

Masterarbeit

zur Erlangung des akademischen Grades
Master

Technische Hochschule Wildau

Fachbereich Wirtschaft, Informatik, Recht

Studiengang Wirtschaftsinformatik (M. Sc.)

Thema (deutsch): Vorteile von adaptiven Multigrid-Methoden bei der Lösung Poisson-ähnlicher Gleichungen

Thema (englisch): Advantages of using adaptive multigrid while solving Poisson-like equations

Autor/in: Janot Phillip George

Seminargruppe: IMT/22

Betreuer/in: Prof. Dr. rer. pol. Ralf Szymanski

Zweitgutachter/in: Dr rer nat Christian Hüttig

Spätestmögliche Abgabe: 25.05.2026

Acknowledgment

First, I would like to thank my supervising professor from the Technical University of Applied Sciences Wildau, Dr. Szymanski and my supervisor from the German Aerospace Center (DLR) Institute of Space Research, Dr. Hüttig, for making this thesis possible and supporting me throughout the process. I am very grateful for the insights into GAIA and its surrounding physics provided by Dr. Hüttig, as well as his patience and encouragement, which inspired me to choose this thesis' topic in the first place. Many thanks also go out to all members of the Department of Planetary Physics, the Department of Planetary Geodesy & Geology at DLR Institute of Space Research and the AMR Group of the High-Performance Computing Department at the DLR Institute of Software Technology, who assisted me with insights and discussions.

Abstract

Simulating the thermal evolution of planetary bodies requires solving large linear matrix systems, which often converge slowly in the presence of sharp material gradients. This thesis investigates whether an adaptive geometric multigrid solver can overcome this limitation. A multigrid solver for the steady-state non-homogeneous heat equation was implemented within the planetary fluid-flow solver GAIA on a 3D domain with over two million (2^{21}) volumes to solve scenarios with high diffusion contrasts. Performance was evaluated by counting memory access across various cycle types and adaptive strategies. `t8code`, a library for adaptive grid management, was used to assist this implementation by managing grid transfer operations and adaptations. ¹

Results show that multigrid solvers outperform GAIA's stabilized BiConjugate gradient solver by a factor of 7 to 10 in uniform environments and maintain high efficiency under strong gradients where current GAIA's solver slows down. The adaptive multigrid approach preserves resolution at regions of abrupt changes in material properties, thereby reducing computational costs compared to non-adaptive variants. Therefore, this geometric multigrid method offers a higher performance alternative. ²

¹Text passage reviewed by Qwen 3.5-122B.

²Text passage reviewed by Qwen 3.5-122B.

Contents

List of Abbreviations	v
List of Figures	ix
List of Tables	x
1 Introduction	1
1.1 Structure	1
1.2 Software	2
1.2.1 GAIA	2
1.2.2 t8code	3
2 Theory	4
2.1 Reference Problem	4
2.2 Finite Volume Method	7
2.3 Multigrid Concept	8
2.4 Heat Equation	11
3 Methodology	14
3.1 Matrix Creation	14
3.1.1 Applying FVM	15
3.1.2 Boundary Conditions	17
3.1.3 Matrix Structure	19
3.2 Multigrid	22
3.2.1 Defect Function	23

3.2.2	Smoothing Function	23
3.2.3	Coarse Grid Correction	26
3.2.4	Fine Grid Correction	30
3.2.5	Multigrid Cycles	32
3.3	Adaptive Multigrid	35
3.3.1	Balanced Grid	38
3.3.2	Multigrid Function Changes	39
3.3.3	Grid Adaptation	42
3.4	Simulation Setup	45
3.4.1	Test Conditions	45
3.4.2	Benchmark Evaluation and the SMRA metric	47
3.4.3	Validation	48
4	Results	49
5	Conclusion and Outlook	57
	Bibliography	61

List of Abbreviations

BiCGSTAB Stabilized BiConjugate Gradient

CG Conjugated Gradient

DLR German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt)

FMG Full Multigrid

FVM Finite Volume Method

GS Gauss-Seidel

GS-LEX Lexicographic Gauss-Seidel

GS-RB Red-Black Gauss-Seidel

MLAT Multi-Level Adaptive Technique

MPI Message Passing Interface

SMRA Sparse Matrix Row Accesses

List of Figures

1.1	Adaptive mesh refinement structure of t8code's grids and its volumes, visualizing the creation of the z-curve and the distribution of volumes among processes. Source: This figure was provided by the AMR Group from the DLR Institute of Software Technology.	3
2.1	Applying discretizations with different resolutions to a 2D square domain, changing the accuracy of the approximation of the physical quantity, here temperature diffusion coefficient α : (a) Distribution of α . (b) Domain split into 32768 squares. (c) Domain split into 64 squares. (d) Colorbar of α for reference.	5
2.2	Discretization of a 3D cubic domain into slightly more than 2 million volumes, divided into two regions with distinct diffusion coefficients α . A blue half-sphere at the bottom of the cube with α_{blue} and a red surrounding region with α_{red} . (a) Full view of the cube. (b) Cross-section of the cube visualizing the spherical shape of the blue region.	6
2.3	2D square environment discretized into smaller squares, to visualize and explain the relations between control volume X and its neighbors $N1$ to $N4$	7
2.4	3D Grids and their matrices, considering 1 physical quantity. (a) A cubical grid with 64 volumes. (b) Matrix of (a) with $64 \times 64 = 4096$ elements. (c) The coarsened grid of (a) with 8 volumes. (d) Matrix of (c) with $8 \times 8 = 64$ elements.	8

2.5	A simple 2-grid-cycle visualized by (Baranoski et al., 2002). Note that the word error in this figure is a simplification. Defect or residual is the technically correct wording. The initial residual 2.2 of formula 2.1 starts out noisy and gets smoothed with formula 2.3. The smooth residual 2.5 gets restricted 2.6 together with matrix A to create a matrix system for the coarse grid 2.7. After solving that system, the solution gets interpolated 2.8 and corrected 2.9. The now introduced noisy residual gets smoothed again, leading to the solution 2.10 after one 2-grid-cycle.	9
2.6	A 2D coarse grid correction Θ of the residual res from child volumes (left) to its parent (right). The interpolation in this example is the sum of child residuals: $\sum res(child) = res(parent)'$	10
2.7	A 2D fine grid correction Ξ of u'_{solved} from a single parent volume (left) to its child volumes (right). The interpolation in this example is the copy or inheritance of the same value $u'_{solved}(parent) = u_f(child)$ for all child volumes.	11
3.1	A 2D grid example with different sized volumes to visualize the creation of the ghost volumes. (a) A 2D adapted grid with different sized volumes. (b) Identification of the boundary volumes in orange. (c) Creation of the ghost volumes in blue, based on their corresponding boundary volume.	18
3.2	A 2D grid with z-curve indexed volumes (a) and its representative matrix A (b).	22
3.3	A 1D example of the aliasing and smoothing effect. The top graphs show residual noise (blue) on the fine grid (blue centers) being restricted (red) to the coarse grid (red centers) without prior smoothing. The bottom graphs show a comparison of the same process after prior smoothing of the residual.	24
3.4	Visible shifting of the error when smoothing is performed only along the z-curve in a fully symmetrical problem.	26
3.5	Two 2D example Grids: (a) Grid $G^{(0)}$. (b) Grid $G^{(1)}$	28
3.6	A Multigrid V-cycle with 6 grids.	32
3.7	Multigrid γ -cycles by (Trottenberg et al., 2001, Page 46).	33
3.8	A Multigrid F-cycle with 6 grids.	33
3.9	A Multigrid FMG-V-cycle with 6 grids.	34

3.10	2D example of FMG-interpolation: (a) Refinement of the inner volume with respect to the neighboring volumes. (b) Barycentric interpolation towards a child Volume with the help of its parent volume and its neighboring volumes.	35
3.11	The geometric adaptive refinement approach, with a locally refined grid around cut cells to accurately capture the circular geometry by (Saberi et al., 2020, Chapter 3).	36
3.12	Strategies for domain partitioning in adaptive grids. (a) Adaptively refined grid. (b) Global coarsening includes all volumes per level. (c) Local smoothing restricts computation to the finest volumes, with coarser volumes as boundaries. (Qin et al., 2025, Chapter 3.1).	37
3.13	Multi-Level Adaptive Technique (MLAT) used on a 3D cube grid with a cross-section view. (a) Distribution of the blue and red region. (b) Locally refined finest grid $G^{(0)}$. (c) Coarse starting grid $G^{(C)}$.	37
3.14	Approximation methods for the face center with neighbors of different sizes.	39
3.15	An adapted 2D grid with z-curve indexed volumes (a) and its representative matrix A (b).	40
3.16	Two 2D example grids: (a) Grid $G^{(0)}$. (b) Adaptively coarsened grid $G^{(1)}$.	41
3.17	Cross section view of a 3D Level 7 grid that gets coarsened and balanced: (a) Distribution of the blue and red regions. (b) Finest grid $G^{(0)}$. (c) Grids $G^{(1)}$ to (g) $G^{(5)}$ are coarsened and balanced iteratively. (h) Color-coded scale of levels.	42
3.18	Cross section view of a 3D Level 7 grid that gets coarsened and balanced initially with a criterion and continued without one: (a) Finest grid $G^{(0)}$. (b) Grids $G^{(1)}$ to (d) $G^{(3)}$ are coarsened and balanced with coarsening criterion iteratively. (d) Grids $G^{(3)}$ to (k) $G^{(10)}$ are coarsened and balanced without coarsening criterion iteratively. (l) Color-coded scale of levels.	44
4.1	The 3D cube test environments solved by GS-LEX, showing the solved temperature $u_{solved}^{(0)}$ in Kelvin, as a cross-section view of the cube. (a) Distribution of blue and red regions. (b) Temperature with $\alpha_{red} = \alpha_{blue} = 1$. (c) Temperature with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$. (d) Temperature with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$.	49

4.2	Performance of all tested BiCGSTAB and multigrid solvers with $\alpha_{red} = \alpha_{blue} = 1$. The Plot on the right is a closeup view of just the multigrid solvers with a logarithmic x-axis for better comparison.	50
4.3	Performance of all tested BiCGSTAB and multigrid solvers with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$	50
4.4	Performance of all tested BiCGSTAB and multigrid solvers with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$	51
4.5	Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with GS-LEX.	52
4.6	Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with GS-LEX. The Plot on the right is a closeup view of just the multigrid solvers with a logarithmic x-axis for better comparison.	52
4.7	Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with BiCGSTAB.	53
4.8	Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with BiCGSTAB.	53
4.9	Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$, using the grid structure from figure 3.18. The Plot on the right is a closeup view of just the multigrid solvers with a logarithmic x-axis for better comparison.	54
4.10	Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$, using the grid structure from figure 3.18.	55

List of Tables

2.1	Number of BiCGSTAB iterations required to reach a relative euclidean residual norm of 10^{-6} for the domain in figure 2.2 under varying diffusion coefficients.	6
2.2	Complexity of 2D Poisson solvers with n unknowns (Trottenberg et al., 2001 , Page 14).	7
3.1	Comparison of Gauss-Seidel smoothers, with N as the number of volumes in grid G (Trottenberg et al., 2001 , Page 34).	24
3.2	Number of volumes per grid. The adaptive multigrid data is based on the example in figure 3.17.	43
3.3	Number of volumes per grid. The adaptive multigrid data is based on the example in figure 3.18.	45
4.1	Total SMRAs required by each solver per test scenario.	51
4.2	Total SMRAs required by each solver per test scenario. The two left columns show the results when the matrix system of $G^{(C)}$ is solved by GS-LEX and the two right columns show that, when BiCGSTAB is used instead.	54
4.3	Total SMRAs required by each solver per test scenario. The two left columns show the results of the non adaptive multigrid solvers and the two right columns show the adaptive multigrid solvers.	55
4.4	Total SMRAs required per solver per approach and per test scenario. The best result for each scenario is highlighted	56
5.1	AI tool usage	61

Chapter 1

Introduction

A primary goal of planetary science is to investigate the thermal history and potential future of planets, moons and other celestial bodies. One way to accomplish this is by simulating these environments using physical equations and known material properties. This allows researchers to simulate the thermal evolution, test and validate physical assumptions, predict the future and model the past for both, existing and hypothetical bodies. This thesis aims to support the software GAIA used by the German Aerospace Center, the Deutsches Zentrum für Luft- und Raumfahrt (DLR). This software is used by many scientists at the DLR Institute of Space Research to study the mentioned topics. The task of this work is to implement and test a multigrid solver for GAIA in order to improve its simulation performance. The topic of this work was chosen based on familiarity with GAIA and the experience gained from the implementation of an adaptive mesh refinement library t8code, which is also used in this thesis and was developed by the DLR Institute of Software Technology.

1.1 Structure

The following chapters explain the theory behind the discretization of physical equations and the multigrid idea followed by how these systems of equations are formulated within GAIA and how specific environmental variations reduce efficiency of the current solver. The idea of a multigrid solver is introduced afterwards, along with how it aims to solve these more complex systems of equations more efficiently.

The following sections explain which representation of the Navier-Stokes equations will be tested and how the system of equations is created in detail. The multigrid and adaptive multigrid method is explained with all its components and how they change the way the system of equations is solved. GAIA's current solver is compared with selected (adaptive)

multigrid methods in predefined physical environments to evaluate their performance. ¹

1.2 Software

The primary objective of this thesis is to analyze and develop a multigrid solver for a possible integration into the GAIA simulation framework. The code developed in this work was implemented directly within GAIA's C++ environment, ensuring the algorithms compatibility and deployability. ²

While GAIA has its own grid system, it currently has no efficient mechanisms for data transfer and management between multiple grids or grid adaptation. To address these limitations, t8code is integrated to take over those tasks. It is a library with optimized algorithms for grid adaption and data transfer between these grids, which are essential for the performance of the adaptive multigrid solver.

1.2.1 GAIA

GAIA is a geophysical fluid flow solver used to study thermal and thermo-chemical evolution of planetary bodies, influence of plate tectonics, magnetic field development, magma oceans, partial melting, mantle differentiation and more ([Ana-Catalina Plesa, 2013](#), [HelmholtzSoftware, 2023](#), [Hüttig and Stemmer, 2008a,b](#), [Hüttig et al., 2013](#)).

"GAIA is a comprehensive framework designed to study the thermo-chemical evolution of rocky and icy bodies (planets, moons, and planetoids) on a global scale within the solar system and beyond. The core part consists of a fluid-dynamics solver for creeping flows under highly spatially varying viscosities with an additional energy solver for natural convection.

Natural convection is a type of heat transfer that occurs due to the movement of a fluid (such as air, water, or in much larger time-scales even rocks) caused by differences in density. When a fluid is heated, it becomes less dense and rises, while cooler, denser fluid sinks. This creates a natural circulation pattern that transfers heat from one area to another without the need for external forces like pumps or fans.

In the context of planetary bodies, natural convection plays a crucial role in

¹Text passage reviewed by Qwen 3.5-122B.

²Text passage reviewed by Qwen 3.5-122B.

the thermal and thermo-chemical evolution of rocky and icy bodies. It drives processes such as mantle convection, which can influence plate tectonics, vulcanism, magnetic field development, and the differentiation of materials within the planet's interior." (HelmholtzSoftware, 2023)

1.2.2 t8code

t8code is a C/C++ library that facilitates the management of parallel adaptive meshes across diverse element types (Holke, 2018, Holke et al., 2023).

"t8code uses a collection (a forest) of multiple connected adaptive space-trees in parallel and scales to at least one million MPI ranks and over 1 Trillion mesh elements. It is licensed under the GNU General Public License 2.0 or later. Copyright (c) 2015 the developers. t8code is intended to be used as a thirdparty library for numerical simulation codes or any other applications that require meshes." (DLR-AMR, 2015)

An important feature of t8code is its space filling curve, or z-curve (Burstedde and Holke, 2016, Knapp, 2017, 2020). It dictates the indexing of the volumes in a Z-shaped pattern. Figure 1.1 shows the structure of a mesh and illustrates how the z-curve is constructed.

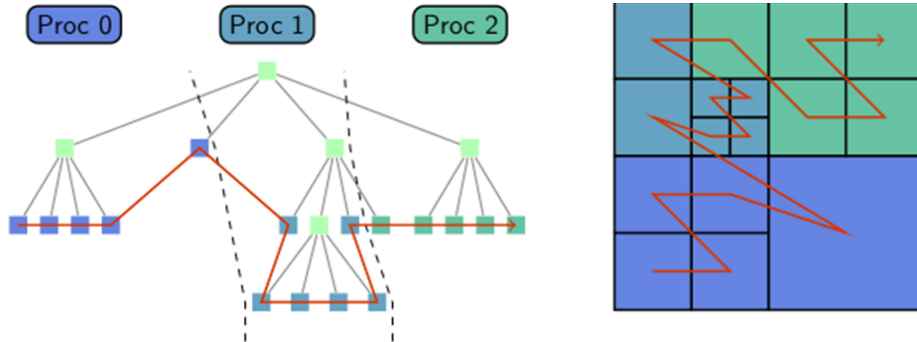


Figure 1.1: Adaptive mesh refinement structure of t8code's grids and its volumes, visualizing the creation of the z-curve and the distribution of volumes among processes. Source: This figure was provided by the AMR Group from the DLR Institute of Software Technology.

In t8code, volumes are described by their level. A level 0 volume fills the entire grid. A level 1 volume is a child of a level 0 volume and so on, creating a tree structure. Levels represent the layers of the tree structure, starting with level 0 at the root and ending with level K at the deepest leaf nodes. Since indexing starts at 0, K is always one less than the layer number. Figure 1.1 also describes this process. ³

³Text passage reviewed by Qwen 3.5-122B.

Chapter 2

Theory

2.1 Reference Problem

When considering physical attributes of environments, such as planetary interiors, the problem involves continuous partial differential equations, that describe the behavior of physical quantities within the domain. As, continuous equations describe infinite points of the domain and therefore cannot be solved directly by computers. Therefore, the environment must be separated by dividing it into finite subdomains in the form of simple geometric shapes like cubes and squares. This process is called discretization, which is commonly performed by numerical methods such as the Finite Volume, Finite Element or Finite Difference methods in physical modeling ([Hüttig and Stemmer, 2008b](#), [Mattiussi, 1997](#), [Richardson, 1911](#)). These methods approximate the physical quantities within the subdomains, transforming the continuous problem over the entire domain into a set of smaller, coupled problems for each subdomain. Thus, these problems can be solved within a finite system of equations using specialized solvers. Figure 2.1 demonstrates the effect of different discretization resolutions, achieved by varying the number of subdomains on the approximation. In this context, the diffusion coefficient α represents the ability of the materials to conduct the physical quantity.

¹

GAIA uses the Finite Volume Method (FVM) to separate the domain into millions of smaller volumes and approximates the physical differential equations with linear equations. This creates one equation for each unknown physical quantity per volume. By assigning an index to each volume and accounting for the fact that each equation depends on its neighboring volumes, this system of equations can be rewritten as a linear matrix equation in the form of:

$$A \cdot u = B \tag{2.1}$$

¹Text passage reviewed by Qwen 3.5-122B.

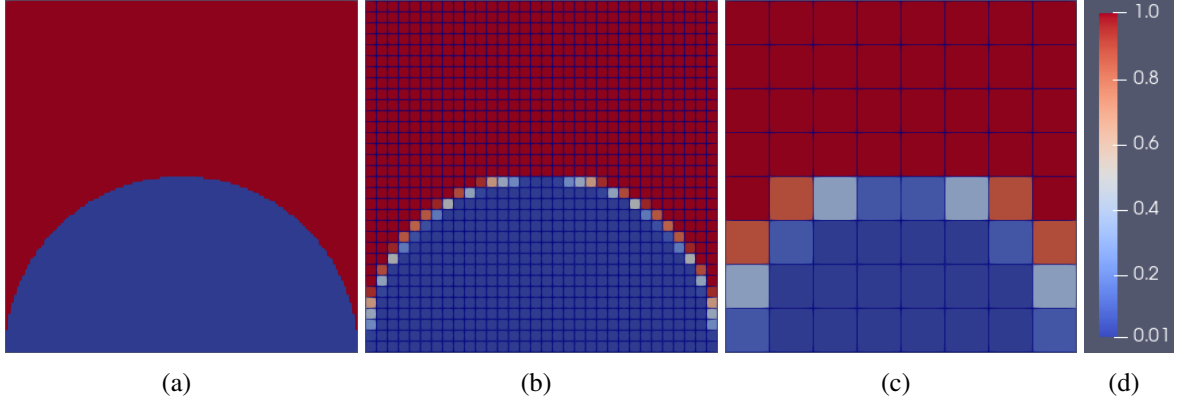


Figure 2.1: Applying discretizations with different resolutions to a 2D square domain, changing the accuracy of the approximation of the physical quantity, here temperature diffusion coefficient α : (a) Distribution of α . (b) Domain split into 32768 squares. (c) Domain split into 64 squares. (d) Colorbar of α for reference.

Where $A \in \mathbb{R}^{N \times N}$ is a banded square sparse matrix, $B \in \mathbb{R}^N$ is the non-zero source term vector of the physical quantities and $u \in \mathbb{R}^N$ is the vector of unknown physical quantities to be solved, where $N \in \mathbb{N}$ is the number of volumes times physical quantities. Many physical processes, particularly those modeled by the FVM, are driven by diffusion and conservation laws. The FVM is designed to conserve quantities such as energy, mass and momentum across each control volume. Instead of solving the differential equations at each point, the FVM integrates these laws over (finite) volumes. This approach then transforms the local differential equation, which applies to a point in the system, into a global system of balance equations. This transformation introduces no approximation error regarding the conservation law itself, as the balance equation is a topological equation. Which means it remains valid regardless of how the material behaves. The discrete global balance is the representation of the conservation law (Mattiussi, 1997, Page 290). Mathematically, these conservation laws often lead to Poisson or Poisson-like equations, which describe how a physical potential such as temperature, pressure or velocity behaves in response to its source B .²

The matrix equation 2.1 often represents or contains such a physical problem. The size of these matrices is immense, making the computation memory-intensive. Most direct solvers for these types of systems have a quadratic operational complexity $O(n^2)$, where n is the number of unknowns, making it impractical to upscale in larger simulations. Iterative solvers promise a higher calculation speed, with the downside of not finding the exact solution in favor of an approximation with a flexible precision parameter.

Even efficient iterative solvers, like the Conjugate Gradient (CG) with a computational complexity of $O(n^{4/3})$ for 3D Poisson problems (Shewchuk, 1994, Page 38) may fail to converge efficiently for Poisson-like problems. GAIA uses another version of this solver, the

²Text passage reviewed by Qwen 3.5-122B.

Stabilized BiConjugate Gradient (BiCGSTAB). It is more robust, but loses efficiency when the diffusion coefficient $\alpha \in \mathbb{R}^N$ of the physical quantity varies drastically between neighboring volumes, creating sharp gradients that slow down convergence. Another disadvantage of pure CG is that it is only applicable to symmetric matrices.³

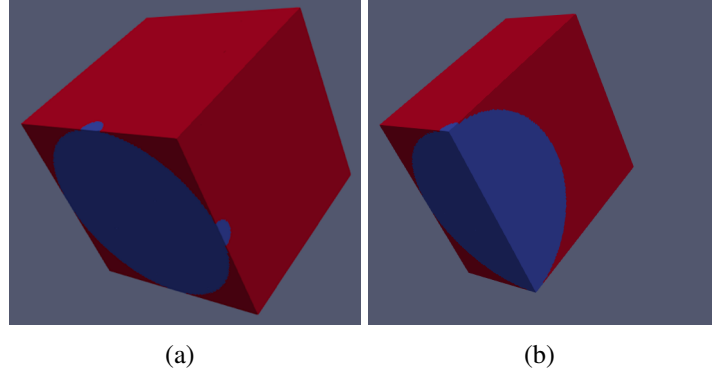


Figure 2.2: Discretization of a 3D cubic domain into slightly more than 2 million volumes, divided into two regions with distinct diffusion coefficients α . A blue half-sphere at the bottom of the cube with α_{blue} and a red surrounding region with α_{red} . (a) Full view of the cube. (b) Cross-section of the cube visualizing the spherical shape of the blue region.

In figure 2.2 (a), an environment can be seen in the form of a cube and in (b) the same cube cut in half in order to visualize the blue region. Sharp gradients in the volumes at the boundary of these regions are created, if their diffusion coefficients differ by 10^5 . To solve this, BiCGSTAB requires about 50% more iterations to reach a the relative euclidean residual norm of 10^{-6} , than if the coefficient is equal, see table 2.1.

α_{red}	α_{blue}	BiCGSTAB Iterations
1	1	180
1	10^5	272
1	10^{-5}	276
10^{-5}	1	286
10^5	1	263

Table 2.1: Number of BiCGSTAB iterations required to reach a relative euclidean residual norm of 10^{-6} for the domain in figure 2.2 under varying diffusion coefficients.

This Master’s thesis analyzes a different solver approach called geometric multigrid, which aims to accelerate this process by utilizing geometric information. Multigrid is equally as or more efficient than specialized Poisson solvers like Buneman and Total Reduction (Trottenberg et al., 2001, Page 15). Table 2.2 lists the computational complexity of these solvers for 2D Poisson problems. However, these boundaries do not necessarily translate to Poisson-like problems with variable coefficients. In contrast, the multigrid method maintains

³Text passage reviewed by Qwen 3.5-122B.

its efficiency even in these more complex scenarios from table 2.1. ⁴

Method	Operation Complexity
Gaussian Elimination (band version)	$O(n^2)$
Jacobi Iteration	$O(n^2)$
Gauss-Seidel Iteration	$O(n^2)$
Conjugate Gradient	$O(n^{3/2})$
Bunemann	$O(n \log(n))$
Total Reduction	$O(n)$
Multigrid	$O(n)$

Table 2.2: Complexity of 2D Poisson solvers with n unknowns (Trottenberg et al., 2001, Page 14).

2.2 Finite Volume Method

FVM is based on balancing the incoming and outgoing fluxes of mass, energy and momentum between control volumes and their neighbors. This is achieved by integrating the physical equations over each control volume, where the fluxes across the boundaries depend on the values of the neighboring volumes. This process leads to one equation per volume per physical quantity. An advantage of FVM is the strict conservation of these quantities at the discrete level. As GAIA uses the Finite Volume Method, it will also be the choice for this thesis' discretization method (Hüttig and Stemmer, 2008b). Furthermore, FVM supports arbitrary grids and geometries (Kolditz, 2002, Page 173), which satisfy the requirements for implementing the adaptive multigrid method, see chapter 3.3.

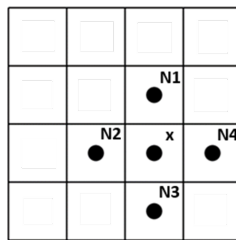


Figure 2.3: 2D square environment discretized into smaller squares, to visualize and explain the relations between control volume X and its neighbors $N1$ to $N4$.

The type of heat equation used in this thesis does not include mass or momentum. So, only the energy conservation is accounted for. This can be accomplished by having the flux of energy be consistent between two neighboring volumes, meaning if the heat flux is positive from one volume to another, it has to be negative from the point of view of the other volume. Figure 2.3 shows a square shaped environment discretized into smaller squares. Volume X is

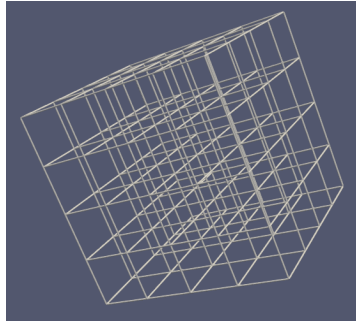
⁴Text passage reviewed by Qwen 3.5-122B.

the control volume, with neighboring volumes $N1$ to $N4$ at each face. If the flux from control volume X to control volume $N1$ is 0.5, the flux from volume $N1$ to X needs to be -0.5 in the formula, where $N1$ is the control volume and X is $N1$'s neighbor. For each volume each physical attribute is stored as a single value in GAIA, therefore, information inside a volume needs to be averaged. This means the flux of a face, is an interpolation of information from neighboring volumes.

2.3 Multigrid Concept

The general idea of a multigrid solver is solving a discretized problem on a fine grid while using additional coarser grids on the same domain to accelerate the computation. Calculations on a coarser grid are faster, as the matrix system that needs to be solved is smaller, since the size of matrix A increases exponentially with the number of volumes. Coarsening a D -dimensional hypercube grid with t8code leads to combining 2^D child volumes into a hypercube parent volume, reducing the number of volumes by 2^3 in 3D for each coarser grid.

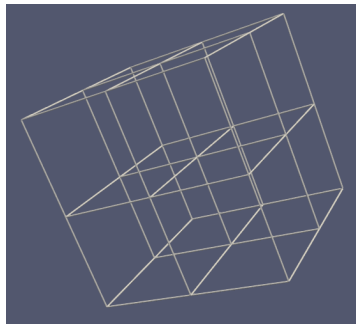
5



(a)

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,63} & a_{1,64} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,63} & a_{2,64} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{63,1} & a_{63,2} & \cdots & a_{63,63} & a_{63,64} \\ a_{64,1} & a_{64,2} & \cdots & a_{64,63} & a_{64,64} \end{bmatrix}$$

(b)



(c)

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,7} & a_{1,8} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,7} & a_{2,8} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{7,1} & a_{7,2} & \cdots & a_{7,7} & a_{7,8} \\ a_{8,1} & a_{8,2} & \cdots & a_{8,7} & a_{8,8} \end{bmatrix}$$

(d)

Figure 2.4: 3D Grids and their matrices, considering 1 physical quantity. (a) A cubical grid with 64 volumes. (b) Matrix of (a) with $64 \times 64 = 4096$ elements. (c) The coarsened grid of (a) with 8 volumes. (d) Matrix of (c) with $8 \times 8 = 64$ elements.

⁵Text passage reviewed by Qwen 3.5-122B.

Since the shape of the matrix A is quadratic, with a size of $N \times N$, where N is the number of volumes times physical quantities, this reduces the number of elements in the matrix during each coarsening iteration by $(2^D)^2 = 4^D$. Figure 2.4 (a) shows a 3D example grid with 64 volumes. (b) shows its matrix, considering 1 physical quantity with $64 \times 64 = 4096$ elements. The coarser grid in (c) has 2^3 fewer volumes than the grid in (a) and its matrix in (d) has 4^3 fewer elements.⁶

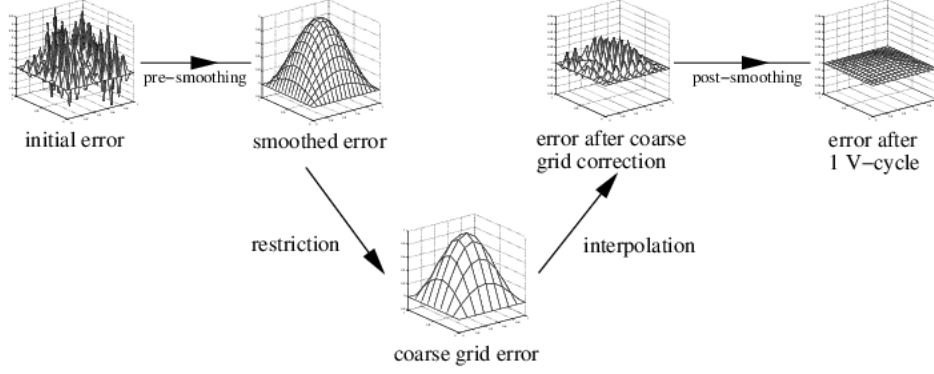


Figure 2.5: A simple 2-grid-cycle visualized by (Baranoski et al., 2002). Note that the word error in this figure is a simplification. Defect or residual is the technically correct wording. The initial residual 2.2 of formula 2.1 starts out noisy and gets smoothed with formula 2.3. The smooth residual 2.5 gets restricted 2.6 together with matrix A to create a matrix system for the coarse grid 2.7. After solving that system, the solution gets interpolated 2.8 and corrected 2.9. The now introduced noisy residual gets smoothed again, leading to the solution 2.10 after one 2-grid-cycle.

Figure 2.5 shows a simple 2D multigrid algorithm with two grids. On the leftmost side is the initial residual of $A \cdot u = B$ visualized, with an initial guess of u_{ini} , which does not solve the equation yet. Its definition is as follows:

$$res(B, A, u_{ini}) = B - A \cdot u \quad (2.2)$$

The solving process starts out with a noisy residual over the whole domain. Before the transfer to the second grid can be applied, noise needs to be smoothed to prevent aliasing. During this step an efficient smoother is desired. The solving algorithm is later applied on the coarsest grid. The smoother function Ψ changes the vector u_{ini} depending on A and B to u_{smooth} :

$$\Psi_{A,B}(u) = u_{smooth} \quad (2.3)$$

This results in the new system of equations:

$$A \cdot \Psi_{A,B}(u) = A \cdot u_{smooth} = B \quad (2.4)$$

⁶Text passage reviewed by Qwen 3.5-122B.

Which leads to a smooth residual:

$$res(B, A, u_{smooth}) = B - A \cdot u_{smooth} \quad (2.5)$$

To transfer data from a fine grid to a coarser one, the matrix A of the initial grid must be coarse grid corrected or restricted. This can be done in multiple ways. Two common options for the restriction are algebraic and geometric. In the algebraic approach the matrix needs to be multiplied by transfer operators to create the new matrix A' (Trottenberg et al., 2001, Page 42), while in the geometric approach the FVM will get reapplied again and a different system of equations is created, $A'u' = B'$. Both methods also need to restrict the residual onto the new grid after the smoothing process. The coarse grid correction Θ is defined as:

$$\Theta(A, res) = (A', res') \quad (2.6)$$

Figure 2.6 shows an example of how to coarse correct the residual, by summing up the residuals of the child volumes.

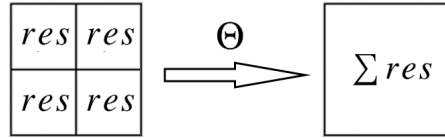


Figure 2.6: A 2D coarse grid correction Θ of the residual res from child volumes (left) to its parent (right). The interpolation in this example is the sum of child residuals: $\sum res(child) = res(parent)'$.

This provides all necessary ingredients for the matrix system of the coarser grid:

$$A' \cdot u' = res(B, A, u_{smooth})' \quad (2.7)$$

The right-hand side of the system of equations of the coarser grid 2.7 is now the restricted residual res' and not the source term B' like in equation 2.1, since the idea of the multigrid is to solve the defect on that coarser grid. As this approach tries to solve for u' , it starts by assuming $u' = 0$ in formula 2.7, since u' represents the correction term for u_{smooth} from formula 2.4 and no correction is done up to this point.

Since the system of equations 2.7 is defined on the coarsened grid, it can now be solved, providing u'_{solved} . Sometimes a smoother is also a sufficient solver, depending on the size of the grid and the desired accuracy of the resulting u'_{solved} . A smoother is a good solver when the grid contains only a few volumes. Finding u'_{solved} means to find values which cancel out the over- and underflow of u_{smooth} . Therefore, the next step is to interpolate or conglomerate u'_{solved} and correct u_{smooth} . This can be done, by first applying a fine grid correction Ξ to

u'_{solved} , which is visualized in figure 2.7 and defined as:

$$\Xi(u'_{solved}) = u_f \quad (2.8)$$

Where u_f is the fine grid corrected u'_{solved} . Afterwards, u_{smooth} is added to it, overwriting u :

$$\Xi(u'_{solved}) + u_{smooth} = u \quad (2.9)$$

The fine grid correction Ξ in both the geometric and algebraic approach revolves around interpolating the found solution u'_{solved} of each parent volume to their respective child volumes, to transfer the correction found in the coarser grid to the finer one.

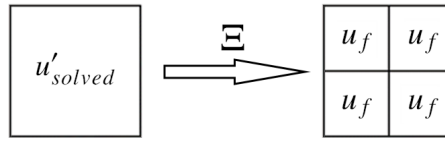


Figure 2.7: A 2D fine grid correction Ξ of u'_{solved} from a single parent volume (left) to its child volumes (right). The interpolation in this example is the copy or inheritance of the same value $u'_{solved}(\text{parent}) = u_f(\text{child})$ for all child volumes.

This interpolation however introduces some over and under-correction resulting in defect noise, which needs to be smoothed again with formula 2.3, overwriting u_{smooth} . This cycle can be repeated until a desired accuracy of the residual $res(B, A, u_{smooth})$ is achieved. If u_{smooth} now solves formula 2.1 with a sufficient precision, it is considered u_{solved} .

$$A \cdot u_{solved} = B \quad (2.10)$$

The multigrid method can also be used as a preconditioner for existing iterative solvers. (Böing, 2022) studied the application of geometric multigrid as a preconditioner for simulations on adaptive meshes for the advection-diffusion equation under difficult physical conditions. This work will also test one simplified version of such a preconditioner, although the focus lies on the multigrid solver itself.

2.4 Heat Equation

The general Poisson equation is defined as:

$$\nabla \cdot \nabla u = -f \quad (2.11)$$

Where u is a twice differentiable scalar field representing the physical quantity to be solved and f is a function, which typically represents the source term of that quantity. For example, if u represents temperature, f represents the heat source. Radioactive materials, for example, emit heat and act as self-heating sources with positive source term values.

The term $\nabla \cdot \nabla u$ can also be written as $\nabla^2 u$ or $\text{div}(\text{grad}(u))$ and describes the divergence of the gradient of u . The Nabla operator ∇ is a vector containing the partial derivatives for each dimension D :

$$\nabla = \left(\frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_D} \right) \quad (2.12)$$

The scalar product of this operator with itself results in:

$$\nabla \cdot \nabla = \frac{\partial^2}{\partial x_1^2} + \dots + \frac{\partial^2}{\partial x_D^2} \quad (2.13)$$

Adding the time derivative of u , where t denotes time, to the Poisson equation results in the diffusion equation with a constant diffusion coefficient $\alpha = 1$:

$$\alpha \cdot \nabla^2 u - \frac{\partial u}{\partial t} = -f \quad (2.14)$$

When u is the temperature, this formulation is known as the heat equation. In the case where $f \neq 0$, the equation is classified as non-homogeneous. Furthermore, if the diffusion coefficient α is not constant, the equation takes this form:

$$\nabla \cdot (\alpha \cdot \nabla u) - \frac{\partial u}{\partial t} = -f \quad (2.15)$$

To reformulate this expression to match the relevant part of the Navier-Stokes equations, the steady state is considered by setting the time derivative $\frac{\partial u}{\partial t}$ to 0. This results in the steady-state non-homogeneous heat equation, which is classified as Poisson-like: ⁷

$$\nabla \cdot (\alpha \cdot \nabla u) = -f \quad (2.16)$$

The specific relevance of this steady-state formulation for the GAIA simulations is explained in the following quote, which highlights the role of spatially varying viscosities in planetary mantles: ⁸

"For the GAIA simulation this has a different background because it is equivalent to a term in the Navier-Stokes equations that describes spatially varying viscosities. These viscosity variations may also come from different materials but mainly from the "problem" that in a planetary mantle the viscosity is strongly

⁷Text passage reviewed by Qwen 3.5-122B.

⁸Text passage reviewed by Qwen 3.5-122B.

temperature and depth dependent and on top even creates phase changes in silicates that introduce many orders of magnitude in sharp spatial gradients. The equation to solve this kind of creeping flow is very similar to the heat equation but in many other terms more complex, which is why this test-scenario was chosen. The main cause of slow convergence was identified to be in that same term that also exists in the heat equation." ([Hüttig, 2026](#))

Chapter 3

Methodology

Since GAIA simulations primarily model fluid dynamics, the main equation to be solved is the Navier-Stokes equations, as it serves as the foundation of this field in physics. This partial differential equation, whose exact solution has not yet been found, is one of the Millennium-Problems. It describes the conservation of momentum in fluid flow and is coupled with the continuity and energy equations to form a system of differential equations (Huber, 2025, Chapter 10.1).¹

This thesis aims to investigate whether the multigrid method can be used in GAIA on this system of equations by solving an analogous Poisson-like equation. Multigrid methods are state of the art solvers that are investigated, tested and developed in physical simulations at the time of writing this thesis. In 2025 the paper of (Lohmann, 2025) researched a multigrid solver for these Navier-Stokes equations in a different context with a different discretization method, the Finite Element Method. The article (Mouhssine et al., 2025) investigates a geometric multigrid solver for linear sparse matrices for elliptic partial differential equations with vector extrapolation.

3.1 Matrix Creation

The chosen steady-state non-homogeneous heat equation can be expressed explicitly as:

$$\nabla \cdot (\alpha(x) \cdot \nabla u(x)) = -f(x) \quad (3.1)$$

Where $\alpha(x) \in \mathbb{R}$ is the diffusion coefficient, $u(x) \in \mathbb{R}$ is the temperature and $f(x) \in \mathbb{R}$ is the heat source at location x , for all x in the domain. In this thesis, the simulated domain in

¹Text passage reviewed by Qwen 3.5-122B.

GAIA is defined as a 3D cube with the extent of $[0, 1]^3$. The location x is represented by a vector of 3D Cartesian coordinates: $(x_1, x_2, x_3) \in [0, 1]^3$. $\nabla u(x)$ denotes the gradient of the temperature at location x and the term $\nabla \cdot (\nabla u(x))$ represents the divergence of this gradient.

3.1.1 Applying FVM

Let G be a 3D cubic grid covering the simulation domain, discretized by uniform cubes like in the example grids in figure 2.4 (a) and (c). V denotes an arbitrary control volume within the grid G and considering $x \in V$. The regular approach in applying FVM is to integrate formula 3.1 over the control volume V (Eymard et al., 2000, Page 8):

$$\int_V \nabla \cdot (\alpha(x) \cdot \nabla u(x)) dV = \int_V -f(x) dV \quad (3.2)$$

For now, the left-hand side term of this equation is focused on:

$$\int_V \nabla \cdot (\alpha(x) \cdot \nabla u(x)) dV \quad (3.3)$$

This term satisfies the conditions of the Divergence Theorem, also known as Gauss's theorem, which states:

$$\int_V \nabla \cdot h(x) dV = \int_{\partial V} h(x) \cdot \vec{n} d\partial V \quad (3.4)$$

Where $V \subset \mathbb{R}^n$ is a compact volume with a piecewise smooth boundary ∂V and $\vec{n}$ is the outward-pointing unit normal vector mapped to each face $S \subset \partial V$ of the volume V . Here, $h(x)$ is a vector field defined on volume V , such that $h : V \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$ (Stoffers, 2015, Page 3).

Applying Gauss's theorem 3.4 to the term 3.3 by setting $h(x) = \alpha(x) \nabla u(x)$ results in:

$$3.3 = \int_{\partial V} \alpha(x) \cdot \nabla u(x) \cdot \vec{n} d\partial V \quad (3.5)$$

This integral over the entire boundary ∂V can be separated into a sum of integrals over each face $S \subset \partial V$, where $\vec{n}_S$ denotes the outward-pointing unit normal vector to face S :

$$3.5 = \sum_{S \subset \partial V} \int_S \alpha(S) \cdot \nabla u(S) \cdot \vec{n}_S dS \quad (3.6)$$

In this thesis, the control volumes are squares in 2D and cubes in 3D, resulting in the number of faces being 4 in 2D and 6 in 3D. To proceed, several approximations are introduced. GAIA stores information of each volume as a single representative value. The position of a volume V is represented by its center x_V and the diffusion coefficient within that volume is averaged

and denoted as the diffusion coefficient of the volumes center $\alpha(x_V)$. The same applies to the temperature $u(x_V)$ and heat source $f(x_V)$.

To access the diffusion coefficient at a face $\alpha(S)$, the diffusion coefficients of the control volume V and its respective neighboring control volume V_S on the other side of said face are averaged:

$$\alpha(S) \approx \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \quad (3.7)$$

The temperature gradient at face S , in the direction of the normal vector $\vec{n}_S$, which points from x_V to x_{V_S} , can be approximated by the directional derivative from x_V to x_{V_S} :

$$\nabla u(S) \cdot \vec{n}_S = \frac{\partial u}{\partial n}(S) \approx \frac{u(x_{V_S}) - u(x_V)}{\|x_{V_S} - x_V\|} \quad (3.8)$$

Applying the approximations from equations 3.7 and 3.8 to the term in equation 3.6 results in:

$$\sum_{S \subset \partial V} \int_S \alpha(S) \cdot \nabla u(S) \cdot \vec{n}_S dS \approx \sum_{S \subset \partial V} \int_S \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S}) - u(x_V)}{\|x_{V_S} - x_V\|} dS \quad (3.9)$$

Since the approximated terms 3.7 and 3.8 are constant over the face S , they can be factored out of the integral:

$$3.9 = \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S}) - u(x_V)}{\|x_{V_S} - x_V\|} \cdot \int_S dS \quad (3.10)$$

The integral over a face S is equal to its surface area, denoted as $|S|$:

$$\int_S dS = |S| \quad (3.11)$$

Applying this equation to term 3.10 provides:

$$3.10 = \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S}) - u(x_V)}{\|x_{V_S} - x_V\|} \cdot |S| \quad (3.12)$$

The left-hand side of formula 3.2 is now approximated by linear terms. The right-hand side of formula 3.2 represents the integral of the heat source over the volume V :

$$\int_V -f(x) dV \quad (3.13)$$

Since GAIA stores the mean value for the heat source of the control volume as $f(x_V)$, the function $f(x_V)$ is treated as constant over the volume V . Therefore, the integral simplifies to

the mean heat source multiplied by the volume of V , denoted as $|V|$:

$$\int_V -f(x) dV \approx -f(x_V) \cdot |V| \quad (3.14)$$

This completes the approximation of formula 3.2:

$$\sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S}) - u(x_V)}{\|x_{V_S} - x_V\|} \cdot |S| = -f(x_V) \cdot |V| \quad (3.15)$$

This equation represents the result of applying the FVM to the steady-state non-homogeneous heat equation 3.1.

3.1.2 Boundary Conditions

Equation 3.15 assumes that every control volume V in grid G has a neighboring volume V_S sharing each face S . This condition cannot always be satisfied at the domains' boundaries, where volumes have at least one face without a neighbor. The standard approach to handle this is by applying boundary conditions (Eymard et al., 2000, Page 10).

This thesis focuses on Dirichlet and Neumann conditions, although the multigrid solver is not limited to these types. Both conditions require the introduction of ghost volumes V_{ghost} in the places a neighboring volume V_S would be missing: x_{ghost} . These ghost volumes of V will be defined by sharing the same physical attributes as V , with exceptions for location and temperature.²

$$\begin{aligned} \text{Heat source: } f(x_{ghost}) &= f(x_V) \\ \text{Diffusion coefficient: } \alpha(x_{ghost}) &= \alpha(x_V) \\ \text{Size: } |V_{ghost}| &= |V| \end{aligned} \quad (3.16)$$

Figure 3.1 visualizes how the ghost volumes are created based on the Grid. (a) shows a 2D adapted grid with 3 different volume sizes. First, the boundary volumes (orange) are identified (b). Then the ghost volumes (blue) are created (c) based on their corresponding boundary volumes by copying their sizes and placing them adjacent to their matching boundary face.

Since ghost volume V_{ghost} has the same diffusion coefficient as the interior volume V , the approximation in formula 3.7 gets simplified whenever the neighbor is a ghost volume:

$$\alpha(S) = \alpha(x_V) \quad (3.17)$$

²Text passage reviewed by Qwen 3.5-122B.

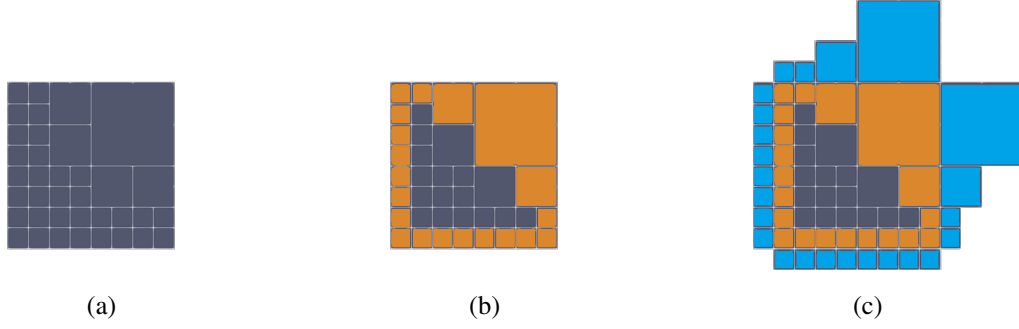


Figure 3.1: A 2D grid example with different sized volumes to visualize the creation of the ghost volumes. (a) A 2D adapted grid with different sized volumes. (b) Identification of the boundary volumes in orange. (c) Creation of the ghost volumes in blue, based on their corresponding boundary volume.

The system of equations would be extended by the ghost volume's temperatures if they remained unknown. This would result in a system containing more unknowns than equations, making it impossible to obtain a unique solution. To prevent this and to simulate the environment around the domain, boundary conditions are introduced. The Dirichlet condition is a first-type boundary condition that dictates specific values for the temperature at the boundary, as shown in formula 3.18. In the context of the heat equation and FVM, this can be imagined as external heat sources, like heating plates, applied to the system boundary. This is applicable when the temperature outside the system is known and fixed, such as modeling the planetary core as a fixed-temperature boundary from below, even if the core itself is not part of the simulation domain. To set a Dirichlet boundary condition $\Phi_D(S)$ on a face S , the temperature at that face is set to $u(S) = \Phi_D(S)$. This translates to an approximated temperature of that ghost volume:³

$$u(x_{ghost}) = 2\Phi_D(S) - u(x_V) \quad (3.18)$$

If the neighbor V_S does not exist and a Dirichlet condition is applied at that face, the face is defined as a Dirichlet face S_D . For all such faces, the term $u(x_{V_S})$ in equation 3.15 is replaced by $2\Phi_D(S_D) - u(x_V)$ and equation 3.17 replaces the approximation of formula 3.7. The term for a control volume V with Dirichlet boundaries changes to:

$$\sum_{S_D \subset \partial V} \alpha(x_V) \cdot \frac{2\Phi_D(S_D) - 2u(x_V)}{\|x_{ghost} - x_V\|} \cdot |S_D| \quad (3.19)$$

If temperature flux should occur at the systems boundary, a gradient or second-type condition is required. The Neumann condition dictates a temperature gradient at boundary S , denoted

³Text passage reviewed by Qwen 3.5-122B.

as $\frac{\partial u}{\partial n}(S)$, instead of a specific value for $u(S)$:

$$\frac{\partial u}{\partial n}(S) = \Phi_N(S) \quad (3.20)$$

This replaces the gradient approximation in equation 3.8 if face S has no neighbor V_S and a Neumann condition is applied. Note that equation 3.17 replaces the approximation of formula 3.7 here as well. Equation 3.15 is modified for all Neumann faces, denoted as S_N , with term:

$$\sum_{S_N \subset \partial V} \alpha(x_V) \cdot \Phi_N(S_N) \cdot |S_N| \quad (3.21)$$

All face types are distinct and form a partition of the boundary:

$$S_N \dot{\cup} S_D \dot{\cup} S = \partial V \quad (3.22)$$

3.1.3 Matrix Structure

The boundary condition terms for the faces 3.19 and 3.21 can be combined with the regular face formula 3.15 to yield an equation accounting for all face types:

$$\begin{aligned} \text{FVM 3.15 (left):} & \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S}) - u(x_V)}{\|x_{V_S} - x_V\|} \cdot |S| \\ \text{Dirichlet 3.19:} & + \sum_{S_D \subset \partial V} \alpha(x_V) \cdot \frac{2\Phi_D(S_D) - 2u(x_V)}{\|x_{ghost} - x_V\|} \cdot |S_D| \\ \text{Neumann 3.21:} & + \sum_{S_N \subset \partial V} \alpha(x_V) \cdot \Phi_N(S_N) \cdot |S_N| \\ \text{FVM 3.15 (right):} & = -f(x_V) \cdot |V| \end{aligned} \quad (3.23)$$

The Neumann term contains no unknowns and can be moved to the right-hand side of the equation:

$$\begin{aligned} \text{FVM:} & \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S}) - u(x_V)}{\|x_{V_S} - x_V\|} \cdot |S| \\ \text{Dirichlet:} & + \sum_{S_D \subset \partial V} \alpha(x_V) \cdot \frac{2\Phi_D(S_D) - 2u(x_V)}{\|x_{ghost} - x_V\|} \cdot |S_D| \\ & = -f(x_V) \cdot |V| - \sum_{S_N \subset \partial V} \alpha(x_V) \cdot \Phi_N(S_N) \cdot |S_N| \end{aligned} \quad (3.24)$$

Next, the Dirichlet term is split into the term with the known boundary value Φ_D and the term with the unknown $u(x_V)$. Similarly, the FVM term is split into the term with the neighbors'

temperatures $u(x_{V_S})$ and the term with the control volume's temperature $u(x_V)$:

$$\begin{aligned}
u(x_{V_S}) \text{ FVM} : & \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S})}{\|x_{V_S} - x_V\|} \cdot |S| \\
u(x_V) \text{ FVM} : & - \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_V)}{\|x_{V_S} - x_V\|} \cdot |S| \\
\Phi_D \text{ Dirichlet} : & + \sum_{S_D \subset \partial V} \alpha(x_V) \cdot \frac{2\Phi_D(S_D)}{\|x_{ghost} - x_V\|} \cdot |S_D| \\
u(x_V) \text{ Dirichlet} : & - \sum_{S_D \subset \partial V} \alpha(x_V) \cdot \frac{2u(x_V)}{\|x_{ghost} - x_V\|} \cdot |S_D| \\
& = -f(x_V) \cdot |V| - \sum_{S_N \subset \partial V} \alpha(x_V) \cdot \Phi_N(S_N) \cdot |S_N|
\end{aligned} \tag{3.25}$$

The Φ_D term is moved to the right-hand side and the terms containing $u(x_V)$ are combined:

$$\begin{aligned}
u(x_{V_S}) : & \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S})}{\|x_{V_S} - x_V\|} \cdot |S| \\
u(x_V) : & - \left(\sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_V)}{\|x_{V_S} - x_V\|} \cdot |S| + \sum_{S_D \subset \partial V} \alpha(x_V) \cdot \frac{2u(x_V)}{\|x_{ghost} - x_V\|} \cdot |S_D| \right) \\
& = -f(x_V) \cdot |V| - \sum_{S_N \subset \partial V} \alpha(x_V) \cdot \Phi_N(S_N) \cdot |S_N| \\
\Phi_D : & - \sum_{S_D \subset \partial V} \alpha(x_V) \cdot \frac{2\Phi_D(S_D)}{\|x_{ghost} - x_V\|} \cdot |S_D|
\end{aligned} \tag{3.26}$$

Additionally, $u(x_V)$ can be factored out of the sum:

$$\begin{aligned}
\text{band} : & \sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{u(x_{V_S})}{\|x_{V_S} - x_V\|} \cdot |S| \\
\text{diagonal} : & -u(x_V) \left(\sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{|S|}{\|x_{V_S} - x_V\|} + \sum_{S_D \subset \partial V} \alpha(x_V) \frac{2|S_D|}{\|x_{ghost} - x_V\|} \right) \\
B\text{-side} : & = -f(x_V)|V| - \sum_{S_N \subset \partial V} \alpha(x_V) \Phi_N(S_N) |S_N| - \sum_{S_D \subset \partial V} \alpha(x_V) \frac{2\Phi_D(S_D)}{\|x_{ghost} - x_V\|} |S_D|
\end{aligned} \tag{3.27}$$

The term B -side contains no unknowns. It consists solely of the heat source term for volume V and the constant terms of the boundary conditions. Therefore, a function $B(V)$ can be defined based on V :

$$B(V) = -f(x_V)|V| - \sum_{S_N \subset \partial V} \alpha(x_V) \Phi_N(S_N) |S_N| - \sum_{S_D \subset \partial V} \alpha(x_V) \frac{2\Phi_D(S_D)}{\|x_{ghost} - x_V\|} |S_D| \tag{3.28}$$

The term named diagonal contains only one unknown, $u(x_V)$, multiplied by constant geometric and material properties. Therefore, a function $A_{dia}(V)$ can be defined: ⁴

$$A_{dia}(V) = - \left(\sum_{S \subset \partial V} \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{|S|}{\|x_{V_S} - x_V\|} + \sum_{S_D \subset \partial V} \alpha(x_V) \frac{2|S_D|}{\|x_{ghost} - x_V\|} \right) \quad (3.29)$$

The term named band contains an unknown variable for each neighbor volume V_S . The remaining terms are constant, but cannot be factored out. Therefore a slightly different function $A_{ban}(V_S)$ is defined based on the neighboring volume V_S :

$$A_{ban}(V_S) = \frac{\alpha(x_V) + \alpha(x_{V_S})}{2} \cdot \frac{|S|}{\|x_{V_S} - x_V\|} \quad (3.30)$$

Applying these definitions to equation 3.27 leads to:

$$\left(\sum_{S \subset \partial V} A_{ban}(V_S) \cdot u(x_{V_S}) \right) + A_{dia}(V) \cdot u(x_V) = B(V) \quad (3.31)$$

This formula can now be applied to every individual volume V in the grid. $B(V)$ can therefore be written as source vector B of the linear matrix system: $A \cdot u = B$. Here, $A_{dia}(V)$ is the coefficient of the volume temperature $u(x_V)$, which means it outputs the diagonal entry of the matrix A in V 's row. The terms $A_{ban}(V_S)$ are coefficients of the respective neighbor volume temperatures $u(x_{V_S})$, forming the band entries of matrix A in V 's row. ⁵

The sum in function 3.31 can be rewritten explicitly for a 2D example with four neighbors (right, left, top, bottom [denoted as F for floor]):

$$\underbrace{A_{ban}(R) \cdot u(x_R)}_R + \underbrace{A_{ban}(L) \cdot u(x_L)}_L + \underbrace{A_{ban}(T) \cdot u(x_T)}_T + \underbrace{A_{ban}(F) \cdot u(x_F)}_F + \underbrace{A_{dia}(V) \cdot u(x_V)}_V = B(V) \quad (3.32)$$

Figure 3.2 (a) shows a 2D example of a small grid with 16 volumes. Volumes are indexed using the space filling z-curve of t8code. The matrix in (b) is constructed by applying formula 3.32 to each volume of the grid in (a) row by row, where the row describes the volume from its own perspective and the column describes the volume from the perspective of the other volumes. Volume 0 has two neighbors: volume 1 (right) and volume 2 (floor). Term $A_{ban}(R)$ (short R) is placed in row 0, column 1, because volume 1 is the right neighbor. Similarly, $A_{ban}(F)$ (short F) is placed in row 0, column 2. Term $A_{dia}(V)$ (short V) is always placed on the diagonal (row 0, column 0). Column 0 also contains two non-diagonal entries. From

⁴Text passage reviewed by Qwen 3.5-122B.

⁵Text passage reviewed by Qwen 3.5-122B.

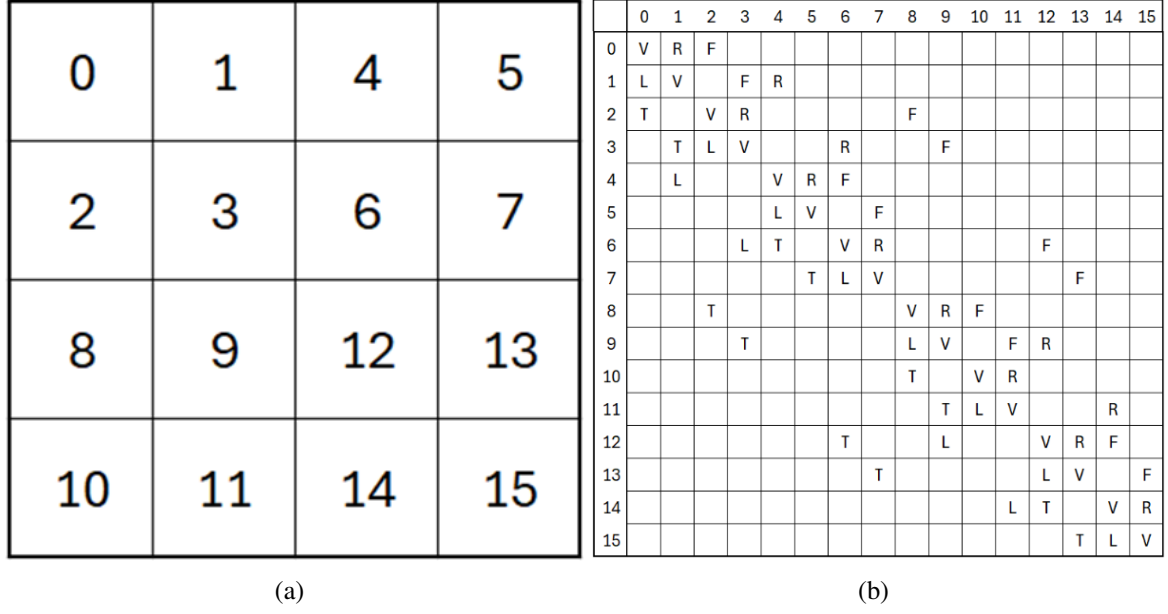


Figure 3.2: A 2D grid with z-curve indexed volumes (a) and its representative matrix A (b).

the perspective of volume 1, volume 0 is its left neighbor (row 1, column 0 with entry L) and from the perspective of volume 2, volume 0 is its top neighbor (row 2, column 0 with entry T). Volume 0 also has two faces without neighbors, the top and left boundary. The boundary conditions are already accounted for in the calculation of $A_{dia}(V)$ and $B(V)$, therefore not creating additional matrix entries. The terms $A_{ban}(T)$ (short T) and $A_{ban}(L)$ (short L) are effectively zero for volume 0 and therefore also do not show up in matrix A . This process is repeated for all volumes until matrix A is fully assembled.

3.2 Multigrid

A multigrid solver consists of multiple components, each with its own set of approaches and configurations to influence the solver's efficiency. There are no strict guidelines or simple rules for selecting optimal ones (Trottenberg et al., 2001, Page 40). Consequently, each component and cycle is explained in the corresponding chapter, with a discussion provided for the choices made in this thesis. ⁶

⁶Text passage reviewed by Qwen 3.5-122B.

3.2.1 Defect Function

As the exact solution u of $A \cdot u = B$ is unavailable, the true error cannot be calculated. Instead, the residual vector res is used as a measure of the defect, with the definition as in chapter 2.3 formula 2.2:

$$res(B, A, u) = B - A \cdot u \quad (3.33)$$

The residual quantifies the local imbalance in the physical conservation law. This means the residual of each volume V describes the heat over- or underflow of the current temperature of the control volume $u(x_V)$.

3.2.2 Smoothing Function

During the transfer from a fine grid to a coarser one, high-frequency error components must be accounted for to prevent aliasing. While low-frequency error components can be accurately represented on coarser grids, high-frequency components may alias during the restriction process of the residual ([Trottenberg et al., 2001](#), Page 17).⁷

Figure 3.3 shows this effect in a 1D setting. The blue curve in the upper graph represents the residual noise on the fine grid without prior smoothing. When this residual is directly restricted to the coarser grid, high-frequency components are lost. The resulting interpolated residual is shown as the red curve. A comparison of both curves shows that the restricted profile differs from the original one. If the residual remains unsmoothed and is restricted, the multigrid solver attempts to correct a profile on the coarser grid that does not represent the fine-grid defect, leading to inefficient convergence and increased computation time.

However, if a smoothing step is applied to the fine-grid residual beforehand, the high-frequency oscillations are damped. The smoother finds a temperature that is close to the current one, where the residual gets reduced and high-frequency components are removed. This is shown in the bottom graphs of Figure 3.3. When this smoothed residual (blue) is restricted to the coarse grid (red), the resulting profile closely matches the fine-grid behavior. Since low-frequency errors are resolved on coarser grids, this process reduces information loss. This initial smoothing step is known as pre-smoothing. After the residual correction is computed on the coarse grid and interpolated or prolonged back to the fine grid, high-frequency errors are reintroduced by the interpolation process. Post-smoothing is then applied to dampen these errors once again and to set up the residual or current temperature solution for the next transfer.⁸

⁷Text passage reviewed by Qwen 3.5-122B.

⁸Text passage reviewed by Qwen 3.5-122B.

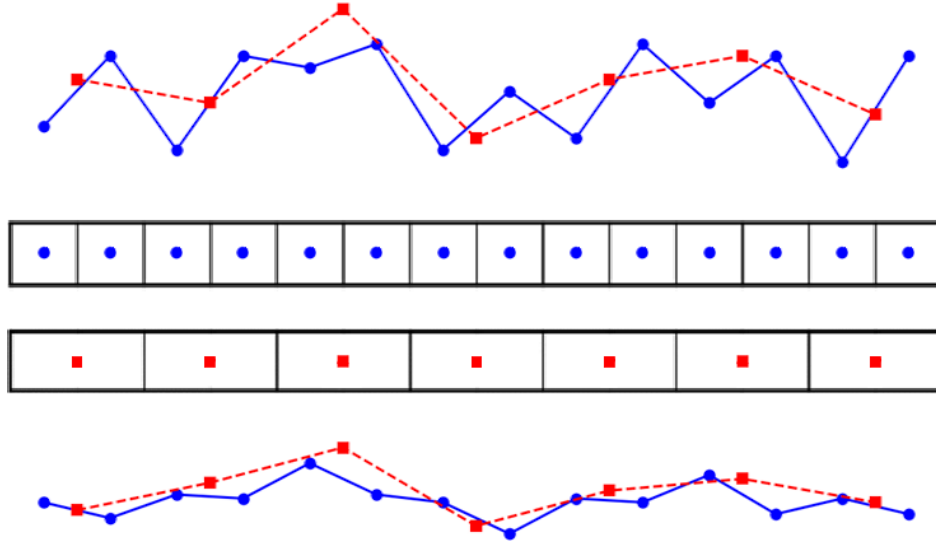


Figure 3.3: A 1D example of the aliasing and smoothing effect. The top graphs show residual noise (blue) on the fine grid (blue centers) being restricted (red) to the coarse grid (red centers) without prior smoothing. The bottom graphs show a comparison of the same process after prior smoothing of the residual.

The book (Trottenberg et al., 2001, Page 34) recommends two Gauss-Seidel methods for a good to very good smoothing factor. In this thesis, the Lexicographic Gauss-Seidel (GS-LEX) smoother is preferred over the Red-Black variant (GS-RB), despite GS-RB promising smoothing quality, compared in table 3.1.

Method	Smoothing Factor	Quality	Parallelism
GS-LEX	0.5	Good	$\leq \sqrt{N}$
GS-RB	0.25	Very Good	$\frac{N}{2}$

Table 3.1: Comparison of Gauss-Seidel smoothers, with N as the number of volumes in grid G (Trottenberg et al., 2001, Page 34).

The reason for this decision lies in the space filling z-curve used by t8code 1.2.2. It inherently provides the lexicographic ordering of the volumes for the GS-LEX smoother, allowing it to operate along its sequential indexing, requiring no additional overhead. The GS-RB scheme requires the grid volumes to be ordered in a checkerboard pattern. Reordering the indices to maintain this pattern would lead to setup costs when using z-curve indexing. While all of these points are factors for using GS-RB, the main reason for not using it is incompatibility with adaptive grids, a main focus of this thesis. The checkerboard pattern is not guaranteed to exist on adaptively refined meshes, where GS-LEX works without adjustments, due to the inherent properties of the z-curve.⁹

⁹Text passage reviewed by Qwen 3.5-122B.

The sparse matrix system has the following form:

$$\underbrace{\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & 0 & 0 & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & 0 & 0 & 0 \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{N-1,N-2} & a_{N-1,N-1} & a_{N-1,N} \\ 0 & 0 & 0 & \cdots & a_{N,N-2} & a_{N,N-1} & a_{N,N} \end{pmatrix}}_{\text{Matrix } A} \cdot \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}}_{\text{Vector } u} = \underbrace{\begin{pmatrix} B_1 \\ B_2 \\ B_3 \\ \vdots \\ B_{N-1} \\ B_N \end{pmatrix}}_{\text{Vector } B} \quad (3.34)$$

Where $n, m \in \{1, \dots, N\}$, with N being the number of volumes in grid G . Note that the right-hand side contains the vector r , which represents either the source vector B or the restricted residual from a finer grid.

To smooth this system with the GS-LEX method, formula 3.35 is applied sequentially along the z-curve for each volume (Herrmann, 1983, Page 45), iterating from $n = 1$ to $n = N$:

$$u_n^{(l)} = \frac{1}{a_{n,n}} \cdot \left(r_n - \sum_{j=1}^{n-1} a_{n,j} \cdot u_j^{(l)} - \sum_{j=n+1}^N a_{n,j} \cdot u_j^{(l-1)} \right) \quad (3.35)$$

It is important to note that this formula contains two summation terms, one where $j < n$ and another where $j > n$. During the iteration, some neighboring volumes may have already been updated in the current step $u_j^{(l)}$, while others $u_j^{(l-1)}$ have not. Here, $u_n^{(l)}$ denotes the value of u at index n after l smoothing iterations.

A disadvantage of this GS-LEX method is that the error reduction is directional. The defect tends to propagate along the z-curve. Therefore, volumes at the beginning of the indexing sequence or at the system boundary often have a smaller defect compared to those at the end. Figure 3.4 shows this effect in a problem with fully symmetrical conditions. In this example, the z-curve starts at the bottom-left corner and ends at the top-right. The distribution of temperature errors is shifted in this direction. It is also visible that the error is lowest at the boundary. This is caused by the applied Dirichlet condition, which fixes the temperature of the ghost volumes. This effectively provides the boundary volumes with neighbors that have a strictly defined temperature, thereby enhancing the accuracy of the smoothing process in those regions.¹⁰

To combat this asymmetry, equation 3.35 is applied twice per smoothing cycle, once forwards from 1 to N with $l = 1$ and once in reverse from N to 1 with $l = 2$. Note that when GS-LEX

¹⁰Text passage reviewed by Qwen 3.5-122B.

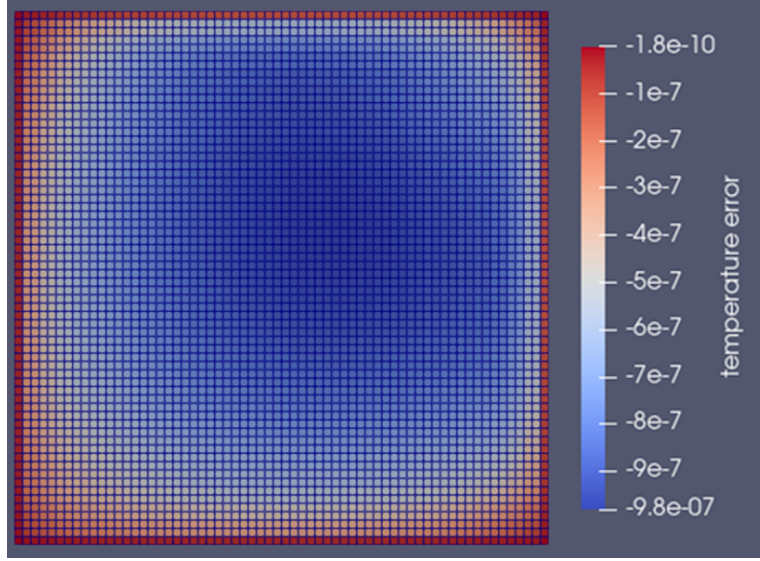


Figure 3.4: Visible shifting of the error when smoothing is performed only along the z -curve in a fully symmetrical problem.

runs in reverse, the roles of the old values $u_j^{(l-1)}$ and new values $u_j^{(l)}$ must be swapped.

$$\begin{aligned}
 \text{Forward pass } (l = 1) : \quad u_n^{(1)} &= \frac{1}{a_{n,n}} \cdot \left(r_n - \sum_{j=1}^{n-1} a_{n,j} \cdot u_j^{(1)} - \sum_{j=n+1}^N a_{n,j} \cdot u_j^{(0)} \right) \\
 \text{Reverse pass } (l = 2) : \quad u_n^{(2)} &= \frac{1}{a_{n,n}} \cdot \left(r_n - \sum_{j=1}^{n-1} a_{n,j} \cdot u_j^{(1)} - \sum_{j=n+1}^N a_{n,j} \cdot u_j^{(2)} \right)
 \end{aligned} \tag{3.36}$$

The vector resulting from the second pass is smooth (Trottenberg et al., 2001, Page 34), gets defined as $u_{smooth} = (u_1^{(2)}, \dots, u_N^{(2)})$ and therefore satisfies the smoothing operator definition 2.3 in chapter 2.3:

$$\Psi_{A,r}(u) = (u_1^{(2)}, \dots, u_N^{(2)}) = u_{smooth} \tag{3.37}$$

3.2.3 Coarse Grid Correction

Chapter 2.3 briefly introduced the concept of coarse grid correction and the restriction operator Θ . The objective is to construct a matrix system for the next coarser grid based on the residual and the matrix of the current grid. This process constructs a new matrix and a new right-hand side vector for the coarser grid. The definition of the operator 2.6 needs to be more generalized:

$$\Theta^{(\phi)} \left(A^{(\phi)}, \text{res} \left(B^{(\phi)}, A^{(\phi)}, u_{smooth}^{(\phi)} \right) \right) = \left(A^{(\phi+1)}, \left(\text{res}(B^{(\phi)}, A^{(\phi)}, u_{smooth}^{(\phi)}) \right)^{(\phi+1)} \right) \tag{3.38}$$

Where $\phi \in \{0, \dots, C-1\}$ denotes the rank in the grid hierarchy, with C being the total number of grids and $G^{(C)}$ the coarsest grid. The finest grid is denoted as $G^{(0)}$. Its components, such as matrix A , are denoted as $A^{(0)}$. Similarly, $B^{(0)}$, $u^{(0)}$ and $res^{(0)}$ refer to the corresponding vectors and operations of $G^{(0)}$. $\Theta^{(\phi)}$ describes the coarse grid correction from $G^{(\phi)}$ to $G^{(\phi+1)}$. To simplify the notation for the restricted residual, the following definition gets applied: ¹¹

$$\left(res(B^{(\phi)}, A^{(\phi)}, u_{smooth}^{(\phi)}) \right)^{(\phi+1)} = res_{(\phi)}^{(\phi+1)} \quad (3.39)$$

Substituting this into the previous equation 3.38 results in:

$$\Theta^{(\phi)} \left(A^{(\phi)}, res \left(B^{(\phi)}, A^{(\phi)}, u_{smooth}^{(\phi)} \right) \right) = \left(A^{(\phi+1)}, res_{(\phi)}^{(\phi+1)} \right) \quad (3.40)$$

In literature (Trottenberg et al., 2001, Page 417), there are two main approaches for coarse grid correction, denoted as $\Theta^{(\phi)}$, the geometric and the algebraic approach. As the name implies, the algebraic approach constructs the coarse grid matrix solely from the data in the finer grid matrix. This results in matrices well-suited to ensuring solver robustness, even when faced with unknown or highly irregular grid structures. In contrast, the geometric approach leverages the known grid structure to recalculate physical fluxes from the actual volume geometry by applying FVM directly on the coarser grid. This ensures that the coarse-grid operator reflects the exact physics of the problem at hand, while utilizing t8code for efficient residual restriction, solution prolongation and Full Multigrid (FMG) interpolation. Due to these advantages in physical consistency and computational efficiency, the geometric approach is selected for the purposes of this thesis. ¹²

Creating the matrix system with the FVM for the finest grid $G^{(0)}$ results in the following equation:

$$A^{(0)} \cdot u_{ini}^{(0)} = B^{(0)} \quad (3.41)$$

$u_{ini}^{(0)}$ represents the initial guess for the temperature. This is typically a zero vector or the solution $u_{solved}^{(0)}$ from the previous time step of the simulation. Smoothing this equation 3.41 with the GS-LEX method results in:

$$A^{(0)} \cdot \Psi_{A^{(0)}, B^{(0)}}^{(0)}(u_{ini}^{(0)}) = A^{(0)} \cdot u_{smooth}^{(0)} = B^{(0)} \quad (3.42)$$

The residual is calculated afterwards with formula 3.33:

$$res \left(B^{(0)}, A^{(0)}, u_{smooth}^{(0)} \right) = B^{(0)} - A^{(0)} \cdot u_{smooth}^{(0)} \quad (3.43)$$

Coarse grid correction is now applied to both the residual and the matrix $A^{(0)}$ with the

¹¹Text passage reviewed by Qwen 3.5-122B.

¹²Text passage reviewed by Qwen 3.5-122B.

operation 3.38:

$$\Theta^{(0)} \left(A^{(0)}, res \left(B^{(0)}, A^{(0)}, u_{smooth}^{(0)} \right) \right) = \left(A^{(1)}, res_{(0)}^{(1)} \right) \quad (3.44)$$

$A^{(1)}$ is built using the FVM on $G^{(1)}$. This process also provides $B^{(1)}$ as a byproduct, which is not needed in this chapter but is necessary for other functions of later chapters.

Before performing the coarse grid correction on the residual, it is essential to understand what the residual represents and how it is treated during restriction to ensure the conservation of energy. The residual represents the local violation of the heat balance in each volume. This imbalance must be preserved when combining child volumes into a parent volume during the coarsening process. This concept is best explained with the example in Figure 3.5.¹³

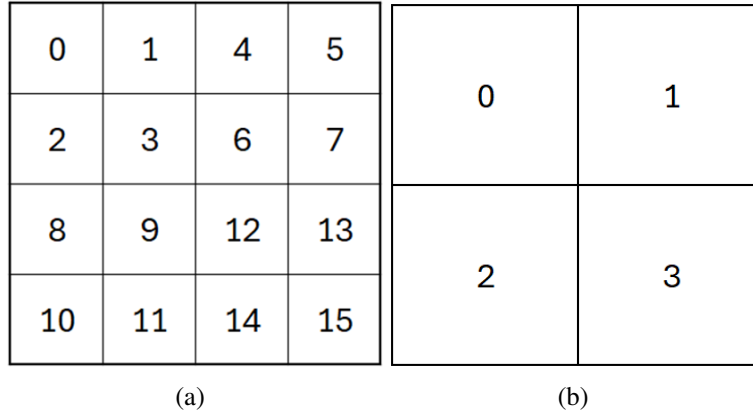


Figure 3.5: Two 2D example Grids: (a) Grid $G^{(0)}$. (b) Grid $G^{(1)}$.

The 2D grids in figure 3.5 represent $G^{(0)}$ (a) and $G^{(1)}$ (b) and share the same physical conditions. Volume 0 from $G^{(1)}$ is the parent volume of the child volumes 0, 1, 2 and 3 from grid $G^{(0)}$. Assuming each child volume has a residual of $2K$, meaning they have a combined $8K$ heat overflow in their region. This results in the same surplus for the parent volume, as its region is the same the child volumes'. To conserve this heat overflow, the residuals of the child-volumes are summed up. The standard mathematical approach involves finding a restriction matrix $R^{(0)}$ such that:

$$res_{(0)}^{(1)} = R^{(0)} \cdot res \left(B^{(0)}, A^{(0)}, u_{smooth}^{(0)} \right) \quad (3.45)$$

Here, $R^{(0)}$ is a staircase or aggregation matrix, as shown in equation 3.46 for figure 3.5. It contains entries of 1 at positions (m, n) , where m is the index of the parent volume in $G^{(1)}$ and n is the index of the corresponding child volume in $G^{(0)}$, with all other entries being 0. Since t8code maintains the connectivity between parent and child volumes, it can directly sum the residuals for the parent volume. The computational speed of this operation is equivalent to

¹³Text passage reviewed by Qwen 3.5-122B.

that of a sparse matrix multiplication in equation 3.45, since it effectively performs the same calculation.¹⁴

$$\underbrace{\begin{pmatrix} res_{(0)}^{(1)} \\ res_1^{(1)} \\ res_2^{(1)} \\ res_3^{(1)} \end{pmatrix}}_{res_{(0)}^{(1)}} = \underbrace{\begin{pmatrix} 1 & 1 & 1 & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & 1 & 1 & 1 & 1 & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & 1 & 1 & 1 & 1 & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & . & 1 & 1 & 1 & 1 \end{pmatrix}}_{R^{(0)}} \cdot \underbrace{\begin{pmatrix} res_0 \\ res_1 \\ \vdots \\ res_{14} \\ res_{15} \end{pmatrix}}_{res(B^{(0)}, A^{(0)}, u_{smooth}^{(0)})} \quad (3.46)$$

Applying formula 3.44 results in the matrix system for the next coarser grid $G^{(1)}$:

$$A^{(1)} \cdot u_{ini}^{(1)} = res_{(0)}^{(1)} \quad (3.47)$$

The system generated by the coarse grid correction $\Theta^{(\phi)}$ is always initialized with $u_{ini}^{(\phi+1)} = 0$. This is done, because resulting formulas like 3.47 are used to solve the matrix system of the next finest grid $G^{(\phi)}$, by finding $u_{solved}^{(\phi+1)}$. $u_{ini}^{(\phi+1)}$ represents the correction for the system of $G^{(\phi+1)}$, which is unknown up to this point and therefore set to zero.

Note that the coarse grid correction from $G^{(0)}$ to $G^{(1)}$ differs from other restrictions. Only the matrix system of $G^{(0)}$ contains the source term vector B . All other matrix systems follow the structure:

$$A^{(\phi)} \cdot u_{ini}^{(\phi)} = res_{(\phi-1)}^{(\phi)} \quad (3.48)$$

The coarse grid correction can also be written in a generalized way to be performed on coarser grids. First, the smoother is applied like before, see formula 3.42:

$$A^{(\phi)} \cdot \Psi_{A^{(\phi)}, B^{(\phi)}}^{(\phi)}(u_{ini}^{(\phi)}) = A^{(\phi)} \cdot u_{smooth}^{(\phi)} = res_{(\phi-1)}^{(\phi)} \quad (3.49)$$

The Residual is calculated as in formula 3.43:

$$res(res_{(\phi-1)}^{(\phi)}, A^{(\phi)}, u_{smooth}^{(\phi)}) = res_{(\phi-1)}^{(\phi)} - A^{(\phi)} \cdot u_{smooth}^{(\phi)} \quad (3.50)$$

This residual and the matrix $A^{(\phi)}$ get coarse grid corrected by the operation 3.38 like in formula 3.44:

$$\Theta^{(\phi)}(A^{(\phi)}, res(res_{(\phi-1)}^{(\phi)}, A^{(\phi)}, u_{smooth}^{(\phi)})) = (A^{(\phi+1)}, res_{(\phi)}^{(\phi+1)}) \quad (3.51)$$

¹⁴Text passage reviewed by Qwen 3.5-122B.

Arriving at the Matrix system of $G^{\phi+1}$:

$$A^{(\phi+1)} \cdot u_{ini}^{(\phi+1)} = res_{(\phi)}^{(\phi+1)} \quad (3.52)$$

3.2.4 Fine Grid Correction

Chapter 2.3 also introduced the fine grid correction or prolongation. The idea behind this is to interpolate the solution $u_{solved}^{(\phi+1)}$ from the coarser grid $G^{(\phi+1)}$ back to the finer grid $G^{(\phi)}$ and apply the prolonged values as a correction to the current smoothed solution $u_{smooth}^{(\phi)}$. The definition of the fine grid operator 2.8 needs to be more generalized in order for this approach as well:

$$\Xi^{(\phi+1)} \left(u_{solved}^{(\phi+1)} \right) = u_{(\phi+1)}^{(\phi)} \quad (3.53)$$

Assuming the matrix system has been solved (or sufficiently smoothed) on the coarsest grid $G^{(C)}$:

$$A^{(C)} \cdot u_{solved}^{(C)} = res_{(C-1)}^{(C)} \quad (3.54)$$

This solution $u_{solved}^{(C)}$ then gets fine grid corrected by the operation $\Xi^{(C)}$ defined in equation 3.53:

$$\Xi^{(C)} \left(u_{solved}^{(C)} \right) = u_{(C)}^{(C-1)} \quad (3.55)$$

The transfer of the temperature correction during fine grid correction differs physically from the residual transfer during the coarse grid correction. While the residual represents a local heat imbalance, which must be summed to conserve energy, u_{solved} represents a temperature correction that scales with the volume. Therefore, this temperature correction is not an overflow or underflow. To illustrate this with an example, the grids from figure 3.5 are reused here, with the additional assumption that $C = 1$ with $G^{(1)} = G^{(C)}$.¹⁵

If the parent volume 0 of the coarse grid $G^{(1)}$ has a temperature correction term of $2K$, the temperature in this large volume must be increased by $2K$. This increase must be transferred to the child volumes of $G^{(0)}$. The transfer from $G^{(1)}$ (b) to $G^{(0)}$ (a) conserves energy if the correction value of the parent is equal to the average correction value of its child volumes 0 to 3. The simplest way to ensure this is by copying the temperature correction value from the parent volume to each of its child volumes. This process uses the transposed restriction matrix $\left(R^{(C-1)} \right)^T$ of the coarse grid correction:

$$\left(R^{(C-1)} \right)^T \cdot u_{solved}^{(C)} = u_{(C)}^{(C-1)} \quad (3.56)$$

¹⁵Text passage reviewed by Qwen 3.5-122B.

$\left(R^{(C-1)}\right)^T$ is also an aggregation matrix, as shown in equation 3.57 for the example in figure 3.5. It contains entries of 1 at positions (m, n) , where m is the index of a child volume in $G^{(C-1)}$ and n is the index of the corresponding parent volume in $G^{(C)}$, with all other entries being 0. The transfer functionalities of t8code can be used here as well, like in chapter 3.2.3:

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{(R^{(C-1)})^T} \cdot \underbrace{\begin{pmatrix} u^{(C)}_0 \\ u^{(C)}_1 \\ u^{(C)}_2 \\ u^{(C)}_3 \end{pmatrix}}_{u^{(C)}_{solved}} = \underbrace{\begin{pmatrix} u^{(C-1)}_0 \\ \vdots \\ u^{(C-1)}_3 \\ u^{(C-1)}_4 \\ \vdots \\ u^{(C-1)}_7 \\ u^{(C-1)}_8 \\ \vdots \\ u^{(C-1)}_{11} \\ u^{(C-1)}_{12} \\ \vdots \\ u^{(C-1)}_{15} \end{pmatrix}}_{u^{(C-1)}_{(C)}} \quad (3.57)$$

The resulting vector $u^{(C-1)}_{(C)}$ represents the temperature correction for the matrix system of grid $G^{(C-1)}$. This correction must be added to the current smoothed solution $u^{(C-1)}_{smooth}$ to obtain a new initial guess for the updated matrix system:

$$u^{(C-1)}_{(C)} + u^{(C-1)}_{smooth} = u^{(C-1)}_{ini} \quad (3.58)$$

Applying the prolongation from equation 3.55 combined with the correction from equation 3.58 results in the updated matrix system for the finer grid $G^{(C-1)}$:

$$A^{(C-1)} \cdot u^{(C-1)}_{ini} = res^{(C)}_{(C-1)} \quad (3.59)$$

The new initial guess $u^{(C-1)}_{ini}$ reduces the residual compared to the previous solution temperature $u^{(C-1)}_{smooth}$, but also reintroduces residual noise. Therefore, post-smoothing is performed after each fine grid correction. This procedure is equal for all fine grid corrections from grid $G^{(\phi+1)}$ to $G^{(\phi)}$. Note that when performing the final fine grid correction from $G^{(1)}$ to $G^{(0)}$ using $\Xi^{(1)}$, the matrix system of $G^{(0)}$ contains the source term $B^{(0)}$ on its right-hand side:

$$A^{(0)} \cdot \left(\Xi^{(1)} \left(u^{(1)}_{smooth} \right) + u^{(0)}_{smooth} \right) = A^{(0)} \cdot \left(u^{(0)}_{(1)} + u^{(0)}_{smooth} \right) = A^{(0)} \cdot u^{(0)}_{ini} = B^{(0)} \quad (3.60)$$

3.2.5 Multigrid Cycles

The last major component that needs to be chosen is the multigrid cycle, which chains all the above operations together. In chapter 2.3 a small example of a multigrid V-cycle is shown in figure 2.5, in which each major operation is used once. Performing multiple iterations of this cycle with a decent convergence factor will lead to a sufficient solution of the initial problem (Trottenberg et al., 2001, Page 45).

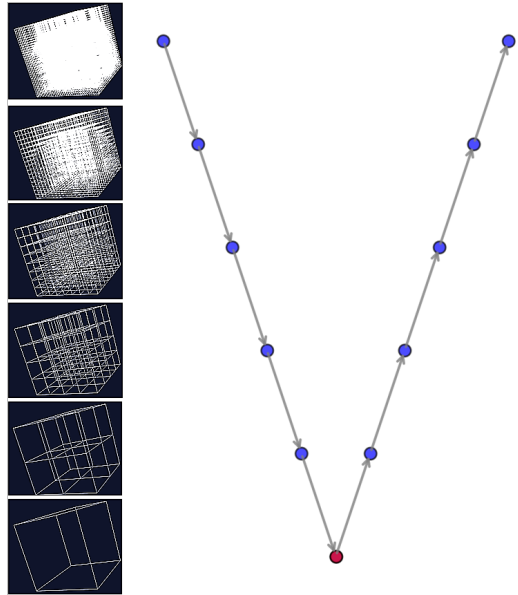


Figure 3.6: A Multigrid V-cycle with 6 grids.

Expanding this simple cycle to a multigrid with more than 2 grids results in a flowchart that looks V-shaped, which gave this cycle its name. Figure 3.6 shows a multigrid V-cycle with 6 grids. Pre-smoothing iterations are displayed by blue dots in the flowchart. Coarse grid corrections are indicated by downward pointing arrows and solving the matrix system on the coarsest grid $G^{(C)}$ is indicated by red dots. This is followed by fine grid corrections, indicated by the upward pointing arrows and post-smoothing iterations, indicated by blue dots. This is a variation of a family of cycles referred to as the γ -cycle family in (Trottenberg et al., 2001, Page 46). $\gamma \in \mathbb{N}$ stands for the cycle index, which is 1 in this example. Increasing γ increases the repetition of certain coarse and fine grid corrections with a pre- and post-smoothing step in between, changing the flowchart form into W-shape. ¹⁶

Figure 3.7 shows a small overview of the γ -cycle family. In practice the relevant cycle index γ is often 1 or 2 (Trottenberg et al., 2001, Page 46), (John, 2013, Page 38).

Another cycle type is the F-cycle. It has similarities to the $\gamma = 2$ cycles with the exception that it prolongates from the coarsest grid up to a finer grid it has not yet reached. Afterwards,

¹⁶Text passage reviewed by Qwen 3.5-122B.

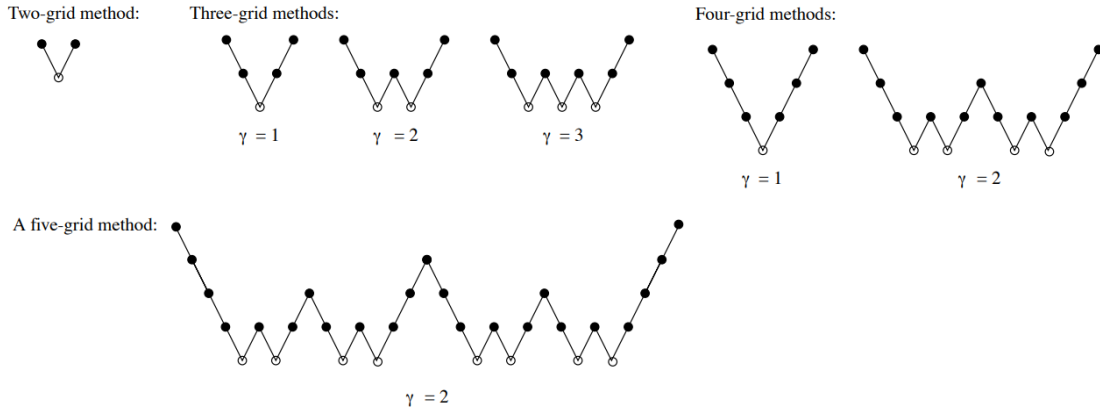


Figure 3.7: Multigrid γ -cycles by (Trottenberg et al., 2001, Page 46).

it repeats the coarse grid correction back down again to the coarsest grid. It requires fewer coarse and fine grid corrections than the $\gamma = 2$ cycles, as shown in figure 3.8.

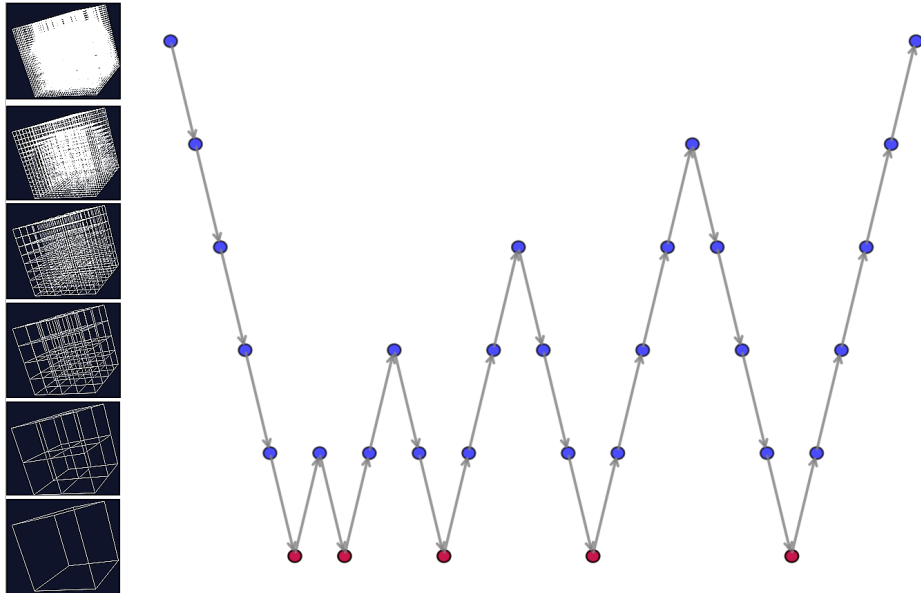


Figure 3.8: A Multigrid F-cycle with 6 grids.

A different method resulting in a similar shape, but very distinct in the theoretical operation speed and methodology is the full multigrid (FMG) cycle (Trottenberg et al., 2001, Page 56). The main difference being, that the FMG-cycles begin at the coarsest grid instead of the finest, skipping the first wave of coarse grid corrections and pre-smoothings. Nested iterations are used in order to solve the matrix system on the coarsest grid, which is faster than operating on the finer grids, as the system of equations is smaller (Trottenberg et al., 2001, Page 56). In figure 3.9 $G^{(C)}$ is a single cube, with a matrix system size of 1 equation. This can also be used as the initial guess $u_{ini}^{(4)}$ for the finer grid $G^{(4)}$ after fine grid correction. The difference here being that no correction is applied as would be the case in formula 3.58, since $u_{smooth}^{(4)}$ does not exist yet. This is an efficient way to get an initial solution, closer to the average of the real solution than a guess or assuming a zero vector. This procedure can be

repeated until the finest grid $G^{(0)}$ is reached. $u_{solved}^{(5)}$ is as accurate as theoretically possible in this discretization scheme of $G^{(5)}$. The book (Trottenberg et al., 2001, Page 56) explains that FMG is asymptotically optimal. This means the number of iterations grows linearly with the number of volumes in $G^{(0)}$.¹⁷

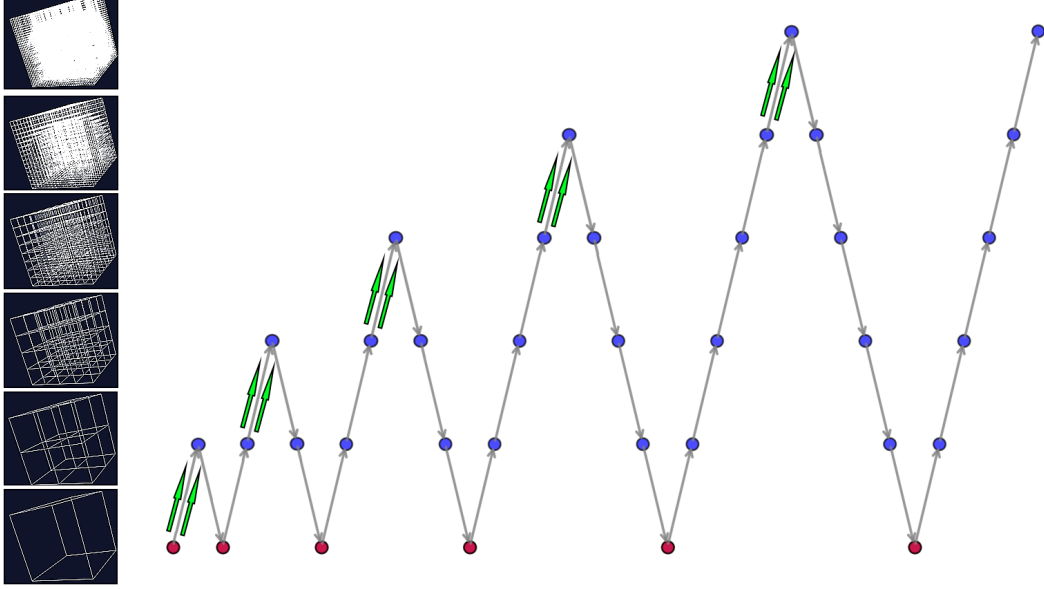


Figure 3.9: A Multigrid FMG-V-cycle with 6 grids.

Starting on the coarsest grid $G^{(C)}$ introduces the problem that $res_{(C-1)}^{(C)}$ doesn't exist yet. The grid construction needs to be changed. Since there is no residual from previous grids, $B^{(C)}$, which is created as a byproduct of the FVM, is used as the right-hand side of the matrix system, since grid $G^{(C)}$ is regarded as the grid to be solved:

$$A^{(C)} \cdot u_{ini}^{(C)} = B^{(C)} \quad (3.61)$$

$u_{ini}^{(C)}$ starts with a zero-vector as well, since getting to $u_{solved}^{(C)}$ is just a single operation in a 1×1 matrix system, assuming $G^{(C)}$ is as coarse as possible. Then, instead of performing a regular fine grid correction $\Xi^{(C)}(u_{solved}^{(C)})$ a special FMG-interpolation $\Xi_{FMG}^{(C)}$ is required with a higher accuracy:

$$\Xi_{FMG}^{(C)}(u_{solved}^{(C)}) = u_{(C)}^{(C-1)} = u_{ini}^{(C-1)} \quad (3.62)$$

This operation is visualized in figure 3.9 by the double green arrows. The regular fine grid correction does not need any neighbor information for its transfer. To improve the interpolation, geometric information, which is stored in neighboring volumes, is required. The interpolation should include as few neighbors as possible, since accessing them is expensive. The interpolation chosen in this thesis uses a barycentric approach (Möbius, 1827, Chapter 3), which is used by GAIA for similar interpolations.

¹⁷Text passage reviewed by Qwen 3.5-122B.

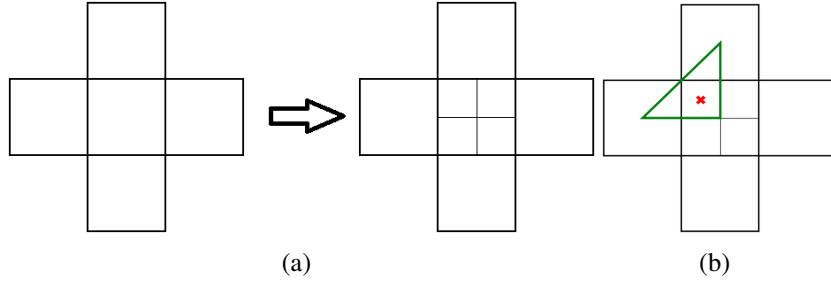


Figure 3.10: 2D example of FMG-interpolation: (a) Refinement of the inner volume with respect to the neighboring volumes. (b) Barycentric interpolation towards a child Volume with the help of its parent volume and its neighboring volumes.

To reduce the amount of neighbor volume accesses to 1 per volume, it is used once on the parent volume of grid $G^{(\phi)}$. In figure 3.10 (a) shows a 2D volume with its surrounding neighbors that need to be accessed. Afterwards, the central volume gets refined from $G^{(\phi)}$ to $G^{(\phi-1)}$. (b) shows the top-left child volume's center with a red X, the interpolation point. $u_{solved}^{(\phi)}$ of the neighboring volumes and the parent volume are used for the interpolation onto each of the child volumes. In 2D the barycentric interpolation needs 3 starting points. The first two points are the centers of the parent's neighbors that are closest to the child volume and the parents center is the third point. These interpolation points are connected by a green triangle in the figure. Barycentric interpolation is used to approximate the temperature correction values of these 3 points onto the child volume. This is repeated for all other children. This approach can be applied in 3D by adding the third closest neighbor and also works adapted grids.

$$A^{(C-1)} \cdot u_{ini}^{(C-1)} = B^{(C-1)} \quad (3.63)$$

Solving the matrix system 3.63, created by the FVM on grid $G^{(C-1)}$ with a V or F-cycle, results in:

$$A^{(C-1)} \cdot u_{solved}^{(C-1)} = B^{(C-1)} \quad (3.64)$$

The FMG-interpolation is used each time to reach a grid that has not yet been reached. After each FMG-interpolation a regular F- or γ -cycle is performed to solve the created matrix system. In the example from figure 3.9 it is a single V-cycle.

3.3 Adaptive Multigrid

The adaptive multigrid method extends the concept of the standard multigrid solver by allowing grids to contain volumes of varying sizes. This flexibility enables the grid to be

tailored towards the specifics of the physical or environmental problem. There are two types of grid refinement approaches used in multigrid: predefined and self-adaptive (Trottenberg et al., 2001, Page 356). Choosing how each grid will be constructed and discretized before starting to solve it is called predefined refinement. While the self-adaptive approach will be partially theoretically explored, the focus of this thesis will be the predefined approach. The self-adaptive approach changes the grid structure and the discretization are dynamically. This allows further customization to account for unwanted behavior, like local singularities. This approach is outside the scope of this thesis, because of its complexity.¹⁸

The paper (Saber et al., 2020, Chapter 3) shows a geometric adaptive multigrid approach, which is also applied in the context of solving Poisson problems, see figure 3.11. The initial grid, here called Ω_{e,h_0} , is refined three times to Ω_{e,h_3} . The reasoning behind this is that some grid volumes are only partly inside the physical domain. These volumes are called cut cells. In figure 3.11, the domain is a circle in the middle of the grid. It is embedded into a larger background domain, which is represented by the entirety of the grid. The grid is refined locally around the cut cells to better capture the shape of the domain, making numerical integration more accurate. This avoids problems with the matrix system that can occur when a volume only contains a very small part of the domain.

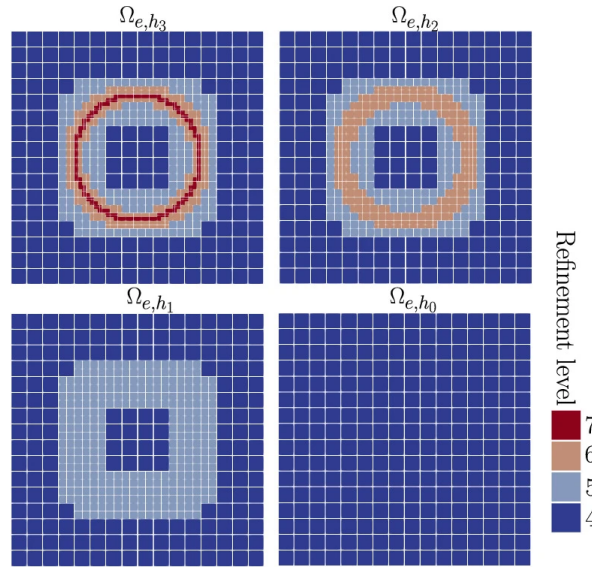


Figure 3.11: The geometric adaptive refinement approach, with a locally refined grid around cut cells to accurately capture the circular geometry by (Saber et al., 2020, Chapter 3).

Figure 3.12 shows two options to separate volumes of an adapted grid into domains (Qin et al., 2025, Chapter 3.1). These approaches are called global coarsening (b) and local smoothing (c) respectively. They can either be used to smooth the entire grid in the case of global coarsening with the trade-off of higher computational cost, or to focus the computational power on the finer volumes in the case of local smoothing, by excluding coarser volumes

¹⁸Text passage reviewed by Qwen 3.5-122B.

and using them as a boundaries. In (a) one can see an adapted grid. The global coarsening approach (b) always includes all volumes of each grid into the domain that gets calculated. This is shown by the gray area. In the local smoothing method, only the finest volumes are considered for the domain to be smoothed, also indicated by the gray area. Additionally, a refinement edge is declared at the boundary of the chosen domain, where boundary conditions need to be defined depending on the neighboring volumes.¹⁹

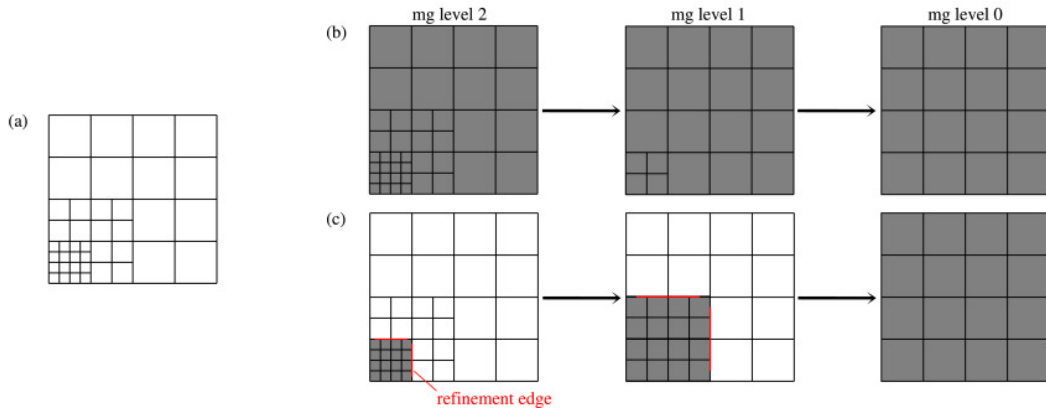


Figure 3.12: Strategies for domain partitioning in adaptive grids. (a) Adaptively refined grid. (b) Global coarsening includes all volumes per level. (c) Local smoothing restricts computation to the finest volumes, with coarser volumes as boundaries. (Qin et al., 2025, Chapter 3.1).

Notable here is that two different ways to handle simultaneous coarsening are shown. In (b) all volumes are adapted simultaneously, leading to all child volumes that are not at the lowest level being coarsened. In (c) only the volumes with the lowest level are coarsened, leaving all coarser volumes unchanged.

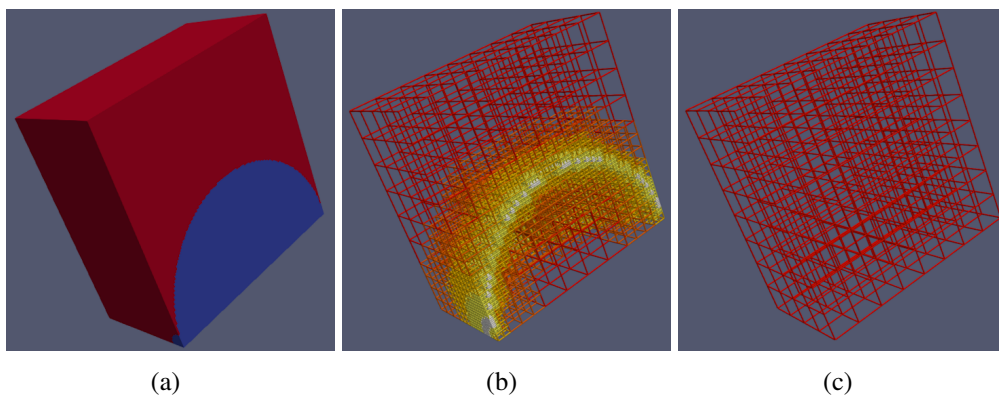


Figure 3.13: Multi-Level Adaptive Technique (MLAT) used on a 3D cube grid with a cross-section view. (a) Distribution of the blue and red region. (b) Locally refined finest grid $G^{(0)}$. (c) Coarse starting grid $G^{(C)}$.

During the Multi-Level Adaptive Technique (MLAT), described by (Baranoski et al., 2002,

¹⁹Text passage reviewed by Qwen 3.5-122B.

Page 357), the multigrid solver does not start on the finest grid $G^{(0)}$, which is similar to the FMG cycle. Instead, a coarser grid is chosen. To better explain this process, it is applied to the test setup from figure 2.2 with differing diffusion coefficients α in the red and blue regions in figure 3.13 (a).

The coarsest grid $G^{(C)}$ chosen for the MLAT is shown in (c). Solving the problem on that grid would not provide a high accuracy of $u_{solved}^{(0)}$, because the residuals of volumes near the edge between the red and blue region would have to be assumed to still be substantial. Therefore, refinement is only necessary in this region, leading to the grid in (b). (Baranoski et al., 2002, Page 357) claims that the accuracy of a solution on a locally refined grid can be similar to the accuracy of the solution of the fully refined grid. For the creation of the in-between grids and domain calculation, the methods mentioned previously for figure 3.12 can be applied.²⁰

While numerous adaptive multigrid approaches exist, each is tailored to a specific physical context, meaning that there is no universal method that can guarantee optimal results compared to an approach custom made for to the problem at hand. Furthermore, the existing simulation environment must be taken into account when searching for an ideal approach. For this work's application, ideas from these methods are adapted in order to find a strategy that fits within the current software framework of GAIA using t8code without requiring overhauls of key functionalities.²¹

3.3.1 Balanced Grid

t8code provides grid balancing as a function. A grid is defined as balanced if all pairs of 2 neighboring volumes only have a size difference of maximum 1 level. The implementation in t8code used for this thesis iteratively checks each highest level volume of a grid. Then it refines all neighbors that are at least 2 levels coarser until they are just 1 level coarser. This process is repeated for every lower level until that grid fulfills the definition of balanced.

This process introduces more child volumes after the coarsening process, resulting in a larger matrix than using an approach with a non-balanced grid. However, there are specific advantages to balancing. It limits the number of neighbors per volume to a maximum of 4 per face in 3D, resulting in a maximum of 24 bands in the matrix. Each volume now either has exactly one neighbor per face, if the neighboring volume is the same or larger size or 2^{D-1} neighbors on faces that border smaller volumes. Without balancing, this maximum number can grow exponentially, reducing the efficiency of the sparse matrix. The t8code version

²⁰Text passage reviewed by Qwen 3.5-122B.

²¹Text passage reviewed by Qwen 3.5-122B.

4.0.0, used in this work, does not support face neighbor accesses on a non-balanced grid, making balancing necessary for this adaptive multigrid solver with the current version. ²²

3.3.2 Multigrid Function Changes

Since the adaptive multigrid approach introduces volumes of different sizes in the same grid, some functionalities need to be adapted.

During the creation of the matrix system, applying FVM results in formula 3.15, which cannot be used for adaptive multigrid in its current form. It considers the volume of each volume, the area of each face and the distance between volume centers, but assumes that all volumes are equidimensional and equidistant. Changes need to be made to the approximation of $\alpha(S)$ in formula 3.7 and the distance between both volumes in formula 3.8. Both approximations were based on the assumption that the mesh is orthogonal, requiring the face center to be exactly in the middle of both neighboring volume centers. This is no longer true on adapted grids. Figure 3.14 shows an example in 2D of this assumption not holding true anymore using this approach. One can see volume X and its neighbors $N1$ to $N6$. Volumes $N6$ and $N5$ have the same size as X , so the approximation method remains the same as in the non-adaptive multigrid approach. However, volumes $N1$ to $N4$ are smaller than X . The left and bottom face of X can no longer be regarded as a single face. They must be split into smaller faces corresponding to the faces of each volume $N1$ and $N2$ or $N3$ and $N4$ respectively. ²³

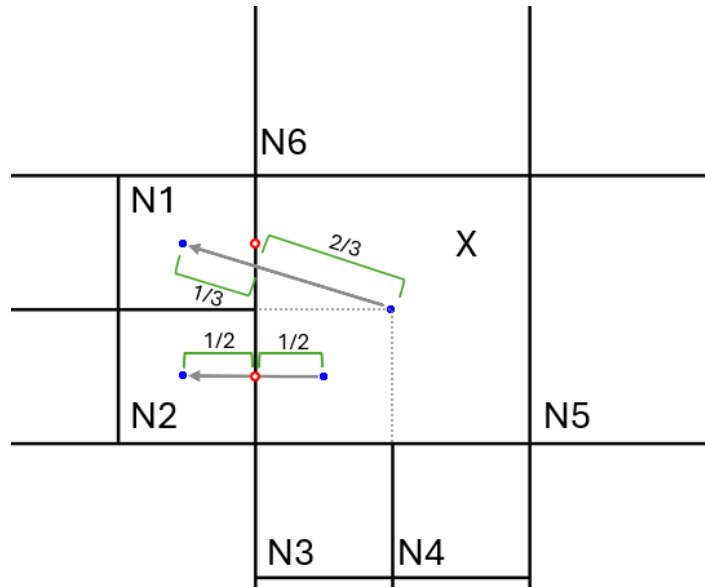


Figure 3.14: Approximation methods for the face center with neighbors of different sizes.

Two approaches are visualized here. The first approach is shown with neighbor $N1$. Instead

²²Text passage reviewed by Qwen 3.5-122B.

²³Text passage reviewed by Qwen 3.5-122B.

of interpolating the face center, visualized as a red dot, the intersection point between the face and the vector (gray arrow) from the volume center of X to the center of $N1$ (blue dots) is used. These points are not equidistant. This means the weights in Formula 3.7 must be adjusted from 50%, 50% to 66.6%, 33.3%, but the distance $\|N1 - X\|$ can be calculated without changing the formula. However, this approach causes the multigrid solvers to fail to converge and could not be tested. The reason for this is likely the FVM. The distance and weight of both volumes are part of the approximation connected with the outward-pointing normal vector of the face, see formula 3.8. But in this approach, the vector is neither orthogonal to the face nor aligned with the normal vector.

The second approach does not encounter these issues. To simulate a vector for the approximation with neighboring volume $N2$, an additional approximation is required. Theoretical child volumes of volume X are created, inheriting the same diffusion coefficient and current temperature as X . The theoretical child, whose center position is the mirror image of the center position of $N2$ across face S , is used for both formulas 3.7 and 3.8 instead. Since the diffusion coefficient α is the same, formula 3.7 does not change. Only the position of the theoretical child of $N1$ is taken into account, instead of the one from its parent. These changes can be applied analogously in 3D and also in the reverse case, if the neighboring volumes are larger in size. The ghost volumes are always the same size as the boundary volumes, therefore, this is already taken into account as seen in figure 3.1.²⁴

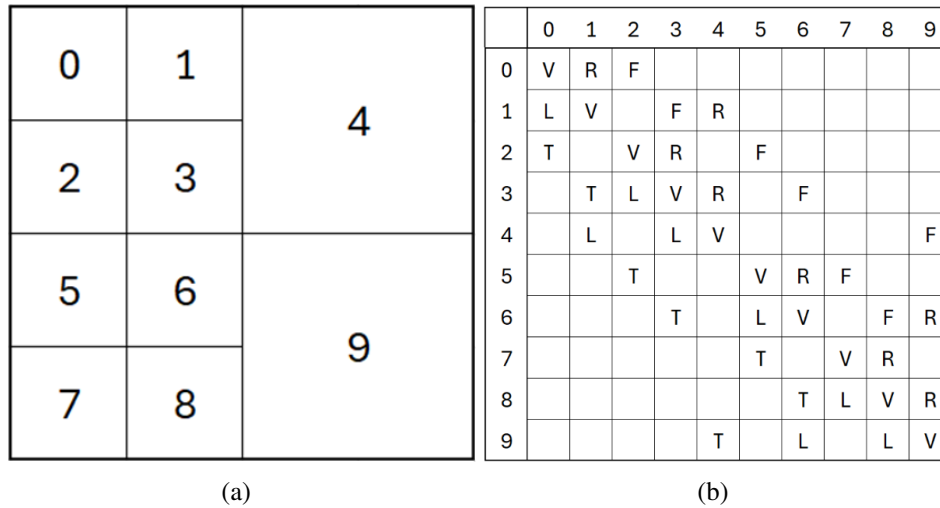


Figure 3.15: An adapted 2D grid with z-curve indexed volumes (a) and its representative matrix A (b).

After accounting for these issues, the procedure from figure 3.2 and formula 3.32 can also be used during the grid construction, see figure 3.15. (a) shows an adapted example grid with different sized volumes. Since it is now possible to have multiple neighbors on the same face, each of these neighbors must appear in the matrix. This can be seen in rows 4 and 9 of

²⁴Text passage reviewed by Qwen 3.5-122B.

matrix (b), with multiple L entries. Note that the L entries do not have the same value. This is a simplification and each coefficient need to be calculated individually.

During coarse and fine grid correction, it is now possible that some volumes do not get refined or coarsened. In the case of t8code's data transfer, only the temperature or residual values are copied from the old grid to the new one. The transfer matrices R change their form. A Volume that does not change only needs one row or column to be transferred.

0	1	4	5	0	1	4
2	3	6	7	2	3	
8	9	12	13	5	6	9
10	11	14	15	7	8	

(a)
(b)

Figure 3.16: Two 2D example grids: (a) Grid $G^{(0)}$. (b) Adaptively coarsened grid $G^{(1)}$.

$$\underbrace{\begin{pmatrix} res^{(1)}_0 \\ res^{(1)}_1 \\ res^{(1)}_2 \\ res^{(1)}_3 \\ res^{(1)}_4 \\ res^{(1)}_5 \\ res^{(1)}_6 \\ res^{(1)}_7 \\ res^{(1)}_8 \\ res^{(1)}_9 \end{pmatrix}}_{res^{(1)}_{(0)}} = \underbrace{\begin{pmatrix} 1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & 1 & 1 & 1 & 1 & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & 1 & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . & 1 & 1 & 1 & 1 \end{pmatrix}}_{R^{(0)}} \underbrace{\begin{pmatrix} res_0 \\ res_1 \\ res_2 \\ res_3 \\ \vdots \\ res_{12} \\ res_{13} \\ res_{14} \\ res_{15} \end{pmatrix}}_{res(B^{(0)}, A^{(0)}, u^{(0)}_{smooth})} \quad (3.65)$$

In figure 3.16, one can see two grids. (a) is the finer grid $G^{(0)}$ and (b) is the adapted coarser grid $G^{(1)}$. To find the coarse grid correction transfer matrix $R^{(0)}$, the transfer matrix as in chapter 3.2.3 must be adapted. Some volumes are coarsened in the same way, while others remain unchanged. The transfer matrix $R^{(0)}$ must be chosen such that the residuals of volumes 0 to 3 remain at the same index, while the residuals of volumes 8 to 11 are mapped to indices 5 to 8 in $G^{(1)}$. Additionally, volumes 4 to 7 and 12 to 15 from $G^{(0)}$ are coarsened into volumes 4 and 9 in $G^{(1)}$, respectively. The transfer matrix $R^{(0)}$ for the example in figure

3.16 is shown in formula 3.65. It changes from a uniform aggregation matrix in formula 3.46 to one with mixed steps, where the shorter steps represent unchanged volumes. $R^{(0)}$ can be transposed to $\left(R^{(0)}\right)^T$ to obtain the transfer matrix for the fine grid correction from $G^{(1)}$ to $G^{(0)}$.²⁵

3.3.3 Grid Adaptation

As seen in chapter 3.3, the adaptive multigrid method has many different approaches tailored to their respective applications. In this work, the adaptive multigrid is applied in a customized way to fulfill the needs of GAIA, as it is desired to be used in regular simulations without changing the simulation setup. The grid on which GAIA's simulation runs is $G^{(0)}$. This grid is fixed and should not be refined locally, since the simulation continues on this static grid. Instead of refining, only coarsening is used. A new grid is built according to the regular multigrid approach, by merging child volumes into a parent volume, but an adjustment is made. Regions with sharp temperature diffusion gradients remain unchanged by the coarsening process.²⁶

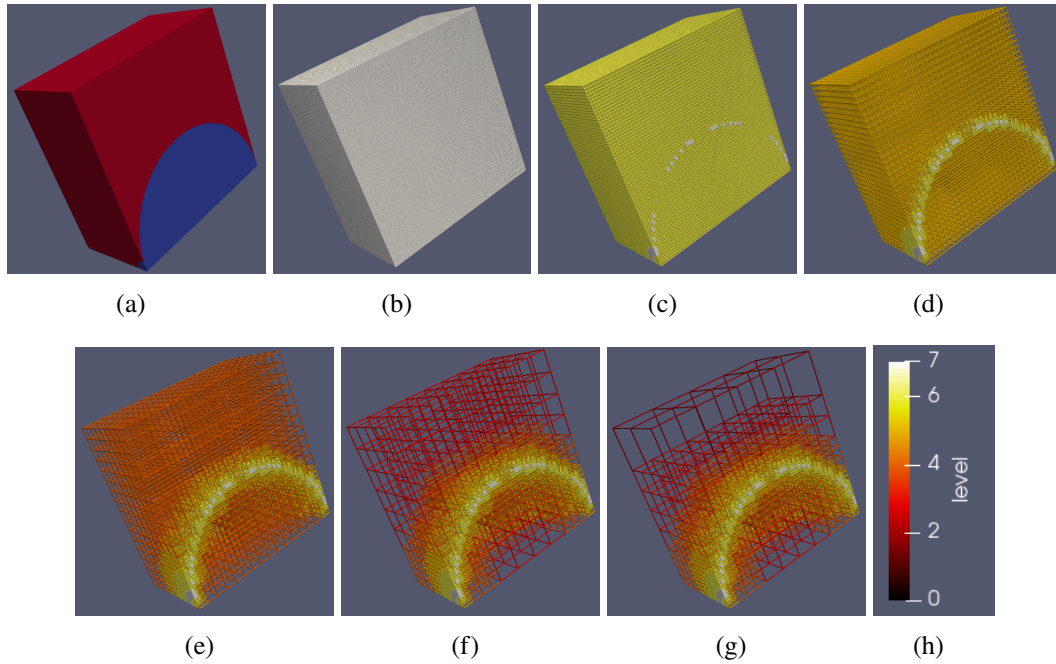


Figure 3.17: Cross section view of a 3D Level 7 grid that gets coarsened and balanced: (a) Distribution of the blue and red regions. (b) Finest grid $G^{(0)}$. (c) Grids $G^{(1)}$ to (g) $G^{(5)}$ are coarsened and balanced iteratively. (h) Color-coded scale of levels.

An adaptation criterion is introduced that evaluates the properties of child volumes and

²⁵Text passage reviewed by Qwen 3.5-122B.

²⁶Text passage reviewed by Qwen 3.5-122B.

decides whether volumes are merged into a parent volume at the beginning of the grid construction. t8code is specialized in using these adaptation callbacks, so no efficiency loss is incurred here. This process is performed only once, at the beginning of the matrix creation. Therefore, the grids are predefined.²⁷

t8code allows the adaptation criterion to be flexible. For example one can account for the variance of the coefficients or the number of volumes with different coefficients. If the criterion is too strict, not enough volumes are coarsened, leading to a high number of volumes on the next grid, leading a larger matrix, which negatively impacts the overall performance of the multigrid solver. If the criterion is too soft, too many volumes are coarsened, even areas with high gradients, making the process too similar to the regular multigrid. Applying this adaptation to the grid, while also balancing it, leads to grid structures visualized in figure 3.17. The coarsening criterion used here is based on the standard deviation σ . If the deviation of the diffusion coefficients of 8 sibling volumes is ≤ 0.5 , they are coarsened into their parent volume. Figure 3.17 shows the process of grid adaptation from the finest grid $G^{(0)}$ (b) to the coarsest one $G^{(5=C)}$ (g). One can see that volumes around the edge between the red and blue region stay on finer levels, while other volumes get coarsened. Volumes on coarser grids get coarser the further they are from finer volumes. This is also a visualization of t8code's balancing.

Grid hierarchy	Multigrid	Adaptive Multigrid
$G^{(0)}$	2,097,152	2,097,152
$G^{(1)}$	262,144	275,864
$G^{(2)}$	32,768	64,548
$G^{(3)}$	4,096	42,008
$G^{(4)}$	512	40,104
$G^{(5)}$	64	39,992
$G^{(6)}$	8	
$G^{(7)}$	1	
Total	2,396,745	2,559,668

Table 3.2: Number of volumes per grid. The adaptive multigrid data is based on the example in figure 3.17.

This approach has two major downsides when compared with the regular multigrid approach. The number of grids in this approach may be lower than in the non-adaptive variant. However, the total number of volumes over all grids is higher and requires more memory, see table 3.2. This increases the total amount of calculations the smoothing and grid transfer operators need, each time they are used. Grids $G^{(3)}$ to $G^{(5)}$ are also similar in volume number and structure. The other downside is the volume count of the coarsest grid $G^{(5)}$. Solving the residual on it requires a lot more iterations, hurting the efficiency of the multigrid method.

²⁷Text passage reviewed by Qwen 3.5-122B.

Since the volume count of grid $G^{(5)}$ is still magnitudes smaller than the one of the finest grid $G^{(0)}$, testing is required to determine if this approach, despite these issues, provides an efficient solver.

Combining this approach with a steady and reverse version of MLAT gets rid of one of these downsides. If grid $G^{(5)}$ from figure 3.17 gets coarsened further without any coarsening criterion, some new grids are introduced. The coarsest grid $G^{(C)}$ now has the smallest possible size of 1 volume, making the solving process on that grid as efficient as in the standard multigrid approach. In figure 3.18, one can see this applied with an adjusted adaptation criterion. It dictates, that all sibling volumes whose standard deviations of the diffusion coefficients is ≤ 0.5 get coarsened, if and only if at least 90% of all volumes would get coarsened. If this fails the first time, the highest level volumes get coarsened without that criterion iteratively, until level 0 is reached.

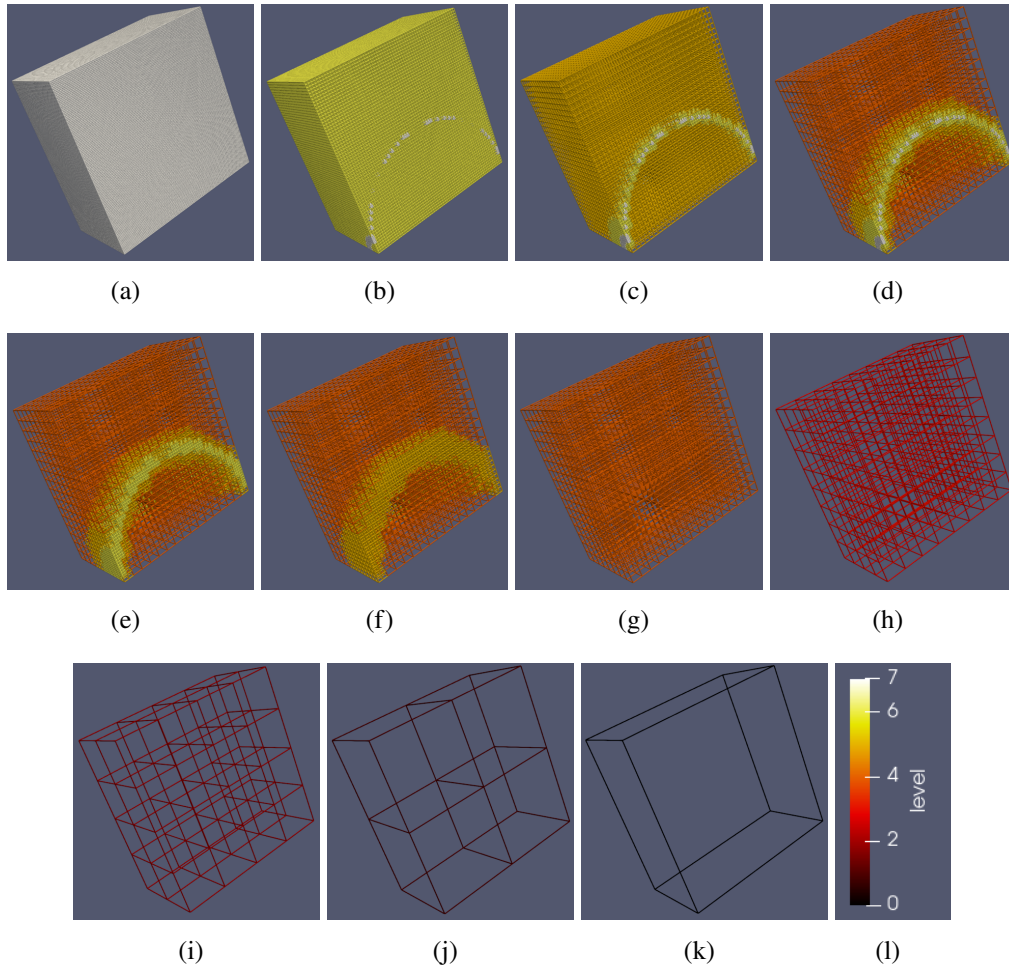


Figure 3.18: Cross section view of a 3D Level 7 grid that gets coarsened and balanced initially with a criterion and continued without one: (a) Finest grid $G^{(0)}$. (b) Grids $G^{(1)}$ to (d) $G^{(3)}$ are coarsened and balanced with coarsening criterion iteratively. (d) Grids $G^{(3)}$ to (k) $G^{(10)}$ are coarsened and balanced without coarsening criterion iteratively. (l) Color-coded scale of levels.

In figure 3.18 and table 3.3 it is visible that with this method the number of grids increases. However, this optimization decreases the total number of volumes compared to previous approach, despite the larger amount of grids.

Grid hierarchy	Multigrid	Adaptive Multigrid
$G^{(0)}$	2,097,152	2,097,152
$G^{(1)}$	262,144	275,864
$G^{(2)}$	32,768	64,548
$G^{(3)}$	4,096	42,008
$G^{(4)}$	512	28,288
$G^{(5)}$	64	10,228
$G^{(6)}$	8	4,096
$G^{(7)}$	1	512
$G^{(8)}$		64
$G^{(9)}$		8
$G^{(10)}$		1
Total	2,396,745	2,522,769

Table 3.3: Number of volumes per grid. The adaptive multigrid data is based on the example in figure 3.18.

3.4 Simulation Setup

This section explains how the performance tests are setup and run. The test environments and settings for the multigrid solvers are specified with the goal of comparing computational costs afterwards. ²⁸

3.4.1 Test Conditions

The test environment used for performance tests of different multigrid settings was partially introduced in chapter 2.1. The following specifications for the environment will be tested:

- The finest grid is a level 7 t8code 3D hypercube mesh. This is a cube containing $(2^3)^7 = 2,097,152$ evenly distributed cubical volumes of equal size, forming a $128 \times 128 \times 128$ grid.
- The steady-state non-homogeneous heat equation $\nabla \cdot (\alpha(x)\nabla u(x)) = -f(x)$ is discretized using the Finite Volume Method on each grid. This results in a matrix of size

²⁸Text passage reviewed by Qwen 3.5-122B.

$2^{21} \times 2^{21}$ for the finest grid $G^{(0)}$.

- As shown in figure 2.2, the domain is split into two regions, a blue and a red one. The blue region is a halfsphere centered on the x_3 -axis with a diameter equal to the side length of the cube, which is 1. A volume V in $G^{(0)}$ belongs to the blue region if its center coordinates $x = (x_1, x_2, x_3)$ satisfy the following condition $\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3)^2} \leq 0.5$. All other volumes belong to the red region. For the tests, both regions are assigned a static diffusion coefficient α with $\alpha_{red} = 1$ for all tests and $\alpha_{blue} = 1$, $\alpha_{blue} = 10^{-2}$ and $\alpha_{blue} = 10^{-5}$ respectively.
- The source term is constant across all tests: $f(x_V) = 1K$ for all $V \in G^{(0)}$.
- The boundary conditions are fixed. For all tests, the Dirichlet condition is applied with $\Phi_D(S) = 0K$ on all boundaries.

Note that this thesis' work accounts for Neumann conditions as well, although they are not used here.

Since the multigrid solver is flexible regarding its components, the following configurations will be tested:

- The GS-LEX smoother will be used along the t8code space filling z-curve first in forward then in reverse order to combat error shifting.
- The defect function is defined as the residual.
- The coarse grid correction of the residual and the fine grid correction of the temperature are performed by t8code, which is equivalent to matrix multiplication by R and R^T , respectively.
- The coarse grid correction of the matrix $A^{(\phi)}$ is performed using the geometric approach, which reapplies the FVM on each grid.
- For FMG-interpolation, the barycentric triangulation based on the parent volumes and their neighbors is used.
- The performance of V-cycles, F-cycles and their FMG-cycles will be tested and compared against pure GS-LEX solving and against GAIA's BiCGSTAB solver.
- In the non adaptive multigrid approach t8code coarsens the finest grid $G^{(0)}$ down to a level 0 3D hypercube, which translates to a single cubical volume covering the entire domain. For each coarsening step from $G^{(\phi)}$ to $G^{(\phi+1)}$, 8 child volumes are merged into a single parent volume, reducing the grid size by a factor of 8 in each iteration.

- The adaptation criterion which dictates the grid coarsening process is based on the standard deviation σ of the diffusion coefficient of the child-volumes. If $\sigma \leq 0.5$ no coarsening is performed.
- For the adaptive multigrid a local smoothing approach is used, where all volumes are smoothed simultaneously.

3.4.2 Benchmark Evaluation and the SMRA metric

In the GAIA framework, a linear system $A \cdot u = B$ is considered solved if the relative euclidean norm of the residual falls below 10^{-6} or if the absolute euclidean norm of the residual drops below 10^{-10} . This thesis also uses the same criterion as GAIA to consider a matrix system solved. The absolute norm is defined as follows:

$$\|\vec{v}\| = \sqrt{\sum_{i=0}^N \left(\text{res}(B^{(0)}, A^{(0)}, u^{(0)})_i \right)^2} \quad (3.66)$$

The relative norm, $\|\vec{v}\|_{rel}$, is defined as:

$$\|\vec{v}\|_{rel} = \frac{\sqrt{\sum_{i=0}^N \left(\text{res}(B^{(0)}, A^{(0)}, u_{ini}^{(0)})_i \right)^2}}{\sqrt{\sum_{i=0}^N \left(\text{res}(B^{(0)}, A^{(0)}, u_{current}^{(0)})_i \right)^2}} \quad (3.67)$$

$u_{ini}^{(0)}$ is the initial temperature before starting to solve and $u_{current}^{(0)}$ is the current temperature of the the finest grid, while solving.

To quantify the computational cost of multigrid cycles and interpolations and to compare them with other solvers, this work employs the Sparse Matrix Row Accesses (SMRA) metric. One SMRA corresponds to a single operation accessing a complete row of a sparse matrix, effectively applying the local stencil. Since memory access is the primary constraint on performance, SMRA offers a robust measure of computational effort. While runtime is often influenced by implementation factors, SMRA serves as an architecture and hardware independent measure. The SMRA count is expected to scale linearly with runtime in a fully optimized implementation. The estimated SMRA costs for specific operations are as follows:

- **BiCGSTAB:** GAIA's BiCGSTAB solver inherently counts iterations. Each iteration involves two matrix multiplications. Since a matrix multiplication requires accessing

²⁹Text passage reviewed by Qwen 3.5-122B.

every row of the sparse matrix once, one iteration corresponds to $2 \cdot N^{(\phi)}$ SMRAs, where $N^{(\phi)}$ is the number volumes of grid $G^{(\phi)}$.

- **Smoothing:** Each call to the smoothing function 3.35 for a volume requires one SMRA. This leads to a total of $2 \cdot N^{(\phi)}$ SMRAs per complete smoothing operation $G^{(\phi)}$.
- **Transfer Operations:** When t8code transfers data between grids, for each volume its children need to be accessed when fine grid correcting or its parent and siblings when coarse grid correcting. The grid transfer corresponds to a matrix-vector multiplication of the vector $u^{(\phi)}$ or $res^{(\phi)}$ with the transfer operators R or R^T . Therefore, this operation is at most as expensive as one SMRA per volume.
- **FMG Interpolation:** Each FMG-iteration requires the same access as the fine grid transfer operator with one additional access to the parent volume's neighbors. Neighbor information is also stored within the bands of the matrix $A^{(\phi)}$. This requires similar access patterns and therefore, an access of all neighbors is counted as one SMRA per volume as well. This translates to $2 \cdot N^{(\phi)}$ SMRA per FMG-interpolation of $G^{(\phi)}$. Since FMG interpolations are only done once per finer grid, the total number of SMRAs are twice the total number of volumes in all grids, excluding the coarsest grid: $2 \cdot \sum_{\phi=0}^{C-1} N^{(\phi)}$

3.4.3 Validation

To independently verify the solution and quantify errors a well established commercial software COMSOL ([COMSOLAB, 2026](#)) within its own framework was used in addition to GAIA's BiCGSTAB. The COMSOL suite uses a finite-element approach and a direct solver. The mesh size was chosen to be extremely fine to produce highly accurate results. The performance of the COMSOL solver is not tracked in this thesis, as its purpose is solely to verify the correctness of the solution $u_{solved}^{(0)}$.³⁰

³⁰Text passage reviewed by Qwen 3.5-122B.

Chapter 4

Results

Solving the three test environments with the settings described in chapter 3.4.1 leads to the solutions $u_{solved}^{(0)}$ visualized in figure 4.1 with different temperature distributions. Note that the scales in (b), (c) and (d) are different. In (b) both diffusion coefficients are set to 1, while in (c) and (d) they differ by magnitudes of 2 and 5, respectively. The differences in $u_{solved}^{(0)}$ of the same scenario obtained with different solvers are negligible, as they all reach the required residual accuracy. ¹

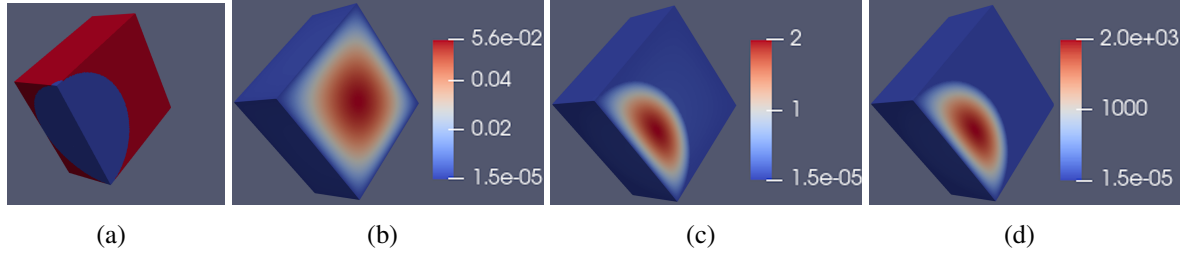


Figure 4.1: The 3D cube test environments solved by GS-LEX, showing the solved temperature $u_{solved}^{(0)}$ in Kelvin, as a cross-section view of the cube. (a) Distribution of blue and red regions. (b) Temperature with $\alpha_{red} = \alpha_{blue} = 1$. (c) Temperature with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$. (d) Temperature with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$.

Figures 4.2, 4.3 and 4.4 plot the current relative euclidean norm of the residual of each solver and cycle type against the number of SMRAs accumulated during the solving process. The "V-cycle + BiCGSTAB" graph represents a single multigrid V-cycle used as a preconditioner combined with GAIA's BiCGSTAB solver. This configuration was tested to determine if a simple multigrid solver could provide an effective initial guess for BiCGSTAB. Note that the euclidean norm of the residual should not be compared directly while the multigrid operates on coarser grids $G^{(\phi \neq 0)}$. These residuals are corrections of residuals and are not the residual of the initial Grid $G^{(0)}$. The multigrid graphs are constructed by plotting points only when

¹Text passage reviewed by Qwen 3.5-122B.

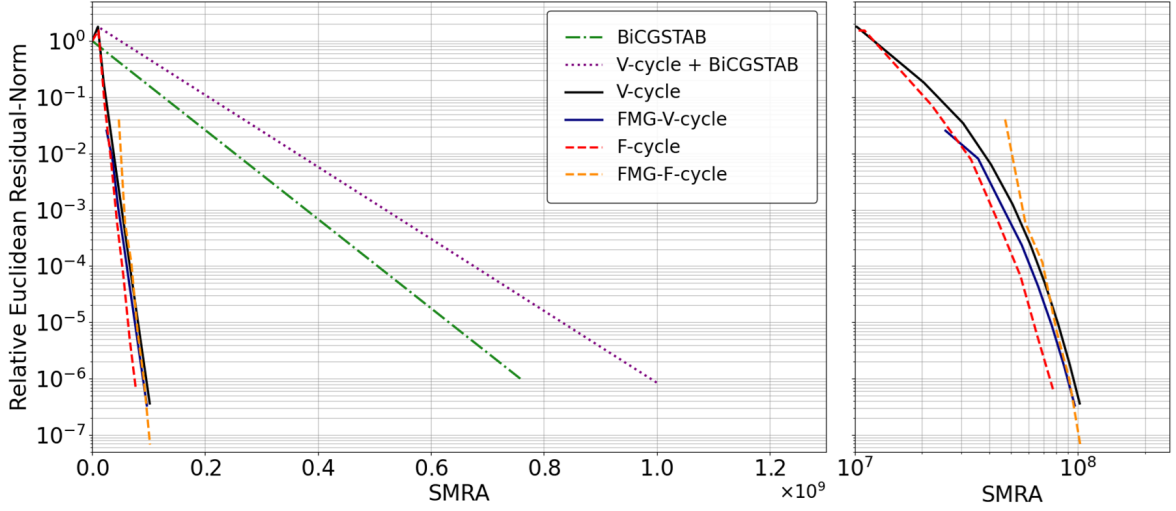


Figure 4.2: Performance of all tested BiCGSTAB and multigrid solvers with $\alpha_{red} = \alpha_{blue} = 1$. The Plot on the right is a closeup view of just the multigrid solvers with a logarithmic x-axis for better comparison.

the solver returns to the finest grid. This explains why the FMG graphs do not start at 0 SMRAs. Table 4.1 provides the exact SMRA counts for each solver, including pure GS-LEX, of all test scenarios for direct comparison.

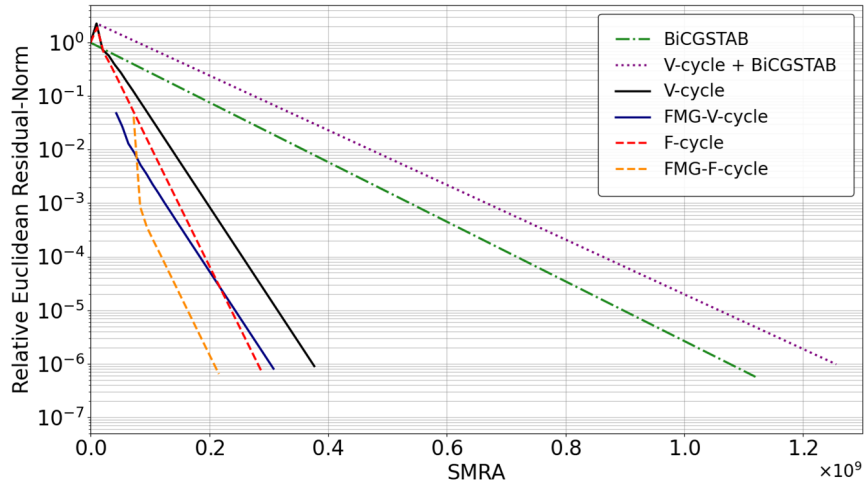


Figure 4.3: Performance of all tested BiCGSTAB and multigrid solvers with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$.

The first key observation is that all multigrid solver types outperform GAIA's BiCGSTAB across all test environments. In the scenario, where the diffusion coefficients are equal $\alpha_{red} = \alpha_{blue}$, the multigrid solvers are approximately 7 to 10 times more efficient than GAIA's BiCGSTAB. It also becomes apparent that all multigrid solvers incur a performance penalty, when increasing the difference between α_{red} and α_{blue} . The pure V-cycle is the least resilient to these changes, while the FMG F-cycle shows the highest resilience. When comparing BiCGSTAB with its preconditioned version, labeled as "V-cycle + BiCGSTAB",

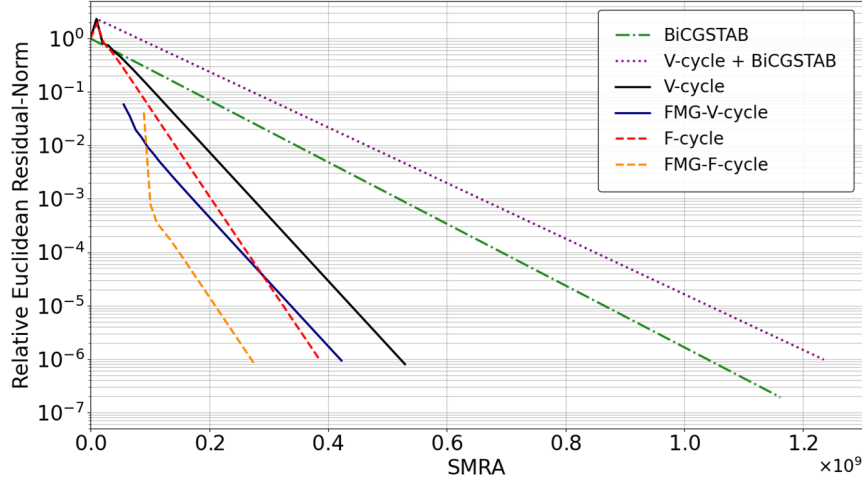


Figure 4.4: Performance of all tested BiCGSTAB and multigrid solvers with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$.

a single multigrid V-cycle does not appear to provide any benefits when used to find a good initial guess. The opposite effect is observed. More SMRAs are required to find a solution.²

Solver	$\alpha_{red} = 1$ $\alpha_{blue} = 1$	$\alpha_{red} = 1$ $\alpha_{blue} = 10^{-2}$	$\alpha_{red} = 1$ $\alpha_{blue} = 10^{-5}$
pure GS-LEX	4.699927e+10	4.346348e+10	4.344670e+10
BiCGSTAB	7.591690e+08	1.119879e+09	1.161822e+09
V-cycle + BiCGSTAB	1.004236e+09	1.255894e+09	1.234923e+09
V-cycle	1.018617e+08	3.768881e+08	5.296806e+08
F-cycle	7.729497e+07	2.870956e+08	3.864748e+08
FMG V-cycle	9.679177e+07	3.082368e+08	4.225526e+08
FMG F-cycle	1.023744e+08	2.162258e+08	2.776235e+08

Table 4.1: Total SMRAs required by each solver per test scenario.

Testing the adaptive multigrid variants on the case of $\alpha_{red} = \alpha_{blue}$ leads to no changes in the grids. No adaptation criterion can be met, as the definition relies on a difference between these coefficients. Figures 4.5 and 4.6 plot the current relative euclidean norm of the residual of each solver and cycle type against the number of SMRAs accumulated during the solving process, using the grid structure from figure 3.17. Since the coarsest grid still contains too many volumes to be solved efficiently with only a few smoothing steps, GS-LEX was applied until the relative euclidean norm of the residual dropped below 10^{-6} . However, this requires an enormous amount of SMRAs, which is visible in the graphs for the F-Cycle and FMG-F-Cycle solvers, since these solvers operate more frequently on the coarser grids. Due to the solvers needing to be applied more often on the coarsest grid, efficiency is lost in this approach. In contrast, the V-Cycle and FMG-V-Cycle are more stable, although they still

²Text passage reviewed by Qwen 3.5-122B.

perform worse than their non-adaptive variant.³

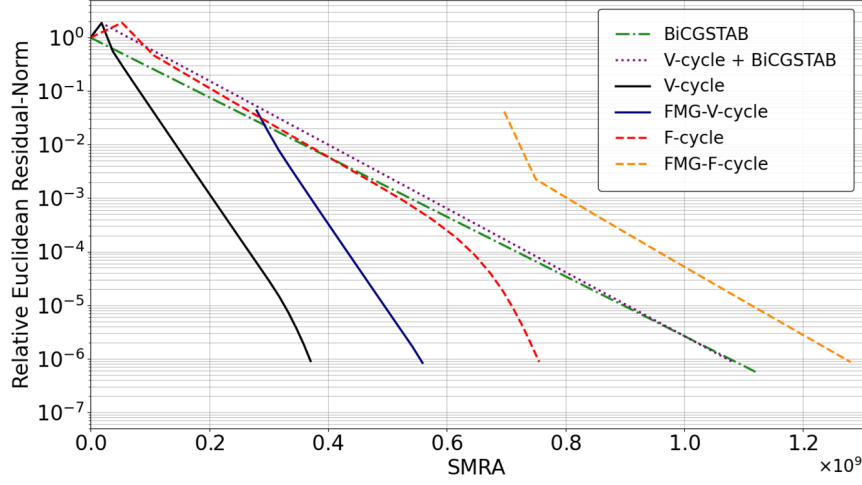


Figure 4.5: Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with GS-LEX.

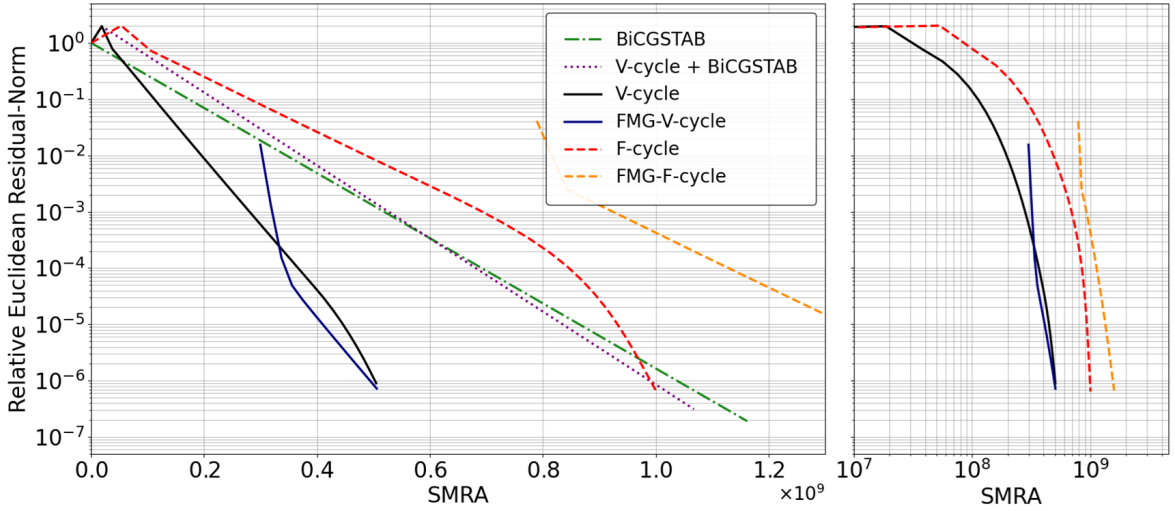


Figure 4.6: Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with GS-LEX. The Plot on the right is a closeup view of just the multigrid solvers with a logarithmic x-axis for better comparison.

The performance of all multigrid solvers improves slightly when GAIA's BiCGSTAB is used instead of GS-LEX to solve the matrix system of $G^{(C)}$. These improvements can be seen in figures 4.7 and 4.8. However, the same issue as with the GS-LEX solver persists for the F-Cycle and FMG-F-Cycle solvers. GAIA's BiCGSTAB still requires too many SMRAs to solve the matrix system of $G^{(C)}$. Despite this, the V-Cycle and FMG-V-Cycle are competitive with their non-adaptive variant. The SMRA counts for these two adaptive methods can be directly compared in table 4.2 with the non-adaptive multigrid results in table 4.1. The SMRA

³Text passage reviewed by Qwen 3.5-122B.

counts of the pure GS-LEX and pure BiCGSTAB solvers do not change in these scenarios, since they only operate on the finest grid $G^{(C)}$, which is equal in all variants.

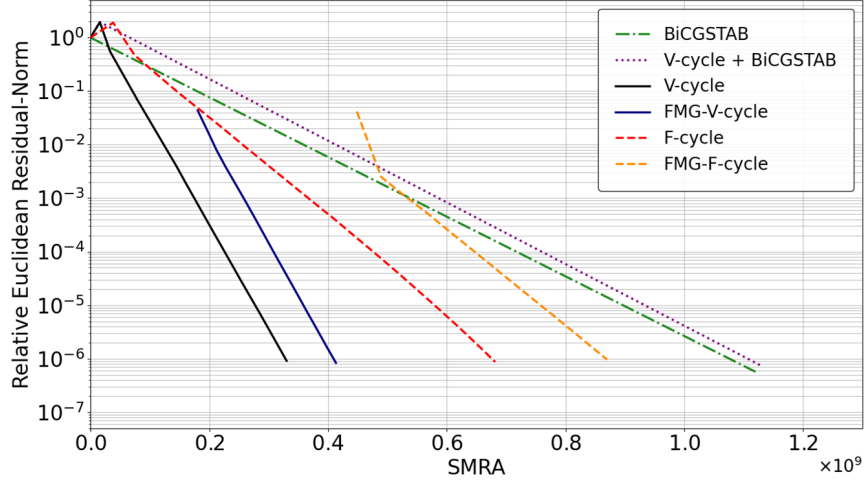


Figure 4.7: Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with BiCGSTAB.

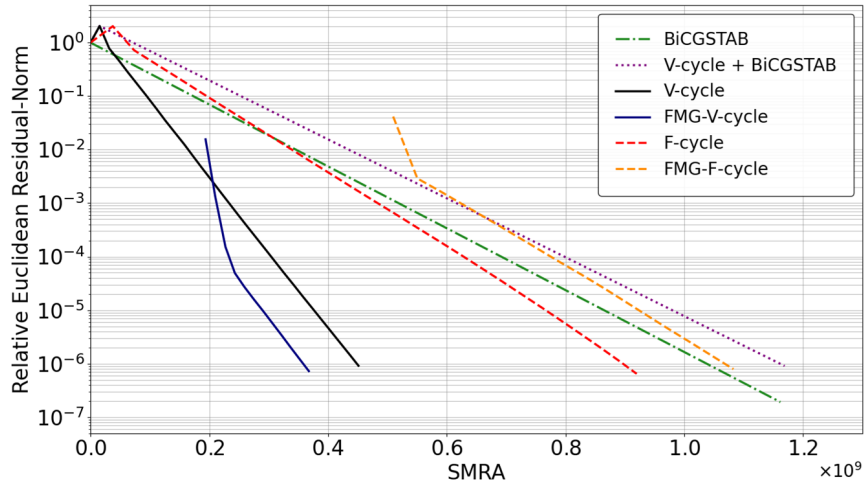


Figure 4.8: Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$, using the grid structure from figure 3.17. The matrix system of $G^{(C)}$ is solved with BiCGSTAB.

These results are outperformed by the adaptive multigrid variant, using the structure of figure 3.18. The use of the modified MLAT approach improved the performance, see figures 4.9 and 4.10. All adaptive multigrid solvers outperform their non-adaptive counterparts according to these results. The SMRA counts can be found in table 4.3 for direct comparison.

Solver	GS-LEX		BiCGSTAB	
	$\alpha_{blue} = 10^{-2}$	$\alpha_{blue} = 10^{-5}$	$\alpha_{blue} = 10^{-2}$	$\alpha_{blue} = 10^{-5}$
V-cycle + BiCGSTAB	1.079904e+09	1.067321e+09	1.127276e+09	1.168819e+09
V-cycle	3.706864e+08	5.048084e+08	3.302945e+08	4.512991e+08
F-cycle	7.556756e+08	1.000918e+09	6.816104e+08	9.198147e+08
FMG V-cycle	5.591370e+08	5.056690e+08	4.133166e+08	3.676870e+08
FMG F-cycle	1.280302e+09	1.584960e+09	8.700252e+08	1.082721e+09

Table 4.2: Total SMRAs required by each solver per test scenario. The two left columns show the results when the matrix system of $G^{(C)}$ is solved by GS-LEX and the two right columns show that, when BiCGSTAB is used instead.

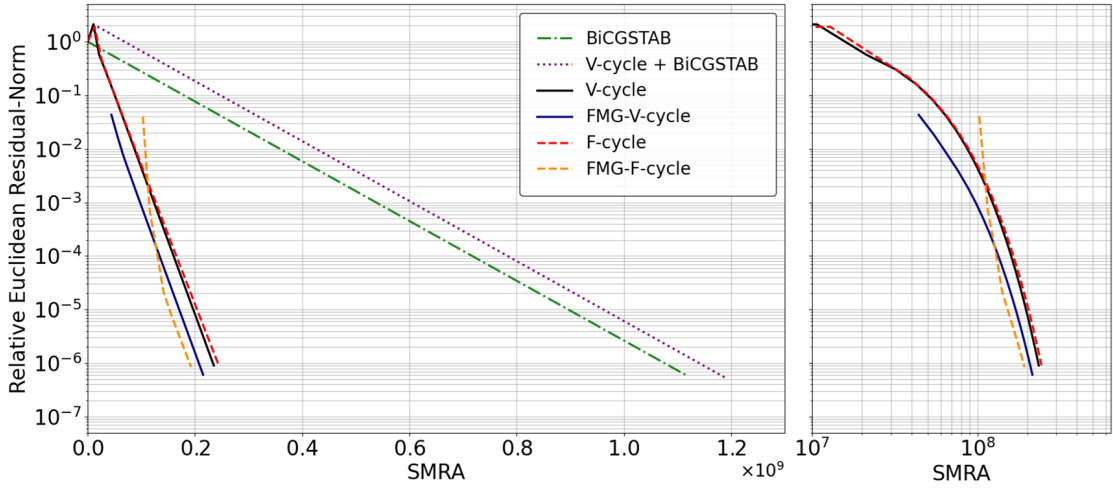


Figure 4.9: Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$, using the grid structure from figure 3.18. The Plot on the right is a closeup view of just the multigrid solvers with a logarithmic x-axis for better comparison.

Considering the scenario where the diffusion coefficient is equal $\alpha_{blue} = \alpha_{red}$, all non adaptive multigrid solvers perform equally well. They range from 7.7×10^7 to 1.0×10^8 SMRAs compared to GAIA's BiCGSTAB with 7.6×10^8 SMRAs, making any multigrid approach desirable for these kinds of problems, see table 4.1 and figure 4.2. However, since this idealized (Poisson) condition has already been thoroughly studied (Trottenberg et al., 2001), this work instead focuses on more complex configurations. The direct comparison between the results of the test scenarios with lesser magnitude difference between diffusion coefficients can be seen in table 4.4. ⁴

Even under conditions where temperature gradients are large, the adaptive FMG-cycles perform the best and with the least variance compared to the ideal scenario. The cycles without FMG are not as efficient, but also provide a performance increase for GAIA's simulations compared to its current solver. Adaptive multigrid solvers can work with sharp gradients,

⁴Text passage reviewed by Qwen 3.5-122B.

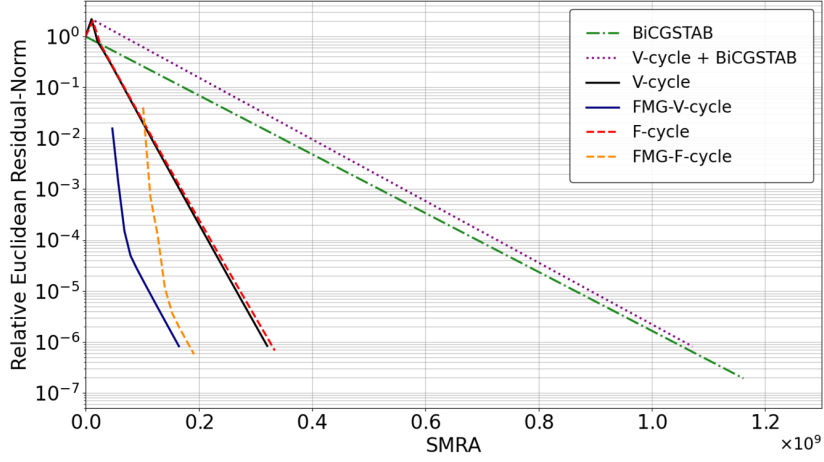


Figure 4.10: Performance of all tested adaptive multigrid solvers compared with BiCGSTAB, with $\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$, using the grid structure from figure 3.18.

Solver	Non-Adaptive		Adaptive	
	$\alpha_{blue} = 10^{-2}$	$\alpha_{blue} = 10^{-5}$	$\alpha_{blue} = 10^{-2}$	$\alpha_{blue} = 10^{-5}$
pure GS-LEX	4.346348e+10	4.344670e+10		
BiCGSTAB	1.119879e+09	1.161822e+09		
V-cycle + BiCGSTAB	1.255894e+09	1.234923e+09	1.189290e+09	1.071849e+09
V-cycle	3.768881e+08	5.296806e+08	2.351858e+08	3.207079e+08
F-cycle	2.870956e+08	3.864748e+08	2.442782e+08	3.342754e+08
FMG V-cycle	3.082368e+08	4.225526e+08	2.152056e+08	1.648677e+08
FMG F-cycle	2.162258e+08	2.776235e+08	1.927555e+08	1.915849e+08

Table 4.3: Total SMRAs required by each solver per test scenario. The two left columns show the results of the non adaptive multigrid solvers and the two right columns show the adaptive multigrid solvers.

where BiCGSTAB loses efficiency.

It can be concluded that using a single multigrid V-cycle as a preconditioner for GAIA's BiCGSTAB provides little benefit and in some cases, even slows it down. The main focus of this thesis does not lie on preconditioners, therefore, other multigrid preconditioner approaches that could possibly improve this result are not explored here. An additional limitation is the measurement of efficiency in SMRAs rather than actual runtime. Considering parallelization and memory optimization the results might differ as new obstacles might be identified. Furthermore, only predefined grids were used for evaluation. No dynamic or self-adaptive approaches were evaluated, meaning that the full potential of t8code and the multigrid solver have not yet been exploited. ⁵

While the approach in this thesis implies a theoretical performance increase in the Navier-Stokes equations, no direct statement can be made based on the fact that the steady-state

⁵Text passage reviewed by Qwen 3.5-122B.

Solver	$\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-2}$			
	Multigrid	Adaptive Multigrid + GS-LEX solver	Adaptive Multigrid + BiCGSTAB	Adaptive Multigrid + MLAT
pure GS-LEX	4.35e+10			
BiCGSTAB	1.12e+09			
V-cycle + BiCGSTAB	1.26e+09	1.08e+09	1.13e+09	1.19e+09
V-cycle	3.77e+08	3.71e+08	3.30e+08	2.35e+08
F-cycle	2.87e+08	7.56e+08	6.82e+08	2.44e+08
FMG V-cycle	3.08e+08	5.60e+08	4.13e+08	2.15e+08
FMG F-cycle	2.16e+08	1.28e+09	8.70e+08	1.93e+08

Solver	$\alpha_{red} = 1$ and $\alpha_{blue} = 10^{-5}$			
	Multigrid	Adaptive Multigrid + GS-LEX solver	Adaptive Multigrid + BiCGSTAB	Adaptive Multigrid + MLAT
pure GS-LEX	4.34e+10			
BiCGSTAB	1.16e+09			
V-cycle + BiCGSTAB	1.23e+09	1.07e+09	1.17e+09	1.07e+09
V-cycle	5.30e+08	5.05e+08	4.51e+08	3.21e+08
F-cycle	3.86e+08	1.00e+09	9.20e+08	3.34e+08
FMG V-cycle	4.26e+08	5.06e+08	3.68e+08	1.65e+08
FMG F-cycle	2.78e+08	1.58e+09	1.08e+09	1.92e+08

Table 4.4: Total SMRAs required per solver per approach and per test scenario. The best result for each scenario is highlighted

non-homogeneous heat equation was used in its stead. Further tests would have to be conducted to see, whether this holds true in practice. An additional benefit in performance could be realized by testing over-relaxation in the GS-LEX smoother function as suggested by (Trottenberg et al., 2001, Page 74) for Poisson problems. Finally, as the approach used in this thesis focuses on the GAIA framework, the options for other adaptive multigrid approaches are limited. ⁶

⁶Text passage reviewed by Qwen 3.5-122B.

Chapter 5

Conclusion and Outlook

This thesis investigated the potential of geometric multigrid solvers in improving the performance of GAIA's simulation. The primary motivation was to address the limitations of GAIA's current BiCGSTAB solver in environments with sharp gradients in the diffusion coefficient α . The objective was to track whether an (adaptive) multigrid approach could provide a reduction in computational cost compared to the existing solver. This was tested on the steady-state non-homogeneous heat equation, which shares the computationally expensive term with the Navier-Stokes equations that are used regularly in GAIA's simulations.

The results presented confirm that the implementation of multigrid methods led to a performance gain. In the baseline scenario with uniform diffusion coefficients, all tested multigrid cycles outperformed the BiCGSTAB solver by a factor of approximately 7 to 10, measured in SMRAs. More importantly, the multigrid approach showed robustness when increasing the diffusion contrast by factors of 10^2 and 10^5 . While the BiCGSTAB solver required 50% more iterations, the multigrid solvers maintained their efficiency, with only a slight increase in computational cost.¹

From the tested scenarios, the adaptive multigrid approach utilizing the modified MLAT proved to be the most efficient strategy for the tested problems with high diffusion contrasts. By preserving fine grid resolution in regions with steep gradients while coarsening elsewhere, this method reduced the total SMRA count by 40% when comparing the best adaptive solver with the best non-adaptive one in the most challenging environment. The FMG F-cycle was the most robust cycle type, consistently being among the two best performing cycles, comparing all non-adaptive multigrid results with the multigrid that applies MLAT. The attempt to use a single V-cycle as a preconditioner for BiCGSTAB did not result in the expected benefits. It often increased the computational effort. This suggests that for this type of problem, one should either investigate further for a better multigrid preconditioner or use

¹Text passage reviewed by Qwen 3.5-122B.

the multigrid solver to solve the problem directly.²

Some limitations of this work must be taken into account. The performance evaluation relied on the SMRA count rather than actual runtime. This was preferred as a metric to compare solvers, because runtime could be influenced by non-optimized memory, cache and parallelization overhead.³

The adaptive multigrid solver was only used on predefined grid structures. The dynamic, self-adaptive refinement, which could further optimize the solving process, was not implemented. The smoothing process was limited to the GS-LEX method without over-relaxation. This could be investigated to increase the performance of the smoother.

Based on the results and limitations of this work, some other suggestions for future research are identified. Hybrid cycle strategies that combine different multigrid cycles could help balance convergence speed and computational cost. The implementation of dynamic self-adaptive grid refinement, allowing the solver to automatically adjust the mesh resolution based on evolving solution features rather than relying on predefined structures, can be investigated. Extending the solver to handle the full Navier-Stokes equations is an important next step to validate the results from this thesis, together with optimizing this work's implementation to compare runtime instead of SMRAs.⁴

In conclusion, this thesis successfully demonstrated that a geometric multigrid solver is a viable and efficient alternative to the current BiCGSTAB solver of GAIA. The adaptive variant, especially, offers promising performance improvements for simulating complex planetary interiors with high material contrasts.

²Text passage reviewed by Qwen 3.5-122B.

³Text passage reviewed by Qwen 3.5-122B.

⁴Text passage reviewed by Qwen 3.5-122B.

Bibliography

- Ana-Catalina Plesa, e. a. (2013). Thermo-chemical convection in planetary mantles: Advection methods and magma ocean overturn simulations. In Rückemann, C.-P., editor, *Integrated Information and Computing Systems for Natural, Spatial, and Social Sciences*, pages 302–323. IGI Global Scientific Publishing.
- Baranoski, G., Wan, J., Rokne, J., and Bell, I. (2002). Simulating the dynamics of the dancing lights.
- Böing, N. (2022). Evaluation of preconditioners for implicit solvers of local dg for the advection-diffusion equation. Master’s thesis, Universität zu Köln.
- Burstedde, C. and Holke, J. (2016). A tetrahedral space-filling curve for nonconforming adaptive meshes. *SIAM Journal on Scientific Computing*, 38(5):C592–C618.
- COMSOLAB (2026). Comsol multiphysics.
- DLR-AMR (2015). t8code parallel algorithms and data structures for tree-based adaptive mesh refinement (amr) with arbitrary element shapes.
- Eymard, R., Thierry, G., and Herbin, R. (2000). Handbook of numerical analysis. 7:731–1018.
- HelmholtzSoftware (2023). GAIA a framework to study the thermo-chemical evolution of rocky and icy bodies (planets, moons, and planetoids) on a global scale in the solar system and beyond.
- Herrmann, D. (1983). *Gauß-Seidel-SOR-Iteration*, pages 45–48. Vieweg+Teubner Verlag, Wiesbaden.
- Holke, J. (2018). *Scalable algorithms for parallel tree-based adaptive mesh refinement with general element types*. PhD thesis, University of Bonn.
- Holke, J., Burstedde, C., Knapp, D., Dreyer, L., Elsweijer, S., Ünlü, V., Markert, J., Lilikakis, I., Böing, N., Ponnusamy, P., and Basermann, A. (2023). t8code v. 1.0 - modular adaptive mesh refinement in the exascale era. In *Proceedings of the SIAM International Meshing Round Table 2023*, Amsterdam, Netherlands.

- Huber, A. (2025). *Navier-Stokes-Gleichungen*, pages 433–454. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Hüttig, C. (2026). Personal communication regarding the chosen heat equation. Face-to-face interview.
- Hüttig, C. and Stemmer, K. (2008a). Finite volume discretization for dynamic viscosities on voronoi grids. *Physics of the Earth and Planetary Interiors*, 171(1):137–146. Recent Advances in Computational Geodynamics: Theory, Numerics and Applications.
- Hüttig, C. and Stemmer, K. (2008b). The spiral grid: A new approach to discretize the sphere and its application to mantle convection. *Geochemistry Geophysics Geosystems*, 9.
- Hüttig, C., Tosi, N., and Moore, W. B. (2013). An improved formulation of the incompressible navier–stokes equations with variable viscosity. *Physics of the Earth and Planetary Interiors*, 220:11–18.
- John, V. (2013). Multigrid methods. Lecture notes, 2013/14.
- Knapp, D. (2017). Adaptive verfeinerung von prismen.
- Knapp, D. (2020). A space-filling curve for pyramidal adaptive mesh refinement. Master’s thesis, Rheinische Friedrich-Wilhelms-Universität Bonn.
- Kolditz, O. (2002). *Finite Volume Method*, pages 173–190. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Lohmann, C. (2025). A geometric multigrid solver for the incompressible navier-stokes equations using discretely divergence-free finite elements in 3d.
- Mattiussi, C. (1997). An analysis of finite volume, finite element, and finite difference methods using some concepts from algebraic topology. *Journal of Computational Physics*, 133(2):289–309.
- Mouhssine, A., Ratnani, A., and Sadok, H. (2025). Vector extrapolation methods applied to geometric multigrid solvers for isogeometric analysis. *Numerical Algorithms*, 100(4):1813–1836.
- Möbius, A. F. (1827). *Der barycentrische Calcul: ein neues Hülfsmittel zur analytischen Behandlung der Geometrie*. Verlag von Johann Ambrosius Barth, Leipzig. Available via ETH-Bibliothek Zürich, e-rara.
- Qin, C., Fu, W., Zhao, N., and Zhou, J. (2025). 3d adaptive finite-element forward modeling for direct current resistivity method using geometric multigrid solver. *Computers & Geosciences*, 196:105840.

Richardson, L. F. (1911). IX. the approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 210(459-470):307–357.

Saberi, S., Vogel, A., and Meschke, G. (2020). Parallel finite cell method with adaptive geometric multigrid. In Malawski, M. and Rzdca, K., editors, *Euro-Par 2020: Parallel Processing*, pages 578–593, Cham. Springer International Publishing.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical report, Pittsburgh, PA, USA.

Stoffers (2015). Die integralsätze von gauß und stokes. Lecture notes, 2015.

Trottenberg, U., Oosterlee, C., and Schuller, A. (2001). *Multigrid Methods*. Elsevier Science.

Table 5.1: AI tool usage

AI Tool	Use Case	Scope	Usage Note
Overleaf AI, Version of May 2026	Grammar, style and spelling checks	Entire thesis	Operated in the background while writing this thesis in Overleaf by highlighting text passages (similar to Microsoft Word or Google Docs).
Qwen 3.5-122B (This AI ran on a DLR server, valuing privacy and did not store chat-logs)	Final review for grammar, spelling, consistency, writing style and clarity	Entire thesis	Applied to the final draft version. Generated suggestions were used as guidance, manually reviewed and not adopted unchanged.
	Suggesting terminology improvements, highlighting imprecise descriptions and improving readability	Selected passages marked by footnotes	Applied to passages written by the author. Generated suggestions were used as guidance, manually reviewed and not adopted unchanged.
	Creating .bib file entries	Bibliography	Used for sources where no .bib file was available.
	Coding assistance for fixing bugs by analyzing error logs and for simple tasks (like sorting arrays)	Solver script	GAIA- and t8code-specific tasks were too complex and this tool could not help.