On the order of accuracy of the divergence theorem (Green-Gauss) method for calculating the gradient in finite volume methods

Alexandros Syrakos, Stylianos Varchanis, Yannis Dimakopoulos, Apostolos Goulas, John Tsamopoulos

On the order of accuracy of the divergence theorem (Green-Gauss) method for calculating the gradient in finite volume methods

Alexandros Syrakos[1], Stylianos Varchanis[1], Yannis Dimakopoulos[1], Apostolos Goulas[2], and John Tsamopoulos[1]

[1]*Laboratory of Fluid Mechanics and Rheology, Dept. of Chemical Engineering, University of Patras, 26500 Patras, Greece*

[2]*Laboratory of Fluid Mechanics and Turbomachinery, Dept. of Mechanical Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece*

Abstract

The divergence theorem (or Green-Gauss) gradient scheme is among the most popular methods for discretising the gradient operator in second-order accurate finite volume methods, with a long history of successful application on structured grids. This together with the ease of application of the scheme on unstructured grids has led to its widespread use in unstructured finite volume methods (FVMs). However, the present study shows both theoretically and through numerical tests that the common variant of this scheme is zeroth-order accurate (it does not converge to the exact gradient) on grids of arbitrary skewness, such as typically produced by unstructured grid generation algorithms. Moreover, we use the scheme in the FVM solution of a diffusion (Poisson) equation problem, with both an in-house code and the popular open-source solver OpenFOAM, and observe that the zeroth-order accuracy of the gradient operator is inherited by the FVM solver as a whole. However, a simple iterative procedure that exploits the outer iterations of the FVM solver is shown to effect first-order accuracy to the gradient and second-order accuracy to the FVM at almost no extra cost compared to the original scheme. Second-order accurate results are also obtained if a least-squares gradient operator is used instead.

Keywords: Finite volume method; Green-Gauss gradient; divergence theorem gradient; diffusion equation; OpenFOAM

Note: This paper is a shorter version of

Syrakos A., Varchanis S., Dimakopoulos Y., Goulas A., Tsamopoulos J. (2016), *"A critical analysis of some popular methods for the discretisation of the gradient operator in finite volume methods"*, arXiv:1606.05556.

The present paper deals only with the divergence theorem (Green-Gauss) gradient, whereas the full paper contains also an analysis of the least-squares gradient scheme.

1 Introduction

The approximation of the gradient operator is of fundamental importance in finite volume methods (FVMs). Even when solving simple partial differential equations (PDEs), such as a Poisson equation, if the domain geometry necessitates the use of non-orthogonal grids, the fluxes across a face separating two finite volumes cannot be expressed as a function of the values of the variables at the centres of these two volumes alone, but corrective terms must be included which typically involve the gradients of these variables as well [1]. The gradient operator is even more significant when solving problems governed by complex PDEs, such as turbulent flows modelled by the RANS and some LES methodologies [1] or non-Newtonian flows [2–4], where properties such as the viscosity may depend on these gradients. Apart from the main task of solving partial differential equations, gradient calculation may also be important in auxiliary activities such as post-processing [5].

A very popular method for calculating the gradient in FVMs is the "divergence theorem" (DT) or "Green-Gauss" method, which derives from the theorem it is named after. It has been in use for nearly three decades [1, 6–10], including in general-purpose commercial and open-source CFD codes. It has many attractive features: it is inexpensive; it is based on the same principles as the FVM discretisation of other PDE terms such as convection terms; it is known to outperform its main rival, the least-squares (LS) gradient method, in the simulation of aerodynamic boundary layer flows over curved surfaces [11, 12]; and it is not algorithmically restricted to a particular grid cell geometry but it can be applied to cells with an arbitrary number of faces. This last property is important, because in recent years the use of unstructured grids is becoming standard practice in simulations of complex engineering processes. The tessellation of the complex geometries typically associated with such processes by structured grids is an arduous and extremely time-consuming procedure for the modeller, whereas unstructured grid generation is much more automated. Therefore, discretisation schemes are sought that are easily applicable on unstructured grids of arbitrary geometry whereas early FVMs either used Cartesian grids or relied on coordinate transformations which are applicable only on smooth structured grids.

The successful use of the DT gradient scheme within second-order accurate FVMs on structured grids combined with the ease of its application on unstructured grids seem to have led to a widespread misconception that the DT scheme is also compatible with second-order accurate FVMs for unstructured grids. Hence, for example, this is the default scheme in the popular open-source CFD solver OpenFOAM. A couple of studies have shown, though, that this scheme is potentially zeroth-order accurate, depending on the grid properties. Syrakos [13] noticed that the DT gradient converged to wrong values on composite Cartesian grids with local refinement patches, in the vicinity of the interfaces between patches of different fineness. His analysis, given here in Sec. 4.4, concludes that this is due to grid skewness. Later, Sozer et al. [12] tested a simpler variant of the DT gradient that uses arithmetic averaging instead of linear interpolation across faces and proved that in the one-dimensional case this variant converges to incorrect values, if the grid is not uniform. Their numerical tests showed that the scheme is inconsistent also on two-dimensional grids of arbitrary topology. In fact, as early as in [6] it was briefly mentioned that the DT gradient fails to calculate exactly the gradients of linear functions, providing a hint to the inconsistency of the scheme. An important question is whether the inconsistency of the DT gradient inhibits convergence of the FVM to the correct solution. The results reported in [9] concerning the FVM solution of a Poisson equation suggest zeroth-order accuracy when the DT gradient is employed on skewed meshes, although the authors did not explicitly attribute this to inconsistency of the DT scheme.

The present paper aims to explore in depth the behaviour of the DT gradient scheme in relation to the grid properties, and to raise awareness of the fact that on unstructured grids it is, in general, zeroth-order accurate, i.e. it does not converge to the exact gradient operator that it approximates; most importantly, this inconsistency inhibits convergence of the FVM to the correct solution of the PDE solved with grid refinement. These conclusions are shown to be true even if the gradient calculation is enhanced by a finite number of "corrector" steps, a common technique (e.g. [9, 14]) whereby an iterative procedure uses the gradient calculated at the previous iteration to improve the accuracy. Theoretically, this procedure increases the order of accuracy only in the limit of infinite iterations. Practically, an acceptable accuracy can be achieved with a finite number of iterations, but this number is a priori unknown and increases with the grid fineness, and furthermore it makes the DT gradient scheme much more expensive than its main competitor, the LS scheme. However, since the FVM solution typically proceeds with a number of outer iterations, we exploit this fact to interweave the gradient iterations with the FVM outer iterations to obtain a first-order accurate DT gradient and an associated second-order FVM at a cost that is nearly the same as that of the uncorrected DT scheme.

In Sections 2 and 3 we present a theoretical analysis explaining why the scheme is second-order accurate on structured grids but zeroth-order accurate on unstructured grids, as well as the role of corrector steps. The theoretical findings are confirmed by numerical tests in Section 4. Then, in Section 5 we proceed to test the accuracy of an in-house FVM code and of the popular open-source FVM code OpenFOAM in solving a heat conduction problem (Poisson equation) on various kinds of unstructured grids. The results show that the FVMs are zeroth-order accurate when they employ the

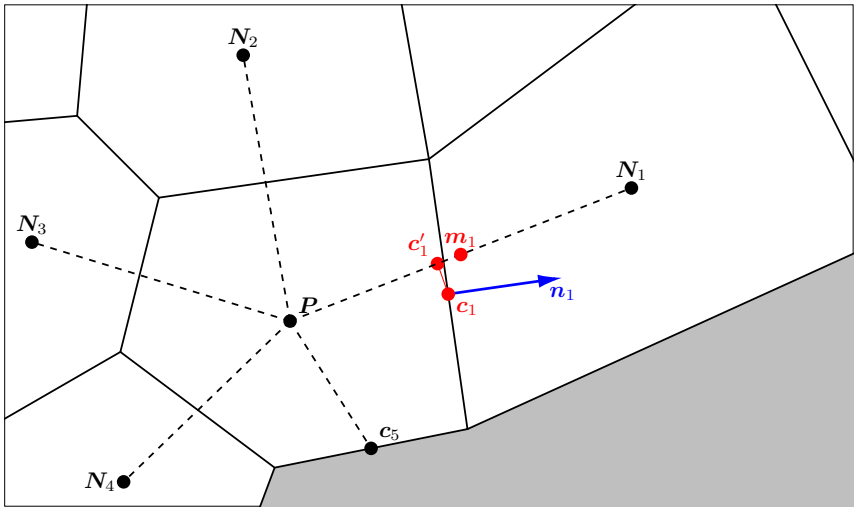


Figure 1: Part of an unstructured grid, showing cell P and its neighbouring cells, each having a single common face with P. The shaded area lies outside the grid. The faces and neighbours of cell P are numbered in anticlockwise order, with face f separating P from its neighbour N_f. The geometric characteristics of its face 1 which separates it from neighbouring cell N_1 are displayed. The position vectors of the centroids of cells P and N_f are denoted by the same characters but in boldface as $\boldsymbol{P}$ and $\boldsymbol{N}_f$; $\boldsymbol{c}_f$ is the centroid of face f and $\boldsymbol{c}'_f$ is its closest point on the line segment connecting $\boldsymbol{P}$ and $\boldsymbol{N}_f$; $\boldsymbol{m}_f$ is the midpoint between $\boldsymbol{P}$ and $\boldsymbol{N}_f$; $\boldsymbol{n}_f$ is the unit vector normal to face f, pointing outwards of P. The shown cell P also has a boundary face (face 5), with no neighbour on the other side.

DT gradient, whereas they are second-order accurate if they employ instead the proposed corrected DT gradient, or an alternative DT gradient scheme, or the least-squares (LS) gradient.

2 Preliminary considerations

We will focus on two-dimensional problems, although the one- and three-dimensional cases are also discussed where appropriate and the conclusions are roughly the same in all dimensions. Let x, y denote the usual Cartesian coordinates, and let $\phi(x, y)$ be a function defined over a domain, whose gradient $\nabla\phi$ we wish to calculate. It is convenient to introduce a convention where subscripts beginning with a dot (.) denote differentiation with respect to the ensuing variable(s), e.g. $\phi_{.x} \equiv \partial\phi/\partial x$, $\phi_{.xy} \equiv \partial^2\phi/\partial x\partial y$ etc. If the variables x, y etc. are used as subscripts without a leading dot then they are used simply as indices, without any differentiation implied. Therefore, we seek the gradient $\nabla\phi = (\phi_{.x}, \phi_{.y})$ of the function ϕ. The domain over which the function is defined is discretised by a grid into a number of non-overlapping finite volumes, or *cells*. A cell can be arbitrarily shaped, but its boundary must consist of a number of straight faces, as in Fig. 1, each separating it from a single other cell or from the exterior of the domain (the latter are called boundary faces). We assume a cell-centred finite volume method, meaning that the values of ϕ are known only at the geometric centres of the cells and at the centres of the boundary faces. The notation that is adopted in order to describe the geometry of the grid is presented in Fig. 1. Also, we will denote vectors by boldface characters.

Our goal is to derive approximate algebraic gradient operators ∇^{a} which return values $\nabla^{\mathrm{a}}\phi(\boldsymbol{P}) \approx \nabla\phi(\boldsymbol{P})$ at each cell centroid $\boldsymbol{P}$, using information only from the immediate neighbouring cells and boundary faces. The components of the approximate gradient are denoted as $\nabla^{\mathrm{a}}\phi = (\phi^{\mathrm{a}}_{.x}, \phi^{\mathrm{a}}_{.y})$. The operators ∇^{a} must be capable of approximating the derivative on grids of arbitrary geometry. As we shall see, the more irregular the grid geometry the harder it is for the operators to maintain high accuracy. Therefore, the need arises to define some indicators of the grid irregularity. With the present grid arrangement, we find it useful to define three kinds of such grid irregularities, which we shall call here "non-orthogonality", "unevenness" and "skewness" (other possibilities exist, see e.g. [15]). We will define these terms with the aid of Fig. 1. With the nomenclature defined in that figure, we will say that face f of cell P exhibits non-orthogonality if $\boldsymbol{N}_f - \boldsymbol{P}$ is not parallel to $\boldsymbol{n}_f$; a measure of non-orthogonality is the angle between the vectors $(\boldsymbol{N}_f - \boldsymbol{P})$ and $\boldsymbol{n}_f$. Also, we will say that face f

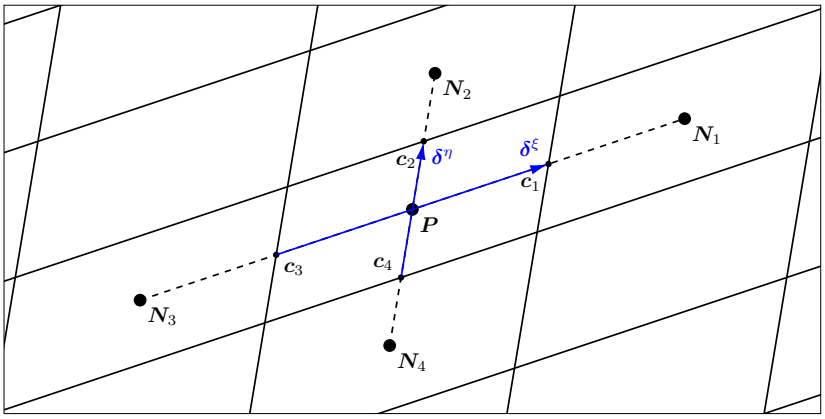


Figure 2: Part of a grid formed by equispaced parallel grid lines. See the text for details.

exhibits unevenness if the midpoint of the line segment joining $\boldsymbol{P}$ and $\boldsymbol{N}_f$, $\boldsymbol{m}_f = (\boldsymbol{P} + \boldsymbol{N}_f)/2$, does not coincide with $\boldsymbol{c}'_f$ (i.e. the cell centres are unequally spaced on either side of the face); a measure of unevenness is $\|\boldsymbol{c}'_f - \boldsymbol{m}_f\|/\|\boldsymbol{N}_f - \boldsymbol{P}\|$. Finally, we will say that face f exhibits skewness if $\boldsymbol{c}'_f$ does not coincide with $\boldsymbol{c}_f$ (i.e. the line joining the cell centres does not pass through the centre of the face); a measure of the skewness is $\|\boldsymbol{c}_f - \boldsymbol{c}'_f\|/\|\boldsymbol{N}_f - \boldsymbol{P}\|$.

Before discussing the DT gradient scheme, it is useful to examine a simpler scheme which is applicable on very plain grids that are formed from two families of equispaced parallel straight lines, intersecting at a constant angle, as in Fig. 2. In this case all the cells of the grid are identical parallelograms. Cartesian grids with constant spacing are a special case of this category where this angle is a right angle, but we will not be restricted to those. Figure 2 shows a cell P belonging to such a grid, and its four neighbouring cells. The vectors $\boldsymbol{\delta}^\xi$ and $\boldsymbol{\delta}^\eta$ are parallel to the grid lines and span the size of the cells. Due to the grid properties it holds that $\boldsymbol{\delta}^\xi = \boldsymbol{c}_1 - \boldsymbol{c}_3 = \boldsymbol{N}_1 - \boldsymbol{P} = \boldsymbol{P} - \boldsymbol{N}_3$ and $\boldsymbol{\delta}^\eta = \boldsymbol{c}_2 - \boldsymbol{c}_4 = \boldsymbol{N}_2 - \boldsymbol{P} = \boldsymbol{P} - \boldsymbol{N}_4$. It can be assumed that two variables, ξ and η, are distributed in the domain, such that in the direction of $\boldsymbol{\delta}^\xi$ the variable ξ varies linearly while η is constant, and in the direction of $\boldsymbol{\delta}^\eta$ the variable η varies linearly while ξ is constant. Then the grid can be considered to be constructed by drawing lines of constant ξ and of constant η, equispaced by $\Delta\xi$ and $\Delta\eta$, respectively. Let us assume also that the grid density can be increased by adjusting the spacings $\Delta\xi$ and $\Delta\eta$, but their ratio must be kept constant, e.g. if $\Delta\xi = h$ then $\Delta\eta = \alpha h$ with α being a constant, independent of h. The variable h determines the grid density. Therefore, the direction of the grid vectors $\boldsymbol{\delta}^\xi$ and $\boldsymbol{\delta}^\eta$ remains constant with grid refinement, but their lengths are proportional to the grid parameter h.

This idealized grid exhibits no unevenness or skewness, but it possibly exhibits non-orthogonality. However, this poses no problem as far as the gradient calculation is concerned. Since points $\boldsymbol{N}_3$, $\boldsymbol{P}$ and $\boldsymbol{N}_1$ are collinear and equidistant, the rate of change of any quantity ϕ in the direction of $\boldsymbol{\delta}^\xi$ can be approximated at point $\boldsymbol{P}$ from the values at $\boldsymbol{N}_3$ and $\boldsymbol{N}_1$ using second-order accurate central differencing. In the same manner, the rate of change in the direction of $\boldsymbol{\delta}^\eta$ at $\boldsymbol{P}$ can be approximated from the values at $\boldsymbol{N}_4$ and $\boldsymbol{N}_2$. So, let the grid vectors be written in Cartesian coordinates as $\boldsymbol{\delta}^\xi = (\delta^\xi_x, \delta^\xi_y)$ and $\boldsymbol{\delta}^\eta = (\delta^\eta_x, \delta^\eta_y)$, respectively. Then, by expanding the function ϕ in a two-dimensional Taylor series along the Cartesian directions, centred at point $\boldsymbol{P}$, and using that to express the values at the points $\boldsymbol{N}_1 = \boldsymbol{P} + \boldsymbol{\delta}^\xi$, $\boldsymbol{N}_3 = \boldsymbol{P} - \boldsymbol{\delta}^\xi$, $\boldsymbol{N}_2 = \boldsymbol{P} + \boldsymbol{\delta}^\eta$ and $\boldsymbol{N}_4 = \boldsymbol{P} - \boldsymbol{\delta}^\eta$ we get

$$\left.\begin{array}{rcl} \phi(\boldsymbol{N}_1) - \phi(\boldsymbol{N}_3) &=& 2\nabla\phi(\boldsymbol{P}) \cdot \boldsymbol{\delta}^\xi \;+\; O(h^3) \\ \phi(\boldsymbol{N}_2) - \phi(\boldsymbol{N}_4) &=& 2\nabla\phi(\boldsymbol{P}) \cdot \boldsymbol{\delta}^\eta \;+\; O(h^3) \end{array}\right\} \Rightarrow$$

$$2\begin{bmatrix} \delta^\xi_x & \delta^\xi_y \\ \delta^\eta_x & \delta^\eta_y \end{bmatrix} \cdot \begin{bmatrix} \phi_{,x}(\boldsymbol{P}) \\ \phi_{,y}(\boldsymbol{P}) \end{bmatrix} = \begin{bmatrix} \phi(\boldsymbol{N}_1) - \phi(\boldsymbol{N}_3) \\ \phi(\boldsymbol{N}_2) - \phi(\boldsymbol{N}_4) \end{bmatrix} + \begin{bmatrix} O(h^3) \\ O(h^3) \end{bmatrix}$$

which can be solved to give

$$\begin{bmatrix} \phi_{,x}(\boldsymbol{P}) \\ \phi_{,y}(\boldsymbol{P}) \end{bmatrix} = \frac{1}{2\Omega_P} \begin{bmatrix} \delta^\eta_y & -\delta^\xi_y \\ -\delta^\eta_x & \delta^\xi_x \end{bmatrix} \cdot \begin{bmatrix} \phi(\boldsymbol{N}_1) - \phi(\boldsymbol{N}_3) \\ \phi(\boldsymbol{N}_2) - \phi(\boldsymbol{N}_4) \end{bmatrix} + \frac{1}{2\Omega_P} \begin{bmatrix} \delta^\eta_y & -\delta^\xi_y \\ -\delta^\eta_x & \delta^\xi_x \end{bmatrix} \cdot \begin{bmatrix} O(h^3) \\ O(h^3) \end{bmatrix}$$

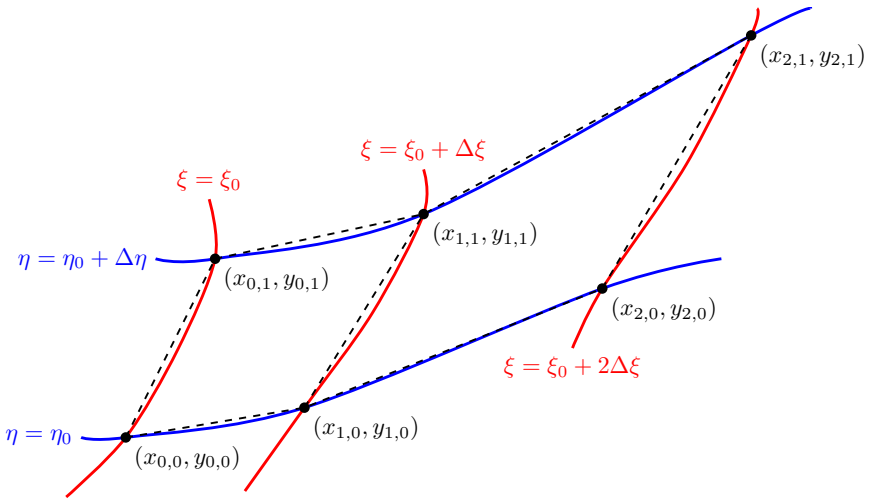


Figure 3: Part of a grid (dashed straight lines) constructed from lines of constant ξ (red curves) and η (blue curves), where ξ, η are variables distributed smoothly in the domain. The lines are equispaced with constant spacings $\Delta\xi$ and $\Delta\eta$, respectively. Grid node (i, j) is located at $(\xi_i, \eta_j) = (\xi_0 + i\,\Delta\xi, \eta_0 + j\,\Delta\eta)$ in the computational space, where (ξ_0, η_0) is a predefined point, and at $(x_{i,j}, y_{i,j})$ in the physical space.

where $\Omega_P = \delta_x^\xi \delta_y^\eta - \delta_y^\xi \delta_x^\eta = \|\boldsymbol{\delta}^\xi \times \boldsymbol{\delta}^\eta\|$ is the volume (area in 2D) of cell P. The last term in the above equation, involving the unknown $O(h^3)$ terms, is of order $O(h^2)$ because $\Omega_P = O(h^2)$ and all the δ's are $O(h)$. So, carrying out the matrix multiplications we arrive at

$$\nabla\phi(\boldsymbol{P}) \;\equiv\; \begin{bmatrix}\phi_{,x}(\boldsymbol{P})\\ \phi_{,y}(\boldsymbol{P})\end{bmatrix} \;=\; \underbrace{\frac{1}{2\Omega_P}\begin{bmatrix}\delta_y^\eta(\phi(\boldsymbol{N}_1)-\phi(\boldsymbol{N}_3)) \;-\; \delta_y^\xi(\phi(\boldsymbol{N}_2)-\phi(\boldsymbol{N}_4))\\ \delta_x^\xi(\phi(\boldsymbol{N}_2)-\phi(\boldsymbol{N}_4)) \;-\; \delta_x^\eta(\phi(\boldsymbol{N}_1)-\phi(\boldsymbol{N}_3))\end{bmatrix}}_{\nabla^{\mathrm{s}}\phi(\boldsymbol{P})} + \begin{bmatrix}O(h^2)\\ O(h^2)\end{bmatrix} \tag{1}$$

Finally, we drop the unknown $O(h^2)$ terms in Eq. (1) and we are left with a second-order approximation to the gradient, ∇^{s}; the dropped terms are called the *truncation error* of the operator ∇^{s}. The fact that this formula has been derived for a grid constructed from equidistant parallel lines may seem too restrictive, but in fact the utility of Eq. (1) goes beyond this narrow context, as will now be explained.

Consider again structured grid generation based on a pair of variables ξ, η distributed smoothly in the domain, where curves of constant ξ and η are drawn at equal intervals of $\Delta\xi$ and $\Delta\eta$, respectively. This time ξ and η are not required to vary linearly nor to be constant along straight lines. Therefore, a curvilinear grid such as that shown in Fig. 3 may result, constructed by joining the points of intersection of these two families of curves by straight line segments (the dashed lines in Fig. 3).

There is a one-to-one correspondence of coordinates (x, y) of the physical space to coordinates (ξ, η) of the computational space. Therefore, not only are the computational coordinates functions of the physical coordinates ($\xi = \xi(x, y)$ and $\eta = \eta(x, y)$), but the physical coordinates (x, y) can also be regarded as functions of the computational coordinates ($x = x(\xi, \eta)$ and $y = y(\xi, \eta)$). Since the latter vary smoothly in the domain, (x, y) can be expanded in Taylor series around a reference point (ξ_0, η_0). For example, for the x coordinate,

$$x(\xi_0 + \delta\xi, \eta_0 + \delta\eta) \;=\; x(\xi_0, \eta_0) \;+\; x_{,\xi}\delta\xi \;+\; x_{,\eta}\delta\eta \;+\; O(h^2) \tag{2}$$

where the derivatives are evaluated at point (ξ_0, η_0). In this way, we can express the coordinates of all the grid vertices $(x_{i,j}, y_{i,j})$ shown in Fig. 3 as functions of $(x_{0,0}, y_{0,0}) \equiv (x(\xi_0, \eta_0), y(\xi_0, \eta_0))$ and of the derivatives $x_{,\xi}$ etc. there. Using these expansions, it is easy to show that

$$\lim_{\Delta\xi,\Delta\eta\to 0}\left(\frac{x_{1,1}-x_{1,0}}{x_{0,1}-x_{0,0}}\right) \;=\; \lim_{\Delta\xi,\Delta\eta\to 0}\left(\frac{y_{1,1}-y_{1,0}}{y_{0,1}-y_{0,0}}\right) \;=\; 1 \tag{3}$$

and

$$\lim_{\Delta\xi,\Delta\eta\to 0}\left(\frac{x_{2,0}-x_{1,0}}{x_{1,0}-x_{0,0}}\right) \;=\; \lim_{\Delta\xi,\Delta\eta\to 0}\left(\frac{y_{2,0}-y_{1,0}}{y_{1,0}-y_{0,0}}\right) \;=\; 1 \tag{4}$$

Equation (3) implies that, as the grid is refined by reducing the $\Delta\xi$, $\Delta\eta$ spacings, neighbouring grid lines of the same family become more and more parallel, so that grid skewness tends to zero. Equation (4) implies that, as the grid is refined, neighbouring cells tend to become of equal size, so that grid unevenness tends to zero. The conclusion is that on structured grids which are constructed from smooth distributions of auxiliary variables (ξ, η), grid refinement[1] causes the geometry of a cell and its neighbours to locally approach that depicted in Fig. 2. This has an important consequence: any numerical scheme for computing the gradient that reduces to Eq. (1) on parallelogram grids (such as that of Fig. 2) is of second-order accuracy on smooth structured grids (such as that of Fig. 3)[2]. It so happens that both the DT scheme and the LS scheme belong to this category. Unfortunately, grid refinement does not engender such a quality improvement when it comes to unstructured meshes.

3 The divergence theorem gradient: theory

For grids like that of Fig. 1, which are of more general geometry than that shown in Fig. 2, a more general method is needed. Let Ω_P denote the volume of cell P and S_P its bounding surface. The DT gradient scheme is based on a derivative of the divergence theorem, which can be expressed as follows:

$$\int_{\Omega_P} \nabla\phi \, \mathrm{d}\Omega \;=\; \int_{S_P} \phi \, \boldsymbol{n} \, \mathrm{d}s$$

where $\boldsymbol{n}$ is the unit vector perpendicular to S_P at each point, pointing outwards of the cell, while $\mathrm{d}\Omega$ and $\mathrm{d}s$ are infinitesimal elements of the volume and surface, respectively. The bounding surface S_P can be decomposed into F faces which are denoted by S_f, $f = 1, \ldots, F$ ($F = 5$ in Fig. 1). These faces are assumed to be straight (planar, in 3 dimensions), as in Fig. 1, so that the normal unit vector $\boldsymbol{n}$ has a constant value $\boldsymbol{n}_f$ along each face f. Therefore, the above equation can be written as

$$\int_{\Omega_P} \nabla\phi \, \mathrm{d}\Omega \;=\; \sum_{f=1}^{F} \left(\boldsymbol{n}_f \int_{S_f} \phi \, \mathrm{d}s \right) \tag{5}$$

According to the midpoint integration rule [1,16], the mean value of a quantity over cell P (or face f) is equal to its value at the centroid $\boldsymbol{P}$ of the cell (or $\boldsymbol{c}_f$ of the face), plus a second-order correction term. Applying this to the mean values of $\nabla\phi$ and ϕ over Ω_P and S_f we get, respectively:

$$\frac{1}{\Omega_P} \int_{\Omega_P} \nabla\phi \, \mathrm{d}\Omega \;=\; \nabla\phi(\boldsymbol{P}) \;+\; \boldsymbol{O}(h^2) \;\Rightarrow\quad \int_{\Omega_P} \nabla\phi \, \mathrm{d}\Omega \;=\; \nabla\phi(\boldsymbol{P}) \, \Omega_P \;+\; \boldsymbol{O}(h^4) \tag{6}$$

$$\frac{1}{S_f} \int_{S_f} \phi \, \mathrm{d}s \;=\; \phi(\boldsymbol{c}_f) \;+\; O(h^2) \;\Rightarrow\quad \int_{S_f} \phi \, \mathrm{d}s \;=\; \phi(\boldsymbol{c}_f) \, S_f \;+\; O(h^3) \tag{7}$$

where h is a characteristic grid spacing (we used the fact that $\Omega_P = O(h^2)$ and $S_f = O(h)$). Substituting these expressions into the divergence theorem, Eq. (5), we get

$$\nabla\phi(\boldsymbol{P}) \;=\; \frac{1}{\Omega_P} \sum_{f=1}^{F} \phi(\boldsymbol{c}_f) \, S_f \, \boldsymbol{n}_f \;+\; \boldsymbol{O}(h) \tag{8}$$

The above formula is exact as long as the unknown $\boldsymbol{O}(h)$ term is retained, which consists mostly of face contributions arising from Eq. (7), whereas the volume contribution from Eq. (6) is only $\boldsymbol{O}(h^2)$. If we drop this term then we are left with a first-order accurate approximation. However, this approximation cannot be the final formula for the gradient because Eq. (8) contains $\phi(\boldsymbol{c}_f)$, the ϕ values at the face

[1] It is stressed that grid refinement must be performed in the *computational* space (ξ, η), by simultaneously reducing the spacings $\Delta\xi$ and $\Delta\eta$. Otherwise, if refinement is performed directly in the physical space (x, y), for example by joining the centroid of each cell to the centroids of its faces thus splitting it into four child cells, then equations (3) – (4) do not necessarily hold.

[2] To be precise, this depends on the skewness decreasing fast enough with grid refinement – see Section 3. For structured grids, using Taylor series such as Eq. (2), it can be shown that the skewness is $O(h)$.

centres, whereas we need a formula that uses only the values at the *cell* centres. The common practice is to approximate $\phi(\boldsymbol{c}_f)$ by $\phi(\boldsymbol{c}'_f)$, the exact values of ϕ at points $\boldsymbol{c}'_f$ rather than $\boldsymbol{c}_f$ (see Fig. 1); these values also do not belong to the set of cell-centre values, but since $\boldsymbol{c}'_f$ lies on the line segment joining cell centres $\boldsymbol{P}$ and $\boldsymbol{N}_f$, the value $\phi(\boldsymbol{c}'_f)$ can in turn be approximated by linear interpolation between $\phi(\boldsymbol{P})$ and $\phi(\boldsymbol{N}_f)$, say $\bar{\phi}(\boldsymbol{c}'_f)$ (the overbar denotes linear interpolation):

$$\bar{\phi}(\boldsymbol{c}'_f) \;\equiv\; \frac{\|\boldsymbol{c}'_f - \boldsymbol{N}_f\|}{\|\boldsymbol{N}_f - \boldsymbol{P}\|}\,\phi(\boldsymbol{P}) \;+\; \frac{\|\boldsymbol{c}'_f - \boldsymbol{P}\|}{\|\boldsymbol{N}_f - \boldsymbol{P}\|}\,\phi(\boldsymbol{N}_f) \;=\; \phi(\boldsymbol{c}'_f) \;+\; O(h^2) \tag{9}$$

Linear interpolation is known to be second-order accurate, hence the $O(h^2)$ term in the above equation. Thus, by using $\bar{\phi}(\boldsymbol{c}'_f)$ instead of $\phi(\boldsymbol{c}_f)$ in the right hand side of Eq. (8), and dropping the unknown $\boldsymbol{O}(h)$ term, we obtain an approximation to the gradient which depends only on cell-centre values of ϕ:

$$\nabla^{\mathrm{d0}}\phi(\boldsymbol{P}) \;\equiv\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\bar{\phi}(\boldsymbol{c}'_f)\,S_f\,\boldsymbol{n}_f \tag{10}$$

This is called the "divergence theorem" (DT) gradient. It applies in both two and three dimensions.

An important question is whether and how the replacement of $\phi(\boldsymbol{c}_f)$ by $\bar{\phi}(\boldsymbol{c}'_f)$, that led from Eq. (8) to the formula (10), has affected the accuracy. To answer this question we first need to deduce how much $\bar{\phi}(\boldsymbol{c}'_f)$ differs from $\phi(\boldsymbol{c}_f)$. This can be done by using the exact $\phi(\boldsymbol{c}'_f)$ as an intermediate value. We will consider structured grids and unstructured grids separately.

Structured grids

As discussed in the previous Section, the skewness of smooth structured grids diminishes with refinement, and in fact expressing the points involved in its definition as Taylor series of the form (2) it can be shown that $\|\boldsymbol{c}_f - \boldsymbol{c}'_f\|/\|\boldsymbol{N}_f - \boldsymbol{P}\| = O(h)$, or $\boldsymbol{c}_f - \boldsymbol{c}'_f = \boldsymbol{O}(h^2)$. Therefore, expanding $\phi(\boldsymbol{c}_f)$ in a Taylor series about point $\boldsymbol{c}'_f$ gives $\phi(\boldsymbol{c}_f) = \phi(\boldsymbol{c}'_f) + \nabla\phi(\boldsymbol{c}'_f)\cdot(\boldsymbol{c}_f - \boldsymbol{c}'_f) + O(h^2) = \phi(\boldsymbol{c}'_f) + O(h^2)$. Next, we note that $\phi(\boldsymbol{c}'_f)$ can be expressed in terms of $\bar{\phi}(\boldsymbol{c}'_f)$ according to Eq. (9). Putting everything together results in the sought relationship:

$$\phi(\boldsymbol{c}_f) \;=\; \phi(\boldsymbol{c}'_f) + O(h^2) \;=\; (\bar{\phi}(\boldsymbol{c}'_f) + O(h^2)) + O(h^2) \;=\; \bar{\phi}(\boldsymbol{c}'_f) + O(h^2) \tag{11}$$

Substituting this into Eq. (8) (which is an exact equation) we get

$$\nabla\phi(\boldsymbol{P}) \;=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\left(\bar{\phi}(\boldsymbol{c}'_f) + O(h^2)\right)\,S_f\,\boldsymbol{n}_f \;+\; \boldsymbol{O}(h)$$

which, considering that $\Omega_P = O(h^2)$ and $S_f = O(h)$, means that

$$\nabla\phi(\boldsymbol{P}) \;=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\bar{\phi}(\boldsymbol{c}'_f)\,S_f\,\boldsymbol{n}_f \;+\; \boldsymbol{O}(h) \tag{12}$$

The above equation is also exact. The first term on its right-hand side is just the approximate gradient ∇^{d0}, Eq. (10). Therefore, the truncation error of ∇^{d0} is $\nabla^{\mathrm{d0}}\phi(\boldsymbol{P}) - \nabla\phi(\boldsymbol{P}) = \boldsymbol{O}(h)$. Actually, the situation is even better because the leading terms of the contributions of opposite faces to the $\boldsymbol{O}(h)$ term in Eq. (12) cancel out leaving a net $\boldsymbol{O}(h^2)$ truncation error, making the method second-order accurate. The proof is tedious and involves writing analytic expressions for the truncation error contributions of each face; it can be performed more easily for a grid such as that shown in Fig. 2 where the divergence theorem procedure described in the present Section is an alternative path to arrive at the exact same formula, Eq. (1), $\nabla^{\mathrm{d0}} \equiv \nabla^{\mathrm{s}}$, with its $\boldsymbol{O}(h^2)$ truncation error.

The error cancellation between opposite faces does not always occur; sometimes the worst-case scenario of $\boldsymbol{O}(h)$ truncation error predicted by Eq. (12) holds. An example is at boundary cells. Fig. 4 shows a boundary cell P belonging to a Cartesian grid which exhibits neither skewness nor unevenness

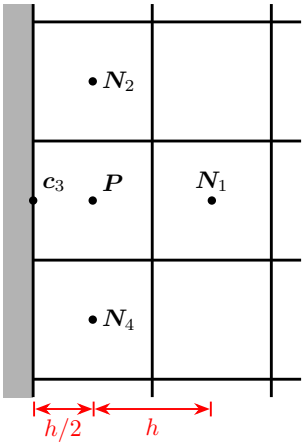


Figure 4: A boundary cell P belonging to a Cartesian grid.

(in fact, it does not even exhibit non-orthogonality). Yet cell P has no neighbour on the boundary side, and so the centre of its boundary face, $\boldsymbol{c}_3$, is used instead of a neighbouring cell centre. This introduces "unevenness" in the x- (horizontal) direction because the distances $\|\boldsymbol{N}_1 - \boldsymbol{P}\| = h$ and $\|\boldsymbol{c}_3 - \boldsymbol{P}\| = h/2$ are not equal. The x- component of the divergence theorem gradient (10) reduces to

$$\phi_{.x}^{\mathrm{d0}}(\boldsymbol{P}) \;=\; \frac{1}{2h}\left(\phi(\boldsymbol{N}_1) + \phi(\boldsymbol{P}) - 2\phi(\boldsymbol{c}_3)\right) \tag{13}$$

which is only first-order accurate, since expanding $\phi(\boldsymbol{N}_1)$ and $\phi(\boldsymbol{c}_3)$ in Taylor series about $\boldsymbol{P}$ gives

$$\phi_{.x}^{\mathrm{d0}}(\boldsymbol{P}) \;=\; \phi_{.x}(\boldsymbol{P}) \;+\; \frac{1}{8}\phi_{.xx}(\boldsymbol{P})\, h \;+\; O(h^2)$$

This offers a nice demonstration of the effect of error cancellation between opposite faces. Equation (10) is obtained from Eq. (8) by using interpolated values $\bar{\phi}(\boldsymbol{c}'_f)$ instead of the exact but unknown values $\phi(\boldsymbol{c}_f)$ at the face centres. Therefore, one might expect that since the cell of Fig. 4 has a boundary face and the exact value at its centre, $\phi(\boldsymbol{c}_3)$, is used rather than an interpolated value, the result would be more accurate; but the above analysis shows the exact opposite: the error increases from $O(h^2)$ to $O(h)$. This is due to the fact that by dropping the interpolation error on the boundary face the corresponding error on the opposite face 1 is no longer counterbalanced.

Another example where error cancellation does not occur and the formal $\boldsymbol{O}(h)$ accuracy predicted by Eq. (12) holds is the common case of grid that consists of triangles that come from dividing each cell of a smooth structured grid along the same diagonal. For example, application of this procedure to the grid shown in Fig. 2 results in the grid of Fig. 5. The latter may be seen to exhibit neither unevenness nor skewness as the face centres $\boldsymbol{c}_f$ coincide with the midpoints of the line segments joining cell centre $\boldsymbol{P}$ to its neighbours $\boldsymbol{N}_f$. Thus Eq. (11) holds, leading to Eq. (12). This time, however, a tedious but straightforward calculation where neighbouring $\phi(\boldsymbol{N}_f)$ values are expressed in Taylor series about $\boldsymbol{P}$ and substituted in (12) shows that there is no cancellation and the truncation error remains $\boldsymbol{O}(h)$. If the grid comes from triangulation of a curvilinear structured grid (Fig. 3) then the skewness is not zero but it diminishes with refinement at an $O(h)$ rate (Eq. (11)) and exactly the same conclusions hold.

In fact, from the considerations leading to Eq. (11) it follows that if a grid generation algorithm is such that skewness diminishes as $O(h^p)$ then the order of accuracy of the DT gradient is at least $\min\{p, 1\}$.

Unstructured grids

Unstructured grids usually consist of triangles (or tetrahedra in 3D), or of polygonal (polyhedral in 3D) cells which are also formed by a triangulation process. They are typically constructed using algorithms that are based on geometrical principles that do not depend on grid fineness, so that there is some self-similarity between coarse and fine grids and refinement does not reduce the skewness, i.e.

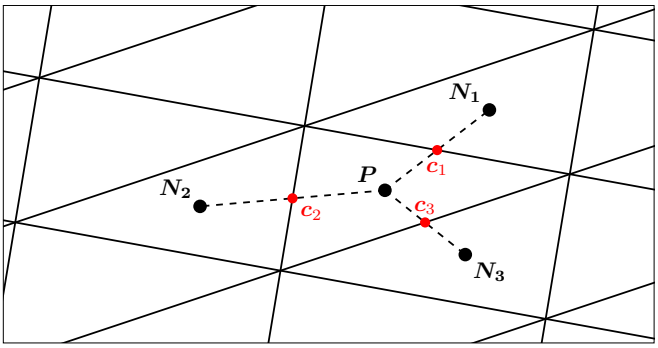


Figure 5: Grid of triangles constructed through bisection of the cells of the structured grid of Fig. 2 along their same diagonal.

$\|\boldsymbol{c}_f - \boldsymbol{c}'_f\|/\|\boldsymbol{N}_f - \boldsymbol{P}\| = O(1)$. This means that $\boldsymbol{c}_f - \boldsymbol{c}'_f = \boldsymbol{O}(h)$ and therefore instead of Eq. (11) we now have

$$\phi(\boldsymbol{c}_f) \;=\; \phi(\boldsymbol{c}'_f) + O(h) \;=\; (\bar{\phi}(\boldsymbol{c}'_f) + O(h^2)) + O(h) \;=\; \bar{\phi}(\boldsymbol{c}'_f) + O(h) \tag{14}$$

Substituting this into the exact equation (8) we get

$$\begin{aligned}\nabla\phi(\boldsymbol{P}) \;&=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\big(\bar{\phi}(\boldsymbol{c}'_f) + O(h)\big)\; S_f\,\boldsymbol{n}_f \;+\; \boldsymbol{O}(h) \\ &=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\bar{\phi}(\boldsymbol{c}'_f)\, S_f\,\boldsymbol{n}_f \;+\; \frac{1}{\Omega_P}\sum_{f=1}^{F} O(h)\, S_f\,\boldsymbol{n}_f \;+\; \boldsymbol{O}(h)\end{aligned}$$

which, considering that $\Omega_P = O(h^2)$ and $S_f = O(h)$, means that

$$\nabla\phi(\boldsymbol{P}) \;=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\bar{\phi}(\boldsymbol{c}'_f)\, S_f\,\boldsymbol{n}_f \;+\; \boldsymbol{O}(1) \;+\; \boldsymbol{O}(h) \tag{15}$$

Again, the first term on its right-hand side is the approximate gradient ∇^{d0}, Eq. (10). But this time, unlike in the structured grid case, the truncation error of ∇^{d0} is $\nabla^{\mathrm{d0}}\phi(\boldsymbol{P}) - \nabla\phi(\boldsymbol{P}) = \boldsymbol{O}(1)$. Nor do the leading terms of the truncation error contributions of the faces cancel out to increase the order of the truncation error, because on unstructured grids the orientations and sizes of the faces are unrelated. This is an unfortunate result, because it means that the approximation (10) is zeroth-order accurate, as the error $\boldsymbol{O}(1)$ does not decrease with grid refinement, but instead $\nabla^{\mathrm{d0}}\phi(\boldsymbol{P})$ converges to a value that is not equal to $\nabla\phi(\boldsymbol{P})$ (see Section 4.4 for an example).

Acknowledging that the lack of accuracy is due to the bad representation of the $\phi(\boldsymbol{c}_f)$ values in Eq. (8) by $\bar{\phi}(\boldsymbol{c}'_f)$, application of the formula (10) is often followed by a "corrector step" where, instead of the values $\bar{\phi}(\boldsymbol{c}'_f)$, the "improved" values $\hat{\phi}(\boldsymbol{c}_f)$ are used, defined as

$$\hat{\phi}(\boldsymbol{c}_f) \;\equiv\; \bar{\phi}(\boldsymbol{c}'_f) \;+\; \overline{\nabla^{\mathrm{d0}}\phi}(\boldsymbol{c}'_f)\cdot(\boldsymbol{c}_f - \boldsymbol{c}'_f) \tag{16}$$

where $\overline{\nabla^{\mathrm{d0}}\phi}(\boldsymbol{c}'_f)$ is obtained by linear interpolation (Eq. (9)) between $\nabla^{\mathrm{d0}}\phi(\boldsymbol{P})$ and $\nabla^{\mathrm{d0}}\phi(\boldsymbol{N}_f)$ at point $\boldsymbol{c}'$ (these were calculated in the previous step (10), the "predictor" step). This results in an approximation which is hopefully more accurate than (10):

$$\nabla^{\mathrm{d1}}\phi(\boldsymbol{P}) \;\equiv\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\hat{\phi}(\boldsymbol{c}_f)\, S_f\,\boldsymbol{n}_f \tag{17}$$

The values $\hat{\phi}(\boldsymbol{c}_f)$ are expected to be better approximations to $\phi(\boldsymbol{c}_f)$ than $\bar{\phi}(\boldsymbol{c}'_f)$ are, because Eq. (16) tries to account for skewness by mimicking the Taylor series expansion

$$\phi(\boldsymbol{c}_f) \;=\; \phi(\boldsymbol{c}'_f) \;+\; \nabla\phi(\boldsymbol{c}'_f)\cdot(\boldsymbol{c}_f - \boldsymbol{c}'_f) \;+\; O(h^2) \tag{18}$$

Unfortunately, since in Eq. (16) only a crude approximation of $\nabla\phi(\boldsymbol{c}'_f)$ is used, namely $\overline{\nabla^{\mathrm{d}0}\phi}(\boldsymbol{c}'_f) = \nabla\phi(\boldsymbol{c}'_f) + \boldsymbol{O}(1)$, what we get by subtracting Eq. (16) from Eq. (18) is $\hat{\phi}(\boldsymbol{c}_f) = \phi(\boldsymbol{c}_f) + O(h)$ which may have greater accuracy but not greater *order* of accuracy than the previous estimate $\bar{\phi}(\boldsymbol{c}'_f) = \phi(\boldsymbol{c}_f) + O(h)$, Eq. (14). Substituting this into Eq. (8) we arrive again at an equation similar to (15) which shows that the error of the approximation (17) is also of order $\boldsymbol{O}(1)$.

Further correction steps may be applied in the same manner; a fixed finite number of such steps may increase the accuracy, but the order of accuracy with respect to grid refinement will remain zero. But if this procedure is repeated until convergence to an operator $\nabla^{\mathrm{d}\infty}$, say, then $\nabla^{\mathrm{d}\infty}$ would simultaneously satisfy both Eqs. (16) and (17), or combined in a single equation:

$$\nabla^{\mathrm{d}\infty}\phi(\boldsymbol{P}) \;-\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\overline{\nabla^{\mathrm{d}\infty}\phi}(\boldsymbol{c}'_f)\cdot(\boldsymbol{c}_f-\boldsymbol{c}'_f)\,S_f\,\boldsymbol{n}_f \;=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\bar{\phi}(\boldsymbol{c}'_f)\,S_f\,\boldsymbol{n}_f \tag{19}$$

where we have moved all terms involving the gradient to the left-hand side. An analogous equation can be derived for the exact gradient, from Eqs. (8), (18), and (9):

$$\nabla\phi(\boldsymbol{P}) \;-\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\overline{\nabla\phi}(\boldsymbol{c}'_f)\cdot(\boldsymbol{c}_f-\boldsymbol{c}'_f)\,S_f\,\boldsymbol{n}_f \;=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\bar{\phi}(\boldsymbol{c}'_f)\,S_f\,\boldsymbol{n}_f \;+\; \boldsymbol{O}(h) \tag{20}$$

By subtracting Eq. (20) from Eq. (19) we get an expression for the truncation error $\boldsymbol{\tau} = \nabla^{\mathrm{d}\infty}\phi - \nabla\phi$:

$$\boldsymbol{\tau}(\boldsymbol{P}) \;-\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\overline{\boldsymbol{\tau}}(\boldsymbol{c}'_f)\cdot(\boldsymbol{c}_f-\boldsymbol{c}'_f)\,S_f\,\boldsymbol{n}_f \;=\; \boldsymbol{O}(h) \tag{21}$$

The left-hand side is a linear combination of not only $\boldsymbol{\tau}(\boldsymbol{P})$ but also $\boldsymbol{\tau}(\boldsymbol{N}_f)$ at all neighbour points. Since $\Omega_P = O(h^2)$, $S_f = O(h)$ and $\boldsymbol{c}_f - \boldsymbol{c}'_f = O(h)$, the coefficients of this linear combination are $O(1)$ i.e. they do not depend on the grid fineness but only on the grid geometry (skewness, unevenness etc.). Expression (21) suggests that $\boldsymbol{\tau} = \boldsymbol{O}(h)$, i.e. that $\nabla^{\mathrm{d}\infty}$ is first-order accurate, although to ascertain this one would have to assemble Eqs. (21) for all grid cells in a large linear system $A\tau = b \Rightarrow \tau = A^{-1}b$ (where τ holds the x- and y- (and z-, in 3D) components of $\boldsymbol{\tau}$ at all cell centres and $b = O(h)$), select a suitable matrix norm $\|\cdot\|$, and ensure that $\|A^{-1}\|$ remains bounded as $h \to 0$.

So, iterating Eqs. (16) and (17) until convergence one would hope that a first-order accurate gradient $\nabla^{\mathrm{d}\infty}$ would be obtained. As reported in [17], convergence of this procedure is not guaranteed and underrelaxation may be needed, leading to a large number of required iterations and great computational cost. In practice, the desired accuracy will be reached with a finite number of iterations, but this number is not known a priori and must be determined by trials; furthermore, in order to maintain the property that grid refinement improves the accuracy, the number of iterations should increase with grid refinement in order to avoid accuracy stagnation. Considering that the main rival of the DT gradient, the LS gradient, costs approximately as much as the DT gradient with a single corrector step, it can be seen that this iterative procedure is impractical and may end up consuming most of the computational time of a FVM solver. Alternatively, one could directly solve all the equations (19) simultaneously in a large linear system; this is also proposed and tested in [17], along with a second-order accurate method that additionally computes the Hessian matrix. Obviously, this approach also involves a very large computational cost.

However, we would like to propose here a way to avoid the extra cost of the iterative procedure. The FVM generally employs outer iterations to solve the problem at hand (e.g. SIMPLE iterations, as opposed to the inner iterations of the linear solvers employed). Outer iterations are necessary when the equations solved are non-linear, but may be used also when solving linear problems if some terms are treated with the deferred-correction approach. FVMs that use gradient schemes such as those presently discussed are almost always iterative, with the gradients computed using the values of the dependent variable from the previous outer iteration. The idea is then to exploit these outer iterations, dividing the "gradient iterations" among them: at each outer iteration n a single "gradient iteration"

is performed according to

$$\nabla^{\mathrm{d}n}\phi(\boldsymbol{P}) \;=\; \frac{1}{\Omega_P}\sum_{f=1}^{F}\left(\overline{\phi^{n-1}}(\boldsymbol{c}'_f) \;+\; \overline{\nabla^{\mathrm{d}(n-1)}\phi}(\boldsymbol{c}'_f)\cdot(\boldsymbol{c}_f-\boldsymbol{c}'_f)\right) S_f\,\boldsymbol{n}_f \tag{22}$$

which means that one has to store not only the solution of the previous outer iteration ϕ^{n-1} but also the gradient of that iteration $\nabla^{\mathrm{d}(n-1)}\phi$, which many codes do already. Since only one calculation is performed per outer iteration, the cost of this procedure is almost as low as that of the uncorrected DT gradient (10), but it arrives at the first-order accurate $\nabla^{\mathrm{d}\infty}$ gradient operator instead of the zeroth-order accurate $\nabla^{\mathrm{d}0}$. It is tested in Section 5.1. For explicit time-dependent FVM methods, Eq. (22) could be applied with $n-1$ denoting the previous time step; in the very first time step it may be necessary to perform several "gradient iterations" to obtain a consistent gradient to begin with.

Alternative schemes can be derived from the Gauss divergence theorem that are consistent on unstructured grids and are worth mentioning, although the present work focuses on the standard method. For example, consider the application of the divergence theorem method not to the actual cell P but to the auxiliary cell marked by dashed lines in Fig. 6. The endpoints of each face of this cell are either cell centroids or boundary face centroids, and so the value of ϕ can be computed at any point on the face to second-order accuracy, by linear interpolation between the two endpoints. Therefore, in Eq. (8) the values $\phi(\boldsymbol{c}_f)$ ($\boldsymbol{c}_f$ being the face centroids of the *auxiliary* cell, marked by empty circles in Fig. 6) are calculated to second-order accuracy rather than first-order, leading to first-order accuracy of the computed gradient. Second-order accuracy is inhibited also by the fact that the midpoint integration rule (Eq. (6)) requires that the gradient be computed at the centroid of the auxiliary cell, which now does not, in general, coincide with point $\boldsymbol{P}$ (the situation changes if $\boldsymbol{P}$ and the centroid of the auxiliary cell tend to coincide with grid refinement). This method, along with a more complex variant, is mentioned in [6]; it is also tested in Section 5.1. Yet another method would be to use cell P itself rather than an auxiliary cell, but, similarly to the previous method, to calculate the values $\phi(\boldsymbol{c}_f)$ in Eq. (8) by linear interpolation from the values at the face endpoints (vertices) rather than from the values at the cell centroids straddling the face. An extra step must therefore precede where ϕ is approximated at the cell vertices to second-order accuracy from its values at the cell centroids. This adds to the computational cost and furthermore requires of the grid data structures to contain lists relating each vertex to its surrounding cells. More information on this method and further references can be found in [10].

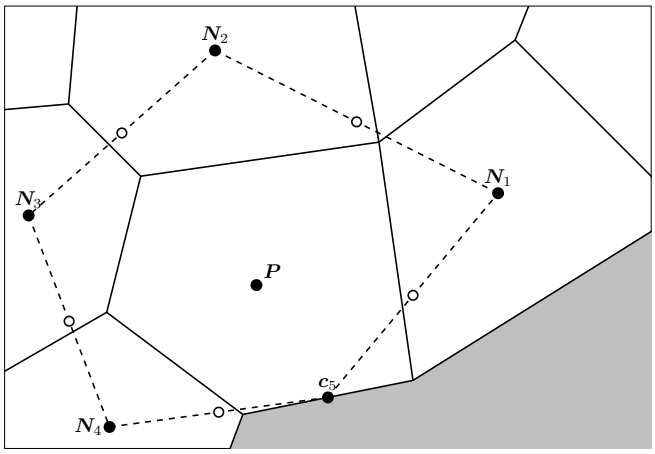


Figure 6: The divergence theorem method can approximate $\nabla\phi(\boldsymbol{P})$ to first-order accuracy on arbitrarily irregular grids if applied to the auxiliary cell bounded by dashed lines, rather than on the actual cell P. This cell is formed by joining the centroids of the neighbouring cells and of the boundary faces of cell P by straight line segments. The open circles denote the centroids of these segments.

4 Numerical tests on the accuracy of the DT gradient

In this section we test the DT gradient on various grid configurations and examine the truncation error in each case. For comparison purposes, the gradient is also computed using the least squares (LS) method, which is also very popular in the finite volume realm [10, 18–21]. Different choices of weights have been used with the LS method but in the present work the LS gradient operator ∇^{ls} is the one that at each cell centre $\boldsymbol{P}$ returns the vector $\nabla^{\text{ls}}\phi(\boldsymbol{P})$ that minimises the sum

$$\sum_f \left(\frac{\Delta\phi_f}{\Delta r_f} \; - \; \nabla^{\text{ls}}\phi(\boldsymbol{P}) \cdot \boldsymbol{d}_f \right)^2 \tag{23}$$

over all neighbours N_f of P, where $\Delta\phi_f \equiv \phi(\boldsymbol{N}_f) - \phi(\boldsymbol{P})$, $\Delta r_f \equiv \|\boldsymbol{N}_f - \boldsymbol{P}\|$, and $\boldsymbol{d}_f \equiv (\boldsymbol{N}_f - \boldsymbol{P})/\Delta r_f$ is the unit vector pointing from $\boldsymbol{P}$ to $\boldsymbol{N}_f$. That is, ∇^{ls} tries to minimise, at each cell centre, the discrepancy between the directional derivatives in the $\boldsymbol{d}_f$ directions and the finite differences $\Delta\phi_f/\Delta r_f$.

4.1 One-dimensional tests

The method is first tested on a one-dimensional problem in order to examine the effect of unevenness, isolated from skewness. So, the derivative of the single-variable function $\phi(x) = \tanh x$ is calculated at 101 equispaced points spanning the $x \in [0, 2]$ interval. The results are compared against the exact solution $\phi_{.x} = 1 - (\tanh x)^2$ and the mean absolute error $\sum_i |\phi_{.x}(x_{(i)}) - \phi^{\text{ls}}_{.x}(x_{(i)})|/101$ is recorded. In order to introduce unevenness, the two neighbours of each point x_i are not chosen from this set of equispaced points but are set at $x_{i,1} = x_i - 0.1/2^r$ and $x_{i,2} = x_i + 0.05/2^r$ where the integer r is the level of grid refinement, while x_i is regarded as the centroid of a cell extending from $x_i - 0.025/2^r$ to $x_i + 0.025/2^r$. By incrementing the level of refinement r by one the distances between x_i and its neighbours are halved, and the effect of this on the error reveals the order of accuracy of the method.

The mean error is plotted in Fig. 7 with respect to r. The slope of each curve reveals the order of accuracy of the corresponding method. It can be seen that both the DT and the LS gradients are first-order accurate (in fact, the respective curves coincide since the two formulae are identical in this case). This is in accordance to theory because in the one-dimensional case there is no skewness and linear interpolation performed to calculate face values of ϕ leads to Eq. (11) and ultimately to (12). The plot also includes results with a simpler variant of the DT method which is often used, denoted "DTa", where the values at face centres are calculated by arithmetic averaging ($\phi(\boldsymbol{c}'_f) = (\phi(\boldsymbol{P}) + \phi(\boldsymbol{N}))/2$) instead of linear interpolation (9). This is seen to be zeroth-order accurate even in the presence of unevenness alone, as noted and explained in [12].

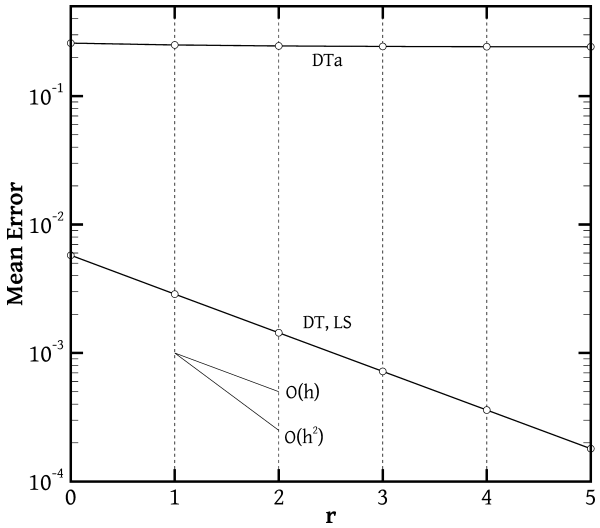


Figure 7: Mean errors of the calculation of the derivative of $\phi(x) = \tanh(x)$ with the DT and LS methods (the results coincide), and with a DT method that uses arithmetic averaging instead of linear interpolation (DTa).

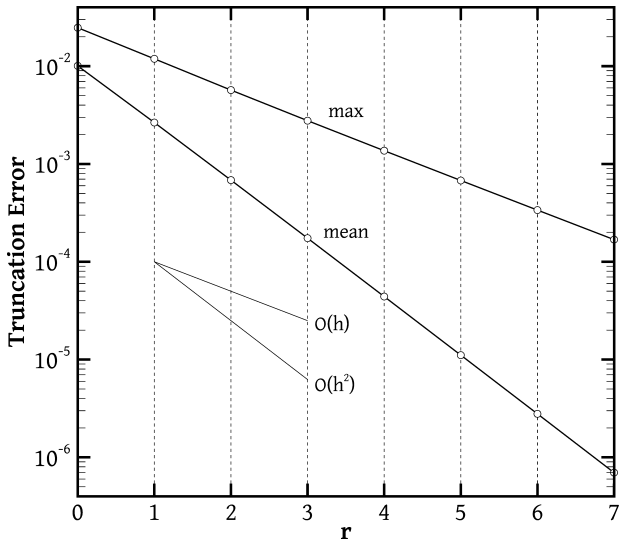


Figure 8: The mean and maximum errors (Eqs. (24) and (25)) of the DT and LS methods (their results coincide) for calculating the gradient of the function $\phi = \tanh(x)\tanh(y)$ on uniform Cartesian grids. The abscissa r designates the grid; grid r has a uniform spacing of $h = 0.25/2^r$.

4.2 Uniform Cartesian grids

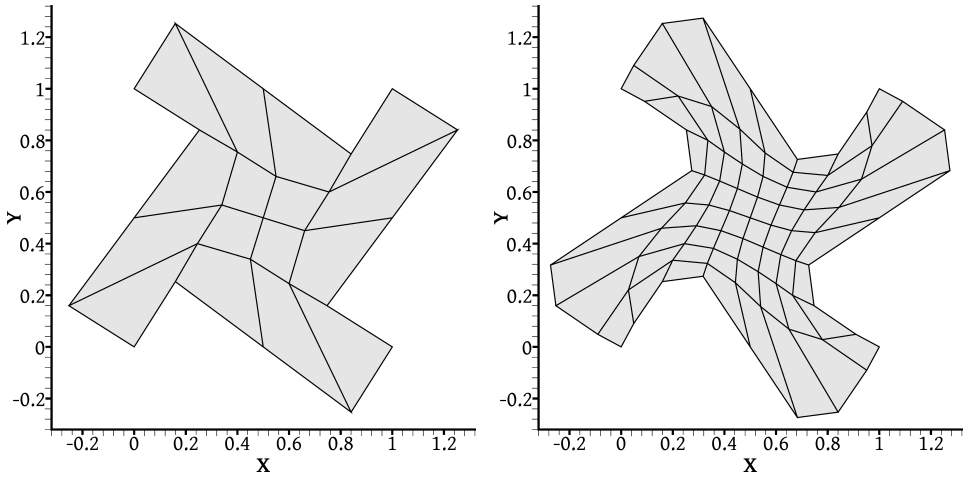
Next, the gradient of the function $\phi(x, y) = \tanh(x)\cdot\tanh(y)$ is calculated on the unit square $(x, y) \in [0, 1] \times [0, 1]$ using uniform Cartesian grids of different fineness. The exact gradient is $\phi_{.x} = (1 - (\tanh x)^2)\tanh y$ and $\phi_y = (1 - (\tanh y)^2)\tanh x$. All grid cells are geometrically identical squares of side $h = 0.25/2^r$ where r is the level of refinement; however, boundary cells are topologically different from interior cells: Whereas at the latter the gradient is calculated using the function values at neighbouring cell centres only, the former have one or more boundary faces where the function value at the face centre has to be used instead (Fig. 4). So the performance of the DT gradient must be inspected separately for these two different classes of cells. It so happens that the function tanh has zero second derivative at the boundaries $x = 0$ and $y = 0$, which may artificially increase the observed order of accuracy there, but the general behaviour of the method at boundary cells can be observed at the $x = 1$ and $y = 1$ boundaries where no such special behaviour of the tanh function applies. Since there is no skewness ($\boldsymbol{c}' = \boldsymbol{c}$ in Eq. (16)), the application of corrector steps is meaningless. The accuracy is evaluated by comparing the mean and maximum truncation errors, defined as

$$\tau_{\text{mean}} \equiv \frac{1}{M_r}\sum_{P=1}^{M_r} \|\nabla^{\text{a}}\phi(\boldsymbol{P}) - \nabla\phi(\boldsymbol{P})\| \tag{24}$$

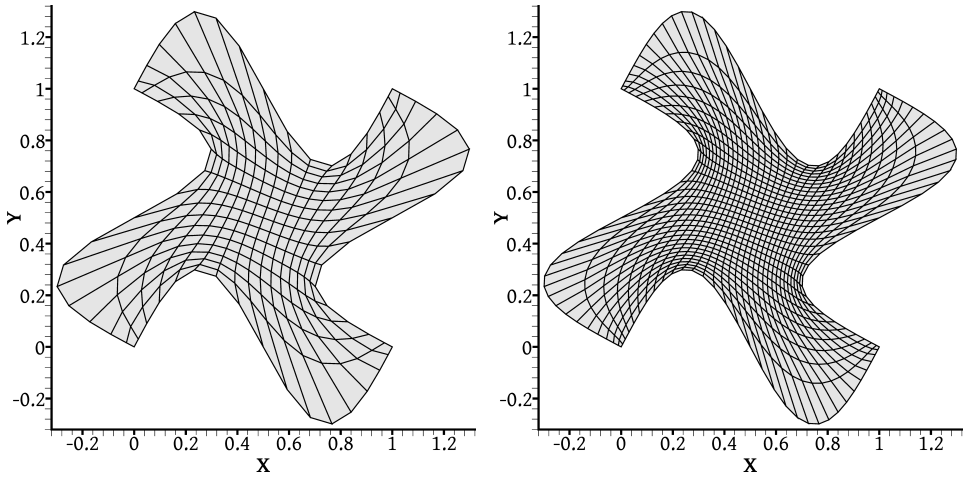
$$\tau_{\text{max}} \equiv \max_{P=1}^{M_r} \|\nabla^{\text{a}}\phi(\boldsymbol{P}) - \nabla\phi(\boldsymbol{P})\| \tag{25}$$

where $\|\cdot\|$ denotes the L_2 norm of a vector in a single cell, M_r is the number of cells of grid r, and ∇^{a} is either the DT or the LS gradient scheme. These errors are plotted in Fig. 8.

The theory predicts that both methods should be second-order accurate at interior cells, i.e. $\tau = O(h^2)$, because they both reduce to formula (1) there. At boundary cells (Fig. 4) the favourable conditions that are responsible for second-order accuracy are lost, and the methods reduce to the first-order accurate formula (13). Indeed, Fig. 8 confirms that the maximum error, which occurs at some boundary cell, is $\tau_{\text{max}} = O(h)$. The mean error is $O(h^2)$, which does not contradict with the $O(h)$ errors at the boundary cells: the number of such cells along each side of the domain equals $1/h$, so that in total there are $O(1/h)$ boundary cells, each contributing an $O(h)$ error. Their total contribution to τ_{mean} in Eq. (24) is therefore $O(1/h)\cdot O(h)/M_r = O(h^2)$ since $M_r = 1/h^2$. The interior cell contribution is $O(h^2)$ also, since $\tau = O(h^2)$ at each individual interior cell.



(a) $r = 0$ (b) $r = 1$



(c) $r = 2$ (d) $r = 3$

Figure 9: The first four of the series of grids constructed on a domain with sinusoidal boundaries via elliptic grid generation. See the text for more details.

4.3 Smooth curvilinear grids

Next, we try the methods on smooth curvilinear grids. The same function $\phi = \tanh(x) \cdot \tanh(y)$ is differentiated, but the domain boundaries now have the shapes of two horizontal and two vertical sinusoidal waves (Fig. 9). The edges of these boundaries are the points $(0, 0)$, $(1, 0)$, $(1, 1)$ and $(0, 1)$. The grid is generated using a very basic elliptic grid generation method [22]. In particular, smoothly varying functions $\xi(x, y)$ and $\eta(x, y)$ are assumed in the domain, and the two families of grid lines are lines of constant ξ and of constant η, respectively. The left, right, bottom and top boundaries correspond to $\xi = 0$, $\xi = 1$, $\eta = 0$ and $\eta = 1$, respectively. In the interior of the domain ξ and η are assumed to vary according to the following Laplace equations:

$$\begin{aligned} \xi_{.xx} \; + \; \xi_{.yy} \; &= \; 0 \\ \eta_{.xx} \; + \; \eta_{.yy} \; &= \; 0 \end{aligned}$$

These equations guarantee that ξ and η vary smoothly in the domain, but their solution $\xi = \xi(x, y)$, $\eta = \eta(x, y)$ is not much help in constructing the grid. Instead, we need the inverse functions $x = x(\xi, \eta)$, $y = y(\xi, \eta)$ which explicitly set the locations of all grid nodes; node (i, j) is located at $(x_{i,j}, y_{i,j}) \equiv (x(\xi = i\,\Delta\xi, \eta = j\,\Delta\eta), y(\xi = i\,\Delta\xi, \eta = j\,\Delta\eta))$. With $\xi, \eta \in [0, 1]$, the constant spacings $\Delta\xi$ and $\Delta\eta$ are adjusted according to the desired grid fineness. Therefore, using the chain rule of

partial differentiation it can be shown that the above equations can be expressed in inverse form as

$$\begin{aligned} g_{22}\, x_{\xi\xi} \;-\; 2g_{12}\, x_{\xi\eta} \;+\; g_{11}\, x_{\eta\eta} \;&=\; 0 \\ g_{22}\, y_{\xi\xi} \;-\; 2g_{12}\, y_{\xi\eta} \;+\; g_{11}\, y_{\eta\eta} \;&=\; 0 \end{aligned}$$

where

$$\begin{aligned} g_{11} \;&=\; x_\xi^2 + y_\xi^2 \\ g_{22} \;&=\; x_\eta^2 + y_\eta^2 \\ g_{12} \;&=\; x_\xi x_\eta + y_\xi y_\eta \end{aligned}$$

In order to cluster the points near the boundaries in the physical domain, we accompany the above equations with the following boundary conditions: at the bottom boundary we set $x = 0.5{+}0.5\sin(\pi(\xi{-}0.5))$ and $y = \sin(2\pi x)$, and at the left boundary we set $y = 0.5{+}0.5\sin(\pi(\eta{-}0.5))$ and $x = -\sin(2\pi y)$. At the top and right boundaries we set the same conditions, respectively, adding 1 to x at the right boundary and 1 to y at the top boundary. Better results can be obtained by using a more elaborate method such as described in [23], but this suffices for the present purposes.

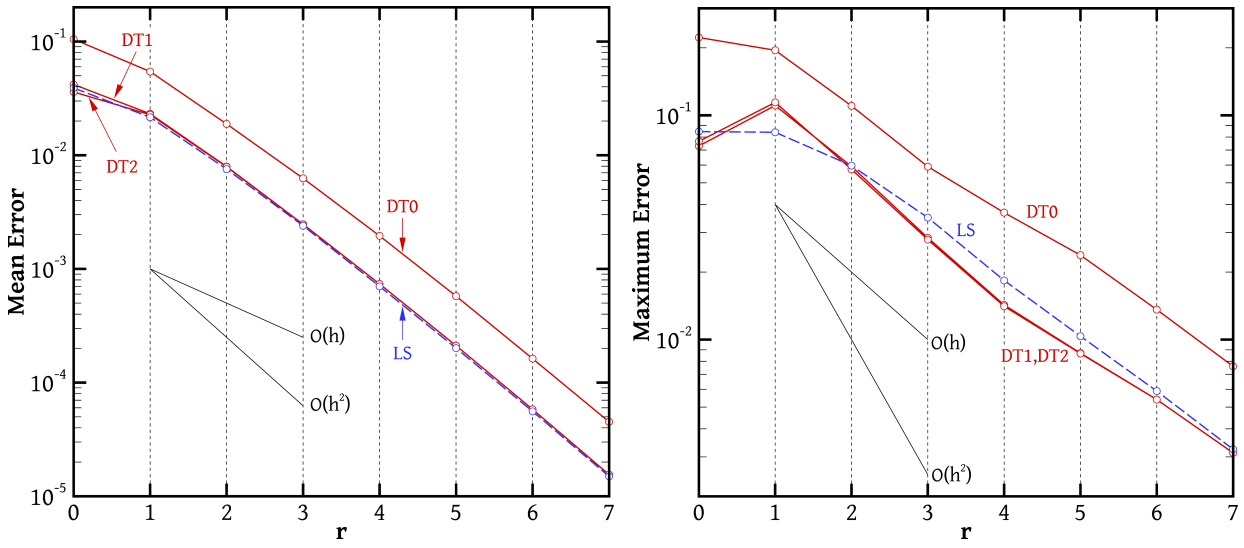
With (x, y) as the dependent variables and $(\xi, \eta) \in [0, 1] \times [0, 1]$ as the independent variables, these equations were solved numerically with a finite difference method on a 513×513 point uniform Cartesian grid. The derivatives were approximated by second-order accurate central differences; for example, at point (i, j), $x_\xi \approx (x_{i+1,j} - x_{i-1,j})/2h$, $x_\eta \approx (x_{i,j+1} - x_{i,j-1})/2h$, $x_{\xi\xi} \approx (x_{i+1,j} - 2x_{i,j} + x_{i-1,j})/h^2$ etc. where $h = 1/512$ is the grid spacing. The resulting system of nonlinear algebraic equations was solved with a Gauss-Seidel iterative method where in the equations of the (i, j) node all terms are treated as known from their current values except for $x_{i,j}$ and $y_{i,j}$ which are solved for. The convergence of the method was accelerated using a minimal polynomial extrapolation technique [24], and iterations were carried out until machine precision was reached.

After obtaining $x(\xi, \eta)$ and $y(\xi, \eta)$, a series of successively refined grids in the physical domain were constructed by drawing lines of constant ξ and lines of constant η at intervals $\Delta\xi = \Delta\eta = 0.25/2^r$, for $r = 0, 1, \ldots 7$. The first four grids are shown in Fig. 9. The gradient of the function ϕ was calculated on these grids with the DT and LS methods, and the errors are plotted in Fig. 10. The grids exhibit all kinds of grid irregularity, but the unevenness and the skewness diminish with grid refinement, as explained in Section 2. Therefore, the methods eventually behave as in the Cartesian case. Indeed, Fig. 10 shows that $\tau_{\text{mean}} = O(h^2)$ and $\tau_{\max} = O(h)$ for both methods. Therefore, like on the Cartesian grids, the methods are second-order accurate at interior cells but revert to first-order accuracy at boundary cells. For the DT method, application of a corrector step (Eq. (17)) now does make a difference, since skewness is present at any finite grid density; it reduces the error by nearly a factor of 3 and brings the accuracy on a par with the LS method. On the other hand, a second corrector step does not bring any noticeable further improvement.

4.4 Grids of localised high distortion

Structured grids that are constructed not by solving partial differential equations, as in Section 4.3, but by algebraic methods may lack the property that unevenness and skewness diminish with grid refinement. This is especially true if the domain boundaries include sharp corners at points other than grid line endpoints. For example, the grid of Fig. 11 is structured, consisting of piecewise straight lines. At the line joining the sharp corners, the intersecting grid lines change direction abruptly. This causes significant skewness which is unaffected by grid refinement.

A similar situation may occur when adaptive mesh refinement is used, depending on the treatment of the interaction between levels. Figure 12 shows multi-level grids, which consist of regions of different fineness. Such grids are often called *composite* grids [25]. One possible strategy is to treat the cells at the level interfaces as topologically polygonal [7, 26, 27]. For example, cell P of Fig. 13(a) has 6 faces, each separating it from a single other cell. Its face f_1 separates it from cell N_1 which belongs to the finer level. Faces such as f_1, which lie on grid level interfaces, exhibit non-orthogonality, unevenness, and skewness. If the grid density is increased throughout the domain, as in the series of grids shown



(a) mean error, τ_{mean} (b) maximum error, τ_{max}

Figure 10: The mean (a) and maximum (b) errors (Eqs. (24) and (25)) of the DT (solid red lines) and LS (dashed blue lines) gradient schemes applied on the function $\phi = \tanh(x)\tanh(y)$ on smooth curvilinear grids (Fig. 9). The abscissa r designates the grid; $r = 0$ is the coarsest grid (Fig. 9(a)), and grid r comes from subdividing every cell of grid $r-1$ into 4 child cells *in the computational space* (see text). For the DT method, the number of corrector steps n is indicated on each curve as DTn.

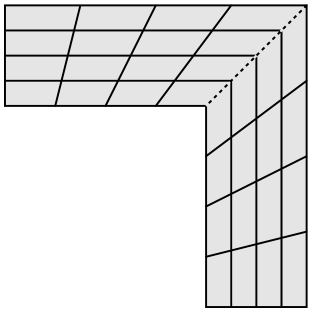


Figure 11: A structured grid where the grid lines belonging to one family change direction abruptly at the dashed line joining the pair of sharp corners, where grid skewness and not diminishing with grid refinement.

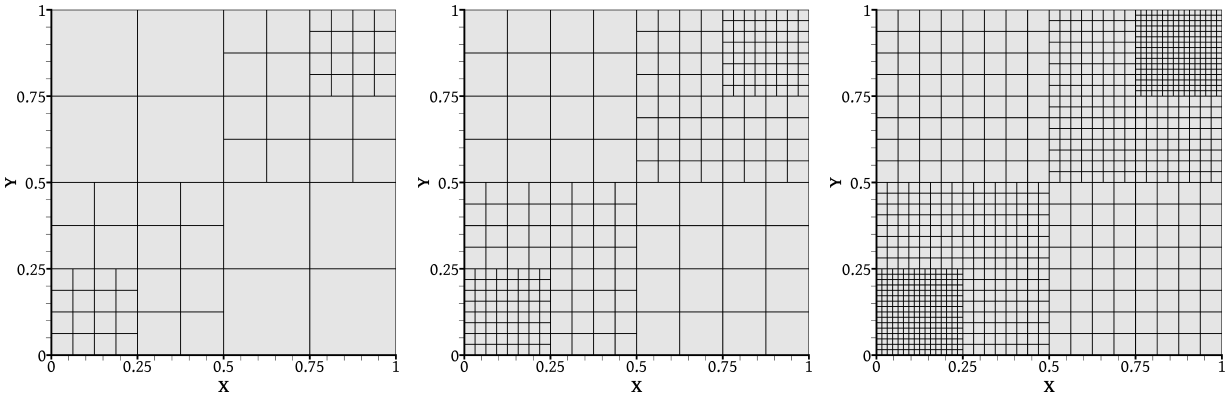
in Fig. 12, then these interface distortions remain insensitive to the grid fineness, like for the marked line in Fig. 11. Alternative schemes exist which avoid changing the topology of the cells by inserting a layer of transitional cells between the coarse and the fine part of the grid (e.g. [28, 29]) but they also lead to high, non-diminishing grid distortions at the interface.

We computed the gradient of the same function $\phi(x, y) = \tanh(x)\tanh(y)$ on a series of composite grids the first three of which are shown in Fig. 12. Figure 14 shows how τ_{mean} and τ_{max} vary with grid refinement. This time, τ_{mean} is defined a little differently than Eq. (24): to account for the different grid levels, the error of each individual cell is weighted by the cell's volume:

$$\tau_{\text{mean}} \equiv \frac{1}{\Omega}\sum_{P=1}^{M_r} \Omega_P \|\nabla^{\text{a}}\phi(\boldsymbol{P}) - \nabla\phi(\boldsymbol{P})\| \tag{26}$$

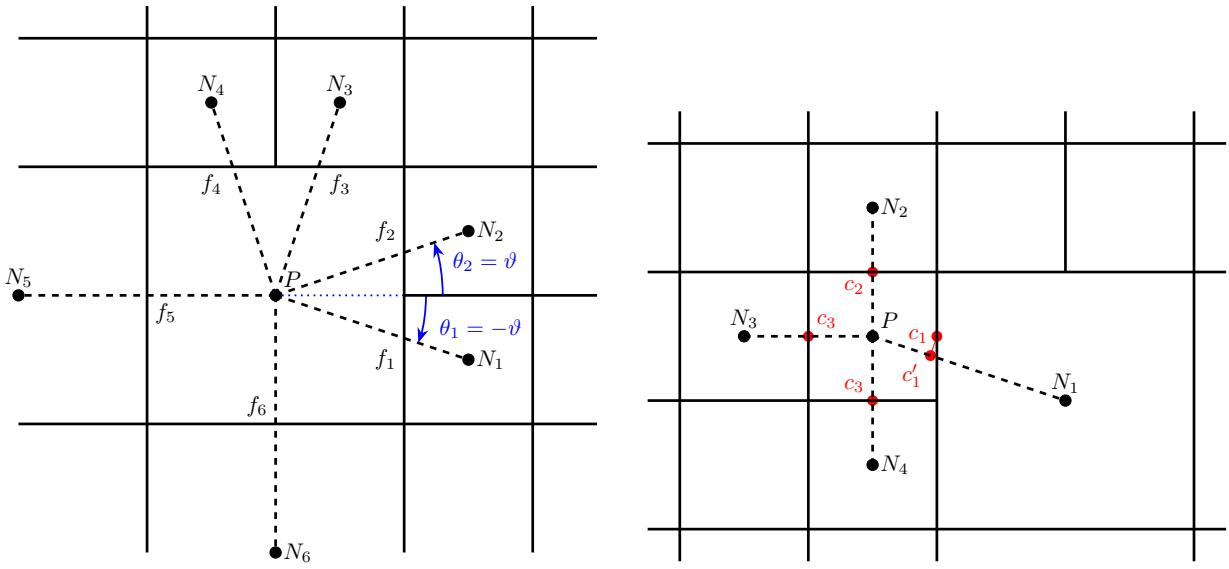
where Ω is the total volume of the domain and Ω_P is the volume of cell P.

We can identify three classes of cells that are topologically different. Apart from the familiar classes of interior and boundary cells, there is now also the class of cells that touch the level interfaces, which shall be called interface cells (these belong to two sub-classes, coarse- and fine-level cells, as in Figs. 13(a) and 13(b), respectively). Interface cells possess high skewness and unevenness that do not diminish with grid refinement. The behaviour of the gradient-calculation methods at interior and boundary cells has already been tested in Sections 4.2 and 4.3, so our interest now focuses on interface



(a) $r = 0$ (b) $r = 1$ (c) $r = 2$

Figure 12: A series of multi-level, or composite, grids. Each grid r comes from the previous grid $r - 1$ by evenly subdividing each cell into four child cells.



(a) coarse cell (b) fine cell

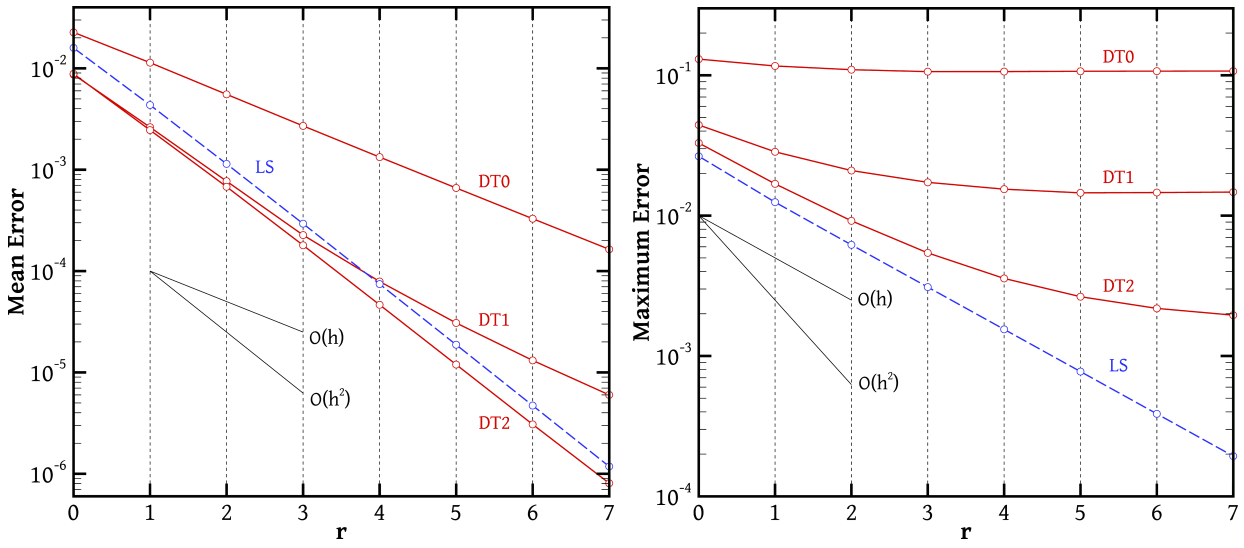
Figure 13: Topological and geometrical characteristics of a coarse cell (a) and of a fine cell (b) adjacent to a level interface, in a composite grid.

cells. Skewness, the most severe grid distortion, is encountered only at those and therefore it is there that the maximum errors (Fig. 14(b)) occur.

In Fig. 14(b) we observe that at the level interfaces the LS method converges to the correct value at a first-order rate, whereas the DT method does not converge to the correct value irrespective of the number of corrector steps performed, although each corrector step brings an accuracy improvement of nearly an order of magnitude. This behaviour is in accordance with the theory of Section 3 as non-diminishing skewness leads to Eq. (15). We can determine the operator to whom ∇^{d0} converges as follows. For the interface cell P of Fig. 13(b), formula (10) amounts to the following series of approximations:

$$\begin{aligned} \phi_{.x}(\boldsymbol{P}) \;&\approx\; \frac{\phi(\boldsymbol{c}_1) - \phi(\boldsymbol{c}_3)}{h} \;\approx\; \frac{\phi(\boldsymbol{c}_1') - \phi(\boldsymbol{c}_3)}{h} \\ &\approx\; \frac{[\alpha\,\phi(\boldsymbol{N}_1) + (1-\alpha)\,\phi(\boldsymbol{P})] \;-\; [0.5\,\phi(\boldsymbol{P}) + 0.5\,\phi(\boldsymbol{N}_3)]}{h} \;\equiv\; \phi_{.x}^{\mathrm{d0}}(\boldsymbol{P}) \end{aligned} \tag{27}$$

where h is the length of the side of cell P. In the last step, the values of ϕ at points $\boldsymbol{c}_1'$ and $\boldsymbol{c}_3$ were approximated with linear interpolation between points $\boldsymbol{N}_1$ and $\boldsymbol{P}$, and $\boldsymbol{P}$ and $\boldsymbol{N}_3$, respectively; α is an interpolation factor which equals $\alpha = 0.3$ for the present geometry. We then substitute in (27)



(a) mean error, τ_{mean}

(b) maximum error, τ_{max}

Figure 14: The mean (a) and maximum (b) errors (Eqs. (26) and (25)) of the DT (red solid lines) and LS (dashed blue lines) gradient schemes applied to the function $\phi = \tanh(x)\tanh(y)$ on locally refined grids (Fig. 12). The abscissa r designates the grid; $r = 0$ is the coarsest grid (Fig. 12(a)), and grid r comes from subdividing every cell of grid $r-1$ into 4 identical child cells. For the DT method, the number of corrector steps n is indicated on each curve as DTn.

$\phi(\boldsymbol{N}_1)$ and $\phi(\boldsymbol{N}_3)$ with their two-dimensional Taylor series about $\boldsymbol{P}$, considering that if $\boldsymbol{P} = (x_0,\, y_0)$ then $\boldsymbol{N}_1 = (x_0{+}3h/2,\, y_0{-}h/2)$ and $\boldsymbol{N}_3 = (x_0{-}h,\, y_0)$ – see Fig. 13(b). The following result is obtained:

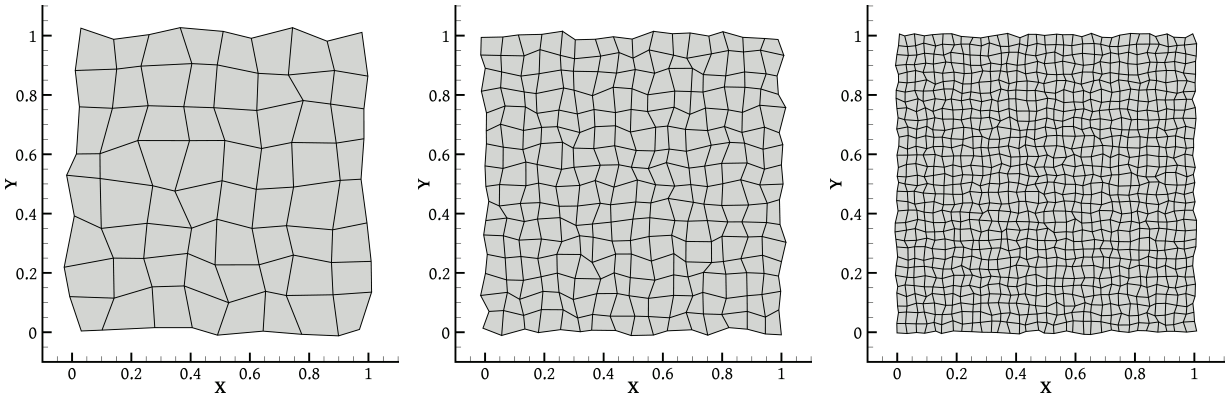
$$\phi_{.x}^{\text{d0}}(\boldsymbol{P}) \;=\; \frac{3\alpha+1}{2}\,\phi_{.x}(\boldsymbol{P}) \;-\; \frac{\alpha}{2}\,\phi_{.y}(\boldsymbol{P}) \;+\; O(h)$$

Therefore, as $h \to 0$, $\phi_{.x}^{\text{d0}}$ converges not to $\phi_{.x}$ but to an operator that involves both $\phi_{.x}$ and $\phi_{.y}$.

Next we examine the mean error in Fig. 14(a). The plot can be interpreted by considering separately the error contributions of each class of cells. The contributions of interior and boundary cells to τ_{mean} are both $O(h^2)$, as discussed in Section 4.2. At interface cells the truncation error of the DT method is $O(1)$. The total length of the level interfaces is constant, so the number of interface cells is $O(h^{-1})$ because it equals this constant length divided by the cell size which is $O(h)$. Their contribution to the mean error in Eq. (26) is (number of cells) $\times$ (volume of one cell) $\times$ (error at a cell) $= O(h^{-1}) \times O(h^2) \times O(1) = O(h)$. Figure 14(a) shows that for the DT0 method (no corrector steps) this $O(h)$ component is so large that it dominates τ_{mean} even at coarse grids. For the DT1 method (one corrector step), at coarse grids this $O(h)$ component is initially small compared to the bulk $O(h^2)$ component that comes from all the other cells, so that τ_{mean} appears to decrease at a second-order rate up to a refinement level of $r = 3$; but eventually it becomes dominant and beyond $r = 5$ the DT1 curve is parallel to the DT0 curve, with a first-order slope. With two corrector steps, the $O(h)$ component is so small that up to $r = 7$ it is completely masked by the $O(h^2)$ component and the method appears to be second-order accurate. More grid refinements are necessary to reveal its asymptotic first-order accuracy.

4.5 Grids with arbitrary distortion

As mentioned in Section 3, general-purpose unstructured grid generation methods result in unevenness and skewness that are insensitive to grid fineness throughout the domain, not just in isolated regions as for the grids of Section 4.4. Therefore, in the present Section the methods are tested under these conditions; grids of non-diminishing distortion were generated by randomly perturbing the vertices of a Cartesian grid. Using such a process we constructed a series of grids, the first three of which are shown in Fig. 15. The perturbation procedure is applied as follows: Suppose a Cartesian grid with grid spacing h. If node (i, j) has coordinates $(x_{ij},\, y_{ij})$ then the perturbation procedure moves



(a) $r = 0$ (b) $r = 1$ (c) $r = 2$

Figure 15: A series of excessively distorted grids. Grid r is constructed by random perturbation of the nodes of Cartesian grid $r + 1$ of Section 4.2.

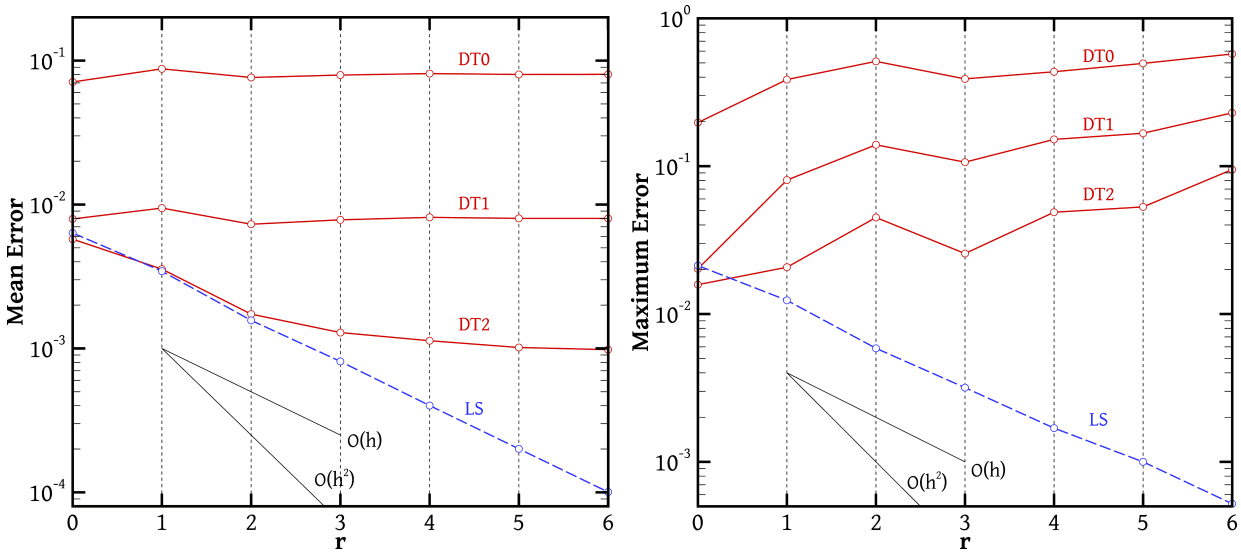
it to a location $(x'_{ij}, y'_{ij}) = (x_{ij} + \delta^x_{ij}, y_{ij} + \delta^y_{ij})$ where δ^x_{ij} and δ^y_{ij} are random numbers in the interval $[-0.25h, 0.25h)$. Because all perturbations are smaller than $h/4$ in both x and y, it is ensured that all grid cells remain simple convex quadrilaterals after all vertices have been perturbed. Grids based on triangles as well as three-dimensional cases will be considered in Section 5.2.

The gradient of the same function $\phi = \tanh(x)\tanh(y)$ is calculated, and the errors are plotted in Fig. 16. This time, all cells belong to a common category. The behaviour of the errors is in agreement with the theory of Section 3, and with Eq. (15) in particular, which predicts zeroth-order accuracy for the DT gradient. The LS method is first-order accurate with respect to both the mean and maximum errors. For the DT method, performing corrector steps improves things, but in every case the convergence eventually stagnates at some grid fineness. In fact, the maximum errors of the DT methods actually increase with grid refinement. Presumably this is due to the fact that as the number of grid nodes is increased the probability of encountering higher degrees of skewness somewhere in the domain increases. Performing corrector steps reduces the maximum error, but it is interesting to note that grid refinement causes a larger error increase when more corrector steps are performed; the deterioration of the $\nabla^{\mathrm{d}0}$ method with grid refinement propagates across the iterative correction procedure. Eventually it is expected that the errors produced by the $\nabla^{\mathrm{d}c}$ operator used as a predictor will become so large that it will provide less (rather than more) accurate face centre values to the resulting "corrected" operator $\nabla^{\mathrm{d}(c+1)}$, which will therefore be worse than $\nabla^{\mathrm{d}c}$ itself. In order to make the corrector steps convergent, underrelaxation would have to be applied, as suggested in [17].

5 Use of the gradient schemes within finite volume PDE solvers

So far we have examined the DT gradient scheme *per se*, examining its truncation error through mathematical tools and numerical experiments. Although there are examples of independent use of a gradient scheme such as in post-processing, the application of main interest is within finite volume methods (FVMs) for the solution of partial differential equations (PDEs). The gradient scheme is but a single component of the FVM and how it affects the overall accuracy depends also on the PDE solved as well as on the rest of the FVM discretisation. In the present section we provide some general comments and some simple demonstrations.

We begin with the observation that the approximation formula (10) is very similar to the formulae used by FVMs for integrating convective terms of transport equations over grid cells. Therefore, according to the same reasoning as in Section 3, such formulae also imply truncation errors of order $O(1)$ on arbitrary grids; this is true even if an interpolation scheme other than (9) is used to calculate $\bar{\phi}(\boldsymbol{c}'_f)$, or even if the exact values $\phi(\boldsymbol{c}'_f)$ are known and used. The order of the truncation error can be increased to $O(h)$ by accounting for skewness through a correction such as (16), provided that the gradient used is at least first-order accurate. These observations may raise concern about the overall accuracy of the FVM; however, it is known that the order of reduction of the discretisation error with



(a) mean error, τ_{mean}

(b) maximum error, τ_{max}

Figure 16: The mean (a) and maximum (b) errors (Eqs. (24) and (25)) of the DT (red solid lines) and LS (blue dashed lines) gradient schemes applied on the function $\phi = \tanh(x)\tanh(y)$ on the series of globally distorted grids (Fig. 15). For the DT method, the number of corrector steps n is indicated on each curve as DTn.

grid refinement is often greater than that of the truncation error. Thus, $O(1)$ *truncation* errors do not necessarily imply $O(1)$ *discretisation* errors; the latter can be of order $O(h)$ or even $O(h^2)$. This phenomenon has been observed by several authors (including the present ones [27]) and a literature review can be found in [30]. The question then naturally arises of whether and how the accuracy of the gradient scheme would affect the overall accuracy of the FVM that uses it. A general answer to this question does not yet exist, and each combination of PDE / discretisation scheme / grid type must be examined separately. For simple cases such as one-dimensional ones the problem can be tackled using theoretical tools but on general unstructured grids this has not yet been achieved [30]. Thus we will resort to some simple numerical experiments that amount to solving a Poisson equation.

5.1 Tests with an in-house solver

We solve the following Poisson equation:

$$\nabla \cdot (-k\nabla\phi) \;=\; b(x,y) \qquad \text{on } \Omega = [0,1]\times[0,1] \tag{28}$$

$$\phi \;=\; c(x,y) \qquad \text{on } S_\Omega \tag{29}$$

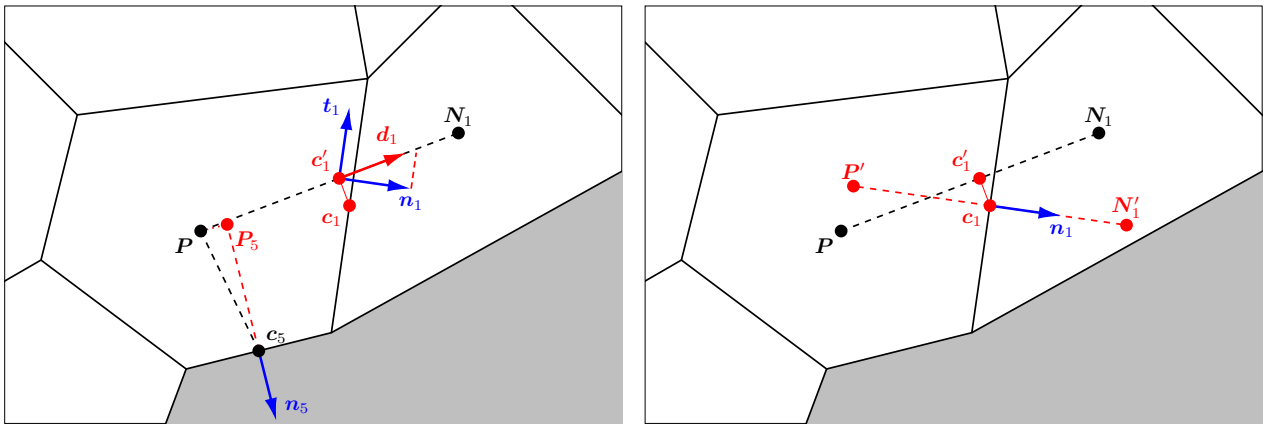
with $k = 1$, where S_Ω is the boundary of the domain Ω (the unit square), and

$$b(x,y) \;=\; 2\tanh(x)\tanh(y)\left[2 - (\tanh(x))^2 - (\tanh(y))^2\right] \tag{30}$$

$$c(x,y) \;=\; \tanh(x)\tanh(y) \tag{31}$$

This is a heat conduction equation with a heat source term and Dirichlet boundary conditions. The source term was chosen such that the exact solution of the above equation is precisely $\phi = c(x,y)$. The domain was discretised by a series of progressively finer grids of 32×32, 64×64, …, 512×512 cells denoted as grids 0, 1, … 4, respectively. The grids were generated by the same randomised distortion procedure as those of Fig. 15, only that their boundaries are straight (Fig. 19(a)). Corresponding undistorted Cartesian grids were also used for comparison. According to the FV methodology, we integrate Eq. (28) over each cell, apply the divergence theorem and use the midpoint integration rule to obtain for each cell P a discrete equation of the following form:

$$\sum_{f=1}^{F} D_f \;=\; b(\boldsymbol{P})\,\Omega_P \tag{32}$$



(a) (b)

Figure 17: (a) Adopted notation for the "over-relaxed" diffusion scheme (Eq. (34)) and for the scheme for boundary faces (Eq. (36)); (b) Notation for the custom scheme (35). Face 1 of cell P is an inner face, while its face 5 is a boundary face.

where $b(\boldsymbol{P})$ is the value of the source term at the centre of cell P and

$$D_f \;=\; \int_{S_f} -k\nabla\phi\cdot\boldsymbol{n}_f\,\mathrm{d}S \;\approx\; -S_f\,k\,\nabla\phi(\boldsymbol{c_f})\cdot\boldsymbol{n}_f \tag{33}$$

is the diffusive flux through face f of the cell. We will test here two alternative discretisations of the fluxes D_f, both of which utilise some discrete gradient operator.

The first is the "over-relaxed" scheme [7, 31] which, according to [32], is very popular and is the method of choice in commercial and public-domain codes such as FLUENT, STAR-CD, STAR-CCM+ and OpenFOAM. This scheme splits the normal unit vector $\boldsymbol{n}_f$ in (33) into two components, one in the direction $\boldsymbol{N}_f - \boldsymbol{P}$ and one tangent to the face. The unit vectors along these two directions are denoted as $\boldsymbol{d}_f$ and $\boldsymbol{t}_f$, respectively (Fig. 17(a)). Thus, if $\boldsymbol{n}_f = \alpha\boldsymbol{d}_f + \beta\boldsymbol{t}_f$ then by taking the dot product of this expression with $\boldsymbol{n}_f$ itself and noting that $\boldsymbol{t}_f$ and $\boldsymbol{n}_f$ are perpendicular ($\boldsymbol{n}_f\cdot\boldsymbol{t}_f = 0$) we obtain $\alpha = (\boldsymbol{d}_f\cdot\boldsymbol{n}_f)^{-1}$. We can therefore split $\boldsymbol{n}_f = \boldsymbol{d}_f^* + \boldsymbol{t}_f^*$ where $\boldsymbol{d}_f^* = \boldsymbol{d}_f/(\boldsymbol{d}_f\cdot\boldsymbol{n}_f)$ and $\boldsymbol{t}_f^* = \beta\boldsymbol{t}_f$ calculated most easily as $\boldsymbol{t}_f^* = \boldsymbol{n}_f - \boldsymbol{d}_f^*$. Then the flux (33) is approximated as

$$\begin{aligned}
D_f \;&\approx\; -S_f\,k\,(\nabla\phi(\boldsymbol{c}'_f)\cdot\overbrace{\boldsymbol{d}_f\frac{1}{(\boldsymbol{n}_f\cdot\boldsymbol{d}_f)}}^{\boldsymbol{d}_f^*} - S_f\,k\,\nabla\phi(\boldsymbol{c}'_f)\cdot\boldsymbol{t}_f^* \\
&\approx\; -S_f\,k\,\frac{\phi(\boldsymbol{N}_f)-\phi(\boldsymbol{P})}{\|\boldsymbol{N}_f-\boldsymbol{P}\|}\,\frac{1}{(\boldsymbol{d}_f\cdot\boldsymbol{n}_f)} \;-\; S_f\,k\,\overline{\nabla^{\mathrm{a}}\phi}(\boldsymbol{c}'_f)\cdot\boldsymbol{t}_f^* \\
&=\; -S_f\,k\,\frac{\phi(\boldsymbol{N}_f)-\phi(\boldsymbol{P})}{(\boldsymbol{N}_f-\boldsymbol{P})\cdot\boldsymbol{n}_f} \;-\; S_f\,k\,\overline{\nabla^{\mathrm{a}}\phi}(\boldsymbol{c}'_f)\cdot\boldsymbol{t}_f^*
\end{aligned} \tag{34}$$

where ∇^{a} is a discretised gradient calculated at the cell centres (either a DT gradient ∇^{d} or the LS gradient ∇^{ls}) whose role we wish to investigate, and $\overline{\nabla^{\mathrm{a}}\phi}(\boldsymbol{c}'_f)$ is its value calculated at point $\boldsymbol{c}'_f$ by linear interpolation (9). This scheme obviously deviates somewhat from the midpoint integration rule, substituting $\nabla\phi(\boldsymbol{c}'_f)\cdot\boldsymbol{n}_f$ instead of $\nabla\phi(\boldsymbol{c}_f)\cdot\boldsymbol{n}_f$ in (33), i.e. it does not account for skewness.

We also try an alternative scheme which is the standard scheme in our code [33] and is a slight modification of a scheme proposed in [1]:

$$D_f \;\approx\; -S_f\,k\,\frac{\phi(\boldsymbol{N}'_f)-\phi(\boldsymbol{P}')}{\|\boldsymbol{N}'_f-\boldsymbol{P}'\|} \tag{35}$$

where

$$\phi(\boldsymbol{P}') \;\approx\; \phi(\boldsymbol{P}) \;+\; \nabla^{\mathrm{a}}\phi(\boldsymbol{P}) \cdot (\boldsymbol{P}' - \boldsymbol{P})$$
$$\phi(\boldsymbol{N}_f') \;\approx\; \phi(\boldsymbol{N}_f) \;+\; \nabla^{\mathrm{a}}\phi(\boldsymbol{N}_f) \cdot (\boldsymbol{N}_f' - \boldsymbol{N}_f)$$

and points $\boldsymbol{P}'$ and $\boldsymbol{N}_f'$ (Fig. 17(b)) are such that the line segment joining these two points has length $\|\boldsymbol{N}_f - \boldsymbol{P}\|$, is perpendicular to face f, and its midpoint is $\boldsymbol{c}_f$. Thus this scheme tries to account for skewness. We will refer to this as the "custom" scheme.

Finally, irrespective of whether scheme (34) or (35) is used for the inner face fluxes, the boundary face fluxes are always calculated as

$$D_f \;\approx\; -S_f\, k\, \frac{\phi(\boldsymbol{c}_f) - \phi(\boldsymbol{P}_f)}{\|\boldsymbol{c}_f - \boldsymbol{P}_f\|} \tag{36}$$

where f is a boundary face, point $\boldsymbol{P}_f$ is the projection of $\boldsymbol{P}$ on the line which is perpendicular to face f and passes through $\boldsymbol{c}_f$ (see Fig. 17(a)), and

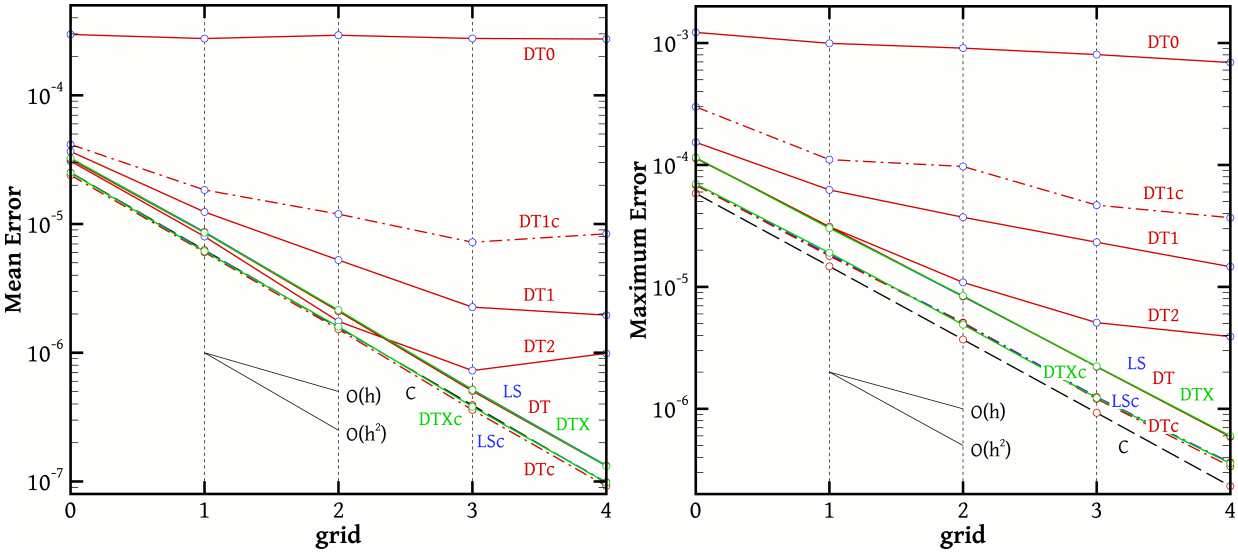
$$\phi(\boldsymbol{P}_f) \;\approx\; \phi(\boldsymbol{P}) \;+\; \nabla^{\mathrm{a}}\phi(\boldsymbol{P}) \cdot (\boldsymbol{P}_f - \boldsymbol{P})$$

Grid non-orthogonality activates in all schemes, (34), (35) and (36), a component that involves the approximate gradient ∇^{a} (it is activated in (35) also by unevenness or skewness). It is not hard to show (although the details are omitted here) that this component contributes $O(1)$ to the truncation error if ∇^{a} is first-order accurate, and $O(1/h)$ if it is zeroth-order accurate. On the undistorted Cartesian grids that were used for comparison the ∇^{a} terms of the schemes are not activated; in fact, schemes (34) and (35) reduce to the same simple formula there.

The system of discrete equations (32) is linear, but for convenience it was solved with a fixed-point iteration procedure where in each outer iteration a linear system is solved (by a few inner iterations of a preconditioned conjugate gradient solver) whose equations involve only the unknowns at a cell and at its immediate neighbours, thus avoiding extended stencils. The matrix of this linear system is assembled only from the parts of Eq. (34) or (35) that directly involve $\phi(\boldsymbol{P})$ and $\phi(\boldsymbol{N}_f)$, while the terms involving the gradients are calculated using the estimate of ϕ from the previous outer iteration and incorporated into the right-hand side of the linear system, as is customary. Outer iterations were carried out until the magnitude of the residual per unit volume had fallen below 10^{-8} in every grid cell, and the number of required such iterations for each diffusion flux scheme and gradient scheme are listed in Table 1, where the operator $\nabla^{\mathrm{d}\infty}$ is obtained with the scheme (22), while the operator ∇^{dx} is the one obtained by applying the divergence theorem to the auxiliary cell of Fig. 6. The Table shows that in most cases the over-relaxed scheme converges faster than the custom scheme, which is not surprising since the former is known for its robustness [32]. Table 1 also shows the percentage of the total calculation time consumed in calculating the gradient. It may be seen that the cost of corrector steps is quite significant, with the ∇^{d2} gradient requiring more than 22% of the total computational effort. On the other hand, the scheme (22) is quite efficient, obtaining the $\nabla^{\mathrm{d}\infty}$ gradient at a cost that is almost as low as that of ∇^{d0}; however, in the case of the over-relaxed scheme, it also requires

Table 1: Number of outer iterations for convergence (residual per unit volume below 10^{-8}) of the over-relaxed (34) and custom (35) schemes for solving the Poisson equation (28) – (31) on the 512×512 distorted grids, employing various gradient schemes. Also shown is the percentage of CPU time spent on computing the gradient.

Scheme		∇^{d0}	∇^{d1}	∇^{d2}	$\nabla^{\mathrm{d}\infty}$	∇^{dx}	∇^{ls}
Over-relaxed	iterations	901	617	601	786	573	585
	CPU %	6.8	14.9	22.1	9.7	17.2	14.3
Custom	iterations	792	874	914	752	919	919
	CPU %	7.4	16.5	23.6	10.5	17.9	15.2



(a) mean error, ϵ_{mean}

(b) maximum error, ϵ_{max}

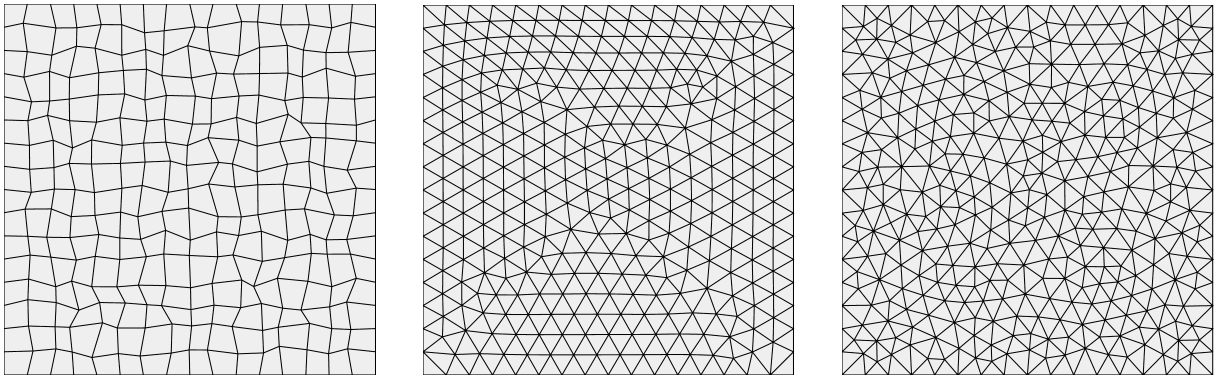
Figure 18: The mean (a) and maximum (b) discretisation errors of various FVM schemes to solve the diffusion problem (28) – (31), on a series of highly distorted quadrilateral grids (Fig. 19(a)). Grids 0, 1, ..., 4 have 32×32, 64×64, ..., 512×512 volumes, respectively. DT0, DT1, DT2, DT, DTX and LS (solid lines) denote the "over-relaxed" scheme (34) with the ∇^{d0}, ∇^{d1}, ∇^{d2}, $\nabla^{\text{d}\infty}$, ∇^{dx} and ∇^{ls} gradient schemes, respectively. DT1c, DTc, DTXc and LSc (dash-dot lines) denote the "custom" scheme (35) with the respective gradient schemes. C denotes results on Cartesian grids, where all methods are identical.

somewhat more outer iterations for convergence. The LS gradient costs about the same as the ∇^{d1} gradient. The cost of the ∇^{dx} gradient appears somewhat inflated due to a quick but not very efficient implementation, but can be reduced by more careful programming.

The discretisation errors with respect to grid refinement are plotted in Fig. 18 for various flux and gradient scheme combinations. The discretisation error is defined as the difference between the exact solution ϕ of the problem (28) – (29) and the numerical (FVM) solution; Fig. 18(a) plots the mean absolute value of the discretisation error over all grid cells, and Fig. 18(b) plots the maximum absolute value.

The first thing that one can notice from Figure 18 is that the common DT gradient (∇^{d0}, ∇^{d1}, ∇^{d2}) leads to zeroth-order accuracy with respect to both the mean and maximum discretisation errors, irrespective of whether the over-relaxed or custom flux scheme is used. Similar trends as those of Fig. 16 are observed: corrector steps reduce the error but are eventually unable to converge to the exact solution. Although on coarse grids the DT1 and DT2 lines give the impression that they converge, eventually grid refinement cannot reduce the error below a certain point. In fact, an increase of the mean error can be observed on grid 4. Things are worse when the DT gradient is used in combination with the custom scheme (line DT1c), presumably because the gradient ∇^{a} plays a more significant role in that scheme than in the over-relaxed scheme.

On the other hand, the $\nabla^{\text{d}\infty}$ gradient obtained through the iterative procedure (22), the "auxiliary cell" gradient ∇^{dx}, and the LS gradient ∇^{ls}, all lead to second-order convergence to the exact solution. As long as the gradient scheme is consistent, the FVM accuracy seems to depend not on the gradient scheme but on the flux scheme, i.e. lines DT, LS and DTX (over-relaxed scheme (34)) nearly coincide, as do lines DTc, LSc and DTXc (custom scheme (35)). The mean discretisation error of the custom scheme is about the same as that obtained on Cartesian grids (line C), or even marginally lower when used in combination with the $\nabla^{\text{d}\infty}$ gradient! The mean discretisation error of the over-relaxed scheme is only about 40% higher, which is a very good performance given that it does not account for skewness and that its iterative convergence is better (Table 1).



(a) (b) (c)

Figure 19: Samples from each grid category employed to solve the 2D Poisson problems using OpenFOAM: (a) a grid of distorted quadrilaterals; (b) a grid constructed with the Netgen algorithm; (c) a grid constructed with the Gmsh algorithm. All domains have unit length along each dimension.

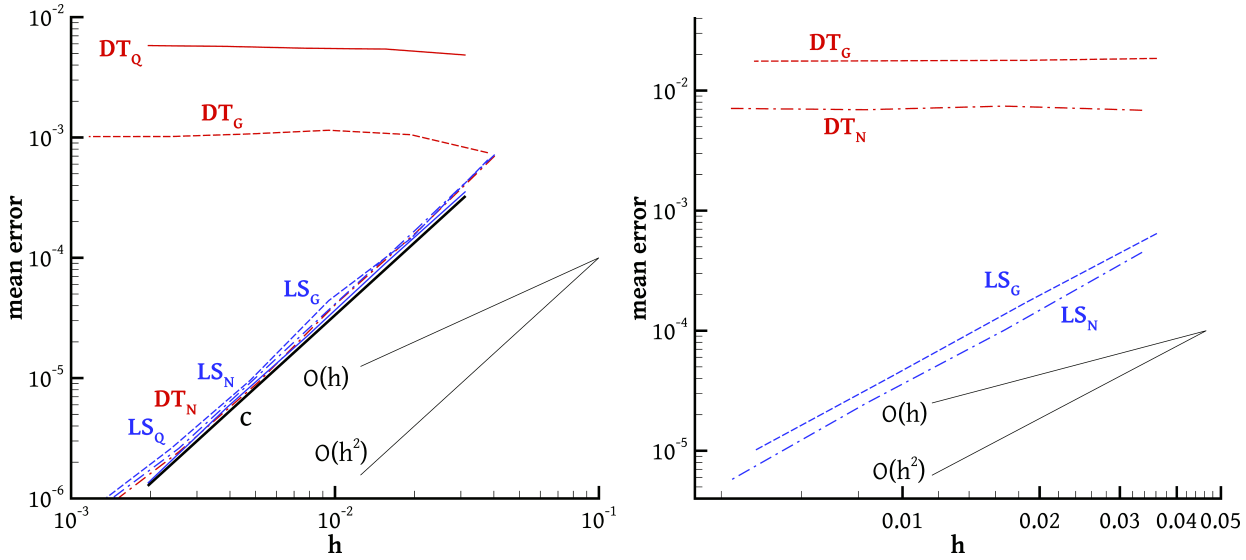
5.2 Tests with OpenFOAM

In order to investigate further the severity of the problem associated with the use of the common DT gradient we performed also a set of experiments using the public domain solver OpenFOAM (`https://openfoam.org`), version 4.0. This is an increasingly popular open-source finite volume solver that can handle a large variety of different flow types and flow phenomena (e.g. [34–36]). We solved again a Poisson problem (28) – (29) but instead of (31) and (30) we have $c(x, y) = \sin(\pi x)\sin(\pi y)$ and $b(x, y) = 2\pi^2 c(x, y)$, respectively. The problem was solved on the same set of grids composed of distorted quadrilaterals (Fig. 19(a)), but, as these are artificially distorted grids, we also repeated the calculations using a couple of popular grid generation procedures which are more typical of real-life engineering applications, namely the Netgen [37] (`www.hpfem.jku.at/netgen`) and Gmsh Delaunay [38] (`http://gmsh.info`) algorithms implemented as plugins in the SALOME preprocessor (`www.salome-platform.org`). Both generate grids of triangular cells, coarse samples of which are shown in Fig. 19. The Netgen algorithm can be seen to be effective in producing smooth grid, reminiscent of that shown in Fig. 5, in four distinct areas within the domain that meet at an "X"-shaped interface where grid distortion is high. The Gmsh grids are less regular. The "LaplacianFoam" component of OpenFOAM was used to solve the equations with default options, which include the DT gradient (10) as the chosen gradient scheme (option "gradSchemes" is set to "Gauss linear"). We repeated the calculations with the gradient option set to LS (gradSchemes: leastSquares).

The discretisation errors are plotted in Fig. 20(a), against the mean cell size[3] h. We note that on the distorted quadrilateral grids and on the Gmsh grids the DT gradient (lines $\mathrm{DT_Q}$ and $\mathrm{DT_G}$) engenders zeroth-order accuracy, like the DT0 case of Fig. 18. On the other hand, the Netgen grids are quite smooth (Fig. 19(b)), resembling grids that come from triangulation of structured grids in most of the domain, and this has the consequence that even with the DT gradient (lines $\mathrm{DT_N}$) the mean error decreases at a second-order rate. With the LS gradient second-order accuracy is exhibited on all grids, and furthermore the accuracy is nearly at the same level as on the Cartesian grids.

Three-dimensional simulations were also performed. The governing equations are again (28) – (29) where we set $c(x, y, z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$ and $b(x, y, z) = 2\pi^2 c(x, y, z)$. The domain is the unit cube $z, y, z \in [0, 1]$ and the grids were generated with the Netgen and Gmsh algorithms, where in the latter we chose the Delaunay method to construct the grid at the boundaries and the frontal Delaunay algorithm to construct the grid in the interior. The discretisation errors are plotted in Fig. 20(b). This time, it may be seen that the DT gradient engenders zeroth-order accuracy even on the Netgen grids. Presumably, in the 3D case although the Netgen algorithm produces relatively smooth surface grids on the bounding surfaces of the cube, in the bulk of the domain it packs the tetrahedra is a way that the skewness is large throughout. On the other hand, the LS gradient again always results in

[3]Unlike in previous plots we use the mean cell size instead of the grid level in Fig. 20 because in the case of the Netgen and Gmsh grids each level does not have precisely 4 (in 2D) or 8 (in 3D) times as many cells as the previous level.



(a) mean error, 2D problem

(b) mean error, 3D problem

Figure 20: The mean errors of the OpenFOAM solutions of the 2D (a) and 3D (b) Poisson equations, for various configurations. The abscissa h is the mean cell size, $h = (\Omega/M)^{1/d}$ where $\Omega = 1$ is the domain volume, M is the total number of grid cells, and $d = 2$ (for 2D) or 3 (for 3D). The black solid line (C) corresponds to the 2D problem solved on undistorted Cartesian grids. Red lines (DT) and blue lines (LS) correspond to results obtained with the DT and LS gradients, respectively. Subscript Q ($\mathrm{DT_Q}$, $\mathrm{LS_Q}$ – solid lines) corresponds to grids of distorted quadrilaterals (Fig. 19(a)). Subscript N ($\mathrm{DT_N}$, $\mathrm{LS_N}$ – dash-dot lines) corresponds to Netgen grids (Fig. 19(b)). Subscript G ($\mathrm{DT_G}$, $\mathrm{LS_G}$ – dashed lines) corresponds to Gmsh Delaunay grids (Fig. 19(c)).

second-order accuracy.

6 Conclusions

The preceding analysis of Sections 2 and 3 has shown that the common DT gradient scheme is second-order accurate on grids whose skewness diminishes with refinement, such as structured grids, and zeroth-order accurate on grids whose skewness does not diminish, such as unstructured grids in most cases. This was confirmed by the numerical tests of Section 4. Most importantly, the additional numerical tests of Section 5 have shown that a finite volume method that employs this scheme inherits its zeroth-order accuracy on grids with non-diminishing skewness. Unfortunately, general-purpose unstructured grid generation algorithms, such as the popular Netgen and Gmsh algorithms tested in Sec. 5, retain high skewness at all levels of refinement (except in the 2D Netgen case) resulting in zeroth-order accuracy for both the DT gradient and the FVM solver that employs it. In our tests, OpenFOAM using the DT gradient was unable to reduce the discretisation errors by grid refinement even though the problems solved were very fundamental, namely linear Poisson problems with sinusoidal solutions on the simplest possible domains, the unit square and cube. Zeroth-order accuracy is very undesirable because it deprives the modeller of his/her main tool for estimating the accuracy of the solution, i.e. the grid convergence study. Therefore the common DT gradient should not be used unless the grid refinement algorithm is known to reduce the skewness towards zero. On the other hand, first-order accuracy of the DT gradient and second-order accuracy of the FVM can be retrieved on unstructured grids by the iterative procedure (22), or by using an alternative consistent DT gradient scheme such as those mentioned at the end of Section 3, or by using the least-squares gradient.

Acknowledgements

This research has been funded by the LIMMAT Foundation under the Project “MuSiComPS”.

References

[1] J. H. Ferziger and M. Peric, *Computational methods for fluid dynamics*. Springer, 3rd ed., 2002.

[2] P. J. Oliveira, F. T. Pinho, and G. A. Pinto, "Numerical simulation of non-linear elastic flows with a general collocated finite-volume method," *J. Non-Newtonian Fluid Mech.*, vol. 79, pp. 1–43, 1998.

[3] A. M. Afonso, M. S. N. Oliveira, P. J. Oliveira, M. A. Alves, and F. T. Pinho, "The finite volume method in computational rheology," in *Finite-Volume Methods – Powerful Means of Engineering Design*, ch. 7, pp. 141–170, In-Tech Open Publishers, 2012.

[4] A. Syrakos, G. Georgiou, and A. Alexandrou, "Solution of the square lid-driven cavity flow of a Bingham plastic using the finite volume method," *J. Non-Newtonian Fluid Mech.*, vol. 195, pp. 19–31, 2013.

[5] C. D. Correa, R. Hero, and K.-L. Ma, "A comparison of gradient estimation methods for volume rendering on unstructured meshes," *IEEE Trans. Visual Comput. Graphics*, vol. 17, pp. 305–319, 2011.

[6] T. J. Barth and D. C. Jespersen, "The design and application of upwind schemes on unstructured meshes," in *AIAA Paper 89-0366*, 1989.

[7] H. Jasak, *Error Analysis and Estimation for the Finite Volume Method with Application to Fluid Flows*. PhD thesis, Imperial College, London, 1996.

[8] Ž. Lilek, S. Muzaferija, M. Perić, and V. Seidl, "An implicit finite-volume method using non-matching blocks of structured grid," *Numer. Heat Transfer*, vol. 32, pp. 385–401, 1997.

[9] J. Wu and P. Traoré, "Similarity and comparison of three finite-volume methods for diffusive fluxes computation on nonorthogonal meshes," *Numer. Heat Transfer B*, vol. 64, pp. 118–146, 2014.

[10] F. Moukalled, L. Mangani, and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics*. Springer, 2016.

[11] D. J. Mavriplis, "Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes," in *AIAA Paper 2003-3986*, 2003.

[12] E. Sozer, C. Brehm, and C. C. Kiris, "Gradient calculation methods on arbitrary polyhedral unstructured meshes for cell-centered CFD solvers," in *AIAA Paper 2014-1440*, 2014.

[13] A. Syrakos, *Analysis of a finite volume method for the incompressible Navier-Stokes equations*. PhD thesis, Aristotle University of Thessaloniki, 2006.

[14] M. S. Karimian and A. G. Straatman, "Discretization and parallel performance of an unstructured finite volume Navier–Stokes solver," *Int. J. Numer. Methods Fluids*, vol. 52, pp. 591–615, 2006.

[15] Y. Kallinderis and C. Kontzialis, "A priori mesh quality estimation via direct relation between truncation error and mesh distortion," *J. Comput. Phys.*, vol. 228, pp. 881–902, 2009.

[16] R. L. Burden and J. D. Faires, *Numerical Analysis*. Brooks/Cole, 9th ed., 2011.

[17] L. J. Betchen and A. G. Straatman, "An accurate gradient and Hessian reconstruction method for cell-centered finite volume discretizations on general unstructured grids," *Int. J. Numer. Methods Fluids*, vol. 62, pp. 945–962, 2010.

[18] T. J. Barth, "A 3-D upwind Euler solver for unstructured meshes," in *AIAA Paper 91-1548-CP*, 1991.

[19] S. Muzaferija and D. A. Gosman, "Finite-volume CFD procedure and adaptive error control strategy for grids of arbitrary topology," *J. Comput. Phys.*, vol. 138, pp. 766–787, 1997.

[20] C. Ollivier-Gooch and M. Van Altena, "A high-order-accurate unstructured mesh finite-volume scheme for the advection–diffusion equation," *J. Comput. Phys.*, vol. 181, pp. 729–752, 2002.

[21] F. Bramkamp, P. Lamby, and S. Müller, "An adaptive multiscale finite volume solver for unsteady and steady state flow computations," *J. Comput. Phys.*, vol. 197, pp. 460–490, 2004.

[22] J. F. Thompson, Z. U. Warsi, and C. W. Mastin, *Numerical grid generation: foundations and applications*. North-holland Amsterdam, 1985.

[23] Y. Dimakopoulos and J. Tsamopoulos, "A quasi-elliptic transformation for moving boundary problems with large anisotropic deformations," *J. Comput. Phys.*, vol. 192, pp. 494–522, 2003.

[24] A. Sidi, "Review of two vector extrapolation methods of polynomial type with applications to large-scale problems," *Journal of Computational Science*, vol. 3, pp. 92–101, 2012.

[25] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*. Academic Press, 2001.

[26] S. Muzaferija, *Adaptive Finite Volume Method for Flow Prediction using Unstructured Meshes and Multigrid Approach*. PhD thesis, Imperial College, London, 1994.

[27] A. Syrakos, G. Efthimiou, J. G. Bartzis, and A. Goulas, "Numerical experiments on the efficiency of local grid refinement based on truncation error estimates," *J. Comput. Phys.*, vol. 231, pp. 6725–6753, 2012.

[28] R. Schneiders, "Refining quadrilateral and hexahedral element meshes," in *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Fluid Simulations*, pp. 679–688, Mississippi State University, 1996.

[29] N. Chatzidai, A. Giannousakis, Y. Dimakopoulos, and J. Tsamopoulos, "On the elliptic mesh generation in domains containing multiple inclusions and undergoing large deformations," *J. Comput. Phys.*, vol. 228, pp. 1980–2011, 2009.

[30] B. Diskin and J. L. Thomas, "Notes on accuracy of finite-volume discretization schemes on irregular grids," *Appl. Numer. Math.*, vol. 60, pp. 224–226, 2010.

[31] P. Traoré, Y. M. Ahipo, and C. Louste, "A robust and efficient finite volume scheme for the discretization of diffusive flux on extremely skewed meshes in complex geometries," *J. Comput. Phys.*, vol. 228, pp. 5148–5159, 2009.

[32] I. Demirdžić, "On the discretization of the diffusion term in finite-volume continuum mechanics," *Numer. Heat Transfer B*, vol. 68, pp. 1–10, 2015.

[33] A. Syrakos and A. Goulas, "Estimate of the truncation error of finite volume discretization of the Navier-Stokes equations on colocated grids," *Int. J. Numer. Methods Fluids*, vol. 50, pp. 103–130, 2006.

[34] E. Robertson, V. Choudhury, S. Bhushan, and D. Walters, "Validation of OpenFOAM numerical methods and turbulence models for incompressible bluff body flows," *Computers & Fluids*, vol. 123, pp. 122–145, 2015.

[35] N. K. Lampropoulos, Y. Dimakopoulos, and J. Tsamopoulos, "Transient flow of gravity-driven viscous films over substrates with rectangular topographical features," *Microfluid. Nanofluid.*, vol. 20, p. 51, 2016.

[36] G. Karapetsas, N. K. Lampropoulos, Y. Dimakopoulos, and J. Tsamopoulos, "Transient flow of gravity-driven viscous films over 3D patterned substrates: conditions leading to Wenzel, Cassie and intermediate states," *Microfluid. Nanofluid.*, vol. 21, p. 17, 2017.

[37] J. Schöberl, "NETGEN an advancing front 2D/3D-mesh generator based on abstract rules," *Computing and Visualization in Science*, vol. 1, pp. 41–52, 1997.

[38] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *Int. J. Numer. Methods Eng.*, vol. 79, pp. 1309–1331, 2009.