# LEAST SQUARES AND SMOLYAK'S ALGORITHM

JAKOB EGGL, ELIAS MINDLBERGER AND MARIO ULLRICH

Abstract. We present novel, large-scale experiments for polynomial interpolation in high dimensional settings using some of the most popular algorithms available. We compare Smolyak's Algorithm (SA) on sparse grids and Least Squares (LSQ) using random data. We empirically confirm that interpolation using LSQ performs equally as good on smooth and better on non-smooth functions if SA is given $n$ and LSQ is given $2 \cdot n$ points.

## 1. Introduction

Smolyak's algorithm on Sparse Grids has been of high theoretical and practical interest for a long time [2, 5, 13, 19]. In [15], it was recently found that Smolyak's algorithm is *not* optimal in the sampling numbers

$$e_n(H) := \inf_{\substack{x_1,\dots,x_n \in D \\ \varphi_1,\dots,\varphi_n \in L_2}} \sup_{f \in B(H)} \left\| f - \sum_{i=1}^{n} f(x_i)\varphi_i \right\|_{L_2}$$

disproving conjecture 5.26 in [7]. We extend upon these theoretical findings with an empirical study comparing the performance of Smolyak's algorithm and Least Squares on simple function recovery problems on the unit cube. Our implementation is available at `https://github.com/th3lias/NumericalExperiments`.

## 2. Notation

We denote the indexset containing all indices $i \in \mathbb{Z}$ where $m \leq i \leq n$ for $m \leq n$ with $[m : n]$. For the set of polynomials $p$ from $D \subseteq \mathbb{C}^K$, $K \in \mathbb{N}$ to the field $\mathbb{F}$ of maximal degree $N$ we use the notation $p \in \mathcal{P}^N(V, \mathbb{F})$. We use $f \lesssim g$ to denote $f \leq cg$ for functions $f, g$. With $B(X)$ we denote the closed unit ball of a normed space $X$, i.e. $B(X) := \{x \in X : \|x\|_X \leq 1\}$. With $X^\star$ we denote the space of linear, bounded functionals on $X$, i.e. $X^\star := \{x^\star \colon X \to \mathbb{R} \mid x^\star \text{ is linear and bounded}\}$.

*Date*: April 14, 2025.

*Key words and phrases.* Polynomial interpolation, Sparse-Grids, Least-Squares, Smolyak.

## 3. Interpolation

Given $i \in \mathbb{N}$ and distinct data points $\mathbf{X}_i := \{x_{i,0}, \dots, x_{i,n}\} \subseteq D \subseteq \mathbb{R}$, function values $\mathbf{Y}_i := \{y_{i,0}, \dots, y_{i,n}\} = \{f(x_{i,0}), \dots, f(x_{i,n})\} \subseteq \mathbb{R}$ for $f : D_i \to \mathbb{R}$, it is a well known fact that there exists a polynomial $\mathcal{I}(\mathbf{Z}_i)$ of degree $n$ such that

$$\mathcal{I}(\mathbf{Z}_i) = y_{i,j}, \quad j \in [0:n].$$

for $\mathbf{Z}_i := (\mathbf{X}_i, \mathbf{Y}_i)$ We then call $\mathcal{I}(\mathbf{Z}_i)$ an *interpolating* polynomial. It can be written down explicitly as

$$\mathcal{I}(\mathbf{Z}_i) = \sum_{j=0}^{n} y_{i,j} \ell_{i,j} \tag{3.1}$$

for basis functions $\ell_{i,j} \in \mathcal{P}^n(\mathbb{R}, \mathbb{R})$. Moreover, in a single dimension, the formula for this basis is, quite intuitively, $\ell_{i,j}(w) = \prod_{m \in [0:n] \setminus \{j\}} \frac{w - x_{i,m}}{x_{i,j} - x_{i,m}}$ [16, 27]. We call $\mathcal{I}$ the *Lagrange interpolator*.

Suppose now we have $d$ point sets $\mathbf{X}_i$. By taking the full cartesian product of point sets

$$\mathbf{X} := \bigtimes_{i=1}^{d} \mathbf{X}_i,$$

we may identify our grid by $\mathbf{X} = \{\mathbf{x_j}\}_{\mathbf{j}}$ with $\mathbf{j} \in \{0, 1, \dots, n\}^d$ and suppose we have according function evaluations $\mathbf{Y} := \{y_{\mathbf{j}}\}_{\mathbf{j}} = \{f(\mathbf{x_j})\}_{\mathbf{j}}$ for $f : D \to \mathbb{R}$, $D \subseteq \mathbb{R}^d$. Let $\mathbf{Z} := (\mathbf{X}, \mathbf{Y})$. Then, we may construct an interpolating polynomial as in (3.1). Indeed, the general formula stays the same but we take tensor product of basis functions, i.e.

$$\ell_{\mathbf{j}}(\mathbf{w}) = \bigotimes_{i=1}^{d} \ell_{j_i}(w_i) = \prod_{i=1}^{d} \ell_{j_i}(w_i) = \prod_{i=1}^{d} \prod_{\substack{k=0 \\ k \neq j_i}}^{n} \frac{w_i - x_{i,k}}{x_{i,j_i} - x_{i,k}} \tag{3.2}$$

and the sum now ranges over all multiindices $\mathbf{j} = (j_1, \dots, j_d) \in \mathbb{N}_0^d$ with $\mathbf{j} \in \{0, 1, \dots, n\}^d$. We denote

$$\mathcal{I}(\mathbf{Z}) := \sum_{j_1=0}^{n} \cdots \sum_{j_d=0}^{n} y_{j_1,\dots,j_d} \ell_{j_1,\dots,j_d} \tag{3.3}$$

for the interpolant on $\mathbb{R}^d$. Formula 3.2 is not a great choice for performing calculations on hardware. Leaving the storage of $(n+1)^d$ grid points aside, for computing one evaluation of the map $\mathbf{w} \mapsto \mathcal{I}(\mathbf{Z})(\mathbf{w})$, one must compute the sum of $(n+1)^d$ terms, where in each summand, the evaluation of $d$ basis functions is computed, each of which is a product of $n$ terms. This evaluation is thus as costly as $\mathcal{O}((n+1)^d dn)$.

One can counter the costly computation of the interpolating polynomial at a given point by reformulating the above (numerically instable) Lagrange interpolator in it's barycentric form

$$\mathcal{I}(\mathbf{Z})(\mathbf{w}) = \frac{\sum_{j_1=0}^{n} \ell_{j_1}^{(1)}(w_1) \cdots \sum_{j_d=0}^{n} \ell_{j_d}^{(d)}(w_d) y_{\mathbf{j}}}{\sum_{j_1=0}^{n} \ell_{j_1}^{(1)}(w_1) \cdots \sum_{j_d=0}^{n} \ell_{j_d}^{(d)}(w_d)} \tag{3.4}$$

where

$$\ell_{j_k}^{(k)}(w) := \frac{m_{j_k}}{w - x_{j_k}} \mid k \in [1:d]. \tag{3.5}$$

with $x_{j_k}$ being the $j_k$–th entry of $\mathbf{x}$. $m_{j_k}$ are the *barycentric* weights which can be computed in $\mathcal{O}(n^2)$ time for general point distributions but admit $\mathcal{O}(1)$ computation for well–known and widely used point sets [14]. For example, due to [3],

$$m_0 = \frac{1}{2}, \quad m_l = (-1)^l,\, l \in [1:n-1], \quad m_n = \frac{1}{2}(-1)^n \tag{3.6}$$

for the *Chebyshev Gauss-Lobatto* (or just Chebyshev extrema), in which

$$x_l := x_{l,n} := -\cos\frac{l\pi}{n},\, l \in [0:n]. \tag{3.7}$$

Similarly, for the *Chebyshev points of the second kind*

$$x_l = x_{l,n} = \cos\frac{l\pi}{n},\, l \in [0:n] \tag{3.8}$$

one gets the closed form

$$m_l = (-1)^l \cdot \begin{cases} 1/2 & l \in \{0, n\} \\ 1 & \text{else.} \end{cases} \tag{3.9}$$

In this case, one achieves an evaluation time of $\mathcal{O}((n+1)^d)$. Hence, in this construction, one cannot get around this bottleneck.

## 4. Smolyak's Algorithm & Sparse Grids

The following description follows [2]. The question arises, how one may reduce the complexity of exact interpolation on grids. The central idea of sparse grids is to *not* take the full tensor product of one dimensional point sets but restrict the number of simultaneously large directions. To that end, take a variable number of points in each direction, i.e. let $\mathbf{X}_j := \left\{x_{1,j}, \ldots, x_{N(j),j}\right\}$ and $\mathcal{I}_j$ the corresponding one–dimensional interpolator with $\mathcal{I}_0 := 0$, $\Delta_j := \mathcal{I}_j - \mathcal{I}_{j-1}$. Then Smolyak's

algorithm is given in a simple recursive manner

$$\mathcal{A}(q,d) := \sum_{\|\mathbf{j}\|_1 \leq q} \bigotimes_{k=1}^{d} \Delta_{j_k} = \mathcal{A}(q-1,d) + \sum_{\|\mathbf{j}\|_1 = q} \bigotimes_{k=1}^{d} \Delta_{j_k} \tag{4.1}$$

with $\mathcal{A}_{d-1,d} = 0$ and $d \leq q$ and $\mathbf{j}$, $\mathcal{I}$ as before. Evidently, only a relatively small number of knots, through the restriction $\|\mathbf{j}\|_1 \leq q$, is needed. Hence $q$ can be thought of as a resolution parameter. By this form, one only needs to assess the function values at the *sparse grid*

$$H(q,d) := \bigcup_{q-d+1 \leq \|\mathbf{j}\|_1 \leq q} \mathbf{X}_{j_1} \times \cdots \times \mathbf{X}_{j_d}$$

where the number of nodes in a given direction can never be *large* for all directions simultaneously. Hence, given that $\mathbf{X}_j \subset \mathbf{X}_{j+1}$, one may write the interpolator given by Smolyak's construction as

$$\mathcal{A}(q,d)(f) = \sum_{\|\mathbf{j}\|_1 \leq q} f(\mathbf{x}_\mathbf{j}) \ell_\mathbf{j},$$

which is familiar from before. The number of knots $N(j)$ used for each one–dimensional interpolation rule $\mathcal{I}_j$ remains to be specified. In order to obtain nested points, i.e. $\mathbf{X}_j \subset \mathbf{X}_{j+1}$ and thus $H(q,d) \subset H(q+1,d)$ together with collocation rules such as (3.7) or (3.8) it is usual to choose a doubling rule, i.e.

$$N(1) = 1, \quad N(j) = 2^{j-1} + 1,\ j > 1.$$

4.1. **Polynomial Exactness.** Without loss of generality, we restrict ourselves to the symmetric cube $[-1,1]^d$ for interpolation of unkowns instead of a general domain $D$. In this case, Smolyak's algorithm is well known to exactly reproduce functions on certain polynomial spaces given that the rules $\mathcal{I}_j$ are exact on nested spaces $V_j$, see [6] or [18].

**Lemma 4.1.** *Assume $\mathcal{I}_j$ is exact on the vector space $V_j \subseteq C([-1,1])$ and assume*

$$V_1 \subset V_2 \subset V_3 \subset \ldots,$$

*then $\mathcal{A}(q,d)$ is exact on*

$$\sum_{\|j\|_1 = q} V_{j_1} \otimes \cdots \otimes V_{j_d}$$

[2] showed the following.

**Lemma 4.2.** *$\mathcal{A}(q,d)$ is exact on*

$$E(q,d) := \sum_{\|\mathbf{j}\|_1 = q} \mathcal{P}^{m_{j_1}-1}(\mathbb{R},\mathbb{R}) \otimes \cdots \otimes \mathcal{P}^{m_{j_d}-1}(\mathbb{R},\mathbb{R})$$

*and $\mathcal{A}(d+k,d)$ is exact for all polynomials of degree $k$.*

Indeed, the following also follows from [2]. Note that we now abuse the notation from 3.1 by writing $\mathcal{I}(\mathbf{Z}) =: \mathcal{I}(f,\mathbf{X}) =: \mathcal{I}(f)$.

**Lemma 4.3.** *Assume $\mathbf{X}_1 \subset \mathbf{X}_2 \subset \dots$ and $\mathcal{I}_j(f)(x) = f(x)$ for every $f \in C([-1,1])$ and every $x \in \mathbf{X}_j$. Then*

$$\mathcal{A}(q,d)(f)(x) = f(x)$$

*for every $f \in C([-1,1]^d)$ and $x \in H(q,d)$.*

4.2. **Error Bounds.** Since the interpolation operator $\mathcal{I}_j$ as defined before is exact on $\mathcal{P}^{N(j)-1}(\mathbb{R},\mathbb{R})$ one concludes

$$\|f - \mathcal{I}_j(f)\|_\infty \leq \mathrm{err}_{N(j)-1}(f)(1+\Lambda_{N(j)})$$

where $\mathrm{err}_n$ is the error of the best approximation by $p \in \mathcal{P}^n(\mathbb{R},\mathbb{R})$, $\Lambda_n$ is the Lebesgue constant for the point set in (3.7) and $n \geq 2$, in which case it is known that

$$\Lambda_n \leq \frac{2}{\pi}\log(n-1) + 1,$$

see for example [9] and [8]. The following bounds can be found in [2] and is well known since [19, 25, 28].

**Lemma 4.4.** *For the space*

$$F_d^k := \left\{ f : [-1,1]^d \to \mathbb{R} \mid D^\alpha f \text{ continuous if } \alpha_i \leq k \text{ for all } i \right\}$$

*the error of $\mathcal{A}(q,d)$ can be bounded as*

$$\|I_d - \mathcal{A}(q,d)\|_{\mathrm{op}} \leq c_{d,k} n^{-k} (\log n)^{(k+2)(d-1)+1}$$

*where $I_d$ is the embedding $F_d^k \hookrightarrow C([-1,1]^d)$ and $c_{d,k}$ is a positive constant only dependent on $d$ and $k$.*

Moreover, for the Sobolev–Hilbert space $H_w^k$ with

$$H_w^k := \left\{ f \in L_w^2 : \|f\|_k^2 = \sum_{\ell \in \mathbb{N}_0} (1+\ell^2)^k \langle f, b_\ell \rangle^2 < \infty \right\} \tag{4.2}$$

with $\langle f, b_\ell \rangle$ being the $\ell$–th Fourier coefficient, one obtains

$$\|f - \mathcal{A}(q,d)(f)\|_0 \leq c_{d,k} n^{-k} (\log n)^{(k+1)(d-1)} \|f\|_k .$$

Here, $L_w^2$ is the $L^2$ weighted by the Chebyshev weight

$$\left(1 - x^2\right)^{-1/2}. \tag{4.3}$$

In that case, it is well known that the Chebyshev polynomials

$$T_n(x) := \cos(n \arccos(x)). \tag{4.4}$$

form an orthonormal basis.

## 5. Least Squares

Contrary to the construction of exactly interpolating approximants in the case of Smolyak's algorithm, Least Squares is a conceptually simpler algorithm. We are given the overdetermined system

$$A\mathbf{z} = \mathbf{y}$$

where $A \in \mathbb{R}^{n\times m}$ with $n > m$. It is well–known that this system may be inconsistent and no exact solution exists. However, one may always pose the optimisation problem solving for $\mathbf{z} \in \mathbb{R}^m$ with the smallest error

$$\inf_{\mathbf{z}\in\mathbb{R}^m} \|A\mathbf{z} - \mathbf{y}\| . \tag{5.1}$$

It is further known that, in case of a full–rank matrix $V$, the unique solution to (5.1) is given by

$$\mathbf{z}_* = \left(A^\top A\right)^{-1} A^\top \mathbf{y} \in \mathbb{R}^m. \tag{5.2}$$

In our specific case of polynomial interpolation, $A$ is the Vandermonde matrix, consisting of basis polynomials $b_1, b_2, \ldots, b_m$ evaluated at the $n$ different sampled points $x_1, x_2, \ldots, x_n$ in $D \subseteq \mathbb{R}^d$ and $\mathbf{y}$ is the vector of function values sampled from the unknown function $f\colon D \to \mathbb{R}$, i.e. $\mathbf{y} = (f(\mathbf{x}_j))_{j=1}^n$. As for such points, the Vandermonde matrix is never singular, the solution to the approximation problem

$$\inf_p \|f - p\|$$

can analytically be expressed as

$$p_*\colon D \to \mathbb{R}, \quad t \mapsto \sum_{j=1}^m z_{*_j} b_j(t)$$

where $\mathbf{z}_* = \left(z_{*_j}\right)_{j=1}^m$ is given by (5.2). For the later implementation, we choose the basis polynomials $b_j$ as the $j$–th weighted Chebyshev polynomial (4.4).

## 6. Theoretical Guarantees

The notation in the following is borrowed from [26]. In this section we introduce a formal setting to the former considerations. That is, we consider a Hilbert space $H$ of real-valued functions on a set $D$ such that point evaluation $\delta_x : f \mapsto \int_D f \, \mathrm{d}\delta_x = f(x)$ is a bounded, linear functional on $H$. The general formulation of Least Squares allows for a broad class of recovery problems. In our specific case, the function recovery of real–valued functions on a $d$–dimensional (compact) subset $D$ using basis functions of a $k$–dimensional subspace $V_k :=$

span $\{b_1, \dots, b_k\}$, we consider the specific form of Least Squares, given by

$$A_{n,k}(f) := \operatorname*{argmin}_{g \in V_k} \sum_{i=1}^{n} \frac{|g(x_i) - f(x_i)|^2}{\varrho_k(x_i)}$$

where

$$\varrho_k(x) = \frac{1}{2}\left(\frac{1}{k}\sum_{j<k} b_{j+1}(x)^2 + \frac{1}{\sum_{j\geq k} a_j^2}\sum_{j\geq k} a_j^2 b_{j+1}(x)^2\right)$$

and $x_1, \dots, x_n \in D$. Whenever $f \in V_k$, then, of course, $f = A_{n,k}(f)$. With

$$e\left(A_{n,k}, H\right) := \sup_{f \in B(H)} \left\|f - A_{n,k}(f)\right\|_{L_2}, \tag{6.1}$$

we denote the worst case error of $A_{n,k}$, where we measure the error of the reconstruction in the space $L_2 := L_2(D, \Sigma, \mu)$ of square integrable functions on $D$ with respect to the measure $\mu$, such that $H$ is embedded into $L_2$. In light of this, the $n$–th minimal error (also called $n$–th sampling number) is denoted by

$$e_n(H) := \inf_{\substack{x_1,\dots,x_n \in D \\ \varphi_1,\dots,\varphi_n \in L_2}} \sup_{f \in B(H)} \left\|f - \sum_{i=1}^{n} f(x_i)\varphi_i\right\|_{L_2}$$

and can be understood as the worst case error of the optimal algorithm using $n$ function values. We get the clear inequality $e_n(H) \leq e(A_{n,k}, H)$ for any point set $\{x_1, \dots, x_n\}$. With

$$a_n(H) := \inf_{\substack{h_1^\star,\dots,h_n^\star \in H^\star \\ \varphi_1,\dots,\varphi_n \in L_2}} \sup_{f \in B(H)} \left\|f - \sum_{i=1}^{n} h_i^\star(f)\varphi_i\right\|_{L_2}$$

we denote the $n$-th approximation number, which is the worst-case error of an optimal algorithm that uses the $n$ best arbitrary linear and bounded functionals as information about the unknown. This quantity is equal to the $n$-th singular value of the embedding $\mathrm{id}\colon H \to L_2$.

The following is known since [15].

**Theorem 6.1** (Krieg–Ullrich)**.** *There exist constants $C, c > 0$ and a sequence of natural numbers $(k_n)$ with each $k_n \geq cn/\log(n+1)$ and for any $n \in \mathbb{N}$ and measure space $(D, \Sigma, \mu)$, and any RKHS $H$ of real-valued functions on $D$ embedded into $L_2(D, \Sigma, \mu)$, we have*

$$e_n(H) \leq \sqrt{\frac{C}{k_n}\sum_{j \geq k_n} a_j(H)^2}.$$

In particular, for

$$a_n(H) \lesssim n^{-s} \log^{\alpha+s}(n) \tag{6.2}$$

with $s > 1/2, \alpha \in \mathbb{R}$, this implies

$$e_n(H) \lesssim n^{-s} \log^{\alpha+s}(n).$$

The following follows from [26].

**Theorem 6.2** (Ullrich)**.** *Given $n \geq 2$ and $c > 0$, let*

$$k_n := \left\lfloor \frac{n}{2^8(2+c)\log n} \right\rfloor,$$

*then, for any measure space $(D, \Sigma, \mu)$ and any RKHS $H$ of real-valued functions on $D$, embedded into $L_2(D, \Sigma, \mu)$, it holds that*

$$e_n\left(A_{n,2k_n}, H\right) \leq \sqrt{\frac{2}{k_n} \sum_{j>k_n} a_j(H)^2}$$

*with probability at least $1 - 8n^{-c}$.*

**Examples.** In particular, (6.2) is satisfied for the approximation numbers on the Sobolev space of dominating mixed smoothness,

$$\begin{aligned} H &:= H^s_{\mathrm{mix}}\left(\mathbb{T}^d\right) \\ &:= \left\{ f \in L_2\left(\mathbb{T}^d\right) : \|f\|_H^2 := \sum_{m \in \mathbb{N}_0^d} \prod_{j=1}^{d} (1 + |m_j|^{2s}) \left\langle f, b_m \right\rangle_{L_2}^2 < \infty \right\} \end{aligned}$$

with $\mathbb{T}^d \cong [0,1)^d$ where $b_m := \otimes_{j=1}^d b^{(1)}_{m_j}$ and $m = (m_1, \ldots, m_d)$ with

$$b^{(1)}_{2m} := \sqrt{2}\cos(2\pi m x)$$
$$b^{(1)}_{2m-1} := \sqrt{2}\sin(2\pi m x)$$

and $b^{(1)}_0 := 1$. This satisfies the assumption for $s > 1/2$. In particular, we can say

$$e_n\left(H^s_{\mathrm{mix}}\left(\mathbb{T}^d\right)\right) \lesssim n^{-s} \log^{sd}(n) \tag{6.3}$$

whenever $s > 1/2$. This disproves a previously posted conjecture (Conjecture 5.26) in [7] and shows that Smolyak's algorithm is not optimal in this case. The surprising fact is that, despite an optimal, deterministic construction of the point sets used for reconstruction being unknown, random i.i.d. points suffice for a reconstruction error that is on the order of optimal points, with probability tending to 1. We verify this by our experimental findings, presented in Section 7 with a much better relative number of points used for LSQ function recovery

vs. SA recovery than guaranteed in this section, i.e. better constants than explicitly known before. It remains an open problem to rigorously improve upon the constants in (6.3).

## 7. Experimental Findings

For assessing the performance of the Least Squares algorithms in comparison to the Sparse Grid alternative, we use the following 12 families of test functions from [24], each defined of the $d$-dimensional unit-cube $[0,1]^d$.

1. Continuous: $f_1(x) = \exp\left(-\sum_{i=1}^d c_i \left|x_i - w_i\right|\right)$
2. Corner Peak: $f_2(x) = \left(1 + \sum_{i=1}^d c_i x_i\right)^{-(d+1)}$
3. Discontinuous: $f_3(x) = \begin{cases} 0, & x_1 > w_1 \vee x_2 > w_2, \\ \exp\left(\sum_{i=1}^d c_i x_i\right), & \text{else} \end{cases}$
4. Gaussian: $f_4(x) = \exp\left(-\sum_{i=1}^d c_i^2 (x_i - w_i)^2\right)$
5. Oscillatory: $f_5(x) = \cos\left(2\pi w_1 + \sum_{i=1}^d c_i x_i\right)$
6. Product Peak: $f_6(x) = \prod_{i=1}^d \left(c_i^{-2} + (x_i - w_i)^2\right)^{-1}$
7. G-Function: $f_7(x) = \prod_{i=1}^d \frac{|4x_i - 2 - w_i| + c_i}{1 + c_i}$
8. Morokoff & Calfisch 1: $f_8(x) = (1 + 1/d)^d \prod_{i=1}^d (c_i x_i + w_i)^{1/d}$
9. Morokoff & Calfisch 2: $f_9(x) = \frac{1}{(d-0.5)^d} \prod_{i=1}^d (d - c_i x_i + w_i)$
10. Roos & Arnold: $f_{10}(x) = \prod_{i=1}^d |4c_i x_i - 2 - w_i|$
11. Bratley: $f_{11}(x) = \sum_{d=1}^d (-1)^i \prod_{j=1}^d (c_i x_i - w_i)$
12. Zhou: $f_{12}(x) = \frac{10^d}{2}\left[\varphi\left(x - \frac{1}{3}\right) + \varphi\left(x - \frac{2}{3}\right)\right]$
    with $\varphi(x) = \frac{10}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\left\|c\,(x - w)\right\|_2^2\right)$

Note that the first 6 function classes are also known as the *Genz Integrand Families* and were introduced by Genz in [10, 11]. In comparison to the original definition in [24], we also introduce the parameters $c$ and

$w$ in each class by making something like an affine-linear transformation of the input $x$. This allows for testing multiple realizations of these functions.

7.1. **Implementation Details.** Generating a function from a specific family is done by sampling the random vectors $c, w \in \mathbb{R}^d$. In our experiments, we sample each entry of $c$ and $w$ from a uniform distribution $\mathcal{U}(0,1)$ and rescale $c$ afterwards such that $\|c\|_1 = d$.

*Remark* 7.1. In [2], experiments were performed for the Genz families, defined on $[-1,1]^d$. We use $[0,1]^d$ as this ensures that also the other families are well-defined for any sampled $c, w \in \mathbb{R}^d$.

In the following experiments, we compare Smolyak's algorithm with two realizations of the *weighted* Least Squares algorithm. In the first realization we use random points that are uniformly distributed in $[0,1]^d$, and we don't reweigh those points. In the second realization we sample the points from $w(x) = (1-x^2)^{1/2}$, as in 4.3 and we use the value of this density at each point as the basis for the weight calculation. For reproducibility, we used seeding in our random number generation. For actually finding the least–squares solution we employ Numpy [12] with it's lstsq method which uses the gelsd driver from the standard linear algebra package Lapack [1] in the backend. Other drivers are a possible choice but did not meet our performance– and numerical precision requirements. All 3 algorithms use the same basis functions (4.4) which, as mentioned, form an ONB for $L^2_w$. For the implementation of Smolyak's algorithm we employ the standard library Tasmanian with it's Python frontend [17, 20–23]. All Algorithms are tested for all families of functions with multiple (more than 10) realizations and for all dimensions $d \in [2:10]$. In each dimension, the resolution $q \in \mathbb{N}$ was varied. Note that $q > d$. Depending on $d$ smaller or larger values for $q$ were possible because of computational bottlenecks based on the exponential increase in the used points, see also [4] for an overview on the number of points in a sparse grid. Both Least Squares algorithms enjoyed twice the amount of points, compared to Smolyak's algorithm, sampled in their respective distributions.

For assessing the quality of the interpolants, we again generated $n$ random points $x_1, \ldots, x_n$ distributed uniformly in $[0,1]^d$, where $n$ is the number of points in the Sparse Grid. Afterwards we compute the

error quantities

$$e_{\ell_2}(A_j, q, f) := \left( \frac{1}{n} \sum_{i=1}^{n} \left( f(x_i) - \left( A_j(q,d)(f) \right)(x_i) \right)^2 \right)^{1/2}$$
$$e_{\ell_\infty}(A_j, q, f) := \max_{i \in [1:n]} \left| f(x_i) - \left( A_j(q,d)(f) \right)(x_i) \right|$$

for all three algorithms $A_j$ with $j = \{\text{Smolyak}, \text{LS-Uniform}, \text{LS-Chebyshev}\}$ and for all sampled functions $f$ in each family of functions. Those quantities are now depicted in Figures 1 to 3 as a function of $q$. Additionally, **??** summarizes in combination with the different distributions for each function class Figures 4 to 6 the performance of each algorithm based on multiple realizations. The results are also depicted in the following tables and figures.[1]

[1]For a concise overview on our data, we did decide to not depict results for coarse resolutions and dimensions. The reader is invited to try out our implementation for such cases himself.

***Reference to the formulas above, i.e. which formula is Least Squares? Section 5?***

***Write something about the setting in which we tested the functions. Maybe this can and should be combined with all the previous stuff. I.e. talk about the error metrics. Tell them which formulas were used for each algorithm. Some implementation details. Then refer to the figures and also to important tables. Tell them, what scale means (Probably need to adapt the notation of the images, i.e. use $q$ and not scale). Maybe adapt notation to $q$ or whichever character we used before for the description of Smolyak.***

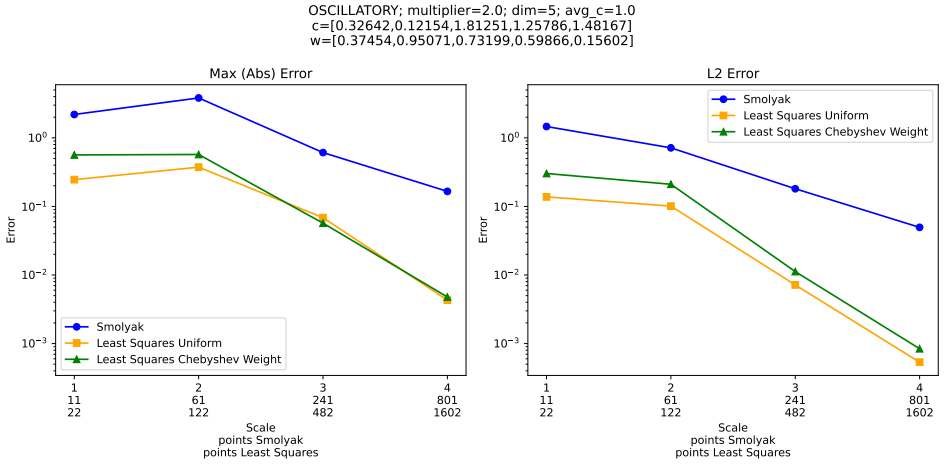


FIGURE 1. Caption

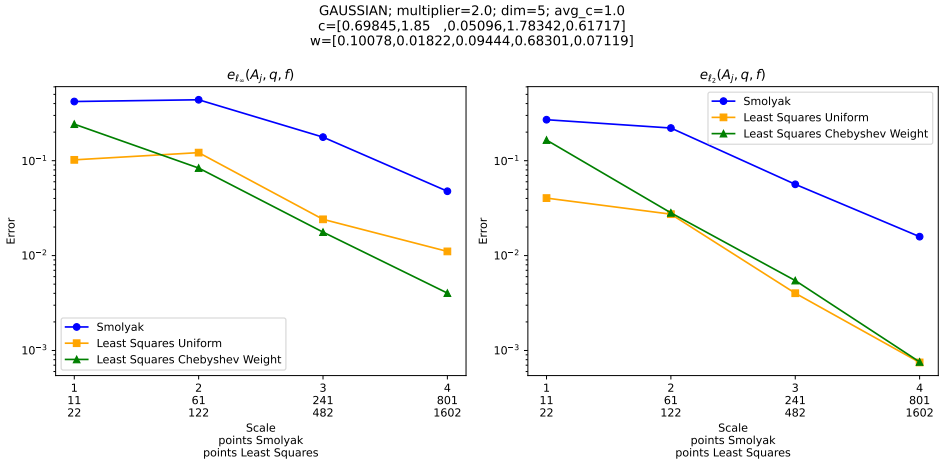


FIGURE 2. Caption

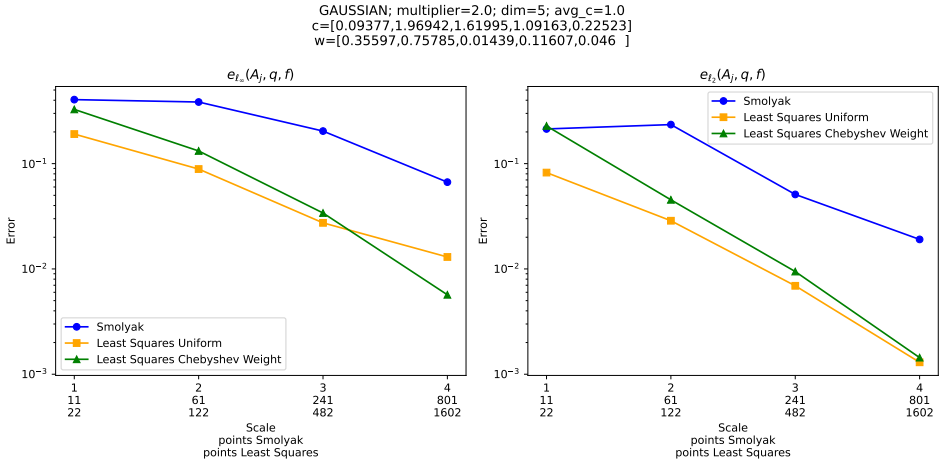


FIGURE 3. Caption

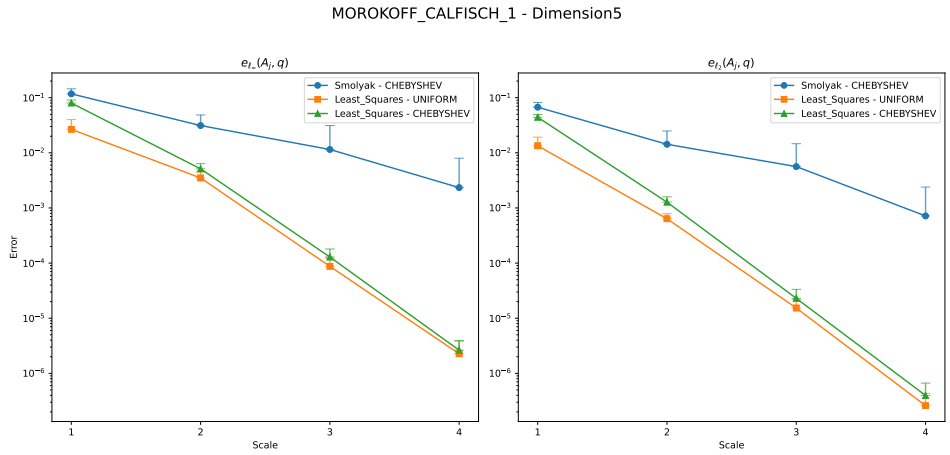


FIGURE 4. Caption

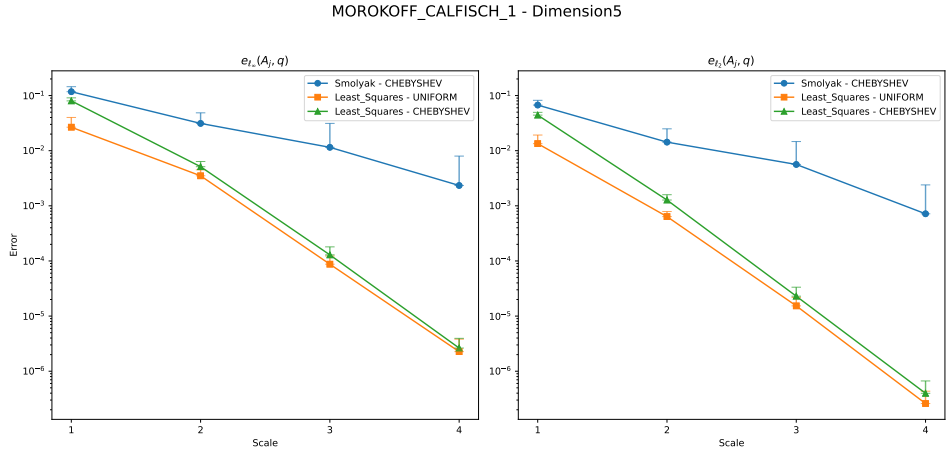


FIGURE 5. Caption

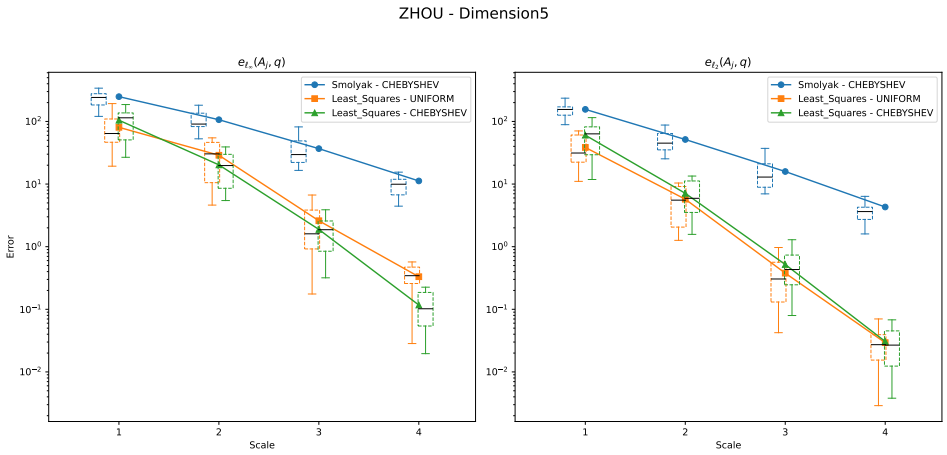


FIGURE 6. Caption

## 8. Conclusion

Ideas

- Same (or usually not a worse—mostly better) order of decay
- $2n$ points seem to suffice compared to the number of the paper
- In some cases, Least Squares outperforms Smolyak a lot
- Tasmainian: Sometimes bad performance: Bad implementation or maybe sometimes Smolyak really bad (Mario: Approxing the 0-Function). People might not be aware of the fact that the approximation quality might be extra-poor
-

## Acknowledgements

## References


[1] E. Anderson et al. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999.

[2] Volker Barthelmann, Erich Novak, and Klaus Ritter. "High dimensional polynomial interpolation on sparse grids". In: *Advances in Computational Mathematics* 12 (2000), pp. 273–288.

[3] Jean-Paul Berrut and Lloyd N Trefethen. "Barycentric lagrange interpolation". In: *SIAM review* 46.3 (2004), pp. 501–517.

[4] John Burkardt. "Counting Abscissas in Sparse Grids". In: (2014).

[5] Chase Coleman and Spencer Lyon. *Efficient Implementations of Smolyak's Algorithm for Function Approximation in Python and Julia*. `https://github.com/EconForge/Smolyak`. 2013.

[6] F.-J. Delvos. "d-Variate Boolean interpolation". In: *Journal of Approximation Theory* 34.2 (1982), pp. 99–114. URL: `https://www.sciencedirect.com/science/article/pii/0021904582900855`.

[7] Dinh Dũng, Vladimir Temlyakov, and Tino Ullrich. *Hyperbolic Cross approximation*. Ed. by Sergey Tikhonov. Birkhäuser, 2018.

[8] V. K. Dzjadyk and V. V. Ivanov. "On asymptotics and estimates for the uniform norms of the Lagrange interpolation polynomials corresponding to the Chebyshev nodal points". In: *Analysis Mathematica* 9.2 (June 1983), pp. 85–97. URL: `https://doi.org/10.1007/BF01982005`.

[9] H. Ehlich and K. Zeller. "Auswertung der Normen von Interpolationsoperatoren". In: *Mathematische Annalen* (June 1966), pp. 105–112.

[10] Alan Genz. "A package for testing multiple integration subroutines". In: *Numerical integration: Recent developments, software and applications* (1987), pp. 337–340.

[11] Alan Genz. "Testing multidimensional integration routines". In: *Proc. of international conference on Tools, methods and languages for scientific and engineering computation*. 1984, pp. 81–94.

[12] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. URL: `https://doi.org/10.1038/s41586-020-2649-2`.

[13] Kenneth L Judd et al. "Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain". In: *Journal of Economic Dynamics and Control* 44 (2014), pp. 92–123.

[14] ANDREAS KLIMKE, KAI WILLNER, and BARBARA WOHLMUTH. "UNCERTAINTY MODELING USING FUZZY ARITHMETIC BASED ON SPARSE GRIDS: APPLICATIONS TO DYNAMIC

SYSTEMS". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 12.06 (2004), pp. 745–759.

[15] David Krieg and Mario Ullrich. "Function Values Are Enough for $L_2$-Approximation". In: *Foundations of Computational Mathematics* 21.4 (Dec. 2020), pp. 1141–1151. URL: http://dx.doi.org/10.1007/s10208-020-09481-w.

[16] Joseph-Louis Lagrange. "Lectures on Elementary Mathematics. On the employment of curves in the solution of problems". Trans. by Thomas J. McCormack. In: (1901). Originally published in French in 1795.

[17] Zack Morrow and Miroslav Stoyanov. "A Method for Dimensionally Adaptive Sparse Trigonometric Interpolation of Periodic Functions". In: *arXiv preprint arXiv:1908.10672* (2019).

[18] Erich Novak and Klaus Ritter. "High Dimensional Integration of Smooth Functions over Cubes". In: *Numerische Mathematik* 75 (Oct. 1996), pp. 79–97.

[19] S. A. Smolyak. "Quadrature and interpolation formulas for tensor products of certain classes of functions". In: *Dokl. Akad. Nauk SSSR* 148.5 (1963), pp. 1042–1045.

[20] M Stoyanov. *User Manual: TASMANIAN Sparse Grids*. Tech. rep. ORNL/TM-2015/596. One Bethel Valley Road, Oak Ridge, TN: Oak Ridge National Laboratory, 2015.

[21] Miroslav Stoyanov. "Adaptive Sparse Grid Construction in a Context of Local Anisotropy and Multiple Hierarchical Parents". In: *Sparse Grids and Applications-Miami 2016*. Springer, 2018, pp. 175–199.

[22] Miroslav Stoyanov et al. *Tasmanian*. Sept. 2013. URL: https://github.com/ORNL/Tasmanian.

[23] Miroslav K Stoyanov and Clayton G Webster. "A dynamically adaptive sparse grids method for quasi-optimal interpolation of multidimensional functions". In: *Computers & Mathematics with Applications* 71.11 (2016), pp. 2449–2465.

[24] S. Surjanovic and D. Bingham. *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Retrieved March 29, 2025, from https://www.sfu.ca/~ssurjano/integration.html.

[25] V. N. Temlyakov. "Approximation of periodic functions of several variables by trigonometric polynomials, and widths of some classes of functions". In: *Math. USSR-Izv.* 27.2 (1986), pp. 285–322.

[26] Mario Ullrich. "On the worst-case error of least squares algorithms for $L_2$-approximation with high probability". In: *Journal of Complexity* 60 (Oct. 2020), p. 101484. URL: http://dx.doi.org/10.1016/j.jco.2020.101484.

[27] Edward Waring. "Problems concerning Interpolations. By Edward Waring, M. D. F. R. S. and of the Institute of Bononia, Lucasian Professor of Mathematics in the University of Cambridge". In: *Philosophical Transactions of the Royal Society of London* 69 (1779), pp. 59–67. URL: http://www.jstor.org/stable/106408 (visited on 04/11/2025).

[28] G.W. Wasilkowski and H. Wozniakowski. "Explicit Cost Bounds of Algorithms for Multivariate Tensor Product Problems". In: *Journal of Complexity* 11.1 (1995), pp. 1–56. URL: https://www.sciencedirect.com/science/article/pii/S0885064X85710011.

J.E., Johannes Kepler University Linz; jakob.eggl@jku.at;
E.M., Johannes Kepler University Linz; elias.mindlberger@jku.at;
M.U., Johannes Kepler University Linz; mario.ullrich@jku.at