Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona,Spain, July 9-13, 2018

ICCFD10-2018-0351

Algebraic implementation of a flux limiter for heterogeneous computing

N.Valle[1*], X.Álvarez[1], F.X.Trias[1], J.Castro[1] and A.Oliva[1]
*Corresponding author: nico@cttc.upc.edu

[1] Heat and Mass Transfer Technological Center (CTTC)
Universitat Politècnica de Catalunya - BarcelonaTech (UPC)
ESEIAAT, Carrer Colom 11, 08222 Terrassa (Barcelona).

1 Introduction

Sharp discontinuities are present in many industrial applications. Such discontinuities appear in both compressible and multiphase flows, among others. In the context of hyperbolic systems, discontinuities develop as shock waves, which present a challenge for numerical calculations. Traditionally, high speed aerodynamics has focused on the appropriate treatment of such discontinuities by developing shock capturing schemes. Advances in this field have been exploited by the multiphase flow community as well, particularly for the advection of the marker function [1].

As a consequence of Godunov's theorem [2] (after [3]) the treatment of shock discontinuities with linear schemes is limited to first order approximations. Higher order linear schemes will result in an unstable discretization and the onset of wiggles. The construction of stable, second order (and higher) discretizations, then, requires the adoption of high resolution non-linear schemes which exhibit a Total Variation Diminishing (TVD) [4] behavior. Among them, flux limiters are a mature and robust method, which has been adopted in a diversity of applications. Sweby [5] generalized several limiters and stated the conditions for 2nd order TVD schemes in a 1D homogeneous mesh in its well known *Sweby diagram*. Despite the known inconsistencies that arise when departing from the 1D homogeneous case [6, 7] these techniques have been ported to non-homogeneous Cartesian [7] and unstructured [8] meshes as well.

Traditionally, both the analysis and the implementation of flux limiters are performed from a stencil-based perspective. On the other hand, the growing interest of the community in mimetic methods, which inherently preserve the spatial structure of the solution [9, 10], demands a new approach to flux limiters. Such an approach constructs discrete operators directly from the inherent incidence matrices that define the mesh. This presents an important advantage both from theoretical and practical points of view, as it is discussed below.

On the one hand, this allows for a flawless discrete mimicking of the continuum operators. In particular, it allows for the exact conservation of important secondary properties, such as kinetic energy [11, 12], among others. On the other hand, the adoption of an algebraic topology approach provides, directly, with a set of algebraic operations which are better suited for both computational implementation and algebraic analysis [13] than its stencil-based counterpart.

By casting discrete operators into algebraic forms (i.e., matrices and vectors) it has been shown that nearly 90% of the operations comprised in a typical CFD algorithm for the DNS of an incompressible flow are comprised of the following: Sparse Matrix-Vector multiplication (SpMV), generalized vector addition (AXPY) and dot product (DOT) [14]. This reduces the implementation of numerical codes to the right combination of a few basic operations.

From a practical point of view, the advent of parallel heterogeneous architectures has motivated a new demand for portability. From this perspective, the use of a unified approach is desired in order to simplify

architecture-oriented implementations to portable ones without a significant lack of performance. This provides an opportunity for High Performance Computing (HPC) optimization, parallelization and portability [15]. In this regard, the casting of a numerical method into a few algebraic operations facilitates an efficient implementation in multiple architectures.

Oyarzun et al. [16] have implemented a Conjugate Gradient (CG) method following such an operator-based approach. Borrell et al. [17] showed the potential of such approach with the aim of pursuing petascale simulations. In all cases the use of an operator-based formulation has provided with robust, portable and optimized implementations. Consequently, the design of operator-based algorithms for its use in massively parallel architectures is a smart strategy towards the efficient solution of both industrial and academic scale problems [18].

In this paper a flux limiter implementation based on algebraic operations is detailed and its implementation tested in heterogeneous architectures.

The rest of the paper is organized as follows: In section 2 a review of basic chain and graph theory is briefly summarized in order to provide some context. Section 3 develops a generalization of flux limiters from an algebraic perspective. Finally, section 4 highlights the capabilities of the method in both structured and unstructured meshes.

2 Algebraic Topology

By using concepts from algebraic topology, mimetic methods preserve the inherent structure of the space, leading to stable and robust discretizations [19, 10]. However, the development of such techniques is out of the scope of this paper, where we rather focus on exploiting the relationships between the different entities of the mesh for the construction of Flux Limiters. The interested reader is referred to [9] and references therein.

For whatever space of interest Ω, we can equip it with a partition of unity, namely a mesh $\mathcal{M}$, by bounding the group of cells, $\mathcal{C}$, with faces, $\mathcal{F}$; those with the set of edges, $\mathcal{E}$, and finally those with the set of vertices, $\mathcal{V}$. In this sequence, groups are related to the next element of the sequence by means of the boundary operator ∂. This is know as a chain complex [19, 20]. A 2D example can be seen in Figure 1, where face and edges collapse into the same entity.

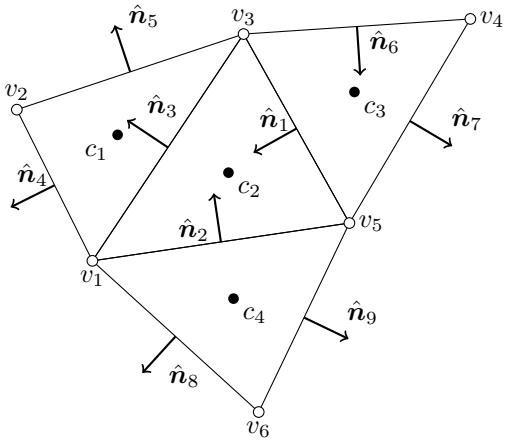

Figure 1: 2D mesh composed of cells c_i, which are bounded by faces f_j oriented in the direction $\hat{\boldsymbol{n}}_j$. Faces, which collapse to the same entities as edges, are bounded by the set of vertices v_k.

$$T_{\mathcal{F}\to\mathcal{V}} = \begin{pmatrix} 0 & +1 & +1 & +1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & +1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 & -1 & 0 & 0 \\ +1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & +1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & +1 & -1 \end{pmatrix} \tag{1}$$

$$T_{\mathcal{C}\to\mathcal{F}} = \begin{pmatrix} 0 & -1 & +1 & 0 \\ 0 & -1 & 0 & +1 \\ -1 & +1 & 0 & 0 \\ +1 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & +1 \end{pmatrix} \tag{2}$$

The relationship between the bounding elements of the cells can be casted in an oriented incidence matrix such as that in equation (1). Conversely, we can define $T_{\mathcal{F}\to\mathcal{C}}$, $T_{\mathcal{E}\to\mathcal{F}}$ and $T_{\mathcal{V}\to\mathcal{E}}$, for the face-to-cell, edge-to-face and vertex-to-edge incidence matrix. In addition, the converse incidence matrices can be seen by

transposing such matrices. Equations (1) and (2) represent the corresponding incidence matrices for the mesh depicted in Figure 1.

Incidence matrices represent the boundary operator between one element of the chain and the next one. Following the example of Figure 1, $T_{\mathcal{F}\to\mathcal{C}}$ provides with the orientation of the boundary faces f_j for cell c_i. On the other hand, $T_{\mathcal{V}\to\mathcal{F}}$ provides with the orientation of the bounding vertices v_k to every face f_j.

Incidence matrices play an essential role in preserving properties of the discrete space. In particular, they form an exact sequence. Exact sequences are those such that the application of the boundary operator twice results in 0. This can be verified by checking $T_{\mathcal{F}\to\mathcal{V}}T_{\mathcal{C}\to\mathcal{F}} = 0$. This property is shared by its continuum counterpart, the de Rahm cohomology [10], which is the ultimate responsible of the following vector calculus identities [19]:

$$\nabla \times \nabla \equiv 0 \tag{3}$$

$$\nabla \cdot \nabla \times \equiv 0 \tag{4}$$

These are powerful identities that mimetic methods preserve by construction. For an extended review of the relationship between the continuum and the discrete counterparts, the reader is referred to [19, 10] and references therein.

In addition to provide a suitable platform for the construction of appropriate mimetic methods, the relations contained in incidence matrices can be studied from a graph theory perspective.

A straightforward use of incidence matrices allows to compute differences across faces. The fact that differences lie in a different space (faces) than variables (cells) is an inherent property of such an approach.

$$\Delta \boldsymbol{u_c} = T_{\mathcal{C}\to\mathcal{F}}\boldsymbol{u_c} \tag{5}$$

Particularly useful is the construction of undirected incidence matrices ($B_{\mathcal{Q}\to\mathcal{S}}$), which are build by taking the absolute value of the elements of the directed ones ($T_{\mathcal{Q}\to\mathcal{S}}$). Considering the index notation between a generic space $\mathcal{Q}$ (e.g., cells, faces) and its boundary $\mathcal{S}$ (e.g., faces, edges), we could proceed as follows:

$$B_{\mathcal{Q}\to\mathcal{S}} = b_{sq} = |t_{sq}| \tag{6}$$

Similarly, one can proceed to compute the degree matrix of the graph, which accounts for the number of connections that an entity has (e.g., the number of cells in contact with a face). Degree matrices are always diagonal and the value of the diagonal elements is obtained as follow:

$$diag(W_{\mathcal{Q}\mathcal{Q}}) = B_{\mathcal{S}\to\mathcal{Q}}\mathbb{1}_{\mathcal{S}} = b_{qs}1_s \tag{7}$$

In particular, undirected incidence matrices can be used to construct suitable shift operators [21] as in equation (8). This provides with a simple face-centered interpolation, weighted with the number of adjacent faces. Note that by taking this approach, boundaries are inherently included from the graph information.

$$\Pi_{\mathcal{C}\to\mathcal{F}} = W^{-1}_{\mathcal{F}\mathcal{F}}B_{\mathcal{C}\to\mathcal{F}} \tag{8}$$

The use of $\Pi_{\mathcal{C}\to\mathcal{F}}$ as it is is restricted to scalar fields. However, following [21], this can be readily extended to vector fields as follows. To do so we shall consider first the discretization of vector fields. First, the discretization of a continuum vector field $\vec{x} = (u, v, w)^T$ is arranged in a single vector as $\vec{\boldsymbol{x}}_{\boldsymbol{c}} = (\boldsymbol{u_c}, \boldsymbol{v_c}, \boldsymbol{w_c})^T$, where $\boldsymbol{u_c}$, $\boldsymbol{v_c}$ and $\boldsymbol{w_c}$ represent the value of u, v and w at cell c. Note that all components are arranged in a single $\mathbb{R}^{|\mathcal{C}|d\times 1}$ column vector, where $|\mathcal{C}|$ is the number of cells and d is the number of dimensions of the problem. Next, the interpolator can be extended component-wise by applying the Kronecker product with the identity matrix of size $\mathbb{R}^{d\times d}$. The final ensemble is as follows:

$$\Gamma_{\mathcal{C}\to\mathcal{F}} = \mathbb{I}_d \otimes \Pi_{\mathcal{C}\to\mathcal{F}} \tag{9}$$

Similarly, normal vectors can be arranged into a $\mathbb{R}^{|\mathcal{F}|\times|\mathcal{F}|d}$ matrix by arranging d diagonal matrices, corresponding to every component of the face vector, next to each other as $N = (N_u|N_v|N_w)$ [21]. Equation (10) shows the N matrix corresponding to the mesh depicted in Figure 1. In such a way, it is straightforward

to either project a discrete vector as $N\vec{\boldsymbol{x}}_{\boldsymbol{f}}$, or to vectorize a scalar quantity as $N^T \boldsymbol{s}_{\boldsymbol{f}}$, provided that both are stored at the faces. An accurate discussion about the construction of this matrix can be found in [21].

$$N = \begin{pmatrix} n_{1x} & 0 & \dots & 0 & n_{1y} & 0 & \dots & 0 \\ 0 & n_{2x} & \dots & 0 & 0 & n_{2y} & \dots & \vdots \\ \vdots & 0 & \ddots & \vdots & \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \dots & n_{9x} & 0 & 0 & \dots & n_{9y} \end{pmatrix} \tag{10}$$

Other basic matrices derived from the graph are both the graph Laplacian ($L_{\mathcal{CC}}$) and the adjacency matrix ($A_{\mathcal{CC}}$).

$$L_{\mathcal{CC}} = B_{\mathcal{F}\to\mathcal{C}} B_{\mathcal{C}\to\mathcal{F}} \tag{11}$$

$$A_{\mathcal{CC}} = W_{\mathcal{CC}} - L_{\mathcal{CC}} \tag{12}$$

Both are constructed based on the incidence matrices and provide information about the propagation of information along the graph. They are constructed by connecting cells to its neighbors trough its bounding faces. In summary, the constructor of such operators provides with tools able to relate different elements of the graph between each others. Equipped with such basic concepts, the development of higher level operators can proceed as in the following section.

3 Flux Limiters

The solution of hyperbolic problems in finite volume methods when sharp discontinuities are present requires the use of high resolution schemes in order to attain second order approximations. In turn, the construction of such schemes is reduced to the appropriate reconstruction of the flux at the faces. Prone to introduce numerical instabilities, such a reconstruction requires, in turn, an appropriate flux reconstruction strategy in order to guarantee TVD behavior (i.e., such that no new minima or maxima are introduced). This is attained by limiting the flux at cell's boundaries. From a physical point of view, this is equivalent to the introduction of some sort of artificial diffusion which stabilizes the method at the expense of smearing out its profile. The limiting approach has been used by several authors [22, 2], who over the years developed several discontinuity sensors in order to limit dissipation to the region near the shock. Among all discontinuity sensors, the most popular is the use of the gradient ratio, which is defined as follows [5, 8, 1]:

$$r_f = \frac{\Delta_U \theta}{\Delta_u \theta} \tag{13}$$

Where $\Delta_U \theta$ is the gradient of θ at the upwind face while $\Delta_u \theta$ correspond with the gradient at the face of interest. Both differences are taken as positive in the flow direction, defined by the velocity field u. This provides with an intuitive intuition of where discontinuities are and its order of magnitude, at the time that keeps a narrow stencil. In addition, it allows to, after proper manipulation by the flux limiter itself, limit the flux in a way which can be interpreted as a diffusion-like term. This is known as "upwinding" as it has the same effect as recovering the 1st order upwind discretization near shocks.

TVD conditions in terms of the gradient ratio where stated by Harten in 1983 [4] for 1D homogeneous grids. These conditions where used by Sweby in 1984 [5] to state 2nd order *and* TVD conditions for different forms of flux limiters. This idea has been extended, with several degrees of accuracy, to multidimensional and irregular grids [6, 7], among others.

3.1 Algebraic Implementation

Typically, flux limiters are stated in the following form [5, 8, 1]:

$$\theta_f = \theta_U + \Psi(r) \left(\frac{\theta_D - \theta_U}{2} \right) \tag{14}$$

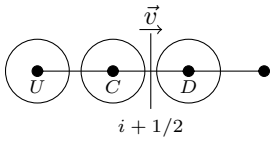

Figure 2: Classical stencil for the computation of the gradient ratio at face $i+1/2$. U, C and D correspond to the upstream, centered and downstream nodes.

Where θ_U and θ_D stand for the upwind and downwind values of θ according to the velocity field u and $\Psi(r)$ is the flux limiter function. Figure 2 depicts this situation. However, this can be recast in the less common form:

$$\theta_f = \frac{\theta_U + \theta_D}{2} + \frac{\Psi(r) - 1}{2} (\theta_D - \theta_U) \tag{15}$$

And successively into:

$$\theta_f = (\Pi_{\mathcal{C}\to\mathcal{F}} + F(r)_{\mathcal{C}\to\mathcal{F}})\, \theta_c \tag{16}$$

Where $\Pi_{\mathcal{C}\to\mathcal{F}}$ is the standard cell-to-face interpolation defined in equation (8) and $F(r)_{\mathcal{C}\to\mathcal{F}}$ absorbs the right term of equation 15. At this point, we may be tempted to analyze the construction of new flux limiters by means of basic algebra concepts. In particular, to bound its spectrum by means of Gershgorin's theorem or to check its entropy conditions [23], among others. The interested reader is referred to Báez et al. [24], where a similar approach is taken for spatial filters.

Because the value of $F(r)_{\mathcal{C}\to\mathcal{F}}$ depends entirely on the value of r, the problem is turned into the accurate computation the gradient ratio for an arbitrary face. There has been several approaches [8, 6] to the construction of such a variable in terms of a least-squares reconstructed gradient. However, the implementation of such schemes can be cumbersome and may not, eventually, recover the 1D homogeneous solution when a homogeneous structured mesh is used.

The construction of the gradient ratio will proceed first by the separate calculation of both the numerator ($\Delta_U\theta$) and the denominator ($\Delta_u\theta$) of equation (13). Both of them will be produced from simple SpMV operations, while the division will be performed element-wise.

In this approach we propose to employ symmetry-preserving gradients (see [11]) into the calculation of both face-centered and upstream gradients in order to preserve, as much as possible, the mimetic properties of the approach. In addition, we aim at recovering the Cartesian formulation as in [1].

Before any calculation, the sign matrix ($S(u)$) is constructed by assigning to a $\mathbb{R}^{F\times F}$ diagonal matrix $+1$ for a positive velocity and -1 for a negative one. This allows for a straightforward calculation of gradient at the face as follows:

$$\Delta_u = S(u) T_{\mathcal{C}\to\mathcal{F}} \tag{17}$$

Where $S(u)$ is used to provide the right direction in which the difference is taken according to the velocity field.

The construction of the upwind delta, Δ_U is more involved. The idea is to construct a partial adjacency matrix which only considers upstream faces, namely the upstream adjacency matrix, $A^U_{\mathcal{F}\mathcal{F}}(u)$, which is responsible to garner upstream information and will be defined further in this paper.

We proceed as follows: Δ is used to compute the difference across every face according to equation (5). In order to asses the contribution of every neighboring face to the face of interest, face differences are vectorized with its corresponding face normal using N^T and added all together with $A^U_{\mathcal{F}\mathcal{F}}(u)$. Finally, the resulting value is projected over the normal of the face of interest by means of N. The overall construction of the operator is as follows:

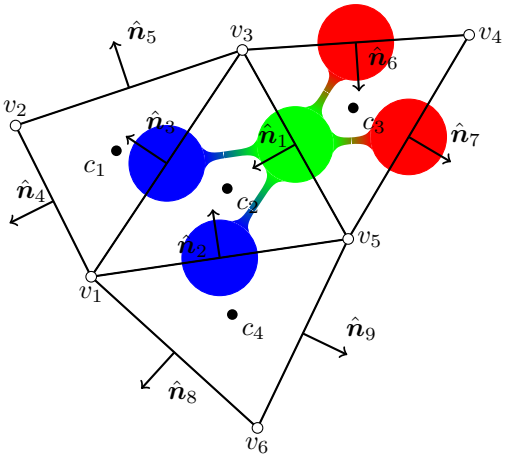

Figure 3: Forward (red) and backward (green) upwind adjacent faces for face 1. Forward faces are those that are upstream of face 1 when the velocity in such a face has a positive component. On the other hand, backward upwind faces are those located upstream of face 1 when the velocity flows in the opposite direction. The selection of the right ones will depend on the sign of the velocity at face 1. The ensemble of such a matrix for all faces results in $A^U_{\mathcal{FF}}(u)$.

$$\Delta_U = N \left(\mathbb{I}_d \otimes A^U_{\mathcal{FF}}(u) \right) N^T \Delta \tag{18}$$

Where, similarly to what we did in equation (9), we have reused the $A^U_{\mathcal{FF}}(u)$ operator for all spatial dimensions. Normal matrices N, defined as in equation (10), are used for both vectorization and projection of the neighboring differences. In this way, orthogonal meshes recover the original 1D formulation, whereas arbitrary ones are handled inherently by the incidence matrix.

The construction of the upstream adjacency matrix, $A^U_{\mathcal{FF}}(u)$, may look, at a first glance, of high computational cost. However, it can be assembled from 2 other simpler matrices.

$$A^U_{\mathcal{FF}}(u) = \frac{1}{2} \left(S(u) A^D_{\mathcal{FF}} - A_{\mathcal{FF}} \right) \tag{19}$$

Where $A_{\mathcal{FF}}$ is the regular adjacency matrix, which contains only positive entries and, consequently, flux direction is not discerned. This is constructed similarly to equation (12) as follows:

$$A_{\mathcal{FF}} = W_{\mathcal{FF}} - B_{\mathcal{C}\to\mathcal{F}} B_{\mathcal{F}\to\mathcal{C}} \tag{20}$$

As seen in section 2, the adjacency matrix is symmetric and contains non-negative entries only. Following on the example depicted in Figure 3, the corresponding $A_{\mathcal{FF}}$ can be seen in equation (21).

$$A_{\mathcal{FF}} = \begin{pmatrix} 0 & +1 & +1 & 0 & 0 & +1 & +1 & 0 & 0 \\ +1 & 0 & +1 & 0 & 0 & 0 & 0 & +1 & +1 \\ +1 & +1 & 0 & +1 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +1 & +1 & 0 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & +1 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & 0 & +1 \\ 0 & +1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 \end{pmatrix} \tag{21}$$

On the other hand, the construction of $A^D_{\mathcal{FF}}$ allows us to distinguish neighboring faces which lie behind or ahead of the face in question. This requires the inclusion of the directed incidence matrix $T_{\mathcal{C}\to\mathcal{F}}$ in the calculation of the adjacency matrix as:

Operation	SpMV	axpy	axdy	shft	scal	vmax/vmin	smax/smin	sign
$S(u)$	0	0	0	0	0	0	0	1
$\Delta_U\theta$	3	1	0	0	0	0	0	0
$\Delta_u\theta$	2	0	0	0	0	0	0	0
r	0	0	1	0	0	0	0	0
$\Psi(r)$	0	0	0	1	1	1	3	0
$\Omega(r)$	0	0	0	0	1	0	0	0
Euler	6	2	0	0	0	0	0	0
total	11	3	1	1	2	1	3	1

Table 1: Operation count for the advance of a time step in a pure advection problem with the use of a SUPERBEE flux limiter. For this case an Euler integration scheme has been used. For details see [25]. Face velocities are given for such a problem and so its operation count is excluded from this count.

$$A^D_{\mathcal{FF}} = W_{\mathcal{FF}} - T_{\mathcal{C}\to\mathcal{F}} B_{\mathcal{F}\to\mathcal{C}} \tag{22}$$

This will provide with a matrix as the one seen in equation (23). Together with $S(u)$ in equation (19), $A^D_{\mathcal{FF}}$ is the responsible to cancel the appropriate side of the unsigned adjacency matrix $A_{\mathcal{FF}}$ and thus provide with the proper upstream information. Note that both $A_{\mathcal{FF}}$ and $A^D_{\mathcal{FF}}$ are constant matrices and that the only matrix that needs to be updated as the velocity field u changes is $S(u)$.

$$A^D_{\mathcal{FF}} = \begin{pmatrix} 0 & +1 & +1 & 0 & 0 & -1 & -1 & 0 & 0 \\ +1 & 0 & +1 & 0 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & 0 & +1 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 & 0 & 0 & +1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \tag{23}$$

The combination of $A^D_{\mathcal{FF}}$ and $A_{\mathcal{FF}}$ in equation (18) results in the computation of Δ_U to be cast a constant plus a term which depends on the orientation of the flow at the face as follows:

$$\Delta_U = N\left(\mathbb{I}_d \otimes \frac{1}{2}\left(S(u)A^D_{\mathcal{FF}} - A_{\mathcal{FF}}\right)\right)N^T\Delta \tag{24}$$

This matrix is computed and stored *a priory* in order to speed up the computations and avoid expensive matrix-matrix products. The operations count for the evaluation of a typical advection scheme that uses a flux limiter is stated in Table 1. As it can be seen, the computation of several local quantities is required in order to construct the final algebraic scheme.

This results in a compact algorithm, which facilitates portability by reducing the number of computing kernels.

3.2 Comparison with the stencil-based formulation

The new approach for the proper construction of gradient ratios is now compared with the classical, stencil-based approach.

In a general, 1D case for a Cartesian homogeneous grid, the computation of the gradient ratio, r, involves the use of, at least, 3 nodes trough a face located at $i+1/2$. Depending on whether the velocity is going in one direction or another, the definition of the upstream nodes will change accordingly. This can be exemplified in Figure 4.

On the other hand, the use of the proposed algebraic approach apparently involves a larger stencil, as it can be seen in Figure 5. However, the computation of the proper upstream adjacency matrix $A^U_{\mathcal{FF}}(u)$

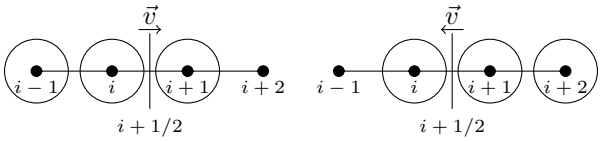

Figure 4: Example of stencils in 1D according to the classical flux limiter approach. The stencil topology is not constant and changes according to the sign of the velocity field $\vec{v}$.

according to equation (19) will results in the appropriate stencil (i.e., the pattern of the sparse matrix) for $A^U_{\mathcal{FF}}(u)$.

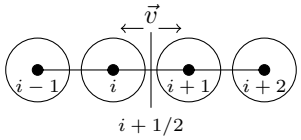

Figure 5: Equivalent stencils used in the computation of an algebraic flux limiter. In this case the adjacency matrices involve the operation with all neighboring nodes.

We note that even when, apparently, the algebraic operation may look more involved at a first glance, the data communication in distributed memory parallelization is the same as in both a classical stencil-based implementation and in an algebraic one. Indeed, the higher number of operations required by the algebraic approach have little effect on the overall system performance [25]. The halo width requires 2 face-adjacent nodes, whereas in [1] 2 vertex-adjacent nodes are required.

4 Numerical Results

Next, the application of this technique is applied to a canonical case. In particular, the advection of a sharp profile has been tested for both structured and unstructured meshes.

Computations are performed in both CPU and GPU computing nodes thanks to HPC^2 [18], a numerical framework designed for the efficient computation of basic algebraic operations in a hybrid systems. Details of the implementation can be found in the companion paper of Álvarez et al. [25].

In this particular case, the advection of a rhodonea, defined in equation (27) after [26], is analyzed. Such a field is subject to a flat velocity field $\vec{u} = (1, 0)$ and periodic boundary conditions. The physical domain is $[-1, 1] \times [-1, 1]$. Integration in time has been performed with a 1st order fully explicit Euler scheme to a total of two complete cycles of advection.

$$x(\theta) = r(\theta)cos(\theta) \tag{25}$$
$$y(\theta) = r(\theta)sin(\theta) \tag{26}$$
$$r(\theta) = r_0 + r_1 sin(\omega\theta) \tag{27}$$

As it can be seen, flux limiters provide with stable solutions at the expense of a smeared out interface. The sharpness of the profiles can be seen in Figures 7, where the structured mesh performs better than its unstructured counterpart due to the fact that the velocity field is perfectly aligned with one of the face normals.

5 Conclusions and Future Work

A flux limiter scheme has been formulated from an algebraic perspective. This results in a compact formulation that allows for an easy implementation on different heterogeneous architectures.

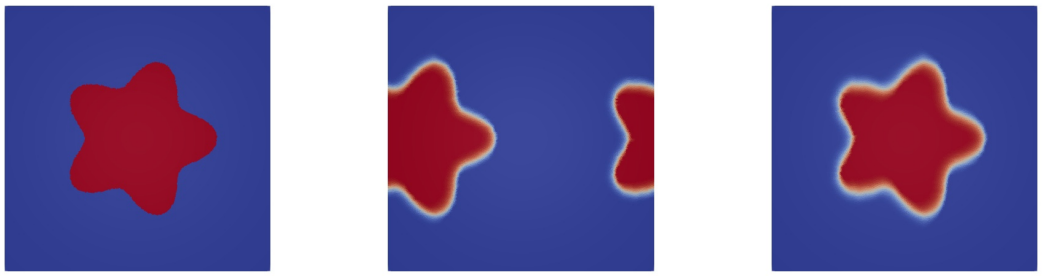

Figure 6: Sequence of the advection of a rhodonea with a constant velocity field with SUPERBEE in an unstructured mesh. Parameters for the initial profile are as follow: $r_0 = 0.5$, $r_1 = 0.1$ and $\omega = 5$

Graph incidence matrices (both directed and undirected) are exploited to construct appropriate gradient ratios. The face velocity sign determines the appropriate side to pick the upstream information. After the sign operation, the remaining operations are all of them either linear or local.

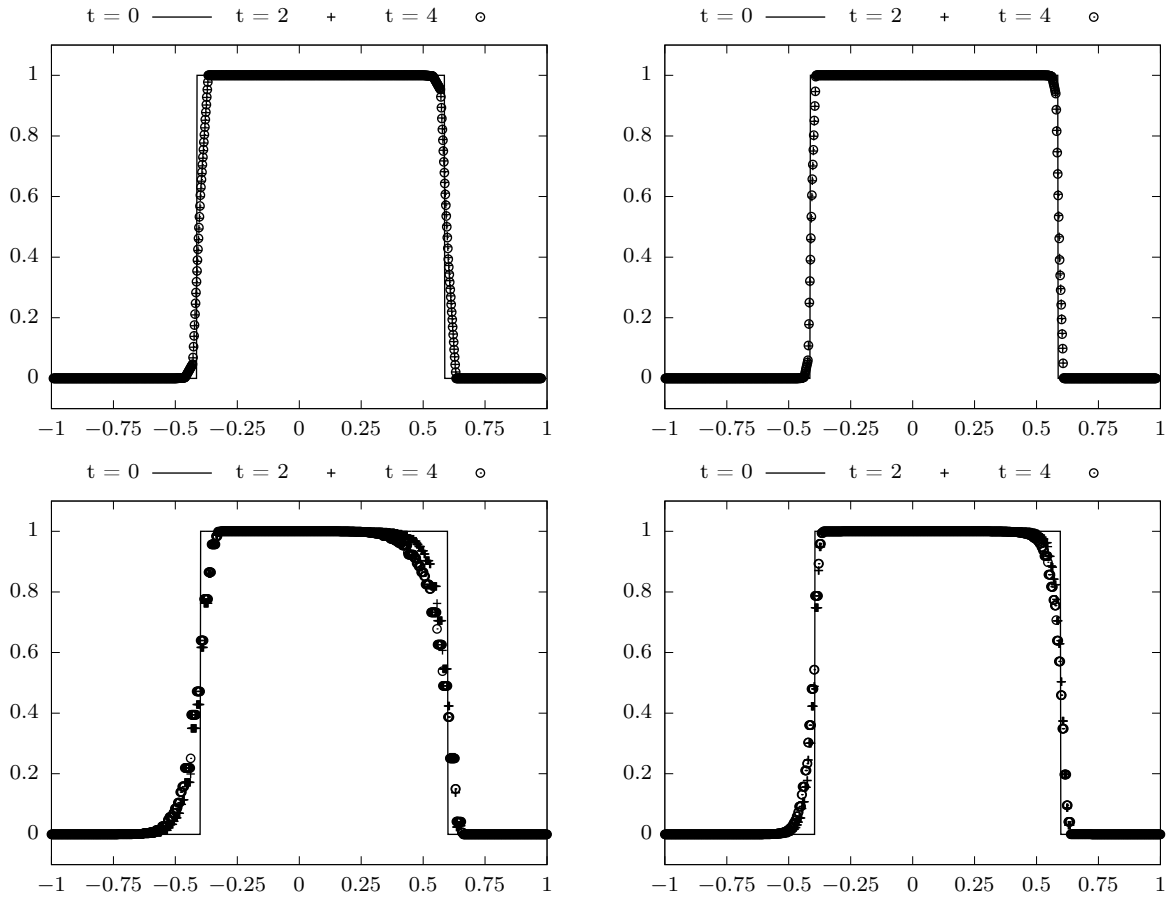

Figure 7: Initial, first and second pass profiles for the horizontal mid line with structured (top) and unstructured (bottom) meshes. Left column corresponds to a meshes with a characteristic lenght of $\Delta x = 1/32$, while right columns are produced with a characteristic length of $\Delta x = 1/64$.

This approach presents several advantages. First of all, casting a flux limiter in algebraic form enhances a higher level of analysis, which has not been considered in this paper. On the other hand, the deployment of such an algebraic form into an heterogeneous computing system reduces the number of computing kernels that need to be ported from one architecture to the other, enhancing portability and improving code development and maintenance.

Regardless the inconsistencies that where already highlighted in sections 1 and 3, the resulting implementation provides with accurate results and collapses to the traditional approach of Sweby [5, 8, 1] when a homogeneous, Cartesian grid is used.

The approach developed in this work can be improved to include the effect of the non-homogeneous distance across upstream faces or its surface in the calculation of the upstream gradient.

6 Acknowledgments

We acknowledge N. Balcázar and A. Bàez for its comments and discussions. The work has been financially supported by the *Ministerio de Economía y Competitividad*, Spain (ENE2017-88697-R and ENE2015-70672-P). N. V. is supported by a *FI AGAUR* predoctoral contract (2018FI_B1_000109). X. Á. is supported by a *FI AGAUR* predoctoral contract (2018FI_B1_00081). F. X. T. is supported by a *Ramón y Cajal* postdoctoral contract (RYC-2012-11996).

References

[1] N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and J. Rigola, "A finite-volume/level-set method for simulating two-phase flows on unstructured grids," *International Journal of Multiphase Flow*, vol. 64, pp. 55–72, may 2014.

[2] C. Hirsch, *Numerical computation of internal and external Flows. Vol. 2: Computational Methods for Inviscid and Viscous Flows*. Wiley-Interscience, 1990.

[3] S. Godunov, "A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics," *Mat. Sb.*, vol. 47, no. 271, pp. 271–306, 1959.

[4] A. Harten, "High resolution schemes for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 49, no. 3, pp. 357–393, 1983.

[5] P. K. Sweby, "High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws," *SIAM Journal on Numerical Analysis*, vol. 21, pp. 995–1011, oct 1984.

[6] M. Berger, M. J. Aftosmis, and S. M. Murman, "Analysis of Slope Limiters on Irregular Grids," tech. rep., 2005.

[7] X. Zeng, "A General Approach to Enhance Slope Limiters in MUSCL Schemes on Nonuniform Rectilinear Grids," *SIAM Journal on Scientific Computing*, vol. 38, no. 2, pp. A789–A813, 2016.

[8] M. S. Darwish and F. Moukalled, "TVD schemes for unstructured grids," *International Journal of Heat and Mass Transfer*, vol. 46, no. 4, pp. 599–611, 2003.

[9] K. Lipnikov, G. Manzini, and M. Shashkov, "Mimetic finite difference method," *Journal of Computational Physics*, vol. 257, no. PB, pp. 1163–1227, 2014.

[10] R. Hiemstra, D. Toshniwal, R. Huijsmans, and M. Gerritsma, "High order geometric methods with exact conservation properties," *Journal of Computational Physics*, vol. 257, pp. 1444–1471, jan 2014.

[11] R. Verstappen and A. Veldman, "Symmetry-preserving discretization of turbulent flow," *Journal of Computational Physics*, vol. 187, no. 1, pp. 343–368, 2003.

[12] F. X. Trias, *Direct numerical simulation and regularization modelling of turbulent flows on loosely coupled parallel computers using symmnetry-preserving discretizations*. PhD thesis, Universitat Politècnica de Catalunya, dec 2006.

[13] F. X. Trias, A. Gorobets, and A. Oliva, "A simple approach to discretize the viscous term with spatially varying (eddy-)viscosity," *Journal of Computational Physics*, vol. 253, pp. 405–417, 2013.

[14] G. Oyarzun, R. Borrell, A. Gorobets, and A. Oliva, "Portable implementation model for CFD simulations. Application to hybrid CPU/GPU supercomputers," *International Journal of Computational Fluid Dynamics*, vol. 31, no. 9, pp. 396–411, 2017.

[15] F. D. Witherden, *On the Development and Implementation of High-Order Flux Reconstruction Schemes for Computational Fluid Dynamics by.* PhD thesis, Imperial Collefe London, 2015.
[16] G. Oyarzun, R. Borrell, A. Gorobets, and A. Oliva, "MPI-CUDA sparse matrix–vector multiplication for the conjugate gradient method with an approximate inverse preconditioner," *Computers & Fluids*, vol. 92, pp. 244–252, 2014.
[17] R. Borrell, J. Chiva, O. Lehmkuhl, I. Rodríguez, and A. Oliva, "Evolving Termofluids CFD Code Towards Peta-Scale Simulations," in *27th International Conference on Parallel Computational Fluid Dynamics*, pp. 1–8, 2015.
[18] X. Álvarez, A. Gorobets, F. X. Trias, R. Borrell, and G. Oyarzun, "HPC^2 - a fully-portable, algebra-based framework for heterogeneous computing. Application to CFD," *Computers & Fluids (published online)*, 2018.
[19] N. Robidoux and S. Steinberg, "A discrete vector calculus in tensor grids," *Computational Methods in Applied Mathematics*, vol. 11, no. 1, pp. 23–66, 2011.
[20] E. Tonti, "Why starting from differential equations for computational physics?," *Journal of Computational Physics*, vol. 257, no. PB, pp. 1260–1290, 2014.
[21] F. Trias, O. Lehmkuhl, A. Oliva, C. Pérez-Segarra, and R. Verstappen, "Symmetry-preserving discretization of Navier–Stokes equations on collocated unstructured grids," *Journal of Computational Physics*, vol. 258, pp. 246–267, feb 2014.
[22] C. Hirsch, *Numerical Computation of Internal and External Flows. Vol. 1: Fundamentals of Numerical Discretization.* Brussels: Wiley-Interscience, 1988.
[23] S. Osher and S. Chakravarthy, "High Resolution Schemes and the Entropy Condition," tech. rep., 1984.
[24] A. Báez Vidal, O. Lehmkuhl, F. Trias, and C. Pérez-Segarra, "On the properties of discrete spatial filters for CFD," *Journal of Computational Physics*, vol. 326, pp. 474–498, dec 2016.
[25] X. Álvarez, N. Valle, A. Gorobets, and F. Trias, "Implementation of a flux limiter into a fully-portable, algebra-based framework for heterogeneous computing," in *Tenth International Conference on Computational Fluid Dynamics*, (Barcelona), 2018.
[26] M. Oevermann and R. Klein, "A Cartesian grid finite volume method for elliptic equations with variable coefficients and embedded interfaces," *Journal of Computational Physics*, vol. 219, no. 2, pp. 749–769, 2006.