3D Adaptive Central Schemes: part I
Algorithms for Assembling the Dual Mesh

S. Noelle[†], W. Rosenbaum[†], and M. Rumpf[‡]

supported by DFG-SPP, ANumE

Abstract. Central schemes are frequently used for incompressible and compressible flow calculations. The present paper is the first in a forthcoming series where a new approach to a 2nd order accurate Finite Volume scheme operating on cartesian grids is discussed. Here we start with an adaptively refined cartesian *primal grid* in 3D and present a construction technique for the staggered *dual grid* based on L^∞-Voronoi cells. The local refinement constellation on the primal grid leads to a finite number of uniquely defined *local patterns* on a primal cell. Assembling adjacent local patterns forms the dual grid. All local patterns can be analysed in advance. Later, running the numerical scheme on staggered grids, all necessary geometric information can instantly be retrieved from lookup-tables. The new scheme is compared to established ones in terms of algorithmic complexity and computational effort.

Keywords: 3D staggered grids, adaptive cartesian meshes, hyperbolic conservation laws, finite volumes, central schemes

1 Introduction

CFD simulations, especially in 3D, produce large amounts of data. In order to minimize memory requirements and computing time without sacrificing high spatial resolution in regions of interest modern numerical schemes are usually based on adaptive grids. Because of their simple structure and the ease of data access, cartesian grids are particularly popular.

Staggered grids are a pair of meshes of the same computational domain whose nodes (in 1D), edges (in 2D) and faces (in 3D) do not coincide. While the shape of a staggered grid is canonical on an uniform mesh it becomes rather complicated for an underlying adaptively refined grid, in particular in three space dimensions. In the present paper we suggest a new construction technique for a staggered grid which is dual to an adaptive cartesian grid.

Our focus of interest is the class of central schemes for hyperbolic conservation laws, which became quite popular during the last decade. In this case, staggered grids circumvent the possibly costly numerical solution of Riemann problems. The prototype of all central schemes is the first-order Lax-Friedrichs scheme [9].

[†]RWTH Aachen, Germany
[‡]University of Duisburg-Essen, Germany

Higher order upgrades were first introduced by Nessyahu and Tadmor [14], and developed further by many authors (see [1, 2, 3, 6, 10, 11, 12] and the references therein). In particular, Arminjon and St-Cyr ([1, 2]) extended the method both onto unstructured tetrahedral and uniform cartesian grids in 3D.

Let us briefly summarize our new staggered grid construction. The cells of the dual grid are defined as the L^∞-Voronoi regions around the vertices of the primal grid. Even though it is in principle simple to construct these Voronoi regions, it becomes a very complex task if the primal grid is adaptively refined, especially in three spcae dimensions. Our approach, motivated by previous work on the 2D case [15], is based on a local decomposition of each primal cell into dual subcells. We call this decomposition of a primal cell the *local pattern*. The dual subcells are later assembled at the vertices. Since we use a cartesian grid with graded refinement, the number of different local decompositions is bounded. In [16], we used Pólya theory to show that there are 227 combinatorially different local patterns in 3D. We summarize this counting in the appendix. All these patterns are analysed and stored in advance. Later, running the numerical scheme on staggered grids, all necessary geometric information can instantly be retrieved from lookup-tables.

At this stage we have implemented a first order staggered finite volume scheme on adaptive cartesian grids. For the first time in this paper, we are able to present numerical results in 3D. Currently, we are developing the ingredients necessary for a second order Nessyahu-Tadmor type extension of our 3D-scheme.

A popular alternative dual grid construction on adaptive cartesian grids is based on "diamond cells", developed for example in [2, 7]. Although this geometrical description is very simple, we show in Section 2 that diamond grids increase the numerical cost of the finite volume scheme considerably.

The paper is organized as follows: In Section 2 we investigate Voronoi decompositions with respect to several norms and compare them with diamond dual grids. In Section 3 we construct local patterns. In order to explain the technique as clearly as possible, we perform the construction first for 2D grids, and generalize it later to 3D grids. Section 4 presents first numerical applications in 3D, discusses algorithmic complexity and poses some open questions to be treated in forthcoming papers.

2 Problem setting and concept

In this section we introduce the necessary notation, formulate properties of the primal and the corresponding dual grid and explain the basic idea of the dual grid construction. We discuss both the popular "diamond cell construction" as well as the Voronoi decomposition with respect to different norms in some detail and argue for our favorite choice.

Notation. Throughout this paper we use the capital letters G, C, F, E and N for a *grid, cell, face, edge* resp. *node*, capital calligraphic letters to denote *sets*, like $\mathcal{N} = \bigcup N$, subscripts p and d to distinguish between *primal* and *dual* objects, an asterisk * to denote objects from the corresponding staggered grid (the dual grid is staggered to the primal grid and vice versa), subscripts i, j and k for node indices, and the superscript $^+$ to label local objects on single cells.

Finite volume schemes. Our primary interest of constructing dual grids is directed towards the numerical solution of systems of conservation laws

$$u_t(x,t) + \operatorname{div} F(u(x,t)) = 0 \tag{1}$$

in three space dimensions. Here, $x \in \mathbb{R}^3$ denotes the space variable, t the evolution time, $u(x,t) \in \mathbb{R}^m$ the solution vector in terms of m conservative variables, and $F = (f_i)_{i=1}^3$ the vector of the three directional flux functions $f_i : \mathbb{R}^m \to \mathbb{R}^m$, $i = 1,2,3$.

The final goal is to implement a 3D-extension of the second order accurate central scheme proposed in [14]. A dimensionless formulation of the finite volume scheme reads as follows:

Let $v(x,t)$ be a cellwise smooth approximate solution of (1) with cell averages

$$\bar{v}_C^n = \frac{1}{|C|}\int\limits_C v(x,t^n)\,dx. \tag{2}$$

The standard finite volume update is

$$\bar{v}_C^{n+1} = \bar{v}_C^n + \frac{1}{|C|}\int\limits_{t^n}^{t^{n+1}}\int\limits_{\partial C} F(v(x,t))\cdot \mathfrak{n}\;dx\,dt \tag{3}$$

Note that the flux integral should be replaced by a quadrature rule, and the approximate solution v needs to be extrapolated in time (for a fully discrete scheme). Since the solution v may be discontinuous at the cell boundary ∂C, one would have to replace the flux function $F(v)$ by a Riemann solver. This can be avoided when using staggered grids: now mean values on the timelevel t^{n+1} are computed on cells C^* of the corresponding staggered grid via

$$\bar{v}_{C^*}^{n+1} = \frac{1}{|C^*|}\sum_{C\cap C^*\neq\emptyset}\left\{\underbrace{\int\limits_{C\cap C^*} v(x,t^n)\,dx}_{I_1} + \underbrace{\int\limits_{t^n}^{t^{n+1}}\int\limits_{C\cap\partial C^*} F(v(x,t))\cdot\mathfrak{n}\;dx\,dt}_{I_2}\right\} \tag{4}$$

As before, one needs quadrature rules for the integrals and time extrapolation for v. For the flux integral I_2 this is possible in such a way that $F(v)$ needs to be computed only at points (x,t) where v is continuous [14, 6, 11]. Therefore, there is no need to solve Riemann problems for central schemes. Still, the evaluation of the flux function may be the most expensive part of the finite volume scheme. Therefore, the numerical scheme should minimize the number of quadrature points in I_2.

Grid adaptation. The cartesian *primal grid* G_p is supposed to be an adaptively refined 3D cartesian mesh. We restrict ourselves to the regular subdivision of a hexahedron into eight child hexahedra. The grid adaptation is subject to a 1-level transition constraint between cells which share a common face or a common edge, see Figure 1. Any refinement technique which observes these rules would be appropriate, as for instance AMR [4], the cartesian refinement based on saturated error indicators [17], or multi-scale analysis [13].

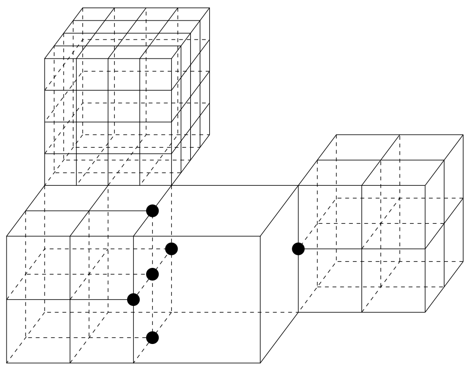


Figure 1: 1-level transition constraint for the primal 3D-grid

Aim. Given a primal grid G_p we are now looking for a corresponding staggered *dual grid* G_d. The numerical algorithms, and in particular those of finite volume type, require at least:

- G_p and G_d are meshes for the same computational domain Ω.
- Interior nodes, edges and faces of G_p and G_d do not coincide.

Moreover, the dual grid G_d should locally reflect the resolution of the primal grid G_p. With respect to the limitation of the timestep size by the CFL-number and the distance of discontinuities in the numerical solution one should also try to maximize the distance between faces of primal and dual cells.

Diamond vs. Voronoi grids. Before we present our new approach for the construction of the staggered grid, we first discuss the well known “diamond grids” (cf. [2, 7]). For reasons of clarity only the 2-dimensional figures are presented. All arguments are formulated also for 3-dimensional grids. The steps to construct dual cells read as follows:

1. Construct edges of dual cells simply by connecting the midpoint of a primal cell C with all of C's corners (see Figure 2(a)). In 2D these edges subdivide C into four triangular dual cell parts, whereas in 3D the dual edges span 12 dual faces which bound six pyramidal dual cell parts.
2. Dual cells result by sticking together adjacent dual cell parts with a common primal face (see Figure 2(b)).

This construction technique is impressively simple, but suffers from excessive numerical cost:

- Since every primal grid cell is subdivided into four (in 2D) resp. six (in 3D) dual cell parts, and a dual cell is normally composed by two cell parts the dual grid contains approximately twice resp. thrice as much cells as the corresponding primal grid.

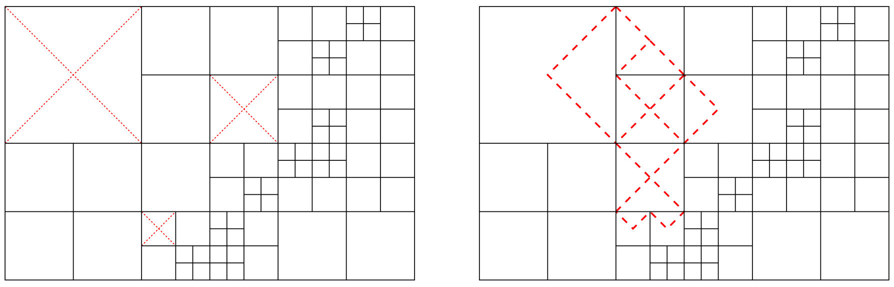

(a) constructing dual edges (b) resulting dual diamond cells

Figure 2: Diamond-type dual cell construction in 2D

- Following the algorithm of Nessyahu and Tadmor from [14] two kinds of integration have to be performed in (4) to determine the numerical solution on the next time level. The far more expensive part is I_2, the integration of flux functions over the boundary of cells. Since the cellwise representation of the numerical data is in general discontinuous at the cell boundaries, the fluxes have to be evaluated in the interior of primal cells (e.g. in the center of gravity of the dual faces in the case of piecewise linear data. Its evaluation in the nodes of the dual cells is not possible since they often coincide with the boundary of primal cells). Since there are four (in 2D) resp. twelve (in 3D) dual faces per primal cell the number of necessary flux evaluations for a timestep onto the dual grid sums up to:

$$\begin{aligned} \#\ \text{fluxes(2D)} &= \ \ 4|\mathcal{C}(G_p)| \\ \#\ \text{fluxes(3D)} &= 12|\mathcal{C}(G_p)| \end{aligned} \tag{5}$$

- Due to the finer resolution of the dual grid the diameter of a dual cell is by factor $\sqrt{2}$ smaller than the diameter of its corresponding primal cell. This leads to a restriction of the global timestep size in the explicit time integration procedure.

Our alternative approach is described by the rule:

1. Construct dual cells always surrounding exactly one primal node N_p by a Voronoi decomposition of the domain Ω respecting all nodes $\mathcal{N}(G_p)$ of the primal grid G_p.

This promises to be less expensive than the Diamond grids because:

- Nodes of dual cells now lie in the interior of primal cells. Therefore we can apply the trapezoidal rule for integration in space with dual nodes as quadrature points. The flux function has to be evaluated there with respect to every spatial dimension. Since in a cartesian grid there are approximately as many nodes as cells, the primal and dual grid are of comparable size. Moreover, since the structure of the primal grid is transfered onto the dual grid in most parts, the number of dual nodes is only slightly larger than the number of primal cells. Thus the number of fluxes to be evaluated amounts to:

$$\begin{aligned} \# \text{ fluxes(2D)} &= 2|\mathcal{N}(G_d)| \approx 2|\mathcal{C}(G_p)| \\ \# \text{ fluxes(3D)} &= 3|\mathcal{N}(G_d)| \approx 3|\mathcal{C}(G_p)| \end{aligned} \tag{6}$$

which is considerably less expensive than the diamond cell approach discussed above.

The cartesian structure of the primal grid and its adaptation constraints (regular refinement, 1-level transition) lead to a rigidly prescribed location of the primal nodes. This allows the Voronoi decomposition to be performed *locally* on every cell C_p of the primal grid (as it will be proved in the next chapter). Depending on the set of primal nodes on the boundary of a primal cell C_p, $\mathcal{N}^+(C_p) := \mathcal{N}(G_p) \cap C_p$, we deduce C_p's decomposition into local Voronoi regions. All these local regions together form a *local pattern* on C_p. Local patterns on adjacent cells of the primal grid automatically match. All local Voronoi regions corresponding to a fixed node N_p of the primal grid finally form the dual cell $C_d(N_p)$.
Due to the primal grid structure and the 1-level transition constraint the number of *essentially different* local patterns is finite, and every possible decomposition can be analysed in advance. Later, the numerical scheme gets instant access to these predefined patterns and uses them in scaled and rotated copies. These copies are retrieved at run time and do not have to be stored.

Voronoi regions: choice of the norm

Given a set $\Omega \subset \mathbb{R}^d$ and a finite set of nodes $\mathcal{N} = \{N_k, k = 1, \ldots, m \,|\, N_k \in \Omega\}$, we decompose Ω into m Voronoi regions V_{N_k} by the following conditions:

(i) $\bigcup_{k=1}^m V_{N_k} = \Omega$

(ii) $\dot{V}_{N_i} \cap \dot{V}_{N_j} = \emptyset, i \neq j$

(iii) $\|x - N_k\| \leq \|x - N_j\|, x \in V_{N_k}, \forall j \neq k.$

Here $\dot{V}_{N_i}$ denotes the interior of V_{N_i}, and $\|\cdot\|$ an arbitrary norm on $\mathbb{R}^d$.
Let us now discuss the choice of an appropriate norm in 2D more in detail. The same arguments hold also for 3D.
A Voronoi decomposition of $\Omega \subset \mathbb{R}^2$ refering to *only two* nodes N_i resp. $N_j \in \Omega$ splits Ω into two Voronoi regions, $V_{N_i(N_j)}$ (around N_i) and $V_{N_j(N_i)}$ (around N_j), sharing a common *separating polygon* S_{ij}. This separating polygon is the locus of the intersection points of circles with same diameter centered at N_i resp. N_j. The shape of the separating polygon depends both on the position of N_i and N_j as well as on the norm on $\mathbb{R}^2$. In general, a Voronoi region V_{N_i} around the node N_i takes the form

$$V_{N_i} = \bigcap_{N_j \in \mathcal{N}} V_{N_i(N_j)} \tag{7}$$

Respecting the ordinary $\|\cdot\|_2$-norm, any S_{ij} is a straight line, hence Voronoi regions V_{N_i} are always convex. In contrast, for the $\|\cdot\|_\infty$-norm, S_{ij} consists, in general, of three straight lines, which are aligned with the cartesian axes (x-axis, y-axis) or the diagonals (of the xy-plane), see Figure 3(a). For special positions of N_i and N_j the separating polygon simplifies to a straight line (see Figure 3(b)), or becomes non-unique in regions away from N_i and N_j (see Figure 3(c)). For the latter case we choose the prolonged straight line from the unique part as the separating polygon. Moreover, Voronoi regions V_{N_i} might happen to be non-convex.

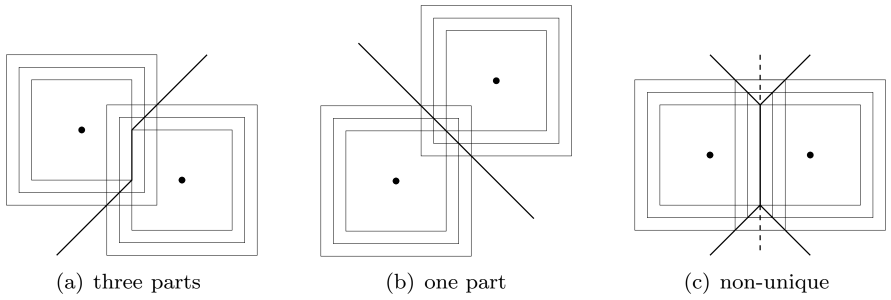

(a) three parts (b) one part (c) non-unique

Figure 3: Different configurations of two points, some $\|\cdot\|_\infty$-norm circles around these points, and the resulting separating polygons in 2D

However, thanks to the cartesian structure of the primal grid the $\|\cdot\|_\infty$-norm even simplifies the construction of local Voronoi regions on cells of the primal grid. Compared to the corresponding construction for the Euclidean norm, boundary faces of Voronoi regions are now aligned only with the axes or the plane diagonals of the primal grid (cf. Figure 4(b) vs. Figure 4(a)). We therefore favour the $\|\cdot\|_\infty$-norm.

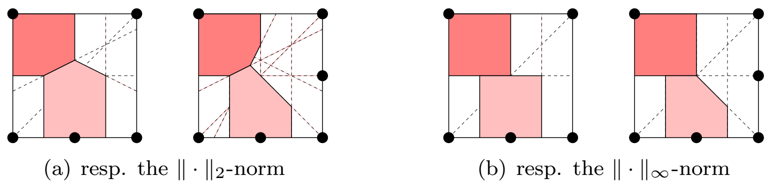

(a) resp. the $\|\cdot\|_2$-norm (b) resp. the $\|\cdot\|_\infty$-norm

Figure 4: Comparison of Voronoi regions and separating polygons on a primal cell in 2D

Local Voronoi construction

Next, we show that the Voronoi regions V_{N_i} can be constructed locally over each cell. This will simplify the algorithmic implementation considerably. We call

$$V_{N_i}^{C_p} := C_p \cap \bigcap_{N_j \in \mathcal{N}(C_p)} V_{N_i(N_j)} \tag{8}$$

the bounding box of a local Voronoi region V_{N_i} on a primal grid cell C_p. $\mathcal{N}(C_p)$ denotes the set of corners of the cell C_p. Figure 5 illustrates the bounding boxes resulting by intersection (9) on a primal cell in 3D. Here, N_i is a corner node (Figure 5(a)), a hanging node in the midpoint of an edge (Figure 5(b)) resp. a hanging node in the midpoint of a face (Figure 5(c)) of C_p.

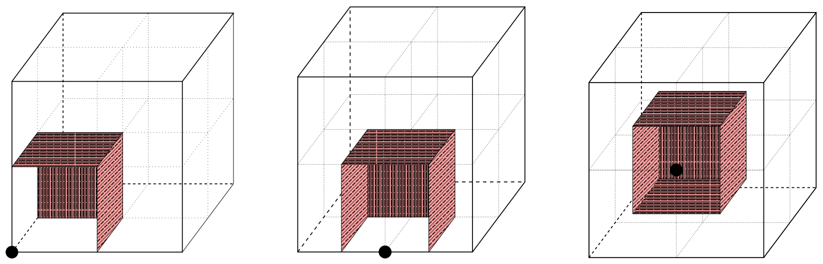

(a) corner node (b) edge node (c) face node

Figure 5: Bounding boxes $V_{N_i}^{C_p}$ for local Voronoi construction in 3D

With (7) we get the relation

$$V_{N_i}^{C_p} \supseteq C_p \cap V_{N_i} \tag{9}$$

In order to get the final shape of a local Voronoi region V_{N_i} on C_p it suffices to perform the intersection (7) for all nodes $N_j \in \mathcal{N}^+(C_p)$, i.e. respecting only the nodes on the boundary of C_p. Shorter, we state

Lemma 1. *Let*

$$(V_{N_i}^{C_p})^+ := C_p \cap \bigcap_{N_j \in \mathcal{N}^+(C_p)} V_{N_i(N_j)} \tag{10}$$

be the restriction of the bounding box $V_{N_i}^{C_p}$ due to the hanging nodes on cell C_p. Then

$$C_p \cap V_{N_i} = (V_{N_i}^{C_p})^+ \tag{11}$$

Proof. Assuming Lemma 1 were false. Then we have

$$C_p \cap V_{N_i} \subsetneq (V_{N_i}^{C_p})^+ \tag{12}$$

i.e.

$$C_p \cap \bigcap_{N_j \in \mathcal{N}(G_p)} V_{N_i(N_j)} \subsetneq C_p \cap \bigcap_{N_j \in \mathcal{N}^+(C_p)} V_{N_i(N_j)} \tag{13}$$

hence $\exists N_k \in \mathcal{N}(G_p) \setminus \mathcal{N}^+(C_p)$ such that $(V_{N_i}^{C_p})^+ \setminus V_{N_i(N_k)} \neq \emptyset$.
Since $(V_{N_i}^{C_p})^+ \subseteq V_{N_i}^{C_p}$ by (10), we also have $V_{N_i}^{C_p} \setminus V_{N_i(N_k)} \neq \emptyset$, which implies

$$\exists\, x \in V_{N_i}^{C_p} \text{ with } \|x - N_k\|_\infty < \|x - N_i\|_\infty \tag{14}$$

i.e. x should not be assigned to V_{N_i}.
Let $\mathfrak{H}_{N_i}(C_p) := \bigcup_{y \in V_{N_i}^{C_p}} B_{\|y-N_i\|_\infty}(y)$. Now, by (14) follows

$$N_k \in \mathfrak{H}_{N_i}(C_p) \tag{15}$$

With $\Delta x = \operatorname{diam}(C_p, \|\cdot\|_\infty)$ we define $\mathfrak{U}_{N_i}(C_p) := \bigcup_{y \in V_{N_i}^{C_p}} B_{\frac{\Delta x}{2}}(y)$ and

$$\mathfrak{H}^+_{N_i}(C_p) = \begin{cases} \mathfrak{U}_{N_i}(C_p) \setminus \hat{C}_p(N_i) & \text{if } N_i \text{ is corner node of } C_p \\ \mathfrak{U}_{N_i}(C_p) & \text{else} \end{cases}$$

where $\hat{C}_p(N_i)$ denotes that neighbour of C_p which shares only the node N_i, i.e. $C_p \cap \hat{C}_p(N_i) = N_i$.

Suppose that we are in the case of Figure 5(a). Let $z \in \hat{C}_p(N_i)$ be fixed. Then we have $\|y - N_i\|_\infty < \|y - z\|_\infty$ for any $y \in V_{N_i}^{C_p}$. Hence there is no $y \in V_{N_i}^{C_p}$ such that $z \in B_{\|y - N_i\|}(y)$, thus $\hat{C}_p(N_i) \cap \mathfrak{H}_{N_i}(C_p) = \emptyset$.

In general, since $\|y - N_i\|_\infty \leq \frac{\Delta x}{2}$ $\forall y \in V_{N_i}^{C_p}$, the relation $\mathfrak{H}_{N_i}(C_p) \subseteq \mathfrak{H}^+_{N_i}(C_p)$ holds for every case of the Figures 5(a), 5(b) and 5(c). Due to the 1-level grading condition over faces and edges of C_p the only grid nodes in the interior of $\mathfrak{H}^+_{N_i}(C_p)$ are the possibly occuring hanging nodes on C_p's boundary. Hence

$$\mathfrak{H}^+_{N_i}(C_p) \cap \mathcal{N} \setminus \mathcal{N}^+(C_p) = \emptyset \tag{16}$$

i.e. there is no N_k fulfilling condition (15). This proves Lemma 1. □

Remark 1. *The special treatment of the case 5(a), where N_i is a corner of C_p, is necessary since the 1-level grading condition over faces and edges still allows a 2-level transition over nodes. In this case $\mathfrak{U}_{N_i}(C_p)$ and $\mathcal{N} \setminus \mathcal{N}^+(C_p)$ would share a node of $\hat{C}_p(N_i)$, and* (16) *would not hold.*
Remark 2. *The arguments in the proof do not depend on the spatial dimension of C_p. However, for a better understanding the reader might prefer to draw sketches of the proof in the 2-dimensional setting.*

3 Local pattern construction

Due to Lemma 1 we can perform the L^∞-Voronoi decomposition locally on every primal cell C_p. The construction process of local Voronoi regions is described by the right-hand side of (11). A decomposition of C_p with respect to all nodes $N_k \in \mathcal{N}^+(C_p)$ is called the *local pattern* on C_p.

In order to prepare the reader for the pattern construction in 3D, which is the focus of this paper, we pause for a moment and study the 2D case for illustration.

Construction in 2D. Depending on the distribution of hanging nodes on the primal cell C_p, every local Voronoi region takes one of the six shapes shown in Figure 6.

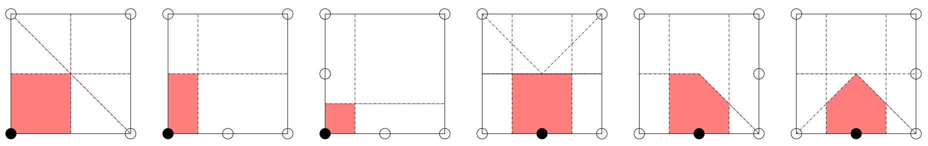


Figure 6: Local Voronoi regions on a primal cell in 2D

Obviously, all cuts of the cell C_p run parallel to the axes or coincide with the plane diagonals. We could therefore describe the steps for the construction of local Voronoi regions on C_p equivalently by the following algorithm:

1. Subdivide the cell C_p into 16 congruent *squares*.
2. Subdivide the four squares containing the midpoint of C_p into two *triangles*. The cuts follow the face diagonals through C_p's midpoint. On C_p we get 8 triangles.

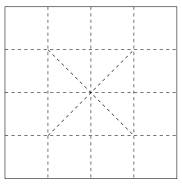


Figure 7: Subdivision of a primal cell in 2D

   These squares and triangles, depicted in Figure 7, will be called *atoms* of C_p.

3. Finally, we assign these 20 atoms of C_p to the nodes of $\mathcal{N}^+(C_p)$ simply by calculating the $\|\cdot\|_\infty$-distance between the nodes and the atom's center of gravity. All atoms which are assigned to the same node $N_k \in \mathcal{N}^+(C_p)$ form the *local Voronoi region* $V_{N_k}(C_p)$ on C_p. The set of all local Voronoi regions on C_p forms the *local pattern*. Figure 8 shows all six essentially different local patterns in 2D.

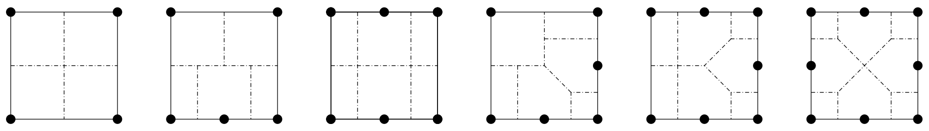


Figure 8: All essentially different local patterns in 2D

Assembling scaled and rotated copies of appropriate local patterns leads to the dual grid. An example is depicted in Figure 9.

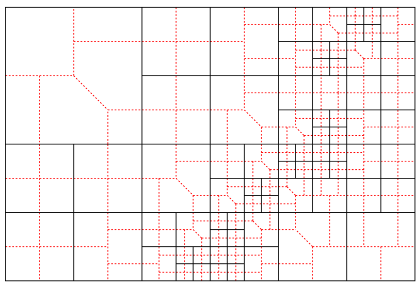


Figure 9: 2D cartesian primal grid (solid) and corresponding dual grid (dashed)

Construction in 3D. Now we extend the steps to create local patterns described above to 3-dimensional grids. Applying the concept of constructing local Voronoi regions leads to an only slightly more complex description than in 2D. The set $\mathcal{N}^+(C_p)$ now contains the eight corner nodes of C_p, and up to 18 possible hanging nodes on the midpoints of C_p's 6 faces and 12 edges. For a general location of two nodes N_i and N_j in $\mathbb{R}^3$ the *separating surface* between the Voronoi regions $V_{N_i(N_j)}$ and $V_{N_j(N_i)}$ consists of up to seven planar patches, whose normals are aligned with the axes (x-, y-, z-axis) or the plane diagonals (xy-, yz-, zx-plane). Fortunately, the variety of positioning two nodes $N_i, N_k \in \mathcal{N}^+(C_p)$ relatively to each other is limited. Hence we get only six essentially different separating surfaces in 3D, depicted in Figure 10. Note that the cases in Figure 10 do not necessarily represent cells of the primal grid. Finally, on a primal cell only scaled, translated and rotated versions of these cuts have to be executed to get the local Voronoi regions.

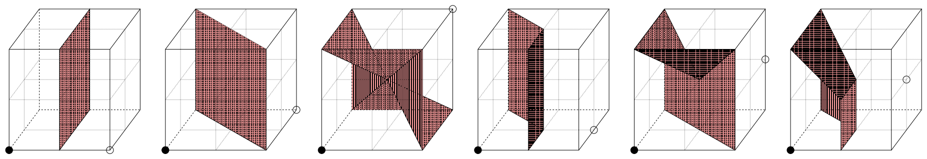


Figure 10: Separating surfaces in 3D

Hence the whole construction can again be described as a subdivision algorithm:

1. Subdivide the hexahedral cell C_p into 64 congruent *cubes*.

2. Subdivide all cubes containing a midpoint of C_p's faces into two *prisms*. The cuts follow the face diagonals through the face midpoints. On C_p we get $8 \times 6 = 48$ prisms.

3. Subdivide all cubes containing the central point of C_p into six *tetrahedra*. The corresponding three cuts equal the diagonal cuts on their adjacent cubes from step 2. This results in $6 \times 8 = 48$ tetrahedra.

These cubes, prisms and tetrahedra will again be called *atoms* of C_p. Figure 11(a) shows a cell C_p and some of its atoms. Next we assign these $64 + 48 - 24 + 48 - 8 = 128$ atoms of C_p to the nodes of $\mathcal{N}^+(C_p)$ by calculating the $\|\cdot\|_\infty$-distance between the nodes and the atom's center of gravity. All atoms which are assigned to the same node $N_k \in \mathcal{N}^+(C_p)$ form the *local Voronoi region* $V_{N_k}(C_p)$ on C_p. Figure 11(b) shows a cell C_p, the set $\mathcal{N}^+(C_p)\backslash\mathcal{N}(C_p)$ (i.e. the corners of C_p left out) and a few corresponding local Voronoi regions. Let us mention that local Voronoi regions might happen to be slightly non-convex (region in the center of Figure 11(c)). But this should not effect the feasibility of our approach, and we do not expect additional problems in constructing a second order central finite volume scheme by the occurrence of these dual cells. The set of all local Voronoi regions on C_p form the *local pattern*. The shape of these three-dimensional local patterns on the boundary faces of a primal cell matches the two-dimensional patterns from Figure 8.

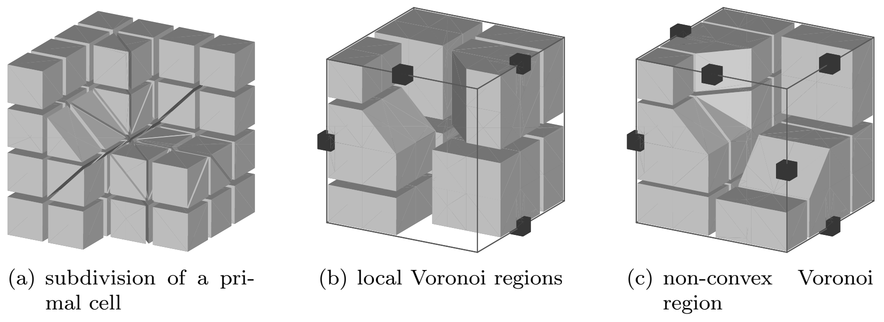

(a) subdivision of a primal cell (b) local Voronoi regions (c) non-convex Voronoi region

Figure 11: Construction of local Voronoi regions in 3D

The set of local patterns in 3D

Since the number of different distributions of nodes on the boundary of a primal cell, and hence the number of different local patterns, is finite, all local patterns can be assembled and stored in advance. Later, running the numerical scheme, these patterns are used in scaled and rotated copies.
For grids in 2D the set of essentially different local patterns (that do not arise from each other by rotation or reflection) can still easily be determined (cf. Figure 8). This task becomes far more complex on grids in 3D. We used tools of Pólya's counting theory ([18], [8]) to determine the number of 227 essentially different local patterns. Details of this counting are presented in the appendix. A complete list of all patterns (some of them with figures) can be found on
`http://www.igpm.rwth-aachen.de/wolfram/local_patterns.html`
Every local refinement situation can be identified with exactly one of these 227 *reference patterns* and an appropriate mapping (i.e. permutation matrix) onto the really occuring pattern.

4 Staggered Finite Volume schemes

The final goal of our work is a second order accurate finite volume scheme on staggered grids. At the current stage, our code includes the primal and dual grid generation, as well as the integration of piecewise linear functions over volumes and boundary faces of cells. These ingredients already suffice for a first order scheme which we have implemented and tested. Further components, like linear data reconstruction and limiting for a second order accurate non-oscillatory method, and the treatment of different boundaries are in progress.

After summarizing the algorithmic demands and presenting our integration strategy, we address the data access, describe our test cases and finally compare the expected numerical effort of our method with the diamond grid approach and the established non-staggered HLL-scheme.

Algorithmic demands

The finite volume update is given by equation (4). The cell $C^* \in G^*$ might belong to the primal grid G_p or to the dual grid G_d constructed in Section 3. Cells C will belong to the staggered grid G.

The evaluation steps in (4) are twofold. Part I_1 calls for an integration of the piecewise linear function $v(\cdot, t^n)$ over a bounded volume and can be evaluated exactly. Part I_2 is a bit more intricate. For an illustration we refer to Figure 12. The normal part of a non-linear flux-function, $F \cdot \mathfrak{n}$, has to be integrated in time over the interval $[t^n, t^{n+1}]$ and in space over a polygonally bounded planar cell face $\mathfrak{p}$. In general, a face $\mathfrak{p}$ of a cell $C^* \in G^*$ consists of several polygonal parts $\mathfrak{p}_i$ in adjacent cells C_i of the corresponding staggered grid G.

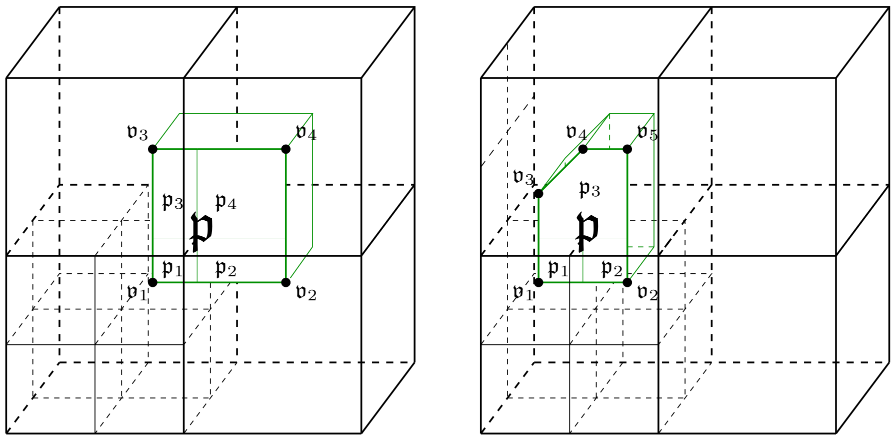
Any second order accurate quadrature formula for I_2 should evaluate F in space-time quadrature points (x, t) where v is uniquely defined. For the temporal integration of I_2 we use the midpoint rule. For that reason v has to be extrapolated to approximate $v(x, t^{n+1/2})$. As time evolves, characteristics spread out from the boundaries of the cells $C_i \in G$ into their interior. In order to avoid the solution of local Riemann problems the quadrature points $(x, t^{n+1/2})$ for the spatial flux integration should not be reached by characteristics emanating from ∂C_i at time t^n. Thus, the quadrature points should be far away from the set $\bigcup_i \partial C_i$ to allow a large timestep.

Though admissible, the choice of quadrature points in the center of gravity of every polygonal part $\mathfrak{p}_i$ would be disadvantageous:

- In a cartesian 3D-grid there are about three times more faces than cells. The majority of polygonal faces consists of four polygonal parts. We would have to evaluate one flux function on every polygonal part.
- The distance of the center of gravity of a polygonal part $\mathfrak{p}_i$ to the discontinuity set $\bigcup_i \partial C_i$ is rather short. This limits the global timestep size unnecessarily.

This leads us to choose the corners of the boundary face $\mathfrak{p}$, i.e. nodes of the grid G^*, as spatial quadrature points. Therefore we have implemented a generalized trapezoidal rule on polygons. The flux function F is evaluated at the nodes of a polygonal face. Later, these values are scaled with barycentric weights respecting the face geometry to achieve a second order accurate integration rule. This approach promises to be more competitive:

- In a cartesian 3D-grid the number of nodes and the number of cells is similar. We evaluate all three directional flux functions at every node $\mathfrak{v}_i$. The numerical effort is by factor ≈ 4 smaller than using the midpoints of the polygonal faces.
- Since the distance of nodes of a polygonal face $\mathfrak{p}$ of a cell C^* to the set $\bigcup_i \partial C_i$ is larger than the distance from the $\mathfrak{p}_i$'s center of gravity to the boundary. The global timestep size can be chosen by factor 2 larger than in the method above.



(a) 4 polygonal parts, 4 vertices (b) 3 polygonal parts, 5 vertices

Figure 12: Compound boundary faces of dual cells

Data access

For fast access all geometric and numerical data, both for the primal and the dual grid, are stored in hashmaps. On every cell C of the grid G the finite volume scheme needs to know

1. the volume of C
2. its neighbouring cells on the grid G (for a higher order reconstruction)
3. the common faces of C and its neighbours, including their normals and their area, as well as the nodes on the grid G that form these faces (to determine fluxes over faces by an appropriate quadrature rule)

On the cells of the cartesian primal grid G_p such information is quite easily accessible by index shift operations. In contrast, for a cell C_d of the dual grid G_d this knowledge is scattered over all local patterns contributing to the construction of C_d and has to be "collected" during a complete traversal of the primal grid G_p. Hence finding the appropriate local pattern on a primal cell is an often used operation and has to be fast.

The local pattern on a primal grid cell G_p is determined by the occurrence of hanging nodes in the midpoints of G_p's 12 edges and six faces (cf. Section 3). These information form an 18-bit key which allows instant access to the corresponding reference pattern (one of 227) as well as the appropriate permutation matrix from a predefined hashmap. This hashmap can easily be initialized in a pre-roll step by applying each of the 48 self-mappings of the cube to all of the 227 essentially different local refinement constellations. Finally, one gets 6210 hashmap entries.

A local pattern consists of a certain number of local Voronoi regions V_{N_k}, each of them composed of several atoms (cubes, prisms and tetrahedra, see Figure 11(a)). We get all information listed above locally:

1. the volume of V_{N_k} equals the sum of the volumes of its atoms

2. neighbouring Voronoi regions on a pattern refer to neighbouring cells on the dual grid G_d
3. common faces of neighbouring Voronoi regions are the union of common faces of their atoms. Thus, the normals, areas and the face-forming nodes are known by construction of the local pattern.

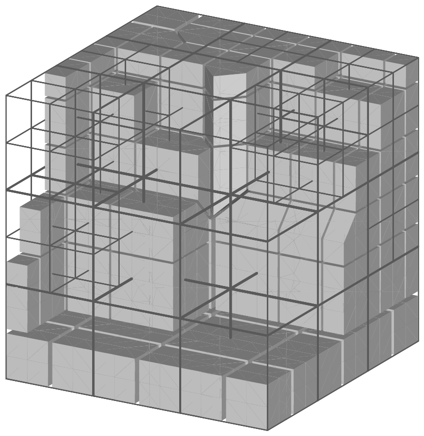
Whereas on the primal grid G_p all cells C_p are cubes and common faces to neighbouring cells are always square-shaped, there is a wider range of different cell- and face-types on the dual grid G_d, see Figure 13(a) for a rough impression.

5 Numerical Experiments

In order to verify the feasibility of our staggered grid approach and to get a deeper insight in the expected complexity of our grids, we run two basic test cases. The first one is a volume integration, and the second a linear advection problem computed with a first order finite volume scheme.

Example I: Volume integration

As discussed in Section 4 the integration over faces and volumes is the essential part of a finite volume scheme. In our first test we performed a *volume integration* over the dual grid G_d. The primal grid has been constructed by adaptively resolving the isosurface of a tilted elliptical paraboloid inside the unit cube, see Figure 13(b).


(a) Close-up view on the dual grid. Lines represent the primal grid.

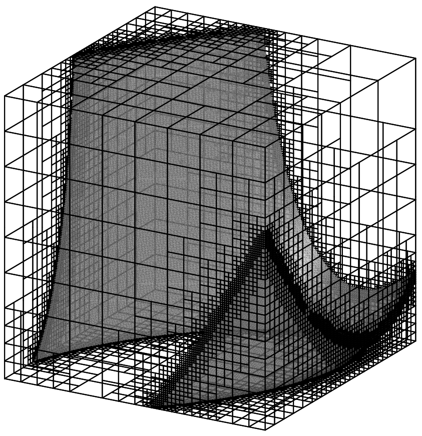

(b) primal test-grid with resolved isosurface on level 7, 112470 cells

Figure 13: Primal and dual grid

We verified our integration algorithm by means of the Gauss integral theorem

$$\int_{\hat{\Omega}} \operatorname{div} \mathfrak{v} \; dx = - \int_{\partial\hat{\Omega}} \mathfrak{v} \cdot \mathfrak{n} \; dx$$

by choosing $\mathfrak{v}(\vec{x}) = \frac{1}{3}\vec{x},\ \forall \vec{x} \in \mathbb{R}^3$, a subdomain $\hat{\Omega} \subseteq \Omega$, and integrating over all cells $C_d \subset \hat{\Omega}$ of the dual grid G_d

1. by summing up the volumes of the involved Voronoi regions
2. by integrating $\frac{1}{3}\langle \vec{x}, \mathfrak{n}_f \rangle$ over all common faces f of neighbouring Voronoi regions (in view of flux integration over cell boundaries). Here $\mathfrak{n}_f$ denotes the oriented normal of f.

We performed the volume integration described above for several resolutions of the primal grid G_p and examined

- the number of different occuring local patterns:
  it leveled off at about 40
- the frequency of occurrence of these patterns:
  approximately 80% of the cells C_p on the primal grid refer to the most simple pattern (no additional node on the boundary of C_p), whose numerical cost are low
- the expected number of flux evaluations for the staggered grid solver by counting the number of nodes in the dual grid

To estimate the profit of our approach, the numerical effort was compared with a cheap non-staggered finite volume scheme where flux integrals are approximated by the Harten-Lax-van Leer (HLL) Riemann solver (cf. [19]). Results are listed in Table 1 below. The numerical cost of both schemes is similar.

level	primal cells	dual cells	fluxes non-staggered	fluxes staggered
4	568	1019	3810	4779
5	4502	6969	30126	33717
6	24781	34710	163899	161001
7	112470	149011	737283	690420
8	493368	630364	3210945	2906742
9	2156547	2684250	13944345	12220641

Table 1: Comparison staggered/non-staggered approach for Exapmle I

Example II: Rotating Cone

Next, we run a standard linear advection problem, the "Rotating cone", with a first order finite volume scheme (note that we have not yet incorporated linear data reconstruction and limiting into our method, which would be essential for a second order scheme). Finally we compare our scheme to the diamond-grid approach as well as to the non-staggered HLL-scheme in terms of numerical effort.

Initial data & numerical solution

As initial data for the rotating cone advection problem we resolve a smooth function f with compact support over an octant of the surface ∂B of a ball

$B = B(c, R)$ (cf. Figure 14(a)) The ball B is centered at $c = (c_x, c_y, c_z) = (0.6, 0.3, 0.2)$ and has a radius $R = 1/4$. The function f is defined as follows:

$$\begin{aligned} r^2 &= (x-c_x)^2 + (y-c_y)^2 + (z-c_z)^2 \\ q &= 4|1-r^2| \\ f(q) &= \begin{cases} 1-2q^2 & \text{for } q < 1/2, x, y, z \geq 0 \\ 2(q-1)^2 & \text{for } 1/2 \leq q \leq 1, x, y, z \geq 0 \\ 0 & \text{else} \end{cases} \end{aligned}$$

This cone is now rotated around c in the plane spanned by $v = (1, 0, 0)^t$ and $w = (0, 1, 0.5)^t$. The advection performs up to time $t = \pi/4$ at constant angle velocity 1. The exact solution is depicted in Figure 14(b). Since our scheme is still only first order accurate the numerical solution suffers from considerable smearing (cf. Figure 14(c)). This would certainly be reduced by a second order scheme, which we plan to develop in the near future.

Grid specifications & numerical effort

Having demonstrated, that a first order finit volume scheme works on our staggered grids, we would now like to extract more information from this experiment. Thus, we compare the number of flux evaluations on the diamond grid, the unstaggered grid and the L^∞-Voronoi grid. These numbers are computed according to Table 2. Results obtained on the grid from Figure 14(c) are listed in Table 3.

timestep	Diamond	HLL	Voronoi
primal $\rightarrow$ dual	12 #primal cells	2 #primal faces	3 #dual nodes
dual $\rightarrow$ primal	1 #primal faces	2 #primal faces	3 #primal nodes

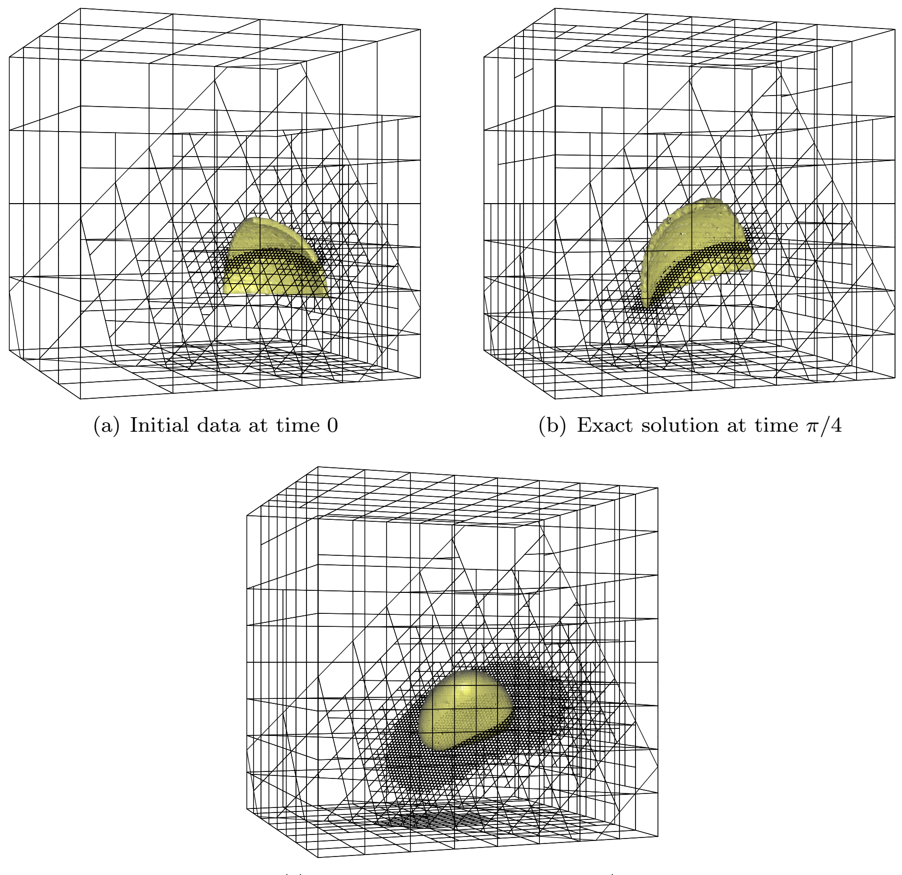
Table 2: Counting flux evaluations in two successive timesteps

Grid specifications	
primal cells	112106
primal faces	345564
primal nodes	121561
dual nodes	152630

# Flux evaluations			
	$\mathbf{G}_p \rightarrow \mathbf{G}_d$	$\mathbf{G}_d \rightarrow \mathbf{G}_p$	$\sum \#f$
Diamond	1.35 M	0.35 M	1.69 M
HLL	0.69 M	0.69 M	1.38 M
Voronoi	0.46 M	0.36 M	0.82 M

Table 3: Grid specifications and numerical effort for Example II

In terms of necessary flux evaluations our L^∞-Voronoi approach proves to be significantly less expensive than the compared schemes. This is founded in the high ratio of 94% of primal cells without hanging nodes. On these cells the local pattern is the easiest possible, containing simply one dual node. This leads to only half as much local costs than incurred by the compared HLL-solver, and only one fourth in comparison to the diamond cell approach. This result confirms our topological statements from Section 2 and encourages further developement of our approach.


(a) Initial data at time 0

(b) Exact solution at time $\pi/4$

(c) Numerical solution at time $\pi/4$

Figure 14: Rotating cone with first order scheme

The high ratio of such cells can be roughly explained by the following argument: in an adaptively octtree-refined cartesian 3D-grid the number of cells on the (locally) deepest refinement level, i.e. those without hanging nodes (we call it *property* $\mathcal{U}$), outranges the number of coarser cells by far. In general, discontinuities in the numerical solution are resolved by (possibly thin) 2-dimensional manifolds. For a rough argument, let N_L be the number of grid cells and A_L the number of cells with property $\mathcal{U}$ on a grid with at most L refinement levels. Resolving a discontinuity by refining k cells on refinement level L effects the numbers A and N as follows: each of the k cells generates 8 child cells, the original cell itself disappears, thus $N_{L+1} \approx N_L + 7k$. The refinement of a cell inserts hanging nodes on its coarser neighbour cells. Since all k cells are supposed to form a thin surface, each of them "destroys" the property $\mathcal{U}$ only in the two face-neighbours in the surface's normal direction. The refined cell itself contributed most likely to A_L. Every child cell on level $L+1$ obviously has property $\mathcal{U}$. Thus $A_{L+1} \approx A_L + 5k$. For $k \gg 1$ one gets the ratio $r = A/N \approx 5/7$. If the k cells on the fine grid level rather form a volume than a thin surface, the ratio r gets even closer to 1.

6 Conclusions

In this paper we have developed a new adaptive staggered grid construction in 3D, which is an alternative to those presented in [1]. The construction of dual grid cells as Voronoi regions respecting the maximum-norm simplifies their geometrical description and, in the end, the implementation. Substantial combinatorial investigation in advance allows our scheme in return to exploit local geometrical structure of the grid at run-time to save a lot of numerical effort. Based on these adaptive grids, we have implemented a first order Lax-Friedrichs type finite volume scheme in 3D. The extension of this approach to a second order Nessyahu-Tadmor type scheme is in progress. It concerns the piecewise linear reconstruction and limitation of cell- and flux-values to end up with a second order scheme, and the data handling for large 3D simulations. A future step will be the incorporation of obstacles in the computational domain and the treatment of divers boundary conditions, especially when more complex geometries (and not only cartesian grid cells) need to be incorporated. Here it should be possible to apply techniques developed by Helzel et al [5].

A Combinatorial investigation

To determine the number of *essentially different* local patterns (that do not arise from each other by rotation or reflection) we use Pólya's counting theory. All information for the counting problem is provided by the group of symmetries $\mathcal{D}(\mathfrak{W})$ of the cube $\mathfrak{W}$. Analysing how the elements of $\mathcal{D}(\mathfrak{W})$ act on the set of $\mathfrak{W}$'s edges and faces leads to the number of 227 essentially different local patterns.

The distribution of the nodes $\mathcal{N}^+(C_p)$ on the boundary of a primal cell C_p determines the shape of the local pattern. Instead of speaking of occuring nodes on the midpoint of an edge or a face of C_p we shortly call this edge resp. face *colored*. If a face f of C_p is colored, i.e. f is refined, consequently all four edges of f are also refined, i.e. automatically colored. The converse is in general not true.

First, we list all 48 elements of $\mathcal{D}(\mathfrak{W})$, the *self-mappings of the cube* by specifying the 10 representatives g_k including the cardinality of the classes of conjugated elements $c(g_k)$:

$$g_1 = \begin{pmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & +1 \end{pmatrix}, c(g_1) = 1, \quad g_2 = \begin{pmatrix} +1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & +1 \end{pmatrix}, c(g_2) = 3,$$

$$g_3 = \begin{pmatrix} 0 & 0 & +1 \\ 0 & +1 & 0 \\ -1 & 0 & 0 \end{pmatrix}, c(g_3) = 6, \quad g_4 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, c(g_4) = 3,$$

$$g_5 = \begin{pmatrix} +1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}, c(g_5) = 6, \quad g_6 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, c(g_6) = 1,$$

$$g_7 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & +1 \\ 0 & +1 & 0 \end{pmatrix}, c(g_7) = 6, \quad g_8 = \begin{pmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & +1 & 0 \end{pmatrix}, c(g_8) = 8,$$
$$g_9 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & -1 & 0 \\ +1 & 0 & 0 \end{pmatrix}, c(g_9) = 6, \quad g_{10} = \begin{pmatrix} 0 & 0 & +1 \\ +1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}, c(g_{10}) = 8.$$

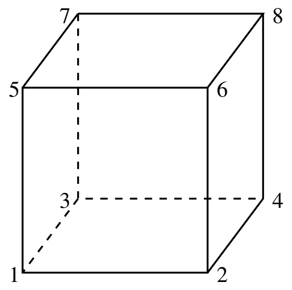


Figure 15: Node numbering on a cube

In order to count the *edge-colored* cubes (where edges may be colored independently from each other, and faces are not colored), we label the edges respecting Figure 15 by indicating their nodes as

$$\begin{aligned} e_1 &= (1,2), & e_5 &= (1,3), & e_9 &= (1,5), \\ e_2 &= (3,4), & e_6 &= (5,7), & e_{10} &= (2,6), \\ e_3 &= (5,6), & e_7 &= (2,4), & e_{11} &= (3,7), \\ e_4 &= (7,8), & e_8 &= (6,8), & e_{12} &= (4,8). \end{aligned}$$

The cycles of permutations on the set $\mathcal{E}(\mathfrak{W}) = \{e_j, j = 1, \ldots, 12\}$ induced by the mappings g_k read as

$$\begin{aligned} g_1^{(e)} &= (1)(2)(3)(4)(5)(6)(7)(8)(9)(10)(11)(12), \\ g_2^{(e)} &= (1\ 2)(3\ 4)(5)(6)(7)(8)(9\ 11)(10\ 12), \\ g_3^{(e)} &= (1\ 9\ 3\ 10)(2\ 11\ 4\ 12)(5\ 6\ 8\ 7), \\ g_4^{(e)} &= (1\ 3)(2\ 4)(5\ 8)(6\ 7)(9\ 10)(11\ 12), \\ g_5^{(e)} &= (1\ 4)(2)(3)(5\ 11)(6\ 9)(7\ 12)(8\ 10), \\ g_6^{(e)} &= (1\ 4)(2\ 3)(5\ 8)(6\ 7)(9\ 12)(10\ 11), \\ g_7^{(e)} &= (1)(2\ 3)(4)(5\ 10)(6\ 12)(7\ 9)(8\ 11), \\ g_8^{(e)} &= (4\ 6\ 11)(10\ 1\ 7)(5\ 12\ 3)(2\ 8\ 9), \\ g_9^{(e)} &= (1\ 12\ 3\ 11)(2\ 10\ 4\ 9)(5\ 7\ 8\ 6), \\ g_{10}^{(e)} &= (1\ 6\ 10\ 4\ 7\ 11)(2\ 5\ 9\ 3\ 8\ 12). \end{aligned}$$

Basing on the cycle representation of the $g_k^{(e)}$ we get the *cycle index* on the set $\mathcal{E}(\mathfrak{W})$ as

$$\mathfrak{Z}(\mathcal{D}, \mathcal{E}(\mathfrak{W})) = \frac{1}{48}(x_1^{12} + 3x_1^4x_2^4 + 12x_1^2x_2^5 + 4x_2^6 + 8x_3^4 + 12x_4^3 + 8x_6^2).$$

Here $d \cdot x_i^r$ denotes r cycles of length i in d different mappings $g \in \mathcal{D}$, e.g. $g_1^{(e)}$ leads to the term x_1^{12}, $g_3^{(e)}$ and $g_9^{(e)}$ both supply x_4^3 with coefficients $c(g_3) = c(g_9) = 6$. The denominator 48 is just the cardinality of $\mathcal{D}$.
In the case of only two different states per edge (colored or not) we substitute each x_i by $1 + x^i$ and get

$$\begin{aligned}\mathfrak{Z}_{\mathcal{E}(\mathfrak{W})}(\mathcal{D}, 1+x) =& \frac{1}{48}\Big((1+x)^{12} + 3(1+x)^4(1+x^2)^4 + 12(1+x)^2(1+x^2)^5 \\ &+ 4(1+x^2)^6 + 8(1+x^3)^4 + 12(1+x^4)^3 + 8(1+x^6)^2\Big) \\ =& 1 + x + 4x^2 + 9x^3 + 18x^4 + 24x^5 + 30x^6 + 24x^7 + 18x^8 \\ &+ 9x^9 + 4x^{10} + x^{11} + x^{12}.\end{aligned}$$

The number of essentially different edge-colored cubes with r colored edges equals the coefficient of the monomial x^r. For example, there are 9 cubes with 3 colored edges, as shown in Figure 16. The whole number of essentialy different edge-colored cubes sums up to $\mathfrak{Z}_{\mathcal{E}(\mathfrak{W})}(\mathcal{D}, 2) = 144$.

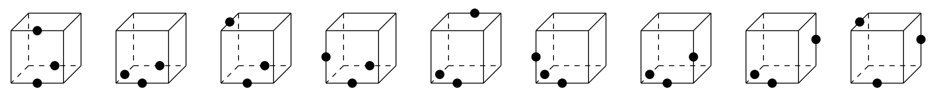


Figure 16: Essentially different cubes with 3 colored edges

Applying this counting on the permutations of faces instead of edges leads to 10 essentially different *face-colored* cubes $\mathfrak{W}^{\mathfrak{f}}$ (where faces may be colored independently from each other, and there is no additional colored edge). For all these $\mathfrak{W}^{\mathfrak{f}}$ we consider the appropriate self-mappings $g^{\mathfrak{f}}$, i.e. only those elements $g \in \mathcal{D}$ which map the colored faces of $\mathfrak{W}^{\mathfrak{f}}$ onto each other. Obviously, these self-mappings $\{g^{\mathfrak{f}}\}$ form a subgroup $\mathcal{D}^{\mathfrak{f}}$ of $\mathcal{D}$. The influence of all $g^{\mathfrak{f}}$ on the edges $\tilde{\mathcal{E}}(\mathfrak{W}^{\mathfrak{f}})$ which are not automatically colored by the colored faces of $\mathfrak{W}^{\mathfrak{f}}$ is summarized in the (reduced) cycle index $\mathfrak{Z}(\mathcal{D}^{\mathfrak{f}}, \tilde{\mathcal{E}}(\mathfrak{W}^{\mathfrak{f}}))$. Again, it supplies the number of essentially different edge-colored cubes now basing on the face-colored cube $\mathfrak{W}^{\mathfrak{f}}$.
As an example we examine a cube $\mathfrak{W}^{\mathfrak{f}}$ with exactly one colored face. Every self-mapping of $\mathfrak{W}^{\mathfrak{f}}$ has to map $\mathfrak{W}^{\mathfrak{f}}$'s only colored face onto itself. The set of self-mappings, again characterized by representatives $g_k^{\mathfrak{f}}$ and the cardinality of their classes of conjugated permutations $c(g_k^{\mathfrak{f}})$, reads as

$$g_1^{\mathfrak{f}} = \begin{pmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & +1 \end{pmatrix}, c(g_1^{\mathfrak{f}}) = 1, \quad g_2^{\mathfrak{f}} = \begin{pmatrix} +1 & 0 & 0 \\ 0 & +1 & 0 \\ 0 & 0 & -1 \end{pmatrix}, c(g_2^{\mathfrak{f}}) = 2,$$

$$g_3^{\mathfrak{f}} = \begin{pmatrix} +1 & 0 & 0 \\ 0 & 0 & +1 \\ 0 & +1 & 0 \end{pmatrix}, c(g_3^{\mathfrak{f}}) = 2, \quad g_4^{\mathfrak{f}} = \begin{pmatrix} +1 & 0 & 0 \\ 0 & 0 & +1 \\ 0 & -1 & 0 \end{pmatrix}, c(g_4^{\mathfrak{f}}) = 2,$$

$$g_5^{\mathfrak{f}} = \begin{pmatrix} +1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}, c(g_5^{\mathfrak{f}}) = 1.$$

The (reduced) cycles of the induced permutations on $\tilde{\mathcal{E}}(\mathfrak{W}^{\mathfrak{f}})$ are

$$g_1^{\mathfrak{f}(e)} = (1)(2)(3)(4)(5)(6)(9)(11), \qquad g_4^{\mathfrak{f}(e)} = (1\ 3\ 4\ 2)(5\ 11\ 6\ 9),$$
$$g_2^{\mathfrak{f}(e)} = (1\ 3)(2\ 4)(5\ 6)(9)(11), \qquad g_5^{\mathfrak{f}(e)} = (1\ 4)(2\ 3)(5\ 6)(9\ 11),$$
$$g_3^{\mathfrak{f}(e)} = (1)(2\ 3)(4)(5\ 9)(6\ 11),$$

and we get

$$\mathfrak{Z}_{\tilde{\mathcal{E}}(\mathfrak{W}^{\mathfrak{f}})}(\mathcal{D}^{\mathfrak{f}}, 1+x) = 1 + 2x + 6x^2 + 10x^3 + 13x^4 + 10x^5 + 6x^6 + 2x^7 + x^8,$$

e.g. 10 (the coefficient of x^3) essentially different colorings with one colored face and 3 (additional) colored edges, as shown in Figure 17.

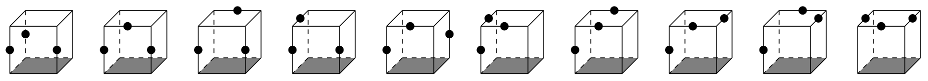


Figure 17: Essentially different cubes with 1 colored face and 3 colored edges

The numbers of essentially different *edge-face-colorings* of the cube, sorted by the numbers of colored faces, are listed in Table 4. All together we get $144 + 51 + 20 + 7 + 3 + 1 + 1 = 227$ essentially different edge-face-colored cubes, and hence 227 essentially different local *reference-patterns.*

colored faces	0	1	2	3	4	5	6
ess. diff. col.	144	51	20	7	3	1	1

Table 4: Numbers of essentially different edge-face-colored cubes

References

[1] P. Arminjon and A. St-Cyr. Nessyahu-Tadmor-type central finite volume methods without predictor for 3D Cartesian and unstructured tetrahedral grids. *Appl. Numer. Math.*, Vol. 46, 135–155, 2003.

[2] P. Arminjon, A. St-Cyr and A. Madrane. New two- and three-dimensional non-oscillatory central finite volume methods on staggered cartesian grids. *Appl. Numer. Math.*, Vol. 40, 367–390, 2002.

[3] P. Arminjon, M.C. Viallon and A. Madrane. A finite volume extension of the Lax-Friedrichs and Nessyahu-Tadmor schemes for conservation laws on unstructured grids. *Int. J. Comput. Fluid Dyn.*, Vol. 9, No. 1, 1–22, 1997.

[4] J. Bell, M. Berger, J. Saltzman, and M. Welcome. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comput.*, Vol. 15, No. 1, 127–138, 1994.

[5] M. Berger, C. Helzel, and R. LeVeque. H-box methods for the approximation of hyperbolic conservation laws on irregular grids. *SIAM J. Numer. Anal.* 41, No.3, 893-918 (2003).

[6] G.-S. Jiang and E. Tadmor. Nonoscillatory central schemes for multidimensionalhyperbolic conservation laws. *SIAM J. Sci. Comput.*, Vol. 19, No. 6, 1892–1917, 1998.

[7] T. Katsaounis, and D. Levy. A Modified Structured Central Scheme for 2D Hyperbolic Conservation Laws *Applied Math Letters*, 12, no. 6, 1999, pp. 89-96.

[8] M.Ch. Klin, R. Pöschel, and K. Rosenbaum. *Angewandte Algebra,* VEB Deutscher Verlag der Wissenschaften, Berlin, 1988.

[9] P.D. Lax. Weak solutions of non-linear hyperbolic equations and their numerical computations. *Comm. Pure Appl. Math.*, 7:159–193, 1954.

[10] D. Levy, G. Puppo and G. Russo. A fourth-order central WENO scheme for multidimensional hyperbolic systems of conservation laws. *SIAM J. Sci. Comput.* 24 (2002), no.2, 480–506.

[11] K.-N. Lie and S. Noelle. An improved quadrature rule for the flux-computation in staggered central difference schemes in multidimensions. *J. Sci. Comput.* 18 (2003), no.1, 69–81.

[12] X.D. Liu and E. Tadmor. Third order nonoscillatory central scheme for hyperbolic conservation laws. *Numer. Math.*, 79:397–425, 1998.

[13] S. Müller. Adaptive multiresolution schemes. *Proceedings of FVCA III, 2002, Porquerolles (France)*, Hermes Penton Science, p. 119 – 136.

[14] H. Nessyahu and E. Tadmor. Non-oscillatory central differencing scheme for hyperbolic conservation laws. *J. Comput. Phys.*, 87:408–463, 1990.

[15] S. Noelle, W. Rosenbaum and M. Rumpf. An adaptive staggered grid scheme for conservation laws. *Int. Ser. Numer. Math.*, 141:775–784, 2001.

[16] S. Noelle, W. Rosenbaum and M. Rumpf. Multidimensional adaptive staggered grids. to appear in DFG-ANumE final report, Springer, 2004.

[17] M. Ohlberger and M. Rumpf. Hierarchical and adaptive visualization on nested grids. *Computing*, Vol. 59, No. 4, 269–285, 1997.

[18] G. Pólya and R.C. Read. Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds. *Springer*, 1987.

[19] E.F. Toro. Riemann Solvers and Numerical Methods for Fluid Dynamics. *Springer*, 1997.