LEIBNIZ UNIVERSITÄT HANNOVER

Seminar

Gradient evaluation on non-orthogonal meshes

Jose Abraham Caravaca Fernández

21/07/2011

Content

1. Introduction: some definitions

The gradient of a scalar function $f(x)$is a measure of how the value of the function changes: the greater the change the greater the magnitude of the gradient. If we consider the case in which the functions are continuous in their domain space, we can calculate the gradient

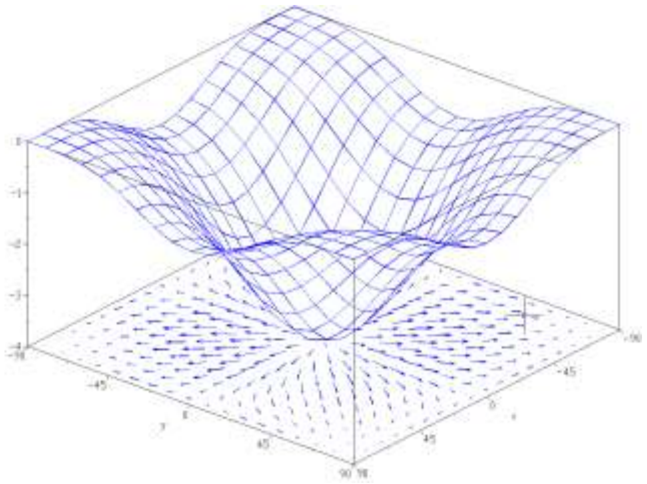
$$\nabla f(x) = \frac{\partial f(x)}{\partial x} = \lim_{h \Rightarrow 0} \frac{f(x) - f(x+h)}{h}$$

Given the case in which the function is multidimensional (i.e. normal rectangular coordinates) we can define the gradient as follows:

$$\nabla f(x,y,z) = \begin{pmatrix} G_x \\ G_y \\ G_z \end{pmatrix} = \begin{pmatrix} \dfrac{\partial f(x,y,z)}{\partial x} \\ \dfrac{\partial f(x,y,z)}{\partial y} \\ \dfrac{\partial f(x,y)}{\partial z} \end{pmatrix}$$

Given this notation, we can define the direction of greatest change as the direction of the normalized gradient vector and as an idea of its magnitude the length of the vector.

As an example let us consider a surface given by a three dimensions function (field). If we intend to see how the function changes over the space, we can calculate the gradient. We can see an example here:



Mathematically, the gradient of a two-variable function (as shown up) is at each image point, a 2D vector with the components given by the derivatives in the horizontal and vertical directions (drawn in the flat plane). At each image point, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction.

1.1 Possible applications of the gradient

What applications or fields may seize the gradient as a tool? A brief proposal for the main uses of the gradient could be:

1) Gradient solvers. These are methods to find minima or extrema of functions or solve linear equation systems where the function or the equations are **known**:

- Gradient descent/ascent
- Conjugate gradient method

2) Gradient approximation methods. In this case values are given on a discrete grid, which are supposed to be the result of some function which is **unknown**. The gradient can help to characterize this function, i.e., compute isovalue sets.

As a possible application of the gradient in one system in which the equations can be solved, because we have all necessary data to apply a direct calculation of the gradient without any approximation, obtaining this way a exact result for the case, would be the case of image processing.

The gradient is a very useful tool in the field of image processing. Image gradients can be used to extract information from images. Gradient images are created from the original image, generally by convolving with a filter, one of the simplest being the *Sobel filter*. [1] The gradient, also known as ***first derivative,*** can be defined the way, above explained, for continuous functions. But attending to the pragmatical view, in an image or, more in general, as a constraint of computational means, where the number of samples is always finite, a continuous derivative cannot be performed. Instead, the difference value between adjacent pixels can be calculated as a "*discrete*" derivative. This difference is somehow noisy but averaging in the direction perpendicular to the derivative can smooth the result.

A typical kernel often used for directional (horizontal and vertical) first derivative is:

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

If we aim to use this kernel on an image **we need a regular grid** (same distribution on the whole structure) and we would just apply the mask pixel by pixel, multiplying the central pixels and their neighborhood and then calculating the mean:

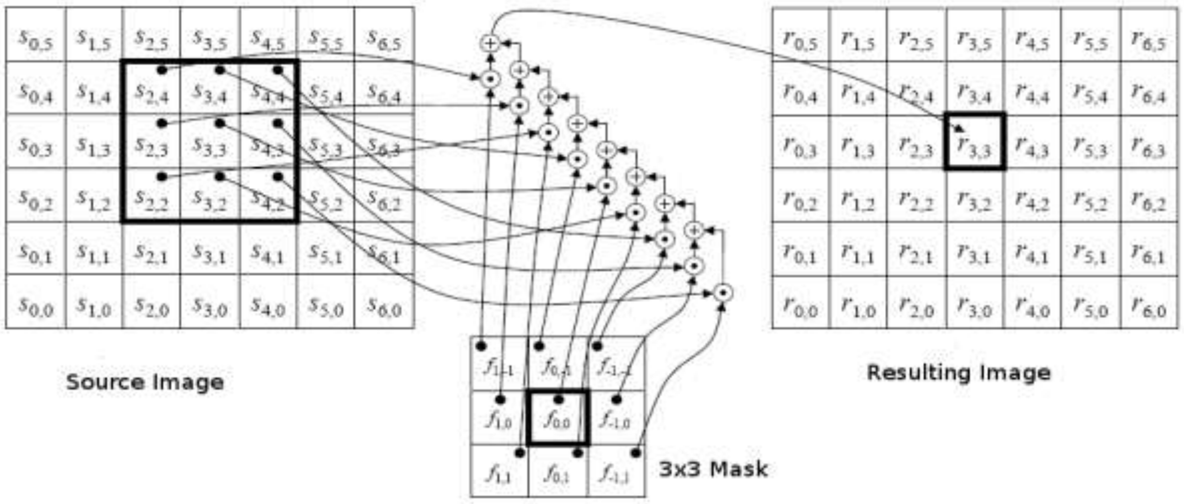


Illustration 1: A 2-D example of filtering is depicted.

Attending to the previous graphic we can see that every square has the same area. But what if we have some images in which we want to apply a mesh, which is not regular because it has different resolutions (distance between vertex) for the divisions, prioritizing this way some parts of the image, that we consider of more interest due to specific reasons (Region of interest)?

Here is where a non-regular mesh might come handy for the analysis. For this kind of analysis with the gradient we need special methods, we will discuss some of them on these pages.

1.2 Definition of bordered subset and its approximation

As I just said we would need to use non-regular meshes in the analysis of some images in which we use this certain methods. But first of all, let me state some definitions.

A mesh can be viewed as a compound of polytopes. In elementary geometry, a **polytope** is a geometric object with flat sides, which exists in any general number of dimensions. A polygon is a polytope in two dimensions, a polyhedron in three dimensions, and so on in higher dimensions (such as a polychoron in four dimensions).

The original approach, broadly followed, begins with the 0-dimensional point as a 0-polytope (vertex). A 1-dimensional 1-polytope (edge) is constructed by bounding a line segment with two 0-polytopes. Then 2-polytopes (polygons) are defined as plane objects whose bounding facets (edges) are 1-polytopes, 3-polytopes (polyhedra) are defined as solids whose facets (faces) are 2-polytopes, and so forth:

Dimension	Element name (in an n-polytope)
-1	Null polytope
0	Vertex
1	Edge
2	Face
3	Cell
…	…
J	j-face – element of rank j = − 1, 0, 1, 2, 3, ..., n
…	…
n-3	Peak – (n−3)-face
n-2	Ridge – (n−2)-face
n-1	Facet – (n-1)-face
N	Body – n-face

A bordered subset $B: \mathfrak{R}^n$ is **linearly bordered**, if the boundary consists of (n-1)-dimensional polytopes.

Given a bordered subset $B: \mathfrak{R}^n$, then a linearly bordered subset $B: \mathfrak{R}^n$ whose boundary consists of polygons is an **approximation of B**, if all vertices ofB_h's boundary lie on the boundary of B. If $n = 2$ (Euclidean plane case), then B is a small area, and an approximation B_h is bounded by a closed polygon.

1.3 Definition of mesh

Given an approximation B_h of B and a finite set $M = \{b_1, \ldots, b_m\}$ of linearly bounded subsets of B_h. If

1 $B_h = b_1 \cup b_2 \ldots \cup b_n$

2 Each b_i is of dimension m.

3 $b_i \cap b_j$ is empty or an n-simplex with $n < m \;\; \forall i, j \in \{1, \ldots, m\} / i \neq j$.

Then M is called **conforming** mesh (or grid). If only 1 and 2 are valid, M is called **non-conforming**. Vertices of b_i that are on the boundary of B_h are called **boundary vertices** all others are called **inner vertices**.

If all elements have identical shape, the mesh is called **regular**. A 2D conforming mesh that consists only of triangles is called Triangulation, that consists only of quadrangles is called Quadrangulation. A 3D conforming mesh is called tetrahedrization, and so forth and so on.

For all $V = \{ v : v \; is \; inner \; vertex \; of \; B_h \}$, $|v|$ is the number of elements of M which have v as vertex. If for all v, $|v|$ is the same, the mesh is called **structured**, else **unstructured**.

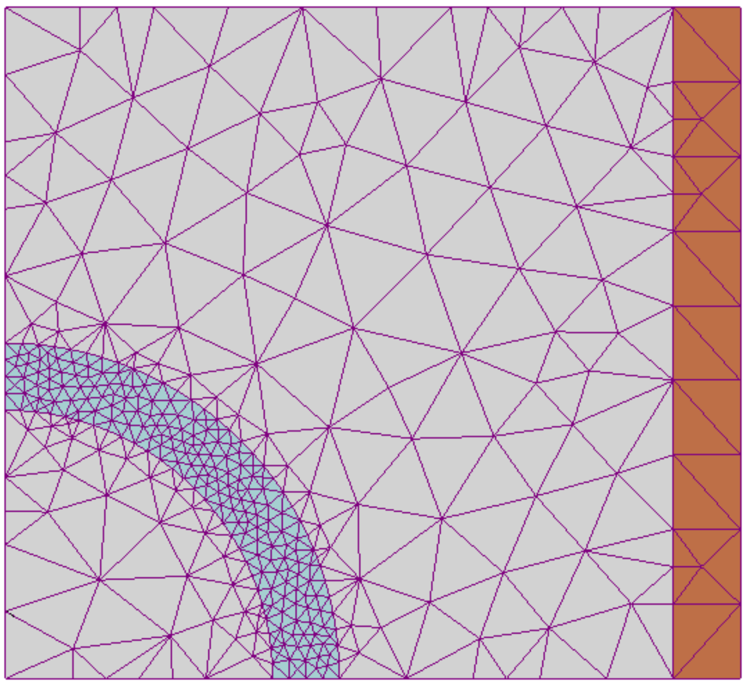


Illustration 2 [5]: example of a 2D FEM (Finite Element Modeling) solution for a cylindrically shaped magnetic shield. The mesh is denser around the object of interest. On the right there is the exciting coil (with electric current).

2. What is the problem?

Gradient estimation is well known and understood for regular grids, and its application is now part of commodity visualization systems. But when dealing with unstructured meshes the partial derivatives of a function with respect to the X,Y and Z dimensions cannot be approximated using finite differences from any alignment of a voxel neighborhood with each of the axes.

The process of estimating the gradient on structured meshes is usually simple, but meshes, as the one used in FEM (finite element methods), are often unstructured, because of the requirement to use different resolutions in different parts of the analyzed volume. Computing the gradient of a scalar function on such a grid is not directly feasible.

Gazing at this problem we can whether:

1 Re-sample the mesh into a structured one, where the computation of gradient is simple. Also after re-sampling might be necessary some [9] mesh adaptation, often referred to as Adaptive Mesh Refinement (AMR), refers to the modification of an existing mesh to improve resolution for certain features under study. But this operation is costly in computational meanings, and it may not be feasible at all in some cases. [7] As in the case of local illumination of unstructured-mesh volume data, for example, re-sampling an unstructured mesh into a 3D regular grid at the Nyquist rate might result in very large volumes that exceed the available system or graphics memory.

2 Approach the problem using Finite Element Methods: a numerical technique for finding approximate solutions of partial differential equations (PDE) as well as of integral equations.

2.1 One possible solution: Isoparametrical transforms

As an instance for clarifying the method a general, non-orthogonal, non-equidistant, 3D structured mesh of hexahedra is considered [7]. The cells can be logically indexed by three indices (I,j,k). Let $\vec{x}^{c}_{i,j,k}$ be the center of the cell (i, j, k). The discrete values $\phi_{i,j,k}$ are the numerical approximations, at the cell centers, of a continuous field ϕ. Our objective is to find an expression for $\Delta\phi(\vec{x}^{c}_{i,j,k})$ (cell-centered gradient) and $\Delta\phi(\vec{x}^{s}_{i,j,k})$ (gradient at one of the cell surface centers). Let us consider a simple case depicted in the next figure:

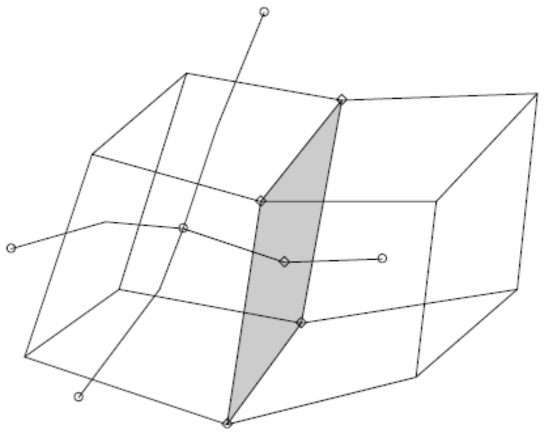


Illustration 1: two cells of a mesh in which ○ represent cell centers, and ◊ represents interpolated points for the cell-surface gradient estimation method.

The problem could be stated as: given a "central" point $\overrightarrow{x_0}$, six points $\overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}, \overrightarrow{x_4}, \overrightarrow{x_5}, \overrightarrow{x_6}$ around $\overrightarrow{x_0}$, and the values ϕ_i, $i = 0, \dots, 6$ of the discrete field ϕ, find $\nabla\phi$ at those points.

Case: cell-centered gradient evaluation

The points to be considered are simply the center cells center: $\vec{x}^{c}_{i,j,k}$ and its six nearest neighbor cell centers

$$\vec{x}^{c}_{i,j,k} \rightarrow \{\vec{x}^{c}_{i+1,j,k}, \vec{x}^{c}_{i-1,j,k}, \vec{x}^{c}_{i,j+1,k}, \vec{x}^{c}_{i,j-1,k}, \vec{x}^{c}_{i,j,k+1}, \vec{x}^{c}_{i,j,k-1}\}$$

The field values are directly known at those points.

Case: surface-centered gradient evaluation

The "central" point is a surface center, while the other six points are the corners of that surface and the two centers of the cells sharing that surface. Nevertheless, one must first interpolate ϕ at the five points located on the surface, since ϕ is known only at cell centers. The next figure shows a cell and one of its neighbors, along with some of the points used for the gradient calculations.

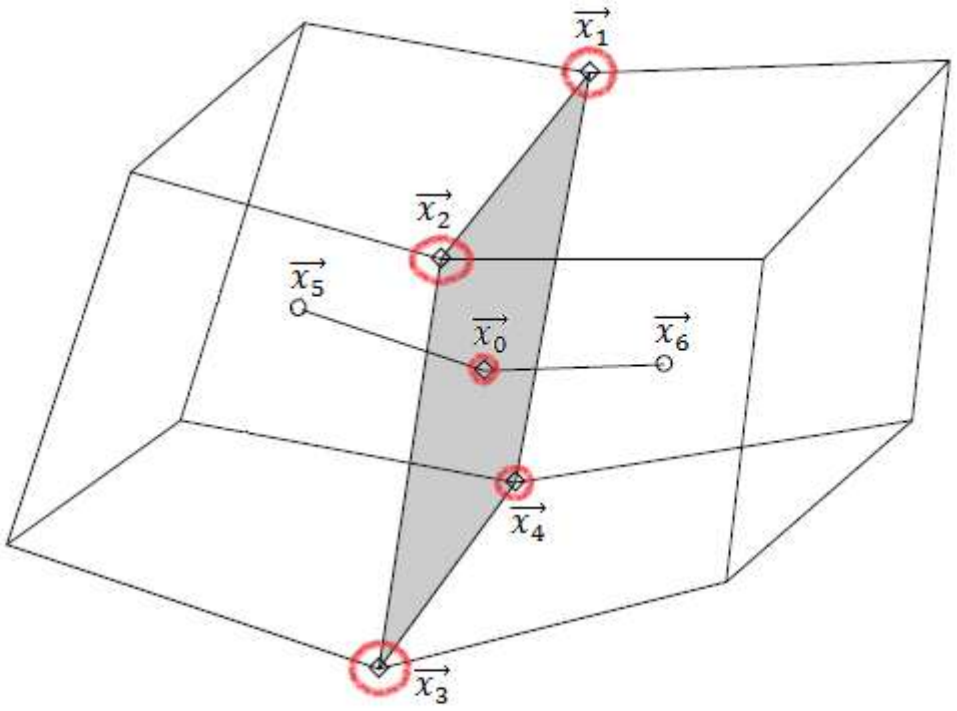


Illustration 2: The points in red must me interpolated

The isoparametric transform, $r(u,v,w)$ that maps a (regular) polyhedron in (u,v,w) coordinates to its (irregular) counterpart in ðx; y; zÞ coordinates may be written:

$$r(u,v,w) = \sum_{\alpha=1}^{n} N_\alpha(u,v,w)\overrightarrow{r_\alpha{}^v}$$

Given such set of points and field values (only at cell centers), it is needed to estimate the gradient at the "central" point. We can do so using the isoparametric transform:

$$\phi(u,v,w) = \sum_{\alpha=1}^{n} N_\alpha(u,v,w)\phi\left(\overrightarrow{r_\alpha{}^v}\right)$$

Where:

n: number of vertices of the polyhedron

$\overrightarrow{r_\alpha{}^v}$: the vertices of the polyhedron in X,Y,Z.

$N_\alpha(u,v,w)$: functions to be determined using interpolation to fulfill the equation

This way we can obtain estimation for $\nabla\phi|_{u,v,w}$ and relate it to $\nabla\phi|_{x,y,z}$ using:

$$\nabla\phi|_{x,y,z} = J^{-1}\cdot\nabla\phi|_{u,v,w}$$

Being the transposed Jacobian of the transform:

$$J^t = \begin{matrix} \frac{\delta u}{\delta x} & \frac{\delta v}{\delta x} & \frac{\delta w}{\delta x} \\ \frac{\delta u}{\delta y} & \frac{\delta v}{\delta y} & \frac{\delta w}{\delta y} \\ \frac{\delta u}{\delta z} & \frac{\delta v}{\delta z} & \frac{\delta w}{\delta z} \end{matrix}$$

It is not possible to write an isoparametric transform mapping the octahedron defined by the seven points considered, however it is possible to partition these points in two sets defining two tetrahedra:

$$\{\overrightarrow{x_0}, \overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}, \overrightarrow{x_4}, \overrightarrow{x_5}, \overrightarrow{x_6}\} \to \{\overrightarrow{x_0}, \overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}\}, \{\overrightarrow{x_0}, \overrightarrow{x_4}, \overrightarrow{x_5}, \overrightarrow{x_6}\}$$

The procedure to evaluate $\nabla\phi(\overrightarrow{x_0})$, given $\{\overrightarrow{x_0}, \dots, \overrightarrow{x_6}, \overrightarrow{\phi_0}, \dots, \overrightarrow{\phi_6}\}$ is then as follows:

1. Consider a first tetrahedron defined by four points ($\{\overrightarrow{x_0}, \overrightarrow{x_1}, \overrightarrow{x_2}, \overrightarrow{x_3}\}$ for example) from the seven given points, and the isoparametric transform it defines.
2. Using $\nabla\phi|_{x,y,z} = J^{-1} \cdot \nabla\phi|_{u,v,w}$, we obtain a first estimation of the gradient at $\overrightarrow{x_0}$: $\nabla\phi^{(1)}$.
3. We proceed similarly for a second tetrahedron, defined by $\overrightarrow{x_0}$ and the remaining three points ($\{\overrightarrow{x_0}, \overrightarrow{x_4}, \overrightarrow{x_5}, \overrightarrow{x_6}\}$ following last example) not used for the first tetrahedron, yielding a second gradient estimation $\nabla\phi^{(2)}$.
4. Finally, we average these two estimates: $\nabla\phi(\overrightarrow{x_0}) = \frac{1}{2}\left(\nabla\phi^{(1)} + \nabla\phi^{(2)}\right)$

Assuming that the isoparametric transform is of the form:

$$\vec{x}(u,v,w) = \overrightarrow{a_1} + \overrightarrow{a_2}u + \overrightarrow{a_3}v + \overrightarrow{a_4}w$$

And is determined by the conditions:

$$\vec{x}(0,0,0) = \overrightarrow{r_1{}^v} \quad \to \quad \overrightarrow{a_1} = \overrightarrow{r_1{}^v}$$

$$\vec{x}(1,0,0) = \overrightarrow{r_2{}^v} \quad \to \quad \overrightarrow{a_2} = \overrightarrow{r_2{}^v} - \overrightarrow{r_1{}^v}$$

$$\vec{x}(0,1,0) = \overrightarrow{r_3{}^v} \quad \to \quad \overrightarrow{a_3} = \overrightarrow{r_3{}^v} - \overrightarrow{r_1{}^v}$$

$$\vec{x}(0,0,1) = \overrightarrow{r_4{}^v} \quad \to \quad \overrightarrow{a_4} = \overrightarrow{r_4{}^v} - \overrightarrow{r_1{}^v}$$

This allows the determination of N_α:

$$N_1(u,v,w) = 1 - u - v - w$$

$$N_2(u,v,w) = u$$

$$N_3(u,v,w) = v$$

$$N_4(u,v,w) = w$$

The Jacobian matrix of the isoparametric transform (application $\mathbb{R}^3 \to \mathbb{R}^3$) is given by:

$$J^{-1} = \begin{bmatrix} \overrightarrow{a_2} \\ \overrightarrow{a_3} \\ \overrightarrow{a_4} \end{bmatrix}^{-1}$$

We can express $\phi\left(\overrightarrow{r_i^v}\right)$ as ϕ_i , $i \in \{1, \dots ,4\}$, and thus write the gradient in the (u, v, w) just like:

$$\frac{\delta\phi}{\delta u} = \phi_2 - \phi_1$$

$$\frac{\delta\phi}{\delta v} = \phi_3 - \phi_1$$

$$\frac{\delta\phi}{\delta w} = \phi_4 - \phi_1$$

Having as a result:

$$\nabla\phi^{(1)}|_{x,y,z} = \begin{bmatrix} \overrightarrow{a_2} \\ \overrightarrow{a_3} \\ \overrightarrow{a_4} \end{bmatrix}^{-1} \cdot \begin{bmatrix} \phi_2 - \phi_1 \\ \phi_3 - \phi_1 \\ \phi_4 - \phi_1 \end{bmatrix}$$

Proceeding similarly we can obtain $\nabla\phi^{(2)}|_{x,y,z}$ and therefore $\nabla\phi|_{x,y,z} = \frac{1}{2}\left(\nabla\phi^{(1)} + \nabla\phi^{(2)}\right)$.

3. A brief review of various gradient estimation methods

[7] Let us define an unstructured mesh as a collection of connected points x that discretize a scalar field f. The linear approximation of this function at a given point $x_0 + h$ is given by:

$$f(x_0 + h) = f(x_0) + \nabla f(x_0) \cdot h + O\left(||h||^2\right)$$

where $\nabla\left(f(x_0)\right)$ is the gradient at point x_0 and h is a discretization step. The goal of gradient estimation is therefore to recover the function ∇f such that the equation holds for any given point. Because the **approximation is linear**, these methods are collectively known as linear gradient reconstruction methods. We can further classify these methods into two groups:

- **averaging-based methods**, which construct the gradient as a **weighted average** of the **neighboring gradients**.
- **regression-based methods**, which posit the equation above as a **least squares problem**.

To understand the sources of error in these methods, we can expand the second term of the linear approximation. The residual of $r_2 = f - \hat{f}$, where $\hat{f}$ is the linear approximation of f, is

$$r_2(x_0 + h) = \frac{1}{2!} \cdot h^T \cdot \nabla^2 f(x_0) h + O\left(||h||^3\right)$$

Where $\nabla^2 f$ is the hessian matrix of f. Furthermore, the absolute error can be bounded as:

$$||r_2|| \leq ||h||^2 \cdot \left|\left|\nabla^2\left(f(\xi, \zeta, \eta)\right)\right|\right| / (\xi, \zeta, \eta) \in (x_0, x_0 + h)$$

Therefore, linear approximation methods are both dependent on the mesh discretization and the complexity of the scalar field. This is true for both structured and **unstructured meshes**. Unlike structured grids, unstructured meshes have a **variable discretization distance h**. Therefore, the shape of the mesh element **is** also **a factor of efficiency**.

3.1 Averaging-based methods

In this family of methods, the gradient is computed as a weighted average of functions of the gradient or scalar values at a neighborhood around a node. This method is the one **used** in the previous part of the paper when using **isoparametric transforms**. In general, this can be expressed as the linear combination

$$\nabla f(x_0) = \sum_i \omega_i \nabla f(i)$$

where ω_i is a weighting factor, and $\nabla f(i)$ is the constant gradient at a cell i. The gradient at a cell can be computed by considering

$$f(x_0 + h) = f(x_0) + \nabla f(x_0) \cdot h + O\left(||h||^2\right)$$

for the 4 vertices of a tetrahedron, here denoted as column vectors x_0, x_1, x_2 and x_3 resulting in the 3x3 linear system:

$$\begin{bmatrix}(x_1 - x_0)\\(x_2 - x_0)\\(x_3 - x_0)\end{bmatrix}\nabla f = \begin{bmatrix}f(x_1 - x_0)\\f(x_2 - x_0)\\f(x_3 - x_0)\end{bmatrix}$$

The left hand side consists of a 3x3 matrix where each row is a displacement and the three columns are the components in each of the spatial dimensions. The right hand side is a column vector of scalar differentials. This system can be solved exactly for non-degenerate tetrahedra (i.e. tetrahedra that do not collapse into a plane, a line or a point).

3.1.1 Cell weighting

Since the cells around a given vertex are not of the same shape, the weighting factors w_i can be computed to give higher importance to those cells that should contribute more to the average gradient:

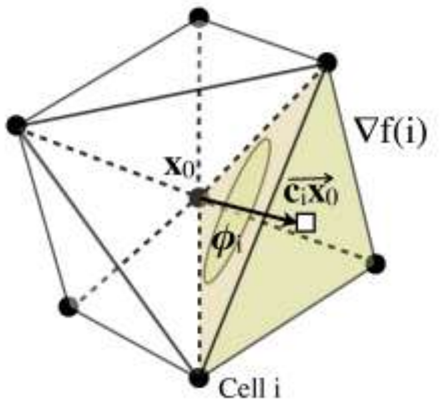


Illustration 3: Weighted average of the neighboring cell gradients. Typical weights are volume, solid angle ϕ_i or inverse centroid distance $\frac{1}{||c_i - x_0||}$

Here, we consider four methods:

- **Uniform**. This is the case when all cells are weighted uniformly. This method is the most commonly used in volume rendering due to its simplicity, but does not adapt to meshes of varying shape.
- **Volume**. Each cell is weighted according to its own volume. Although it adapts better to meshes of varying shape, some elements may exhibit small aspect ratio while having the same volume of other more regular elements. Later on, we show that this method is equivalent to obtaining the gradient using the Green Gauss theorem.
- **Solid Angle**. A cell is weighted by the solid angle subtended by the cell at the central vertex x0, measured as the surface area of a unit sphere covered by the opposite face to vertex x0.
- **Inverse centroid distance**. Each cell is weighted by the inverse of the distance between the central vertex x0 and the centroid of cell i.

A different derivation of the gradient is obtained using the Green-Gauss theorem, which states that for a volume Ω enclosed by a surface S,

$$\int_{\Omega} \nabla f d\Omega = \int_{\vartheta\Omega} f \cdot \vec{n} dS$$

where $\vec{n}$ denotes the outward pointing normal vector to the surface S, as shown:

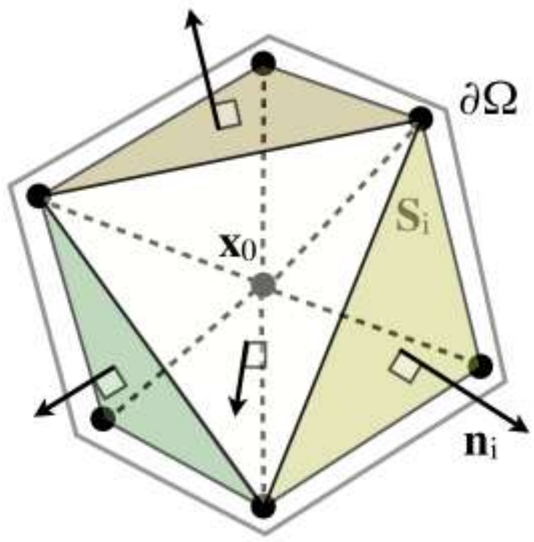


Illustration 4: The Green Gauss method approximates the gradient as the surface integral of the control volume Ω.

In an unstructured mesh, the average gradient at a node can be approximated by:

$$\nabla f(x_0) \approx \frac{1}{|\Omega|} \int_{\vartheta\Omega} f \cdot \vec{n} dS \approx \frac{1}{|\Omega|} \sum_{i \in S_1 \dots S_n} \bar{f}_i \vec{n}_i$$

The first approximation replaces the volume integral of the region enclosing the vertex by the total volume.

The second approximation is done over the surface integral using the trapezoidal rule on each of the faces Si defining the surface. $\vec{n}_i$ denotes the outward pointing normal vector of the face S_i. The scalar value at a face, f_i, is obtained as the linearly interpolated scalar value at the barycenter of faceS_i.

We can see that the Green Gauss method is equivalent to computing the volume weighted cell average gradient. Let us define the volume integral of the gradient in a neighborhood of cells around a central vertex x_0. Assuming that the gradient at the cell is constant

$$\int_{\Omega} \nabla f \, d\Omega = \sum_i \nabla f(i) \int_{\Omega} d\Omega = \sum_i \nabla f(i) \, V_i$$

where V_i is the volume of cell i. That is, if we convert each element volume integral into an element Green-Gauss surface integral, the contributions from shared internal faces will cancel out in the summation over the entire region, resulting in the Green-Gauss approximation.

3.2 Regression based methods

Another family of methods can be derived from

$$f(x_0 + h) = f(x_0) + \nabla f(x_0) \cdot h + O\left(\|h\|^2\right)$$

by fitting a **hyperplane** that best satisfies the equation for a number of sample points, as depicted:

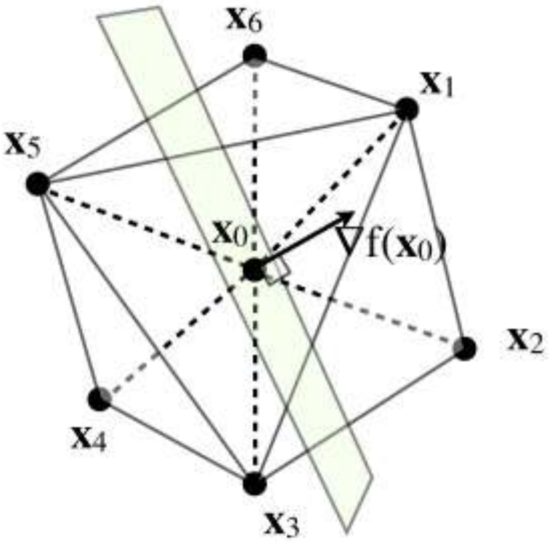


Illustration 5: Regression fits a plane for the gradient based on the contribution of the direct neighbors of vertex x_0

In the case of node-centered gradients, the equation above can be generalized to an over-constrained system of equations.

$$\begin{bmatrix} (x_1 - x_0)^T \\ \dots \\ (x_k - x_0)^T \end{bmatrix} \nabla f = \begin{bmatrix} f(x_1) - f(x_0) \\ \dots \\ f(x_k) - f(x_0) \end{bmatrix}$$

Where $x_1, x_2 \dots x_k$ are the vertex neighbors of vertex x_0. Equivalently, this system can be expressed In matrix form,

$$\bar{\bar{X}} \nabla f = b$$

Where $\bar{\bar{X}}$ is a kx3 matrix whose columns are the displacement of each vertex in the spatial dimensions, and b is a column vector of dimensions k × 1 of scalar value differentials. The problem can be solved using linear least squares.

As can be seen, this method extends naturally to arbitrary element shapes and neighborhoods. In particular, considering all the vertices in a spatial neighborhood of x_0 leads to a meshless gradient reconstruction scheme. The same cannot be said about averaging methods. As pointed out by Mavriplis, the Green Gauss approximation is generally not exact for discretizations other than tetrahedra, although different control volumes can be defined for such cases.

4. References

[1] John C. Russ (2006). *The image processing handbook*. Fifth edition. CRC Press. Taylor & Francis Group. ISBN 0-8493-7254-2 (p. 291-306)

[2] http://en.wikipedia.org/wiki/Image_gradient

[3] Tinku Acharya, Ajoy K. Ray (2005). *Image Processing: Principles and Applications*. Wiley-Interscience . ISBN 0-4717-1998-6 (p. 135-140)

[4] http://en.wikipedia.org/wiki/Jacobian

[5] http://en.wikipedia.org/wiki/File:Example_of_2D_mesh.png

[6] http://en.wikipedia.org/wiki/Finite_element_method

[7] L. Klinger, J.B. Vos, K. Appert (2001). A simplified gradient evaluation on non-orthogonal meshes; application to a plasma torch simulation method. Computers & Fluids 33 (2004) 643–654. (p. 644–647)

[8] Carlos D. Correa, Robert Hero, Kwan-Liu Ma (2009). A Comparison of Gradient Estimation Methods for Volume Rendering on Unstructured Meshes.

[9] http://www.cfd-online.com/Wiki/Mesh_adaptation