Approximating Functions Using Epsilon-Distinguishable Sets

Alice Feng

August 2024

Abstract

Traditional function approximation methods approximate functions on sets such as the Chebyshev grid, Monte Carlo set, quasi-Monte Carlo sets, etc. These methods usually assumes a simple geometry such as hypercube. However, in practice we may need to approximate functions on arbitrary geometries. Moreover, sets such as Monte-Carlo sets are only ex-ante uniform, but do not possess good uniformity ex-post. In this paper, we use the concept "ϵ-distinguishable" sets(EDS), which are pre-computed good uniform sets that are geometry-independent and good for approximating function at any sample size. Experiments in this paper show that EDS can compete with widely used sets such as the Chebyshev grid, Monte Carlo set, quasi-Monte Carlo sets, etc.. Function approximation using EDS can greatly reduce the number of function evaluations needed when the target function is costly to evaluate.

1 Introduction

Given a C^2 function $f : A \to \mathbf{R}$, where A$\subset \mathbf{R^n}$ is a compact subset (called the "domain"), we want to choose a finite set of points $X \subset \mathbf{R^n}$ so that we can construct a surrogate function $\hat{f}(\cdot, X, f(X))$ where the maximum approximation error $\|\hat{f} - f\|_\infty$ is minimized.

Existing sets for function approximation can be divided into 4 categories: tensor product grid (including uniform grid, Chebyshev grid, etc), Monte-Carlo Grid, low discrepancy sets and Monte-Carlo Latin Hypercube. It is well known that each category of set comes with their disadvantages.

We first introduce some measures of a datasets quality

Separation distance:
We define the separation distance of the set as

$$q = \frac{1}{2} \min_{x_i \neq x_j \in X} \|x_i - x_j\|_2$$

Separation distance measures the minimum distance between data points.

Fill distance:
The fill distance of X over the domain A is

$$h_{X,A} = \max_{x \in A} \min_{x_i \in X} \|x - x_i\|_2$$

which is the radius of largest ball inside the domain A having no points from X.

Discrepancy:
We define the L_∞discrepancy of a set as the following:

$$discr(X) = \sup_B |\frac{C(B;X)}{|X|} - \mu(B)|$$

where $|X|$ is the size of the set, $C(B;X) = |B \cap X|$ is the number of points in X that fall inside the box B. (B is the tensor product of intervals of form $[a_i, b_i), i = 1...d)$.
Kuipers, L.; Niederreiter, H. (1974). Uniform Distribution of Sequences. John Wiley & Sons Inc. Page 93

Note that the computation of discrepancy using the formula above can be very costly. In this paper I will use the L2-Star discrepancy for computation, which is defined as

$$L_2^* discr(X) = (\int_{[0,1]^d} (\frac{C(B_x;X)}{|X|} - \mu(B_x))^2 dx)^{\frac{1}{2}}$$

Where the box $B_x = \prod_{i=1}^{d} [0, x_i)$ [1]

1.1 Curse of dimensionality

For tensor product grids, their sizes grow exponentially with dimensions. If we use n data points along each dimension, then the size of the grid is n^d, where d is the dimension of the grid. The curse of dimensional imposes computational difficult on approximating function on grids such as uniform grid, Chebyshev tensor grid, etc..

1.2 Data clustering

Due to the random nature of Monte-Carlo sets, there is some unnecessary clustering of points. Formally, When data clustering occur, the separation distance becomes very small. When we are doing interpolation, low separation distance often means the condition number of the interpolation matrix is large, causing numerical instability. Moreover, data clustering also increases the number of data points needed to cover the domain. As we will show later, an EDS displays no clustering of points.

1.3 Empty region of Monte-Carlo sets

Another issue with Monte-Carlo sets are the existence of empty region in the set. This phenomenon can be quantified through the size of the fill distance. Figure 1 illustrates the data clusters and empty regions of a two dimensional Monte-Carlo.
Make notation on the graph showing the large hole.

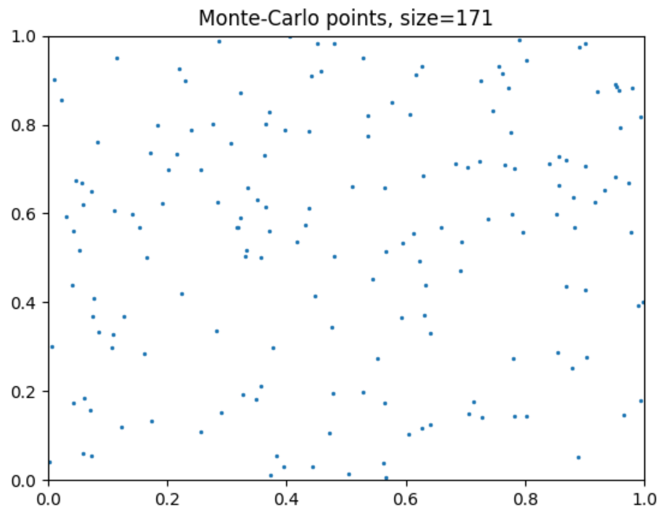


Figure 1

1.4 Small Sample nonuniformity of low discrepancy sets

Low discrepancy sets such as Halton and Sobol (a.k.a quasi Monte-Carlo set) are extensively used in numerical intergration and function approximation.

Example: Halton Sequence
The Halton Sequence of dimension d is constructed in this way:
1. Choose a sequence of mutually prime numbers (called the ”base”) $\{b_i\}$ of size d.
2. Let the 0-th Halton point to be

$$y_0 = (0, ..., 0)$$

3. For $n \geq 1$, the i-th coordinate of the n-th Halton point is computed as

$$y_{ni} = \sum_{j=1}^{J_{ni}} a_{nij} b_i^{-j}$$

where

$$a_{nij} = n \mod b_i^{j-1}$$

and

$$J_{ni} = 1 + \log_{b_i} n$$

Figure 2 shows an example of Halton sequence in the first 2 dimensions.

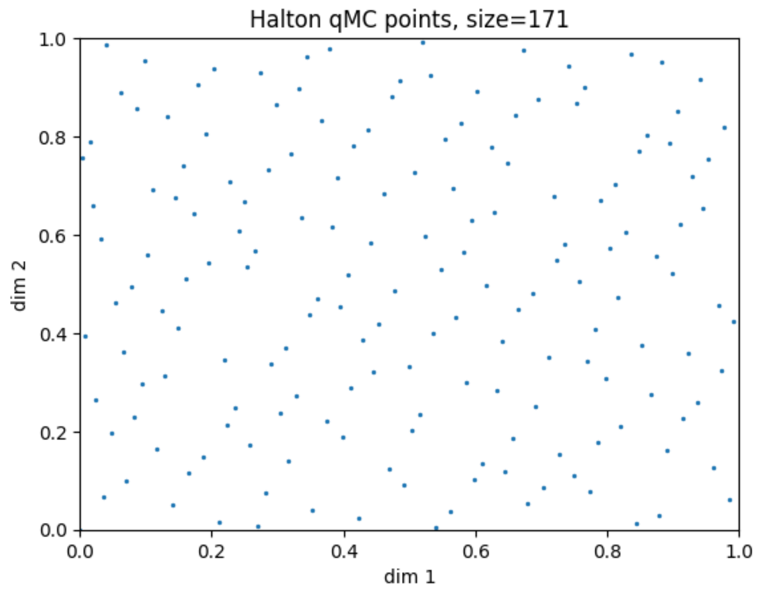


Figure 2

However, low discrepancy are only "balanced" in certain sizes. For example, Sobol sequence are only good when their size are powers of 2. Quasi-Monte carlo shows significant imbalance in some projected dimensions when the dimension is high.

Figure 3, 4 and 5 shows how halton can behave badly in higher dimensions.

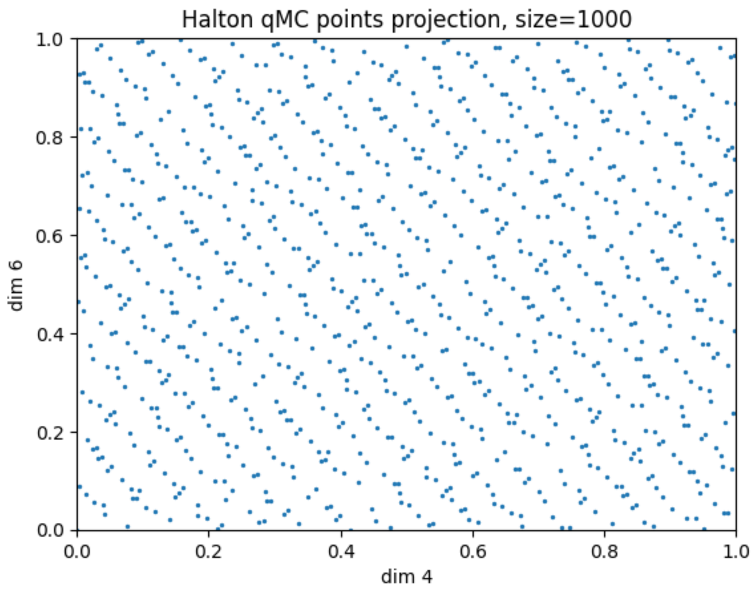


Figure 3

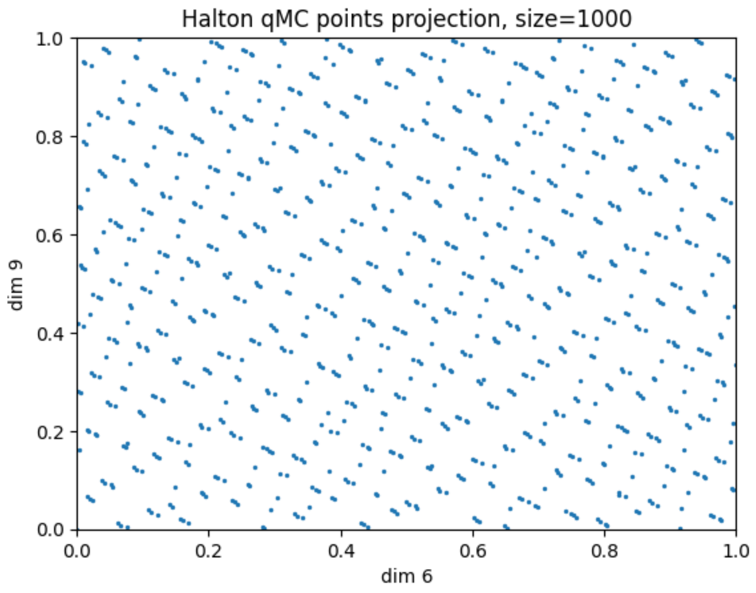


Figure 4

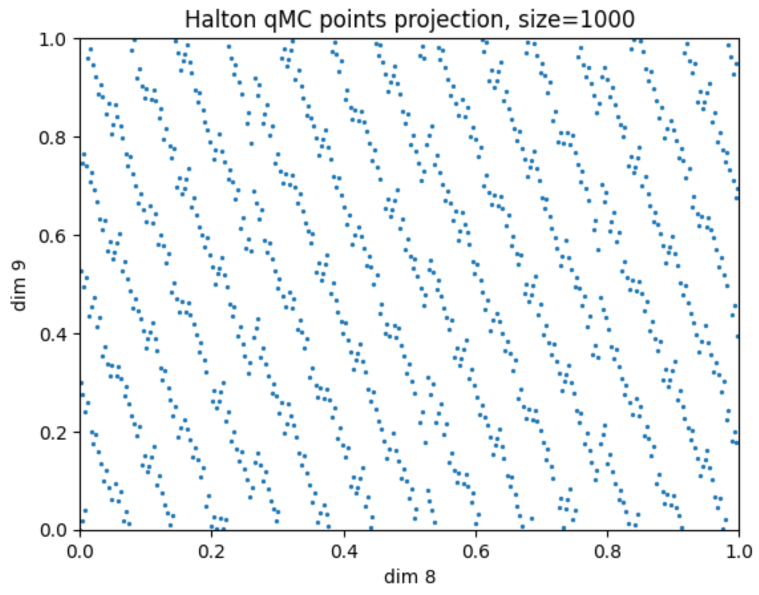


Figure 5

1.5 Monte-Carlo Latin Hypercube

The Latin Hypercube method divide the coordinate of each dimension into small subintervals, and require that no two point's projection on any coordinate fall into the same subinterval. (This definition is wrong, need to find out the correct one)
Figure 6 shows a Monte-Carlo Latin-hypercube in 2 dimension.

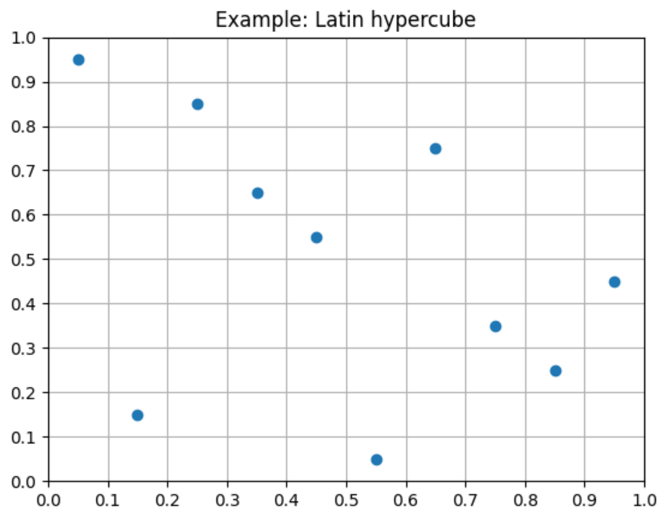


Figure 6

The Latin-hypercube has good uniformity along the projection on each axis, which is an advantage over low discrepancy points. However, being uniform on each dimension does not mean good uniformity over the entire space.

1.6 Epsilon Distinguishable Set (EDS)

Definition 1: Let Ω to be a metric space where we have the metric function $d : \Omega \times \Omega \to \mathbf{R}$ defined on it. The finite subset $X_\epsilon \subseteq \Omega$ is said to be ϵ-distinguishable iff for all x,y$\in X_\epsilon$

$$d(x, y) \geq \epsilon$$

In this paper we will focus on the construction and analysis of EDS in $\mathbf{R^n}$ where n is a finite number.

EDS is related to what is defined as the separation distance. An ϵ–distinguishable set has separation distance of at least $\frac{\epsilon}{2}$

Definition 2: We say an ϵ-distinguishable set (EDS) X_ϵ is a fully filled EDS iff its fill distance satisfies

$$h_{X_\epsilon,A} < \epsilon$$

In this paper, when we are referring to EDS, we always mean fully filled EDS.

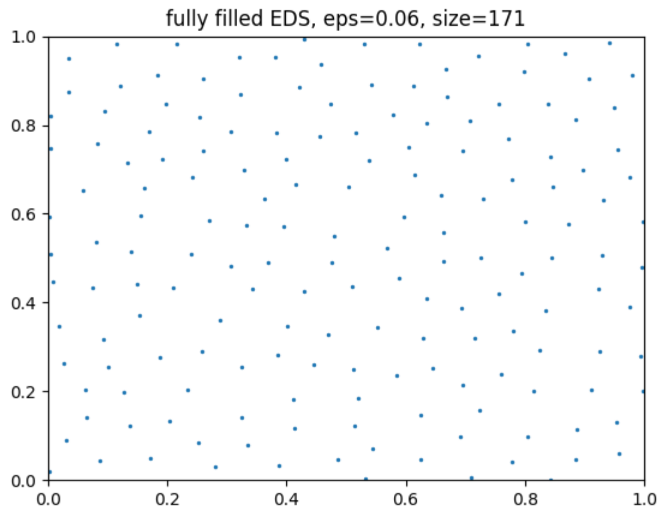
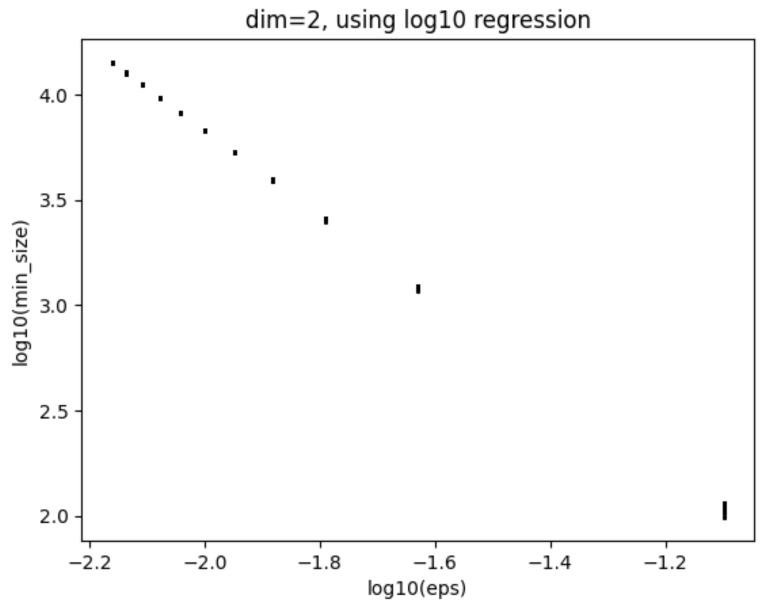


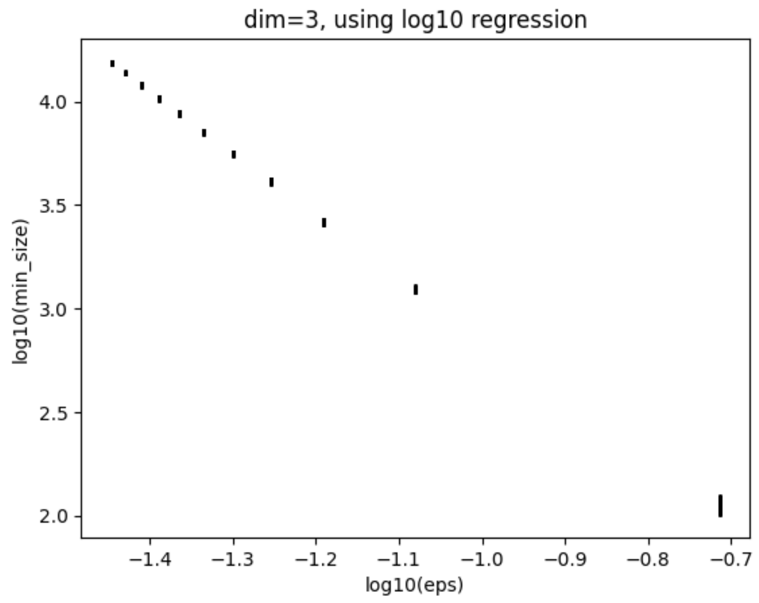
Figure 7

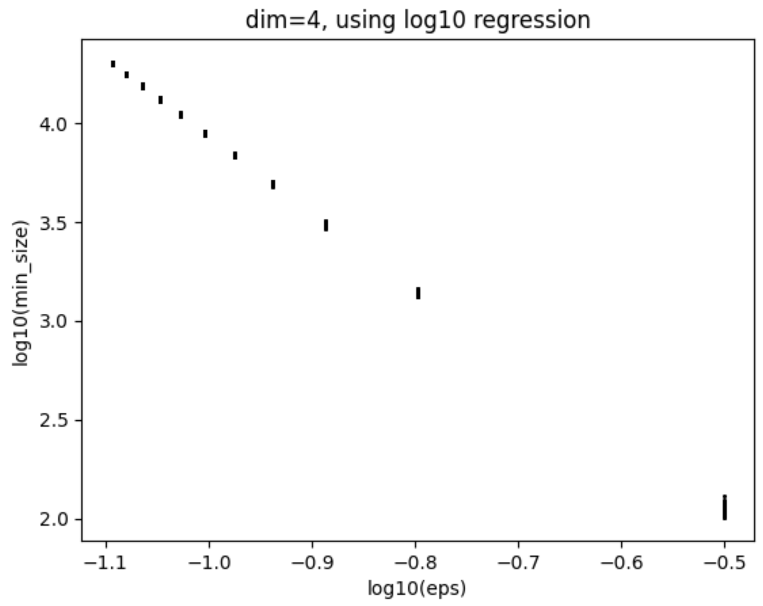
1.7 min-size EDS

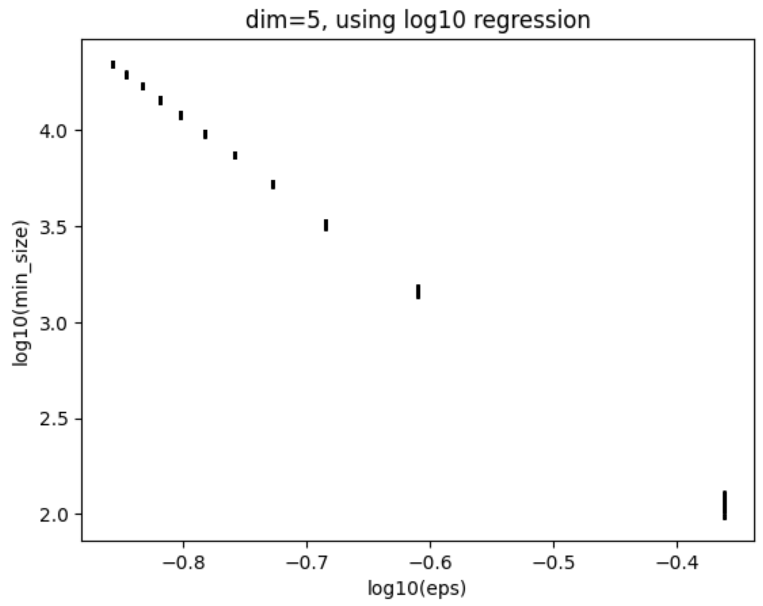
If we fix an ϵ, the corresponding EDS over a domain is not unique. Sometimes we may want to find the fully-filled EDS of minimum size. Unfortunately, it

is computationally intractable to find out the min-size EDS. Alternatively, we can find an approximately min-size EDS by trying different random seeds during the generation process(see next section for detail), and use the smallest set found.
The following plots show that the difference in sizes of fully-filled EDS are small when we are doing 100 repeated runs.






dim=4, using log10 regression
log10(min_size)
log10(eps)
4.0
3.5
3.0
2.5
2.0
−1.1
−1.0
−0.9
−0.8
−0.7
−0.6
−0.5


dim=5, using log10 regression
log10(min_size)
log10(eps)
4.0
3.5
3.0
2.5
2.0
−0.8
−0.7
−0.6
−0.5
−0.4

2 Generating EDS

Following is a simple algorithm to generate EDS

Algorithm gen_EDS

```
1: Generate a random point $x_0$ in the domain A
2: eds_set← $\{x_0\}$
3: n_fails←0
4: while n_fails≤n_fails_max do
5:     Generate a random point $\hat{x}$ in the domain A
6:     add← True
7:     for $x_i \in$ eds_set do
8:         if $\|x_i - \hat{x}\| < \epsilon$ then
9:             add←False
10:            n_fails← n_fails+1
11:            break
12:        end if
13:    end for
14:    if add=True then
15:        eds_set←eds_set$\cup\{\hat{x}\}$
16:        n_fail← 0
17:    end if
18: end while
19: return eds_set
```

2.1 Probability test analysis of the stopping criteria

The algorithm stops when we examined n_fail consecutive random points and conclude they are all too close to some existing point. We notice that if there exists an empty ball of radius $\alpha\epsilon$ ($\alpha > 1$), then any random point that fall into the inner circle of radius $(\alpha - 1)\epsilon$ would have been added to the EDS. Thus the volume of the space where we can add an addition point is at least $V_n\epsilon^n(\alpha-1)^n$, where n is the dimension and V_n is the volume of n-dimensional ball of radius 1. (Note that points that falls into the outer circle may also be added to EDS, that's why we use "at least".) Therefore we have

$$Pr(h_{X,A} \geq \alpha\epsilon) = (1 - \frac{\mu(B_{(\alpha-1)\epsilon})}{\mu(A)})^{n\_fails\_max} = (1 - \frac{V_n\epsilon^n(\alpha-1)^n}{\mu(A)})^{n\_fails\_max}.$$

When $(\alpha - 1)\epsilon$ is sufficiently small (e.g., $(\alpha - 1)^n\epsilon^n$), we have

$$Pr(h_{X,A} \geq \alpha\epsilon) \approx e^{-\frac{V_n(\alpha-1)^n\epsilon^n}{\mu(A)}n\_fails\_max}.$$

In practice, we usually use some heuristics to determine the stopping criteria. For the rest of the paper, I will set the maximum fail attempts to be

$\max(1000, c|X|)$, where c is some constant and $|X|$ is the size of the EDS. The intuition behind this is that when ϵ is already large, we want α to be smaller so that the fill distance does not become too large.

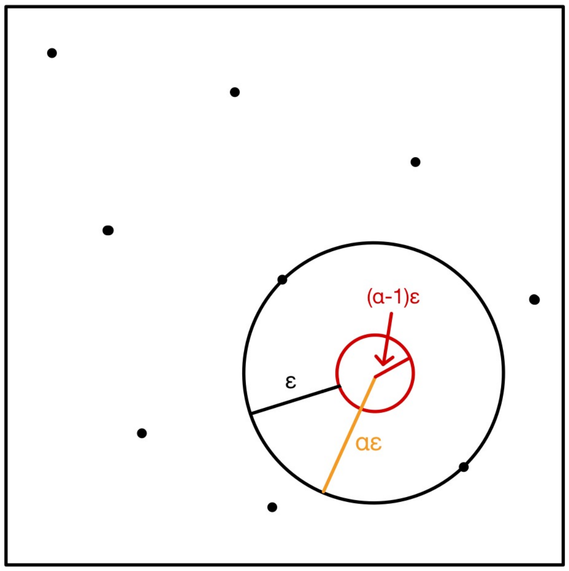


Figure 8: The algorithm will add a random point to the EDS if it falls into the red inner circle

2.1.1 Example: EDS in 5 dimensional unit hypercube

In a five dimensional unit hypercube, we have $V_5 = \frac{8\pi}{15}$. Therefore we have

$$Pr(h_{X,A} \geq \alpha\epsilon) = (1 - \frac{8\pi(\alpha-1)^5\epsilon^5}{15})^{n\_fails\_max} \approx e^{-\frac{8\pi(\alpha-1)^5\epsilon^5}{15}n\_fails\_max}$$

If we require $Pr(h_{X,A} \geq (\alpha-1)\epsilon) = 0.01$, then we have the following relationship

$$n\_fail = -\frac{15}{8\pi(\alpha-1)^5\epsilon^5}ln(0.01) \approx \frac{2.7485}{(\alpha-1)^5\epsilon^5}$$

where $\alpha\epsilon$ is the fill distance.

Alternatively, can fix n_fail_max and express the probability in terms of fill distance. For example, if we fix n_fail_max to be 1000, then we have following results (see the figures)

2.2 Properties of EDS

In addition to fill distance and separation distance, we can also recall the definition of discrepancy: When we fix the size of the data set (the table below uses various two dimensional sets of size 171), the EDS has a small fill distance and a large separation distance.

Grid type	Fill distance	Separation distance	L2-Star discrepancy
EDS	0.0592	0.0300	0.0131
Monte-Carlo	0.0997	0.0004	0.0270
Halton qMC	0.1009	0.0175	0.0103
Latin-Hypercube	0.1059	0.0083	0.0119

2.3 Estimation of ϵ

If we have a target size for EDS that we want to generate, we need to estimate the ϵ used in the generation process. Empirically, the relationship between EDS size n and ϵ satisfies

$$log(n) = alog(\epsilon) + b \qquad a, b \in \mathbf{R}$$

where a,b are some constants dependent on the dimension and the shape of the domain.

The following display the relationship of size and ϵ in a 5-dimensional hypercube. The log-linear relationship is obvious.

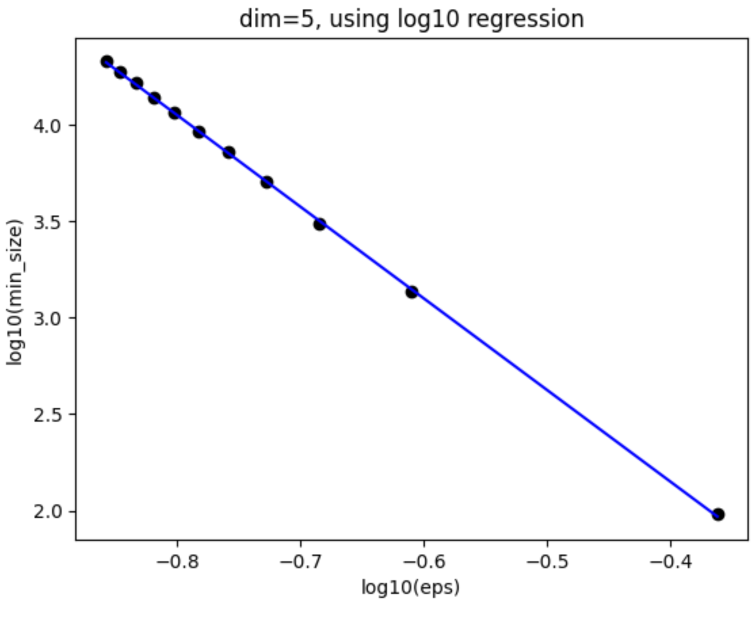


Figure 9: a=-5.59, b=0.50

Therefore, the following algorithm can generate an EDS close to the target size.

2.4 EDS and probabilities

We can use EDS to estimate probabilities of a point being in a set.

3 EDS and approx methods

3.1 Radial function interpolation

The simplest form of radial basis function interpolation is interpolation with a universal scale factor. The interpolants are of the form

$$\hat{f}(x) = \sum_{i=1}^{n} a_i \phi(\frac{\|x - x_i\|}{c})$$

where c is sometimes called "width/scale/..." of the RBF interpolant. According to [2], for the "homogeneously" scaled case, the error over domain A is given by

$$\|f - \hat{f}\|_{L_\infty(A)} \leq C h_{X,A}^{\beta - d/2} \|f\|_{W_2^\beta(A)}$$

Where C is some constant, $h_{X,A}$ being the fill distance and W_2^β is the Sobolev norm.

However, the homogeneously scaled RBF interpolation suffer from the Runge phenomenon and the ill-condition-ness. One way to overcome this disadvantage is to use the "spacially variably scaled RBF"[3] which takes the form

$$\hat{f}(x) = \sum_{i=1}^{n} a_i \phi(\frac{\|x - x_i\|}{c_i})$$

where c_i is different for each RBF "center". The Fornberg paper suggests a formula for finding the c

$$c_i = \frac{c}{d_i}$$

where

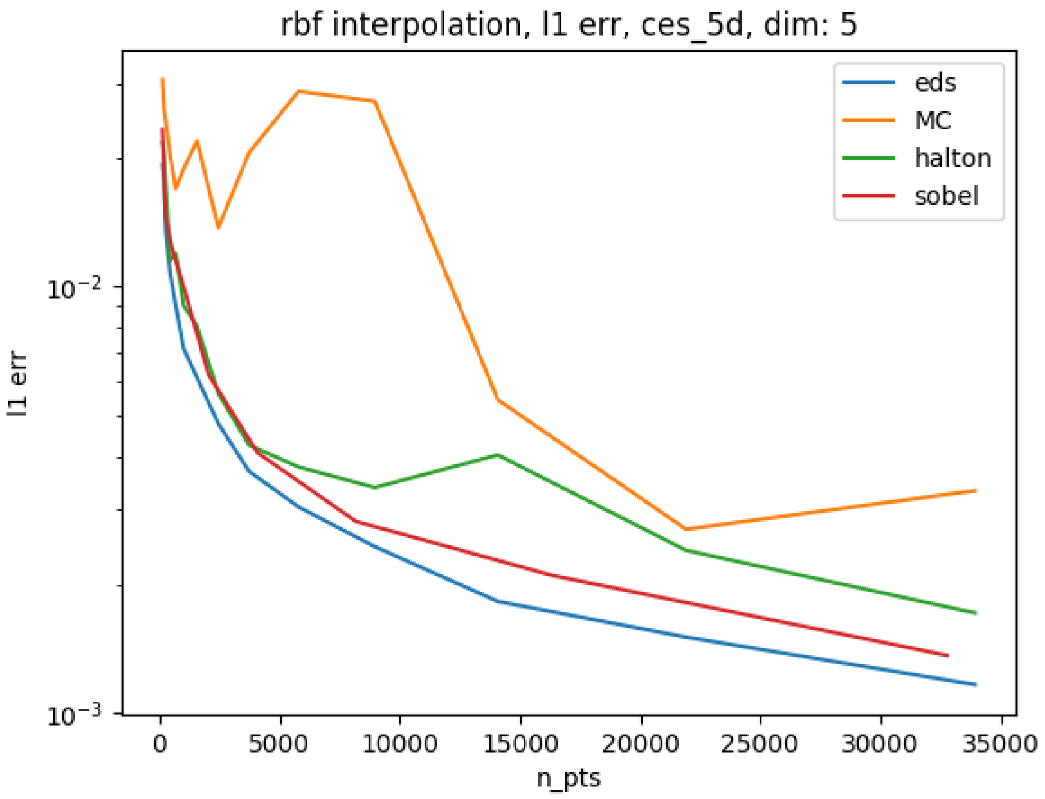
$$d_i = \min_{j \neq i} \|x_j - x_i\|$$

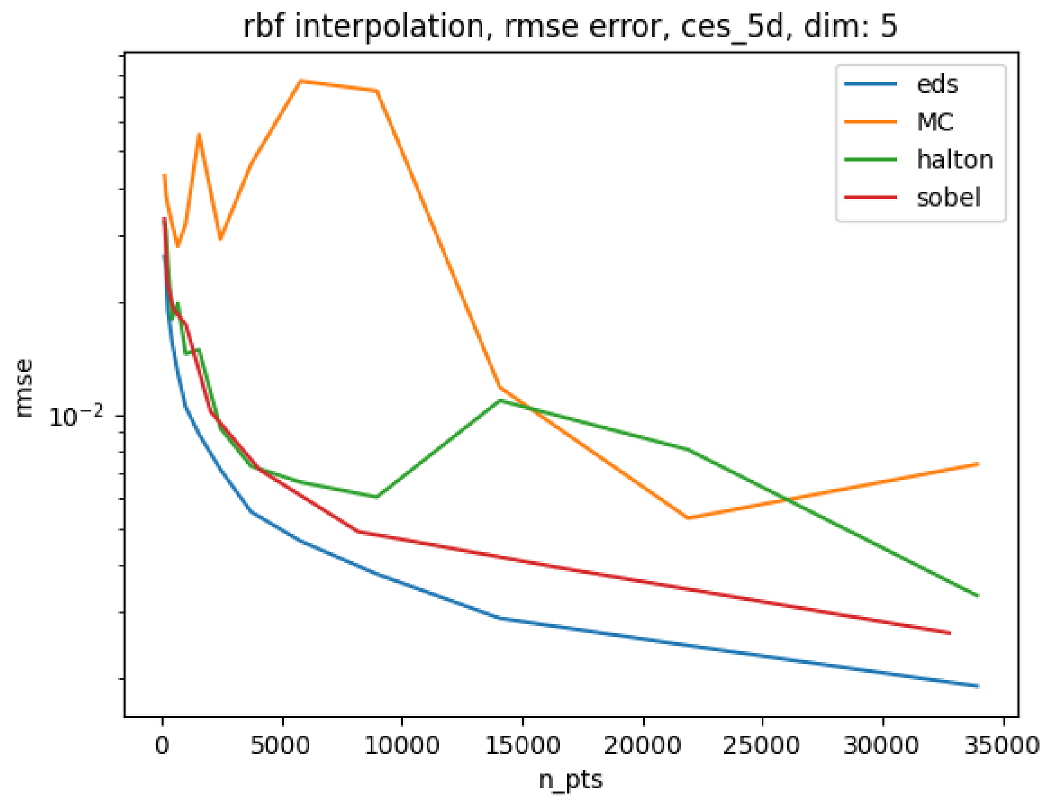
My RBF experiments are all implemented using the above formula. The theories behind "spatially variably scaled RBFs" are based on differential geometry and are beyond the scope of this paper.[4]

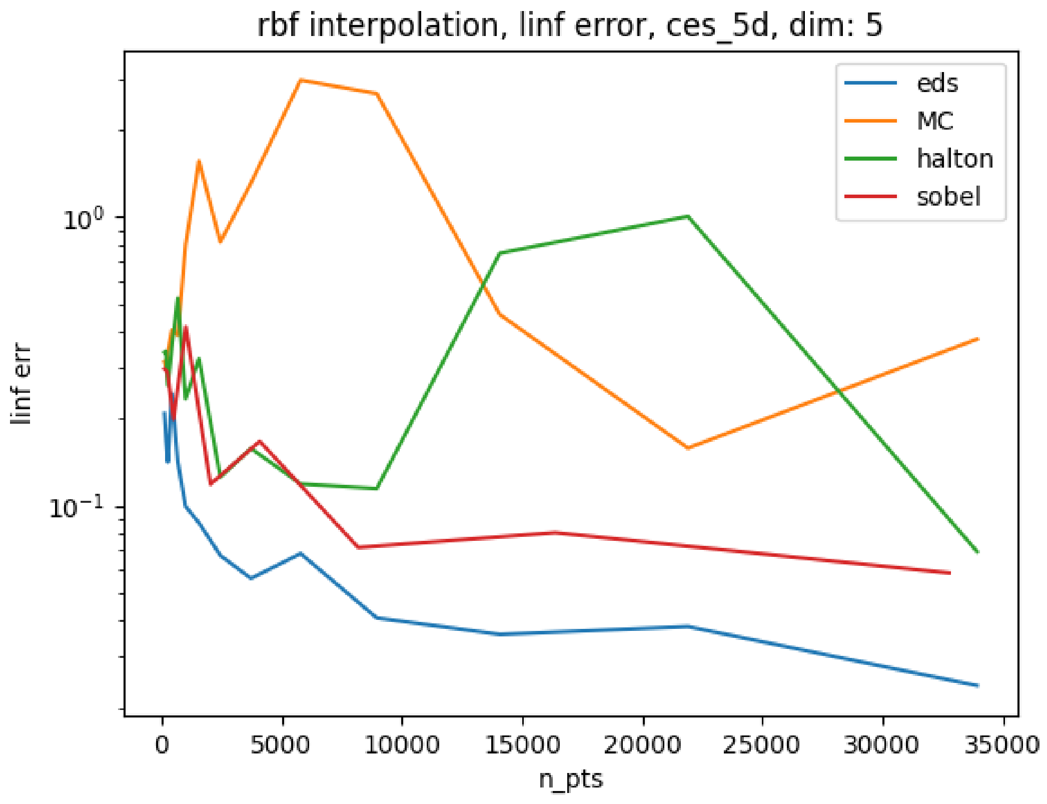
The graphs below showed the results of approximation of 5-dimensional CES function.

$$y = (\sum_{i=1}^{d} a_i x_i^r)^{\frac{1}{r}}$$

Where we use r=-0.3 and let $\{a_i\}$ to be randomly chosen values that somes up to 1. The domain of approximation is $[0.1, 5]^5$




rbf interpolation, rmse error, ces_5d, dim: 5
eds
MC
halton
sobel
rmse
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000
n_pts


rbf interpolation, linf error, ces_5d, dim: 5
eds
MC
halton
sobel
linf err
10^{0}
10^{-1}
0
5000
10000
15000
20000
25000
30000
35000
n_pts

3.2 Results using chebyshev regression

Chebyshev complete polynomial L1/L2/Linf regression. (Add definition of chebyshev polynomial?) We let α to be a multi-index with $|\alpha| \leq deg$ (we use $||$ to denote the some of all entries of α), then the formula for L1 regression (without regularization) is

$$\min_{\mu_i, a_\alpha} \sum_i \mu_i$$

subject to

$$-\mu_i \leq \hat{f}(x_i, \{a_\alpha\}) - ytrue_i \leq \mu_i$$
$$\mu_i \geq 0$$
$$i = 1, 2, ..., n\_data, |\alpha| \leq deg$$

Formula for L-inf regression (without regularization) is

$$\min_{\mu, a_\alpha} \mu$$

subject to

$$-\mu \leq \hat{f}(x_i, \{a_\alpha\}) - ytrue_i \leq \mu$$
$$\mu \geq 0$$
$$i = 1, 2, ..., n\_data, |\alpha| \leq deg$$

If we add regularization for terms that satisfy

$$deg_0 \leq |\alpha| \leq deg,$$

the formula for L1 regression will become,

$$\min_{\mu_i, a_\alpha} \sum_i \mu_i + p \sum_{deg_0 \leq |\alpha| \leq deg} |\alpha| a_\alpha^+$$

subject to

$$-\mu_i \leq \hat{f}(x_i, \{a_\alpha\}) - ytrue_i \leq \mu_i$$
$$-a_\alpha^+ \leq a_\alpha \leq a_\alpha^+$$
$$\mu_i \geq 0$$
$$a_\alpha^+ \geq 0$$
$$i = 1, 2, ..., n\_data, |\alpha| \leq deg$$

and the formula for L-inf regression with regularization will become

$$\min_{\mu_i, a_\alpha} \mu + p \sum_{deg_0 \leq |\alpha| \leq deg} |\alpha| a_\alpha^+$$

subject to

$$-\mu \leq \hat{f}(x_i, \{a_\alpha\}) - ytrue_i \leq \mu$$

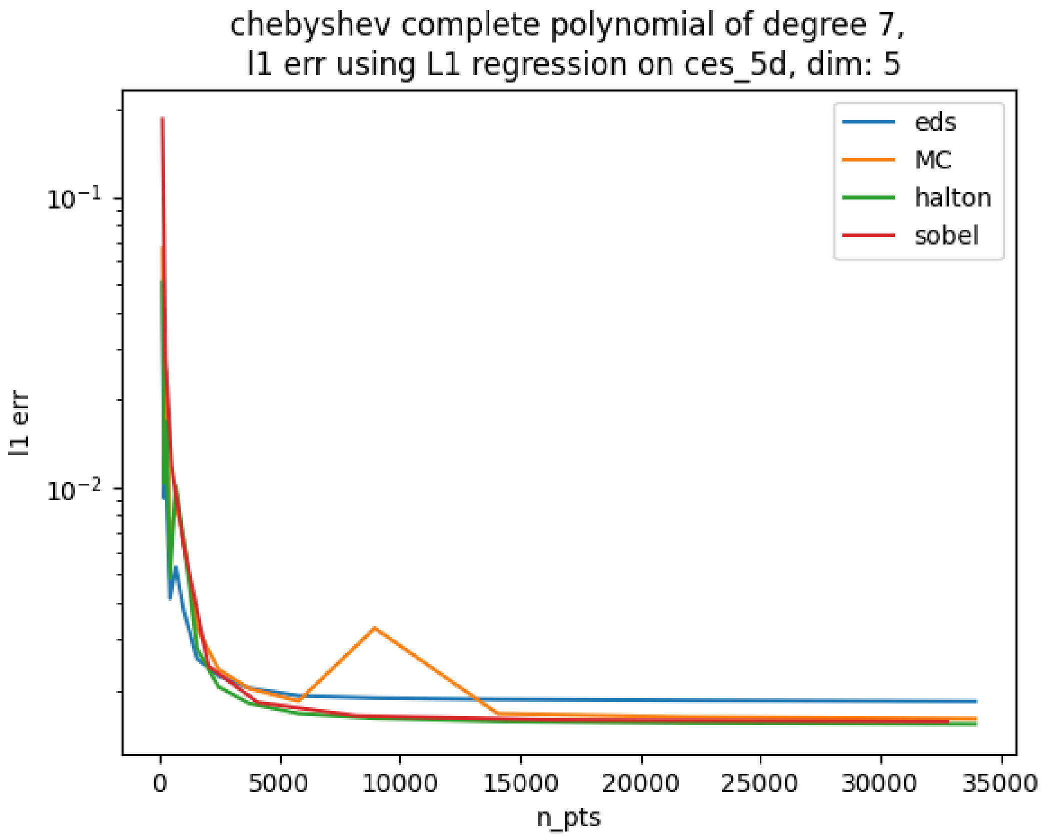
$$-a_\alpha^+ \leq a_\alpha \leq a_\alpha^+$$

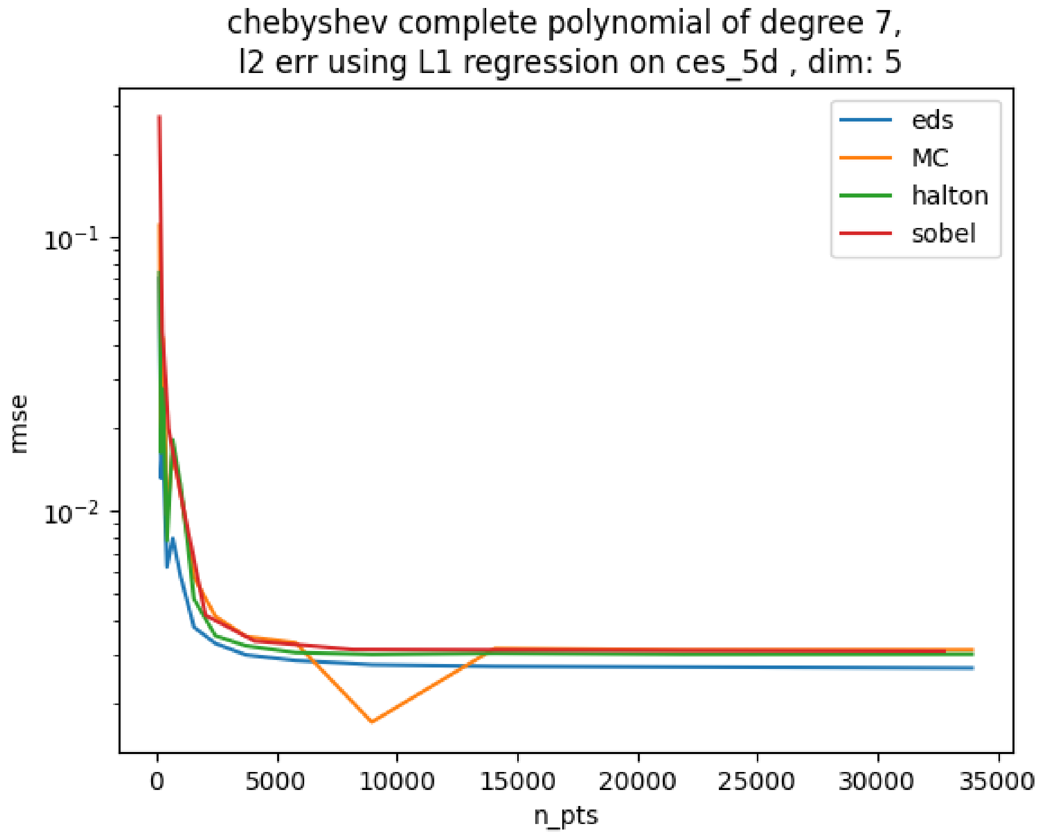
$$\mu \geq 0$$

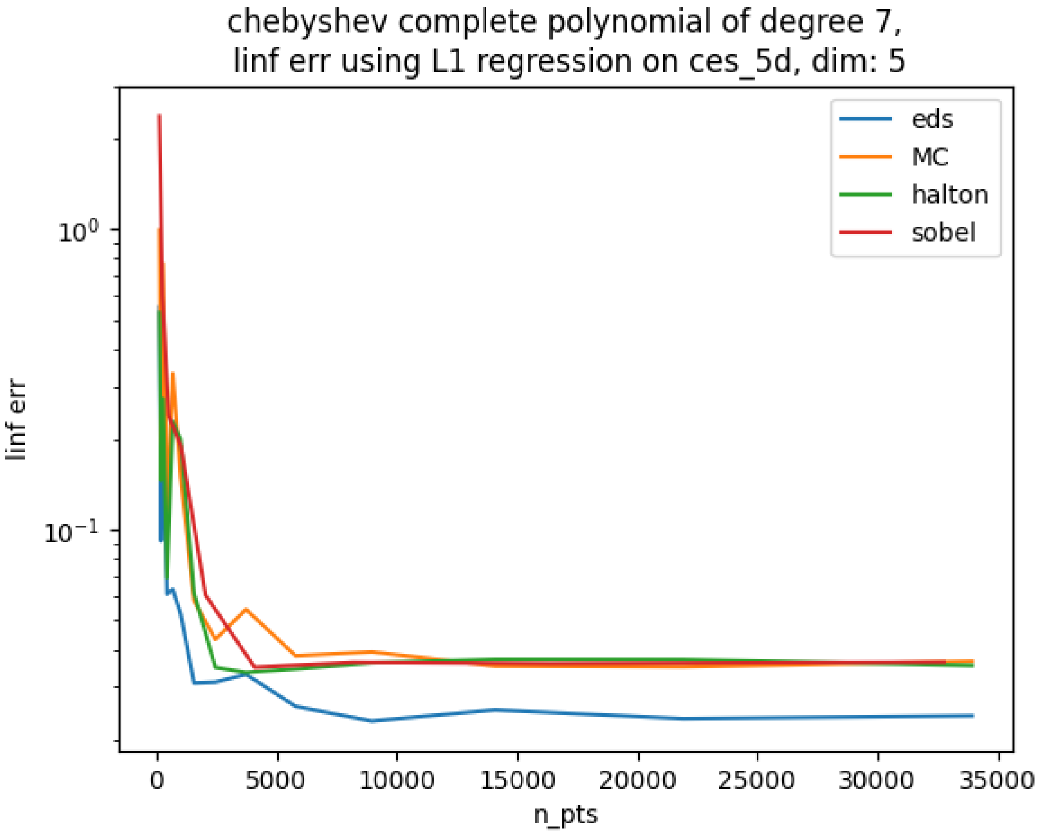
$$a_\alpha^+ \geq 0$$

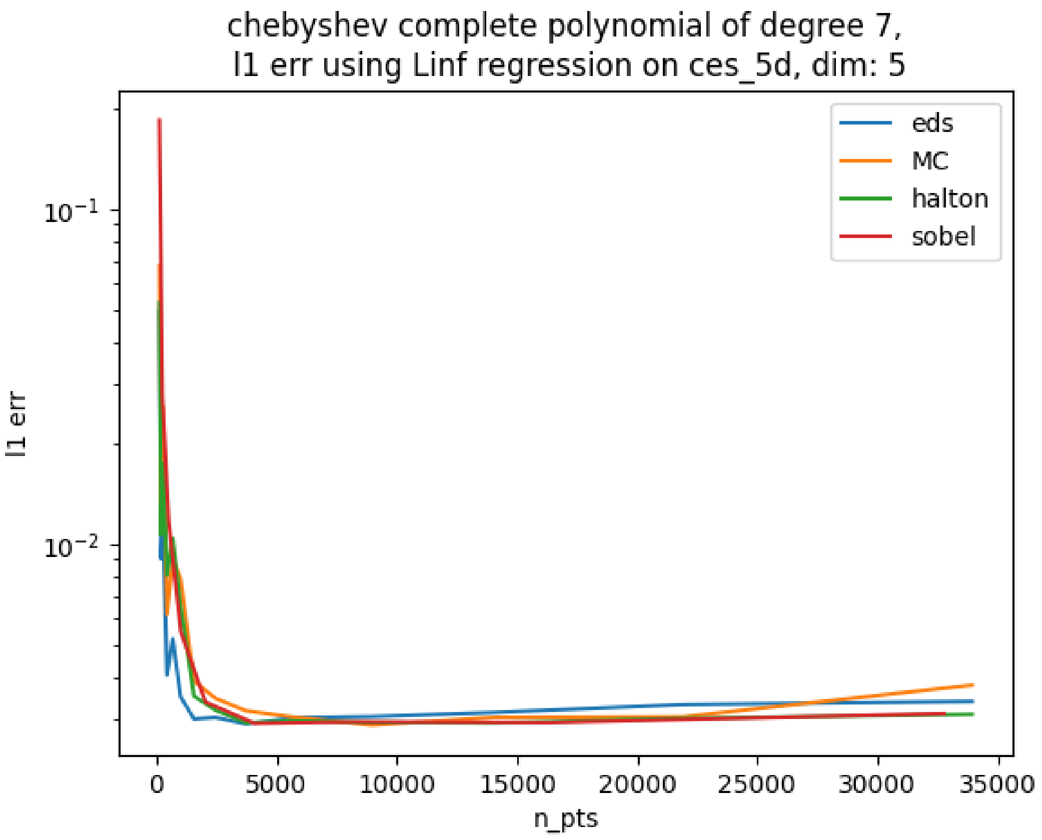
$$i = 1, 2, ..., n\_data, |\alpha| \leq deg$$

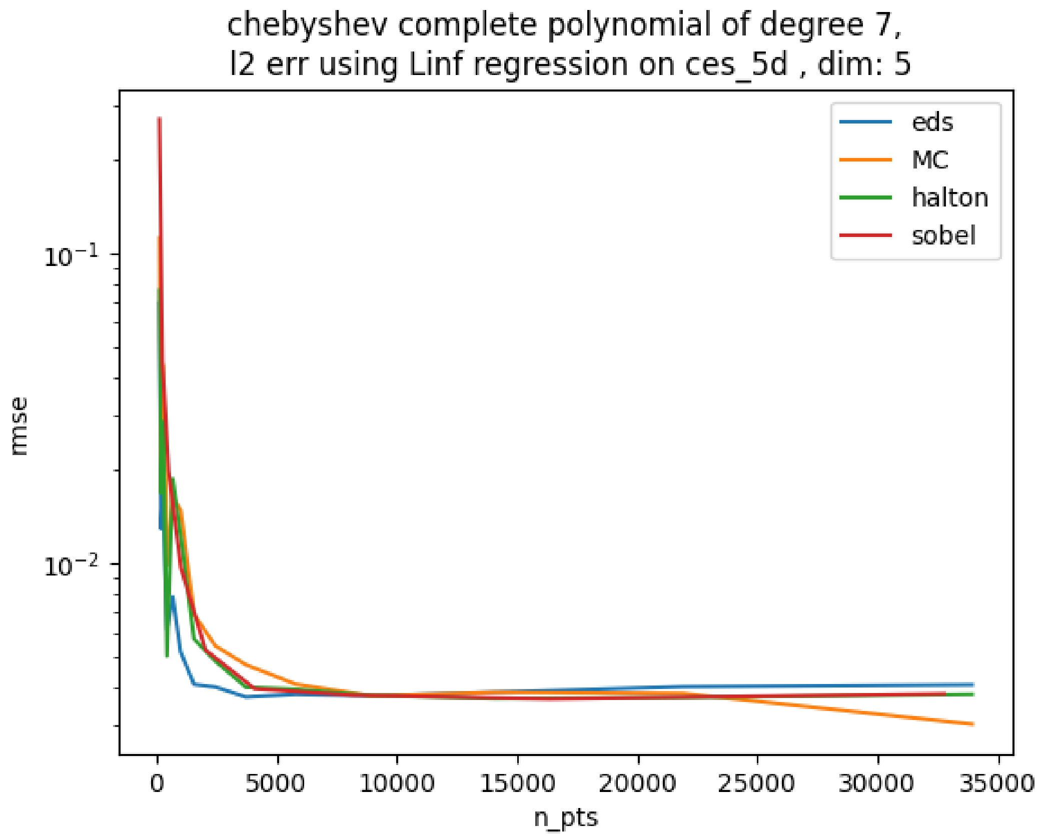
Following are the results of using Chebyshev regression on various data points, we can see that EDS(the blue line) have significant advantage when it comes to the L-inf errors of approximation.

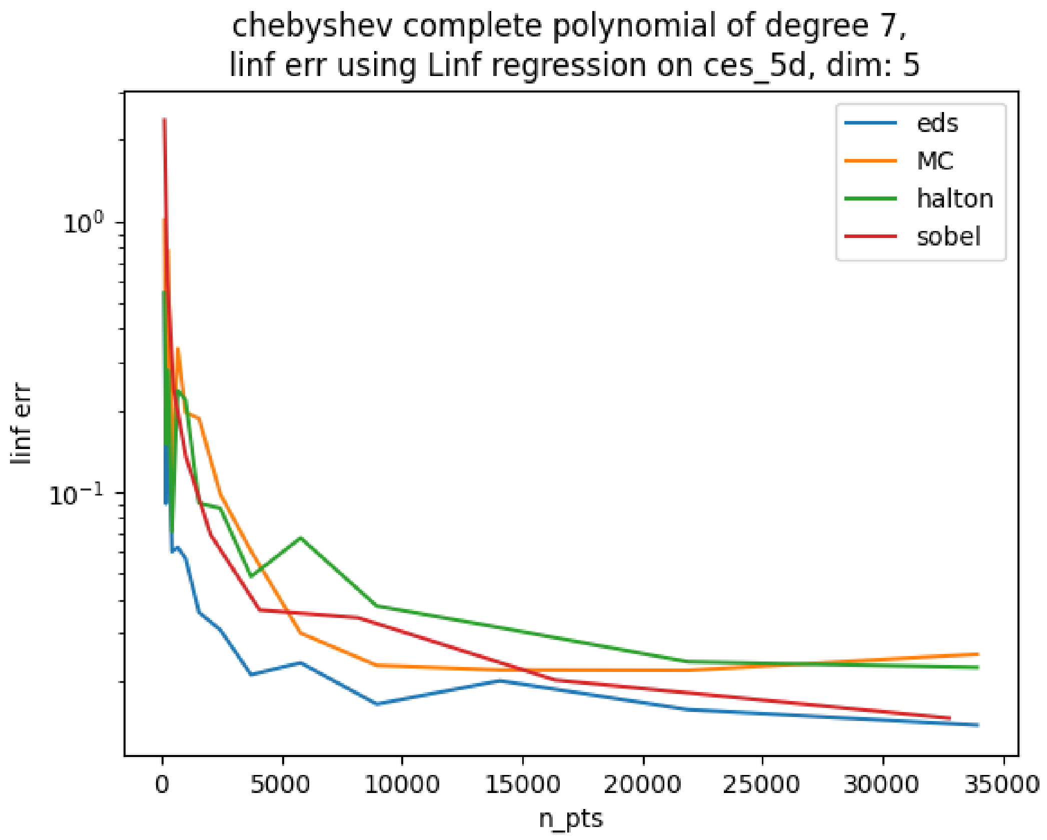


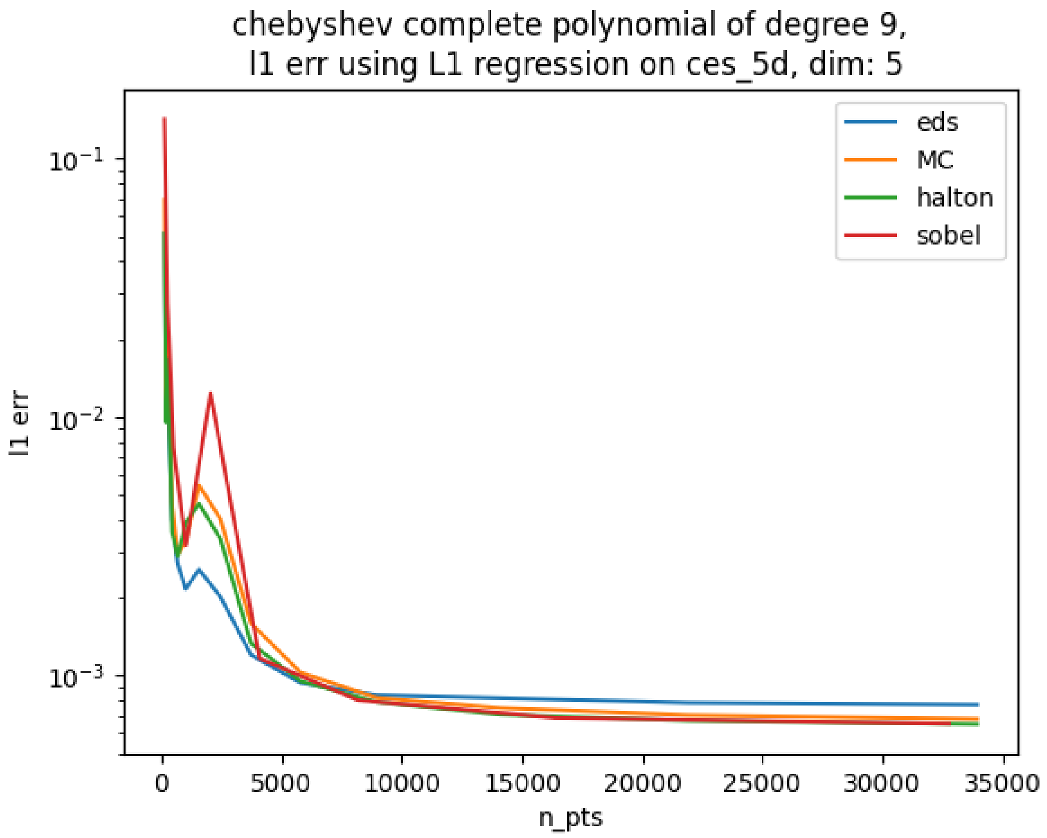

chebyshev complete polynomial of degree 7,
l2 err using L1 regression on ces_5d , dim: 5
mse
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000

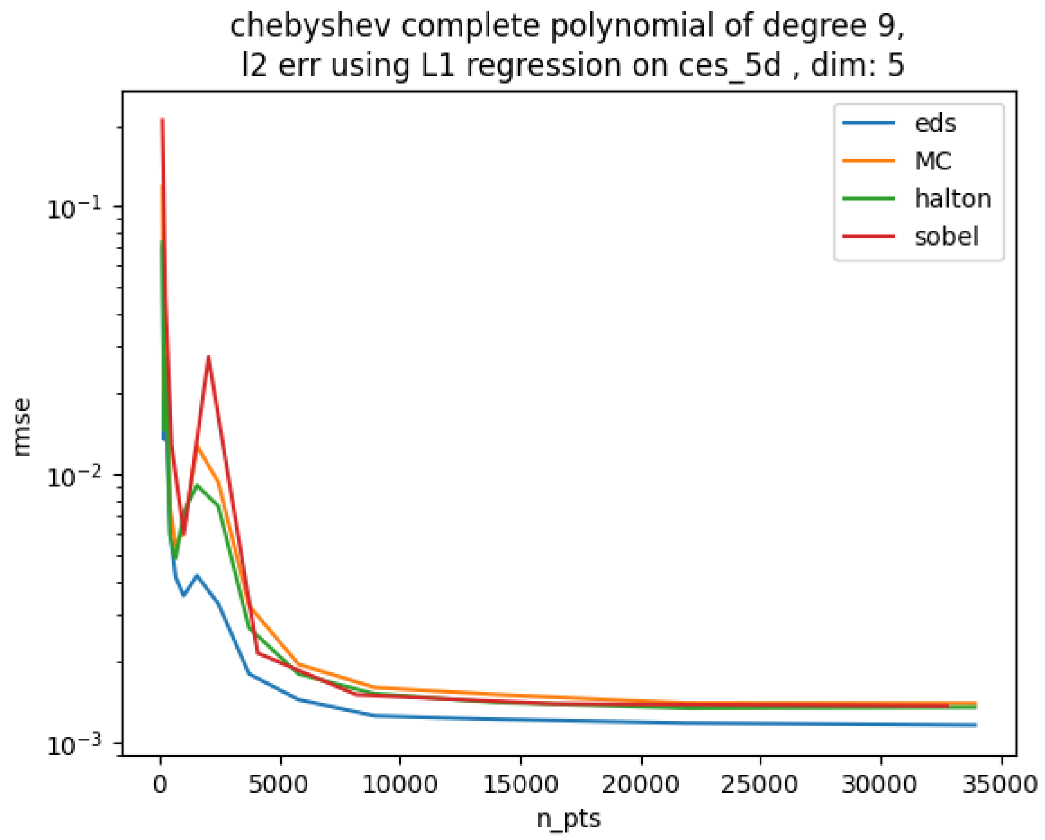

chebyshev complete polynomial of degree 7,
linf err using L1 regression on ces_5d, dim: 5
linf err
n_pts
eds
MC
halton
sobel
10^{0}
10^{-1}
0
5000
10000
15000
20000
25000
30000
35000

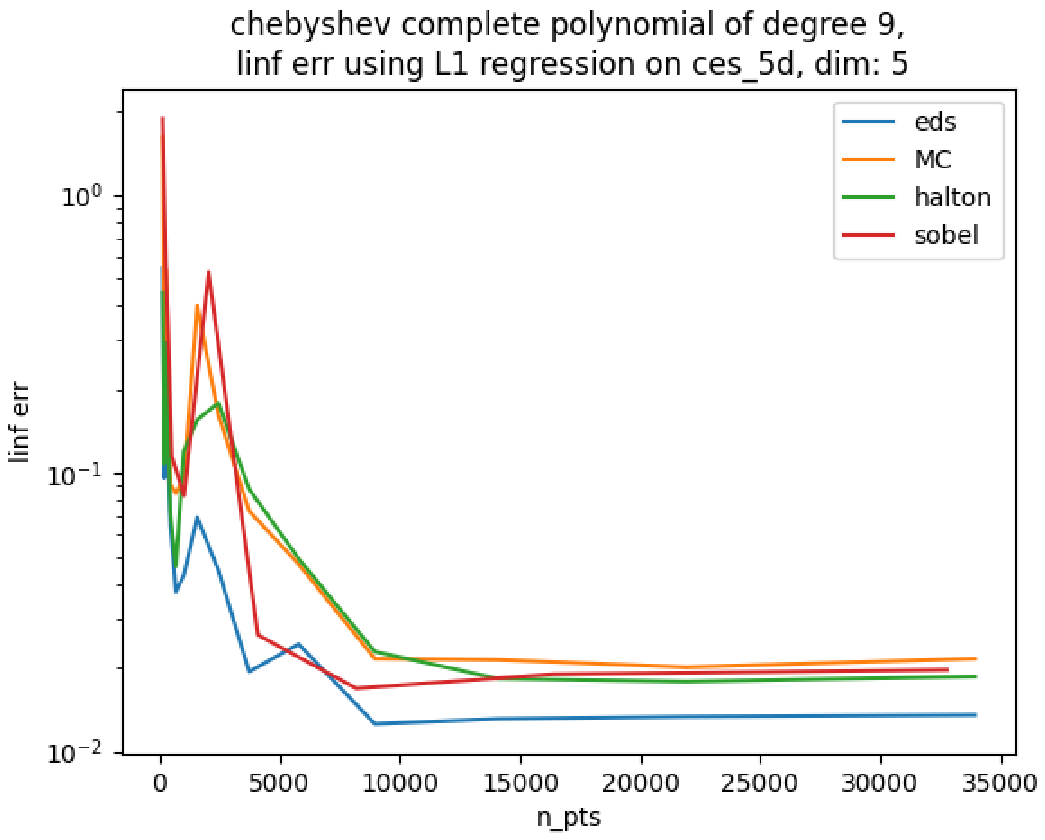

chebyshev complete polynomial of degree 7,
l1 err using Linf regression on ces_5d, dim: 5
l1 err
n_pts
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000
eds
MC
halton
sobel

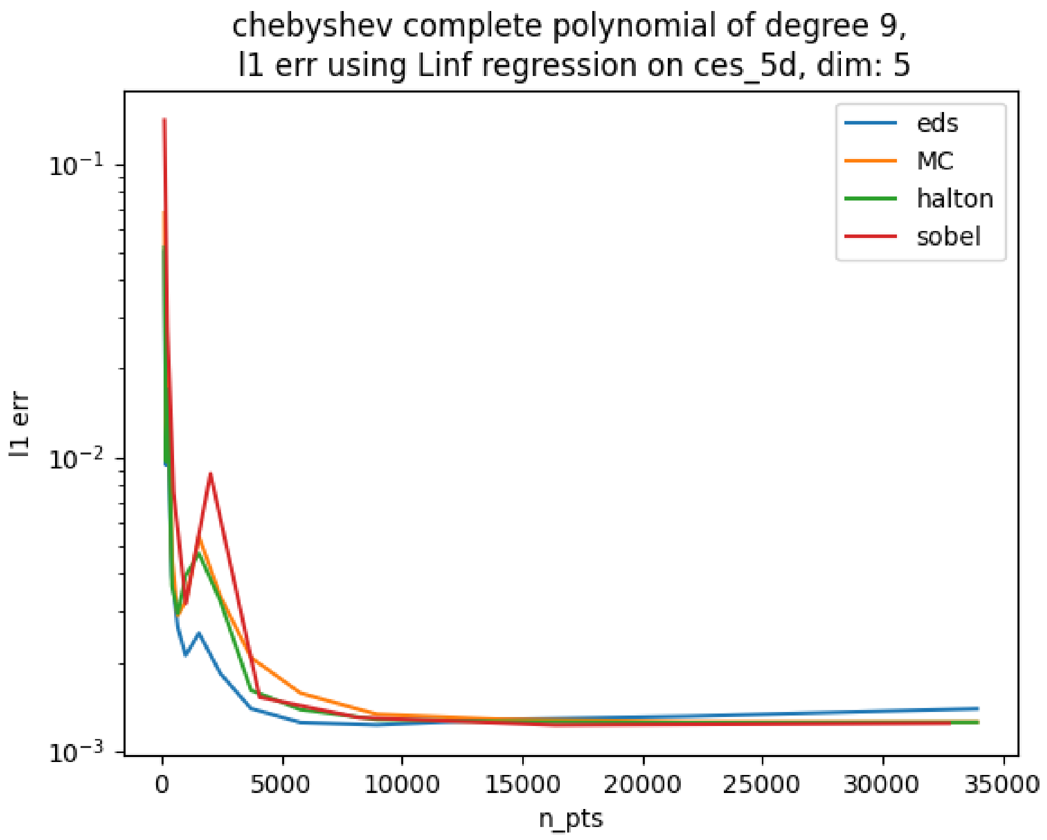

chebyshev complete polynomial of degree 7,
l2 err using Linf regression on ces_5d , dim: 5
mse
n_pts
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000
eds
MC
halton
sobel

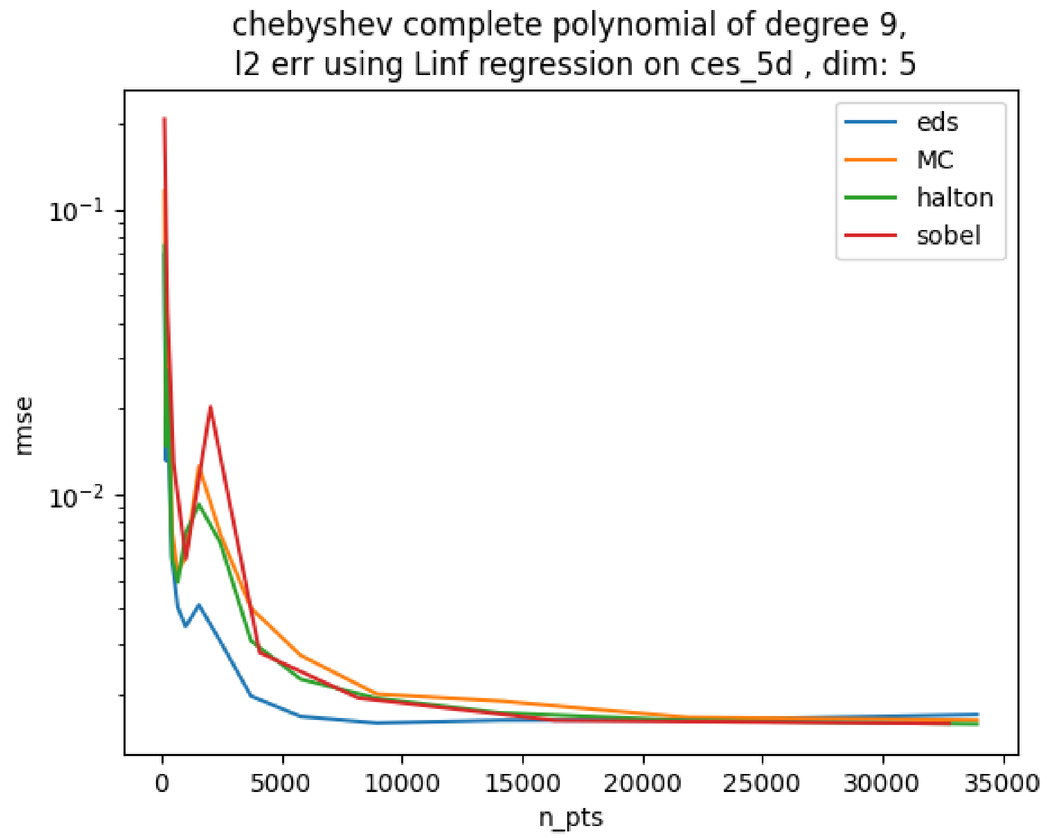

chebyshev complete polynomial of degree 7,
linf err using Linf regression on ces_5d, dim: 5
eds
MC
halton
sobel
linf err
n_pts
0
5000
10000
15000
20000
25000
30000
35000
10^0
10^{-1}

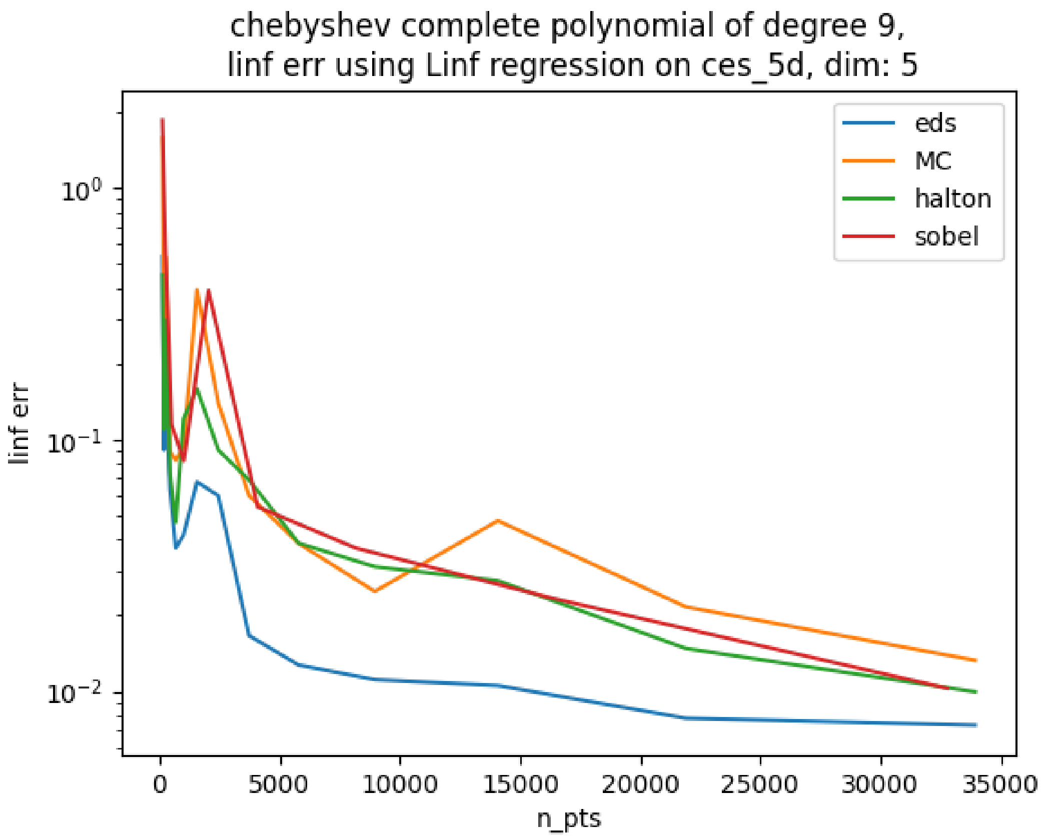

chebyshev complete polynomial of degree 9,
l1 err using L1 regression on ces_5d, dim: 5
eds
MC
halton
sobel
l1 err
n_pts
0
5000
10000
15000
20000
25000
30000
35000
10^{-1}
10^{-2}
10^{-3}

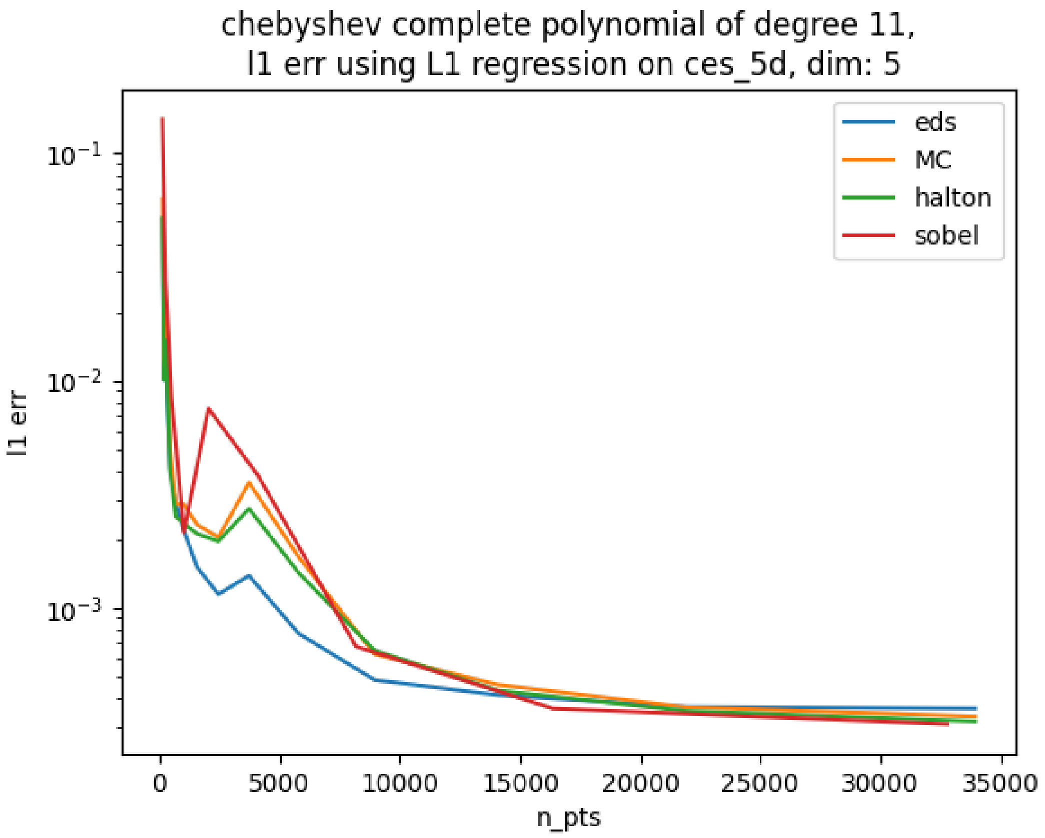

chebyshev complete polynomial of degree 9,
l2 err using L1 regression on ces_5d , dim: 5
eds
MC
halton
sobel
mse
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000
n_pts

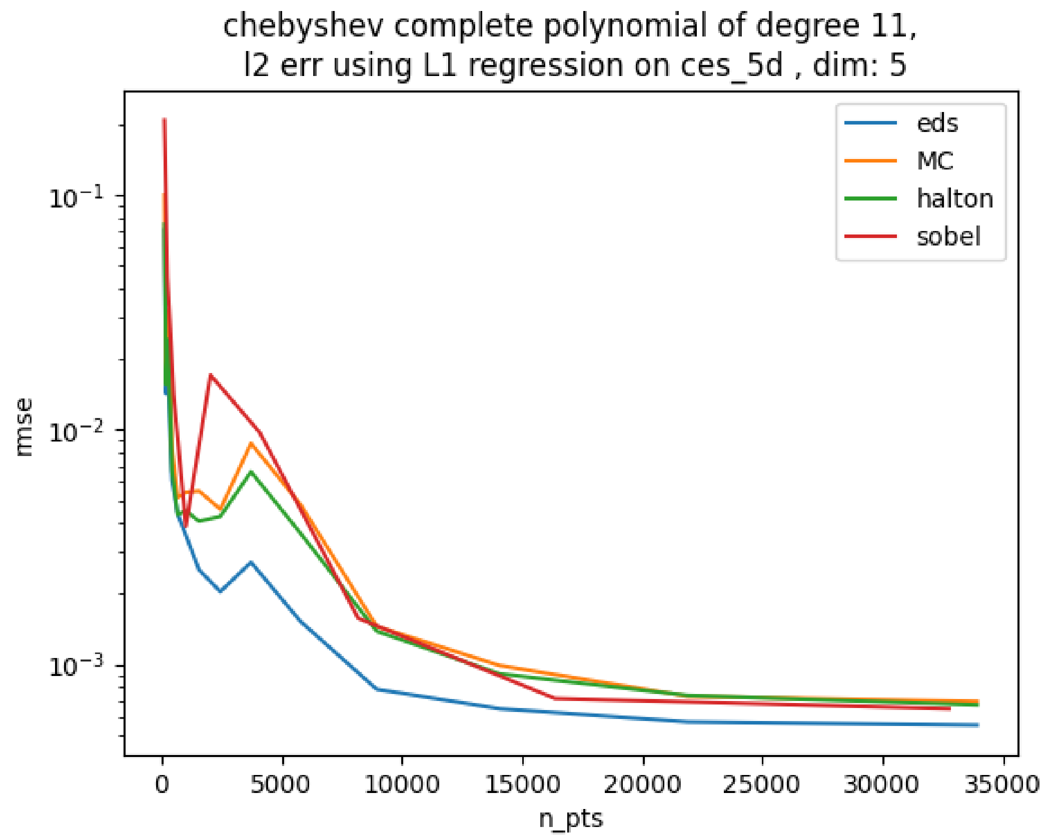

chebyshev complete polynomial of degree 9,
linf err using L1 regression on ces_5d, dim: 5
eds
MC
halton
sobel
linf err
10^{0}
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000
n_pts

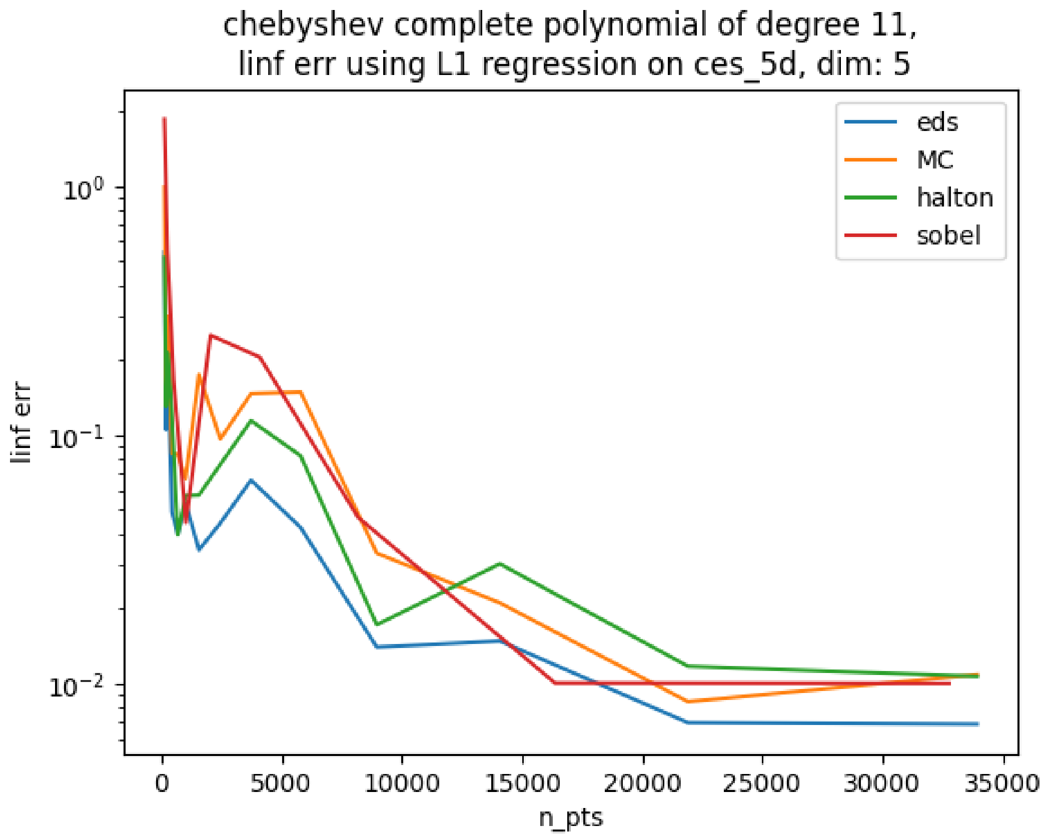

chebyshev complete polynomial of degree 9,
l1 err using Linf regression on ces_5d, dim: 5
l1 err
n_pts
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000
eds
MC
halton
sobel

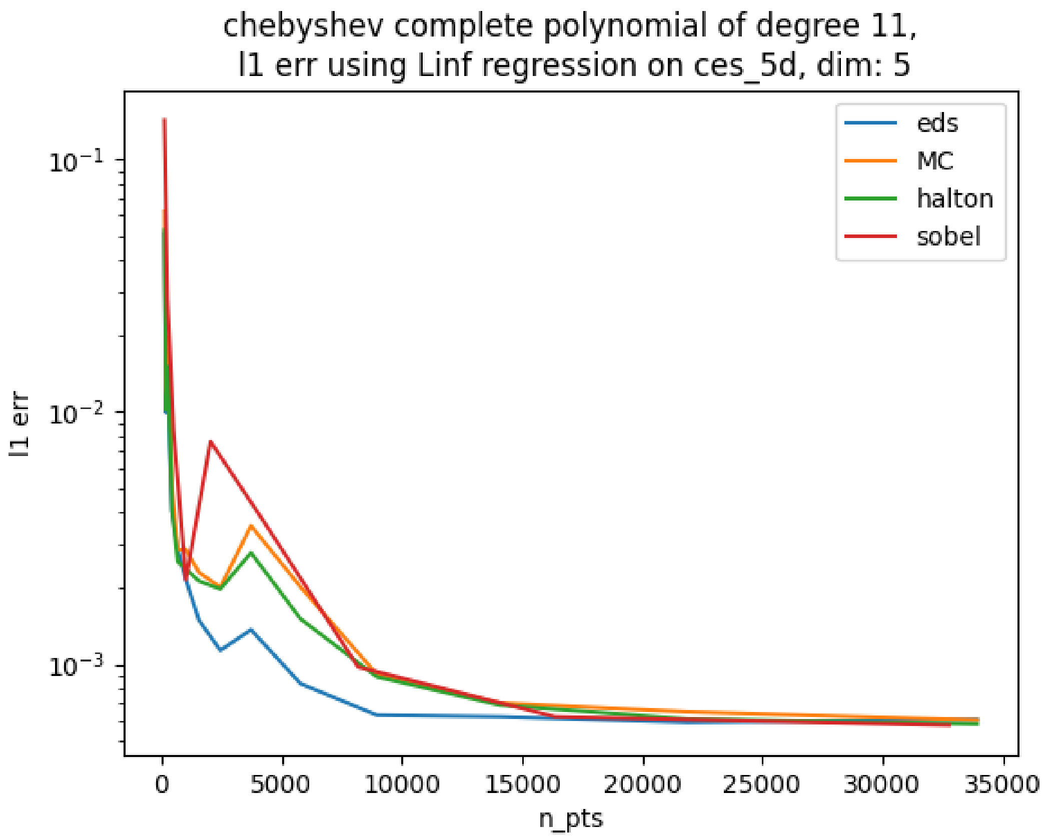

chebyshev complete polynomial of degree 9,
l2 err using Linf regression on ces_5d , dim: 5
mse
n_pts
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000
eds
MC
halton
sobel

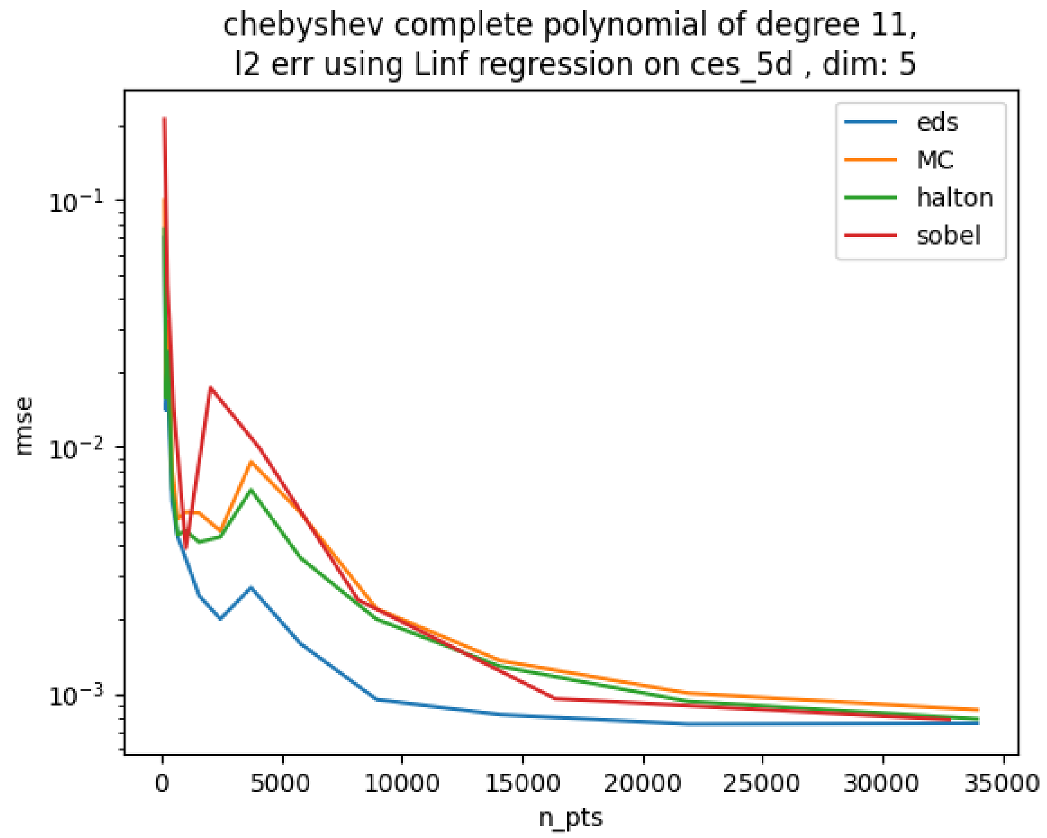

chebyshev complete polynomial of degree 9,
linf err using Linf regression on ces_5d, dim: 5
linf err
n_pts
eds
MC
halton
sobel
10^0
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000

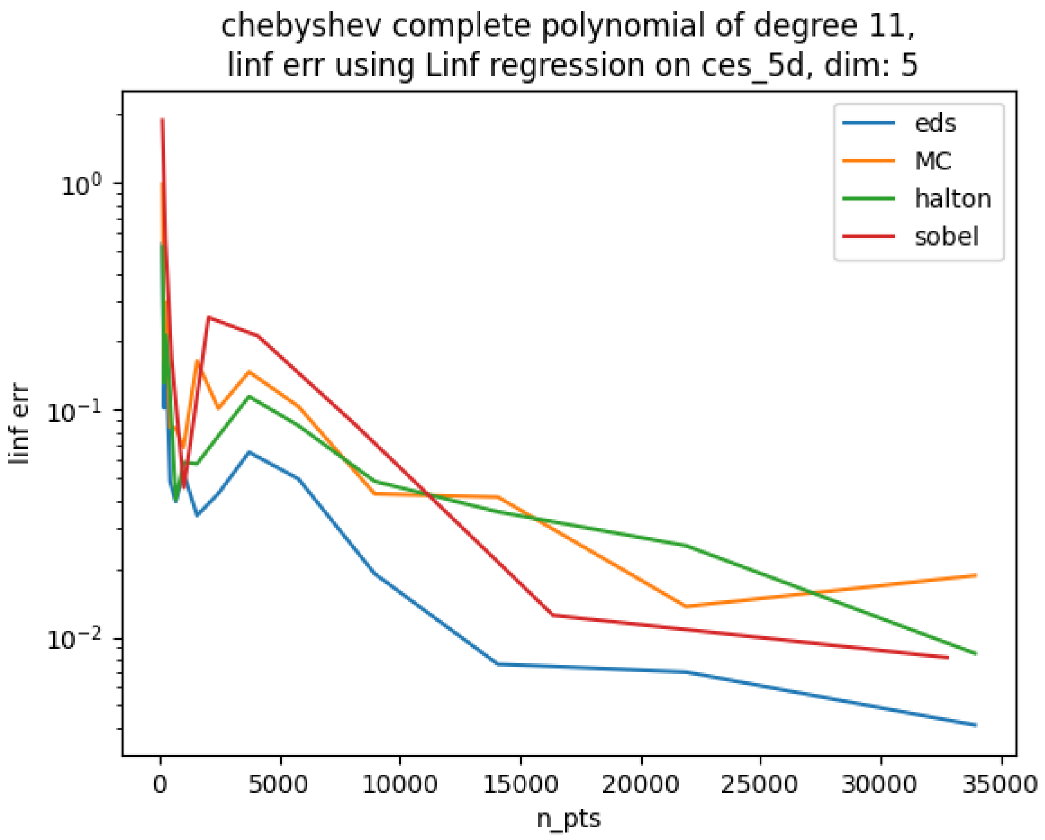

chebyshev complete polynomial of degree 11,
l1 err using L1 regression on ces_5d, dim: 5
l1 err
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000


chebyshev complete polynomial of degree 11,
l2 err using L1 regression on ces_5d , dim: 5
mse
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000


chebyshev complete polynomial of degree 11,
linf err using L1 regression on ces_5d, dim: 5
linf err
n_pts
eds
MC
halton
sobel
10^{0}
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000


chebyshev complete polynomial of degree 11,
l1 err using Linf regression on ces_5d, dim: 5
l1 err
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000

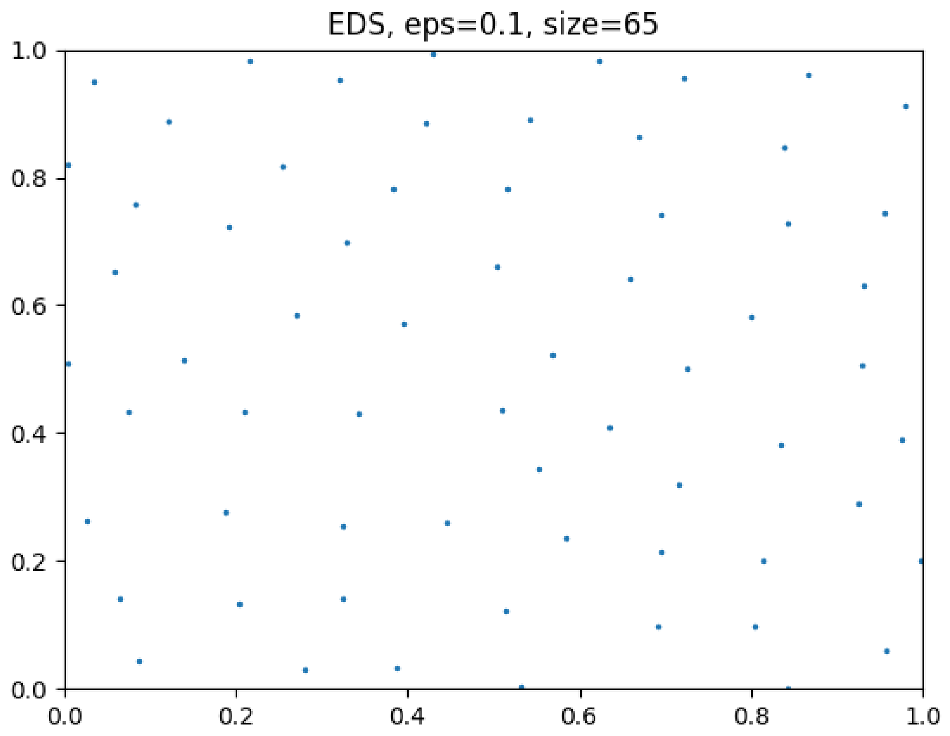

chebyshev complete polynomial of degree 11,
l2 err using Linf regression on ces_5d , dim: 5
mse
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000

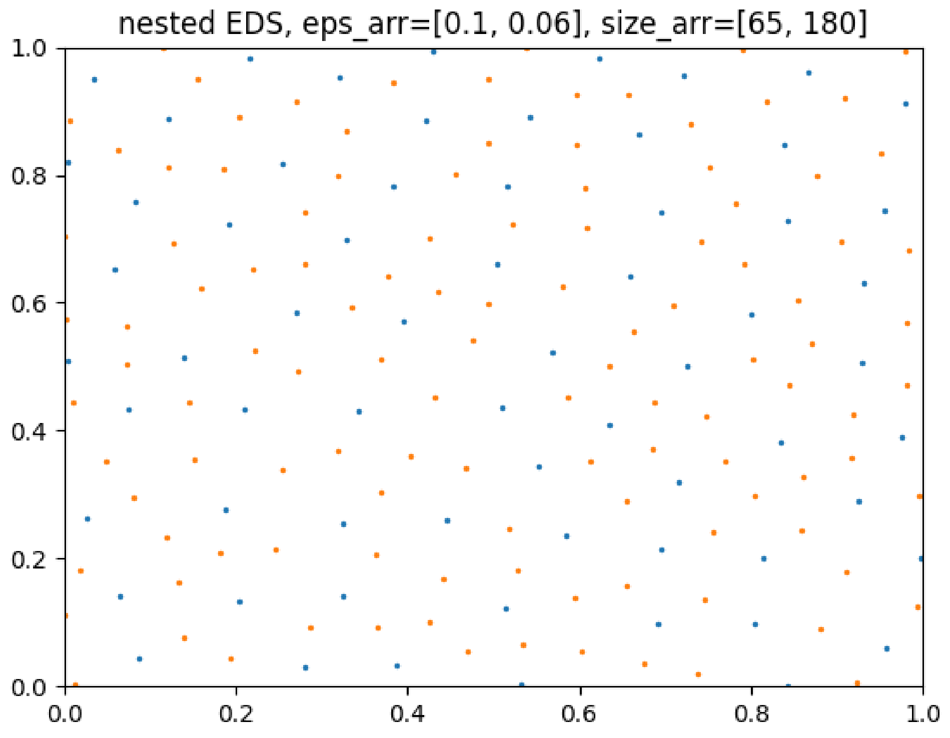


3.3 Nested Epsilon Distinguishable Set (NEDS)

Sometimes we want to reduce the epsilon parameter (say, we want to reduce from ϵ_1 to ϵ_2)and get a larger EDS set. We can do so by inserting new points to the old (ϵ_1-distinguishable set) while making sure the new set is ϵ_2-distinguishable. The new set is then a nested EDS set.

Definition: An ϵ_n-distinguishable set $\mathbf{X}$ is a nested epsilon distinguishable set of $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\}$ ($\epsilon_1 < \epsilon_2 < \ldots < \epsilon_n$) iff for each $\epsilon_i \in \{\epsilon_1, \epsilon_2, \ldots, \epsilon_n\}$, there exists $X_i \subseteq \mathbf{X}$ such that each X_i is a fully filled ϵ_i-distinguishable set. (first example is not nested EDS)


EDS, eps=0.1, size=65
0.0
0.2
0.4
0.6
0.8
1.0


nested EDS, eps_arr=[0.1, 0.06], size_arr=[65, 180]
0.0
0.2
0.4
0.6
0.8
1.0

4 Using adaptive nested EDS for function approximation (purpose: can easily build up larger sets)

4.1 NEDS and adpative rbf

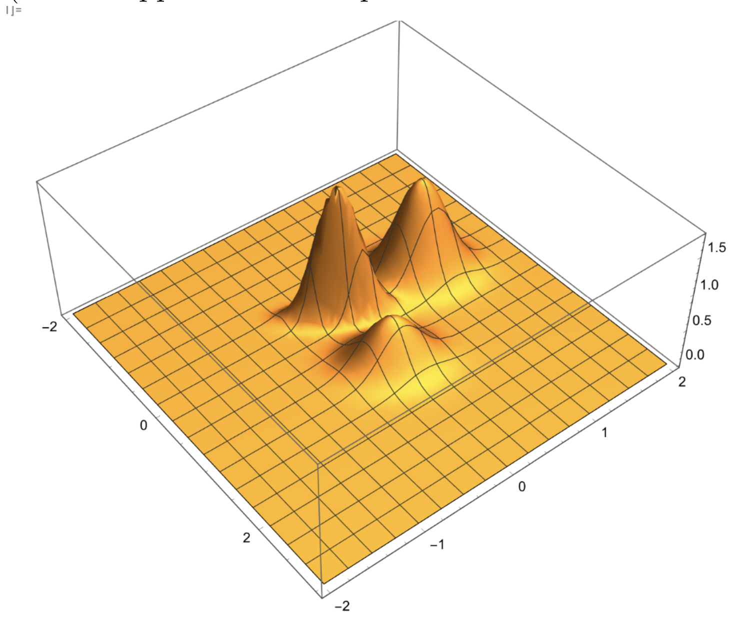
Given a callable function f, we want to construct a surrogate of the original function by computing a set of Radial Basis Function (RBF) centers, the corresponding of RBF interpolation coefficients and the optimal RBF kernel parameters for interpolation.

Assumptions and requirements: Each call to the original function f may be expensive, so the total evaluations of the original function need to be limited.

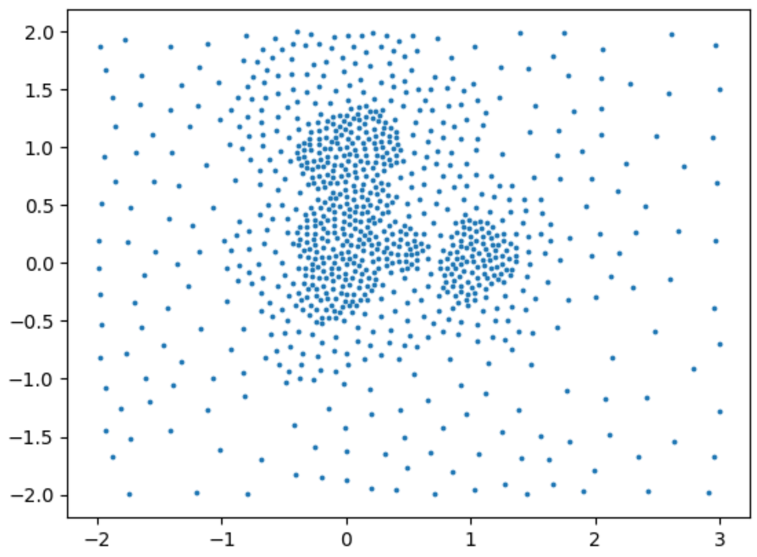
Algorithm overview:
The algorithm begins with sparse EDS sets with large ϵ parameter and smaller data size. Using leave-one-out cross-validation (LOOCV) the algorithm determines the points with the largest approximation errors, and generates refined nested-EDS sets (NEDS) with smaller ϵ within *delta* radius around these "bad points".
The algorithm ends when the target (leave-one-out) cross-validation accuracy has been achieved.

4.2 Example of adaptive EDS

We approximate the following 2-dimensional "mixtures of normals distribution test function" (see the appendix for the precise definition of this function)



The adaptive EDS points used to approximate this function would look like



Note that in the above graph, we have various local EDS sets with different density according the leave-one-out cross-validation(LOOCV) error. When the LOOCV error in some region is high, we want to use denser EDS sets with small ϵ in that region. Tradition LOOCV requires solving n different linear systems, leading to computational time of $O(n^4)$, where n is the size of interpolation points used. In 1999, Shmuel Rippa has proposed a formula that can reduce the computational complexity of LOOCV on RBF to $O(n^3)$, which is the same complexity of solving a single linear system. [5]

Using Rippa's formula, the LOOCV error on x_i using homogeneously scaled RBF interpolant can be computed as

$$\epsilon(x_i) = \frac{a_i}{K_{ii}^{-1}}$$

where $K_{ij} = \phi(\frac{\|x_i - x_j\|}{c})$, and a_i satisfies $Ka = y$, For spatially variably scaled RBF interpolant, it can be shown that the formula above still holds, with $K_{ij} = \phi(\frac{\|x_i - x_j\|}{c_j})$

Computing K^{-1} would cost $O(n^3)$, while the coefficient array a can be computed using $a = K^{-1}y$ in $O(n^2)$ time. So the total complexity to compute the LOOCV error would be $O(n^3)$

5 Future works

5.1 Using compactly supported RBF kernel

Interpolation using globally supported RBF kernels will require solving a dense linear system, which could be costly when the data size is large. A recent paper [6] provides details for constructing optimal compactly supported RBF kernels in Sobolev spaces.

5.2 Approximating discontinuous functions

Many economic problem (such as taxation problems) will be easy solved if we can construct discontinuous surrogate functions. However, traditional function fitting which uses smooth function basis suffer from the gibbs phenomenon. A promising new tools for discontinuous function approximation is using a special type of RBF function called "variably scaled discontinuous kernels (VSDK)".[7]

If we know where the discontinuity of the function is in advance, we can then construct an approximant using VSDK radial basis functions. In the meantime, there is another paper that proposes to use another type of RBF function called "variably scaled kernels(VSK)"[8] to locate the discontinuity of the original function.

By combining the methods in these 2 papers, we may eventually be able to use RBF to approximate discontinuous functions.

6 Appendix

6.1 explicit form of mixture distribution test function

The mixture distribution test function used to test adaptive EDS has the following formula

$$pdf = \frac{e^{\frac{1}{2}(-\frac{(-1+x)(y\rho\sigma_1+\sigma_2-x\sigma_2)}{(-1+\rho^2)\sigma_1^2\sigma_2} - \frac{y(-y\sigma_1-\rho\sigma_2+x\rho\sigma_2)}{(-1+\rho^2)\sigma_1\sigma_2^2})}}{2\pi\sqrt{\sigma_1^2\sigma_2^2 - \rho^2\sigma_1^2\sigma_2^2}}$$

We define the following 3 distributions:

$$pdf_1 : \mu_1 = 0, \mu_2 = 0, \sigma_1 = 1/6, \sigma_2 = 1/5, \rho = 0.75$$

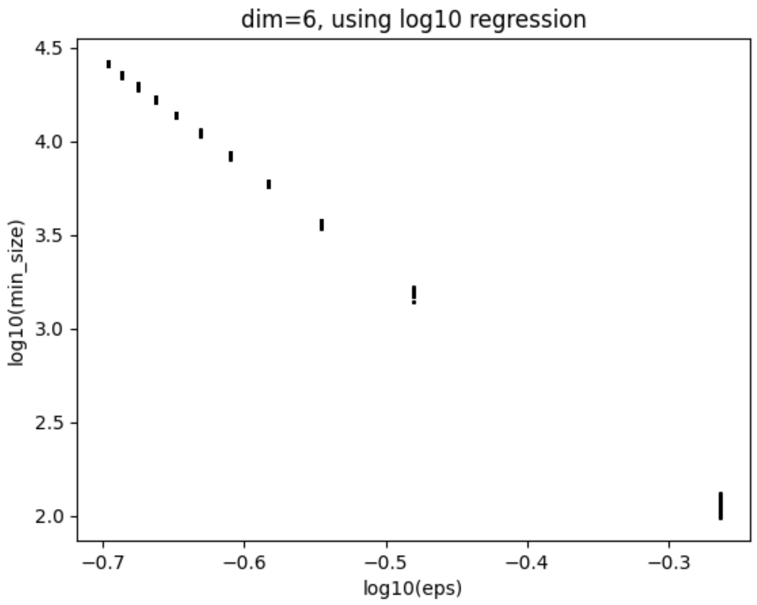
$$pdf_2 : \mu_1 = 0, \mu_2 = 1, \sigma_1 = 0.25, \sigma_2 = 0.25, \rho = 0.25$$

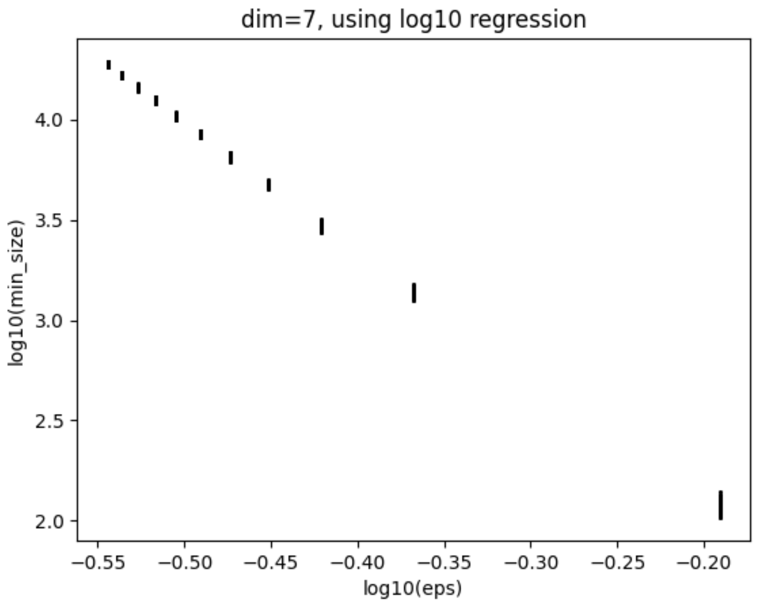
$$pdf_3 : \mu_1 = 1, \mu_2 = 0, \sigma_1 = 0.25, \sigma_2 = 0.25, \rho = 0.25$$

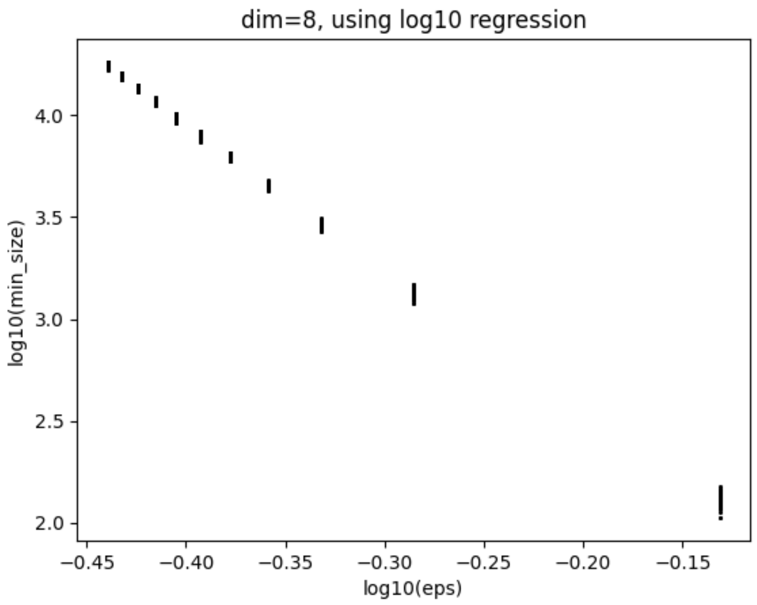
The mixture distribution is thus:

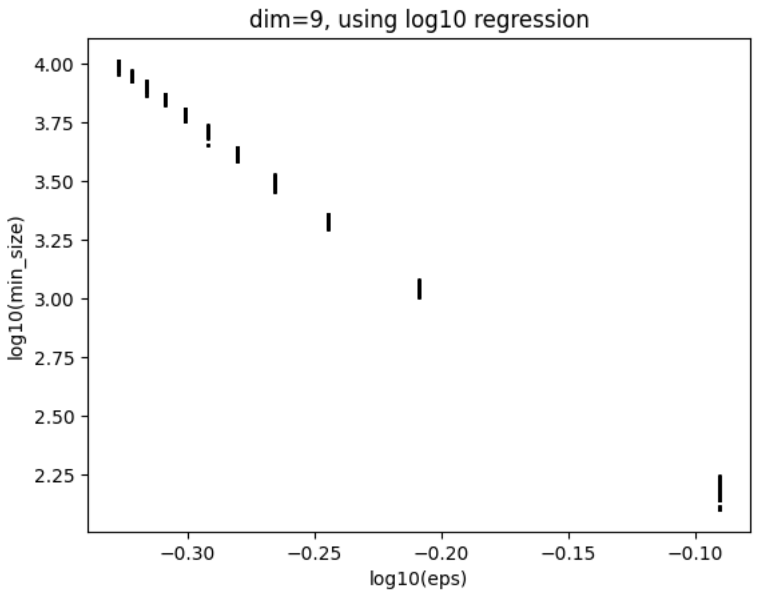
$$pdf_{mixture} = 0.25pdf_1 + 0.5pdf_2 + 0.25pdf_3$$

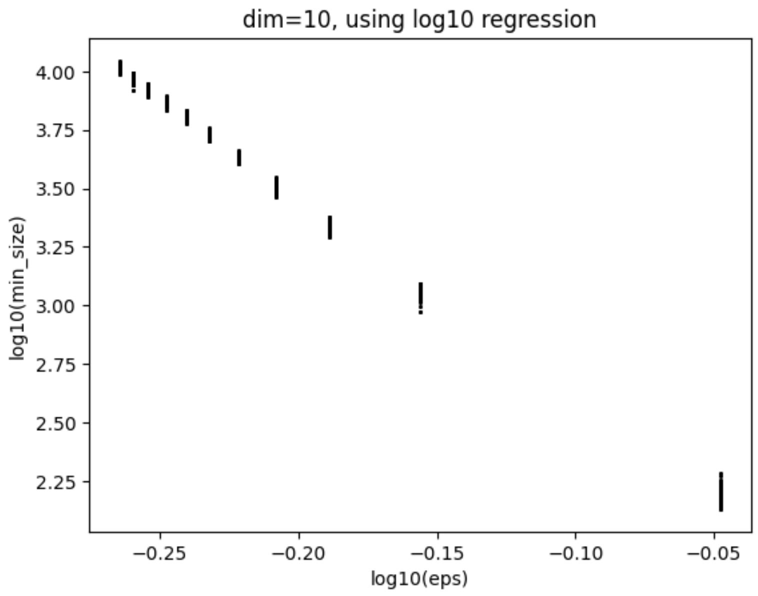
6.2 Other EDS size distributions (100 repeated runs)





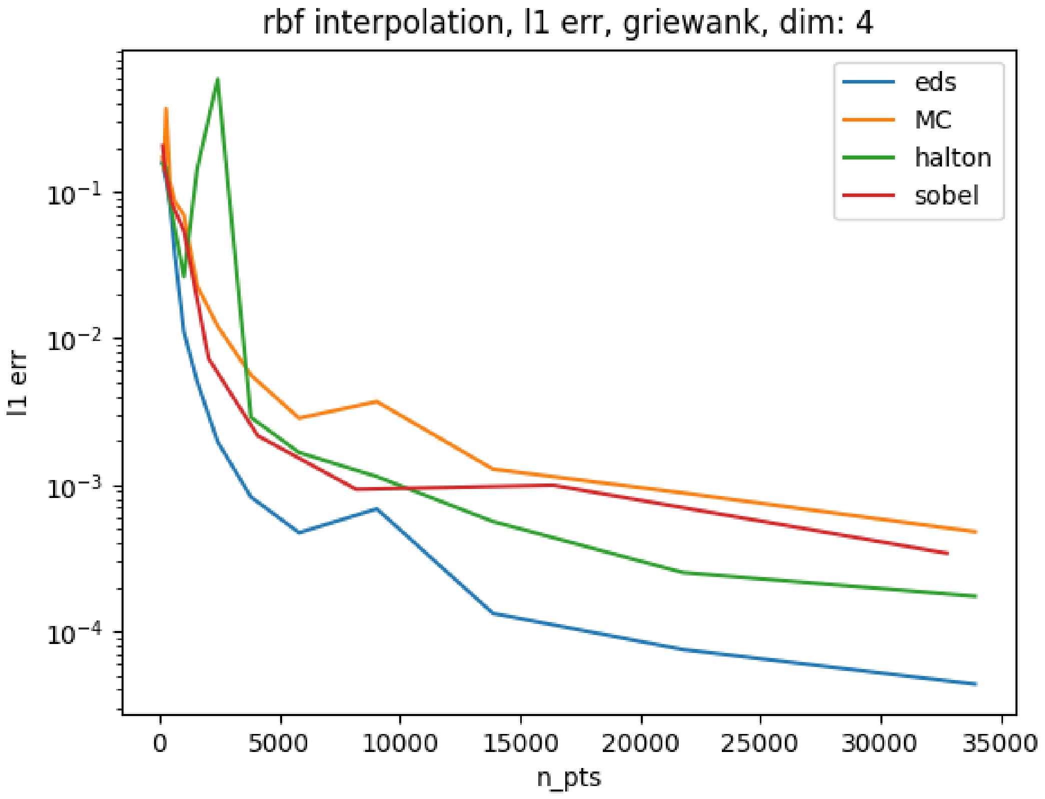

dim=8, using log10 regression
log10(min_size)
log10(eps)
4.0
3.5
3.0
2.5
2.0
−0.45
−0.40
−0.35
−0.30
−0.25
−0.20
−0.15

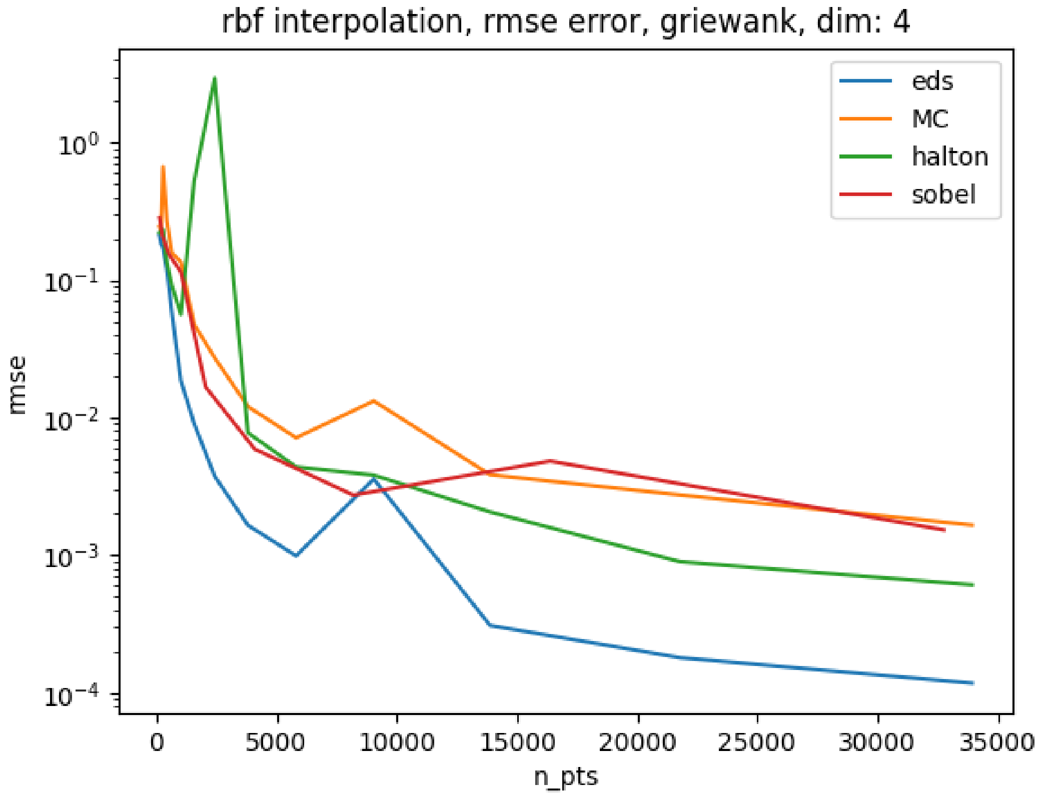

dim=9, using log10 regression
log10(min_size)
log10(eps)
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
−0.30
−0.25
−0.20
−0.15
−0.10

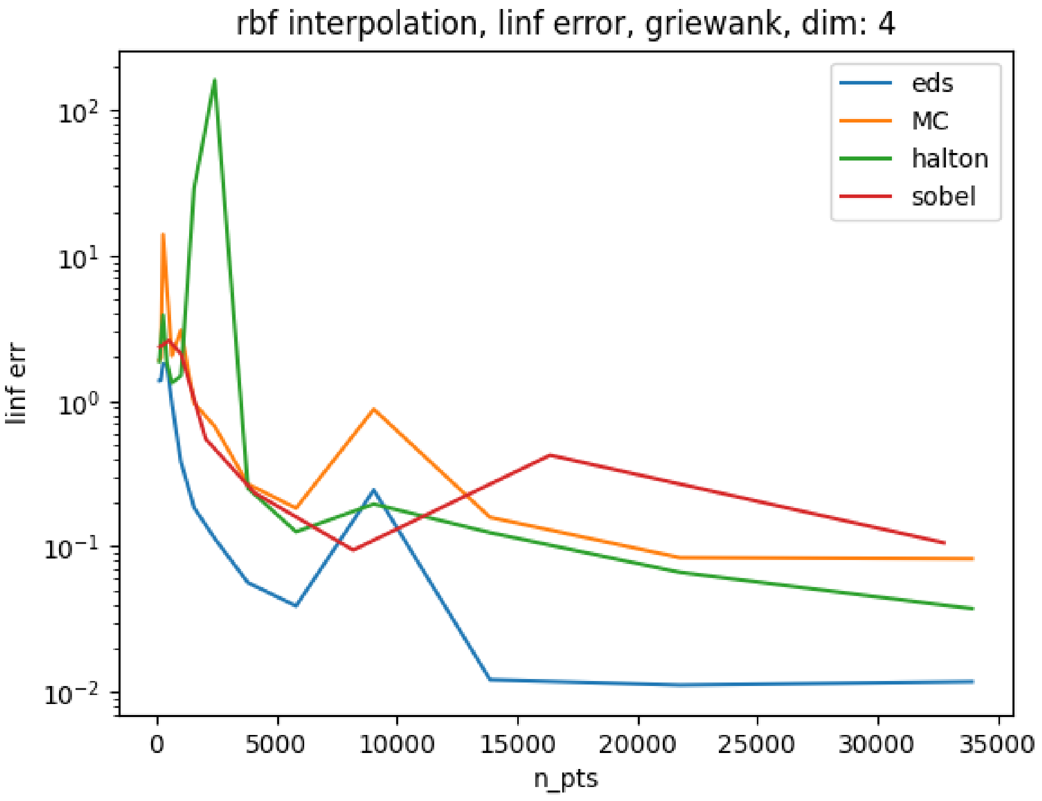


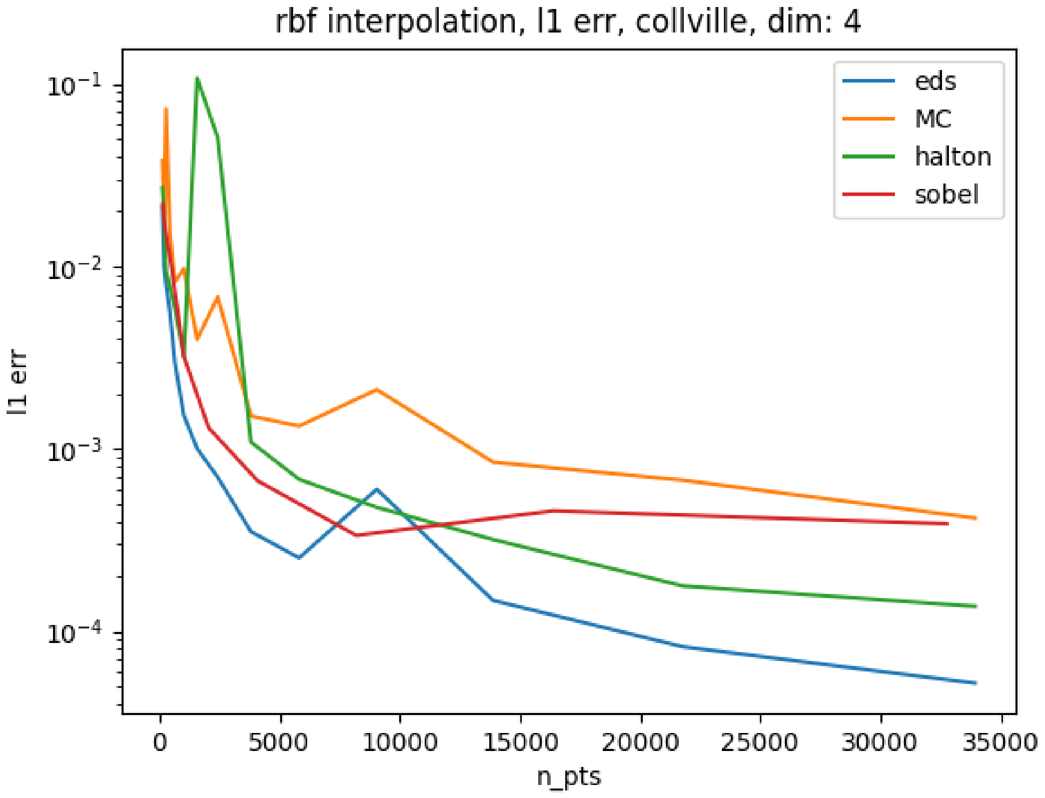
6.3 Other test functions and experiment results

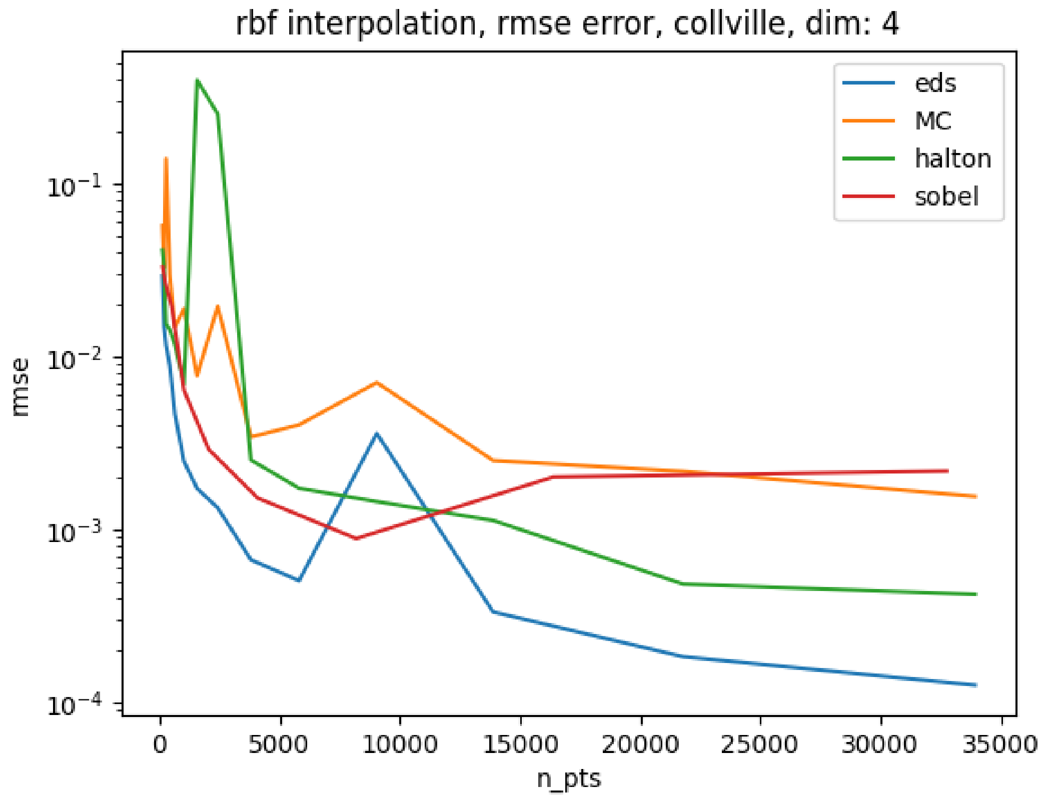
Details about test functions to be added later

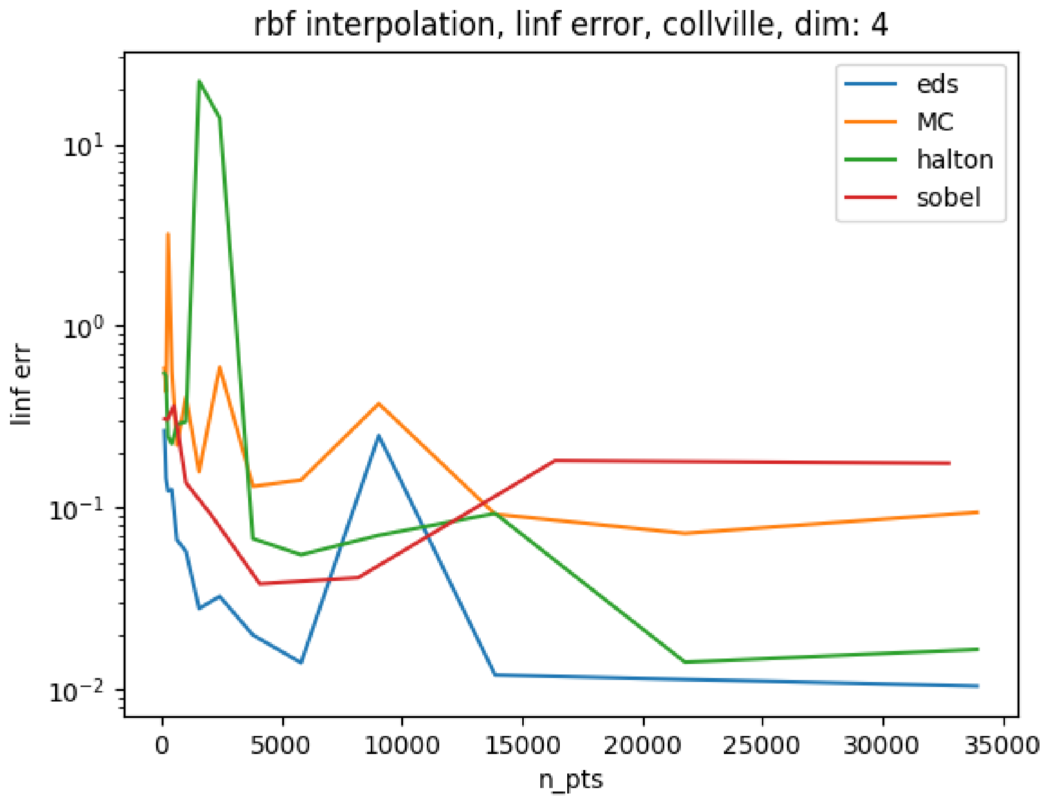


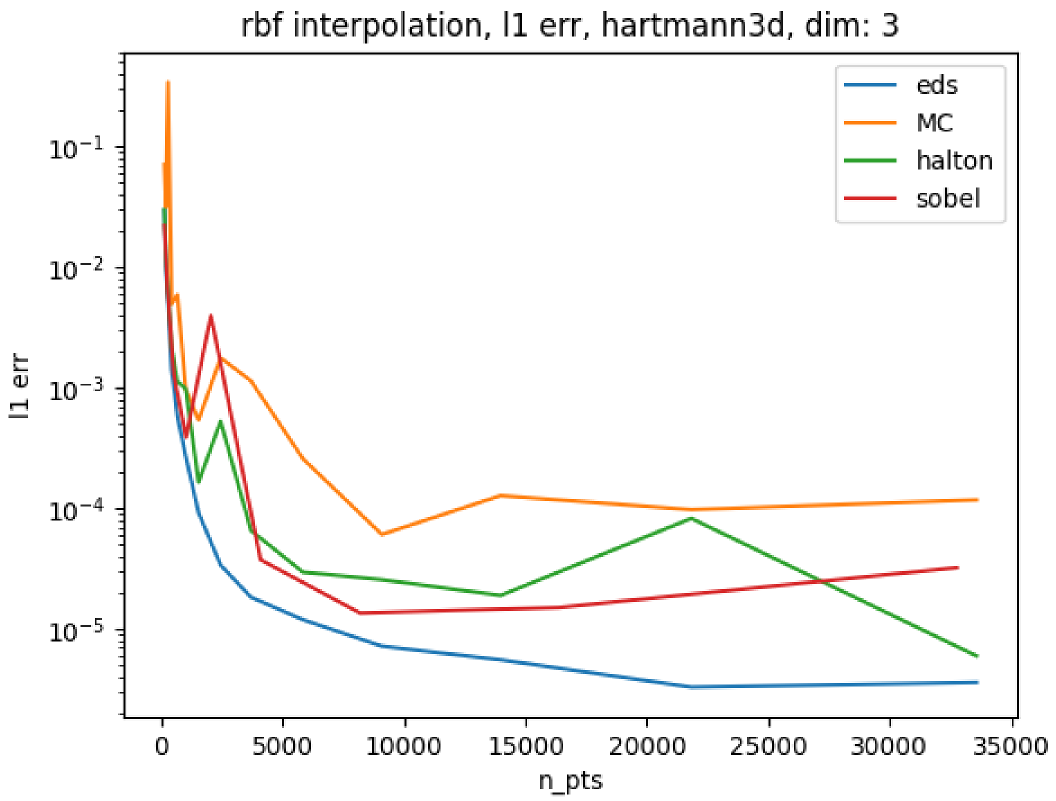

rbf interpolation, rmse error, griewank, dim: 4
rmse
n_pts
eds
MC
halton
sobel

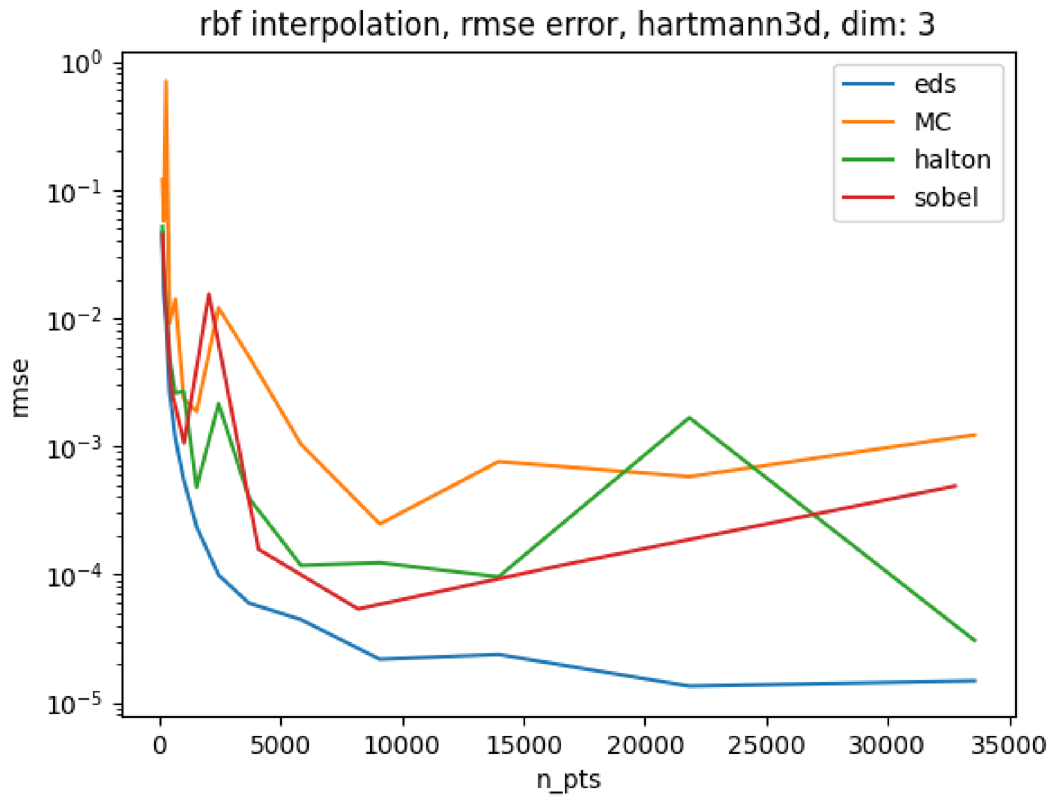

rbf interpolation, linf error, griewank, dim: 4
linf err
n_pts
eds
MC
halton
sobel


rbf interpolation, l1 err, collville, dim: 4
l1 err
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5000
10000
15000
20000
25000
30000
35000

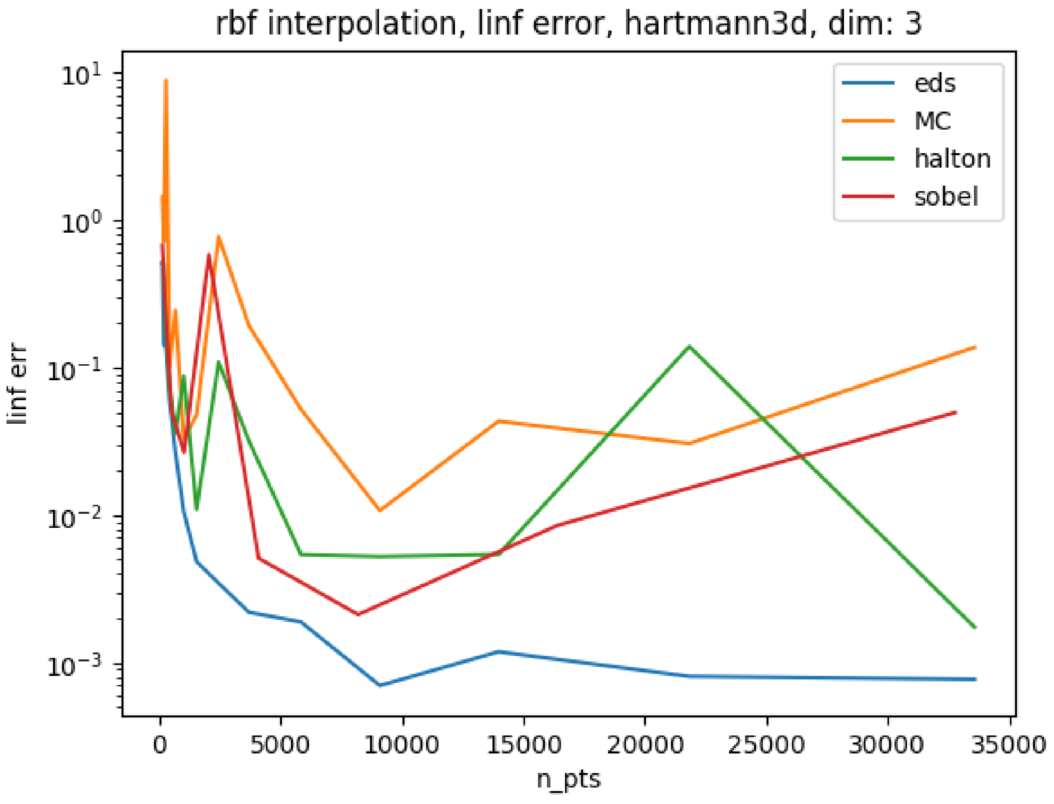

rbf interpolation, rmse error, collville, dim: 4
rmse
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5000
10000
15000
20000
25000
30000
35000


rbf interpolation, linf error, collville, dim: 4
linf err
n_pts
eds
MC
halton
sobel
10^{1}
10^{0}
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000

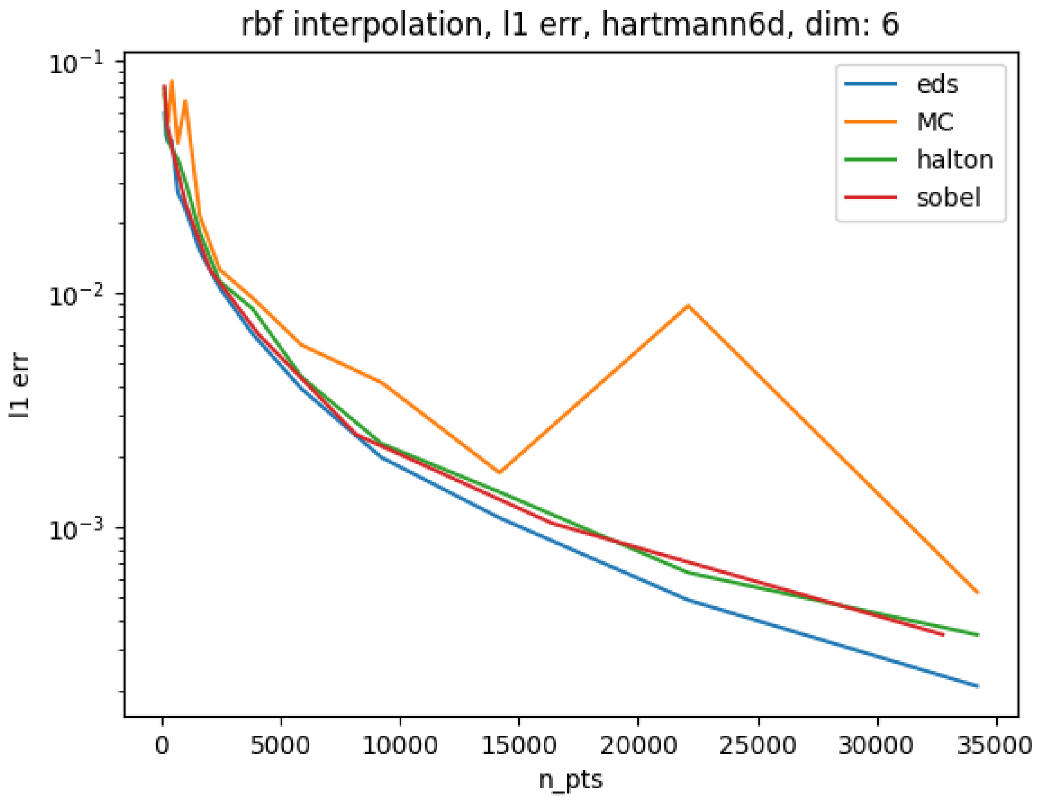

rbf interpolation, l1 err, hartmann3d, dim: 3
l1 err
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
5000
10000
15000
20000
25000
30000
35000

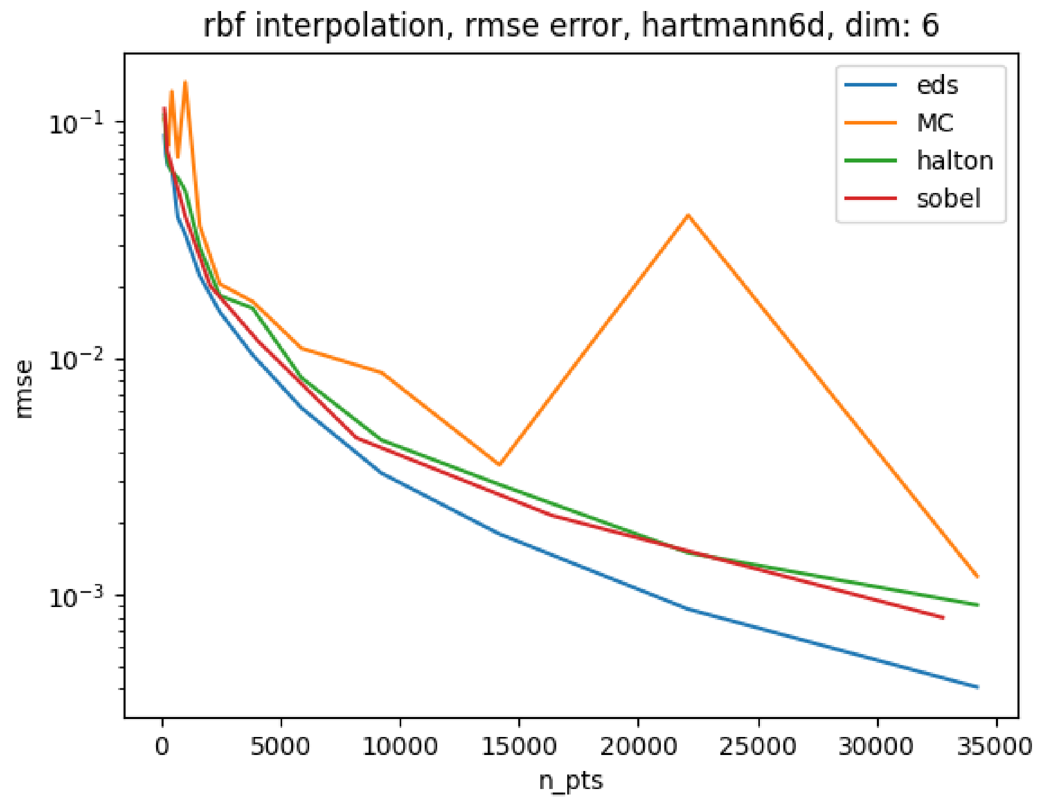


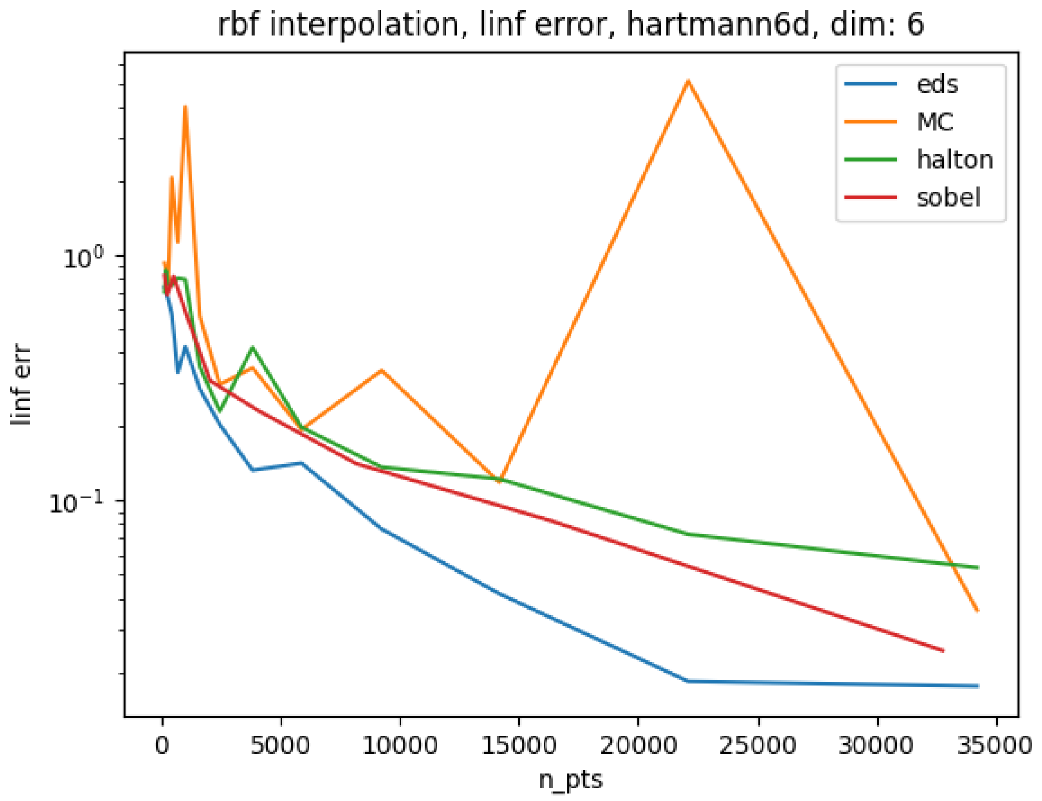
rbf interpolation, rmse error, hartmann3d, dim: 3
rmse
n_pts
eds
MC
halton
sobel
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
0
5000
10000
15000
20000
25000
30000
35000



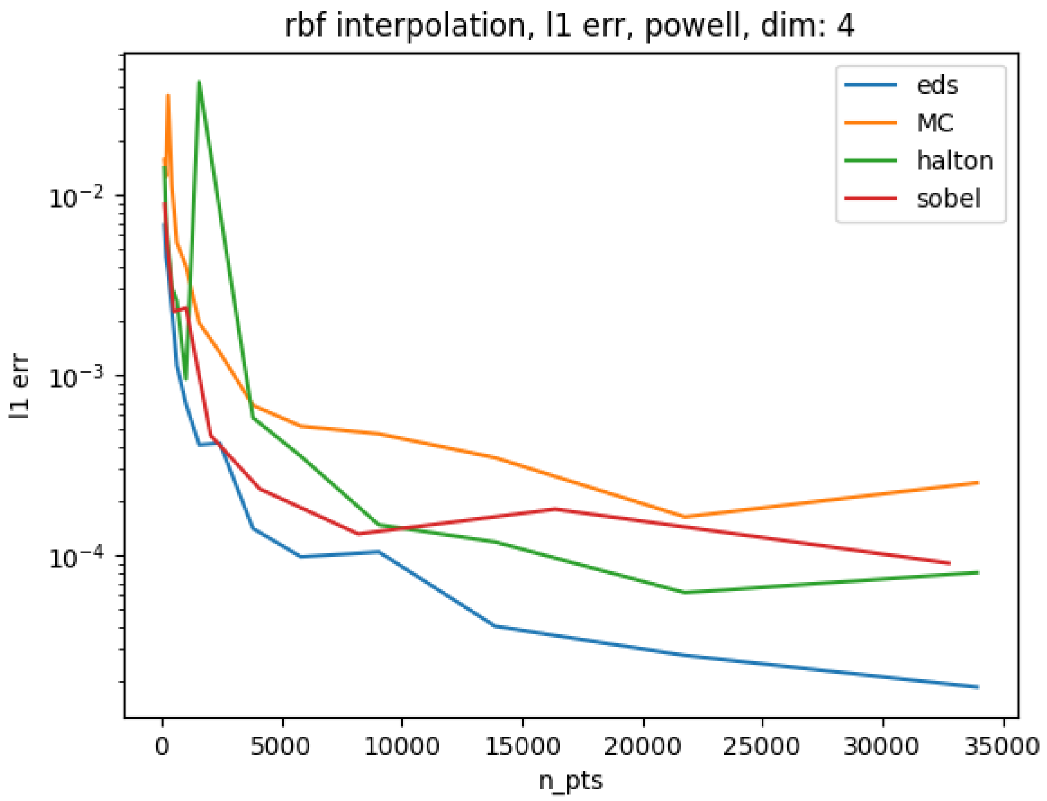
rbf interpolation, linf error, hartmann3d, dim: 3
linf err
n_pts
eds
MC
halton
sobel
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000


rbf interpolation, l1 err, hartmann6d, dim: 6
l1 err
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000

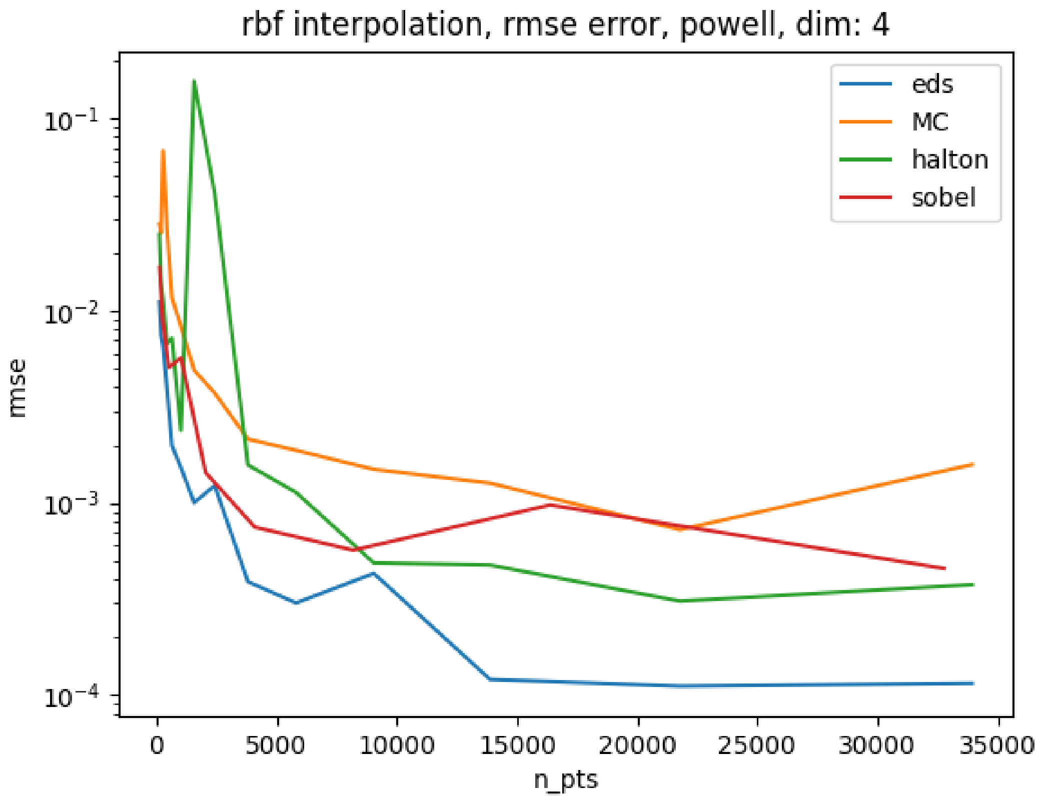

rbf interpolation, rmse error, hartmann6d, dim: 6
rmse
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000

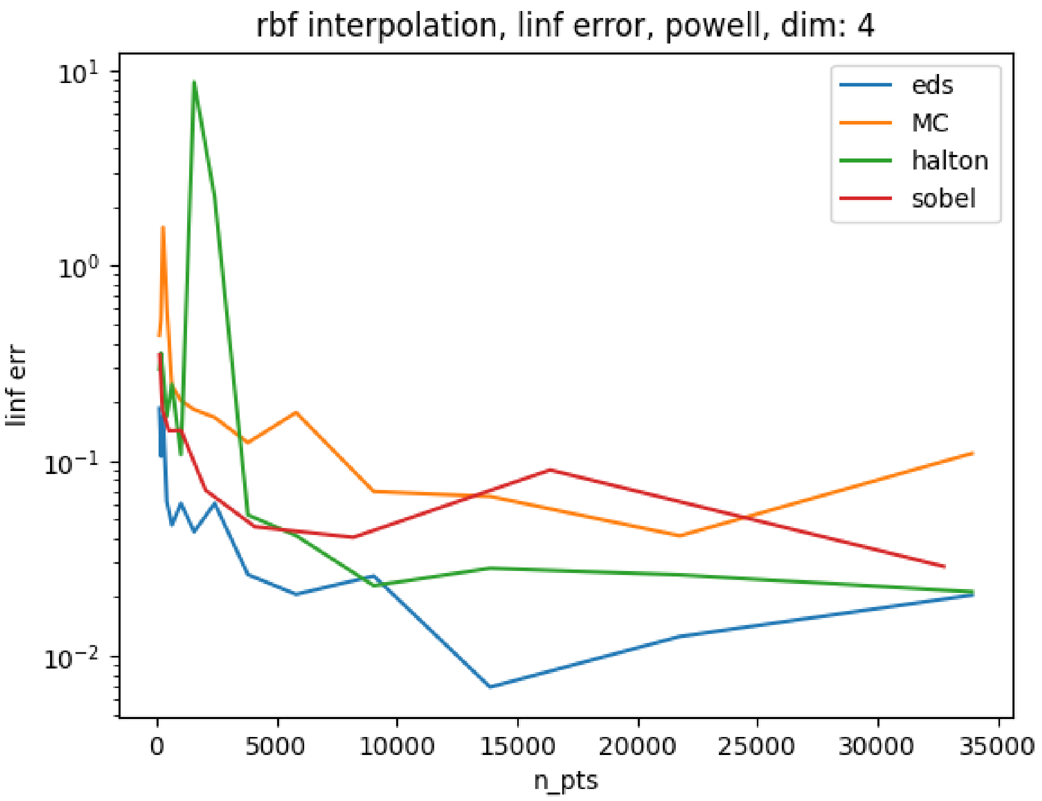


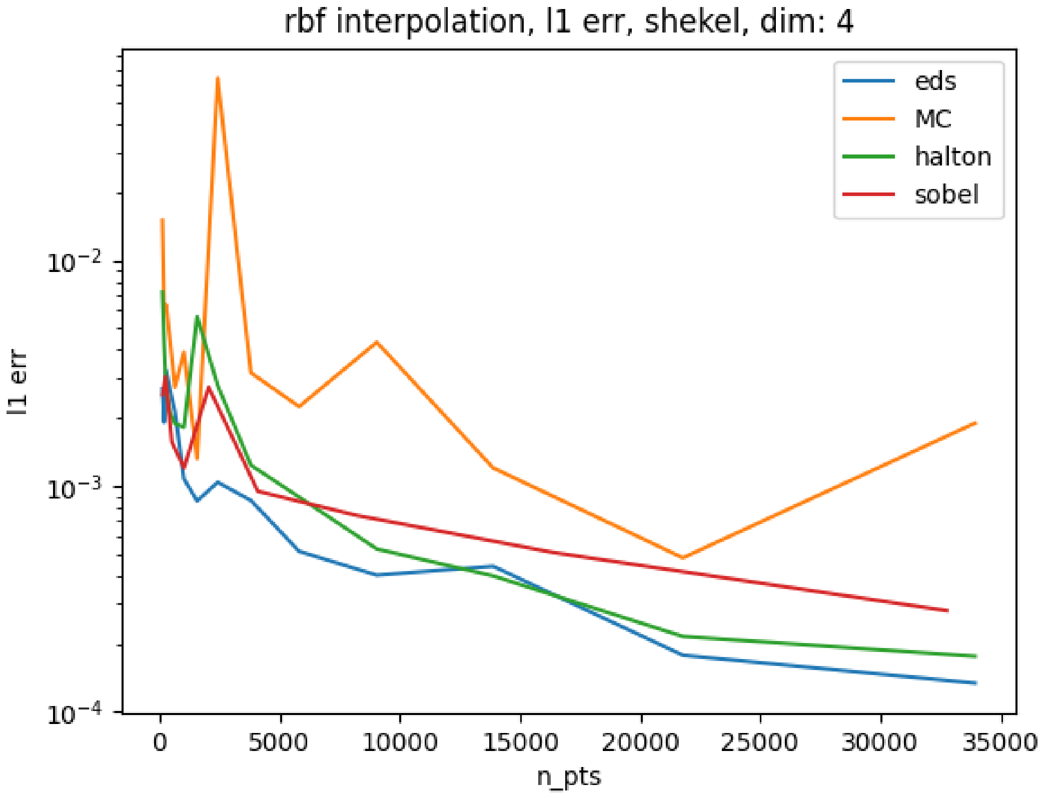
rbf interpolation, linf error, hartmann6d, dim: 6
eds
MC
halton
sobel
10^0
10^{-1}
linf err
0
5000
10000
15000
20000
25000
30000
35000
n_pts

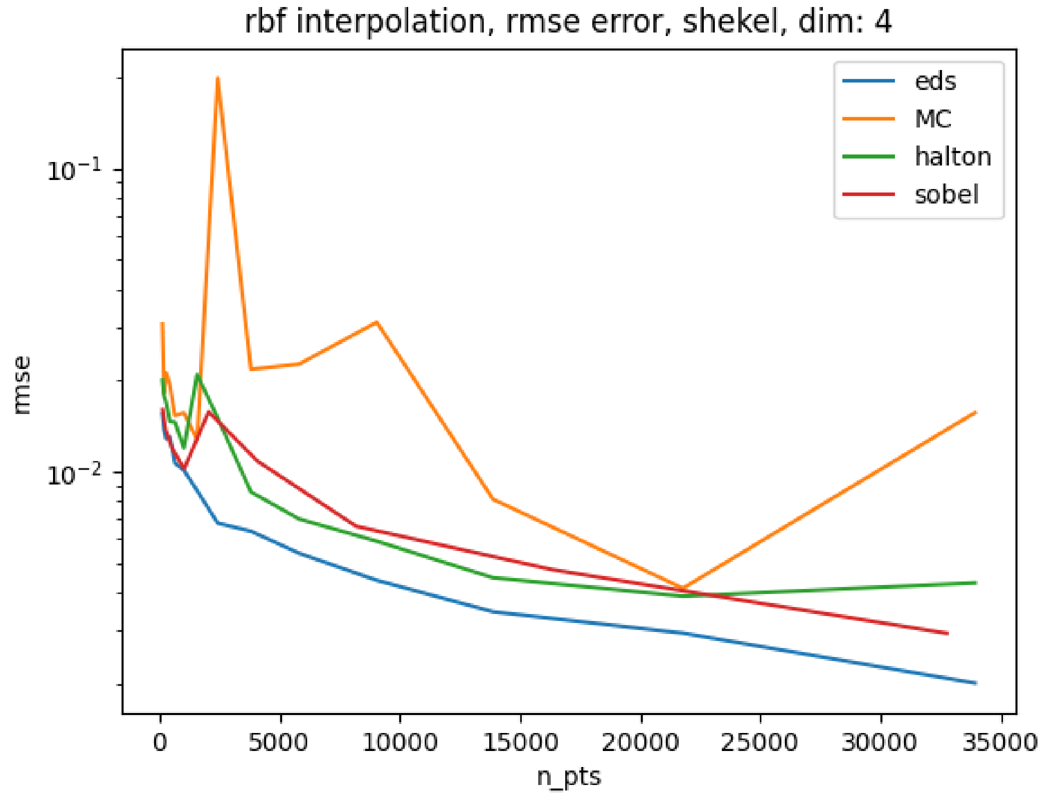


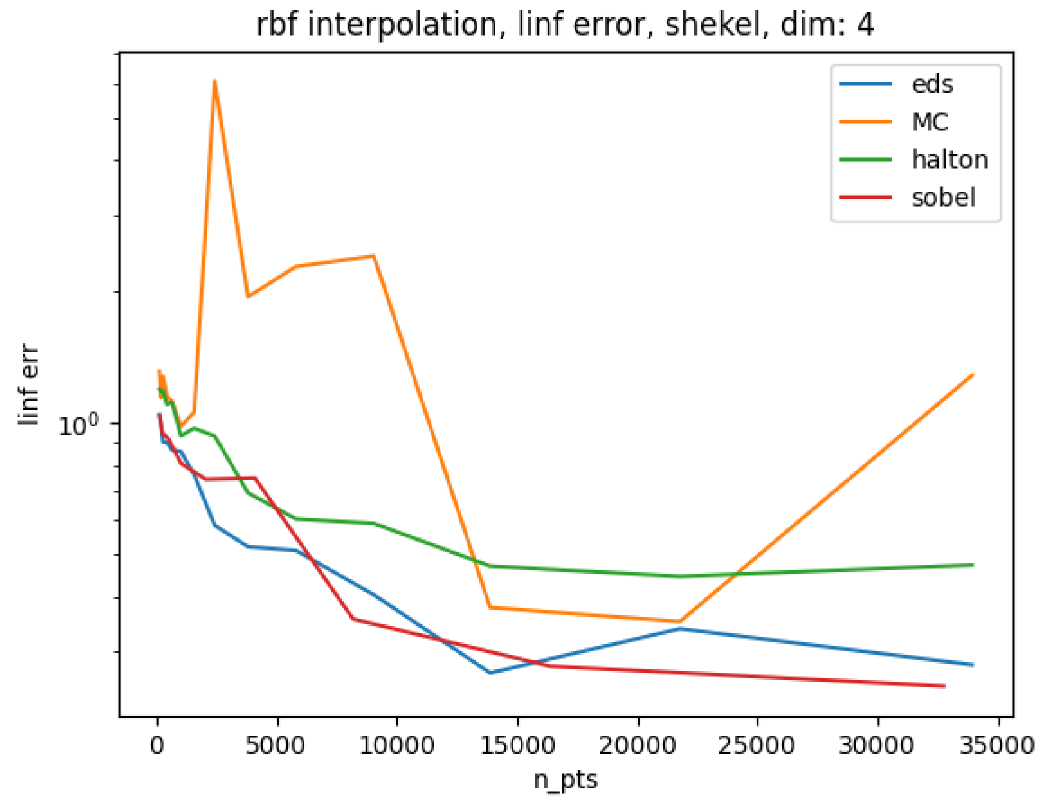
rbf interpolation, l1 err, powell, dim: 4
eds
MC
halton
sobel
10^{-2}
10^{-3}
10^{-4}
l1 err
0
5000
10000
15000
20000
25000
30000
35000
n_pts

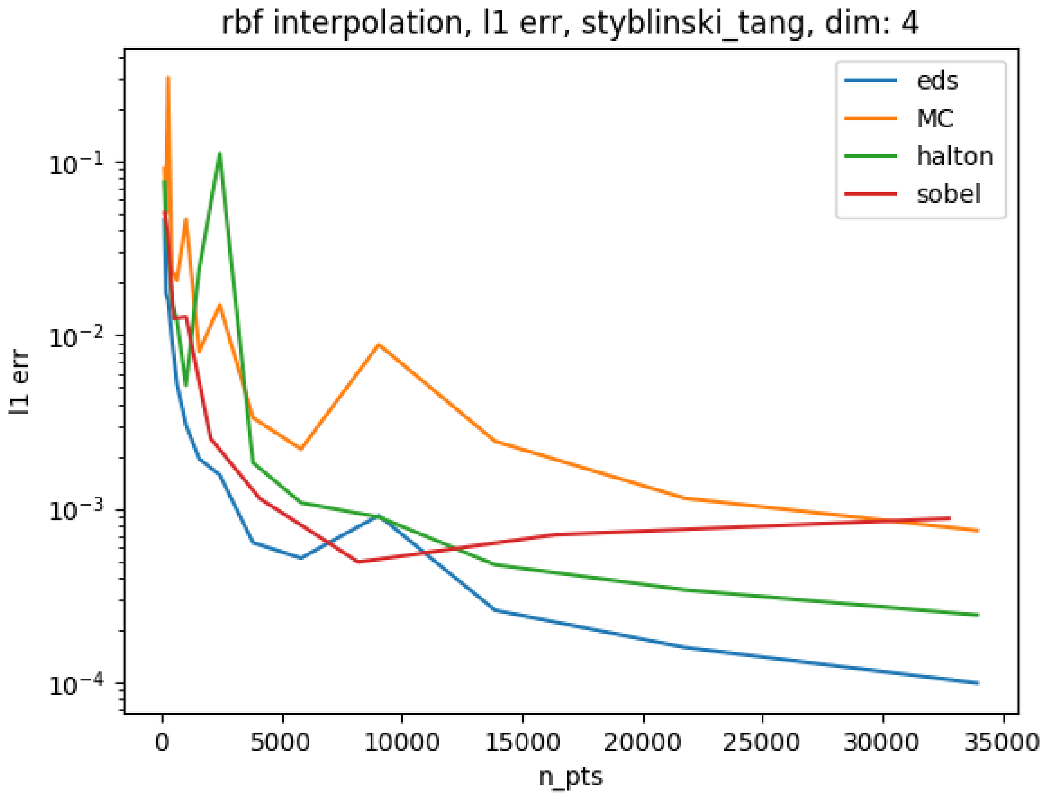

rbf interpolation, rmse error, powell, dim: 4
rmse
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5000
10000
15000
20000
25000
30000
35000

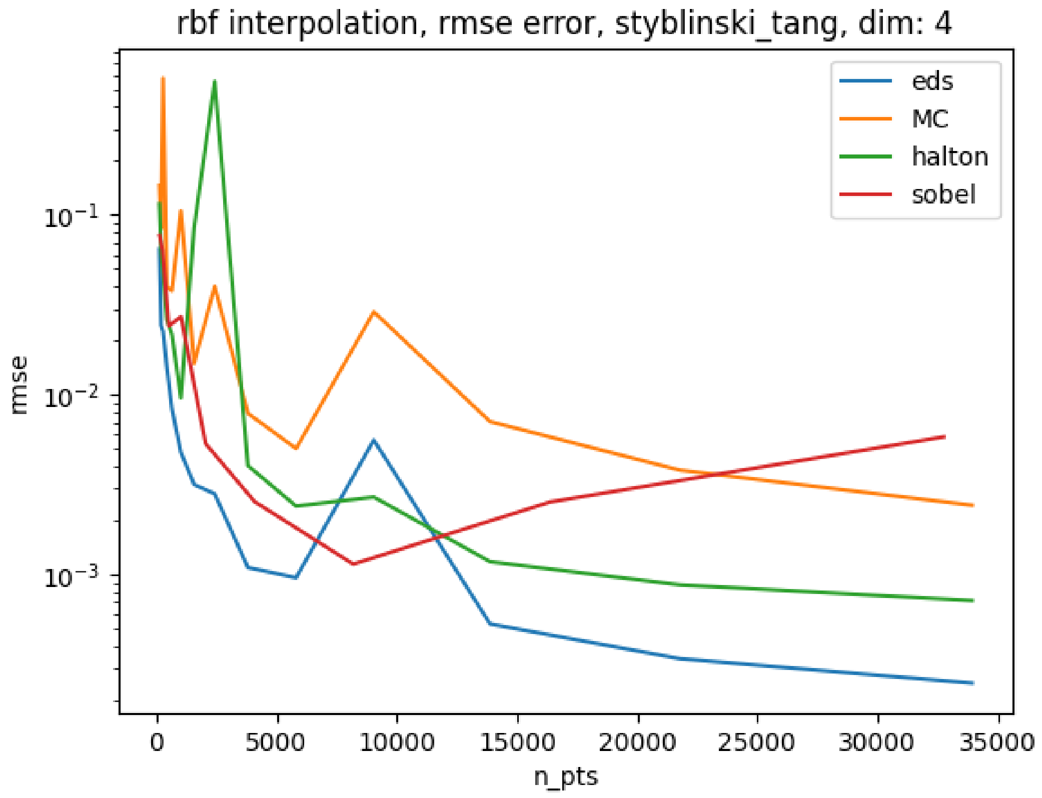

rbf interpolation, linf error, powell, dim: 4
linf err
n_pts
eds
MC
halton
sobel
10^{1}
10^{0}
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000

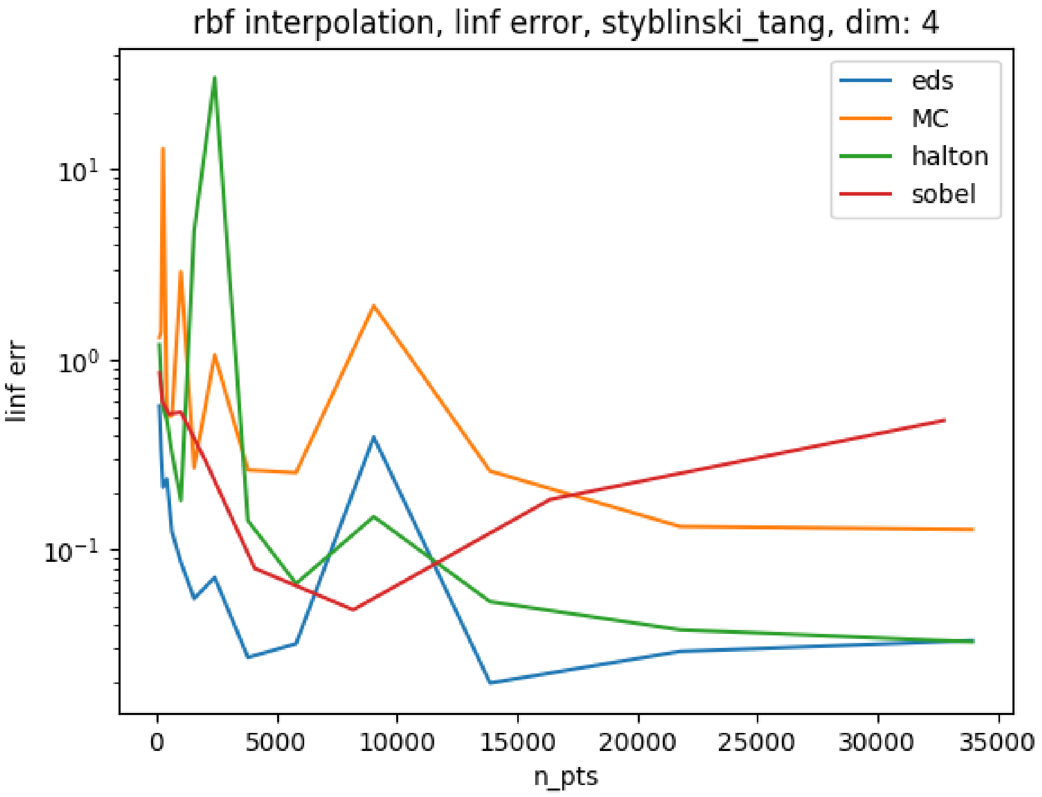

rbf interpolation, l1 err, shekel, dim: 4
l1 err
n_pts
eds
MC
halton
sobel
10^{-2}
10^{-3}
10^{-4}
0
5000
10000
15000
20000
25000
30000
35000


rbf interpolation, rmse error, shekel, dim: 4
rmse
n_pts
eds
MC
halton
sobel
10^{-1}
10^{-2}
0
5000
10000
15000
20000
25000
30000
35000


rbf interpolation, linf error, shekel, dim: 4
linf err
10^0
n_pts
0
5000
10000
15000
20000
25000
30000
35000
eds
MC
halton
sobel


rbf interpolation, l1 err, styblinski_tang, dim: 4
l1 err
10^{-1}
10^{-2}
10^{-3}
10^{-4}
n_pts
0
5000
10000
15000
20000
25000
30000
35000
eds
MC
halton
sobel


rbf interpolation, rmse error, styblinski_tang, dim: 4
eds
MC
halton
sobel
rmse
10^{-1}
10^{-2}
10^{-3}
0
5000
10000
15000
20000
25000
30000
35000
n_pts


rbf interpolation, linf error, styblinski_tang, dim: 4
eds
MC
halton
sobel
linf err
10^{1}
10^{0}
10^{-1}
0
5000
10000
15000
20000
25000
30000
35000
n_pts

6.4 Pseudocode for adaptive and nested eds

Input:
f, **D**: A callable function and the corresponding domain.
$\{\epsilon_i\}$ (will be represented by **eps_arr** in the pseudocode): EDS parameters for each refinement level i, which are the minimum-allowed Euclidean distances between each point.
tol: We add additional points near an existing point if the cross-validation error exceeds the tolerance on this points.
delta_rate: Parameter used to determine the refinement radius, which is calculated as:

$$delta\_i = delta\_rate * \epsilon_i$$

n_rep: Parameter to be passed to the local min-EDS generator, denoting the number of repeated runs among which a min-size eds refinement is chosen.
RBF_type: Indicating the type of RBF (i.e. Gaussian, Multiquadric, etc.)
scale_RBF: The radius parameter of the radius basis function for approximating the target function

6.5 Adaptive NEDS Algorithm

When doing the refinement, the algorithm tests whether each points can be accurately predicted by other data points, and add refinements near the ”bad points”. Each data points may be tested multiple times.

Following is the pseudocode of the algorithm. Note that NEDS, NEDS_values, bad_pts are all ordered sets that group the data according to the refinement level during which the point is added.

Algorithm Nested_EDS_with_RBF_v2

```
1: max_iter ← ||eps_arr||
2: NEDS← {∅, ∅, ...}
3: NEDS_values← {∅, ∅, ...}
4: bad_pts← {∅, ∅, ...}
5: for $i = 1, 2, \ldots$,max_iter do
6:     if i=1 then
7:         EDS_orig← gen_EDS(∅, D, eps_arr[i])
8:         vals_orig← f(EDS_orig)
9:         NEDS[i]← EDS_orig
10:        NEDS_values[i]← vals_orig
11:    else
12:        for j=1,2,...,i-1 do
13:            delta=delta_rate*eps_arr[j+1]
14:            EDS_new← gen_refine(NEDS, bad_pts[j], D, eps_arr[j+1], delta, n_reps)
15:            vals_new←f(EDS_new)
16:            NEDS[j+1]← NEDS[j+1]∪EDS_new
17:            NEDS_values[j+1]← NEDS_values[j+1]∪vals_new
18:        end for
19:    end if
20:    bad_pts←find_bad_v2(NEDS, NEDS_values, n_surrounding, n_bad, RBF_type,
       scale_rbf, i)
21: end for
22: coefs←RBF_interp(NEDS, NEDS_values, RBF_type, scale_rbf)
```

6.5.1 Subroutine that finds out the ”bad points”

```
1: function FIND_BAD_V2(NEDS, NEDS_values, n_surrounding, n_bad, RBF_type, scale_rbf,
   n_level)
2:     cross_valid_err← {∅, ∅, ...}
3:     for j=1,2,...,n_level do
4:         Allocate space for an empty array err_j of size ||NEDS[j]||
5:         for k=1,2,...,||NEDS[j]|| do
6:             $x_k$ ←NEDS[j][k]
7:             $y_k$ ←NEDS_values[j][k]
8:             pts_test← The nearest n_surrounding points near the tested point.
9:             idx_test←get_idx(NEDS,pts_test)
```

```
10:           vals_test← NEDS_values[idx_test]
11:           coef_temp←RBF_interp(NEDS, NEDS_values, RBF_type, scale_rbf)
12:           We find out the function value of $y_k$ ∈NEDS_values
13:           $y_{pred}$ ←RBF_pred(pts_test,coef_temp,$x_k$)
14:           err_test_temp← $|y_k - y_{pred}|$
15:           err_j[k]←err_test_temp
16:       end for
17:       cross_valid_err[j]←err_j
18:   end for
19:   bad_pts←find_max_err_pts(cross_valid_err,NEDS,n_bad)
20:   return bad_pts
21: end function
```

6.6 Subroutine: gen_refine

This subroutine calls gen_EDS to generate refinement points (which is a min-EDS set with a smaller epsilon value) around the "bad points".

```
1: function GEN_REFINE(NEDS_old, pts_bad, D, $\epsilon$, delta, n_reps)
2:     D_refine← $(\bigcup_{x_b \in pts\_bad} B_{delta}(x_b)) \cap D$
3:     min_EDS← $\emptyset$
4:     min_size← $\infty$
5:     for $i = 1, 2, \ldots,$ n_reps do
6:         Set a new seed for the random point generator.
7:         _, EDS_new←gen_EDS_incremental(NEDS_old, D_refine, $\epsilon$)
8:         if ||EDS_new|| <min_size then
9:             min_EDS←EDS_new
10:            min_size← ||EDS_new||
11:        end if
12:    end for
13:    return min_EDS
14: end function
```

6.7 Subroutine: gen_EDS_incremental

Given an existing set NEDS_old, an ϵ and a domain D_local, this subroutine fills the old NEDS set with new points that are at least ϵ distances from other points.

```
1: function GEN_EDS_INCREMENTAL(NEDS_old, D_local, $\epsilon$)
2:     NEDS_new←NEDS_old
3:     pts_new← $\emptyset$
4:     n_fails←0
5:     while n_fails≤ max{5*‖eds_set‖,1000} do
6:         Generate a random point $\hat{x}$ in the domain D_local
7:         add← True
8:         for $x_i \in$ NEDS_new do
9:             if $\|x_i - \hat{x}\| > \epsilon$ then
10:                add←False
11:                n_fails← n_fails+1
12:                break
13:            end if
14:        end for
15:        if add=True then
16:            NEDS_new←NEDS_new$\cup\{\hat{x}\}$
17:            pts_new←pts_new$\cup\{\hat{x}\}$
18:        end if
19:    end while
20:    return NEDS_new, pts_new
21: end function
```

Note on line 4: We require the algorithm to do at least 1000 trials in any cases to prevent large holes when ϵ is not small.

References

[1] Y.-D. Zhou, K.-T. Fang, and J.-H. Ning, “Mixture discrepancy for quasi-random point sets,” *Journal of Complexity*, vol. 29, no. 3, pp. 283–301, 2013.

[2] S. De Marchi, R. Schaback, and H. Wendland, “Near-optimal data-independent point locations for radial basis function interpolation,” *Advances in Computational Mathematics*, vol. 23, pp. 317–330, Oct 2005.

[3] B. Fornberg and J. Zuev, “The runge phenomenon and spatially variable shape parameters in rbf interpolation,” *Computers & Mathematics with Applications*, vol. 54, no. 3, pp. 379–398, 2007.

[4] M. Heidari, M. Mohammadi, and S. De Marchi, “Curvature based characterization of radial basis functions: application to interpolation,” *Mathematical Modelling and Analysis*, vol. 28, pp. 415–433, 09 2023.

[5] S. Rippa, “An algorithm for selecting a good parameter c in radial basis function interpolation,” *Advances in Computational Mathematics*, vol. 11, pp. 193–210, 11 1999.

[6] R. Schaback, “Optimal compactly supported functions in sobolev spaces,” 09 2024.

[7] S. De Marchi, F. Marchetti, and E. Perracchione, "Jumping with variably scaled discontinuous kernels (vsdks)," *BIT*, vol. 60, p. 441–463, June 2020.

[8] L. Romani, M. Rossini, and D. Schenone, "Edge detection methods based on rbf interpolation," *Journal of Computational and Applied Mathematics*, vol. 349, pp. 532–547, 2019.