FINITE VOLUME SCHEMES ON 2D NON-UNIFORM GRIDS

Gabriella Puppo

Dip. di Scienza e Alta Tecnologia, Università Insubria
Via Valleggio 11
22100, Como, Italy

Matteo Semplice

Dip. di Matematica, Università di Torino
Via Carlo Alberto 10
10123 Torino, Italy

Abstract. In this work we briefly describe a technique to define second order finite volume schemes on non uniform cartesian grids. The purpose is to couple this scheme with an error indicator to drive mesh adaptivity. In this context, it is crucial that the underlying scheme uses a very compact stencil. Here we illustrate an algorithm that matches this goal, without giving up accuracy, while relaying on a non oscillatory reconstruction.

1. **Introduction.** We are interested in the integration in 2D of systems of conservation laws of the form

$$u_t + f_x(u) + g_y(u) = 0 \tag{1}$$

where the system is assumed to be hyperbolic, namely any linear combination of the Jacobian matrices $f'(u)$ and $g'(u)$ is diagonalizable with real eigenvalues. The purpose of this work is to integrate such a system with a finite volume, second order scheme, using a locally adaptive grid with a cartesian structure. The advantage of this procedure is that schemes based on cartesian grids are easier to parallelize than schemes based on unstructured grids, because it is easy to store the grid in a tree structure, allowing for easier communication between the grid elements. It is frequently said that the main drawback of cartesian grids is due to the difficulty in dealing with curved boundaries, but this problem can be solved using a combination of level sets to describe the boundary and the ghost fluid method, see [1], [2], and it will not be discussed further in this work.

The scheme described here will be applied in the context of adaptive grid refinement, driven by an error indicator based on the entropy production, as in [3]. As opposed to the rectangular patches of finer grids used in [4], our approach allows to refine every single cell independently of its neighbors and thus, even in the cartesian setting, the location of cells in the neighborhood and their sizes can vary in a wide pool of possible patterns. Moreover, the amplitude of each cell may change as a result of the error diagnosed within a time step, and in particular the solution just computed in a cell that undergoes refinement must be recomputed on the refined

2000 *Mathematics Subject Classification.* Primary: 65M08; Secondary: 65M50.

Key words and phrases. Finite Volume Schemes, Conservation Laws, Non-uniform cartesian grids.

This work is partly supported by PRIN-2009588FHJ002.

cells to improve its accuracy. Consequently the stencil on which the scheme is based must be as compact as possible, so that the necessity of recomputing the solution spreads as little as possible on neighboring cells. The reconstruction proposed here is targeted on this particular requirement.

2. **Reconstruction.** We suppose that the computational domain is covered with a non uniform grid composed of non overlapping square cells. Each cell has a side which is a submultiple by a power of 2 of a basic length h. The cells are labelled with a single integer index, $j = 1, \dots, N$. The cell V_j is centered around the point (x_j, y_j) and has a width $h_j = 2^{-l_j} h$, where l_j denotes the current cell level of refinement, $l_j = 0, \dots, l_{\max}$, and $l_{\max}$ is the maximum level of refinement. The grid structure is stored in a tree, as described in [3]. The leaves of the tree represent the active cells. Note that the following procedure can be extended to rectangular cells, with some tedious modifications.

The time step starts with a knowledge of the cell averages of the unknown u on each cell, $\overline{u}_j$, for $j = 1, \dots, N$. The first task is to compute a piecewise linear reconstruction from the cell averages. Suppose that $v(x, y)$ is a smooth function with cell averages $\overline{u}_j$ on the current grid. In this case the reconstruction ought to be given by

$$u(x, y) = \sum_{j=1}^{N} P_j^1(x, y) \chi_j(x, y),$$

where χ_j is the characteristic function of the j-th cell, and

$$P_j^1(x, y) = \overline{u}_j + \sigma_j^x (x - x_j) + \sigma_j^y (y - y_j),$$

with

$$\sigma_j^x = \partial_x v|_{(x_j, y_j)} + O(h_j), \quad \sigma_j^y = \partial_y v|_{(x_j, y_j)} + O(h_j), \tag{2}$$

so that $u(x, y) - v(x, y) = O(h)^2$. Moreover, the reconstruction must be non-oscillatory. The reconstruction is computed starting from the large cells in the grid. We illustrate the evaluation of the slope σ_j^x in the two cases illustrated in Fig. 1, which account for the typical cases that may occur. We need a left and a right slope, σ_j^L and σ_j^R, from which the actual slope will be computed via the MinMod function, or some other limiter. Referring to the figure, in both cases, the left slope is simply given by $\sigma_j^L = (\overline{u}_j - \overline{u}_1)/(x_j - x_1)$.

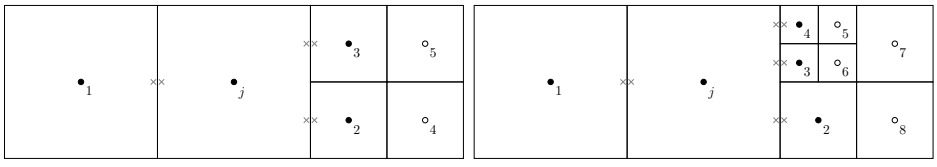

FIGURE 1. Computing the horizontal slopes

The right slope is different in the two cases shown in the figure. In the example appearing on the left, the cell j interfaces two smaller cells, with the same level of refinement. Clearly the centers of these cells have the same abscissas, and one can define

$$\sigma_j^R = c_2(\overline{u}_j - \overline{u}_2)/(x_j - x_2) + c_3(\overline{u}_j - \overline{u}_3)/(x_j - x_3),$$

where c_2 and c_3 are two positive weights adding up to 1. It is clear that if the weights are chosen equal (thus in this case, $c_2 = c_3 = \frac{1}{2}$), σ_j^R satisfies the requirement (2). The same behaviour occurs in the case in which the cell j interfaces with a larger number of cells, but all of them have the same level of refinement: we just need to modify the weights, which will be $c_r = 2^{-(l_r - l_j)}$. The second case (right of Fig. 1) is trickier. Here, the cells on the right do not have a uniform size, and their cell centers do not have the same abscissas. We still compute

$$\begin{aligned}\sigma_j^R &= c_2 \frac{\overline{u}_j - \overline{u}_2}{x_j - x_2} + c_3 \frac{\overline{u}_j - \overline{u}_3}{x_j - x_3} + c_4 \frac{\overline{u}_j - \overline{u}_4}{x_j - x_4} \\ &= \left(\sum_{k=2}^{4} c_k \right) v_x + \left(\sum_{k=2}^{4} c_k \frac{y_j - y_k}{x_j - x_k} \right) v_y + O(h)\end{aligned}$$

The coefficients in the first sum must add up to 1, while the sum of the coefficients in the second term must be zero. We further choose $c_3 = c_4$. In this fashion, recalling that $h_3 = h_4 = \frac{1}{4} h_j; h_2 = \frac{1}{2} h_j$, one easily finds $c_2 = \frac{6}{11}, c_3 = c_4 = \frac{5}{22}$, which give the correct weights to ensure that the slope in the large cell has the desired accuracy, as in (2). Clearly, Fig. 1 does not represent all possible cases. What is left is the possibility that the large cell j borders with cells of mixed type that have an even lower degree of refinement. In this case, it is still possible to compute constant weights, depending on the local pattern of refinement, but not on the solution itself, which ensure that the correct accuracy is matched, but this would complicate the scheme exceedingly. Alternatively, one could enlarge the stencil, including also the cells immediately to the right of V_3 and V_4 (namely the cells V_5 and V_6 in the figure), but this alternative would make the stencil less compact.

On the other hand, we note that c_3 is close to $\frac{1}{4}$, while c_2 is quite close to $\frac{1}{2}$. Thus in general we choose:

$$c_i = \begin{cases} 1 & \text{if } h_i = h_j \\ \frac{6}{11} & \text{if } h_i/h_j = \frac{1}{2} \\ \frac{5}{22} & \text{if } h_i/h_j = \frac{1}{4} \\ \frac{1}{2^{l_i - l_j}} & \text{if } h_i/h_j = 2^{l_j - l_i} < \frac{1}{4}, \end{cases} \tag{3}$$

where j denotes the index of the cell on which the reconstruction is sought, and i is the index of the neighboring cell which contributes to the slope in the cell V_j. The right slope will than be computed as

$$\sigma_j^R = \left(\sum_i c_i \frac{\overline{u}_j - \overline{u}_i}{x_j - x_i} \right) / \sum_i c_i,$$

where the sum is extended to all cells on the right of the cell V_j. In this fashion, accuracy is preserved exactly in the two common cases in which an edge of a cell V_j is either facing cells which are all of the same size (even if they are much smaller) or is facing cells which are at most two levels finer (in any possible disposition). On the other hand accuracy is almost preserved in the unlikely case in which a cell is surrounded by cells with a very uneaven refinement pattern. In all cases, the stencil consists only of the cells that are adjecent to the current cell.

Once the reconstruction of each cell of a given level l has been obtained, the algorithm proceeds to the evaluation of the reconstruction of cells of level $l+1$. For the sake of illustration, we still consider the cases of Fig. 1. We suppose we have

the reconstruction in the cell j and we compute the reconstruction in the cell V_2. The left slope will be given by

$$\sigma_2^L = \frac{P^1(x_j, y_j - h_2/2) - \overline{u}_2}{x_j - x_2},$$

while for the right slope we use the algorithm described above. In this case too accuracy is preserved. This completes the description of the evaluation of the reconstruction.

Equation (1) is written in finite volume formulation, namely

$$\frac{d\overline{u}_j}{dt} = -\frac{1}{h^2}\int_{\partial V_j} [F, G]^T(t) \cdot \mathbf{n}, \tag{4}$$

where F and G denote the numerical fluxes, consistent with f and g respectively, ∂V_j is the boundary of the cell V_j, and $\mathbf{n}$ is the external normal to ∂V_j. The line integrals are computed with the mid point rule, and the quadrature nodes depend on the cells surrounding the cell V_j. To compute the numerical fluxes, we need two boundary extrapolated data, across each quadrature node, which are computed from within each cell using the reconstruction. The location where the boundary extrapolated data are computed is shown by the crosses in Fig. 1. Once the boundary extrapolated data are known, we compute the numerical fluxes using the Local Lax Friedrichs formula. For the time integration, a second order explicit TVD Runge Kutta scheme is applied.

The main focus of this paper is on the numerical integration of 2D problems on non uniform grids, but we also provide, as a comparison, results obtained using an adaptive grid, in which the level of refinement and coarsening is computed dynamically, following the solution, with the aid of an a-posteriori error indicator. In this paper, we consider the error indicator based on the numerical residual in an entropy inequality for the original equation (1), see [3] and [5]. Let η and ψ be an entropy-entropy flux pair for (1). Then we define the density of entropy production for the cell V_j during the time step t^n, t^{n+1} as

$$S_j^n = \frac{1}{\Delta t}\left(\eta(\overline{u}_j^{n+1}) - \eta(\overline{u}_j^{n+1}) + \frac{\Delta t}{h^2}\sum_{i=1}^{2}\int_{\partial V_j}\Psi(i)\right) \tag{5}$$

where $\Psi(i)$ are numerical entropy fluxes, consistent with ψ, computed corresponding to the i-th Runge Kutta stage value of the solution u. The integrals are evaluated with the same quadrature points as in the time advancement scheme, and the numerical entropy fluxes are based on the boundary extrapolated data already computed to advance the solution in time. Here, we will use a variant of the Local Lax Friedrichs formula. For more details, see [3, 5]. For alternative error indicators, see for example [6] and [7].

3. **An example.** We consider a linear advection problem $u_t + a(x,y)u_x + b(x,y)u_y = 0$, in which a bump rotates around the origin. The computational domain is the square $[-2,2] \times [-2,2]$. The initial condition is

$$u_0(x,y) = (1 - 4r^2)^3 \chi_{(r \le \frac{1}{2})}, \qquad r = ||(x,y) - (0.75, 0.75)||_2,$$

that is u_0 is a peaked hump, centered in the top right quadrant, that is prolongated to zero with a C^2 degree of smoothness. Note that $\max(u_0) = 1$, and that the maximum is quite narrow. The rotation field is $a(x,y) = -2\pi y$, $b(x,y) = 2\pi x$. In this fashion, the bump completes a full rotation in 1 unit of time. During its

rotation, the hump will intersect the grid with no predefined angle. We devised this test to remove any effect due to a biased alignement of the grid with the data, which often pollutes results with cartesian meshes.

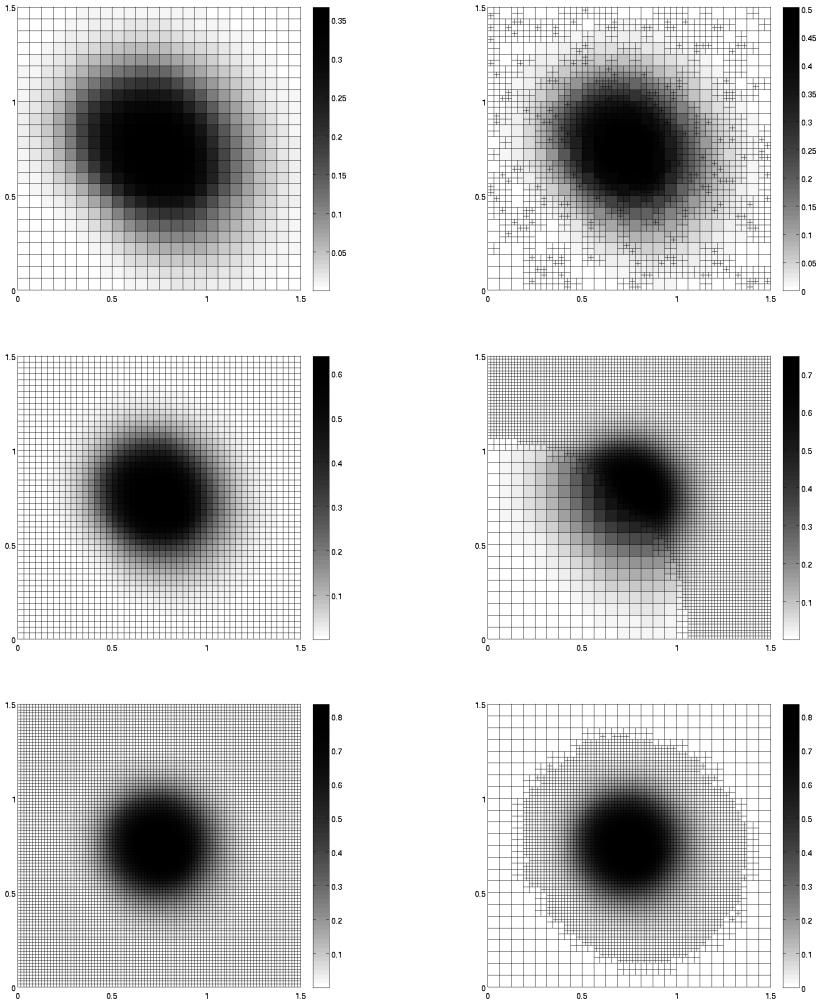

FIGURE 2. Top right quadrant of the solution after one complete rotation. Left column: solution computed on a uniform grid, for decreasing values of h (from top to right). Right column: solution computed on several non-uniform grids.

Fig. 2 shows the solution obtained after one complete rotation. The column on the left shows the solution obtained on a uniform grid, with $h = 4/2^l, l = 4, 5, 6$ from top to bottom. From the color bar we note that the height of the peak decreases slightly faster than linearly, as a consequence of the artificial diffusion induced by the MinMod limiter. Moreover, the effect of the numerical error is also apparent in the deformation of the peak in the direction of its advection, especially on the coarse grid. The right column contains the solution obtained on several non

uniform grids based on 3 levels of refinemnt. In the figure at the top the level of refinement is chosen randomly (grid R), and naturally this provides the worst case. The shape of the bump is similar to the coarse grid case, although the spreading is more contained. Note also that the height of the peak is less smeared than on the coarse grid. In the middle row, the grid is refined outside a circle of radius 0.75, so that there is a always discontinuity in the grid size at the center of the hump during the whole revolution (grid C). Here one can note that there is no apparent distortion due to grid effects (the hump preserves its shape, although of course it is more diffused in the portion solved by the coarse grid). Finally, the figure at the bottom is obtained with the adaptive grid algorithm. Here the patch on which the grid is refined travels with the hump, and the quality of the solution is the same as the one provided when the uniform fine grid is used throughout (grid A).

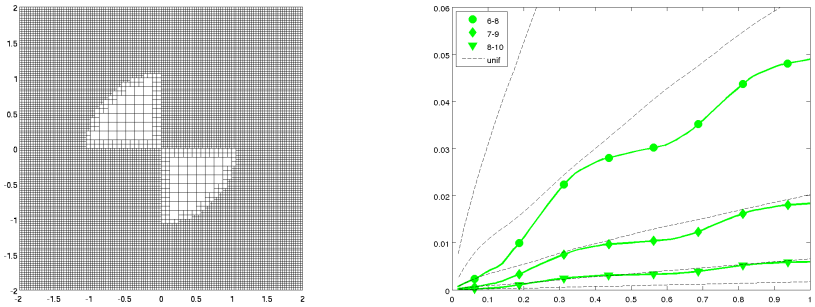

FIGURE 3. Error as a function of time on several grids. The dashed lines refer to uniform grids, while the green curves refer to non uniform grids with the structure illustrated on the left of the figure.

Fig. 3 shows the behavior of the error as a function of time when the grid has the shape seen in the left part of the figure. The dashed lines correspond to the error computed on several uniform grids with mesh size $h = 4/2^l$: from the top the bottom curve (which corresponds to the minimum error) $l = 6, \dots 10$. The green curves are the errors obtained on the grid shown on the left, and they involve three levels of refinement, i.e. $h = 4/2^l, 4/2^{l+1}, 4/2^{l+2}$. Here $l = 6$ for the curve at the top, while $l = 8$ for the curve at the bottom. The behavior of the green curve shows clearly that the error grows faster while the hump crosses the coarse patches in the grid, while it grows slowly when the patch is located on the refined parts. In any case, the behavior of the error remains smooth, with no oscillations or accuracy losses due to the irregularity in the grid.

Finally, Fig. 4 contains the behavior of the error with time for the solutions obtained on the three non uniform grids shown on the right of Fig. 2. The dashed lines refer to results obtained on uniform grids, as in the previous figure. The green curves are obtained on each of the grids illustrated in Fig. 2, but they involve cells with different levels of refinement, as in the previous figure. Clearly the random grid R provides the highest error, which is quite close to the one corresponding to its coarsest grid. It is noteworthy that the adaptive grid yields an error which almost coincides with the error found on its finest grid, although the number of grid points is much smaller.

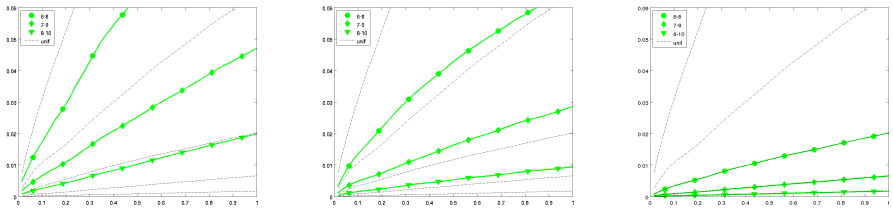

FIGURE 4. Error versus time for the three grids shown in Fig. 2, for different levels of refinement. From left to right, grids R, C and A. The dashed lines refer to uniform grids, while the green curves refer to solutions obtained on non uniform grids.

Acknowledgments. All simulations were performed with `dune-fv`, a module for computing solutions of conservation laws written by the second author and depending on the DUNE [8] and the ALUGRID [9] libraries. The source code (GPL licence) is available from the web-pages [8, 9] and upon request to the second authos (for `dune-fv`).

REFERENCES

[1] F. Chantalat, C.H. Bruneau, C. Galusinski, A. Iollo, *Level-set, penalization and cartesian meshes: A paradigm for inverse problems and optimal design*, J. Comp. Phys., **228** (2009), 6291-6315.

[2] A. Coco, G. Russo, *Second order multigrid methods for elliptic problems with discontinuous coefficients on an arbitrary interface, I: One dimensional problems*, Numerical Mathematics, **5** (2012), 19-42.

[3] G. Puppo, M. Semplice, *Numerical entropy and adaptivity for finite volume schemes*, Comm. Comp. Phys., **10** (2011), 1132-1160.

[4] M. Berger and R. Leveque. *Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems.* SIAM J. Numer. Anal., **35** (1998), 2298-2316.

[5] G. Puppo *Numerical Entropy Production for Central Schemes*, SIAM J. Sci. Comp. **25**, (2003), 1382-1415.

[6] S. Karni and A. Kurganov. *Local error analysis for approximate solutions of hyperbolic conservation laws.* Adv. Comput. Math., **22** (2005), 79–99.

[7] D. Kröner and M. Ohlberger. *A posteriori error estimates for upwind finite volume schemes for nonlinear conservation laws in multidimensions.* Math. Comp., **69** (2000), 25–39.

[8] P. Bastian et al. A Generic Grid Interface for Parallel and Adaptive Scientific Computing. Part I and Part II *Computing*, **82**(2008), 103–119 and 121–138. See also http://www.dune-project.org.

[9] A. Burri, A. Dedner, R. Klöfkorn, and M. Ohlberger. An efficient implementation of an adaptive and parallel grid in dune. *Notes on Numerical Fluid Mechanics*, 91:67–82, 2006. See also http://aam.mathematik.uni-freiburg.de/IAM/Research/alugrid.

E-mail address: gabriella.puppo@uninsubria.it
E-mail address: matteo.semplice@unito.it