TESIS DE LA UNIVERSIDAD
DE ZARAGOZA

2013

128

Pablo Salinas Cortés

Semi-Structured multigrid methods on Voronoi meshes to the resolution of the Darcy-Oberbeck-Boussinesq model

Departamento

Ingeniería Mecánica

Director/es

Gaspar Lorenz, Francisco José
Lisbona Corté, Francisco Javier

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

Tesis Doctoral

SEMI-STRUCTURED MULTIGRID METHODS ON VORONOI MESHES TO THE RESOLUTION OF THE DARCY-OBERBECK-BOUSSINESQ MODEL

Autor

Pablo Salinas Cortés

Director/es

Gaspar Lorenz, Francisco José
Lisbona Corté, Francisco Javier

UNIVERSIDAD DE ZARAGOZA

Ingeniería Mecánica

2013

Semi-Structured multigrid methods on Voronoi meshes to the resolution of the Darcy-Oberbeck-Boussinesq model

Pablo Salinas Cortés

Applied mathematics

University of Zaragoza

A thesis submitted for the degree of

Doctor (PhD) of Computational Mechanics

2013 June

To my beloved Cristina Gargallo

Acknowledgements

Regarding the mathematical team, firstly I would like to thank Francisco Gaspar for offering me the possibility of doing a PhD, and for his help during these years whenever I needed it. Also, I want to expose my gratitude to Francisco Lisbona since he always explained me any problem I faced, he is like the encyclopaedia of the group. And last but not least, I want to give many thanks to Carmen Rodrigo because she always was there for quick doubts, except when she was in the other side of the world. I want to emphasize my gratitude for their help correcting this thesis.

Special thanks to Kees Oosterlee for hosting me in Amsterdam and showing me Delft during my staying in the Netherlands, where I had an incredible experience.

I would like to express my gratefulness to my family, Javier, Margarita and David, for their support, patience and for cheering me on during this project despite at the beginning it seemed very difficult to fulfil.

And finally, I want to thank to my engaged Cristina since without her constant support and patience listening to me talking about "triangulitos", I would not have been able to finish this thesis.

Resumen

La mayoría de los problemas en física e ingeniería se modelan mediante ecuaciones en derivadas parciales. Algunos de los modelos más representativos serían, por ejemplo, las ecuaciones de Maxwell, el problema de la elasticidad o las ecuaciones de Navier-Stokes.

Generalmente, la resolución de las ecuaciones en derivadas parciales se basa en el uso de métodos numéricos que comienzan con un proceso de discretización y finalizan con la resolución de un gran sistema de ecuaciones algebraicas que puede llegar a ser muy costosa. El desarrollo de métodos eficientes para esta última tarea es muy importante pues repercute en la posibilidad de realizar más ensayos con un menor costo computacional, y es el principal objetivo de esta tesis.

Hay dos grandes grupos de métodos numéricos para la resolución de estos sistemas de ecuaciones, los métodos basados en los subespacios de Krylov, y los métodos multimalla. En aplicaciones especialmente difíciles, es habitual combinar ambas metodologías, ya sea usando un método multimalla como precondicionador de un método basado en un subespacio de Krylov o este último como suavizador en el método multimalla. Esta tesis se va a centrar en los métodos multimalla.

Los métodos multimalla (4), que son conocidos por ser métodos muy rápidos para la resolución de problemas de tipo parabólico y elíptico (49), se pueden clasificar en dos familias. Los geométricos que tienen un bajo consumo de memoria, pero solo pueden aplicarse sobre dominios regulares y mallas estructuradas, y los métodos multimalla algebraicos, que requieren más memoria pero se pueden aplicar a dominios de cualquier complejidad. Una alternativa que pretende obtener lo mejor de ambos métodos es el uso de métodos multimalla geométricos en mallas semi-estructuradas. Esta se basa en considerar una malla inicial no estructurada que se adapte a la geometría del dominio, para posteriormente refinar regularmente esta malla inicial hasta obtener una malla suficientemente fina en la que podamos obtener una solución con la precisión deseada.

De esta manera, obtenemos una malla que se adapta al dominio, y que es localmente estructurada, donde podemos aplicar métodos multimalla geométricos. Se han aplicado estos métodos sobre mallas rectangulares y triangulares, en estas últimas considerando discretizaciones centradas en los vértices (20, 21).

Las mallas de Delaunay son ampliamente utilizadas para la triangulación de dominios debido a sus buenas propiedades. La malla dual de una triangulación de Delaunay es conocida como malla de Voronoi. Entre otras buenas propiedades, se cumple que las rectas que unen los nodos de la malla de Voronoi son perpendiculares a los lados de la triangulación de Delaunay. Por tanto, su uso para la discretización de ecuaciones en derivadas parciales es muy interesante, ya que para discretizar las derivadas en la dirección normal basta con un cociente de diferencias. Vamos a considerar triangulaciones acutángulas, por lo que el punto de Voronoi siempre estará dentro del triángulo correspondiente. Sin embargo, su posición cambia en función de la forma del triángulo, lo cual puede dar lugar a grandes anisotropías debidas al mallado y no al problema.

Uno de los componentes más importantes de los métodos multimalla es el suavizador. El estudio de suavizadores para discretizaciones centradas en celdas es escaso, y nulo para mallas semi-estructuradas. Por tanto, el diseño de nuevos suavizadores es uno de nuestros principales objetivos. Para ello, deberemos estudiar las características del problema y diseñar nuevos suavizadores que sean capaces de tener en cuenta dichas características así como de tratar con las posibles anisotropías inducidas por la malla.

A la vez que desarrollamos un método multimalla genérico para dichas discretizaciones, nos centraremos en la resolución del modelo Darcy-Oberbeck-Boussinesq (42), el cual se puede escribir de forma adimensional como sigue:

$$\begin{aligned} \nabla \cdot \mathbf{u} &= 0, && (1)\\ \mathbf{u} &= -(\nabla P - C\mathbf{z}), && (2)\\ \frac{\partial C}{\partial t} &= -\mathbf{u} \cdot \nabla C + \frac{1}{Ra}\Delta C, && (3) \end{aligned}$$

donde $\mathbf{z}$ es el vector $(0, -1)$ en coordenadas cartesianas, y Ra denota el parámetro de Rayleigh, el cual establece la dominancia de la difusión o la convección.

Comenzaremos con el diseño de un método multimalla para discretizaciones de problemas escalares en mallas de Voronoi asociadas a triangulaciones semi-estructuradas, más concretamente consideraremos el problema de Laplace. Tras obtener un multimalla eficiente para este problema, iremos añadiendo complejidad al modelo, materiales no homogéneos, convección dominante, y finalmente el problema de evolución dado en la ecuación (3).

Sin embargo, para la resolución del modelo de Darcy-Oberbeck-Boussinesq, hay que tener en cuenta además el sistema de ecuaciones acopladas dado por (1) y (2), con incógnitas de presión, P, y velocidad, $\mathbf{u}$. Para ello, localizaremos las proyecciones de la velocidad en los puntos medios de los lados de los triángulos y la presión en los puntos de Voronoi. En cuanto al diseño del método multimalla correspondiente tenemos dos opciones, el uso de suavizadores de tipo Vanka, que involucran la resolución de un sistema de ecuaciones para cada punto de la malla, o suavizadores de tipo distributivo. A lo largo de esta tesis, analizaremos el comportamiento de ambos tipos de suavizadores.

El modelo Darcy-Oberbeck-Boussinesq es un modelo que se utiliza para calcular el comportamiento de flujos en medios porosos saturados, los cuales se mueven debido a diferencias de densidad dentro del fluido. Este modelo puede ser utilizado para la simulación del problema de almacenaje de CO_2 en acuíferos salinos. Notemos que la velocidad y la concentración están acopladas, haciendo que el problema sea altamente no lineal. De hecho, al poco tiempo del comienzo del proceso se crean inestabilidades ("fingering") difícilmente modelizables que exigen pasos de tiempo muy pequeños.

Más concretamente esta tesis está dividida en los siguientes capítulos:

En el Capítulo 2 comenzaremos con un problema simple, el problema de Laplace, para desarrollar un método multimalla para discretizaciones sobre mallas de Voronoi asociadas con mallas triangulares semi-estructuradas. Primero, discretizaremos el problema, luego desarrollaremos un método multimalla en mallas estructuradas y analizaremos su comportamiento para diferentes geometrías de la malla, diseñando nuevos suavizadores cuando sea requerido. Tras este proceso, desarrollaremos un análisis local de Fourier para comprobar que estos nuevos componentes funcionan correctamente. Finalmente, aplicaremos esta metodología en mallas semi-estructuradas en dos experimentos, uno para el problema de Laplace, y otro para un problema de convección-difusión con difusión dominante. Además, explicaremos la estructura de datos utilizada para trabajar con nodos centrados en celdas así como algunos detalles para extender un método multimalla sobre mallas estructuradas a una malla semi-estructurada.

En el Capítulo 3, vamos a añadir complejidad al problema de Laplace. Primero, nos enfrentaremos con materiales no homogéneos, para lo cual deberemos usar el método de Galerkin para la discretización en las mallas bastas. En la siguiente sección, añadiremos un término de convección dominante, el cual discretizaremos mediante un esquema upwind, y luego comprobaremos la buena convergencia del método multimalla. Finalmente, añadiremos un término dependiente del tiempo para resolver la ecuación de advección. Tras ser capaces de resolver la ecuación de advección, en el Capítulo 4 nos centraremos en la Ley de Darcy acoplada con la ecuación de conservación de masa:

$$\mathbf{u} = -(\nabla P' - C\mathbf{z}), \tag{4}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{5}$$

Para ello, extenderemos la metodología desarrollada en el caso de mallas centradas en celdas a mallas decaladas, donde las proyecciones de la velocidad son almacenadas en los puntos medios de los lados y la presión en los centros de las celdas. Sin embargo, la extensión directa de estos métodos a mallas decaladas no nos proporciona un método multimalla tan eficiente como deseamos, por ello, desarrollaremos un suavizador distributivo que se mostrará muy rápido y eficaz.

Finalmente, en el Capítulo 5, tras haber comprobado que hemos desarrollado unos métodos de resolución robustos y estables, nos enfrentaremos al modelo Darcy-Oberbeck-Boussinesq para simular el almacenamiento de CO_2 en diferentes dominios. Estudiaremos las inestabilidades que aparecen dependiendo de la importancia del término difusivo frente al convectivo. Para resolver el modelo, usaremos un método de punto fijo para desacoplar las ecuaciones.

El Apéndice A contiene una introducción básica de los métodos multimalla, sus componentes (suavizadores, operadores de transferencia y el operador en las mallas bastas), tipos de ciclos y también se proporciona una pequeña explicación del análisis local de Fourier.

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

The majority of the problems in physics and engineering are modelled by using partial differential equations. Some representative models are Maxwell equations in electromagnetism, elasticity equation in deformable solid theory and Navier-Stokes equations in fluid dynamics.

Usually, the solution of partial differential equations is based on the use of numerical methods that start with a discretization process and finish with the resolution of a large system of algebraic equations, which can be very costly. The development of efficient methods for this latter task is fundamental since affects the possibility of performing complex simulations with a low computational cost. That is the main objective of this thesis. There are two main groups of numerical methods for the resolution of these systems of equations, the Krylov subspace methods, and the multigrid methods. In some difficult applications, it is usual to combine both methodologies, a multigrid method can be used as preconditioner of a Krylov subspace method or this latter as smoother for a multigrid method. This thesis will be focused on multigrid methods.

Multigrid methods (4) are known to be fast solvers for solving parabolic and elliptic problems (49). They can be classified in two families. Geometric multigrid methods that have a low memory consumption but with the drawback that they can only be used in regular domains with a structured mesh, and algebraic multigrid methods that require more memory but they can be used in any kind of domain, no matter its complexity. One alternative taking the best of both approaches is the use of geometric multigrid methods in semi-structured meshes. This methodology consists in creating an initial unstructured mesh that fits the geometry of the domain, and after to refine it in a regular way to reach a target fine grid. In this way, we obtain a mesh that fits the geometry of the domain and is locally structured, in which we can apply geometric multigrid methods. This methodology has been successfully apply on rectangular and triangular meshes, this latter by using vertex-centred discretizations (20, 21).

Delaunay grids are widely used to triangularize a domain due to their good properties. The dual of the Delaunay mesh is known as Voronoi grid. The edges between the vertexes of the Voronoi mesh are perpendicular to the edges of the Delaunay triangulation. Therefore, its use to discretize PDEs is very interesting, as to discretize the flux across the boundaries of a triangle it is enough to use a difference quotient between Voronoi nodes. We will use acute Delaunay triangulations, so the Voronoi points will always fall inside the corresponding triangles. Nevertheless, the position of the Voronoi points varies depending on the shape of the triangles, which may yield strong anisotropies based on geometrical facts and not due to the differential problem.

One of the key components in multigrid methods is the smoother. The study of smoothers for cell-centred discretizations is almost missing in the literature. Therefore, the design of new relaxation methods is one of our main objectives. For that, we will have to design new smoothers capable to deal with the peculiarities of the problem and with the anisotropy introduced by the mesh.

As a final application of the numerical solvers previously developed, we focus ourselves in the resolution of the Darcy-Oberbeck-Boussinesq model (42), which can be written in nondimensional form as follows:

$$\begin{aligned}
\nabla \cdot \mathbf{u} &= 0, && (1.1)\\
\mathbf{u} &= -(\nabla P - C\mathbf{z}), && (1.2)\\
\frac{\partial C}{\partial t} &= -\mathbf{u} \cdot \nabla C + \frac{1}{Ra}\Delta C, && (1.3)
\end{aligned}$$

where $\mathbf{z}$ is the vector $(0, -1)$ in cartesian coordinates and Ra denotes the Rayleigh parameter, which establishes the dominance of the diffusion or the convection. Suitable boundary conditions must be added to close the model.

We will start with the development of a multigrid method for the discretization of scalar problems on Voronoi grids associated with a semi-structured triangulation, more concretely we will consider Laplace problem. After obtaining an efficient multigrid method for this problem, we will add more complexity to the model, non-homogeneous materials, and the parabolic problem given in equation (1.3). Nonetheless, for the solution of the Darcy-Oberbeck-Boussinesq, it is necessary to take also into account the coupled system of equations given by (1.1) and (1.2), with the pressure P and the velocity $\mathbf{u}$ as unknowns. For that, we will locate the normal components of the velocity at the midpoints of the edges of the triangles and the pressure at the Voronoi points. Regarding the design of the corresponding multigrid method, we have two possibilities, the use of Vanka-type smoothers, which have to solve a system of equations for each point of the grid, or distributive smoothers. Throughout this thesis, we will compare their behaviour.

Let us remark that the Darcy-Oberbeck-Boussinesq model is a model used to calculate the behaviour of density driven flows inside saturated porous media. This model can be used in the simulation of CO_2 storage in saline aquifers (24). Note that the velocity and the concentration are coupled, making the problem non-linear. In fact, after a small period of time some instabilities start to grow ("fingering"), which are very difficult to model and require the use of small time steps.

The synopsis of the structure of the thesis is as follows. In Chapter 2, we will use a simple model problem, Laplace equation, in order to develop a good multigrid method for discretizations on Voronoi grids associated with triangular semi-structured meshes. First, we will discretize the problem, next we will develop a multigrid method in structured meshes and we will analyse the influence of different triangular geometries, developing new smoothers whenever it is necessary. After this process, we will develop a Local Fourier Analysis to check whether these new methods work properly or not. Finally, we will apply this methodology to semi-structured grids for two different problems, a Laplace problem and a convection-diffusion problem in which the diffusion is dominant. Besides, we will explain the data storage used to work with cell-centred discretizations on triangular grids and also some details to extend the structured multigrid methods to a semi-structured mesh.

In Chapter 3, we will add complexity to the initial model problem. First, we will consider non-homogeneous materials. For that, we will have to use the Galerkin approach as discretization on coarser meshes. In the next section, we will add a dominant convective term, which we will discretize using the upwind scheme, and we will test the performance of the multigrid method. Finally, we will add a time dependent term to solve the transient convection diffusion equation.

After solving the advection model, in Chapter 4 we will focus on the Darcy Law coupled with the continuity equation:

$$\begin{aligned} \mathbf{u} &= -(\nabla P' - C\mathbf{z}), && (1.4) \\ \nabla \cdot \mathbf{u} &= 0. && (1.5) \end{aligned}$$

For this problem, we will extend what we developed in the case of semi-structured cell-centred multigrid in Voronoi meshes to a staggered triangular grid, where the projections of the velocities are located at the midpoints of the edges and the pressure at the Voronoi points. Nevertheless, the simplest extension of these methods will not provide an efficient multigrid method. For that, we will develop a distributive smoother that will result to be a very efficient smoother.

Finally, in Chapter 5, after having tested that we have developed robust and stable methods, we will confront the Darcy-Oberbeck-Boussinesq model in different domains. We will study the instabilities that appear depending on the dominance of the diffusive term or the convective term. In order to solve the problem, we will use a fixed point method to solve the equations.

Appendix A contains a basic introduction to multigrid, the components of multigrid (smoothers, inter-grid transfer operators and operator in coarser meshes), kinds of cycles and also an introduction to the local Fourier analysis.

Chapter 2

Scalar Multigrid methods on semi-structured Voronoi grids

Although algebraic multigrid can also handle very easily cell-centred discretizations on unstructured grids (18), the focus on this work is to consider geometric multigrid methods. Not many authors have applied geometric multigrid methods on cell-centred discretizations and most of these works have been done in rectangular grids. Historically, the pioneer work is due to Wesseling (55), where a multigrid method for interface problems was constructed to simulate oil reservoir problems. This work started a chain reaction of papers focused on this subject, see (28, 29, 30, 31, 53, 54, 56). The W-cycle convergence of these multigrid methods, in the case of natural injection as prolongation, was theoretically analyzed by Bramble et al. (3), and in the case of V-cycle, the convergence was proved by Kwak et al. in (34) using certain weighted prolongation operators. Furthermore, on triangular grids, Kwak et al. (36, 37), proposed a new multigrid method, extending their previous works.

For an irregular domain, it is very common to apply regular refinement to an unstructured input grid. In this way, a hierarchy of locally structured grids is generated. To perform this refinement, each triangle is divided into four congruent ones by connecting the midpoints of their edges. In this way, a hierarchy of grids is obtained, where transfer operators between two consecutive levels can be defined. These grids provide a suitable framework for the implementation of a geometric multigrid algorithm, permitting the use of stencil-based data structures, see (2), requiring only a few stencils to represent the discrete operator for piece-wise constant coefficient problems, reducing drastically the memory required. Here, very simple local inter-grid transfer operators have been chosen to make easier the communication between different input blocks. For this reason, to overcome the weakness of the coarse-grid correction, the design of powerful smoothers results mandatory, since standard relaxation does not provide satisfactory results in most of the cases. Then, here different novel smoothers suitable for the considered cell-centred discretization are introduced. These smoothers are used on each input block of the initial unstructured grid, depending on their shape, to overcome possible anisotropies of the arising Voronoi mesh. Notice that these smoothers could not be implemented on a pure unstructured grid, what involves an advantage of the geometric multigrid method presented here against the algebraic multigrid approach. We are speaking about, for example, the line-type smoothers, very necessary in the case of anisotropic problems.

2.1 Discretization of an homogeneous diffusion problem on Voronoi grids

In this section, the finite volume discretization on Voronoi meshes of a diffusion problem is presented. Such discretization is introduced for unstructured as well as for structured triangular meshes. However, we want to remark that in the numerical experiments section a convection-diffusion problem will be considered, and its corresponding discretization will be obtained following the same steps as for the pure diffusion problem, as we will specify at that point.

2.1.1 Discretization on unstructured triangular grids

We are going to construct a finite volume discretization scheme on the Voronoi mesh associated with a Delaunay triangulation for the following boundary value problem:

$$\begin{aligned} -\Delta v &= f, \text{ in } \Omega, && (2.1)\\ v &= 0, \text{ on } \partial\Omega. && (2.2)\end{aligned}$$

Firstly, we suppose to have a Delaunay triangulation $\mathcal{T}$ on the domain Ω, that satisfies the usual admissibility assumption (see (9)), i.e. the intersection of two different triangles is either empty, a vertex, or a whole edge. Besides, we restrict ourselves to an acute triangulation, since the use of non-acute triangles makes the Voronoi points to fall outside their corresponding triangle.

A Voronoi mesh asociated with a triangulation $\mathcal{T}$, see Figure 2.1, is defined by the centres of the circumscribed circles of the triangles in $\mathcal{T}$. Moreover, the segment connecting two neighboring Voronoi points goes trough the midpoint of the common edge of their corresponding triangles, being also perpendicular to it. This latter property allows us to approximate the derivative in the normal direction, by the difference quotient on both Voronoi points. Using the divergence theorem on a triangle T of the triangulation $\mathcal{T}$, the following balance equation holds:

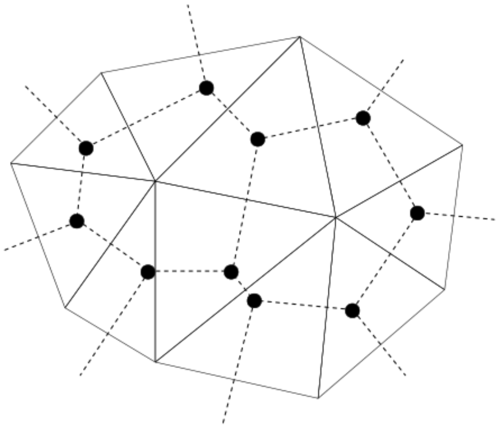


Figure 2.1: Unstructured mesh and its associated Voronoi grid.

$$-\int_{l_1} \nabla v \cdot \mathbf{n}_1 \, \mathrm{d}l_1 - \int_{l_2} \nabla v \cdot \mathbf{n}_2 \, \mathrm{d}l_2 - \int_{l_3} \nabla v \cdot \mathbf{n}_3 \, \mathrm{d}l_3 = \int_T f(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{2.3}$$

where $\mathbf{n}_i$ is the unit outward normal vector to the corresponding edge l_i of triangle T. Now, each of these line integrals is approximated as the length of the corresponding edge multiplied by the flux evaluated in the midpoint of the edge. Afterwards, we approximate such fluxes using the Voronoi points. With regard to the integral in the right-hand side, we consider the following approximation:

$$\int_T f(\mathbf{x}) \, \mathrm{d}\mathbf{x} \approx \mathrm{meas}(T) \, f(\mathbf{x}_c), \tag{2.4}$$

where $\mathbf{x}_c$ denotes the Voronoi point of triangle T, and where meas(T) is the area of T.

Denoting $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ the Voronoi points of the triangles adjacent to T with common edges l_1, l_2 and l_3, respectively, and d_i the distance between points $\mathbf{x}_c$ and $\mathbf{x}_i$,

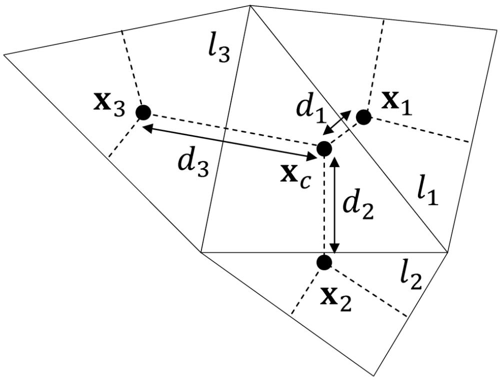


Figure 2.2: Notation for neighbouring Voronoi points on an unstructured grid.

$d_i = \text{dist}(\mathbf{x}_c, \mathbf{x}_i)$, with $i = 1, 2, 3$, (see Figure 2.2), we finally obtain the equation corresponding to node $\mathbf{x}_c$:

$$-\frac{1}{\text{meas}(T)} \sum_{i=1}^{3} \left(\text{meas}(l_i) \, \frac{v_h(\mathbf{x}_i) - v_h(\mathbf{x}_c)}{d_i} \right) = f(\mathbf{x}_c), \tag{2.5}$$

where v_h is the grid function approximating the solution of the continuous problem on the Voronoi points. This scheme is first order accurate in a discrete H^1-norm, whereas it is computationally observed that second order is achieved in L^2-norm. The theoretical analysis of this approach can be seen in (26, 41, 52).

2.1.2 Discretization on structured triangular grids

Now, we are going to consider the particular case of the discretization of problem (2.1) on a structured triangular grid. In this kind of grids, it is very usual to work in stencil notation because it takes advantage of the structured ordering of the unknowns which contribute in the discretization of a fixed grid-point. In a structured grid, any point is surrounded by the same grid-pattern, and using a suitable numbering of the grid-points it is easy to capture this pattern in a small matrix or "stencil" which stores the contributions of the neighbouring unknowns. Then, first of all, a suitable numbering of the grid-points is needed. In triangular grids, a unitary basis of $\mathbb{R}^2$, $\{\mathbf{e}_1, \mathbf{e}_2\}$, where $\mathbf{e}_1$, and $\mathbf{e}_2$ are unit vectors defining the oblique coordinate system, is considered fitting the geometry of the triangle, as can be seen in Figure 2.3 (a). Hence, a local numeration

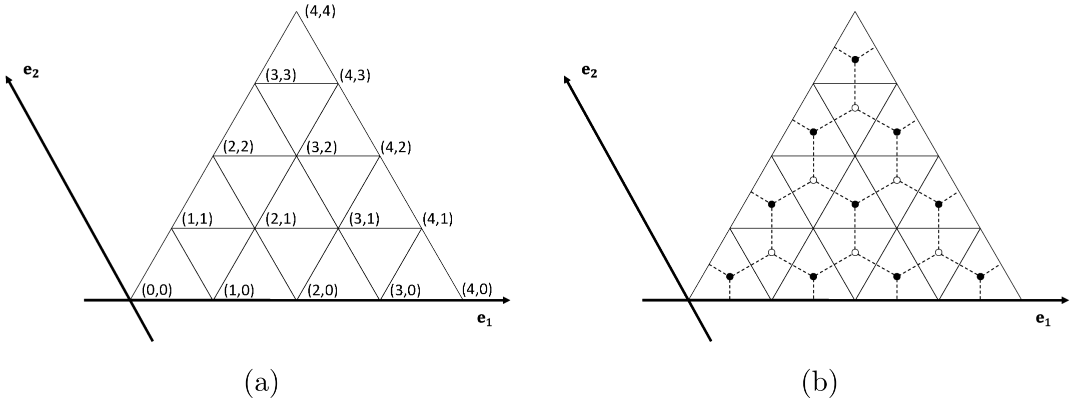


Figure 2.3: (a) New basis in $\mathbb{R}^2$ fitting the geometry of a uniform triangular grid, and local numeration for the regular Delaunay grid obtained on a triangular domain. (b) Corresponding Voronoi mesh.

can be fixed according to the definition of the spatial basis. In this way, a manner of numbering nodes very convenient for identifying the neighbouring nodes can be defined. We consider a triangular grid arising on a triangular domain by applying a fixed number of regular refinement steps ℓ. This is done in the way that, on each refinement step every triangle is divided into four congruent ones by connecting the midpoints of their edges. Then, we can define the corresponding grid in the following way:

$$G_\ell = \{\mathbf{x} = k_1 h_1 \,\mathbf{e}_1 + k_2 h_2 \,\mathbf{e}_2 \,|\, k_1 = 0, \ldots, 2^\ell,\ k_2 = 0, \ldots, k_1\}, \tag{2.6}$$

where $h = (h_1, h_2)$ is the grid spacing associated with the refinement level ℓ (h_1 is the grid spacing in the direction of $\mathbf{e}_1$, and h_2 in the direction of $\mathbf{e}_2$), so that the grid G_ℓ can also be denoted by G_h. Thus, for a refinement level ℓ, a local numeration with double index (k_1, k_2), $k_1 = 0, \ldots, 2^\ell$, $k_2 = 0, \ldots, k_1$, is used in such a way that the indexes of the vertices of the triangle are $(0, 0)$, $(2^\ell, 0)$, $(2^\ell, 2^\ell)$, as it can also be observed in Figure 2.3(a) for $\ell = 2$.

On the other hand, the considered discretization is based on the dual Voronoi mesh, represented in Figure 2.3(b). In the particular case in which a structured grid as considered here is used, the obtained finite difference scheme becomes different depending on the grid-point. More concretely, one-half of the grid-points, those corresponding to an up-oriented triangle, have the same equation and the other half, those corresponding to a down-oriented triangle, have a “mirror image stencil”, see Figure 2.4. In this

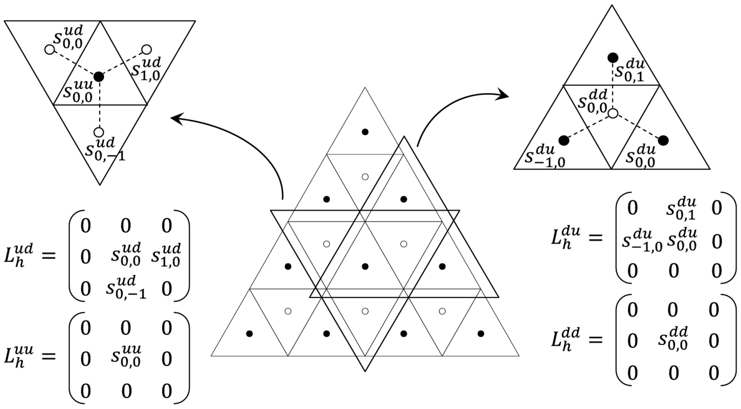

$s_{0,0}^{ud}$
$s_{1,0}^{ud}$
$s_{0,0}^{uu}$
$s_{0,-1}^{ud}$
$L_h^{ud} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s_{0,0}^{ud} & s_{1,0}^{ud} \\ 0 & s_{0,-1}^{ud} & 0 \end{pmatrix}$
$L_h^{uu} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s_{0,0}^{uu} & 0 \\ 0 & 0 & 0 \end{pmatrix}$
$s_{0,1}^{du}$
$s_{0,0}^{dd}$
$s_{-1,0}^{du}$
$s_{0,0}^{du}$
$L_h^{du} = \begin{pmatrix} 0 & s_{0,1}^{du} & 0 \\ s_{-1,0}^{du} & s_{0,0}^{du} & 0 \\ 0 & 0 & 0 \end{pmatrix}$
$L_h^{dd} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & s_{0,0}^{dd} & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Figure 2.4: Stencils corresponding to two different grid-points: one associated with an up-oriented triangle and the other with a down-oriented triangle.

sense, the Voronoi mesh, denoted by V_h could be split up into two sub-grids V_h^u (associated with the up-oriented triangles) and V_h^d, (corresponding to the down-oriented triangles), as seen in Figure 2.3(b). These sub-grids can be defined from the grid G_h, in the following way:

$$V_h^u = \{\mathbf{x}_{k_1,k_2}^u = (k_1+\delta_1)h_1\,\mathbf{e}_1 + (k_2+\delta_2)h_2\,\mathbf{e}_2 \mid k_1h_1\,\mathbf{e}_1 + k_2h_2\,\mathbf{e}_2 \in G_h\}, \quad (2.7)$$

$$V_h^d = \{\mathbf{x}_{k_1,k_2}^d = (k_1+\delta_1')h_1\,\mathbf{e}_1 + (k_2+\delta_2')h_2\,\mathbf{e}_2 \mid k_1h_1\,\mathbf{e}_1 + k_2h_2\,\mathbf{e}_2 \in G_h\}, \quad (2.8)$$

where δ_i, δ_i', with $i = 1, 2$, are suitable scalar values to reach Voronoi points from the primal ones following the considered local coordinate system, see Figure 2.5. Then, a grid-function, v_h, defined on the Voronoi mesh V_h, could be split up into two different sub-grid functions, v_h^u and v_h^d, associated with sub-grids V_h^u and V_h^d, respectively. In this way, given an arbitrary pair (k_1, k_2) associated with a node of G_h, the equations corresponding to the two Voronoi points $\mathbf{x}_{k_1,k_2}^u$ and $\mathbf{x}_{k_1,k_2}^d$, are given by

$$L_h^{uu} v_h^u(\mathbf{x}_{k_1,k_2}^u) + L_h^{ud} v_h^d(\mathbf{x}_{k_1,k_2}^d) = f_h^u(\mathbf{x}_{k_1,k_2}^u), \quad (2.9)$$

$$L_h^{du} v_h^u(\mathbf{x}_{k_1,k_2}^u) + L_h^{dd} v_h^d(\mathbf{x}_{k_1,k_2}^d) = f_h^d(\mathbf{x}_{k_1,k_2}^d), \quad (2.10)$$

where these “scalar” operators are given in stencil form as:

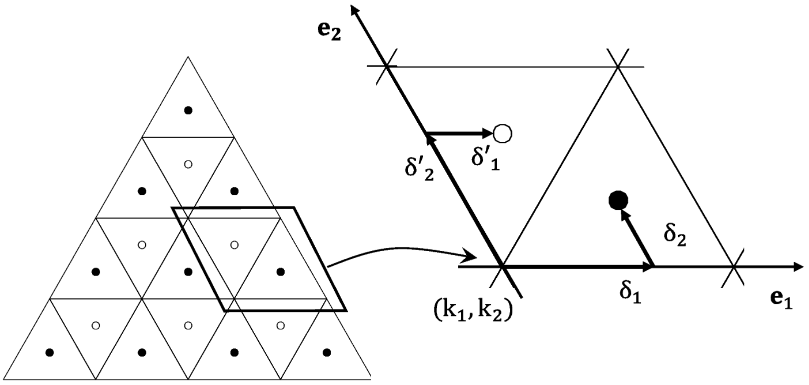


Figure 2.5: Voronoi mesh split into two sub-grids, and corresponding values of δ_i and δ'_i.

$$L_h^{uu} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sum_{i=1}^{3} \frac{l_i}{d_i} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L_h^{ud} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\frac{l_1}{d_1} & -\frac{l_3}{d_3} \\ 0 & -\frac{l_2}{d_2} & 0 \end{bmatrix},$$

$$L_h^{du} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & -\frac{l_2}{d_2} & 0 \\ -\frac{l_3}{d_3} & -\frac{l_1}{d_1} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L_h^{dd} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sum_{i=1}^{3} \frac{l_i}{d_i} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

2.1.2.1 Stencil depending on two angles characterizing the triangular grid.

An important feature to take into account in a stencil is the strength of the connections between the involved unknowns. Each entry of the stencil defines the coefficient of the corresponding unknown in the equation associated with the central point. Then, if some coefficient is large relative to the other coefficients in the same equation, then a small change in the value of the associated unknown has more effect on the value of the unknown at the central point, and therefore, we will say that these both unknowns are strongly connected, see (7).

In order to know a priori the strong and weak connections between neighbouring unknowns depending on the grid geometry, we are going to rewrite the stencils as a function of some parameters characterizing the grid, that is, two angles, α and β, and the length l, of one edge of an arbitrary triangle of the grid, see Figure 2.6. As we will see, this is going to be very useful for the design of smoothers for different geometries, taking into account the strong connections appearing in the stencils. Therefore, we are going to describe in detail the computation of the stencil for a Voronoi grid-point, associated with an arbitrary down-oriented triangle, $\mathbf{x}^d_{k_1,k_2}$ in V^d_h. Since all the parameters involved in the stencil (area, distances between Voronoi points and lengths of the edges) are independent of the chosen coordinate system, for simplicity, the coordinates of the points involved in such stencil can be computed in the Cartesian coordinate system with respect to the origin, see Figure 2.6. And, in terms of the previously explained geometric parameters, they result in the following:

$$\begin{aligned}
\mathbf{x}^d_{k_1,k_2} &= \frac{l}{2}\left(\frac{2\cos\alpha\sin\beta}{\sin(\alpha+\beta)}+1, \frac{2\sin\alpha\sin\beta}{\sin(\alpha+\beta)}+\cot(\alpha+\beta)\right), &(2.11)\\
\mathbf{x}^u_{k_1,k_2} &= \frac{l}{2}\left(3,-\cot(\alpha+\beta)\right), &(2.12)\\
\mathbf{x}^u_{k_1-1,k_2} &= \frac{l}{2}\left(1,-\cot(\alpha+\beta)\right), &(2.13)\\
\mathbf{x}^u_{k_1,k_2+1} &= \frac{l}{2}\left(\frac{2\cos\alpha\sin\beta}{\sin(\alpha+\beta)}+1, \frac{2\sin\alpha\sin\beta}{\sin(\alpha+\beta)}-\cot(\alpha+\beta)\right). &(2.14)
\end{aligned}$$

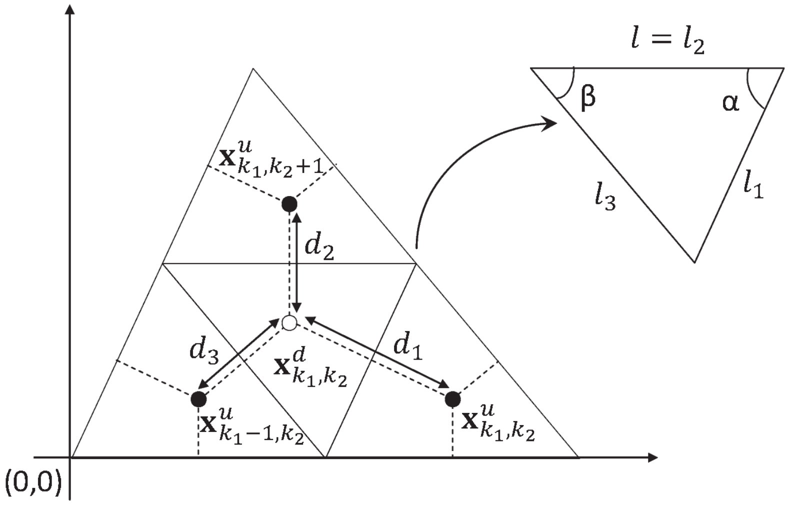


Figure 2.6: Notation for neighbouring Voronoi points on a structured grid, characterized by angles α and β.

Due to the fact that the area of an arbitrary triangle T is given in terms of the geometric parameters as

$$\text{meas}(T) = \frac{l^2 \sin\alpha \sin\beta}{2\sin(\alpha+\beta)}, \tag{2.15}$$

and the lengths of the sides of T are $l_2 = l$, $l_1 = \dfrac{l\sin\beta}{\sin(\alpha+\beta)}$, and $l_3 = \dfrac{l\sin\alpha}{\sin(\alpha+\beta)}$, after computing d_i $i = 1,2,3$ from (2.11)-(2.14), we can finally obtain the stencils:

$$L_h^{du} = \frac{2\sin(\alpha+\beta)}{l^2\sin\alpha\sin\beta}\begin{bmatrix} 0 & \tan(\alpha+\beta) & 0 \\ -\tan\alpha & -\tan\beta & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.16}$$

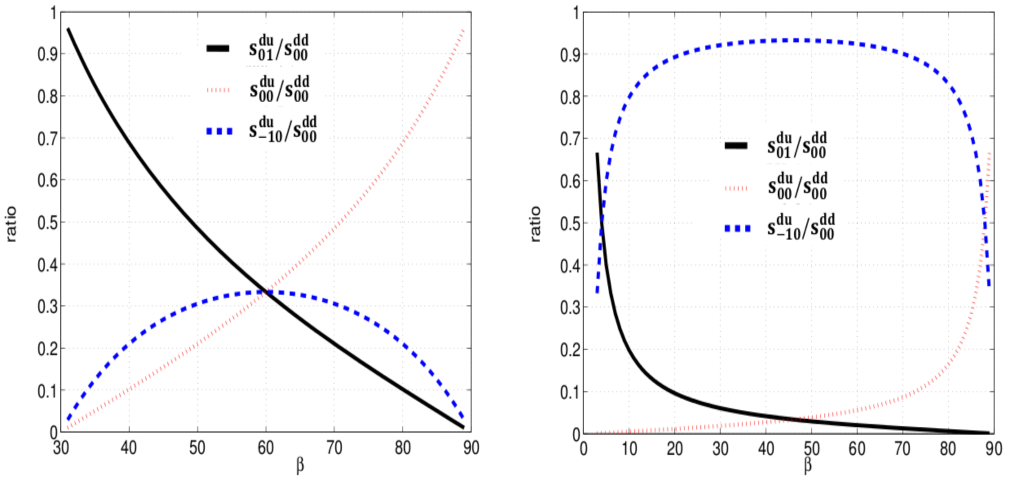
$$L_h^{dd} = \frac{2\sin(\alpha+\beta)}{l^2\sin\alpha\sin\beta}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -\tan(\alpha+\beta)+\tan\alpha+\tan\beta & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{2.17}$$

As previously commented, for a Voronoi grid-point, associated with an arbitrary up-oriented triangle, $\mathbf{x}^u_{k_1,k_2}$ in V_h^u the stencil would be the "mirror image stencil" of (2.16)-(2.17), that is,

$$L_h^{uu} = \frac{2\sin(\alpha+\beta)}{l^2\sin\alpha\sin\beta}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -\tan(\alpha+\beta)+\tan\alpha+\tan\beta & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{2.18}$$

$$L_h^{ud} = \frac{2\sin(\alpha+\beta)}{l^2\sin\alpha\sin\beta}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -\tan\beta & -\tan\alpha \\ 0 & \tan(\alpha+\beta) & 0 \end{bmatrix}. \tag{2.19}$$

Notice that depending on the angles characterizing the grid, some strong connections appear between some unknowns. In particular, in Figure 2.7 we show the variability of the stencil weights, for a Voronoi point $\mathbf{x}^d_{k_1,k_2}$ associated with a down-oriented triangle, depending on one of the angles of the triangulation. More concretely, we display the ratios between the three extra-diagonal coefficients in the equation and the central term, that is, s^{du}_{00}/s^{dd}_{00}, s^{du}_{01}/s^{dd}_{00}, and s^{du}_{-10}/s^{dd}_{00}, (see notation in Figure 2.4) for two different situations in which angle α is fixed and angle β varies among all its possible values satisfying the requirement of an acute triangulation. In Figure 2.7(a), α is 60^{O}, and we can see that when $\beta = 60^{\text{O}}$, that is for an equilateral triangle, the strength of the three connections in the stencil is the same. When angle β is small or large enough, we observe that two of the connections are weak whereas the other one is very strong. On the other hand, in Figure 2.7(b), α is fixed as 88^{O}. We can see that for almost the whole range of values of β one of the connections in the stencil is strong and the other



(a)

(b)

Figure 2.7: Variability of the ratios between the coefficients of a stencil associated with triangles with (a) $\alpha = 60^{\mathrm{o}}$ or (b) $\alpha = 88^{\mathrm{o}}$, and β varying among all its possible values satisfying the requirement of an acute triangulation.

ones are much weaker, whereas when β is either very small or very large, then two of the connections in the stencil are of equal strength. These strong connections between the unknowns involved in a stencil will have to be taken into account in the design of the smoothers in a geometric multigrid method, as will be discussed in next section.

2.2 Multigrid method

In this section we are going to present the multigrid components that we are going to use. For a basic explanation of multigrid methods, we refer the reader to Appendix A.

The performance of geometric multigrid methods is strongly dependent on the choice of adequate components to the considered problem. The main components are the smoother $\mathbf{S}_h$, inter-grid transfer operators: restriction $\mathbf{I}_h^{2h}$ and prolongation $\mathbf{I}_{2h}^h$, and the coarse-grid operator $\mathbf{L}_{2h}$. These components have to be chosen so that they efficiently interplay with each other in order to obtain a good connection between the relaxation and the coarse-grid correction. In the following subsections the proposed cell-centred multigrid algorithm is described. All the attention is focused in the detailed explanation of the considered smoothers and the special features appearing due to the cell-centred character of the discretization.

Although the presentation of such components is done on a regular structured grid, our purpose is to apply the proposed multigrid method in the framework of semi-structured grids. Therefore, the choice of the corresponding components is done also with a view to this application. In this case, we will use a block-wise multigrid algorithm, where each triangle of the coarsest grid is treated as a different block with regard to the smoothing process. This block-wise strategy is suitable thanks to the possibility of choosing different smoothers for triangles having different geometries, thus resulting in an improvement of the characteristics of our algorithm. Besides, we will have to take care in the communication among the triangles of the coarsest triangulation. Next, we are going to describe the components of the algorithm that we are going to consider throughout all this chapter.

2.2.1 Coarse-grid correction

In the application of geometric multigrid, a hierarchy of grids is needed in order to accelerate the convergence of the smoother, by using solutions obtained on the coarser meshes as corrections. In order to obtain such hierarchy of grids, we divide the initial triangles into four congruent ones by connecting the midpoints of the edges, and so forth until the mesh has the fine scale to obtain the desired approximation of the solution. Once the hierarchy of grids is defined, we have to choose suitable inter-grid transfer operators as well as the corresponding discrete operators on the coarse grids. Regarding the coarse-grid approximation, a direct PDE discretization has been used.

When vertex-centred discretizations are considered on triangular grids, grid-points lying on coarser grids also belong to the finer grids, giving rise to a so-called nested hierarchy of grids. However, when the considered cell-centred discretizations are used, it is worth to note that except in the case of equilateral triangles, the grid hierarchy results to be non-nested, see Figure 2.8. This makes the interplay between smoothing and coarse-grid correction specially difficult, requiring the design of new smoothers or new inter-grid transfer operators. With a view to the application of the proposed multigrid method on semi-structured grids, our proposal is to consider very simple inter-grid operators, since this will facilitate the communication between the blocks composing these grids. In particular, injection operator is considered as the prolongation, and its adjoint is chosen as the restriction, resulting in the fact that only the four fine-grid points surrounding a coarse-grid point contribute to its restriction.

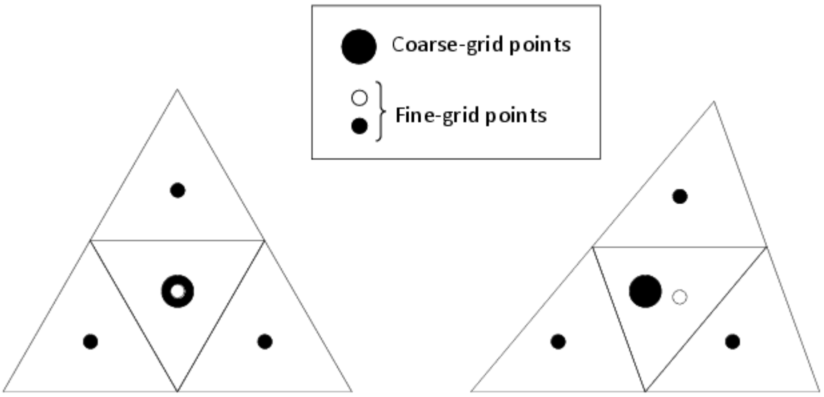


Figure 2.8: Nested (left) and non-nested (right) hierarchies.

More concretely, the considered restriction operator, $\mathbf{I}_h^{2h}$, is given in the following way

$$\mathbf{I}_h^{2h} = \begin{bmatrix} (I_h^{2h})^{uu} & (I_h^{2h})^{ud} \\ (I_h^{2h})^{du} & (I_h^{2h})^{dd} \end{bmatrix}, \text{ with } \left\{ \begin{array}{ll} (I_h^{2h})^{uu} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & (I_h^{2h})^{ud} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1/4 & 1/4 \\ 0 & 1/4 & 0 \end{bmatrix}, \\ (I_h^{2h})^{du} = \begin{bmatrix} 0 & 1/4 & 0 \\ 1/4 & 1/4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, & (I_h^{2h})^{dd} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1/4 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \end{array} \right. \tag{2.20}$$

as shown in Figure 2.9, and the corresponding prolongation fulfils $\mathbf{I}_{2h}^h = 4\,\mathbf{I}_h^{2h}$.

The choice of these inter-grid transfer operators leads us to make an effort in the smoothing process. Then, we must design efficient smoothers capable of taking charge of the remaining components of the error, which cannot be eliminated by the coarse-grid correction part of the algorithm. Actually, the design of suitable smoothers for cell-centred grids is a challenge in this context.

2.2.2 Smoothers

The smoother usually plays an important role in multigrid algorithms, mainly in the geometric approach. The choice of a suitable smoother is an important feature to

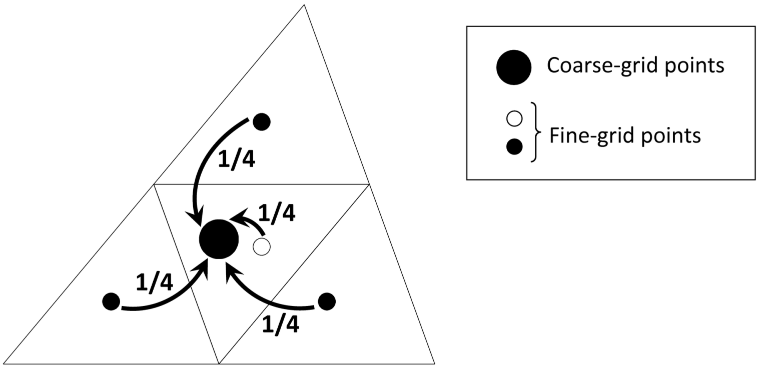


Figure 2.9: Restriction operator.

guarantee the efficiency of these methods. Moreover, as previously commented, in the framework we are working with, this choice is even more relevant. Due to the general observation that errors become smooth if strongly connected unknowns are collectively updated, see (5, 49), appropriate smoothers have been designed depending on the magnitude of the coefficients of the stencils, given in (2.16)-(2.19). The following smoothers have been considered and tested in order to fulfil the previous requirement.

Jacobi smoother: For almost-equilateral triangles, the magnitude of all the entries of the stencils is similar, and therefore a point-wise smoother is enough to satisfactorily reduce the high-frequency components of the error. The easiest smoother to perform is a Jacobi type smoother, which consists of computing the approximation of each unknown, by using non-updated values of the rest of the unknowns. Notice that this implies that the order in which the grid-points are visited makes no difference, what makes Jacobi scheme well-suited for parallel processing. However, for difficult problems, usually this smoother does not give enough satisfactory results, and some variants have to be considered. As is the case for Jacobi smoother presented here, some standard smoothers are based on a decomposition on the positive and negative parts of the operator, which correspond to the updated and non-updated unknowns before the current step, see (49). That is, any discrete operator L_h can be split in the following way $L_h = L_h^+ + L_h^-$, where the choices of the positive and negative parts give rise to different classical iterative methods. Here we are going to present the corresponding decomposition for Jacobi smoother. In order to do this, only the positive part of the operator is displayed. Taking into account that only the diagonal blocks of the operator contribute in this positive part, it holds

$$(L_h^{uu})^+ = L_h^{uu}, \quad (L_h^{dd})^+ = L_h^{dd}. \tag{2.21}$$

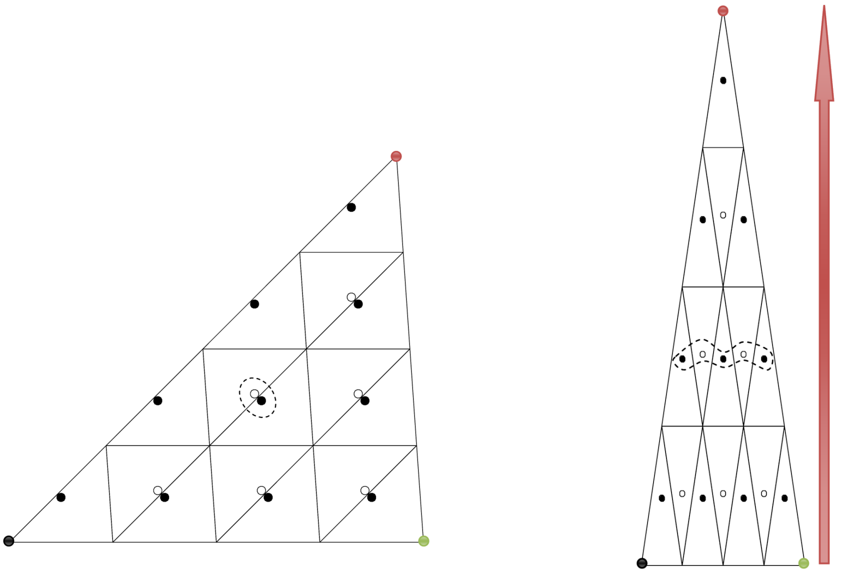
Red-Black smoother: Due to the fact that unknowns related to up or down-oriented triangles have no direct connection with each other, it seems natural to simultaneously update all unknowns associated with equally oriented triangles, giving rise to a pattern relaxation scheme. Since two different types of grid-points are distinguished, a two-color relaxation process, called here red-black smoother, is considered. More concretely, one iteration of this relaxation scheme consists of two partial steps. In the first one, unknowns corresponding to up-oriented triangles are updated, and in the second step those associated with the down-oriented triangles are relaxed by using the updated values. Thus, the complete smoothing operator $\mathbf{S}_h$ is given by the composition of two partial step operators, $\mathbf{S}_h^u$ and $\mathbf{S}_h^d$, which correspond to apply a Jacobi step on each type of grid-points, that is, $\mathbf{S}_h = \mathbf{S}_h^d\,\mathbf{S}_h^u$. These partial step operators are characterized by a different decomposition than the previous Jacobi over all the grid-points, in the way that for $\mathbf{S}_h^u$, for example, the positive parts of the scalar operators are:

$$(L_h^{uu})^+ = L_h^{uu}, \quad (L_h^{dd})^+ = I_h, \tag{2.22}$$

and if $\mathbf{S}_h^d$ is considered, the identity operator will correspond to $(L_h^{uu})^+$, and $(L_h^{dd})^+ = L_h^{dd}$.

Diamond smoothers: For almost-right triangles, a strong connection between only two nodes involved in the stencil appears, due to the anisotropy of the Voronoi mesh. Therefore, since the common lore claims that smoothing must be done in the direction of the strong connection, in this case, both unknowns will have to be simultaneously relaxed. These unknowns are associated with the closest Voronoi points corresponding to different-oriented triangles, as seen in Figure 2.10(a). Therefore, a small (2×2)-system must be solved for each of these pairs of unknowns. Different orderings can be chosen to visit these blocks. We have chosen the lexicographic one, but of course, many orderings are possible.

In triangular grids, three different diamond smoothers, associated with the three edges of a triangle, can be defined. If a triangle characterized by angles α and β is considered, we can assign a different color to each of its vertex, in the way that, for example, black color is associated with the vertex corresponding to angle α, green color with that vertex associated with β, and red color corresponds to vertex of angle $180^{\text{O}} - (\alpha + \beta)$. In this manner, each diamond smoother can be named with the color corresponding to the vertex opposite to the edge across which the strong coupling appears. Following this rule, diamond smoother appearing in Figure 2.10(a) is called green-diamond smoother.



(a)

(b)

Figure 2.10: (a) Green-diamond smoother and (b) Red-wormy smoother.

This smoother is based on a decomposition of positive and negative parts of the operator. Although three different diamond smoothers can be considered, the corresponding decomposition can be done analogously. Then, in order to do this description, we consider the green-diamond smoother, which can be seen in Figure 2.10(a). The positive parts of the involved scalar operators are given as follows:

$$(L_h^{uu})^+ = L_h^{uu},\ (L_h^{ud})^+ = \begin{bmatrix} 0 & 0 & 0 \\ 0 & s_{0,0}^{ud} & 0 \\ 0 & s_{0,-1}^{ud} & 0 \end{bmatrix},\ (L_h^{du})^+ = \begin{bmatrix} 0 & 0 & 0 \\ s_{-1,0}^{du} & s_{0,0}^{du} & 0 \\ 0 & 0 & 0 \end{bmatrix},\ (L_h^{dd})^+ = L_h^{dd}. \tag{2.23}$$

Wormy smoothers: When a very small angle characterizes the triangulation, the strong coupling appears between the Voronoi points associated with the triangles situated in the direction of the edge opposite to the vertex corresponding to this small angle, see Figure 2.10(b). Therefore, all those points have to be simultaneously updated and a tridiagonal system must be solved for each of these "wormy-lines". For this reason, this smoother will be called wormy smoother.

Similarly to the previous case of the diamond smoother, in triangular grids, three different wormy-smoothers can be defined, associated with the three edges of a triangle, and they can be named with the color corresponding to the vertex opposite to such edge. Following this criterion, wormy smoother appearing in Figure 2.10(b) is called red-wormy smoother. Notice that these smoothers can be performed by visiting the "lines" in a lexicographic order, from vertex to edge or, on the contrary, from edge to vertex, and this latest is the chosen option, as it is represented by the arrow in Figure 2.10(b). From the description of this smoother, we can obtain the decomposition of the discrete operator which gives rise to wormy smoother. Analogously to the diamond smoother, three different wormy smoothers can be defined, whose decomposition can be obtained in a similar way. In order to present the corresponding decomposition of the operator, the red-wormy smoother, displayed in Figure 2.10(b), is considered. In this way, the positive parts of the scalar operators result as follows:

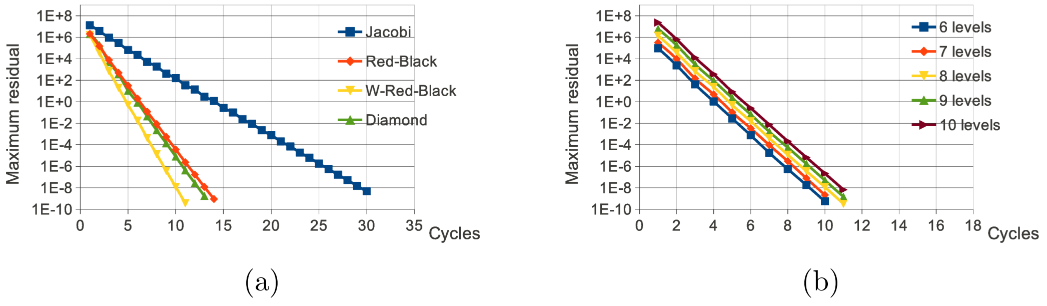
$$(L_h^{uu})^+ = L_h^{uu},\ (L_h^{ud})^+ = \begin{bmatrix} 0 & 0 & 0 \\ 0 & s_{0,0}^{ud} & s_{1,0}^{ud} \\ 0 & s_{0,-1}^{ud} & 0 \end{bmatrix},\ (L_h^{du})^+ = \begin{bmatrix} 0 & 0 & 0 \\ s_{-1,0}^{du} & s_{0,0}^{du} & 0 \\ 0 & 0 & 0 \end{bmatrix},\ (L_h^{dd})^+ = L_h^{dd}. \tag{2.24}$$

Concluding, we can say that each of these wormy-smoothers will be suitable when the angle corresponding to the vertex of its color is small, and as we will see in next section, any possible triangulation will have associated a wormy smoother giving a satisfactory convergence factor.

2.2.3 Results of the proposed multigrid method on structured grids

In order to see the suitability of the introduced smoothers in the design of an efficient multigrid algorithm, we present some experiments comparing their behaviour in regular structured grids. In particular, model problem (2.1)-(2.2) is solved in the structured grid arising from the regular refinement of a triangular domain, characterized by two of their angles. Unless stated otherwise, homogeneous Dirichlet boundary conditions, a random initial guess, and a zero right-hand side are considered in the numerical experiments to avoid round-off errors. Initially, an F-cycle is considered since it is cheaper than a W-cycle, and more robust than a simple V-cycle (49). However, in the section of numerical experiments its behaviour will be compared to that of a V-cycle. Regarding the number of smoothing steps, we have fixed two pre- and two post-smoothing steps.

We begin by considering an equilateral triangular domain. Since any anisotropy arises from the grid resulting of a regular refinement, it could be seen natural to think in applying a simple point-wise smoother, like Jacobi or lexicographic Gauss-Seidel, to this kind of triangulations. However, as previously commented, for this situation a pattern relaxation scheme could be more appropriate. Thus, we are going to present some convergence results on a regular equilateral grid, comparing the behaviour of multigrid by considering: undamped Jacobi smoother, red-black smoother, $\omega-$red-black smoother (with $\omega = 1.15$) and diamond smoother. In Figure 2.11(a), we show the history of the convergence on a grid obtained after eight refinement levels, and by considering as stopping criterion to reduce the maximum residual until 10^{-8}. First of all, a rather surprising observation could be concluded from this figure: undamped Jacobi appears to be a satisfactory choice as smoother for cell-centred discretizations on triangular grids (as also seen for other type of discretizations on triangular grids (20), and in the context of full-multigrid on rectangular grids (44)), despite the well-known lack of smoothing property of this iterative scheme. Notwithstanding this unusual behaviour, the obtained Jacobi results are largely improved by red-black smoother and diamond



(a)

(b)

Figure 2.11: (a) Comparison of smoothers on an equilateral triangular grid. (b) History of the convergence for different numbers of refinement levels by using $\omega-$red-black smoother.

smoother. At the same time, the convergence factors provided by both smoothers are enhanced by the red-black smoother with relaxation parameter $\omega = 1.15$, which has been obtained by experimental tests. This improving effect by using a relaxation parameter was pointed out in (43) for cell-centred discretizations on rectangular grids, where it was also validated by a local Fourier analysis. Moreover, the good behaviour of the multigrid based on the $\omega-$red-black smoother is confirmed in Figure 2.11(b), where its robustness with regard to the discretization parameter is shown. In this figure, the history of the convergence of the method is displayed for different numbers of refinement levels, resulting to be independent. Therefore, we conclude from this experiment that $\omega-$red-black smoother seems to be a good choice for almost-equilateral triangulations. However, this good behaviour deteriorates quickly when the shape of the triangle tends to be rectangular or is characterized by a very small angle.

In the case of almost-right triangles, point-wise smoothers are not suitable anymore due to the anisotropy of the Voronoi mesh. On the other hand, diamond smoother results in a very efficient smoother when this kind of grids are considered. As an example, a triangular domain characterized by angles $\alpha = 45^{\text{O}}$ and $\beta = 85^{\text{O}}$ is fixed. In Figure 2.12(a), the history of the multigrid convergence by using different smoothers is displayed. More concretely, red-black, $\omega-$red-black and diamond smoothers are used in this comparison. In all cases, the finest grid results by applying eight refinement levels to the initial triangular domain. As we can observe, very bad rates are obtained when

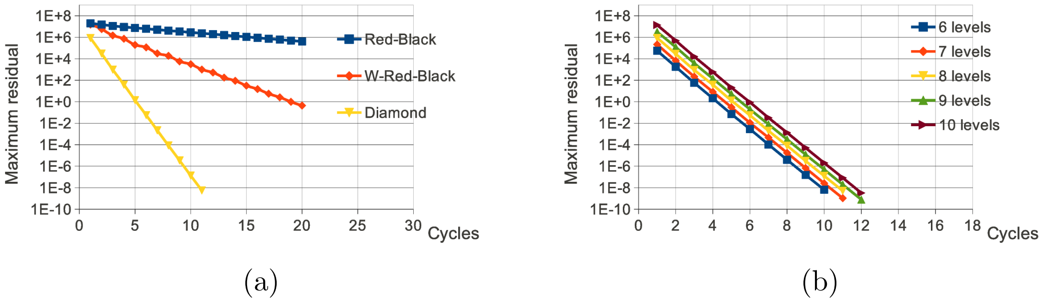


Figure 2.12: (a) Comparison of smoothers on an almost-right triangular grid. (b) History of the convergence for different numbers of refinement levels by using diamond smoother.

both red-black smoothers are considered, whereas the convergence factors provided by the new diamond smoother are very satisfactory, achieving the convergence in only eleven cycles. Besides, in Figure 2.12(b), where the history of the convergence is shown for different numbers of refinement levels, the robustness of this smoother with respect to the space discretization parameter is demonstrated. Although convergence factors provided by diamond smoother are very satisfactory for many grid configurations, when a triangulation characterized by a very small angle is used, this smoother gives rise to poor rates. This behaviour can be seen in Figure 2.13, where asymptotic convergence factors of the diamond smoother based multigrid are shown for a wide range of pairs of angles (α, β) characterizing the grid.

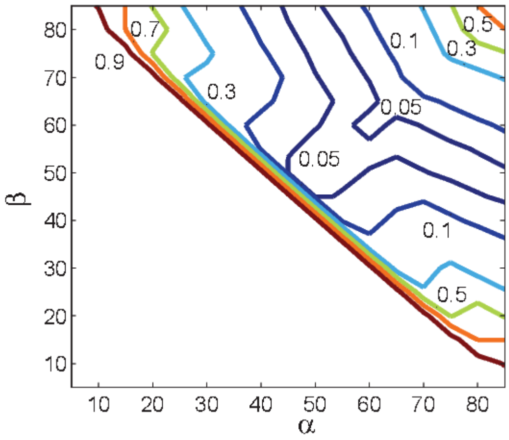
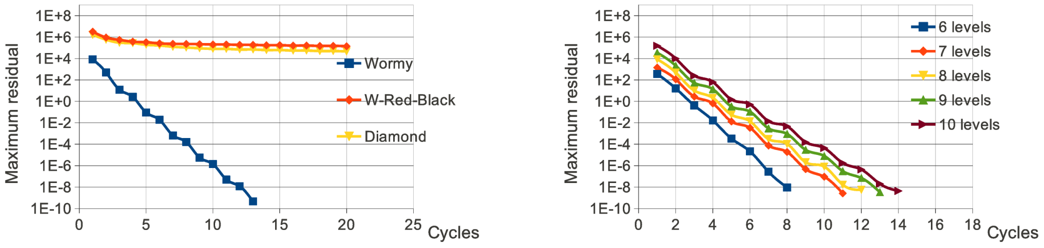


Figure 2.13: Experimentally computed convergence factors for the diamond smoother based multigrid and four smoothing steps, for different triangles in function of two of their angles.

To overcome these troubles appearing when the primal mesh is anisotropic, wormy smoother in the direction of the anisotropy is a suitable smoother, largely improving the convergence factors provided by the rest of point-wise or block-wise smoothers. To validate this statement, we are going to compare the multigrid convergence by using each one of the smoothers proposed in this work, when an isosceles triangle with a small angle of $10^{\rm O}$ is considered as domain of our problem. With this purpose, in Figure 2.14(a), the multigrid convergence provided by using $\omega-$red-black, diamond and wormy smoothers is depicted for eight refinement levels. From this picture, it is clear that wormy smoother is the best choice for this type of triangulations. Moreover, an $h-$independent convergence is also shown in Figure 2.14(b).

		Equilateral ($\omega-$red-black)		Right-Triangle (Diamond smoother)		Sharp triangle (Wormy smoother)	
Levels	Unknowns	Cycles	Time (s)	Cycles	Time (s)	Cycles	Time (s)
6	4096	7	0.03	8	0.03	6	0.04
7	16384	7	0.09	9	0.10	6	0.13
8	65536	7	0.35	9	0.39	7	0.58
9	262144	7	1.35	9	1.55	7	2.28
10	1048576	7	5.76	9	6.81	7	9.66

Table 2.1: Number of cycles necessary to reduce the initial residual in a factor of 10^{-10}, by using an F-cycle and the corresponding CPU-times.



(a)

(b)

Figure 2.14: (a) Comparison of smoothers on an isosceles triangular grid with smallest angle 10^{O}. (b) History of the convergence for different numbers of refinement levels by using wormy smoother.

From the results presented in this section, it seems that a reasonable strategy to follow would be to apply the point-wise $\omega-$red-black smoother for almost-equilateral triangles, the collective diamond smoother for almost-right triangles, and finally the appropriate block collective wormy smoother when triangulations with a small angle appear. To summarize, in Table 2.1 we show for the three considered representative triangles the results corresponding to the best smoother for each geometry. In particular, for different numbers of refinement levels, the number of cycles necessary to reduce the initial residual in a factor of 10^{-10}, and the CPU-time are shown. As we can observe, a convergence independent on the number of unknowns is obtained in all the three cases.

From the practical point of view, for any given triangular geometry it would be nice to be capable of choosing a suitable smoother in order to reach a desired convergence factor. Moreover, for semi-structured grids, it is imperative to know the smoother to use for each triangle of the input grid to achieve a local desired convergence factor (45). In order to reach this, a set-up phase has been implemented in the multigrid code; it consists of reading an already calculated database containing the most efficient strategy depending on the angles of the triangle. That is, taking into account that wormy smoother is about twice expensive than $\omega-$red-black smoother, and diamond smoother is about ten per cent more expensive than this latter, we choose for each triangle the cheapest smoother which gives a convergence factor below a desired one. Following this strategy, the corresponding guideline to reach a global convergence factor about 0.1 is shown in Figure 2.15.

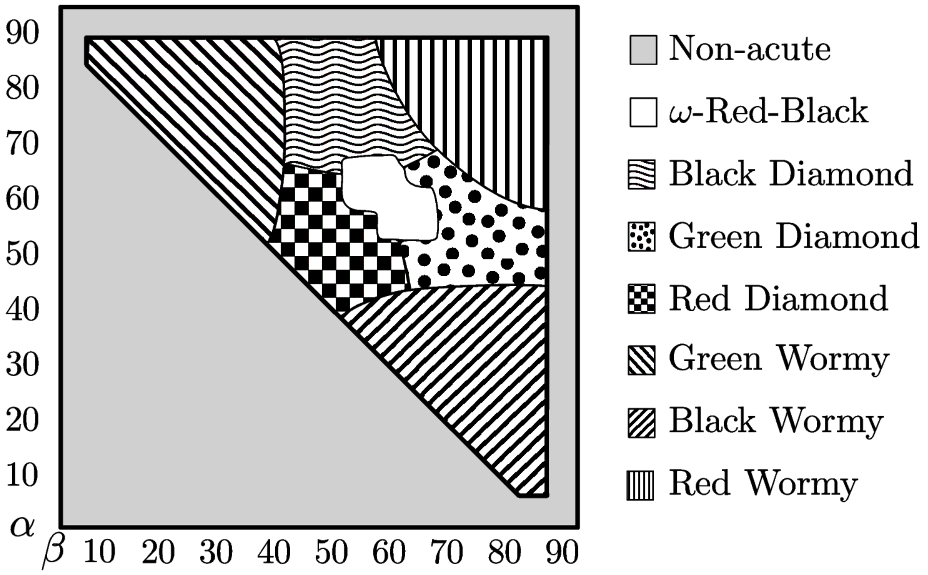


Figure 2.15: Guideline to choose suitable smoothers to reach an asymptotic convergence factor about 0.1 on different triangles.

2.3 Local Fourier analysis

As we have seen in previous sections, the design of an efficient multigrid method may not be easy. The necessity of a good smoothness, and its interplay with the coarse-grid correction is essential to achieve a satisfactory convergence rate. Up to this point, we have been working with numerical experiments, which means that we had to code the whole multigrid method in order to know whether it works or not. Taking into account the amount of work that this requires, it seems logical to try to know a priori if what we are going to perform is a good strategy. For that reason, the Local Fourier analysis, LFA, was initially developed by Brandt in (4) and afterwards extended by him in (6). It was focused on rectangular grids and recently, it was extended to triangular meshes in (20). This tool calculates the asymptotic convergence rate of a multigrid algorithm. To do that, it requires some assumptions: it neglects the effect of boundary conditions, by considering the discrete operator to be defined on an infinite grid, and also if the equation is non-linear or it has non-constant coefficients it must be locally linearized (49). Despite these limitations, this method is a very useful tool to design efficient multigrid methods for new problems. Also, it can be used to test if a multigrid code has any bugs, since LFA predicts the asymptotic convergence rate that the code should provide, see Figure 2.16. In Appendix A, we have included a basic introduction to the smoothing analysis.

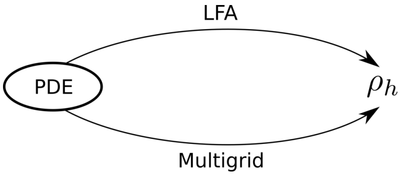


Figure 2.16: The Local Fourier Analysis and the multigrid code should provide the same asymptotic convergence rate.

Local Fourier analysis cannot be straightforwardly applied to discretizations on regular Voronoi meshes, since two different stencils appear for different grid-points, depending on the orientation of the corresponding triangles. This latter, together with the fact that triangular meshes are considered, make this analysis different from the usual one. Regarding the extension to triangular meshes, the key is to write the Fourier transform using coordinates in non-orthogonal bases fitting the structure of the grid, as in Section 2.1.2. For local Fourier analysis, a regular infinite grid is assumed. With this purpose, we extend the definition of grid G_h in (2.6) to the corresponding infinite grid in the following way:

$$\mathcal{G}_h = \{\mathbf{x}_{k_1,k_2} = k_1 h_1 \, \mathbf{e}_1 + k_2 h_2 \, \mathbf{e}_2 \,|\, k_1,\, k_2 \in \mathbb{Z}\}, \tag{2.25}$$

as seen in Figure 2.17. In this way the associated Voronoi mesh is also extended to an infinite grid $\mathcal{V}_h$, which can be decomposed again into two infinite sub-grids as $\mathcal{V}_h = \mathcal{V}_h^u \bigcup \mathcal{V}_h^d$. Now, we must extend the discrete problem to the whole infinite grid $\mathcal{V}_h$.

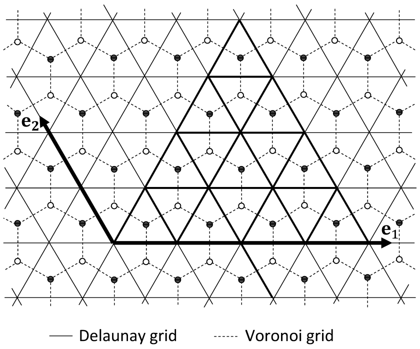


Figure 2.17: Infinite triangular Delaunay grid with the associated infinite Voronoi mesh and the basis fitting the grid

However, for this purpose, we have to take into account that equations at points on $\mathcal{V}_h^u$ and $\mathcal{V}_h^d$ are different. Thus, we can define the application of the discrete operator to a grid-function v_h on $\mathcal{V}_h$ as:

$$L_h \, v_h(\mathbf{x}) = \begin{cases} \displaystyle\sum_{(i,j)\in I^{uu}} s_{i,j}^{uu} v_h(\mathbf{x}_{k_1+i,k_2+j}^u) + \sum_{(i,j)\in I^{ud}} s_{i,j}^{ud} v_h(\mathbf{x}_{k_1+i,k_2+j}^d), \; \mathbf{x} = \mathbf{x}_{k_1,k_2}^u \in \mathcal{V}_h^u, \\ \displaystyle\sum_{(i,j)\in I^{du}} s_{i,j}^{du} v_h(\mathbf{x}_{k_1+i,k_2+j}^u) + \sum_{(i,j)\in I^{dd}} s_{i,j}^{dd} v_h(\mathbf{x}_{k_1+i,k_2+j}^d), \; \mathbf{x} = \mathbf{x}_{k_1,k_2}^d \in \mathcal{V}_h^d, \end{cases} \tag{2.26}$$

where the coefficients in (2.26) are

$$\begin{aligned} s_{0,1}^{du} &= \frac{2\sin(\alpha+\beta)}{l^2 \sin\alpha \sin\beta} \tan(\alpha+\beta), \quad & s_{-1,0}^{du} &= -\frac{2\sin(\alpha+\beta)}{l^2 \sin\alpha \sin\beta} \tan\alpha, \\ s_{0,0}^{du} &= -\frac{2\sin(\alpha+\beta)}{l^2 \sin\alpha \sin\beta} \tan\beta, & s_{0,0}^{dd} &= -(s_{0,0}^{du} + s_{1,0}^{du} + s_{0,-1}^{du}). \end{aligned} \tag{2.27}$$

Due to the geometric relations between an up- and a down-oriented triangle, it is easy to see that the coefficients satisfy $s_{0,-1}^{ud} = s_{0,1}^{du}$, $s_{1,0}^{ud} = s_{-1,0}^{du}$, $s_{0,0}^{ud} = s_{0,0}^{du}$, $s_{0,0}^{uu} = s_{0,0}^{dd}$.

and the subsets I^{uu}, I^{ud}, I^{du}, and I^{dd} give the connections of Voronoi points associated with up-oriented triangles with themselves and with those associated with down-oriented triangles, and vice versa. In particular, in our case we have $I^{uu} = \{(0,0)\}$, $I^{ud} = \{(0,0),(1,0),(0,-1)\}$, $I^{du} = \{(0,0),(-1,0),(0,1)\}$, and $I^{dd} = \{(0,0)\}$.
A local Fourier analysis is based on the Fourier modes, which have the following form: $\varphi_h(\boldsymbol{\theta},\mathbf{x}) = e^{\imath\boldsymbol{\theta}\cdot\mathbf{x}/\mathbf{h}} = e^{\imath(\theta_1 x_1/h_1+\theta_2 x_2/h_2)}$, with $\boldsymbol{\theta} = (\theta_1,\theta_2) \in \boldsymbol{\Theta}_h = [-\pi,\pi)^2$, and where $\mathbf{x}$ is a grid-point. For node-based problems, these Fourier modes turn out to be eigenvectors of grid operators that can be represented by a single stencil, see (4, 57). However, for the analysis that we want to develop, $\varphi_h(\boldsymbol{\theta},\mathbf{x})$ is not directly useful, and we have to define the corresponding "Fourier modes", taking into account that we consider a cell-centered discretization on $\mathcal{V}_h$, and the fact that the discrete operator cannot be represented by a single stencil. Then, we define the following grid-functions:

$$\phi_h(\boldsymbol{\theta},\mathbf{x}) = \alpha^u\phi_h^u(\boldsymbol{\theta},\mathbf{x}) + \alpha^d\phi_h^d(\boldsymbol{\theta},\mathbf{x}), \quad \mathbf{x}\in\mathcal{V}_h, \quad \alpha^u,\alpha^d\in\mathbb{C}, \tag{2.28}$$

where

$$\phi_h^u(\boldsymbol{\theta},\mathbf{x}) = \left\{\begin{array}{ll} \varphi_h(\boldsymbol{\theta},\mathbf{x}), & \mathbf{x}\in\mathcal{V}_h^u \\ 0, & \mathbf{x}\in\mathcal{V}_h^d \end{array}\right., \quad \text{and} \quad \phi_h^d(\boldsymbol{\theta},\mathbf{x}) = \left\{\begin{array}{ll} 0, & \mathbf{x}\in\mathcal{V}_h^u \\ \varphi_h(\boldsymbol{\theta},\mathbf{x}), & \mathbf{x}\in\mathcal{V}_h^d \end{array}\right. . \tag{2.29}$$

This splitting in (2.28) is a direct consequence of the fact that the stencils for the points on $\mathcal{V}_h^u$ and $\mathcal{V}_h^d$ are different. In this way, these grid-functions will play the same role in the LFA for the discretizations considered here as the Fourier modes $\varphi_h(\boldsymbol{\theta},\mathbf{x})$ in the LFA of standard node-based discretizations. Then, we define the following space of grid-functions:

$$\mathcal{F}(\mathcal{V}_h) = \{\phi_h(\boldsymbol{\theta},\cdot) = \alpha^u\phi_h^u(\boldsymbol{\theta},\cdot) + \alpha^d\phi_h^d(\boldsymbol{\theta},\cdot),\ \alpha^u,\alpha^d\in\mathbb{C},\ \boldsymbol{\theta}\in\boldsymbol{\Theta}_h\}, \tag{2.30}$$

which will play the role of our "Fourier space". In this way, we can prove that any discrete operator as in (2.26) leaves invariant subspace $\mathcal{F}(\mathcal{V}_h)$. In fact, if we apply operator L_h to a grid-function $\phi_h\in\mathcal{F}(\mathcal{V}_h)$, depending on the grid-point and taking into account definitions in (2.7)-(2.8) the following holds:

$$
\begin{aligned}
&(L_h\phi_h(\boldsymbol{\theta},\cdot))(\mathbf{x}^u_{k_1,k_2}) = \alpha^u \sum_{(i,j)\in I^{uu}} s^{uu}_{i,j}\phi^u_h(\mathbf{x}^u_{k_1+i,k_2+j}) + \alpha^d \sum_{(i,j)\in I^{ud}} s^{ud}_{i,j}\phi^d_h(\mathbf{x}^d_{k_1+i,k_2+j}) \\
&= e^{\imath\boldsymbol{\theta}\cdot\mathbf{x}^u_{k_1,k_2}/\mathbf{h}} \left(\alpha^u \sum_{(i,j)\in I^{uu}} s^{uu}_{i,j} e^{\imath(\theta_1 i+\theta_2 j)} + \alpha^d \sum_{(i,j)\in I^{ud}} s^{ud}_{i,j} e^{\imath(\theta_1(i+\delta_1'-\delta_1)+\theta_2(j+\delta_2'-\delta_2))} \right), \\
&(L_h\phi_h(\boldsymbol{\theta},\cdot))(\mathbf{x}^d_{k_1,k_2}) = \alpha^u \sum_{(i,j)\in I^{du}} s^{du}_{i,j}\phi^u_h(\mathbf{x}^u_{k_1+i,k_2+j}) + \alpha^d \sum_{(i,j)\in I^{dd}} s^{dd}_{i,j}\phi^d_h(\mathbf{x}^d_{k_1+i,k_2+j}) \\
&= e^{\imath\boldsymbol{\theta}\cdot\mathbf{x}^d_{k_1,k_2}/\mathbf{h}} \left(\alpha^u \sum_{(i,j)\in I^{du}} s^{du}_{i,j} e^{\imath(\theta_1(i+\delta_1-\delta_1')+\theta_2(j+\delta_2-\delta_2'))} + \alpha^d \sum_{(i,j)\in I^{dd}} s^{dd}_{i,j} e^{\imath(\theta_1 i+\theta_2 j)} \right).
\end{aligned}
$$

Then, from these expressions we obtain

$$
\begin{aligned}
(L_h\phi_h(\boldsymbol{\theta},\cdot)) &= L_h \begin{bmatrix} \phi^u_h & \phi^d_h \end{bmatrix} \begin{bmatrix} \alpha^u \\ \alpha^d \end{bmatrix} \\
&= \left(\alpha^u \sum_{(i,j)\in I^{uu}} s^{uu}_{i,j} e^{\imath(\theta_1 i+\theta_2 j)} + \alpha^d \sum_{(i,j)\in I^{ud}} s^{ud}_{i,j} e^{\imath(\theta_1(i+\delta_1'-\delta_1)+\theta_2(j+\delta_2'-\delta_2))} \right) \phi^u_h \\
&+ \left(\alpha^u \sum_{(i,j)\in I^{du}} s^{du}_{i,j} e^{\imath(\theta_1(i+\delta_1-\delta_1')+\theta_2(j+\delta_2-\delta_2'))} + \alpha^d \sum_{(i,j)\in I^{dd}} s^{dd}_{i,j} e^{\imath(\theta_1 i+\theta_2 j)} \right) \phi^d_h \\
&= \begin{bmatrix} \phi^u_h & \phi^d_h \end{bmatrix} \widetilde{L}_h(\boldsymbol{\theta}) \begin{bmatrix} \alpha^u \\ \alpha^d \end{bmatrix} = \begin{bmatrix} \phi^u_h & \phi^d_h \end{bmatrix} \begin{bmatrix} \beta^u \\ \beta^d \end{bmatrix}, \qquad (2.31)
\end{aligned}
$$

from which the invariance property is demonstrated, and where

$$
\widetilde{L}_h(\boldsymbol{\theta}) = \begin{bmatrix} \displaystyle\sum_{(i,j)\in I^{uu}} s^{uu}_{i,j} e^{\imath(\theta_1 i+\theta_2 j)} & \displaystyle\sum_{(i,j)\in I^{ud}} s^{ud}_{i,j} e^{\imath(\theta_1(i+\delta_1'-\delta_1)+\theta_2(j+\delta_2'-\delta_2))} \\ \displaystyle\sum_{(i,j)\in I^{du}} s^{du}_{i,j} e^{\imath(\theta_1(i+\delta_1-\delta_1')+\theta_2(j+\delta_2-\delta_2'))} & \displaystyle\sum_{(i,j)\in I^{dd}} s^{dd}_{i,j} e^{\imath(\theta_1 i+\theta_2 j)} \end{bmatrix}, \tag{2.32}
$$

is the representation of L_h in $\mathcal{F}(\mathcal{V}_h)$. Notice that in this case, this representation is a $2\times 2-$matrix.

In the particular case of the discretization considered for our model problem, the Fourier domain representation of the discrete operator depending on the angles characterizing the triangular grid reads

$$
\widetilde{L}_h(\boldsymbol{\theta}) = \frac{1}{meas(T)} \begin{bmatrix} t_\alpha + t_\beta - t_{\alpha\beta} & E(t_{\alpha\beta}e^{-\imath\theta_2} - t_\beta - t_\alpha e^{\imath\theta_1}) \\ E^{-1}(t_{\alpha\beta}e^{\imath\theta_2} - t_\beta - t_\alpha e^{-\imath\theta_1}) & t_\alpha + t_\beta - t_{\alpha\beta} \end{bmatrix} \tag{2.33}
$$

where the following abbreviations have been used: $t_\alpha = \tan(\alpha)$, $t_\beta = \tan(\beta)$, $t_{\alpha\beta} = \tan(\alpha+\beta)$, and $E = e^{\imath(\theta_1(\delta_1'-\delta_1)+\theta_2(\delta_2'-\delta_2))}$.

2.3.1 Smoothers and their representations in the Fourier space

In this work some classical smoothers will be considered together with some novel smoothers which have been appropriately designed to deal with the particular anisotropies that can appear in the Voronoi meshes. All these smoothers are based on a splitting of the discrete operator L_h as $L_h = L_h^+ + L_h^-$, that is, a decomposition on the positive and negative parts of the operator which correspond to the updated and non-updated unknowns before the current step, see (49). Next, each smoother is described and the corresponding Fourier domain representation is given. For the sake of simplicity, only the positive part of the operator is displayed.

2.3.1.1 Jacobi

First, we consider the damped Jacobi iteration. In this case, L_h^+ is just equal to the diagonal part of operator L_h, denoted by D_h, and the iteration matrix of this smoother is given by $S_h = I_h - \omega D_h^{-1} L_h$, where I_h is the identity operator and ω is a relaxation parameter. The Fourier domain representation of operator S_h, $\widetilde{S}_h(\boldsymbol{\theta})$, is $\widetilde{S}_h(\boldsymbol{\theta}) = \widetilde{I}_h(\boldsymbol{\theta}) - \omega \widetilde{D}_h(\boldsymbol{\theta})^{-1} \widetilde{L}_h(\boldsymbol{\theta})$, where $\widetilde{I}_h(\boldsymbol{\theta})$ is the $2 \times 2-$identity matrix, $\widetilde{L}_h(\boldsymbol{\theta})$ is given in (2.33), and

$$\widetilde{D}_h(\boldsymbol{\theta}) = \frac{1}{meas(T)} \left[\begin{array}{cc} t_\alpha + t_\beta - t_{\alpha\beta} & 0 \\ 0 & t_\alpha + t_\beta - t_{\alpha\beta} \end{array} \right].$$

2.3.1.2 Gauss-Seidel

Next, we consider the lexicographic Gauss-Seidel method, in which the grid-points are updated with an order dictated by the numbering of the Voronoi points. Since two Voronoi points are associated with the same pair of indexes (k_1, k_2), first the Voronoi point associated with a down-oriented triangle is relaxed and then that associated with the up-oriented triangle is updated. Figure 2.18 illustrates the update order. Considering this order, the Fourier representation of the positive part of the splitting is in this case

$$\widetilde{L}_h^+(\boldsymbol{\theta}) = \frac{1}{meas(T)} \left[\begin{array}{cc} t_\alpha + t_\beta - t_{\alpha\beta} & E(t_{\alpha\beta} e^{-\imath\theta_2} - t_\beta) \\ -E^{-1} t_\alpha e^{-\imath\theta_1} & t_\alpha + t_\beta - t_{\alpha\beta} \end{array} \right]. \tag{2.34}$$

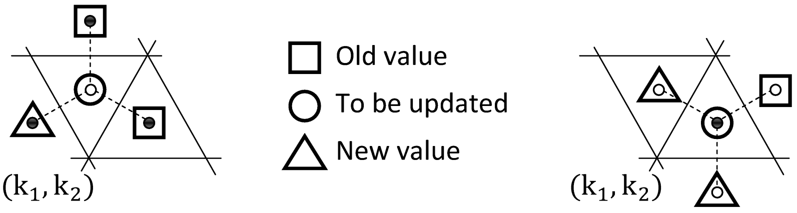


Figure 2.18: Update order in Gauss-Seidel smoother for a Voronoi point associated with a down-oriented triangle (left) and a Voronoi point associated with an up-oriented triangle (right).

2.3.1.3 Red-black smoother

Now, we consider a point-wise pattern relaxation that will be very efficient for equilateral triangulations. Since the Voronoi grid is naturally subdivided into two sub-grids $\mathcal{V}_h^u$ and $\mathcal{V}_h^d$, we can consider a scheme which updates first the grid-points in $\mathcal{V}_h^u$, and secondly those points in $\mathcal{V}_h^d$. Thus, this scheme consists of two partial relaxation steps, in the first one the Voronoi points associated with up-oriented triangles are relaxed, and in the second one those grid-points associated with down-oriented triangles are updated. In this case, the Fourier representation of the positive part of the decomposition reads

$$\widetilde{L}_h^+(\boldsymbol{\theta}) = \frac{1}{meas(T)} \begin{bmatrix} (t_\alpha + t_\beta - t_{\alpha\beta})/\omega^u & 0 \\ E^{-1}(t_{\alpha\beta}e^{\imath\theta_2} - t_\beta - t_\alpha e^{-\imath\theta_1}) & (t_\alpha + t_\beta - t_{\alpha\beta})/\omega^d \end{bmatrix}, \qquad (2.35)$$

where ω^u and ω^d are relaxation parameters associated with each one of the partial steps.

2.3.1.4 Diamond smoother

Here we introduce a novel block-wise smoother suitable for one type of the Voronoi anisotropic grids arising when almost-right triangular grids are considered, see Figure 2.19. It consists of simultaneously relaxing two Voronoi points, each one associated with a different oriented triangle. This smoother will be efficient when these two points are very close. For instance, in the case of the almost-right triangulation displayed in Figure 2.19, the coupling of the Voronoi points will be as shown in Figure 2.20 (a), and then the Fourier domain representation of the positive part of the operator is

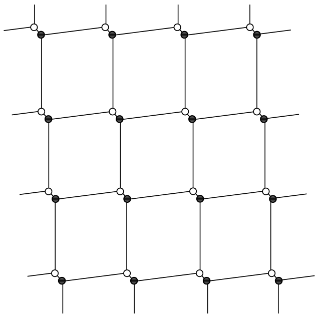
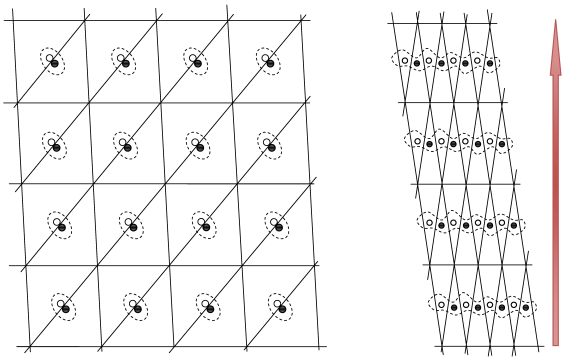


Figure 2.19: Voronoi anisotropy appearing when almost-right triangular grids are considered.

$$\widetilde{L}_h^+(\boldsymbol{\theta}) = \frac{1}{meas(T)} \begin{bmatrix} t_\alpha + t_\beta - t_{\alpha\beta} & E(t_{\alpha\beta}e^{-\imath\theta_2} - t_\beta - t_\alpha e^{\imath\theta_1}) \\ -E^{-1}t_\beta & t_\alpha + t_\beta - t_{\alpha\beta} \end{bmatrix}. \tag{2.36}$$



(a) (b)

Figure 2.20: (a) Diamond smoother and (b) wormy smoother on infinite grids.

2.3.1.5 Wormy smoother

We finish the description of the smoothers with the so-called wormy-smoother. As in the previous case, this is a block-wise smoother, and as will be seen in the results section it will be very efficient for the remaining case of Voronoi anisotropic meshes, that is, in the case of triangulations with a very small angle. This smoother consists of simultaneously updating all the Voronoi points associated with the triangles situated

	Jacobi		Red-Black	
ν	ρ_{2g}	ρ_h	ρ_{2g}	ρ_h
1	0.75	0.71	0.53	0.52
2	0.56	0.55	0.24	0.23
3	0.41	0.42	0.13	0.13
4	0.31	0.31	7.2×10^{-2}	7.0×10^{-2}
5	0.23	0.22	4.1×10^{-2}	4.0×10^{-2}

Table 2.2: LFA two-grid convergence factors, ρ_{2g}, and measured two-grid convergence rates ρ_h for an equilateral triangular grid.

in the direction of the edge opposite to the vertex corresponding to this small angle, see Figure 2.20 (b). Notice that in this case, tridiagonal systems must be solved and therefore it is more expensive than the other previously considered smoothers. However, in some situations it is mandatory. For this smoother, the Fourier representation of the positive part of the operator is given by

$$\widetilde{L}_h^{+}(\boldsymbol{\theta}) = \frac{1}{meas(T)} \begin{bmatrix} t_\alpha + t_\beta - t_{\alpha\beta} & E(t_{\alpha\beta}e^{-\imath\theta_2} - t_\beta - t_\alpha e^{\imath\theta_1}) \\ -E^{-1}(t_\beta + t_\alpha e^{-\imath\theta_1}) & t_\alpha + t_\beta - t_{\alpha\beta} \end{bmatrix}. \qquad (2.37)$$

2.3.2 Results of local Fourier analysis

Now, we are going to compare the results from the multigrid experiments and those obtained by the LFA in order to validate both. Hence, we have chosen the same inter-grid transfer operators presented in Section 2.2.1. For the first experiment we will consider an equilateral triangular grid and we will use Jacobi, and red-black smoothers. In Table 2.2 we show the asymptotic convergence factor predicted by the LFA and the one computationally obtained by the multigrid code. From this table, we can observe that the convergence factors are very well predicted by the LFA in all the cases. Using this analysis we have looked for the damping parameters to improve the considered smoothers. For example, if three smoothing steps are used, we have found that in the case of Jacobi, the use of a relaxation parameter $\omega = 0.9$ improves the two-grid convergence factor to $\rho_{2g} = 0.34$. For the red-black smoother if we use optimal relaxation parameters $\omega^u = 1.2$ and $\omega^d = 1.1$, the improvement is more than half, reducing the convergence factor from 0.13 to 0.06.

	Red-Black		(ω^u, ω^d)-Red-Black		Diamond	
ν	ρ_{2g}	ρ_h	ρ_{2g} (ω^u, ω^d)	ρ_h	ρ_{2g}	ρ_h
1	0.97	0.96	0.96 (1.3, 0.6)	0.95	0.45	0.45
2	0.90	0.90	0.36 (7.35, 1)	0.36	0.20	0.20
3	0.84	0.83	0.27 (1.05, 5.1)	0.27	9.2×10^{-2}	9.0×10^{-2}
4	0.78	0.78	0.24 (1.1, 3.55)	0.23	4.1×10^{-2}	4.0×10^{-2}
5	0.73	0.72	0.21 (1.15, 2.85)	0.20	2.6×10^{-2}	2.4×10^{-2}

Table 2.3: LFA two-grid convergence factors, ρ_{2g}, and measured two-grid convergence rates ρ_h for an almost-right triangular grid.

Next, we deal with an almost-right triangle. In Table 2.3 we show the behaviour of diamond smoother together with the red-black and the $(\omega^u, \omega^d)-$red-black smoothers. For different number of smoothing steps, the two-grid convergence factors predicted by LFA are provided together with the experimentally computed asymptotic two-grid convergence factors. As we can observe, the LFA tool results very useful since the prediction of the analysis matches very accurately the computational results. In the case of the diamond smoother, the use of an extra-relaxation parameter is not useful at all, hence we do not include any results for the relaxed variant. On the other hand, since the results obtained by red-black smoother give significant improvements in the case of equilateral triangulations, we wonder if this can be still useful for this type of anisotropic grids. Then, in Table 2.3, we have included for each number of smoothing steps, the optimal relaxation parameters predicted by LFA and the corresponding two-grid convergence factors. We can observe a very large improvement in all the cases by using very high values for the relaxation parameters. Anyway, despite this big improvement in the behaviour of the red-black smoother, diamond smoother is by far more suitable, and therefore we still recommend this smoother for this type of anisotropic grids.

Finally, we focus on the wormy smoother and to test its behaviour we will use a sharp isosceles triangle with two angles of 85^o . In Table 2.4, three different smoothers are considered: diamond, wormy and a relaxed variant of this latter. Again, the two-grid convergence factors predicted by LFA are compared with those experimentally computed factors for different numbers of smoothing steps. Notice that the introduc-

	Diamond		Wormy		ω-Wormy	
ν	ρ_{2g}	ρ_h	ρ_{2g}	ρ_h	ρ_{2g}	ρ_h
1	0.97	0.96	0.57	0.52	0.46	0.46
2	0.94	0.94	0.33	0.29	0.21	0.20
3	0.92	0.90	0.19	0.19	9.4×10^{-2}	8.7×10^{-2}
4	0.89	0.88	0.11	0.10	4.3×10^{-2}	4.6×10^{-2}
5	0.86	0.85	6.2×10^{-2}	5.6×10^{-2}	1.9×10^{-2}	2.1×10^{-2}

Table 2.4: LFA two-grid convergence factors, ρ_{2g}, and measured two-grid convergence rates ρ_h for an almost-right triangular grid.

tion of the relaxation parameter $\omega = 0.9$ yields to big improvements. However, this parameter ω highly depends on the triangle shape.

Next, we want to analyse the behaviour of a V-cycle. Therefore, we will perform a three-grid analysis for the three illustrative triangular configurations, using the recommended smoother in each one. In Table 2.5, with two pre- and two post-smoothing steps, the three grid convergence factors predicted by the LFA and the experimentally computed factors by using a V-cycle with three and nine levels (the finest grid is obtained by applying nine refinement levels) are shown. From these results we can see that if the recommended smoother is chosen, a V-cycle results an efficient strategy. However, we have observed that if the chosen smoother is not strong enough, the V-cycle provides convergence factors depending on the number of levels used in the calculations. In Figure 2.21, for three different triangular domains, we display the experimentally computed asymptotic convergence rates obtained by using different number of levels. The first picture corresponds to a triangle with angles $\alpha = 60^o$ and $\beta = 75^o$, and the results for red-black and wormy smoothers are presented. Whereas the wormy smoother provides convergence factors independent of the used number of levels, the convergence factors of the other smoothers get worse. Notice that in this latter case, even divergence is achieved. In the second picture, a triangle with angles $\alpha = 45^o$ and $\beta = 55^o$ is considered. For this case, red-black and diamond smoothers are analysed, resulting the convergence factors provided by diamond smoother independent on the number of used levels. Again, red-black smoother yields even divergence in some cases. The last picture corresponds to a triangle with angles $\alpha = 70^o$ and $\beta = 85^o$,

	equilateral	almost-right	isosceles 85^o
	(RB)	(Diamond)	(Wormy)
ρ_{3g}	7.2×10^{-2}	4.2×10^{-2}	0.11
ρ_h^3	8.1×10^{-2}	4.2×10^{-2}	9.8×10^{-2}
ρ_h^9	0.11	4.1×10^{-2}	9.5×10^{-2}

Table 2.5: LFA three-grid convergence factors, ρ_{3g}, and experimentally computed factors using three and nine multigrid levels (the number of nodes is the same).

and in this case the analysed smoothers are diamond and wormy. Whereas if diamond is considered, we observe a deterioration with regard to the number of levels, wormy smoother provides very good results. We can conclude that the V-cycle can be used in practice whenever a strong enough smoother is considered.

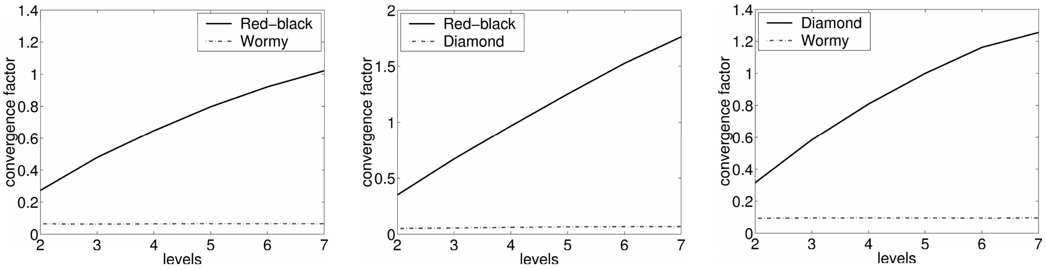


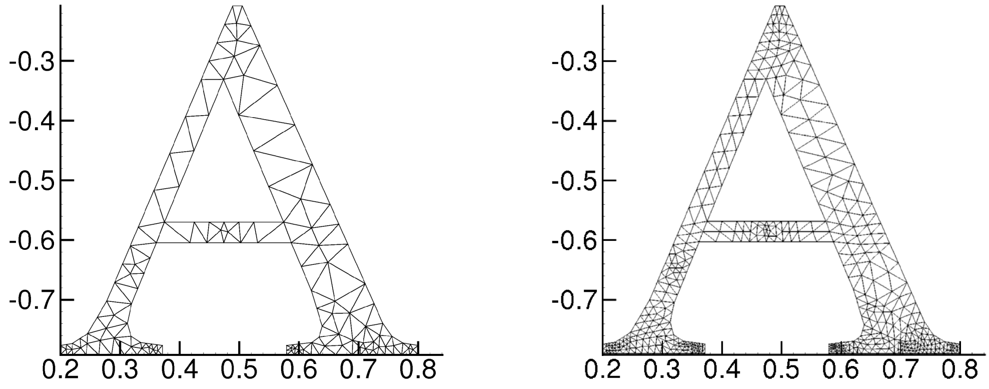
Figure 2.21: Experimentally computed asymptotic convergence factors of a V-cycle by using different numbers of levels, for three different triangular configurations.

2.4 Numerical experiments on semi-structured grids

In this subsection, we consider two model problems: a Laplace problem in an A-shaped domain, and a convection-diffusion problem in a square domain. The strategy shown in Figure 2.15 has been applied to perform the numerical experiments. Throughout all the thesis, *aCute* software (15, 16), which is based on *Triangle* (47, 48), has been used to generate the initial unstructured acute Delaunay triangulation.

2.4.1 Laplace problem in an A-shaped domain

In the first numerical experiment, model problem (2.1) is solved in an A-shaped domain, as shown in Figure 2.22. To this purpose, an initial unstructured grid composed of 201 triangles is considered, as depicted in Figure 2.22(a). From this mesh, a hierarchy of grids is built by applying regular refinement on each coarse triangle, and the grid resulting after one refinement step is shown in Figure 2.22(b), as an example.



(a) (b)

Figure 2.22: (a) Coarsest unstructured grid. (b) Grid obtained after one regular refinement level.

Following the guideline displayed in Figure 2.15, we have chosen the most efficient smoother for each triangle of the input unstructured triangulation. Selected smoothers can be seen in Figure 2.23. It is important to remark that to achieve the desired global convergence factor, an extra relaxation on Voronoi nodes close to the internal boundaries of the initial coarsest grid, has been necessary.

After applying the proposed strategy by using a $F(2,2)-$cycle, the number of cycles necessary to reduce the initial residual in a factor of 10^{-10} and the CPU-time, together with the asymptotic convergence factors are shown in Table 2.6 for different numbers of refinement levels. It is observed that the convergence is independent of the space discretization parameter h, and that in few iterations the residual is reduced as desired. Moreover, an asymptotic convergence factor about 0.12 is obtained. In order to know

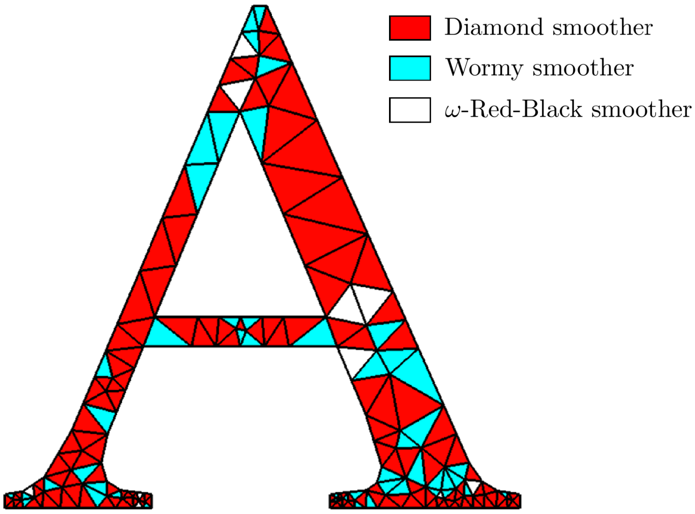


Figure 2.23: Different smoothers for the triangles composing the initial triangulation of the A-shaped domain.

the behaviour of the V-cycle, in the same table, we have also added the results corresponding to a $V(2,2)-$cycle, following the same strategy than for the F-cycle, that is, the guideline shown in Figure 2.15. From these results, we can observe a slight deterioration in the asymptotic convergence factors with respect to the number of unknowns. This degradation of the convergence may be due to the fact that the accuracy of the transfer operators is insufficient to fulfil the well-known rule of thumb, see (5, 25),

$$m_P + m_R > m,$$

where m_P and m_R are defined as the highest degree plus one of the polynomials that are interpolated exactly by the prolongation and the restriction respectively, and m is the order of the partial differential equation to be solved. Here, injection is considered, and then, m_P and m_R are both equal to one. Since the order of the equation is two, the strict inequality is not satisfied. Thus, it is not possible to guarantee efficient convergence rates due to the weakness of the coarse-grid correction. Anyway, from the practical point of view, it could be interesting the use of V-cycles for solving this type of problems, since in very few iterations the desired convergence is achieved.

		V(2,2)			F(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)	Cycles	ρ_h	Time (s)
4	51456	9	0.16	1.78	8	0.09	3.27
5	205824	9	0.19	3.62	8	0.11	6.32
6	823296	10	0.21	10.16	9	0.11	15.96
7	3293184	11	0.23	34.69	9	0.12	43.66
8	13172736	11	0.26	123.65	9	0.12	142.52

Table 2.6: Number of iterations to reduce the initial residual in a factor of 10^{-10}, corresponding asymptotic convergence rates and CPU-times for different numbers of refinement levels, by using a V-cycle and an F-cycle.

2.4.2 Convection-diffusion problem on a square domain

The strategy proposed can be applied to more complex problems as, for example convection-diffusion, which can be written as:

$$-\nabla\cdot(\nabla v + \mathbf{b}\, v) = f, \text{ in } \Omega, \tag{2.38}$$

where $\mathbf{b}(\mathbf{x})$ is a given velocity field, whose divergence is assumed to be zero. In order to obtain a difference scheme by the cell-centred finite volume method, we follow the same approach that we have explained in detail in Section 2.1, by using a central difference scheme to approximate the convective term, see (41). In this numerical experiment an square domain of unit length and Dirichlet boundary conditions are considered, and a constant vector $\mathbf{b} = (1, 0)$ is fixed in the whole domain. Thus, the following equation on each of the grid-nodes $\mathbf{x}_c$ results:

$$-\frac{1}{\text{meas}(T)}\sum_{i=1}^{3}\left(\text{meas}(l_i)\left(\frac{v_h(\mathbf{x}_i)-v_h(\mathbf{x}_c)}{d_i}+\mathbf{b}\cdot\mathbf{n}_i\frac{v_h(\mathbf{x}_i)+v_h(\mathbf{x}_c)}{2}\right)\right) = f(\mathbf{x}_c). \tag{2.39}$$

We consider an initial unstructured grid, composed of 96 triangles, as seen in Figure 2.24, in which, for illustration, the dual Voronoi mesh has been displayed. The hierarchy of grids is obtained by regular refinement. As the convective part of the problem is not dominant, and its derivatives are of lower order, the behaviour of the multigrid will be similar to that obtained for a pure diffusive problem. Therefore we will follow the guideline given in Figure 2.15 to choose the suitable local smoother on each input triangle, and this selection is displayed in Figure 2.25. The proposed

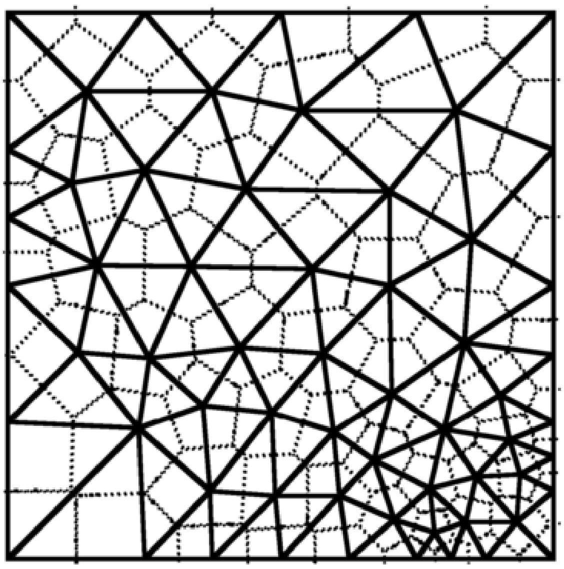


Figure 2.24: Coarsest unstructured grid together with the associated Voronoi mesh.

geometric multigrid method is applied to solve the corresponding large sparse linear system of equations. First, an $F(2,2)-$cycle is used to test the independence of the multigrid convergence with regard to the discretization parameters. In Table 2.7, for different numbers of refinement levels, the asymptotic convergence rate, ρ_h, and the number of iterations necessary to reduce the initial residual in a factor of 10^{-10}, are displayed together with the CPU-time. Similarly to the previous numerical experiment, the results corresponding to apply a $V(2,2)-$cycle are also shown in this table. Again, a deterioration of the asymptotic convergence factor is observed, but the application of this cycle can be interesting due to the small number of iterations necessary to reach the convergence.

2.5 Implementation

In this section, we are going to go deeper into the details of programming a multigrid algorithm for cell-centred discretizations on Voronoi meshes associated with semi-structured triangular grids.

The semi-structured approach requires the use of a connectivity array for the unstructured triangulation. Nevertheless, the triangles resulting from the regular refine-

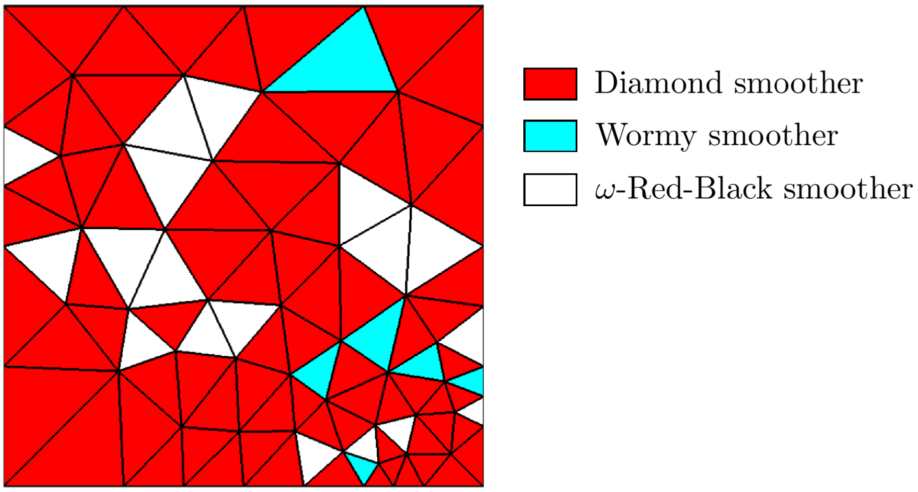


Figure 2.25: Different smoothers considered on each triangular block of the input grid.

ment of this initial unstructured mesh, have a structure ordering and therefore, the connectivity matrix is unnecessary. Hence, its use for all the triangles would result in a waste of resources, and stencils are used for that purpose. However, storing information about triangles is a challenge by itself since the data structures use to have a square shape.

Also, we will comment how to overcome the problems that appear in the application of multigrid on semi-structured grids, by using extra-relaxation processes. And more specifically, the particular case in which the domain is composed of two extremely sharp triangles connected.

2.5.1 Data storage for cell-centred discretizations on structured triangular grids

In all the numerical methods for the solution of partial differential equations, it is necessary to read data from the memory several times, apart from performing many mathematical operations. While the mathematical operations are done extremely fast, the access to the memory is slow and strongly depends on the data structure. Therefore, the way in which we store the information as well as the speed to read and write data from the memory will be a key point in the performance of the overall code. Thus,

		V(2,2)			F(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)	Cycles	ρ_h	Time (s)
4	24576	9	0.19	0.74	8	0.09	1.19
5	98304	10	0.20	1.76	8	0.09	2.54
6	393216	10	0.21	4.54	8	0.09	6.07
7	1572864	11	0.22	15.53	8	0.09	17.11
8	6291456	11	0.22	55.16	8	0.09	56.33

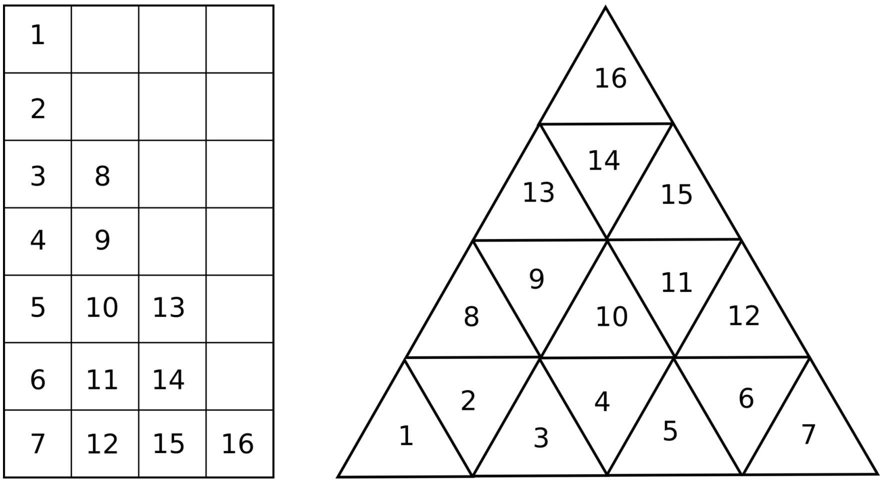
Table 2.7: Number of iterations necessary to reduce the initial residual in a factor of 10^{-10}, corresponding asymptotic convergence rates and CPU-times for different numbers of refinement levels, by using a V-cycle and an F-cycle.

we can improve or reduce the performance of the algorithm depending on the used data structure. A desired data structure must be easy to use by the user and suitable for the performance of the computer. Therefore, one has to design a user friendly data structure depending also on the peculiarities of the programming language. In the particular case of this thesis, Fortran90 was the chosen language. In Fortran, the natural order of storage is column-major. Not following this order may make the data access time three times slower (11).

Many programmers that need to solve PDEs claim that the most efficient way to store the data is by using a simple one-dimensional array. This is commonly used when rectangular grids are considered. In this case, each row has the same number of nodes, which makes easy the use of one dimensional arrays. To access to the data in column x, one just need to read from the memory position $(x-1) * (size(column)) + 1$ to $x * size(column)$. However, in the case of triangles, the size of the column is not fixed. Therefore, to obtain the information from a certain column, one needs to use a formula instead of a constant to know the beginning and the end of that column.

Due to the already mentioned difficulties, we propose a simpler data structure based on 2-dimensional arrays. The storage of triangles have the complication that, if one wants to keep the logical structure of the triangle, this means that each row of a triangle is in a different column of the array, that array would be half empty, see Figure 2.26.

This way of storage leaves a lot of empty space, and therefore it is undesirable. Nonetheless, if we add an extra row in the upper part of the array, then we have the same number of empty places, where we can store, for example, the right hand side of



1			
2			
3	8		
4	9		
5	10	13	
6	11	14	
7	12	15	16

Figure 2.26: A triangle structure stored in an array, keeping the row and column coherence

the equation. However it is still not easy to move through nodes. For example, triangle number two is just below triangle number eight but in the array that coherence is not kept. Besides, to access to the right hand side data there is no a simple way. For example, node 1, that would be stored in the position $(1, 2)$ of the array, would have its right hand side in position $(4, 7)$. Despite possible, this way of storing is not easy and therefore likely to make mistakes. Hence, we will consider a new data structure.

If we have a look to Figure 2.26, we can distinguish two sorts of triangles. Some of them are up-oriented triangles and the rest are down-oriented triangles. Each up-oriented triangle is surrounded only by down-oriented triangles and vice-versa. This means, that to access to the neighbouring triangles the up-oriented triangles will only need information from the down-oriented triangles and the other way around. In this way, a new possible data structure, a modification of the previous one, arises in a natural way, to store each kind of triangles in a different array. However, we still have the problem of the inherent difficulty to access to the right hand side data. Regarding this latter, one solution is to keep the diagonal of the array empty, and to store the data of the right hand side in a symmetrical fashion, making simpler the access to the right hand side data, since the solution and the right-hand side data corresponding to the same node will have a symmetric numbering. For example: following Figure 2.27 the position of node 1 is $(1, 2)$ and its right hand side is stored in $(2, 1)$, which is much easier to find than in the previous case, were both position had nothing in common. Also, the access to those triangles that are surrounding a concrete one is very easy. For example, if we want to find the neighbours of the up-oriented triangle number 10, whose storage position is $(2, 4)$, we just have to go to the array of down-oriented triangles and get the nodes corresponding to $(2, 4)$, $(2, 3)$ and $(1, 3)$, nodes 11, 9 and 4 respectively. That is, the way to access to the neighbours is more or less the same as graphically we can observe, the left triangle is also in the left position in the array, the down triangle is also below and only in the right node one has to remember that it is stored in the same position of the other array. Similarly, for a down-oriented triangle, the method to obtain the neighbours is symmetric. That is, considering triangle 9 in the position $(2, 3)$ of the down-oriented triangles array, the neighbours, 8 (left), 10 (right) and 13 (up), are stored in the up-oriented triangles array in positions $(2, 3)$, $(2, 4)$ and $(3, 4)$ respectively.

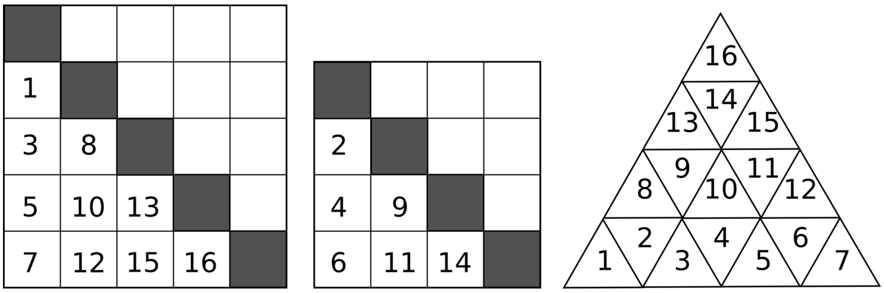
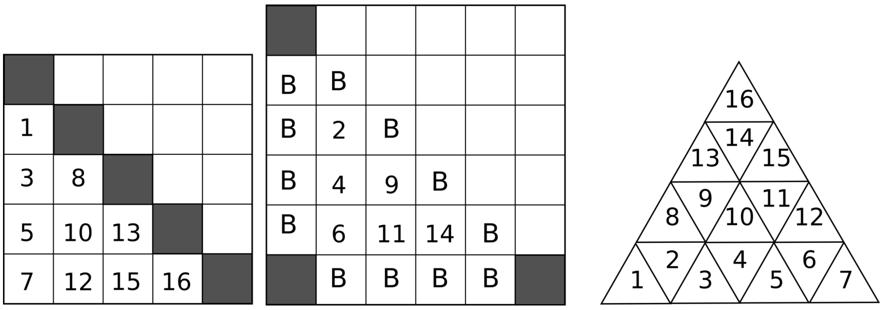


Figure 2.27: A triangle structure stored in two arrays, one for up-oriented triangles and another for down-oriented triangles.

On the other hand, there is still memory that is not going to be used. More concretely, the amount of useless memory is exactly of the size of the length of the array, $lenght(array)$, whereas the total amount of data is $lenght(array)^2$. Hence, the memory that we are not going to use is almost irrelevant compared with the total amount of used memory.

Regarding the storage of data corresponding to Dirichlet boundary conditions, the idea is to modify the down-oriented triangles array, by adding space to store the boundary data. We should store it in the places where we would look for neighbours from the up-oriented triangles that are touching the boundaries. For that, it is necessary to make the down-oriented array bigger, then it has to start from node $(0, 0)$, while the up-oriented one must begin from $(1,1)$. In this way, taking into account the boundary data will be as natural as accessing the neighbouring triangles. For example, for triangle 1, in position $(1,2)$, we would look for neighbours in the positions $(1,2)$, $(1,1)$ and $(0,1)$ in the down-oriented triangles array. Therefore, the natural way to store the boundary data is to use the diagonal and also to add a column before the first one and an extra row after the last one that we can see by applying the same idea to node 7, see Figure 2.28.



■				
1	■			
3	8	■		
5	10	13	■	
7	12	15	16	■

■					
B	B				
B	2	B			
B	4	9	B		
B	6	11	14	B	
■	B	B	B	B	■

Figure 2.28: A triangle structure stored in two arrays, one for up-oriented triangles and another for down-oriented triangles, with boundary data storage, B.

Finally, we have obtained a data storage which does not differentiate from boundary nodes to internal ones and keeps the logical structure of a triangle. It has the problem that one needs to store the data by pairs (left side of the diagonal and right side of the diagonal), and that one will always work with two different arrays. Nevertheless, these two drawbacks are not a big problem in the solution of PDEs. The information is usually required by pairs: solution and right hand side, residual and previous step time solution. Regarding the use of two arrays, it may affect the performance if we are not cautious. Thus, we will always try to work first with one kind of triangles and later with the other one.

2.5.2 Extra-relaxation process in semi-structured grids

The idea of applying multigrid on semi-structured grids is very simple. We will treat differently all the triangles that form the initial unstructured triangulation, see Figure 2.29. Nonetheless, we have to communicate them. The natural way is to consider each triangle to have Dirichlet boundary conditions, where these boundary conditions contain the overlap from neighbouring triangles, see Figure 2.30. However, these "boundaries" have to be updated after every modification of their values, see Figure 2.31, for example, after the smoothing process and the prolongation update. Nevertheless, this methodology some times yields some problems that will require the use of an extra-relaxation process.

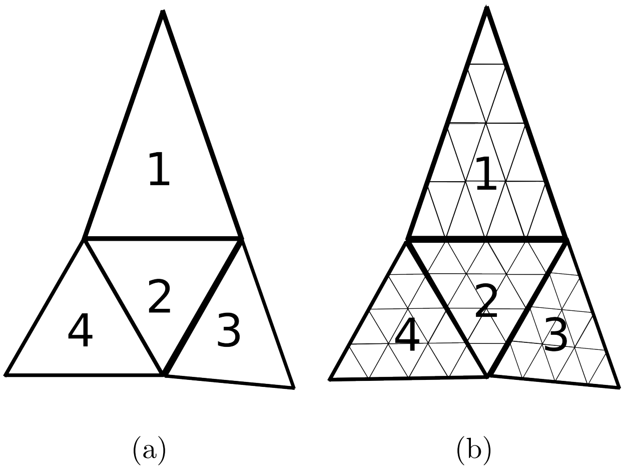


Figure 2.29: (a) Initial unstructured grid of a domain. (b) Semi-structured grid obtained after one refinement level.

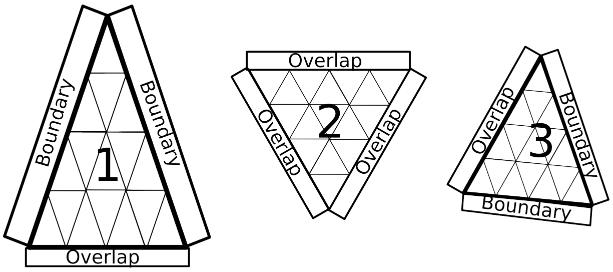


Figure 2.30: Three isolated triangles after a virtual splitting.

The first reason of considering an extra-relaxation process can be easily seen when considering the red-black smoother. To perform this method, we subdivide all the triangles in the grid, and therefore, the nodes into two different types: up- and down-oriented triangles, each of which can be updated simultaneously since each one only uses information from the other one. This means that, if we have two triangles of the initial grid sharing a common edge, and we want to apply a red-black smoother

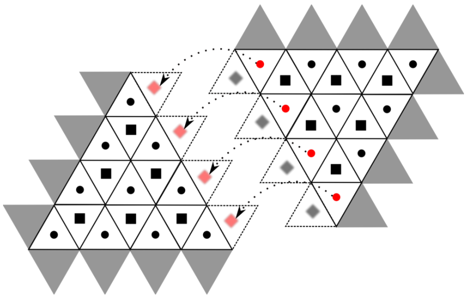
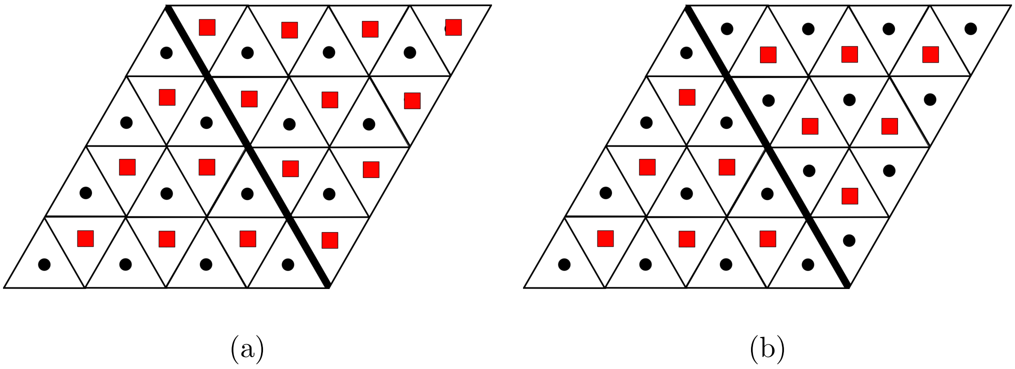


Figure 2.31: Overlapping update process.

we should keep the coherence across the triangles, see Figure 2.32 (a). Nevertheless, this can be very difficult if we consider not two triangles but hundreds of them. The splitting of the nodes for the red-black smoother applied on two coarse triangles is depicted in Figure 2.32 (b). We can see that the nodes at the overlapping regions belong to the same group (Black nodes). This means that one is not carrying out a true red-black smoother in the whole domain since the rule that defines a red-black smoother is not fulfilled by all the nodes. Hence, we are performing an "almost" red-black smoother, which, in the best case scenario will be as good as the true red-black smoother. Nonetheless, it seems that for the considered cell-centred discretizations we obtain a worse performance.



(a)

(b)

Figure 2.32: (a) Red-Black smoother in two triangles keeping the coherence between them. (b) Practical red-black smoother.

The second reason comes from the error smoothness that we obtain in the whole domain after smoothing each triangle by its own. We obtain a smooth error on the triangles, but it seems not to be smooth enough at the nodes close to the common edges between the triangles. This effect is due to the fact that, as each triangle is treated separately, we obtain a smooth error on each single domain, but there can be a jump in the values from one triangle to its neighbours, which is sufficient to affect to the multigrid convergence factor. This behaviour can be easily appreciated in Figure 2.33, where we have depicted the error after two smoothing steps. The error on the whole domain seems smooth, but at the common edges of the triangles non smooth lines appear, deteriorating the convergence rate from a theoretical value of 0.10 to about 0.90. However, this effect can be even more dramatic, making the multigrid even to diverge in more complicated geometries. As conclusion, this issue must be solved in order to obtain an optimal method.

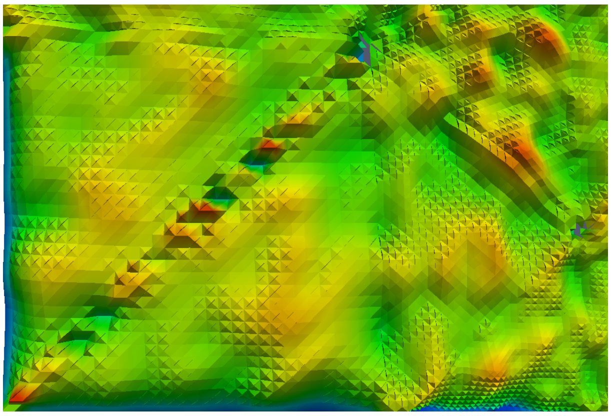


Figure 2.33: Error after two smoothing steps, following the relaxation strategy presented in Figure 2.15.

In order to get global smooth error it seems a good idea to apply a strong smoother in the nodes of the triangles that are close to the internal boundaries. In our case, the strongest considered relaxation scheme is the wormy smoother. Nevertheless, we would like to simultaneously update all the nodes, from both triangles, that are close to their common edge. Thus, we can consider a variant of the wormy smoother which smooths the lines at both sides at the same time, see Figure 2.34. However, we obtain a block-tridiagonal system of equations to solve, which makes this method really costly. Nonetheless, since the amount of nodes to be smoothed by this extra-relaxation process is of the order of $\sqrt{Totalnodes}$, the required amount of time to do that is small compared with the cost of the rest of the multigrid process, which is proportional to the total number of nodes.

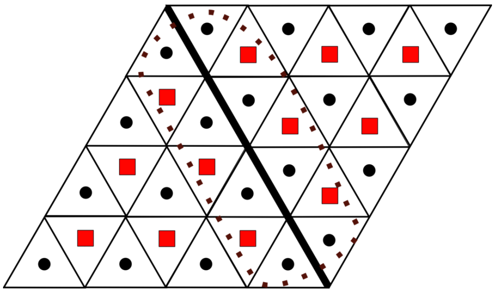


Figure 2.34: Wormy smoother extended to two lines of two different triangles at the same time.

Nevertheless, it is not always enough to perform the extra-relaxation of the error at the common edges of the triangles. The nodes closest to the vertexes of the triangles composing the initial grid are more complicated to smooth. To overcome this difficulty we could apply the same idea as before to those points, this means to simultaneously smooth all the nodes closest to a vertex. However, in our experience, we have realized that it is more robust to apply to each edge of the triangle a wormy smoother after the extra-relaxation process. This is more expensive from a computational point of view but its cost is still affordable.

In order to show the efficiency of this methodology, we have performed again the previous example given in Figure 2.33, by applying the extra-relaxation method. The result is presented in Figure 2.35, where we can see that now the error is smooth over the whole domain.

2.5.3 The case of extremely sharp triangles

We have previously commented the necessity of the extra-relaxation to overcome the smoothing problems derived from the use of cell-centred discretizations on semi-structured meshes. In the majority of the times, the presented approach is more than enough. However, there is a special case for which it is not enough. This concrete situation appears when two extremely-sharp triangles are in contact by the edge that is in the direction of the high anisotropy. In this case, the explained extra-relaxation is not enough to obtain the expected convergence factor predicted by the analysis. This is

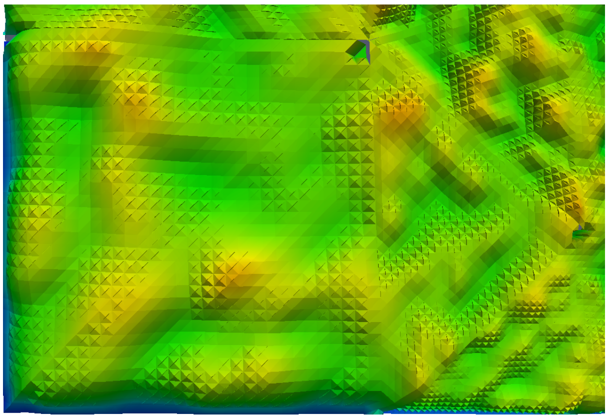


Figure 2.35: Error after two smoothing steps, following the relaxation strategy presented in Figure 2.15, and applying the extra-relaxation process.

due to the strong coupling between the two triangles. Theoretically, these two triangles should be smoothed together. The application of this smoothing strategy would lead to the same results obtained by the LFA. However, this would go against the essence of the semi-structured approach. Therefore, in order to keep the possibility of working with isolated triangles, in this section we are going to present two different possibilities to face this problem.

Since this problem seems to be caused by the splitting of a smoother in two, one option is to overlap the smoother of both triangles so the interchange of information is somehow kept. To this purpose, we are going to create a wormy smoother that will use some nodes of the other triangle, see Figure 2.36. However, in order to reduce the interchange of data between triangles to its bare minimum, it will not update the values from the neighbouring triangle. We will perform this smoother on both triangles so in this way, we have to take information from the other triangle but we do not modify its values. Hence, it is a compromise between the semi-structured strategy and the search for a good performance.

In Table 2.8, we show the convergence factors obtained with this smoother, by using an overlapping of just one node, for two sharp isosceles triangles. We compare the experimentally obtained convergence rate with the predictions by the LFA and we also consider a standard wormy smoother in the comparison. The improvement is significant for the first three cases, whereas for the last one both methods works very bad. The solution for this latter is to extend further the overlap, for example by using two or more nodes from the neighbouring triangle. As we increase the number of nodes from the other triangle used in the overlap, we are approaching to the optimal solution that will correspond to smooth both triangles together.

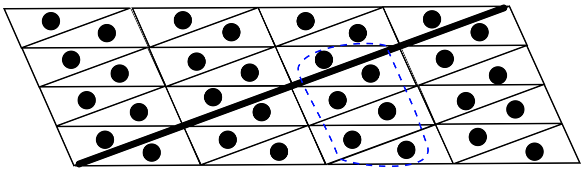


Figure 2.36: Extended wormy smoother, with an overlap of one node.

α	ω	LFA	Wormy modified	Wormy
85^o	1	0.1	0.1	0.20
86.5^o	1.5	0.1	0.12	0.25
87^o	1.5	0.11	0.2	0.33
88.5^o	1.5	0.11	0.45	0.56

Table 2.8: Convergence rates using the modified wormy and a the standard one, with different relaxation parameters and for different very sharp isosceles triangles.

Another different approach would be to use a more complex extra-relaxation method, that is, instead of performing the extra-relaxation explained in Section 2.5.2, to use a bigger one with four nodes deep, see Figure 2.37. The use of extra-relaxations allows us to keep the essence of the semi-structured approach better than the previous method. Nonetheless, we will have to solve a block tridiagonal system of 4 × 4 blocks, which is very expensive, and as a consequence the time spent on the extra-relaxation increases dramatically, requiring more time than the whole smoothing process. Nevertheless, the obtained results are satisfactory, as we can see in Table 2.9. In this table, we can appreciate that the experimentally computed convergence factors stay close to those predicted by the LFA up to a triangle with base angle of 87.6 degrees. In the case of an isosceles triangle with bases angle of 88.5 degrees the convergence deteriorates, although the convergence rate is still acceptable. Regarding the time consumption, one can always apply this special extra-relaxation smoother to the particular case of these two connected triangles and the cheaper strategy to the rest of the triangles of the domain.

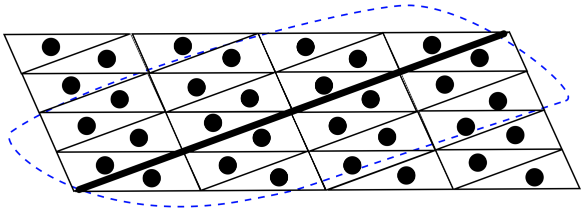


Figure 2.37: Four nodes deep extra-relaxation.

α	ω	LFA	Code
86.8^o	1	0.11	0.09
87.6^o	1	0.11	0.16
88.5^o	1.5	0.12	0.26

Table 2.9: Convergence rates using the special extra-relaxation method for three very sharp isosceles triangles.

Finally, as a conclusion we recommend the use of the modified wormy smoothers whenever possible instead of the extended extra-relaxation method. The algorithm is faster, and also the programming is much easier than the extended extra-relaxation method, where debugging is more complicated.

Chapter 3

Extension to more complex models

3.1 Multigrid methods with discontinuous coefficients

Elliptic equations with discontinuous coefficients arise from many areas of application, such as heat conduction through heterogeneous materials, flow in porous media with different porosities, etc. Consequently, the development of robust and efficient numerical algorithms for the solution of such problems has been an active area of research, and still remains a very important challenge of computational mathematics. Among the numerical methods approximating elliptic equations with jump discontinuities, we consider a cell-centred finite difference scheme on triangular grids. This latter can be derived from the simplification of the lowest order Raviart-Thomas mixed finite element method, as done in (17).

The diffusion equation with discontinuous coefficients is widely used in numerical simulation, and it has been established as a benchmark problem in books dedicated to multigrid methods (49, 56). It is known that when standard multigrid is applied for solving equations with highly varying discontinuous coefficients, a deterioration in the convergence of the method can appear, and even divergence can be observed. Then, many authors have intended to construct inter-grid transfer operators adequate to deal with rough coefficients, for example Wesseling and Khalil (31) and Kwak et al. (35, 37). In the case of vertex centred approximations, in (12, 14) it is suggested to use matrix-dependent transfer operators. The interpolator is motivated by the continuity of the flux, and following this idea, some prolongators based on the flux continuity have been successfully applied to accelerate the convergence of multigrid algorithms (1, 36, 39). However, the implementation of these methods requires additional storage and preliminary work, and can be fairly costly. In the case of cell-centred finite volume schemes, the main idea is to use Galerkin coarse-grid approximation. The disadvantage usually observed when using this approach is that the stencils of the coarse-grid operators are often larger than the corresponding fine-grid stencil, what is problematic especially in three dimensions. However, the use of simple transfer operators preserves the size of the fine-grid stencil.

Most of the existing geometric multigrid techniques to solve two-dimensional elliptic equations with discontinuous coefficients are only applied on simple domains and mainly on rectangular grids, and therefore, the design of geometric multigrid methods for complex domains is missing in the literature. To close this gap, in this section we will propose a block-wise multigrid algorithm for cell-centred discretizations on semi-structured triangular grids, which is suitable for relatively complex domains. Notice that many of the applications modelled by the diffusion equation with discontinuous coefficients deal with composite materials whose components have nearly constant diffusivity, but vary between them by several orders of magnitude. In these cases, it is quite common to idealize the diffusivity by a piecewise constant function. Semi-structured grids are suitable for capturing these piecewise functions, in the way that an initial unstructured input grid is considered fitting both the geometry of the domain and the piecewise diffusion coefficient.

In this section, very simple local inter-grid transfer operators are considered to make easier the communication between different input blocks. The accuracy of these transfer operators is insufficient to fulfil the rule of thumb, see (5, 25),

$$m_P + m_R > m,$$

where m_P and m_R are defined as the highest degree plus one of the polynomials that are interpolated exactly by the prolongation and the restriction respectively, and m is the order of the partial differential equation to be solved. Here, injection operator is considered as the prolongation and its adjoint is chosen as the restriction, and then, m_P and m_R are both equal to one. Since the order of the equation is two, the strict inequality is not satisfied. Thus, it is not possible to guarantee efficient convergence rates due to the weakness of the coarse-grid correction. Anyway, from the practical point of view, we will see that, in combination with strong smoothers, we will obtain very efficient multigrid methods.

3.1.1 Discretization on semi-structured grids of a diffusion problem with discontinuous coefficients

We deal with the solution of the following diffusion equation:

$$-\nabla \cdot (\kappa(x,y)\nabla v) = f, \text{ in } \Omega, \tag{3.1}$$

$$v = g, \text{ on } \Gamma, \tag{3.2}$$

where Ω is an open bounded domain, Γ is its boundary, f is a source term, g gives the Dirichlet boundary condition and $\kappa(x,y)$ is the diffusion coefficient, that may be discontinuous. In particular, here we are interested in problems where κ is piecewise constant.

As done in Chapter 2, here we consider here a cell-centred finite volume discretization of problem (3.1)-(3.2) on a semi-structured grid. Firstly, we construct an acute Delaunay triangulation $\mathcal{T}$ on the domain Ω, as the coarsest unstructured grid. We assume this triangulation to be fairly coarse, resolving the large-scale features of the domain, that is, this triangulation is taken fitting the geometry of the domain and the discontinuities of the diffusion coefficient in the way that on each triangle the value of κ remains constant. Secondly, we apply a regular refinement process to the triangles of $\mathcal{T}$, that is, every triangle is divided into four congruent ones by connecting the midpoints of their edges, and so on until getting the desired fine scale to approximate the solution of the problem.

Now, we are going to obtain the equation on the Voronoi point $\mathbf{x}_c$ corresponding to a triangle T of the triangulation $\mathcal{T}$, see Figure 2.2. Note that T is connected with three different triangles at most, T_1, T_2 and T_3. We denote by κ_c the diffusion coefficient on the present triangle T, and by κ_i those coefficients on T_i. Using the divergence theorem on equation (3.1) at triangle T, we obtain

$$-\int_{\partial T} (\kappa(x,y)\nabla v) \cdot \mathbf{n}\, dl = \int_T f(\mathbf{x})\, d\mathbf{x},$$

where $\mathbf{n}$ is the unit outward normal vector to ∂T and $\partial T = l_1 \cup l_2 \cup l_3$. Then, by splitting the integral on the left-hand side into the sum of line integrals corresponding to edges

l_1, l_2 and l_3, and by approximating them as the length of the corresponding edge, l_i, multiplied by the flux evaluated in the midpoint of the edge, we obtain the left-hand side of the equation. Regarding the right-hand side integral, it is replaced by the area of the triangle multiplied by the value of f at the Voronoi point. Finally, the equation reads:

$$-\frac{1}{meas(T)}\sum_{i=1}^{3}\left(\kappa_i^H meas(l_i)\,\frac{v_h(\mathbf{x}_i)-v_h(\mathbf{x}_c)}{d_i}\right) = f(\mathbf{x}_c), \tag{3.3}$$

where $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ are the Voronoi points of the triangles adjacent to T with common edges l_1, l_2 and l_3, respectively, and d_i the distance between points $\mathbf{x}_c$ and $\mathbf{x}_i$, with $i = 1, 2, 3$, (see Figure 2.2). The coefficients κ_i^H appearing in (3.3) are the harmonic average between the corresponding diffusion coefficients, given by:

$$\kappa_i^H = \frac{2\kappa_c\kappa_i}{\kappa_c + \kappa_i}, \tag{3.4}$$

which is the most accurate method of known techniques of averaging (46, 56).

Next, we are interested in obtaining the equations for the Voronoi points on the structured patches of the semi-structured grid. These equations follow from (3.3) by properly choosing the coefficients and using a double indexing for the notation of Voronoi points, more appropriate when structured grids are dealt with.

Following the notation introduced in Section 2.1.2, given an arbitrary pair (k_1, k_2) associated with a node of G_h, the equations corresponding to the two Voronoi points $\mathbf{x}^u_{k_1,k_2}$ and $\mathbf{x}^v_{k_1,k_2}$ are given by

$$L_h^{uu} v_h^u(\mathbf{x}^u_{k_1,k_2}) + L_h^{ud} v_h^d(\mathbf{x}^d_{k_1,k_2}) = f_h^u(\mathbf{x}^u_{k_1,k_2}), \tag{3.5}$$

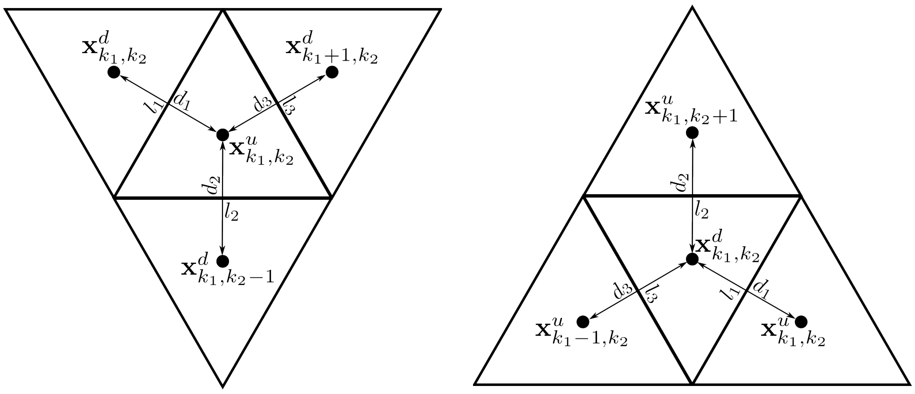
$$L_h^{du} v_h^u(\mathbf{x}^u_{k_1,k_2}) + L_h^{dd} v_h^d(\mathbf{x}^d_{k_1,k_2}) = f_h^d(\mathbf{x}^d_{k_1,k_2}), \tag{3.6}$$

where operators L_h^{uu}, L_h^{ud}, L_h^{du} and L_h^{dd} can be obtained from equation (3.3), and are given in stencil form as:

$$L_h^{uu} = \frac{1}{meas(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sum_{i=1}^{3} \kappa_i^H \frac{l_i}{d_i} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad L_h^{ud} = \frac{1}{meas(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\kappa_1^H \frac{l_1}{d_1} & -\kappa_3^H \frac{l_3}{d_3} \\ 0 & -\kappa_2^H \frac{l_2}{d_2} & 0 \end{bmatrix},$$

$$L_h^{du} = \frac{1}{meas(T)} \begin{bmatrix} 0 & -\kappa_2^H \frac{l_2}{d_2} & 0 \\ -\kappa_3^H \frac{l_3}{d_3} & -\kappa_1^H \frac{l_1}{d_1} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad L_h^{dd} = \frac{1}{meas(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sum_{i=1}^{3} \kappa_i^H \frac{l_i}{d_i} & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where the distances d_1, d_2, d_3 and the lengths l_1, l_2, l_3 are defined depending on the orientation of the triangle, as seen in Figure 3.1. For example, for an up-oriented triangle d_2 is defined as the distance between $\mathbf{x}^u_{k_1,k_2}$ and $\mathbf{x}^d_{k_1,k_2-1}$, and l_2 as the length of the edge between those Voronoi points.



(a) (b)

Figure 3.1: Notation used to construct the stencil on a Voronoi point at (a) an up-oriented triangle or at (b) a down-oriented triangle.

3.1.2 Block-wise Multigrid

For this type of problems, the same methodology considered in the previous chapter will be used. That is, a block-wise multigrid in which different smoothers are chosen for the triangles of the coarsest grid, is considered. Due to the fact that a different diffusion coefficient does not affect the anisotropies of the grid, we will follow the guideline depicted in 2.15 for the choice of the smoothers. Regarding the coarse-grid correction, as in the previous chapter, simple and local inter-grid transfer operators have been chosen in order to reduce to the bare minimum the communication between the coarsest triangles. In particular, the natural injection is selected as the prolongation operator $\mathbf{I}_{k-1}^{k}$; in fact, this is the only possible operator that is local for cell-centred discretizations, and the restriction is chosen as the adjoint of the prolongation. In this way, only the four fine-grid points surrounding a coarse-grid point are related to it by both inter-grid transfer operators. We would like to remark that the choice of these operators preserves four-point stencils when Galerkin approach is applied, which is very convenient for the simplicity of the connections between the triangular blocks of the grid.

When large jumps in the diffusion coefficient κ occur in the domain, a direct discretization on coarse grids may not work properly (49, 56). However, in most of the multigrid methods proposed for discontinuous coefficient problems, the Galerkin approach has provided satisfactory results. This means that the coarse-grid operator is defined in terms of the fine-grid operator, $\mathbf{L}_k$, the restriction, $\mathbf{I}_k^{k-1}$ and the prolongation, $\mathbf{I}_{k-1}^{k}$, in the following way:

$$\mathbf{L}_{k-1} = \mathbf{I}_k^{k-1}\mathbf{L}_k\mathbf{I}_{k-1}^{k}. \tag{3.7}$$

However, when simple injection is selected for the inter-grid transfer, the Galerkin operator (3.7) results to be inconsistent with the differential operator, and it must be replaced by the following expression:

$$\mathbf{L}_{k-1} = \frac{1}{2}\mathbf{I}_k^{k-1}\mathbf{L}_k\mathbf{I}_{k-1}^{k}. \tag{3.8}$$

This modified Galerkin approach provides better multigrid convergence rates than the original Galerkin operator (3.7), see (32, 40, 59).

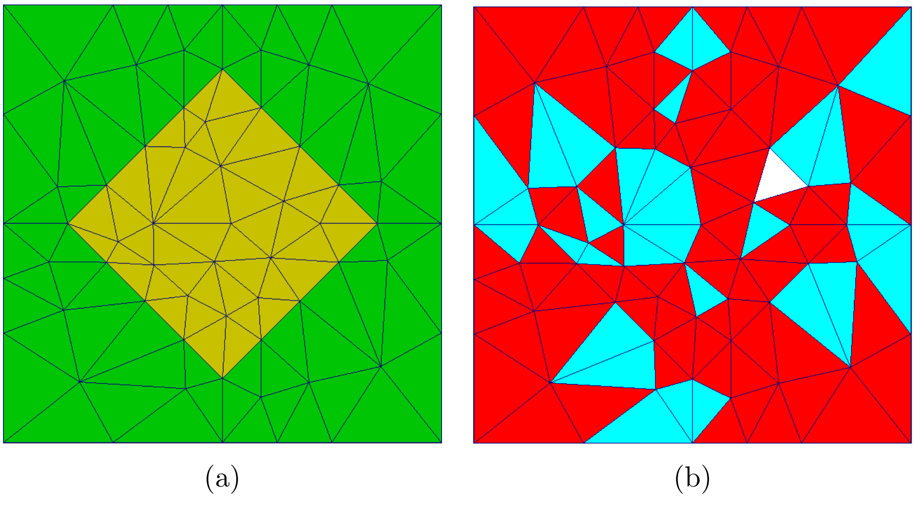
3.1.3 Numerical experiments

In this section, we are going to present two numerical experiments to demonstrate the efficiency of the proposed block-wise multigrid algorithm based on the modified Galerkin approach.

In the first experiment, problem (3.1) is solved on the unit square with two different distributions of diffusion coefficients. In the second one, the same problem is solved on a more complex domain. In the first numerical experiment, the proposed multigrid algorithm is applied by using an F-cycle with two pre- and two post-smoothing steps, whereas in the second one the influence of the type of cycle and the number of pre- and post-smoothing steps on the performance of the multigrid algorithm will be investigated. The computer used for the timing results in the subsequent experiments is an AMD at x2 at 2.9 GHz.

3.1.3.1 Diffusion problem on the unit square with discontinuous coefficients

In the first experiment, we solve problem (3.1)-(3.2) on the unit square with different diffusion coefficients defined piecewise on two subdomains of different shapes, as shown in Figures 3.2(a) and 3.3(a). It consists of a benchmark problem taken from the literature (56). More concretely, in the first test case, the inner subdomain has a rhombus shape with a 0.5 side length, see Figure 3.2(a); whereas the second one consists of an hexadecagon with diameter 0.5 simulating a circle, see Figure 3.3(a). The right-hand side is defined as $f(x, y) = xy$, the Dirichlet boundary conditions are given by $g(x, y) = x^2 + y^2$, and the diffusion coefficient values are $\kappa = 0.333 \times 10^5$ for the internal subdomains and $\kappa = 2$ for the rest of the domain, see Figures 3.2(a) and 3.3(a). In the same figures the corresponding coarsest grids are also represented.


(a)

(b)

Figure 3.2: (a) Coarsest unstructured mesh for the first test case, and distribution of diffusion coefficients: $\kappa = 0.333 \times 10^5$ at the yellow region and $\kappa = 2$ at the green part. (b) Different smoothers for the triangles of the coarsest grid: white corresponds to red-black smoother, diamond smoother is represented by red, and wormy smoother by blue.

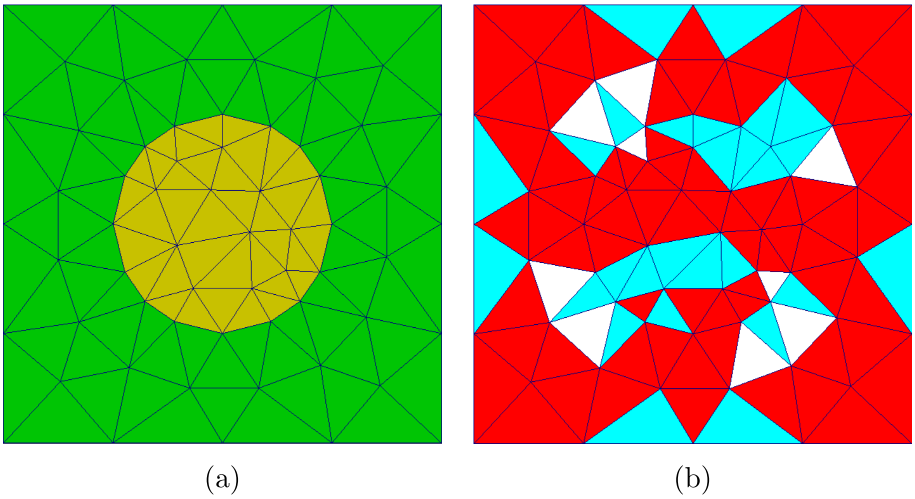


Figure 3.3: (a) Coarsest unstructured mesh for the second test case, and distribution of diffusion coefficients: $\kappa = 0.333 \times 10^5$ at the yellow region and $\kappa = 2$ at the green part. (b) Different smoothers for the triangles of the coarsest grid: white corresponds to red-black smoother, diamond smoother is represented by red, and wormy smoother by blue.

The proposed block-wise multigrid method has been applied to solve both test cases. Red-black, wormy and diamond smoothers have been used for different triangles of the coarsest grid, as shown in Figures 3.2(b) and 3.3(b). Regarding the obtained multigrid convergence, in Table 3.1, the number of iterations necessary to reduce the initial residual in a factor of 10^{-10} are shown for both test cases. We observe an h-independent convergence for both problems, and although these results are slightly worse than those obtained in the case of constant diffusion coefficients. As expected, the method shows a very satisfactory convergence. On the other hand, when direct discretization is used on coarse grids, a very poor convergence rate is obtained.

3.1.3.2 Diffusion problem on a composite material

In the second experiment, problem (3.1)-(3.2) is solved on a rectangular domain composed of two different materials with different diffusion coefficients: $\kappa = 1$ and $\kappa = 0.001$, as we can see in Figure 3.4(a). The considered coarsest grid is shown in the same figure, and also we can observe that it is composed of triangles with very disparate shapes. For this reason, different smoothers are considered for the different triangles of the coarsest triangulation. In particular the smoothers chosen for these

Levels	Unknowns	Rhombus	Circle
4	6912	8	8
5	27648	8	9
6	110592	8	9
7	442368	8	9
8	1769472	9	9

Table 3.1: Number of iterations to reduce the initial residual in a factor of 10^{-10} for both test cases: the inner rhombus subdomain (Figure 3.2) and the inner circle subdomain (Figure 3.3).

triangles are shown in Figure 3.4(b). In this way, the proposed block-wise multigrid is used for solving this problem.

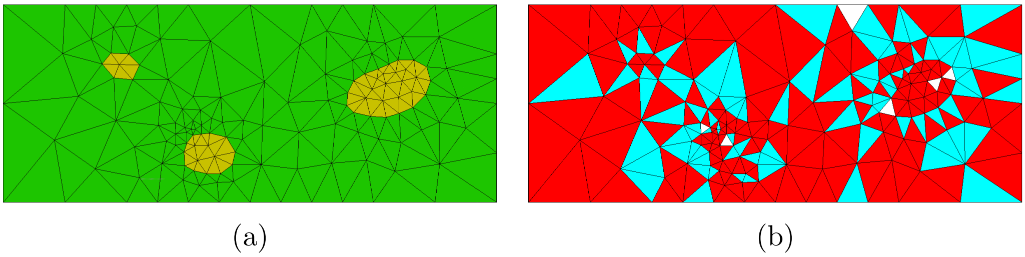


Figure 3.4: (a) Coarsest unstructured mesh and distribution of diffusion coefficients for the second experiment. Yellow color represents $\kappa = 0.001$ and green $\kappa = 1$. (b) Different smoothers for the triangles of the coarsest grid: white corresponds to red-black, diamond smoother is represented by red and wormy smoother by blue.

Firstly, we want to compare the behaviour of the multigrid algorithm by considering both, direct discretization on coarse grids and the Galerkin approach. For this purpose, in Figure 3.5, the history of the convergence of the method by using an F(2,2)-cycle, for different numbers of refinement levels, is displayed. The stopping criterion is chosen as the final maximum residual to be less than 10^{-7}. We can see that the method based on direct discretization leads to divergence, while that based on Galerkin approach yields very satisfactory and robust results. Moreover, we observe that the convergence is independent of the discretization parameter, and with only twelve/thirteen cycles the residual reaches the desired value. Note, that in this experiment the use of Galerkin coarse-grid operator becomes mandatory.

Next, we want to investigate the performance of the block-wise multigrid method

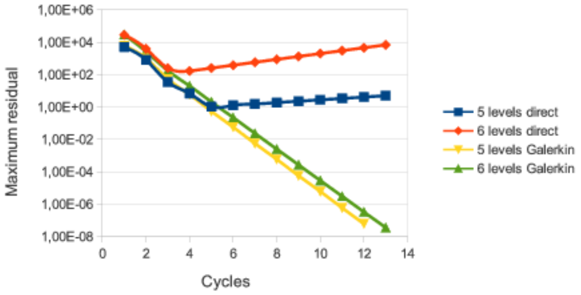


Figure 3.5: Comparison between direct discretization and Galerkin approach for different numbers of refinement levels.

proposed here, depending on the type of cycle and the number of smoothing steps. Varying these parameters, the number of iterations necessary to reduce the initial residual in a factor of 10^{-10} and the CPU time, together with the asymptotic convergence factor are shown in Table 3.2, for different number of refinement levels. In the case of an F-cycle, it is observed that the convergence is independent of the space discretization parameter, and that in few iterations the residual is reduced as desired. Regarding the behaviour of the V-cycle, it is clearly observed a very quickly deterioration in the asymptotic convergence factor with respect to the size of the mesh, even achieving divergence in some cases. This degradation of the convergence in the case of a V-cycle may be due to the lack of accuracy of the chosen inter-grid transfer operators. Notice that the price of obtaining an h-independent convergence rate is the use of an F-cycle. On the other hand, very simple inter-grid operators are used facilitating the communication between two connected patches, as well as strong and cheap smoothers are carefully chosen depending on the shape of the triangle.

3.2 Dominant convection

The combination of diffusion and convection processes is extensively used. For instance, we can find many examples related to flow in porous media, including oil recovery, tracking of contaminants in groundwater flows and nuclear waste storage. Many schemes are known to fail if the convection is dominant, yielding solutions which are physically meaningless. The main difficulty comes from the interplay of the convection, a large scale effect, and diffusion, that comes from the Brownian motion which is a small scale

		5 levels		6 levels		7 levels		8 levels	
		it (ρ)	CPU	it (ρ)	CPU	it (ρ)	CPU	it (ρ)	CPU
2	V	31(0.71)	17.29	131(0.95)	167.39	-	-	-	-
	F	16(0.30)	19.59	16(0.30)	39.51	16(0.32)	96.99	17(0.32)	306.64
3	V	13(0.58)	10.03	21(0.78)	36.90	-	-	-	-
	F	11(0.17)	17.98	11(0.18)	36.77	11(0.19)	90.11	11(0.18)	277.02
4	V	9(0.48)	8.88	16(0.65)	35.70	35(0.83)	217.04	-	-
	F	8(0.15)	16.36	9(0.14)	37.99	9(0.15)	92.85	9(0.15)	273.17

Table 3.2: Number of iterations to reduce the initial residual in a factor of 10^{-10}, corresponding asymptotic convergence rates (between brackets) and CPU-time for different numbers of refinement levels, by using a V-cycle and an F-cycle, and different numbers of smoothing steps $\nu = 2, 3, 4$. The symbol $-$ indicates divergence of the method.

effect. Among all the numerical methods to stabilize the convection-diffusion problems we will use the simplest one, which is the upwind discretization.

An overview of multigrid methods for dominant convection problems can be seen in (19, 49), where the authors point out some inherent problems that multigrid has when the convection is dominant, and also how to overcome those difficulties.

3.2.1 Discretization on unstructured triangular grids

In the numerical experiment presented in Section 2.4.2, we presented a convection-diffusion model, discretized by a central difference scheme, which was solved for a dominant diffusion situation. Nevertheless, it is more common to face a physical phenomena where the convection is dominant over the diffusion. In this case, the discretization used in Section 2.4.2 requires the use of very fine grids to obtain a physically acceptable solution, making its resolution prohibitive (49). On the other hand, the central difference scheme is a second order discretization, which is a desirable property for a scheme. However, in practice, this scheme is not always possible to use. When the convection is dominant, the discretization must be coherent with the direction of the flow. Using central differences, we consider all the directions to have the same importance, which is not true when there is a velocity field. As an alternative, we use an upwind discretization for the convective term, that despite having only first order, it will make the method stable no matter how big is the convective term.

The equation for the convection-diffusion model written in divergence form is:

$$\nabla \cdot (\nabla v - \mathbf{b}\, v) = f, \text{ in } \Omega, \tag{3.9}$$

where $\mathbf{b}(\mathbf{x})$ is a given velocity field, whose divergence is assumed to be zero.

We will use the same finite volume strategy used in the previous chapter. By using the divergence theorem on a triangle T of the triangulation $\mathcal{T}$, the following balance equation holds:

$$\int_{l_1} (\nabla v - \mathbf{b} v) \cdot \mathbf{n}_1 \, \mathrm{d}l_1 + \int_{l_2} (\nabla v - \mathbf{b} v) \cdot \mathbf{n}_2 \, \mathrm{d}l_2 + \int_{l_3} (\nabla v - \mathbf{b} v) \cdot \mathbf{n}_3 \, \mathrm{d}l_3 = \int_T f(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{3.10}$$

where $\mathbf{n}_i$ is the unit outward normal vector to the corresponding edge l_i of triangle T. Each of these line integrals can be split in two parts, the diffusion one, and the convective term. The diffusion part can be approximated as we did in Section 2.1.1. Regarding the convective part, we have the following line integral:

$$-\int_{l_i} \mathbf{b} v \cdot \mathbf{n}_i \, \mathrm{d}l_i, \tag{3.11}$$

which can be approximated by the length of the edge multiplied by an approximation of $\mathbf{b}v \cdot \mathbf{n}_i$ at the midpoint of the edge i. In this case, to approximate the value of v, an average between the values stored in the two triangles that share that common edge can be considered. This approach yields a second order approximation. However, it will not work when convection is dominant. In order to perform an upwind discretization, we have to approximate v by using only the value from one of those two triangles, following the direction of the flow. We can define a general formula for the upwind discretization:

$$-\,\overline{b}_i(\,\xi\, v_h(\mathbf{x}_c) + (1-\xi)\, v_h(\mathbf{x}_i))\,\mathrm{meas}(l_i), \; \left\{ \begin{array}{l} \xi = 1, \text{ if } \overline{b}_i > 0 \\ \xi = 0, \text{ if } \overline{b}_i \leq 0 \end{array} \right. , \tag{3.12}$$

where $\overline{b}_i$ is the evaluation of $\mathbf{b}\cdot\mathbf{n}_i$ at the midpoint of edge i, $\mathbf{x}_c$ is the node of the triangle in which we are currently discretizing, and $\mathbf{x}_i$ is the node of the neighbouring triangle touching the corresponding edge, and ξ is a parameter which is one or zero depending on the direction of the velocity. If we assemble the whole equation, we obtain:

$$\begin{gathered} \frac{1}{\mathrm{meas}(T)} \sum_{i=1}^{3} \left[\mathrm{meas}(l_i) \left(\frac{v_h(\mathbf{x}_i) - v_h(\mathbf{x}_c)}{d_i} - \overline{b}_i(\,\xi\, v_h(\mathbf{x}_c) + (1-\xi)\, v_h(\mathbf{x}_i)) \right) \right] = f(\mathbf{x}_c), \\ \text{with } \xi \text{ defined as: } \left\{ \begin{array}{l} \xi = 1, \text{ if } \overline{b}_i > 0 \\ \xi = 0, \text{ if } \overline{b}_i \leq 0 \end{array} \right. , \end{gathered} \tag{3.13}$$

where meas(T) is the area of triangle T, meas(l_i) the length of the edge i and d_i the Voronoi distance between the corresponding Voronoi points $\mathbf{x}_c$ and $\mathbf{x}_i$. This formula has second order for the diffusion term and the right-hand side and first order for the convective part, and then the resulting scheme has first order accuracy.

3.2.2 Discretization on structured triangular grids

In this subsection our intention is to obtain a stencil form of equation (3.13) for structured triangular grids. Starting with equation (3.13) and following the notation and the explanation carried out in Section 2.1.2, we can obtain the stencils for a structured triangular mesh as follows:

$$L_h^{uu} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\sum_{i=1}^{3} \text{meas}(l_i)(\frac{1}{d_i} + \bar{b}_i \xi_i) & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$L_h^{ud} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \text{meas}(l_1)\left(\frac{1}{d_1} - \bar{b}_1(1-\xi_1)\right) & \text{meas}(l_3)\left(\frac{1}{d_3} - \bar{b}_3(1-\xi_3)\right) \\ 0 & \text{meas}(l_2)\left(\frac{1}{d_2} - \bar{b}_2(1-\xi_2)\right) & 0 \end{bmatrix},$$

$$L_h^{du} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & \text{meas}(l_2)\left(\frac{1}{d_2} - \bar{b}_2(1-\xi_2)\right) & 0 \\ \text{meas}(l_3)\left(\frac{1}{d_3} - \bar{b}_3(1-\xi_3)\right) & \text{meas}(l_1)\left(\frac{1}{d_1} - \bar{b}_1(1-\xi_1)\right) & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$L_h^{dd} = \frac{1}{\text{meas}(T)} \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sum_{i=1}^{3} \text{meas}(l_i)(\frac{1}{d_i} + \bar{b}_i \xi_i) & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

In these formulae the values of ξ_1, ξ_2 and ξ_3 follow the rule defined in equation (3.12).

Regarding the anisotropy related to the convective part of the equation it is important to remark that, as we are using an upwind discretization, the effect of the anisotropy does not only depend on the mesh itself, but also on the direction of the flux and the orientation of the mesh.

3.2.3 Multigrid results on structured grids

We are going to present some numerical experiments on structured triangular grids by using multigrid methods. The inter-grid transfer operators and the smoothers are the same that those presented in Section 2.2. However, in this situation, we should expect different results depending on the velocity field. In order to analyse the convergence rate, we will use an initial random guess, and a right-hand side equal to zero in order to avoid round-off errors. Also, in order to study better the effects of the convection, we will add a parameter ϵ multiplying the diffusion term, so we can reduce its dominance in the experiment:

$$\nabla \cdot (\epsilon \nabla v - \mathbf{b}\, v) = f, \text{ in } \Omega. \tag{3.14}$$

In order to show the importance of the direction of the flow and the updating ordering, we performed some experiments for the same geometry with different velocity fields. The corresponding results are shown in Table 3.3. The most remarkable convergence factors are those that have a reduction of seven orders of magnitude in each step. This effect is due to the fact that the updating ordering is coherent with the direction of the flow. When the convection is very dominant, the problem is almost hyperbolic, and if the updating process follows the direction of the flow, the smoother is almost an exact solver. As an illustrative example, we can see that the red wormy smoother has the best and worst convergence factors for the velocity vectors $\mathbf{b} = (0, 1)$ and $\mathbf{b} = (0, -1)$ respectively. This is because the considering updating ordering goes in the direction $(0, 1)$.

Regarding the red-black smoother, it has a very bad convergence rate in all the cases. On the other hand, the convergence rate of diamond smoother depends on the relation between the updating ordering and the direction of the flow.

Please note that the theory says that the worst multigrid convergence rate should be around 0.5 due to the bad approximation of the convective term on coarse meshes (49). However, that theoretical prediction does not take into account the boundaries, which are crucial in these cases. Then, these predictions can be very pesimistic, since the effect of boundary conditions can improve the convergence rate.

	$\mathbf{b} = (0, 1)$	$\mathbf{b} = (0, -1)$	$\mathbf{b} = (1, 1)$	$\mathbf{b} = (-1, -1)$
Green diamond	0.15	0.28	0.29	0.21
Red diamond	0.47	1.2×10^{-6}	0.28	$7,2 \times 10^{-2}$
Black diamond	0.15	0.28	8.4×10^{-7}	0.45
Green wormy	0.16	4.3×10^{-7}	2.7×10^{-7}	8.3×10^{-2}
Red wormy	5.3×10^{-7}	0.27	2.6×10^{-7}	0.21
Black wormy	0.16	4.3×10^{-7}	0.28	4.8×10^{-7}
Red-Black	0.29	0.53	0.50	0.30

Table 3.3: Convergence factors for different smoothers using a V(1,1)-cycle with different flow directions.

3.2.4 Multigrid results on semi-structured grids

Finally, to finish this section we are going to show the behaviour of multigrid on semi-structured grids. We are going to consider again the same domain as in the experiment in Section 2.4.2, so we can compare only the effect of dominant convection, avoiding the influence of the domain. Therefore, an unstructured grid of 96 triangles is considered, see Figure 2.25. Since the effect of the convective part depends on the direction of the flow and the geometry of the triangles, we decided to follow the strategy that only depends on the geometry and not on random information. Hence, we will use the strategy proposed in Figure 2.15 to choose the smoothers. Regarding the velocity field considered in the experiment, in order to make the experiment more realistic, we impose a circulatory flow $\mathbf{b}$ defined by:

$$\mathbf{b} = (-\sin(\pi x)\cos(\pi y), \sin(\pi y)\cos(\pi x)). \tag{3.15}$$

Thus, the streamlines of the flow are depicted in Figure 3.6. In Table 3.4, we can see the asymptotic convergence rate, the number of cycles to reduce the initial residual in ten orders of magnitude, and the CPU time used to achieve the desired convergence by using an AMD athlon X2 at 2.9 GHz. For this experiment, we have used the smoothers depicted in Figure 2.25 and $\epsilon = 10^{-3}$. The results for the F-cycle are very similar to those obtained for the case of non-dominant convection by using central differences. Regarding the computational cost, it is higher in this case due to the

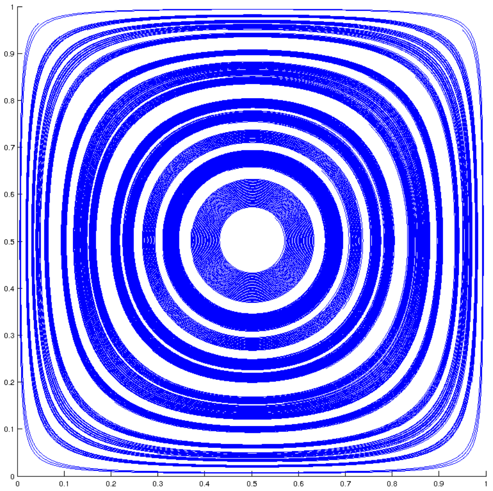


Figure 3.6: Streamlines of the circulatory flow in the square domain.

dynamic calculation of the stencil that we have to perform to correctly apply the upwind discretization.

Focusing on the V-cycle, the results are much worse. Not only the value of the asymptotic convergence rate is worse, but also it shows a dependence on the number of nodes. Hence, a V-cycle does not seem recommendable.

		V(2,2)			F(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)	Cycles	ρ_h	Time (s)
4	24576	14	0.11	1.86	14	0.12	3.97
5	98304	16	0.23	5.19	14	0.12	8.7
6	393216	23	0.36	21.2	13	0.12	19.5
7	1572864	34	0.53	102.8	11	0.12	48.7
8	6291456	48	0.65	549.7	10	0.12	222.4

Table 3.4: Number of iterations to reduce the initial residual in a factor of 10^{-10}, corresponding asymptotic convergence rates and CPU-times for different numbers of refinement levels, using a V-cycle and an F-cycle.

However, we see in Table 3.3 that the wormy smoothers handle well the dominant convection. Therefore we have repeated the experiment by only using wormy smoothers.

Remember that for any triangle there is always a wormy smoother that can be used to obtain an optimal convergence rate. Thus, in Table 3.5 results for a V-cycle but using only wormy smoothers are displayed. In this case, the necessary number of iterations is reduced, and also the CPU time and the convergence rate.

In order to see the importance of the influence of the dominant convection on the obtained results, we perform an experiment by considering $\epsilon = 10^{-5}$ and by using F-cycle. Seven refinement levels are considered and both strategies, only wormy smoothers and the usual smoothing strategy with different smoothers, are tested. We obtain that by using only wormy smoothers the asymptotic convergence rate is about 0.16, whereas, the usual strategy leads to divergence. Hence, the use of wormy smoothers is recommended for dominant convection problems.

		V(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)
4	24576	13	0.10	1.8
5	98304	17	0.20	6.0
6	393216	19	0.27	20.0
7	1572864	24	0.37	86.8
8	6291456	29	0.43	404.9

Table 3.5: Number of iterations to reduce the initial residual in a factor of 10^{-10}, corresponding asymptotic convergence rates and CPU-times for different numbers of refinement levels, and by using only wormy smoothers.

3.3 Time dependent convection-diffusion problem

3.3.1 Discretization on unstructured triangular grids

In this section, we consider the time dependent case of the model presented in Section 3.2, that is:

$$\frac{\partial v}{\partial t} - \nabla \cdot (\epsilon \nabla v - \mathbf{b}\, v) = f, \text{ in } \Omega, \tag{3.16}$$

where $\mathbf{b}(\mathbf{x})$ is a given velocity field, whose divergence is assumed to be zero.

Regarding the diffusion and convective terms, the discretization is exactly the same that we did previously in their respective sections. Now, we have to decide how to approach the time dependent term. To this purpose we can consider $\theta-$methods, with θ equal to zero (explicit Euler), 1/2 (Crank-Nicolson) or one (implicit Euler). The advantages of using explicit Euler are that it is easy to implement and its application is faster, since it only depends on old values. On the other hand, the use of explicit methods is restricted to very small time steps that makes their use almost prohibitive. Therefore, we are not interested in this method. Using an implicit method (Crank-Nicolson or implicit Euler) means that a system of equations must be solved in each time step, which is very time consuming. Nonetheless, in (49) it is stated that the use of implicit methods increases the value of the diagonal of the system, making easier its resolution. Therefore, we have two possibilities, the use of Crank-Nicolson which has second order, or implicit Euler, that has only first order. Nevertheless, by using Crank-Nicolson, there is a restriction for the time step that assures that the method possesses the max-min property (23). Also, second order methods are known to be more unstable than first order methods, and as we will see, the Darcy-Oberbeck-Boussinesq model will be very sensitive to instabilities. Therefore, we will use the implicit Euler method. Now, we discretize our equation obtaining:

$$\frac{v_h^{n+1}(\mathbf{x}_c)-v_h^n(\mathbf{x}_c)}{\tau} - \frac{\epsilon}{\text{meas}(T)}\sum_{i=1}^{3}\text{meas}(l_i)\left(\frac{v_h^{n+1}(\mathbf{x}_i)-v_h^{n+1}(\mathbf{x}_c)}{d_i}\right)$$
$$+\frac{1}{\text{meas}(T)}\sum_{i=1}^{3}\text{meas}(l_i)\left(\bar{b}_i(\,\xi\, v_h^{n+1}(\mathbf{x}_c)+(1-\xi)\, v_h^{n+1}(\mathbf{x}_i))\right) = \; f^{n+1}(\mathbf{x}_c), \left\{\begin{array}{l}\xi=1,\ \text{if}\,\bar{b}_i>0\\ \xi=0,\ \text{if}\,\bar{b}_i\leq 0\end{array}\right., \tag{3.17}$$

where τ is the time-discretization step, and $v_h^n(\mathbf{x}_i)$ denotes the approximation at time $t_n=n\tau$ of v at the point $\mathbf{x}_i$.

As we have used implicit Euler, we have a first order discretization in time, and note that our scheme is of first order in space because although the diffusion term is discretized by a second order scheme, the convective term is discretized by an upwind scheme which is a first order discretization.

3.3.2 Multigrid results on structured grids

In order to see the influence of the time-dependent term on the multigrid performance, we repeat the experiment done in Section 2.2.3 for an almost-right triangular grid and a triangular grid characterized by a small angle. For this experiment, we consider

$\mathbf{b} = \mathbf{0}$ and $\epsilon = 1$. We use a random initial guess and a zero right hand side to avoid round-off errors. Regarding the rest of the multigrid components, we have used the same inter-grid transfer operators presented in Section 2.2.1, direct discretization on coarse grids and two pre- and two post-smoothing steps. Since we want to compare the efficiency of the method, we will perform the experiments with only one time step. The time discretization parameter is fixed as $\tau = 0.01$. The spatial grid is composed of 16384 triangles. In Table 3.6, we can see the results for an almost-right triangular grid and for a triangular grid characterized by a small angle. In all the cases the results are better than those obtained in Section 2.2.3. Despite the fact that the improvement is small, it is enough to realise that the convergence rate now also depends on the time discretization step. More concretely, it depends on the relation between τ and the space discretization parameter. From this and considering the fact that the time discretization step can only improve the performance of the multigrid method, we can conclude that the strategy presented in Figure 2.15 to choose suitable smoothers is also valid for the time dependent convection-diffusion equation, since the steady problem would be the worst case.

	Right-Triangle		Sharp triangle	
Smoother	F(2,2)	V(2,2)	F(2,2)	V(2,2)
Red-Black	0.55	–	0.93	0.94
w-Red-Black	0.43	–	0.91	0.91
Diamond	3.6×10^{-2}	3.7×10^{-2}	0.87	0.88
Wormy	6.3×10^{-2}	6.6×10^{-2}	2.7×10^{-2}	2.7×10^{-2}

Table 3.6: Asymptotic convergence rates for an almost right triangular grid characterized by angles (45, 85), and for an isosceles triangular grid characterized by angles (85,85), by using four different smoothers. The time discretization is $\tau = 0.01$ and a grid with 16384 nodes is considered.

Now, we are going to consider an equilateral triangular grid. The same experiment as presented in Table 3.3 is performed for the time dependent convection-diffusion equation, with $\tau = 0.01$. The corresponding results are shown in Table 3.7, and one can see that in all the cases the results have improved.

	$\mathbf{b} = (0, 1)$	$\mathbf{b} = (0, -1)$	$\mathbf{b} = (1, 1)$	$\mathbf{b} = (-1, -1)$
Green diamond	4.3×10^{-2}	0.10	0.13	9.5×10^{-2}
Red diamond	0.19	5.3×10^{-8}	0.12	$2,7 \times 10^{-2}$
Black diamond	4.2×10^{-2}	0.10	4.6×10^{-8}	0.22
Green wormy	4.2×10^{-2}	1.3×10^{-8}	1.2×10^{-8}	2.9×10^{-2}
Red wormy	1.7×10^{-8}	0.10	1.2×10^{-8}	$9.3\ 10^{-2}$
Black wormy	4.2×10^{-2}	1.3×10^{-8}	0.13	2.0×10^{-8}
Red-Black	0.10	0.20	0.24	0.13

Table 3.7: Convergence factors for different smoothers using a V(1,1)-cycle with different flow directions and $\tau = 0.01$.

3.3.3 Multigrid results on semi-structured grids

Finally we are going to show the suitability of the proposed multigrid when semi-structured triangular grids are considered. To this purpose, we solve equation (3.16) in a polygonal domain with three isolated holes inside. We have considered the circulatory flow given in equation (3.15). For these experiments we will fix $\epsilon = 1$. The coarsest mesh is formed by 108 triangles with different geometries, and it is shown in Figure 3.7. The choice of the smoothers has been done by using the guide shown in Figure 2.15, and they are displayed in the same figure. In order to test the effects of the time dependent term, we will consider two different time discretization steps. In Table 3.8 we show the results for $\tau = 0.01$ and in Table 3.9 for $\tau = 0.00001$. We present the number of iterations to reduce the initial residual in ten orders of magnitude, the necessary CPU-time by using an AMD X2 at 2.9 GHz and the asymptotic convergence rates. When $\tau = 0.01$, F-cycle presents an h-independent convergence rate, while that of V-cycle deteriorates as we increase the number of nodes. Despite this effect, the application of V-cycle seems to be more interesting as it requires less computational cost. Regarding the case in which $\tau = 0.00001$, we observe that the convergence rate for V- and F-cycles is pretty similar. Thus, the use of the F-cycle seems completely unnecessary for small time steps.

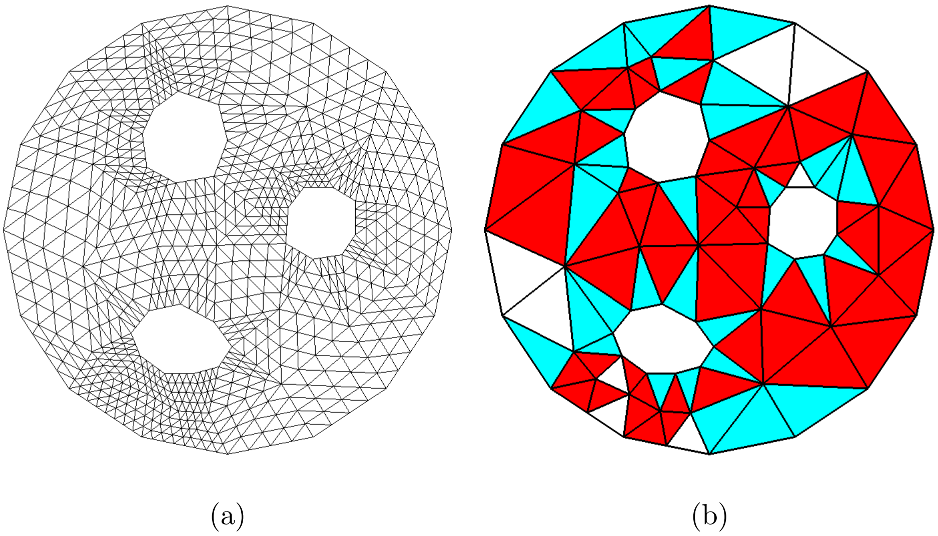


Figure 3.7: (a) Grid after two refinement levels (b) Different smoothers for the triangles of the coarsest grid: white corresponds to red-black, diamond smoother is represented by red and wormy smoother by blue.

		V(2,2)			F(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)	Cycles	ρ_h	Time (s)
4	27648	10	0.12	1.7	9	0.10	1.7
5	110592	10	0.13	3.8	9	0.10	4.5
6	442368	11	0.15	11.8	9	0.11	13.7
7	1769472	11	0.17	39.5	9	0.12	44.7
8	7077888	11	0.18	190.8	9	0.12	221.4

Table 3.8: Number of iterations to reduce the initial residual in a factor of 10^{-10}, corresponding asymptotic convergence rates and CPU-times for different numbers of refinement levels, by using V-cycle and F-cycle, where $\tau = 0.01$.

		V(2,2)			F(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)	Cycles	ρ_h	Time (s)
4	27648	4	6.3×10^{-4}	0.6	4	6.3×10^{-4}	0.8
5	110592	6	1.3×10^{-2}	2.3	5	1.3×10^{-2}	3.1
6	442368	8	5.3×10^{-2}	8.4	8	5.3×10^{-2}	12.1
7	1769472	9	0.11	32.2	9	0.10	45.2
8	7077888	10	0.12	161.2	9	0.11	207.6

Table 3.9: Number of iterations to reduce the initial residual in a factor of 10^{-10}, corresponding asymptotic convergence rates and CPU-times for different numbers of refinement levels, by using V-cycle and F-cycle, where $\tau = 0.00001$.

Chapter 4

Multigrid methods on staggered grids

Until this chapter we have only considered scalar equations. Nevertheless, this is very limited, since in many areas the use of scalar fields is insufficient. Consider, for example, computational fluid dynamics where the velocity is represented by vectors. Moreover, we may be more interested in the interaction between the velocity of a fluid and other variables like the pressure, or the concentration of a contaminant.

Therefore, in this chapter we are going to deal with the use of vectors and scalar values together. Besides, we will also have to deal with the anisotropy created by the meshes. For that, we will adapt the multigrid components that we have developed for scalar equations to work with staggered grids.

4.1 Discretization on unstructured triangular grids

In this section, a discretization for the following system of equations is presented:

$$\begin{array}{l} \mathbf{u} + \triangledown P = \mathbf{f} \\ \triangledown \cdot \mathbf{u} = 0 \end{array} \quad , \ in\ \Omega, \tag{4.1}$$

$$\mathbf{u} \cdot \mathbf{n} = 0,\ on\ \partial\Omega, \tag{4.2}$$

where $\mathbf{u}$ is the velocity field, and P a scalar value denoting the pressure. We will approximate the normal component of the velocity at the midpoint of the edges, see Figure 4.1, while the scalar variables will be approximated at the Voronoi points. For the normal components of the velocity, we will use the same notation as in Section 3.2. Regarding the properties of the triangulation, we assume the same restrictions established in Section 2.1.1.

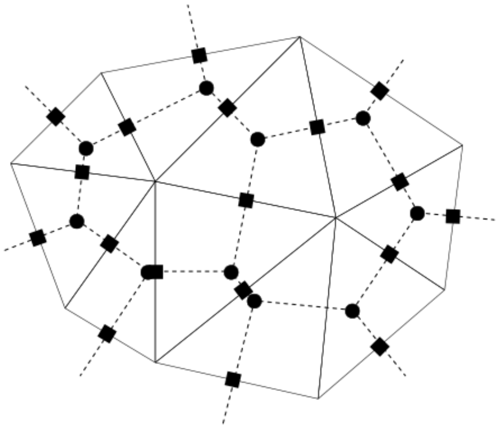


Figure 4.1: Unstructured mesh and its associated Voronoi grid with dots for the scalar values, and squares denoting the places where the projection of the vector values are stored.

For the continuity equation, we will use the discretization technique called VAGO, presented in (50). In this way, the resulting discretization for any triangle T reads as:

$$\frac{1}{\text{meas}(T)} \sum_{i=1}^{3} (\text{meas}(l_i)\, u_i) = 0, \tag{4.3}$$

where u_i is the numerical approximation of the outward normal component of the velocity on edge i. Once we have discretized the scalar equation, now we deal with Darcy's law, whose discretization at the midpoint of edge i of triangle T is:

$$u_i + \frac{P(\mathbf{x}_i) - P(\mathbf{x}_c)}{d_i} = f_i. \tag{4.4}$$

Here, u_i is the normal projection of the velocity on edge i, $P(\mathbf{x}_c)$ is the pressure at the Voronoi point of triangle T, $P(\mathbf{x}_i)$ the pressure corresponding to the Voronoi point $\mathbf{x}_i$, and f_i is the normal projection of the right hand side evaluated in the middle point of the edge i.

As seen in (50),the resulting scheme is a first order discretization.

4.2 Multigrid method

4.2.1 Coarse-grid correction

In this subsection we are going to deal with the prolongation and restriction for discretizations based on edges. For cell-centred nodes we have used the same transfer operators introduced in Section 2.2.1. In order to define the restriction and the prolongation for discretizations based on edges, we are going to distinguish three different type of nodes, one for each edge. Then the restriction and prolongation for each type of node only involves grid-points of its same type.

4.2.1.1 Restriction

We considered a six-point restriction operator with weights of $\frac{1}{4}$ or $\frac{1}{12}$ depending on the relative location of the nodes, see Figure 4.2 (a). For example, for the grid-points located at edges of type L1:

$$\begin{aligned} r_c(\mathbf{x}_{ic,jc}) = (3r_f(\mathbf{x}_{i,j}) + 3r_f(\mathbf{x}_{i,j+1}) + 3r_f(\mathbf{x}_{i+1,j+1}) + r_f(\mathbf{x}_{i,j-1}) \\ +r_f(\mathbf{x}_{i+1,j}) + r_f(\mathbf{x}_{i+2,j+1}))/12, \end{aligned} \tag{4.5}$$

where r_c and r_f are the residual grid functions defined on the coarse and fine grid respectively, and the coarse indices i_c and j_c are related to those on the fine grid by the following expressions: $i = 2i_c - 1$ and $j = 2j_c - 1$.

4.2.1.2 Prolongation

Regarding the prolongation, given a fine-grid node belonging to an up-oriented coarse triangle, we apply a simple injection from the coarse-grid point located at the same coarse triangle. In other case, we apply an average of the three closest coarse-grid points, see Figure 4.2 (b).

4.2.2 Smoothers

Since the considered system is a saddle point problem, it is well known that standard smoothers do not provide good convergence, and then more sophisticated smoothers have to be considered. More concretely, Vanka type smoothers (51), as well as distributive smoothers (58), are common choices for these type of problems. Next, we develop novel Vanka type smoothers adapted to staggered discretizations on structured

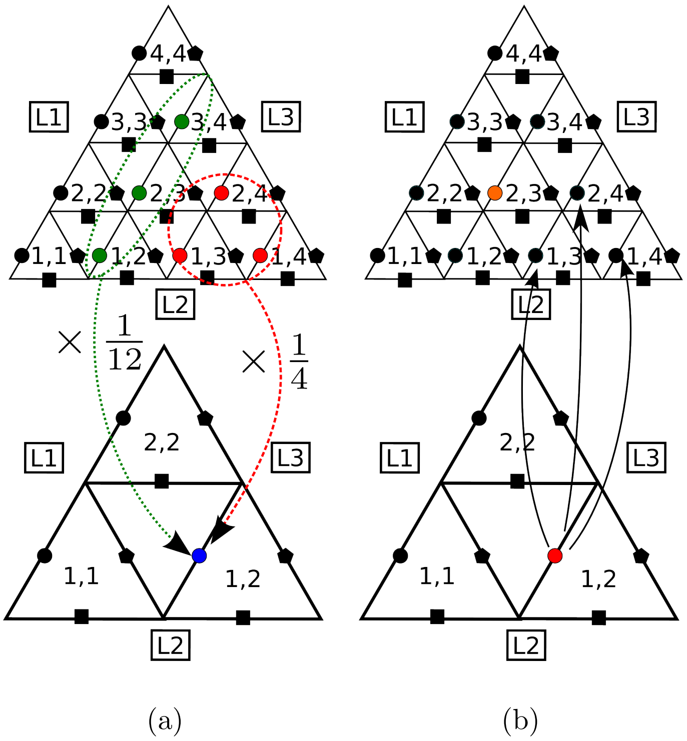


Figure 4.2: (a) Restriction for a L1 type node (circles), where the red nodes have a weight of $\frac{1}{4}$ and the green ones of $\frac{1}{12}$. (b) Prolongation for a L1 type node, where some nodes are updated by injection, whereas others, like the orange node, need to be calculated from the three coarse grid surrounding nodes.

triangular grids. Due to the properties of the discrete operators considered here, we can also design a suitable distributive smoother.

4.2.2.1 Vanka type smoothers

In general, Vanka type smoothers consist of simultaneously updating a set of unknowns. Of course, these blocks can be updated in different orderings. As a generalization of the smoothers introduced in Chapter 2, next we are going to present three different Vanka type smoothers.

Vanka red-black smoother: For this smoother a block of four unknowns belonging to the same triangle is updated. In particular, the three velocity unknowns located at the edges, denoted here as v_1, v_2, v_3, and the scalar unknown associated with the Voronoi point, represented by P, are simultaneously updated.

Following the idea of a red-black ordering, we first update all the blocks corresponding to up-oriented triangles, and secondly all the blocks associated with down-oriented triangles. Notice, that the unknowns located at the edges are relaxed twice per smoothing step, see Figure 4.3.

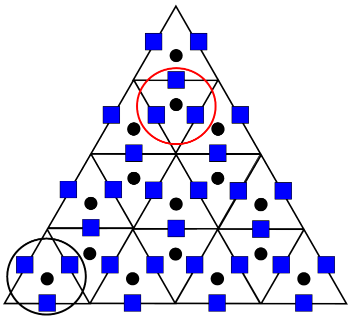


Figure 4.3: Vanka red-black smoother for staggered triangular grids.

The application of this smoother requires us to solve for each triangle the following small 4 × 4 system of equations:

$$\begin{bmatrix} 1 & 0 & 0 & \frac{1}{d_{v_1}} \\ 0 & 1 & 0 & \frac{1}{d_{v_2}} \\ 0 & 0 & 1 & \frac{1}{d_{v_3}} \\ \frac{l_1}{\text{meas}(T)} & \frac{l_2}{\text{meas}(T)} & \frac{l_3}{\text{meas}(T)} & 0 \end{bmatrix} \begin{bmatrix} \delta v_1 \\ \delta v_2 \\ \delta v_3 \\ \delta P \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_s \end{bmatrix}, \qquad (4.6)$$

where r_i is the residual on the node corresponding to the edge i and r_s the residual corresponding to the scalar unknown.

Vanka diamond smoother: In a Vanka diamond smoother we have to update simultaneously all the unknowns located at two neighbouring triangles. In this case, we have to solve a 7 × 7 system of equations for each block:

$$\begin{bmatrix} 1 & 0 & 0 & \frac{1}{d_{v_1}} & 0 & 0 & 0 \\ 0 & 1 & 0 & \frac{1}{d_{v_2}} & 0 & 0 & \frac{-1}{d_{v_2}} \\ 0 & 0 & 1 & \frac{1}{d_{v_3}} & 0 & 0 & 0 \\ \frac{l_1}{\text{meas}(T)} & \frac{l_2}{\text{meas}(T)} & \frac{l_3}{\text{meas}(T)} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \frac{1}{d_{v_5}} \\ 0 & 0 & 0 & 0 & 0 & 1 & \frac{1}{d_{v_4}} \\ 0 & -\frac{l_2}{\text{meas}(T)} & 0 & 0 & \frac{l_5}{\text{meas}(T)} & \frac{l_4}{\text{meas}(T)} & 0 \end{bmatrix} \begin{bmatrix} \delta v_1 \\ \delta v_2 \\ \delta v_3 \\ \delta P_1 \\ \delta v_4 \\ \delta v_5 \\ \delta P_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_{s1} \\ r_4 \\ r_5 \\ r_{s2} \end{bmatrix}. \tag{4.7}$$

Vanka wormy smoother: In this case, we have to update simultaneously the unknowns located at a row of triangles. This means that, whereas a tridiagonal system had to be solved in the scalar case, now a block tridiagonal system must be solved, which means that the computational cost will be very high.

4.2.2.2 Distributive smoother

In this kind of smoother, the idea is to decouple the system of equations by introducing a right preconditioner in the smoothing procedure, in order to apply fast smoothers to the decoupled system, see Figure 4.4. More concretely it consists of two steps:

- Predictor step: After decoupling the system, we smooth the new variables by smoothing the scalar and vector unknowns separately.
- Corrector step: The final step consists in updating the original variables by using the values obtained in the predictor step.

Next, we are going to describe the concrete distributive smoother for our system:

$$\begin{pmatrix} I_h & G_h \\ D_h & 0 \end{pmatrix} \begin{pmatrix} U_h \\ P_h \end{pmatrix} = \begin{pmatrix} f_h \\ 0 \end{pmatrix}. \tag{4.8}$$

By introducing the new variables

$$\begin{pmatrix} U_h \\ P_h \end{pmatrix} = \begin{pmatrix} I_h & G_h \\ 0 & -I_h \end{pmatrix} \begin{pmatrix} U'_h \\ P'_h \end{pmatrix}, \tag{4.9}$$

the new decoupled system yields:

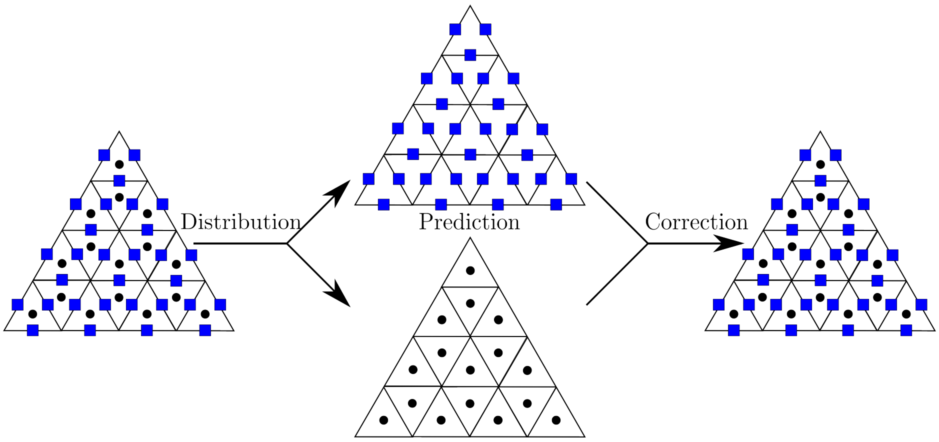


Figure 4.4: Process of a distributive smoother.

$$\begin{pmatrix} I_h & 0 \\ D_h & \triangle_h \end{pmatrix} \begin{pmatrix} U_h' \\ P_h' \end{pmatrix} = \begin{pmatrix} f_h \\ 0 \end{pmatrix}. \tag{4.10}$$

In the new system (4.10), we obtain the discrete laplacian operator because the discretization scheme obtained by the VAGO method is consistent with the properties of the differential operator (50) and then $D_h G_h = \triangle_h$. Since the obtained system is decoupled, we can use the appropiate smoothers for the Laplace discrete operator extensively studied in Chapter 2. Regarding the vector unknowns, the smoothing process is very simple since only an identity operator appears in the equation. Note that the distributive smoother has an inconvenient: there are no boundary conditions for the new variables. However, this problem can be overcome if after the distributive smoother we perform a wormy smoothing step at the boundaries.

4.3 Results of the proposed multigrid method on structured grids

In this section, we are going to see how well the smoothers perform for different structured triangular grids. Again, we will test the three characteristic triangles that we have been using throughout the thesis. In all the cases, we will also test the distributive smoother, in which we use the smoother recommended by the guide shown in Figure 2.15 to relax the transformed system. However, since the computational cost

	Time (s)
Vanka red-black	2.9
Vanka diamond	3.3
Vanka wormy	15.2
Distributive	1.0

Table 4.1: CPU time to perform 100 iterations of the relaxation scheme by using a grid composed of 65536 triangles.

will affect our final decision for the choice of the smoother, we first compare the efficiency of the smoothers. The test consists of performing 100 iterations of the relaxation scheme by using a grid composed of 65536 triangles. For these calculations, we have used an AMD athlon X2 at 2.9 GHz. The results can be seen in Table 4.1. Now the difference between the computational cost of the smoothers is higher than in the scalar case. For example, the Vanka wormy smoother is five times more expensive than the Vanka red-black smoother. The distributive smoother has a very good performance, is about three times faster than the Vanka red-black smoother. In the distributive smoother we have used a point-wise red-black smoother to relax the scalar equation of the decoupled system. Of course, the behaviour of the distributive smoother depends on the choice of the smoother used for the scalar equation of the transformed system. For example, if we use a wormy smoother the required time is about twice more expensive. Without taking into account the convergence factor, it seems that the distributive smoother would be the best choice, if it would be able to deal with the anisotropies of the grid.

Once that we have compared the computational cost of all the smoothers, we are going to study their behaviour depending on the geometry of the grid. In all these experiments, we have used an F-cycle with two pre- and two post-smoothing steps, a random initial guess and a zero right-hand side. We have used 8 refinement levels, which corresponds to a grid with 65536 triangles. The chosen stopping criterion consists of reducing the maximum residual until 10^{-8}. When the distributive smoother is considered, we have included the Vanka wormy extra-relaxation in the boundaries.

First, we compare the behaviour of the four considered smoothers for an equilateral triangular grid. In Figure 4.5, we show the maximum residual versus the number of cycles. All the smoothers provide a convergence factor lower than 0.10, and therefore, all of them provide very satisfactory results. Taking into account the computational cost of the smoothers, the recommended smoother is the distributive relaxation, since its slightly slower convergence rate with that of the Vanka wormy smoother is compensated by far by the fact that the distributive smoother is much faster.

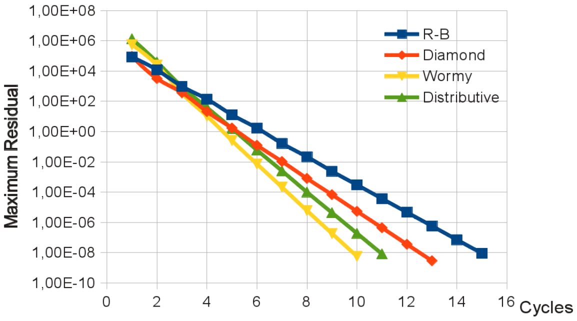


Figure 4.5: History of the convergence for different smoothers in an equilateral triangular grid composed of 65536 triangles.

We have just seen that the distributive smoother is the best choice when there is no anisotropy. Now we are going to see what happens when almost right triangular grids are considered. In Figure 4.6, we can see the performance of three smoothers in an almost right-triangular grid characterized by the angles, $\alpha = 45^o$ and $\beta = 85^o$. The Vanka red-black smoother diverges in this case, and therefore it is not considered. Again all the smoothers show a convergence factor lower than 0.10. The Vanka diamond is the best smoother, whereas the Vanka wormy smoother and the distributive relaxation have a similar behaviour. Since, the distributive smoother can handle this anisotropy, we also recommend for this case the use of the distributive smoother.

The final case is to test the behaviour of the smoothers on a very sharp triangular domain characterized by the angles, $\alpha = 85^o$ and $\beta = 85^o$. In this case, all the smoothers except the Vanka wormy smoother diverge. Please note that depending on the strategy used in the boundaries the distributive smoother would be able to provide a better convergence rate. The recommended smoother is then the Vanka wormy relaxation, which shows a behaviour very similar to that presented in the scalar case. In Figure 4.7, the history of convergence of such smoother for different numbers of refinement levels is displayed. As it can be seen, an independent convergence with respect to the space

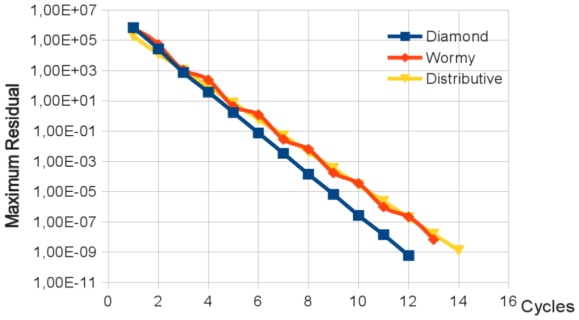


Figure 4.6: History of the convergence for different smoothers in an almost right-triangular grid composed of 65536 triangles.

discretization paramter is obtained.

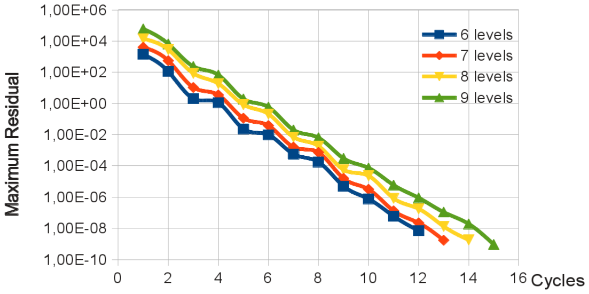


Figure 4.7: History of the convergence for the wormy smoother with different number of refinement levels.

However, due to the high computational cost of the Vanka wormy smoother we prefer to use it only when it is strictly necessary. Hence, being the distributive smoother the cheapest one, we want to analyse which degree of anisotropy it is able to handle. For that, we perform some tests for different isosceles triangular grids. The results are plotted in Figure 4.8, where we can see that in the case of $\alpha = \beta = 70^o$ the smoother has a satisfactory convergence, whereas the sharper the triangular domain is, the worse the convergence rate becomes. To see if the distributive smoother has an h-independent convergence rate, we perform a test for a triangular grid characterized by angles, $\alpha = \beta = 80^o$, for different number of refinement levels. Figure 4.9, shows the independence of the convergence rate with respect to the number of refinement levels.

Finally, we show in Figure 4.10 a new strategy to choose suitable smoothers for

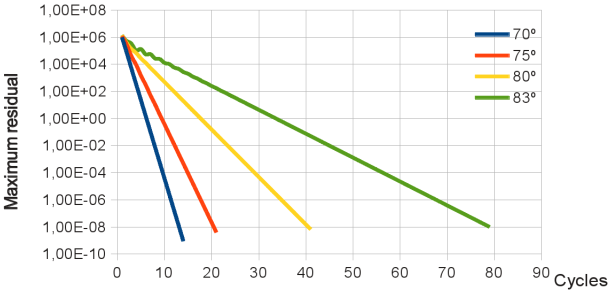


Figure 4.8: History of the convergence for the distributive smoother for different isosceles triangular grids.

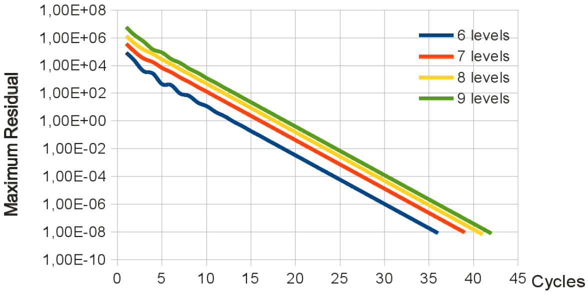


Figure 4.9: History of the convergence for the distributive smoother for different numbers of refinement levels.

different staggered triangular grids characterized by two angles, α and β to achieve a convergence factor about 0.10.

4.4 A numerical experiment on semi-structured grids

We are going to design a block-wise multigrid on semi-structured staggered triangular grids to solve the problem (4.1). We consider the domain Ω as shown in Figure 4.11, in which, we can also observe the initial coarsest grid. If we use the guideline in Figure 4.10 to choose the suitable smoother for each triangle of the coarsest grid, the result would be that we would have to apply a distributive smoother on all the triangles. In this case we have two options. The first one would be to apply the distributive smoother locally on each triangle of the coarsest grid, whereas, in the second option we could apply the distributive smoother globally on the whole domain. In this latter, on each smoothing step we have to relax a scalar equation on the complete domain following

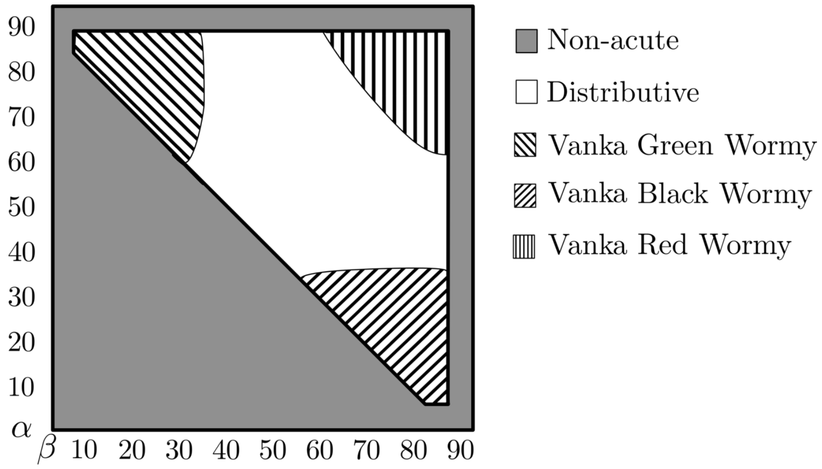


Figure 4.10: Guideline to choose suitable smoothers to reach an asymptotic convergence factor about 0.1 for different triangular grids characterized by two angles, α and β.

the methodology explained in Chapter 2. Notice that different smoothers will be used depending on the shape of the coarsest triangles. We have observed that this second alternative provides better results, and therefore, we will choose this strategy to perform the numerical experiments.

In this experiment, we have used an AMD athlon X2 at 2.9 GHz. The right-hand side is zero and the initial guess is random. The stopping criterion is to reduce the initial residual by a factor of 10^{-10}. In Table 4.2, for different numbers of refinement levels we show the number of cycles necessary to reach the convergence, the corresponding asymptotic convergence factors and the CPU time in seconds. Two different types of cycles are considered in this experiment V- and F-cycles and two pre- and two post-smoothing steps are fixed. As we can observe, the proposed multigrid method based on distributive smoothers results in a very fast solver because only about ten cycles are necessary to reach the desired convergence. Besides, an h-independent convergence rate is achieved. Comparing the performance of V- and F-cycles, we see that the choice of V-cycle is preferable.

As Vanka smoothers are widely used for these type of saddle point problems, it is worth to analyse a block-wise multigrid strategy only based on Vanka type smoothers. To this end, we will use the appropriated Vanka smoothers for each triangle of the coarsest triangulation. Figure 4.11 shows the smoothers considered for this experiment.

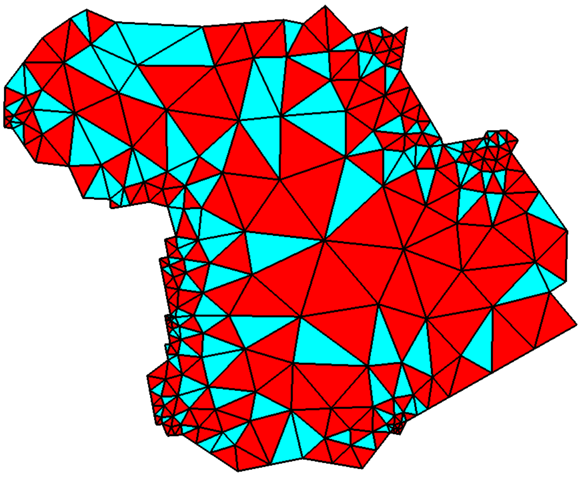


Figure 4.11: Different smoothers for the triangles of the coarsest grid: diamond smoother is represented by red, and wormy smoother by blue.

In Table 4.3 results similar to those presented in the previous experiment are shown. Again very good results are obtained in this case, which makes this strategy also suitable in practice.

Finally, as we expected from the study performed on the previous section, the strategy based only on distributive relaxation is prefered due to its lower computational cost. Therefore, this methodology will be used in next chapter for solving the most real problem considered in this thesis.

		V(2,2)			F(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)	Cycles	ρ_h	Time (s)
3	24192	10	0.12	18.3	9	8.9×10^{-2}	25.6
4	96768	10	0.13	45.2	9	8.9×10^{-2}	61.8
5	387072	11	0.12	148.6	10	9.0×10^{-2}	174.3
6	1548288	11	0.12	491.7	10	9.8×10^{-2}	528.4
7	6193152	11	0.12	1710.4	11	9.7×10^{-2}	1879.1

Table 4.2: Number of cycles, asymptotic convergence factors and CPU times for different type of cycles and different numbers of refinement levels, by using the global distributive smoother.

		V(2,2)			F(2,2)		
Levels	Unknowns	Cycles	ρ_h	Time (s)	Cycles	ρ_h	Time (s)
3	24192	14	0.25	25.7	11	0.16	31.6
4	96768	14	0.23	80.6	12	0.16	96.8
5	387072	14	0.23	207.5	12	0.16	203.5
6	1548288	14	0.23	718.3	12	0.14	727.7
7	6193152	14	0.22	2399.7	13	0.16	2512.2

Table 4.3: Number of cycles, asymptotic convergence factors and CPU times for different type of cycles and different numbers of refinement levels, by using the smoothers depicted in Figure 4.11.

Chapter 5

Multigrid method on semi-structured grids for the resolution of the Darcy-Oberbeck-Boussinesq model

5.1 Density driven flows

Density driven flows are flows carried out by differences in the density of the fluid (27). These flows may appear in all the situations in which there is a variation of density or when there are two immiscible fluids with a different density. Also, if there is only one phase, density variations across the fluid may be produced by a gradient of temperatures or a concentration of a solute. These types of flows appear frequently in porous media, where the speed of the stream is characterized by Darcy's law. More concretely, when underwater bears are used to store hot water in summer to be pumped out in winter. The introduced water, that is in steady state, has a different temperature and therefore, a gradient of temperatures will appear creating a slow flow driven by the different densities. Another example of density driven flow in porous media appears in the storage of a chemical waste underground. The presence of the chemical material in the underwater bear creates a flow due to the variation of its concentration, which may lead to a leakage of the residual. Clean coal technologies are another example of density driven flows in porous media. This technology consists in introducing the CO_2 produced by the combustion in underwater saline aquifers.

Nevertheless, not in all the cases where there is a variation of the density a density driven flow models the behaviour of the process. To consider a density driven flow, the distribution of the density must be coupled with the velocity of the flow. As the density is directly related with the temperature or with the concentration of a solute, then we consider this variable the variable of interest.

5.1.1 The Darcy-Oberbeck-Boussinesq model

In order to model density driven flows, we are going to work out the Darcy-Oberbeck-Boussinesq model, DOB from now on, by using the basic equations to model a fluid. We start with Navier-Stokes equations:

$$\begin{aligned} \rho\frac{D\mathbf{v}}{Dt} &= -\nabla P + \mu\nabla^2\mathbf{v} + \rho g\mathbf{z}, && (5.1)\\ \frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) &= 0, && (5.2) \end{aligned}$$

where ρ is the density of the fluid, P is the pressure, g the gravity force, μ the viscosity, $\mathbf{v}$ the velocity field, $\mathbf{z}$ is the vector $(0, -1)$ in Cartesian coordinates, and $\frac{D}{Dt}$ denotes the material derivative. Now, we consider the usual assumptions to achieve the Stokes equation, incompressibility, creeping flow and also stationary. Then, the material derivative of the left side of equation (5.1) and the derivative of ρ are equal to zero, and then we obtain the Stokes equations:

$$\begin{aligned} -\mu\nabla^2\mathbf{v} + \nabla P &= \rho g\mathbf{z}, && (5.3)\\ \nabla\cdot\mathbf{v} &= 0. && (5.4) \end{aligned}$$

We can consider the fluid to have an almost constant volume, which means that assuming the divergence of the velocity to be zero, condition (5.4), is acceptable even though we are going to consider variations of density later. In (8), it is shown that the variation of the volume is very small for the majority of the fluids, below 10^{-3}.

Regarding the viscosity term, if we consider an isotropic porous media, an approximation of this expression can be done by considering a linear response of the viscous term:

$$\mu\nabla^2\mathbf{v} = -\mu\frac{\phi\mathbf{v}}{\kappa}, \tag{5.5}$$

in which ϕ is the porosity, and κ the permeability.

In order to obtain the "Darcy velocity", we multiply the porosity by the velocity field:

$$\mathbf{u} = \phi \mathbf{v}. \tag{5.6}$$

Introducing equations (5.5) and (5.6) into equation (5.3), we obtain the Darcy's law:

$$\mathbf{u} = -\frac{\kappa}{\mu}(\nabla P - \rho g \mathbf{z}). \tag{5.7}$$

Darcy's law was formulated by Henry Darcy after some experiments about the flow of water through sand, and it was published in (10).

The change of the density of a fluid can be produced by a variation of the temperature, pressure or concentration of a solution. We will consider only the variation of concentration as our intention is to model the sequestration of CO_2. Therefore, the next equation to consider is the advection equation:

$$\frac{DC}{Dt} = D\nabla^2 C. \tag{5.8}$$

In this equation, D is the diffusion coefficient relative to the concentration C. Next, we proceed to expand the material derivative:

$$\frac{\partial C}{\partial t} + \mathbf{v} \cdot \nabla C = D\nabla^2 C. \tag{5.9}$$

Substituting $\mathbf{v}$ in equation (5.9), by using equation (5.6), we obtain the desired equation:

$$\phi \frac{\partial C}{\partial t} = -\mathbf{u} \cdot \nabla C + \phi D \nabla^2 C. \tag{5.10}$$

Despite the fact that we have considered an incompressible fluid, we must take into account the variations of density in the buoyancy term carried out by the variation of concentration of a solute. Then, the variation of density follows the equation:

$$\rho = \rho_0 + \delta\rho C, \tag{5.11}$$

where the density ρ is a linear function of concentration in which ρ_0 is the density of the lighter fluid and $\delta\rho$ is the difference between the density of the two fluids.

5.2 The Darcy-Oberbeck-Boussinesq model in CO_2 storage

We are going to focus on the use of the DOB model in the simulation of storage of CO_2 in underwater saline aquifers. Hence, the variation of the density of the flow will be due to the dissolution of a solute, CO_2, in the water.

Firstly, we want to explain the clean coal technology, so we can understand its interest. Next, we will adapt the DOB model to this particular case.

5.2.1 The CO_2 sequestration process

The carbon capture and storage is a process that consists of obtaining the CO_2 from a source, usually from an important source like a power plant. After that the CO_2 is compressed, transported and finally stored.

5.2.1.1 Carbon capture

There are three capturing processes: Post-combustion, pre-combustion and oxy-fuel. The first, post-combustion, is the simplest one. This method burns the fuel in the old-fashion way. After the combustion, there is a process to capture the CO_2 from the flue gases. This method is very interesting because it is an external procedure to the burning process, which means that the old power stations can be upgraded with this technology to turn them into clean power stations. Pre-combustion consists in capturing the CO_2 before the fuel is burned. In this process, initially the fuel is partially oxidized, turning the fuel into carbon monoxide, hydrogen and water. The separation before the combustion yields cleaner gases to deal with, and also the gases can be compressed, making this last process simpler. However, this method cannot be applied to old power stations. The third option, oxy-fuel combustion, consists in burning the fuel directly with oxygen instead of air. The resulting flue gases only contain CO_2 and water. Therefore, all the gases can be treated after the process, not only a fraction of the total flue gases, as it was the case in the first method.

5.2.1.2 Compression and transport

Once the CO_2 is obtained from the flue gases, it must be compressed to a super-critical state to facilitate its transportation. The compression means that part of the energy obtained in the power station will have to be spent in that process. The amount of necessary energy is about the 40% of the energy produced. Hence, the cost of the electricity will be increased in an important factor. However, it is believed that this percentage can be reduced to 20%.

After the compression of CO_2, it has to be transported. For this part of the process, many existing pipelines, which are already in use for other purposes, can also be used to pump CO_2.

5.2.1.3 Storage mechanisms

The final step of the sequestration process is to store the CO_2. For that there are many possibilities. One possibility is to dissolve CO_2 in the oceans, however, the introduction of CO_2 in the water changes its acidity. We could consider storing it in tanks, however, this would lead to a situation close to the storage of nuclear waste, since these residuals, despite not being so dangerous are long-living as well. Also, there is always the possibility of these tanks to explode. A better option which has been used since many years, is to store CO_2 in the reservoirs where the oil is stored. Introducing CO_2 in the reservoirs started in order to facilitate the extraction of oil. Despite the fact that this method could be considered as an option, the main problem is that not all countries have oil and hence such reservoirs to store the CO_2. Another option is to store CO_2 underground, in other kind of places. Among all the possible places, the most interesting is the use of saline aquifers. Saline aquifers have been already used to store industrial wastes, since they are of no use for humankind and also they are very extended, making easy the transportation. The use of saline aquifers is also very interesting because the use of water to dilute CO_2 means that after the necessary time the CO_2 will turn into a mineral, making the storage permanent. However, before this final storage, the CO_2 have to be saved for several years. In that time, the CO_2 must be stored without leakages to the surface. To understand better the effectiveness of the storage method, we are going to see in more detail the way in which the CO_2 is stored. Firstly, the CO_2 is pumped into the saline aquifer. Here, there are four mechanisms that avoid the CO_2 to return to the surface:

- Structural trapping: The CO_2 may try to scape to the surface, however, it is stored due to the walls of the formation.

- Dissolution trapping: The CO_2 can be dissolved in the water. When this happens, the density of the water is changed making the water rich in CO_2 to go to the bottom.

- Residual trapping: This trapping occurs when pumping the CO_2. The porous rock acts like a sponge, then the supercritical CO_2 displaces the fluid stored in the rock. However, after that initial displacement, the fluid returns, displacing the majority of the CO_2, but leaving behind some residual CO_2 stored in the pores of the rock.

- Mineral trapping: This trapping process is the slowest one, but it is permanent. When the CO_2 is dissolved into water, it creates carbonic acid. After a long time, this carbonic acid reacts with the rocks of the boundaries to form a mineral, finally trapping the CO_2.

5.2.2 Nondimensionalization of the Darcy-Oberbeck-Boussinesq model

We present the nondimensionalization of the Darcy-Oberbeck-Boussinesq model:

$$\begin{aligned}
\mathbf{u} &= -\frac{\kappa}{\mu}(\nabla P - \rho g \mathbf{z}), && (5.12)\\
\phi\frac{\partial C}{\partial t} &= -\mathbf{u}\cdot\nabla C + \phi D \Delta C, && (5.13)\\
\nabla\cdot\mathbf{u} &= 0, && (5.14)\\
\rho &= \rho_0 + \delta\rho C. && (5.15)
\end{aligned}$$

For the nondimensionalization of this model, we are going to follow the process presented in (24). Firstly, we need to define the basic parameters to be used in the nondimensionalization : μ, $\mathbf{u}$, H (the domain width), κ, ϕ and $\delta\rho$.

After selecting the basic parameters, we define the dimensionless parameters:

- $\mathbf{u}^* = \dfrac{\kappa[m^2]\delta\rho[\frac{Kg}{m^3}]g[\frac{m}{s^2}]}{\mu[\frac{Kg}{s\,m}]}$

- $P^* = \dfrac{\mu[\frac{Kg}{s\,m}]\mathbf{u}[\frac{m}{s}]H[m]}{\kappa[m^2]} = \delta\rho g H$

- $t^* = \dfrac{\phi[-]H[m]}{\mathbf{u}[\frac{m}{s}]}$

- $\rho^* = \delta\rho[\frac{Kg}{m^3}]$

Using these parameters we can define new variables:

$$P^n = \frac{P}{P^*},\ \mathbf{u^n} = \frac{\mathbf{u}}{\mathbf{u}^*},\ t^n = \frac{t}{t^*},\ x^n = \frac{x}{H},\ z^n = \frac{z}{H}. \tag{5.16}$$

The final step is to introduce the nondiomensional variables into the original system. We begin with equation (5.12):

$$\mathbf{u}^n \mathbf{u}^* = -\frac{\kappa}{\mu}(\frac{\nabla^n}{H}(P^n \delta\rho g H) - \rho g \mathbf{z}), \tag{5.17}$$

and after some algebraic manipulations we obtain the following equation:

$$\mathbf{u}^n = -\frac{\kappa}{\mu \mathbf{u}^*}\delta\rho g(\frac{\nabla^n}{H}(P^n H) - \frac{\rho_0 + \delta\rho C}{\delta\rho}\mathbf{z}). \tag{5.18}$$

As $\delta\rho g = \dfrac{\mu \mathbf{u}^*}{\kappa}$ we can modify the equation as follows:

$$\mathbf{u}^n = -(\nabla^n P^n - \frac{\rho_0 \mathbf{z}}{\delta\rho} - C\mathbf{z}). \tag{5.19}$$

Finally, if we regroup all the terms affected by the gradient in a new variable $P' = P^n - \dfrac{\rho_0 z^n}{\delta\rho H}$, where we have introduced the variable z^n in the gradient in order to keep the equation the same, we obtain the following dimensionless equation:

$$\mathbf{u}^n = -(\nabla^n P' - C\mathbf{z}). \tag{5.20}$$

Next step is the nondimensionalization of the equation (5.13), by substituting the old variables by the dimensionless ones:

$$\phi\frac{\partial C}{\partial t^n}\frac{\mathbf{u}^*}{\phi H} = -\mathbf{u}^n \mathbf{u}^* \frac{\nabla^n C}{H} + \phi D \frac{\Delta^n}{H^2}C. \tag{5.21}$$

Multiplying the equation by $H/\mathbf{u}^*$ we can remove the coefficients of almost all terms:

$$\frac{\partial C}{\partial t^n} = -\mathbf{u}^n \nabla^n C + \frac{\phi D}{\mathbf{u}^* H}\Delta^n C. \tag{5.22}$$

Equation (5.22) can be simplified to contain a single parameter, since $\frac{\phi D}{\mathbf{u}^* H}$ is equal to the inverse of the Rayleigh parameter. Therefore, the final equation reads:

$$\frac{\partial C}{\partial t^n} = -\mathbf{u}^n \nabla^n C + \frac{1}{Ra}\Delta^n C. \tag{5.23}$$

It is very easy to get the dimensionless equivalent of equation (5.14):

$$\frac{\nabla^n}{H} \cdot \frac{\mathbf{u}^n}{\mathbf{u}^*} = 0 = \nabla^n \cdot \mathbf{u}^n. \tag{5.24}$$

If we finally assemble all the equations, we obtain the desired dimensionless equation system:

$$\begin{aligned} \frac{\partial C}{\partial t^n} &= -\mathbf{u}^n \nabla^n C + \frac{1}{Ra} \Delta^n C, && (5.25) \\ \mathbf{u}^n &= -(\nabla^n P' - C\mathbf{z}), && (5.26) \\ \nabla^n \cdot \mathbf{u}^n &= 0. && (5.27) \end{aligned}$$

5.3 Solver strategy

In this section, we explain the numerical strategy that we have used to solve the DOB model. Regarding the discretization methods used to discretize the equations of the model, we have used the techniques explained in previous chapters.
Regarding the non-linearity in the term $-u^n \nabla^n C$, we use a fixed point method. Thus, we will split the system into two sub-systems:

$$\frac{\partial C}{\partial t^n} = -\mathbf{u}^n_{old} \nabla^n C + \frac{1}{Ra} \Delta^n C, \tag{5.28}$$

and

$$\begin{aligned} \mathbf{u}^n &= -(\nabla^n P' - C_{old}\mathbf{z}) && (5.29) \\ \nabla^n \cdot \mathbf{u}^n &= 0. && (5.30) \end{aligned}$$

The fixed point method follows the next scheme, also represented in Figure 5.1:

- Using the multigrid method explained in Section 3.3, the parabolic equation (5.28) is solved using an old guess of $\mathbf{u}^n$. In this way, a new approximation of C is calculated.

- Next, using the already calculated C we solve the system of equations (5.29) and (5.30) with the multigrid method that we presented in Section 4.

- After obtaining a new velocity, we will repeat the process until the concentration does not change its value more than 1% from one step to the next one.

Note that as we are using the same multigrid methods that we used in previous chapters, their behaviour is same as we already presented.

Regarding the fixed point method, to advance one time step, we have to perform about 100 iterations for the first time step, whereas for the next steps two fixed point iterations are sufficient.

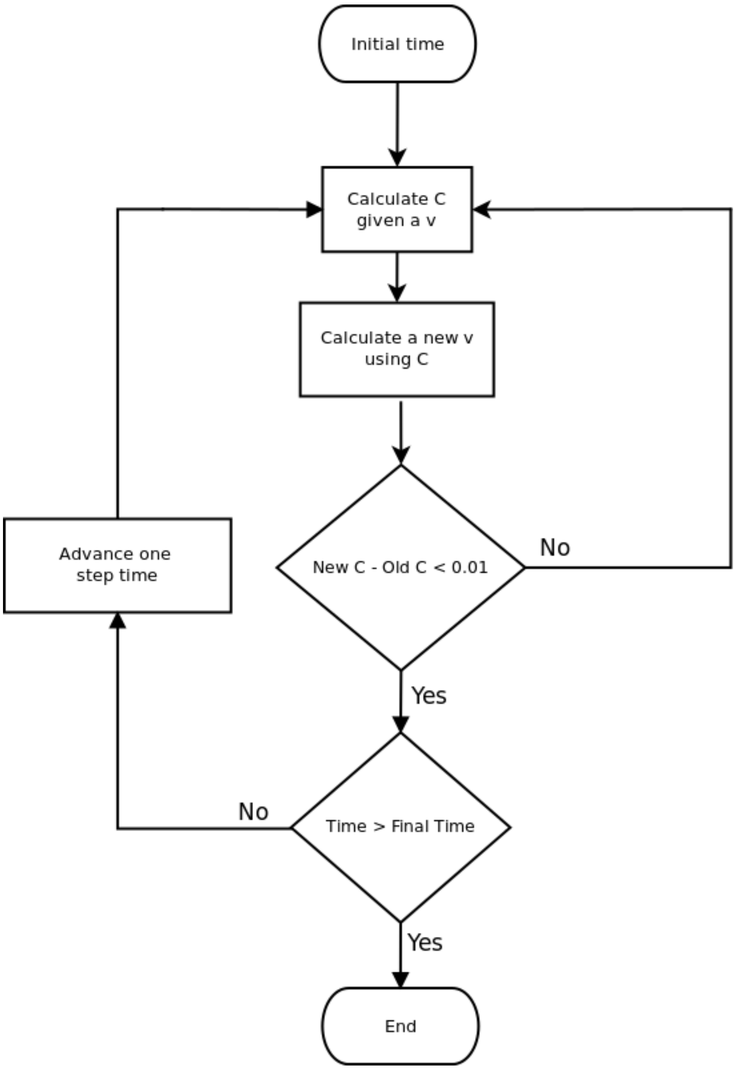


Figure 5.1: Flowchart showing the fixed point method.

5.4 Numerical experiments

In the CO_2 sequestration process, we have two physical phenomena competing. The diffusion and the convection processes. Initially, the diffusion effect rules the evolution of the experiment. However, it rapidly becomes unstable creating some "fingers" that will drive the rest of the process. These "fingers" are columns of fluid with a higher density due to the dissolved CO_2. The higher density makes them to weigh more than water, making them to fall, creating streams that grow by absorbing more CO_2 from the upper boundary or by ingesting other fingers.
This instability created by the fingers means that the numerical experiments cannot represent exactly the reality (24). Since we are going to consider averaged parameters of permeability, viscosity and diffusion, we cannot consider small perturbations of the parameters across the domain. For instance, a higher permeability in one zone of the domain would increase the chances of a finger to start to grow there, making the result different from the one calculated. Nevertheless, although the finger would appear in a different position, the behaviour of the process would be similar. Therefore, the interest is not to perfectly match the reality but to be able to model and understand the general behaviour of the process.

Unless stated otherwise, the initial and boundary conditions are going to be the same for all the experiments. Regarding the initial condition, we will always consider that the experiment starts from a rest state. Concerning the boundary conditions for the concentration, we will consider an upper layer of concentration equal to one, as the representation of the super-critical CO_2 that is intended to be stored. This layer will remain constant during all the process. In (38) it is stated that this approximation is physically and mathematically acceptable. Since our intention is to consider a closed domain, we will impose to the other frontiers mirror Neumann boundary conditions, $\nabla C \cdot \mathbf{n} = 0$. The boundary conditions for the velocity will be Dirichlet boundary conditions, where the normal direction of the velocity $\mathbf{u}$ with regard to the frontier is zero. In the solver, in all the experiments we have used a V-cycle with two pre- and two post-smoothing steps for the resolution of both, the scalar equation and the staggered system of equations. Regarding the discretization parameters, we have considered a time step of 0.001, and 4096 triangles per coarse triangle of the initial unstructured triangulation. Concerning the smoothers used, for the resolution of the Darcy problem we have used the previously introduced distributive smoother in the whole domain, whereas for the resolution of the advection equation the strategy depicted in Figure 2.15 is considered.

5.4.1 Numerical experiments in homogeneous media

In this section, we perform two numerical experiments for different geometries where the physical properties of the media are equal in all the domain.

5.4.1.1 Rectangular domain

In these experiments our domain is a rectangle, whose dimensions are 1 width and 0.5 high. First, we analyse the influence of the mesh on the obtained results. Next, we increase the Rayleigh number in order to analyse the connection between the number of fingers and this parameter. Also, for this latter experiment, we consider the effect of the artificial diffusion induced by the mesh in the solution.

Rayleigh number equal to 1000

Due to the inherent instabilities of the physical problem, the results are very sensitive to small perturbations in the physical conditions and of course also to the errors introduced by the numerical method. In order to show this effect induced by the numerical part, we consider the results obtained by two very unlikely initial unstructured triangulations that create approximation errors of diverse magnitude in different parts of the domain. As we will present, this fact creates solutions that are quantitatively different, but they present a similar qualitative behaviour. This is typical in all the “fingering” processes in saturated and not saturated porous media. In this field, it is normally enough to obtain a comparable number of fingers and a similar velocity (22, 24).

In Figure 5.2, we have depicted the two initial unstructured triangulations for that rectangular domain that we are going to consider. We keep the coherence in the other figures, that is, all the figures (a) are associated with the triangulation showed in Figure 5.2 (a) and the same for figures (b).

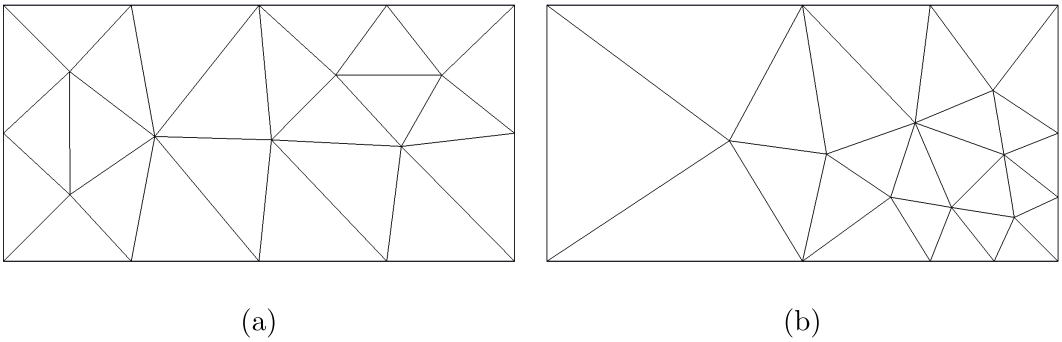
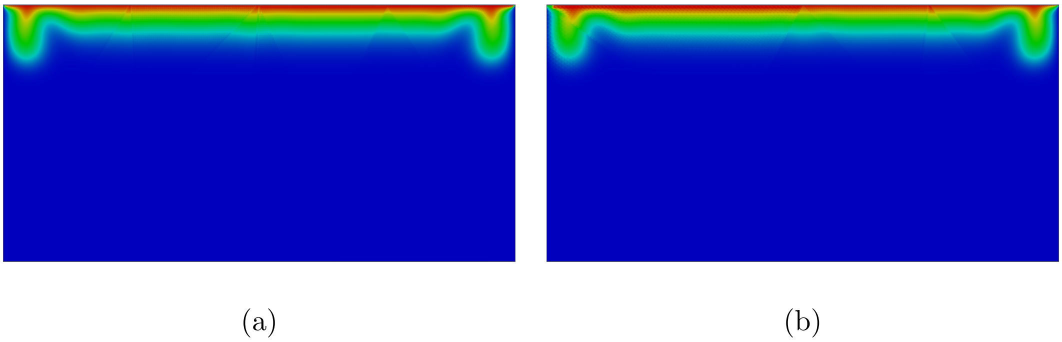


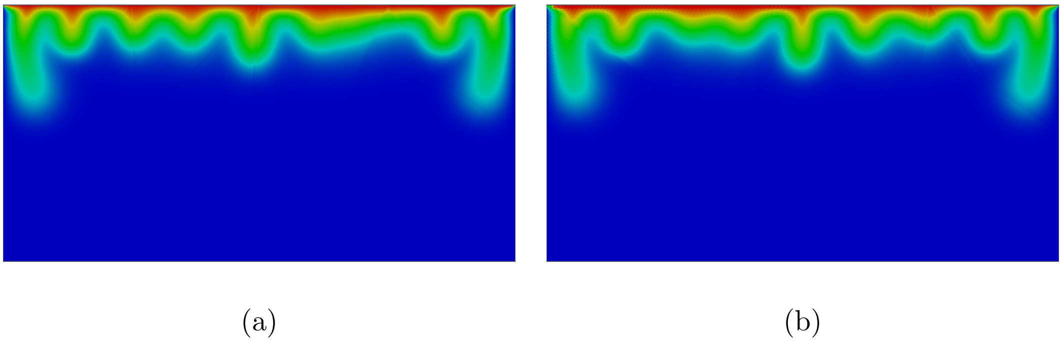
Figure 5.2: Initial unstructured meshes.

As we want to analyse the differences due to the unstructured mesh, we are interested in the unstable part of the experiment. Since up to 0.5 dimensionless seconds the behaviour of the model is driven by the diffusivity, we start our analysis from that moment. Figure 5.3 shows that two fingers appear close to the boundaries of the domain for both cases. When one dimensionless second has passed the diffusion frontier becomes unstable creating fingers in both experiments, see Figure 5.4. In this case, the difference is very small, but in Figure 5.5 the distribution of the fingers start to be more different. Nevertheless, it is important to remark that the size of the fingers and the total number of them is very similar. In Figures 5.6, 5.7 and 5.8, we can appreciate how the behaviour of both numerical experiments become more different. However, from a qualitative point of view, both experiments show an almost alike response. Finally, in Figure 5.9, after almost 10 dimensionless seconds, the domain has only one finger in both cases. In the (a) case it has two ingestion sources but they do not go as deep as the ingestion source of the finger we can see in the (b) case. Note that a dimensionless second corresponds to an average situation to a scale of 10 or 100 years.



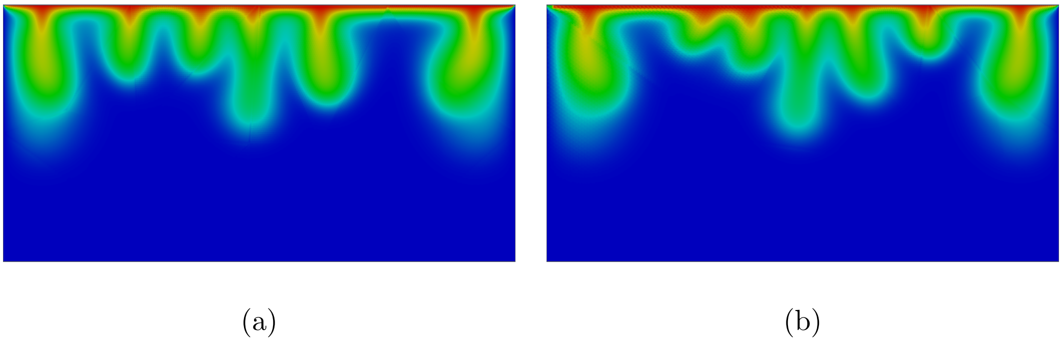
(a) (b)

Figure 5.3: CO_2 concentration after 0.53 dimensionless seconds for the two considered meshes.



(a) (b)

Figure 5.4: CO_2 concentration after 1.02 dimensionless seconds for the two considered meshes.



(a) (b)

Figure 5.5: CO_2 concentration after 1.52 dimensionless seconds for the two considered meshes.

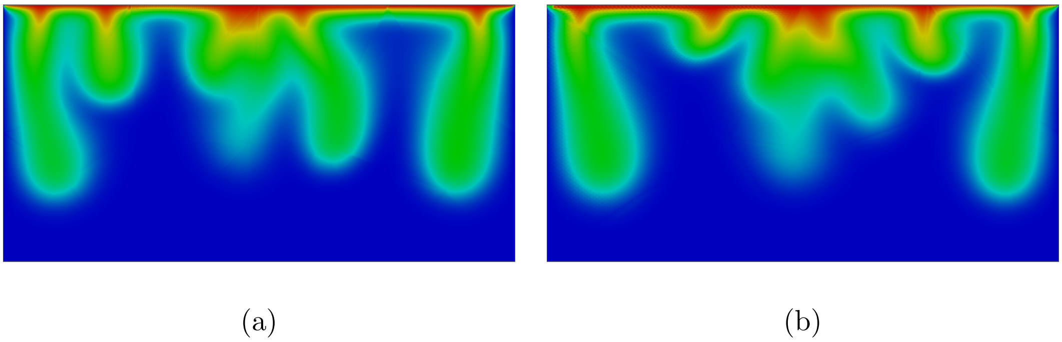


Figure 5.6: CO_2 concentration after 2.01 dimensionless seconds for the two considered meshes.

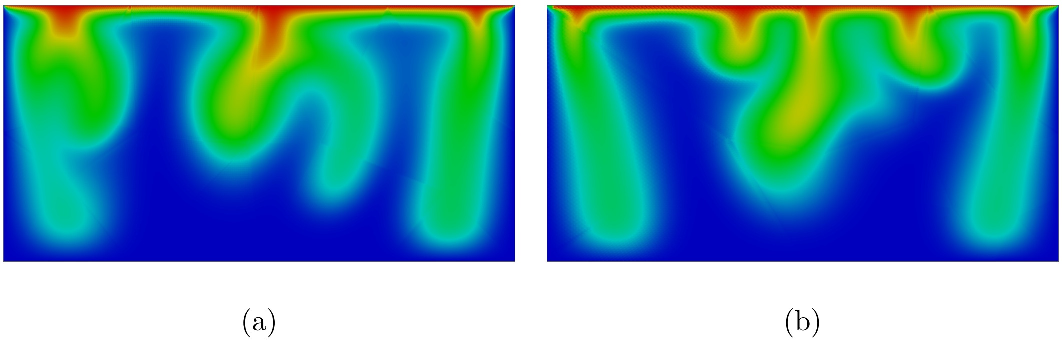


Figure 5.7: CO_2 concentration after 2.51 dimensionless seconds for the two considered meshes.

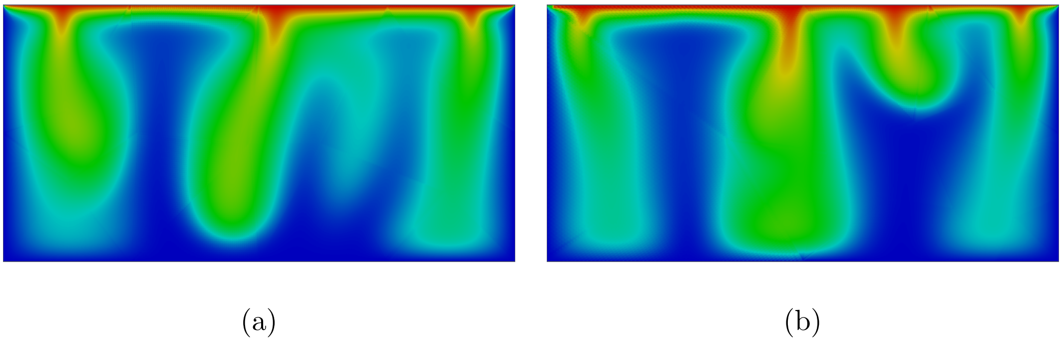


Figure 5.8: CO_2 concentration after 3.01 dimensionless seconds for the two considered meshes.

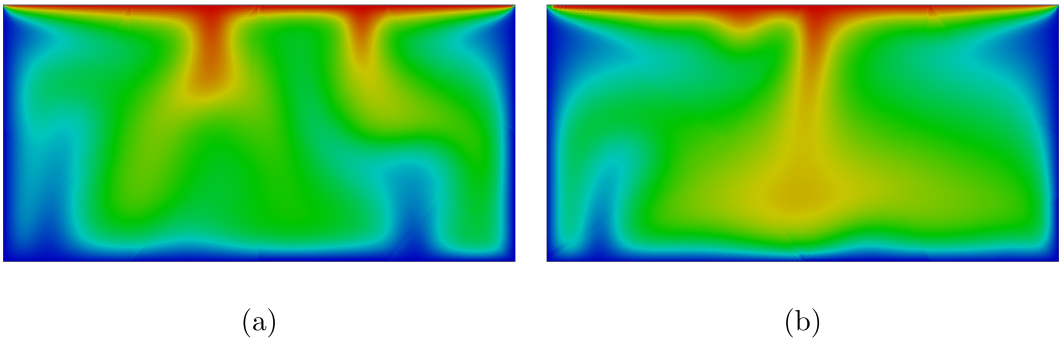


Figure 5.9: CO_2 concentration after 9.97 dimensionless seconds for the two considered meshes.

Rayleigh number equal to 10000

In this experiment, we increase the Rayleigh number in one order of magnitude. Hence, we expect a result driven more by the convection than by the diffusion process. The mesh used is shown in Figure 5.2 (a). In this case, as we reduce the diffusion term, we must be careful with the artificial diffusion introduced by the discretization method. If it becomes bigger or similar to the real diffusion, then the obtained results will not be acceptable. In order to know whether the artificial diffusion may modify our results or not we study the order of magnitude of the artificial and real diffusion. As it is known, the introduced artificial diffusion is proportional to the space discretization parameter. In our case, the initial unstructured triangulation contains 24 triangles, and each one contains 4096 scalar nodes. This means that the total amount of scalar nodes is about 10^5. As the Rayleigh number is 10^4 and the domain measures are 0.5 high and 1 width, then we can say that the space discretization is about 10^{-5}. Therefore, we can conclude that the artificial diffusion is of one order of magnitude smaller than the real diffusion.

Now, we proceed to study the behaviour of the experiment. In this case, after 0.07 dimensionless seconds, we can see in Figure 5.10 (a) that some fingers appear in the boundaries and also in the centre of the domain. Therefore, we can conclude that the instabilities have started earlier than for the previous case. Besides, more fingers are created, see Figure 5.10 (b).

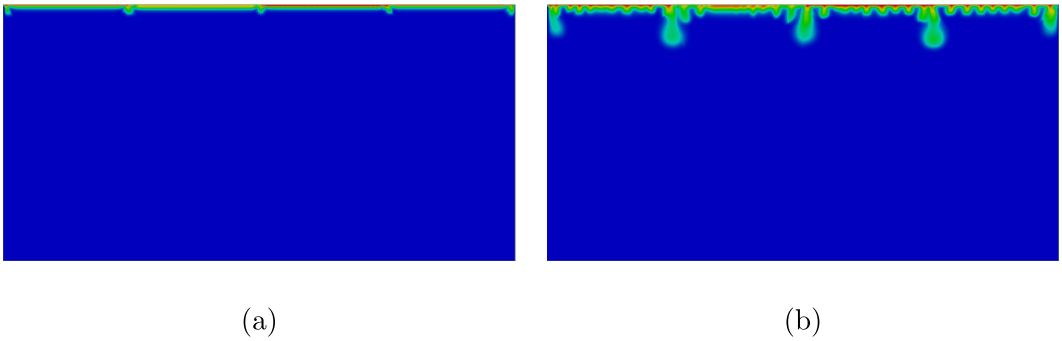


Figure 5.10: (a) CO_2 concentration after 0.07 dimensionless seconds. (b) CO_2 concentration after 0.25 dimensionless seconds.

In this experiment, the bigger fingers do not ingest the small ones so easily. Thus, after 0.39 dimensionless seconds, we can see in Figure 5.11 (a) that the majority of the fingers are growing. Nonetheless, some have been absorbed reducing the initial number of fingers, see Figure 5.11 (b) and Figure 5.12 (a). In the last picture, Figure 5.12 (b), some fingers have already reached the bottom. Also we can observe that some fingers are almost detached from the upper boundary. Nevertheless, the connection is still kept, although it is weak.

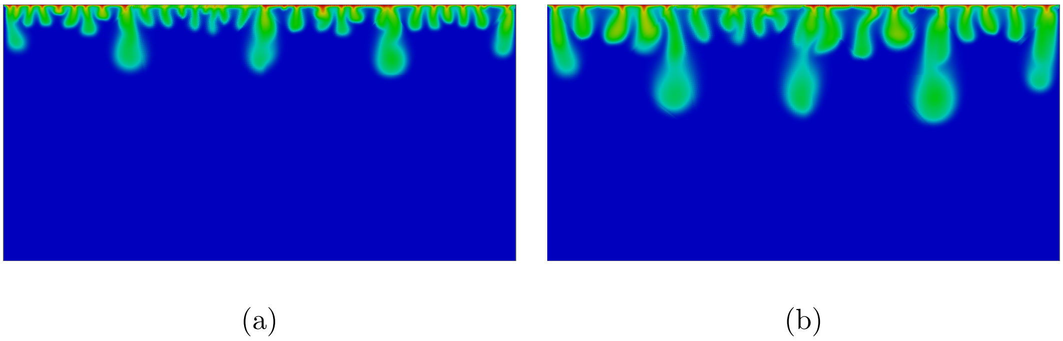


Figure 5.11: (a) CO_2 concentration after 0.39 dimensionless seconds. (b) CO_2 concentration after 0.68 dimensionless seconds.

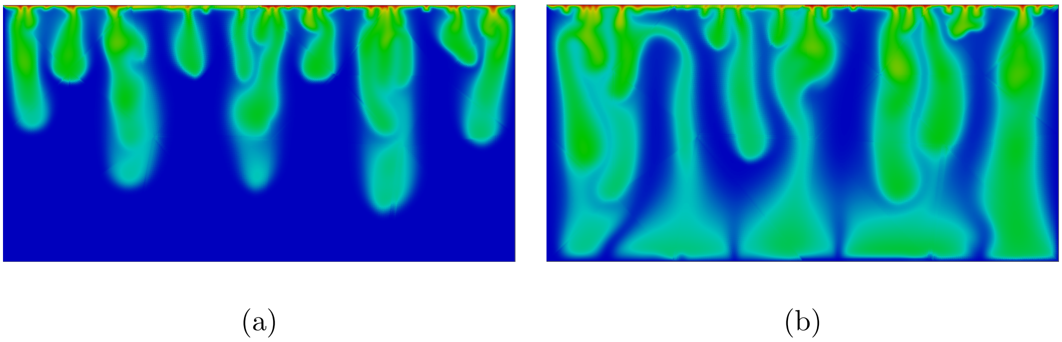


Figure 5.12: (a) CO_2 concentration after 1.22 dimensionless seconds. (b) CO_2 concentration after 2.50 dimensionless seconds.

5.4.1.2 Pipe shape domain

Finally, we perform two experiments in a pipe-shape domain in order to test the influence of the shape of the domain in the creation of fingers. In the first experiment, we impose a low Rayleigh value, making the behaviour of the test to be driven by the diffusivity. In the second experiment, a Rayleigh value of 5000 is considered. A domain with a curve part on the boundary is considered in order to see how the shape of the domain influences the CO_2 absorption process. In Figure 5.13 the smoothers used for the resolution of the advection equation are depicted.

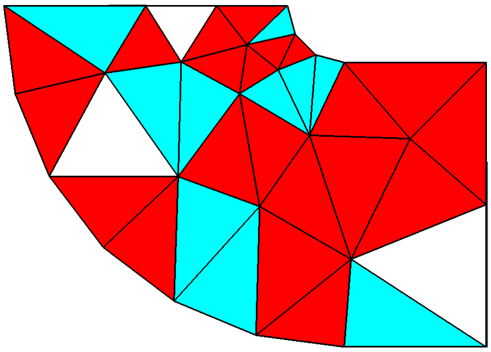


Figure 5.13: Different smoothers for the triangles of the coarsest grid: white corresponds to red-black, diamond smoother is represented by red and wormy smoother by blue.

Rayleigh number equal to 100

In this example, we expect a high diffusion component since the Rayleigh number is very low and the instabilities appear for a Rayleigh number higher than 380 (27). Looking at Figures 5.14, 5.15 and 5.16, we can see that in this case, a single finger has been created.

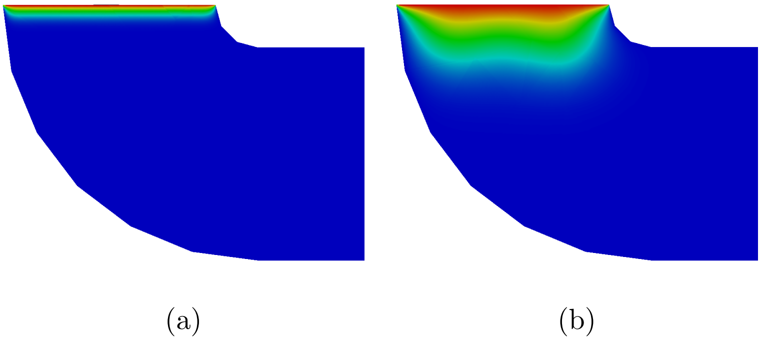


Figure 5.14: (a) CO_2 concentration after 0.03 dimensionless seconds. (b) CO_2 concentration after 1.00 dimensionless seconds.

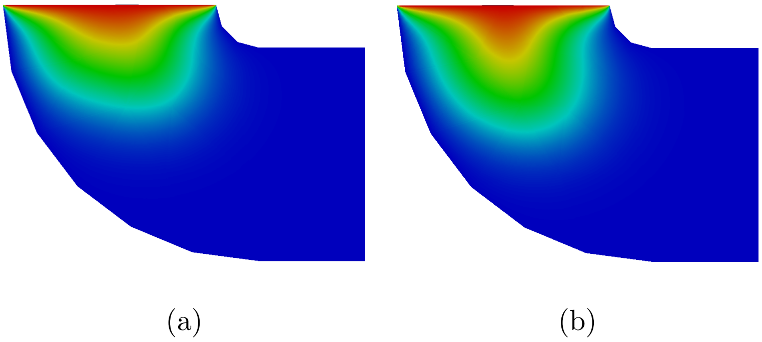


Figure 5.15: (a) CO_2 concentration after 2.08 dimensionless seconds. (b) CO_2 concentration after 2.51 dimensionless seconds.

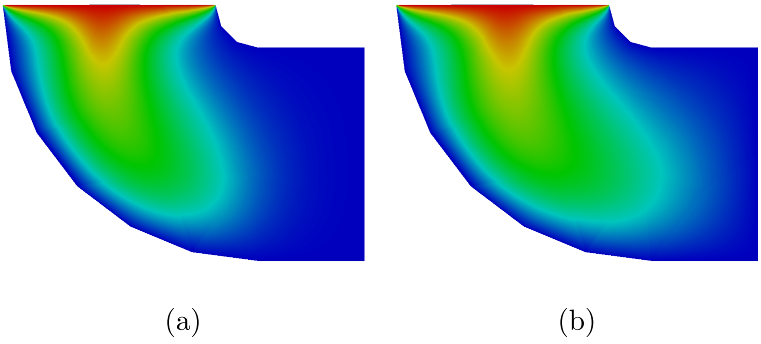


Figure 5.16: (a) CO_2 concentration after 5.02 dimensionless seconds. (b) CO_2 concentration after 9.99 dimensionless seconds.

Rayleigh number equal to 5000

Again, we face a high Rayleigh value. As expected, several fingers appear, see Figure 5.17. Later, they start to grow and merge between themselves, see Figure 5.18. Finally, due to the shape of the domain a large finger is created that ingest all the new fingers that have been created, see Figure 5.19.

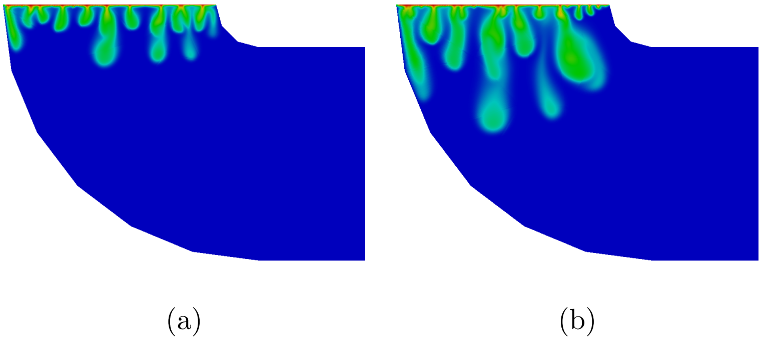


Figure 5.17: (a) CO_2 concentration after 1.02 dimensionless seconds. (b) CO_2 concentration after 2.08 dimensionless seconds.

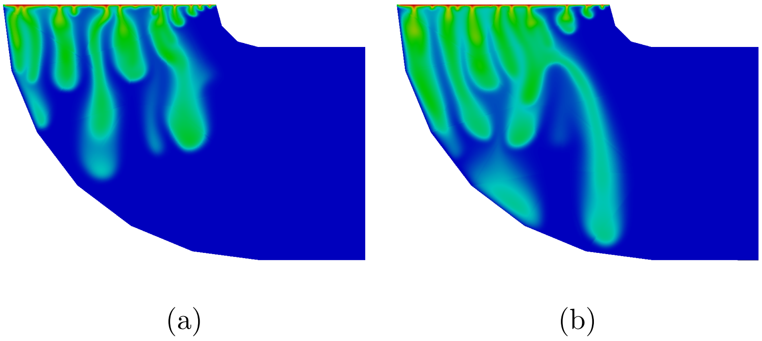


Figure 5.18: (a) CO_2 concentration after 3.00 dimensionless seconds. (b) CO_2 concentration after 4.02 dimensionless seconds.

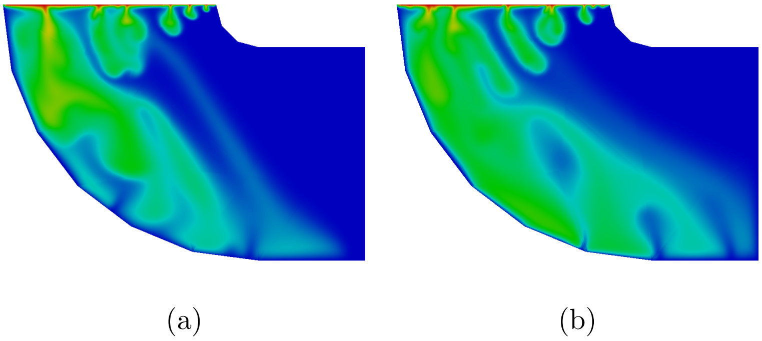


Figure 5.19: (a) CO_2 concentration after 6.07 dimensionless seconds. (b) CO_2 concentration after 8.02 dimensionless seconds.

5.4.2 Numerical experiments in heterogeneous media

This section deals with domains where different layers of soil are considered. We perform two experiments in which we will study the effect of these layers in the absorption of CO_2.

5.4.2.1 Hourglass shape domain

In this first experiment, we consider an hourglass shape domain in which the layer at the middle of the hourglass has a Rayleigh number value of 8000, while the rest of the domain has a value of 1000, see Figure 5.20. The considered smoothers for the resolution of the advection equation are depicted in Figure 5.20 (b).

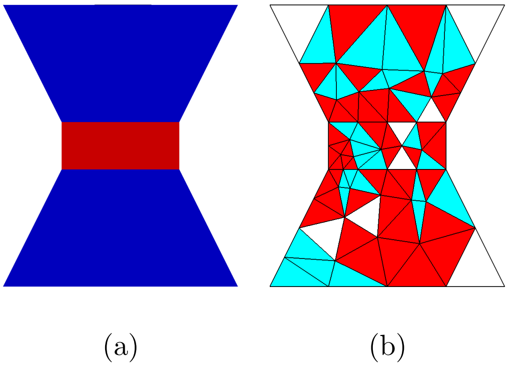


Figure 5.20: (a) Rayleigh number values in the considered domain. Blue stands for a Rayleigh number of 1000 and red for a number of 8000. (b) Different smoothers for the scalar case: white corresponds to red-black, diamond smoother is represented by red and wormy smoother by blue.

Before the fingers reach the layer at the centre of the domain, the behaviour of the experiment is similar to the homogeneous media experiments, see Figure 5.21. Nevertheless, once the fingers are approaching the centre layer, they start to reduce their speed,see Figure 5.22 (a), and even they retract themselves until a bigger finger is created, see Figure 5.22 (b). Finally, once a finger of CO_2 has reached the other section of the hourglass, Figure 5.23 (a), the stream of CO_2 is stable and a single finger of CO_2 goes through the rest of the domain, Figure 5.23 (b).

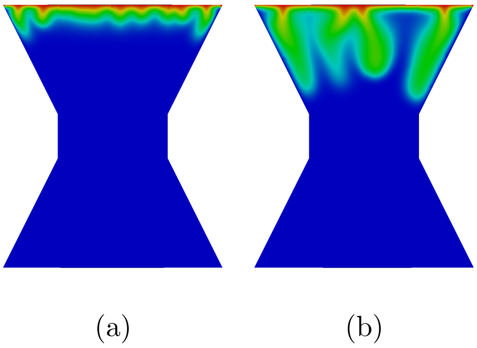


Figure 5.21: (a) CO_2 concentration after 1.09 dimensionless seconds. (b) CO_2 concentration after 2.47 dimensionless seconds.

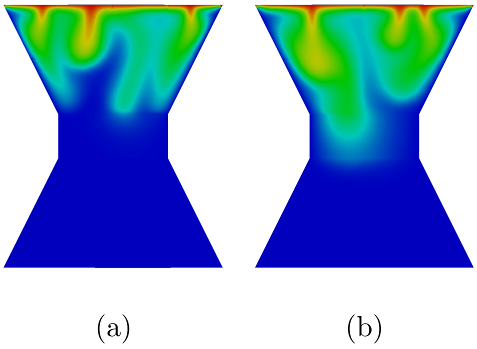


Figure 5.22: (a) CO_2 concentration after 3.56 dimensionless seconds. (b) CO_2 concentration after 5.01 dimensionless seconds.

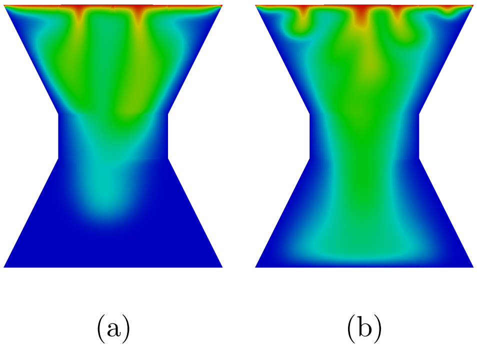


Figure 5.23: (a) CO_2 concentration after 7.02 dimensionless seconds. (b) CO_2 concentration after 9.99 dimensionless seconds.

5.4.2.2 Strata domain

Now, we want to analyse the behaviour of the process when many different layers are considered in the domain. The jump in the Rayleigh number between the layers is different in all the cases. Figure 5.24 (a) shows the four different layers. The light green area has a Rayleigh value of 100 (diffusive response), the blue one has a value of 1000, the red zone a value 5000 and the yellow of 8000. For this experiment, we have used 1024 scalar nodes per coarse triangle, and a time step of 0.0001, while the rest of the parameters are the same as before. For the advection equation, the smoothers used are depicted in Figure 5.24 (b). In Figure 5.25 (a), we see an initial stable layer of CO_2. However, once the CO_2 reaches the zone of a higher Rayleigh value, Figure 5.25 (b),

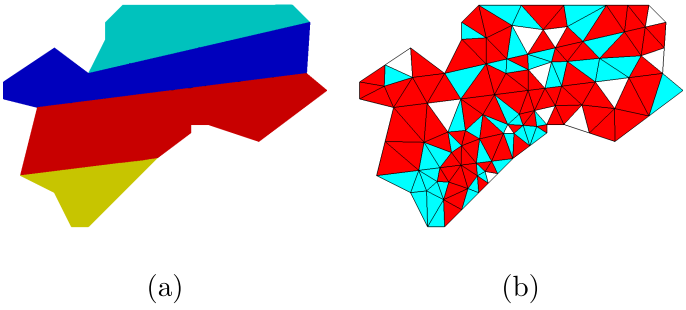


Figure 5.24: (a) Rayleigh numbers in the considered domain. Light green represents a Rayleigh number of 100, Blue stands for 1000, red for 5000 and yellow for 8000. (b) Different smoothers for the scalar case: white corresponds to red-black, diamond smoother is represented by red and wormy smoother by blue.

many fingers start to grow, which continue growing, as we seen in Figure 5.26 (a). In this case, when the fingers reach the layer with a Rayleigh number value of 5000, the fingers continue their movements without any visible effect, Figure 5.26 (b).

In order to study whether the layer with a Rayleigh number value of 5000 actually modifies the results or not, we decided to run this experiment with only two layers, that with a Rayleigh value of 100 was kept unchanged while the other three layers were merged into a single one with a Rayleigh number value of 1000. Comparing those two experiments, some differences in the behaviour of the fingers appeared. In the case of four layers, the fingers tended to reduce their number, but on the other hand they were bigger. In Figure 5.27 (a) and (b) the fingers have reached the bottom of the domain; some plumes of low concentration start to appear close to the boundaries. Finally, in Figure 5.28 (a) and (b) we can see that as the domain is almost full, the dissolution of CO_2 becomes slower.

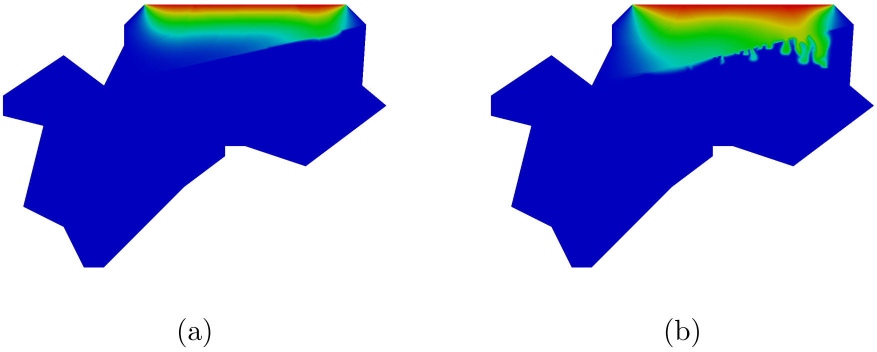


Figure 5.25: (a) CO_2 concentration after 0.33 dimensionless seconds. (b) CO_2 concentration after 0.83 dimensionless seconds.

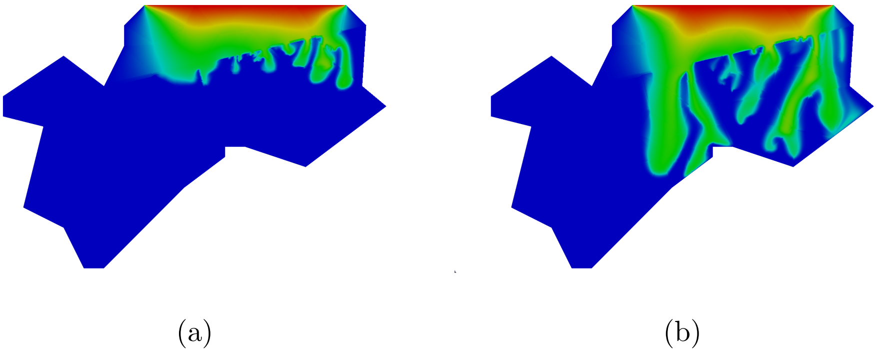


Figure 5.26: (a) CO_2 concentration after 1.17 dimensionless seconds. (b) CO_2 concentration after 2.33 dimensionless seconds.

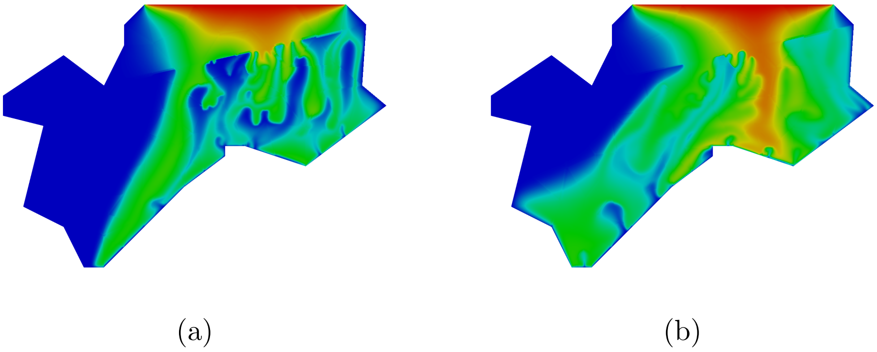


Figure 5.27: (a) CO_2 concentration after 3.33 dimensionless seconds. (b) CO_2 concentration after 4.99 dimensionless seconds.

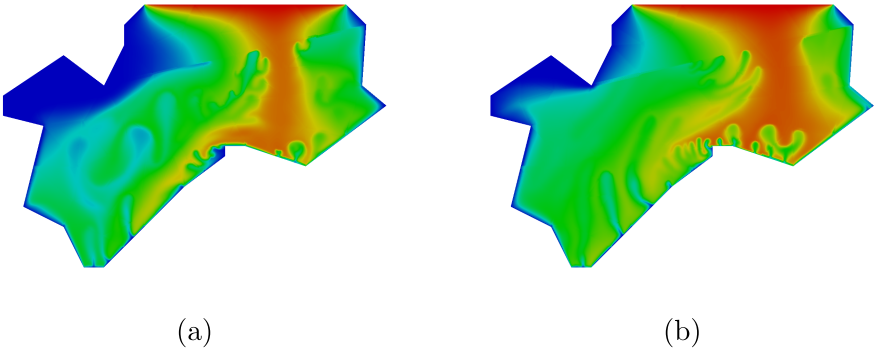


Figure 5.28: (a) CO_2 concentration after 6.66 dimensionless seconds. (b) CO_2 concentration after 9.99 dimensionless seconds.

Chapter 6

Conclusions

In this thesis, we have shown the viability of multigrid methods for discretizations on Voronoi meshes associated with semi-structured Delaunay triangulations. The use of semi-structured grids allows us to use geometric multigrid methods in relatively complex domains, obtaining a fast solver in domains where the direct application of a geometric multigrid method is not possible. The methodology presented in this thesis has been already successfully applied for finite element discretizations on rectangular and triangular grids (20, 21). This thesis is focused on the use of cell-centred discretizations, which has not been widely studied, except some works in which equilateral triangular grids were considered (35). Besides, the use of the Voronoi points as nodes yields a very simple discretization of the flux, whenever it is possible to obtain an acute triangulation, which is mathematically proved for two-dimensional domains (33). In order to overcome the anisotropies induced by the mesh, we have developed novel relaxation schemes like diamond and wormy smoothers and their Vanka-type versions. Moreover, a special local Fourier analysis has been developed taking into account the particularities of these discretizations on Voronoi grids.

The use of semi-structured grids allows us to choose the proper smoother depending on the geometry of each triangle of the coarsest grid, in the way that we can locally obtain a desired convergence rate. By considering the block-wise multigrid that uses these smoothers for the different triangles, the desired convergence rate can be globally obtained in the whole domain.

For the coarse-grid operators, we usually consider the direct discretization of the PDE on such grids. However, for the case of heterogeneous materials, we have to use the Galerkin operator, which yields very good results, and it does not produce a bigger stencil due to the choice of the inter-grid transfer operators, making its use very easy.

For solving the Darcy problem, in which vector and scalar unknowns appear, we have extended the methods developed for the scalar problem, obtaining very good results. Besides, for our particular problem, we have also proposed a distributive smoother that permits us to solve the system of equations with a better performance than the usual Vanka strategy, and also with a lower computational cost.

Finally, we have shown the good behaviour of this methodology by solving the Darcy-Oberbeck-Boussinesq problem that models density driven flows in porous media and can be used in the simulation of CO_2 sequestration in underwater saline aquifers. Despite the apparent simplicity of the model, it is very unstable and sensitive to small perturbations, and then it has to be handled carefully in order to obtain physically correct solutions.

As final conclusion, we can say that we have obtained a very robust and stable multigrid method that has proved its efficiency by solving standard model problems like the Laplace, convection-diffusion and the Darcy problems. Also, a more complex model, the Darcy-Oberbeck-Boussinesq model has been solved by using this method.

Many results from this thesis have been published in different international journals:

- P. SALINAS, C. RODRIGO, F.J. GASPAR, F.J. LISBONA: “Multigrid methods for cell-centered discretizations on triangular meshes” in Numerical Linear Algebra with applications, DOI: 10.1002/nla.1864, 2012.

- P. SALINAS, C. RODRIGO, F.J. GASPAR, F.J. LISBONA: “An efficient cell-centered multigrid method for problems with discontinuous coefficients on semi-structured triangular grids” in Computer and Mathematics with Applications, Volume 65, 1978–1989, 2013.

- C. RODRIGO, P. SALINAS, F.J. GASPAR, F.J. LISBONA: “Local Fourier analysis for cell-centered multigrid methods on triangular grids” in Journal of Computational and Applied Mathematics, DOI: 10.1016/j.cam.2013.03.040, 2013.

Regarding the material shown in Chapters 4 and 5, our intention is to publish a paper after the submission of this thesis.

Conclusiones

En esta tesis hemos mostrado la eficiencia de los métodos multimalla para discretizaciones sobre mallas de Voronoi asociadas con triangulaciones de Delaunay semi-estructuradas. El uso de mallas semi-estructuradas permite aplicar métodos multimalla geométricos en dominios relativamente complejos, obteniendo un buen factor de convergencia junto con una gran velocidad de resolución en dominios donde la aplicación directa del multimalla geométrico no es posible. La metodología presentada en esta tesis ya ha sido aplicada anteriormente con éxito para discretizaciones por elementos finitos sobre mallas tanto rectangulares como triangulares (20, 21). En este trabajo nos hemos centrado en discretizaciones centradas en celdas, lo cual apenas ha sido investigado, salvo en algunos trabajos en los que consideraban mallas de triángulos equiláteros (35). Además, el uso de los centros de Voronoi como nodos proporciona una discretización muy simple del flujo, siempre que sea posible obtener una triangulación formada por triángulos acutángulos, lo cual esta matemáticamente demostrado para dos dimensiones (33). Para superar las dificultades inducidas por las anisotropías originadas por el tipo de malla, nos hemos visto obligados a desarrollar nuevos suavizadores como diamond, wormy y sus versiones de tipo Vanka. Además, el uso de discretizaciones sobre los puntos de Voronoi, ha implicado el desarrollo de un análisis de Fourier especial, teniendo en cuenta las particularidades de estas discretizaciones.

El uso de mallas semi-estructuradas nos permite escoger el suavizador más adecuado dependiendo de la geometría de cada triángulo, pudiendo elegirlo de forma que localmente en ese triángulo podemos obtener el factor de convergencia deseado.

Para llevar a cabo la discretización en las mallas bastas, normalmente usamos la discretización directa, sin embargo cuando se consideran materiales heterogéneos, hemos observado que el uso del operador de Galerkin no solamente proporciona muy buenos resultados, si no que además, no da lugar a una molécula más compleja, haciendo su uso muy simple.

Para el problema de Darcy, hemos extendido los métodos desarrollados para un problema escalar, obteniendo muy buenos resultados. Además, para nuestro problema concreto, hemos propuesto también un suavizador distributivo que nos permite resolver el sistema de ecuaciones con un rendimiento superior al obtenido mediante los suavizadores de tipo Vanka, y además con un menor costo computacional.

Finalmente, hemos comprobado el buen rendimiento de este método mediante la resolución del modelo Darcy-Oberbeck-Boussinesq que modela flujos inducidos por densidad en medios porosos y que puede aplicarse a la simulación del almacenamiento de CO_2 en acuíferos subterráneos salinos. A pesar de la aparente simplicidad del modelo, es un problema muy inestable y muy sensible a pequeñas perturbaciones. Éste debe de ser tratado cuidadosamente para obtener soluciones físicamente razonables.

Como conclusión final, podemos afirmar que hemos obtenido un método multimalla robusto y estable que ha probado su eficiencia mediante la resolución de problemas estándar como el problema de Laplace, de convección-diffusión y el problema de Darcy. Y finalmente ha sido aplicado a un modelo más complejo, esto es el problema de Darcy-Oberbeck-Boussinesq.

Muchos de los resultados obtenidos en esta tesis han sido publicados en diferentes revistas internacionales:

- P. SALINAS, C. RODRIGO, F.J. GASPAR, F.J. LISBONA: “Multigrid methods for cell-centered discretizations on triangular meshes” en Numerical Linear Algebra with applications, DOI: 10.1002/nla.1864, 2012.

- P. SALINAS, C. RODRIGO, F.J. GASPAR, F.J. LISBONA: “An efficient cell-centered multigrid method for problems with discontinuous coefficients on semi-structured triangular grids” en Computer and Mathematics with Applications, Volume 65, 1978–1989, 2013.

- C. RODRIGO, P. SALINAS, F.J. GASPAR, F.J. LISBONA: “Local Fourier analysis for cell-centered multigrid methods on triangular grids” en Journal of Computational and Applied Mathematics, DOI: 10.1016/j.cam.2013.03.040, 2013.

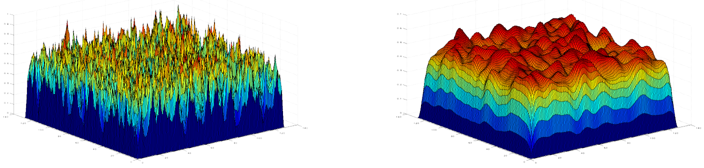
En relación al material presentado en los Capítulos 4 y 5, nuestra intención es publicar un artículo tras la presentación de esta tesis.

Appendices

Appendix A

Basic multigrid

The efficient numerical solution of many real problems modelled by partial differential equations strongly depends on the resolution of the corresponding large linear systems resulting from their discretization. Since their development in the 60's, multigrid methods have been proved to be among the most efficient numerical algorithms for solving this kind of systems, achieving asymptotically optimal complexity at least for elliptic problems. That is, the required computational work to solve a discrete problem is of the order of the number of unknowns of the corresponding system. These methods are iterative solvers with a characteristic property: the convergence of a multigrid method is independent of the discretization grid size. Multigrid methods are mainly based on the acceleration of the convergence of common iterative methods by using solutions obtained on coarser meshes as corrections. Most basic relaxation schemes suffer when smooth components are present in the error, however these components become high frequency components on a coarser grid, so that it makes sense to go down to such a grid and relax directly on the error by using the residual equation. We then return to the fine grid to correct the approximation obtained there with the correction from the coarser grid. Two principles are involved in the development of multigrid methods: the first one is the fact that some classical iterative methods have a strong smoothing effect on the components of the error corresponding to the high frequencies (high oscillating error components). This effect is due to the fact that relaxation schemes (as iterative methods are known in the multigrid context) basically average neighbouring discrete values, leading to a reduction of the oscillating character of the error. However, when the error is mainly composed of low frequency components, the differences between neighbouring discrete points are small and this leads to a slowdown in the convergence of these methods (7). In Figure A.1 this behaviour is shown, by choosing a random initial guess and applying some iterations of a classical iterative method, in particular, a Gauss-Seidel method. It is observed that in a few iterations the oscillating character of the algebraic error, is reduced resulting in a very smooth error. Notice that although the error is smooth, it is not significantly reduced in magnitude. The components associated with the low frequencies are annihilated by relaxation methods in a very slow way. When the error becomes smooth, the convergence of these methods gets worse. In order to avoid this, the second fundamental idea of these methods is employed, which is that a smooth error can be well represented on a coarser grid, where all the computations can be done faster because of the reduction of the number of grid-points. This strategy can be applied in a recursive way using coarser and coarser grids until a level of refinement is reached on which the application of an exact solver would be feasible. Consequently, multigrid methods are based on using a hierarchy of grids in order to eliminate all the different components of the error.



Initial error

Error after 10 iterations

Figure A.1: Gauss-Seidel smoothing effect on the error.

There are two possibilities to achieve an efficient interplay between the smoothing and the coarse–grid correction, which give rise to two different approaches to multigrid algorithms: geometric and algebraic. Algebraic multigrid (AMG) appears suitable for problems on unstructured grids. The use of semi-structured grids appears as an alternative to the use of AMG for unstructured grids for relatively complex domains, since these grids allow the efficient implementation of a geometric multigrid algorithm.

A.1 Multigrid components

Defining a sequence of grids $G_0, G_1, \ldots, G_l$, characterized by grid sizes $h_0 > h_1 > \ldots > h_l$, the multigrid iteration operator, M_k, can be recursively defined as follows:

$$\begin{aligned} M_0 &= 0, \\ M_k &= (S_k)^{\nu_2} C_k (S_k)^{\nu_1}, \ k = 1, \ldots, l, \text{ where} \qquad \text{(A.1)} \\ & C_k = I_k - I_{k-1}^{k}(I_{k-1} - (M_{k-1})^{\gamma}) L_{k-1}^{-1} I_k^{k-1} L_k, \end{aligned}$$

where S_k is the smoother on grid G_k, C_k represents the coarse-grid correction part of the algorithm, and the fact that the coarse grid equation is approximately solved by applying γ iterations of a multigrid cycle, is indicated by the term $(I_{k-1} - (M_{k-1})^{\gamma})L_{k-1}^{-1}$ (49). For defining (A.1), a discrete operator $L_k : \mathcal{G}(G_k) \rightarrow \mathcal{G}(G_k)$ approximating L_l, and a suitable relaxation process $S_k : \mathcal{G}(G_k) \rightarrow \mathcal{G}(G_k)$ must be defined on each grid $k = 0, \ldots, l-1$, and the corresponding transfer grid operators between consecutive grids are denoted by

$$I_k^{k+1} : \mathcal{G}(G_k) \rightarrow \mathcal{G}(G_{k+1}), \qquad \text{(A.2)}$$

$$I_{k+1}^{k} : \mathcal{G}(G_{k+1}) \rightarrow \mathcal{G}(G_k). \qquad \text{(A.3)}$$

The idea of using a sequence of grids permits us to process them in different ways. These manners of processing the grids are determined by the cycle index γ, which indicates the number of multigrid steps to perform on coarser grids. The value $\gamma = 1$ yields the V-cycle, which is the easiest recursive definition of a multigrid cycle. By increasing the number of times coarser grids are processed, other type of cycles can be defined. For example, $\gamma = 2$ yields the so-called W-cycle, and another type of cycle, as the F-cycle, is also commonly used. The F-cycle consists of going down to the coarsest grid and recursively interpolating to the next finer grid and applying a V-cycle until the finest mesh is reached (49). The different types of cycles are represented in Figure A.2.

From the definition of the multigrid operator (A.1), it is deduced that many details are open for discussion and decision. All its components, that is, the smoother, intergrid transfer operators (restriction and prolongation), the choice of the coarser grids and the operators on them, the type of cycle and the number of pre- and post-smoothing

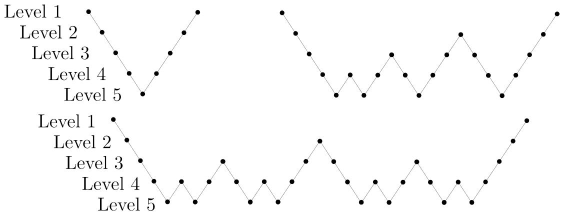


Figure A.2: Types of cycles.

steps, have to be properly chosen. For each concrete problem they have to be specified, and although for some problems the components which give good results are known, when dealing with new applications it is always a challenge to find the most suitable ones.

The coarsening strategy defines the hierarchy of grids necessary for the implementation of the geometric multigrid algorithm. Different ways of building the coarser grids can be considered, the more standard being the strategy of doubling the mesh size from G_k to G_{k-1}, that is, $h_{k-1} = 2h_k$.

Once the hierarchy of grids has been built, suitable discrete operators L_k on each coarse grid G_k, $k = 0, \ldots, l-1$ have to be chosen approximating the fine-grid discrete operator L_l. The most natural way to define these operators is to use the direct discretization of the equations on each coarse grid, and it is known as discretization coarse-grid approximation (DCA). This choice usually gives rise to reasonable approximations to L_l, ensuring the overall consistency of the discrete problem. An alternative choice of L_k is the so-called Galerkin coarse-grid approximation (GCA), which consists of using the inter-grid transfer operators, I_k^{k+1} and I_{k+1}^k, to define L_k from L_{k+1}, that is,

$$L_k = I_{k+1}^k L_{k+1} I_k^{k+1}. \tag{A.4}$$

When Galerkin coarse-grid operators are used, the restriction and the prolongation are often selected to be adjoint to each other for symmetric positive definite operators. This approximation is necessary for problems with discontinuous coefficients, and also in the application of algebraic multigrid, for example.

The choice of inter-grid transfer operators is, of course, closely related to the coarsening strategy. Transfer of information between two consecutive grids G_k, and G_{k+1} is done by the restriction and prolongation operators, I_{k+1}^k and I_k^{k+1}, which map

$(k+1)-$grid functions on $k-$grid functions and $k-$grid functions on $(k+1)-$grid functions, respectively.

The simplest way to define a prolongation operator is through linear interpolation. Other possible choices for the prolongation can be chosen, like high-order interpolations, for example. More sophisticated interpolation methods are sometimes helpful, depending on the problem and on the quantity to be transferred. For instance, matrix-dependent prolongations (13, 14), whose construction is based on the discrete operator under consideration, are useful for problems with discontinuous coefficients and in the algebraic multigrid framework.

On the other hand, the simplest restriction operator which comes into mind is the injection operator. Injection is easily programmed, but restriction operators which define a coarse-grid value by a certain averaging of neighbouring fine-grid values provide robust algorithms for more general problems. The full-weighting restriction is commonly used.

The smoother usually plays an important role in multigrid algorithms, above all in the geometric approach. From the early stages of multigrid development until now, a wide variety of smoothing procedures has been investigated apart from the standard ones like Jacobi and Gauss-Seidel type relaxations in all their varieties. Smoothers as ADI- or CG-type methods, the incomplete LU-decomposition (ILU), distributive relaxations, Vanka-type smoothers, and so on, have been widely used in the multigrid framework, and this list could be extended with many more, (49).

A.2 Local Fourier Analysis

The local Fourier analysis (LFA), introduced by Brandt in 1977 (4), is a quantitative analysis for multigrid algorithms, in the sense that it provides accurate predictions of the asymptotic convergence rates of such methods. This analysis is based on the Fourier transform theory, and a good introduction can be found in the books by Trottenberg et al. (49), and Wienands and Joppich (57). A k-level local Fourier analysis is a very useful tool to predict the convergence rate of a multigrid algorithm. The main idea of this analysis is formally to extend all multigrid components to an infinite grid, neglecting the boundary conditions, and to restrict the analysis to discrete linear operators with constant coefficients. Despite these restrictions, if boundary conditions are appropriately treated, in general this analysis matches the numerical results satisfactorily.

In this Appendix, our intention is just to explain the easiest application of LFA, that is, the smoothing analysis. This analysis is enough to study the performance of the smoothers and to understand the LFA concept. Nevertheless, it is not enough to simulate the multigrid behaviour. For that, at least a two-grid analysis has to be carried out. For a deeper understanding of this analysis we refer the reader to (57), where LFA is explained in detail.

A.2.1 Smoothing analysis

In order to explain the smoothing analysis, we will consider the standard finite difference discretization of Laplace operator on a Cartesian grid. LFA is based on the idea that a discrete grid function can be written as a formal linear combination of Fourier modes.

$$\varphi(\boldsymbol{\theta}, \mathbf{x}) = e^{\iota \boldsymbol{\theta} \cdot \mathbf{x}/h}, \tag{A.5}$$

where $\mathbf{x}$ is a point of the grid and $\boldsymbol{\theta}$ is a frequency in $(-\pi, \pi]^2$.

It is well-known that for each frequency the corresponding Fourier mode is an eigenfunction of the discrete Laplace operator, with corresponding eigenvalue, called symbol in this context,

$$\tilde{L}_h(\boldsymbol{\theta}) = \frac{1}{h^2}(4 - e^{\iota\theta_1/h} - e^{\iota\theta_2/h} - e^{-\iota\theta_1/h} - e^{-\iota\theta_2/h}). \tag{A.6}$$

Any standard smoothing operator can be described by a splitting of the discrete operator $L_h = L_h^+ + L_h^-$. Then, the symbol of any relaxation operators is given by

$$\tilde{S}_h(\boldsymbol{\theta}) = -\frac{\tilde{L}_h^-(\boldsymbol{\theta})}{\tilde{L}_h^+(\boldsymbol{\theta})}. \tag{A.7}$$

Since the idea of smoothing analysis is to study how the smoother eliminates the high-frequency components of the error, we will only consider $\tilde{S}_h(\boldsymbol{\theta})$, for $\boldsymbol{\theta}$ being a high frequency, see Figure A.3. Then, the smoothing factor is defined as the maximum of the modulus of these values. For example, Gauss-Seidel method provides a smoothing factor of 0.5 which is obtained for the frequency $(\theta_1, \theta_2) = (\pi/2, 0.6435)$.

The smoothing analysis is a good start to know a priori the behaviour of a method.

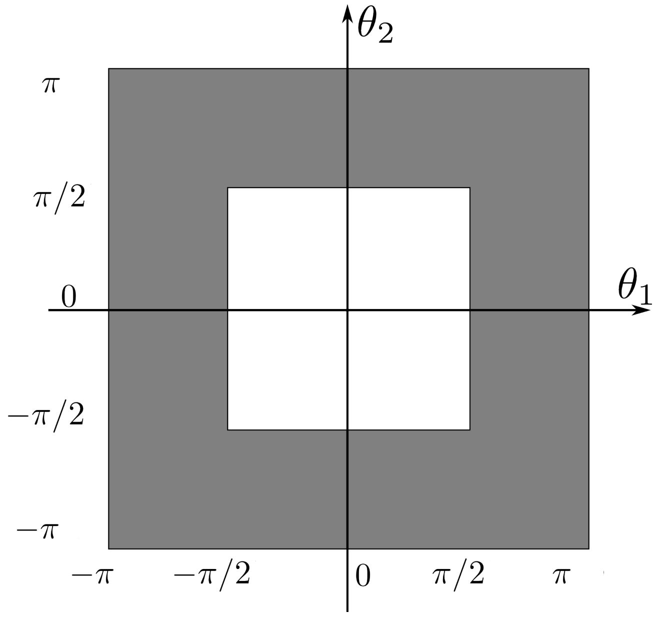


Figure A.3: High and low frequencies, grey and white areas respectively, for standard coarsening.

Bibliography

[1] R. E. ALCOUFFE, A. BRANDT, J. E. DENDY, J. W. PAINTER: “The multi-grid method for the diffusion equation with strongly discontinuous coefficients” in SIAM J. Sci. Stat. Comput., Volume 2, 430–455, 1981. 54

[2] B. BERGEN, T. GRADL, F. HÜLSEMANN, U. RUEDE: “A massively parallel multigrid method for finite elements” in Comput. Sci. Eng., Volume 8, 56–62, 2006. 5

[3] J. BRAMBLE, R. EWING, J. PASCIAK, J. SHEN: “The analysis of multigrid algorithms for cell centered finite difference methods” in Adv. Comput. Math.,Volume 5, 15–29, 1996. 5

[4] A. BRANDT: “Multi–level adaptive solutions to boundary-value problems” in Math. Comput., Volume 31, 333–390, 1977. iii, 1, 26, 28, 127

[5] A. BRANDT: “Guide to multigrid development” in Multigrid Methods, Lecture Notes in Mathematics, Volume 960, 220–312, 1982. 17, 38, 54

[6] A. BRANDT: “Rigorous quantitative analysis of multigrid I. Constant coefficients two level cycle with L2 norm” in SIAM J.Numer. Anal., Volume 31, 1695–1730, 1994. 26

[7] W.L. BRIGGS, V.E. HENSON, S.F. McCORMICK: “A multigrid tutorial”. SIAM publications, 2000. 11, 124

[8] S. CHANDRASEKHAR: “Hydrodynamic and hydromagnetic stability”. New York, 1961. 90

[9] P.G. CIARLET: “The finite element method for elliptic problems”. North-Holland, Amsterdam: Series “Studies in Mathematics and its Applications”, 1978. 6

[10] H. DARCY: “Les Fontaines Publiques de la Ville de Dijon” in Paris: 1856. 91

[11] R. DAVIES, A. REA, D. TSAPTINOS: “Introduction to Fortran 90 An introduction Course for Novice Programmers”. Cardiff: HPC training and education centres. 42

[12] J.E. DENDY, Jr.: “Black box multigrid” in J. Comput. Phys., Volume 48, 366-386, 1982 53

[13] J.E. DENDY Jr. “Black box multigrid for nonsymmetric problems” in Appl. Math. Comput., Volume 8, 366-386, 1983. 127

[14] P.M. de ZEEUW: “Matrix-dependent prolongations and restrictions in a black box multigrid solver” in J. Comput. Appl. Math., Volume 33, 1–27, 1990. 53, 127

[15] H. ERTEN, A. ÜNGÖR: “Quality Triangulations with Locally Optimal Steiner Points” in SIAM J. Sci. Comput., Volume 31, 2103–2130, 2009. 36

[16] H. ERTEN, A. ÜNGÖR: “Computing Triangulations without Small and Large Angles” in International Symposium on Voronoi Diagrams (ISVD) 2009. 36

[17] R.E. EWING, O. SAEVAREID, J. SHEN: “Discretization schemes on triangular grids” in Comp. Meth. Appl. Mech. Eng., Volume 152, 219–238, 1998. 53

[18] R.E. EWING, R.D. LAZAROV, P.S. VASSILEVSKI: “Local refinement techniques for elliptic problems on cell-centered grids” in Numerische Mathematik, Volume 59, 431–452, 1991. 5

[19] F.J. GASPAR: “Multi-grid methods for higher-order discretisation in convection-diffusion problems”. University of Zaragoza: 1997. 63

[20] F.J. GASPAR, J.L. GRACIA, F.J. LISBONA: “Fourier analysis for multigrid methods on triangular grids” in SIAM J. Sci. Comput., Volume 31, 2081–2102, 2009. iv, 2, 21, 26, 115, 117

[21] F.J. GASPAR, J.L. GRACIA, F.J. LISBONA, C. RODRIGO: “Multigrid finite element methods on semi-structured triangular grids for planar elasticity” in Numer. Linear Algebra Appl, Volume 17, 473–493, 2010. iv, 2, 115, 117

[22] H. GOMEZ, L. CUETO-FELGUEROSO, R. JUANES: “Three-dimensional simulation of unstable gravity-driven infiltration of water into a porous medium” in Journal of Computational Physics, Volume 238, 217–239, 2013. 99

[23] D. GREENSPAN, V. CASULLI: “Numerical analysis for applied mathematics, science, and engineering”. Addison-Wesley publishing Company, Inc., 1988. 70

[24] M.A. HESSE: “Mathematical modeling and multiscale simulation of CO_2 storage in saline aquifers”. Stanford University: 2008. 3, 94, 98, 99

[25] P.W. HEMKER: “On the order of prolongations and restrictions in multigrid procedures” in J. Comput. Appl. Math., Volume 32, 423–429, 1990. 38, 54

[26] R. HERBIN: “An error estimate for a finite volume scheme for a diffusion-convection problem on a triangular mesh” in Numer. Meth. for Partial Diff. Equat., Volume 11, 165–174, 1995. 8

[27] E. O. HOLZBECHER: “Modeling density-driven flow in porous media”. Berlin: Springer, 1998. 89, 106

[28] M. KHALIL: “Analysis of linear multigrid methods for elliptic differential equations with discontinuous and anisotropic coefficients” Technical University Delft: 1989. 5

[29] M. KHALIL, P. WESSELING: “A cell–centered multigrid method for three-dimensional anisotropic-diffusion and interface problems” in Mandel, J., McCormick, S.F. (eds.), Preliminary Proc. of the 4th Copper Mountain Conference on Multigrid Methods, Volume 3, 99–117, 1989. 5

[30] M. KHALIL, P. WESSELING: “Vertex-centered and cell-centered multigrid for interface problems” in Mandel, J., McCormick, S.F. (eds.), Preliminary Proc. of the 4th Copper Mountain Conference on Multigrid Methods, Volume 3, 61–97, 1989. 5

[31] M. KHALIL, P. WESSELING: “Vertex–centered and cell–centered multigrid for interface problems” in J. Comput. Phys., Volume 98, 1–20, 1992. 5, 53

[32] B. KOOBUS, M.H. LALLEMAND, A. DERVIEUX: “Unstructured volume-agglomeration MG: Solution of the Poisson equation” in INRIA report n 1946, 1993. 58

[33] E. KOPCZYŃSKI, I. PAK, P. PRZYTYCKI: “Acute triangulations of polyhedra and $\mathbb{R}^n$”: DOI: arXiv:0909.3706v3. 115, 117

[34] D.Y. KWAK: “V–cycle multigrid for cell-centered finite differences” in SIAM J. Sci. Comput., Volume 21, 552–564, 1999. 5

[35] D.Y. KWAK: “V–cycle multigrid for cell-centered finite differences” in SIAM J. Sci. Comput, Volume 21, 552–564, 1999. 53, 115, 117

[36] D.Y. KWAK, J.S. LEE: “Multigrid algorithm for the cell-centered finite difference method II: Discontinuous coefficient case” in Numer. Meth. Partial Differential Eqs., Volume 20, 723–741, 2004. 5, 54

[37] D.Y. KWAK, J.S. LEE: “Comparison of V–cycle Multigrid Method for Cell–Centered Finite Difference on Triangular Meshes” in Numer. Meth. Partial Differential Eqs., Volume 22, 1080–1089, 2006. 5, 53

[38] E. LINDENBERG, P. BERGMO: “The long-term fate of CO_2 injected into an aquifer” in Proceedings of the 6th International Conference on Greenhouse Gas Technologies, Volume 1, 489–494, 2003. 98

[39] C. LIU, Z. LIU, S. McCORMICK: “An efficient multigrid scheme for elliptic equations with discontinuous coefficients” in Comm. Appl. Numer. Meth., Volume 8, 621–631, 1991. 54

[40] D.J. MAVRIPLIS, V. VENTAKRISHNAN, "Agglomeration multigrid for viscous turbulent flows" in NASA Contractor Report 194953, ICASE Report No. 94-62. 58

[41] I.D. MISHEV: "Finite volume methods on Voronoi meshes" in Numer. Methods Partial Differential Equations, Volume 14, 193–212, 1998. 8, 39

[42] J. OTERO, L.A. DONTCHEVA, H, JOHNSTON, R.A. WORTHING, A. KURGANOV, G. PETROVA, C.R. DOERING: "High-Rayleigh-number convection in a fluid-saturated porous layer" in J. Fluid Mech., Volume 500, 263–281, 2004. iv, 2

[43] M. Mohr, R. Wienands: "Cell–centered multigrid revisited" in Comput. Visual. Sci., Volume 7, 129–140, 2004. 22

[44] C. RODRIGO, F.J. GASPAR, C.W. OOSTERLEE, I. YAVNEH: "Accuracy Measures and Fourier Analysis for the Full Multigrid Algorithm" in SIAM J. Sci. Comput., Volume 32, 3108–3129, 2010. 21

[45] C. RODRIGO, F.J. GASPAR, F.J. LISBONA: "Geometric multigrid methods on triangular grids. Application to semi-structured meshes" in Lambert academy, 2012. 25

[46] A.A. SAMARSKII: "The theory of difference schemes". New York-Basel: Marcel Dekker Inc., Volume 1148, 203–222 2001. 56

[47] J.R. SHEWCHUCK: "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator" in Applied Computational Geometry: Towards Geometric Engineering, Volume 22, 21–74, 1996. 36

[48] J.R. SHEWCHUCK: "Delaunay Refinement Algorithms for Triangular Mesh Generation" in Comput. Geom., 2002. 36

[49] U. TROTTENBERG, C.W. OOSTERLEE, A. SCHÜLLER: "Multigrid". New York: Academic Press, 2001. iii, 1, 17, 21, 26, 30, 53, 58, 63, 66, 70, 125, 127

[50] P.N. VABISHCHEVICH: "Finite-difference approximation of mathematical physics problems on irregular grids" in Comp. methods in applied mathematics, Volume 5, 294–330, 2005. 76, 81

[51] S.P. VANKA: "Block-Implicit Multigrid solution of Navier-Stokes equations in primitive Variables" in Journal of computational physics, Volume 65, 138–158, 1986. 77

[52] P.S. VASSILEVSKI, S.I. PETROVA, and R.D. LAZAROV: "Finite difference schemes on triangular cell-centered grids with local refinement" in SIAM J. Sci. Stat. Comput., Volume 13, 1287–1313, 1992. 8

[53] P. WESSELING: "Cell–centered multigrid for interface problems" in J. Comput. Phys., Volume 79, 85–91, 1988. 5

[54] P. WESSELING: "Cell–centered multigrid for interface problems" in New York: Marcel Dekker, Volume 110, 631–641, 1988. 5

[55] P. WESSELING: "A multigrid method for elliptic equations with a discontinuous coefficient" in Proceedings of the First International Conference on Industrial and Applied Mathematics, Volume 87, 173–183, 1988. 5

[56] P. WESSELING: "An introduction to multigrid methods". Chichester: John Wiley, 1992. 5, 53, 56, 58, 59

[57] R. WIENANDS, W. JOPPICH: "Practical Fourier analysis for multigrid methods". Chapman and Hall/CRC Press, 2005. 28, 127, 128

[58] G. WITTUM: "Multi-grid methods for Stokes and NavierStokes equations with transforming smoothers: algorithms and numerical results"in Numer. Math., Volume 54, 543–563, 1989. 77

[59] I. YAVNEH: "Coarse-Grid Correction for Nonelliptic and Singular Perturbation Problems" in SIAM J. Sci. Comput., Volume 19, 1682–1699, 1998. 58