Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 267

Implementation of a Parallel Multigrid Solver for the Solution of Higher-dimensional PDEs on Anisotropic Grids in DUNE

Daniel Pfister

Studiengang:	Informatik
Prüfer/in:	Jun.-Prof. Dr. Dirk Pflüger
Betreuer/in:	Mario Heene, M.Sc., Dr. Steffen Müthing
Beginn am:	2015-11-02
Beendet am:	2016-05-03
CR-Nummer:	G.1.0, G.1.3, G.1.8

Abstract

English

In this bachelor thesis a parallel geometric multigrid solver is extended to be used with the sparse grid combination technique. To test this method the stationary advection-diffusion equation is used, which is discretized by the finite volume element method. Suitable multigrid components are chosen and their convergence and parallel efficiency is tested in numerical experiments. The implementation uses the Distributed and Unified Numerics Environment (DUNE) for the solver and discretization. The SG^{++} Distributed Combigrid module is employed for the combination technique. A small weak scaling study is conducted to measure the influence of the aspect ratio of the grids.

Deutsch

In dieser Bachelorarbeit wird ein paralleler geometrischer Mehrgitterlöser erweitert um mit der der Dünngitter-Kombinationstechnik genutzt werden zu können. Zum Testen dieser Methode wird die stationäre Advektions-Diffusions Gleichung verwendet, welche mit der Finite-Volumen-Elemente Methode diskretisiert wird. Es werden geeignete Komponenten für den Mehrgitterlöser gewählt und ihre Konvergenz und parallele Effizienz wird in numerischen Experimenten getestet. Die Implementierung nutzt die Distributed and Unified Numerics Environment (DUNE) für den Löser und die Diskretisierung. Das SG^{++} Distributed Combigrid Modul wird für die Kombinationstechnik eingesetzt. Eine kleine Studie zur schwachen Skalierbarkeit wird durchgeführt um den Einfluss der Streckung der Gitter zu messen.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> *"In view of all that we have said in the foregoing sections, the many obstacles we appear to have surmounted, what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientist from earliest days."*
>
> — Richard Bellman, *Adaptive Control Processes* [9]

The term "curse of dimensionality" was coined in 1961 by Richard Bellman in a book of his [9]. Therein it refers to the volume of a mathematical space which grows exponentially with its dimensionality. One prominent application that is affected by this so-called curse is the numerical solution of partial differential equations (PDEs).

In the case of PDEs high dimensionality typically means four or more dimensions. Such equations appear among others in finance and physics. In financial mathematics the well known Black-Scholes equation can have arbitrarily high dimensionality [11]. In plasma physics the gyrokinetic approach for the simulation of turbulence in plasma fusion leads to a five-dimensional PDE [24].

The difficulty arises when the function domain is discretized, which leads to $\mathcal{O}(h^{-d})$ degrees of freedom (DOFs) when a d-dimensional regular full grid with spacing h is used. One approach to tackle this problem are sparse grid methods. These were originally introduced for just this purpose [38, 21, 15]. A discretization using sparse grids results in only $\mathcal{O}(h^{-1} \cdot \log(h^{-1})^d)$ degrees of freedom, which is a considerable improvement over full grids. This comes at the cost of some loss in accuracy that can however be bounded if the function satisfies certain smoothness conditions.

The approach used in this bachelor thesis is the *sparse grid combination technique*, which was introduced by Griebel et al. [22] in 1992. Here the problem is discretized on a number of coarse but anisotropic subspaces of the full grid. The solutions to these discretizations are then linearly combined to approximate the solution to the original problem discretized on a sparse grid. The strongly anisotropic grids which occur in this method may result in slow rates of convergence

for standard iterative solvers like the conjugate gradient method. However this effect can be countered by using appropriate preconditioners or solvers.
The goal of this bachelor thesis is threefold:

1. A finite volume element (FVE) discretization is implemented for the linear advection-diffusion equation with d-linear trial functions on rectangular grids. This serves as a toy problem with arbitrary dimensionality, that can be used to examine the behavior of advection-dominated flow problems.

2. An existing linear geometric multigrid (GMG) solver is extended to overcome the specific difficulties of this application. It will be compared to an algebraic multigrid (AMG) solver with respect to the rate of convergence and parallel execution time.

3. This solver is then implemented to be used with a sparse grid combination technique library. The scaling behavior of this constellation is then examined.

The focus of this work is not a rigorous analysis of the presented methods, but an empirical examination based on numerical experiments.

1.1 Outline

The second chapter is about the formulation of the problem to be solved. First the model problem and its continuous mathematical formulation are established. Thereafter the focus lies on the discretization of this model problem. At last basics of the combination technique are explained.

The third chapter goes into the method for solving the formulated problem. The basic concepts of multigrid algorithms are explained. Here the choices for the various components of the multigrid method are shown.

In the fourth chapter all the relevant implementation details are given. At first the used libraries are presented. After that an overview is given for the code written for this work.

The fifth chapter is about the results of the numerical experiments. These regard the accuracy of the discretization, the comparison of the multigrid solvers and the scaling of the combination technique utilizing the geometric multigrid solver.

1.2 Related Work

As there are are multiple goals involved in this bachelor thesis, there is also plenty of related work. In the following some of the most influencing works are listed.

The FVE method [16] has gained some attention for the simulation of fluid dynamics in the last decades and is also known as vertex-centered finite volume method [39, 27], box method [3, 23], control-volume finite element method [2, 31]

or covolume method [17]. It was introduced by Baliga et al. [2] and has been extensively studied for the advection-diffusion equation with linear trial functions on simplicial meshes.

In this work the FVE method is used with d-linear trial functions on rectangular grids. There recently have been some advances in the error analysis for this case. An error bound of $\mathcal{O}(h^2)$ in the L_2 norm has been derived for the Poisson equation with a bilinear finite element space and quadrilateral meshes by Lv et al. [29].

Chou et al. [17] examined the convergence of multigrid algorithms for the FVE method on triangular meshes, which showed the feasibility of this combination. The effect of different coarsening strategies on the convergence of multigrid algorithms was studied by Sprengel [36], by using stable subspace splittings. A multigrid algorithm was used in the original paper on the combination technique [22]. Zubair et al. [10, 11] also used a multigrid algorithm with the combination technique to solve high-dimensional PDEs. Therein the focus lies on the diffusion equation discretized with the standard finite element method. Unlike in the previous works, we study the scaling with two levels of parallelization, where the first level is the combination technique and the second level is a domain decomposition on the component grids.

Chapter 2

Problem Formulation

The overarching goal of this work is to efficiently solve the advection-diffusion equation on the high-dimensional unit hypercube. To this end the first step is a mathematical formulation of the continuous problem. Thereafter a discretization is needed so that the problem can be numerically solved by a computer. This discretization will first be done on regular full grids, on which the combination technique will later build upon.

2.1 Advection-Diffusion Equation

The advection-diffusion equation is well suited as a model problem because it can describe a variety of phenomena. It is a PDE defined as

$$\frac{\partial u}{\partial t} + \nabla \cdot (-D\nabla u + au) = f, \qquad u \in C^2(\mathbb{R}^d \times \mathbb{R}). \tag{2.1}$$

This equation can model the following physical process. The function u to be determined describes the concentration of a material that is suspended in a fluid. The concentration is however too thin to have any influence on the velocity of this fluid. There are three terms of which each models a different physical process. The term $\nabla \cdot D\nabla u$ is responsible for the diffusion of the material. Therein the second-order tensor field D determines how fast this process takes place. The term $\nabla \cdot au$ models the advection of the material, where the vector field a describes the velocity of the fluid. The scalar field f models occurring sources or sinks.

For different choices of D, a and f there are various special cases of the advection-diffusion equation. In this work we will only consider the case of coefficients that are dependent on the spacial position, from which follows that equation (2.1) is a linear PDE. Furthermore we assume that the coefficients are smooth on the domain and that $D = \varepsilon \cdot \mathbb{I}$ for some constant $\varepsilon \geq 0$ and the identity operator $\mathbb{I}$.

2.1.1 Steady-State

In the remainder of this work we will only consider stationary problems. Once a state of equilibrium is reached, there are no further changes in time. This corresponds to a time derivative equal to zero, hence the instationary equation (2.1) becomes

$$\nabla \cdot (-D\nabla u + au) = f \tag{2.2}$$

which is called the steady-state advection-diffusion equation. If $D = \mathbb{I}$ and $a = 0$ equation (2.2) becomes the Poisson equation

$$-\nabla^2 u = f \tag{2.3}$$

where ∇^2 is the Laplace operator. This is a second-order elliptic PDE. If $D = 0$ equation (2.2) becomes the stationary advection equation

$$\nabla \cdot (au) = f \tag{2.4}$$

which is a hyperbolic PDE. These three equations will be used as model problems for the numerical experiments.

2.1.2 Boundary Value Problem

The function u shall now be restricted to an open connected domain $\Omega \subseteq \mathbb{R}^d$. From here on out $\Omega := (0,1)^d$, that is the open d-dimensional unit hypercube. In order to get a unique solution additional constraints are enforced on the domain's boundary $\partial\Omega = \overline{\Omega} \setminus \Omega$, which is the closure of Ω without its interior points.

The boundary conditions of interest in this work are Dirichlet boundary conditions. These are defined on $\partial\Omega$ and constrain the function value of u. This is called a boundary value problem and a function

$$u \in C^2(\Omega) \cap C^0(\overline{\Omega})$$

that satisfies

$$\nabla \cdot (-D\nabla u + au) = f \quad \text{in } \Omega \tag{2.5a}$$

$$u = g \quad \text{on } \partial\Omega \tag{2.5b}$$

in a pointwise sense is called its classical solution.

Other kinds of boundary conditions like Neumann and outflow boundary conditions were not further investigated in this work.

2.2 Finite Volume Element Method

This section gives a brief view on the construction of a FVE method on rectangular grids. A more in-depth analysis can be found in the dissertation of Schmidt [34].

Now that the continuous problem has been formulated, it must be discretized. For this the FVE method is used, which is a mix of the classical finite volume (FV) method and the finite element (FE) method. There are mainly two reasons to choose the FVE method.

1. For advection dominated flow problems (i.e. with a high Péclet number) the standard Galerkin FE method becomes unstable [35].
2. With the FVE method the DOFs are located at the vertices of the grid, which is where they are needed for the combination technique. This is not the case for the classical FV method where the DOFs are located at the cell centers.

Arguably a finite difference method would also have been a good choice as only structured grids are used. However it might be easier to extend the FVE implementation to a higher order scheme in the future (cf. [28], section 3.2.2).

2.2.1 Control Volumes

First we need some definitions to describe the occurring regular full grids. Let $\vec{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ be a d-dimensional multi-index, with operators on multi-indices being applied element-wise. Let there also be special multi-indices for constant values like $\vec{2} = (2, \ldots, 2)$. Then $\Omega_{\vec{l}}$ denotes the regular grid on $\overline{\Omega}$ with spacing $h_{\vec{l}} := (h_{l_1}, \ldots, h_{l_d}) := 2^{-\vec{l}} = (2^{-l_1}, \ldots, 2^{-l_d})$ and level $\vec{l}$. Furthermore we need two kinds of entities of the grid $\Omega_{\vec{l}}$

$$E^0_{\vec{l}} := \{E^0_{\vec{l},\vec{i}} \in \overline{\Omega} \ : \ \vec{0} \leq \vec{i} \leq 2^{\vec{l}}\}, \qquad E^0_{\vec{l},\vec{i}} := (h_{l_1} \cdot i_1, \ldots, h_{l_d} \cdot i_d) \tag{2.6a}$$

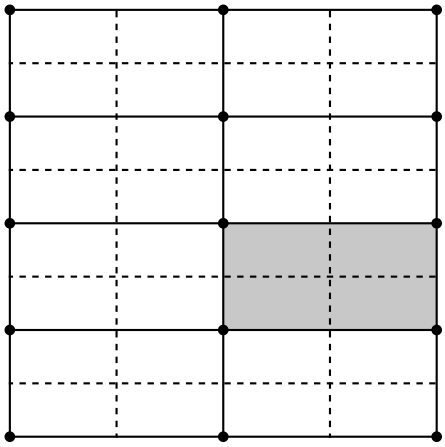
$$E^d_{\vec{l}} := \{E^d_{\vec{l},\vec{i}} \subseteq \Omega \ : \ \vec{0} \leq \vec{i} < 2^{\vec{l}}\}, \qquad E^d_{\vec{l},\vec{i}} := \prod_{1 \leq j \leq d}]h_{l_j} \cdot i_j, h_{l_j} \cdot (i_j + 1)[\tag{2.6b}$$

where (2.6a) denotes the vertices—or entities of dimensionality 0—and (2.6b) denotes the cells—or entities of dimesionality d (see fig. 2.1a). The number of interior vertices is denoted by $N_{\vec{l}} = |E^0_{\vec{l}} \setminus \partial\Omega|$.

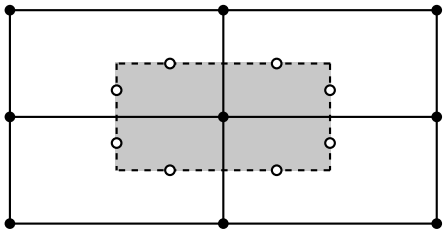
One essential feature of all finite volume methods is that they operate on conservation laws. These describe processes where some quantity—like the mass of the material in the example model of section 2.1—is conserved in an isolated system. As a first step of the discretization the domain Ω is partitioned into a finite set of control volumes $B_{\vec{l}}$. A local conservation law of equation (2.5a) can be obtained by integrating over each control volume and applying the Gauss Divergence Theorem

$$\int_{\partial B} (-D\nabla u + au) \cdot n \, ds = \int_B f \, dx, \quad \forall B \in B_{\vec{l}} \tag{2.7}$$

where n is the unit outer normal vector of the boundary ∂B. This equation describes a balance between the flux over the control volume boundary on the left hand side and the sources inside the control volume on the right hand side.



(a) Full grid $\Omega_{(1,2)}$ (solid lines), its dual grid (dashed lines), its vertices $E^0_{(1,2)}$ (black dots), and cell $E^d_{(1,2),(1,1)}$ (gray)



(b) One control volume (gray) and the integration points for the surface integrals (white dots)

Figure 2.1: A full grid and its dual in 2D.

There are a few possible choices for how to partition the domain, but certain conditions must hold (cf. [34], section 3.1). In this work we choose the control volumes as the centered boxes, which form a dual grid. This dual grid is constructed by connecting the centroid of every cell to the centroids of all faces on its boundary, see figure 2.1a for the 2-dimensional case. Formally we can define these control volumes as

$$B_{\vec{l}} := \{B_{\vec{l},\vec{i}} \subseteq \Omega \ : \ \vec{0} \leq \vec{i} \leq 2^{\vec{l}}\}, \qquad B_{\vec{l},\vec{i}} := \Omega \cap \prod_{1 \leq j \leq d}]h_{l_j} \cdot (i_j - \tfrac{1}{2}), h_{l_j} \cdot (i_j + \tfrac{1}{2})[$$

which is essentially just a translation of the cells $E^d_{\vec{l}}$ by half the grid spacing in all directions (except for the boundary of the domain). Each vertex $E^0_{\vec{l},\vec{i}}$ of the primal grid is now associated with one control volume $B_{\vec{l},\vec{i}}$ which surrounds it. Also for every vertex there is one balance equation in (2.7) which must be fulfilled.

2.2.2 Finite Element Space

Note that the integral formulation of equation (2.7) weakened the requirements for the differentiability of the solution function u compared to equation (2.2). The next step in the discretization process is to determine the flux values at the boundaries of the control volumes in equation (2.7). This is done by a finite element approximation. We first define a suitable space of trial functions

$$U_{\vec{l}} = \{u \in C^0(\overline{\Omega}) \ : \ \forall E \in E^d_{\vec{l}} : u|_E \in Q_1, u|_{\partial\Omega} = 0\} \subset H^1_0(\Omega)$$

where Q_1 is the set of polynomials of degree less or equal to 1 in each variable and $H^1_0(\Omega)$ is the Sobolev space $W^{1,2}(\Omega)$ with zero trace. $U_{\vec{l}}$ is a space of piecewise

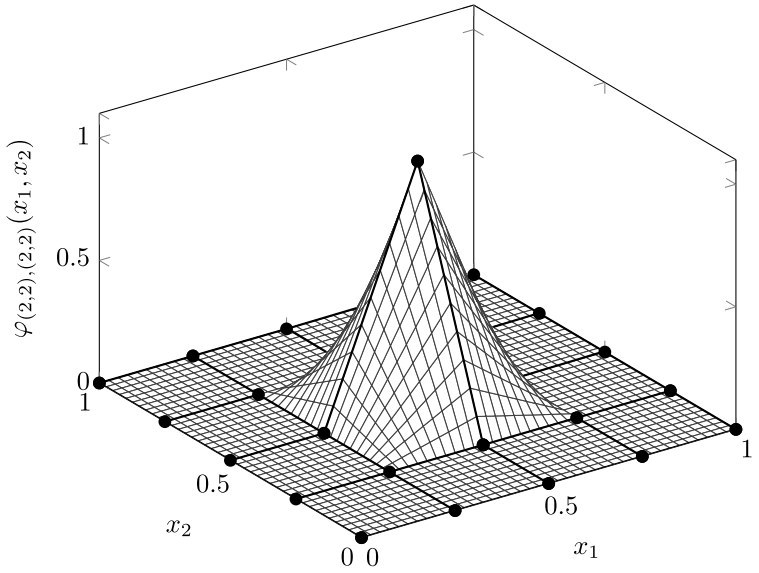


Figure 2.2: 2D hat function $\varphi_{(2,2),(2,2)}$ on the grid $\Omega_{(2,2)}$

d-linear functions that vanish on the boundary. It is a finite-dimensional vector space that has the d-dimensional hat functions $\varphi_{\vec{l},\vec{i}}$ (cf. [32], section 2.1.3) on the interior nodes as a basis (see fig. 2.2).

By now approximating u with a function $u_{\vec{l}} \in U_{\vec{l}}$ in equation (2.7) it becomes

$$\int_{\partial B} (-D\nabla u_{\vec{l}} + a u_{\vec{l}}) \cdot n \; ds = \int_B f \; dx, \quad \forall B \in B_{\vec{l}}. \tag{2.8}$$

If $u_{\vec{l}}$ is represented by its basis functions

$$u_{\vec{l}} = \sum_{\vec{0} < \vec{i} < 2^{\vec{l}}} \mathbf{u}_{m(\vec{i})} \, \varphi_{\vec{l},\vec{i}}$$

with $\mathbf{u} \in \mathbb{R}^{N_{\vec{l}}}$ and m as some mapping to a linear index, this is an algebraic finite system of equations.

One essential component that is still missing is the treatment of advection dominated problems. For this we separately choose the function value $u_{\vec{l}}$ on the control volume boundary for the advective term $a u_{\vec{l}}$ such that the value of the upwind node is chosen (cf. [4], section 2.7). The result of this is a first order scheme with respect to the advective flow. It is also possible to construct an upwind scheme by choosing the control volumes problem dependent [25], which becomes interesting once higher order schemes are used. This was however not further considered in this work.

It is also possible to derive this method from a FE method standpoint alone, by using the characteristic functions of the control volumes as a test space. This is then classified as a nonconforming Petrov-Galerkin method [34].

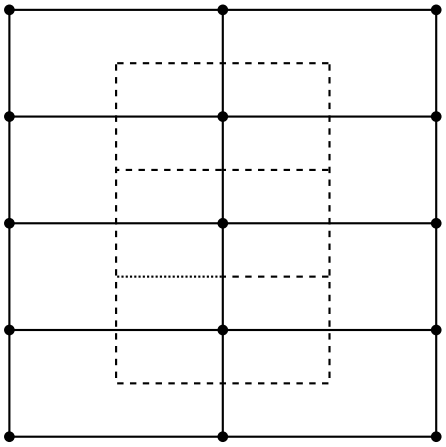


Figure 2.3: Full grid $\Omega_{(1,2)}$ (solid lines), its dual grid (dashed lines) for Dirichlet boundary conditions and the dual grid skeleton entity $\gamma_{(0,1),(1,1),(1,2)}$ (dotted line).

2.2.3 Boundary Conditions

Until now we disregarded the boundary conditions required by equation (2.5b). As for Dirichlet boundary conditions the function value is given by the function g, the corresponding equations and control volumes can be eliminated from the system of equations (2.8) (cf. [4], section 2.7). We call $\tilde{B}_{\vec{l}} \subset B_{\vec{l}}$ the set of control volumes not corresponding to a node on the Dirichlet boundary.

Furthermore we now choose $u_{\vec{l}} \in w + U_{\vec{l}}$ where $w \in H^1(\Omega)$ with $w|_{\partial\Omega} = g$ and

$$w + U_{\vec{l}} = \{u \in H^1(\Omega) \ : \ u = w + v, \ v \in U_{\vec{l}}\}$$

This way the function $u_{\vec{l}}$ can have the prescribed values at the boundary.

As a side note Neumann boundary conditions can also be easily enforced by keeping the node as a DOF but prescribing the flux over the boundary with the given value.

2.2.4 Assembly

The last step of the discretization is to actually assemble the system of equations. Because the implementation expects the problem to be formulated as a weighted residual method, we write the residual form $r_{\vec{l},\vec{i}}$ of equations (2.8) as

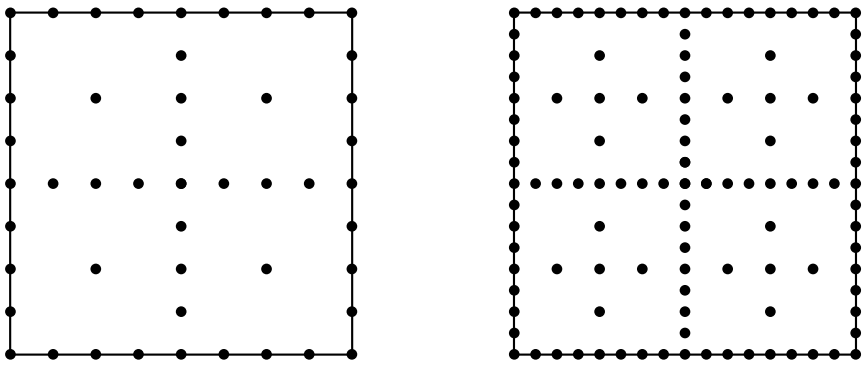
$$r_{\vec{l},\vec{i}}(u_{\vec{l}}) = \int_{\partial\tilde{B}_{\vec{l},\vec{i}}} (-D\nabla u_{\vec{l}} + a u_{\vec{l}}) \cdot n \ ds - \int_{\tilde{B}_{\vec{l},\vec{i}}} f \ dx.$$

Let us also define the map $R : \mathbb{R}^{N_{\vec{l}}} \to \mathbb{R}^{N_{\vec{l}}}$ with

$$(R(\mathbf{u}))_{m(\vec{i})} = r_{\vec{l},\vec{i}}(u_{\vec{l}}).$$

So the problem now reads (cf. [20], section 1.3)

$$\text{find } \mathbf{u} \in \mathbb{R}^{N_{\vec{l}}} \text{ s.t. } R(\mathbf{u}) = 0.$$



(a) (b)

Figure 2.4: 2D regular sparse grid points with level 3 (a) and level 4 (b).

To solve this problem the residual $R(\mathbf{u})$ and its jacobian matrix $\mathbf{A} = \nabla R(\mathbf{u})$ must be assembled.

To this end we define an additional subset of the domain Ω, the skeleton of the dual grid (see fig. 2.3). It consists of the common border entities of the control volumes inside a primal grid cell and can be defined as

$$\Gamma_{\vec{l}} = \{\gamma_{\vec{i},\vec{j},\vec{k}} \ : \ E^d_{\vec{l},\vec{i}} \cap \partial \tilde{B}_{\vec{l},\vec{j}} \cap \partial B_{\vec{l},\vec{k}} \cap \Omega\}.$$

With this the residual form $r_{\vec{l},\vec{i}}$ corresponding to node $E^0_{\vec{l},\vec{i}}$ can be calculated by adding up the contributions from all adjacent grid cells

$$r_{\vec{l},\vec{i}}(u_{\vec{l}}) = \sum_{\vec{j} \,:\, \partial E^d_{\vec{l},\vec{j}} \cap E^0_{\vec{l},\vec{i}} \neq \emptyset} \left(\sum_{\gamma \in \Gamma_{\vec{l}} \cap E^d_{\vec{l},\vec{j}} \cap \partial \tilde{B}_{\vec{l},\vec{i}}} \int_\gamma (-D \nabla u_{\vec{l}} + a u_{\vec{l}}) \cdot n \; ds \right) - \int_{\tilde{B}_{\vec{l},\vec{i}}} f \; dx. \tag{2.9}$$

The surface and volume integrals in equation (2.9) can be approximated by the midpoint rule [34] (see fig. 2.1b).

2.3 Sparse Grids

For an introduction to sparse grid methods we refer to the dissertation of Pflüger [32]. In the following the basic ideas of sparse grids are given.

Sparse Grid methods start off with the hierarchical basis representation of the function space $U_{\vec{l}}$. From there only the hierarchical subspaces are chosen, which contribute most to the interpolation $u_{\vec{l}} \in U_{\vec{l}}$ of a function u for a some norm.

For an isotropic grid with spacing h this procedure reduces the number of points from $\mathcal{O}(h^{-d})$ to $\mathcal{O}(h^{-1}(\log h^{-1})^{d-1})$ (see fig. 2.4). If the function u is smooth enough the approximation error only deteriorates from $\mathcal{O}(h^2)$ to $\mathcal{O}(h^2(\log h^{-1})^{d-1})$ [32].

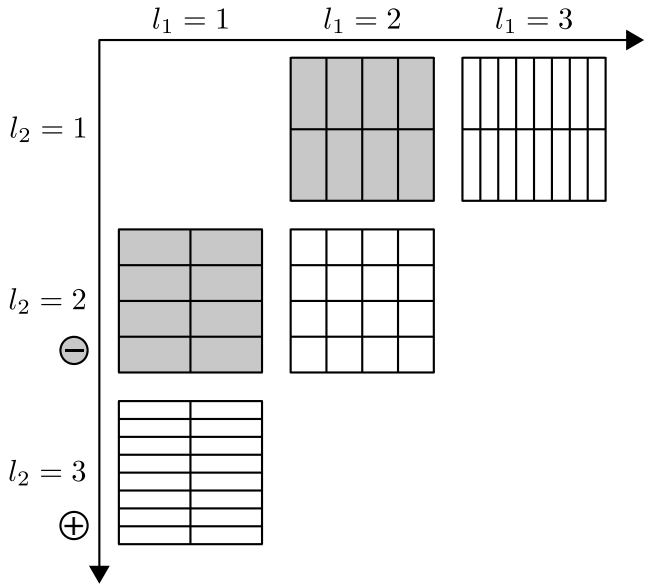


Figure 2.5: Combination technique for 2D regular sparse grid with level 3.

2.3.1 Combination Technique

Rather than using the hierarchical basis the combination technique uses the function subspaces on coarse but anisotropic full grids. The functions on these component grids are then linearly combined to form a solution in the sparse grid space (see fig 2.5). So the solution for the combination technique is defined as (cf. [10], section 3)

$$u_n^c = \sum_{i=n}^{n+d-1} (-1)^{i+1} \binom{d-1}{i-n} \sum_{|\vec{l}|_1 = i} u_{\vec{l}}$$

Where $|\cdot|_1$ is the sum of all elements of a multi-index, and n is the level of the regular sparse grid space that is utilized.

For the interpolation of a function the combination technique solution and the sparse grid solution are identical, however when solving PDEs the solutions may differ [32].

By using the combination technique the total amount of DOFs is higher than with hierarchical sparse grids, but the benefit is that the component grid problems are independent from each other. Therefore they can be solved in parallel and communication is only needed in the combination step, when the solution function u_n^c is evaluated.

Chapter 3

Multigrid Methods

This chapter will give a short introduction to multigrid methods. For a more thorough explanation of the topic we refer to the well known books *A Multigrid Tutorial* [14] by Briggs and *Multigrid* [37] by Trottenberg, Osterlee and Schüller. We follow up on the results of Zubair et al. [10] for the use of multigrid algorithms with the combination technique.

From the discretization we now have a residual $R(\mathbf{u}^{(0)})$ for some initial guess $\mathbf{u}^{(0)}$ and the jacobian matrix $\mathbf{A}$. These can be used to calculate a function $u_{\vec{l}}$ that fulfills equation (2.8)

$$\mathbf{u} = \mathbf{u}^{(0)} - \underbrace{\mathbf{A}^{-1} R(\mathbf{u}^{(0)})}_{=\mathbf{e}}.$$

The question that still needs to be answered is how to solve the system of linear equations $\mathbf{Ae} = R(\mathbf{u}^{(0)})$. By using standard iterative methods like Gauss-Seidel or Jacobi solvers the convergence rate quickly deteriorates on finer grids. To get convergence rates independent of the grid spacing so-called multigrid methods must be employed.

3.1 Coarse Grid Correction

There are two observations that lead to multigrid methods. The first one is, that for some problems certain iterative solver can smooth out the error (i.e. reduce the high frequency error components) after a few iterations. This process is called relaxation. The second observation is that the error on a fine grid appears more oscillatory on a coarser grid [14].

In a coarse grid correction scheme both of these observations are used. First smooth out the error on the fine grid, then restrict the residual to a coarser grid. On the coarse grid solve the defect equation, then prolongate the error back to the fine grid. With that the solution on the fine grid can be corrected.

3.2 Multiplicative Multigrid

The previous idea can be used in a recursive way, instead of solving the problem on the coarse grid. For this we use a sequence of grids

$$\left(\Omega_{c^i(\vec{l_f})}\right)_{i=0,\dots,k} = \left(\Omega_{\vec{l_f}},\ \Omega_{c(\vec{l_f})},\ \Omega_{c^2(\vec{l_f})}, \dots, \Omega_{c^k(\vec{l_f})}\right)$$

where c is some function that maps $\vec{l}$ to a level such that $c(\vec{l}) < \vec{l}$, the level $\vec{l_f}$ is that of the finest grid and k is the maximum depth of the grid hierarchy. For each grid we need an operator $\mathbf{A}_{\vec{l}}$ analogous to the operator $\mathbf{A}$ from the discretization on the fine grid. Furthermore there is a need for the prolongation $\mathbb{I}^{\vec{l}}_{c(\vec{l})}$ and restriction $\mathbb{I}^{c(\vec{l})}_{\vec{l}}$ operations. With these components the classical multiplicative multigrid solver is then defined like in the following.

Function MG($\mathbf{v}_{\vec{l}}$, $\mathbf{b}_{\vec{l}}$)

Data: $\left(\mathbf{A}_{c^i(\vec{l_f})}\right)_{i=0,\dots,k}$
if $\vec{l} = c^k(\vec{l_f})$ **then** /* coarsest grid reached */
  return $\mathbf{A}^{-1}_{\vec{l}}\mathbf{v}_{\vec{l}}$;
else
  for $i \leftarrow 1$ **to** α_1 **do** /* pre-smoothing */
    $\mathbf{v}_{\vec{l}} \leftarrow$ smooth $\mathbf{A}_{\vec{l}}\,\mathbf{v}_{\vec{l}} = \mathbf{b}_{\vec{l}}$;
  end
  $\mathbf{b}_{c(\vec{l})} \leftarrow \mathbb{I}^{c(\vec{l})}_{\vec{l}}(\mathbf{b}_{\vec{l}} - \mathbf{A}_{\vec{l}}\,\mathbf{v}_{\vec{l}})$; /* restrict residual */
  $\mathbf{v}_{c(\vec{l})} \leftarrow 0$;
  for $i \leftarrow 1$ **to** β **do**
    $\mathbf{v}_{c(\vec{l})} \leftarrow \mathrm{MG}(\mathbf{v}_{c(\vec{l})}, \mathbf{b}_{c(\vec{l})})$; /* recursion */
  end
  $\mathbf{v}_{\vec{l}} \leftarrow \mathbf{v}_{\vec{l}} + \mathbb{I}^{\vec{l}}_{c(\vec{l})}\mathbf{v}_{c(\vec{l})}$; /* coarse grid correction */
  for $i \leftarrow 1$ **to** α_2 **do** /* post-smoothing */
    $\mathbf{v}_{\vec{l}} \leftarrow$ smooth $\mathbf{A}_{\vec{l}}\,\mathbf{v}_{\vec{l}} = \mathbf{b}_{\vec{l}}$;
  end
  return $\mathbf{v}_{\vec{l}}$;
end

This function is then called iteratively in the assignment

$$\mathbf{u}^{(i+1)} \leftarrow \mathbf{u}^{(i)} - \mathrm{MG}(0,\, R(\mathbf{u}^{(i)}))$$

until a stopping criterion is met (e.g residual was reduced by a certain factor).

As we can see there are a number of components that must be chosen. In fact it is critical to the performance of the method to choose the correct components for a given problem. In the following we take a look at how the components were chosen for this work.

3.2.1 Coarse Grid Operator

The operator $\mathbf{A}_{\vec{l}}$ is obtained by simply discretizing the problem on $\Omega_{\vec{l}}$ the same way as on the fine grid. This is called the direct coarse approximation (DCA).

There is also the possibility to construct an operator for the coarse grid in a purely algebraic way by using the prolongation and restriction operators. This so-called Galerkin coarse approximation (GCA) has some advantages over a DCA, and can lead to efficient multigrid methods for first order upwind discretizations [19]. There are however unresolved issues when constructing a GCA for higher dimensions [10] so the DCA is the preferred choice.

3.2.2 Grid Transfer Function

As the model problem is a second order PDE, d-linear interpolation is chosen as the prolongation operator (cf. [37], section 2.7.1). For the restriction operator the transpose of the prolongation operator is used.

3.2.3 Smoother

The choice for the smoother is not trivial once an upstream discretization is used as a simple ω-Jacobi method is not viable (cf. [37], section 7.2.1). If the velocity vector field is constant, downstream relaxation—where the order of visited grid points corresponds to the characteristics of the problem—is a good choice. This is however to complicated once variable coefficients and parallelism occur [37].

One approach is to start a Gauss-Seidel sweep from every corner of the grid [37], but for higher dimensions this results in an exponential amount of sweeps.

Another alternative are alternating line smoothers [37]. For higher dimensions these become hyper-plane smoothers. They are more complicated to implement and were analyzed for the sequential case by Reisinger et al. [33] for the combination technique. It is not clear how well they would scale in a parallel setting.

As there seems to be no practical solution for this problem a symmetric Gauss-Seidel point smoother is used. This results in at least 2 flow directions which retain good smoothing qualities, but for some special cases the method might not converge.

3.2.4 Schedules

The multigrid schedule is described by the α_1, α_2 and β parameter in the MG function definition.

For different values of β the traversal order of grid hierarchy changes. Setting $\beta = 1$ results in the V-cycle (see fig. 3.1 left), setting $\beta = 2$ results in the W-cycle (see fig. 3.1 right). Both of these are viable options but the V-cycle is expected to show a better weak scalability (cf. [37], section 6.3.3).

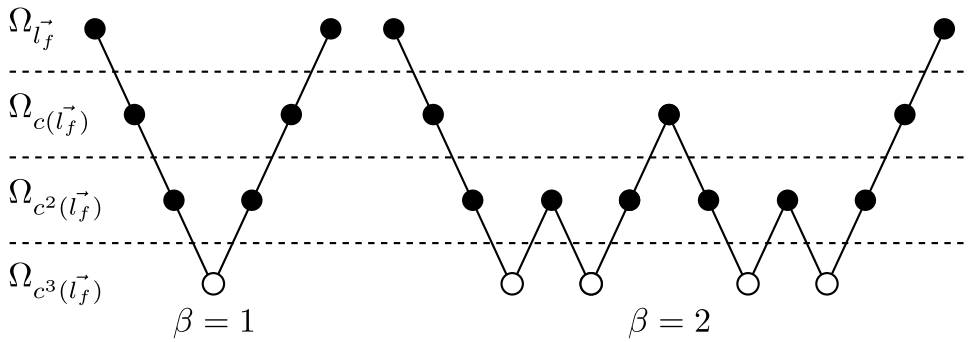


Figure 3.1: Multigrid schedules for a grid hierarchy of depth 3. This shows restriction and prolongation operations (lines), the smoothing steps (black dots) and the coarse grid solution steps (white dots).

i	$c^i(\vec{l}_f)$	
	strategy 1	strategy 2
0	(6,2,5,3)	(6,2,5,3)
1	(5,2,5,3)	(5,2,5,3)
2	(4,2,4,3)	(3,2,3,3)
3	(3,2,3,3)	(2,2,2,2)
4	(2,2,2,2)	(1,1,1,1)
5	(1,1,1,1)	

Table 3.1: Coarsening strategies for $\vec{l}_f = (6, 2, 5, 3)$

The α_1 and α_2 parameters decide how many pre- and post-smoothing steps are used. A schedule will be called $V(\alpha_1, \alpha_2)$ for a V-cycle and $W(\alpha_1, \alpha_2)$ for a W-cycle with given smoothing iteration count.

3.3 Coarsening Strategy

The coarsening strategy has to deal with the anisotropy in multiple dimensions, which is given by the component grids of the combination technique. As only isotropic diffusion coefficients are considered the task is simplified because the strategy can be chosen independent from the problem.

We will test both of the proposed strategies from Zubair et al. [11]. Strategy 1 partially coarsens along the dimensions with the strongest coupling by doubling the grid spacing. Once an equidistant grid is reached it proceeds with standard coarsening. This strategy is also called standard refinement by Sprengel [36]. Strategy 2 is similar to strategy 1, but instead of doubling the grid spacing in the partial coarsening phase it quadruples the grid spacing (see fig. 3.2). Table 3.1 shows an example for both strategies.

According to Zubair et al. [10] strategy 1 shows good convergence behavior even if the relaxation parameter of the smoother was not chosen optimally. On

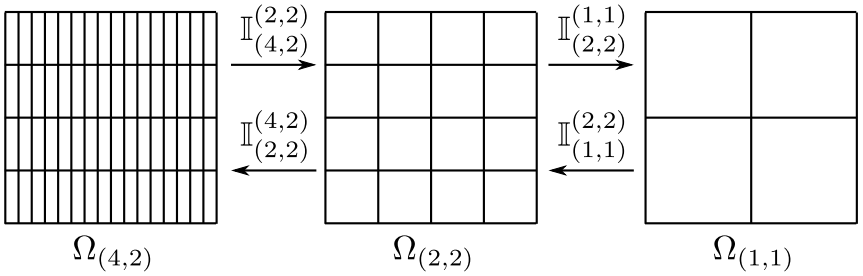


Figure 3.2: Grid hierarchy for strategy 2.

the other hand strategy 2 is more dependent on the smoothing quality but can offers a speedup over strategy 1.

3.4 Parallelization

There are two level of parallelization involved in solving the problem. The first level of parallelization comes from the combination technique, because the problems on the component grids can be solved independent from each other. The second level of parallelization comes from the parallel solution on the component grids. This is done with a domain decomposition approach. The parts of the algorithm that must be parallelized are the smoother method and the transfer operators.

The grids $\Omega_{\vec{l}}$ of the hierarchy are now decomposed into a number of rectangular subgrids. Each of these subgrids has a layer of overlapping cells with its neighboring subgrids. The grid hierarchy is truncated at the smallest level where there are more cells than the number of processes p involved in the parallelization

$$p > |E^d_{c^k(\vec{l_f})}|.$$

This way it is not possible that processors run idle while coarser grids are traversed. For the coarse grid solver the Bi-CGSTAB method was chosen, which can handle the non-symmetric matrix resulting from the upwind discretization.

The symmetric Gauss-Seidel smoother is inherently sequential, because every step in the calculation uses data which was updated in a previous step. To run it in parallel the additive Schwarz method is used. Here the processes first apply the smoother independent from each other on their own subgrid. After that the values from the overlapping region are updated by adding up the solutions from all processes it is contained in.

The grid transfer operators are trivially parallelizable as they can be carried out locally on the subgrids. However for the restriction of the residual there is communication required, because on the overlap boundary not all the necessary information is available (cf. [5], section 6.4).

Chapter 4

Implementation

This chapter is about the implementation details of this work. At first the libraries which were used are presented. After that the implementation for each of the goals is explained.

4.1 DUNE Project

The Distributed and Unified Numerics Environment (DUNE) is a toolbox for solving PDEs with grid-based methods. It is modular by design which makes it easy to implement further functionality. Its main goals are a separation of data structures and algorithms by abstract interfaces, an efficient implementation of these interfaces by generic programming, and reuse of existing simulation packages [1]. It is written in modern C++ and makes heavy use of static polymorphism through templates for more compile-time optimizations. This enables the use of abstract interfaces without the runtime overhead this typically entails.

The code of this work will be combined in a module with the title DUNE-MG. It is compatible to DUNE 2.4, which is the stable version at the time of writing. The overall dependency relations are shown in Figure 4.1.

4.1.1 Core Modules

The core modules are the basic framework which all other modules can use. In the following the most relevant of them are described in more detail.

DUNE-Common

The Common module provides classes for the basic infrastructure, like exception handling. Among other helper classes it also contains classes for dense vectors and matrices.

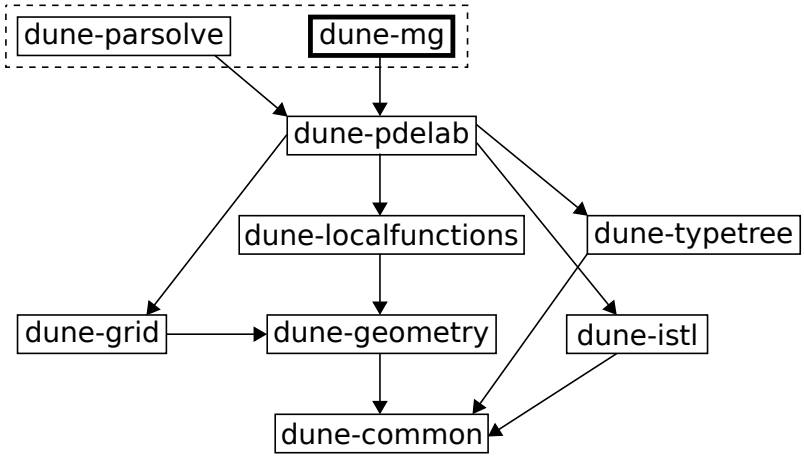


Figure 4.1: Dependency graph of the DUNE modules

DUNE-Grid

The Grid module [7, 6] defines the interface which every grid implementation must adhere to. Due to this abstraction it is possible to exchange the underlying grid implementation if it supports all the needed features. In DUNE a grid does not actually hold any data, but provides just the topological information.

DUNE-Geometry

In the Geometry module everything related to the reference elements is defined. This module is used in the implementation of the discretization method.

DUNE-ISTL

The Iterative Solver Template Library (ISTL) [13] contains classes for sparse vectors and matrices. It also provides implementations for Krylov Subspace methods and an aggregation-based algebraic multigrid preconditioner. These will be used as a comparison to the geometric multigrid method of this work.

4.1.2 External Modules

There are two external modules on which the implementation depends. In the following they are described in detail.

DUNE-PDELab

The PDELab module [8] provides the framework for discretizing PDEs. The finite volume element method implemented in this work, uses the interfaces of PDELab. It also provides access to the solvers defined in ISTL, to solve the discretized systems.

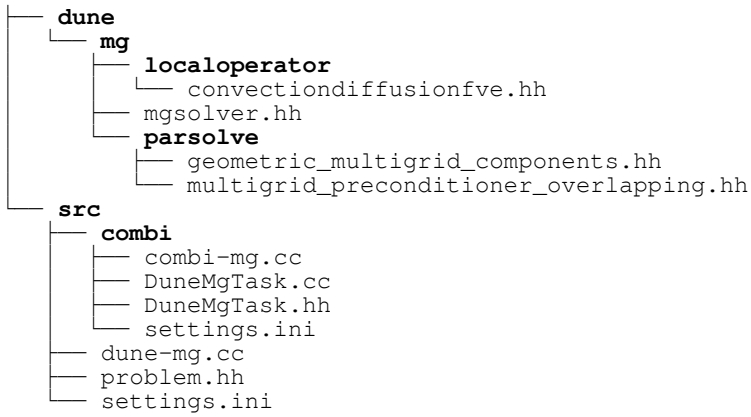


Figure 4.2: The most important files in the DUNE-MG module.

DUNE-Parsolve

The Parsolve module is part of a lecture at the University of Heidelberg [5]. In it the basic geometric multigrid method is implemented, as described in the function MG from section 3.2.

4.2 SG++ Distributed Combigrid

The SG++ toolbox [32] drastically reduces the amount of work required to use sparse grids. It is mainly concerned with adaptive sparse grids, but it also contains a module for the combination technique.

The SG++ Distributed Combigrid module is based on the manager-worker pattern. A manager process distributes the computational tasks—corresponding to the component grids—to groups of worker processes. These worker process groups then solve their tasks in parallel. The implementation of the task interface is up to the user.

4.3 Implementation

A part of the file tree of the DUNE-MG module can be seen in figure 4.2. The discretization implementation is inside the file `convectiondiffusionfve.hh`. The multigrid solver extensions are in the file `mgsolver.hh`. Files taken from the parsolve module are in the directory `parsolve`.

The `src` directory contains the driver code to test the implementation. Tests for the full grid are implemented in the file `dune-mg.cc`. In the `problem.hh` file the definition of the model problems are contained. Tests for the combination technique are implemented in the files from the `combi` directory. The `settings.ini` files allow to change some of the parameters without recompiling.

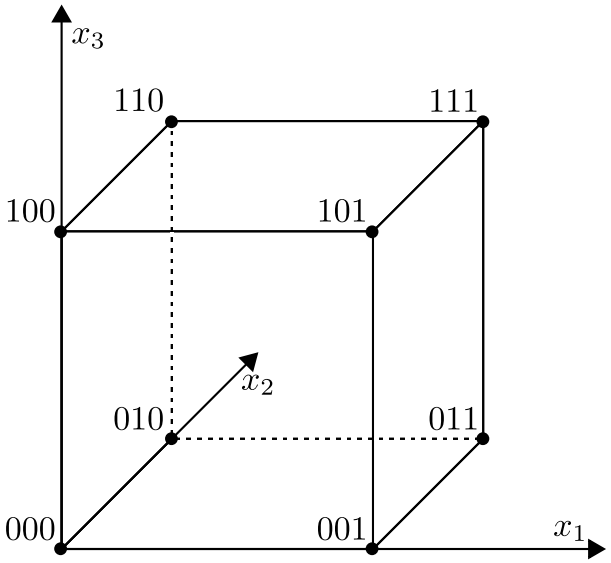


Figure 4.3: 3D reference element with binary corner indices.

4.3.1 Finite Volume Element Method

The theoretical background for the FVE method was already covered in section 2.2, so we now take a look on how this carries over into actual code.

It is clear that we somehow need to construct the dual grid, to get the geometry information for the control volumes. This construction only has to be done once on the reference element. Because a rectangular grid is used, the reference element is the unit hypercube. The corners are enumerated in row-major order as seen in figure 4.3 for the 3-dimensional case.

In the following an algorithm is presented that constructs the neighbor relation between every corner of the reference element, where two corners are neighbors if they are connected by an edge and the second corner has a larger index number. This is equivalent to constructing the hypercube graph with directed edges.

Algorithm 1: Calculate the neighbor relation on the reference element.

Input: dimension D
Output: neighbor relation $\mathcal{N}$
$\mathcal{N} \leftarrow \emptyset$;
for $i \leftarrow 0$ **to** $2^D - 1$ **do**
  for $d \leftarrow 0$ **to** $D - 1$ **do**
    $j \leftarrow i \veebar 2^d$; `/* bitwise xor */`
    if $i > j$ **then**
      $\mathcal{N} \leftarrow \mathcal{N} \cup (j, i)$;
    end
  end
end

The neighbor relation can now be used to find the control volume boundaries inside the reference element. For every neighboring corner pair there is exactly one control volume boundary. The asymmetry of the neighbor relation is necessary for the direction of the outer normal vector of the associated boundary. To help find the integration points, outer normal vectors and inside- and outside-cells of these boundaries the `StaticRefinement` class from DUNE-Geometry is used. This results in the skeleton information of the dual grid, that is utilized to calculate the surface integrals.

Local Operator

The DUNE-PDELab interface for discretization methods requires the residual form (2.9) to be split into three parts:

$$r_{\vec{l},\vec{i}}(u_{\vec{l}}) = \alpha^{\text{vol}}_{\vec{l},\vec{i}}(u_{\vec{l}}) + \alpha^{\text{skel}}_{\vec{l},\vec{i}}(u_{\vec{l}}) + \alpha^{\text{bnd}}_{\vec{l},\vec{i}}(u_{\vec{l}})$$

During the assembly process the local contribution to the residual is calculated. Each of the summands corresponds to the iteration over certain grid entities. In the local operator implementation each of the summands is calculated in a method with the fitting name (e.g. `ConvectionDiffusionFVE::alpha_volume` for $\alpha^{\text{vol}}_{\vec{l},\vec{i}}$).

All the calculations are done on the reference element, therefore a transformation between the actual element and the reference element is needed. As we are using axis aligned rectangular grids this only involves a translation and scaling. The inverse and jacobian determinant of this transformation is trivial to calculate and supplied by the geometry implementation of DUNE.

For the FVE method only $\alpha^{\text{vol}}_{\vec{l},\vec{i}}$ and $\alpha^{\text{bnd}}_{\vec{l},\vec{i}}$ are relevant, which are calculated while iterating over the grid cells and the domain boundary faces. In case there are only Dirichlet boundary conditions $\alpha^{\text{bnd}}_{\vec{l},\vec{i}} = 0$ holds, so the residual can be calculated in one iteration over the grid cells. For every visited cell the contributions to the residual form from (2.9) are added up.

If Neumann or outflow boundary conditions are used $\alpha^{\text{bnd}}_{\vec{l},\vec{i}}$ does no longer necessarily vanish, because these would become natural boundary conditions in this FVE method. The implementation of the discretization does support homogeneous Neumann boundary conditions.

The jacobian of the residual can be explicitly programmed or calculated numerically. In this work it was explicitly programmed.

For future work we already implemented a temporal local operator, which can be used to solve the instationary advection-diffusion equation.

4.3.2 Multigrid Method

The first change for the multigrid solver was to wrap it inside a class conforming to the PDELab interface for linear solvers. This way it can be used just like any other already existing solver in PDELab.

The extensions made in this work required changes in the code of DUNE-Parsolve. The necessary code of it is therefore integrated in the DUNE-MG module.

Coarsening Strategy

The major change is the implementation of the coarsening strategy described in section 3.3. The DUNE grid interface supports hierarchically nested grids, which was originally used by Parsolve to construct the operator and prolongation hierarchies. One starts with the coarsest grid space and builds up the grid hierarchy by refining until the finest grid is reached.

The DUNE-Grid module comes with an implementation of its interface called YaspGrid. It implements a parallel regular grid with arbitrary dimensionality. One of its shortcomings is however that it does not support anisotropic refinement, which is necessary for the partial coarsening strategy. To circumvent this problem a wrapper class was written that stores a separate grid for every level in the hierarchy. As grids only contain the topology information this is not much worse than using the built-in hierarchy capabilities.

This is implemented in the `GridHierarchy` class. It specifically wraps around YaspGrid by accessing the implementation through the experimental grid extensions configuration flag. This is required for the methods `GridHierarchy::father` and `GridHierarchy::geometryInFather` which return the father entity and the geometry relative to the father entity. These are normally implemented in the YaspEntity class, and are needed to build the prolongation hierarchy.

There is an external grid manager that supports anisotropic refinement called SPGrid [30]. However in contrast to YaspGrid it does not support a user defined partitioning. This is important for the recombination step in the combination technique once instationary problems are considered, as the partitioning must be the same on every component grid. Also it is not possible to restrict the size of the partition overlap in the refinement process.

4.3.3 Combigrid Task Class

To use the SG^{++} Distributed Combigrid module a task class must be implemented. In the `DuneMgTask::init` method the DUNE specific objects are initialized with the information supplied by the Task class (e.g. local MPI communicator, component grid level, ...). In the `DuneMgTask::run` method the solver is applied to the problem. For a detailed understanding of the driver code we refer to the DUNE-PDELab howto [20].

To further process the solution data it must be transferred to a DistributedFullGrid data structure from SG^{++} Distributed Combigrid. There is still a problem left for future work once time dependent problems are a concern. The order of the ranks in the partitioning of YaspGrid is row-major, but for the DistributedFullGrid it is column-major. So one of the implementations must be changed or otherwise

almost the entire volume of the domain has to be sent through the network, in each recombination step.

Chapter 5

Numerical Experiments

In this chapter the implementation is tested. First the model problems which will be used in the tests are defined. After that the discretization error is examined in multiple dimensions. Lastly the multigrid methods are compared with respect to their parallel efficiency and convergence rate.

5.1 Model Problems

There are four special cases of equation (2.2) which will be considered as model problems. Three of them are defined for the 2-dimensional case and one is defined for arbitrary dimensionality.

5.1.1 Problem 1 (Diffusive Flow)

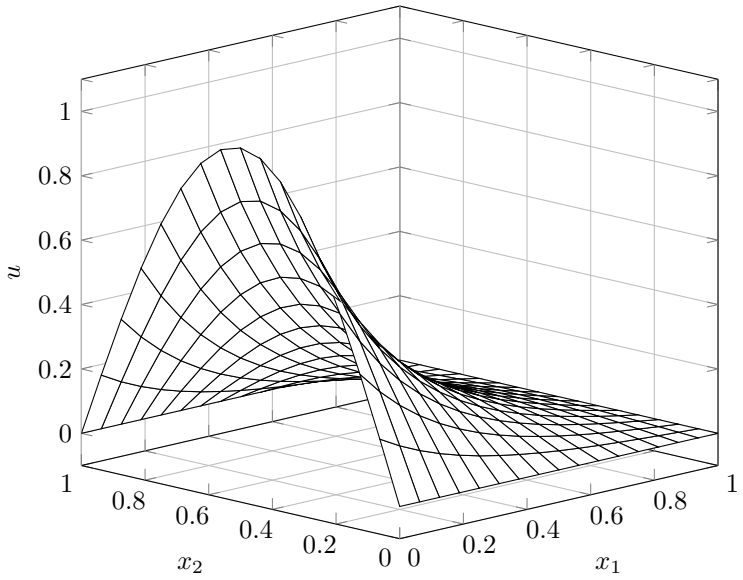
The first class of problems are special cases of the Poisson equation (2.3).

(a) The first problem of this class has $f = 0$, which is also known as Laplace equation. The boundary conditions are chosen as $u(0, x_2) = \sin(\pi x_2)$ and zero otherwise. The exact solution for this problem (cf. [22], see fig. 5.1a) is

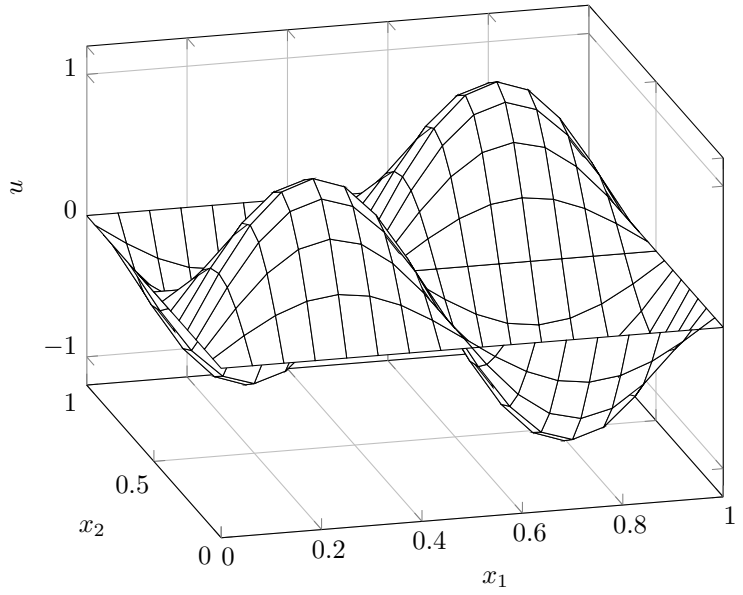
$$u(x_1, x_2) = \sin(\pi x_2) \frac{\sinh(\pi(1 - x_1))}{\sinh(\pi)}$$

(b) The second problem of this class has $f = 4\pi^2 d \prod_{i=0}^{d} \sin(2\pi x_i)$ and homogeneous Dirichlet boundary conditions. It can easily be checked that the exact solution (see fig. 5.1b) is

$$u(x) = \prod_{i=0}^{d} \sin(2\pi x_i)$$

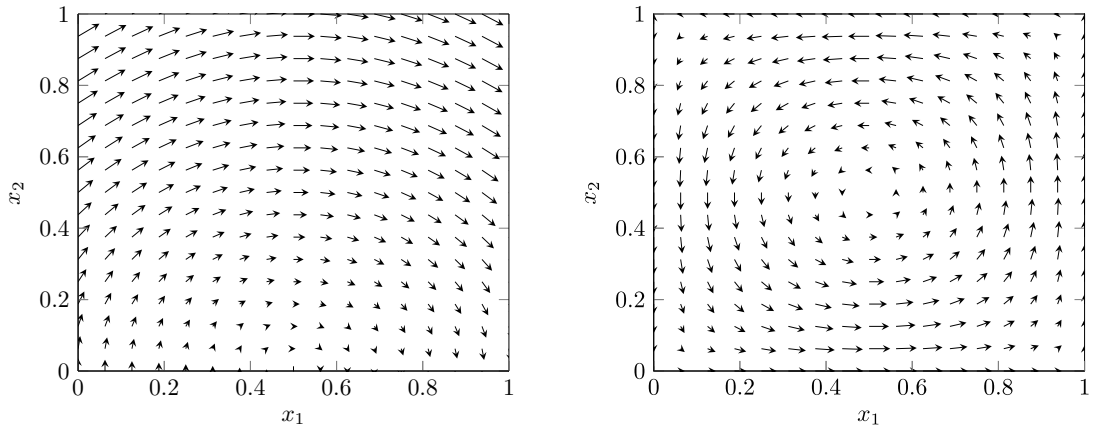


(a) Problem 1a.



(b) Problem 1b with $d = 2$.

Figure 5.1: Graph of solution functions.



(a) Problem 2. (b) Problem 3.

Figure 5.2: Advective flow.

5.1.2 Problem 2 (Advective Flow)

The second problem is a special case of the stationary advection equation (2.4). It has $f = 0$ and $a = (\sin(\frac{\pi}{2}x_2), \cos(\frac{\pi}{2}(x_1 + \frac{1}{2})))^T$ which is a circular flow around the point $(\frac{1}{2}, 0)$ (see fig. 5.2a). The boundary conditions are chosen as $u(x_1, 0) = 1$ and zero otherwise. As the characteristics form concentric half circles around the inflow boundary one finds the solution to be

$$u(x_1, x_2) = \begin{cases} 1 & \text{if } (x_1 - \frac{1}{2})^2 + x_2^2 \leq \frac{1}{4} \\ 0 & \text{otherwise.} \end{cases}$$

5.1.3 Problem 3 (Recirculating Flow)

The last problem is a special case of the stationary advection-diffusion equation (2.2). It has $f = 0$, $D = \varepsilon \cdot \mathbb{I}$, $\varepsilon > 0$ and $a = (\sin(\pi x_1)\cos(\pi x_2), -\cos(\pi x_1)\sin(\pi x_2))^T$ which is a recirculating flow (see fig. 5.2b). The boundary conditions are chosen as $u(0, x_2) = 1$ and zero otherwise.

5.2 Discretization Error

Before we can compare the multigrid methods we must first verify that the implemented discretization method is correct. As formal verification is not an option here, we are testing a number of model problems to see if implausible results might occur.

To measure the discretization error the discrete L_2-norm is defined as

$$\|\mathbf{x}\|_2 = \frac{\sqrt{\mathbf{x}^T\mathbf{x}}}{N_{\vec{l}}}, \quad \mathbf{x} \in \mathbb{R}^{N_{\vec{l}}}$$

$\vec{l}$	L_2 discretization error		
	Problem 1a	Problem 1b ($d = 4$)	Problem 2
$\vec{2}$	1.95016e-03	9.50285e-02	1.42581e-01
$\vec{3}$	1.77864e-04	2.15153e-03	4.35312e-02
$\vec{4}$	1.93067e-05	9.23531e-05	1.58520e-02
$\vec{5}$	2.26806e-06		6.21729e-03
$\vec{6}$	2.79756e-07		2.53737e-03
$\vec{7}$	3.71901e-08		1.05137e-03
$\vec{8}$	6.27906e-09		4.41520e-04
$\vec{9}$	1.80573e-09		1.87906e-04

Table 5.1: Discretization error on full grids.

$\vec{l}$	L_2 discretization error		
	Problem 1a	Problem 1b ($d = 4$)	Problem 2
$\vec{2}$	3.94774e-03	4.93827e-02	1.74777e-01
$\vec{3}$	5.26787e-04	6.66389e-03	6.10977e-02
$\vec{4}$	7.54388e-05	1.26420e-03	2.38143e-02
$\vec{5}$	1.10713e-05	1.19301e-04	9.70069e-03
$\vec{6}$	1.62448e-06		4.07973e-03
$\vec{7}$	2.35407e-07		1.76295e-03
$\vec{8}$	3.41297e-08		7.76766e-04
$\vec{9}$	4.30903e-09		3.46877e-04

Table 5.2: Discretization error with the combination technique solution interpolated to a full grid.

on the grid $\Omega_{\vec{l}}$ (cf. [22]) which approximates the continuous L_2-norm. The discretization error is then calculated by subtracting the exact solution from the sufficiently accurate numerical solution at the grid points and taking the discrete L_2-norm.

The result for full grids can be seen in table 5.1. The expected error bound of $\mathcal{O}(h^2)$ [29] is valid for problems 1a and 1b. Also as expected for problem 2 the error is bounded by $\mathcal{O}(h)$ because first-order upwinding was used.

It is also important to verify that the model problems converge with the combination technique. The results can be seen in table 5.2. As we can see all the error bounds hold, and problem 2 converges even though its solution function is discontinuous.

5.3 Multigrid Comparison

So now that we know the discretization works as expected, the next step is to analyze the multigrid methods. In this section variations of the geometric multigrid solver are compared to a Bi-CGSTAB solver preconditioned by the algebraic multigrid method from DUNE-ISTL. The stopping criterion for all solvers is a reduction of the residual $R(\mathbf{u}^{(0)})$ by a factor of 10^{-6} in the L_2-norm. The initial guess $\mathbf{u}^{(0)}$ is always zero.

We call the variations of the GMG solver as follows (cf. section 3.2.4):

- $V_i(1,1)$ uses the V-cycle schedule with coarsening strategy $i \in \{1,2\}$.
- $W_i(1,1)$ uses the W-cycle schedule with coarsening strategy $i \in \{1,2\}$.

The algebraic multigrid method is denoted by its abbreviation AMG. It is treated as a black box solver as provided in PDELab with the default parameters set from `DUNE::PDELab::ISTLBackend_BCGS_AMG_SSOR`.

5.3.1 Convergence Behavior

In the following we will take a look at how each method is affected under the influence of changing grid anisotropy and advection dominance. To measure the influence we use the average convergence rate defined as (cf. [33])

$$\bar{\rho} = \sqrt[k]{\frac{\|R(\mathbf{u}^{(k)})\|_2}{\|R(\mathbf{u}^{(0)})\|_2}}.$$

All tests in this section are carried out with the sequential multigrid algorithm. Because of the additive Schwarz domain decomposition the parallel algorithm is not numerically identical. The convergence in the parallel version will be slower than in the sequential version.

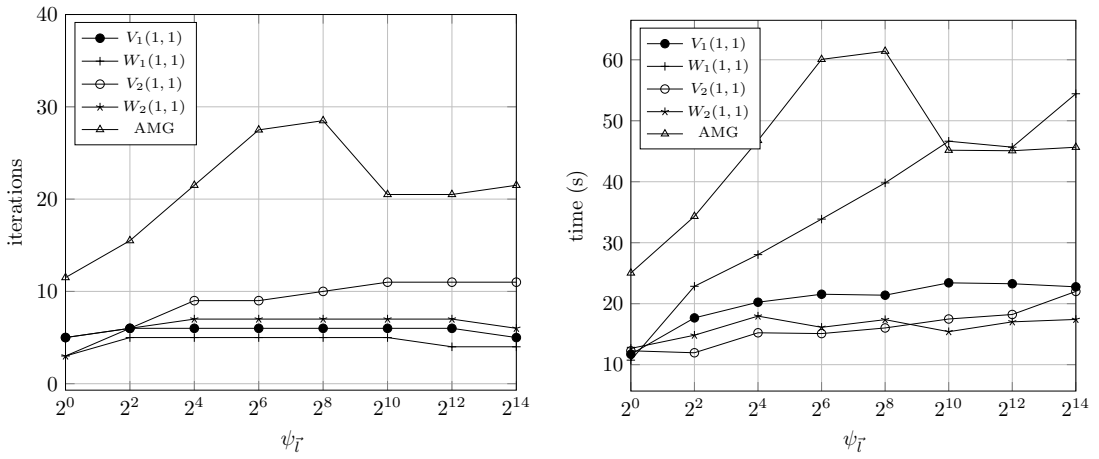
For more experiments on the convergence behavior of sequential multigrid methods in this context we refer to the detailed surveys of Zubair et al. [10, 11].

Grid Anisotropy

To investigate the effect of anisotropy we look at the relation between convergence rate and the aspect ratio of the grid. We define the aspect ratio as the quotient of the largest cell extent divided by the smallest cell extent parallel to the axes

$$\psi_{\vec{l}} = \frac{\max(2^{\vec{l}})}{\min(2^{\vec{l}})}.$$

To mask out the effects of advection we use model problem 1b with $d = 2$. We consider a sequence of grids where the level of the first dimension is halved in



(a) Iterations until convergence.

(b) Wall-clock time.

Figure 5.3: Behavior of multigrid methods for for problem 1b with $d = 2$ and different aspect ratios.

every step, whereas the level of the second dimension is doubled. The first grid in this sequence is the isotropic grid $\Omega_{(8,8)}$, so the whole sequence reads

$$(\Omega_{(8,8)}, \Omega_{(7,9)}, \ldots, \Omega_{(1,15)}).$$

The number of DOFs is reduced in every step due to a growing number of points on the Dirichlet boundary. It should be noted that the aspect ratio quadruples in every step. It should also be noted that these are component grids for the combination technique in 2 dimensions, if the sparse grid is of level 15 or 16.

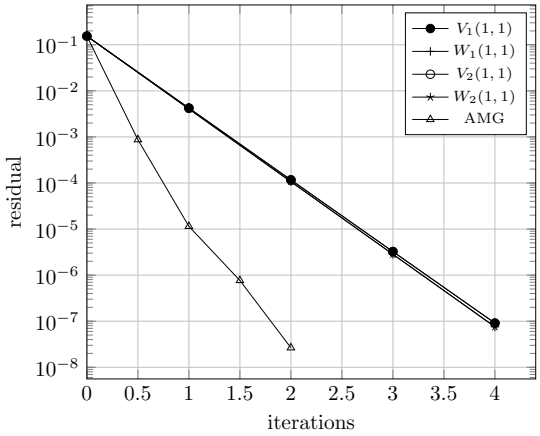
The result can be seen in figure 5.3a. With exception of $V_2(1,1)$ all GMG methods approximately retain a convergence rate independent of the aspect ratio. The influence of the aspect ratio is most obvious for the AMG method which converges the slowest on grid $\Omega_{(4,12)}$.

Next we look at the effect of anisotropy in higher dimensions, where they can occur in multiple directions. For this we solve problem 2 in 4 dimensions. Figure 5.4 shows the convergence history of an isotropic grid, a grid with anisotropy in multiple direction and a grid where there is a strong anisotropy in one direction.

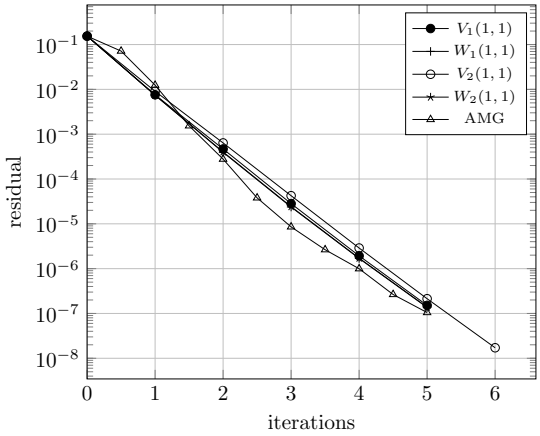
For the isotropic grid $\Omega_{(4,4,4,4)}$ with $\psi_{(4,4,4,4)} = 1$ (see fig. 5.4a) the AMG method shows the best convergence ratio. The GMG variations behave basically the same, with only marginal differences between the schedules. As expected the coarsening strategies have no influence as only full coarsening occurs.

For the anisotropic grid $\Omega_{(6,2,5,3)}$ with $\psi_{(6,2,5,3)} = 2^4$ (see fig. 5.4b) the GMG methods behave well, but for AMG the number of iteration almost tripled compared to the isotropic grid.

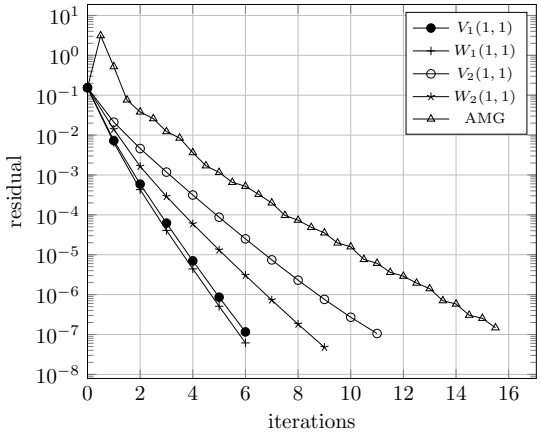
With the anisotropic grid $\Omega_{(10,2,2,2)}$ with $\psi_{(10,2,2,2)} = 2^8$ (see fig. 5.4c) there is a drastic increase in the number of iterations—almost by a factor of 8—for the AMG method. The GMG solvers fair better with the anisotropy especially



(a) On grid $\Omega_{(4,4,4,4)}$.



(b) On grid $\Omega_{(6,2,5,3)}$.



(c) On grid $\Omega_{(10,2,2,2)}$.

Figure 5.4: Convergence history for problem 1b with $d = 4$.

the ones using strategy 1. These results suggest that the AMG method does not converge independent of grid spacing if anisotropy grows.

Advection Dominance

To measure the influence of the advective flow we solve problem 3 with differing values for ε on the isotropic grid $\Omega_{(8,8)}$. Because the coarsening strategies are the same for isotropic grids, we only test one method for each schedule. This problem is well known to be hard to solve with multigrid methods as solver (cf. [37], section 7.8.1). Therefore we will additionally use the GMG methods as a preconditioner for the Bi-CGSTAB solver. These are then denoted by $\tilde{V}_1(1,1)$ and $\tilde{W}_1(1,1)$.

The results can be seen in table 5.3. The convergence rate for the GMG solvers deteriorates quickly. As expected, using the multigrid methods as a preconditioner leads to a significant improvement. The W-cycle turns out to be the best method and the V-cycle is comparable to AMG when used as a preconditioner. However

ε	$\bar{\rho}$ (#iterations)				
	$V_1(1,1)$	$\tilde{V}_1(1,1)$	$W_1(1,1)$	$\tilde{W}_1(1,1)$	AMG
1e-00	0.0260 (4)	0.0001 (2)	0.0266 (4)	0.0001 (2)	0.1366 (7.5)
1e-01	0.0459 (5)	0.0008 (2)	0.0267 (4)	0.0001 (2)	0.1555 (7.5)
1e-02	0.3399 (13)	0.0304 (4.5)	0.0280 (4)	0.0002 (2)	0.4802 (19.5)
1e-03	0.7909 (59)	0.2358 (10.5)	0.1351 (7)	0.0156 (3.5)	0.3871 (15)
1e-04	0.9350 (206)	0.4379 (17.5)	0.5418 (23)	0.1260 (7.5)	0.4747 (20.5)
1e-05		0.5467 (23)	0.7483 (48)	0.2058 (9)	0.4569 (19.5)
1e-06		0.5302 (22)	0.8231 (71)	0.2274 (11.5)	0.4805 (19)

Table 5.3: Multigrid convergence rate for problem 3.

we do not expect mesh independent convergence for any of our methods (cf. [37], table 7.15).

5.3.2 Running Time

The calculations in the following were done on a shared memory system with 8 Intel® Xeon® CPU E7-8880 v3 and 515 871MB RAM. With this system there are 144 available virtual cores.

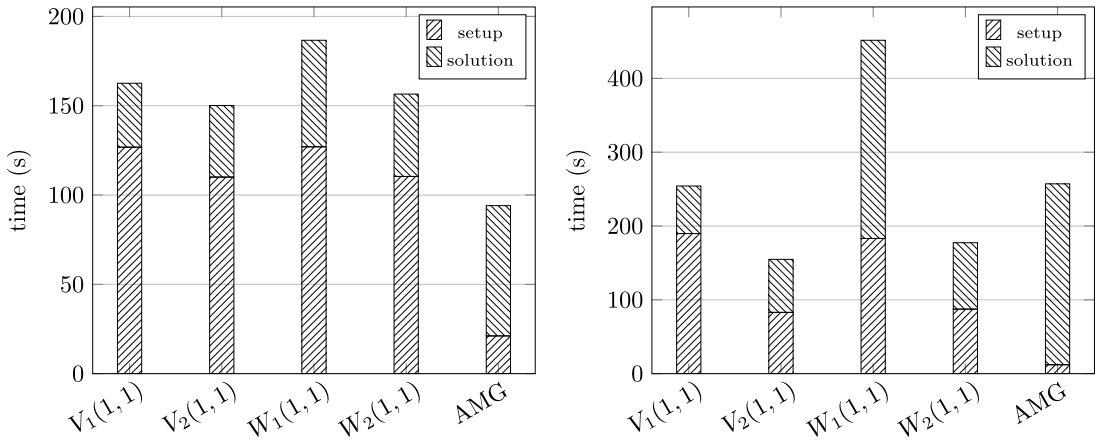
As we have seen above, the GMG variants show a better convergence behavior than the AMG method. However in figure 5.3b one can see that $W_1(1,1)$ is the slowest method on high aspect ratios even though it has the best convergence rate. This is shows that $W_1(1,1)$ has a significantly higher cost per multigrid cycle than all the other methods for high aspect ratios. So one can not only judge the methods by their convergence rate.

For multigrid methods the running time is the sum of a setup phase and a solving phase. In the setup phase the coarse grid operators and grid transfer functions are prepared, as these do not change during the solution phase. We now take a look at the time needed for the different phases in the experiments corresponding to figure 5.4 with the sequential multigrid methods.

The results are shown in figure 5.5. Like we would expect the GMG methods using strategy 1 have a longer setup time than for strategy 2 because their grid hierarchy is deeper. This is the reason for the speed-up that can be observed from using strategy 2. We can also see that the AMG method uses most of its time for the solution phase.

The results from this section suggest the following conclusions

- V-cycles are faster on high aspect ratio grids whereas W-cycles are faster on low aspect ratio grids.
- Methods using strategy 2 retain their running time better for varying aspect ratios than methods using strategy 1.
- The AMG method and the $W_1(1,1)$ GMG method are significantly slower on grids with high aspect ratio.



(a) On grid $\Omega_{(6,2,5,3)}$.

(b) On grid $\Omega_{(10,2,2,2)}$.

Figure 5.5: Wall-clock time for problem 1b with $d = 4$.

With this knowledge we can now leave the sequential version behind and take a look at the parallel version.

5.3.3 Parallel Scaling

The challenge for tomorrow's exascale computing is to solve larger and larger problems, instead of solving today's problems faster. So when we look at the parallel multgrid methods we are most interested in how they behave if the processor count is grows with the problem size. This is called the weak scaling and to measure it we define the sizeup as (cf. [26], chapter 10)

$$\text{sizeup}(N, p) = \frac{N(p)}{N(1)} \cdot \frac{T_{\text{seq}}(N(1))}{T_{\text{par}}(N(p), p)}$$

where N is the problem size dependent on the number of processes, T_{seq} is the running time for the sequential algorithm and T_{par} is the running time for the parallel algorithm. The weak scaling efficiency then reads (cf. [26], chapter 10)

$$E(N, p) = \frac{\text{sizeup}(N, p)}{p}.$$

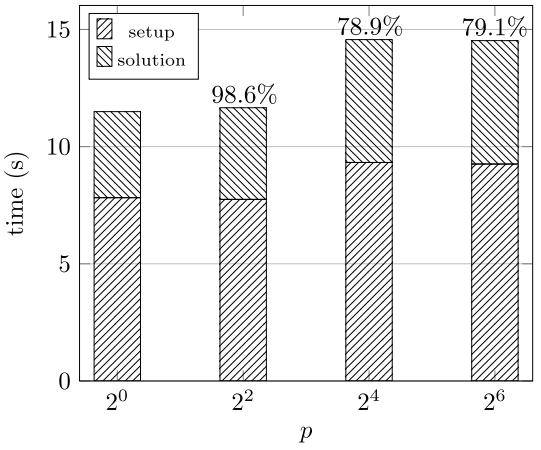
Due to the restriction in available processing cores we only do a small weak scaling study for the 2 dimensional case. We solve problem 1b again for 2 dimensions with an isotropic and an anisotropic test case. The sequential problems have the level $(8, 8)$ and $(15, 1)$ respectively and the grid spacing is doubled in each direction for the next larger problem, so the aspect ratio is conserved. Furthermore we choose p proportional to the problem size. We check the scaling for at most 64 processes, therefore the largest problem has 4 194 304 DOFs on the fine grid with 65 536 DOFs per process. In the previous sections we have seen that the

AMG method and the $W_1(1,1)$ GMG variation are insufficiently efficient in the anisotropic case, so they are not further considered.

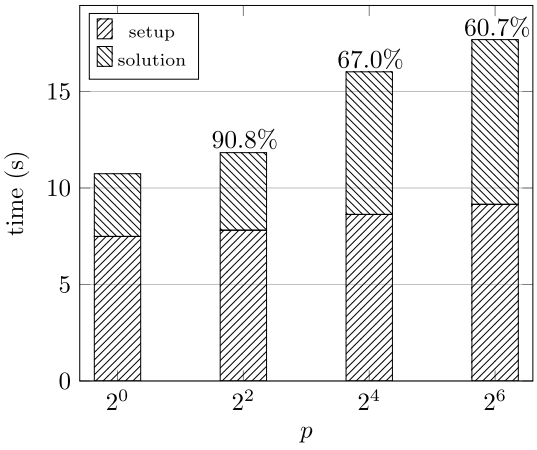
From figure 5.6 we can see that for isotropic grids the $V_1(1,1)$ method scales up best. Like we expected in section 3.2.4, V-cycles scale better than W-cycles. The AMG method has the worst scaling behavior of the three. The results for the AMG method approximately accord to the results from Blatt [12] table 5.2, where the weak scalability for a Poisson problem discretized with the FE method is shown.

For the anisotropic case the picture looks worse as can be seen in figure 5.7. All the methods are already rather inefficient on 4 processes and continue to get worse. The most likely reason for this seems to be the employed smoothing method which is known to cause bad scaling behavior (cf. [18], section 3.3.2).

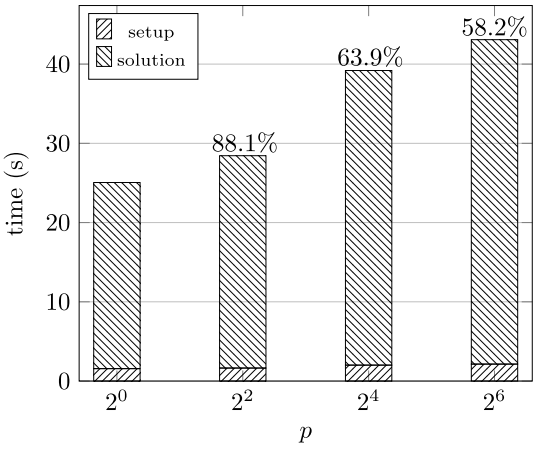
In both cases the major sequential time component comes from the solution phase. To isolate the cause one would have to make additional measurements of the individual multigrid components.



(a) $V_1(1,1)$ method.

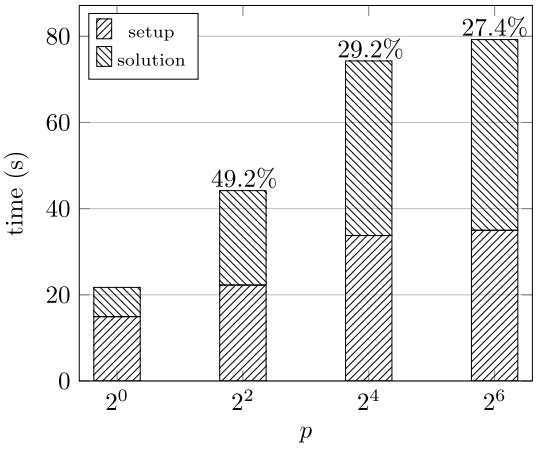


(b) $W_2(1,1)$ method.

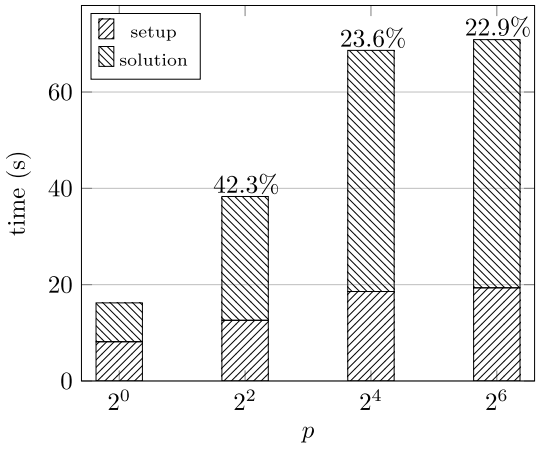


(c) AMG method.

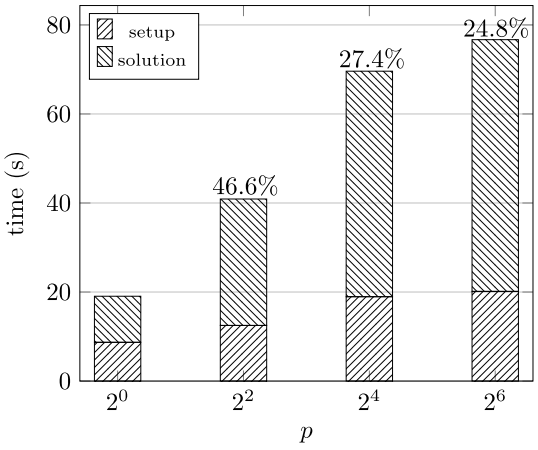
Figure 5.6: Weak scalability for problem 1b with $d = 2$ and $\psi_{\bar{l}} = 1$. The weak scaling efficiency is given as percentage above the bars.



(a) $V_1(1,1)$ method.



(b) $W_2(1,1)$ method.



(c) $V_2(1,1)$ method.

Figure 5.7: Weak scalability for problem 1b with $d = 2$ and $\psi_{\vec{l}} = 2^{14}$. The weak scaling efficiency is given as percentage above the bars.

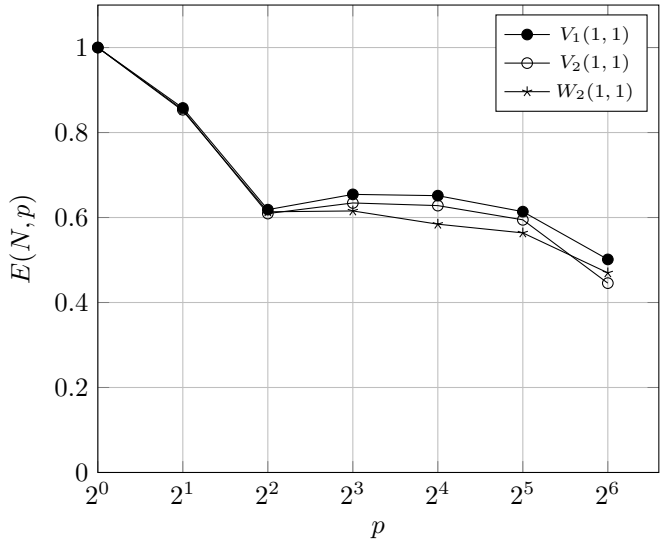


Figure 5.8: Strong scalability for problem 1b with $d = 2$.

5.4 Combination Technique

The GMG methods are superior to the AMG method for the model problems, but until there is a solver that scales up better for high aspect ratios, we deem it not sensible to conduct massively parallel experiments.

Nevertheless we want to demonstrate that the implementation works. We take a look at the running time for a fixed problem with a growing number of processes. This is called strong scaling and we define it through the speedup measure (cf. [26], chapter 9)

$$\text{speedup}(N,p) = \frac{T_{\text{seq}}(N)}{T_{\text{par}}(N,p)}.$$

The strong scaling efficiency is then defined analogous to the weak scaling efficiency (cf. [26], chapter 9)

$$E(N,p) = \frac{\text{speedup}(N,p)}{p}.$$

We use the same computing system and model problem as in the weak scaling study. The solution lies in the 2-dimensional sparse grid space of level 15 with a minimum component grid level of 2, therefore there are 27 component grids in the combination technique.

Because there are two levels of parallelization it is not clear how to distribute the growing amount of processes to these levels. We decide to first increase the number of processes in the domain decomposition up to 4, and then to increase the number of processing groups in the combination technique up to 16, so that we can observe the effects of both levels. This way we use 64 processes in total and take no advantage of the virtual cores, because these would probably distort the results. Keep in mind that we only measure the solution process and not the combination of the component grids.

The result is shown in figure 5.8 and we can clearly see the different effects of the level of parallelization. In the beginning the efficiency quickly drops most likely due to the inefficient domain decomposition on small grids. When we start to add new process groups to the combination technique the efficiency stops dropping because there are enough component grids available so no group runs idle. Once the number of process groups approaches the amount of component grids the processor load drops and with it the efficiency.

Chapter 6

Conclusion

In this work we explored the possibility to solve the higher dimensional advection-diffusion equation with a parallel multigrid method on anisotropic grids. This was done through experimentation on different model problems.

To do these experiments a suitable discretization method was implemented using DUNE-PDELab. Furthermore a parallel geometric multigrid method was extended to handle the difficulties of the problem. Both of these implementations were then used with the sparse grid combination technique using the SG^{++} Distributed Combigrid module.

Different variations for the geometric multigrid method were compared to each other, together with an algebraic multigrid method. The results of the experiments suggest that the $V_1(1,1)$, $V_2(1,1)$ and $W_2(1,1)$ variations respectively are the best choice for certain problems. The V-cycles show better scalability than the W-cycles for both low and high aspect ratios. The W-cycles are generally better for advection dominant equations if they are used as a precondtioner for Bi-CGSTAB. The algebraic multigrid method was, apart from some isolated cases, worse then the geometric multigrid methods. However this should come as no surprise because only structured grids were involved.

Even though some improvements were made in comparison to the previously available solvers, none of these multigrid methods is adequate for massively parallel experiments yet, due to their low efficiency. Future research is required to find suitable multigrid components for upwind discretizations on higher-dimensional parallel anisotropic grids.

Bibliography

[1] Dune project description. `https://dune-project.org/dune.html`. Accessed: 2016-05-02.

[2] B. R. Baliga and S. V. Patankar. A new finite-element formulation for convection-diffusion problems. In *Numerical Heat Transfer*, volume 3, pages 393–409, 1980.

[3] R. E. Bank and D. J. Rose. Some error estimates for the box method. In *SIAM Journal on Numerical Analysis*, volume 24, pages 777–787, 1987.

[4] P. Bastian. *Parallele adaptive Mehrgitterverfahren*. PhD thesis, Universität Heidelberg, 1994.

[5] P. Bastian. Lecture notes on parallel solution of large sparse linear systems, 2015.

[6] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, R. Kornhuber, M. Ohlberger, and O. Sander. A generic grid interface for adaptive and parallel scientific computing. part ii: Implementation and tests in dune, 2007.

[7] P. Bastian, M. Blatt, A. Dedner, C. Engwer, R. Klöfkorn, M. Ohlberger, and O. Sander. A generic grid interface for parallel and adaptive scientific computing. part i: Abstract framework, 2007.

[8] P. Bastian, F. Heimann, and S. Marnach. Generic implementation of the finite element methods in the distributed and unified numerics environment (dune). In *Kybernetika*, volume 46, pages 294–315, 2010.

[9] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[10] H. bin Zubair and C. W. Oosterlee. Multigrid method for high dimensional elliptic equations. In *European Conference on Computational Fluid Dynamics*, 2006.

[11] H. bin Zubair, C. W. Oosterlee, and R. Wienands. Multigrid for high-dimensional elliptic partial differential equations on non-equidistant grids. In *SIAM Journal on Scientific Computing*, volume 14, pages 178–194, 2007.

[12] M. Blatt. *A Parallel Algebraic Multigrid Method for Elliptic Problems with Highly Discontinuous Coefficients*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2010.

[13] M. Blatt and P. Bastian. The iterative solver template library. In *Lecture Notes in Scientific Computing*, volume 4699, pages 666–675. Springer, 2007.

[14] W. L. Briggs. *A Multigrid Tutorial*. SIAM, 2000.

[15] H.-J. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*. PhD thesis, Technische Universität München, 1992.

[16] Z. Cai. On the finite volume element method. In *Numerische Mathematik*, volume 58, pages 713–735, 1991.

[17] S.-H. Chou and D. Y. Kwak. Multigrid algorithms for a vertex–centered covolume method for elliptic problems. In *Numerische Mathematik*, volume 90, pages 441–458, 2002.

[18] E. Chow, R. D. Falgout, J. J. Hu, R. S. Tuminaro, and U. M. Yang. A survey of parallelization techniques for multigrid solvers, 2006.

[19] P. M. de Zeeuw. Matrix-dependent prolongations and restrictions in a black-box multigrid solver. In *Journal of Computational and Applied Mathematics*, volume 33, pages 1–27, 1990.

[20] Dune Team. dune-pdelab howto, 2014.

[21] M. Griebel. A parallelizable and vectorizable multi-level algorithm on sparse grids. In *Parallel Algorithms for Partial Differential Equations*, volume 31, pages 94–100. Vieweg-Verlag, 1991.

[22] M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In *Iterative Methods in Linear Algebra*, pages 263–281. North Holland, 1992.

[23] W. Hackbusch. On first and second order box schemes. In *Computing*, volume 41, pages 277–296, 1989.

[24] P. Hupp, R. Jacob, M. Heene, D. Pflüger, and M. Hegland. Global communication schemes for the numerical solution of high-dimensional pdes. In *Advances in Parallel Computing*, volume 52, pages 564–573, 2014.

[25] L. Ju, L. Tian, X. Xiao, and W. Zhao. Covolume-upwind finite volume approximations for linear elliptic partial differential equations. In *Journal of Computational Physics*, volume 231, pages 6097–6120, 2012.

[26] A. Kaminisky. *BIG CPU, BIG DATA Solving the World's Toughest Computational Problems with Parallel Computing*. 2015.

[27] T. Lin and X. Ye. A posteriori error estimates for finite volume method based on bilinear trial functions for the elliptic equation. In *Journal of Computational and Applied Mathematics*, 2013.

[28] Y. Lin, J. Liu, and M. Yang. Finite volume element methods: An overview on recent developments. In *Numerical Analysis and Modeling*, volume 4, pages 14–34, 2013.

[29] J. Lv and Y. Li. L2 error estimate of the finite volume element methods on quadrilateral meshes. In *Advances in Computational Mathematics*, volume 33, pages 129–148, 2010.

[30] M. Nolte. *Efficient Numerical Approximation of the Effective Hamiltonian*. PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau, 2011.

[31] C. Parkash and S. V. Patankar. A control volume-based finite-element method for solving the navier stokes equations using equal-order velocity-pressure interpolation. In *Numerical Heat Transfer*, volume 8, pages 259–280, 1985.

[32] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, Aug. 2010.

[33] C. Reisinger and G. Wittum. On multigrid for anisotropic equations and variational inequalities. In *Computing and Visualization in Science*, volume 7, pages 189–197, 2004.

[34] T. Schmidt. *Analyse zweier Finite-Volumen-Methoden für elliptische partielle Differentialgleichungen 2. Ordnung auf Viereckgittern*. PhD thesis, Christian-Albrechts-Universität zu Kiel, 1992.

[35] I. A. Segal. Finite element methods for the incompressible navier-stokes equations, 2015.

[36] F. Sprengel. Comparing multilevel coarsening strategies. In *Electronic Transactions on Numerical Analysis*, volume 14, pages 178–194, 2002.

[37] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2000.

[38] C. Zenger. Sparse grids. In *Parallel Algorithms for Partial Differential Equations*, volume 31, pages 241–251. Vieweg-Verlag, 1991.

[39] Z. Zhang and Q. Zou. Some recent advances on vertex centered finite volume element methods for elliptic equations. In *Science China Mathematics*, volume 56, pages 2507–2522, 2013.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift