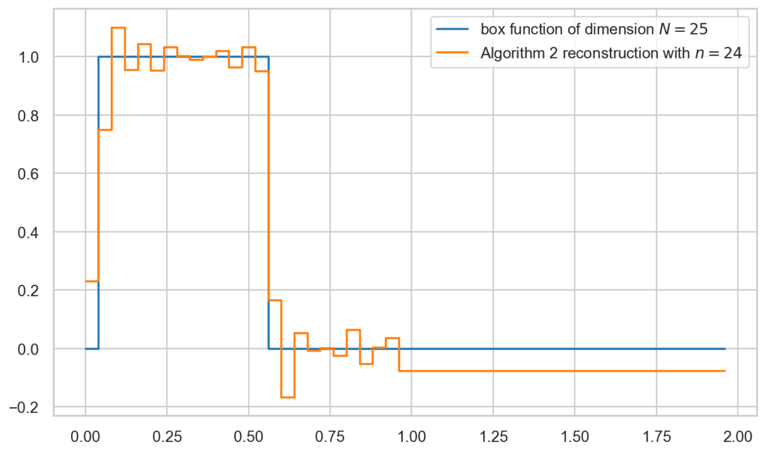
Figure 3 show the ratio ρ between Algorithm 1 and 2's approximation of $d_n(S_N, v_N)$. We see that ρ peaks around $n = N/2$. This is where the difference between Algorithm 1 and 2 is largest. For small n, there is a smaller space of bases to sample and the difference between Algorithm 1 and 2 is smaller. For large n, close to N, the choice of basis is less important (in the extreme case $n = N$, any basis spans V_N and gives a perfect approximation). Here the difference between Algorithm 1 and 2 is smaller again.

Notably, we see in Figure 3 that when n is very close to N, Algorithm 1 gives a lower estimate than Algorithm 2. Figure 4 shows Algorithm 1 and 2's worst best-reconstruction (the reconstruction of $\arg\max_{s \in S_N} \inf_{v \in V_n} ||s - v||_V$.) for their chosen subspace V_n of a box function in S_N when n is very close to N ($N = 25$, $n = 24$). Here Algorithm 1 outperforms Algorithm 2. Minimizing the sum of least squares error in (4) is different from minimizing the worst best-approximation error in (2). Algorithm 2 is optimal in the sum of least square sense, and therefore it weights the squared approximation error of every element in S_N. The Kolmogorov n-width is defined solely based on the error of the worst best-approximation in S_N. The definition is independent of the best-approximation error of other elements in S_N.

This is illustrated in Figure 5, which shows the approximation error using the bases chosen in Algorithm 1 and 2 for all box functions $b_{a,l} \in S_N$, when n is close to N ($n = 24$, $N = 25$). The worst best-approximation error for the basis in Algorithm 2 is 44% higher than the worst best-approximation error of the basis chosen in Algorithm 1. This is, despite the average approximation error of the basis in Algorithm 2 being close to three times lower than the average approximation error of the basis chosen in Algorithm 1.

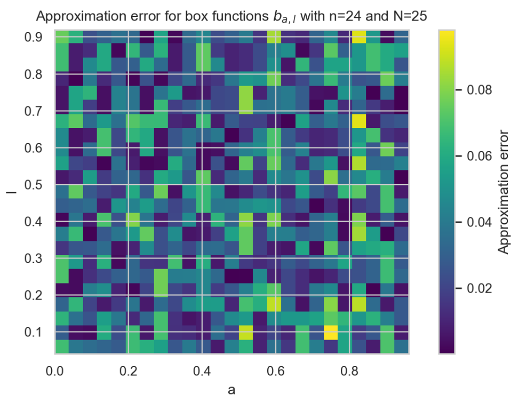


(a) Algorithm 1 with $N_s = 10^5$ and the approximation of $d_n(S_N, V_N) = 0.089$.

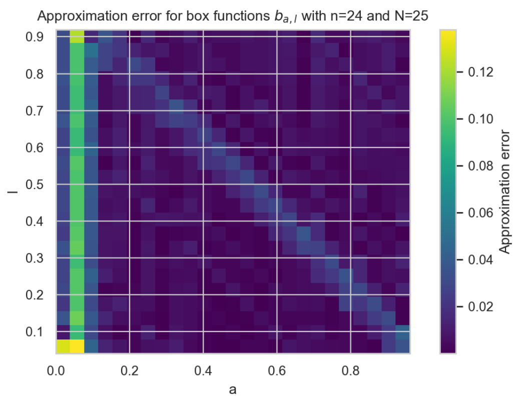


(b) Algorithm 2 with the approximation of $d_n(S_N, V_N) = 0.092$.

Figure 4: The worst best-reconstructions of Algorithm 1 and 2 for $N = 25$ and $n = 24$. (The box functions plotted are the $\arg\max_{s \in S_N} \inf_{v \in V_n} ||s - v||_V$ for the subspace V_n chosen in Algorithms 1 and 2 with their respective reconstructions.)



(a) Algorithm 1



(b) Algorithm 2

Figure 5: Approximation error of all box functions $b_{a,l} \in S_N$ with a on the x-axis and l on the y-axis for the bases chosen in Algorithm 1 and 2. (n=24, N=25, $N_s = 10^4$)

We now look at the random bases sampling and the basis from POD methods for the polynomial Kolmogorov $(n, 2)$-width with Algorithm 3 and 4 respectively. The results are almost identical to the results for the Kolmogorov n-width. Figure 6 shows a sample reconstruction with Algorithm 2 and 4. The green and orange line segments indicate the reconstruction of Algorithm 2 and 4 of the box function in blue, respectively. The difference in height between the line segments is very small, showing that Algorithm 2 and 4 produce similar reconstructions. For this example setting, the quadratic term seems to have little impact on the reconstruction. This suggests that Algorithm 2 and 4 might behave similarly to Algorithm 1 and 3.

Figure 7 compares Algorithm 3 and 4, similarly to how Figure 1 does. The relation between Algorithm 3 and 4 seems to be the same as between Algorithm 1 and 2. In the next section, we also see the similarity between the approximations of Algorithm 2 and 4.

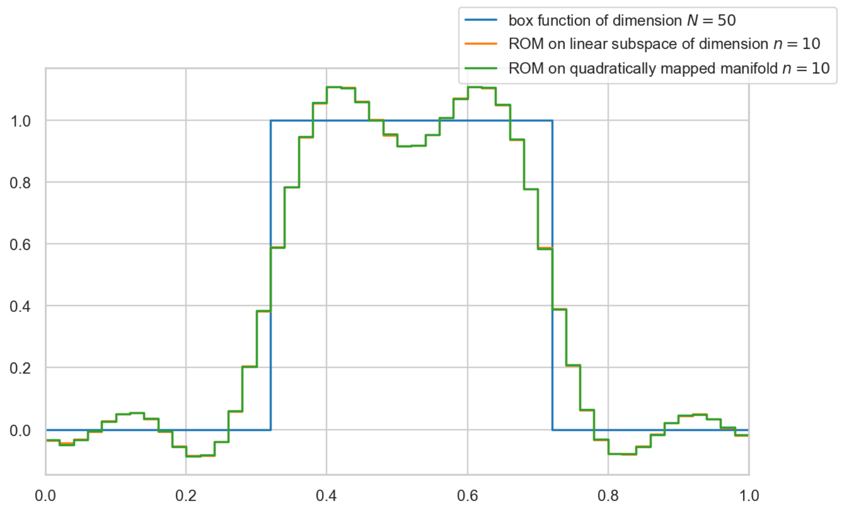


Figure 6: A sample box function reconstruction with Algorithm 2 and 4 for $N = 25$ and $n = 10$.

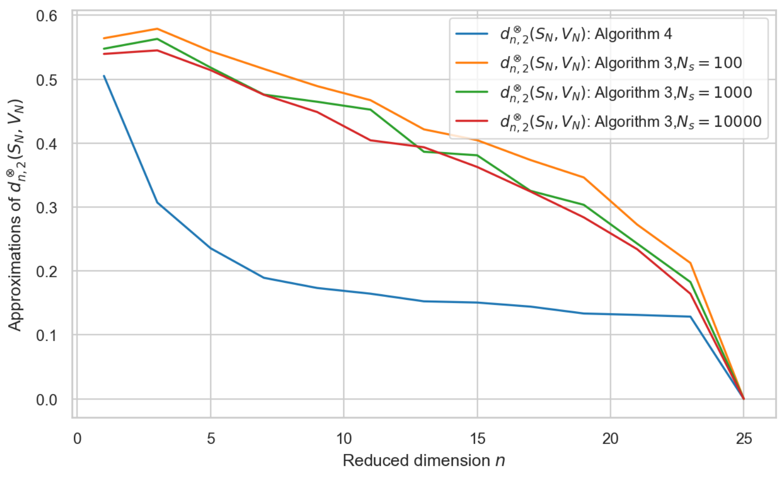


Figure 7: Algorithm 3 vs Algorithm 4 for different values of N_s with full dimension $N = 25$ and $\gamma = 20$.

4.2.2 Approximation bounds for polynomial Kolmogorov (n,2)-width

Theorem 2 gives us bounds for the polynomial Kolmogorov (n, p)-width in terms of the Kolmogorov n-width and $t(n, p)$-width. We now look at how $d^{\otimes}_{n,2}(S_N, V_N)$ behaves within the bounds of Theorem 2.

Figure 8 shows a comparison of the widths in Theorem 2. For Algorithm 4, γ was chosen to minimize the average value of the estimated width over the possible values of the reduced dimension n less than N. The approximations for $d_n(S_N, V_N)$ and $d^{\otimes}_{n,2}(S_N, V_N)$ are very close and it is unclear from Figure 8 alone, if the approximations satisfy the inequality in Theorem 2. Figure 9 shows the ratio ρ between $d_n(S_N, V_N)$ and $d^{\otimes}_{n,2}(S_N, V_N)$. It confirms that the approximation of $d^{\otimes}_{n,2}$ is indeed lower than the approximation of $d_n(S_N, V_N)$.

The approximation of $d^{\otimes}_{n,2}(S_N, V_N)$ is between the approximations of $d_n(S_N, V_N)$ and $d_{t(n,2)}(S_N, v_N)$ as would be expected by Theorem 2. However, for this choice of V_N and S_N, the difference between the approximations of $d_n(S_N, V_N)$ and $d^{\otimes}_{n,2}(S_N, V_N)$ is between 0.1% and 1% as seen in Figure 9.

Figure 10 compares the approximation bounds in Theorem 2 using Algorithm 1 and 3 instead of Algorithm 2 and 4 like in Figure 8. Here, the difference between the estimations of $d_n(S_N, V_N)$ and $d^{\otimes}_{n,2}(S_N, V_N)$ is bigger (up to 10% difference) than in Figure 8. The estimations in Figure 10 do satisfy the inequality in Theorem 2. In Figure 8, $d_n(S_N, V_N)$ and $d^{\otimes}_{n,2}(S_N, V_N)$ decay (as a function of n) quickly at first and then slow down. Whereas, the estimations in Figure 10 by Algorithm 1 and 3 mostly decay close to linearly.

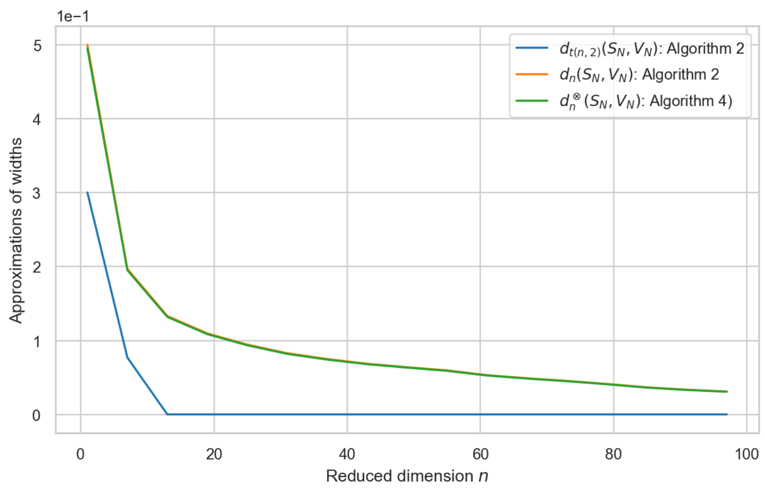


Figure 8: Comparison of the approximation bounds in Theorem 2 using Algorithm 2 and 4. ($N = 100$, $\gamma = 20$)

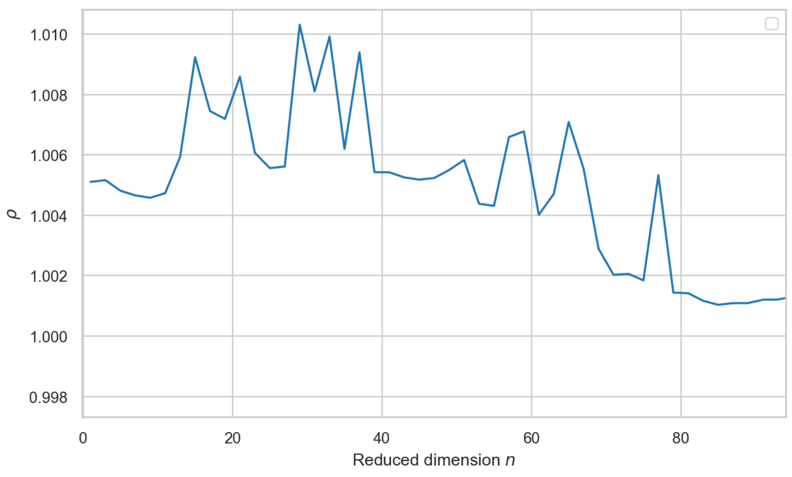


Figure 9: The ratio of approximations of Algorithm of 2 and 4 denoted by $\rho := \frac{d_n^{(2)}(S_N,V_N)}{d_{n,2}^{\otimes(4)}(S_N,V_N)}$ with (i) indicating the approximation of Algorithm i. ($N = 100$)

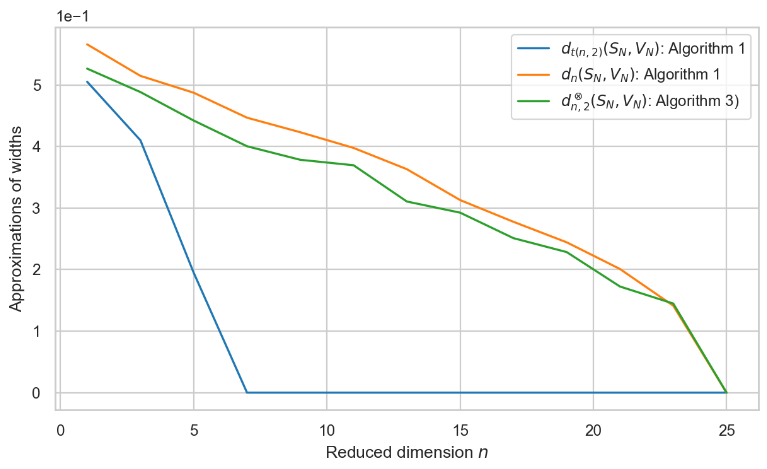


Figure 10: Comparison of the approximation bounds in Theorem 2 using Algorithm 1 and 3. ($N = 25$, $N_s = 10^3$ and $\gamma = 20$)

5 Conclusion and future work

A theoretical approximation bound (lower and upper) can be given for the polynomial Kolmogorov (n, p)-width is terms of the Kolmogorov n-width (see Theorem 2). In this thesis, a numerical investigation is done on these approximation bounds, to investigate how the polynomial Kolmogorov (n, p)-width behaves within these bounds on a test setting. For this purpose, two approaches are presented for estimating the Kolmogorov n-width and polynomial Kolmogorov $(n, 2)$-width. On the test setting (and tested values of N, n, N_s and γ), the estimated polynomial Kolmogorov (n, p)-width decayed closely to the estimated upper bound of the approximation bounds.

The first approach, applied to Algorithm 1 and 3, relies on randomly sampling the basis that defines the ROM. In Algorithm 1, the Kolmogorov n-width is estimated. First, a random basis is sampled and transformed into an orthonormal basis using the SVD. Then, this basis is evaluated using the worst best-approximation error. This process is repeated for N_s samples. The Kolmogorov n-width is estimated by the minimum worst best-approximation error encountered in the samples. Algorithm 3, extends this to the polynomial Kolmogorov $(n, 2)$-width. The basis for the linear term is sampled in the same as in Algorithm 1. Then, another random orthonormal basis is sampled for the quadratic term. This basis is projected on the orthogonal complement of the space spanned by the first basis to ensure orthogonality between the bases. The polynomial Kolmogorov (n, p)-width is again estimated as the minimum over the worst best-approximation errors of the ROMs derived from the bases. The second approach, applied in Algorithm 2 and 4, relies on the POD to find an orthonormal basis that minimizes the least squares reconstruction error. The basis for Algorithm is obtained from the first n left singular vectors of the snapshot matrix containing all coordinate vectors of elements in S. This basis minimizes the sum of least squares error of the reconstruction of S (for ROMs on linear subspaces). After deriving the linear basis, Algorithm 4 constructs the quadratic basis through the sum of least squares minimization of the residual error, that is not captured by the linear basis.

The algorithms are tested on the set of discrete periodic "box functions" contained in the space of left-continuous piecewise constant periodic functions with a period of 1. In this setting, the estimations from Algorithm 2 of the Kolmogorov n-width were significantly lower than the estimations of Algorithm 1, with the exception when the reduced dimension was very close to the full dimension. This was tested up to $N_s = 10^7$ random bases for Algorithm 1 (Table 6), where Algorithm 1's estimation was still higher by a big margin. As the full dimension N increases, Algorithm 1's estimates of the Kolmogorov n-width are higher, relative to the estimations of Algorithm 2 as seen in Table 6. This is presumably because of the increased degrees of freedom in choosing a random basis.

The algorithms for the polynomial Kolmogorov (n,2)-width produced results close to their Kolmogorov n-width counterparts, with Algorithm 2 and 4 showing minimal difference in their reconstruction of box functions. This indicates, that for this problem setting and chosen parameters, the quadratic term has little impact on reducing the approximation error compared to the linear term. Because of this, the results for Algorithm 1 and 2, apply in a similar way for Algorithm 3 and 4.

The four methods were used to investigate the approximation bounds in Theorem 2 numerically. The inequality of Theorem 2 was satisfied by the estimated widths, on the example setting. According to the estimations, the polynomial Kolmogorov $(n, 2)$-width decayed closely to the Kolmogorov n-width. ROMs on quadratically mapped manifold do not perform well on this test testing according to the estimations methods used and the used values of n, N, N_s and γ. There are multiple success cases, where ROMs on quadratically mapped manifolds produced a significantly lower approximation error over linear subspace ROMs, such as in [6] and [8]. Their performance needs to be evaluated on additional test settings, before any conclusions can be made about their general performance. Additionally, ROMs on polynomially mapped manifolds of higher order are yet to be evaluated.

A limitation, when testing the approximation bounds in Theorem 2, is the fact that we estimate the widths. The results are less certain, because there is no upper- or lower bound on the error of our estimations. A comparison is needed, on a test setting, where an exact form of the Kolmogorov n-width is known like in [9], where, an exact form of the kolmogorov $2n$-width is given by $(2\pi m)^{-1}$ for a specific type of Sobolev class of functions. This could provide insight into the accuracy of the Algorithm 2 and potentially Algorithm 4 as well, because of its similarity to Algorithm 2.

Another constraint of this thesis, is the low dimension used for the full dimensional states. In model order reduction, the dimension of the FOMs can be in the order of 10^5 - 10^9 [11]. It is uncertain if the results for the lower dimensions, used in this thesis, translate to higher dimension. With more time and computational resources, future work could be done to investigate this. Improving the execution time of the proposed algorithms can also aid in this.

With slight modifications, Algorithm 1 and 3 can be parallelized to improve their execution time. Each random basis sample, can be evaluated for the worst best-approximation error independently of each other. The results can then be combined back together to obtain an estimate of the width. Especially when done on a GPU, this could massively improve the execution time compared to execution on a single core. Algorithm 2 and 4 do not rely on separate evaluation of bases, and thus cannot be parallelized this way. However, they mostly consist of linear algebra operations that can be done in parallel on a GPU. Matrix multiplication, matrix-vector multiplication and matrix inversion can be done concurrently with libraries such as cuBLAS [4]. The SVD can also be done in parallel on the GPU, using a GPU-optimized library like cuSOLVER [5]. When n and N are large, the matrices used will be large and GPU-accelerated linear algebra operations will be very effective at reducing the execution time. Evaluating the best-approximation error of elements $s \in S$ in Algorithm 2 and 4, can be done independently of each other. This means batches of elements of S can be evaluated in parallel. When implemented, all of these modifications can be used to evaluate the widths for higher values of N, n and N_s in less time.

The time complexity of the proposed algorithms is analysed and summarized in Table 5. Notably, Algorithm 1 has an additional factor in the number of basis samples N_s compared to Algorithm 2.

The decay of Algorithm 1 is shortly mentioned in Figure 2, where a power law is used to fit the decay. Using the horizontal asymptote of this fit, gives similar results to Algorithm 2 in approximating the Kolmogorov n-width. The accuracy and computational feasibility of such methods could be worthy of further investigation.

References

[1] Patrick Buchfink, Silke Glas, and Bernard Haasdonk. "Approximation Bounds for Model Reduction on Polynomially Mapped Manifolds". In: (2023). (Accepted in Comptes Rendus Mathématique https://comptes-rendus.academie-sciences.fr/mathematique). DOI: 10.48550/arXiv.2312.00724.

[2] A. Burian, J. Takala, and M. Ylinen. "A fixed-point implementation of matrix inversion using Cholesky decomposition". In: *2003 46th Midwest Symposium on Circuits and Systems*. Vol. 3. 2003, pp. 1431–1434. DOI: 10.1109/MWSCAS.2003.1562564.

[3] Albert Cohen et al. "Nonlinear compressive reduced basis approximation for PDE's". en. In: *Comptes Rendus. Mécanique* 351.S1 (2023), pp. 357–374. DOI: 10.5802/crmeca.191.

[4] NVIDIA Corporation. *NVIDIA cuBLAS Library Documentation*. https://docs.nvidia.com/cuda/cublas/index.html. 2023.

[5] NVIDIA Corporation. *NVIDIA cuSOLVER Library Documentation*. https://docs.nvidia.com/cuda/cusolver/index.html. 2023.

[6] Rudy Geelen, Stephen Wright, and Karen Willcox. "Operator inference for non-intrusive model reduction with quadratic manifolds". In: *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115717. DOI: 10.1016/j.cma.2022.115717.

[7] G. Golub and W. Kahan. "Calculating the Singular Values and Pseudo-Inverse of a Matrix". In: *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis* 2.2 (1965), pp. 205–224. DOI: 10.1137/0702016.

[8] Shobhit Jain et al. "A quadratic manifold for model order reduction of nonlinear structural dynamics". In: *Computers Structures* 188 (2017), pp. 80–94. DOI: https://doi.org/10.1016/j.compstruc.2017.04.005.

[9] A. Kolmogoroff. "Über die beste Annäherung von Funktionen einer gegebenen Funktionenklasse". In: *Annals of Mathematics* 37.1 (1936). Translated in Michael J. Fischer, *On the Best Approximation of Functions of a Given Class*, available at https://www.mn.uio.no/math/english/people/aca/michaelf/translations/kolmogoroff_english.pdf, pp. 107–110. DOI: 10.2307/1968691.

[10] Allan Pinkus. *n-Widths in Approximation Theory*. Berlin, Heidelberg: Springer, 1985. DOI: 10.1007/978-3-642-69894-1.

[11] Wil Schilders. "Introduction to Model Order Reduction". In: *Model Order Reduction: Theory, Research Aspects and Applications*. Ed. by Wilhelmus H. A. Schilders, Henk A. van der Vorst, and Joost Rommes. Berlin, Heidelberg: Springer, 2008, pp. 3–32. DOI: 10.1007/978-3-540-78841-6_1.

[12] Thomas S. Shores. "GEOMETRICAL ASPECTS OF ABSTRACT SPACES". In: *Applied Linear Algebra and Matrix Analysis*. Ed. by Thomas S. Shores. Cham: Springer International Publishing, 2018, pp. 391–444. DOI: 10.1007/978-3-319-74748-4_6.

[13] Thomas S. Shores. "THE EIGENVALUE PROBLEM". In: *Applied Linear Algebra and Matrix Analysis*. Ed. by Thomas S. Shores. Cham: Springer International Publishing, 2018, pp. 331–390. DOI: 10.1007/978-3-319-74748-4_5.

[14] Lawrence Sirovich. "Turbulence and the dynamics of coherent structures. I. Coherent structures". In: *Quarterly of Applied Mathematics* 45.3 (1987), pp. 561–571. DOI: 10.1090/qam/910462.

A Appendix. Code implementation

This appendix contains implementation details in python, that can be used to reproduce this work. It contains an explanation of the code needed for the implementation of the four algorithms, including the implementations of the algorithms themselves. Source code is available at https://gitlab.utwente.nl/s2840790/thesis-sourcecode.

The following package imports are assumed:

```python
import numpy as np
import scipy.linalg as lalg
import itertools
import math
```

A.1 Creating the test setting

We start with the implementation of the test case defined in Section 4.1. The coordinate vector c_f of a box function $f := b_{a,l}$ in the basis used for V_N is obtained with:

```python
def create_box_function_basis(a, l):
    c = []
    for x in np.arange(0,1,1/N):
        if a + l > 1:
            c.append(1/(N**0.5) if (x >= a or x < (a+l)%1) else 0)
        else:
            c.append(0 if x < a or x >= a + l else 1/(N**0.5))
    return c
```

The coordinate vector for every box function in S_N is obtained as the coordinate vector of $b_{a,l}$ for all possible a and l values.

```python
def create_S_N(N):
    as_ = np.arange(0,1,1/N)
    ls = np.arange(lmin, 1-lmin,1/N)
    cs = []
    for a in as_:
        for l in ls:
            cs.append(create_box_function_basis(a, l))
    return np.array(cs)
```

A.2 Sampling orthonormal bases

The following snippet can be used to generate the orthonormal random basis matrices used in Algorithm 1 and 3.

```python
def sample_linear_and_quadratic_basis(N, n):
    linearBasis = sample_linear_basis(N,n)
    return linearBasis, sample_quadratic_basis(N, n, linearBasis)

def sample_linear_basis(N, n):
    return lalg.orth(np.random.rand(N,n))

def sample_quadratic_basis(N, n, linBasis):
    num_columns = t(n,2) - n - 1
    orth_random_matrix = lalg.orth(np.random.rand(N,num_columns))

    if orth_random_matrix.shape[1] < num_columns: # if the SVD truncated
                                                    the random matrix
```

```
        zeros_padding = np.zeros((N, num_columns - orth_random_matrix.shape
                                                        [1]))
        orth_random_matrix = np.hstack((orth_random_matrix, zeros_padding))

    quadBasis = (np.eye(N) - linBasis @ linBasis.T) @ orth_random_matrix
    return quadBasis
```

A.3 Creating the snapshot matrix

The following snippit creates the snapshot matrix P containing snapshots of all elements in S_N used in Algorithm 2 and 4.

```
def create_snapshot_matrix():
    all_a = np.arange(0,1,1/N)
    all_l = np.arange(lmin, 1-lmin,1/N)
    paramaters = np.array(list(itertools.product(all_a, all_l)))

    num_parameters = paramaters.shape[0]
    snapshot_matrix = np.zeros((N, num_parameters))

    for i in range(num_parameters):
        a,l = paramaters[i]
        coeff = create_box_function_basis(a, l)
        snapshot_matrix[:,i] = coeff

    return snapshot_matrix
```

A.4 Kronecker product

The symmetric Kronecker product K_s of order 2 used in Algorithm 3 and 4.

```
def symmetric_kronecker_product_order_2(x):
    unique_combinations = itertools.combinations_with_replacement(x,2)
    sym_kron2 = np.zeros(math.comb(len(x) + 2 - 1, 2))
    for i,term in enumerate(unique_combinations):
        sym_kron2[i] = math.prod([_ for _ in term])
    return sym_kron2
```

A.5 Algorithm 1

```
def kolmogorov_n_width_random_bases_method(n, N_s, S_N):
    n_width = np.inf
    I_N = np.eye(N)
    for temp in range(N_s):
        basis = sample_random_orthonormal_basis(N, n)
        worst_best_approximation_error = 0
        for x in S_N:
            temp = np.dot(I_N - basis@basis.T, x)
            best_approximation_error = np.dot(temp,temp) ** 0.5
              worst_best_approximation_error = \
            max(worst_best_approximation_error, best_approximation_error)
        n_width = min(n_width, worst_best_approximation_error)
    return n_width
```

A.6 Algorithm 2

```
def kolmogorov_n_width_pod_method(n, S_N):
    n_width = 0
    I_N = np.eye(N)
    snapshot_matrix = create_snapshot_matrix()
    podBasis = lalg.orth(snapshot_matrix)[:, :n]

    for x in S_N:
        temp = np.dot(I_N - podBasis@podBasis.T, x)
        best_approximation_error = np.dot(temp,temp) ** 0.5
        n_width = max(n_width, best_approximation_error)

    return n_width
```

A.7 Algorithm 3

```
def poly_kolmogorov_n2_width_random_bases_method(n, N_s, S_N):
    n2_width = np.inf
    for _ in range(N_s):
        V1,V2 = sample_linear_and_quadratic_basis(N, n)
        worst_best_approximation_error = 0
        for x in S_N:
            reduced_coordinates = np.dot(V1.T,x)
            kron_reduced = \
                symmetric_kronecker_product_order_2(reduced_coordinates)

            x_tilde = np.dot(V1, reduced_coordinates) \
                  + np.dot(V2, kron_reduced)

            diff = x - x_tilde
            best_approximation_error = np.dot(diff, diff) ** 0.5
              worst_best_approximation_error = \
                max(best_approximation_error, \
                worst_best_approximation_error)
        n2_width = min(n2_width, worst_best_approximation_error)
    return n2_width
```

A.8 Algorithm 4

For Algorithm 4, the code for computing the quadratic basis A_2 from the linear POD basis A_1 is included.

```
def poly_kolmogorov_n2_width_extended_pod(n, S, S_N, reg):
    linPodBasis = lalg.orth(S)[:,:n]
    quadPodBasis = create_quadratic_pod_basis(linPodBasis, S, reg)
    n2_width = 0
    for x in S_N:
        reduced_coordinates = np.dot(linPodBasis.T,x)
        kron_reduced = \
          symmetric_kronecker_product_order_2(reduced_coordinates)
        x_tilde = np.dot(linPodBasis, reduced_coordinates) \
        + np.dot(quadPodBasis, kron_reduced)
        diff = x - x_tilde
        best_approximation_error = np.dot(diff, diff) ** 0.5
        n2_width = max(best_approximation_error, n2_width)
    return n2_width
```

```python
def create_quadratic_pod_basis(A1, P, gamma):
    I = np.eye(P.shape[0])
    ER = (I - (A1 @ A1.T)) @ P

    P_HAT = A1.T @ P
    W = create_W(P_HAT)

    WW_T = W @ W.T
    gamma_I = gamma * np.eye(WW_T.shape[0])
    A2 = ER @ W.T @ np.linalg.inv(WW_T + gamma_I)
    return A2

def create_W(P_HAT):
    k = P_HAT.shape[1]
    W = []

    for i in range(k):
        pi_hat = P_HAT[:, i]
        kron = symmetric_kronecker_product_order_2(pi_hat)
        W.append(kron)

    W = np.stack(W, axis=1)
    return W
```