MPFA on Polyhedral Grids in Atmospherical Applications

Leibniz Institut für Troposphärenforschung

Martin-Luther Universität Halle-Wittenberg

Anna Maria Hartkopf
Matr.Nr. 205108459

Contents

Introduction

In the equations of theoretical meteorology the diffusion equation plays an important role:

$$\frac{\partial u}{\partial t} = \operatorname{div}\left(K \cdot \operatorname{grad}\, u\right),$$

whereas u is the concentration and K the conductivity. Diffusion also appears in the Euler equations and in the diffusion terms of the Navier Stokes equation:

$$\begin{pmatrix} \frac{\partial u}{\partial t} \\ \frac{\partial v}{\partial t} \end{pmatrix} = \operatorname{div}[u[\frac{1}{2}\nabla\vec{u} + \nabla\vec{u}^T]] = \operatorname{grad}[\operatorname{div}\vec{u}] - \frac{1}{2}\operatorname{curl}[\operatorname{rot}\vec{u}].$$

Having an accurate discretization of diffusion is thus a crucial task for atmospherical simulation.
The MUSCAT, the simulation model of the Leipzig Institute for tropospheric research is currently working only with vertical diffusion, because advection forces preponderates in the horizontal diffusion. Now a more exact discretization of diffusion shall be implemented in the MUSCAT and the ASAM, the other model of the institute.
The Multi Point Flux Approximation (abbr.: MPFA) method is a method to simulate diffusion on rough polyhedral grids. It is a control volume method that was developed for simulation on porous media. In numerical testing the method performs a convergence rate up to two, depending on the computational mesh. The first chapter will give a detailed description of the method.
We implemented the method for convex polyhedral grids with the restriction that in each corner of a cell exactly three edges and faces meet. This excludes for example pyramids with a quadrilateral base. This very general implementation gives us the ability to test the MPFA method on a wide variety of the different shaped grid used for atmospherical simulation. These meshes need to meet different requirements in vertical and horizontal direction. In horizontal direction the curvature of the earth, i.e. the spherical shape, must be taken into account and in vertical the orography of the earth surface, i.e. the mountains and valleys.
There are different approaches to meet this needs. We will describe four different methods of partitioning the sphere. Besides the rectangular latitude and longitude grid we will introduce quadrilateral, hexagonal and polygonal grids on the sphere. In vertical direction the orography can be resembled by terrain following coordinates in which the horizontal grid lines follow the earth surface. Another approach is to cut the orography out of a cartesian coordinate system. The second chapter covers the grid configurations.
In the third chapter we investigate the numerical properties of the method on the different grids. We performed two test scenarios on the horizontal grids and each with one on the 2D and 3D vertical grids to compare the terrain following coordinates with the cut cells.
The last chapter summarizes the investigations and points out open questions. It tries to put the previous development of the theoretical convergence results in a nutshell and integrates the further proceeding into it.

1 Multipoint Flux Approximation Method

The Multi Point Flux Approximation Method is designed to discretize the elliptic pressure equation on general rough meshes. It belongs to the family of control volume methods and provides a local explicit flux with respect to the pressure. [12] We will just refer to the MPFA O-Method which is most discussed in literature. It was developed for oil reservoir simulation. The geology with its non parallel layers has to be respected in simulation [13]. The method is suitable for non cartesian meshes. In meteorological modeling we also use these grids as we will see in section (...). The detailed description of the discretization will firstly only refer to quadrilateral grids in the two dimensional space. We will afterwards explain the 3D extension because the implementation was conducted for polyhedral cells.

1.1 Diffusion Equation and the Divergence Theorem

Consider the stationary diffusion equation

$$Q = -\operatorname{div}\left(K \cdot \operatorname{grad}\, u\right), \tag{1}$$

whereas the conductivity K is a symmetric and positive definite tensor, u is the concentration and Q the source term. To solve equation (1) it must be equipped with a boundary condition. [1]
We aim for a discrete form:

$$\mathbf{Au} = \mathbf{Q}, \tag{2}$$

where $\mathbf{A}$ is a sparse matrix and $\mathbf{u}$ and $\mathbf{Q}$ are discrete cell values. By introduction of a new variable:

$$q = -K \operatorname{grad} u,$$

the Diffusion Equation reduces to:

$$\operatorname{div} q = Q. \tag{3}$$

Let Ω be a compact domain with boundary $\partial\Omega$ and outer unit normal $\vec{n}$
Applying the divergence theorem (1) leads to:

$$\begin{aligned} \textstyle\int_\Omega Q d\omega &= \textstyle\int_\Omega \operatorname{div} q d\omega \\ &= \textstyle\int_{\partial\Omega} q\vec{n} ds \end{aligned} \tag{4}$$

$q\vec{n}$ may be considered as the *flux* over the boundary of Ω.
On a polygonal domain Ω, a cell of the grid, with vertices $(v_1, ..., v_n)$, edges $(e_1, ..., e_n)$, surface area $|\Omega| = V$ and the consideration of Q being constant on each cell, equation (4) becomes:

$$\int_\Omega Q d\omega = VQ = \sum_i f_i,$$

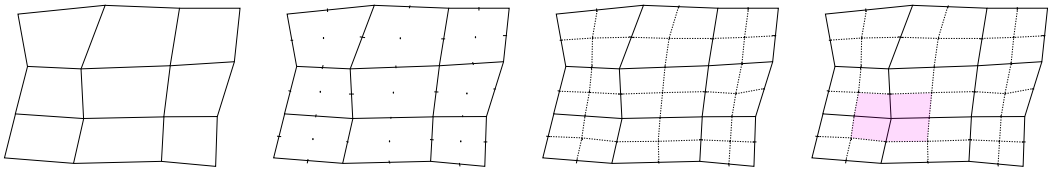
with

$$f_i = -\int_{e_i} q\vec{n}_i ds.$$

Now we want to find a discrete version of the flux f_i that only depends on the discrete cell values of u. For this discretization we need to define a dual grid.

1.2 Generating the Dual Grid

Let $\Omega \in \mathbb{R}^2$ be the domain with partition G consisting of vertices V, edges E and faces F. The vertices $\tilde{V}$ of the dual grid $\tilde{\mathrm{G}}$ are the *midpoints*[1] of the cells of the primal grid. The

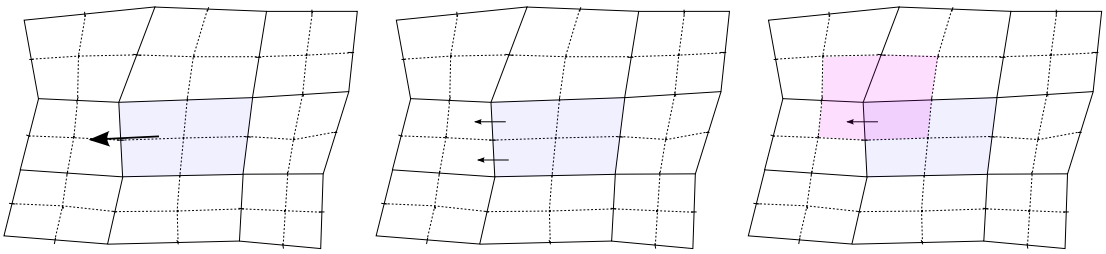


primal grid barycenters edges dual cell

dual edges link the barycenters/midpoint of two neighbored cells. The edge is bend so that it exactly crosses the intersection edge in its midpoint. The primal cells and their edges are intersected by this dual edges. The halved primal edges are denoted *subedges*. Analogously intersected cells are denoted *subcells*. Note that these subcells are always quadrilaterals in 2D and cuboids in 3D. The notion *interaction volume* is also current for the dual cells.

1.3 Discretization of the Flux

The expression for the flux over an entire edge is a combination of the flux expressions of its two subedges. To find a discretization of the partial flux we focus on the interaction



entire flux the two partial fluxes partial flux in the respective interaction volume

volumes, pictured in figure (1)
Let v_k be a non-boundary vertex. We will discuss the proceeding at the boundary with distinct conditions separately.
Let $C_j, j \in \{1, ..., j_k\}$ be the cells meeting in v_k and $c_j, j \in \{1, ..., j_k\}$ the subcells that

[1]The notion of midpoint is not uniquely defined on a quadrilateral, especially when the vertices are not planar. For the implementation we use the barycenters of the faces. For further investigations concerning the geometric notions see appendix B.

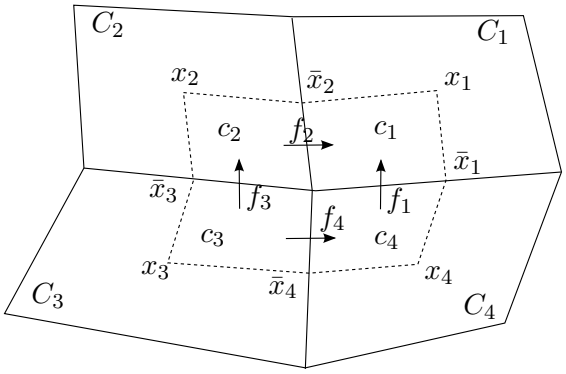


Figure 1: Notations

correspond to the respective intersection volume. Without loss of generality we set $j_k = 4$. The concentration u is resembled by linear functions U_j in each of the four subcells c_j. Let u_j be a discrete value of u in the subcell c_j. There is no unique definition of the discrete value. One might use for example the integral of u over the subcell or the value of the concentration in the dual vertex.
In addition we assume for the moment that we also know the discrete values at the centers of the edges, denoted:

$$\bar{u_i} = u_{\bar{i}}.$$

The linear function U_1 shall be spanned over the triangle $x_1, \bar{x_1}, \bar{x_2}$ (see figure (2). The

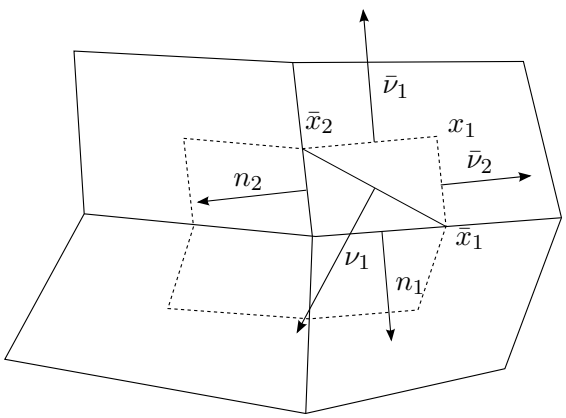


Figure 2: Configuration of the triangle and associated outer normal vectors in subcell c_1

interpolation constraints are:

$$\begin{aligned} U_1(x_1) &= u_1 \\ U_1(\bar{x_1}) &= \bar{u_1} \\ U_1(\bar{x_2}) &= \bar{u_2} \end{aligned} \tag{5}$$

Set

$$U_1(x) = \sum_{1,\bar{1},\bar{2}} u_i \phi_i \tag{6}$$

where ϕ_i is the linear basis function defined by $\phi_i(x_j) = \delta_{ij}$.
The gradient of ϕ_i is given by:

$$\mathrm{grad}(\phi_i) = -\frac{1}{2F}\nu_i. \tag{7}$$

ν_i are the unit normals respective the dual subedges and F is the surface of the triangle $x_i, i = 1, \bar{1}, \bar{2}$.[2] The normal vectors of the triangle are of the same length as the respective edge. For these vectors the following relation holds:

$$\sum_{i=1,\bar{1},\bar{2}} \nu_i = 0.$$

This gives us the ability to eliminate ν_1

$$\nu_1 = -(\bar{\nu_1} + \bar{\nu_2}). \tag{8}$$

The gradient of U_1 may now together with (7) and (8) be written as:

$$\mathrm{grad}(U_1) = -\frac{1}{2F}\sum_{i=1,\bar{1},\bar{2}} u_i\nu_i = -\frac{1}{2F}[(\bar{u_1} - u_1)\bar{\nu_1} + (\bar{u_1} - u_1)\bar{\nu_2}.] \tag{9}$$

Remember that the flux over an edge e_i was given by:

$$f_i = \int_{e_i} (-\mathrm{K}\,\mathrm{grad}\,u)\vec{n_i}ds.$$

Equation (9) gives us a discrete expression for $\mathrm{grad}\,u$. Now together with the normal vector $\vec{n_i}$ we are able formulate a discrete flux expression.[2] The formulation for the partial fluxes in subcell c_1, that just depends on u_1 , $\bar{u_1}$ and $\bar{u_2}$ is:

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \underbrace{-\frac{1}{2F_1}\begin{bmatrix} n_1^T \\ n_2^T \end{bmatrix}\mathrm{K}\begin{bmatrix} \bar{\nu_1} & \bar{\nu_2} \end{bmatrix}}_{G_1}\begin{bmatrix} \bar{u_1} - u_1 \\ \bar{u_2} - u_1 \end{bmatrix}. \tag{10}$$

In the following we may refer to G_1 as the the *geometry matrix* of subcell c_1.[1]
Since now we did not mention the *direction* of the normals. The divergence theorem enforces an *outer* normal vector. Each inner edge is the boundary of two cells, so the normal that is dedicated to the edge must change sign. To determine the direction of the normal respective a certain cell in a polyhedral grid without a global direction we use the notion of orientation[3]. In brief the orientation of an edge respective a cell is positive if its normal points out of the cell and vice versa. Expression (10) considering the orientation becomes:

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = -\frac{1}{2F_1}\begin{bmatrix} o_{n_1,c_1}n_1^T \\ o_{n_2,c_1}n_2^T \end{bmatrix}\mathrm{K}\begin{bmatrix} o_{\nu_1,c_1}\bar{\nu_1} & o_{\nu_2,c_1}\bar{\nu_2} \end{bmatrix}\begin{bmatrix} \bar{u_1} - u_1 \\ \bar{u_2} - u_1 \end{bmatrix}. \tag{11}$$

[2]Here again we are faced with non-uniqueness of definition if the vertices are non planar. We use a constant vector. For the approximation see appendix A.

[3]For a detailed definition and association with the proper mathematical context see appendix B

We will get analogous expressions for the fluxes within the other subcells c_k in the interaction volume surrounding the vertex v_j. The geometry matrices $G_k \in \mathbb{R}^{2\times 2}$ with entries g_{ij}^k depend only on metrical quantities (e.g. normals, their orientation, area of the triangle in the subcell) and the conductivity tensor $\mathcal{K}$.

$$G_k = \frac{1}{2F_k} \begin{bmatrix} o_{n_{k1},c_k} n_{k1}^T \\ o_{n_{k2},c_k} n_{k2}^T \end{bmatrix} \mathrm{K} \begin{bmatrix} o_{\nu_{k1},c_k} \bar{\nu_{k1}} & o_{\nu_{k2},c_1} \bar{\nu_{k2}} \end{bmatrix},$$

For the fluxes f_i^j in the interaction volume we yield the following system:

$$\begin{bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{bmatrix} = -G_1 \begin{bmatrix} \bar{u_1} - u_1 \\ \bar{u_2} - u_1 \end{bmatrix}, \quad \begin{bmatrix} f_2^{(2)} \\ f_3^{(2)} \end{bmatrix} = -G_2 \begin{bmatrix} \bar{u_2} - u_2 \\ \bar{u_3} - u_2 \end{bmatrix}, \tag{12}$$

$$\begin{bmatrix} f_3^{(3)} \\ f_4^{(3)} \end{bmatrix} = -G_3 \begin{bmatrix} \bar{u_3} - u_3 \\ \bar{u_4} - u_3 \end{bmatrix}, \quad \begin{bmatrix} f_4^{(4)} \\ f_1^{(4)} \end{bmatrix} = -G_4 \begin{bmatrix} \bar{u_4} - u_4 \\ \bar{u_1} - u_4 \end{bmatrix}. \tag{13}$$

Enforcing continuity of flux in the intersection points yields:

$$f_i^j = f_i^{j'}.$$

Since we have two conditions for each flux we are able to eliminate the unknowns $\bar{u}_i$ We define the vectors $\mathbf{u} = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 \end{pmatrix}^T$, $\bar{\mathbf{u}} = \begin{pmatrix} \bar{u_1} & \bar{u_2} & \bar{u_3} & \bar{u_4} \end{pmatrix}^T$ and $\mathbf{f} = \begin{pmatrix} f_1 & f_2 & f_3 & f_4 \end{pmatrix}^T$. By sorting the elements of the respective geometrical matrices in their designated position we gain the following two equations:

$$\mathbf{f} = E\mathbf{u} + C\bar{\mathbf{u}} = H\mathbf{u} + D\bar{\mathbf{u}} \tag{14}$$

The 4×4 matrices A,B,C and D are given by:

$$E = \begin{pmatrix} g_{11}^1 + g_{12}^1 & 0 & 0 & 0 \\ g_{21}^1 + g_{22}^1 & 0 & 0 & 0 \\ 0 & g_{21}^2 + g_{22}^2 & 0 & 0 \\ 0 & 0 & g_{21}^3 + g_{22}^3 & 0 \end{pmatrix} C = \begin{pmatrix} -g_{11}^1 & -g_{12}^1 & 0 & 0 \\ -g_{21}^1 & -g_{22}^1 & 0 & 0 \\ 0 & -g_{21}^2 & -g_{22}^2 & 0 \\ 0 & 0 & -g_{21}^3 & -g_{22}^3 \end{pmatrix}$$

$$H = \begin{pmatrix} 0 & 0 & 0 & g_{21}^4 + g_{22}^4 \\ 0 & g_{11}^2 + g_{12}^2 & 0 & 0 \\ 0 & 0 & g_{11}^3 + g_{12}^3 & 0 \\ 0 & 0 & 0 & g_{11}^4 + g_{12}^4 \end{pmatrix} D = \begin{pmatrix} -g_{21}^4 & 0 & 0 & -g_{22}^4 \\ 0 & -g_{11}^2 & -g_{12}^2 & 0 \\ 0 & 0 & -g_{11}^3 & -g_{12}^3 \\ -g_{11}^4 & 0 & 0 & -g_{12}^4 \end{pmatrix}$$

From equation (1.3) follows an expression for $\bar{\mathbf{u}}$:

$$\bar{\mathbf{u}} = (D - C)^{-1}(E - H)\mathbf{u}. \tag{15}$$

Finally the flux expression can be written only with the unknowns $\mathbf{u}$:

$$\mathbf{f} = T\mathbf{u}$$

$$T = E + C(D - C)^{-1}(E - H). \tag{16}$$

T is denoted *transmissibility matrix*.[1]

1.4 Assembling of the Flux

Let T_k be the transmissibility matrices of the vertices v_k. These matrices give us expressions for the partial fluxes in the intersection volumes that surround the vertices. These expressions need to be allocated for the flux balance in each cell. The cell C_1 that is shown

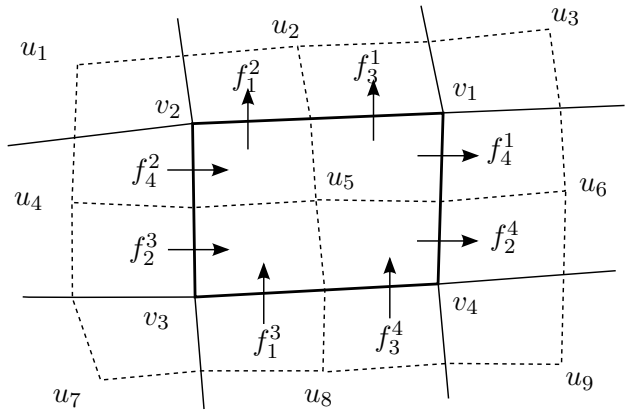


Figure 3: Primal Cell (thick line) with Partial Fluxes f_i^k

in the figure with area $\bar{C}_1$ and constant source term G has the following flux balance:

$$Q\bar{C}_1 = f_1^2 + f_3^1 + f_4^1 + f_2^4 - f_4^2 - f_2^3 - f_1^3 - f_3^4. \tag{17}$$

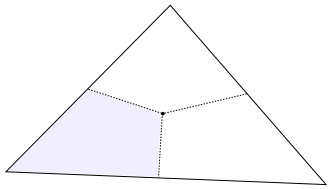
Assembly leads to the desired sparse matrix $\mathbf{A}$ (cf. (2)).

1.5 Extension to Polygonal Grids and Higher Dimensional Domains

For simplicity's sake the MPFA O-method was just described for quadrilateral grids in the two dimensional space. Because the implementation was conducted for polyhedral grids in $\mathbb{R}^3$ we first want to explain the extension to polygonal grids and then the generalization for three dimensional polyhedral meshes.

1.5.1 Polygonal Grids

The extension to polygonal grids in 2D is quite straightforward. The dual grid can emerge from the primal by the same rules as in the quadrilateral case. Let n_k be the number of cells and edges that adjoin in the vertex v_k. In 2D these numbers must be equal. The intersection volume is spanned over the midpoints of the cells and the midpoints of the edges. In every polygon the subcells that emerge by connecting the midpoint of the polygon with the midpoints of two neighbored edges are of quadrilateral shape.[11] This gives us the ability to apply the discretization of the flux via spanning a linear function over these



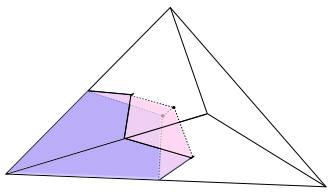
quadrilaterals (cf. (6)).
The size of the transmissibility matrix that gives the expression for the flux over the edges in dependance on the concentration in the cells equals the number of cells times the number of edges of the vertex: $T_k \in \mathbb{R}^{n_k \times n_k}$.

1.5.2 3D Domains

Note that the term *cell* always refers to the highest dimensional element of the grid. Hence, in 2D we may refer to the faces as cells whereas in 3D the same term denotes a polyhedron. Let $\Omega \in \mathbb{R}^3$ be a domain. Let $\mathbf{G}$ be a polyhedral partition of Ω. We have to make two restrictions to the cells $C \in \mathbf{G}$:

- the cell need to be convex and
- each corner of the cell consists of precisely three edges and faces.

The second constraint is called the *corner constraint*. A tetrahedra does satisfy the property, a pyramid with a quadrilateral base does not. Let v_k be a vertex of $\mathbf{G}$. In v_k cells $C_i, i = 1, ..., n_C$ meet, that are intersected by faces $F_i, i = 1, ..., n_F$. The interaction volumes that appear in the dual grid are formed by the linking of the midpoints of the cells x_i, $i = 1, ..., n_C$, of the faces $\bar{x}_i$, $i = 1, ..., n_F$ and of the edges that are adjoined in one vertex. The corner constraint ensures the cuboidal shape of the subcells. To discretize the gradient in the three



dimensional subcell we need to formulate the interpolation constraints. Let u_k be a discrete value of the concentration in the cell and $\bar{u}_i, i = j_1, j_2, j_3$ the discrete values on the faces. To satisfy the four degrees of freedom of a linear function in 3D, the interpolation constraints are:

$$\begin{aligned} U_1(x_1) &= u_1 \\ U_1(\bar{x_{k_1}}) &= \bar{u_{k_1}} \\ U_1(\bar{x_{k_2}}) &= \bar{u_{k_2}} \\ U_1(\bar{x_{k_3}}) &= \bar{u_{k_3}}. \end{aligned} \tag{18}$$

The elimination of the unknowns $\bar{u}_i, i = 1, ..., n_F$ approaches analogously to the two dimensional case.

The transmissibility matrix gives flux expressions for the flux over the faces in dependance of the cells. Hence, the size of T_k is given by $n_F \times n_C$.[1]

1.6 Boundary Conditions

1.6.1 Implementation Strategy for the Boundary Cells

In the description of the MPFA method we restricted the definition of the transmissibility matrices to non-boundary cells. If the interaction volume is located at the boundary of the domain we do not have two conditions for the fluxes over the edges at the boundary. The elimination of $\bar{u}$ can not be conducted as given in (15). We use an equivalent formalization of the transmissibility matrix. The two equations for the fluxes

$$\begin{aligned} f &= Eu + C\bar{u} \\ f &= Hu + D\bar{u}, \end{aligned} \tag{19}$$

may be written in the form:

$$\begin{pmatrix} f \\ f \end{pmatrix} = \begin{pmatrix} E & C \\ H & D \end{pmatrix} \begin{pmatrix} u \\ \bar{u} \end{pmatrix}.$$

For eliminating $\bar{u}$ we split the matrix in parts for u and $\bar{u}$:

$$\begin{pmatrix} f \\ f \end{pmatrix} = \begin{pmatrix} E & \bar{0} \\ H & \bar{0} \end{pmatrix} \begin{pmatrix} u \\ 1 \end{pmatrix} + \begin{pmatrix} 0 & C \\ 0 & D \end{pmatrix} \begin{pmatrix} 0 \\ \bar{u} \end{pmatrix},$$

where $\bar{0}$ is a vector filled with zeros.
With the identity I this leads us to:

$$\underbrace{\begin{pmatrix} I & -C \\ I & -D \end{pmatrix}}_{T_{f\bar{u}}} \begin{pmatrix} f \\ \bar{u} \end{pmatrix} = \underbrace{\begin{pmatrix} E & \bar{0} \\ H & \bar{0} \end{pmatrix}}_{T_u} \begin{pmatrix} u \\ 1 \end{pmatrix}. \tag{20}$$

By inverting the matrix on the left hand side we gain an equation for the flux that depends only on the concentration in the cell centers u:

$$\begin{pmatrix} f \\ \bar{u} \end{pmatrix} = \begin{pmatrix} I & -C \\ I & -D \end{pmatrix}^{-1} \begin{pmatrix} E & \bar{0} \\ H & \bar{0} \end{pmatrix} \begin{pmatrix} u \\ 1 \end{pmatrix}$$

In an interaction volume located at the boundary empty rows in $T_{f\bar{u}}$ occur, because we have only one condition for the flux through a boundary face. The boundary conditions, weather they are Neumann, Dirichlet or mixed, will be inserted in this rows to keep the solvability. We calculate a transmissibility matrix for each node. The Matrix $\tilde{T} := T_{f\bar{u}}^{-1} T_u$ has number of adjacent faces rows and number of adjacent cells plus one columns.

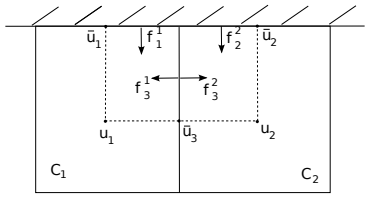


Figure 4: Cells C_1 and C_2 at the top boundary of the domain.

1.6.2 Insertion of Boundary Condition

To describe the insertion of the proper boundary conditions we refer to the situation given in figure (4). Extension to other boundary situations and other dimensions is straight forward. In equation (14) we have two equations for each flux. In our case only flux f_3 has two conditions. Thus we can just eliminate $\bar{u}_3$. The matrices H and D have no entries for the fluxes f_1 and f_2. Hence $T_{f\bar{u}}$ is not regular. We are given a mixed boundary condition:

$$\alpha f + \beta \bar{u} = g(\bar{v}), \text{ with } \alpha, \beta \in \mathbb{R}$$

and $g(\cdot)$ a function at the boundary. By writing out the whole system (20) one can clearly see its regularity:

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & & & \\ 0 & 1 & 0 & & C \in \mathbb{R}^{3\times 3} & \\ 0 & 0 & 1 & & & \\ \hline \alpha & 0 & 0 & \beta & 0 & 0 \\ 0 & \alpha & 0 & 0 & \beta & 0 \\ 0 & 0 & 1 & h_{31} & h_{32} & h_{33} \end{array}\right) \left(\begin{array}{c} f_1 \\ f_2 \\ f_3 \\ \hline \bar{u}_1 \\ \bar{u}_2 \\ \bar{u}_3 \end{array}\right) = \left(\begin{array}{cc|c} & & \\ E \in \mathbb{R}^{3\times 3} & & \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \\ & & \\ \hline 0 & 0 & g(\bar{v}_1) \\ 0 & 0 & g(\bar{v}_1) \\ d_{31} & d_{32} & 0 \end{array}\right)$$

Note that this approach can be applies to pure Neumann or Dirichlet boundary conditions by setting α equal to one and β equal to zero and vice versa.

2 Atmospherical Simulation Grids

The spherical shape of the earth and its non smooth surface has to be respected in atmospherical and meteorological simulation. The horizontal coordinate system respects the spherical shape whereas the vertical fits to the orography of the earth's surface. There exist different approaches of horizontal and vertical coordinate systems.
The German weather service (DWD) uses spherical coordinates for horizontal and terrain following coordinates for vertical coordinates in their local model (LM=Lokal Modell). In the model only small domains are considered and the equator of the sphere will be moved into the domain to guarantee a maximal quadratic shape of the cells and avoid pole singularity problems [18].
The global numerical weather prediction model of the German weather service is called GME. It uses an almost uniform icosahedral-hexagonal grid. It emerges from a regular icosahedron whose triangular faces are gradually refined, by partitioning them into four smaller triangles. The respective dual grid gives the icosahedral-hexagonal grid, which consists only of hexagonals except in the vertices of the original icosahedral. The grid avoids

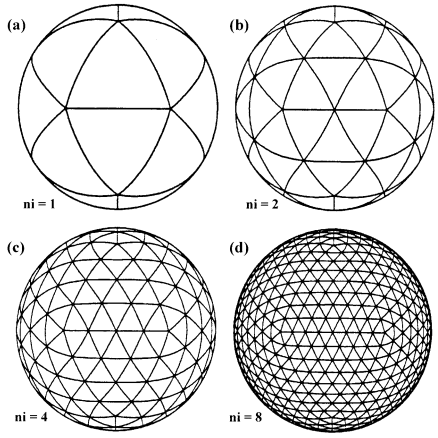


Figure 5: Grid generation by successively halving the triangle edges to form new triangles. Parameter n_i is the number of intervals on a major triangle edge.

pole singularity problems and is well designed for parallel computation [14].
The OLAM, i.e. the Ocean-Land-Atmosphere Model, designed to take all scales of atmospheric flows into account. It works with a triangular mesh in horizontal and shaved cells in vertical direction [19]. We will refer to shaved cells as *cut cells*.
A new type of the atmospheric general circulation model based on the nonhydrostatic system and the icosahedral grid is developed. It is called Nonhydrostatic Icosahedral Atmospheric Model (NICAM) and designed to make use of the extending computer power of the Earth Simulator, a Japanese Supercomputer. It works with an icosahedral-hexagonal mesh in horizontal direction. The grid is constructed as in [14] and smoothen by using spring dynamics [17].
The climate department of the Los Alamos National Laboratory use centroidal Voronoi tessellation on the sphere. The grid is iteratively generated from randomly distributed points

on the sphere. This type of grid is simple to generate and discretizes the sphere very uniformly. The numerical solutions turn out to be very robust [16].
This section will give an overview over the construction of some of the grids named above. First we want to introduce the possibilities for vertical coordinates, namely cut cells and terrain following coordinates. Then we want to present some grids on the sphere. We will refer to the implementation of the different grid types and describe the data structure for general polyhedral grids.

2.1 Horizontal Grids

For large scale modeling the spherical shape of the earth must be taken into account. The simplest spherical grid is the classical *orange slice* shaped rectangular longitutde/latitude grid, that is used in geography. There are several disadvantages of this kind of grid. The size of the cells strongly varies and there appear singularities at the poles. New approaches for avoiding this disadvantages are the use of centroidal Voronoi tessellation girds, with uniform hexahedral cells and a icosahedral-hexagonal grid that is generated by refinement of a icosahedron. In this work we introduce four different types of horizontal grids. The first is the classical spherical grid, then centroidal Voronoi tessellation and a refinement of the dodecahedron, which will lead to a grid very similar to the icosahedral-hexagonal grid. Finally we introduce a quadrilateral grid on the sphere, which emerges from a wrapping of a cartesian grid around the sphere.

2.1.1 Rectangular Longitude/Latitude Grids

This classical spherical grid is shaped like the latitudes and longitudes used in geography. It is the conventional mapping of the spherical coordinate system in a cartesian. Let S be the spherical coordinate system on a sphere with radius r

$$S = [-\frac{\pi}{2}, \frac{\pi}{2}] \times [-\pi, \pi] \ni (\lambda, \varphi),$$

and G be an equidistant partition of S

$$G = \{(\lambda_i, \varphi_j) \mid \lambda_i = -\frac{\pi}{2} + \frac{i\,\pi}{n_\lambda}, \varphi_j = -\pi + \frac{2\,j\,\pi}{n_\varphi}\} \quad i = 0, ..., n_\lambda, \quad j = 1, ..., n_\varphi.$$

The vertices of G are mapped via:

$$\begin{aligned} x &= r\,\sin\varphi\,\cos\lambda \\ y &= r\,\sin\varphi\,\sin\lambda \\ z &= r\,\cos\varphi. \end{aligned}$$

The main problem of this kind of grid are the poles. In the poles n_φ cells meet. On a fine-meshed grid this may lead to computational problems. We avoided this problem by subtracting and adding a small $\varepsilon \in \mathbb{R}^+$ on the boundaries of the interval of λ. The cells do not meet in the poles anymore. The logical cartesian property of the grid is kept.

2.1.2 Centroidal Voronoi Tessellation Grid

Before explaining the algorithm for generating the centroidal Voronoi tessellation we need some definitions. Let Ω be an open, bounded domain on $\mathbb{R}^d$ and $\{x_i\}_{i=1}^n \in \Omega$ a set of distinct points. For each x_i the corresponding *Voronoi region* or *Voronoi cell* is defined by:

$$V_i = \{x \in \Omega \mid \|x - x_i\| < \|x - x_j\| \text{for } j = 1, ..., n \ , j \neq i\},$$

whereas $\|\cdot\|$ is the Euclidean distance. Clearly $V_i \cap V_j = \emptyset$ for $i \neq j$ and $\cap_{i=1}^n \bar{V} i = \bar{\Omega}$. The partition $\{V_i\}_{i=1}^n$ is called the *Voronoi tessellation* of Ω corresponding to $\{x_i\}_{i=1}^n$. The single point x_i is denoted the *generator* of V_i. The dual grid of the Voronoi tessellation consists of triangles in 2D and tetrahedra in 3D and is called *Delaunay tessellation.* [15].
Let $\rho(x) \geq 0$ be a density function on Ω. For any region $V \in \Omega$ the *standard mass center* x^* of V is given by:

$$x^* = \frac{\int_V x\rho(x)dx}{\int_V \rho(x)dx}.$$

A Voronoi tessellation in which the generators x_i for the Voronoi regions V_i are themselves the mass centroids of those regions is denoted a *centroidal Voronoi tessellation*[6].
For constraining the centroidal Voronoi tessellation to a sphere we use the *constrained centroidal Voronoi tessellation.* The generation of the grid uses an algorithm known as *Lloyd's method for CCVTs*:
ALGORITHM Given a surface S, a density function $\rho(x)$ defined for all $x \in S$, and a positive integer k
0. select an initial set of k points $\{z_i\}_{i=1}^k$ on S e.g., by using a Monte Carlo method;
1. construct the Voronoi sets $\{V_i\}_{i=1}^k$ of S associated with $\{z_i\}_{i=1}^k$;
2. determine the constrained mass centroids of the Voronoi sets$\{V_i\}_{i=1}^k$;
3. these constrained centroids form the new set of points $\{z_i\}_{i=1}^k$;
4. if the new points meet some convergence criterion, terminate; otherwise, step 1. [7]

For numerical testing we compared grids with fewer iteration steps that are less rectangular

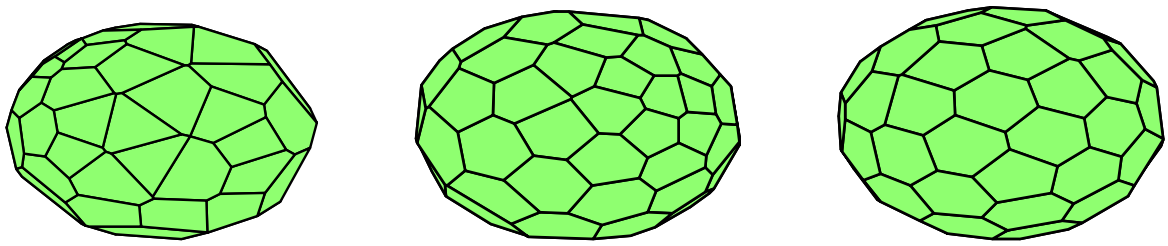
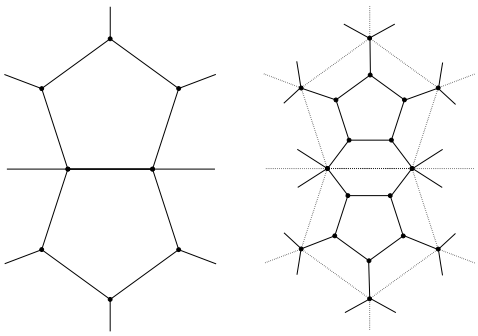


Figure 6: Centroidal Voronoi Tessellation with 64 Cells after 1, 5 and 50 Iteration Steps

to those with a higher number of iteration steps that get more consistent. After a certain number of iterations the number of vertices of the polygons shall be between four and seven.

2.1.3 Uniform Refinement of the Platonic Dodecahedron

This grid satisfies the demand for uniform mesh refinement of a polygonal grid on the sphere. The basis is a platonic dodecahedron. In the first refinement step each pentagon contracts and is encased by five halved hexagons. At each edge of the pentagon two halved hexagons



emerge which fuse to form a hexagon themselves. In the following refinement steps the hexagons and pentagons shrink again and will be encased by six respectively five halved hexagons. For implementation we inserted a yet existing grid that was generated with a

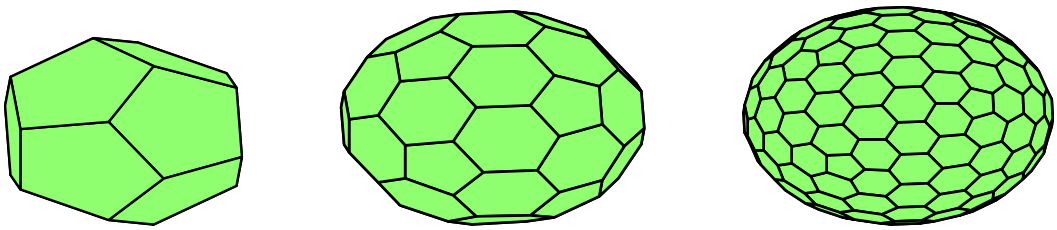


Figure 7: Dodecahedron after none, one and two steps of refinement

PASCAL code.

2.1.4 Quadrilateral Grid on the Sphere

The quadrilateral grid on the sphere avoids pole singularities. The cell size is nearly uniform. The ratio of the smallest to the biggest cell is nearly two and does not increase under refinement. The main idea is to map the computational domain $[-3, 1] \times [-1, 1]$ on the

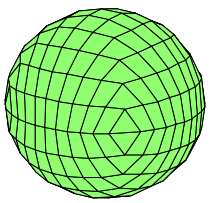
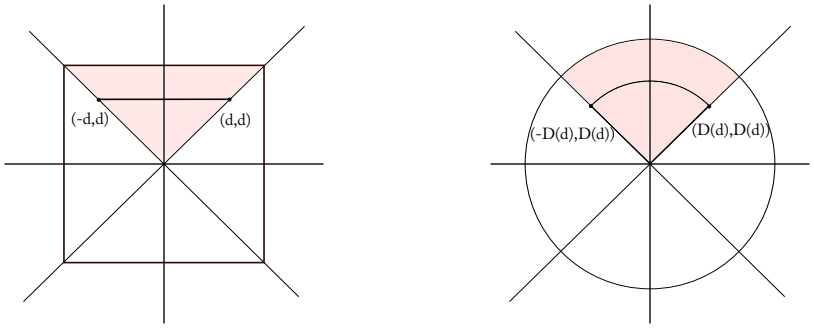


Figure 8: Quadrilateral Grid on the Sphere

sphere. Figuratively one can describe the approach by folding the domain in the middle and

inflate it to a sphere. For the construction of the grid we first focus on the mapping of the unit square $[-1,1] \times [-1,1] \ni (\xi, \eta)$ on a unit disk. The north sector of the computational domain is the part of the square satisfying $|\xi| \leq \eta$.
The main idea is to map a horizontal line segment between points $(-d, d)$ and (d, d) on a circular arc of radius $R(d)$, passing through points $(-D(d), D(d)))$ and $(D(d), D(d))$. Due to



the radius not necessarily being equal to one, the center of the circle is located in $(0, c_y)$, with $c_y = D(d) - \sqrt{R(d)^2 - D(d)^2}$. For each point (ξ, η) in the north sector of the computational domain, i.e. $|\xi| \leq \eta$ the mapping on the disk is given by:

$$\begin{aligned} X_n(\xi, \eta) &= D(\eta)\tfrac{\xi}{\eta} \\ Y_n(\xi, \eta) &= D(\eta) - \sqrt{R(d)^2 - D(d)^2} - \sqrt{R(d)^2 - X_n(\xi, \eta)^2} \end{aligned}$$

The mapping of the other three parts of the unit square is obtained by negating and/or swapping the arguments (ξ, η) or functions X_n, Y_n.
By mapping of the disk on the hemisphere we gain a mapping from the unit square to the hemisphere. The computational domain $[-3,1] \times [-1,1]$ consists of two clotted unit squares that will be mapped on two hemispheres which form an entire sphere. If the distance between the arcs on the disk is uniform, a mapping of the disk on the hemisphere will lead to elongated cells near the equator. The formula:

$$D(d) = \sin(\frac{\frac{\pi d}{2}}{\sqrt{2}})$$

compresses the grid lines near the boundary of the disk. With given functions $X(\xi, \eta)$ and $Y(\xi, \eta)$ for the mapping from the unit square to the unit disk we define the mapping functions $X_S(\xi, \eta)$, $Y_S(\xi, \eta)$ and $Z_S(\xi, \eta)$ for the mapping of the computational domain on the unit square: [4]

$$\begin{aligned} X_S(\xi, \eta) &= \begin{cases} X(-(\xi+2), \eta) & \text{if } |\xi+2| \leq 1 \\ X(\xi, \eta) & \text{if } |\xi| \leq 1 \end{cases} \\ Y_S(\xi, \eta) &= Y(\xi, \eta) \\ Z_S(\xi, \eta) &= \begin{cases} -\sqrt{1 - (X(-(\xi+2), \eta) + Y(\xi, \eta)^2} & \text{if } |\xi+2| \leq 1 \\ -\sqrt{1 - (X(\xi, \eta) + Y(\xi, \eta)^2} & \text{if } |\xi| \leq 1 \end{cases} \end{aligned}$$

2.2 Vertical Grids

Beneath the spherical shape of the earth the orography of the earth ,i.e. the shape of the earth, the mountains and valleys, must be resembled in the horizontal coordinates of the computational mesh. This vertical modifications can e applied to all of the above introduced horizontal grids. If the computational domain for modeling is sufficiently small, the spherical shape may be neglected. Let $D \in \mathbb{R}^3, v = (x, y, z)^T \in D$ be such a domain in which the mesh shall be inscribed into and $O(\cdot, \cdot)$ the orography function, i.e. $O(x, y)$ returns the height of the ground level at a certain point $(x, y) \in D$.
As mentioned above there are two possibilities to insert the orography information into the grid, cut cells or shaved cells which literally cut the orography out of a grid that does not take the orography into account and terrain following coordinates that squeeze the grid from the ground level, but keeps the grid logically cartesian.

2.2.1 Terrain Following Coordinates

Terrain following coordinates are generated by a transformation of a cartesian coordinate system in its vertical direction. The cartesian property is, in contrary to the cut cells, kept by this transformation. Several transformations of the grid are conceivable but for implementation we use the traditional transformation, developed by Gal-Chen and Somerville in 1974 to solve the Navier-Stokes Equation [9]. The horizontal grid lines strongly follow the orography near the ground and get shallower the farer away from the ground level they are. Let $G \subset \mathbb{R}^n$ be a cartesian grid with coordinates $\bar{x}, \bar{y}$ and $\bar{z}$ on the domain $[0, H_x] \times [0, H_y] \times [0, H_z]$ and $O(x, y)$ the orography function that should be applied on G to get the transformed grid $\hat{G}$ with coordinates x, y and z. The transformation is given by:

$$\begin{aligned} x &= \bar{x}, \\ y &= \bar{y}, \\ z &= [\bar{z}(H_z - O(x, y))/H_z] + O(x, y). \end{aligned}$$

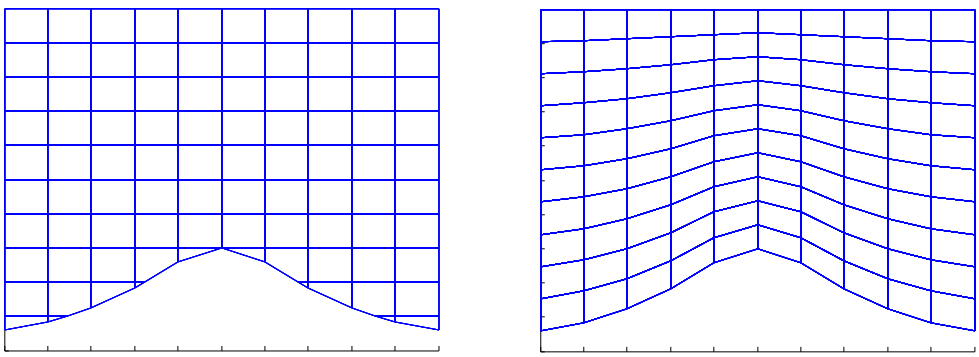


Figure 9: Cut cell grid and terrain following coordinates in a 2D domain matched to the same mountain.

2.2.2 Cut Cells

The basis of a cut cell grid may be any polyhedral grid. We restrict our investigations to a cubic cartesian grid $G \subset \mathbb{R}^3$, consisting of sets of p-dimensional elements $p \in \{0, ..., 3\}$. Of course the definition can be downsized to two dimensions, but as the implementation was conducted for three dimensions we will give this description. The p-dimensional element sets are:

- 0-dimensional vertices $V = \{v_i = (x_{v_i} y_{v_i} z_{v_i})^T, i = 1, ..., n_v\}$,
- 1-dimensional edges $E = \{e_i, i = 1, ..., n_e\}$, with boundary $\partial e_i = \{v_{i1}, v_{i2}\}$ consisting of vertices,
- 2-dimensional faces $F = \{f_i, i = 1, ..., n_f\}$, with boundary $\partial f_i = \{e_{i1}, e_{i2}, e_{i3}, e_{i4}\}$ consisting of edges that are ordered to orbit the face and
- 3-dimensional cells $C = \{C_i, i = 1, ..., n_C\}$, with boundary $\partial C_i = \{f_{i1}, f_{i2}, f_{i3}, f_{i4}, f_{i5}, f_{i6}\}$ consisting of faces

The border of a domain $D \subset \mathbb{R}^3$, in our case the orography function, is literally cut out of the grid. Elements that are located entirely out of the manifold will be rejected. Those that intersect the manifolds boundary are modified and new elements (faces, edges and vertices) emerge to mimic the smooth boundary surface ∂D. In figure 2.2.2 we can see modified 3-dimensional elements which cross the boundary. The *new* elements are the colored faces which built together with their edges and vertices a part of the boundary D. Instead of

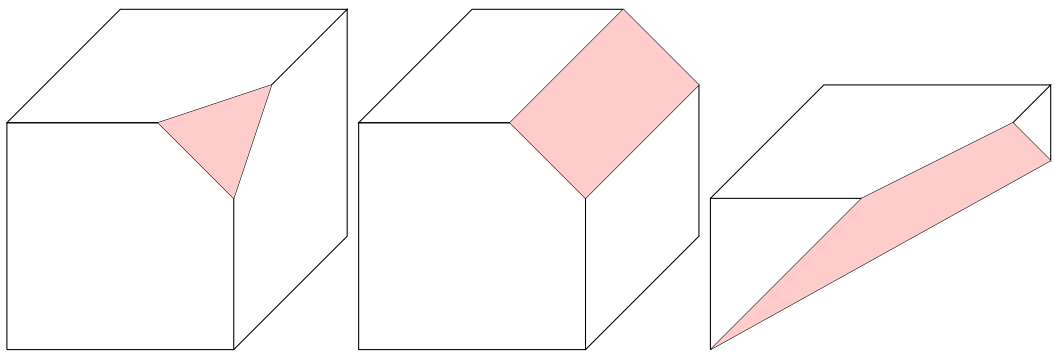


Figure 10: Cut Cells in 3D

the boundary of a general manifold we will consider the orography function $O(x, y)$ to be cut out of the grid. Note that we exclude the case of having caves on the earth surface by forcing O to be a function. A suitable way to describe the boundary is a *levelset* function.

$$\varphi(x, y, z) = z - O(x, y).$$

With the help of a level set function φ the boundary is described by the set $\varphi(x, y, z) = zero$. We introduce a *cutting operator* $\mathcal{C}$. This operator will be consecutively applied to the sets of p-dimensional elements (V, E, F and C) of G. It is firstly applied to the vertices to distinguish weather they are located above or underneath the earth's surface. Those

vertices that are not rejected build the set $\hat{V}_1$. Then the cutting operator will be applied to the set of edges. It induces the the set of edges whose ends both lie above the ground and those that intersect the boundary. Those form the set $\hat{E}_1$. The intersection point of an intersecting edges with the boundary is the null of the levelset function on the interval of the edge. These intersection points form the set $\hat{V}_2$ which merged with $\hat{V}_1$ establish the total set of vertices of the cut cell grid $\hat{G}$. By applying the cutting operator we yield the set of faces $\hat{F}_1$ that are above the boundary or intersecting it. The intersection edges are formed by joining the intersection points of the intersecting edges that belong to the face (see below for details). Thus the total set of edges consist of these edges $\hat{E}_2$ together with $\hat{E}_1$. Analogously we apply the cutting operator to the cells. In this step the cross section of the cells with the boundary form the set of faces $\hat{F}_2$ that again merges with $\hat{F}_1$ to the total set of faces.
We see that the procedure of applying $\mathcal{C}$ must be successive. It is not a mapping because more than one element may emerge as an image of a single element. We focus on the single steps of the approach of applying the cutting operator:

Step 1: $\mathcal{C}(V)$ In the first step the operator is applied to the vertices. Those that are located below the orography or at the boundary are rejected.

$$\mathcal{C}(v_i) = \begin{cases} v_i & \text{if } \varphi(x_{v_i}, y_{v_i}, z_{v_i}) > 1 \\ \emptyset & \text{else} \end{cases}$$

$$\mathcal{C}(V) = \hat{V}_1 = \{v_i \in G \mid \mathcal{C}(v_i) \neq \emptyset\}$$

Step 2: $\mathcal{C}(E)$ In the second step $\mathcal{C}$ will be applied to the edges E. The edges that are located below the orography are rejected, those that are entirely located above are kept unmodified. For those edges that intersect the ground level a *new* vertex, the intersection point, emerges. The vertex of the edge that is below the orography is replaced by the intersection point. The edge is shortened, but keeps its topological characteristics. Note that the vertices that are located on the boundary, which were rejected in step 1 will appear as intersection points again.
Let $e_i \in E$ be an edge with $\partial e_i = [v_{i1}, v_{i2}]$ and length $|e|$. Formally the application of the cutting operator to the edges can be expressed as follows:

$$\mathcal{C} := \begin{cases} e_i & \text{, if } \varphi(v_{i1}) > 0 \hat{\varphi}(v_{i2}) > 0 \\ \emptyset & \text{, if } \varphi(v_{i1}) \leq 0 \hat{\varphi}(v_{i2}) \leq 0 \\ (\tilde{e}_i, \tilde{v}_i) & \text{, else} \end{cases}$$

The modified edge $\tilde{e}_i$ is defined by its boundary:

$$\partial \tilde{e}_i := \begin{cases} \{\tilde{v}_i, v_{i2}\} & \text{, if } \varphi(v_{i1}) \leq 0 \\ \{v_{i1}, \tilde{v}_i\} & \text{, if } \varphi(v_{i2}) \leq 0. \end{cases}$$

For a sufficiently monotonous levelset function φ the intersection vertex is a root of φ on the interval of the edge:

$$\varphi(\tilde{v}_i) = 0.$$

As an image of E under $\mathcal{C}$ we gain:

$$\mathcal{C}(E) = \hat{E}_1 \cup \hat{V}_2,$$

with

$$\hat{E}_1 = \{e_i \in E \mid \mathcal{C}(e_i) \neq \emptyset\}$$
$$\hat{V}_2 = \{\tilde{v}_i \mid i \in I\},\ I = \{i \in \mathbb{N} \mid e_i \text{ intersects the orography}\}.$$

Step 3: $\mathcal{C}(F)$ The operator $\mathcal{C}$ will be applied to the faces F. Faces that are entirely below or above the ground level will be rejected or kept unmodified. Let f_i be a face orbited by the edges $\{e_i1, e_i2, e_i3, e_i4\}$. And let the set $\{v_i1, v_i2, v_i3, v_i4\}$ be the vertices of f_i. Application of $\mathcal{C}$ to f_i:

$$\mathcal{C}(f_i) := \begin{cases} f_i & \text{if } \varphi(v_ij) > 0 \forall j = 1, ..., 4 \\ \emptyset & \text{if } \varphi(v_ij) \leq 0 \forall j = 1, ..., 4 \\ (\tilde{f}_i, \tilde{e}_i) & \text{else} \quad . \end{cases}$$

For the construction of the modified face $\tilde{f}_i$ we consider the images of the edges of f_i under the operator $\mathcal{C}$. We define:

$$\partial\, \bar{\tilde{f}}_i = \{\mathcal{C}(e_{i1}), \mathcal{C}(e_{i2}), \mathcal{C}(e_{i3}), \mathcal{C}(e_{i4}), \tilde{e}_i\}.$$

Let e_{ik} and e_{il} be the edges that intersect the orography function, i.e. $\mathcal{C}(e_{ij}) = \tilde{e}_{ij}, j = k, l$. The vertices $\tilde{v}_{il}$ and $\tilde{v}_{ik}$ are the associated intersection points. The edge $\tilde{e}_i$ that nestles up against the orography is the linking between these two intersection points. Without loss of generality we set:

$$\partial \tilde{e}_i = \{\tilde{v}_{ik}, \tilde{v}_{il}\}.$$

For the modified face $\tilde{f}_i$ the edges must be ordered to orbit the face again. The image of F under the cutting operator are the modified faces $\hat{F}_1$ and the edges that nestle up against the orography $\hat{E}_2$.

Step 4: $\mathcal{C}(C)$ In the forth and last step we finally generate the *cut cells*. We consider a cell C_i with boundary $\partial C_i = \{f_{ij}, (j = 1, ..., 6)\}$ and vertices $\{v_{ij}\ , j = 1, ..., 8\}$. The cutting operator is analogously defined by:

$$\mathcal{C}(C_i) := \{ \begin{cases} C_i & \text{if } \varphi(v_ij) > 0 \forall j = 1, ..., 8 \\ \emptyset & \text{if } \varphi(v_ij) \leq 0 \forall j = 1, ..., 8 \\ (\tilde{C}_i, \tilde{f}_i) & \text{else} \end{cases}$$

Again we consider the image of the boundary faces to compose $\tilde{C}_i$ and define for the boundary of the modified cell:

$$\partial\tilde{C}_i := \{\mathcal{C}(f_{ij})_{(j=1,...,6)}, \tilde{f}_i\}$$

Let E be the set of edges of $\mathcal{C}(f_{ij})_{(j=1,...,6)}$:

$$\tilde{E}_i := \{\partial(\mathcal{C}(f_{ij}))_{(j=1,...,6)}\}$$

The edges of $\tilde{E}$ that are also elements of $\hat{E}_2$, i.e. that emerge from the construction of the modified faces build the intersection face of the cell $\tilde{f}_i$. The intersection set $\tilde{\tilde{E}} = \tilde{E}_i \cap \hat{E}_2$ contains the edges that form $\tilde{f}_i$.
The edges have to be put into the right sequence to make sure that the face is orbited.

$$\partial\tilde{f}_i = \{e_{l1}, ..., e_{ln_i}\},\ e_{lj} \in \tilde{\tilde{E}}\ (j = 1, ..., n_i)$$

By this four steps all the elements of the cut cell grid are constructed.

2.3 Implementation and Data Organization

The different grid configurations mentioned above can be transformed into the framework of general polyhedral grids. The only constraint of the grid configuration is that each corner of each cell must consist of exactly three faces. This includes all the above grids but for example excludes girds that consist of pyramids with a quadratic base. For special applications we stick closely to the architecture of the grid. So we need an organization of the data that provides information but minimizes the computational cost. For this the combinatorial properties of the mesh are separated from the metrical. We only store geometrical information, i.e. the coordinates, for the vertices. Furthermore only incidence will be stored. In graph theory incidence is resembled by an incidence matrix. It expresses the relation of two classes of objects. For example the incidence matrix of cells and faces has a row for each cell and a column for each face. If a cell and a face are incident, i.e. the face is a boundary element of the cell, the entry in matrix is 1 or -1, weather the orientation of the face is positive ore negative.
The incidence matrices are not efficient to handle in an implementation. Instead, we use pointers which point to the respective elements. As the implementation is written in Matlab, the pointers are resembled by structs that are attached to each element. A struct contains the numbers of the positions of the associated elements. By generating these lists and attaching the boundary information of the single elements the combinatorics of the grid is fully described.
The metrical quantities, like midpoints or surface areas, are computed in a separate routine and attached to the struct of the respective elements. We have still not yet defined the notion of *orientation*. It is a generalization of left and right in a vicinity without global directions. The term *orientation* is used in linear algebra and differential geometry. We will consider the linear algebra definition and amplify it for our needs. Let V be a vector space

```
Number: 1                          Mid =
  Edge: [41 107 49 105]
  Node: [1 3 13 11]                    x: 0.1250
  Cell: [1 1]                          y: 0.3018
   Mid: [1x1 struct]                   z: 0.8536
  Area: 0.1937
Normal: [1x1 struct]
```

Figure 11: Example of a Struct of a Face in the Actual Matlab Implementation

and B_1 and B_2 two bases of V. Let $E : V \to V$ be the linear transformation that maps B_1 into B_2. We denote the bases equally oriented if:

$$det(E) > 0.$$

Thus an orientation is an equivalence class of bases. Let $v, w \in \mathbb{R}$ be linearly independent. Then the bases $(e_1, e_2, e_3)$and$(v, w, v \times w)$ are equally oriented [8]. If we assign the orientation *positive* to the canonical basis, we detect that the basis that is formed by two linearly independent vectors and their cross product, i.e. the normal vector on the plane that is spanned by them, are positive oriented too. We may assign a negative orientation to the half-space into which the normal vector points.
This property is extendable to define orientation on the mesh. The faces of the mesh are not necessarily planar. The edges and so the vertices that define the boundary of a face are ordered to orbit the face. This is necessary to define a normal vector on this skew face[4]. The direction of the normal vector is unique. So the cell in which the normal vector points is assigned a negative orientation and vice versa.
To adapt the by definition two dimensional spherical grids into the framework of general polyhedral 3dimensional grids, it will be *inflated*. The approach of the inflation is as follows:

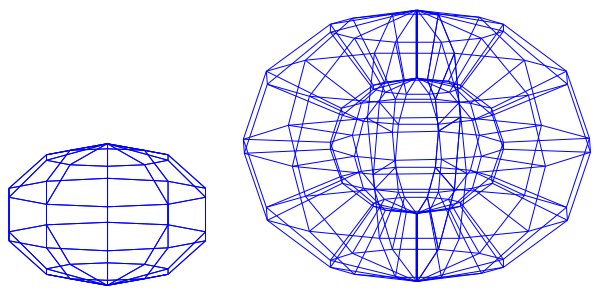


Figure 12: Flat and Inflated Rectangular Longitude/Latitude Spherical Grid.

Let G_1 be any grid on the sphere S_1 around the origin with radius $r_1 \in \mathbb{R}$. G_1 consists of Vertices V, Edges E and Faces F. Let S_2 be a second sphere around the origin with radius $r_2 > r_1, \in \mathbb{R}$. By multiplying the coordinates of the vertices of G with the factor $\alpha = \frac{r_2^2}{r_1^2}$ we get a mapping of the vertices V on the sphere S_2. With this mapping we can construct

[4]c.f. Appendix A

a copy G_2 of the entire grid G_1 on S_2.
The linking of an origin element with its copy form an element of higher dimension:

- the linking of a vertex with its copy form an edge,
- the linking of an edge with its copy form a face and
- the linking of a face with its copy form a cell.

The elements of G_1, G_2 and the elements emerged by linking form the inflated spherical grid that is a 3dimensional analogon of the flat 2dimensional.

3 Numerical Tests

In this section we want to investigate the convergence behavior of the MPFA Method on the different grids which were introduced in section (3). In the first part we compare horizontal grids. The comparison of cut cells and terrain following coordinates occurs in the second part of the chapter. We will see convergence near to quadratic behavior for all test scenarios. The Multi Point Flux Approximation Method, described in chapter (2) discretizes the diffusion equation

$$Q = -\operatorname{div}(K \cdot \operatorname{grad} u).$$

If the conductivity is the identity the right hand side of the equation becomes the Laplacian:

$$Q = -\Delta u.$$

The discretization yields a system of linear equations

$$\mathbf{A}\mathbf{u} = \mathbf{q}. \tag{21}$$

For numerical testing we investigate the discrete solution of (21) with the exact solution. The entries of the vectors $\mathbf{q}$ and $\mathbf{u}$ are the discrete values that attributed to the cell centers. Note that $\mathbf{A}$ is sparse, thus we use the biconjugate gradient method to solve the system. Let $u(x, y, z) \in C(D)$ be a twice continuously differentiable real function on the domain $D \in \mathbb{R}^d$ with dimension d . For numerical testing we calculate $\mathbf{u}$, the right hand side of (21). Let C_i be a cell with boundary $\partial C_i = \{F_j \mid F_j \in \partial C_i\}$. The flux balance u_i of C_i is computed by the sum over the fluxes f_j over the faces F_j:

$$u_i = \sum_j f_j.$$

We remember that the flux over the face F_j with outer normal unit vector n is given by:

$$f_j = \int_{F_j} \nabla u \cdot n \, \mathrm{ds}.$$

In the implementation we approximated the integral by the gradient taken in the midpoint $m_(F_j)$ of the face times a constant *mean* outer normal unit vector n_j factorized with the area of the face A_{F_j}:

$$f_j = u(m_{F_j}) n_j A_{F_j}.$$

As a discrete value of the exact solution we take the function value in the cell centers. Let n_C be the number of cells and Vol_i the volume of cell C_i. The error will be measured in the following norm:

$$e = \frac{\sum_{i=1}^{n_C} \mathrm{Vol}_i(\tilde{u}_i - u_i)}{\sum_{i=1}^{n_C} \mathrm{Vol}_i},$$

In addition we investigate the maximal error that is measured by:

$$e_{max} = \max_{i=1}^{n_C},$$

3.1 Horizontal Grids

3.1.1 Description of the Setting

Test case (1) is performed with a Gaussian bell curve which is *wrapped* around the sphere with its peak located in the spherical coordinates $[\phi_0, \lambda_0]$ ($\phi_0 \epsilon [-\frac{\pi}{2}, \frac{\pi}{2}]$, $\lambda_0 \epsilon [\pi, \pi]$) as the solution u. It is given by:

$$u(\text{dist}) = e^{-\text{dist}^2},$$

$$\text{dist} = \arccos(\sin(\phi_0)\sin(\phi) + \cos(\phi_0)\cos(\phi)\cos(\lambda - \lambda_0))$$

$$\phi = \arccos \frac{z}{\sqrt{x^2 + y^2 + z^2}} - \frac{\pi}{2},$$

$$\lambda = \text{atan2}(y, x) = \begin{cases} \arccos \frac{x}{\sqrt{x^2+y^2}} & \text{für } y \geq 0, \\ 2\pi - \arccos \frac{x}{\sqrt{x^2+y^2}} & \text{für } y < 0; \end{cases}$$

Test case (2) is performed with two added bell curves with different maxima located in different positions on the sphere.

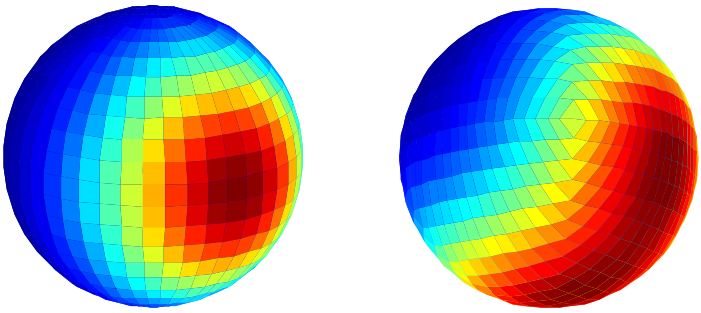


Figure 13: Solutions of Testcases (1) and (2) on a lat/long bzw. Quadrilateral grid

3.1.2 Results

We performed the computing on grids with $n \times 2n$ grid cells. For the spherical grids (abbr.: sphere) we use $2n$ cells along the equator and n cells in longitudinal direction to make sure the cells are rather quadratic. The alternative quadrilateral grid (abbr.: Quad) requires a rectangle with side lengths $l \times 2l$ to be *wrapped* around the sphere. The centroidal Voronoi tessellation is performed with $n \times 2n$ initial points. We investigate grids generated by centroidal Voronoi Tessellation with 50 steps of iteration (abbr.:CVT50), whose cells are rather uniform and grids generated by only 10 steps of iteration (abbr.:CVT10). The shape of the cells of the CVT10 grid are less uniformly allocated on the sphere than those of CVT50. The skewer grids give us the possibility to see the full scope of the MPFA Method. The fifth grid is the uniformly refined dodecahedron (abbr.: Dode). Due to the refinement we have no influence in the number of cells N_D. As we plot the errors in dependence of

$h = 1/n$, for the dodecahedron h is given by: $h = \sqrt{\frac{2}{N_D}}$. The diagrams of the errors and maximal errors (fig.(14) and fig.(15)) show quadratic convergence behavior of the error and maximal error.The errors and maximal errors can be found in tables (1), (2), (3) and (4) in appendix B.

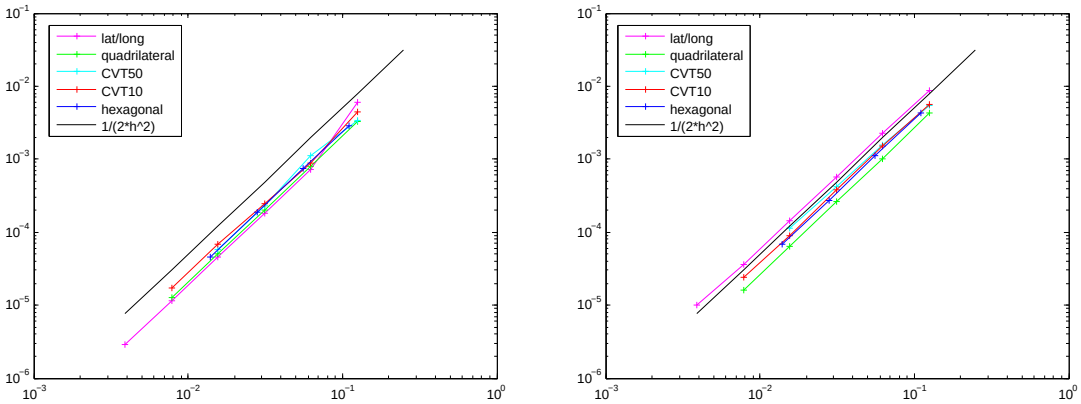


Figure 14: Plots of the error for Test Cases (1) and (2) on Horizontal Grids.

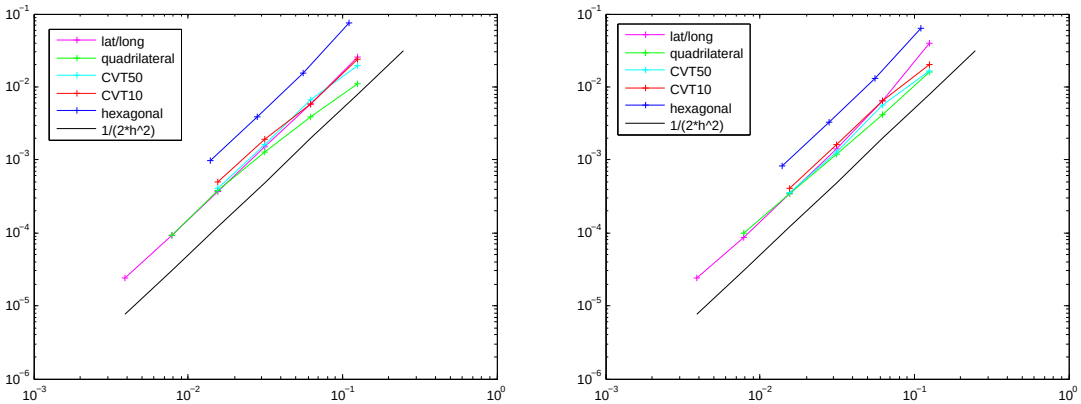


Figure 15: Plots of the maximal error for Test Cases (1) and (2) on Horizontal Grids.

3.2 Vertical Grids

In this subsection we want to present the numerical tests of the vertical grid configurations. We compare the convergence of diffusion on terrain following coordinates with cut cells.

3.2.1 Description of the Setting

The computational domain is the unit cube in $\mathbb{R}^d$ with $d = 2, 3$. We design a mountain as an orography function on the ground level. The mountain and therewith the z-Koordinate

des ground level is defined by the function:

$$z = \frac{hw}{1+x} \text{ in 2D} \quad \text{and} \quad z = \frac{hw}{1+\sqrt{x^2+y^2}}, \text{in 3D}$$

with height h and width w, known as the *witch of agnesi*. Figure (16) shows a very coarse 3D grid of the domains with a mountain of height $h = 0.5$ and width $w = 0.25$. The test is

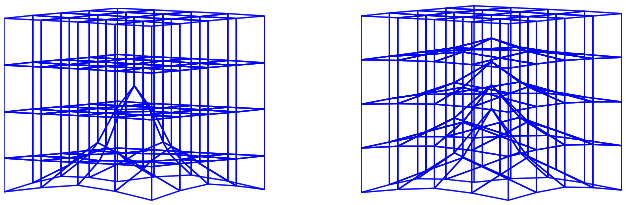


Figure 16: Grid Configurations for the Orography Grids

performed with the solution:

$$u = \sin(\pi x)\sin(\pi y)\sin(\pi z)$$

and Dirichlet boundary conditions. Figure (17) shows a plots of the solution on the horizontal grid layers of both grid configurations in 3D.

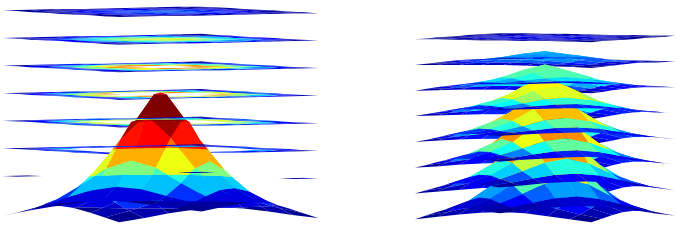


Figure 17: Solution of the Vertical Grids.

3.2.2 Results

We performed tests in the 2D and 3D scenarios. The grids are generated by $n \times n$ or $n \times n \times n$ grid cells. Figures (18) shows the errors and maximal of the 2D case in dependance of $h = 1/n$, figures (19) the 3D case. the values of the errors can be looked up in tables (5) and (6) in appendix B. All convergence plots show quadratic behavior. For our test scenario the terrain following coordinates show a smaller *general* error, but when we investigate the *maximal* error the cut cells succeed, even though in the 2D case not as distinctive as in 3D.

3.3 Discussion of the Results

Summing up the convergence results from our test cases we observed quadratic behavior in all cases. In the literature we find the same results [1]. The quadratic convergence is

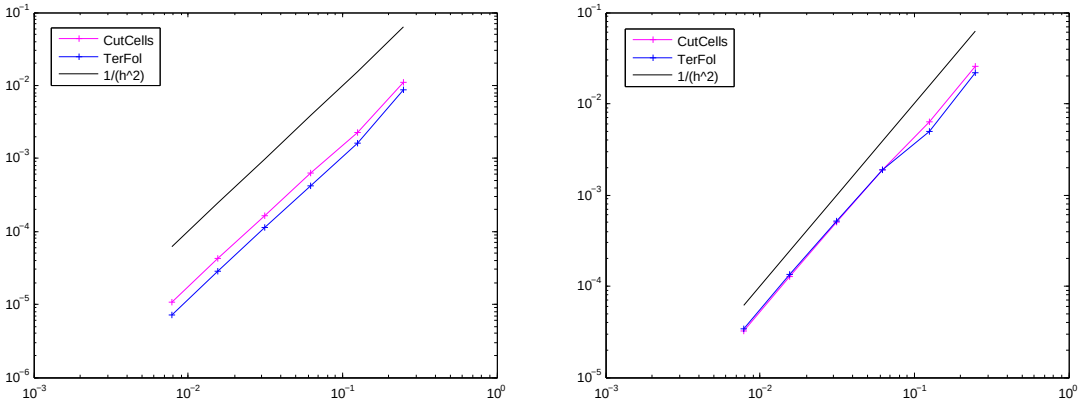


Figure 18: Convergence Plots

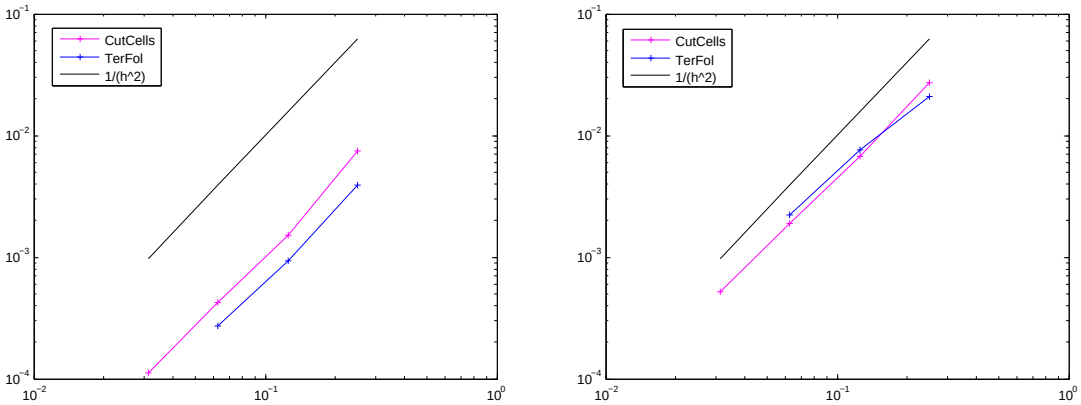


Figure 19: Convergence Plots

proofed for almost all polyhedral grids [3] One constraint that is made in this proofs is about the size. The volume must exceed a certain lower boundary. This constraint is not fulfilled by cut cells, which might be very small. The quadratic convergence behavior of the maximal error for the cut cells is thus not just a proof of proper implementation of the MPFA method.

4 Summary and Future Prospects

4.1 Summary

The main question of this work was to investigate if the Multi Point Flux Approximation Method is a suitable way to discretize diffusion in atmospherical simulation. In atmospherical modeling many different kinds of meshes are used. In horizontal direction the spherical shape of the earth must be taken into account. I implemented four different kinds of spherical grids; a rectangular longitude/latitude grid, a hexahedral grid emerging by refinement of the dodecahedron, a grid generated by centroidal voronoi tessellation and a logical cartesian quadrilateral grid on the sphere, that is constructed by wrapping a cartesian grid around the sphere. In horizontal direction the orography, the mountains and valleys of the earth's surface must be taken into account. I implemented terrain following coordinates and cut cell coordinates. I developed an approach to generate cut cells out of a cartesian grid. To be able to apply the Multi Point Flux Approximation Method on this various grids, we put all the grids in a framework of general polyhedral 3dimensional grids. I developed a Matlab code to apply the MPFA Method into this general framework.
I ran tests on the horizontal and vertical grids to investigate the convergence behavior. The test cases on the horizontal grid yield quadratic convergence for diffusion. The results of vertical tests yield the same convergence rate. The general error was smaller for terrain following coordinates whereas the maximal error under-runs for the cut cells. The quadratic convergence for the cut cells is notable, since the convergence is only proven for meshes which satisfy a lower bound of cell size [12].
The quadratic convergence for all tested grids yield a satisfying result to the main question of applicability of the MPFA method to atmospheric simulation. Further investigation seems worthwhile.

4.2 Future Prospects and Open Questions

This work tested the MPFA method on various meshes that are used for meteorological simulation. Due to the wide variety the tests we can give a general assumption over the applicability. But deeper tests have to be performed to support this assumption and to figure out the full scope and advantages and disadvantages of the MPFA method especially in meteorological modeling. The development of the error must be compared to hitherto existing methods of discretization of diffusion. The performance of more realistic test scenarios might yield different results than computation on an artificial mountain in the unit cube. To get a solid statement in the comparison between cut cells and terrain following coordinates one may design more test examples, steepen the hills, create valley etc. The computational effort of the MPFA method rather high. The accuracy must be considered under the aspect of computational effort.
Theoretical investigation of the convergence of the discretization of diffusion on cut cells is another task that is constituted by this work. The cut cells may be very small, which may cause problems and excludes them from hitherto theoretical results.

For the application of the MPFA method into a general polyhedral framework, I touched the field of discrete differential geometry. [5] Theoretical results over polyhedral meshes use the notions and some results from this mathematical field. [3] For theoretical investigations of cut cells one might find fruitful parallels.
To apply the plain spherical meshes into the code that was conducted for polyhedral, 3dimensional grids, we inflated the mesh. One might refer to offset meshes as well. In case of a sphere not much confusion might happen. Thus another risen question is the inflation of objects with non constant (or even non smooth or non continuous) curvature, the development of a discrete Laplace-Beltrami-operator.
In the implementation many things of course could be enhanced. In the case of cut cells the MPFA method takes way to much effort on the non boundary cells, because on cartesian grids the MPFA method becomes the simple 5point stencil.

Appendix A - Geometry

In section (MPFA) geometrical notions like midpoints, areas and normal vectors of polygons and the volume of polyhedra were used without proper definition. This shall be made up here.
The implementation was conducted for polyhedral grids. The faces of the cells are generally non-planar polygons. In the following we want to give the exact definitions for the notions that we used for implementation and show some alternatives and the development of the discretization error.

Midpoint

The notion of midpoints is used for edges, faces and cells. For an edge there might not happen any confusion because the edges are assumed to be straight and not bent and the midpoint M_e of two points of an edge (P_{e1}, P_{e2}) is uniquely defined:

$$P_{ei} = \begin{pmatrix} x_{ei} \\ y_{ei} \\ z_{ei} \end{pmatrix}, \ i = 1, 2 \tag{22}$$

$$M_e = \frac{1}{2} \begin{pmatrix} (x_{e1} + x_{e2}) \\ (y_{e1} + y_{e2}) \\ (z_{e1} + z_{e2}) \end{pmatrix} \tag{23}$$

For the Midpoints of higher dimensional elements we might use an generalization of formula (25). Let $P_i, i = 1, ..., n_i$ the vertices of the object.

$$P_{ei} = \begin{pmatrix} x_{ei} \\ y_{ei} \\ z_{ei} \end{pmatrix}, \ i = 1, ..., n_i \tag{24}$$

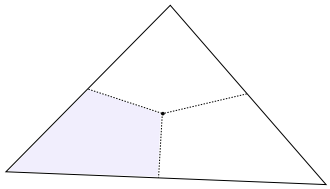
The *barycenter of vertices* is defined by:

$$M_e = \frac{1}{n_i} \begin{pmatrix} \sum_{i=1}^{n_i} x_i \\ \sum_{i=1}^{n_i} y_i \\ \sum_{i=1}^{n_i} z_i \end{pmatrix} \tag{25}$$

Note that the faces and cells need to be convex to make sure that the midpoint is inside the object [1].

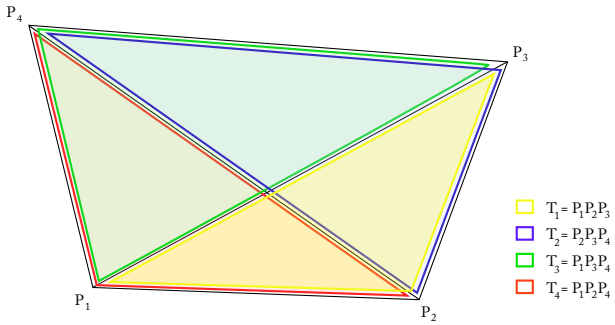
Area of a non-planar convex Polygon

Each Polygon can be partitioned into quadrilaterals by joining the midpoint of the polygon with the midpoints of the edges. The calculation of the area of a non-planar polygon occurs via summation over the areas of these quadrilaterals. Though the task is to compute the area of a non planar quadrilateral.



Averaging

The area of a non-planar quadrilateral can be estimated by twice the average value of the areas of the triangles that can be inscribed.



Bilinear Form

The non planar quadrilateral, spanned by the vertices P_1, P_2, P_3, P_4 can be approximated by the bilinear function:

$$S(u,v) = c_{00} + c_{01}u + c_{10}v + c_{11}uv \quad 0 \leq u, v \leq 1, \tag{26}$$

where

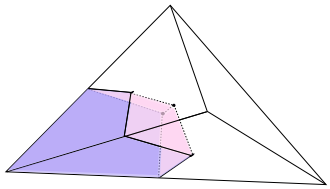
$$c_{00} = P_1, \quad c_{01} = P_2 - P_1, \quad c_{10} = P_4 - P_1, \quad c_{11} = P_3 - P_2 - P_4 + P_1.$$

The surface area of the quadrilateral can be computed by integrating the surface element. This leads to [4]:

$$\int_0^1 \int_0^1 \parallel S_u \times S_v \parallel dudv = \int_0^1 \int_0^1 \parallel (c_{01} - c_{11}v) \times (c_{10} - c_{11}u) \parallel dudv \tag{27}$$

Volume

The computation of the volume of a polyhedron occurs analogously to the computation of the area of a polygon. The only restriction to the grid is the corner constraint, namely that exactly three edges and faces meet in each corner of every polyhedron. This ensures that the subcells of the polyhedron are cuboids. So again, we will compute the volume of the



cuboids and add them together to gain the volume of the entire polyhedron.
For the computing of the volumes of the cuboids we use a formula excerpted from [10]:

$$\begin{aligned} 12\,\mathrm{vol} = \; & [(\vec{v}_7 - \vec{v}_0), (\vec{v}_1 + \vec{v}_6) - (\vec{v}_4 + \vec{v}_5). - (\vec{v}_1 - \vec{v}_6) + (\vec{v}_4 - \vec{v}_5)] + \\ & [(\vec{v}_7 - \vec{v}_0), (\vec{v}_1 + \vec{v}_6) - (\vec{v}_3 + \vec{v}_2), (\vec{v}_1 - \vec{v}_6) + (\vec{v}_3 - \vec{v}_2)]. \end{aligned} \tag{28}$$

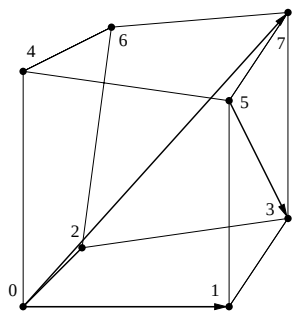


Figure 20: Enumeration of the Vertices of a Cuboid.

Normal Vector

The normal vector of a non planar surface is not a constant. An average is estimated by integrating the normal over the quadrilateral subcell. We consider the bilinear form that describes the quadrilateral (26). The normal vector of a certain point (u_p, v_p) of the surface is given by the cross product of the two partial derivatives.

$$\vec{n}(u_p, v - p) = S_u(u_p, v_p) \times S_v(u_p, v_p) \tag{29}$$

The partial derivatives are given by:

$$S_u = (c_{01} - c_{11}v) = v(P_4 - P_1) + (1 - v)(P_3 - P_2), S_v = (c_{10} - c_{11}u). \tag{30}$$

Integration over the unit square leads to:

$$\hat{n} = \frac{1}{4}((P_2 - P_1) \times (P_4 - P_2) + (P_3 - P_4) \times (P_4 - P_2) + (P_2 - P_1) \times (P_3 - P_2) + (P_3 - P_4) \times (P_3 - P_2)) \tag{31}$$

Appendix B - Material from Section 4

Gradients

In this subsection we present the gradients of the solution functions that were used for the numerical testing in section (4)

Horizontal Grids The solution function is given by:

$$u(\text{dist}) = e^{-\text{dist}^2},$$

$$\text{dist} = \arccos(\sin(\phi_0)\sin(\phi) + \cos(\phi_0)\cos(\phi)\cos(\lambda - \lambda_0))$$

$$\phi = \arccos\frac{z}{\sqrt{x^2+y^2+z^2}} - \frac{\pi}{2},$$

$$\lambda = \text{atan2}(y,x) = \begin{cases} \arccos\frac{x}{\sqrt{x^2+y^2}} & \text{für } y \geq 0, \\ 2\pi - \arccos\frac{x}{\sqrt{x^2+y^2}} & \text{für } y < 0; \end{cases}$$

The gradient, which we use for computation of the right hand side is:

$$\nabla u = u_{\text{dist}} \nabla \text{dist}$$

with the derivatives:

$$u_{\text{dist}} = -2(\text{dist})u,$$

$$\nabla \text{dist} = \text{dist}_\phi \nabla\phi + \text{dist}_\lambda \nabla\lambda$$

$$\text{dist}_\phi = (-\cos(\phi)\sin(\phi_0) + \sin(\phi)\cos(phi_0) * \cos(\lambda - \lambda_0))/\sin(\text{dist})$$

$$\text{dist}_\lambda = \cos(\phi)\cos(\phi_0)\sin(\lambda - \lambda_0)/\sin(\text{dist})$$

$$\nabla\phi = \begin{pmatrix} \frac{xz}{\sqrt{x^2+y^2}(x^2+y^2+z^2)} \\ \frac{yz}{\sqrt{x^2+y^2}(x^2+y^2+z^2)} \\ -\frac{\sqrt{x^2+y^2}}{x^2+y^2+z^2} \end{pmatrix}$$

$$\nabla\lambda = \begin{pmatrix} -\frac{y}{y^2+x^2} \\ \frac{x}{y^2+x^2} \\ 0 \end{pmatrix}$$

Vertical Grids The solution function for the vertical test cases is given by:

$$u = \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

Though the gradients is:

$$\nabla u = \begin{pmatrix} \pi\cos(\pi x)\sin(\pi y)\sin(\pi z) \\ \pi\cos(\pi y)\sin(\pi x)\sin(\pi z) \\ \pi\cos(\pi z)\sin(\pi y)\sin(\pi x) \end{pmatrix}$$

Tables of Errors and Maximal Errors

Table 1 The first table shows the values of the error e of different partitions for test case (1) on the horizontal grids.

n	h		Sphere	Quad	CVT50	CVT10	Dode
$2 \cdot 2^4$	2^{-2}	e	0.4869	0.0138	0.0156	0.0159	0.0115 (n=42)
$2 \cdot 2^6$	2^{-3}	e	0.0060	0.0033	0.0034	0.0045	0.0029 (n=162)
$2 \cdot 2^8$	2^{-4}	e	7.1493e-4	8.0157e-4	0.0011	8.9501e-4	7.4452e-4 (n=642)
$2 \cdot 2^{10}$	2^{-5}	e	1.8101e-4	2.0133e-4	2.3042e-4	2.4592e-4	1.8461e-4 (n=2562)
$2 \cdot 2^{12}$	2^{-6}	e	4.5210e-5	5.0673e-5	5.743e-5	6.7342e-5	4.587e-5 (n=10242)
$2 \cdot 2^{14}$	2^{-7}	e	1.1313e-5	1.2805e-5		1.6911e-5	
$2 \cdot 2^{16}$	2^{-8}	e	2.9230e-6				

Table 2 The table shows the maximal error e_{max} of the different partitions for test case (1) on the horizontal grids.

n	h		Sphere	Quad	CVT50	CVT10	Dode
$2 \cdot 2^4$	2^{-2}	e_{max}	1.8569	0.0323	0.0871	0.00843	0.0763 (n=42)
$2 \cdot 2^6$	2^{-3}	e_{max}	0.0256	0.0111	0.0196	0.0242	0.0156 (n=162)
$2 \cdot 2^8$	2^{-4}	e_{max}	0.0058	0.0039	0.0067	0.0059	0.0039 (n=642)
$2 \cdot 2^{10}$	2^{-5}	e_{max}	0.0015	0.0013	0.0016	0.0019	9.8597e-4 (n=2562)
$2 \cdot 2^{12}$	2^{-6}	e_{max}	3.6904e-4	3.7702e-4	4.0998e-4	4.8430e-4	2.4800e-4 (n=10242)
$2 \cdot 2^{14}$	2^{-7}	e_{max}	9.2597e-5	9.3040e-5			
$2 \cdot 2^{16}$	2^{-8}	e_{max}	2.4111e-5				

Table 3 The table shows the values of the error of horizontal grids with n cells for the second test case.

n		Sphere	Quad	VTS50	VTS10	Dode
32	e	0.4447	0.0219	0.0314	0.0265	0.0180 (n=42)
128	e	0.0088	0.0043	5.457e-3	0.0057	0.0043 (n=162)
512	e	0.0023	0.0010	1.571e-3	0.0015	0.0011 (n=642)
2048	e	5.7377e-4	2.5965e-4	4.253e-4	3.8023e-4	2.715e-4 (n=2562)
8192	e	1.432e-4	6.4868e-5	1.135e-4	8.8233e-5	6.75e-5 (n=10242)
32768	e	3.6050e-5	1.6311e-5		2.3859e-5	
131072	e	1.0094e-5				

Table 4 The table contains the values of the maximal horizontal grids with n cells for the second test case.

n		Sphere	Quad	VTS50	VTS10	Dode
32	e	1.5892	0.0611	0.0800	0.0805	0.0645(n=42)
128	e	0.0392	0.0160	0.0165	0.0199	0.0131(n=162)
512	e	0.0064	0.0041	0.0056	0.0064	0.0033(n=642)
2048	e	0.0014	0.0012	0.0013	0.0016	8.2335e-4 (n=2562)
8192	e	3.4563e-4	3.4843e-4	3.5565e-4	4.0362e-4	2.0592e-4(n=10242)
32768	e	8.7174e-5	9.9137e-5			
131072	e	2.4150e-5				

Table 5 The fifth table shows the general and maximal error of the vertical 2D tests.

n	e	Cut Cells	TerFol	e_{max}	CutCell	TerFol
1	e	0.0110	0.0087	e_{max}	0.0259	0.0218
2	e	0.0023	0.0016	e_{max}	0.0064	0.0050
3	e	6.3229e-4	4.2399e-4	e_{max}	0.0019	0.0019
4	e	1.6650e-4	1.1212e4	e_{max}	4.967e-4	5.1976e-4
5	e	4.274e-5	2.8784e-5	e_{max}	1.282e-4	1.3480e-4
6	e	1.0828e-5	7.2848e-6	e_{max}	3.2581e-5	3.4231e-5

Table 6 The table contains the errors and maximal errors of the vertical 3D tests.

n	e	Cut Cells	TerFol	e_{max}	CutCell	TerFol
1	e	0.0074	0.0039	e_{max}	0.0270	0.0207
2	e	0.0015	9.3627e-4	e_{max}	0.0067	0.0076
3	e	4.2183e-4	2.7230e-4	e_{max}	0.0019	0.0022
4	e	1.1108e-4		e_{max}	5.1827e-4	

Hiermit versichere ich, dass ich diese Arbeit selbstständig und nur mit den angegebenen Quellen angefertigt habe.

Anna Hartkopf

References

[1] I. Aavatsmark, G. Eigestad, and R. Klausen. Numerical convergence of the MPFA O-method for general quadrilateral grids in two and three dimensions. *Compatible spatial discretizations*, pages 1–21, 2006.

[2] J. Alberty, C. Carstensen, and S.A. Funken. Remarks around 50 lines of Matlab: short finite element implementation. *Numerical Algorithms*, 20(2):117–137, 1999.

[3] F. Brezzi, K. Lipnikov, and M. Shashkov. Convergence of mimetic finite difference method for diffusion problems on polyhedral meshes. 43(5):1872–1896, 2004.

[4] D. Calhoun and C. Helzel. A finite volume method for solving parabolic equations an logically cartsian curved surface meshes. *Journal of Scientific Computing*, (31):4066–4099, 2009.

[5] M. Desbrun, E. Kanso, and Y. Tong. Discrete differential forms for computational modeling. In *ACM SIGGRAPH 2006 Courses*, pages 39–54. ACM, 2006.

[6] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[7] Q. Du, M.D. Gunzburger, and L. Ju. Constrained centroidal Voronoi tessellations for surfaces. *SIAM Journal on Scientific Computing*, 24(5):1488–1506, 2003.

[8] G. Fischer. Lineare Algebra, Vieweg, 9, 1986.

[9] Tzvi Gal-Chen and Richard C. J. Somerville. On the use of a coordinate transformation for the solution of the navier-stokes equations. *Journal of Computational Physics*, (17):209–228, 1975.

[10] J. Grandy. Efficient computation of volume of hexahedral cells. Technical report, Lawrence Livermore National Lab., CA (United States), 1997.

[11] R.A. Klausen and A.F. Stephansen. Mimetic MPFA. In *Proc. 11th European Conference on the Mathematics of Oil Recovery*, 2008.

[12] R.A. Klausen and A.F. Stephansen. Convergence of the MPFA O-method on general grids, 2010.

[13] R.A. Klausen and R. Winther. Convergence of multipoint flux approximations on quadrilateral grids. *Numerical Methods for Partial Differential Equations*, 22(6):1438–1454, 2006.

[14] D. Majewski, D. Liermann, P. Prohl, B. Ritter, M. Buchhold, T. Hanisch, G. Paul, W. Wergen, and J. Baumgardner. The operational global icosahedral-hexagonal grid-point model GME: Description and high-resolution tests. *Monthly Weather Review*, 130(2):319–338, 2002.

[15] T.D. Ringler, M. Gunzburger, and L. Ju. A multi-resolution method for climate system modeling: application of Spherical Centroidal A multi-resolution method for climate

system modeling: Application of Spherical Centroidal Voroni Tessellations. Technical report, Los Alamos National Laboratory (LANL), 2008.

[16] T.D. Ringler, M. Gunzburger, and L. Ju. Voronoi Tessellations and their Application to Climate and Global Modeling. Technical report, Los Alamos National Laboratory (LANL), 2009.

[17] M. Satoh, T. Matsuno, H. Tomita, H. Miura, T. Nasuno, and S. Iga. Nonhydrostatic icosahedral atmospheric model (NICAM) for global cloud resolving simulations. *Journal of Computational Physics*, 227(7):3486–3514, 2008.

[18] J. Steppeler, G. Doms, U. Schättler, HW Bitzer, A. Gassmann, U. Damrath, and G. Gregoric. Meso-gamma scale forecasts using the nonhydrostatic model LM. *Meteorology and Atmospheric Physics*, 82(1):75–96, 2003.

[19] R.L. Walko and R. Avissar. The ocean-land-atmosphere model (OLAM). Part II: formulation and tests of the nonhydrostatic dynamic core. *Monthly Weather Review*, 136(11):4045–4062, 2008.