Computers & Graphics (2019)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

A Comparison of Methods for Gradient Field Estimation on Simplicial Meshes

Claudio Mancinelli[a], Marco Livesu[b], Enrico Puppo[a]

[a]DIBRIS - University of Genoa (Italy)
[b]IMATI - CNR, Genoa (Italy)

ARTICLE INFO

Article history:
Received May 17, 2019

Keywords: Simplicial meshes, Gradient estimation, Discrete geometry

ABSTRACT

The estimation of the differential properties of a function sampled at the vertices of a discrete domain is at the basis of many applied sciences. In this paper, we focus on the computation of function gradients on triangle and tetrahedral meshes. We study one cell-based method (the standard the facto), plus three vertex-based methods. Comparisons regard accuracy, ability to perform on different domain discretizations, and efficiency. We performed extensive tests and provide an in-depth analysis of our results. Besides some common behaviour, we found that some methods perform better than others, considering both accuracy and efficiency. This directly translates to useful suggestions for the implementation of gradient estimators in research and industrial code.

1. Introduction

Geometric meshes play a central role in a number of domains, including computer graphics and scientific visualization, mechanical, structural and electrical engineering, and computational methods in physics, chemistry and geology. Quite often, a scalar field is sampled at the vertices of a geometric mesh, which discretizes a given domain. Computational analysis of such a field may require evaluating its gradient and, possibly, tracing its integral curves.

In the geometry processing and FEM literature, a scalar field is often extended from vertices to the interior of higher dimensional cells by linear interpolation. Under this approach, a constant gradient is associated to each cell with a straightforward computation, thus providing the simplest form of evaluation of the gradient field. Although sufficient for many applications, the piece-wise constant gradient has several limitations and drawbacks: being not continuous, the gradient has divergent covariant derivative at the interface between cells; singularities are forced to lie just at vertices of the mesh; and integral curves are discretised into polylines that travel parallel to each other inside each element, so that their distribution does not depend just on the field, but also on the orientation of edges and vertex valences of the underlying mesh (see Figure 1).

As discussed in [1], discrete vector fields can be given per face, per edge, or per vertex. Despite the entity to which the gradient estimate is attached, the gradient field can be extended to the whole mesh either in a piece-wise constant manner – i.e., defining a local region surrounding each vertex/edge/face and assuming the gradient to be constant within it, or via interpolation – e.g., linearly inside each simplex. When vector fields are extended to the whole mesh by linear interpolation, they can be traced exactly inside each region [2, 3]. However, there exist surprisingly few works dealing with the estimation and assessment of piece-wise linear gradient fields.

In this work, we review common methods for estimating a per-vertex gradient field linearly interpolated within each element, and we compare their accuracy with respect to the standard method to compute a per-face constant gradient field. Since our interest is to study the performances of the vertex-based methods in every point of the domain, such comparisons will be made considering not only the vertices of the mesh, as shown in Sec.4.3. This choice is further motivated by the expectation of applying these results to the tracing of integral curves. We limit our study to the most common cases of planar and volumetric complexes. Since all methods can be extended to arbi-

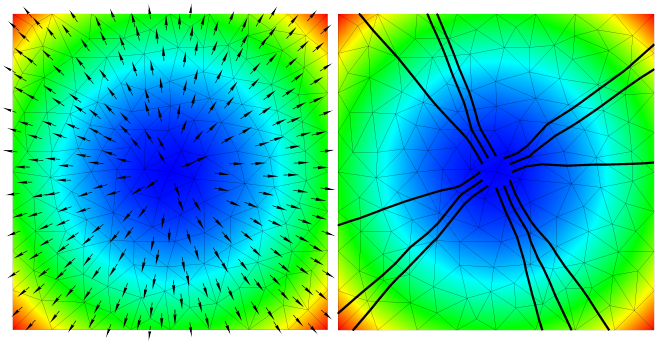


Fig. 1. Per-face gradient estimation on a paraboloid with minimum at the center of the domain (left). Although the estimate is very accurate, integral lines traced from a neighborhood of the minimum do not diverge uniformly, but rather travel in four bundles of nearly parallel lines (right). This behaviour is induced by the discrete piece-wise constant nature of the gradient field. Starting the tracing from the star of a vertex with low valence exacerbates the problem.

trary dimensions, given the coherent results that we obtain in 2D and 3D, we postulate a similar behaviour also on higher dimensional complexes. Performances of the various methods are measured according to a family of parametric functions sampled at mesh vertices. The influence of the underlying tessellation is also studied, considering different domain discretisation strategies. Note that the extension to surface meshes embedded in 3D is out of the scope of this paper, and will be briefly discussed in Sec. 8.2.

The analysis of our results shows that – not surprisingly – piece-wise constant gradients perform worse than gradients linearly interpolated within each cell in almost all our experiments. The only exception being performances on boundaries, and the estimation of the sole magnitude of the gradient, where separate computation per cell offers an advantage to methods that use a bigger stencil. Restricting to vertex-based approaches, we found that all methods performed similarly in many experiments in terms of accuracy, with some exceptions regarding resiliency to anisotropy and signal noise. We broke ties combining accuracy with an analysis of scalability and computational cost associated to each technique, being able to provide useful suggestions for the implementation of per-vertex gradients in research and industrial code.

A preliminary version of this paper was presented at *Smart Tool and Apps for Graphics (STAG)* [4]. While the former work was devoted to 2D triangulated domains, in this version we extend it by a more thorough analysis of the 2D case, as well as by adding an experimental part on volumetric meshes. Experiments on volumes substantially confirm the performances of the gradient estimators that we already observed on triangle meshes.

2. Related Works

Sozer and colleagues [5] surveyed different approaches to gradient estimation for a variety of meshes used in Computational Fluid Dynamics (CFD), ranging from tetrahedral and hexahedral, to general polyhedral meshes. In all the considered examples, the function is encoded at the vertices of the mesh and the average per-cell gradient is computed according to different techniques, always leading to a piece-wise constant gradient field. In a recent course, De Goes and colleagues [1] discuss gradient fields as a special case of the more general vector fields; the course focuses on triangular meshes embedded in $\mathbb{R}^3$, and does not extend to other spaces or discretisations. Hyman and Shashkov [6] introduced a framework for discrete calculus of fields defined on the vertices of a 2D domain meshed with quads, formulating a discrete counterpart of continuous operators such as gradient, divergence and curl. Neumann and colleagues [7] propose an efficient method to estimate surface normals of a density field encoded in a voxel grid, and use this information for shading. The method is based on a gradient estimation obtained with 4D linear regression. Jirka and Skala [8] studied gradient estimation based on fitting quadrics on mesh vertices; the regression strategy we study is based on the method they describe. Correa et al. [9] studied the accuracy of gradient estimation in the context of scientific visualization. Their study is similar in spirit to ours (similar gradient estimation techniques are considered), but it is focused on volumetric meshes only. For the 2D case, Cerbato et al. [10] tested the performances of the Green Gauss gradient estimation on a variety of different polygonal meshes (simplicial meshes were not considered). To the best of our knowledge, no comprehensive study of gradient field estimation on simplicial meshes is present in literature.

3. Methods for gradient field estimation

We introduce here the four methods we consider in our study, also fixing our notation. Let $\Omega \subset \mathbb{R}^d$ be a compact domain. In the following, we focus on the cases $d = 2, 3$; nevertheless concepts extend to any dimension.

Let Σ be a simplicial mesh having Ω as carrier. Mesh Σ can be described by the collection V of its vertices, and the collection T of its maximal cells – i.e., triangles and tetrahedra, for $d = 2$ and $d = 3$, respectively. Cells of intermediate dimensions, such as edges and faces, can be derived uniquely by subsets of vertices belonging to the same maximal cell. We assume the standard topological relations among adjacent and incident cells. In this context, we recall that the *star* (or *1-ring*) of a vertex $v \in V$ is defined as the collection of all cells (edges, triangles, tetrahedra) of Σ that are incident at v. With abuse of notation, we will refer to vertices in the star of v by meaning the vertices sharing an edge with v. The star provides a discrete version of a mathematical *neighbourhood*, and will be generically denoted by $\mathcal{N}(v)$. The *k-ring*, for $k > 1$, can be defined inductively as the union of the stars of all vertices in the $(k-1)$-ring.

Let $f : \Omega \longrightarrow \mathbb{R}$ be a smooth scalar function. We assume to know the value of f only at the vertices of Σ. The discrete version of f is therefore a collection $F = \{f_1, \ldots, f_n\}$, where n is the number of vertices in V and each f_i corresponds to the function value sampled at vertex v_i, for all $i = 1, \ldots, n$.

The problem addressed in the following is that of estimating the gradient ∇f on Ω, on the basis of the discretisations F and Σ. The gradient will be estimated at cells of Σ and extended to the whole domain via an either piece-wise constant (Section 3.1), or vertex-based (Sections 3.2.1, 3.2.2 and 3.2.3) approach.

3.1. Per-Cell linear Estimation (PCE)

Our bottom line is a method that estimates a constant gradient at each maximal cell. We start by extending the values in F to the cells of Σ by linear interpolation. This is well defined on cells of all orders because Σ is a simplicial mesh, hence there exists a unique linear function that interpolates the values in F at all vertices of any cell $\sigma \in \Sigma$, namely

$$\tilde{f}_\sigma(p) = \sum_{v_i \in \sigma} \lambda_i f_i,$$

where p is a generic point of σ, the v_i's are the vertices of cell σ and the λ_i's are the barycentric coordinates of p with respect to the v_i's. In this model, function $\tilde{f}$ estimates f as a piecewise-linear function, which is continuous over Ω and differentiable only in the interior of its maximal cells. The gradient of $\tilde{f}$ is thus constant inside every cell σ and it is associated either to the whole σ, or conventionally to its centroid c_σ, depending on the applications. For a triangle t with vertices v_i, v_j, v_k it is easy to show that we have

$$\nabla f_t = (f_j - f_i)\frac{(v_i - v_k)^\perp}{2A_t} + (f_k - f_i)\frac{(v_j - v_i)^\perp}{2A_t}. \tag{1}$$

Analogously, for a tetrahedron τ with vertices v_i, v_j, v_k, v_h we have

$$\begin{aligned}\nabla f_\tau = &(f_j - f_i)\frac{(v_i - v_k) \times (v_h - v_k)}{2V_\tau} \\ &+ (f_k - f_i)\frac{(v_i - v_h) \times (v_j - v_h)}{2V_\tau} \\ &+ (f_h - f_i)\frac{(v_k - v_i) \times (v_j - v_i)}{2V_\tau},\end{aligned} \tag{2}$$

where $e^\perp$ denotes edge e rotated by 90° w.r.t. the normal of the triangle we are considering, and A_t, V_τ are the area of t and the volume of τ, respectively.

3.2. Per-vertex gradient estimation

As already observed, in the piece-wise linear model of $\tilde{f}$ the gradient is not defined at vertices of Σ. The methods we review in the following assume that f is a higher order function, smooth at edges and vertices of Σ. The different methods exploit different facts that hold in the continuous case, and try to bring them to the discrete setting. All such methods work either by averaging (discrete integration) or by approximation (fitting), because no exact model can be assumed for f in the generic case.

Note that, once the gradient has been estimated at all vertices, the gradient field can be extended by linear interpolation inside cells of any order. It is therefore continuous in Ω and overall more accurate than the piece-wise constant field reviewed in the previous section, as we will see in our experiments.

3.2.1. Average Gradient on Star (AGS)

A common procedure in discrete differential geometry consists of estimating a differential property at a point p as the average value of the same property in a neighbourhood of p [11]. More formally, in our case, we can write

$$\nabla f(p) \simeq \frac{1}{V_{B(p)}} \int_{B(p)} \nabla f dV, \tag{3}$$

where $B(p)$ is a neighbourhood of p and $V_{B(p)}$ is its area/volume.

Now, given a vertex v of Σ, we can use the method in the previous section to estimate (an average value of) ∇f in the maximal cells of the star of v, and compute the integral as a weighted sum of constant terms, obtaining

$$\nabla f_v \simeq \frac{1}{\sum_{\sigma \in \mathcal{N}(v)} w_\sigma} \sum_{\sigma \in \mathcal{N}(v)} w_\sigma \nabla f_\sigma, \tag{4}$$

where ∇f_σ is the value computed with Equation 1 and w_σ is the weight assigned to cell σ incident at v. Correa et al. [12] studied the influence of different weights on the accuracy of results on tetrahedral meshes, They experimented using the *inverse distance of centroid of σ from v,* the *(solid) angle of σ at v*, and the *volume of σ*. Note that weighting with the volume corresponds to applying the Green-Gauss theorem to compute Equation 3 on the star of v (or, equivalently, on a centroidal decomposition of cells), by assuming the linear model inside each incident cell. According to the experiments in [12, 13], however, the inverse distance of centroid and the angle weights perform similarly, and overall better than the volume weight. Furthermore, in our experiments we found the angle weight to be more robust against anisotropic meshes (i.e., meshes containing elongated elements), hence we use such weight throughout. In summary, w_σ in Equation 4 will be the measure of the angle (solid angle in the 3D case) of σ at v (where dependence of w_σ on v has been omitted to keep a lighter notation).

3.2.2. Least Squares fit of Directional Derivatives (LSDD)

This approach consists in estimating first a few directional derivatives of f at v_i, and imposing their relation with the gradient. Let be $\{v_0, ..., v_{k_i}\}$ the vertices belonging to the 1-ring of v_i. Taylor's expansion of f at the first order allows us to write:

$$f(v_j) - f(v_i) \approx \nabla f \cdot (v_i - v_j),$$

for every $j = 0, ..., k_i$. The idea is to build a linear system exploiting the above approximation, i.e writing

$$f(v_j) - f(v_i) = \nabla f \cdot (v_i - v_j), \qquad j = 0, ..., k_i. \tag{5}$$

Note that the second term of (5) is the directional derivative of f along vector $(v_i - v_j)$, hence the name. Since k_i is usually greater than the dimension of the space, the linear system is usually overdetermined and it only admits a least squares solution. As observed in [12], this can be addressed as a weighted least squares problem, where a weight is assigned to each equation, which is inversely proportional to the square of the corresponding edge length. Let A_i be the $k_i \times d$ matrix obtained by collecting all the $(v_j - v_i)$, let W_i the diagonal matrix of weights, and let D_i be the column matrix consisting of all the $f(v_j) - f(v_i)$. Then, the weighted least squares solution is obtained by resolving the $d \times d$ linear system

$$A_i^T W_i A_i \nabla f(v_i) = A_i^T W_i D_i, \tag{6}$$

where $W_i(j, j) = 1/d_{ij}^2$ and d_{ij} is the length of edge $v_i v_j$. Note that such a system must be solved at every vertex. We have also experimented with the unweighted solution (i.e., $W_i = I$), but the weighted one resulted superior in all our experiments, so we use the weighted version throughout.

3.2.3. Linear Regression (LR)

The last approach we review consists of approximating function f in the neighbourhood of v_i with a polynomial π_i of given degree, by setting a system of linear equations that asks π_i to assume the given values of F at all vertices of a given k-ring of v_i. After the fitting polynomial has been obtained, the gradient of f at v_i is estimated analytically as the gradient of π_i.

In our experiments we consider quadratic polynomials and 1-rings, which are extended to 2-rings only if the number of neighbours of v_i is insufficient to fix all degrees of freedom. In 2D we have

$$\pi_i(x, y) = a_i x^2 + b_i y^2 + c_i xy + d_i x + e_i y + f_i,$$

where coefficients $P_i^T = [a_i, b_i, c_i, d_i, e_i, f_i]$ are unknown. For each vertex v_j in the neighbourhood of v_i (including v_i itself), we impose $\pi_i(v_j) = f_j$, thus obtaining a linear system with as many equations as the vertices in the neighbourhood of v_i. Again, we address it as a weighted least squares problem, by assigning a weight to the each equation, and we obtain the coefficients of the best fitting polynomial by solving the system

$$A_i^T W_i A_i P_i = A_i^T W_i F_i, \tag{7}$$

where:

- A_i is a $k_i \times 6$ matrix containing one row per vertex in the neighbourhood of v_i (including v_i itself); the row corresponding to vertex $v_j = (x_j, y_j)$ contains values $(x_j^2, y_j^2, x_j y_j, x_j, y_j, 1)$;
- F_i is a column vector containing the values f_j corresponding to the vertices v_j in the neighbourhood of v_i;
- W_i is a diagonal matrix of weights, each inversely proportional to distance from v_i with a Gaussian decay:

$$W_i(j, j) = \frac{1}{L\sqrt{2\pi}} e^{-\frac{d_{ij}^2}{L^2}}$$

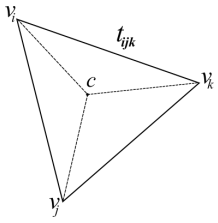
  with d_{ij} being the length of edge $v_i v_j$ ($d_{ii} = 0$) and L being the average edge length in the mesh.

Once the coefficients of π_i are known, the gradient at $v_i = (x_i, y_i)$ is given trivially by

$$\nabla f(v_i) = (2a_i x_i + c_i y_i + d_i, 2b_i y_i + c_i x_i + e_i).$$

For $d = 3$ the solution is analogous: the polynomial has 10 unknown coefficients and the method needs solving a 10×10 linear system. Note that we have to solve such a system at every vertex. As for the previous method, we found the weighted solution of the least squares problem to perform better than the unweighted solution in all our experiments, so we use it throughout.

Remark: Notice that, if we assume function f to be piecewise-linear (linear inside each triangle) and the gradient to be estimated at centroids (or any internal point) of triangles, then all the methods produce the same estimate of PCE, which is exact in the piecewise-linear model. This can be seen easily by considering a triangle t_{ijk}, a point c internal to t_{ijk} (where function f_c at c is computed by linear interpolation), and a conventional split of t_{ijk} into three triangles obtained by connecting c with its three vertices. Then PCE applied to either t_{ijk}, or any of its three sub-triangles returns the same gradient, hence AGS averages three equal values and also returns the same gradient. LSDD applied to the three edges cv_i, cv_j and cv_k also returns the same gradient, since the three edges (lifted by the values of f at the four points) span the same plane. And also LR applied to the same four points returns the same plane, hence the same gradient, once the quadratic terms of the polynomial are forced to zero to be compliant with the piecewise-linear model. So, there is no point comparing the different methods if the piecewise-linear model is assumed. The main argument here is that the piecewise-linear model is in fact too coarse and inaccurate, while per-vertex gradient estimation methods assume a more general model, which not only supports gradient estimation at the vertices of the mesh, but also allows us to extend it to the interior of triangles (e.g., with linear interpolation) providing a better estimation over the whole domain.



4. Experimental setup

We evaluate the four techniques presented in Section 3 on analytic functions, comparing numerical estimates with the ground truth. In order to analyze different situations, we use a parametric family of non-polynomial periodic functions, and meshes with different characteristics.

4.1. Test functions

For the planar case, we consider the domain $\Omega = [0, 1]\times[0, 1]$ and the following parametric family of functions:

$$f_{a,b}(x, y) = a\ \sin(bx)\ \cos(by).$$

Parameters a and b control the amplitude and the frequency of the function, respectively. Figure 2 shows four plots of $f_{a,b}$ for different values of a and b. Likewise, for the 3D setting, we use the parametric family of functions

$$f_{3,a,b}(x, y, z) = a\ \sin(bx)\ \cos(by)\ \sin(bz),$$

defined in the domain $\Omega_3 = [0, 1] \times [0, 1] \times [0, 1]$. We denote $P = 2\pi/b$ the period of function $f_{a,b}$, hence $1/P$ its frequency.

4.2. Meshes

We aim to test the performances of the various gradients on different discrete settings, where the domain is tessellated according to different strategies. To this end, we select three meshes, which we believe are representative of ubiquitous scenarios in applied sciences. Specifically, for the 2D case we consider:

- Σ_S: a *structured* mesh made of equilateral triangles;
- Σ_U: an *unstructured* mesh obtained computing a Constrained Delaunay Tessellation of a Poisson sampling of the domain Ω;

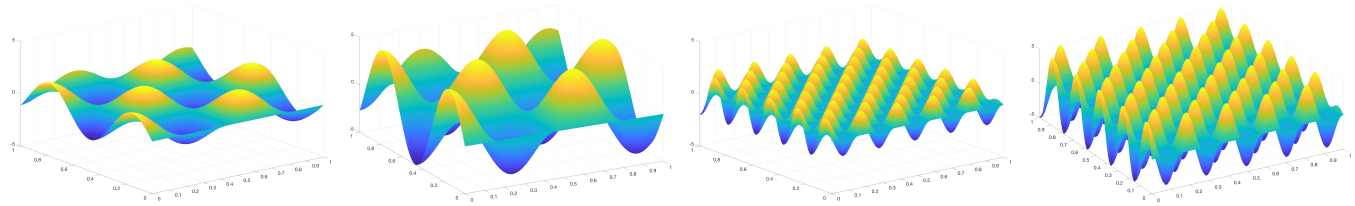


Fig. 2. Four examples of our test function $f_{a,b}$. Parameter a controls the amplitude, whereas parameter b controls the frequency. From left to right: $f_{2,10}$, $f_{5,10}$, $f_{2,30}$, $f_{5,30}$.

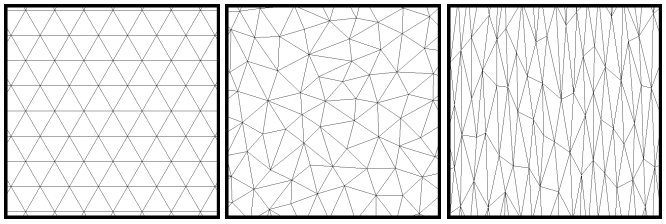


Fig. 3. Close up of our test meshes for the planar case. From left to right: structured, unstructured and anisotropic ($K = 5$).

- Σ_A: an *anisotropic* mesh obtained by computing a CDT of a sampling of the $[0, K] \times [0, 1]$ domain, and squeezing it to fit the unit square. Value K is called the coefficient of anisotropy.

Mesh Σ_S was built procedurally, while for Σ_U and Σ_A we used Triangle [14] for mesh generation. In order to get a fairly uniform distribution of points, we impose the min angle of a triangle in Σ_U to be 20° and the maximum area of a triangle to be $1/(1.6n^2)$ where n is the number of sampled points (consider that the mesh contains about n^2 triangles with an average area of about $1/(2n^2)$ each). Mesh Σ_A is built with similar figures, prior to squeezing it. Each mesh contains approximately 1K triangles. Closeups of our meshes are shown in Figure 3. Analogously, for the 3D case the regular mesh was constructed splitting each element of a regular lattice into five tetrahedra using a standard scheme [15], and Σ_U, Σ_A were created as in 2D, substituting Triangle with Tetgen [16] for the computation of the CDT and using similar figures to bound maximal volumes and minimum solid angles.

For a given mesh, we define L to be its average edge length, hence $1/L$ to be the *sampling frequency*. In our experiments, we do not investigate progressively finer meshes; we rather decided to keep the tessellation fixed and to act on the frequency of the test function (i.e., parameter b), in order to study the interaction between domain discretisation and signal frequency. Note that the two approaches are equivalent. In order to be scale independent, most our graphs use the ratio L/P (signal frequency vs the sampling frequency) on the abscissa. It is worth pointing out that meshes have been constructed with the same approximate average edge length in both the 2D an the 3D case, in order to be able to consider consistent frequency ranges in all our experiments.

4.3. Error metrics

Depending on applications, either the vector value of the gradient, or just its direction, or just its magnitude may be relevant. For this reason, we evaluate the behaviour of the various methods by adopting three separate error measures: for angles we measure the Root Mean Square Error (RMSE); while for vector and magnitude, we adopt the Symmetric Mean Absolute Percentage Error (SMAPE) [17], in order to be as scale independent as possible.

Considering the gradient of the analytic function ∇f, and its numerical estimation $\tilde{\nabla} F$ (computed on the discrete sampling F of f at the vertices of the mesh), our error metrics are defined as:

$$RMSE_{ang} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \measuredangle(\nabla f(v_i), \tilde{\nabla} F(v_i))^2}\,.$$

$$SMAPE_{tot} = \frac{1}{N}\sum_{i=1}^{N} \frac{\|\nabla f(v_i) - \tilde{\nabla} F(v_i)\|_2}{\|\nabla f(v_i)\|_2 + \|\tilde{\nabla} F(v_i)\|_2}$$

$$SMAPE_{mag} = \frac{1}{N}\sum_{i=1}^{N} \left| \frac{\|\nabla f(v_i)\|_2 - \|\tilde{\nabla} F(v_i)\|_2}{\|\nabla f(v_i)\|_2 + \|\tilde{\nabla} F(v_i)\|_2} \right|$$

In order to measure error on the whole domain, we sample data on a regular grid of 500×500 points in 2D, and $100 \times 100 \times 100$ points in 3D. We also add some measures just limited to either the vertices (for the piecewise-linear methods), or the centroids (for the piecewise-constant method) of triangles. In general, we plot angle and magnitude errors only in cases where they show significantly different behaviours (e.g., Figure 9). If not stated differently, we always assume the total error is considered, and the other two errors exhibit a similar behaviour.

5. Evaluation

We report here our experiments and analysis. We consider the four different gradient estimators described before:

- **PCE:** Per-Cell Estimator (Section 3.1);
- **AGS:** Average Gradient on Star (Section 3.2.1);
- **LSDD:** Least Squares Directional Derivatives (Section 3.2.2);
- **LR:** Linear Regression (Section 3.2.3).

Gradients evaluated with PCE are considered constant inside each cell, while vertex-based gradients (i.e., AGS, LSDD and LR) are linearly interpolated inside each cell using barycentric coordinates. We test the performances of the different methods under different conditions, namely: by varying the frequency of

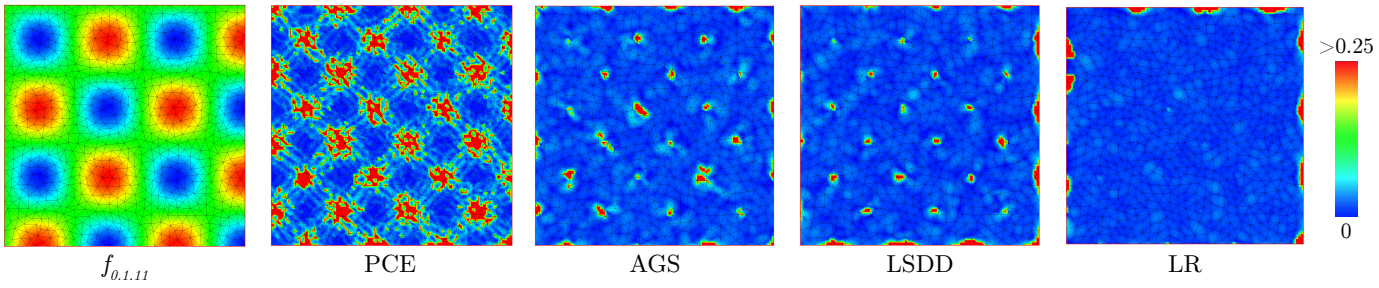


Fig. 4. Heatmaps of the total error measured on the 2D unstructured mesh with test function $f_{0.1,11}$ ($L/P \simeq 0.07$).

our parametric test function; by considering the three different domain discretisations; and by using data affected with noise.

All methods have been implemented as functions of the *CinoLib* [18] geometry processing library, which relies on the Eigen [19] library for numerical computations. Analysis was performed through interactive programs to support data visualization and batch programs to support numerical evaluations. All our implementations run as single threaded C++ applications. The programs are made available in the public domain at `https://github.com/mlivesu/gradient_benchmark`. Experiments were run on a MacBook Pro equipped with an Intel i5 with 2.7 GHz and 8 GB of RAM.

Evaluations are reported separately for the 2D and the 3D case. For the sake of brevity, some discussions are made just in the 2D case, for all situations in which we found a consistent behaviour in the 3D case, too.

5.1. Overview

In order to get a first assessment of the performances of the various methods, we show qualitative results with the help of heatmaps, with a color transfer function mapping low-to-high values to a blue-to-red scale (through green for mid-range values). Here, we show only heatmaps for the 2D case; our benchmark programs support heatmaps also for the 3D case; 3D results are qualitatively consistent with the 2D case.

Heatmaps are used to show both the scalar function (leftmost image in Figure 4), and the total error estimates for the different methods on the same scale (other images in the same figure). It is immediately apparent that the distribution of error is not uniform for all methods:

- PCE, AGS and LSDD exhibit the largest error near critical values of the signal; while LR appears to be less sensitive to the variation of the signal in the interior of the domain;
- AGS, LSDD and LR exhibit a much larger error at the boundary of the domain, which also appears to be amplified near critical points of the signal; conversely, PCE exhibits a rather uniform behaviour both in the interior and at the boundary;
- PCE seems to perform worse than the other methods (at least at this signal frequency) and to have a wider distribution of error.

On the basis of such observations, in the following we perform quantitative analyses by studying separately the behaviour of the different methods at the boundary and in the interior of the domain. For the boundary, we just investigate the major sources of error for the different methods. For the interior, we study sensitivity to the ratio between average edge length and period of the function (L/P), to anisotropy, and to noise, which is evaluated by sweeping the corresponding parameters within ranges that may be relevant in practical applications, and reporting the overall error according to the metrics defined in Section 4.3. Since the different methods exhibit different error distributions, we also report box plots for various situations, by considering a small number of values for each parameter (namely, the extreme values and an intermediate value for each range in the sweeps).

5.2. Boundaries

PCE computes the gradient at any point inside a cell σ by using only data at its vertices. In this respect, a boundary cell is not different from a cell internal to the domain, so it is not surprise that this method exhibits the same performance in the interior and at the boundary.

The situation is rather different for the vertex-based methods.

Remember that all such methods use a stencil (i.e., the 1-*ring*) to sample the function in a local neighborhood and estimate the gradient. More specifically, AGS and LSDD are both based on discrete integration (averaging) and use the stencil as integration domain; while LR is an approximation method, using the vertices in the stencil as data to fit. In both cases, the underlying principle assumes that the stencil provides a discrete version of the mathematical neighbourhood of point v where the gradient is estimated. But this assumption is violated at boundary vertices, where the stencil only captures a "half-neighborhood" (i.e., a set with the topology of a half-space containing v on its boundary).

Specific instances of the boundary effects for the three vertex-based methods are depicted in Figures 5, 6 and 7 and discussed below. For all three methods, it is apparent that the largest error occurs in the proximity of critical points of the signal, where information on the "half-neighborhood" provided by the stencil is not sufficient to guess the behaviour of the signal on the other (unknown) half. Away from critical points, the signal would extend outside the boundary in a more monotone way, which is easier to guess just on the basis of the "half-neighborhood".

More specifically, both AGS and LSDD exhibit very similar behaviour and fail at all critical points on the boundary, just because there are no triangles/edges outside the boundary to

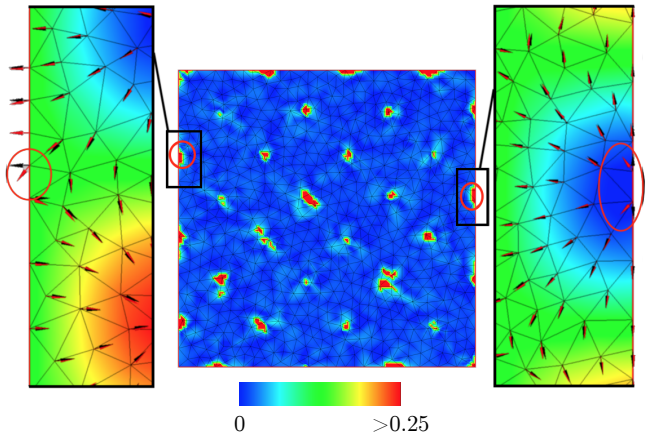


Fig. 5. Heatmap of the total relative error of AGS (center) and the gradient directions estimated at a saddle point (left, red arrow) and at a minimum (right, red arrow). The gradient field depicted by black arrows is the ground truth from test function $f_{0.1,11}$).

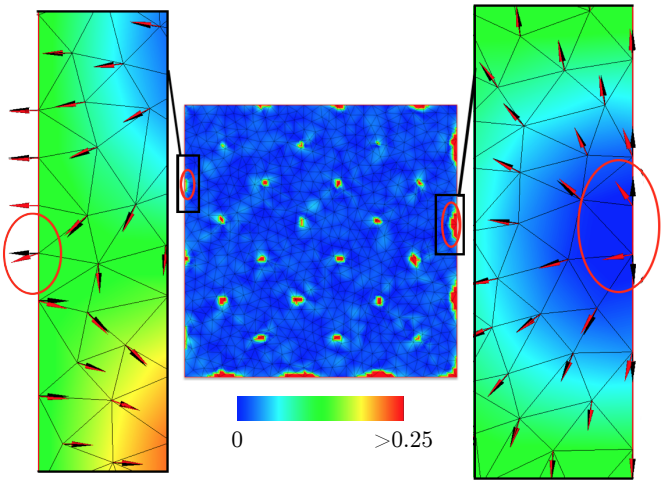


Fig. 6. Heatmap of the total relative error of LSDD (center) and the gradient directions estimated at a saddle point (left, red arrow) and at a minimum (right, red arrow). The gradient field depicted by black arrows is the ground truth from test function $f_{0.1,11}$.

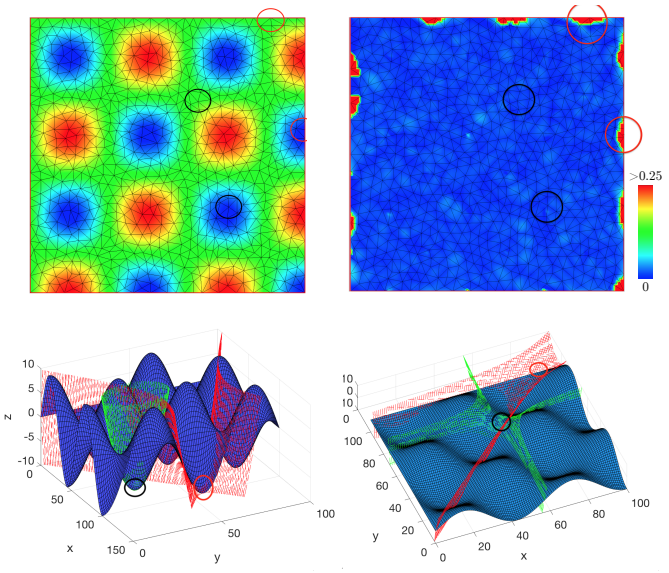


Fig. 7. Top left: heatmap of the scalar field $f_{0.1,11}$ with two boundary vertices (red circles) and two internal vertices (black circles) near critical points; the internal points are placed in positions rather equivalent to the boundary vertices with respect to the period of the signal, respectively. Top right: the heatmap of the total error made by LR shows how the error inside red circles is much higher than inside black circles. Bottom: 3D plots of the signal with superimposed the functions fitted by LR at the corresponding points in the circles: red plots correspond to red circles, green plots correspond to black circles (minimum at the left, saddle at the right).

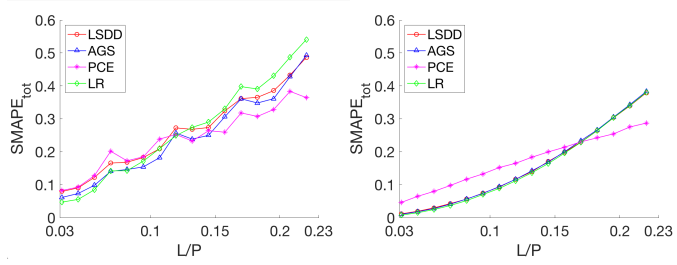


Fig. 8. Error plots at growing signal frequencies, for boundary (left), and interior (right) points. See Section 4.2 for the definition of L/P.

"pull" the gradient to a different direction. For LR the situation is rather different. Note that estimating a gradient at the boundary with LR is equivalent to using LR to extrapolate the signal beyond the boundary. It is well known that extrapolation methods are very unstable, unless a prior on the data can be assumed. The assumption underlying LR is that the signal is a quadric, which is not true in general, and in particular for our test function. In fact, LR does not fail in a systematic way at all critical points, but when it does, it may return very poor results, as the examples illustrated in Figure 7: in one case, a minimum, which is fit rather well inside the domain, is extrapolated into a saddle on the boundary; and a saddle, which is also fit well inside the domain, is extrapolated into an unbalanced and misaligned saddle at the boundary.

In Figure 8, we plot the $SMAPE_{tot}$ of the four methods evaluated on boundaries (left) and interior vertices (right),while varying the ratio L/P. Note that on the boundary all methods show a similar increase, with LR performing best at low frequencies and PCE performing best at high frequencies, while AGS and LSDD perform very similarly and overall stay between the other two methods. Oscillations in such graphs are due to the presence of more or less critical points on the boundary, which in turn depends on the signal frequency. Conversely, in the interior PCE shows a trend similar to the boundary case (except for the bumps, which are absorbed by the distribution of critical points), which is roughly linear with L/P; while vertex-based methods show a non-linear trend, very similar for all of them: they perform better than PCE at low frequencies and worse than it at high frequencies. The better performance of PCE at high frequencies is given to its higher locality (smaller stencil).

In order to avoid the bias introduced by boundaries, in the remainder of the paper we always discard errors measured at sample points lying inside cells of the outer layer, i.e., which have at least one vertex on the boundary of the domain. In a

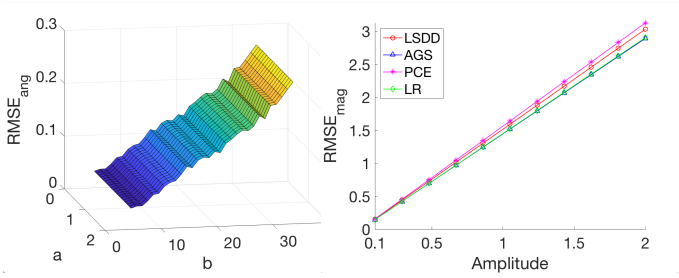


Fig. 9. Left: plot of angle error, obtained varying both the function amplitude a and the frequency parameter b. Right: plot of the *RMSE* of the magnitude for fixed frequency and varying amplitude. For growing values of a, angle error remains constant, while magnitude error grows linearly.

sense, we act as if around the domain we had a padding layer of size compatible with the local stencil used by numerical algorithms.

5.3. The 2D case

Here we report performances of the four gradient estimators on different instances of our test function $f_{a,b}$ for the case $d = 2$. As stated in Section 4.1, parameters a and b set the amplitude and frequency of the function, respectively.

Amplitude. We tested different values of parameter a, ranging from 0.1 to 2. As it can be noticed from Figure 2, changes in amplitude affect the magnitude of the gradient (the bells become steeper), leaving unchanged its direction (maxima and minima do not change). This is confirmed by our experiments, where we see that angle error is constant while varying a (Figure 9, left), whereas magnitude error grows linearly for increasing frequencies (Figure 9, right).

Frequency. We investigate the response of gradient estimators versus growing frequencies of the signal, varying the ratio L/P in the range $[0.03, 0.22]$, hence in a range where we have at least four samples per signal period in all directions. Pushing the signal frequency closer to the Nyquist frequency would not make much sense, as it introduces reconstruction artefacts that are independent of the specific estimator.

Differently from the amplitude, in this case both the angular and the magnitude components of the gradient are affected. For this experiment, we considered structured and unstructured meshes. Overall, we noticed a coherent behaviour for all piecewise-linear methods, and a rather different behaviour for the PCE method. We analyse the differences with the help of the graphs in Figure 10 and of the heatmaps in Figure 4 . We observe that PCE is the least accurate in terms of angle error, while being the most accurate in terms of magnitude error; in terms of total error, the vertex-based methods perform better at low frequencies, while PCE prevails at high frequencies.

Large angle error for PCE is not surprising, as this method cannot catch the non-linear variation of the function within each face, due to its piecewise-constant nature (Figure 4). On the other hand, the smaller stencil of PCE implies a higher locality thus favouring a better estimate of the magnitude of the gradient. Among piecewise-linear approaches, the only noticeable

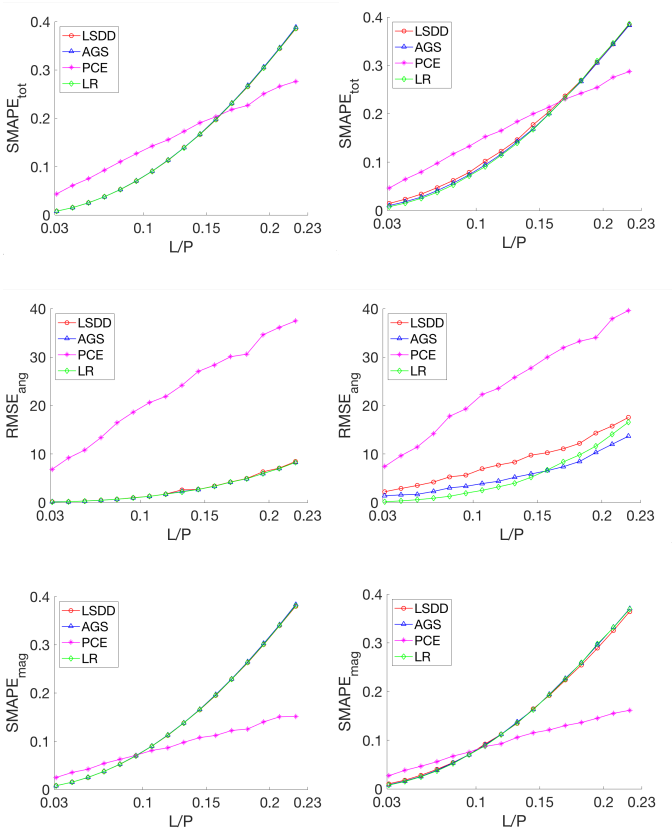


Fig. 10. Plot of total error (top row), angle error (middle row), and magnitude error (bottom row) on the 2D structured (left column) and 2D unstructured mesh (right column).

differences are found in estimating the angle on the unstructured mesh. In this case, Linear Regression (LR) performs better at low frequencies, but above a certain frequency its error grows faster and it starts to perform worse than AGS. Overall, AGS exhibits the best performance in terms of resilience to increase in signal frequency. LSDD exhibits a behaviuor similar to LR, but with a consistently larger error.

Magnitude error offers an opposite landscape, with PCE consistently prevailing on the vertex-based methods. However, looking at data, we noticed that vertex-based methods tend to smoothly distribute the error on the whole domain, without relevant positive or negative peaks. Conversely, the magnitude error of PCE tends to be overall lower, while showing spikes near critical points, where also its angle error becomes much larger. Therefore, even in terms of magnitude and total error, PCE performs better on average, but with a less uniform behaviour.See also Section 5.5 on error distribution.

Notice that both our regular and unstructured meshes come from a uniform sampling; consequently, the stars of internal vertices provide neighbourhoods of nearly constant radius. This is in fact a desirable property, which is pursued by all meshing tools used in practical applications. We made further experiments to test the dependency of accuracy from the valence of vertices, but we could not find any evident correlation (besides, as pointed out in Section 4.2, even in the unstructured meshes, most vertices have a valence equal to the average).

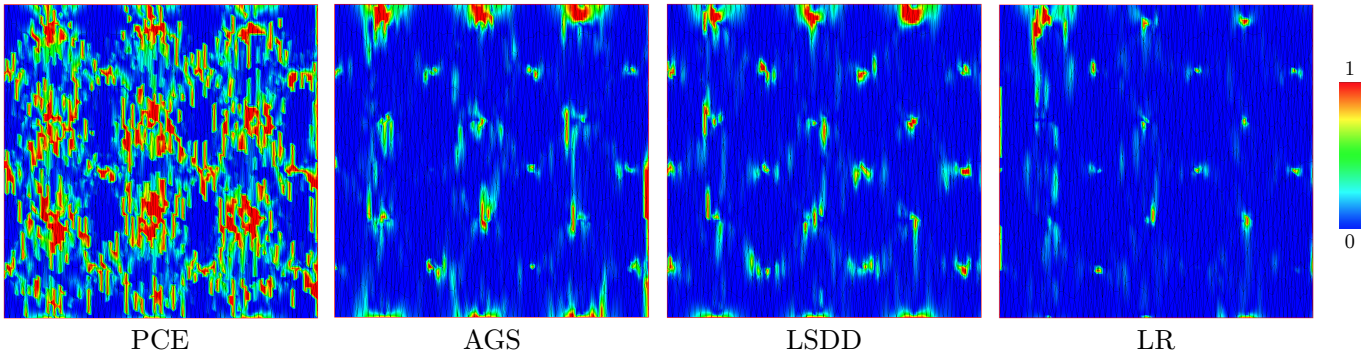


Fig. 11. Plot of the $RMSE_{ang}$ of the four methods, computed on an unstructured mesh with anisotropy factor $K = 9$.

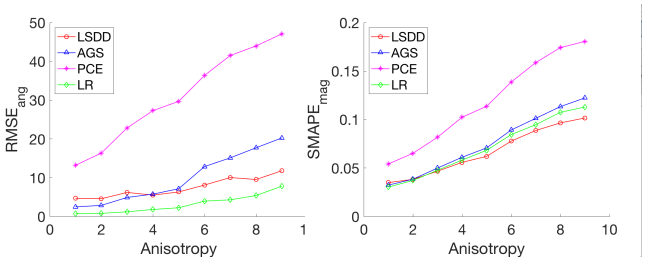


Fig. 12. Plot of angle error (left) and magnitude error (right) of the four methods computed on unstructured meshes with increasing levels of anisotropy. We set the input signal with frequency parameter $b = 10$.

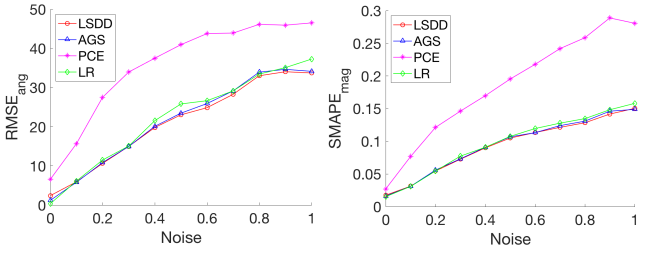


Fig. 13. Plot of angle error (left) and magnitude error (right) of the four methods on the unstructured mesh, where the input signal has been perturbed at vertices to simulate an increasingly high amount of noise. We set the input signal with frequency parameter $b = 10$.

Anisotropy. We test here the performances of the four gradient estimators with respect to meshes exposing increasing levels of anisotropy (Figure 3, right). We report both angle and magnitude errors in Figure 12. As for the frequency tests, there is a neat separation between piecewise-constant and vertex-based methods, which exhibit a consistently superior performance throughout. Among vertex-based methods, linear regression (LR) performs remarkably better than the others just on estimating the direction (Figure 11); while AGS turns out to be generally worse than the other two vertex-based methods.

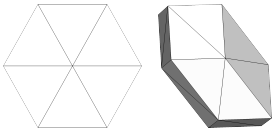
Noise. In many experiments involving real data (e.g. acquired with sensors), the signal measured at vertices is affected by noise. We test here the resiliency of all methods with respect to perturbations of the test function at the mesh vertices. Perturbations are computed as random deviations from the ground truth function. In details, we perturbed the data with additive noise with uniform distribution, random sampled in the range $(-a\varepsilon, a\varepsilon)$, where a is the amplitude of the signal (from Section 4.1) and ε is a parameter to tune the entity of perturbation. Abscissas in Figure 13 are labeled according the values chosen for ε during the experiments. In Figure 13, we report both angle and magnitude errors for growing values of noise. Again, we observe a clear separation between PCE and the vertex-based methods, with PCE exhibiting a much faster increase of error. The vertex-based methods exhibit similar behavior, resulting more sensitive to noise in terms of angle error and more resilient in terms of magnitude error.

5.4. The 3D case

Results in the volumetric case substantially confirm the behaviour of the planar case. Amplitude variations of the test function do not lead to relevant considerations; we therefore study the performances of the four methods with respect to variations in frequency, mesh anisotropy, and resiliency to noise.

Frequency. The tetrahedral meshes considered in our experiments have been constructed in order to have approximately the same average edge length of the meshes used in the planar case. We consider the function $f_{3,a,b}(x, y, z)$ introduced in Section 4 and we vary the ratio L/P in the range $[0.035, 0.27]$.

Overall the 2D (Figure 10) and 3D (Figure 14) cases exhibit a quite coherent behaviour. The same considerations made Section 5.3 about comparative performances remain true also in this case: PCE performs a poor estimation of direction, but it is more resilient to the increase of frequency in estimating the magnitude: on the other hand, vertex-based methods are more accurate in terms of angle error. The main difference is that, in the planar case, the estimation of the gradient direction provided by the vertex-based methods is much more accurate on the structured mesh than on the unstructured one. Indeed, the 1-rings in the volumetric mesh are not as isotropic as in the planar mesh (see inset – 2D ring left, 3D ring



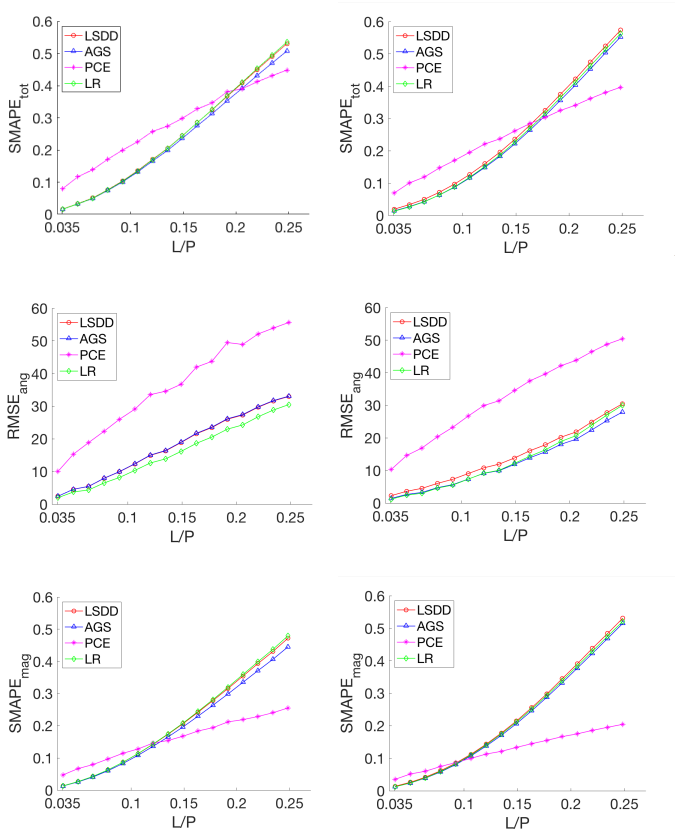


Fig. 14. Plot of total error (top line), angle error (middle line), and magnitude error (bottom line) on the 3D structured (left column) and unstructured mesh (unstructured column).

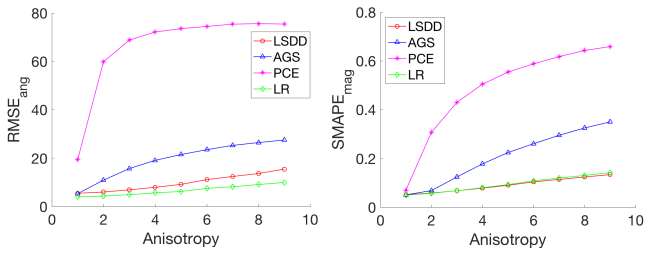


Fig. 15. Plot of angle error (left) and magnitude error (right) of the four methods computed on unstructured meshes with increasing levels of anisotropy. We set the input signal with frequency parameter $b = 10$.

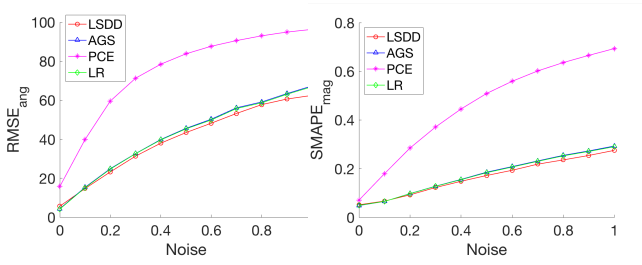


Fig. 16. Plot of angle error (left) and magnitude error (right) of the four methods computed on a structured mesh, where the function sampled at its vertices was affected by increasingly high displacement to simulate noisy data. We set the input signal with frequency parameter $b = 10$.

right); we conjecture that less regular shape of cells and 1-rings have a negative effect on the accuracy of all methods.This seems confirmed by the fact that LR is most resilient to this kind of 1-rings, consistently with its superior behaviour on anisotropic meshes.

Anisotropy. The results of the experiments made on the variation of the coefficient of anisotropy are shown in Figure 15. Here results lead to slightly different considerations compared to what we observed in the 2D case: PCE suffers much more the increase of coefficient K and this, in turn, affects the performance of AGS, which turns out to be the less accurate among the vertex-based methods. LR seems to be almost insensitive to the increase in anisotropy of the mesh, while LSDD has an intermediate behaviour.

Noise. Figure 16 reports both angle and magnitude error for growing values of noise in the test function. As in the planar case, we observe a super-linear increase of the magnitude error for PCE, while all vertex-based methods exhibit a similar behaviour in terms of magnitude error. Concerning angle error, we see that PCE has the worse performance. In this case, we notice that the differences in term of accuracy between the three vertex-based methods are less evident compared to the 2D case.

5.5. *Error distribution*

In this section, we analyze the distribution of the total error for the various methods, by using box plots. For each experiment made in the previous sections, we compute the error distributions at the starting value, at an intermediate value, and at the final value of the parameter scale. For the case of variable frequency (Figure 17) and variable anisotropy (Figure 18), we show the box plots of both the 2D and the 3D case, in order to highlight differences; for the case of variable noise (Figure 19), we show just the box plot for the 2D case, as its 3D counterpart shows a nearly identical behaviour for all methods.

The red line inside each box indicates the value of the median error; the box encloses all data between the 25th and 75th percentiles, respectively; the whiskers extend to the most extreme data points not considered outliers; and the outliers are plotted individually using blue dots. We choose to use the maximal whisker length, e.g. $1.5 * IQR$, where IQR is the difference between the 75th and 25th percentiles of the sample data.

In the variable frequency experiment, PCE shows an error distribution much wider than all other methods; while all the vertex-based methods exhibit a similar distribution. Not surprisingly, the distribution becomes wider as the L/P ratio increases. At high frequencies, the better resiliency of PCE to the L/P ratio becomes apparent as its box lies below the median of the other methods. Note how the median is higher in the 3D case for vertex-based methods, although at low frequencies they present much less outliers w.r.t. the 2D case.

In the 2D variable anisotropy experiment, PCE exhibits both a worse mean and a much wider distribution than the vertex-based methods, while AGS exhibits a slightly wider distribu-

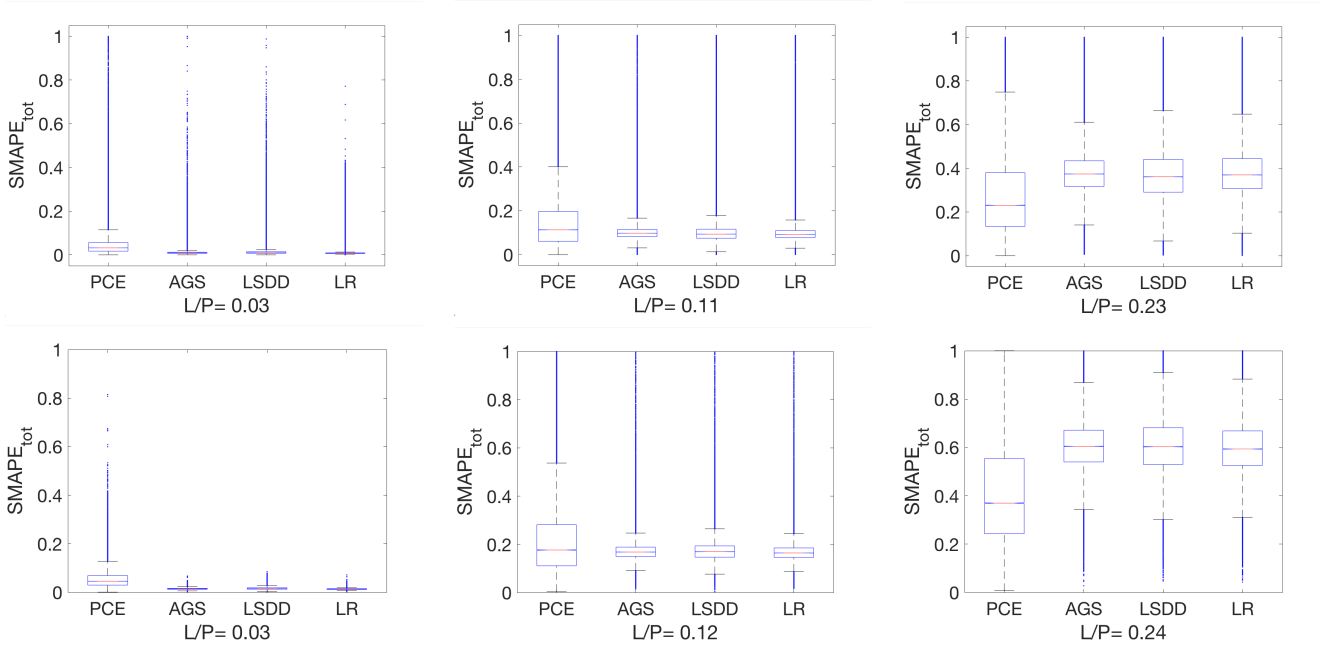


Fig. 17. Box plots of error distribution for $L/P = 0.03, 0.11, 0.23$ for the 2D case(top row) and for $L/P = 0.03, 0.12, 0.24$ for the 3D case (bottom row).

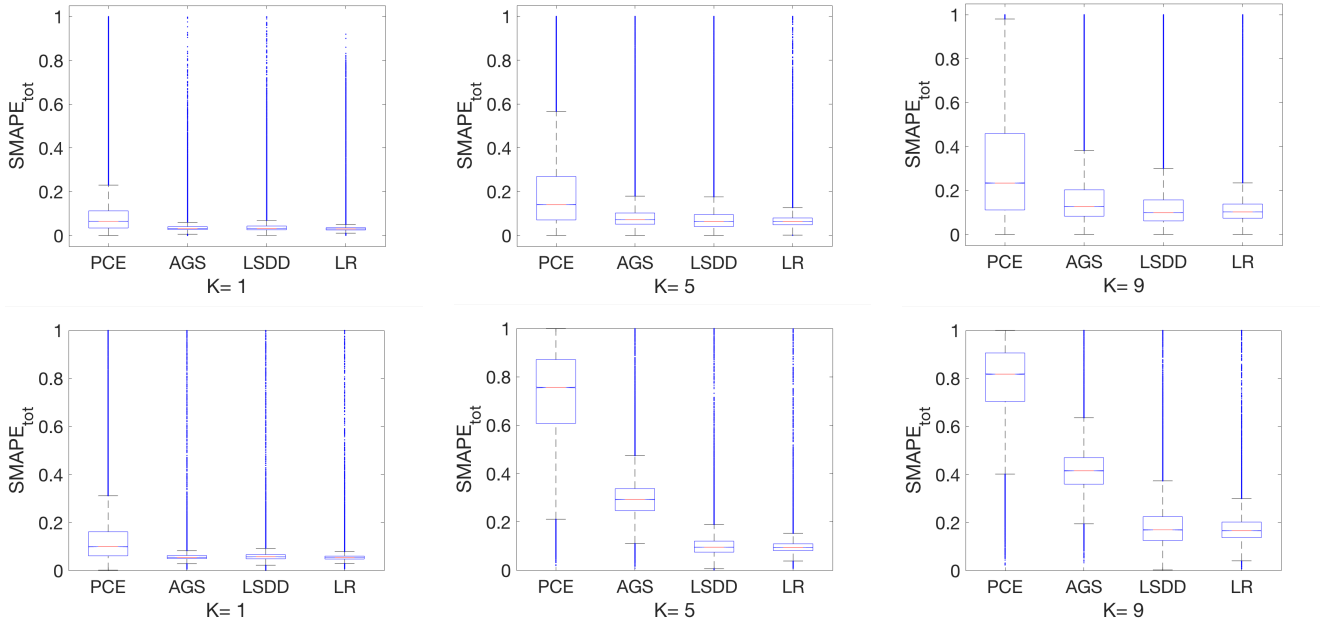


Fig. 18. Box plots of the error distribution for varying anisotropy with $K = 1, 5, 9$ for the 2D case (top row) and for the 3D case (bottom row).

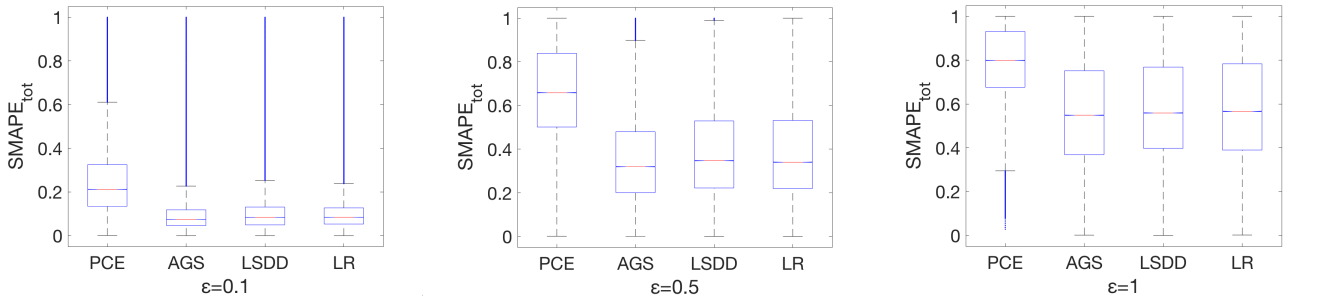


Fig. 19. Box plots of the error distribution for varying noise level with $\varepsilon = 0.1, 0.5, 1$ for the 2D case.

tion than the LSDD and LR. In the 3D case, the increase in error of PCE becomes dramatic and also the difference between the distribution of AGS from the distributions of LSDD and LR becomes more relevant, at intermediate levels of anisotropy already. The slightly better resiliency of LR over LSDD at high anisotropy levels is witnessed by a thinner distribution, the two medians being comparable, though.

The variable noise experiment confirms a definitely worse overall performance of PCE over the vertex-based methods, but the error distribution of PCE maintains a nearly constant variance at increasing noise, while the variance of the vertex-based methods increases dramatically, becoming wider than that of PCE at high levels of noise. Distributions of the three vertex-based methods are very similar, with a slight superiority of AGS.

6. Computational cost

Here we discuss the four tested methods with respect to their computational cost. All methods support two possible implementations: in a *one-shot* implementation, the gradient is estimated directly by taking in input the mesh and the signal; in an *amortised* implementation, some pre-computation, which depends just on the mesh, is performed once and for all, while the gradient evaluation is computed efficiently from the pre-computed structures each time the input signal changes. The cost of the one-shot implementation is roughly equal to the sum of costs of the two phases in the amortised implementation; while the cost of gradient evaluation is expected to be much lower than the cost of pre-computation in the amortised implementation. Therefore, the amortised implementation is always convenient when gradient computation is repeated for multiple signals. Given a mesh with T triangles/tetrahedra and V vertices, we summarise the pre-computation and the gradient estimation phases for each method below:

- **PCE** can be efficiently packed into a $dT \times V$ sparse matrix G, having just four $d+1$ non-zero entries per row, where $d = 2, 3$ for the 2D and 3D case, respectively. Multiplying G for a column vector containing the function values at each vertex in the mesh, returns a dT long column vector containing the serialised per triangle gradient.

- **AGS** is similar to PCE, but matrix G will have size $dV \times V$, where the number of non-zero entries per row is equal to the valence of the corresponding vertex, being 6 and 13 on average for the 2D and 3D case, respectively. Multiplying G with the scalar field vector returns a dV vector containing the serialised per vertex gradient. Note that, for both PCE and AGS, having the gradient in matrix form is useful to define a discrete divergence operator, which is nothing but the matrix G^T, which transforms gradients into per vertex divergence values by means of matrix vector multiplication [20].

- **LSDD** consists of a collection of dV linear systems of dimension $d \times d$ each. The matrix in the left term can be assembled and pre-factorised; each system can be solved efficiently after pre-factorisation. Note that the signal part in the right term must be reassembled each time the signal changes. In order to exploit the parallelism of linear algebra libraries, all systems can be assembled in a unique sparse block matrix consisting of 3×3 blocks along the diagonal. Since all linear systems are independent, however, this further optimization does not reduce the total number of operations.

- **LR** can be implemented similarly to LSDD, only in this case there will be dV linear systems of dimension 6×6 and 10×10 each, for the 2D and 3D case, respectively.

Method	2D		3D	
	pre-computation	*gradient estimation*	*pre-computation*	*gradient estimation*
PCE	$38V$	$20V$	$221V$	$106V$
AGS	$44V$	$22V$	$185V$	$75V$
LSDD	$129V$	$44V$	$469V$	$125V$
LR	$822V$	$174V$	$4317V$	$553V$

Table 1. Computational cost of each gradient estimator, obtained counting the number of floating point operations for both the pre-processing and the gradient computation phases. Costs are presented in terms of the number of vertices V. Since the complexity depends on the number of incident vertices and faces, whenever needed we substitute those values with the corresponding average valences in a CDT.

In Table 1, we report an estimate of the computational cost of such implementations in terms of number of floating-point operations for the pre-processing and for the gradient computation phases, both for the 2D and for the 3D case, respectively. Note that we disregard completely the cost of retrieving incidence and adjacency relations between simplices in the mesh, as this cost is highly dependent on the underlying data structure. All figures are referred to the number of vertices V in the input mesh; since some figures actually also depend on the number of triangles/tetrahedra in the mesh, we reported all such parameters to V. For the 2D case, we rely on the Euler formula, giving $T \approx 2V$; likewise, the average number of neighbors and triangles in the star of a vertex is 6. Since the Euler formula for the 3D case does not provide an estimation of the number of tetrahedra in terms of V, we rely on figures from our unstructured mesh (CDT), where $T \approx 5V$; likewise, the average number of neighbors and tetrahedra in the star of a vertex is 13 and 21, respectively.

The pre-computation phase of PCE involves just a straight computation of Eq. 1 and 2, for $d + 2, 3$, respectively; while gradient estimation has just the cost of a sparse matrix vector product. For the pre-computation phase of AGS, instead of implementing Eq. 4, which involves computing PCE first, we directly compute the weighted coefficients corresponding to the neighbours of each vertex, as in the traditional Green-Gauss method of Eq. 3: this approach simplifies computation by avoiding terms corresponding to faces incident at the central vertex v, which cancel in the Green-Gauss method, while limiting computations to faces opposite to v. The gradient estimation phase for AGS is again a sparse matrix vector computation, with about the same number of non-zero entries of PCE in 2D, and a smaller number of non-zero entries than PCE in 3D. For the pre-computation phase of LSDD and LR, we count all the operations necessary to build the square systems in Eq. 6 and

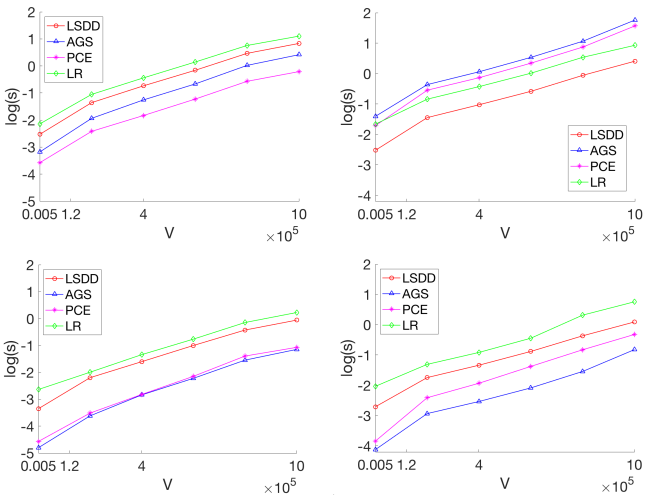


Fig. 20. Computational times associated to each gradient estimator in 2D (left) and 3D (right) for the *one-shot* implementation (top) and the gradient estimation phase of the *amortized* implementation (bottom). Mesh size varies between 500 and one million vertices, both in 2D and in 3D; times are reported in log scale.

Eq. 7, respectively, and then we add the complexity of a QR factorization with Householder (see [21], Lec.10). While the gradient estimation phase for LSDD and LR involves the cost of a matrix vector computation to build the right hand term, plus the cost of resolving a triangular system.

In Figure 20 we plot computation times of the four methods we tested for meshes with growing size. We report times for both the one-shot implementation and the gradient estimation phase of the amortised implementation. In the one-shot implementations, in 2D all four methods exhibit the same asymptotic behaviour, with LR being the more expensive, and PCE being the least expensive. In 3D, PCE and AGS have a higher cost, and seem to scale worse. While there are no conceptual differences between the 2D and 3D case, we believe this increased cost is due to the specific data structures we used, which are tailored for general polygonal and polyhedral meshes [18], and do not fully exploit memory locality, leading to an overly high cost to access topological relations between cells. In the amortised implementation, all methods run faster by at least one order of magnitude than their related one-shot implementation. In 2D, PCE and AGS run almost equally fast and overall at least one order of magnitude faster than LSDD and LR, LSDD being faster than LR for about a factor of two. In 3D, differences become more dramatic: AGS runs faster than PCE, being based on a smaller matrix, and almost three orders of magnitude faster than its one-shot version; PCE also runs about two orders of magnitude faster than its one-shot version, but it is about three times slower than AGS. LSDD is just about twice faster than its one-shot version, and about one order of magnitude slower than AGS. LR exhibits the worst performance, being between two and three times faster than its one-shot version, but still much slower than LSDD, especially when the size of the mesh grows.

7. Summary

In Table 2, we summarize on an ordinal scale the performances of the various methods. As it can be easily seen, there is no clear winner.

In terms of overall performances on semi-regular meshes, the vertex-based methods do almost equally well, and definitely better than PCE, on estimating the direction of the gradient; while PCE provides a better estimate of the magnitude of gradient; the total error is lower for the vertex-based methods at low L/P ratio, while PCE shows a better resilience to the increase of such ratio, thanks to its smaller stencil.

All methods provide lousy estimates at the boundaries: PCE in fact exhibits the same (relatively poor) performances at the interior and at the boundary, while the vertex-based methods have a much worse behaviour at the boundary (not worse than PCE, though); LR exhibits the most unpredictable behaviour at the boundary, given to the effect of extrapolation error on the second order polynomial fit.

On the other hand LR excels on the anisotropic mesh; while all three vertex-bases methods result equally resilient to noise, and do definitely better than PCE.

In terms of computational cost, PCE and AGS result faster than LSDD and LR, both in 2D and in 3D, both in the one-shot and in the amortised case; in particular, LR results much slower than the other methods and its difference in speed is particularly relevant in 3D.

Overall, AGS seems a good choice for processing semi-regular meshes, while LR should be preferred for anisotropic meshes.

8. Extensions

The scope of our analysis is limited to the evaluation of a gradient field on simplicial meshes covering a Euclidean domain. There are several important extensions of this problem that we briefly review in the following.

8.1. Polygonal/polyhedral meshes

Linear interpolation of function f cannot be used on non-simplicial cells. Per-cell gradient estimation can be obtained by applying the Green-Gauss formula to each cell σ

$$\int_\sigma \nabla f dV = \int_{\partial\sigma} f \mathbf{n} dA$$

where $\mathbf{n}$ is the normal direction to the boundary of σ. In [5] this approach is investigated in detail and some computational alternatives are proposed. Note that this formula gives the same result of per-cell linear estimation in case σ is a simplicial cell.

The extension of per-vertex methods is straightforward. Once per-cell constant gradient has been obtained, the method described in Section 3.2.1 can be applied with no change, by averaging gradients in the incident cells of each vertex. Also the methods described in Sections 3.2.2 and 3.2.3 can be applied directly, since they only require retrieving vertices in the star (or k-ring) of a given vertex. In the latter case, since the direct neighbors of each vertex are usually fewer in a polyhedral mesh, all vertices of cells incident at a given vertex might be considered as neighbors.

Method	Overall performance			Resilience to:			Comp. cost	Pathologies
	interior		boundary	poor L/P	anisotropy	noise		
	direction	*magnitude*						
PCE	- -	+	-	+	-/- -	- -	++/+	critical points
AGS	++	-	-	-	+/-	+	++/+	critical points
LSDD	++	-	-	-	+	+	-	critical points
LR	++	-	- -	-	++	+	- -	critical points on boundary

Table 2. Summary of the performance of the various methods on an ordinal scale (- -, -, +, ++) on all aspects analyzed. We obtained coherent results for the 2D and 3D cases with respect to most aspects: in that case just one symbol is reported; we report two symbols separated with a slash for the 2D / 3D case, respectively, just where we found relevant differences.

8.2. Manifold domains

When domain Ω is a manifold, such as a surface embedded in 3D space, gradient fields are defined in tangent space. Per-face gradient estimation can be applied unchanged, because the tangent plane at a face of a surface mesh coincides with the plane of the face itself. Conversely, the estimation of gradients at vertices, as well as their extension to the whole domain, need to bring the necessary information to the tangent planes at vertices, either through an affine connection (for AGS), or through an exponential map (for LSDD and LR). An excellent reference about computing exponential maps and affine connections is provided by [22]. There exists a vast literature on the design and analysis of vector fields on surfaces. However, to the best of our knowledge, the explicit problem of estimating gradient fields per-vertex on a surface has not been investigated in the literature. In a recent survey [1], several techniques for vector field processing are reviewed, yet gradient estimation is reported only under the per-face approach.

8.3. Field tracing

Once a gradient has been estimated on a mesh Σ, it is possible to trace its integral lines. Tracing a piecewise-constant gradient is straightfoward and gives piecewise-linear integral lines, but it often produces artifacts.

A piecewise-linear gradient model is much more reliable, but it is also much more difficult to trace. As shown in [3], the integral lines of a linear vector field on a simplex may be expressed in analytic form with a rather complex parametrization, which cannot be easily intersected with the boundary of the cell. Tracing such exact lines has been attempted in [2], but it may lead to numerical issues. One straightforward alternative consists of computing a piecewise-linear approximation of an integral line: starting at a given point, the line is traced by considering a polyline that follows the gradient for a given step at each node; with this approach, the intersection with the boundary of cells is easy, but the tracing procedure is prone to a potentially large drift which may propagate at each step. Although this solution converges to the exact one, as the length of the step tends to zero, it is hard to set a step that guarantees a bound to the drift. One dangerous consequence of accumulated numerical errors and/or drift is that integral lines that should proceed parallel may meet and intersect, thus corrupting the topology of the field.

Some recent approaches try to address the topological correctness of a piecewise-linear vector field on a surface, by detecting coherent bundles of integral lines that travel parallel inside a cell [23, 24, 25]. Under these approaches, big drifts are avoided and the overall topological structure of the field is maintained, at the cost of a rougher approximation of lines inside a given cell.

9. Concluding remarks

We presented an experimental evaluation in 2D and 3D of four different gradient estimators for simplicial meshes. We started from the consideration that the ubiquitous piece-wise constant gradients, obtained computing the partial derivatives of the function sampled at the mesh vertices and linearly extended within each cell, suffer from a number of limitations (Section 1).

In our study, we have considered a family of periodic functions, and tested each method on a variety of discrete meshes which are common in applied sciences, also testing resiliency to noise and computational performances.

Experiments confirm that overall – due to its piece-wise constant nature – PCE performs worse than any vertex-based gradient estimator that we tested. The worst performances come for anisotropic meshes or signals affected by noise, where PCE has angle error which grows as the same rate observed for vertex-based gradients, and magnitude error which grows at a much higher rate (Figures 12 , 15 , 13 and 16)x. This is also probably due to its higher locality, which makes it extremely sensitive to perturbations in the signal or sampling, although it is not yet clear why this effect manifests itself more on the magnitude component than on angles. On the positive sides, the locality of PCE makes it more resilient to the increase of the signal frequency in estimating the magnitude of the gradient (Figures 10 and 14). Furthermore, being computed separately on each triangle, PCE is basically insensitive to boundaries (Figure 8).

Restricting to vertex-based methods, linear regression (LR) seems to do slightly better than AGS and LSDD when the mesh is quite refined with respect to signal frequency (Figures 10 and 14), and has a superior behavior on anisotropic meshes (Figures 12 and 15). However, LR suffers the most on boundaries, (Figures 4). Furthermore, LR is the most expensive method, as it requires solving a linear system for each vertex, making it hardly scalable on big meshes (Figure 20). AGS and LSDD often show the same asymptotic behaviour, but AGS has lower error in every experiments, except the one made on the anisotropic 3D mesh, and it has lower computational cost.

For a standard use, AGS seems to offer the best compromise between stability, accuracy and computational cost. LR performs better on anisotropic meshes and may offer more uniform accuracy at a slightly larger computational cost, but it has a quite unpredictable behaviour at the boundaries.

A common trait of all the meshes we considered – both procedural and Voronoi based – is to expose a fairly even distribution of per vertex valences. Meshes that do not fall in this category may possibly bias the results of our study. While valence irregular meshes could certainly be created with interactive modeling tools; we did not investigate such meshes because we are not aware of any automatic tool capable to synthesize them. Conversely, the scientific literature offers various techniques to harmonize mesh connectivity and transform irregular meshes into semi-regular or regular ones [26].

For future works, we aim at extending this study considering estimation on manifold domains and integral curve tracing problems. The latter seems to be particularly interesting (and challenging) for the vertex-based methods, where singularities may arise at any point inside elements domain and not only at the mesh vertices. Global methods, e.g., based on radial basis functions, could also offer a valid alternative to the local methods investigated here: it would be interesting to investigate the trade-off between accuracy and computational cost, which is probably higher.

Acknowledgements

This work is supported by the Italian Ministry of Education, University and Research under Program PRIN 2015, Project DSurf, Grant N. 2015B8TRFM˙002, and by the EU ERC Advanced Grant CHANGE, grant agreement No 694515.

References

[1] De Goes, F, Desbrun, M, Tong, Y. Vector field processing on triangle meshes. In: ACM SIGGRAPH 2016 Courses, SIGGRAPH 2016. Pixar Animation Studios, United States; New York, New York, USA: ACM Press; 2016, p. 1–49.

[2] Kipfer, P, Reck, F, Greiner, G. Local Exact Particle Tracing on Unstructured Grids. Computer Graphics Forum 2003;22(2):133–142.

[3] Nielson, GM, Jung, IH. Tools for computing tangent curves for linearly varying vector fields over tetrahedral domains. IEEE Transactions on Visualization and Computer Graphics 1999;.

[4] Mancinelli, C, Livesu, M, Puppo, E. Gradient Field Estimation on Triangle Meshes. In: Smart Tools and Apps for Graphics - Eurographics Italian Chapter Conference. The Eurographics Association; 2018,.

[5] Sozer, E, Brehm, C, Kiris, CC. Gradient calculation methods on arbitrary polyhedral unstructured meshes for cell-centered CFD solvers. In: 52nd AIAA Aerospace Sciences Meeting - AIAA Science and Technology Forum and Exposition, SciTech 2014. Science and Technology Corporation, Hampton, Hampton, United States; Reston, Virginia: American Institute of Aeronautics and Astronautics; 2014,.

[6] Hyman, JM, Shashkov, M. Natural discretizations for the divergence, gradient, and curl on logically rectangular grids. Computers and Mathematics with Applications 1997;33(4):81–104.

[7] Neumann, L, Csebfalvi, B, Konig, A, Groller, E. Gradient estimation in volume data using 4D linear regression. Computer Graphics Forum 2000;19(3):C351–C357.

[8] Jirka, T, Skala, V. Gradient Vector Estimation Using Quadratic Regression Function. In: Int. Conf. on Computer Vision and Graphics. Zakopane, Poland; 2002,.

[9] Correa, CD, Hero, R, Ma, KL. A comparison of gradient estimation methods for volume rendering on unstructured meshes. IEEE Transactions on Visualization & Computer Graphics 2009;(3):305–319.

[10] Cerbato, G, Hurtado, FS, da Silva, AFC, Maliska, CR. Analysis of gradient reconstruction methods on polygonal grids applied to petroleum reservoir simulation. In: 15th Brazilian Congress of Thermal Science and Engineering, Belém. ENCIT Proceedings. 2014,.

[11] Meyer, M, Desbrun, M, Schröder, P, Barr, AH. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In: Visualization and Mathematics III. Berlin, Heidelberg: Springer, Berlin, Heidelberg; 2003, p. 35–57.

[12] Correa, CD, Ma, K, Hero, R. A comparison of gradient estimation methods for volume rendering on unstructured meshes. IEEE Transactions on Visualization and Computer Graphics 2009;17:305–319.

[13] Thrmer, G, Wthrich, CA. Normal computation for discrete surfaces in 3d space. Computer Graphics Forum 1997;16(3):C15–C26. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00138. doi:10.1111/1467-8659.00138. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.0

[14] Shewchuk, JR. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In: Applied computational geometry towards geometric engineering. Springer; 1996, p. 203–222.

[15] Dompierre, J, Labbé, P, Vallet, MG, Camarero, R. How to subdivide pyramids, prisms, and hexahedra into tetrahedra. In: IMR. 1999, p. 195–204.

[16] Si, H. Tetgen, a delaunay-based quality tetrahedral mesh generator. ACM Transactions on Mathematical Software (TOMS) 2015;41(2):11.

[17] Flores, BE. A pragmatic view of accuracy measurement in forecasting. Omega(Oxford) 1986;14:93–98.

[18] Livesu, M. cinolib: a generic programming header only c++ library for processing polygonal and polyhedral meshes. 2017. Https://github.com/mlivesu/cinolib/.

[19] Guennebaud, G, Jacob, B, et al. Eigen v3. http://eigen.tuxfamily.org; 2010.

[20] Livesu, M. A heat flow relaxation scheme for n dimensional discrete hyper surfaces. Computers & Graphics 2018;71:124 – 131. doi:10.1016/j.cag.2018.01.004.

[21] Trefethen, LN, Bau, D. Numerical Linear Algebra. SIAM; 1997. ISBN 0898713617.

[22] Liu, B, Tong, Y, Goes, FD, Desbrun, M. Discrete connection and covariant derivative for vector field analysis and design. ACM Trans Graph 2016;35(3):23:1–23:17. URL: http://doi.acm.org/10.1145/2870629. doi:10.1145/2870629.

[23] Bhatia, H, Jadhav, S, Bremer, P, Chen, G, Levine, JA, Nonato, LG, et al. Flow Visualization with Quantified Spatial and Temporal Errors Using Edge Maps. Visualization and Computer Graphics, IEEE Transactions on 2012;18(9):1383–1396.

[24] Ray, N, Sokolov, D. Robust polylines tracing for n-symmetry direction field on triangulated surfaces. ACM Transactions on Graphics (TOG) 2014;33(3):30.

[25] Myles, A, Pietroni, N, Zorin, D. Robust field-aligned global parametrization. ACM Transactions on Graphics 2014;33(4):1–14.

[26] Alliez, P, Ucelli, G, Gotsman, C, Attene, M. Recent advances in remeshing of surfaces. In: Shape analysis and structuring. Springer; 2008, p. 53–82.