



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

The Delaunay Density Diagnostic

A. K. Gillette, E. Kur


March 20, 2023

Association for Computing Machinery: Transactions on
Mathematical Software

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

THE DELAUNAY DENSITY DIAGNOSTIC

 **Andrew Gillette***

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550
gillette7@llnl.gov

 **Eugene Kur**

Strategic Deterrence
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550
kur1@llnl.gov

ABSTRACT

Accurate approximation of a real-valued function depends on two aspects of the available data: the density of inputs within the domain of interest and the variation of the outputs over that domain. There are few methods for assessing whether the density of inputs is *sufficient* to identify the relevant variations in outputs—i.e., the “geometric scale” of the function—despite the fact that sampling density is closely tied to the success or failure of an approximation method. In this paper, we introduce a general purpose, computational approach to detecting the geometric scale of real-valued functions over a fixed domain using a deterministic interpolation technique from computational geometry. The algorithm is intended to work on scalar data in moderate dimensions (2-10). Our algorithm is based on the observation that a sequence of piecewise linear interpolants will converge to a continuous function at a quadratic rate (in L^2 norm) if and only if the data are sampled densely enough to distinguish the feature from noise (assuming sufficiently regular sampling). We present numerical experiments demonstrating how our method can identify feature scale, estimate uncertainty in feature scale, and assess the sampling density for fixed (i.e. static) datasets of input–output pairs. We include analytical results in support of our numerical findings and have released lightweight code that can be adapted for use in a variety of data science settings.

Key words. function approximation, scale detection, interpolation, Delaunay triangulation

AMS subject classifications. 65D05, 65D04, 68W20

1 Introduction

Identification of sufficient sampling density is an essential and ongoing challenge in data science and function modeling. For any problem context, too little data raises concerns of over-fitting while too much data risks under-fitting and inefficient computational pipelines. While theorems and error estimates can provide rough bounds on requisite sampling density, more often density is selected by heuristics, trial and error, or rules of thumb.

In this work, we introduce the *Delaunay density diagnostic*: a computational technique that can help identify the presence and scale of geometric features in a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^1$ as values of f are iteratively collected. By geometric features we mean any non-linear behavior that is not noise, and hence the presence of a feature requires sufficient sampling density to distinguish it from noise. More formally, the goal of the diagnostic is to provide a robust computation of the rate at which a piecewise linear interpolant of f changes as additional sample points are incorporated in batches of user-specified size.

A simple illustration of the challenges present in this effort can be seen by examining the Griewank function on \mathbb{R}^1 ($d = 1$), defined by

$$g_1(x) := \frac{1}{4000}x^2 - \cos(x) + 1.$$

*<https://people.llnl.gov/gillette7>

At a large scale, e.g. $x \in [-10^4, 10^4]$, the quadratic term is dominant and only an extremely dense sampling of the interval would be able to distinguish the cosine term from random noise. At a small scale, e.g. $x \in [-1, 1]$, the cosine term is dominant and even a very dense sampling of the interval would be insufficient to pick up the quadratic term behavior. At intermediate scales, the interplay between sampling density and feature representation is more subtle, motivating the need for a computational approach to the problem.

We first describe the Delaunay density diagnostic in the simplest context and in a manner slightly distinct from how it is actually implemented. Assume we can rapidly compute the output of a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^1$ for any $\mathbf{x} \in \mathbb{R}^d$. For now, we assume there is no uncertainty or significant numerical error in the output computation, as can be the case when f is given by an analytical formula or a trained neural network of mild complexity. The following must then be specified by the user:

- A d -dimensional **bounding box** $\mathcal{B} \subset \mathbb{R}^d$. All data and computations will be confined to this box. In practice, it is best to rescale the box to a standard hypercube (such as $[0, 1]^d$), which would induce a similar rescaling on any data contained within.
- An initial set of **sample points** $\{\mathbf{x}_j\} \subset \mathcal{B}$. Ideally, these points are spread uniformly over \mathcal{B} . The initial Delaunay interpolant will be determined by this set.
- An **upsampling growth factor** $b \in (1, 2]$. Sample points will increase by b^d in each iteration, typically via uniform upsampling. Each iteration determines a finer-grained Delaunay interpolant than the previous.
- A set of **query points** $\{\mathbf{q}_i\} \subset \mathcal{B}$. The Delaunay interpolant will be evaluated at these points. To avoid too many query points falling outside the convex hull of the sample points (thus resulting in geometric extrapolation), the query points should be supported in a proper subset of \mathcal{B} . In general, the larger d is, the smaller the subset bounding the query points must be, but the nature of the sample points also plays a role (e.g. if the samples are intentionally clustered in some region).

In the simplest case, we choose the query points to be a regular, axis-aligned lattice of p^d points forming a cube in \mathbb{R}^d , for some p , and \mathcal{B} to be a box with the same center as the lattice but larger diameter. The main computation is an iterative procedure:

1. For each query point \mathbf{q}_i , compute and store $\hat{f}(\mathbf{q}_i)$, the unique Delaunay piecewise-linear interpolant defined by $\{\mathbf{x}_j, f(\mathbf{x}_j)\}$, evaluated at \mathbf{q}_i .
2. Let $\|\{\mathbf{x}_j\}\| = n$. Let ℓ be the integer closest to $\lceil b n^{1/d} - (b-1) \rceil^d - n$. Randomly sample ℓ more points from \mathcal{B} and add these points to the set of samples so that $\|\{\mathbf{x}_j\}\| = n + \ell$.
3. Repeat until $\|\{\mathbf{x}_j\}\|$ exceeds a specified threshold.

If some \mathbf{q}_i lies outside the convex hull of the $\{\mathbf{x}_j\}$ at a step of the process, the Delaunay interpolant is not defined and we just store a nan value. This case can be avoided by increasing the diameter of \mathcal{B} or increasing the initial number of points in $\{\mathbf{x}_j\}$.

Observe that the values of $\hat{f}(\mathbf{q}_i)$ may change at each step, since the introduction of new samples may change the sample points defining the computation of $\hat{f}(\mathbf{q}_i)$. We use this observation to compute an approximate rate of convergence of the interpolants to the true function f over the region defined by the query points. Let $\hat{f}_k(\mathbf{q}_i)$ denote the interpolated value at \mathbf{q}_i at the k th step of the iterative procedure. For each $k \geq 2$, we compute a rate r_k defined by

$$r_k := \log_b \left(\frac{\sqrt{\text{MSD}_i(\hat{f}_{k-1}(\mathbf{q}_i) - \hat{f}_{k-2}(\mathbf{q}_i))}}{\sqrt{\text{MSD}_i(\hat{f}_k(\mathbf{q}_i) - \hat{f}_{k-1}(\mathbf{q}_i))}} \right).$$

Here, $\text{MSD}_i(\cdot) = \frac{1}{|\{\mathbf{q}_i\}|} \sum_i (\cdot)^2$ denotes the mean squared difference over the index i , which provides a measure of the difference between successive interpolants in the procedure; the square root of the mean squared difference is a discretization of the L^2 norm. The definition of r_k is inspired by the computation of convergence rates for finite element methods over a sequence of increasingly refined meshes [Brenner and Scott(1994)]. We now make the following claims, which will be supported by numerical and analytical results later in the paper:

Claim 1 *If the set $\{(\mathbf{x}_j, f(\mathbf{x}_j))\}$ used to define \hat{f}_k contains sufficient information to reconstruct the geometric features present in f over the set of query points $\{\mathbf{q}_i\}$ then the mean squared difference rate r_k will be approximately 2.*

Claim 2 *If the set $\{(\mathbf{x}_j, f(\mathbf{x}_j))\}$ used to define \hat{f}_k cannot distinguish geometric features from random noise over the set of query points $\{\mathbf{q}_i\}$ then the mean squared difference rate r_k will be approximately 0.*

We envision the results of these claims being relevant to the multitude of scientific machine learning problems in contemporary literature where the goal is to approximate some unknown function based on unstructured numerical data. Many such problems have inputs in \mathbb{R}^d for $d > 2$, making visualization difficult, but not $d \gg 2$, making techniques designed for high-dimensional data not necessarily applicable. In such settings, there are few computational techniques for robust identification of sufficient sampling density and hence the proposed diagnostic (which works best for $2 < d < 10$) could aid in determining whether a given sampling procedure is insufficient, sufficient, or excessive in the context of a specific problem or application. We note, however, that sampling density is important throughout computational mathematics and the technique has no intrinsic relation to machine learning.

The diagnostic effectively evaluates nonlinear feature presence at different scales, but has several limitations that need to be considered:

- The Delaunay triangulation becomes expensive to construct in higher dimensions, even partially, which limits the maximum dimension to around 10.
- Much of our work on the diagnostic assumes (nearly) uniform sampling for the sample points. For non-uniform sampling, the “features” detected by the diagnostic would be a mix of features of the function and the sampling density. This could potentially be resolved by using a norm adjusted for the sampling density, but we leave such considerations for future work. If the sampling density is very non-uniform (such as if data lies on a lower-dimensional manifold), this could significantly degrade the performance of the Delaunay interpolant and prevent computation of the diagnostic. Of note, the software we will use for computing the Delaunay interpolant can sometimes detect if the data lies on or nearly on a lower-dimensional manifold.
- The diagnostic does not distinguish between geometric features present in all coordinates and those present in only some coordinates. For example, suppose a 4-dimensional function only depended on two of the four coordinates and the features had characteristic size ℓ . For a bounding box with side length L , the diagnostic would require somewhat more than $(L/\ell)^4$ sample points to resolve the features, while an algorithm that could take advantage of the lower dimensionality could fit the function well with only somewhat more than $(L/\ell)^2$ points. In such cases, the diagnostic is too conservative, claiming more points are needed to adequately resolve features even when appropriate algorithms can fully resolve them without additional sampling.

2 Context in the literature

Techniques for fitting a function to unstructured numerical data have been studied in multiple disciplines for many decades. Some prominent examples include the Kriging interpolation method [Sacks et al.(1989), Simpson et al.(2001), Van Beers and Kleijnen(2004)], radial basis functions [Buhmann(2000), Buhmann(2003)], response surface methodology [Khuri and Mukhopadhyay(2010)], and generalized linear models [Nelder and Wedderburn(1972), McCullagh and Nelder(2019)]. Interpolation based on “nearest neighbors” is appealing for its simplicity, but has not emerged as a feasible competitor to the above methods. In general, methods have been developed to serve specific needs in engineering and statistics communities but have trouble scaling to the size and dimension of modern datasets.

Perhaps the most closely related research area—and in fact the inspiration for this work—is the notion of data oscillation in finite element methods (FEM); see e.g. [Morin et al.(2000)]. In FEM, the goal is to approximate the solution u to a partial differential equation $Du = f$ by creating a piecewise polynomial with respect to a mesh of the domain. Data oscillation, $osc_{\mathcal{M}}$, is a quantity that measures the variation of f over a fixed mesh \mathcal{M} . If $osc_{\mathcal{M}}$ is large, f has fine scale geometric structures that cannot be resolved by approximation with respect to \mathcal{M} , meaning a finer mesh must be used. Hence, attaining optimal rates of convergence in adaptive FEM (in which meshes are partially refined in an iterative process) depends on controlled data oscillation; control can be attained by assumption, by luck, or by computational detection and mesh refinement, often based on the classical theory of Richardson extrapolation [Richardson(1911), Richardson and Gaunt(1927)].

The issue of function-dependent sampling density requirements is by no means restricted to the finite element world. Metrics to assess data variation with respect to sample points have been devised for application-specific contexts but have no standard nomenclature. These include the “grid convergence index” [Celik and Karatekin(1997)] and “index of resolution quality” [Celik et al.(2005)] for computational fluid dynamics, “local feature size” for homeomorphic surface reconstruction from point clouds [Amenta et al.(2000)], and the “coefficient of variation” for LIDAR sampling [Guo et al.(2010)]. In machine learning communities, various notions of “holdout sets” and “cross validation” are in vogue [Blum et al.(1999), Bengio and Grandvalet(2003), Bates et al.(2023)]. The list could go on.

This work introduces a methodology for assessing data variation on unstructured numerical data with input dimension d up to ≈ 10 . Our approach is not meant to generalize or replace the application-specific techniques and metrics described above. Rather, we aim to provide users working with high volume but relatively low-dimensional numerical datasets a means to assess data variation and sampling density in a rigorous framework, subject to clearly stated assumptions. Such datasets are now ubiquitous in scientific disciplines, but analyzing function variation with respect to a mesh is often stymied by the so-called “curse of dimensionality.” For $d > 3$, it quickly becomes infeasible to compute, store, or manipulate the complete mesh structure of a collection of unstructured data points.

As we will demonstrate, the lack of scalability of mesh management can be circumvented for interpolation tasks if the interpolated value can be determined using only a sparse subset of an implied—but not computed—mesh data structure. Delaunay theory provides the requisite mathematical results for an implied mesh structure and the recently developed algorithm `DelaunaySparse` provides a practical tool for such computations.

2.1 Delaunay interpolation

Let $f : \mathbb{R}^d \rightarrow \mathbb{R}^\ell$ be a multivariate function whose outputs are known at a collection of n data points $\mathcal{D} \subset \mathbb{R}^d$. Assume that \mathcal{D} is truly a d -dimensional sample in the sense that it does not lie entirely in a hyperplane of \mathbb{R}^d ; we remark that if there was such a hyperplane, the interpolation algorithm would detect this. Then, the *convex hull* of \mathcal{D} , denoted $CH(\mathcal{D})$, is a d -dimensional, flat-faced, convex region in \mathbb{R}^d . The *Delaunay triangulation*, denoted $DT(\mathcal{D})$, is an unstructured mesh of $CH(\mathcal{D})$ consisting of d -simplices with vertices in \mathcal{D} that satisfy the open circumball property, which is described in the caption of Figure 1. The Delaunay triangulation exists and is unique, except for some corner cases, which can still be handled robustly. For instance, data stored on a rectilinear lattice does not have a unique Delaunay triangulation, but in such situations, the geometry of the data is known *a priori*. For a randomly generated data set in \mathbb{R}^d , as is employed here, the probability of a unique Delaunay triangulation is 1.

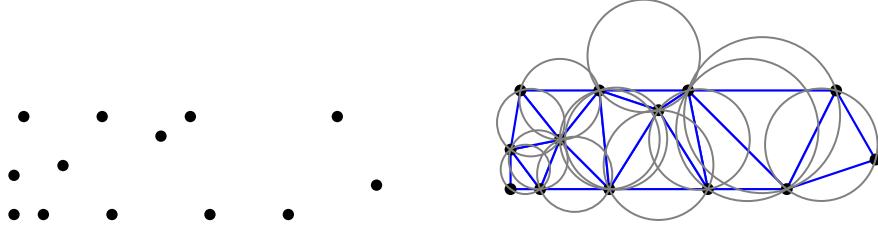


Figure 1: An arbitrary collection of points \mathcal{D} in \mathbb{R}^2 (left) has a unique triangular mesh (right, blue lines), called the Delaunay triangulation. Every triangle in the Delaunay mesh satisfies the “empty ball criterion”: the open circumball whose boundary passes through the vertices of the triangle does not contain any points from \mathcal{D} . The boundaries of the circumballs for the Delaunay triangulation in the figure above are shown as grey circles. For $d > 2$, these properties generalize to meshes of d -simplices and associated d -dimensional circumballs.

The Delaunay triangulation can be used to define a unique piecewise linear interpolant, called the *Delaunay interpolant*, of the values $f(\mathcal{D})$. The Delaunay triangulation is widely considered to be an optimal simplicial mesh for the purposes of multivariate function interpolation [Chen and Xu(2004), Rajan(1994)], and the Delaunay interpolant (defined below) has been studied in the context of both regression [Liu and Yin(2019), Lux et al.(2021)] and classification [Belkin et al.(2018)] problems.

Given any point $\mathbf{q} \in CH(\mathcal{D})$, the *Delaunay interpolant* at \mathbf{q} is defined as follows. Let \mathcal{S} be the simplex in $DT(\mathcal{D})$ with vertices $\mathbf{s}_1, \dots, \mathbf{s}_{d+1} \in \mathcal{D}$ such that $\mathbf{q} \in \mathcal{S}$. Then there exist non-negative weights $w_1, \dots, w_{d+1} \in \mathbb{R}$ such that $\mathbf{q} = \sum_{i=1}^{d+1} w_i \mathbf{s}_i$ and $\sum_{i=1}^{d+1} w_i = 1$. The value of the Delaunay interpolant \hat{f}_{DT} at \mathbf{q} is given by

$$\hat{f}_{DT}(\mathbf{q}) := w_1 f(\mathbf{s}_1) + w_2 f(\mathbf{s}_2) + \dots + w_{d+1} f(\mathbf{s}_{d+1}). \quad (1)$$

The approximation defined by $\hat{f}_{DT}(\mathbf{q})$ is a continuous, piecewise linear interpolant of f for \mathcal{D} . In particular, if \mathbf{q} lies at the interface of multiple d -simplices in $DT(\mathcal{D})$, the value of $\hat{f}_{DT}(\mathbf{q})$ is not dependent on the choice of simplex used to compute it.

2.2 The DelaunaySparse algorithm

While computing a data structure for the full $DT(\mathcal{D})$ is not computationally feasible beyond very low dimensions, the value of $\hat{f}_{DT}(\mathbf{q})$ only requires detection of a simplex in $DT(\mathcal{D})$ that contains \mathbf{q} . `DelaunaySparse` is a recently

developed algorithm and software package that exploits this observation to provide efficient computation of the Delaunay interpolant in high dimensions at a user-provided set of input points. Algorithm 1 outlines the general strategy of DelaunaySparse; further details can be found in [Chang et al.(2020), Chang et al.(2018)].

Algorithm 1 DelaunaySparse [Chang et al.(2020)]

```

1:  $\mathcal{D}$  contains  $n$  points in  $\mathbb{R}^d$ 
2:  $f(\mathcal{D})$  contains values of  $f(\mathbf{x}_j)$  for all  $\mathbf{x}_j \in \mathcal{D}$ 
3:  $\mathbf{q} \in CH(\mathcal{D})$  is an interpolation point
4: Set  $\tilde{\mathbf{x}} \leftarrow \operatorname{argmin}_{\mathbf{x}_j \in \mathcal{D}} \|\mathbf{x}_j - \mathbf{q}\|$   $\triangleright \tilde{\mathbf{x}}$  is the closest point in  $\mathcal{D}$  to  $\mathbf{q}$ 
5: Find a  $d$ -simplex  $\mathcal{S}$  in  $DT(\mathcal{D})$  incident to  $\tilde{\mathbf{x}}$ 
6: while  $\mathbf{q} \notin \mathcal{S}$  do
7:   Select the facet  $\mathcal{F}$  of  $\mathcal{S}$  from which  $\mathbf{q}$  is visible
8:   Complete a new  $d$ -simplex  $\mathcal{S}^*$  from the facet  $\mathcal{F}$ 
9:   Update  $\mathcal{S} \leftarrow \mathcal{S}^*$ 
10: end while
11: Since the loop has terminated,  $\mathbf{q} \in \mathcal{S}$ 
12: return  $\hat{f}_{DT}(\mathbf{q})$   $\triangleright$  Computed according to (1), using  $f(\mathcal{D})$ 
    
```

Note that the cost to build the initial seed simplex is $\mathcal{O}(nd^3)$, where $n = \|\mathcal{D}\|$, and the cost to compute each subsequent simplex is $\mathcal{O}(nd^2)$. Therefore, the total cost of Algorithm 1 is $\mathcal{O}(nd^3 + nd^2p)$, where p is the number of flips required. Empirically, for uniformly distributed \mathcal{D} , p tends to be a super-linear but sub-quadratic function of n and independent of d [Chang et al.(2018)]. In particular, typically $p \gg d$ so that Algorithm 1 is effectively $\mathcal{O}(nd^2p)$.

While Algorithm 1 only allows interpolation points in the geometric sense, it can also extrapolate a value $f(\mathbf{y})$ for a point \mathbf{y} outside $CH(\mathcal{D})$. For this, the point \mathbf{y} is projected onto the convex hull by solving a quadratic programming problem, and the resulting projection $\hat{\mathbf{y}} \in CH(\mathcal{D})$ is interpolated on the face of a simplex in $DT(\mathcal{D})$ via Algorithm 1. In this work, we detect and flag when a query point corresponds to geometric extrapolation for a given sample set, but intentionally set up experiments so that extrapolation does not occur.

3 Computing the Delaunay density diagnostic

We now describe in detail how the Delaunay density diagnostic is computed. As outlined in Section 1, the user specifies the following: a set of **query points** $\{\mathbf{q}_i\} \subset \mathbb{R}^d$; a d -dimensional **bounding box** $\mathcal{B} \subset \mathbb{R}^d$, such that $\{\mathbf{q}_i\} \subset \mathcal{B}$; an initial set of **sample points** $\{\mathbf{x}_j\} \subset \mathbb{R}^d$, drawn randomly from \mathcal{B} ; and an **upsampling growth factor** $b \in (1, 2]$. In addition, a stopping criterion should be specified, in the form of a maximum number of upsampling iterations and/or a maximum size for the set of sample points. The algorithm is as follows:

Algorithm 2 Delaunay density diagnostic (for MSD rate)

```

1:  $k=0$ 
2: while stopping criteria not met do
3:    $n_k \leftarrow \|\{\mathbf{x}_j\}\|$ 
4:    $\hat{f}_k(\mathbf{q}_i) \leftarrow$  results of Algorithm 1 with  $\mathcal{D} = \{\mathbf{x}_j\}$ , for each  $\mathbf{q} \in \{\mathbf{q}_i\}$ 
5:   if  $k > 0$  then
6:      $\text{diff}_{k,i} \leftarrow (\hat{f}_k(\mathbf{q}_i) - \hat{f}_{k-1}(\mathbf{q}_i))$ , for each  $\mathbf{q} \in \{\mathbf{q}_i\}$ 
7:   end if
8:   if  $k > 1$  then
9:      $r_k \leftarrow \log_b \left( \frac{\sqrt{\text{MSD}_i(\text{diff}_{k-1,i})}}{\sqrt{\text{MSD}_i(\text{diff}_{k,i})}} \right)$ 
10:  end if
11:   $\ell \leftarrow \text{round} \left( \left[ b n_k^{1/d} - (b-1) \right]^d - n_k, 0 \right)$ 
12:  Generate  $\ell$  points from  $\mathcal{B}$  randomly and add them to the collection  $\{\mathbf{x}_j\}$ 
13:   $k \leftarrow k + 1$ 
14: end while
15: return  $\{n_k, r_k\}$ 
    
```

Each output (n_k, r_k) from Algorithm 2 is an estimate of the rate (r_k) at which piecewise linear interpolants are converging to f at the query points $\{q_i\}$ when n_k samples are drawn from \mathcal{B} . Since Algorithm 2 involves random sampling, we can run it multiple times with different initial random seeds to generate a distribution of outputs. A collection of outputs $\{n_k, r_k\}_1, \{n_k, r_k\}_2, \dots$ can then be used to assess the accuracy of the estimates and test sensitivity to the randomized aspect of the algorithm.

Computing rates in other norms To compute the rate in a different norm than mean squared difference (MSD), only a few lines from Algorithm 2 need to be modified. An informative alternate rate to consider is the mean squared difference of the *gradients* of the interpolants, presented in Algorithm 3. We call this rate grad-MSD, for clarity, but note that it is essentially a discrete version of the Sobolev semi-norm H^1 , just as the square root of MSD is a discrete version of the L^2 norm.

Algorithm 3 Delaunay density diagnostic (for grad-MSD rate)

```

1: ...
4:  $\nabla \hat{f}_k(q_i) \leftarrow$  results of Algorithm 1 with  $\mathcal{D} = \{x_j\}$ , for each  $q \in \{q_i\}$ 
5: if  $k > 0$  then
6:    $\text{diff}_{k,i} \leftarrow (\nabla \hat{f}_k(q_i) - \nabla \hat{f}_{k-1}(q_i))$ , for each  $q \in \{q_i\}$ 
7: end if
8: if  $k > 1$  then
9:    $r_k \leftarrow \log_b \left( \frac{\text{norm}(\text{diff}_{k-1,i})}{\text{norm}(\text{diff}_{k,i})} \right)$ 
10: end if
14: ...
15: return  $\{n_k, r_k\}$ 
    
```

We briefly comment on two key changes from Algorithm 2 to Algorithm 3. First, the calculation of $\nabla \hat{f}_k(q_i)$ in line 4 can be done accurately and robustly since \hat{f}_k is piecewise linear. Details of this computation are provided in Appendix A. Second, we interpret the term `norm` used in line 9 as the regular Euclidean norm for vectors in \mathbb{R}^d , which is available as `numpy.linalg.norm` in Python.

We have equivalent claims to Claim 1 and Claim 2 for the grad-MSD rate; the only change is the expected value for r_k in each case, namely, 1 replaces 2 and -1 replaces 0.

Claim 3 *If the set $\{(x_j, f(x_j))\}$ used to define \hat{f}_k contains sufficient information to reconstruct the geometric features present in f over the set of query points $\{q_i\}$ then the grad-MSD rate r_k will be approximately 1.*

Claim 4 *If the set $\{(x_j, f(x_j))\}$ used to define \hat{f}_k cannot distinguish geometric features from random noise over the set of query points $\{q_i\}$ then the grad-MSD rate r_k will be approximately -1.*

Other norms could be considered as well. For instance, computing rates in L^p for $p \in [1, \infty)$ could be assessed by replacing MSD with “mean p th power differences” and the square roots with p th roots in line 9 from Algorithm 2. The L^∞ rate could be similarly accommodated. Likewise, convergence in the Sobolev norm $W^{1,p}$ can be attained by using the L^p norm instead of the L^2 norm for the term `norm` used in line 9 of Algorithm 3. We leave study of these possibilities for future work.

It is important to note that the point of using a linear interpolant is not because it is necessarily the best fit for the data. Indeed, there are many algorithms that will likely fit the data “better,” in the sense of having lower generalization error. We use a linear interpolant here because it has a known convergence rate for smooth data, which is required to serve as a diagnostic. Most algorithms, especially ones requiring any type of iterative fitting or training procedure, do not have known convergence rates and thus cannot be used as intended here. Further, since we are only interested in the convergence *rate* from successive approximations, the quality of the interpolant as a surrogate function is irrelevant. We choose the Delaunay linear interpolant because of its unique definition for arbitrary datasets and for the speed with which DelaunaySparse is able to work in moderate dimensions.

4 Numerical results

We have implemented and tested the feasibility of the Delaunay density diagnostic in various scenarios for data in \mathbb{R}^d from $d = 2$ to $d = 5$. After collecting the data, we convert the number of samples at the k th step, n_k , to average sample

spacing by

$$\langle \text{samp} \rangle := \text{average sample spacing} := \frac{L}{n_k^{1/d}}, \quad (2)$$

where L is the average side length of the bounding box \mathcal{B} . For simplicity, we have always taken \mathcal{B} to be a d -dimensional cube in \mathbb{R}^d ; the definition of L could be modified if a more complicated choice of \mathcal{B} was desired. The values of average sample spacing are used for the horizontal axes in our figures. Thus, the smallest average sample spacing in each plot corresponds to the *largest* n_k value attained before the stopping criteria.

To assess the sensitivity of Algorithm 2 and Algorithm 3 to the location of points $\{\mathbf{x}_j\}$, we use a seed to initialize a random number generator. At the start of the code, we set `rng=numpy.random.default_rng(globalseed)`, where `globalseed` is a user-specified integer; we then use `rng` to generate samples $\{\mathbf{x}_j\}$ subsequently in the code. After running the code with multiple seeds, we report the “mean rate” computed over all seeds, as well as the inter-quartile and inter-decile range for the computed rates, thereby providing an estimate of uncertainty in the computation.

Distinguishing features from noise We begin by demonstrating that the rate computations can reliably distinguish features from noise for analytic functions $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, shown in Figures 2 and 3. For these examples, we set the query points $\{\mathbf{q}_i\}$ to be a 20×20 uniformly spaced grid including the corners of $[-10, 10]^2$ for (a) and (b), $[-100, 100]^2$ for (c), and $[-1000, 1000]^2$ for (d). The bounding box \mathcal{B} is taken to be a square centered at the origin with side length 25% longer than the query point lattice, e.g., $\mathcal{B} = [-12.5, 12.5]^2$ for (a). The upsampling growth factor b is 1.4641. We initialize Algorithm 2 and Algorithm 3 with $\|\{\mathbf{x}_j\}\| = 9$ and use a stopping criterion of $\|\{\mathbf{x}_j\}\| > 200,000$.

When calling `DelaunaySparse` (Algorithm 1), we compute and pass values of $f(\{\mathbf{x}_j\})$ for the current collection of sample points $\{\mathbf{x}_j\}$. In (a), we assign $f(\mathbf{x}_j)$ to be a random number drawn uniformly from $[-1, 1]$. In (b)–(d), we assign $f(\mathbf{x}_j) := g_2(\mathbf{x}_j)$, where g_2 is the Griewank function [Griewank(1981)] on \mathbb{R}^2 , given by:

$$g_d(x_1, \dots, x_d) := \sum_{i=1}^d \frac{x_i^2}{4000} - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1. \quad (3)$$

The top rows of Figures 2 and 3 show a visualization of f for each case, over a domain matching the region defined by the query points. The middle and bottom rows show the MSD and `grad-MSD` rates, respectively, as a function of average sample spacing. Here, we have filtered out (n_k, r_k) outputs from the code with $n_k < 500$, i.e. larger average sample spacings. These outputs had larger variations in r_k values due to the small number of points involved, which distracted from the success of the method for larger n_k values. We will discuss this issue further later. Thus, in Figures 2 and 3, the average sample spacing values correspond to $n_k \in \{173832, 81275, 38039, 17830, 8376, 3947, 1869, 891\}$, from left to right, in each graph. The captions of Figures 2 and 3 describe high level takeaway messages from each experiment.

Effect of the upsampling factor b We next examine the effect of varying b , the parameter that controls the rate at which samples are added during each iteration (line 11 from Algorithm 2). For this experiment, we fix f to be the Ackley function [Ackley(1987)], visualized in Figure 4, given by the formula

$$f_{\text{(Ackley)}}(x, y) := -20 \exp\left(-0.2\sqrt{\frac{x^2 + y^2}{2}}\right) - \exp\left(\frac{\cos(2\pi x) + \cos(2\pi y)}{2}\right) + 20 + e.$$

Like the Griewank function, the Ackley function is commonly used to test optimization algorithms due to its dense collection of local extrema at a fine scale. We again fix the query points $\{\mathbf{q}_i\}$ to be a 20×20 uniformly spaced grid including the corners of $[-10, 10]^2$ and set $\mathcal{B} = [-12.5, 12.5]^2$. We initialize each trial with $\|\{\mathbf{x}_j\}\| = 9$ and use a stopping criterion of $\|\{\mathbf{x}_j\}\| > 1,000,000$.

We carried out 100 trials for three different values of the upsampling growth factor b : 1.1, 1.21, and 1.4641, shown in Figure 5. As in the previous example, we filter out results with very small n_k values (in this case $n_k < 20$) due to the wide variation in r_k values, as evidenced by the larger error bars on the right side of each graph. Regardless, the trend picked up by the mean rate in each experiment is quite clear: the Ackley function has many small oscillations with a period of about 1 unit in each coordinate, so we can fully resolve the function as long as the sample density has $\langle \text{samp} \rangle \ll 1$. Figure 5 indicates further that we can expect to fully recover features when $\langle \text{samp} \rangle \lesssim 0.1$. Additionally, when looking at the Ackley function over larger scales (several units), the oscillations are less important to the value of the function than the global bowl-shaped trend. Figure 5 demonstrates this as well: for $\langle \text{samp} \rangle > 5$, we can fully recover the large-scale behavior of the function. In between these regimes, our diagnostic is detecting “features” (the

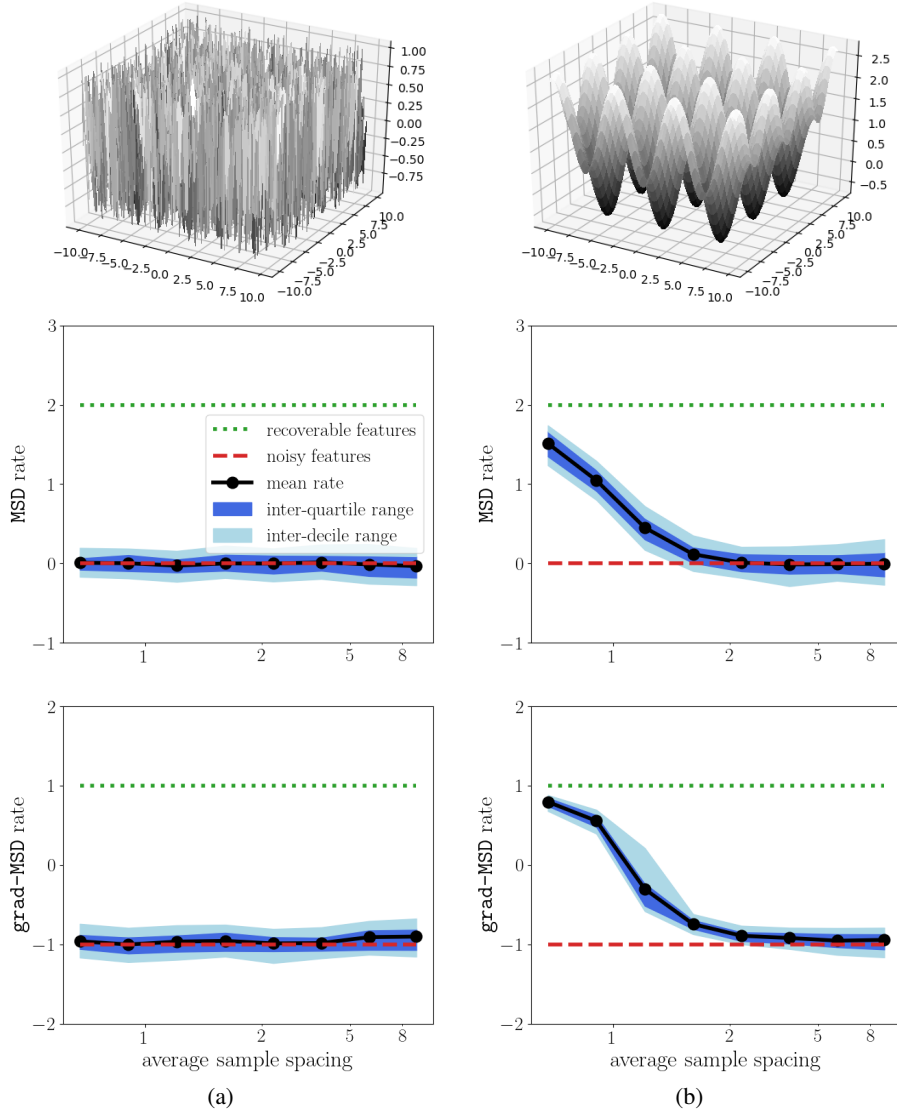


Figure 2: We validate Algorithm 2 and Algorithm 3 by assessing the computed rate over a range of average sample spacings, for f shown in the top row. The mean rate (black dot series) shows the average of the computed rate over 100 trials with different random initial seeds. The inter-quartile (between 25th and 75th percentiles) and inter-decile (between 10th and 90th percentiles) ranges are shown in dark blue and light blue bands, respectively. (a) Pure noise is consistently detected as having the “noisy features” rate (0 for MSD, -1 for grad-MSD). (b) Fine scale features are only recoverable if average sample spacing is small enough.

oscillations) at about $\langle \text{samp} \rangle \sim 0.8$, roughly matching the scale of the oscillations (the period of the cosine terms, i.e. 1 unit).

Experiments in higher dimensions While many Delaunay methods for computational geometry apply exclusively to data in \mathbb{R}^2 or \mathbb{R}^3 , the Delaunay density diagnostic algorithms have no formal restriction on input dimension d . We explore practical considerations of assessing data sets with dimension 4 or higher by a series of experiments with the Griewank functions, defined in (3). While multiple parameters must be selected in order to run the code, each parameter has a clear geometric interpretation of how it affects the computational cost and accuracy of the result, as we will explain. The results of the experiments are shown in Figure 6; the parameters used are given in Table 1 and explained below.

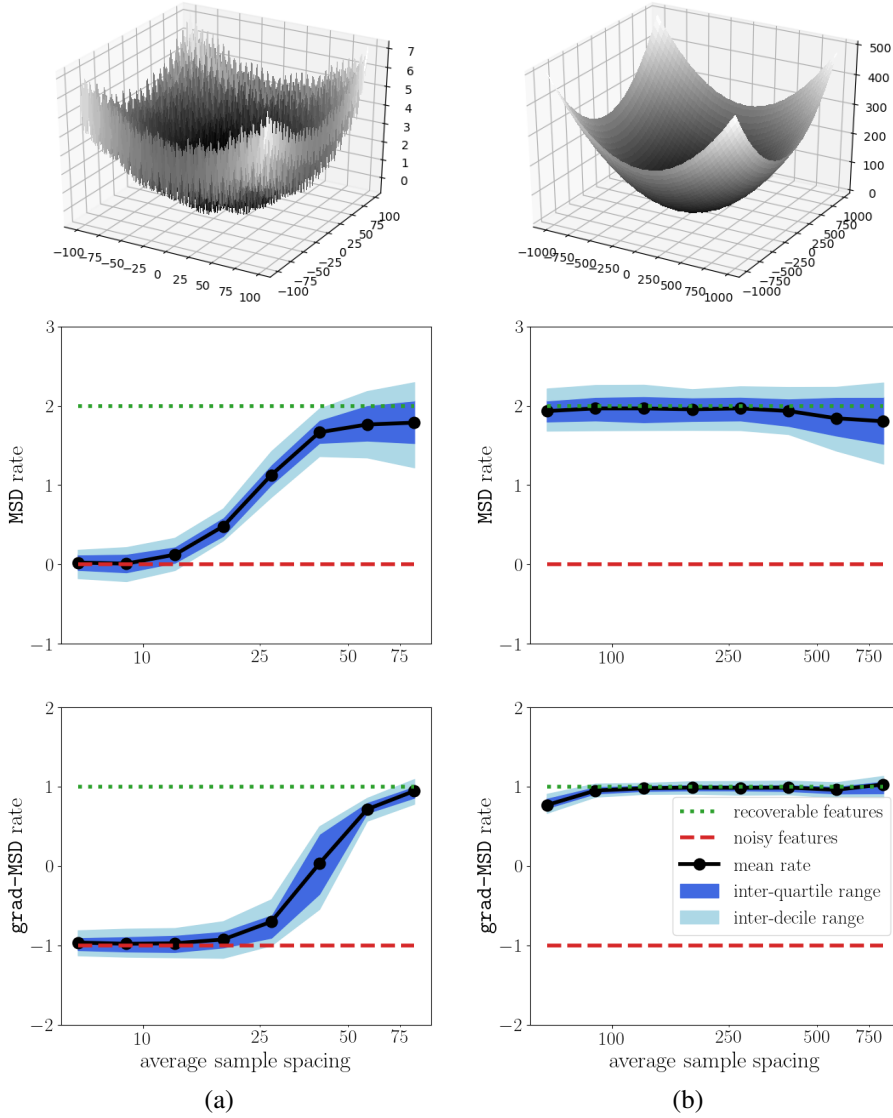


Figure 3: Additional experiments as in Figure 2. (a) Insufficiently sampled fine scale features are detected as noise; the large scale quadratic feature is recoverable with larger average sample spacing. (b) Smooth variation (at the scale of inquiry) is consistently detected as having the “recoverable features” rate (2 for MSD, 1 for grad-MSD).

For $d = 2, 3, 4$, we fix a uniformly spaced d -dimensional lattice of Q^d query points, centered at the origin in \mathbb{R}^d . For $d = 2, 3$ we use $Q = 20$ and for $d = 4$ we use $Q = 10$. Note that Q can be any positive integer ($Q = 1$ corresponds to a single query point), however, the size of Q^d will become a main driver of computational cost as d increases. Letting M denote the side length of the query point lattice, and L the side length of the bounding box to be used (as in Eq. (2)), we define

$$\text{qpdf} := \text{query points dimension fraction} := \frac{M}{L}.$$

For $d = 2, 3$, we use $\text{qpdf} = 0.8$ and for $d = 4$ we use $\text{qpdf} = 0.6$, with the bounding box always centered at the origin. The purpose in decreasing qpdf for larger d is to reduce the probability that a query point lies outside the convex hull of the samples $\{\mathbf{x}_j\}$, a case we exclude from consideration in this work, per the discussion at the end of Section 2.2.

We select b as indicated in Table 1. The effect of b on accuracy was discussed above and in Figure 5. We initialize each trial with $\|\{\mathbf{x}_j\}\| = 3000$ for $d = 3$ and 5000 for $d = 4$ and set a stopping criteria of $\|\{\mathbf{x}_j\}\| = 100,000$ or 200,000. After collecting results, we filter out small n_k values to produce the min and max values indicated in Table 1. From these values, we compute $\langle \text{samp} \rangle$ and then determine the number and extent of scaling needed so that $\langle \text{samp} \rangle$ will have

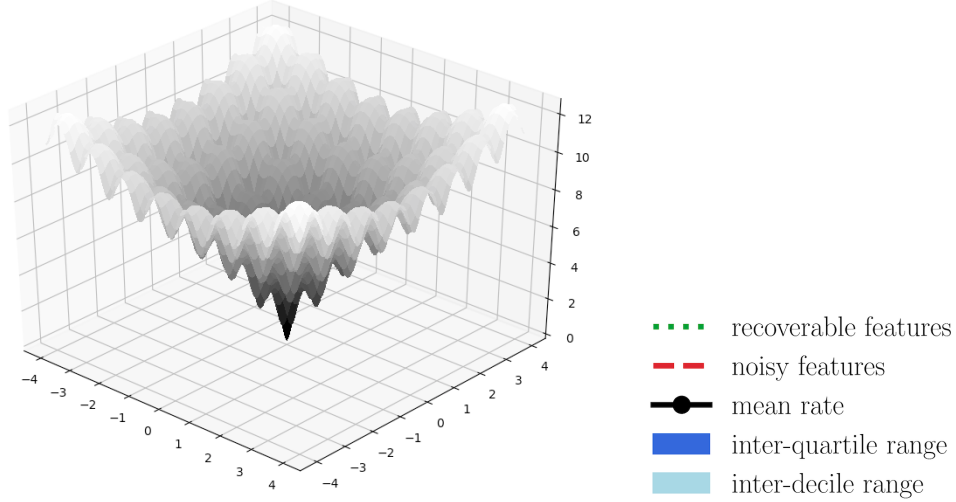


Figure 4: *Left:* A basic visualization of the Ackley function on $[-4, 4]^2$. *Right:* Legend for the plots in Figure 5.

range $\approx [10^{-1}, 10^{2.5}]$ across all experiments. This last point is key. There are not enough computational resources to support exploring the Griewank function at scales ranging from 10^{-1} to $10^{2.5}$ using a single bounding box (too many points are required to reach the smallest scales). So instead, we scale down the bounding box when we reach our limit of points (100k-200k) and then glue these different scales together by matching $\langle \text{samp} \rangle$.

| dim | b | $\ \{q_i\}\ $ | qpdf | min n_k | max n_k | # scales | # trials |
|-----|--------|---------------|------|-----------|-----------|----------|----------|
| 2 | 1.4641 | 400 | 0.8 | 3,947 | 173,832 | 4 | 100 |
| 3 | 1.2 | 8,000 | 0.8 | 8,400 | 69,321 | 10 | 25 |
| 4 | 1.3 | 10,000 | 0.6 | 33,423 | 89,109 | 14 | 25 |

Table 1: Parameters used to generate results shown in Figure 6.

We use high performance computing resources at Lawrence Livermore National Laboratory to compute all of our examples. Each trial was run on a single node with 64 GB memory, consisting of 16 cores. We call the built-in “level 1 parallelism” feature of `DelaunaySparse`, which exploits a speedup strategy on the loop over query points; this approach temporarily stores discovered parts of the Delaunay mesh structure to accelerate subsequent interpolation queries. With these resources and parameter choices, the wall clock time per trial was 45 seconds, 8.5 minutes, and 11.1 minutes in $d = 2, 3$, and 4, respectively. Thus, the cumulative compute time to produce Figure 6 was approximately 5 hours, 35 hours, and 65 hours for $d = 2, 3$, and 4, respectively.

The reported run times indicate the feasibility of the algorithm in higher dimensions, not an actual assessment of its scalability with respect to dimension. We did not attempt to optimize the code for speed. In practice, the dimension of the data is fixed by the application context. A user would only need to adjust the other parameters—i.e., the column labels of Table 1—to ensure the error bounds met their requirements and the compute time fit within their resource capabilities.

Fixed datasets in \mathbb{R}^d for small values of d We now demonstrate how the algorithm can be modified to assess sampling density of static, existing datasets. At a high level, the only major change required to employ Algorithm 2 or Algorithm 3 on a static dataset is a change to the random point generation process. No bounding box is used. Instead, the index for the static dataset is randomly shuffled and the initial collection of sample points $\{x_j\}$ is defined to be the first n_0 points indicated by the shuffled index. Subsequent additions to $\{x_j\}$ are attained by including the next ℓ points according to the shuffled index. This process emulates the random selection of points, avoids drawing duplicates from the dataset, and is limited by the size of the dataset. We summarize this modification in Algorithm 4.

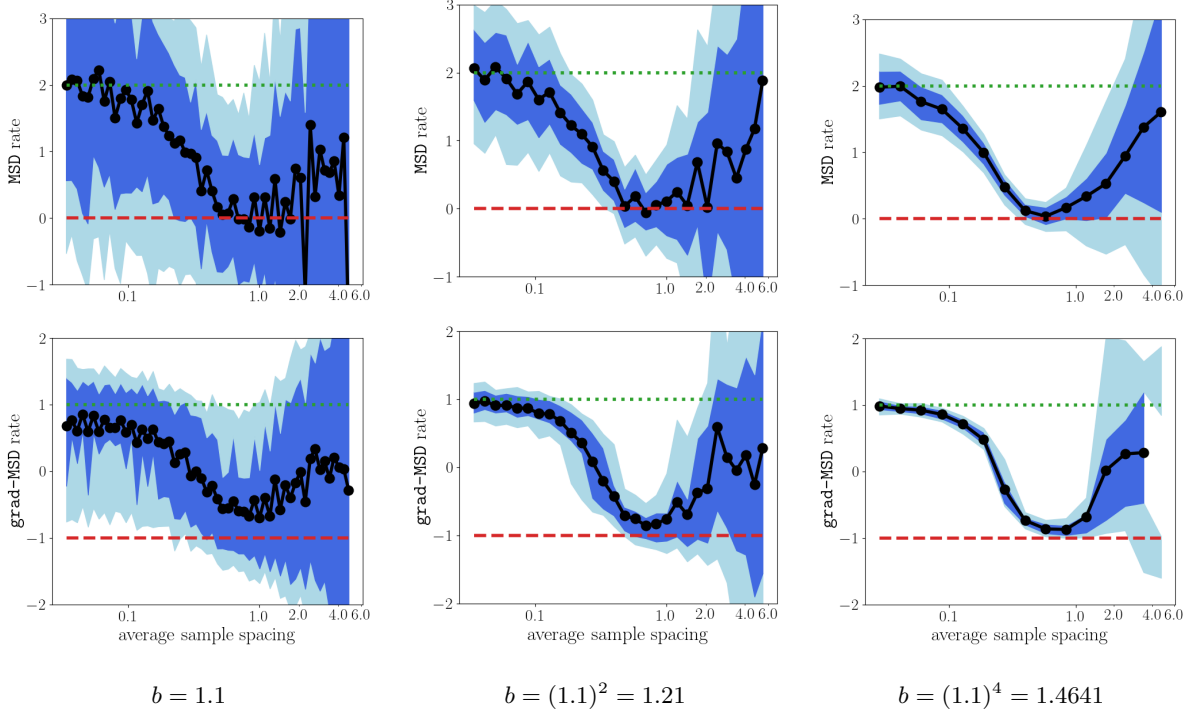


Figure 5: For the Ackley function on \mathbb{R}^2 —see Figure 4—we examine the effect of changing the upsampling rate b on the rates computed by Algorithm 2 and Algorithm 3. Using the values of b indicated, we confirm that a larger b value corresponds to a larger step size (in the horizontal axis) and a smaller variation in the computed rate, as evidenced by the narrower inter-quartile and inter-decile ranges as b increases. For larger average sample spacings, the small number of samples is the cause of the increased variation. In each case, the mean rate has the same trend, reflecting the multi-scale nature of the Ackley function.

Algorithm 4 Delaunay density diagnostic (for static data sets)

```

1: ...
12: Select the next  $\ell$  points from the static data with shuffled index. If  $\ell$  points are not available, break.
14: ...
15: return  $\{n_k, r_k\}$ 
    
```

We have validated Algorithm 4 on a variety of datasets, including datasets from the UCI Machine Learning Repository [Kelly et al.(2023)] as well as fixed datasets of different sizes and dimensions generated from analytical functions. We describe one such validation study in detail, using a dataset from OpenTopography [Cochran(2023)]. The full dataset is twenty million x, y, z coordinates collected via lidar scanning over a roughly 4 km² region of the San Bernadino Mountains in California. A simple point cloud rendering reveals that there are large scale “mountain size” features as well as small scale “tree size” features present in the dataset; see Figure 7, right column. We randomly sample ten thousand points from the full dataset and save this as a fixed dataset for testing Algorithm 4.

The topography dataset is treated as samples from a function $f : (x, y) \mapsto z$. By applying Algorithm 4 to this dataset, we are seeking to determine the scale of features present in the ten thousand point sample of f . From other experiments, we find that setting $\max n_k$ to the size of the dataset is acceptable due to the random nature of subsampling and that setting $\min n_k$ to 1% of the dataset size often yields informative results. Accordingly, we set $\min n_k = 100$ and $\max n_k = 10,000$. We set $b = 1.333521432163324$, derived using a heuristic to produce a handful of computed rates (details on the heuristic appear in the code [Gillette and Kur(2022)]). We construct a grid of 1600 query points over the 10th–90th percentiles of the x and y coordinate data, which corresponds to $\text{qpdf} = 0.8$, since the samples are close to uniformly distributed over a rectangular region. We set $\# \text{ seeds} = 100$. Under these settings, each seed took a few seconds to compute on a MacBook Pro.

In Figure 7, middle, we show the result of the experiment just described. The rates display trends similar to the Griewank function on $[-100, 100]^2$, as studied in Figure 3(a). Notice that both rates trend upward as the sample spacing increases, from “noisy” toward “recoverable”. Accordingly, we may infer that the sample encompasses some non-trivial

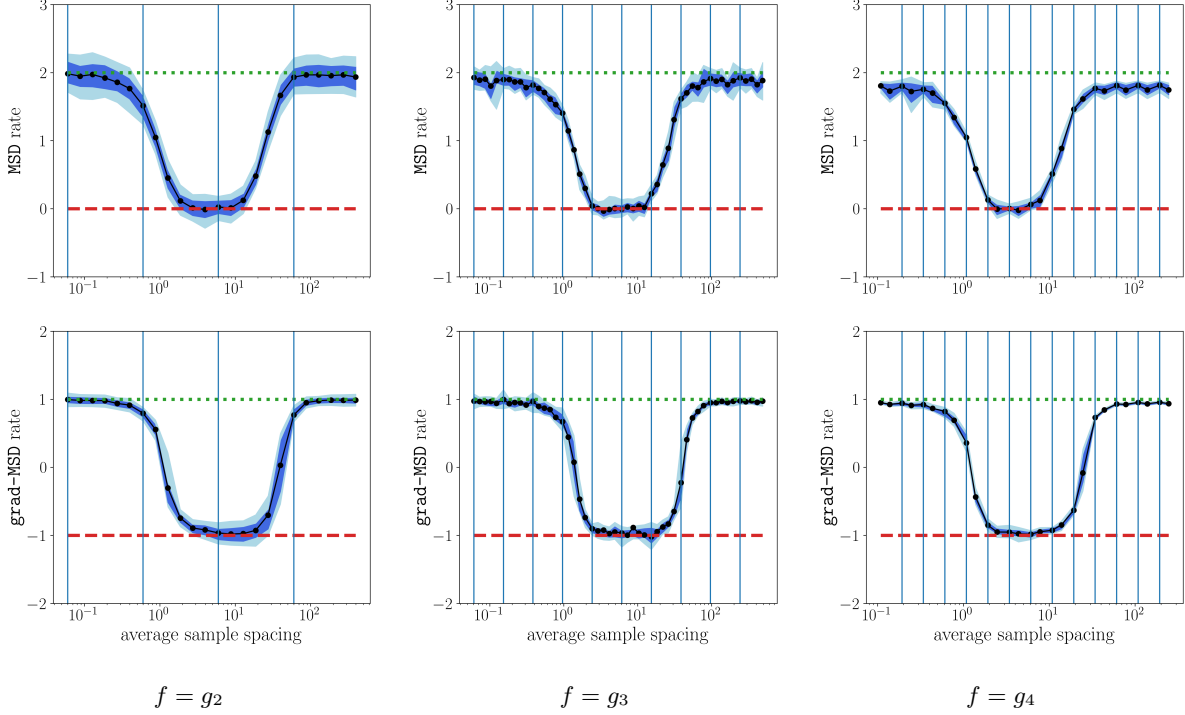


Figure 6: We examine the effect of stepping up the input dimension for the Griewank function $g_d : \mathbb{R}^d \rightarrow \mathbb{R}$, given in Eq. (3). We fix query points $\{q_i\}$ and a bounding box \mathcal{B} , centered at the origin, then run distinct experiments by scaling both by pre-determined amounts. Vertical blue bars separate the distinct scales of experiments, i.e., places where we adjusted the size of the bounding box. As the dimension increases, we use more distinct scales with fewer iterations per scale, demonstrating one approach for scaling with dimension. The computed rates have the same behavior in each dimension, further validating the method.

large-scale features (i.e. the major contours of the mountains) but also has under-resolved small-scale features (i.e. the trees, which register as noise in this dataset). The larger sample spacing results do not coalesce around the recoverable features lines, since those rates are computed starting from $\min n_k$, which is only one hundred points. A takeaway message from these findings is as follows: the ten thousand point sample captures large-scale features to moderate accuracy, but a regression or smoothing technique would be advisable to mitigate the effect of noise from under-resolved small-scale features.

In our final example, we apply Algorithm 4 to two higher-dimensional static datasets in \mathbb{R}^5 . The datasets, denoted BURN_OFF and BURN_ON, consist of input-output pairs from HYDRA, a multi-physics simulation code developed at LLNL over the past twenty years that informs experiments at the National Ignition Facility [Marinak et al.(2001)]. Both datasets consist of pairs $\{(\mathbf{x}_j, f(\mathbf{x}_j))\}$ where $\mathbf{x}_j \in \mathbb{R}^8$ and $f(\mathbf{x}_j) \in \mathbb{R}^1$ is a quantity of interest. Physicists familiar with the problem context have indicated that the response of f is primarily dependent on only five of the eight inputs. Thus, we filter out the three less-important input variables, reducing inputs to $\mathbf{x}_j \in \mathbb{R}^5$. The filtering process creates “near-duplicates” in the dataset, i.e., points $(\mathbf{x}_m, f(\mathbf{x}_m))$ and $(\mathbf{x}_n, f(\mathbf{x}_n))$ where $\mathbf{x}_m \neq \mathbf{x}_n$ but $\|\mathbf{x}_m - \mathbf{x}_n\|_{L^2(\mathbb{R}^5)} < \delta$, with δ small enough that DelaunaySparse marks them as identical. We fix a tolerance level δ , then identify clusters of points within the dataset such that each point in a cluster is within δ of some other point in the cluster. For clusters with more than one point, we keep the mean of the points and values as a “new” data point and discard the points defining the cluster.

By the filtering process, the BURN_OFF dataset is reduced from 17,450 to 13,016 input-output pairs while the BURN_ON dataset is reduced from 17,406 to 12,989 pairs. Each dataset is then rescaled so that the min and max in each coordinate are 0 and 1, respectively, including the output coordinate. Rescaling in this way removes bias due to choice of units.

To construct a query point lattice that lies inside the convex hull of the input points, we compute the 25th and 75th percentiles for each of the five input coordinates. These percentiles define an interval for each dimension, from which we can build a lattice centered around the geometric mean of the inputs. We use five points per dimension, for a total of $5^5 = 3125$ query points. The notion of qpdf does not extend directly to these datasets since the inputs were, by design, not drawn uniformly from a bounding box but instead clustered closer to the mean of the inputs.

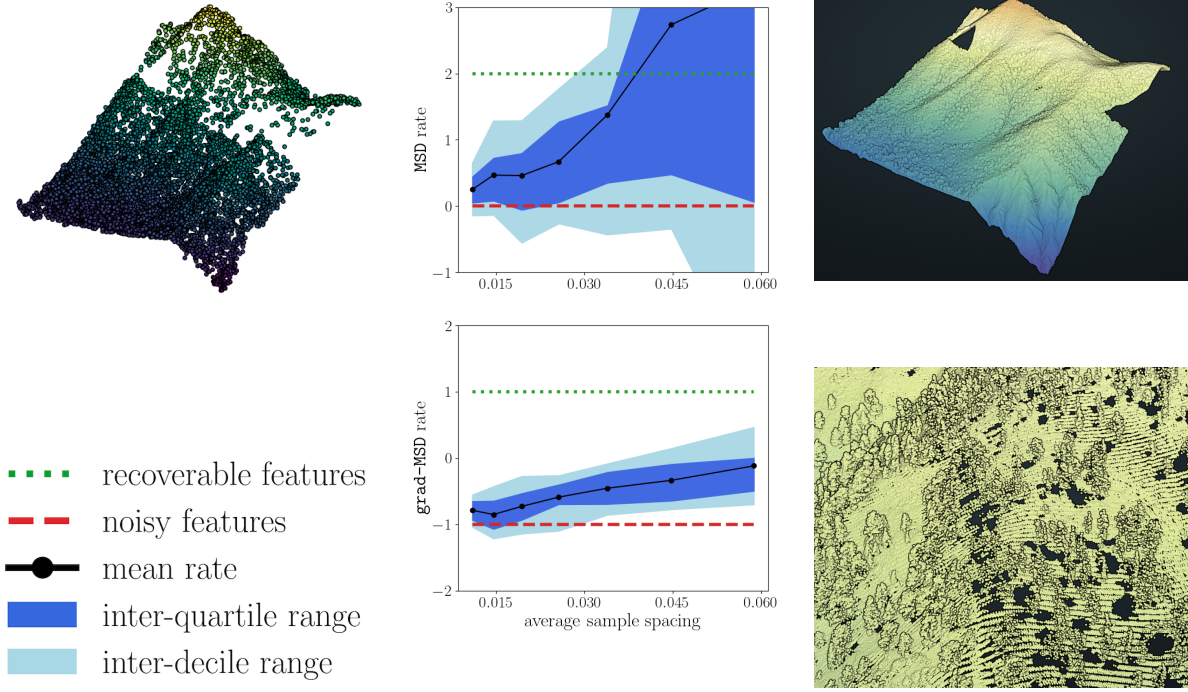


Figure 7: We test Algorithm 4 on a static dataset of 10,000 points, collected from a lidar sampling of a mountain range, retrieved from OpenTopography [Cochran(2023)] (left, top). The computed rates (middle) indicate that small scale features are present in the data set but not fully resolved. These findings are consistent with visualizations of the original twenty million point cloud data from which the sample was drawn, indicating both coarse features at the scale of mountains (right, top) and fine features at the scale of trees (right, bottom).

In Figure 8, we show the results of our experiments for $b = 1.2$, $\min n_k = 200$, $\max n_k = 13,016$ for BURN_OFF, $\max n_k = 12,989$ for BURN_ON, and $\# \text{ seeds} = 50$. With these options, the wall clock time was approximately 2.3 minutes per trial using the same compute resources that produced Figure 6. For both datasets, both rates display trends similar to the Griewank function on $[-10, 10]^2$, as studied in Figure 2(b). Accordingly, the results suggest that the dataset has sufficient density *only* at the smallest possible $\langle \text{samp} \rangle$ value to resolve the features present in the domain of consideration, i.e. the five dimensional volume of parameters space enclosed by the 25th to 75th percentiles of each coordinate. The smallest $\langle \text{samp} \rangle$ value occurs when using *all* of the available data, meaning using anything less than the full dataset risks confusing geometric features for noise.

Finally, we note that both datasets are modeling the same physical process: an inertial confinement fusion implosion. The BURN_OFF dataset artificially turns off the deposition of energy generated by the fusion reactions, while the BURN_ON dataset keeps the deposition in place. The BURN_ON dataset thus experiences “yield amplification” [Lindl et al.(2018), Christopherson et al.(2019)] relative to the BURN_OFF dataset. Such a process is unlikely to generate features at scales smaller than those present in the BURN_OFF dataset, though it can change the structure at larger scales. Comparing the rate computations between the two datasets in 8, we confirm that small scale features are accurately resolved in both, while the BURN_ON dataset exhibits more variation in rates at lower sample densities (i.e., larger scales). The differences are not extreme and the similarity in trends between the two datasets serves as a validation of the correlation between the simulations that generated them.

5 Analytical results

Our numerical results provide evidence for Claim 1, Claim 2, Claim 3, and Claim 4. In this section, we sketch out theoretical support for these claims by employing order-of-magnitude analysis for the quantities appearing in Algorithm 2. We follow that up with a more formal argument. Throughout the section we assume sample points (those supporting the Delaunay triangulation) and query points (where we evaluate the Delaunay interpolant) are uniformly drawn from the bounding box. Our results are technically valid in the limit of a large number of draws, as we neglect errors coming

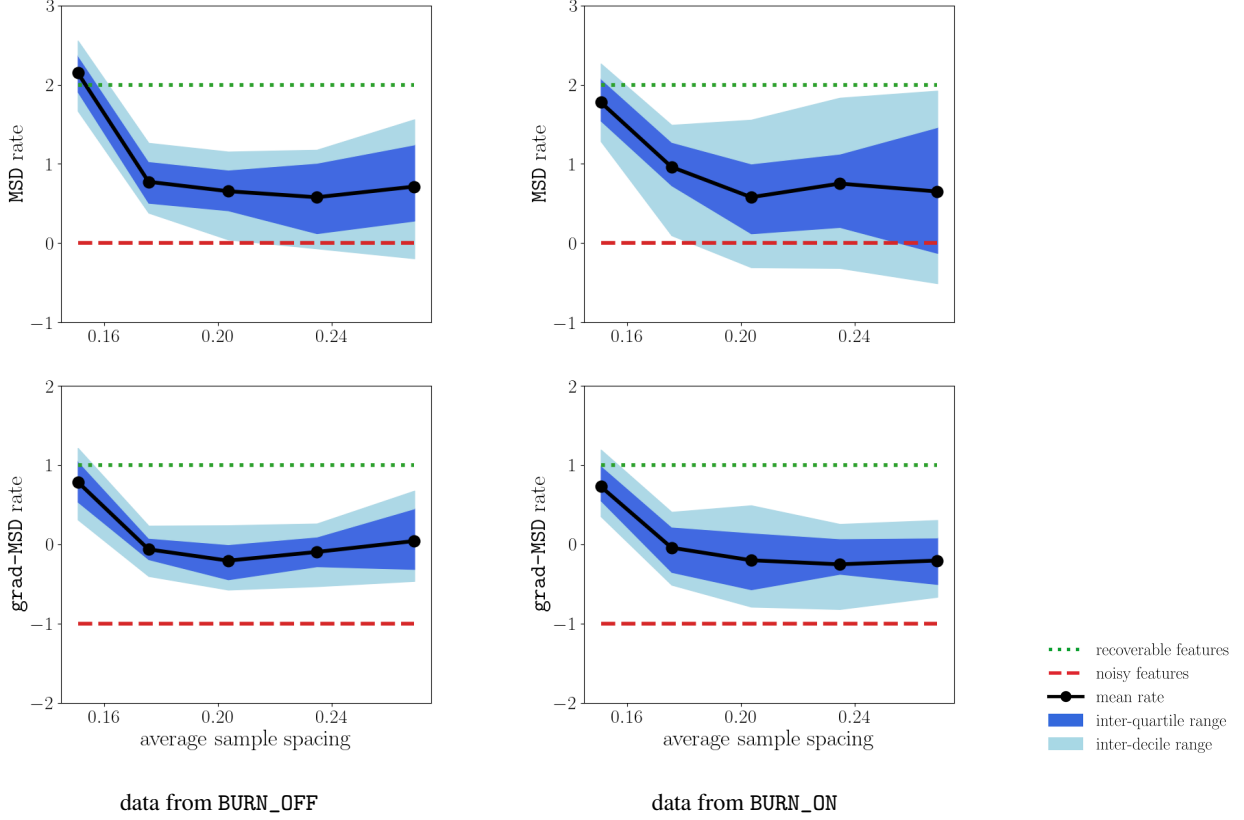


Figure 8: We apply Algorithm 4 to two static datasets of $\approx 13,000$ data points each, gathered from simulations of inertial confinement fusion, produced by the HYDRA codebase. The computed rates suggest that the datasets sufficiently capture small scale geometric features, since the rates are nearly equal to the recoverable features rate at the smallest possible $\langle \text{samp} \rangle$ value. In addition, using less than the full data set to create a surrogate model risks introducing spurious features and potentially akin to modeling pure noise. Decreasing $\langle \text{samp} \rangle$ further would potentially uncover more features, but would require generating more data, demonstrating how the Delaunay density diagnostic can be used to inform the need for data collection.

from finite sampling. In practice, we have found that our results hold as long as deviations from these assumptions are not too significant (e.g. the data does not lie on a lower-dimensional submanifold).

We start with the case of smooth functions, where the sampling resolves all of their features (Claim 1). For such functions, the linear interpolant is a reasonable approximation up to quadratic order. In particular the difference between successive interpolants will behave as $\|\hat{f}_k - \hat{f}_{k-1}\| \sim \mathcal{O}(\langle \text{samp} \rangle_k^2)$, where $\|\cdot\|$ refers to a discrete approximation to any L^p norm (e.g., $\sqrt{\text{MSD}_i(\bullet)}$) and $\langle \text{samp} \rangle_k = L/n_k^{1/d}$ is the average sample spacing (the subscript k reminds us of the iteration for which it is computed). Taking ratios of norms on successive iterations gives us

$$\frac{\|\hat{f}_{k-1} - \hat{f}_{k-2}\|}{\|\hat{f}_k - \hat{f}_{k-1}\|} \sim \frac{\langle \text{samp} \rangle_{k-1}^2}{\langle \text{samp} \rangle_k^2} \sim \left(\frac{n_k}{n_{k-1}} \right)^{2/d} \sim b^2, \quad (4)$$

where the last step uses the approximate relation between the number of points and the upsample rate b (i.e., line 11 from Algorithm 2). Taking $\log_b(\cdot)$ of the expressions in (4) gives $r_k \sim 2$, verifying Claim 1.

With additional notation, we can further formalize the above argument. We write the difference between the linear interpolant \hat{f}_k (at iteration k) and the true function f as

$$\hat{f}_k(\mathbf{x}) - f(\mathbf{x}) = \sum_{n=2}^{\infty} a_n^{(k)}(\mathbf{x}; \sigma_x^{(k)}) \langle \Delta x \rangle_{\sigma_x^{(k)}}^n. \quad (5)$$

Here the point \mathbf{x} is in the d -dimensional simplex $\sigma_x^{(k)}$ of the Delaunay triangulation, the discrepancy coefficients $a_n^{(k)}$ depend implicitly on the simplex containing \mathbf{x} , and the average edge length in $\sigma_x^{(k)}$, $\langle \Delta x \rangle_{\sigma_x^{(k)}}$, has been factored out.

This construction, together with our uniform sampling assumption, guarantees (in a suitable average sense) that the $a_n^{(k)}$ depend only weakly on the refinement iteration k , $a_n^{(k)} \approx a_n^{(k-1)}$.

We now write $\langle \Delta x \rangle_{\sigma_x^{(k)}} = w^{(k)}(x) \langle \text{samp} \rangle_k$, which introduces the weight function $w^{(k)}(x)$ that connects the overall average edge length $\langle \text{samp} \rangle_k$ in the Delaunay mesh to the local average edge length within $\sigma_x^{(k)}$. The weight function will be larger in regions of sparse sampling (large simplices) and smaller in regions of dense sampling (small simplices), averaging out to 1, in a suitable sense. As with the $a_n^{(k)}$, $w^{(k)}(x)$ will depend only weakly on k as a result of our uniform sampling assumption.

We are now ready to construct a discrete L^1 distance between successive linear interpolants using the query points $\{q_j\}$; a similar argument holds for any discrete L^p norm. We have:

$$\begin{aligned} |||\hat{f}_k - \hat{f}_{k-1}||| &= \frac{1}{||\{q_j\}||} \sum_j \|\hat{f}_k(q_j) - \hat{f}_{k-1}(q_j)\| \\ &= \frac{1}{||\{q_j\}||} \sum_j \left\| \sum_{n=2}^{\infty} [a_n^{(k)}(q_j) w^{(k)}(q_j) - a_n^{(k-1)}(q_j) w^{(k-1)}(q_j)] \langle \text{samp} \rangle_k^n \right\|, \end{aligned} \quad (6)$$

where we have used $\langle \text{samp} \rangle_{k-1} \approx b \langle \text{samp} \rangle_k$. Since $a_n^{(k)}$ and $w^{(k)}$ (on average) depend only weakly on k , we expect the only k -dependence of $|||\hat{f}_k - \hat{f}_{k-1}|||$ to come from $\langle \text{samp} \rangle_k$. This can be used to more rigorously define the average sense in which $a_n^{(k)}$ and $w^{(k)}$ are independent of k (though we do not attempt that here). If we further assume the $n = 2$ term dominates the inner sum, we can conclude that

$$\frac{|||\hat{f}_{k-1} - \hat{f}_{k-2}|||}{|||\hat{f}_k - \hat{f}_{k-1}|||} \approx \frac{\langle \text{samp} \rangle_{k-1}^2}{\langle \text{samp} \rangle_k^2} \approx b^2, \quad (7)$$

confirming our initial estimate from Equation (4).

A similar argument can be made using the gradient of the linear interpolant (and its corresponding norm). The discrepancy between linear interpolants would scale linearly rather than quadratically. This causes the ratio of norms to be approximately b , and the log-rate to be 1, as required for Claim 3.

We now turn to the case of noisy or highly-oscillatory functions (Claim 2). With insufficient sampling, the linear interpolant will be a poor approximation to the function and will oscillate from iteration to iteration with the amplitude of the noise, which we write as A_f :

$$\|\hat{f}_k(q_j) - \hat{f}_{k-1}(q_j)\| \approx A_f \implies \frac{|||\hat{f}_{k-1} - \hat{f}_{k-2}|||}{|||\hat{f}_k - \hat{f}_{k-1}|||} \approx 1. \quad (8)$$

Taking the log of the above expression to compute the rate gives us a rate of 0, verifying Claim 2.

A similar argument holds for the rate of the gradient norm. In this case, the gradient estimate gets *worse* with refinement, scaling as $A_f / \langle \text{samp} \rangle_k$. Thus, the ratio of successive norms will go as $1/b$ and \log_b of the ratio will go to -1 , as required for Claim 4.

Analysis of upsampling factor b Finally, we comment on the trends with upsampling b uncovered in Figure 5. Equation (7) deviates from b^2 due to the presence of cubic (and higher order) terms in (6). If we write this deviation as $b^2(1 + \varepsilon)$, then the rate becomes

$$r_k = \log_b[b^2(1 + \varepsilon)] = 2 + \log_b(1 + \varepsilon) \approx 2 + \frac{\varepsilon}{\ln b}, \quad (9)$$

where we have made the approximation that the deviation ε is small.

From (9), we see that the deviation from the recoverable features MSD-rate (i.e., 2) gets worse as $b \rightarrow 1$. This causes stronger fluctuations around the estimated rate for smaller b , as seen in Figure 5. We can approximate how much additional sampling would be needed to meaningfully reduce these fluctuations. Suppose we are upsampling near 1, so that $b = 1 + \delta$. Since the deviation ε is sourced by a cubic correction to the rate, we can estimate it as $\varepsilon \approx (a_3/a_2) \langle \text{samp} \rangle$ (the leading order contribution, b^2 , already involved two powers of $\langle \text{samp} \rangle$, leaving just one power remaining in ε). If we want small fluctuations around 2 for the rate, we would need $\varepsilon / \ln b \ll 2$, which simplifies approximately to $\langle \text{samp} \rangle \ll 2\delta a_3/a_2$. So, reducing δ by a factor of two would require a factor of 2^d greater total sampling (halving the

average sample spacing $\langle \text{samp} \rangle$) to get the same level of fluctuations around the target rate. Thus, our analysis confirms that fluctuations in the computed rates can be controlled by increasing b , increasing n_k , or both.

This analysis shows that ideally we would select a large value of b to reduce fluctuations, but this choice comes with two drawbacks. First, a large b makes it difficult to resolve large scales. The diagnostic requires two iterations to compute the first rate, so if we start with an initial spacing of $\langle \text{samp} \rangle_0 = L/n_0^{(1/d)}$, the first diagnostic point will be at a spacing of $\langle \text{samp} \rangle_2 = \langle \text{samp} \rangle_0/b^2$, which might be significantly smaller than the box side L . Unfortunately, there is little that can be done to alleviate this loss of large scales. Fluctuations only decrease with increasing b or decreasing spacing, both of which lose track of large scales. Secondly, a large b means fewer points on the diagnostic curve. This will give us a less precise estimate of how much data is too much (if the diagnostic shows features can be resolved). This is less of an issue if we primarily care about the ability of the full dataset to resolve features. In that case, the only concern with large b is having a sufficiently large initial sample n_0 to avoid large fluctuations. This depends somewhat on the structure of the function and is best determined empirically.

6 Conclusions, extensions, and code

We have demonstrated in this paper how the convergence rate of iteratively refined, piecewise linear approximations to a function can be used to assess if the function has been sampled densely enough *relative* to its variation. Our computational technique eschews nearly any assumption on f as it detects pure noise and undersampled oscillations as equivalent phenomena. Many extensions of the approach are plausible, including assessing convergence in other norms (as mentioned previously) and consideration of higher dimensional data sets, additional static data sets, discontinuous functions, functions with singularities, time-dependent functions, and so forth.

To aid any interested parties in exploring these and other directions, we have released Python code and two driver scripts that effectively replicate the numerical results shown in Figure 6 for $f = g_2$ and in Figure 7 for the static topography dataset. The parameters were adjusted so that the requisite data and figures can be generated in minutes using a typical laptop with a standard modern python environment. The code is included as part of the published version of this work. A version of the code is also available on Github [Gillette and Kur(2022)].

Acknowledgments

We would like to thank Peer-Timo Bremer, Luc Peterson, Brian Spears, and many other colleagues at Lawrence Livermore National Laboratory for providing feedback, encouragement, and financial support for this research. We would also like to thank Tyler Chang at Argonne National Laboratory for many helpful discussions regarding the DelaunaySparse software. Finally, we would like to thank the anonymous reviewers of the manuscript whose suggestions greatly improved the manuscript and accompanying code. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and the LLNL-LDRD Program under Project tracking No. 21-ERD-028. Release number LLNL-JRNL-846568.

References

- [Ackley(1987)] David Ackley. 1987. *A connectionist machine for genetic hillclimbing*. Vol. 28. Kluwer Academic Publishers, -.
- [Amenta et al.(2000)] Nina Amenta, Sunghee Choi, Tamal K Dey, and Naveen Leekha. 2000. A simple algorithm for homeomorphic surface reconstruction. In *Proceedings of the sixteenth annual symposium on Computational geometry*. 213–222.
- [Bates et al.(2023)] Stephen Bates, Trevor Hastie, and Robert Tibshirani. 2023. Cross-validation: what does it estimate and how well does it do it? *J. Amer. Statist. Assoc.* (2023), 1–12.
- [Belkin et al.(2018)] Mikhail Belkin, Daniel J. Hsu, and Partha Mitra. 2018. Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate. In *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2300–2311.
- [Bengio and Grandvalet(2003)] Yoshua Bengio and Yves Grandvalet. 2003. No unbiased estimator of the variance of k-fold cross-validation. *Advances in Neural Information Processing Systems* 16 (2003).
- [Blum et al.(1999)] Avrim Blum, Adam Kalai, and John Langford. 1999. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the twelfth annual conference on Computational learning theory*. 203–208.

- [Brenner and Scott(1994)] Susanne C. Brenner and L. Ridgway Scott. 1994. Polynomial Approximation Theory in Sobolev Spaces. In *The Mathematical Theory of Finite Element Methods*. Springer New York, New York, NY, 91–122.
- [Buhmann(2000)] Martin D Buhmann. 2000. Radial basis functions. *Acta numerica* 9 (2000), 1–38.
- [Buhmann(2003)] Martin D Buhmann. 2003. *Radial basis functions: Theory and implementations*. Vol. 12. Cambridge University Press, -.
- [Celik and Karatekin(1997)] Ismail Celik and Ozgur Karatekin. 1997. Numerical experiments on application of Richardson extrapolation with nonuniform grids. *ASME Journal on Fluids Engineering* 119 (1997), 584–590.
- [Celik et al.(2005)] I. B. Celik, Z. N. Cehreli, and I. Yavuz. 2005. Index of resolution quality for large eddy simulations. *Journal of Fluids Engineering* (2005), 949–958.
- [Chang et al.(2020)] Tyler H Chang, Layne T Watson, Thomas CH Lux, Ali R Butt, Kirk W Cameron, and Yili Hong. 2020. Algorithm 1012: DELAUNAYSPARSE: Interpolation via a Sparse Subset of the Delaunay Triangulation in Medium to High Dimensions. *ACM Transactions on Mathematical Software (TOMS)* 46, 4 (2020), 1–20.
- [Chang et al.(2018)] Tyler H. Chang, Layne T. Watson, Thomas C. H. Lux, Bo Li, Li Xu, Ali R. Butt, Kirk W. Cameron, and Yili Hong. 2018. A polynomial time algorithm for multivariate interpolation in arbitrary dimension via the Delaunay triangulation. In *Proceedings of the ACMSE 2018 Conference*. ACM, Richmond, Kentucky, 12:1–12:8.
- [Chen and Xu(2004)] Long Chen and Jinchao Xu. 2004. Optimal Delaunay triangulations. *Journal of Computational Mathematics* 18, 2 (2004), 299–308.
- [Christopherson et al.(2019)] A. R. Christopherson, R. Betti, and J. D. Lindl. 2019. Thermonuclear ignition and the onset of propagating burn in inertial fusion implosions. *Physical Review E* 99, 2 (2019), 021201.
- [Cochran(2023)] W. Cochran. Dec 15 2023. San Bernardino Mountains, CA 2016 airborne lidar survey. National Center for Airborne Laser Mapping (NCALM). Distributed by OpenTopography. <https://doi.org/10.5069/G9SX6B51>
- [Gillette and Kur(2022)] Andrew Gillette and Eugene Kur. 2022. Delaunay Density Diagnostic [Github repository]. (March 2022). <https://github.com/LLNL/ddd>.
- [Griewank(1981)] Andreas O Griewank. 1981. Generalized descent for global optimization. *Journal of optimization theory and applications* 34, 1 (1981), 11–39.
- [Guo et al.(2010)] Qinghua Guo, Wenkai Li, Hong Yu, and Otto Alvarez. 2010. Effects of topographic variability and LIDAR sampling density on several DEM interpolation methods. *Photogrammetric Engineering & Remote Sensing* 76, 6 (2010), 701–712.
- [Kelly et al.(2023)] Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. 2023. The UCI Machine Learning Repository. <https://archive.ics.uci.edu/>
- [Khuri and Mukhopadhyay(2010)] André I Khuri and Siuli Mukhopadhyay. 2010. Response surface methodology. *Wiley Interdisciplinary Reviews: Computational Statistics* 2, 2 (2010), 128–149.
- [Lindl et al.(2018)] J. D. Lindl, S. W. Haan, O. L. Landen, A. R. Christopherson, and R. Betti. 2018. Progress toward a self-consistent set of 1D ignition capsule metrics in ICF. *Physics of Plasmas* 25, 12 (2018), 122704.
- [Liu and Yin(2019)] Yehong Liu and Guosheng Yin. 2019. Nonparametric Functional Approximation with Delaunay Triangulation Learner. In *2019 IEEE International Conference on Big Knowledge (ICBK)*. IEEE, Beijing, China, 167–174.
- [Lux et al.(2021)] Thomas C. H. Lux, Layne T. Watson, Tyler H. Chang, Yili Hong, and Kirk Cameron. 2021. Interpolation of sparse high-dimensional data. *Numerical Algorithms* 88, 1 (2021), 281–313.
- [Marinak et al.(2001)] Michael M Marinak, GD Kerbel, NA Gentile, O Jones, D Munro, S Pollaine, TR Dittrich, and SW Haan. 2001. Three-dimensional HYDRA simulations of National Ignition Facility targets. *Physics of Plasmas* 8, 5 (2001), 2275–2280.
- [McCullagh and Nelder(1919)] Peter McCullagh and John A Nelder. 2019. *Generalized linear models*. Routledge, -.
- [Morin et al.(2000)] Pedro Morin, Ricardo H. Nochetto, and Kunibert G. Siebert. 2000. Data Oscillation and Convergence of Adaptive FEM. *SIAM J. Numer. Anal.* 38, 2 (2000), 466–488.
- [Nelder and Wedderburn(1972)] John Ashworth Nelder and Robert WM Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)* 135, 3 (1972), 370–384.
- [Rajan(1994)] VT Rajan. 1994. Optimality of the Delaunay triangulation in \mathbb{R}^d . *Discrete & Computational Geometry* 12, 2 (1994), 189–202.

- [Richardson(1911)] Lewis Fry Richardson. 1911. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character* 210, 459-470 (1911), 307–357.
- [Richardson and Gaunt(1927)] Lewis Fry Richardson and J Arthur Gaunt. 1927. The deferred approach to the limit. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character* 226, 636-646 (1927), 299–361.
- [Sacks et al.(1989)] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. 1989. Design and analysis of computer experiments. *Statistical science* 4, 4 (1989), 409–423.
- [Simpson et al.(2001)] Timothy W Simpson, Timothy M Mauery, John J Korte, and Farrokh Mistree. 2001. Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA journal* 39, 12 (2001), 2233–2241.
- [Van Beers and Kleijnen(2004)] Wim CM Van Beers and Jack PC Kleijnen. 2004. Kriging interpolation in simulation: a survey. In *Proceedings of the 2004 Winter Simulation Conference, 2004.*, Vol. 1. IEEE.

A Computing the gradient of the Delaunay interpolant

We can compute $\nabla \hat{f}_k(\mathbf{q}_i)$ in Algorithm 3 by exploiting some standard techniques of differential geometry and linear algebra. Since $\hat{f}_k : CH(\mathcal{D}) \rightarrow \mathbb{R}^1$, and $CH(\mathcal{D}) \subset \mathbb{R}^d$, we can write

$$x_{d+1} = \hat{f}_k(x_1, \dots, x_d),$$

where $\{x_i\}$ are standard Euclidean coordinates for \mathbb{R}^{d+1} . Thus,

$$\hat{f}_k(x_1, \dots, x_d) - x_{d+1} = 0 \quad (10)$$

defines an implicit d -dimensional surface in \mathbb{R}^{d+1} that is piecewise flat. Define $\hat{n}_k(\mathbf{q}_i)$ by implicit differentiation as

$$\hat{n}_k(\mathbf{q}_i) = \left(\partial_1 \hat{f}_k(\mathbf{q}_i), \dots, \partial_d \hat{f}_k(\mathbf{q}_i), -1 \right) = \left(\nabla \hat{f}_k(\mathbf{q}_i), -1 \right). \quad (11)$$

Observe $\hat{n}_k(\mathbf{q}_i) / \|\hat{n}_k(\mathbf{q}_i)\|$ is a unit normal vector to the piecewise flat implicit surface (10).

Now, near \mathbf{q}_i , the surface (10) is determined by the values of the Delaunay d -simplex \mathcal{S} that contains \mathbf{q}_i . Let $\{s_1, \dots, s_{d+1}\}$ denote the vertices of \mathcal{S} , which are found during the computation of $\hat{f}_k(\mathbf{q}_i)$ by Algorithm 1. Recall that each $s_\ell \in \{\mathbf{x}_j\}$, meaning $f(s_\ell)$ is known by assumption. Thus, $\{(s_\ell, f(s_\ell))\}$ is a collection of $d+1$ points in \mathbb{R}^{d+1} lying on the surface (10) near \mathbf{q}_i .

Set A to be the $(d+1) \times (d+1)$ matrix whose rows are formed by the vectors $(s_\ell, f(s_\ell))$. Subtract the column-wise average of A from each row of A , which has the effect of translating the barycenter of \mathcal{S} to the origin. Let $U\Sigma V^*$ be the SVD of A . Then V^* is an orthonormal set whose first d vectors form a basis for (10), meaning the last vector, call it \mathbf{v}_{d+1} , is a unit normal to (10). By scaling \mathbf{v}_{d+1} so that its last coordinate is -1 , we have found $\hat{n}_k(\mathbf{q}_i)$ and can recover $\nabla \hat{f}_k(\mathbf{q}_i)$ from (11).

We remark briefly on the case where \mathbf{q}_i lies at the interface of one or more Delaunay simplices. The gradient is not continuous across mesh elements (since the function is piecewise flat) and thus $\nabla \hat{f}_k(\mathbf{q}_i)$ has no unique definition. In practice, however, this is very unlikely to occur and such instances could be detected and managed robustly in a number of ways.