universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation /Title of the Doctoral Thesis

„A 3D Helmholtz solver and efficient time integration methods for viscous flows in the ANTARES framework“

verfasst von / submitted by

Dipl.-Phys. Patrick Blies

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doktor der Naturwissenschaften (Dr. rer. nat.)

Wien, 2017 / Vienna, 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 796 605 405
Dissertationsgebiet lt. Studienblatt / field of study as it appears on the student record sheet:	Mathematik
Betreut von / Supervisor:	Univ.-Prof. i.R. Dr. Herbert Muthsam
Mitbetreut von / Co-Supervisor:	Mag. Dr. Friedrich Kupka, Privatdoz.

Danksagung

Die vorliegende Arbeit wurde durch die Projekte P25229-N27, “Numerical Simulation of Semiconvection in Exoplanets”, und P29172-N27, “Turbulent Convection and Pulsation Interaction in Stars”, des Fonds zur Förderung der wissenschaftlichen Forschung (FWF) teilweise finanziell unterstützt. Meine Rechnungen am VSC-2 wurden im Rahmen von P70708 “Numerical simulations of semiconvection” durchgeführt.

Meinem Erstbetreuer Herbert J. Muthsam danke ich für die Möglichkeit, in seiner Gruppe die Dissertation zu verfassen. Besonderer Dank gilt meinem Zweitbetreuer Friedrich Kupka, der mir stets mit qualitativ hervorragendem Fachwissen zur Verfügung stand, seine Aufgaben als Betreuer stets zu meiner vollsten Zufriedenheit erledigt hat und meinen Erwartungen in jeder Hinsicht optimal entsprochen hat.

Ich danke meinen Kollegen Natalie Happenhofer und Hannes Grimm-Strele für die angenehme Arbeitsatmosphäre und dafür, dass sie mich mit ANTARES vertraut gemacht haben. Ohne ihre Hilfe hätte die Einarbeitung bedeutend länger gedauert. Hannes danke ich überdies für die iterative Erkundung der Dachterrasse.

Weiterhin danke ich den Teilnehmern des International PhD Cafés—allen voraus Sam, Allison und Hemma—und den Mitgliedern der lunch group—Izabela, Elisa, Sarah, Tobias und Fabian—für die notwendige Ablenkung neben der Dissertation.

Ganz besonderer Dank gilt meinen Eltern, ohne deren finanzielle Unterstützung diese Dissertation niemals möglich gewesen wäre.

Abschließender Dank gilt den beiden Frauen in meinem Leben: meiner Tochter Anna und meiner Partnerin Jenny. Ich danke für euer Verständnis und eure Unterstützung in der nicht enden wollenden Zeit, in der ich die Abende und Wochenenden im Büro auf der Suche nach Bugs verbracht habe. Ihr seid die Besten!

Abstract

In fluid dynamics, depending on the nature of the flow, different time scales govern the physical processes. This involves a significant hurdle for numerical simulations of these systems since the smallest formal time scale determines the size of the overall time step of the numerical simulation. This can slow down computations considerably. To overcome the obstacle of small formal time scales and speed up the integration of the governing equations, different methods have been implemented into the ANTARES code – the code used by our group to simulate convection in pulsating and non-pulsating stars and double–diffusive convection, among others.

The overarching topic of this thesis is the advancement of the applicability of one of these methods: a strong stability preserving implicit-explicit (IMEX) Runge-Kutta scheme as introduced into the ANTARES code by F. Kupka et al., (2012). Up to now, these IMEX-RK schemes have been implemented for the case of two spatial dimensions only and take into account only heat and solute diffusion as the processes possibly introducing the smallest time scale into the entire problem. One long–term goal of our group is the realistic hydrodynamic simulation of the double–diffusive convection that occurs in giant gas planets, as it is called for by J. Leconte and G. Chabrier, (2012), e.g., because it is one of the candidates that could explain the luminosity anomaly of Saturn (J. Leconte and G. Chabrier, 2013). To be able to achieve that, several steps must be undertaken: first of all, we have to be able to use the IMEX method for flows that are not only limited by heat and solute diffusion, but by viscosity, too, because the viscous time scale of the flow problem also induces a limitation to the time step in simulations of convection in giant planets with moderate Prandtl numbers. One part of this theses is the derivation of the IMEX equations for these by viscosity limited flows. Secondly, the simulations must be able to be performed in three dimensions because the rotation of planets does call for a third dimension. Thirdly, another simplification that has been used up to now in simulations of double–diffusive convection is the assumption that the thermal conductivity K_T and the concentration diffusivity κ_S are constants. This is not the case in most physical flows and thus, we need to address this simplification and introduce a way to solve equations where K_T and κ_S are not constant. A consequence of this is that the partial differential equation which results from the implicit part of the IMEX scheme has now non-constant coefficients which are either dependent on space only – in which case the equation to be solved is linear – or on space and temperature, e.g. – in which case the equation is nonlinear.

To solve the arising (non-) linear equations of Helmholtz type, we derive and implement a multigrid method for both the linear and nonlinear, variable coefficients Helmholtz equation in three dimensions. This solver is based on the excellent multigrid solver for the two–dimensional Helmholtz equation that has been designed and implemented by Happenhofer, (2014) and is crucial both for efficient use of the IMEX methods developed herein but also for future extensions of ANTARES, as e.g. the use of the Eddington approximation for the radiative transport.

Finally, it is demonstrated that the modified code is more efficient than its predecessor for simulations of convection in a two-component fluid, where time step limitations are introduced by either heat diffusion or viscosity, for the case of the Boussinesq approximation.

Zusammenfassung

Eine signifikante Schwierigkeit bei der Simulation physikalischer Systeme stellen die unterschiedlichen Zeitskalen dar, auf denen die verschiedenen physikalischen Prozesse ablaufen: die kleinste dieser Zeitskalen legt den maximal erlaubten Zeitschritt für die gesamte Simulation fest. Dies kann Berechnungen erheblich verlangsamen. Um dieses Hindernis zu umgehen und die Integration der hydrodynamischen Grundgleichungen zu beschleunigen, wurden unterschiedliche Methoden in den ANTARES Code – den Code unserer Gruppe, der u.a. zur Simulation von Konvektion in pulsierenden und nicht-pulsierenden Sternen und doppelt-diffusiver Konvektion benutzt wird – implementiert.

Das übergreifende Thema dieser Dissertation ist die Weiterentwicklung einer dieser Methoden: einer stark Stabilität erhaltenden implizit-expliziten (IMEX) Runge–Kutta Methode, die von F. Kupka et al., (2012) für den ANTARES Code vorgestellt wurde. Bisher sind diese IMEX Methoden für 2 Dimensionen implementiert und berücksichtigen ausschließlich Wärme- und Konzentrationsdiffusion als die Prozesse, welche den kleinsten Zeitschritt diktieren. Ein langfristiges Ziel unserer Gruppe ist die realistische hydrodynamische Simulation doppelt-diffusiver Konvektion in Gasriesenplaneten, was z.B. von J. Leconte and G. Chabrier, (2012) gefordert wird, da es einer der Kandidaten dafür ist, die Luminositätsanomalie von Saturn zu erklären (J. Leconte and G. Chabrier, 2013). Um dies zu erreichen, sind mehrere Schritte notwendig: zunächst muss dafür gesorgt werden, dass die IMEX Methode auch für Strömungen genutzt werden kann, deren Zeitschritt nicht nur durch Wärme- und Konzentrationsdiffusion beschränkt wird, sondern dass auch solche Strömungen effektiv berechnet werden können, deren Zeitintegration von viskösen Prozessen gebremst wird. Der Grund ist, dass die visköse Zeitskala bei Simulationen von Konvektion in Gasriesenplaneten mit moderater Prandtl Zahl eine Beschränkung darstellt. Ein Teil dieser Dissertation besteht aus der Herleitung der IMEX-Gleichungen für diese, durch Viskosität limitierten, Strömungen. Zweitens müssen die Simulationen in drei Dimensionen durchführbar sein, da die Rotation von Planeten eine dritte Dimension verlangt. Drittens muss eine weitere Vereinfachung fallen gelassen werden, die bisher genutzt wurde: in Simulationen doppelt-diffusiver Konvektion wird angenommen, dass die thermische Leitfähigkeit K_T und die Konzentrationsdiffusion κ_S konstant sind. Dies ist in dem meisten physikalischen Systemen nicht der Fall. Wir müssen also eine Methode entwickeln, die Gleichungen zu lösen, wenn K_T und κ_T nicht konstant sind. Eine Konsequenz daraus ist, dass die partielle Differentialgleichung, die aus dem impliziten Teil der IMEX Methode herrührt, nicht-konstante Koeffizienten hat, die entweder nur von den Raumkoordinaten abhängen – in welchem Fall die zu lösende Gleichung linear ist – oder von den Raumkoordinaten und z.B. der Temperatur abhängen – in welchem Fall die zu lösende Gleichung nichtlinear ist.

Um diese Gleichung zu lösen, wird ein Multigrid Löser entwickelt, der sowohl die lineare, als auch die nichtlineare generalisierte Helmholtzgleichung in drei Dimensionen lösen kann. Der Löser basiert auf dem exzellenten Multigrid Löser für die zweidimensionale Helmholtzgleichung, der von Happenhofer, (2014) entwickelt wurde. Der entwickelte Löser ist sowohl für die effiziente Nutzung der in dieser Dissertation entwickelten IMEX Methode, als auch für zukünftige Erweiterungen von ANTARES wie z.B. die Nutzung der Eddington-Approximation von fundamentaler Wichtigkeit.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

As Mortensen, (2011) puts it, computer simulations are nowadays an integral part of basic and applied research in the sciences and have evolved themselves to be a third standalone discipline besides theory and experiment. Applications of computer simulations span across a multitude of disciplines: computational materials science, bioinformatics, computational mathematics and mechanics, computational chemistry and so on. Simulations enable the scientist to conduct research in areas that are otherwise inaccessible: there may be no analytic solution due to the complexity of the system and putting up an experiment would be too expensive or just impossible to achieve (e.g. studying the effects of the collision of two black holes). One big research area in computational science is the numerical simulation of flows: computational fluid dynamics.

To underline the importance of this field we give one example: Anderson, (1995) says, speaking about trans-atmospheric vehicles, that "... anyone steeped in the history of aeronautics, where the major thrust has always been to fly faster and higher, knows that such vehicles will someday be a reality. But they will be made a reality only when computational fluid dynamics has developed to the point where the complete three-dimensional flow field over the vehicle and through the engines can be computed expeditiously with accuracy and reliability.". One can see from these two sentences alone that computational fluid dynamics is a field with important applications and that still is an area of very active research. Being able to solve the full governing equations of fluid dynamics efficiently and accurately in three dimensions while including every possible physical sub-process is still something that is not feasible today. Although we are purely focusing on astrophysical applications in this thesis, the numerical methods that are developed can be applied to all the fields where CFD is used. Thus, it is also an important contribution to the general computational community as a whole.

Due to its interdisciplinarity, computational astrophysics has close relations to numerical analysis and computer science: numerical analysis is concerned with the approximation of func-

tions and the integration of differential equations. Convergent, consistent and stable algorithms are developed in this branch of mathematics that are all essential for their use in the simulations of computational astrophysics and the development of new algorithms has to use the tools of numerical analysis to be effective.

The overlap with computer science stems from working on software problems or the development of standards for parallel processing such as MPI and OpenMP, which both are methods that have been implemented into our own code ANTARES, of which we give a short overview next.

1.1 ANTARES

ANTARES is a numerical code which was originally conceived for simulations in stellar hydrodynamics. It was first mentioned by H. J. Muthsam, Löw-Baselli, et al., (2007), followed by a comprehensive description by H. Muthsam et al., (2010). Numerous updates on numerical methods and applications have been published until today as e.g. H. J. Muthsam, (2011), H. J. Muthsam, F. Kupka, et al., (2011), Mundprecht, H. J. Muthsam, and F. Kupka, (2013) or Blies, F. Kupka, and H. J. Muthsam, (2015) to name a few. ANTARES has been designed with the following principles in mind:

- general: time dependent compressible hydrodynamics and extensions (such as MHD) in 1D, 2D and 3D; Fortran90 code with a modular structure
- numerics: various high resolution numerical schemes of conservative form implemented:
  - hyperbolic terms discretized with ENO, WENO, CNO schemes with adaptive stencils
  - parabolic terms discretized by dissipative finite difference schemes of fourth order
  - time integration done with total variation diminishing Runge-Kutta methods
- radiative transfer: short characteristics method (use of the diffusion approximation where appropriate); either gray or non-gray by the binning method based on state of the art opacities;
- micro-physics: realistic (or idealized);
- gridding: logically rectangular; either rectangular or spherical coordinates; equidistant or (vertically) logarithmically spaced grid points; grid refinement in pre-assigned rectangular patches, even recursively, nested grids: at the moment, ANTARES provides up to three levels of nested grids;
- parallelization: via MPI and OpenMP;
- portability: is running on

  - AMD, IBM and Intel processors
  - AIX, Linux and OSX operating systems

The area of application of ANTARES is threefold: extremely high resolution 3D simulations of solar surface convection (H. Muthsam et al., 2010) or of other types of stars for which the same computational approach is applicable, 2D simulations of the convection zones of pulsating stars (Mundprecht, H. J. Muthsam, and F. Kupka, 2013)—both with realistic microphysics and opacities—and idealized 2D and 3D simulations of semiconvection (F. Zaussinger and Spruit, 2013). An overview on the code was given by H. Muthsam et al., (2010). Here, we shortly summarize which new features have since been added. The newest developments of ANTARES include

- the capability to calculate two-component flows via (1.4) (F. Zaussinger, 2010),
- the capability to solve the equations for incompressible flow in the Boussinesq approximation (F. Zaussinger, 2010),
- an operator splitting method to handle low-Mach number flows without modification of the basic equations (such as the anelastic approximation) (Happenhofer et al., 2013),
- a strong stability preserving implicit-explicit Runge–Kutta method for the set of compressible equations (Higueras, 2006; Higueras, 2009; F. Kupka et al., 2012; Happenhofer, 2014; Higueras et al., 2014) when diffusion processes limit the maximal time step,
- a parallel multigrid solver for the two–dimensional, generalized, non–linear Helmholtz equation (Happenhofer, 2014) and
- the capability to solve the Navier–Stokes equations on curvilinear grids (Grimm-Strele, 2014; Grimm-Strele, F. Kupka, and H. Muthsam, 2014).

The features that are developed for ANTARES in this thesis are

1. a strong stability preserving implicit-explicit Runge–Kutta method for the set of incompressible equations in the Boussinesq approximation when diffusion processes or diffusion and viscous processes limit the maximal time step and
2. a parallel multigrid solver for the three–dimensional, generalized, non–linear Helmholtz equation.

These additions open up ANTARES to the following new applications and enhancements:

- simulating compressible and incompressible flows with non–constant diffusivities,
- increasing the efficiency of simulations of incompressible flow where diffusive or viscous processes limit the size of the time step,

- they lay the foundation for the implementation of the Eddington approximation, which severely speeds up simulations of stars where radiative transfer is important.

1.2 The Governing Equations of Fluid Dynamics

ANTARES is capable to solve the equations of fluid dynamics in several different forms.

1.2.1 The Equations for Compressible Flow

The most general form of the equations are the compressible Navier–Stokes equations:

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \boldsymbol{u}), \tag{1.1}$$

$$\frac{\partial (\rho \boldsymbol{u})}{\partial t} = -\nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}^{\mathrm{T}}) - \nabla p + \nabla \cdot \sigma + \rho \boldsymbol{g}, \tag{1.2}$$

$$\frac{\partial (\rho E)}{\partial t} = -\nabla \cdot (\rho E \boldsymbol{u} + p \boldsymbol{u}) + \nabla \cdot (\boldsymbol{u}^{\mathrm{T}} \sigma) + \rho \boldsymbol{u}^{\mathrm{T}} \boldsymbol{g} + Q_{rad} \tag{1.3}$$

with the recent addition of the concentration of a second species:

$$\frac{\partial (\rho c)}{\partial t} = -\nabla \cdot (\rho c \boldsymbol{u}) + \nabla \cdot (\rho \kappa_c \nabla c). \tag{1.4}$$

In stellar simulations Q_{rad} in (1.3) is obtained either by the diffusion approximation or by solving the stationary limit of the radiative transfer equation

$$\boldsymbol{r} \cdot \nabla I_\nu = \chi_\nu (S_\nu - I_\nu) \tag{1.5}$$

for all ray directions $\boldsymbol{r}$ and for all frequencies ν. This makes ANTARES a full radiation hydrodynamics code. We do not treat compressible flows in this thesis, however. Instead, we look at a form of the equations often used when treating incompressible flows: the Boussinesq approximation.

1.2.2 The Equations in the Boussinesq Approximation

It is often useful to simplify the basic equations to reduce the effort to solve them. One of these simplifications is the *Boussinesq approximation*. It was applied for the first time by Boussinesq, (1903). Spiegel and Veronis, (1960) summarized the approximation as follows:

1. “The fluctuations in density which appear with the advent of motion result principally from thermal (as opposed to pressure) effects.

2. In the equations for the rate of change of momentum and mass, density variations may be neglected except when they are coupled to the gravitational acceleration in the buoyancy force.” (Spiegel and Veronis, 1960)

As seen in Cohen and Kundu, (2002, p. 118), the incompressibility assumption is not valid in

1. steady flows with large Mach numbers ($Ma > 0.3$) because large pressure changes cause large density changes at high Mach numbers.

2. unsteady flows: waves would propagate at infinite speed if the density variations were neglected.

3. flows where the vertical scale of the flow is so large that the hydrostatic pressure variations cause large changes in density. The Boussinesq Approximation requires that the vertical scale of the flow is $L \ll c^2/g$ with c being the speed of sound in the medium.

The governing equations take the following form in the Boussinesq approximation (e.g. Cohen and Kundu, 2002):

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.6}$$

$$\left(\frac{\partial \boldsymbol{u}}{\partial t}\right) + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\frac{\nabla p}{\rho_0} + \nu \nabla^2 \boldsymbol{u} + \frac{\rho}{\rho_0}\boldsymbol{g}, \tag{1.7}$$

$$\frac{\partial T}{\partial t} + (\boldsymbol{u} \cdot \nabla)T = \kappa_T \nabla^2 T, \tag{1.8}$$

$$\frac{\partial S}{\partial t} + (\boldsymbol{u} \cdot \nabla)S = \kappa_S \nabla^2 S, \tag{1.9}$$

with $\rho = \rho_0[1 - \alpha(T - T_0) + \beta(S - S_0)]$. $\boldsymbol{u}$ is the velocity of the flow, p the pressure, ρ the density, ρ_0 a reference density, ν the kinematic viscosity, T the temperature, T_0 the reference temperature where $\rho = \rho_0$, S the salinity, S_0 the reference salinity where $\rho = \rho_0$, κ_T the thermal diffusivity and κ_S the molecular diffusivity. α is the thermal expansion coefficient $-\rho_0^{-1}(\partial\rho/\partial T)_S$, β is the saline expansion coefficient $\rho_0^{-1}(\partial\rho/\partial S)_T$.

1.2.3 Non–Dimensionalization of the Boussinesq Equations

It is usual in fluid dynamics to non–dimensionalize the equations. This makes it possible to use the same simulations to describe physical systems vastly differing in magnitudes, as long as the geometry is similar. Ones begins by choosing scales that are typical for the system to be simulated. For ANTARES, the equations are non-dimensionalized with the scales

$$r = Lr^*, \quad T - T_0 = \Delta T T^*, \quad S - S_0 = \Delta S S^*, \quad \boldsymbol{u} = \frac{\kappa_T}{L}\boldsymbol{u}^*, \quad t = \frac{L^2}{\kappa_T}t^*. \tag{1.10}$$

L is a typical length scale, ΔT and ΔS are the differences of temperature and salinity between lower and upper boundary. The time scale used is the thermal diffusion time scale.

1.2.3.1 Non-Dimensionalization of the Continuity Equation

We start with the continuity equation (1.6):

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1.11}$$

$$\frac{\kappa_T}{L^2}\nabla^* \boldsymbol{u}^* = 0 \tag{1.12}$$

Dividing by κ_T/L^2 and dropping * for better readability leads to

$$\nabla \cdot \boldsymbol{u} = 0. \tag{1.13}$$

We see that the equation has exactly the same form as the equation with dimensions.

1.2.3.2 Non-Dimensionalization of the Momentum Equation

We start with (1.7):

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{\nabla p}{\rho_0} + \nu\nabla^2\boldsymbol{u} + \frac{\rho}{\rho_0}\boldsymbol{g} \tag{1.14}$$

Non–dimensionalizing with above scales gives

$$\frac{\kappa_T^2}{L^3}\frac{D\boldsymbol{u}^*}{Dt^*} = -\frac{\nabla p}{\rho_0} + \frac{\kappa_T}{L^3}\nu\nabla^{*2}\left(\boldsymbol{u}^*\right) + \left[1 - \alpha\Delta T T^* + \beta\Delta S S^*\right]\boldsymbol{g}. \tag{1.15}$$

Multiplying with L^3/κ_T^2 and κ_T/ν gives

$$\underbrace{\frac{\kappa_T}{\nu}}_{\mathrm{Pr}^{-1}}\frac{D\boldsymbol{u}^*}{Dt^*} = \underbrace{-\frac{L^3}{\kappa_T\nu\rho_0}\nabla p + \frac{L^3}{\kappa_T\nu}\boldsymbol{g}}_{-\nabla p_{\mathrm{eff}}} + \nabla^{*2}\boldsymbol{u}^* - \underbrace{\frac{\alpha L^3\Delta T\boldsymbol{g}}{\kappa_T\nu}}_{\mathrm{Ra_T}}T^* + \underbrace{\frac{\beta L^3\Delta S\boldsymbol{g}}{\kappa_T\nu}}_{\mathrm{Ra_S}}S^*. \tag{1.16}$$

In ANTARES, the x-axis is pointing towards the center of the star/planet, so the vector of gravitational acceleration is $\boldsymbol{g} = (-g, 0, 0)^{\mathrm{T}}$. Using that and dropping * for better legibility gives

$$\frac{1}{\mathrm{Pr}}\frac{D\boldsymbol{u}}{Dt} = -\nabla p_{\mathrm{eff}} + \nabla^2\boldsymbol{u} + \mathrm{Ra_T}T\boldsymbol{e_x} - \mathrm{Ra_S}S\boldsymbol{e_x}. \tag{1.17}$$

Here, we have introduced three typical non-dimensional numbers: $\mathrm{Pr}, \mathrm{Ra_T}$ and $\mathrm{Ra_S}$ and end up with the following three equations for the velocity vector $\boldsymbol{u} = (u, v, w)^{\mathrm{T}}$:

$$\frac{1}{\mathrm{Pr}}\frac{Du}{Dt} = -\nabla p_{\mathrm{eff}} + \nabla^2 u + \mathrm{Ra_T}T - \mathrm{Ra_S}S, \tag{1.18}$$

$$\frac{1}{\mathrm{Pr}}\frac{Dv}{Dt} = -\nabla p_{\mathrm{eff}} + \nabla^2 v, \tag{1.19}$$

$$\frac{1}{\mathrm{Pr}}\frac{Dw}{Dt} = -\nabla p_{\mathrm{eff}} + \nabla^2 w. \tag{1.20}$$

Note that ∇p_{eff} is different for u than it is for v and w.

1.2.3.3 Non-Dimensionalization of the Temperature equation

Starting with (1.8),

$$\frac{DT}{Dt} = \kappa_T\nabla^2 T, \tag{1.21}$$

non-dimensionalizing leads to

$$\frac{\kappa_T}{L^2}\frac{D(\Delta T T^* + T_0)}{Dt^*} = \frac{\kappa_T}{L^2}\nabla^{*2}(\Delta T T^* + T_0). \tag{1.22}$$

Multiplying with L^2/κ_T and dividing by ΔT gives

$$\frac{DT^*}{Dt^*} = \nabla^{*2} T^* \tag{1.23}$$

and dropping * leads to

$$\frac{DT}{Dt} = \nabla^2 T. \tag{1.24}$$

As with the continuity equation there is no additional dimensionless number occurring in the temperature equation.

1.2.3.4 Non-Dimensionalization of the Salinity equation

Finally, starting with (1.9),

$$\frac{DS}{Dt} = \kappa_S \nabla^2 S, \tag{1.25}$$

non-dimensionalizing leads to

$$\frac{\kappa_T}{L^2}\frac{D(\Delta S^* + S_0)}{Dt} = \kappa_S \frac{1}{L^2}\nabla^{*2}(\Delta S S^* + S_0) \tag{1.26}$$

Multiplying with L^2/κ_T and dividing by ΔS gives

$$\frac{DS^*}{Dt^*} = \underbrace{\frac{\kappa_S}{\kappa_T}}_{\mathrm{Le}} \nabla^{*2} S^*. \tag{1.27}$$

Dropping * gives

$$\frac{DS}{Dt} = \mathrm{Le}\nabla^2 S. \tag{1.28}$$

Here, we have introduced the Lewis number Le.

1.2.4 Summary

For the Boussinesq approximation, we end up with the five non-dimensionalized equations

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.29}$$

$$\mathrm{Pr}^{-1}\left[\left(\frac{\partial \boldsymbol{u}}{\partial t}\right) + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\right] = -\nabla p_{\mathrm{eff}} + \nabla^2 \boldsymbol{u} + Ra_T T \boldsymbol{e_x} - Ra_S S \boldsymbol{e_x}, \tag{1.30}$$

$$\frac{\partial T}{\partial t} + (\boldsymbol{u}\cdot\nabla)T = \nabla^2 T, \tag{1.31}$$

$$\frac{\partial S}{\partial t} + (\boldsymbol{u}\cdot\nabla)S = \mathrm{Le}\nabla^2 S, \tag{1.32}$$

for which we have introduced the dimensionless numbers

$$\mathrm{Pr} = \frac{\nu}{\kappa_T}, \quad \mathrm{Le} = \frac{\kappa_S}{\kappa_T}, \quad Ra_T = \frac{\alpha L^3 \Delta T g}{\kappa_T \nu}, \quad Ra_S = \frac{\beta L^3 \Delta S g}{\kappa_T \nu}. \tag{1.33}$$

$\mathrm{Ra_T}$ and $\mathrm{Ra_S}$ are related to each other by the stability parameter $R_\rho = Ra_S/Ra_T$. We have also introduced the effective pressure p_{eff} which is the gradient of the scaled original pressure plus a constant term. This can be written in this form because the constant term vanishes in the course of solving the equations.

Even though the equations in the Boussinesq form are significantly easier to handle and solve than the full Navier–Stokes equations, there are still important considerations to make when trying to simulate a system that is governed by these equations. One problem that requires a special consideration are the different physical processes which the different terms in the equations represent and the different time scales on which these processes operate. The advective terms in the equations are of hyperbolic type while the diffusive and viscous processes are of the parabolic type. This demands extra care when deciding on the time step to use for the simulations because parabolic type partial differential equations significantly limit the time step that can be taken with explicit time integration methods. Equations where the terms change on vastly different time scales are called stiff equations and are introduced in the next section.

1.3 Time Step Restrictions by Stiff Equations

The following example is taken out of Hoffman, (2001) to demonstrate the behavior of stiff equations. There, stiff equations are demonstrated with the following ordinary differential equation, which Gear, (1971) considered:

$$\begin{aligned} y' &= -\alpha(y - F(t)) + F'(t), \\ y(t_0) &= y_0 \end{aligned} \tag{1.34}$$

The exact solution is

$$y(t) = (y_0 - F(0))\exp(-\alpha t) + F(t) \tag{1.35}$$

If $\alpha \gg 0$ and $F(t)$ is a smooth and slowly varying function, two different time scales become apparent:

- the rapidly varying component $(y_0 - F(0))\exp(-\alpha t)$
- the slowly changing term $F(t)$

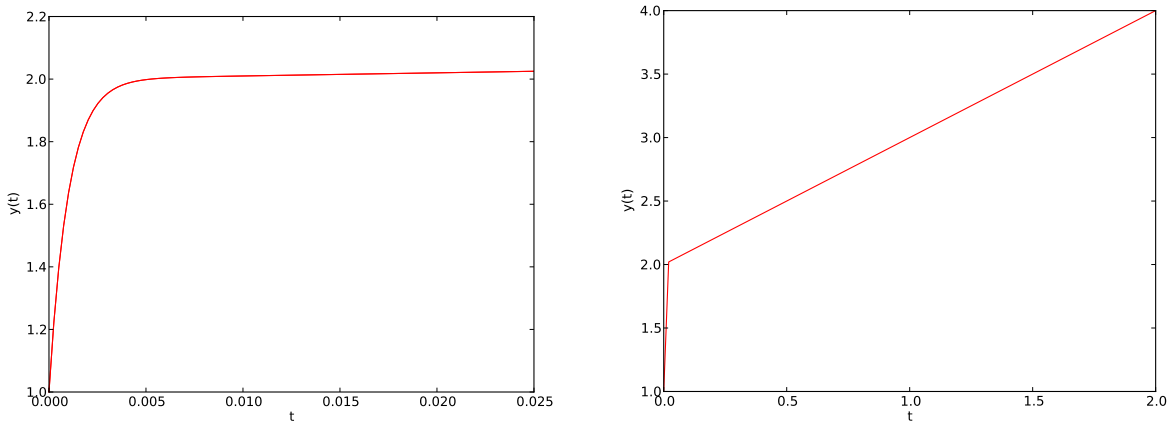
As a concrete example, let $\alpha = 1000$, $F(t) = t + 2$ and $y(0) = 1$. Then, (1.34) becomes

$$\begin{aligned} y' &= -1000(y - (t-2)) + 1, \\ y(0) &= 1 \end{aligned} \tag{1.36}$$

with the solution

$$y(t) = -\exp(-1000t) + t + 2. \tag{1.37}$$

If one were interested in the solution of (1.36) for large values of t only, one might try to use



(a) Small times

(b) Large times

Figure 1.1: Plot of the analytical solution (1.37) for small and large values of t.

a simple explicit time integration method with a large time step Δt. This leads to an unstable simulation, however: using, e.g., Euler forward,

$$y_{n+1} = y_n + \Delta t f_n, \tag{1.38}$$

leads to

$$y_{n+1} = y_n + \Delta t(-1000(y_n - (t_n + 2)) + 1). \tag{1.39}$$

Figure 1.2 shows the exact and numerical solutions for time steps $0.005, 0.01, 0.02$ and 0.025. One can see that while the numerical solution is a somewhat close approximation to the exact solution for very small time steps, for larger time steps the solution is unstable. This is a typical behavior of explicit method when treating stiff equations: the stability depends crucially on the time step Δt taken. This means that even if one was only interested in the solution of (1.36) for, say, $t > 10$, one still needed to take time steps of about 0.001 until $t = 10$ was reached. That is 10000 unnecessary steps before the time of interest is reached.

This is exactly the situation which we have when doing fluid dynamical simulations. The physical processes with the smallest time scales can be, depending on the specific setting, diffusion processes, viscous processes or the propagation of sound waves. Whatever process is the one operating on the smallest time scale demands the strictest time step restriction because it must be resolved numerically when using explicit methods. This leads to computationally very expensive calculations.

One way to alleviate this problem is the use of implicit time integration methods. These are methods which are especially suited for stiff equations. A severe drawback of implicit methods

are their high computational cost, however. They require the solution of a (non–) linear system of equations each time step which is more expensive to perform than just a simple explicit step. The advantage is the often unconditional stability of the method, however, which means that time steps much larger that the one limiting the maximal time step in explicit methods can be used in the simulation. So we get to the interesting parts of the simulation with much fewer (but more expensive) time steps.

For this reason, one tries to limit the use of implicit methods to those parts of the governing equations which are stiff, i.e., one splits the equations in a stiff and a non–stiff part and uses implicit schemes for the stiff parts and explicit schemes for the non–stiff parts. This is exactly the method which we use in part II to develop an implicit-explicit Runge–Kutta scheme for the Boussinesq equations.

One major obstacle in the development of these methods is the occurence of a a three-dimensional Helmholtz equation with varying coefficients either in linear form,

$$-\nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))+\xi(\boldsymbol{x})u(\boldsymbol{x})=f(\boldsymbol{x}), \tag{1.40}$$

or in nonlinear form

$$-\nabla\cdot(\kappa(u)\nabla u(\boldsymbol{x}))+\xi(u)=f(\boldsymbol{x}) \tag{1.41}$$

with $u:\mathbb{R}^3\to\mathbb{R}$, $\boldsymbol{x}\in\mathbb{R}^3$, $\kappa:\mathbb{R}^3\to\mathbb{R}^+$, $\xi:\mathbb{R}^3\to\mathbb{R}^+$ and $f:\mathbb{R}^3\to\mathbb{R}$. The boundary conditions can either be of Dirichlet type

$$u=g_D \quad \text{on } \Gamma_D \tag{1.42}$$

or of Neumann type

$$\nabla u\cdot\boldsymbol{n}=\frac{\partial u}{\partial n}=g_N \quad \text{on } \Gamma_N. \tag{1.43}$$

The equations are to be solved over a rectangular domain $\Omega\subset\mathbb{R}^3$.

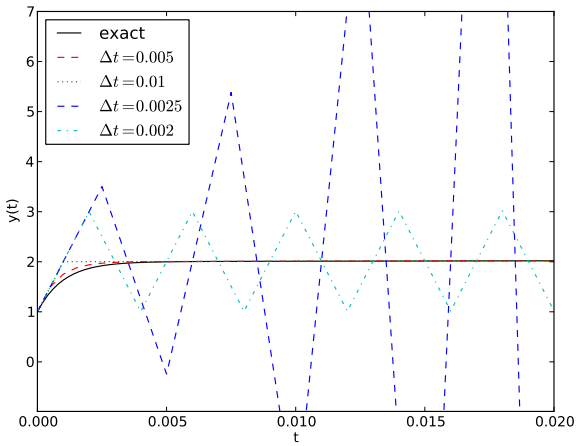


Figure 1.2: The influence of the time step Δt of an explicit scheme on the stability of the solution when solving 1.36 with Euler forward.

Up to this point, there is no method implemented in ANTARES which is able to solve this problem in three dimensions. Happenhofer, (2014) has developed a solver for the two- dimensional form of the equation but before we will be able to tackle the problem of stiff equations with IMEX methods we have to develop a solver for (1.40) and (1.41). This is done in part I which follows now.

Part I

The Multigrid Solver

Introduction

In ANTARES, depending on the physical system to be simulated, we end up with a linear generalized Helmholtz equation of type

$$-\nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))+\xi(\boldsymbol{x})u(\boldsymbol{x})=f(\boldsymbol{x}) \tag{1.44}$$

or with a nonlinear generalized Helmholtz equation of type

$$-\nabla\cdot(\kappa(u)\nabla u(\boldsymbol{x}))+\xi(u)=f(\boldsymbol{x}), \tag{1.45}$$

which are to be solved over a domain $\Omega\subset\mathbb{R}^3$. The boundary conditions can either be of Dirichlet type

$$u=g_D \quad \text{on } \Gamma_D \tag{1.46}$$

or of Neumann type

$$\nabla u\cdot\boldsymbol{n}=\frac{\partial u}{\partial n}=g_N \quad \text{on } \Gamma_N. \tag{1.47}$$

Mixed boundary conditions are not considered here.

To solve the arising Helmholtz equations we discretize them and solve the resulting linear system of equations with the multigrid method which is one of the fastest methods available for this purpose. The derivation of the solver follows this outline:

- discretization of the
  - linear equation (chapter 2.2)
  - nonlinear equation (chapter 2.3)
- derivation of the multigrid components for the
  - linear equation (chapter 3)
  - nonlinear equation (chapter 3.3)
- testing the multigrid solver by running several test cases for the
  - linear equation
  - nonlinear equation

General introductory remarks regarding the discretization, the multigrid method and the test outline are presented in the parts for the linear equation.

CHAPTER 2

DISCRETIZATION OF THE HELMHOLTZ EQUATION

2.1 Introductory Remarks

2.1.1 The Finite Elements Method

The discretization of (1.44) and (1.45) is done with the finite elements method. In general, the following steps are required to discretize a differential equation with the finite elements method:

1. write the problem in variational formulation
2. choose the specific FEM method (e.g. Collocation, Galerkin)
3. choose a finite dimensional subspace V_h and the basis function φ_μ
4. calculate the resulting matrices and vectors

Details on the mathematical theory behind the finite elements method can be found in Brenner and Scott, (2008) or Quarteroni and Valli, (1994). A more introductory exposition can be found, e.g., in Hanke-Bourgeois, (2009).

2.2 Discretization of the Linear Equation

In this part we discretize the linear Helmholtz equation

$$-\nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))+\xi(\boldsymbol{x})u(\boldsymbol{x})=f(\boldsymbol{x}). \tag{2.1}$$

The boundary condition is of either Dirichlet type or Neumann type. Mixed boundary conditions are not implemented in ANTARES. We derive the discretized system of equations for

homogeneous Dirichlet, non-homogeneous Dirichlet and Neumann boundary conditions. The derivation follows along the lines of Quarteroni, (2009).

2.2.1 Deriving the Variational Formulation

2.2.1.1 Variational Formulation of the Homogeneous Dirichlet Problem

The homogeneous Dirichlet problem reads:
find u such that

$$\begin{cases} -\nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))+\xi(\boldsymbol{x})u(\boldsymbol{x})=f(\boldsymbol{x}) & \text{in } \Omega \\ u=0 & \text{on } \partial\Omega=\Gamma_D \end{cases} \tag{2.2}$$

where $\Omega\subset\mathbb{R}^3$ is a bounded domain with boundary $\partial\Omega$.

To obtain the variational formulation of (2.2) we multiply by an arbitrary test function $v(\boldsymbol{x})$ and integrate over the domain Ω:

$$-\int_\Omega \nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))v(\boldsymbol{x})\,\mathrm{d}\Omega+\int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega=\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega. \tag{2.3}$$

By applying Green's formula to the first integral we obtain

$$\int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega-\int_{\Gamma_D}\kappa(\boldsymbol{x})\frac{\partial u}{\partial n}v(\boldsymbol{x})\,\mathrm{d}\Gamma+\int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega=\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega. \tag{2.4}$$

For the homogeneous Dirichlet problem, we impose the condition that the (arbitrary) testfunction $v(\boldsymbol{x})$ is zero on the boundary $\partial\Omega$, by which the boundary integral vanishes and we are left with

$$\int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega+\int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega=\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega. \tag{2.5}$$

So we get the following weak formulation for problem (2.2):

$$\begin{aligned} &\text{find } u\in \mathrm{H}^1_0(\Omega):\\ &\int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega+\int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega=\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \quad \forall v\in \mathrm{H}^1_0(\Omega), \end{aligned} \tag{2.6}$$

where $f\in \mathrm{L}^2(\Omega)$ and

$$\begin{aligned} &\mathrm{H}^1_0=\{v\in \mathrm{H}^1(\Omega): v=0 \text{ on } \partial\Omega\},\\ &\mathrm{H}^1=\{v:\Omega\to\mathbb{R} \text{ s.t. } v\in \mathrm{L}^2(\Omega), \frac{\partial v}{\partial x_i}\in \mathrm{L}^2(\Omega), i=1,2,3\}. \end{aligned} \tag{2.7}$$

Setting $V=\mathrm{H}^1_0$, we define the bilinear form

$$a: V\times V\to\mathbb{R}, \quad a(u,v)=\int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega+\int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \tag{2.8}$$

and the linear and continuous functional

$$F : V \to \mathbb{R}, \quad F(v) = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega \tag{2.9}$$

whereby the problem can be written as

$$\text{find } u \in V : \quad a(u,v) = F(v) \quad \forall v \in V. \tag{2.10}$$

2.2.1.2 Variational Formulation of the Non–homogeneous Dirichlet Problem

The non–homogeneous Dirichlet problem reads:
find u such that

$$\begin{cases} -\nabla \cdot (\kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x})) + \xi(\boldsymbol{x}) u(\boldsymbol{x}) = f(\boldsymbol{x}) & \text{in } \Omega \\ u = g_D & \text{on } \partial\Omega = \Gamma_D \end{cases} \tag{2.11}$$

where $\Omega \subset \mathbb{R}^3$ is a bounded domain with boundary $\partial\Omega$.

To obtain the variational formulation of (2.11) we multiply by an arbitrary test function $v(\boldsymbol{x})$ and integrate over the domain Ω:

$$-\int_\Omega \nabla \cdot (\kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x})) v(\boldsymbol{x}) \, \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) u(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega. \tag{2.12}$$

By applying Green's formula to the first integral we obtain

$$\int_\Omega \kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x}) \nabla v(\boldsymbol{x}) \, \mathrm{d}\Omega - \int_{\Gamma_D} \kappa(\boldsymbol{x}) \frac{\partial u}{\partial n} v(\boldsymbol{x}) \, \mathrm{d}\Gamma + \int_\Omega \xi(\boldsymbol{x}) u(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega. \tag{2.13}$$

For the non–homogeneous Dirichlet problem, we again impose the condition that the (arbitrary) testfunction $v(\boldsymbol{x})$ is zero on the boundary $\partial\Omega$, by which the boundary integral vanishes and we are left with

$$\int_\Omega \kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x}) \nabla v(\boldsymbol{x}) \, \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) u(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega. \tag{2.14}$$

The difference to the homogeneous Dirichlet problem (2.2) becomes apparent in the weak formulation:

$$\text{find } u \in V_g : \quad \int_\Omega \kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x}) \nabla v(\boldsymbol{x}) \, \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) u(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\Omega \quad \forall v \in \mathrm{H}^1_{\Gamma_D}(\Omega), \tag{2.15}$$

where $f \in \mathrm{L}^2(\Omega)$ and

$$\begin{aligned} \mathrm{H}^1_{\Gamma_D} &= \{v \in \mathrm{H}^1(\Omega) : v = 0 \text{ on } \Gamma_D\}, \\ V_g &= \{v \in \mathrm{H}^1(\Omega) : v = g_D \text{ on } \Gamma_D\}, \\ \mathrm{H}^1 &= \{v : \Omega \to \mathbb{R} \text{ s.t. } v \in \mathrm{L}^2(\Omega), \frac{\partial v}{\partial x_i} \in \mathrm{L}^2(\Omega), i = 1,2,3\}. \end{aligned} \tag{2.16}$$

This formulation is problematic because $u \in V_g$ while $v \in \mathrm{H}^1_{\Gamma_D}$. To deal with that, we have to do a lifting of the boundary data in the following way.:

Lifting of Boundary Conditions

To handle non-homogeneous Dirichlet boundary conditions we introduce the continuous, piecewise linear function $u_{g_D} \in V_g$ s.t.

$$u = u^0 + u_{g_D}, \tag{2.17}$$

where u^0 stands for the homogeneous part of u, s.t. $u^0|_{\Gamma_D} = 0$, i.e. $u^0 \in \mathrm{H}^1_{\Gamma_D}$. We can now formulate the following variational formulation of the problem (2.11) in terms of u^0:

$$\begin{aligned}\text{find } u^0 \in \mathrm{H}^1_{\Gamma_D}(\Omega):&\\ &\int_\Omega \kappa(\boldsymbol{x})\nabla u^0(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x})u^0(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega\\ &\quad = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega - \int_\Omega \kappa(\boldsymbol{x})\nabla u_{g_D}(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega - \int_\Omega \xi(\boldsymbol{x})u_{g_D}(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \qquad \forall v \in \mathrm{H}^1_{\Gamma_D}(\Omega).\end{aligned}$$

We now have both u^0 and $v \in \mathrm{H}^1_{\Gamma_D}(\Omega)$. Setting $V = \mathrm{H}^1_{\Gamma_D}(\Omega)$, we define the bilinear form

$$a : V \times V \to \mathbb{R}, \quad a(u,v) = \int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \tag{2.18}$$

and the linear and continuous functional

$$F : V \to \mathbb{R}, \quad F(v) = -a(u_{g_D}, v) + \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \tag{2.19}$$

The problem can then be written as

$$\text{find } u^0 \in V: \quad a(u^0, v) = F(v) \quad \forall v \in V. \tag{2.20}$$

Note that the formulation is the same as in the homogeneous Dirichlet case. The differences lie in the form of $F(v)$ and in the space V.

2.2.1.3 Variational Formulation of the Neumann Problem

The Neumann problem reads:
find u such that

$$\begin{cases} -\nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})) + \xi(\boldsymbol{x})u(\boldsymbol{x}) = f(\boldsymbol{x}) & \text{in } \Omega \\ \nabla u \cdot \boldsymbol{n} = g_N & \text{on } \partial\Omega \end{cases} \tag{2.21}$$

where $\Omega \subset \mathbb{R}^3$ is a bounded domain with boundary $\partial\Omega$. For the pure Neumann problem to have a unique solution, we must require $\xi(\boldsymbol{x}) > 0$ everywhere.

To obtain the variational formulation of (2.21) we multiply by an arbitrary test function $v(\boldsymbol{x})$ and integrate over the domain Ω:

$$-\int_\Omega \nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega. \tag{2.22}$$

By applying Green's formula to the first integral we obtain

$$\int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega - \int_{\Gamma_N} \kappa(\boldsymbol{x})\frac{\partial u}{\partial n}v(\boldsymbol{x})\,\mathrm{d}\Gamma + \int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega. \tag{2.23}$$

We know from the problem that $\dfrac{\partial u}{\partial n} = g_N$, so we can immediately put the boundary integral to the right hand side and get

$$\int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_{\Gamma_N} \kappa(\boldsymbol{x})g_N v(\boldsymbol{x})\,\mathrm{d}\Gamma. \tag{2.24}$$

Thus, the weak formulation of (2.21) reads

$$\begin{aligned} &\text{find } u \in \mathrm{H}^1(\Omega): \\ &\int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \\ &\quad = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_{\Gamma_N} \kappa(\boldsymbol{x})g_N v(\boldsymbol{x})\,\mathrm{d}\Gamma \quad \forall v \in \mathrm{H}^1(\Omega), \end{aligned} \tag{2.25}$$

where $f \in \mathrm{L}^2(\Omega)$ and

$$\mathrm{H}^1 = \{v : \Omega \to \mathbb{R} \text{ s.t. } v \in \mathrm{L}^2(\Omega), \frac{\partial v}{\partial x_i} \in \mathrm{L}^2(\Omega), i = 1,2,3\}. \tag{2.26}$$

Setting $V = \mathrm{H}^1(\Omega)$, we define the bilinear form

$$a : V \times V \to \mathbb{R}, \quad a(u,v) = \int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \tag{2.27}$$

and the linear and continuous functional

$$F : V \to \mathbb{R}, \quad F(v) = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega + \int_{\Gamma_N} \kappa(\boldsymbol{x})g_N v(\boldsymbol{x})\,\mathrm{d}\Gamma \tag{2.28}$$

The problem can then be written as

$$\text{find } u \in V: \quad a(u,v) = F(v) \quad \forall v \in V. \tag{2.29}$$

Note that the formulation is the same as in both Dirichlet cases. The differences lie in the form of $F(v)$ and in the space V.

2.2.2 The Galerkin Finite Element Method

Having derived the variational formulations for different boundary conditions, we now turn towards the approximation with finite elements. Each boundary condition yields the same structure for the variational problem:

$$\text{find } u \in V: \quad a(u,v) = F(v) \quad \forall v \in V. \tag{2.30}$$

V is different for each boundary condition but they all are Hilbert spaces and subspaces of H^1.

Let V_h now be a family of spaces that depend on the positive parameter h, s.t.

$$V_h \subset V, \quad \dim V_h \equiv N_h < \infty \quad \forall h > 0. \tag{2.31}$$

The Galerkin problem that approximates (2.30) is then given by

$$\text{find } u_h \in V_h: \quad a(u_h, v_h) = F(v_h) \quad \forall v_h \in V_h. \tag{2.32}$$

We denote the basis of V_h as $\{\varphi_\mu, \mu = 1,2,...,N_h\}$ where N_h depends on the triangulation chosen and is equal to the number of nodes minus the number of Dirichlet nodes. Because all the functions in V_h are linear combinations of the basis functions φ_μ it suffices to verify (2.32) for each φ_μ of the subspace. We thus require that

$$a(u_h, \varphi_\mu) = F(\varphi_\mu), \quad \mu = 1,2,...,N_h. \tag{2.33}$$

Here, μ denotes a node on the computational grid G, which is defined as

$$G := \{(x,y,z) : x = x_i = ih_x, y = y_j = jh_y, z = z_k = kh_z; i,j,k \in \mathbb{Z}\}. \tag{2.34}$$

Since $u_h \in V_h$ we can expand it in terms of the basis functions of V_h as

$$u_h(\boldsymbol{x}) = \sum_{\nu=1}^{N_h} u_\nu \varphi_\nu(\boldsymbol{x}) \tag{2.35}$$

where the u_ν are unknown coefficients. Inserting (2.35) into (2.33) gives

$$\sum_{\nu=1}^{N_h} u_\nu a(\varphi_\nu, \varphi_\mu) = F(\varphi_\mu), \quad \mu = 1,2,...,N_h. \tag{2.36}$$

This is equivalent to the linear system

$$A\boldsymbol{u} = \boldsymbol{f}, \tag{2.37}$$

where A denotes the matrix with the elements

$$A_{\mu\nu} = a(\varphi_\nu, \varphi_\mu), \tag{2.38}$$

$\boldsymbol{f}$ is the vector with components

$$f_\mu = F(\varphi_\mu) \tag{2.39}$$

and $\boldsymbol{u}$ denotes the vector of unknown coefficients $\{u_\nu, \nu = 1,...,N_h\}$.

In order to solve (2.37), we need to specify the subspace V_h, to which we turn next.

2.2.3 The Finite Elements

We begin defining the finite dimensional subspace V_h by partitioning Ω into subdomains Ω_e, $e = 1,...,n_{\text{el}}$, where $\Omega_e \subset \Omega \subset \mathbb{R}^3$ and n_{el} is the number of partitions with the following properties:

- Ω_e is open in $\mathbb{R}^3$
- $\bar{\Omega} = \bigcup_{e=1}^{n_{\text{el}}} \Omega_e$
- $\Omega_i \cap \Omega_j = \emptyset$, $i \neq j$

The shape of the partitions Ω_e can be chosen at will but heavily depend on the form of Ω and especially its boundary. It is one of the ingredients needed to define finite elements.

Formally, a finite element, denoted by $(K, \mathscr{P}, \mathscr{N})$, consists of three items (see Brenner and Scott, 2008, e.g.):

- the element domain $K \subseteq \mathbb{R}^n$, where K is a bounded closed set with nonempty interior and piecewise smooth boundary
- the finite–dimensional space of shape functions $\mathscr{P}$ on K and
- the set of $n = \dim(P)$ nodal variables $\mathscr{N} = \{N_1, N_2, ..., N_k\}$ which define the degrees of freedom.

The nodal basis of $\mathscr{P}$ is denoted by $\{\varphi_1, \varphi_2, ..., \varphi_k\}$ and is dual to $\mathscr{N}$. This means that $N_\nu(\phi_\mu) = \delta_{\nu\mu}$, which is an important property for deriving the shape functions later.

We now turn to specifying the properties $K, \mathscr{P}$ and $\mathscr{N}$ for the finite elements we are using.

The Choice of K

Our simulations are run in a rectangular domain with straight boundaries so we can use one of the easiest choices for K: rectangular cuboids (see figure 2.1).

The Choice of $\mathscr{P}$

For a rectangular domain, the space of polynomials is usually denoted by $\mathscr{Q}$ instead of $\mathscr{P}$. $\mathscr{Q}_k(\mathbb{R}^n)$ denotes the space of polynomials of degree at most k in each variable. For three dimensions, it is

$$\mathscr{Q}_k(\mathbb{R}^3) = \left\{ \sum_{j=0}^{k} \alpha_j p_j(x) q_j(y) r_j(z) : p_j, q_j, r_j \text{ polynomials of degree} \leq k. \right\} \tag{2.40}$$

Since our domain Ω is perfectly rectangular, a choice of $k = 1$ leads to a sufficiently accurate approximation, so we use trilinear polynomials for our finite element discretization:

$$\mathscr{P} = \mathscr{Q}_1(\mathbb{R}^3) = \text{span}\{1, x, y, z, xy, xz, yz, xyz\}. \tag{2.41}$$

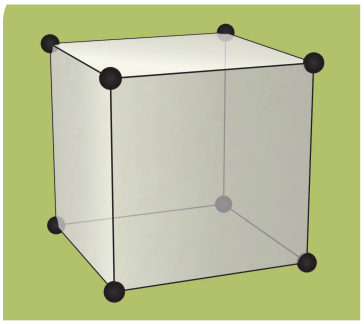


Figure 2.1: The element $\mathcal{Q}_1(\mathbb{R}^3)$ and its degrees of freedom. Taken from Arnold and Logg, (2014).

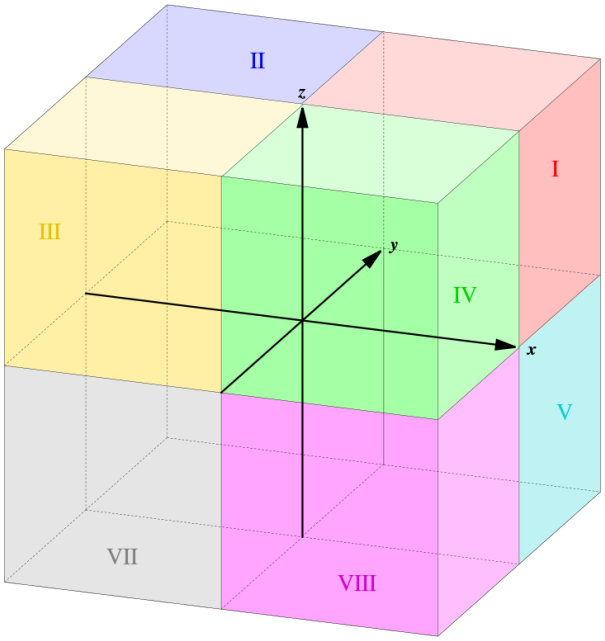


Figure 2.2: The subset Ω_8 of the domain Ω which shows eight elements around one grid point with coordinates (x_i, y_j, z_k) (taken from Wikipedia).

The Choice of $\mathcal{N}$

Having specified K and $\mathcal{P}$ we must now specify the nodes we use. Since $\dim(\mathcal{P}) = 8$ we need 8 nodes. A standard choice for Lagrangian degrees of freedom on $\mathcal{P} = \mathcal{Q}_1(\mathbb{R}^3)$ are the 8 corners of the cubicle (see figure 2.1).

2.2.3.1 Calculations of the Shape Functions

Next, we demonstrate how the shape functions are derived . Consider the subset $\Omega_8 \subset \Omega$ which consists of eight elements around the grid point μ with the coordinates (x_i, y_j, z_k). Figure 2.2 shows a representation of Ω_8 and the numbering of elements around the grid point μ.

Remember that the linear system to be solved is

$$A\boldsymbol{u} = \boldsymbol{f}, \tag{2.42}$$

i.e.,

$$\sum_{\nu=1}^{N_h} u_\nu a(\varphi_\nu, \varphi_\mu) = F(\varphi_\mu), \quad \mu = 1,2,...,N_h. \tag{2.43}$$

The bilinear form $a(\cdot,\cdot)$ and the linear functional $F(\cdot)$ are (in the homogeneous Dirichlet case) defined as

$$a(u,v) = \int_\Omega \kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x}) \nabla v(\boldsymbol{x}) \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) u(\boldsymbol{x}) v(\boldsymbol{x}) \mathrm{d}\Omega \tag{2.44}$$

and

$$F(v) = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \mathrm{d}\Omega, \tag{2.45}$$

so that (2.43) becomes

$$\sum_{\nu=1}^{N_h} u_\nu \left[\int_\Omega \kappa(\boldsymbol{x}) \nabla\varphi_\nu(\boldsymbol{x}) \nabla\varphi_\mu(\boldsymbol{x}) \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) \varphi_\nu(\boldsymbol{x}) \varphi_\mu(\boldsymbol{x}) \mathrm{d}\Omega \right] = \int_\Omega f(\boldsymbol{x}) \varphi_\mu(\boldsymbol{x}) \mathrm{d}\Omega, \quad \mu = 1,2,...,N_h. \tag{2.46}$$

The integrals are evaluated with the three–dimensional trapezoidal rule,

$$\int_\Omega f(x,y,z) \mathrm{d}x \mathrm{d}y \mathrm{d}z = \frac{f_1 + f_2 + f_3 + f_4 + f_5 + f_6 + f_7 + f_8}{8} h_x h_y h_z, \tag{2.47}$$

where f_μ is the value of $f(x,y,z)$ evaluated at node μ of the cubicle and the indices 1-8 indicate the node. See figure 2.3 for the numbering of the nodes. This quadrature rule requires us to calculate φ and $\nabla\varphi$ at each of the eight nodes of each element in Ω. We now derive a method to do that.

Consider the arbitrarily chosen inner grid point with index μ and coordinates (x_l, y_n, z_m). It is surrounded by eight elements $Q_1, Q_2, ..., Q_8$ (figure 2.2). Since the integrals in (2.46) are evaluated over all of Ω, each cubicle around point μ contributes to the integral. In order to calculate (2.46) we need explicit forms for the shape functions φ, which are the basis functions of the subspace V_h and their gradients $\nabla\varphi$. We have specified earlier that $\varphi \in \mathscr{P}$. A generic polynomial from our chosen $\mathscr{P}$ is of the form

$$\varphi(x,y,z) = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 z + \alpha_4 xy + \alpha_5 xz + \alpha_6 yz + \alpha_7 xyz. \tag{2.48}$$

The gradient of (2.48) is

$$\nabla\varphi(x,y,z) = \begin{pmatrix} \alpha_1 + \alpha_4 y + \alpha_5 z + \alpha_7 yz \\ \alpha_2 + \alpha_4 x + \alpha_6 z + \alpha_7 xz \\ \alpha_3 + \alpha_5 x + \alpha_6 y + \alpha_7 xy \end{pmatrix}. \tag{2.49}$$

The condition for shape functions is

$$\varphi_{l,m,n}(x_i, y_j, z_k) = \delta_{li}\delta_{mj}\delta_{nk} \tag{2.50}$$

where $\varphi_{l,m,n}$ is the shape function at the point (x_l, y_m, z_n) of the grid and (x_i, x_j, x_k) is the point on the grid where the contribution of this particular shape function is to be evaluated. This shows that only nodes which are points of the element itself have a contribution. All other contributions are zero due to the lack of support.

Let us now derive an algorithm to calculate the contributions of any shape function on any point in the grid. To this end, we choose an arbitrary element Q_e, where e stands for example. With our chosen degrees of freedom, Q_e looks like figure 2.3.

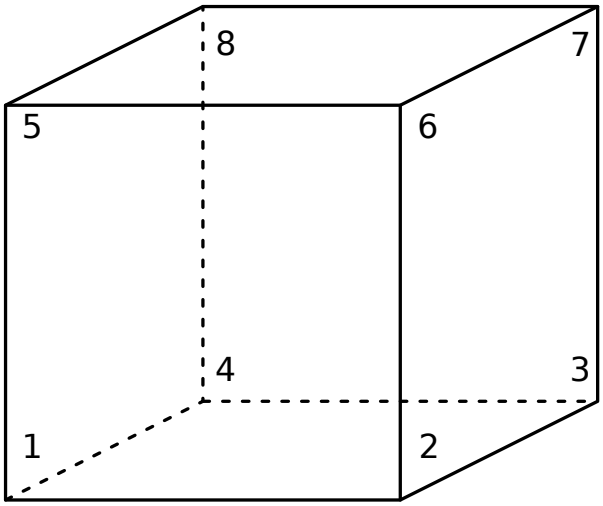


Figure 2.3: A typical element in our discretization of the domain. The numbers represent the Lagrangian degrees of freedom.

Every inner node in the grid is surrounded by eight of these elements. That means that we have to be able to calculate the shape function contributions for any combination of (l, m, n) and (i, j, k).

Let us now calculate the contributions of the shape functions to the node (x_i, y_j, z_k). The node is surrounded by the 8 elements $Q_1, Q_2, Q_3, Q_4, Q_5, Q_6, Q_7, Q_8$ (see figure 2.2). Each of these elements contributes to the value of the function at the node. The contribution of all the other elements is zero because of (2.50). So we now calculate the contributions for each cubicle, starting with Q_1, the cubicle with all positive coordinates. We demonstrate the calculation for this cubicle in detail. The calculations for the other cubicles have been done with Mathematica.

Cubicle $Q_1 = [x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}]$

We start by inserting the coordinates of each of the eighth nodes of the cubicle Q_1, i.e., (x_i, y_j, z_k), (x_{i+1}, y_j, z_k), (x_{i+1}, y_{j+1}, z_k), (x_i, y_{j+1}, z_k), (x_i, y_j, z_{k+1}), (x_{i+1}, y_j, z_{k+1}),

$(x_{i+1}, y_{j+1}, z_{k+1})$ and (x_i, y_{j+1}, z_{k+1}) into the general form of the shape function (2.48). We end up with the eight equations

$$\begin{aligned}\varphi_{l,m,n}(x_i, y_j, z_k) = \alpha_0 &+ \alpha_1 x_i + \alpha_2 y_j + \alpha_3 z_k + \alpha_4 x_i y_j \\ &+ \alpha_5 x_i z_k + \alpha_6 y_j z_k + \alpha_7 x_i y_j z_k,\end{aligned} \tag{2.51}$$

$$\begin{aligned}\varphi_{l,m,n}(x_{i+1}, y_j, z_k) = \alpha_0 &+ \alpha_1 x_{i+1} + \alpha_2 y_j + \alpha_3 z_k + \alpha_4 x_{i+1} y_j \\ &+ \alpha_5 x_{i+1} z_k + \alpha_6 y_j z_k + \alpha_7 x_{i+1} y_j z_k,\end{aligned} \tag{2.52}$$

$$\begin{aligned}\varphi_{l,m,n}(x_{i+1}, y_{j+1}, z_k) = \alpha_0 &+ \alpha_1 x_{i+1} + \alpha_2 y_{j+1} + \alpha_3 z_k + \alpha_4 x_{i+1} y_{j+1} \\ &+ \alpha_5 x_{i+1} z_k + \alpha_6 y_{j+1} z_k + \alpha_7 x_{i+1} y_{j+1} z_k,\end{aligned} \tag{2.53}$$

$$\begin{aligned}\varphi_{l,m,n}(x_i, y_{j+1}, z_k) = \alpha_0 &+ \alpha_1 x_i + \alpha_2 y_{j+1} + \alpha_3 z_k + \alpha_4 x_i y_{j+1} \\ &+ \alpha_5 x_i z_k + \alpha_6 y_{j+1} z_k + \alpha_7 x_i y_{j+1} z_k,\end{aligned} \tag{2.54}$$

$$\begin{aligned}\varphi_{l,m,n}(x_i, y_j, z_{k+1}) = \alpha_0 &+ \alpha_1 x_i + \alpha_2 y_j + \alpha_3 z_{k+1} + \alpha_4 x_i y_j \\ &+ \alpha_5 x_i z_{k+1} + \alpha_6 y_j z_{k+1} + \alpha_7 x_i y_j z_{k+1},\end{aligned} \tag{2.55}$$

$$\begin{aligned}\varphi_{l,m,n}(x_{i+1}, y_j, z_{k+1}) = \alpha_0 &+ \alpha_1 x_{i+1} + \alpha_2 y_j + \alpha_3 z_{k+1} + \alpha_4 x_{i+1} y_j \\ &+ \alpha_5 x_{i+1} z_{k+1} + \alpha_6 y_j z_{k+1} + \alpha_7 x_{i+1} y_j z_{k+1},\end{aligned} \tag{2.56}$$

$$\begin{aligned}\varphi_{l,m,n}(x_{i+1}, y_{j+1}, z_{k+1}) = \alpha_0 &+ \alpha_1 x_{i+1} + \alpha_2 y_{j+1} + \alpha_3 z_{k+1} + \alpha_4 x_{i+1} y_{j+1} \\ &+ \alpha_5 x_{i+1} z_{k+1} + \alpha_6 y_{j+1} z_{k+1} + \alpha_7 x_{i+1} y_{j+1} z_{k+1},\end{aligned} \tag{2.57}$$

$$\begin{aligned}\varphi_{l,m,n}(x_i, y_{j+1}, z_{k+1}) = \alpha_0 &+ \alpha_1 x_i + \alpha_2 y_{j+1} + \alpha_3 z_{k+1} + \alpha_4 x_i y_{j+1} \\ &+ \alpha_5 x_i z_{k+1} + \alpha_6 y_{j+1} z_{k+1} + \alpha_7 x_i y_{j+1} z_{k+1}.\end{aligned} \tag{2.58}$$

Now we choose (l, m, n) to be the coordinates of one of the 8 nodes and use the shape function condition (2.50) on each equation.

This results in eight linear system of equations, one for each set of (l, m, n). For $(l, m, n) = (i, j, k)$, which is the first node in cubicle Q_1, the resulting linear system is

$$M^{(1)}\boldsymbol{\alpha} = \boldsymbol{\delta}^{(1)} \tag{2.59}$$

with $M^{(1)} =$

$$\begin{bmatrix}
1 & ih_x & jh_y & kh_z & ih_x jh_y & ih_x kh_z & \dots \\
1 & (i+1)h_x & jh_y & kh_z & (i+1)h_x jh_y & (i+1)h_x kh_z & \dots \\
1 & (i+1)h_x & (j+1)h_y & kh_z & (i+1)h_x(j+1)h_y & (i+1)h_x kh_z & \dots \\
1 & ih_x & (j+1)h_y & kh_z & ih_x(j+1)h_y & ih_x kh_z & \dots \\
1 & ih_x & jh_y & (k+1)h_z & ih_x jh_y & ih_x(k+1)h_z & \dots \\
1 & (i+1)h_x & jh_y & (k+1)h_z & (i+1)h_x jh_y & (i+1)h_x(k+1)h_z & \dots \\
1 & (i+1)h_x & (j+1)h_y & (k+1)h_z & (i+1)h_x(j+1)h_y & (i+1)h_x(k+1)h_z & \dots \\
1 & ih_x & (j+1)h_y & (k+1)h_z & ih_x(j+1)h_y & ih_x(k+1)h_z & \dots
\end{bmatrix}$$

$$\begin{bmatrix} \dots & jh_y kh_z & ih_x jh_y kh_z \\ \dots & jh_y kh_z & (i+1)h_x jh_y kh_z \\ \dots & (j+1)h_y kh_z & (i+1)h_x (j+1)h_y kh_z \\ \dots & (j+1)h_y kh_z & ih_x (j+1)h_y kh_z \\ \dots & jh_y (k+1)h_z & ih_x jh_y (k+1)h_z \\ \dots & jh_y (k+1)h_z & (i+1)h_x jh_y (k+1)h_z \\ \dots & (j+1)h_y (k+1)h_z & (i+1)h_x (j+1)h_y (k+1)h_z \\ \dots & (j+1)h_y (k+1)h_z & ih_x (j+1)h_y (k+1)h_z \end{bmatrix}, \tag{2.60}$$

where the vector $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, ..., \alpha_7]^{\mathrm{T}}$ contains the unknown coefficients α_i and the vector $\boldsymbol{\delta}^{(1)} = [1,0,0,0,0,0,0,0]$ contains the shape function conditions (2.50) for the current node at (x_i, y_j, z_k). Solving the system gives us the coefficients $\alpha_i, i = 1, ..., 7$ for (2.49):

$$\begin{aligned} \alpha_0 &= (i+1)(j+1)(k+1), & \alpha_1 &= -\frac{(j+1)(k+1)}{h_x}, & \alpha_2 &= -\frac{(i+1)(k+1)}{h_y}, \\ \alpha_3 &= -\frac{(i+1)(j+1)}{h_z}, & \alpha_4 &= \frac{k+1}{h_x h_y}, & \alpha_5 &= \frac{j+1}{h_x h_z}, \\ \alpha_6 &= \frac{(i+1)}{h_y h_z}, & \alpha_7 &= -\frac{1}{h_x h_y h_z}. \end{aligned} \tag{2.61}$$

With these coefficients, the shape function (2.48) and its gradient (2.49) can be evaluated for the first element Q_1. The procedure is the same for every other element and has been done with Mathematica. The results can be seen in the following section where the integrals is evaluated.

2.2.4 Evaluation of the Integrals

Reminder: we have the following problem to solve:

$$\text{find } u_h \in V_h(\Omega): \quad a(u_h, v_h) = F(v_h) \quad \forall v_h \in V_h(\Omega) \tag{2.62}$$

or written in the Galerkin form:

$$\text{find } u_h \in V_h(\Omega): \quad \sum_{\nu=1}^{N_h} u_\nu a(\varphi_\nu, \varphi_\mu) = F(\varphi_\mu), \quad \mu = 1, 2, ..., N_h. \tag{2.63}$$

ν and μ denote points in the computational grid G. Having derived the shape functions, we can now evaluate the integrals.

2.2.4.1 Calculating the Bilinear Forms

The bilinear form $a(\cdot,\cdot)$ was defined in (2.8). It is the same for each boundary condition considered:

$$a: V \times V \to \mathbb{R}, \quad a(u,v) = \int_\Omega \kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x}) \nabla v(\boldsymbol{x}) \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) u(\boldsymbol{x}) v(\boldsymbol{x}) \mathrm{d}\Omega. \tag{2.64}$$

Following (2.63), it is to be calculated for every combination of ν and μ. Due to the limited support of the shape function the calculations are reduced substantially and we are left with the calculation of the contribution of neighboring points only. Looking at figure 2.2 shows us that each node (i,j,k) is surrounded by 26 other nodes. That means that we only have to calculate 27 bilinear forms per grid point instead of N_h ones per grid point. For the node μ with coordinates (x_i, x_j, y_k), the forms to calculate are

$$\begin{array}{lll}
a(\varphi_{i-1,j+1,k+1}, \varphi_{i,j,k}), & a(\varphi_{i,j+1,k+1}, \varphi_{i,j,k}), & a(\varphi_{i+1,j+1,k+1}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j,k+1}, \varphi_{i,j,k}), & a(\varphi_{i,j,k+1}, \varphi_{i,j,k}), & a(\varphi_{i+1,j,k+1}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j-1,k+1}, \varphi_{i,j,k}), & a(\varphi_{i,j-1,k+1}, \varphi_{i,j,k}), & a(\varphi_{i+1,j-1,k+1}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j+1,k}, \varphi_{i,j,k}), & a(\varphi_{i,j+1,k}, \varphi_{i,j,k}), & a(\varphi_{i+1,j+1,k}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j,k}, \varphi_{i,j,k}), & a(\varphi_{i,j,k}, \varphi_{i,j,k}), & a(\varphi_{i+1,j,k}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j-1,k}, \varphi_{i,j,k}), & a(\varphi_{i,j-1,k}, \varphi_{i,j,k}), & a(\varphi_{i+1,j-1,k}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j+1,k-1}, \varphi_{i,j,k}), & a(\varphi_{i,j+1,k-1}, \varphi_{i,j,k}), & a(\varphi_{i+1,j+1,k-1}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j,k-1}, \varphi_{i,j,k}), & a(\varphi_{i,j,k-1}, \varphi_{i,j,k}), & a(\varphi_{i+1,j,k-1}, \varphi_{i,j,k}), \\
a(\varphi_{i-1,j-1,k-1}, \varphi_{i,j,k}), & a(\varphi_{i,j-1,k-1}, \varphi_{i,j,k}), & a(\varphi_{i+1,j-1,k-1}, \varphi_{i,j,k}).
\end{array} \tag{2.65}$$

One advantage of a triangulation by cubicles instead of tetrahedrons is that all bilinear forms of nodes which are diagonal to each other are zero. This reduces above 27 forms to merely 7 forms (omitting dependencies for readability):

$$a(\varphi_{i,j,k}, \varphi_{i,j,k}) = \int_\Omega \kappa \nabla\varphi_{i,j,k} \nabla\varphi_{i,j,k} \,\mathrm{d}\Omega + \int_\Omega \xi \varphi_{i,j,k} \varphi_{i,j,k} \,\mathrm{d}\Omega, \tag{2.66}$$

$$a(\varphi_{i+1,j,k}, \varphi_{i,j,k}) = \int_\Omega \kappa \nabla\varphi_{i+1,j,k} \nabla\varphi_{i,j,k} \,\mathrm{d}\Omega + \int_\Omega \xi \varphi_{i+1,j,k} \varphi_{i,j,k} \,\mathrm{d}\Omega, \tag{2.67}$$

$$a(\varphi_{i,j+1,k}, \varphi_{i,j,k}) = \int_\Omega \kappa \nabla\varphi_{i,j+1,k} \nabla\varphi_{i,j,k} \,\mathrm{d}\Omega + \int_\Omega \xi \varphi_{i,j+1,k} \varphi_{i,j,k} \,\mathrm{d}\Omega, \tag{2.68}$$

$$a(\varphi_{i-1,j,k}, \varphi_{i,j,k}) = \int_\Omega \kappa \nabla\varphi_{i-1,j,k} \nabla\varphi_{i,j,k} \,\mathrm{d}\Omega + \int_\Omega \xi \varphi_{i-1,j,k} \varphi_{i,j,k} \,\mathrm{d}\Omega, \tag{2.69}$$

$$a(\varphi_{i,j-1,k}, \varphi_{i,j,k}) = \int_\Omega \kappa \nabla\varphi_{i,j-1,k} \nabla\varphi_{i,j,k} \,\mathrm{d}\Omega + \int_\Omega \xi \varphi_{i,j-1,k} \varphi_{i,j,k} \,\mathrm{d}\Omega, \tag{2.70}$$

$$a(\varphi_{i,j,k+1}, \varphi_{i,j,k}) = \int_\Omega \kappa \nabla\varphi_{i,j,k+1} \nabla\varphi_{i,j,k} \,\mathrm{d}\Omega + \int_\Omega \xi \varphi_{i,j,k+1} \varphi_{i,j,k} \,\mathrm{d}\Omega, \tag{2.71}$$

$$a(\varphi_{i,j,k-1}, \varphi_{i,j,k}) = \int_\Omega \kappa \nabla\varphi_{i,j,k-1} \nabla\varphi_{i,j,k} \,\mathrm{d}\Omega + \int_\Omega \xi \varphi_{i,j,k-1} \varphi_{i,j,k} \,\mathrm{d}\Omega. \tag{2.72}$$

The domain of integration Ω also reduces due to the lack of support of the shape functions. It consists of only those cubicles which contain both nodes in question. Those are:

$$\text{for } a(\varphi_{i,j,k}, \varphi_{i,j,k}): \quad Q_1 \cup Q_2 \cup Q_3 \cup Q_4 \cup Q_5 \cup Q_6 \cup Q_7 \cup Q_8, \tag{2.73}$$

$$\text{for } a(\varphi_{i+1,j,k},\varphi_{i,j,k}): \quad Q_1 \cup Q_4 \cup Q_5 \cup Q_8, \tag{2.74}$$

$$\text{for } a(\varphi_{i,j+1,k},\varphi_{i,j,k}): \quad Q_1 \cup Q_2 \cup Q_5 \cup Q_6, \tag{2.75}$$

$$\text{for } a(\varphi_{i-1,j,k},\varphi_{i,j,k}): \quad Q_2 \cup Q_3 \cup Q_6 \cup Q_7, \tag{2.76}$$

$$\text{for } a(\varphi_{i,j-1,k},\varphi_{i,j,k}): \quad Q_3 \cup Q_4 \cup Q_7 \cup Q_8, \tag{2.77}$$

$$\text{for } a(\varphi_{i,j,k+1},\varphi_{i,j,k}): \quad Q_1 \cup Q_2 \cup Q_3 \cup Q_4, \tag{2.78}$$

$$\text{for } a(\varphi_{i,j,k-1},\varphi_{i,j,k}): \quad Q_5 \cup Q_6 \cup Q_7 \cup Q_8 \tag{2.79}$$

with

$$\begin{aligned}
Q_1 &= [x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}], \\
Q_2 &= [x_{i-1}, x_i] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}], \\
Q_3 &= [x_{i-1}, x_i] \times [y_{j-1}, j_j] \times [z_k, z_{k+1}], \\
Q_4 &= [x_i, x_{i+1}] \times [y_{j-1}, y_j] \times [z_k, z_{k+1}], \\
Q_5 &= [x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k-1}], \\
Q_6 &= [x_{i-1}, x_i] \times [y_j, y_{j+1}] \times [z_k, z_{k-1}], \\
Q_7 &= [x_{i-1}, x_i] \times [y_{j-1}, j_j] \times [z_k, z_{k-1}], \\
Q_8 &= [x_i, x_{i+1}] \times [y_{j-1}, y_j] \times [z_k, z_{k-1}].
\end{aligned} \tag{2.80}$$

Calculation of the Stiffness Matrix

For each bilinear form we calculate the stiffness matrix and the mass matrix separately. To evaluate the integrals we use the trapezoidal rule (2.47). The first integral in the forms (2.66)-(2.72) gives the entries for the stiffness matrix S. The 7 non-empty entries are

$$S_{ijk,ijk} = \sum_{\xi=1}^{8} \iiint_{Q_\xi} \kappa(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k)}(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k)}(x,y,z) \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \tag{2.81}$$

$$S_{i+1jk,ijk} = \sum_{\xi=1,4,5,8} \iiint_{Q_\xi} \kappa(x,y,z) \nabla \varphi^{(\xi)}_{(i+1,j,k)}(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k)}(x,y,z) \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \tag{2.82}$$

$$S_{ij+1k,ijk} = \sum_{\xi=1,2,5,6} \iiint_{Q_\xi} \kappa(x,y,z) \nabla \varphi^{(\xi)}_{(i,j+1,k)}(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k)}(x,y,z) \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \tag{2.83}$$

$$S_{i-1jk,ijk} = \sum_{\xi=2,3,6,7} \iiint_{Q_\xi} \kappa(x,y,z) \nabla \varphi^{(\xi)}_{(i-1,j,k)}(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k)}(x,y,z) \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \tag{2.84}$$

$$S_{ij-1k,ijk} = \sum_{\xi=3,4,7,8} \iiint_{Q_\xi} \kappa(x,y,z) \nabla \varphi^{(\xi)}_{(i,j-1,k)}(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k)}(x,y,z) \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \tag{2.85}$$

$$S_{ijk+1,ijk} = \sum_{\xi=1,2,3,4} \iiint_{Q_\xi} \kappa(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k+1)}(x,y,z) \nabla \varphi^{(\xi)}_{(i,j,k)}(x,y,z) \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z, \tag{2.86}$$

$$S_{ijk-1,ijk} = \sum_{\xi=5,6,7,8} \iiint_{Q_\xi} \kappa(x,y,z) \nabla\varphi^{(\xi)}_{(i,j,k-1)}(x,y,z) \nabla\varphi^{(\xi)}_{(i,j,k)}(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z. \tag{2.87}$$

Evaluation of those stiffness matrix entries by the trapezoidal rule (2.47) with Mathematica gives

$$\begin{aligned} S_{ijk,ijk} =& \frac{h_y h_z}{2h_x}(\kappa_{i-1,j,k} + \kappa_{i+1,j,k}) + \frac{h_x h_z}{2h_y}(\kappa_{i,j-1,k} + \kappa_{i,j+1,k}) \\ &+ \frac{h_x h_y}{2h_z}(\kappa_{i,j,k-1} + \kappa_{i,j,k+1}) + \left(\frac{h_y h_z}{h_x} + \frac{h_x h_z}{h_y} + \frac{h_x h_y}{h_z}\right)\kappa_{i,j,k}, \end{aligned} \tag{2.88}$$

$$S_{i+1jk,ijk} = -\frac{h_y h_z\left(\kappa_{i,j,k} + \kappa_{i+1,j,k}\right)}{2h_x}, \tag{2.89}$$

$$S_{i-1jk,ijk} = -\frac{h_y h_z\left(\kappa_{i-1,j,k} + \kappa_{i,j,k}\right)}{2h_x}, \tag{2.90}$$

$$S_{ij+1k,ijk} = -\frac{h_x h_z\left(\kappa_{i,j,k} + \kappa_{i,j+1,k}\right)}{2h_y}, \tag{2.91}$$

$$S_{ij-1k,ijk} = -\frac{h_x h_z\left(\kappa_{i,j-1,k} + \kappa_{i,j,k}\right)}{2h_y}, \tag{2.92}$$

$$S_{ijk+1,ijk} = -\frac{h_x h_y\left(\kappa_{i,j,k} + \kappa_{i,j,k+1}\right)}{2h_z}, \tag{2.93}$$

$$S_{ijk-1,ijk} = -\frac{h_x h_y\left(\kappa_{i,j,k-1} + \kappa_{i,j,k}\right)}{2h_z}. \tag{2.94}$$

Calculating the Mass Matrix M

Calculating the mass matrix involves considerable less effort than calculating the stiffness matrix. The reason is that the shape functions themselves are calculated (in contrast to their gradients, as has been the case in the calculation of the stiffness matrix) at the corners of the cubicles. They are defined to be zero there, however. Only the shape function at (x_i, y_j, z_k) is nonzero. We use the entry $M_{i+1jk,ijk}$ to illustrate this.

The second integral in the forms (2.66)-(2.72) gives the entries for the mass matrix M:

$$M_{i+1jk,ijk} = \sum_{\xi=1,4,5,8} \iiint_{Q_\xi} \xi(x,y,z)\varphi^{(\xi)}_{i+1,j,k}(x,y,z)\varphi^{(\xi)}_{i,j,k}(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z \tag{2.95}$$

Inserting the shape function condition (2.50) yields

$$M_{i+1jk,ijk} = \sum_{\xi=1,4,5,8} \iiint_{Q_\xi} \xi(x,y,z)\delta_{i+1,x}\delta_{j,y}\delta_{k,z}\delta_{i,x}\delta_{j,y}\delta_{k,z}\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z. \tag{2.96}$$

This is always zero because $\delta_{i+1,x}\delta_{i,x}$ is zero unless $i+1 = i$ which is obviously never the case. That means that all terms of the mass matrix are zero with the exception of $M_{ijk,ijk}$, which is

$$M_{ijk,ijk} = \sum_{\xi=1}^{8} \iiint_{Q_\xi} \xi(x,y,z)\varphi^{(\xi)}_{i,j,k}(x,y,z)\varphi^{(\xi)}_{i,j,k}(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z. \tag{2.97}$$

This is a sum of eight volume integrals. We illustrate the calculation for cubicle Q_1. The integral for Q_1 is

$$\iiint_{Q_1} \xi(x,y,z)\varphi^{(1)}_{i,j,k}(x,y,z)\varphi^{(1)}_{i,j,k}(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z. \tag{2.98}$$

Before using the trapezoidal rule we need to specify the 8 nodes of cubicle Q_1 which are

$$\begin{aligned}(x_i,y_j,z_k),(x_{i+1},y_j,z_k),(x_{i+1},y_{j+1},z_k),(x_i,y_{j+1},z_k),\\ (x_i,y_j,z_{k+1}),(x_{i+1},y_j,z_{k+1}),(x_{i+1},y_{j+1},z_{k+1}),(x_i,y_{j+1},z_{k+1}).\end{aligned} \tag{2.99}$$

Using the trapezoidal rule for cubicles (2.47) now gives

$$\iiint_{Q_1} \xi(x,y,z)\varphi^{(1)}_{i,j,k}(x,y,z)\varphi^{(1)}_{i,j,k}(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = \frac{h_x h_y h_z}{8}\,q_1 \tag{2.100}$$

with

$$\begin{aligned}q_1 =&\xi(i,j,k)\varphi^{(1)}_{i,j,k}(i,j,k)\varphi^{(1)}_{i,j,k}(i,j,k)\\ &+\xi(i+1,j,k)\varphi^{(1)}_{i,j,k}(i+1,j,k)\varphi^{(1)}_{i,j,k}(i+1,j,k)\\ &+\xi(i+1,j+1,k)\varphi^{(1)}_{i,j,k}(i+1,j+1,k)\varphi^{(1)}_{i,j,k}(i,j,k)\\ &+\xi(i,j+1,k)\varphi^{(1)}_{i,j,k}(i,j+1,k)\varphi^{(1)}_{i,j,k}(i,j+1,k)\\ &+\xi(i,j,k+1)\varphi^{(1)}_{i,j,k}(i,j,k+1)\varphi^{(1)}_{i,j,k}(i,j,k+1)\\ &+\xi(i+1,j,k+1)\varphi^{(1)}_{i,j,k}(i+1,j,k+1)\varphi^{(1)}_{i,j,k}(i+1,j,k+1)\\ &+\xi(i+1,j+1,k+1)\varphi^{(1)}_{i,j,k}(i+1,j+1,k+1)\varphi^{(1)}_{i,j,k}(i+1,j+1,k+1)\\ &+\xi(i,j+1,k+1)\varphi^{(1)}_{i,j,k}(i,j+1,k+1)\varphi^{(1)}_{i,j,k}(i,j+1,k+1).\end{aligned} \tag{2.101}$$

Using the shape function condition again, q_1 reduces to

$$q_1 = \xi(i,j,k), \tag{2.102}$$

so (2.100) becomes

$$\iiint_{Q_1} \xi(x,y,z)\varphi^{(1)}_{i,j,k}(x,y,z)\varphi^{(1)}_{i,j,k}(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = \frac{h_x h_y h_z}{8}\,\xi(i,j,k). \tag{2.103}$$

The calculation for the other cubicles is similar. Summation over all eight elements yields the mass matrix entry $M_{ijk,ijk}$ which is

$$M_{ijk,ijk} = \sum_{\mu=1}^{8} \iiint_{Q_\mu} \xi(x,y,z)\varphi^{(\mu)}_{i,j,k}(x,y,z)\varphi^{(\mu)}_{i,j,k}(x,y,z)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}z = h_x h_y h_z\,\xi(i,j,k). \tag{2.104}$$

Result: The Bilinear Forms

After having evaluated stiffness and mass matrix entries for node (i,j,k) we can now explicitly write out the seven non-vanishing bilinear forms:

$$\begin{aligned} a(ijk,ijk) = &\frac{h_y h_z}{2h_x}(\kappa_{i-1,j,k}+\kappa_{i+1,j,k}) + \frac{h_x h_z}{2h_y}(\kappa_{i,j-1,k}+\kappa_{i,j+1,k}) \\ &+ \frac{h_x h_y}{2h_z}(\kappa_{i,j,k-1}+\kappa_{i,j,k+1}) + \left(\frac{h_y h_z}{h_x}+\frac{h_x h_z}{h_y}+\frac{h_x h_y}{h_z}\right)\kappa_{i,j,k} \\ &+ h_x h_y h_z \xi_{i,j,k}, \end{aligned} \tag{2.105}$$

$$a(i+1jk,ijk) = -\frac{h_y h_z\left(\kappa_{i,j,k}+\kappa_{i+1,j,k}\right)}{2h_x}, \tag{2.106}$$

$$a(i-1jk,ijk) = -\frac{h_y h_z\left(\kappa_{i-1,j,k}+\kappa_{i,j,k}\right)}{2h_x}, \tag{2.107}$$

$$a(ij+1k,ijk) = -\frac{h_x h_z\left(\kappa_{i,j,k}+\kappa_{i,j+1,k}\right)}{2h_y}, \tag{2.108}$$

$$a(ij-1k,ijk) = -\frac{h_x h_z\left(\kappa_{i,j-1,k}+\kappa_{i,j,k}\right)}{2h_y}, \tag{2.109}$$

$$a(ijk+1,ijk) = -\frac{h_x h_y\left(\kappa_{i,j,k}+\kappa_{i,j,k+1}\right)}{2h_z}, \tag{2.110}$$

$$a(ijk-1,ijk) = -\frac{h_x h_y\left(\kappa_{i,j,k-1}+\kappa_{i,j,k}\right)}{2h_z}. \tag{2.111}$$

2.2.4.2 The Calculation of the Right–Hand Side $F(v)$

The equation to solve, (2.43), is repeated for the reader's convenience:

$$\sum_{j=1}^{N_h} u_j\, a(\varphi_\nu,\varphi_\mu) = F(\varphi_\mu), \quad i = 1,2,...,N_h. \tag{2.112}$$

We have calculated an explicit formulation of the bilinear form $a(\cdot,\cdot)$ in the previous section. Now we turn to the calculation of the right–hand side $F(\cdot)$ which has been defined, depending on the boundary conditions, in section 2.2.1 as

$$F : \mathrm{H}_0^1 \to \mathbb{R}, \quad F(v) = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x})\,\mathrm{d}\Omega \tag{2.113}$$

for homogeneous Dirichlet boundary conditions, as

$$F : \mathrm{H}_{\Gamma_D}^1 \to \mathbb{R}, \quad F(v) = -a(u_{g_D}, v) + \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x})\,\mathrm{d}\Omega \tag{2.114}$$

for hon-homogeneous Dirichlet boundary conditions and as

$$F : \mathrm{H}^1 \to \mathbb{R}, \quad F(v) = \int_{\Gamma_N} \kappa(\boldsymbol{x}) g_N v(\boldsymbol{x})\,\mathrm{d}\Gamma + \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x})\,\mathrm{d}\Omega \tag{2.115}$$

for Neumann boundary conditions.

Using the Galerkin approximation and inserting φ_μ for v gives us the following three terms to calculate: $\int_\Omega f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega$, $-a(u_{g_D},\varphi_\mu)$ and $\int_{\Gamma_N} \kappa(\boldsymbol{x}) g_N \varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Gamma$. It is important for the calculation to know if $\mu \in \partial\Omega$. We begin with the calculation of $\int_\Omega f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega$ for the case that μ is not situated on the boundary.

The Domain Contributions of the Right–Hand–Side

For the inner grid points, no matter the boundary conditions, $F(\cdot)$ reduces to

$$F(\varphi_\mu) = \int_\Omega f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega, \quad \mu = 1,2,...,N_{\text{inner}}. \tag{2.116}$$

N_{inner} represents the number of grid points which are not situated on the domain boundary. The calculation of F for the grid point with coordinates (x_i, y_j, z_k) is straightforward and similar to the calculation of the mass matrix M. First, the domain of integration Ω is reduced due to the lack of support by the shape functions: only the cubicles containing node (i,j,k) have non-zero contributions, so

$$\Omega = Q_1 \cup Q_2 \cup Q_3 \cup Q_4 \cup Q_5 \cup Q_6 \cup Q_7 \cup Q_8, \tag{2.117}$$

with the same definitions of Q_ξ, $\xi = 1,..,8$, as in (2.80). Thus, (2.116) becomes

$$F(\varphi_\mu) = \sum_{\xi=1}^{8} \int_{Q_\xi} f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega, \quad \mu = 1,2,...,N_h. \tag{2.118}$$

After doing the calculation (in the same fashion as was done for the mass matrix before) one sees that, as in the case of the mass matrix, each cubicle contributes

$$q_{\text{partial}} = \frac{h_x h_y h_z}{8} f(i,j,k) \tag{2.119}$$

to the full $F(\varphi_\mu)$. There are eight cubicles so the overall value of F at grid point $(x_i, y_j, z_k) \notin \partial\Omega$ is

$$F(\varphi_{ijk}) = h_x h_y h_z f(i,j,k). \tag{2.120}$$

2.2.4.3 The Boundary Contributions

On the boundary the calculation of $F(\cdot)$ is a little more involved because, depending on the kind of boundary condition used, the term $a(u_{g_D},\varphi_\mu)$ or $\int_{\Gamma_N} \kappa(\boldsymbol{x}) g_N \varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Gamma$ enters the equation. Also, the support at the boundaries is only half of what it is at the inner points which changes the value of $\int_\Omega f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega$.

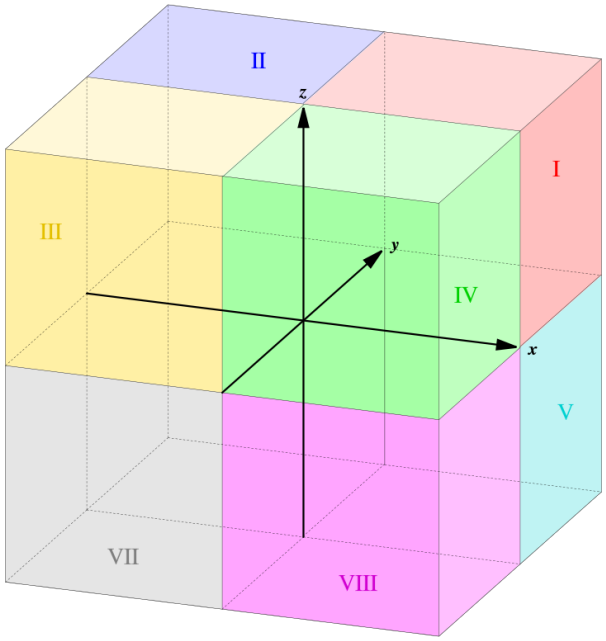


Figure 2.4: The domain of integration (taken from Wikipedia).

Calculation of $\int_\Omega f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\mathrm{d}\Omega$ on the boundary.

This is very similar to the calculation in the domain, only that the support, and thus the overall contribution, is halved. We start with

$$F_f(\varphi_\mu) = \int_\Omega f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\mathrm{d}\Omega, \quad \mu = 1,2,...,N_{\text{outer}}, \tag{2.121}$$

where $\mu \in \partial\Omega$ and N_{outer} stands for the number of grid points on the boundary $\partial\Omega$. Looking at figure 2.4 we imagine the yz-plane at $x = 0$ to be the lower boundary which then looks like figure 2.5. We immediately see that the area of integration is reduced from what it was in the calculation of the inner domain,

$$\Omega_{\text{int}} = Q_1 \cup Q_2 \cup Q_3 \cup Q_4 \cup Q_5 \cup Q_6 \cup Q_7 \cup Q_8, \tag{2.122}$$

to

$$\Omega_{\text{low\_bnd}} = Q_1 \cup Q_4 \cup Q_5 \cup Q_8 \tag{2.123}$$

for the lower x-boundary and

$$\Omega_{\text{up\_bnd}} = Q_2 \cup Q_3 \cup Q_6 \cup Q_7 \tag{2.124}$$

for the upper x-boundary. Thus, we arrive at

$$F_f(\varphi_\mu) = \sum_{\xi=1,4,5,8} \int_{Q_\xi} f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\mathrm{d}\Omega, \quad \mu \in \text{LBIndex} \tag{2.125}$$

for the lower boundary and

$$F_f(\varphi_\mu) = \sum_{\xi=2,3,6,7} \int_{Q_\xi} f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega, \quad \mu \in \mathrm{UBIndex} \tag{2.126}$$

for the upper boundary, where LB- and UBIndex are the index sets for lower and upper boundary, respectively. Again, each cubicle contributes

$$q_{\text{partial}} = \frac{h_x h_y h_z}{8} f(i,j,k) \tag{2.127}$$

to $F_f(\varphi_\mu)$. There are only four cubicles now, so the overall value of F_f at grid point $(x_i, y_j, z_k) \in \partial\Omega$ is

$$F_f(\varphi_{ijk}) = \frac{1}{2} h_x h_y h_z f(i,j,k) \tag{2.128}$$

at the lower and upper x-boundary.

2.2.4.4 Calculation of the Neumann Boundary Term

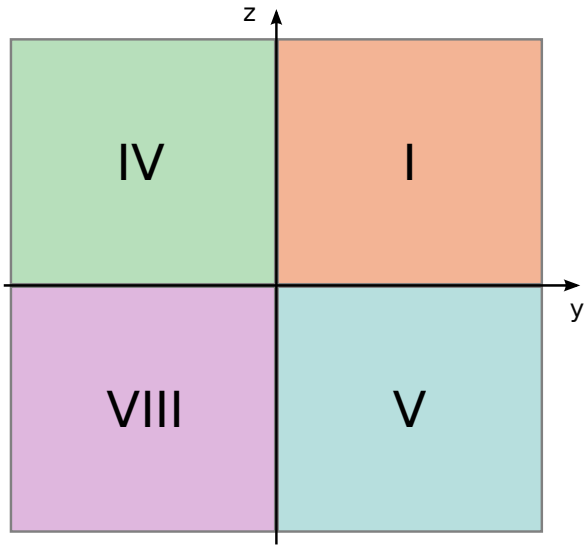


Figure 2.5: The lower boundary yz–plane.

We turn to the calculation of the Neumann boundary term

$$F_{\text{Neum}} = \int_{\Gamma_N} \kappa(\boldsymbol{x}) g_N \varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Gamma. \tag{2.129}$$

This is a surface integral over the domain shown in figure 2.5 so the quadrature rule changes accordingly. We use the trapezoidal rule in two dimensions for the evaluation of the integral.

First, as before, the lack of support reduces the domain Γ_N to the four squares in figure 2.5, T_1, T_4, T_5, T_8:

$$F_{\text{Neum}} = \int_{\Gamma_N} \kappa(\boldsymbol{x}) g_N \varphi_\mu(\boldsymbol{x}) \mathrm{d}\Gamma = \sum_{\xi=1,4,5,8} \int_{T_\xi} \kappa(\boldsymbol{x}) g_N \varphi_\mu(\boldsymbol{x}) \mathrm{d}\Gamma \tag{2.130}$$

Using the points of the square corners as limits we obtain

$$\begin{aligned} F_{\text{Neum}} = &\int_{z_k}^{z_{k+1}} \mathrm{d}z \int_{y_j}^{y_{j+1}} \mathrm{d}y \, \kappa(x_i,y,z) g_N(y,z) \varphi_{ijk}(x_i,y,z) \\ &+ \int_{z_k}^{z_{k+1}} \mathrm{d}z \int_{y_{j-1}}^{y_j} \mathrm{d}y \, \kappa(x_i,y,z) g_N(y,z) \varphi_{ijk}(x_i,y,z) \\ &+ \int_{z_{k-1}}^{z_k} \mathrm{d}z \int_{y_j}^{y_{j+1}} \mathrm{d}y \, \kappa(x_i,y,z) g_N(y,z) \varphi_{ijk}(x_i,y,z) \\ &+ \int_{z_{k-1}}^{z_k} \mathrm{d}z \int_{y_{j-1}}^{y_j} \mathrm{d}y \, \kappa(x_i,y,z) g_N(y,z) \varphi_{ijk}(x_i,y,z) \end{aligned} \tag{2.131}$$

where x_i is situated on the lower Neumann boundary. Using the trapezoid rule gives

$$\begin{aligned} F_{\text{Neum}} &= \frac{h_y h_z}{4} \kappa(x_i,y_j,z_k) g_N(y_j,z_k) \cdot 4 \\ &= h_y h_z \, \kappa(x_i,y_j,z_k) g_N(y_j,z_k) \end{aligned} \tag{2.132}$$

2.2.4.5 Overall Contribution at Neumann boundaries

So the overall value of $F(\cdot)$ at the Neumann boundary node with coordinates (x_i, y_j, z_k) in the yz-plane is

$$F(\varphi_{ijk}) = \frac{1}{2} h_x h_y h_z f(i,j,k) + h_y h_z \kappa(i,j,k) g_N(j,k). \tag{2.133}$$

2.2.4.6 Contribution at Non–homogeneous Dirichlet Boundaries

We turn to the calculation of the non–homogeneous Dirichlet boundary term

$$F_{\text{Dir}} = a(u_{g_D}, v), \tag{2.134}$$

which is, using the definition of the bilinear form, (2.18),

$$a(u_{g_D}, v) = \int_\Omega \kappa(\boldsymbol{x}) \nabla u_{g_D}(\boldsymbol{x}) \nabla v(\boldsymbol{x}) \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) u_{g_D}(\boldsymbol{x}) v(\boldsymbol{x}) \mathrm{d}\Omega. \tag{2.135}$$

By using the Galerkin approximation one obtains

$$a(u_{g_D}, v) = \sum_{\nu=1}^{N_o} u_{g_D,\nu} \left[\int_\Omega \kappa(\boldsymbol{x}) \nabla \varphi_\nu(\boldsymbol{x}) \nabla \varphi_\mu(\boldsymbol{x}) \mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x}) \varphi_\nu(\boldsymbol{x}) \varphi_\mu(\boldsymbol{x}) \mathrm{d}\Omega \right] \quad \mu = 1, \ldots, N_o. \tag{2.136}$$

Now, u_{g_D} is nonzero only at the boundaries, so all terms with indices ν and $\mu \notin \partial\Omega$ vanish. The terms in parenthesis are essentially the same as have been calculated in (2.66)-(2.72) and we end up with

$$F_{\text{Dirichlet}}(x_i, y_j, z_k) = f_{i,j,k} - a(\varphi_{i,j,k}, \varphi_{i-1,j,k})\, u_{g_D,(i,j,k)} \tag{2.137}$$

for the lower x–boundary and

$$F_{\text{Dirichlet}}(x_i, y_j, z_k) = f_{i,j,k} - a(\varphi_{i,j,k}, \varphi_{i+1,j,k})\, u_{g_D,(i,j,k)} \tag{2.138}$$

for the upper x–boundary.

2.2.5 Interlude – The Data Structure in ANTARES

To store the evaluated integrals in ANTARES we make use of the stencil notation (see Wesseling, 1992): Let $A : \mathbb{R}^d \to \mathbb{R}^d$ be a linear operator. Then, using stencil notation, $A\boldsymbol{u}$ can be represented by

$$(A\boldsymbol{u})_i = \sum_{j \in \mathbb{Z}^d} A(i,j) u_{i+j}, \quad i \in G. \tag{2.139}$$

The subscript $i = (i_1, i_2, ..., i_d)$ identifies a point in the computational grid G which is defined by

$$G = \{x \in \mathbb{R}^d : x = ih, i = (i_1, i_2, ..., i_d), h = (h_1, h_2, .., h_d), \\ i_\alpha = 0, 1, 2, ..., n_\alpha, h_\alpha = 1/n_\alpha, \alpha = 1, 2, ..., d\}. \tag{2.140}$$

Note: i is not the index for the x-variable here but a d-dimensional vector. The set S_A, defined by

$$S_A = \{j \in \mathbb{Z}^d : \exists i \in G \text{ with } A(i,j) \neq 0\}, \tag{2.141}$$

is called the structure of A. The set of values $A(i,j)$ with $j \in S_A$ is called the stencil of A at grid point i. It can be visualized by an array of values $[A]_i$ in which the values of $A(i,j)$ are given. In our case, 3D, the stencil looks like this:

$$[A]_i^{(-1)} = \begin{bmatrix} A(i,-e_1+e_2-e_3) & A(i,e_2-e_3) & A(i,e_1+e_2-e_3) \\ A(i,-e_1-e_3) & A(i,-e_3) & A(i,e_1-e_3) \\ A(i,-e_1-e_2-e_3) & A(i,-e_2-e_3) & A(i,e_1-e_2-e_3) \end{bmatrix}, \tag{2.142}$$

$$[A]_i^{(0)} = \begin{bmatrix} A(i,-e_1+e_2) & A(i,e_2) & A(i,e_1+e_2) \\ A(i,-e_1) & A(i,0) & A(i,e_1) \\ A(i,-e_1-e_2) & A(i,-e_2) & A(i,e_1-e_2) \end{bmatrix}, \tag{2.143}$$

$$[A]_i^{(1)} = \begin{bmatrix} A(i,-e_1+e_2+e_3) & A(i,e_2+e_3) & A(i,e_1+e_2+e_3) \\ A(i,-e_1+e_3) & A(i,+e_3) & A(i,e_1+e_3) \\ A(i,-e_1-e_2+e_3) & A(i,-e_2+e_3) & A(i,e_1-e_2+e_3) \end{bmatrix} \tag{2.144}$$

with

$$e_1 = (1,0,0), \quad e_2 = (0,1,0), \quad e_3 = (0,0,1). \tag{2.145}$$

In the FEM case, the stencil entries correspond to the following bilinear forms:

$$[A]^{(-1)}_{i,j,k} = \begin{bmatrix} a(\varphi_{i,j,k},\varphi_{i-1,j+1,k-1}) & a(\varphi_{i,j,k},\varphi_{i,j+1,k-1}) & a(\varphi_{i,j,k},\varphi_{i+1,j+1,k-1}) \\ a(\varphi_{i,j,k},\varphi_{i-1,j,k-1}) & a(\varphi_{i,j,k},\varphi_{i,j,k-1}) & a(\varphi_{i,j,k},\varphi_{i+1,j,k-1}) \\ a(\varphi_{i,j,k},\varphi_{i-1,j-1,k-1}) & a(\varphi_{i,j,k},\varphi_{i,j-1,k-1}) & a(\varphi_{i,j,k},\varphi_{i+1,j-1,k-1}) \end{bmatrix}, \quad (2.146)$$

$$[A]^{(0)}_{i,j,k} = \begin{bmatrix} a(\varphi_{i,j,k},\varphi_{i-1,j+1,k}) & a(\varphi_{i,j,k},\varphi_{i,j+1,k}) & a(\varphi_{i,j,k},\varphi_{i+1,j+1,k}) \\ a(\varphi_{i,j,k},\varphi_{i-1,j,k}) & a(\varphi_{i,j,k},\varphi_{i,j,k}) & a(\varphi_{i,j,k},\varphi_{i+1,j,k}) \\ a(\varphi_{i,j,k},\varphi_{i-1,j-1,k}) & a(\varphi_{i,j,k},\varphi_{i,j-1,k}) & a(\varphi_{i,j,k},\varphi_{i+1,j-1,k}) \end{bmatrix}, \quad (2.147)$$

$$[A]^{(1)}_{i,j,k} = \begin{bmatrix} a(\varphi_{i,j,k},\varphi_{i-1,j+1,k+1}) & a(\varphi_{i,j,k},\varphi_{i,j+1,k+1}) & a(\varphi_{i,j,k},\varphi_{i+1,j+1,k+1}) \\ a(\varphi_{i,j,k},\varphi_{i-1,j,k+1}) & a(\varphi_{i,j,k},\varphi_{i,j,k+1}) & a(\varphi_{i,j,k},\varphi_{i+1,j,k+1}) \\ a(\varphi_{i,j,k},\varphi_{i-1,j-1,k+1}) & a(\varphi_{i,j,k},\varphi_{i,j-1,k+1}) & a(\varphi_{i,j,k},\varphi_{i+1,j-1,k+1}) \end{bmatrix}. \quad (2.148)$$

The variable names in ANTARES correspond to the direction in which the entries are written in these stencils. They go from southwest to northeast. The middle term is named dd as in the 2D case. The same stencil as above in ANTARES notation is thus written

$$[A]^{(-1)}_{i,j,k} = \begin{bmatrix} \texttt{nw\_zm1(i,j,k)} & \texttt{n\_zm1(i,j,k)} & \texttt{ne\_zm1(i,j,k)} \\ \texttt{w\_zm1(i,j,k)} & \texttt{dd\_zm1(i,j,k)} & \texttt{e\_zm1(i,j,k)} \\ \texttt{sw\_zm1(i,j,k)} & \texttt{s\_zm1(i,j,k)} & \texttt{se\_zm1(i,j,k)} \end{bmatrix}, \quad (2.149)$$

$$[A]^{(0)}_{i,j,k} = \begin{bmatrix} \texttt{nw\_z0(i,j,k)} & \texttt{n\_z0(i,j,k)} & \texttt{ne\_z0(i,j,k)} \\ \texttt{w\_z0(i,j,k)} & \texttt{dd\_z0(i,j,k)} & \texttt{e\_z0(i,j,k)} \\ \texttt{sw\_z0(i,j,k)} & \texttt{s\_z0(i,j,k)} & \texttt{se\_z0(i,j,k)} \end{bmatrix}, \quad (2.150)$$

$$[A]^{(1)}_{i,j,k} = \begin{bmatrix} \texttt{nw\_zp1(i,j,k)} & \texttt{n\_zp1(i,j,k)} & \texttt{ne\_zp1(i,j,k)} \\ \texttt{w\_zp1(i,j,k)} & \texttt{dd\_zp1(i,j,k)} & \texttt{e\_zp1(i,j,k)} \\ \texttt{sw\_zp1(i,j,k)} & \texttt{s\_zp1(i,j,k)} & \texttt{se\_zp1(i,j,k)} \end{bmatrix}. \quad (2.151)$$

2.2.5.1 The Final Form of the Stencils

The values of the bilinear forms have been calculated before. Their values are given in (2.105)-(2.111). The fact that only 7 of these forms are non-vanishing leads to this simplified version of the operator:

$$[A]^{(-1)}_{i,j,k} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a(\varphi_{i,j,k},\varphi_{i,j,k-1}) & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$[A]^{(0)}_{i,j,k} = \begin{bmatrix} 0 & a(\varphi_{i,j,k},\varphi_{i,j+1,k}) & 0 \\ a(\varphi_{i,j,k},\varphi_{i-1,j,k}) & a(\varphi_{i,j,k},\varphi_{i,j,k}) & a(\varphi_{i,j,k},\varphi_{i+1,j,k}) \\ a & a(\varphi_{i,j,k},\varphi_{i,j-1,k}) & 0 \end{bmatrix}, \quad (2.152)$$

$$[A]^{(1)}_{i,j,k} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & a(\varphi_{i,j,k},\varphi_{i,j,k+1}) & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Written out the stencil (2.152) is

$$\begin{aligned}(Au)(i,j,k) =& a(\varphi_{i,j,k},\varphi_{i,j,k-1})u(i,j,k-1) + a(\varphi_{i,j,k},\varphi_{i,j-1,k})u(i,j-1,k)+\\ & a(\varphi_{i,j,k},\varphi_{i-1,j,k})u(i-1,j,k) + a(\varphi_{i,j,k},\varphi_{i,j,k})u(i,j,k)+\\ & a(\varphi_{i,j,k},\varphi_{i+1,j,k})u(i+1,j,k) + a(\varphi_{i,j,k},\varphi_{i,j+1,k})u(i,j+1,k)+\\ & a(\varphi_{i,j,k},\varphi_{i,j,k+1})u(i,j,k+1)\end{aligned} \tag{2.153}$$

This is the equation for the point (x_i, y_j, z_k) of the linear system of equations which has to be solved on the finest grid for lexicographical ordering of the grid points.

2.3 Discretization of the Nonlinear Helmholtz Equation

We now turn to the discretization of the nonlinear equation (1.45), i.e.,

$$-\nabla\cdot(\kappa(u)\nabla u(\boldsymbol{x}))+\xi(u)=f(\boldsymbol{x}). \tag{2.154}$$

The discretization for the nonlinear problem is similar to the discretization of the linear one. One major difference is the treatment of ξ: it is not multiplied with u but is directly dependent on u. This has implications for the discretization.

2.3.1 Deriving the Variational Formulation

Let V be a Hilbert space on Ω. Multiplying (2.154) with an arbitrary test function $v(\boldsymbol{x})\in \mathrm{V}(\Omega)$ and integrating over the domain Ω gives

$$-\int_\Omega \nabla\cdot(\kappa(u)\nabla u(\boldsymbol{x}))v(\boldsymbol{x})\,\mathrm{d}\Omega+\int_\Omega \xi(u)v(\boldsymbol{x})\,\mathrm{d}\Omega=\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega. \tag{2.155}$$

After using integration by parts on the first integral one arrives at the variational form of (2.154), which reads (recalling (2.21) for the definition of g_N):

$$\begin{aligned}&\text{find } u\in \mathrm{V}(\Omega):\\ &\int_\Omega \kappa(u)\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\,\mathrm{d}\Omega+\int_\Omega \xi(u)v(\boldsymbol{x})\,\mathrm{d}\Omega-\int_{\Gamma_N}\kappa(u)g_N v(\boldsymbol{x})\,\mathrm{d}\Gamma\\ &\qquad=\int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\Omega \qquad \forall v\in \mathrm{V}.\end{aligned} \tag{2.156}$$

This has the same form as the linear case, (2.4), except that κ and ξ depend on u instead of $\boldsymbol{x}$. The treatment of boundary conditions is equivalent to the treatment of boundary conditions in the linear case (section 2.2.1). In short: the Dirichlet part of the boundary integral vanishes because the arbitrary test function is chosen to be zero on the boundary. Choosing a subspace V_h and the basis functions φ_μ as test functions leads to

$$\begin{aligned}&\text{find } u_h\in \mathrm{V_h}(\Omega):\\ &\int_\Omega \kappa(u)\nabla u_h(\boldsymbol{x})\nabla\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega+\int_\Omega \xi(u)\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega-\int_{\Gamma_N}\kappa(u)g_N\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Gamma\\ &\qquad=\int_\Omega f(\boldsymbol{x})\varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Omega, \qquad \mu=1,2,\ldots,\dim(V_h).\end{aligned} \tag{2.157}$$

By writing $u_h(\boldsymbol{x})$ in terms of the basis functions of the subspace V_h,

$$u_h(\boldsymbol{x})=\sum_{\nu=1}^{N_h} u_\nu\varphi_\nu(\boldsymbol{x}), \tag{2.158}$$

where $N_h = \dim(V_h)$, one obtains

$$\begin{aligned} &\text{find } u \in \mathrm{V_h}(\Omega): \\ &\sum_{\nu=1}^{N_h} \left[\int_\Omega \kappa(u) \nabla(u_\nu \varphi_\nu(\boldsymbol{x})) \nabla \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega \right] + \int_\Omega \xi(u) \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega \\ &\quad - \int_{\Gamma_N} \kappa(u) g_N \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Gamma = \int_\Omega f(\boldsymbol{x}) \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega, \qquad \mu = 1,2,...,N_h. \end{aligned} \tag{2.159}$$

The first integral,

$$\sum_{\nu=1}^{N_h} \int_\Omega \kappa(u) \nabla(u_\nu \varphi_\nu(\boldsymbol{x})) \nabla \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega, \tag{2.160}$$

can be written as

$$\sum_{\nu=1}^{N_h} \left[u_\nu \int_\Omega \kappa(u) \nabla \varphi_\nu(\boldsymbol{x}) \nabla \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega \right], \tag{2.161}$$

which in turn can be written as a matrix–vector product $\bar{A} \cdot \boldsymbol{u}$ of the matrix

$$\bar{A} = \begin{bmatrix} <\varphi_1,\varphi_1> & <\varphi_1,\varphi_2> & \cdots & <\varphi_1,\varphi_{N_h}> \\ <\varphi_2,\varphi_1> & <\varphi_2,\varphi_1> & \cdots & <\varphi_1,\varphi_{N_h}> \\ \vdots & \vdots & \ddots & \vdots \\ <\varphi_{N_h},\varphi_1> & <\varphi_{N_h},\varphi_2> & \cdots & <\varphi_{N_h},\varphi_{N_h}> \end{bmatrix} \tag{2.162}$$

and the vector $\boldsymbol{u} = [u_1, u_2, ..., u_{N_h}]^{\mathrm{T}}$, where the entries $\bar{A}_{\mu\nu} = <\varphi_\mu, \varphi_\nu>$ are defined as

$$<\varphi_\mu, \varphi_\nu> = \int_\Omega \kappa(u) \nabla \varphi_\nu(\boldsymbol{x}) \nabla \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega. \tag{2.163}$$

Thus, (2.159) can be written as

$$\bar{A} \cdot \boldsymbol{u} + \underbrace{\int_\Omega \xi(u) \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega}_{\boldsymbol{\xi}} \underbrace{- \int_{\Gamma_N} \kappa(u) g_N \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Gamma}_{\boldsymbol{g}} = \underbrace{\int_\Omega f(\boldsymbol{x}) \varphi_\mu(\boldsymbol{x}) \,\mathrm{d}\Omega}_{\boldsymbol{b}}, \qquad \mu = 1,2,...,N_h, \tag{2.164}$$

where $\boldsymbol{\xi}, \boldsymbol{g}$ and $\boldsymbol{b}$ are vectors $\in \mathbb{R}^{N_h}$. Here, the difference to the linear case is obvious: while ξ entered the mass matrix in the linear case, it enters the calculation as a vector here.

The sum of $\bar{A} \cdot \boldsymbol{u}$, $\boldsymbol{\xi}$ and $\boldsymbol{g}$ can be summarized as the action of the nonlinear operator $\mathscr{A}$ on the vector $\boldsymbol{u}$:

$$\mathscr{A}(\boldsymbol{u}) = \boldsymbol{b}. \tag{2.165}$$

To solve this system of nonlinear equations we use the Newton method.

2.3.2 Calculation of the Nonlinear Operator $\mathscr{A}(\boldsymbol{u})$ for the Current Iteration

The calculation of the nonlinear operator $\mathscr{A}(\boldsymbol{u})$ must be performed at every iteration step because it depends on $\boldsymbol{u}$. It is defined as

$$\mathscr{A}(\boldsymbol{u}) = \bar{A}\cdot\boldsymbol{u} + \int_\Omega \xi(u)\varphi_\mu(\boldsymbol{x})\mathrm{d}\Omega - \int_{\Gamma_N} \kappa(u) g_N \varphi_\mu(\boldsymbol{x})\mathrm{d}\Gamma, \qquad \mu = 1,2,...,N_h. \tag{2.166}$$

We calculate the three terms separately.

2.3.2.1 Calculation of $\bar{A}\cdot\boldsymbol{u}$

To wit, $\bar{A}\cdot\boldsymbol{u}$ is defined as the product of

$$\bar{A} = \begin{bmatrix} <\varphi_1,\varphi_1> & <\varphi_1,\varphi_2> & \cdots & <\varphi_1,\varphi_{N_h}> \\ <\varphi_2,\varphi_1> & <\varphi_2,\varphi_1> & \cdots & <\varphi_1,\varphi_{N_h}> \\ \vdots & \vdots & \ddots & \vdots \\ <\varphi_{N_h},\varphi_1> & <\varphi_{N_h},\varphi_2> & \cdots & <\varphi_{N_h},\varphi_{N_h}> \end{bmatrix} \tag{2.167}$$

with the vector $\boldsymbol{u} = [u_1,u_2,...,u_{N_h}]^{\mathrm{T}}$, where the entries $\bar{A}_{\mu\nu} = <\varphi_\mu,\varphi_\nu>$ are defined as

$$<\varphi_\mu,\varphi_\nu> = \int_\Omega \kappa(u)\nabla\varphi_\nu(\boldsymbol{x})\nabla\varphi_\mu(\boldsymbol{x})\mathrm{d}\Omega. \tag{2.168}$$

This is very similar to the bilinear form which was defined for the linear problem in (2.8) and reads

$$a(u,v) = \int_\Omega \kappa(\boldsymbol{x})\nabla u(\boldsymbol{x})\nabla v(\boldsymbol{x})\mathrm{d}\Omega + \int_\Omega \xi(\boldsymbol{x})u(\boldsymbol{x})v(\boldsymbol{x})\mathrm{d}\Omega. \tag{2.169}$$

There are only two differences between the linear and the nonlinear problem:

1. κ is dependent on u in the nonlinear case
2. The term containing $\boldsymbol{\xi}$ is missing in the nonlinear case (because it is calculated separately in the next step.)

Because of the similarity to the linear problem we can use exactly the same calculations as we did for the linear case, bearing in mind the two differences stated above. The resulting entries for the matrix associated with grid point (i,j,k) are the same as the stiffness matrix entries (2.88)–(2.94),

$$\begin{aligned} \bar{A}_{ijk,ijk} = &\frac{h_y h_z}{2h_x}(\kappa_{i-1,j,k}+\kappa_{i+1,j,k}) + \frac{h_x h_z}{2h_y}(\kappa_{i,j-1,k}+\kappa_{i,j+1,k}) \\ &+ \frac{h_x h_y}{2h_z}(\kappa_{i,j,k-1}+\kappa_{i,j,k+1}) + \left(\frac{h_y h_z}{h_x}+\frac{h_x h_z}{h_y}+\frac{h_x h_y}{h_z}\right)\kappa_{i,j,k}, \end{aligned} \tag{2.170}$$

$$\bar{A}_{i+1jk,ijk} = -\frac{h_y h_z\left(\kappa_{i,j,k}+\kappa_{i+1,j,k}\right)}{2h_x}, \tag{2.171}$$

$$\bar{A}_{i-1jk,ijk} = -\frac{h_y h_z \left(\kappa_{i-1,j,k} + \kappa_{i,j,k}\right)}{2h_x}, \tag{2.172}$$

$$\bar{A}_{ij+1k,ijk} = -\frac{h_x h_z \left(\kappa_{i,j,k} + \kappa_{i,j+1,k}\right)}{2h_y}, \tag{2.173}$$

$$\bar{A}_{ij-1k,ijk} = -\frac{h_x h_z \left(\kappa_{i,j-1,k} + \kappa_{i,j,k}\right)}{2h_y}, \tag{2.174}$$

$$\bar{A}_{ijk+1,ijk} = -\frac{h_x h_y \left(\kappa_{i,j,k} + \kappa_{i,j,k+1}\right)}{2h_z}, \tag{2.175}$$

$$\bar{A}_{ijk-1,ijk} = -\frac{h_x h_y \left(\kappa_{i,j,k-1} + \kappa_{i,j,k}\right)}{2h_z}, \tag{2.176}$$

with the difference that $\kappa_{i,j,k}$ stands for $\kappa(u(x_i, y_j, z_k))$ here.

2.3.2.2 Calculation of $\boldsymbol{\xi}$

The calculation of $\boldsymbol{\xi}$ is straightforward and essentially the same as the calculation of the domain contributions for the linear case in section 2.2.4.2 (again with the difference that ξ is dependent on u now), thus it is omitted and only the result is stated:

$$(\boldsymbol{\xi}(u, \boldsymbol{x}))_{ijk} = \int_\Omega \xi(u)\varphi_{ijk}(\boldsymbol{x})\,\mathrm{d}\Omega = \cdots = h_x h_y h_z\, \xi(u(x_i, y_j, z_k)) \tag{2.177}$$

If the point (x_i, y_j, z_k) lies on the boundary $\partial\Omega$, the contribution is halved, exactly as it was in section 2.2.4.3:

$$(\boldsymbol{\xi}(u, \boldsymbol{x}))_{ijk} = \int_\Omega \xi(u)\varphi_{ijk}(\boldsymbol{x})\,\mathrm{d}\Omega = \cdots = \frac{1}{2} h_x h_y h_z\, \xi(u(x_i, y_j, z_k)) \tag{2.178}$$

Assuming, e.g., that only the first and last point of the grid lie on the boundary, this would result in the vector

$$\boldsymbol{\xi} = \begin{bmatrix} 1/2\, h_x h_y h_z\, \xi(u(x_i, y_j, z_k)) \\ h_x h_y h_z\, \xi(u(x_i, y_j, z_k)) \\ \vdots \\ h_x h_y h_z\, \xi(u(x_i, y_j, z_k)) \\ 1/2\, h_x h_y h_z\, \xi(u(x_i, y_j, z_k)) \end{bmatrix} \tag{2.179}$$

2.3.2.3 Calculation of the Neumann Boundary Term g_N in the Nonlinear Case

The calculation of the Neumann boundary terms also follows along the lines of the linear problem in section 2.2.4.5. It results in the following contribution on Neumann boundaries:

$$\int_{\Gamma_N} \kappa(u) g_N \varphi_\mu(\boldsymbol{x})\,\mathrm{d}\Gamma = \cdots = h_y h_z\, \kappa(u(x_i, y_j, z_k)) g_N(y_j, z_k) \tag{2.180}$$

Again, assuming that only the first and last point of the grid lie on the boundary, this would result in the vector

$$\boldsymbol{g}_N = \begin{bmatrix} h_y h_z \kappa(u(x_i, y_j, z_k)) g_N(y_j, z_k) \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ h_y h_z \kappa(u(x_i, y_j, z_k)) g_N(y_j, z_k) \end{bmatrix} \tag{2.181}$$

2.3.3 Summarizing: The Nonlinear Operator $\mathscr{A}(\boldsymbol{u})$ in Explicit Form

In summary, $\mathscr{A}(\boldsymbol{u}) = \bar{A} \cdot \boldsymbol{u} + \boldsymbol{\xi} + \boldsymbol{g}_N$ can be written as

$$\begin{aligned} \mathscr{A}(\boldsymbol{u}) = & \begin{bmatrix} <\varphi_1,\varphi_1> & <\varphi_1,\varphi_2> & \cdots & <\varphi_1,\varphi_{N_h}> \\ <\varphi_2,\varphi_1> & <\varphi_2,\varphi_1> & \cdots & <\varphi_1,\varphi_{N_h}> \\ \vdots & \vdots & \ddots & \vdots \\ <\varphi_{N_h-1},\varphi_1> & <\varphi_{N_h-1},\varphi_2> & \cdots & <\varphi_{N_h-1},\varphi_{N_h-1}> \\ <\varphi_{N_h},\varphi_1> & <\varphi_{N_h},\varphi_2> & \cdots & <\varphi_{N_h},\varphi_{N_h}> \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N_h-1} \\ u_{N_h} \end{bmatrix} \\ & + \begin{bmatrix} 1/2\, h_x h_y h_z \xi(u(x_i, y_j, z_k)) \\ h_x h_y h_z \xi(u(x_i, y_j, z_k)) \\ \vdots \\ h_x h_y h_z \xi(u(x_i, y_j, z_k)) \\ 1/2\, h_x h_y h_z \xi(u(x_i, y_j, z_k)) \end{bmatrix} - \begin{bmatrix} h_y h_z \kappa(u(x_i, y_j, z_k)) g_N(y_j, z_k) \\ 0 \\ \vdots \\ 0 \\ h_y h_z \kappa(u(x_i, y_j, z_k)) g_N(y_j, z_k) \end{bmatrix}, \end{aligned} \tag{2.182}$$

assuming that only grid points with indices 1 and N_h lie on the boundary. Of course, this assumption is nonsense in 3D but it serves its purpose of demonstrating the structures of the vectors that make up the nonlinear operator.

2.3.4 The Data Structure in the Nonlinear Case

Inner Grid Points For every non-boundary point $\mu \in G$ with the coordinates (x_i, y_j, z_k) we have

$$\begin{aligned} (\mathscr{A}(\boldsymbol{u}))_{ijk} = & <\varphi_{i,j,k}, \varphi_{i,j,k-1}> \cdot u_{i,j,k-1} + <\varphi_{i,j,k}, \varphi_{i,j-1,k}> \cdot u_{i,j-1,k} \\ & + <\varphi_{i,j,k}, \varphi_{i-1,j,k}> \cdot u_{i-1,j,k} + <\varphi_{i,j,k}, \varphi_{i,j,k}> \cdot u_{i,j,k} \\ & + <\varphi_{i,j,k}, \varphi_{i+1,j,k}> \cdot u_{i+1,j,k} + <\varphi_{i,j,k}, \varphi_{i,j+1,k}> \cdot u_{i,j+1,k} \\ & + <\varphi_{i,j,k}, \varphi_{i,j,k+1}> \cdot u_{i,j,k+1} + h_x h_y h_z \cdot \xi_{i,j,k}. \end{aligned} \tag{2.183}$$

We save the entries in the same data structure that we used for the linear case, i.e., the action of the nonlinear operator on the node μ with coordinates (x_i, y_j, z_k), using ANTARES notation, is as

follows:

$$\begin{aligned}(\bar{A}\cdot\boldsymbol{u})_{i,j,k} = {} & \texttt{dd\_zm1(i,j,k)}\cdot u_{i,j,k-1} + \texttt{s\_z0(i,j,k)}\cdot u_{i,j-1,k} \\ & + \texttt{w\_z0(i,j,k)}\cdot u_{i-1,j,k} + \texttt{dd\_z0(i,j,k)}\cdot u_{i,j,k} \\ & + \texttt{e\_z0(i,j,k)}\cdot u_{i+1,j,k} + \texttt{n\_z0(i,j,k)}\cdot u_{i,j+1,k} \\ & + \texttt{dd\_zp1(i,j,k)}\cdot u_{i,j,k+1} + h_x h_y h_z \cdot \xi_{i,j,k}\end{aligned} \tag{2.184}$$

Boundary Points For every boundary point $\mu \in G$ with the coordinates (x_i, y_j, z_k) we have

$$\begin{aligned}(\mathcal{A}(\boldsymbol{u}))_{ijk} = {} & <\varphi_{i,j,k},\varphi_{i,j,k-1}>\cdot u_{i,j,k-1} + <\varphi_{i,j,k},\varphi_{i,j-1,k}>\cdot u_{i,j-1,k} \\ & + <\varphi_{i,j,k},\varphi_{i-1,j,k}>\cdot u_{i-1,j,k} + <\varphi_{i,j,k},\varphi_{i,j,k}>\cdot u_{i,j,k} \\ & + <\varphi_{i,j,k},\varphi_{i+1,j,k}>\cdot u_{i+1,j,k} + <\varphi_{i,j,k},\varphi_{i,j+1,k}>\cdot u_{i,j+1,k} \\ & + <\varphi_{i,j,k},\varphi_{i,j,k+1}>\cdot u_{i,j,k+1} + \frac{1}{2} h_x h_y h_z \cdot \xi_{i,j,k} \\ & - h_y h_z \cdot \kappa(u_{ijk}) g_N(x_{j,k}),\end{aligned} \tag{2.185}$$

where the contributions of points left/right of lower/upper boundaries are zero.

2.4 Summary

In this chapter, we have discretized the linear and nonlinear equations with the finite elements method. We have obtained a (non-)linear system of equations. We can now turn to its solution. To this end, we use the multigrid method which is the topic of the next chapter.

CHAPTER 3

THE MULTIGRID METHOD

" In my program, a 48x40 grid was used so that the unknown grid function and the right hand side vector occupied almost all operational memory. I started working with the only iterative method familiar to me then, the relaxation method [here, apparently, simple iteration or Richardson method is meant-M.B.], and soon became convinced in its poor efficiency. If at that time I was aware of the ADM (Alternated Direction Method) with optimally chosen parameters (by that time it was already known), or, at least of the overrelaxation (SOR) method, then I would probably not have looked for something else. But "fortunately" I was sufficiently ignorant in this topic. I was trying to understand what caused the slow convergence by looking at the residual evolution in the course of iterations and easily discovered the well known fact: first, the nonsmooth residual decreased fast and became smooth. After this the decrease became desperately slow. How the idea to formulate the correction equation as a problem on a coarse grid with the residual at the right hand side crossed my mind is difficult to recall now. Apparently, a certain hint was given by the Newton method which, for linear equations, leads to the same problem. This problem can be eased by switching to a coarse grid and the smoothness of the right hand side (the residual) justifies such an approach."

— R.P. Fedorenko, the pioneer of multigrid
http://wwwhome.math.utwente.nl/~botchevma/fedorenko/index.php

3.1 Introductory Remarks

Multigrid is considered as one of the fastest methods for the solution of systems of equations today. A good introduction to multigrid methods can be found in Briggs and McCormick, (2000) or Köckler, (2012). A thorough treatment is given in Trottenberg, Oosterlee, and Schüller, (2001).

The underlying principle of multigrid methods it the smoothing property of the standard iterative schemes for the solution of linear systems of equations. We only give a very short introduction to multigrid here. The interested reader can read more about the theory in the mentioned expositions. There are two pillars the whole multigrid principle rests on:

- error smoothing, as stated by the **smoothing principle**: many iterative methods like Jacobi or Gauss–Seidel have a strong smoothing effect on the error of any approximation if appropriately applied to a discrete elliptical problem

- and the fact that any smooth quantity on a certain grid can be, without any essential loss of information, be approximated on a coarser grid, stated by the **coarse grid principle**: a smooth error term is well approximated on a coarse grid and a coarse grid procedure is substantially less expensive than a fine grid procedure.

So, in essence, multigrid can be summarized the following way: apply an iterative method on the system of equations to be solved until the error is smooth. Then, approximate the now smooth error on a coarser grid and continue the computation on the coarser grid where the computation is much cheaper. This process can be repeated as long as the grid can be made coarser. In ANTARES, a grid coarsening factor of two is chosen, so that a sequence of grids looks like figure 3.1.

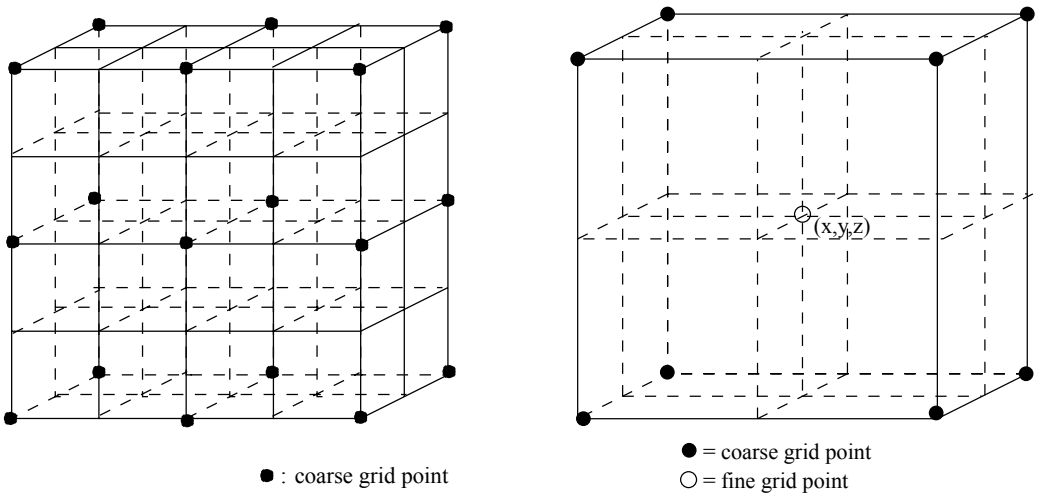


Figure 3.1: A sequence of four two–dimensional grids. From Trottenberg, Oosterlee, and Schüller, (2001).

We further demonstrate the core concept of the multigrid technique by means of the two–grid correction scheme.

The nonlinear multigrid method is developed in section 3.3.

3.2 Linear Multigrid

The goal is to solve the linear system of equations (2.42),

$$A\boldsymbol{u} = \boldsymbol{f} \tag{3.1}$$

which resulted from the discretization of (2.1) on the discrete grid

$$G_h = \{(x,y,z) : x = x_i = ih_x,\, y = y_j = jh_y,\, z = z_k = kh_z;\, i,j,k \in \mathbb{Z}\}, \tag{3.2}$$

where $\boldsymbol{h} = (h_x, h_y, h_z)^{\mathrm{T}}$ is a vector of fixed mesh sizes. The subscript $_h$ is used to label the starting (fine) grid while the subscript $_{2h}$ is used to label to coarser grid, i.e., we write the system of equations as

$$A_h \boldsymbol{u}_h = \boldsymbol{f}_h. \tag{3.3}$$

The multigrid two–grid correction scheme has the following structure:

1. Choose an (arbitrary) initial guess $\boldsymbol{v}_h^{(0)}$ on the finest grid G_h.
2. Relax $A_h \boldsymbol{v}_h^{(0)} = \boldsymbol{f}_h$ ν_1 times on the finest grid G_h to obtain the improved value $\boldsymbol{v}_h^{(\nu_1)}$.
3. Calculate the residual $\boldsymbol{r}_h = \boldsymbol{f}_h - A_h \boldsymbol{v}_h^{(\nu_1)}$.
4. Transfer the residual $\boldsymbol{r}_h$ to the coarser grid G_{2h} to obtain $\boldsymbol{r}_{2h}$.
5. Solve $A_{2h}\boldsymbol{e}_{2h} = \boldsymbol{r}_{2h}$ on G_{2h} to obtain an approximation to the error $\boldsymbol{e}_{2h}$.
6. Transfer the error $\boldsymbol{e}_{2h}$ to the finer grid G_h to obtain $\boldsymbol{e}_h$.
7. Correct the fine–grid approximation $\boldsymbol{v}_h^{(\nu_1)}$ by adding the error: $\boldsymbol{v}_h^{(\nu_1+1)} = \boldsymbol{v}_h^{(\nu_1)} + \boldsymbol{e}_h$.
8. Relax $A_h \boldsymbol{v}_h^{(\nu_1+1)} = \boldsymbol{f}_h$ ν_2 times on the finest grid G_h to obtain the improved value $\boldsymbol{v}_h^{(\nu_1+1+\nu_2)}$.

By one iteration of this algorithm the initial guess $\boldsymbol{v}^{(0)}$ is corrected $\nu_1 + 1 + \nu_2$ times in total. In what follows we present the elements or components of the multigrid algorithm and describe the implementation of each component into ANTARES. Although it is well known how to choose suitable multigrid components for large classes of problems, it is very difficult to define the right components in new applications. Since ANTARES is a new application it is up to the future users to change the components accordingly if the performance of this initial setup is not satisfactory.

3.2.1 The Initial Guess

1. Choose an (arbitrary) initial guess $\boldsymbol{v}_h^{(0)}$ on the finest grid G_h.

This first step in the outline does not need a special implementation. In ANTARES, the initial guess stems from the previous Runge–Kutta time step so there is no need to take a guess.

3.2.2 The Smoothing Routine

2. Relax $A_h \boldsymbol{v}_h^{(0)} = \boldsymbol{f}_h$ ν_1 times on the finest grid G_h to obtain the improved value $\boldsymbol{v}_h^{(\nu_1)}$.

Principally, one could use any iterative scheme to smooth the high–frequency error components. One of the best iterative schemes for multigrid smoothing is the Gauss–Seidel method because of its superior smoothing properties. It is implemented in ANTARES in the lexicographic variant only. A future improvement could be an implementation of block-iterative techniques for 3D as e.g. plane–relaxation Gauss–Seidel.

3.2.3 Applying the Linear Operator and Calculating the Residual

3. Calculate the residual $\boldsymbol{r}_h = \boldsymbol{f}_h - A_h \boldsymbol{v}_h^{(\nu_1)}$.

This is straightforward and does not need any discussion.

3.2.4 The Restriction and Prolongation Operators

3.2.4.1 The Restriction Operator, Part I

4. Transfer the residual $\boldsymbol{r}_h$ to the coarser grid G_{2h} to obtain $\boldsymbol{r}_{2h}$.

The transfer of the residual to the coarser grid is the first point in the two–grid correction scheme which needs an elaborate explanation since this is crucial in the multigrid process. The following exposition follows along the lines of Wesseling, (1992) and Elman, Silvester, and Wathen, (2014).

Before we can transfer anything to a coarser grid we must first define the coarser grid itself. The standard coarsening approach that is used with multigrid is that to take the original grid G_h with mesh width h and double the mesh width so that the new coarse grid G_{2h} has a mesh width $2h$. This is also the approach that is used in ANTARES.

As it turns out, the restriction operator is canonically defined for a discretization by the finite elements method and can be derived immediately from the prolongation operator—the operator that is used for transferring the residual back to the fine grid. The relationship between the prolongation operator P and the restriction operator R is

$$P = R^*, \tag{3.4}$$

where R^* denotes the adjoint of R.

This means that we have to calculate the prolongation operator first, which we turn to next.

3.2.4.2 The Prolongation Operator

We define an operator P (standing for prolongation) which transfers any vector from the coarse grid G_{2h} to the fine grid G_h, i.e., we are searching for a $P : V_{2h} \to V_h$ with

$$P\boldsymbol{u}_{2h} = \boldsymbol{u}_h, \tag{3.5}$$

where V_h and V_{2h} are the vector spaces of $\boldsymbol{u}_h$ and $\boldsymbol{u}_{2h}$, respectively. In stencil notation the prolongation operator is defined the following way (see Wesseling, 1992):

Let $P : V_{2h} \to V_h$ be a prolongation operator. The stencil of P can be represented as

$$(P\boldsymbol{u}_{2h})_i = \sum_{j \in \mathbb{Z}^d} P^*(j, i-2j)\, u_{2h,j} \tag{3.6}$$

where P^* denotes the adjoint of the prolongation operator and i and j are points in the grid.

Remark: in the stencil, as always, the second entry of P^* is the displacement in the stencil and the first entry is the point where the stencil is evaluated.

To obtain the stencil of P we need a rule which gives $P\boldsymbol{u}_{2h}$ for a given $\boldsymbol{u}_{2h}$. For Finite Element Methods the prolongation is defined through the natural inclusion of the coarse grid subspace V_{2h} into V_h,

$$P\boldsymbol{u}_{2h} = \boldsymbol{u}_h, \tag{3.7}$$

for all functions $\boldsymbol{u}_{2h} \in V_{2h}$. A representation of P is determined by the finite element bases. Suppose that φ^{2h} is a nodal basis function for V_{2h}. Any coarse grid vector $\boldsymbol{u}_{2h}$ can be expanded independently in terms of the fine and coarse-grid basis vectors

$$\boldsymbol{u}_{2h} = \sum_{n=1}^{\dim(V_{2h})} u_{n,2h}\, \varphi_n^{2h} = \sum_{m=1}^{\dim(V_h)} u_{m,h}\, \varphi_m^h, \tag{3.8}$$

where $u_{n,2h}$ and $u_{m,h}$ are the coefficients of the expansions on the coarse and fine grid, respectively. Since we have chosen the finite element shape functions as basis functions for each φ_μ^h there exists a unique $\boldsymbol{p}_i \in \Omega$ such that

$$\varphi_j^h(\boldsymbol{p}_i) = \delta_{ij}, \qquad i,j = 1,2,\ldots,\dim(V_h), \tag{3.9}$$

where $\boldsymbol{p}_i$ denotes the fine grid vertex associated with φ_μ^h. Combining (3.8) and (3.9) gives

$$\boldsymbol{u}_i^h = \sum_{j=1}^{\dim(V_{2h})} u_{j,2h}\, \varphi_j^{2h}(\boldsymbol{p}_i), \qquad i = 1,2,\ldots,\dim(V_h). \tag{3.10}$$

(To reduce notational clutter we have written the grid indices h and $2h$ of $\boldsymbol{u}$ as superscripts and will keep doing so in this subsection.) The matrix entries of the prolongation operator are then defined as the coarse-grid shape functions evaluated at the fine-grid vertices:

$$P(i,j) = \varphi_j^{2h}(\boldsymbol{p}_i^h). \tag{3.11}$$

We demonstrate the process for the cubicle Q_1, assuming that the domain consists of only that cubicle and ignoring boundary conditions. This gives us for $\dim(V_{2h}) = 9$ and $\dim(V_h) = 27$. By using (3.10) one obtains

$$\begin{aligned}\boldsymbol{u}_1^h &= \sum_{j=1}^{4} u_{j,2h}\, \varphi_j^{2h}(\boldsymbol{p}_1^h) \\ &= u_{1,2h} \underbrace{\varphi_1^{2h}(\boldsymbol{p}_1^h)}_{=1} + u_{2,2h} \underbrace{\varphi_2^{2h}(\boldsymbol{p}_1^h)}_{=0} + u_{3,2h} \underbrace{\varphi_3^{2h}(\boldsymbol{p}_1^h)}_{=0} + u_{4,2h} \underbrace{\varphi_4^{2h}(\boldsymbol{p}_1^h)}_{=0}. \end{aligned} \tag{3.12}$$

The same holds for every fine grid vertex that coincides with a coarse grid vertex. Further,

$$\begin{aligned}\boldsymbol{u}_2^h &= \sum_{j=1}^{4} u_{j,2h}\, \varphi_j^{2h}(\boldsymbol{p}_2^h) \\ &= u_{1,2h} \underbrace{\varphi_1^{2h}(\boldsymbol{p}_2^h)}_{=0.5} + u_{2,2h} \underbrace{\varphi_2^{2h}(\boldsymbol{p}_2^h)}_{=0.5} + u_{3,2h} \underbrace{\varphi_3^{2h}(\boldsymbol{p}_2^h)}_{=0} + u_{4,2h} \underbrace{\varphi_4^{2h}(\boldsymbol{p}_2^h)}_{=0}. \end{aligned} \tag{3.13}$$

The same holds for every fine grid vertex that lies between two coarse grid vertices. It doesn't matter in which direction we move. Finally,

$$\begin{aligned}\boldsymbol{u}_5^h &= \sum_{j=1}^{4} u_{j,2h}\, \varphi_j^{2h}(\boldsymbol{p}_5^h) \\ &= u_{1,2h} \underbrace{\varphi_1^{2h}(\boldsymbol{p}_5^h)}_{=0.25} + u_{2,2h} \underbrace{\varphi_2^{2h}(\boldsymbol{p}_5^h)}_{=0.25} + u_{3,2h} \underbrace{\varphi_3^{2h}(\boldsymbol{p}_5^h)}_{=0.25} + u_{4,2h} \underbrace{\varphi_4^{2h}(\boldsymbol{p}_5^h)}_{=0.25}. \end{aligned} \tag{3.14}$$

The same holds for every fine grid vertex that lies in the coarse grid plane between four coarse grid vertices.

If we move into the cube to the middle point there are no corresponding coarse grid vertices in that plane which we could use. We have to use the already calculated points to determine the values of the prolongation operator. The value we get is $P(14, j) = 0.125$.

We see that the prolongation operator we get from the finite element methods is the same as the canonical trilinear prolongation operator (see Wesseling, 1992). The effect of trilinar interpolation is:

$$\left(P\boldsymbol{u}^{2h}\right)_{2\boldsymbol{i}} = \boldsymbol{u}_{\boldsymbol{i}}^{2h}, \tag{3.15}$$

$$\left(P\boldsymbol{u}^{2h}\right)_{2\boldsymbol{i}\pm\boldsymbol{e}_1} = \frac{1}{2}\left(\boldsymbol{u}_{\boldsymbol{i}}^{2h} + \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_1}^{2h}\right), \tag{3.16}$$

$$\left(P\boldsymbol{u}^{2h}\right)_{2\boldsymbol{i}\pm\boldsymbol{e}_1\pm\boldsymbol{e}_2} = \frac{1}{4}\left(\boldsymbol{u}_{\boldsymbol{i}}^{2h} + \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_1}^{2h} + \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_2}^{2h}\boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_1\pm\boldsymbol{e}_2}^{2h}\right), \tag{3.17}$$

$$\left(P\boldsymbol{u}^{2h}\right)_{2\boldsymbol{i}\pm\boldsymbol{e}_1\pm\boldsymbol{e}_2\pm\boldsymbol{e}_3} = \frac{1}{8}\left(\boldsymbol{u}_{\boldsymbol{i}}^{2h} + \sum_{\alpha=1}^{3} \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_\alpha}^{2h} + \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_1\pm\boldsymbol{e}_2}^{2h} + \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_2\pm\boldsymbol{e}_3}^{2h} + \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_3\pm\boldsymbol{e}_1}^{2h} + \boldsymbol{u}_{\boldsymbol{i}\pm\boldsymbol{e}_1\pm\boldsymbol{e}_2\pm\boldsymbol{e}_3}^{2h}\right) \tag{3.18}$$

The resulting stencil is

$$\left[P^*\right] = \left[\begin{bmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \end{bmatrix} \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \begin{bmatrix} \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \end{bmatrix}\right]. \tag{3.19}$$

It becomes important for the calculation of the coarse grid operator A_{2h} in section 3.2.5.1.

3.2.4.3 The Restriction Operator, Part II

Using (3.4) we can immediately calculate the restriction operator. The stencil of the restriction operator reads

$$[R] = \frac{1}{64}\left[\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\quad\begin{bmatrix}2&4&2\\4&8&4\\2&4&2\end{bmatrix}\quad\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\right]. \tag{3.20}$$

As it turns out, this coincides with the three–dimensional full–weighting operator.

3.2.5 Solving the Linear System on the Coarse Grid

5. Solve $A_{2h}\boldsymbol{e}_{2h} = \boldsymbol{r}_{2h}$ on G_{2h} to obtain an approximation to the error $\boldsymbol{e}_{2h}$.

To solve the system on the coarse grid, we need to know what the operator A_{2h} looks like. It can be calculated in various ways. We have chosen to do it via the Galerkin coarse grid approximation (GCA) (see section 3.2.5.1). The coarse grid operator A_{2h} is calculated with the help of both the prolongation and the restriction operator.

3.2.5.1 The Coarse Grid Operator A_{2h}

For a multigrid method there are two possibilities to obtain the coarse grid operator A_{2h}: the straightforward way of discretizing the equation that is to be solved on the coarse grid directly ("discretization coarse grid approximation (DCA)") and the more involved method called "Galerkin coarse grid approximation (GCA)" where the operator A_{2h} is constructed with the help of prolongation and restriction operators. One big advantage of the GCA is the fact that is ensures perfect coupling between the operators of the different grid levels which plays a big role when using spherical grids (for details on this see Happenhofer, 2014). Because one major application of ANTARES is the simulation of Cepheids, this solver might be used for Cepheid simulations in the future. And since the grids in those simulations are spherical we have implemented the GCA method.

Wesseling, (1992) gives an explicit formula for the calculation of the coarse grid operator:

$$A_{2h}(i,n) = \sum_{m\in S_R}\sum_{k\in S_A} R(i,m)A_h(2i+m,k)P^*(i+n,m+k-2n), \tag{3.21}$$

where S_A and S_R are the structures of A and R, defined as

$$S_A = \{j \in \mathbb{Z}^d : \exists i \in G_h \text{with } A_h(i,j) \neq 0\} \tag{3.22}$$

$$S_R = \{j \in \mathbb{Z}^d : \exists i \in G_{2h} \text{with } R(i,j) \neq 0\} \tag{3.23}$$

The calculation of (3.21) is rather involved and was thus done with Mathematica (see appendix). To give a sample of what they look like we present the entry for $A_{2h}((i,j,k),(0,0,0))$:

$A_{2h}((i,j,k),(0,0,0)) =$

$$\frac{1}{8}\mathrm{ddZ0}(2i,2j,2k)+\frac{1}{32}\mathrm{ddZ0}(2i,2j,2k-1)+\frac{1}{32}\mathrm{ddZ0}(2i,2j,2k+1)+\frac{1}{32}\mathrm{ddZ0}(2i,2j+1,2k)+$$
$$\frac{1}{32}\mathrm{ddZ0}(2i,2j-1,2k)+\frac{1}{128}\mathrm{ddZ0}(2i,2j-1,2k-1)+\frac{1}{128}\ \mathrm{ddZ0}(2i,2j-1,2k+1)+$$
$$\frac{1}{128}\mathrm{ddZ0}(2i,2j+1,2k-1)+\frac{1}{128}\mathrm{ddZ0}(2i,2j+1,2k+1)+\frac{1}{32}\mathrm{ddZ0}(2i-1,2j,2k)+$$
$$\frac{1}{128}\mathrm{ddZ0}(2i-1,2j,2k-1)+\frac{1}{128}\mathrm{ddZ0}(2i-1,2j,2k+1)+\frac{1}{128}\mathrm{ddZ0}(2i-1,2j-1,2k)+$$
$$\frac{1}{512}\mathrm{ddZ0}(2i-1,2j-1,2k-1)+\frac{1}{512}\mathrm{ddZ0}(2i-1,2j-1,2k+1)+\frac{1}{128}\ \mathrm{ddZ0}(2i-1,2j+1,2k)+$$
$$\frac{1}{512}\mathrm{ddZ0}(2i-1,2j+1,2k-1)+\frac{1}{512}\mathrm{ddZ0}(2i-1,2j+1,2k+1)+\frac{1}{32}\mathrm{ddZ0}(2i+1,2j,2k)+$$
$$\frac{1}{128}\mathrm{ddZ0}(2i+1,2j,2k-1)+\frac{1}{128}\ \mathrm{ddZ0}(2i+1,2j,2k+1)+\frac{1}{128}\mathrm{ddZ0}(2i+1,2j-1,2k)+$$
$$\frac{1}{512}\mathrm{ddZ0}(2i+1,2j-1,2k-1)+\frac{1}{512}\ \mathrm{ddZ0}(2i+1,2j-1,2k+1)+\frac{1}{128}\mathrm{ddZ0}(2i+1,2j+1,2k)+$$
$$\frac{1}{512}\mathrm{ddZ0}(2i+1,2j+1,2k-1)+\frac{1}{512}\ \mathrm{ddZ0}(2i+1,2j+1,2k+1)+$$
$$\frac{1}{16}\mathrm{ddZm1}(2i,2j,2k)+\frac{1}{16}\mathrm{ddZm1}(2i,2j,2k+1)++\frac{1}{128}\mathrm{nZm1}(2i-1,2j,2k+1)$$
$$\frac{1}{64}\mathrm{ddZm1}(2i,2j-1,2k)+\frac{1}{64}\mathrm{ddZm1}(2i,2j-1,2k+1)+\frac{1}{64}\mathrm{ddZm1}(2i,2j+1,2k)+$$
$$\frac{1}{64}\mathrm{ddZm1}(2i,2j+1,2k+1)+\frac{1}{64}\mathrm{ddZm1}(2i-1,2j,2k)+\frac{1}{64}\mathrm{ddZm1}(2i-1,2j,2k+1)+$$
$$\frac{1}{256}\mathrm{ddZm1}(2i-1,2j-1,2k)+\frac{1}{256}\mathrm{ddZm1}(2i-1,2j-1,2k+1)+\frac{1}{256}\mathrm{ddZm1}(2i-1,2j+1,2k)+$$
$$\frac{1}{256}\mathrm{ddZm1}(2i-1,2j+1,2k+1)+\frac{1}{64}\mathrm{ddZm1}(2i+1,2j,2k)+\frac{1}{64}\mathrm{ddZm1}(2i+1,2j,2k+1)+$$
$$\frac{1}{256}\mathrm{ddZm1}(2i+1,2j-1,2k)+\frac{1}{256}\mathrm{ddZm1}(2i+1,2j-1,2k+1)+\frac{1}{256}\mathrm{ddZm1}(2i+1,2j+1,2k)+$$
$$\frac{1}{256}\mathrm{ddZm1}(2i+1,2j+1,2k+1)+\frac{1}{16}\mathrm{ddZp1}(2i,2j,2k)+\frac{1}{16}\mathrm{ddZp1}(2i,2j,2k-1)+$$
$$\frac{1}{64}\mathrm{ddZp1}(2i,2j-1,2k)+\frac{1}{64}\mathrm{ddZp1}(2i,2j-1,2k-1)+\frac{1}{64}\mathrm{ddZp1}(2i,2j+1,2k)+$$
$$\frac{1}{64}\mathrm{ddZp1}(2i,2j+1,2k-1)+\frac{1}{64}\mathrm{ddZp1}(2i-1,2j,2k)+\frac{1}{64}\mathrm{ddZp1}(2i-1,2j,2k-1)+$$
$$\frac{1}{256}\mathrm{ddZp1}(2i-1,2j-1,2k)+\frac{1}{256}\mathrm{ddZp1}(2i-1,2j-1,2k-1)+$$
$$\frac{1}{256}\mathrm{ddZp1}(2i-1,2j+1,2k)+\frac{1}{256}\mathrm{ddZp1}(2i-1,2j+1,2k-1)+$$
$$\frac{1}{64}\mathrm{ddZp1}(2i+1,2j,2k)+\frac{1}{64}\mathrm{ddZp1}(2i+1,2j,2k-1)+\frac{1}{256}\mathrm{ddZp1}(2i+1,2j-1,2k)+$$
$$\frac{1}{256}\mathrm{ddZp1}(2i+1,2j-1,2k-1)+\frac{1}{256}\mathrm{ddZp1}(2i+1,2j+1,2k)+\frac{1}{16}\mathrm{eZ0}(2i,2j,2k)+$$
$$\frac{1}{256}\mathrm{ddZp1}(2i+1,2j+1,2k-1)+\frac{1}{64}\mathrm{eZ0}(2i,2j,2k-1)+\frac{1}{64}\mathrm{eZ0}(2i,2j,2k+1)+$$
$$\frac{1}{64}\mathrm{eZ0}(2i,2j-1,2k)+\frac{1}{256}\mathrm{eZ0}(2i,2j-1,2k-1)+\frac{1}{256}\mathrm{eZ0}(2i,2j-1,2k+1)+$$

$$\frac{1}{64}\mathrm{eZ0}(2i,2j+1,2k)+\frac{1}{256}\mathrm{eZ0}(2i,2j+1,2k-1)+\frac{1}{256}\mathrm{eZ0}(2i,2j+1,2k+1)+$$
$$\frac{1}{16}\mathrm{eZ0}(2i-1,2j,2k)+\frac{1}{64}\mathrm{eZ0}(2i-1,2j,2k-1)+\frac{1}{64}\mathrm{eZ0}(2i-1,2j,2k+1)+$$
$$\frac{1}{64}\mathrm{eZ0}(2i-1,2j-1,2k)+\frac{1}{256}\mathrm{eZ0}(2i-1,2j-1,2k-1)+\frac{1}{256}\mathrm{eZ0}(2i-1,2j-1,2k+1)+$$
$$\frac{1}{64}\mathrm{eZ0}(2i-1,2j+1,2k)+\frac{1}{256}\mathrm{eZ0}(2i-1,2j+1,2k-1)+\frac{1}{256}\mathrm{eZ0}(2i-1,2j+1,2k+1)+$$
$$\frac{1}{32}\mathrm{eZm1}(2i,2j,2k)+\frac{1}{32}\mathrm{eZm1}(2i,2j,2k+1)+\frac{1}{128}\mathrm{eZm1}(2i,2j-1,2k)+$$
$$\frac{1}{128}\mathrm{eZm1}(2i,2j-1,2k+1)+\frac{1}{128}\mathrm{eZm1}(2i,2j+1,2k)+\frac{1}{128}\mathrm{eZm1}(2i,2j+1,2k+1)+$$
$$\frac{1}{32}\mathrm{eZm1}(2i-1,2j,2k)+\frac{1}{32}\mathrm{eZm1}(2i-1,2j,2k+1)+\frac{1}{128}\mathrm{eZm1}(2i-1,2j-1,2k)+$$
$$\frac{1}{128}\mathrm{eZm1}(2i-1,2j-1,2k+1)+\frac{1}{128}\mathrm{eZm1}(2i-1,2j+1,2k)+\frac{1}{128}\mathrm{eZm1}(2i-1,2j+1,2k+1)+$$
$$\frac{1}{32}\mathrm{eZp1}(2i,2j,2k)+\frac{1}{32}\mathrm{eZp1}(2i,2j,2k-1)+\frac{1}{128}\mathrm{eZp1}(2i,2j-1,2k)+\frac{1}{64}\mathrm{neZm1}(2i,2j-1,2k+1)$$
$$+\frac{1}{128}\mathrm{eZp1}(2i,2j-1,2k-1)+\frac{1}{128}\mathrm{eZp1}(2i,2j+1,2k)+\frac{1}{128}\mathrm{eZp1}(2i,2j+1,2k-1)+$$
$$\frac{1}{32}\mathrm{eZp1}(2i-1,2j,2k)+\frac{1}{32}\mathrm{eZp1}(2i-1,2j,2k-1)+\frac{1}{128}\mathrm{eZp1}(2i-1,2j-1,2k)+$$
$$\frac{1}{128}\mathrm{eZp1}(2i-1,2j-1,2k-1)+\frac{1}{128}\mathrm{eZp1}(2i-1,2j+1,2k)+\frac{1}{128}\mathrm{eZp1}(2i-1,2j+1,2k-1)+$$
$$\frac{1}{32}\mathrm{neZ0}(2i,2j,2k)+\frac{1}{128}\mathrm{neZ0}(2i,2j,2k-1)+\frac{1}{128}\mathrm{neZ0}(2i,2j,2k+1)+$$
$$\frac{1}{32}\mathrm{neZ0}(2i,2j-1,2k)+\frac{1}{128}\mathrm{neZ0}(2i,2j-1,2k-1)+\frac{1}{128}\mathrm{neZ0}(2i,2j-1,2k+1)+$$
$$\frac{1}{32}\mathrm{neZ0}(2i-1,2j,2k)+\frac{1}{128}\mathrm{neZ0}(2i-1,2j,2k-1)+\frac{1}{128}\mathrm{neZ0}(2i-1,2j,2k+1)+$$
$$\frac{1}{32}\mathrm{neZ0}(2i-1,2j-1,2k)+\frac{1}{128}\mathrm{neZ0}(2i-1,2j-1,2k-1)+\frac{1}{128}\mathrm{neZ0}(2i-1,2j-1,2k+1)+$$
$$\frac{1}{64}\mathrm{neZm1}(2i,2j,2k)+\frac{1}{64}\mathrm{neZm1}(2i,2j,2k+1)+\frac{1}{64}\mathrm{neZm1}(2i,2j-1,2k)+$$
$$\frac{1}{64}\mathrm{neZm1}(2i-1,2j,2k)+\frac{1}{64}\mathrm{neZm1}(2i-1,2j,2k+1)+\frac{1}{64}\mathrm{neZm1}(2i-1,2j-1,2k)+$$
$$\frac{1}{64}\mathrm{neZm1}(2i-1,2j-1,2k+1)+\frac{1}{64}\mathrm{neZp1}(2i,2j,2k)+\frac{1}{64}\mathrm{neZp1}(2i,2j,2k-1)+$$
$$\frac{1}{64}\mathrm{neZp1}(2i,2j-1,2k)+\frac{1}{64}\mathrm{neZp1}(2i,2j-1,2k-1)+\frac{1}{64}\mathrm{neZp1}(2i-1,2j,2k)+$$
$$\frac{1}{64}\mathrm{neZp1}(2i-1,2j,2k-1)+\frac{1}{64}\mathrm{neZp1}(2i-1,2j-1,2k)+\frac{1}{64}\mathrm{neZp1}(2i-1,2j-1,2k-1)+$$
$$\frac{1}{32}\mathrm{nwZ0}(2i,2j,2k)+\frac{1}{128}\mathrm{nwZ0}(2i,2j,2k-1)+\frac{1}{128}\mathrm{nwZ0}(2i,2j,2k+1)+$$
$$\frac{1}{32}\mathrm{nwZ0}(2i,2j-1,2k)+\frac{1}{128}\mathrm{nwZ0}(2i,2j-1,2k-1)+\frac{1}{128}\mathrm{nwZ0}(2i,2j-1,2k+1)+$$
$$\frac{1}{32}\mathrm{nwZ0}(2i+1,2j,2k)+\frac{1}{128}\mathrm{nwZ0}(2i+1,2j,2k-1)+\frac{1}{128}\mathrm{nwZ0}(2i+1,2j,2k+1)+$$

$$\frac{1}{32}\text{nwZ0}(2i+1,2j-1,2k)+\frac{1}{128}\text{nwZ0}(2i+1,2j-1,2k-1)+\frac{1}{128}\text{nwZ0}(2i+1,2j-1,2k+1)+$$
$$\frac{1}{64}\text{nwZm1}(2i,2j,2k)+\frac{1}{64}\text{nwZm1}(2i,2j,2k+1)+\frac{1}{64}\text{nwZm1}(2i,2j-1,2k)+$$
$$\frac{1}{64}\text{nwZm1}(2i,2j-1,2k+1)+\frac{1}{64}\text{nwZm1}(2i+1,2j,2k)+\frac{1}{64}\text{nwZm1}(2i+1,2j,2k+1)+$$
$$\frac{1}{64}\text{nwZm1}(2i+1,2j-1,2k)+\frac{1}{64}\text{nwZm1}(2i+1,2j-1,2k+1)+\frac{1}{64}\text{nwZp1}(2i,2j,2k)+$$
$$\frac{1}{64}\text{nwZp1}(2i,2j,2k-1)+\frac{1}{64}\text{nwZp1}(2i,2j-1,2k)+\frac{1}{64}\text{nwZp1}(2i,2j-1,2k-1)+$$
$$\frac{1}{64}\text{nwZp1}(2i+1,2j,2k)+\frac{1}{64}\text{nwZp1}(2i+1,2j,2k-1)+\frac{1}{64}\text{nwZp1}(2i+1,2j-1,2k)+$$
$$\frac{1}{64}\text{nwZp1}(2i+1,2j-1,2k-1)+\frac{1}{16}\text{nZ0}(2i,2j,2k)+\frac{1}{64}\text{nZ0}(2i,2j,2k-1)+$$
$$\frac{1}{64}\text{nZ0}(2i,2j,2k+1)+\frac{1}{16}\text{nZ0}(2i,2j-1,2k)+\frac{1}{64}\text{nZ0}(2i,2j-1,2k-1)+$$
$$\frac{1}{64}\text{nZ0}(2i,2j-1,2k+1)+\frac{1}{64}\text{nZ0}(2i-1,2j,2k)+\frac{1}{256}\text{nZ0}(2i-1,2j,2k-1)+$$
$$\frac{1}{256}\text{nZ0}(2i-1,2j,2k+1)+\frac{1}{64}\text{nZ0}(2i-1,2j-1,2k)+\frac{1}{256}\text{nZ0}(2i-1,2j-1,2k-1)+$$
$$\frac{1}{256}\text{nZ0}(2i-1,2j-1,2k+1)+\frac{1}{64}\text{nZ0}(2i+1,2j,2k)+\frac{1}{256}\text{nZ0}(2i+1,2j,2k-1)+$$
$$\frac{1}{256}\text{nZ0}(2i+1,2j,2k+1)+\frac{1}{64}\text{nZ0}(2i+1,2j-1,2k)+\frac{1}{256}\text{nZ0}(2i+1,2j-1,2k-1)+$$
$$\frac{1}{256}\text{nZ0}(2i+1,2j-1,2k+1)+\frac{1}{32}\text{nZm1}(2i,2j,2k)+\frac{1}{32}\text{nZm1}(2i,2j,2k+1)+$$
$$\frac{1}{32}\text{nZm1}(2i,2j-1,2k)+\frac{1}{32}\text{nZm1}(2i,2j-1,2k+1)+\frac{1}{128}\text{nZm1}(2i-1,2j,2k)+$$
$$\frac{1}{128}\text{nZm1}(2i-1,2j-1,2k)+\frac{1}{128}\text{nZm1}(2i-1,2j-1,2k+1)+\frac{1}{128}\text{nZm1}(2i+1,2j,2k)+$$
$$\frac{1}{128}\text{nZm1}(2i+1,2j,2k+1)+\frac{1}{128}\text{nZm1}(2i+1,2j-1,2k)+\frac{1}{128}\text{nZm1}(2i+1,2j-1,2k+1)+$$
$$\frac{1}{32}\text{nZp1}(2i,2j,2k)+\frac{1}{32}\text{nZp1}(2i,2j,2k-1)+\frac{1}{32}\text{nZp1}(2i,2j-1,2k)+$$
$$\frac{1}{32}\text{nZp1}(2i,2j-1,2k-1)+\frac{1}{128}\text{nZp1}(2i-1,2j,2k)+\frac{1}{128}\text{nZp1}(2i-1,2j,2k-1)+$$
$$\frac{1}{128}\text{nZp1}(2i-1,2j-1,2k)+\frac{1}{128}\text{nZp1}(2i-1,2j-1,2k-1)+\frac{1}{128}\text{nZp1}(2i+1,2j,2k)+$$
$$\frac{1}{128}\text{nZp1}(2i+1,2j,2k-1)+\frac{1}{128}\text{nZp1}(2i+1,2j-1,2k)+\frac{1}{128}\text{nZp1}(2i+1,2j-1,2k-1)+$$
$$\frac{1}{32}\text{seZ0}(2i,2j,2k)+\frac{1}{128}\text{seZ0}(2i,2j,2k-1)+\frac{1}{128}\text{seZ0}(2i,2j,2k+1)+$$
$$\frac{1}{32}\text{seZ0}(2i,2j+1,2k)+\frac{1}{128}\text{seZ0}(2i,2j+1,2k-1)+\frac{1}{128}\text{seZ0}(2i,2j+1,2k+1)+$$
$$\frac{1}{32}\text{seZ0}(2i-1,2j,2k)+\frac{1}{128}\text{seZ0}(2i-1,2j,2k-1)+\frac{1}{128}\text{seZ0}(2i-1,2j,2k+1)+$$
$$\frac{1}{32}\text{seZ0}(2i-1,2j+1,2k)+\frac{1}{128}\text{seZ0}(2i-1,2j+1,2k-1)+\frac{1}{128}\text{seZ0}(2i-1,2j+1,2k+1)+$$

$$\frac{1}{64}\mathrm{seZm1}(2i,2j,2k)+\frac{1}{64}\mathrm{seZm1}(2i,2j,2k+1)+\frac{1}{64}\mathrm{seZm1}(2i,2j+1,2k)+$$
$$\frac{1}{64}\mathrm{seZm1}(2i,2j+1,2k+1)+\frac{1}{64}\mathrm{seZm1}(2i-1,2j,2k)+\frac{1}{64}\mathrm{seZm1}(2i-1,2j,2k+1)+$$
$$\frac{1}{64}\mathrm{seZm1}(2i-1,2j+1,2k)+\frac{1}{64}\mathrm{seZm1}(2i-1,2j+1,2k+1)+\frac{1}{64}\mathrm{seZp1}(2i,2j,2k)+$$
$$\frac{1}{64}\mathrm{seZp1}(2i,2j,2k-1)+\frac{1}{64}\mathrm{seZp1}(2i,2j+1,2k)+\frac{1}{64}\mathrm{seZp1}(2i,2j+1,2k-1)+$$
$$\frac{1}{64}\mathrm{seZp1}(2i-1,2j,2k)+\frac{1}{64}\mathrm{seZp1}(2i-1,2j,2k-1)+\frac{1}{64}\mathrm{seZp1}(2i-1,2j+1,2k)+$$
$$\frac{1}{64}\mathrm{seZp1}(2i-1,2j+1,2k-1)+\frac{1}{32}\mathrm{swZ0}(2i,2j,2k)+\frac{1}{128}\mathrm{swZ0}(2i,2j,2k-1)+$$
$$\frac{1}{128}\mathrm{swZ0}(2i,2j,2k+1)+\frac{1}{32}\mathrm{swZ0}(2i,2j+1,2k)+\frac{1}{128}\mathrm{swZ0}(2i,2j+1,2k-1)+$$
$$\frac{1}{128}\mathrm{swZ0}(2i,2j+1,2k+1)+\frac{1}{32}\mathrm{swZ0}(2i+1,2j,2k)+\frac{1}{128}\mathrm{swZ0}(2i+1,2j,2k-1)+$$
$$\frac{1}{128}\mathrm{swZ0}(2i+1,2j,2k+1)+\frac{1}{32}\mathrm{swZ0}(2i+1,2j+1,2k)+\frac{1}{128}\mathrm{swZ0}(2i+1,2j+1,2k-1)+$$
$$\frac{1}{128}\mathrm{swZ0}(2i+1,2j+1,2k+1)+\frac{1}{64}\mathrm{swZm1}(2i,2j,2k)+\frac{1}{64}\mathrm{swZm1}(2i,2j,2k+1)+$$
$$\frac{1}{64}\mathrm{swZm1}(2i,2j+1,2k)+\frac{1}{64}\mathrm{swZm1}(2i,2j+1,2k+1)+\frac{1}{64}\mathrm{swZm1}(2i+1,2j,2k)+$$
$$\frac{1}{64}\mathrm{swZm1}(2i+1,2j,2k+1)+\frac{1}{64}\mathrm{swZm1}(2i+1,2j+1,2k)+\frac{1}{64}\mathrm{swZm1}(2i+1,2j+1,2k+1)+$$
$$\frac{1}{64}\mathrm{swZp1}(2i,2j,2k)+\frac{1}{64}\mathrm{swZp1}(2i,2j,2k-1)+\frac{1}{64}\mathrm{swZp1}(2i,2j+1,2k)+$$
$$\frac{1}{64}\mathrm{swZp1}(2i,2j+1,2k-1)+\frac{1}{64}\mathrm{swZp1}(2i+1,2j,2k)+\frac{1}{64}\mathrm{swZp1}(2i+1,2j,2k-1)+$$
$$\frac{1}{64}\mathrm{swZp1}(2i+1,2j+1,2k)+\frac{1}{64}\mathrm{swZp1}(2i+1,2j+1,2k-1)+\frac{1}{16}\mathrm{sZ0}(2i,2j,2k)+$$
$$\frac{1}{64}\mathrm{sZ0}(2i,2j,2k-1)+\frac{1}{64}\mathrm{sZ0}(2i,2j,2k+1)+\frac{1}{16}\mathrm{sZ0}(2i,2j+1,2k)+$$
$$\frac{1}{64}\mathrm{sZ0}(2i,2j+1,2k-1)+\frac{1}{64}\mathrm{sZ0}(2i,2j+1,2k+1)+\frac{1}{64}\mathrm{sZ0}(2i-1,2j,2k)+$$
$$\frac{1}{256}\mathrm{sZ0}(2i-1,2j,2k-1)+\frac{1}{256}\mathrm{sZ0}(2i-1,2j,2k+1)+\frac{1}{64}\mathrm{sZ0}(2i-1,2j+1,2k)+$$
$$\frac{1}{256}\mathrm{sZ0}(2i-1,2j+1,2k-1)+\frac{1}{256}\mathrm{sZ0}(2i-1,2j+1,2k+1)+\frac{1}{64}\mathrm{sZ0}(2i+1,2j,2k)+$$
$$\frac{1}{256}\mathrm{sZ0}(2i+1,2j,2k-1)+\frac{1}{256}\mathrm{sZ0}(2i+1,2j,2k+1)+\frac{1}{64}\mathrm{sZ0}(2i+1,2j+1,2k)+$$
$$\frac{1}{256}\mathrm{sZ0}(2i+1,2j+1,2k-1)+\frac{1}{256}\mathrm{sZ0}(2i+1,2j+1,2k+1)+\frac{1}{32}\mathrm{sZm1}(2i,2j,2k)+$$
$$\frac{1}{32}\mathrm{sZm1}(2i,2j,2k+1)+\frac{1}{32}\mathrm{sZm1}(2i,2j+1,2k)+\frac{1}{32}\mathrm{sZm1}(2i,2j+1,2k+1)+$$
$$\frac{1}{128}\mathrm{sZm1}(2i-1,2j,2k)+\frac{1}{128}\mathrm{sZm1}(2i-1,2j,2k+1)+\frac{1}{128}\mathrm{sZm1}(2i-1,2j+1,2k)+$$
$$\frac{1}{128}\mathrm{sZm1}(2i-1,2j+1,2k+1)+\frac{1}{128}\mathrm{sZm1}(2i+1,2j,2k)+\frac{1}{128}\mathrm{sZm1}(2i+1,2j,2k+1)+$$

$$\frac{1}{128}\mathrm{sZm1}(2i+1,2j+1,2k)+\frac{1}{128}\mathrm{sZm1}(2i+1,2j+1,2k+1)+\frac{1}{32}\mathrm{sZp1}(2i,2j,2k)+$$
$$\frac{1}{32}\mathrm{sZp1}(2i,2j,2k-1)+\frac{1}{32}\mathrm{sZp1}(2i,2j+1,2k)+\frac{1}{32}\mathrm{sZp1}(2i,2j+1,2k-1)+$$
$$\frac{1}{128}\mathrm{sZp1}(2i-1,2j,2k)+\frac{1}{128}\mathrm{sZp1}(2i-1,2j,2k-1)+\frac{1}{128}\mathrm{sZp1}(2i-1,2j+1,2k)+$$
$$\frac{1}{128}\mathrm{sZp1}(2i-1,2j+1,2k-1)+\frac{1}{128}\mathrm{sZp1}(2i+1,2j,2k)+\frac{1}{128}\mathrm{sZp1}(2i+1,2j,2k-1)+$$
$$\frac{1}{128}\mathrm{sZp1}(2i+1,2j+1,2k)+\frac{1}{128}\mathrm{sZp1}(2i+1,2j+1,2k-1)+\frac{1}{16}\mathrm{wZ0}(2i,2j,2k)+$$
$$\frac{1}{64}\mathrm{wZ0}(2i,2j,2k-1)+\frac{1}{64}\mathrm{wZ0}(2i,2j,2k+1)+\frac{1}{64}\mathrm{wZ0}(2i,2j-1,2k)+$$
$$\frac{1}{256}\mathrm{wZ0}(2i,2j-1,2k-1)+\frac{1}{256}\mathrm{wZ0}(2i,2j-1,2k+1)+\frac{1}{64}\mathrm{wZ0}(2i,2j+1,2k)+$$
$$\frac{1}{256}\mathrm{wZ0}(2i,2j+1,2k-1)+\frac{1}{256}\mathrm{wZ0}(2i,2j+1,2k+1)+\frac{1}{16}\mathrm{wZ0}(2i+1,2j,2k)+$$
$$\frac{1}{64}\mathrm{wZ0}(2i+1,2j,2k-1)+\frac{1}{64}\mathrm{wZ0}(2i+1,2j,2k+1)+\frac{1}{64}\mathrm{wZ0}(2i+1,2j-1,2k)+$$
$$\frac{1}{256}\mathrm{wZ0}(2i+1,2j-1,2k-1)+\frac{1}{256}\mathrm{wZ0}(2i+1,2j-1,2k+1)+\frac{1}{64}\mathrm{wZ0}(2i+1,2j+1,2k)+$$
$$\frac{1}{256}\mathrm{wZ0}(2i+1,2j+1,2k-1)+\frac{1}{256}\mathrm{wZ0}(2i+1,2j+1,2k+1)+\frac{1}{32}\mathrm{wZm1}(2i,2j,2k)+$$
$$\frac{1}{32}\mathrm{wZm1}(2i,2j,2k+1)+\frac{1}{128}\mathrm{wZm1}(2i,2j-1,2k)+\frac{1}{128}\mathrm{wZm1}(2i,2j-1,2k+1)+$$
$$\frac{1}{128}\mathrm{wZm1}(2i,2j+1,2k)+\frac{1}{128}\mathrm{wZm1}(2i,2j+1,2k+1)+\frac{1}{32}\mathrm{wZm1}(2i+1,2j,2k)+$$
$$\frac{1}{32}\mathrm{wZm1}(2i+1,2j,2k+1)+\frac{1}{128}\mathrm{wZm1}(2i+1,2j-1,2k)+\frac{1}{128}\mathrm{wZm1}(2i+1,2j-1,2k+1)+$$
$$\frac{1}{128}\mathrm{wZm1}(2i+1,2j+1,2k)+\frac{1}{128}\mathrm{wZm1}(2i+1,2j+1,2k+1)+\frac{1}{32}\mathrm{wZp1}(2i,2j,2k)+$$
$$\frac{1}{32}\mathrm{wZp1}(2i,2j,2k-1)+\frac{1}{128}\mathrm{wZp1}(2i,2j-1,2k)+\frac{1}{128}\mathrm{wZp1}(2i,2j-1,2k-1)+$$
$$\frac{1}{128}\mathrm{wZp1}(2i,2j+1,2k)+\frac{1}{128}\mathrm{wZp1}(2i,2j+1,2k-1)+\frac{1}{32}\mathrm{wZp1}(2i+1,2j,2k)+$$
$$\frac{1}{32}\mathrm{wZp1}(2i+1,2j,2k-1)+\frac{1}{128}\mathrm{wZp1}(2i+1,2j-1,2k)+\frac{1}{128}\mathrm{wZp1}(2i+1,2j-1,2k-1)+$$
$$\frac{1}{128}\mathrm{wZp1}(2i+1,2j+1,2k)+\frac{1}{128}\mathrm{wZp1}(2i+1,2j+1,2k-1) \tag{3.24}$$

The presentation of this entry must suffice to hint at the complexity of the underlying code for the Galerkin coarsening approach. There are 26 more entries like this in total, all of them nonzero. This is, in fact, one of the drawbacks of the GCA. The DCA would deliver the same 7-point stencil structure as the fine grid operator A_h, i.e., 7 non–zero entries while the remaining 20 entries would be zero. However, the important coupling for Cepheids would not be present, as indicated before.

The subroutine to calculate the coarse grid operator with the GCA has about 5000 lines of code in ANTARES and must be called once for each multigrid level.

3.2.5.2 The Coarse Grid Solver

We can now come back to the problem of solving

$$A_{2h}\boldsymbol{e}_{2h} = \boldsymbol{r}_{2h}. \tag{3.25}$$

on the coarse grid. Having derived an explicit form of the operator A_{2h} the only thing that remains to be chosen is the method with which to solve (3.25). Happenhofer, (2014) has used the Conjugate Gradient method as the coarse–grid solver in her 2D solver. We did the same for three dimensions. Implementing the CG method for three dimensions did not pose much of a problem because of its relatively simple algorithmic structure. Details on the development of a CG solver in the ANTARES framework can be found in Grimm-Strele, (2010). The system is solved up to a sufficiently small error tolerance ϵ_{coarse}.

3.2.6 Restricting the Error to the Finer Grid and Correcting the Approximation

6. Transfer the error $\boldsymbol{e}_{2h}$ to the finer grid G_h to obtain $\boldsymbol{e}_h$.

The error obtained on the coarsest grid is transferred to the finer grid with the help of the restriction operator from section 3.2.4:

$$\boldsymbol{e}_h = R\boldsymbol{e}_{2h} \tag{3.26}$$

3.2.7 Correcting the Approximation and Final Smoothing

7. Correct the fine–grid approximation $\boldsymbol{v}_h^{(\nu_1)}$ by adding the error: $\boldsymbol{v}_h^{(\nu_1+1)} = \boldsymbol{v}_h^{(\nu_1)} + \boldsymbol{e}_h$.

8. Relax $A_h\boldsymbol{v}_h^{(\nu_1+1)} = \boldsymbol{f}_h$ ν_2 times on the finest grid G_h to obtain the improved value $\boldsymbol{v}_h^{(\nu_1+1+\nu_2)}$.

The last two steps do not need any further explanation. The smoother in step 8 is the same one that was used in step 1, i.e., lexicographic Gauss–Seidel. This completes one two–grid correction sweep. The next natural step in the development of a multigrid solver is to use more than two grids, which we turn to next.

3.2.8 From Two–grid to Multigrid

The multi–grid method works principally exactly like the two–grid method. The only difference is that the restriction to a finer grid (and prolongation to the coarse one, of course) does not take place only once, but several times. There are different methods of how exactly to traverse the different grids (V–cycle, W–cycle, F–cycle,...). Examples are shown in figure 3.2. The interested reader can find more details in the literature. For our purposes is suffices to say that the implemented three–dimensional multigrid solver can use anything from two to five grid levels and has V–cycling and W–cycling implemented. To see which cycling yields better results, the reader is referred to section 4.1.3.

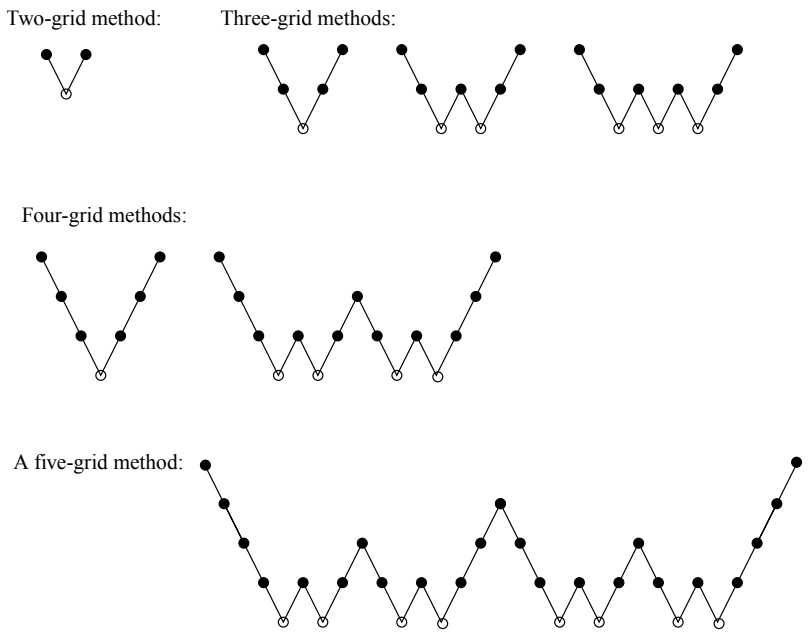


Figure 3.2: A few examples for different multigrid cycles (from Trottenberg, Oosterlee, and Schüller, 2001).

3.3 Non-Linear Multigrid

We now turn towards the case where the elliptic equation to be solved is nonlinear, as in

$$-\nabla\cdot(\kappa(u)\nabla u(\boldsymbol{x}))+\xi(u)=f(\boldsymbol{x}) \tag{3.27}$$

with suitable boundary conditions.

There are two ways to treat a nonlinear problem with multigrid methods: the full approximation storage scheme (FAS), which is allows the development of a completely matrix- and derivative-free nonlinear solver and the Multigrid–Newton procedure, which basically is a linearization of the nonlinear equation with a subsequent use of the linear multigrid procedure to solve the resulting linear system. Happenhofer, (2014) tested both approaches for the two–dimensional solver and has come to the following conclusion:

"The advantage of the MG-FAS method clearly lies in the possibility of creating a completely matrix- and derivative-free nonlinear solver. Apart from having good convergence properties, it allows a memory efficient implementation. However, the nonlinear formulation requires the determination of the nonlinear operator anew on every grid level and multiple times on the coarsest grid level, depending on the solution method employed. Since the determination of the nonlinear operator A entails a call to the EOS, which is, in case that a model employing realistic microphysics is considered, computationally expensive [...], this is considered a disadvantage. Furthermore, if the fixed-point iteration is used, the additional timestep restriction [...] has to be taken into account. [...] Based on the estimation of the impact of the call to the EOS on the solution time, the MG-FAS approach is discarded and the Multigrid-Newton technique is adopted to solve the nonlinear equation [...] arising in the context of IMEX Runge-Kutta methods."

Since the nonlinear three–dimensional multigrid solver is to be used for the same physical simulations as the two–dimensional one they also entail a call to the EOS. Thus, following Happenhofer's argumentation, the 3D nonlinear solver was coded with the Multigrid–Newton method.

3.3.1 The Newton Scheme

Details on the Newton scheme for multigrid methods can be found,e.g., in Köckler, (2012). A remark: the Newton procedure does not guarantee convergence, i.e., the starting guess must be rather accurate in order for the method to converge. In ANTARES, the starting value stems from a previous Runge–Kutta time step so it is reasonable so suppose a sufficiently accurate value.

The starting point of the Newton procedure is the Taylor expansion of the nonlinear operator $\mathscr{A}(\boldsymbol{u})$,

$$\mathscr{A}(\boldsymbol{u})=\mathscr{A}(\boldsymbol{u}^{(r)})+J(\boldsymbol{u}^{(r)})(\boldsymbol{u}-\boldsymbol{u}^{(r)})+\mathscr{O}(\boldsymbol{u}-\boldsymbol{u}^{(r)})^2, \tag{3.28}$$

where J is the Jacobian matrix

$$J(\boldsymbol{u})=\frac{\partial\mathscr{A}(\boldsymbol{u})}{\partial\boldsymbol{u}} \tag{3.29}$$

and r is the iteration count. Neglecting terms of second order and setting $\mathscr{A}(\boldsymbol{u}) = \boldsymbol{b}$ and $\boldsymbol{u} = \boldsymbol{u}^{(r+1)}$ leads to

$$\boldsymbol{b} = \mathscr{A}(\boldsymbol{u}^{(r)}) + J(\boldsymbol{u}^{(r)})(\boldsymbol{u}^{(r+1)} - \boldsymbol{u}^{(r)}). \tag{3.30}$$

Defining $\boldsymbol{d}^{(r)} \equiv \boldsymbol{u}^{(r+1)} - \boldsymbol{u}^{(r)}$ leads to the iterative scheme

$$\begin{aligned} J(\boldsymbol{u}^{(r)})\boldsymbol{d}^{(r)} &= \boldsymbol{b} - \mathscr{A}(\boldsymbol{u}^{(r)}) \\ \boldsymbol{u}^{(r+1)} &= \boldsymbol{u}^{(r)} + \boldsymbol{d}^{(r)}, \quad r = 0,1,... \end{aligned} \tag{3.31}$$

This is now a *linear* system of equations which can be solved with a linear solver. For our purposes this is the linear multigrid solver that was developed in section 3.2. We now have a nested iterative scheme with the outer iteration (with counter r) being the linearization by the Newton method and the inner iterative scheme being the linear multigrid solver.

To be able to solve the system, we first need an explicit form for the Jacobian matrix $J(\boldsymbol{u})$ and the right hand side $\boldsymbol{b} - \mathscr{A}(\boldsymbol{u})$, calculated at the current iteration step r. To obtain these we first need to calculate $\mathscr{A}(\boldsymbol{u})$, to which we turn now.

3.3.2 Calculation of the Jacobian Matrix

Now that we know what $\mathscr{A}$ looks like, we can calculate the Jacobian matrix. It is defined as

$$J_{ijk,lmn} = \frac{\partial[\mathscr{A}(\boldsymbol{u})]_{ijk}}{\partial u_{lmn}} \tag{3.32}$$

Using definition (2.184) for the nonlinear operator $\mathscr{A}(\boldsymbol{u})$, this results in

$$\begin{aligned} J_{ijk,lmn} = & u_{ijk}\frac{\partial \texttt{dd\_z0(i,j,k)}}{\partial u_{lmn}} + \texttt{dd\_z0(i,j,k)}\frac{\partial u_{ijk}}{\partial u_{lmn}} \\ & + u_{ijk-1}\frac{\partial \texttt{dd\_zm1(i,j,k)}}{\partial u_{lmn}} + \texttt{dd\_zm1(i,j,k)}\frac{\partial u_{ijk-1}}{\partial u_{lmn}} \\ & + u_{ijk+1}\frac{\partial \texttt{dd\_zp1(i,j,k)}}{\partial u_{lmn}} + \texttt{dd\_zp1(i,j,k)}\frac{\partial u_{ijk+1}}{\partial u_{lmn}} \\ & + u_{i+1jk}\frac{\partial \texttt{e\_z0(i,j,k)}}{\partial u_{lmn}} + \texttt{e\_z0(i,j,k)}\frac{\partial u_{i+1jk}}{\partial u_{lmn}} \\ & + u_{i-1jk}\frac{\partial \texttt{w\_z0(i,j,k)}}{\partial u_{lmn}} + \texttt{w\_z0(i,j,k)}\frac{\partial u_{i-1jk}}{\partial u_{lmn}} \\ & + u_{ij+1k}\frac{\partial \texttt{n\_z0(i,j,k)}}{\partial u_{lmn}} + \texttt{n\_z0(i,j,k)}\frac{\partial u_{ij+1k}}{\partial u_{lmn}} \\ & + u_{ij-1k}\frac{\partial \texttt{s\_z0(i,j,k)}}{\partial u_{lmn}} + \texttt{s\_z0(i,j,k)}\frac{\partial u_{ij-1k}}{\partial u_{lmn}} \\ & + h_x h_y h_z \cdot \frac{\partial \xi_{i,j,k}}{\partial u_{lmn}} \end{aligned} \tag{3.33}$$

The stencil entries `dd_zm1(i,j,k)`, `s_z0(i,j,k)`, `e_z0(i,j,k)`, `dd_z0(i,j,k)`, `w_z0(i,j,k)`, `n_z0(i,j,k)`and `dd_zm1(i,j,k)`were calculated in (2.170)-(2.176). To wit, they are

$$\texttt{dd\_z0(i,j,k)} = \frac{h_y h_z \kappa_{i,j,k}}{h_x} + \frac{h_x h_z \kappa_{i,j,k}}{h_y} + \frac{h_x h_y \kappa_{i,j,k}}{h_z}$$

$$+\frac{h_x h_y \kappa_{i,j,k-1}}{2h_z}+\frac{h_x h_y \kappa_{i,j,k+1}}{2h_z}+\frac{h_x h_z \kappa_{i,j-1,k}}{2h_y} \tag{3.34}$$
$$+\frac{h_x h_z \kappa_{i,j+1,k}}{2h_y}+\frac{h_y h_z \kappa_{i-1,j,k}}{2h_x}+\frac{h_y h_z \kappa_{i+1,j,k}}{2h_x},$$

$$\texttt{e\_z0(i,j,k)} = -\frac{h_y h_z(\kappa_{i,j,k}+\kappa_{i+1,j,k})}{2h_x}, \tag{3.35}$$

$$\texttt{w\_z0(i,j,k)} = -\frac{h_y h_z(\kappa_{i,j,k}+\kappa_{i-1,j,k})}{2h_x}, \tag{3.36}$$

$$\texttt{n\_z0(i,j,k)} = -\frac{h_x h_z(\kappa_{i,j,k}+\kappa_{i,j+1,k})}{2h_y}, \tag{3.37}$$

$$\texttt{s\_z0(i,j,k)} = -\frac{h_x h_z(\kappa_{i,j,k}+\kappa_{i,j-1,k})}{2h_y}, \tag{3.38}$$

$$\texttt{dd\_zp1(i,j,k)} = -\frac{h_x h_y(\kappa_{i,j,k}+\kappa_{i,j,k+1})}{2h_z}, \tag{3.39}$$

$$\texttt{dd\_zm1(i,j,k)} = -\frac{h_x h_y(\kappa_{i,j,k}+\kappa_{i,j,k-1})}{2h_z}. \tag{3.40}$$

With this we can calculate the derivatives in (3.33) and obtain

$$\begin{aligned}
J_{ijk,lmn} = &-\frac{h_x h_y u_{ijk-1}\left(\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}+\frac{\partial \kappa_{ijk-1}}{\partial u_{lmn}}\right)}{2h_z}-\frac{h_x h_y u_{ijk+1}\left(\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}+\frac{\partial \kappa_{ijk+1}}{\partial u_{lmn}}\right)}{2h_z}\\
&-\frac{h_x h_z u_{ij-1k}\left(\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}+\frac{\partial \kappa_{ij-1k}}{\partial u_{lmn}}\right)}{2h_y}-\frac{h_x h_z u_{ij+1k}\left(\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}+\frac{\partial \kappa_{ij+1k}}{\partial u_{lmn}}\right)}{2h_y}\\
&-\frac{h_y h_z u_{i-1jk}\left(\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}+\frac{\partial \kappa_{i-1jk}}{\partial u_{lmn}}\right)}{2h_x}-\frac{h_y h_z u_{i+1jk}\left(\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}+\frac{\partial \kappa_{i+1jk}}{\partial u_{lmn}}\right)}{2h_x}\\
&+\left(h_x h_y h_z\frac{\partial \xi_{ijk}}{\partial u_{lmn}}+\frac{h_x h_y\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}}{h_z}+\frac{h_x h_z\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}}{h_y}+\frac{h_y h_z\frac{\partial \kappa_{ijk}}{\partial u_{lmn}}}{h_x}+\frac{h_x h_y\frac{\partial \kappa_{ijk-1}}{\partial u_{lmn}}}{2h_z}\right.\\
&\left.+\frac{h_x h_y\frac{\partial \kappa_{ijk+1}}{\partial u_{lmn}}}{2h_z}+\frac{h_x h_z\frac{\partial \kappa_{ij-1k}}{\partial u_{lmn}}}{2h_y}+\frac{h_x h_z\frac{\partial \kappa_{ij+1k}}{\partial u_{lmn}}}{2h_y}+\frac{h_y h_z\frac{\partial \kappa_{i-1jk}}{\partial u_{lmn}}}{2h_x}+\frac{h_y h_z\frac{\partial \kappa_{i+1jk}}{\partial u_{lmn}}}{2h_x}\right)u_{ijk}\\
&+\left(\xi_{ijk}h_x h_y h_z+\frac{h_x h_y \kappa_{ijk}}{h_z}+\frac{h_x h_z \kappa_{ijk}}{h_y}+\frac{h_y h_z \kappa_{ijk}}{h_x}+\frac{h_x h_y \kappa_{ijk-1}}{2h_z}\right.\\
&\left.+\frac{h_x h_y \kappa_{ijk+1}}{2h_z}+\frac{h_x h_z \kappa_{ij-1k}}{2h_y}+\frac{h_x h_z \kappa_{ij+1k}}{2h_y}+\frac{h_y h_z \kappa_{i-1jk}}{2h_x}+\frac{h_y h_z \kappa_{i+1jk}}{2h_x}\right)\frac{\partial u_{ijk}}{\partial u_{lmn}}\\
&-\frac{h_x h_y(\kappa_{ijk}+\kappa_{ijk-1})\frac{\partial u_{ijk-1}}{\partial u_{lmn}}}{2h_z}-\frac{h_x h_y(\kappa_{ijk}+\kappa_{ijk+1})\frac{\partial u_{ijk+1}}{\partial u_{lmn}}}{2h_z}\\
&-\frac{h_x h_z(\kappa_{ijk}+\kappa_{ij-1k})\frac{\partial u_{ij-1k}}{\partial u_{lmn}}}{2h_y}-\frac{h_x h_z(\kappa_{ijk}+\kappa_{ij+1k})\frac{\partial u_{ij+1k}}{\partial u_{lmn}}}{2h_y}\\
&-\frac{h_y h_z(\kappa_{ijk}+\kappa_{i-1jk})\frac{\partial u_{i-1jk}}{\partial u_{lmn}}}{2h_x}-\frac{h_y h_z(\kappa_{ijk}+\kappa_{i+1jk})\frac{\partial u_{i+1jk}}{\partial u_{lmn}}}{2h_x}.
\end{aligned} \tag{3.41}$$

This is as far as we can go without knowing the explicit dependencies of κ and ξ on u and without any further knowledge of the underlying physics. As it turns out, we can use an essential simplification for the applications we are interested in.

3.3.3 Simplifications by the Local Nature of κ and ξ

The three–dimensional nonlinear solver which was developed is used primarily for simulations of semiconvection with a possible later addition of simulations of Cepheid–like pulsating stars. In these simulations, $\kappa(u)$ and $\xi(u)$ stem from micro-physical processes. Depending on the specific kind of simulation that is performed, they be the opacity, radiative diffusivity or the concentration diffusivity among others. What is common to all of the choices for $\kappa(u)$ and ξ is that they stem from micro-physical processes. This means that they stem from very fast processes on microscopic scales. Since we are describing dynamic processes with the equations of hydrodynamics, the time scales are significantly larger than the time scales on which the micro-processes change. This allows us to the separation of scales to be able to average the fast micro-processes and use these averages for the discussion of their influence on the dynamics of the whole system.

A consequence of that is that these averages are local quantities in that they have no explicit dependence on $\boldsymbol{x}$ which means that the derivatives with respect to all other points are zero:

$$\frac{\partial \kappa_{ijk}}{\partial u_{lmn}} = \delta_{ijk,lmn} \quad \text{and} \quad \frac{\partial \xi_{ijk}}{\partial u_{lmn}} = \delta_{ijk,lmn} \tag{3.42}$$

That means that many terms in the entries of the Jacobian matrix are zero which allows us to write down the explicit form of the entries of the Jacobian matrix in the next section.

3.3.3.1 The Final Form of the Jacobian Matrix

We can now evaluate the final form of the Jacobian matrix for our purposes. It has the same structure as the nonlinear operator has, i.e., 7 nonzero entries. They read

$$JwZ0 = -\frac{h_y h_z \left(\kappa_{ijk} + (u_{i-1jk} - u_{ijk})\frac{\partial \kappa_{i-1jk}}{\partial u_{i-1jk}} + \kappa_{i-1jk}\right)}{2h_x} \tag{3.43}$$

$$JeZ0 = -\frac{h_y h_z \left(\kappa_{ijk} + (u_{i+1jk} - u_{ijk})\frac{\partial \kappa_{i+1jk}}{\partial u_{i+1jk}} + \kappa_{i+1jk}\right)}{2h_x} \tag{3.44}$$

$$JsZ0 = -\frac{h_x h_z \left(\kappa_{ijk} + (u_{ij-1k} - u_{ijk})\frac{\partial \kappa_{ij-1k}}{\partial u_{ij-1k}} + \kappa_{ij-1k}\right)}{2h_y} \tag{3.45}$$

$$JnZ0 = -\frac{h_x h_z \left(\kappa_{ijk} + (u_{ij+1k} - u_{ijk})\frac{\partial \kappa_{ij+1k}}{\partial u_{ij+1k}} + \kappa_{ij+1k}\right)}{2h_y} \tag{3.46}$$

$$JddZm1 = -\frac{h_x h_y \left(\kappa_{ijk} + (u_{ijk-1} - u_{ijk})\frac{\partial \kappa_{ijk-1}}{\partial u_{ijk-1}} + \kappa_{ijk-1}\right)}{2h_z} \tag{3.47}$$

$$JddZp1 = -\frac{h_x h_y \left(\kappa_{ijk} + (u_{ijk+1} - u_{ijk})\frac{\partial \kappa_{ijk+1}}{\partial u_{ijk+1}} + \kappa_{ijk+1}\right)}{2h_z} \tag{3.48}$$

$$\begin{aligned} JddZ0 = \frac{1}{2h_x h_y h_z}\Bigg\{ & 2h_x^2 h_y^2 h_z^2 u_{ijk}\frac{\partial \xi_{ijk}}{\partial u_{ijk}} + 2\xi_{ijk} h_x^2 h_y^2 h_z^2 \\ & + \frac{\partial \kappa_{ijk}}{\partial u_{ijk}}\Big[h_x^2\left(h_y^2(2u_{ijk} - u_{ijk-1} - u_{ijk+1}) + h_z^2(2u_{ijk} - u_{ij-1k} - u_{ij+1k})\right) \\ & + h_y^2 h_z^2(2u_{ijk} - u_{i-1jk} - u_{i+1jk})\Big] + 2h_x^2 h_y^2 \kappa_{ijk} + h_x^2 h_y^2 \kappa_{ijk-1} \\ & + h_x^2 h_y^2 \kappa_{ijk+1} + 2h_x^2 h_z^2 \kappa_{ijk} + h_x^2 h_z^2 \kappa_{ij-1k} + h_x^2 h_z^2 \kappa_{ij+1k} \\ & + 2h_y^2 h_z^2 \kappa_{ijk} + h_y^2 h_z^2 \kappa_{i-1jk} + h_y^2 h_z^2 \kappa_{i+1jk}\Bigg\} \end{aligned} \tag{3.49}$$

3.3.3.2 Lower Boundary Terms

The calculation for the boundaries is analogous to the previous calculation in the domain. We only give the results here. The stencil entries of the nonlinear operator for the lower boundary terms are

$$\begin{aligned} \text{ddZ0LB} = & 0.5\xi_{ijk} h_x h_y h_z + \frac{h_y h_z \kappa_{ijk}}{2h_x} + \frac{h_x h_z \kappa_{ijk}}{2h_y} \\ & + \frac{h_x h_y \kappa_{ijk}}{2h_z} + \frac{h_x h_y \kappa_{ijk-1}}{4h_z} + \frac{h_x h_y \kappa_{ijk+1}}{4h_z} \\ & + \frac{h_x h_z \kappa_{ij-1k}}{4h_y} + \frac{h_x h_z \kappa_{ij+1k}}{4h_y} + \frac{h_y h_z \kappa_{i+1jk}}{2h_x} \end{aligned} \tag{3.50}$$

$$\text{eZ0LB} = -\frac{h_y h_z(\kappa_{ijk} + \kappa_{i+1jk})}{2h_x} \tag{3.51}$$

$$\text{wZ0LB} = 0 \tag{3.52}$$

$$\text{nZ0LB} = -\frac{h_x h_z(\kappa_{ijk} + \kappa_{ij+1k})}{4h_y} \tag{3.53}$$

$$\text{sZ0LB} = -\frac{h_x h_z(\kappa_{ijk} + \kappa_{ij-1k})}{4h_y} \tag{3.54}$$

$$\text{ddZp1LB} = -\frac{h_x h_y(\kappa_{ijk} + \kappa_{ijk+1})}{4h_z} \tag{3.55}$$

$$\text{ddZm1LB} = -\frac{h_x h_y(\kappa_{ijk} + \kappa_{ijk-1})}{4h_z}. \tag{3.56}$$

The resulting entries of the Jacobian matrix are

$$JddZm1LB = -\frac{h_x h_y \left(\kappa_{ijk} + (u_{ijk-1} - u_{ijk})\frac{\partial \kappa_{ijk-1}}{\partial u_{ijk-1}} + \kappa_{ijk-1}\right)}{4h_z} \tag{3.57}$$

$$JsZ0LB = -\frac{h_x h_z \left(\kappa_{ijk} + (u_{ij-1k} - u_{ijk})\frac{\partial \kappa_{ij-1k}}{\partial u_{ij-1k}} + \kappa_{ij-1k}\right)}{4h_y} \tag{3.58}$$

$$JwZ0LB = 0 \tag{3.59}$$

$$\begin{aligned} JddZ0LB = &\frac{1}{h_x h_y h_z} 0.5 h_x^2 h_y^2 h_z^2 u_{ijk} \frac{\partial \xi_{ijk}}{\partial u_{ijk}} + 0.5 \xi_{ijk} h_x^2 h_y^2 h_z^2 \\ &+ \frac{\partial \kappa_{ijk}}{\partial u_{ijk}} \Big(h_x^2 \Big(h_y^2 (0.5 u_{ijk} - 0.25 u_{ijk-1} - 0.25 u_{ijk+1}) \\ &+ h_z^2 (0.5 u_{ijk} - 0.25 u_{ij-1k} - 0.25 u_{ij+1k}) \Big) + h_y^2 h_z^2 (0.5 u_{ijk} - 0.5 u_{i+1jk}) \Big) \\ &+ 0.5 h_x^2 h_y^2 \kappa_{ijk} + 0.25 h_x^2 h_y^2 \kappa_{ijk-1} + 0.25 h_x^2 h_y^2 \kappa_{ijk+1} \\ &+ 0.5 h_x^2 h_z^2 \kappa_{ijk} + 0.25 h_x^2 h_z^2 \kappa_{ij-1k} \\ &+ 0.25 h_x^2 h_z^2 \kappa_{ij+1k} + 0.5 h_y^2 h_z^2 \kappa_{ijk} + 0.5 h_y^2 h_z^2 \kappa_{i+1jk} \end{aligned} \tag{3.60}$$

$$JeZ0LB = -\frac{h_y h_z \left(\kappa_{ijk} + (u_{i+1jk} - u_{ijk}) \frac{\partial \kappa_{i+1jk}}{\partial u_{i+1jk}} + \kappa_{i+1jk} \right)}{2 h_x} \tag{3.61}$$

$$JnZ0LB = -\frac{h_x h_z \left(\kappa_{ijk} + (u_{ij+1k} - u_{ijk}) \frac{\partial \kappa_{ij+1k}}{\partial u_{ij+1k}} + \kappa_{ij+1k} \right)}{4 h_y} \tag{3.62}$$

$$JddZp1LB = -\frac{h_x h_y \left(\kappa_{ijk} + (u_{ijk+1} - u_{ijk}) \frac{\partial \kappa_{ijk+1}}{\partial u_{ijk+1}} + \kappa_{ijk+1} \right)}{4 h_z} \tag{3.63}$$

3.3.3.3 Upper Boundary Terms

The stencil entries of the nonlinear operator at the upper boundary read

$$\text{ddZm1UB} = -\frac{h_x h_y (\kappa_{ijk} + \kappa_{ijk-1})}{4 h_z} \tag{3.64}$$

$$\text{sZ0UB} = -\frac{h_x h_z (\kappa_{ijk} + \kappa_{ij-1k})}{4 h_y} \tag{3.65}$$

$$\text{wZ0UB} = -\frac{h_y h_z (\kappa_{ijk} + \kappa_{i-1jk})}{2 h_x} \tag{3.66}$$

$$\begin{aligned} \text{ddZ0UB} = &0.5 \xi_{ijk} h_x h_y h_z + \frac{h_y h_z \kappa_{ijk}}{2 h_x} + \frac{h_x h_z \kappa_{ijk}}{2 h_y} \\ &+ \frac{h_x h_y \kappa_{ijk}}{2 h_z} + \frac{h_x h_y \kappa_{ijk-1}}{4 h_z} + \frac{h_x h_y \kappa_{ijk+1}}{4 h_z} \\ &+ \frac{h_x h_z \kappa_{ij-1k}}{4 h_y} + \frac{h_x h_z \kappa_{ij+1k}}{4 h_y} + \frac{h_y h_z \kappa_{i-1jk}}{2 h_x} \end{aligned} \tag{3.67}$$

$$\text{eZ0UB} = 0 \tag{3.68}$$

$$\text{nZ0UB} = -\frac{h_x h_z (\kappa_{ijk} + \kappa_{ij+1k})}{4 h_y} \tag{3.69}$$

$$\text{ddZp1UB} = -\frac{h_x h_y (\kappa_{ijk} + \kappa_{ijk+1})}{4 h_z}. \tag{3.70}$$

The resulting entries of the Jacobian matrix are

$$JddZm1UB = -\frac{h_x h_y \left(\kappa_{ijk} + (u_{ijk-1} - u_{ijk}) \frac{\partial \kappa_{ijk-1}}{\partial u_{ijk-1}} + \kappa_{ijk-1} \right)}{4 h_z} \tag{3.71}$$

$$JsZ0UB = -\frac{h_x h_z \left(\kappa_{ijk} + (u_{ij-1k} - u_{ijk})\frac{\partial \kappa_{ij-1k}}{\partial u_{ij-1k}} + \kappa_{ij-1k}\right)}{4h_y} \tag{3.72}$$

$$JwZ0UB = -\frac{h_y h_z \left(\kappa_{ijk} + (u_{i-1jk} - u_{ijk})\frac{\partial \kappa_{i-1jk}}{\partial u_{i-1jk}} + \kappa_{i-1jk}\right)}{2h_x} \tag{3.73}$$

$$\begin{aligned} JddZ0UB = &\frac{1}{h_x h_y h_z} 0.5 h_x^2 h_y^2 h_z^2 u_{ijk} \frac{\partial \xi_{ijk}}{\partial u_{ijk}} + 0.5 \xi_{ijk} h_x^2 h_y^2 h_z^2 \\ &+ \frac{\partial \kappa_{ijk}}{\partial u_{ijk}} \Big(h_x^2 \Big(h_y^2 (0.5 u_{ijk} - 0.25 u_{ijk-1} - 0.25 u_{ijk+1}) \\ &+ h_z^2 (0.5 u_{ijk} - 0.25 u_{ij-1k} - 0.25 u_{ij+1k}) \Big) + h_y^2 h_z^2 (0.5 u_{ijk} - 0.5 u_{i-1jk}) \Big) \\ &+ 0.5 h_x^2 h_y^2 \kappa_{ijk} + 0.25 h_x^2 h_y^2 \kappa_{ijk-1} + 0.25 h_x^2 h_y^2 \kappa_{ijk+1} \\ &+ 0.5 h_x^2 h_z^2 \kappa_{ijk} + 0.25 h_x^2 h_z^2 \kappa_{ij-1k} + 0.25 h_x^2 h_z^2 \kappa_{ij+1k} \\ &+ 0.5 h_y^2 h_z^2 \kappa_{ijk} + 0.5 h_y^2 h_z^2 \kappa_{i-1jk} \end{aligned} \tag{3.74}$$

$$JeZ0UB = 0 \tag{3.75}$$

$$JnZ0UB = -\frac{h_x h_z \left(\kappa_{ijk} + (u_{ij+1k} - u_{ijk})\frac{\partial \kappa_{ij+1k}}{\partial u_{ij+1k}} + \kappa_{ij+1k}\right)}{4h_y} \tag{3.76}$$

$$JddZp1UB = -\frac{h_x h_y \left(\kappa_{ijk} + (u_{ijk+1} - u_{ijk})\frac{\partial \kappa_{ijk+1}}{\partial u_{ijk+1}} + \kappa_{ijk+1}\right)}{4h_z} \tag{3.77}$$

This completes the derivation of the Jacobian matrix. With this, all components of the nonlinear solver have been derived and we can solve the system (3.31).

After having derived the necessary components for the discretization, linearization and solution of the systems of equations the next step is to implement them into the ANTARES framework. This is the step which demanded a great amount of my time. Getting the algorithms in the correct form and implementing them into the existing code has been a major part of my work. Unfortunately, I cannot represent the difficulty of the coding part of the thesis here.

After having implemented the multigrid solver it is necessary to thoroughly test it and to determine if the discretization accuracy of the finite elements method is met. This is done in the following chapter.

3.4 Summary

In this chapter, we have given a few general introductory remarks about multigrid methods in general and have given information about which components have been chosen for our implementation of a multigrid solver for the three–dimensional, (non–)linear, generalized Helmholtz equation in the ANTARES framework. Finally, we have derived the chosen components for the linear and nonlinear case.

CHAPTER 4

TESTING THE MULTIGRID SOLVER

After having implemented the solver into the ANTARES framework, we now have to validate its accuracy in order for it to be used in production simulations. This section serves two purposes:

- to test the MG solver with equations for which the analytical solutions are known in order to be able to make the statement “the solver is producing correct results” and
- to demonstrate and explain to future users of ANTARES the impact of some parameters that can be adjusted when using the solver. Those parameters are:
  - the number of points on the coarse grid,
  - the number of multigrid levels,
  - the number of pre- and post-smoothing steps ν_1 and ν_2,
  - the multigrid cycling used: V–cycling or W–cycling
  - and the stopping criterion or accuracy to which the solver approximates the solution.

The chapter is subdivided as follows:

- We start with tests of the linear multigrid solver with different boundary conditions:
  - Homogeneous Dirichlet boundary conditions (section 4.1): in this section, various combinations of the parameters mentioned above are tested and their influence on convergence rate, wall clock time and multigrid cycles needed to achieve convergence is investigated;
  - non-homogeneous Neumann boundary conditions (section 4.2);
  - non-homogeneous Dirichlet boundary conditions (section 4.3).

- We then verify the correctness of parallelization done and investigate the strong and weak scaling behavior (section 4.4).
- Next, the nonlinear solver is tested with the following boundary conditions:
  - non-homogeneous Dirichlet boundary conditions (section 4.5);
  - non-homogeneous Neumann boundary conditions (section 4.6).
- Finally, the correctness of parallelization is verified in the nonlinear case (section 4.7)

A short remark on accuracy: as Trottenberg, Oosterlee, and Schüller, (2001) mention: in practice, it is usually not necessary to reduce the residual by a factor of 10^{-12}, as is done in the following investigations. Convergence to discretization accuracy $O(h^2)$ is sufficient in most cases and, naturally, considerably faster. Because we are interested in studying the implemented solver with analytic problems here, we are not so much interested in the speed of the solver but in the asymptotic convergence properties. Hence, we investigate the behavior of the residual to much smaller scales than it is actually necessary.

The test cases are of the following structure: let $\Omega = [0,2\pi] \times [0,2\pi] \times [0,2\pi]$. Each of the linear test equations is of form

$$-\nabla \cdot (\kappa(\boldsymbol{x}) \nabla u(\boldsymbol{x})) + \xi(\boldsymbol{x}) u = f(\boldsymbol{x}) \tag{4.1}$$

with $\boldsymbol{x} \in \Omega$; $\kappa, \xi : \Omega \to \mathbb{R}^+$; $f, u : \Omega \to \mathbb{R}$ and each of the nonlinear test equations is of form

$$-\nabla \cdot (\kappa(u) \nabla u(\boldsymbol{x})) + \xi(u) = f(\boldsymbol{x}) \tag{4.2}$$

with $\boldsymbol{x} \in \Omega$; $\kappa, \xi : \mathbb{R} \to \mathbb{R}^+$; $f, u : \Omega \to \mathbb{R}$. The fact that κ and ξ are positive in Ω ensures strong ellipticity.

4.1 Test Case 1 — Linear Equation with Homogeneous Dirichlet B.C.

The problem considered is

$$-\nabla \cdot (x \nabla u(\boldsymbol{x})) + x\, u(\boldsymbol{x}) = 4x \sin(x) \sin(y) \sin(z) - \cos(x) \sin(y) \sin(z) \tag{4.3}$$

with the homogeneous Dirichlet boundary conditions

$$\begin{aligned} u(0,y,z) &= u(2\pi,y,z) = 0, \\ u(x,0,z) &= u(x,2\pi,z), \\ u(x,y,0) &= u(x,y,2\pi). \end{aligned} \tag{4.4}$$

The exact solution is

$$u(x,y,z) = \sin(x)\sin(y)\sin(z). \tag{4.5}$$

The numerical results of the test can be found in appendix A.1.1. We summarize the most interesting results here.

4.1.1 Calculating the Order of Accuracy

To investigate the accuracy of the solver we are following the procedure which LeVeque, (2007) outlines: first, the order of accuracy of the discretization is confirmed. It is expected to be two because that is the order of accuracy of the discretization with the finite element method discussed in section 2. When we have the exact solution u_{exact} at each grid point (which we have) we can estimate the absolute numerical error directly by calculating the point-wise error vector

$$\boldsymbol{e} = \boldsymbol{u}_{\text{exact}} - \boldsymbol{u}_{\text{numeric}} = \begin{bmatrix} u_{\text{exact},1} - u_{\text{numeric},1} \\ u_{\text{exact},2} - u_{\text{numeric},2} \\ \vdots \\ u_{\text{exact},\dim(V_h)} - u_{\text{numeric},\dim(V_h)} \end{bmatrix} \tag{4.6}$$

where the vector components represent the values of the exact and numerical solutions at each grid point. We then take some norm of $\boldsymbol{e}$ to obtain $e(h) = \|\boldsymbol{e}\|$, which is a function of the grid spacing h. In what follows we use the discrete 2-norm for calculating e, i.e.,

$$e(h) = \|\boldsymbol{e}\|_2(h) = \left(h \sum_{j=1}^{\dim(V_h)} |e_j^2| \right)^{1/2} \tag{4.7}$$

The order p of the method can then be estimated by

$$p \approx \frac{\log(e(h_1)/e(h_2))}{\log(h_1/h_2)}, \tag{4.8}$$

i.e., we need to run two simulations with a different grid spacing. The values of e can be seen in the tables in the appendix. Taking e.g. the errors at the resolutions $100 \times 100 \times 100$ (table A.2, $h_1 = 6.28 \cdot 10^{-2}$) and $200 \times 200 \times 200$ (table A.3, $h_2 = 3.14 \cdot 10^{-2}$), we arrive at

$$p \approx \frac{\log(e(h_1)/e(h_2))}{\log(h_1/h_2)} = \frac{\log\left(1.33 \cdot 10^{-3}/3.33 \cdot 10^{-4}\right)}{\log\left(6.28 \cdot 10^{-2}/3.14 \cdot 10^{-2}\right)} = 2.00 \tag{4.9}$$

which is what we expect for a second order accurate discretization. Having a second order accurate method means that

$$e(h) = Ch^2 + o(h^2) \tag{4.10}$$

where "little-oh" o means that if

$$f(h) = o(g(h)) \text{ as } h \to 0 \tag{4.11}$$

then

$$\left| \frac{f(h)}{g(h)} \right| \to 0 \text{ as } h \to 0. \tag{4.12}$$

4.1.2 Adjusting the Multigrid Component Parameters

Having ascertained that our implementation yields the correct order of accuracy we can now move to figure out the best choice for the parameters for the multigrid components. Based on the conclusions of Happenhofer, (2014) for her two–dimensional solver we start with the following parameters:

- the number of grid points in one direction on the coarsest grid, denoted by N_c. This parameter is chosen automatically, based on the actual resolution setting in the input file. This works as follows:

  1. divide the number of fine grid points per direction, N, by the coarsening ratio. (Remark: the coarsening ratio is always 2 in this implementation.)
  2. If the result is an even number: repeat step 1 until the result is an uneven number.
  3. The number of grid levels used in the multigrid solver is the number of divisions done plus one.
  4. The number of coarse grid points per direction, N_c, is the resulting uneven number.

  For example, if one wants to run a simulation with a resolution of around $200 \times 200 \times 200$ the procedure is the following:

  $$200/2 = 100 \quad \text{result even: continue dividing by 2} \tag{4.13}$$

  $$100/2 = 50 \quad \text{result even: continue dividing by 2} \tag{4.14}$$

  $$50/2 = 25 \quad \text{result even: stop dividing} \tag{4.15}$$

  $$\Longrightarrow N_c = 25, \text{ number of grid levels: } 4$$

  If we wanted, say, a coarse grid resolution of 99 grid points per direction, but still a fine grid resolution of around $200 \times 200 \times 200$, we would have to choose a fine grid resolution of 198×198 such that

  $$198/2 = 99 \quad \text{result uneven: stop dividing} \tag{4.16}$$

  $$\Longrightarrow N_c = 99, \text{ number of grid levels: } 2$$

  So by choosing the fine grid resolution, one is at the same time choosing the number of multigrid levels and the resolution on the coarsest grid. One important note: due to parallelization issues and the necessary presence of ghost cells the number of coarse grids points cannot be chosen too small.

- the number of pre-smoothing steps ν_1 and the number of post-smoothing steps ν_2,
- the accuracy limit for the multigrid solver,
- and the multigrid cycling strategy.

4.1.2.1 Determining the Influence of the Number of Multigrid Levels and the Coarse Grid Resolution

The Setting

- Accuracy setting: exit the solver when the initial residual $\|\boldsymbol{r}\|_2^{(0)}$ is reduced by a factor of 10^{-10} or when $\|\boldsymbol{r}\|_2^{(k)} < 10^{-13}$, whichever comes first. The values for the coarse grid solver are $10^{-12}\|\boldsymbol{r}\|_2^{(0,CG)}$ and $\|\boldsymbol{r}\|_2^{(k,CG)} < 10^{-13}$, respectively.
- Multigrid cycling: V-grid traversal
- $\nu_1 = \nu_2 = 2$

We ran simulations with the following coarse and fine grid resolutions:

- coarse grid resolution $N_c = 25^3$:
  - 2 grid levels, resulting in a fine grid resolution of $N = 50^3$ (results in table A.1),
  - 3 grid levels, resulting in a fine grid resolution of $N = 100^3$ (results in table A.2),
  - 4 grid levels, resulting in a fine grid resolution of $N = 200^3$ (results in table A.3).
- coarse grid resolution $N_c = 51^3$:
  - 2 grid levels, resulting in a fine grid resolution of $N = 102^3$ (results in table A.4)
  - 3 grid levels, resulting in a fine grid resolution of $N = 204^3$ (results in table A.5),
- coarse grid resolution $N_c = 99^3$:
  - 2 grid levels, resulting in a fine grid resolution of $N = 198^3$ (results in table A.6)

The numerical results can be found in the given tables in appendix A.1.1. Here, we give an overview of the results.

The Results: Influence on the Convergence Rate

The impact of the different resolutions and number of multigrid levels on the convergence rate can be seen in figure 4.1. We recognize three distinct groups of graphs: the solid lines represent all simulations with two grid levels. They lead to the fastest convergence in terms of iterations. Their slope, i.e., their convergence rate, is steepest among these simulations. Next, we have the dashed lines which represent the simulations with three grid levels. They lead to medium convergence rates. Lastly, we have the dash–dotted plot which represents the simulation with four grid levels. Judging from these simulations we arrive at our first conclusions:

- the convergence rate is hardly dependent on the fine or coarse grid resolution,
- the convergence rate is dependent on the number of grid levels used: less grid levels lead to a better convergence rate

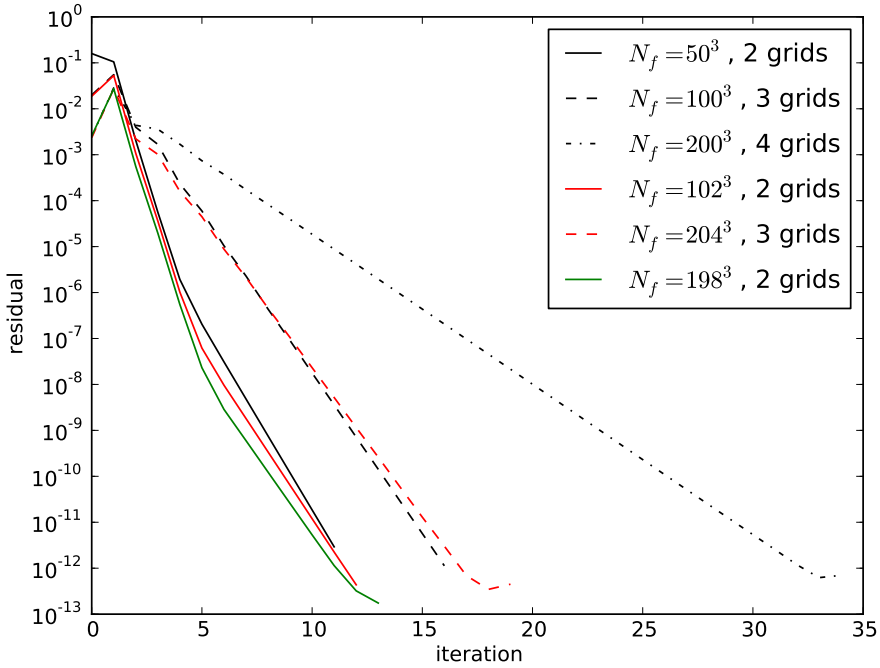


Figure 4.1: Plot of the residual norm against the number of iterations for different coarse grid resolutions. The black lines correspond to simulations with a coarse grid resolution of $N_c = 25^3$, the red lines correspond to $N_c = 51^3$ and the green line corresponds to $N_c = 99^3$. N_f is the fine grid resolution.

The Results: Influence on the Wall Clock Times

To measure the efficiency of the solver, both the convergence rate and the total computational work – measured by the wall clock time – are important measures. We now investigate the effect the different resolutions had on the wall clock times. These have been measured by a call to `system_clock` immediately before and after the call to the multigrid solver.

Figure 4.2 shows a comparison of the normalized wall clock times, mean convergence rates and necessary multigrid cycles of the three simulations with fine grid resolutions of $200 \times 200 \times 200$, $204 \times 204 \times 204$ and $198 \times 198 \times 198$ (tables A.6, A.5, A.3). Obviously the solution with four grid levels and a fine grid resolution $N_f = 200^3$ has the worst wall clock time, convergence rate and necessary grid cycles among the three solutions. The best wall clock time ($N_f = 204^3$, 3 grid levels) is not reached by the simulation with the best average convergence rate ($N_f = 198^3$, 2 grid levels), however. This can be explained by the fact that while the convergence rate for the two–grid V-cycle is better than for the three–grid cycle in this implementation for this particular problem, the time needed to solve the system of equations on the coarsest grid is significantly higher for the case with a coarse grid resolution $N_c = 98^3$ (and a corresponding fine grid resolution of $N_f = 198^3$) than it is for the case with the significantly lower coarse grid resolution $N_c = 51^3$ and the fine grid resolution of 204^3. We thus arrive at another conclusion:

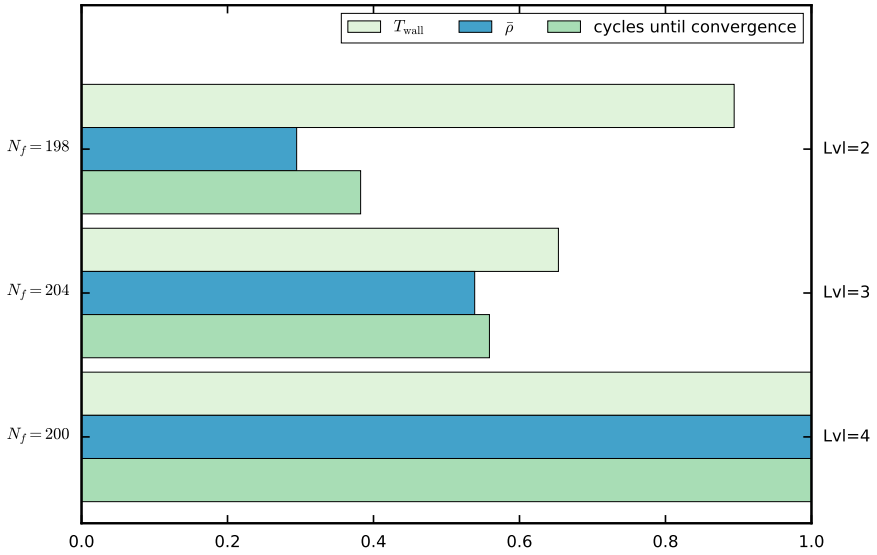


Figure 4.2: Normalized wall clock time, average convergence rate and multigrid iterations until convergence for fine grid resolutions of $N_\mathrm{f} \approx 200^3$ and different number of grid levels for $\nu_1 = \nu_2 = 2$ and multigrid V-cycling.

- the wall clock time is not always smallest for the case with best convergence rate

Next, we investigate the influence of a change in the number of pre- and post-smoothing steps.

4.1.2.2 Determining the Influence of the Number of Smoothing Steps

According to Trottenberg, Oosterlee, and Schüller, (2001), the number of pre- and post-smoothing steps should be chosen so that $\nu_1 + \nu_2 \leq 3$. We repeated the above simulations with the following settings for ν_1 and ν_2:

- $\nu_1 = 2$ and $\nu_2 = 1$.

  The numerical results can be seen in tables A.7 - A.9 for the simulations with a coarse grid resolution of $25 \times 25 \times 25$, tables A.10 - A.11 for $51 \times 51 \times 51$ and table A.12 for $99 \times 99 \times 99$.

- $\nu_1 = 1$ and $\nu_2 = 2$.

  The numerical results can be seen in tables A.13 - A.15 for the simulations with a coarse grid resolution of $25 \times 25 \times 25$, tables A.16 - A.17 for $51 \times 51 \times 51$ and table A.18 for $99 \times 99 \times 99$.

- $\nu_1 = 1$ and $\nu_2 = 1$.

  The numerical results can be seen in tables A.19 - A.21 for the simulations with a coarse grid resolution of $25 \times 25 \times 25$, tables A.22 - A.23 for $51 \times 51 \times 51$ and table A.24 for $99 \times 99 \times 99$.

We have investigated the same as before: the effects on the wall clock times, average convergence rates and iterations until convergence.

The Results: The Influence on the Convergence Rate

Plots of the residuals against the iteration count can be seen in figure 4.3. We observe the following properties in these plots:

- more smoothing cycles lead to faster convergence
- for two grid levels, using only one pre- and post-smoothing step leads to worse convergence properties than when using two pre- and one post-smoothing step.

The Results: The Influence on the Wall Clock Times

The results are shown in figure 4.6. We observe the following:

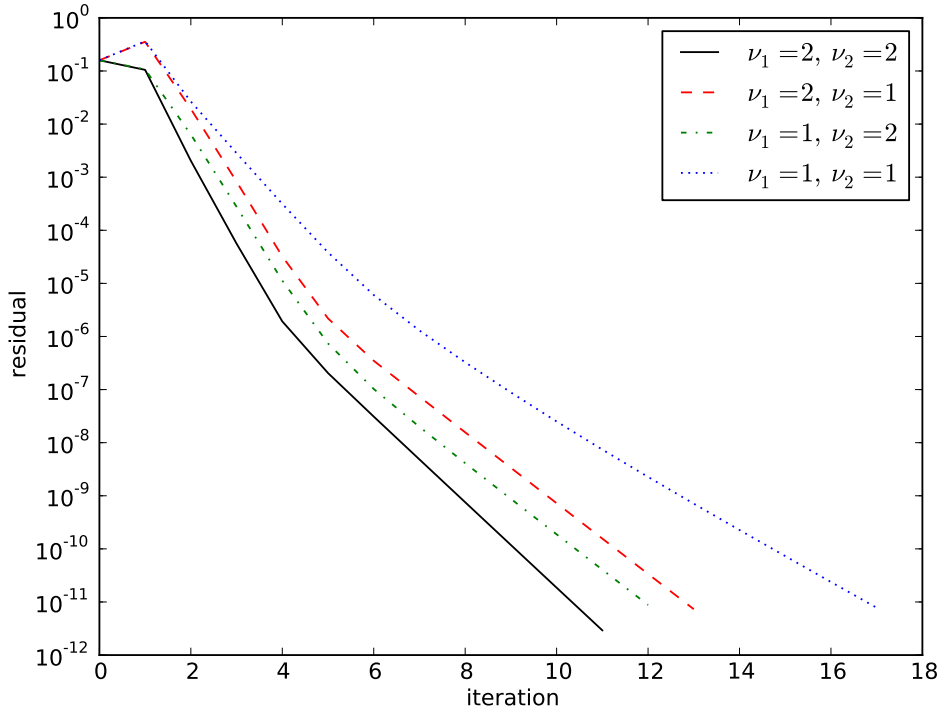
- The average convergence rate is worst when using $\nu_1 = \nu_2 = 1$ smoothing cycles. This was already apparent in the previous plot.
- The wall clock time for almost every simulation is shortest when using $\nu_1 = \nu_2 = 1$ smoothing cycles. That means that the convergence rate is not a good measure of multigrid performance.
- When comparing simulations with similar fine grid resolutions, i.e., $N_\mathrm{f} = 100^3$ and 102 or $N_\mathrm{f} = 198^3, 200, 204$ we see that simulations with three grid levels always outperform simulations with two grid levels. Four grid levels do not yield a better result than three multigrid levels at this problem, though.

4.1.2.3 Conclusions from this Series of Experiments

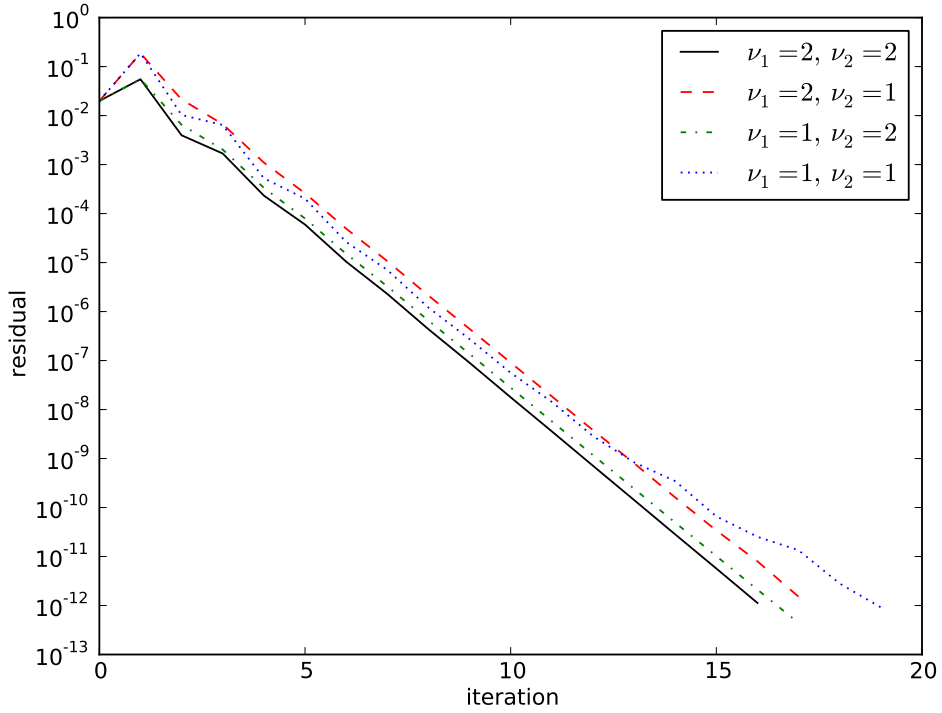
For this kind of problem, it seems that the following set of parameters yields the best performance in terms of the wall clock time:

- coarse grid resolution: $N_\mathrm{c} = 51^3$;
- number of grid levels: 3 levels;
- resulting fine grid resolution: $N_\mathrm{f} = 204^3$;
- number of pre- and post-smoothing cycles: $\nu_1 = \nu_2 = 1$.

We use these settings for the next tests. Keep in mind that these parameters have proven best for this particular problem only. If you want to achieve optimal performance with the solver when using ANTARES, you must find out the best values for your specific problem by doing numerical experiments yourself. The next tests we perform are

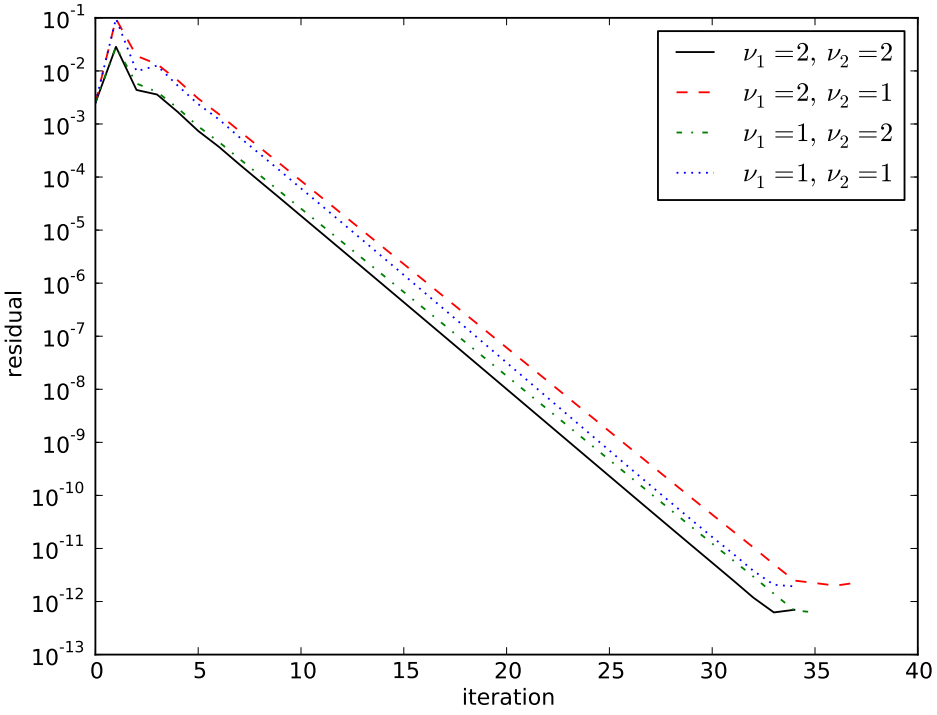


(a) $N_\mathrm{f} = 50^3$, $N_\mathrm{c} = 25^3$, 2 grid levels

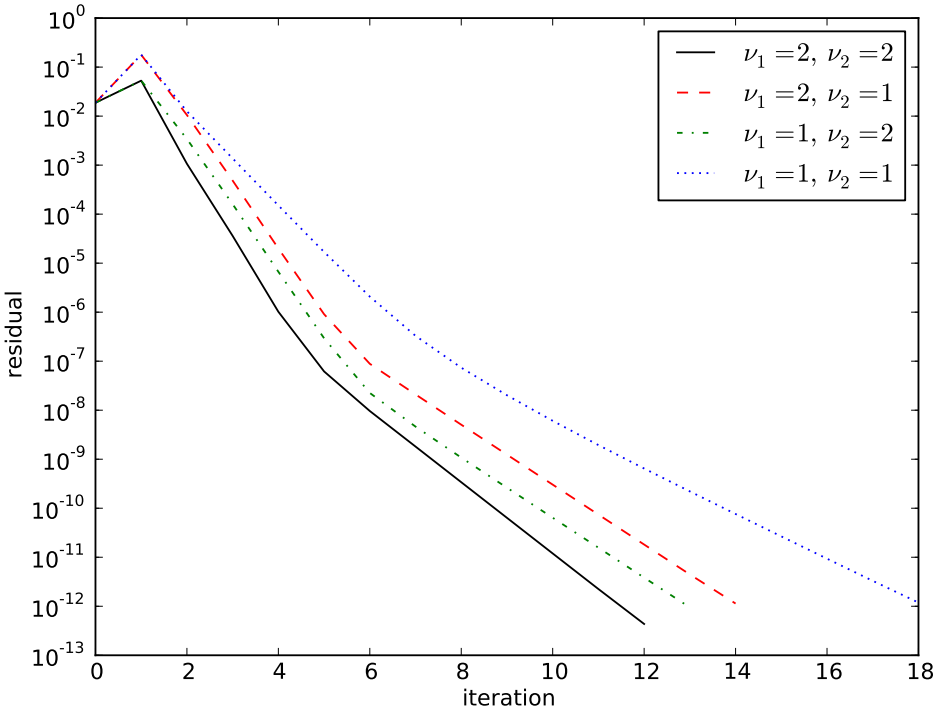


(b) $N_\mathrm{f} = 100^3$, $N_\mathrm{c} = 25^3$, 3 grid levels

Figure 4.3: Plot of the norm of the residual against the multigrid iteration for different coarse grid resolutions and numbers of pre- and post-smoothing steps.

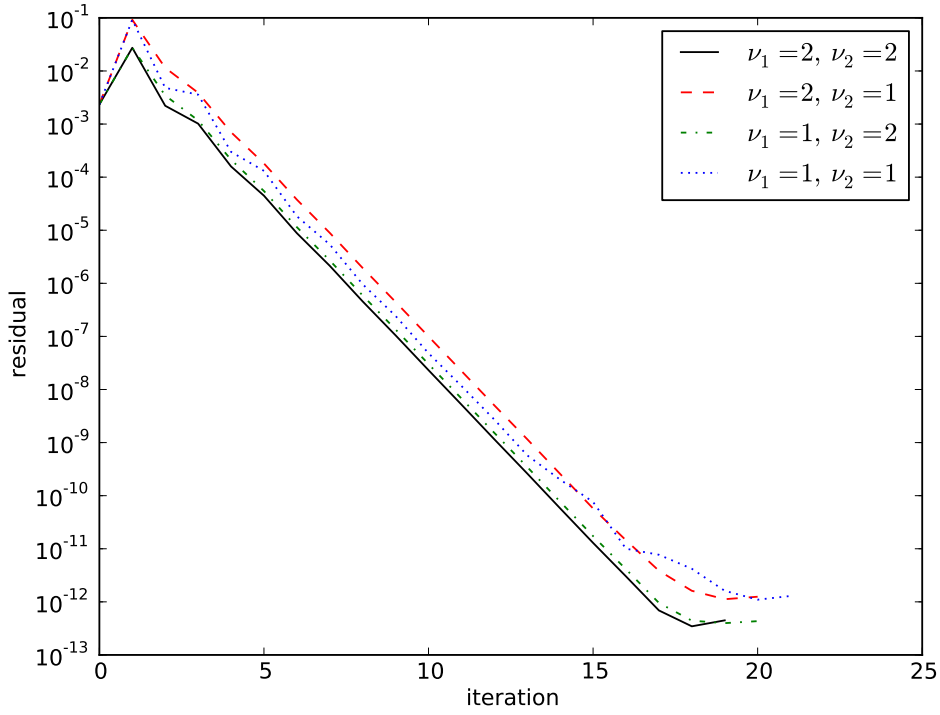


(a) $N_\mathrm{f} = 200^3$, $N_\mathrm{c} = 25^3$, 4 grid levels

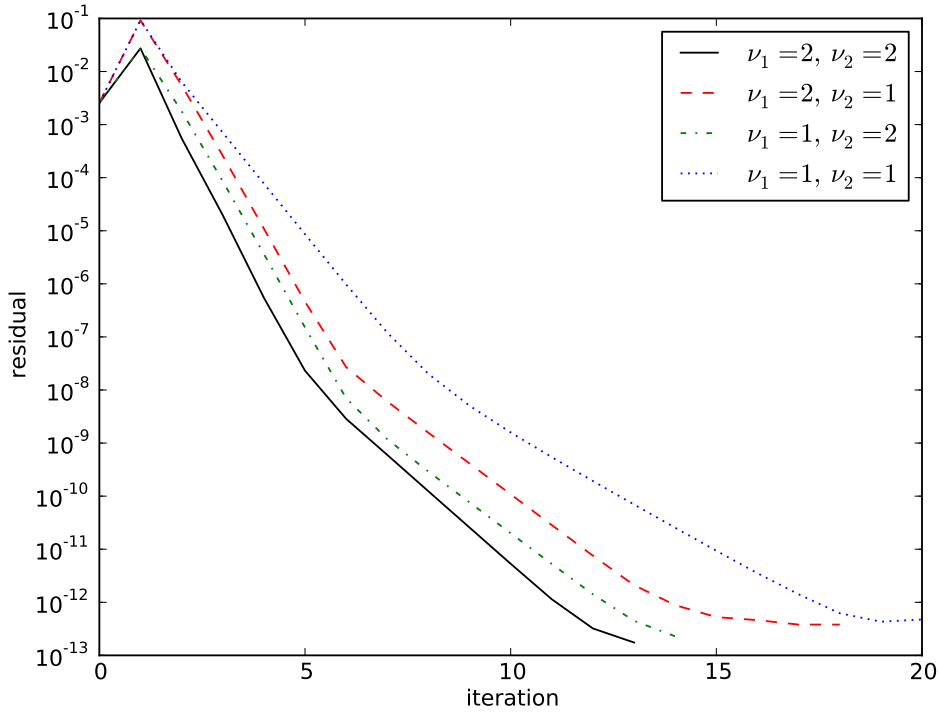


(b) $N_\mathrm{f} = 102^3$, $N_\mathrm{c} = 51^3$, 2 grid levels

Figure 4.4: Plot of the norm of the residual against the multigrid iteration for different coarse grid resolutions and numbers of pre- and post-smoothing steps.

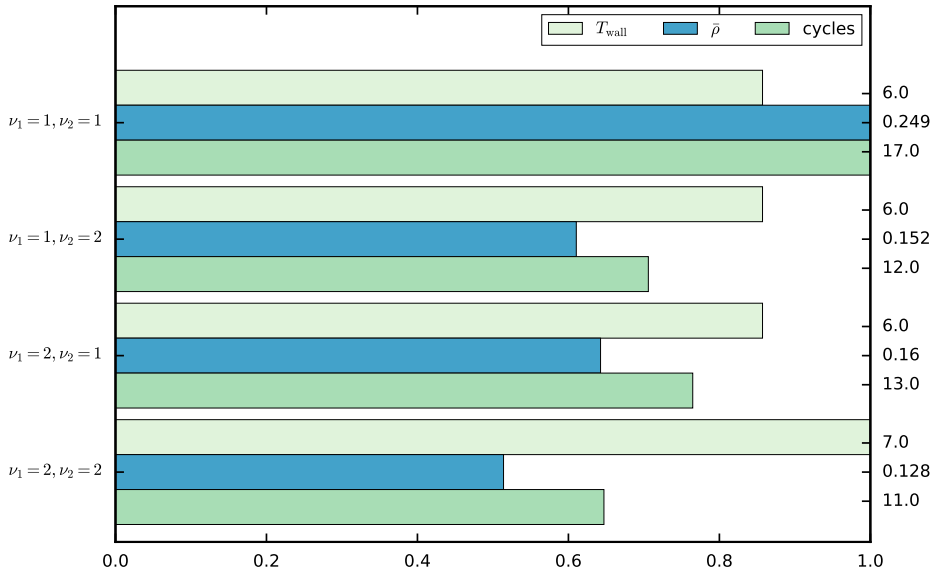


(a) $N_\mathrm{f} = 204^3$, $N_\mathrm{c} = 51^3$, 3 grid levels

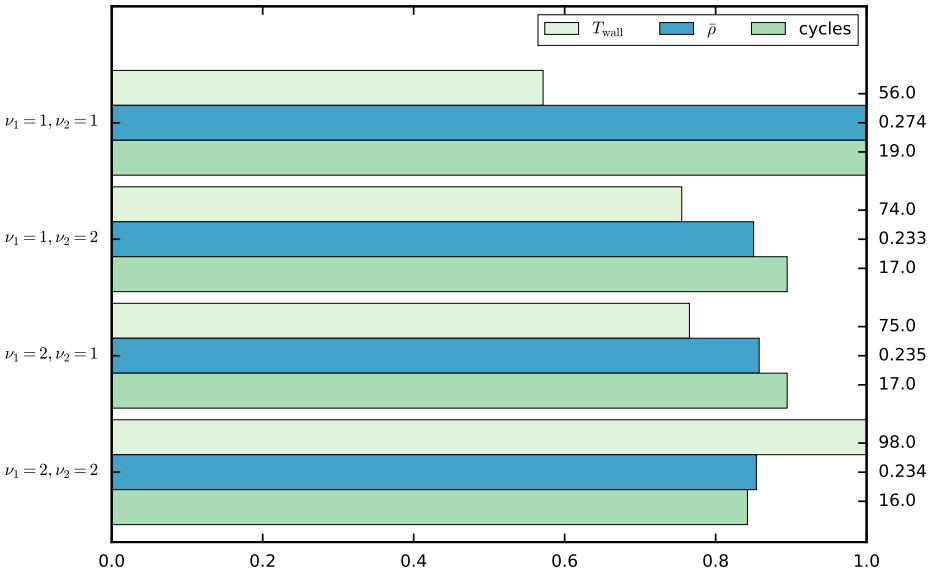


(b) $N_\mathrm{f} = 198^3$, $N_\mathrm{c} = 99^3$, 2 grid levels

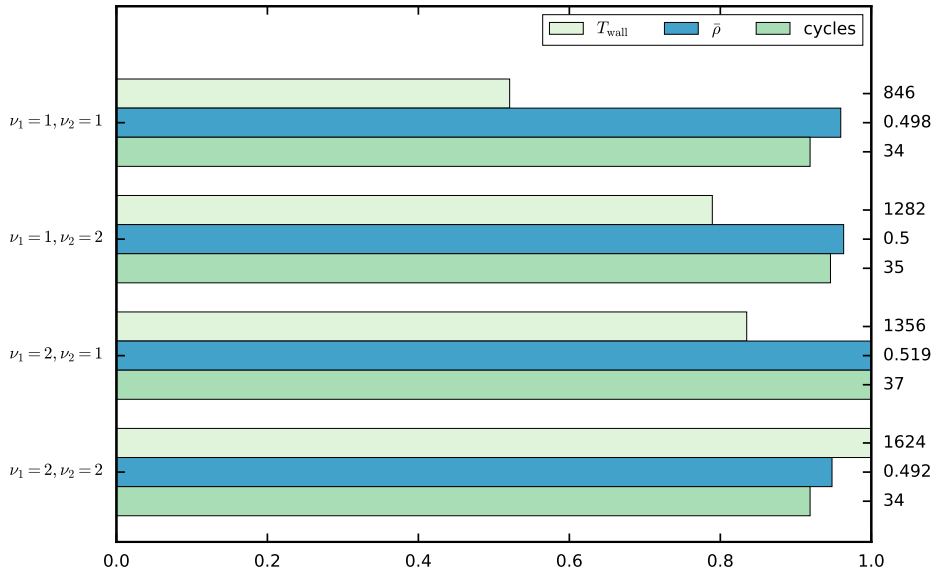
Figure 4.5: Plot of the norm of the residual against the multigrid iteration for different coarse grid resolutions and numbers of pre- and post-smoothing steps.



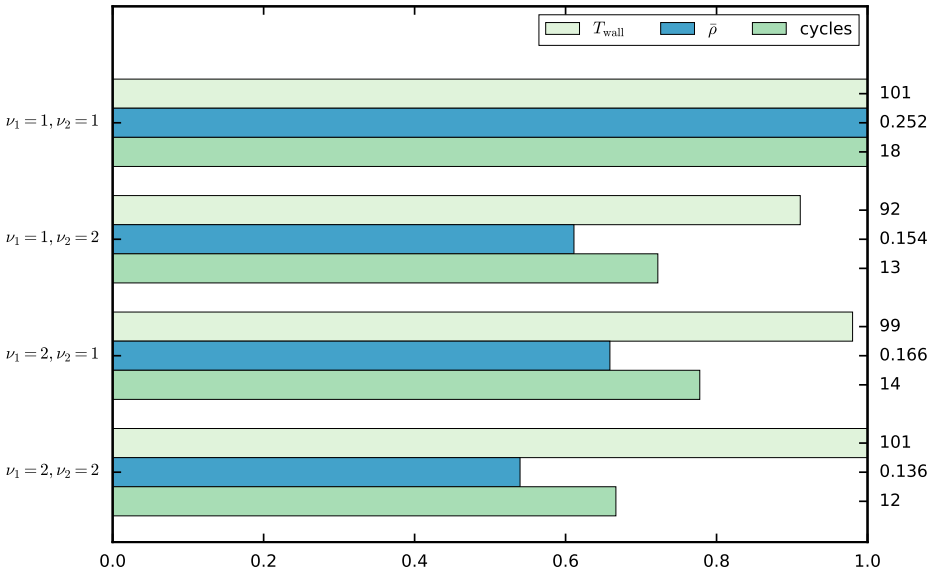
(a) $N_{\text{f}} = 50^3$, $N_{\text{c}} = 25^3$, 2 grid levels



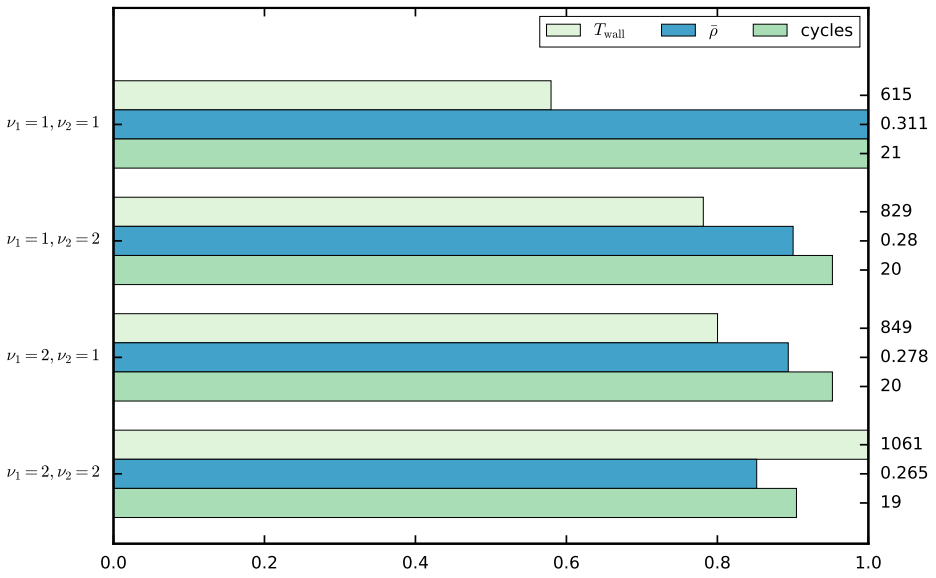
(b) $N_{\text{f}} = 100^3$, $N_{\text{c}} = 25^3$, 3 grid levels



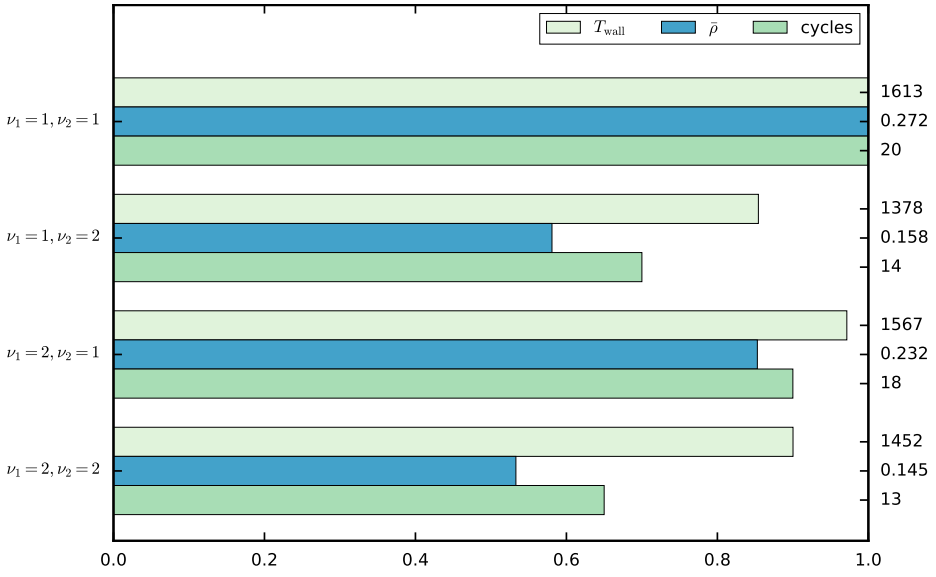
(a) $N_{\text{f}} = 200^3$, $N_{\text{c}} = 25^3$, 4 grid levels



(b) $N_{\text{f}} = 102^3$, $N_{\text{c}} = 51^3$, 2 grid levels



(a) $N_{\text{f}} = 204^3$, $N_{\text{c}} = 51^3$, 3 grid levels



(b) $N_{\text{f}} = 198^3$, $N_{\text{c}} = 99^3$, 2 grid levels

Figure 4.6: Normalized wall clock time, average convergence rate and multigrid iterations until convergence for varying fine grid resolutions and different values for ν_1 and ν_2. Multigrid cycling: V-cycling.

- investigation of the influence of the type of multigrid cycling: V-cycling or W-cycling,
- investigation of the influence of the accuracy settings.

4.1.3 Third Series of Experiments — The Influence of the Grid Traversal

In this section we investigate the influence of the type of grid traversal used. Up to now, each experiment has been done with V–traversal. Now we test using W–traversal.

4.1.3.1 The Setting

As stated in the previous section, we use a coarse grid resolution of $N_c = 51^3$ grid points per direction because it resulted in the best wall clock times. Three simulation settings have been used:

- coarse grid resolution $N_c = 25^3$ and fine grid resolutions of $N_f = 100^3$ and $N_f = 200$, corresponding to 3 and 4 grid levels respectively
- coarse grid resolution of $N_c = 51^3$ and a fine grid resolution of $N_c = 204$, i.e., 3 grid levels

Each setting was done once with $\nu_1 = \nu_2 = 1$ and once with $\nu_1 = \nu_2 = 2$. W-grid traversal was used in each simulation. The numerical results can be found in tables A.25 - A.30.

4.1.3.2 Results

Residuals vs. Multigrid Iteration

Again, we first take a look at the residual evolution in terms of multigrid cycles in figure 4.7. We immediately note the same influence of the number of pre- and post-smoothing cycles as in the previous section: more smoothing leads to faster convergence in terms of complete multigrid iterations.

But there are some noteworthy differences, too:

- the convergence rate seems to be independent of the numbers of grid levels and also almost independent of the fine grid resolution. This is much more satisfying than the level-dependent convergence rate which we had observed for V-traversal.
- the convergence rate (i.e., the slope in figure 4.7) for W-grid traversal) is better

Let us look at the differences more directly by plotting the results in one figure and compare the simulations with fine grid resolutions of $N_f = 100^3$, $N_f = 200^3$ and $N_f = 204^3$ grid points per direction for V- and W-cycling.

First Setting: $N_f = 100^3$**,** $N_c = 25^3$

Looking at figure 4.8, we see that the W-cycle leads to better convergence rates when using $\nu_1 = \nu_2 = 2$ smoothing steps. For $\nu_1 = \nu_2 = 1$ we have an ambivalent result: while the convergence rate is better after only a few iterations, it becomes worse towards the end. However, since the

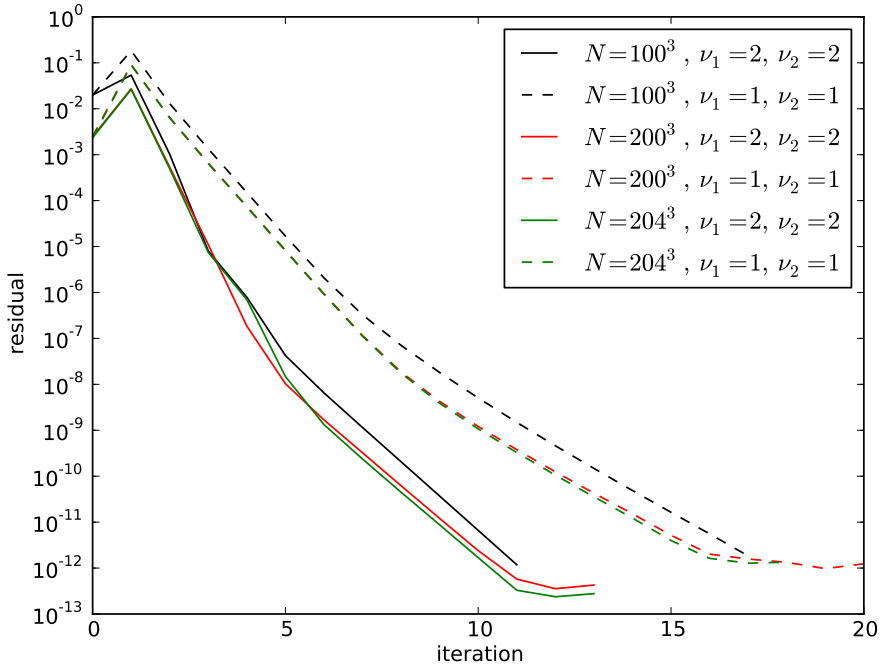


Figure 4.7: Evolution of 2-norm of residual against number of multigrid iterations.

accuracy to which to reduce the residual ϵ is chosen larger in production runs this does not matter because less iterations are performed in these runs, anyway. So in summary, for production simulations, W-cycling leads to better convergence.

Second Setting: $N_\mathbf{f} = 204^3$**,** $N_\mathbf{c} = 51^3$

We see in figure 4.9 that the case looks similar to the simulation with fine grid resolution of $N_\mathrm{f} = 204^3$. For small iteration counts the W-cycle leads to faster convergence and using more pre- and post-smoothing steps also leads to faster convergence.

Third Setting: $N_\mathbf{f} = 200^3$**,** $N_\mathbf{c} = 25^3$

The difference to the two settings before is that we are using four grid levels now. When using 4 grid levels the W-cycle leads to a very distinct improvement of the convergence rate as can be seen in figure 4.10.

Although the V-cycle has a nice constant convergence rate all the way down to the accuracy limit, reducing the residual by an order of magnitude takes significantly more grid cycles than with the W-cycle. From the standpoint of minimizing the overall convergence rate, the W-cycle with $\nu_1 = \nu_2 = 2$ leads to the best result. However, the wall clock time is what really counts in practical simulations so that is what we investigate next.

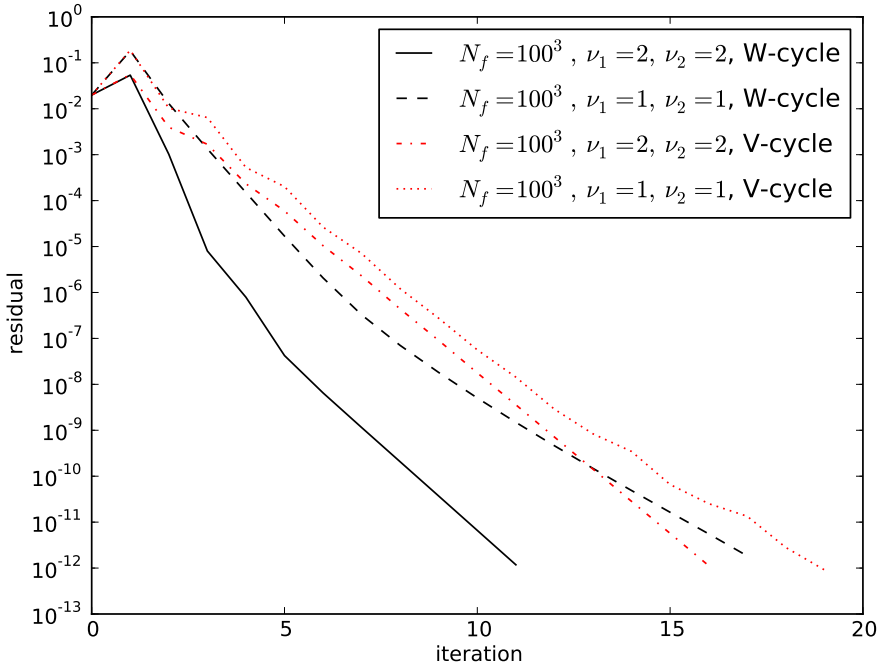


Figure 4.8: Plot of the norm of the residual against the multigrid iteration for a fine grid resolution of $N_f = 100^3$ for different values of the number of pre- and post-smoothing steps and different grid traversal strategies.

4.1.4 Wall Clock Times for W-cycling

Let us now investigate the wall clock times for the simulations with W-grid traversal. First, we look at the simulations where three grid levels have been used, then we look at the simulations where four grid levels have been used.

4.1.4.1 Three Grid Levels

The results can be found in figure 4.11. Both sub-figures look very similar. While the absolute values of wall clock times, average convergence rate and multigrid cycles needed to achieve convergence differ, the normalized values look very similar for both the case where the fine grid resolution is $N_f = 100^3$ and the case where it is $N_f = 204^3$ points. One possible explanation could be that that for three grid levels, the differences for using a different number of pre- and post-smoothing steps and a different cycling pattern do not depend on the actual number of fine or coarse grid points but only on the number of grid levels used in the multigrid algorithm. For both settings the following points can be observed:

- the smallest wall clock time is achieved when using less ($\nu_1 = \nu_2 = 1$ here) smoothing steps and W-grid traversal,

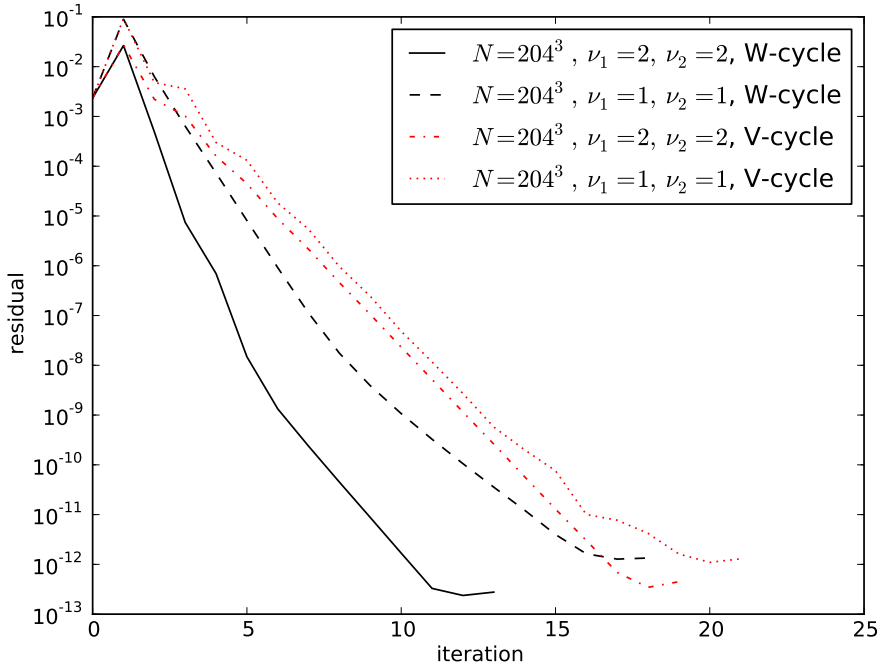


Figure 4.9: Residual decline comparison for $N_\mathrm{f} = 204^3$.

- the best smoothing rate is achieved when using more ($\nu_1 = \nu_2 = 2$ here) smoothing steps and W-cycle grid traversal,
- fewest multigrid iterations are needed when using more ($\nu_1 = \nu_2 = 2$ here) smoothing steps and W-grid traversal,
- the difference between V- and W-cycles is more pronounced when using a higher number of smoothing steps.

Summarizing, when using three grid levels, the smallest wall clock time and hence least computational work is achieved with the W-cycle and $\nu_1 = \nu_2 = 1$ pre- and post-smoothing steps.

4.1.4.2 Four Grid Levels

Figure 4.12 shows the comparison of wall clock time, average smoothing rate and number of multigrid cycles needed for the case of using a fine grid resolution of $N_\mathrm{f} = 200^3$, which corresponds to a coarse grid resolution of $N_\mathrm{c} = 25^3$. Here, we can see a pronounced difference between V- and W-grid traversal, especially in the number of iterations needed and in the average smoothing rate. For the V-cycle we have a mediocre smoothing rate of about 0.5 while for the W-cycle we have a good value of about 0.15 for 2 pre- and post-smoothing steps.

This can be explained by the fact that the 4 level W-cycle visits the coarsest grid four times in one multigrid iteration while the V-cycle visits it only once, which means that 34 multigrid

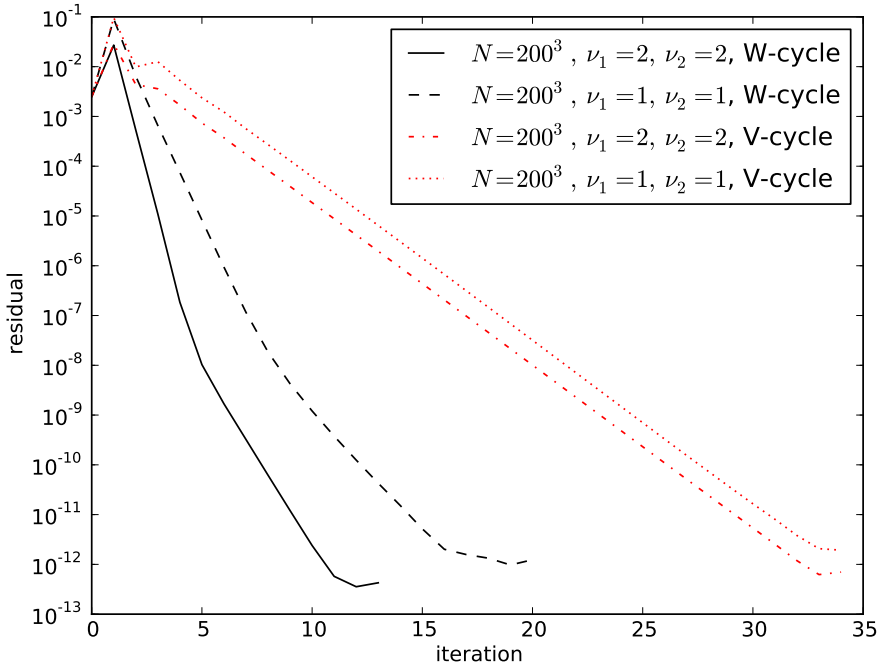
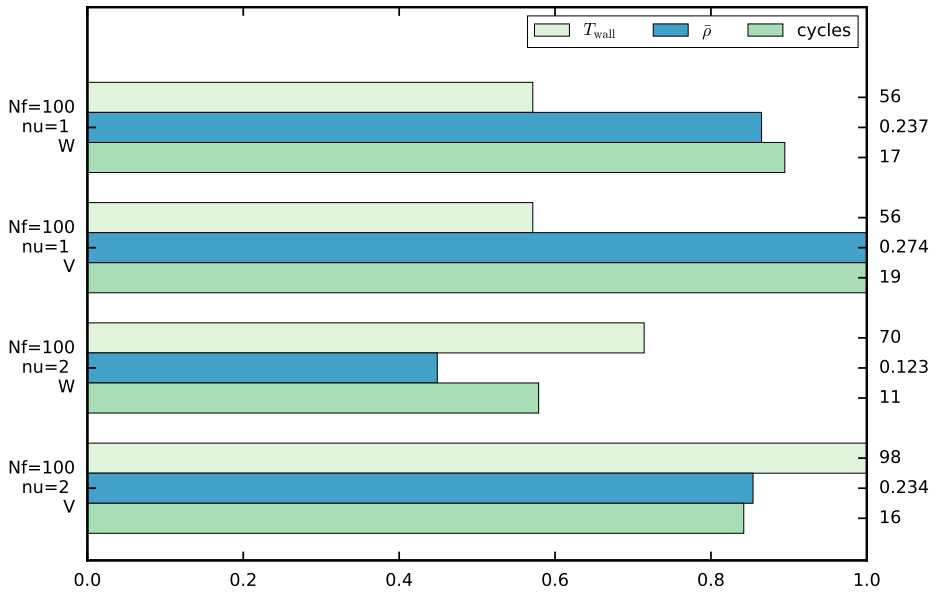


Figure 4.10: Plot of the norm of the residual against the multigrid iteration for a fine grid resolution of $N_f = 200^3$ and a coarse grid resolution of $N_c = 25^3$ which corresponds to four grid levels.

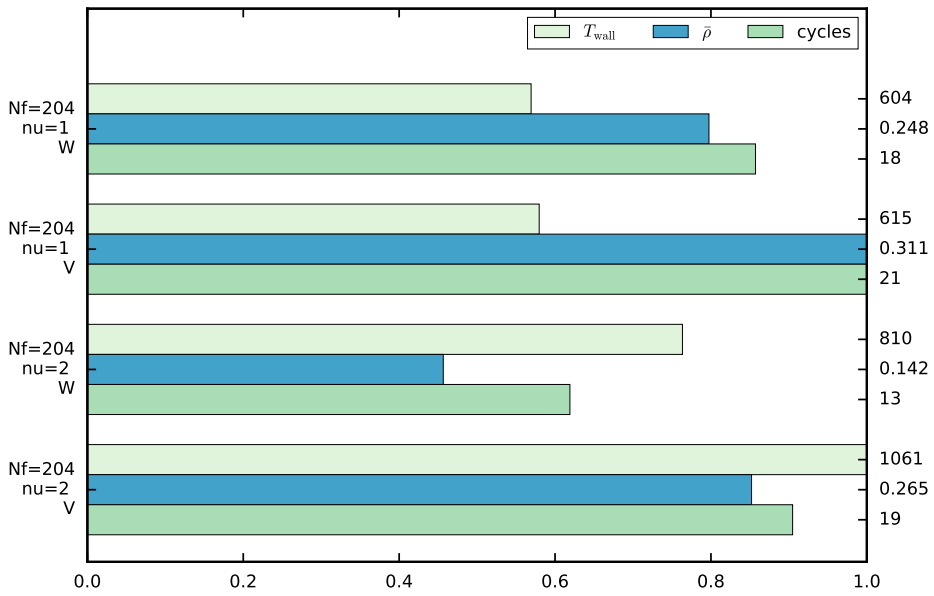
iterations with the V-cycling pattern lead to 34 coarse grid solver calls while 13 multigrid cycles with the W-cycling pattern lead to $13 \cdot 4 = 52$ coarse grid solver calls. But the wall clock time is still much better, at least for this particular test case. The main observations for four grid levels are:

- the smallest wall clock time is achieved using less ($\nu_1 = \nu_2 = 1$ here) smoothing steps and W-cycle grid traversal,
- the best smoothing rate is achieved when using more ($\nu_1 = \nu_2 = 2$ here) smoothing steps and W-cycle grid traversal,
- fewest multigrid iterations are needed when using more ($\nu_1 = \nu_2 = 2$ here) smoothing steps and W-cycle grid traversal,
- the difference between V- and W-cycles is more pronounced when using a higher number of smoothing steps.

So the observations for three and four grids are the same.



(a) $N_{\text{f}} = 100^3$



(b) $N_{\text{f}} = 204^3$

Figure 4.11: Wall clock times, average smoothing rates and MG-cycles needed to achieve convergence for 3 grid levels.

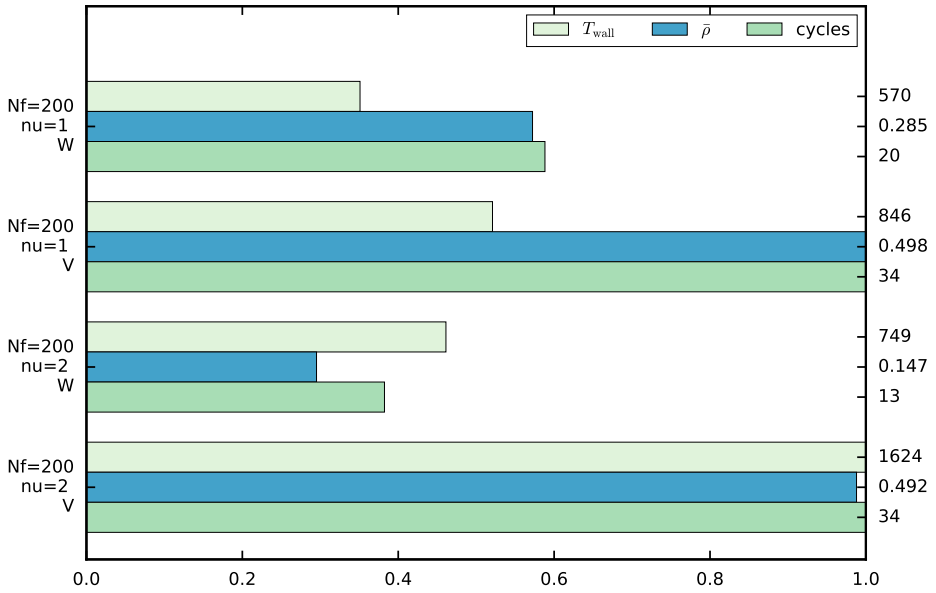


Figure 4.12: Wall clock times, average smoothing rates and MG-cycles needed to achieve convergence for 4 grid levels.

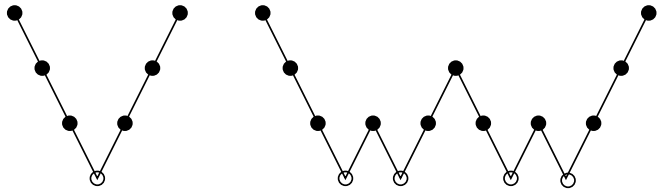


Figure 4.13: V- and W-cycle for four grid levels (taken from Trottenberg, Oosterlee, and Schüller, 2001))

4.1.5 Result from W-cycle Testing

The question remains which setup is the best for solving a system with, say, about 200 grid points in each direction. To this end, one should compare figures 4.11 and 4.12. For the fastest simulations with $\nu_1 = \nu_2 = 1$ and W-cycling it does not seem to matter whether 3 or 4 grid levels are used. The absolute wall clock times differ but the $N_f = 200^3$ simulation has only about 94% of the grid points of the $N_f = 204^3$ simulation. At the same time, the wall clock time is also only about 94 % that of the $N_f = 204^3$ case so these numbers are very similar. If in doubt, one should use W-cycling for this test case to be able to use more grid levels.

In summary, for this particular test case the multigrid setup should be: the smallest wall clock time and hence least computational work is achieved with the W-cycle and $\nu_1 = \nu_2 = 1$ pre- and post-smoothing steps, no matter whether three or four grid levels are used. This is a better result than the V-cycle result which stated that using four grid levels leads to an inefficient algorithm.

4.1.6 Fourth Series of Experiments — Adjusting the Accuracies

Up to now, we performed the simulations up to almost machine precision in order to test the convergence properties. We now turn to a more realistic scenario because it is not necessary for the residual to become so small. After all, our solver does not solve the differential equation but the system of equations that resulted from our finite element discretization. This means that it is not necessary to achieve an accuracy with the solver which is smaller than the discretization accuracy.

Formally, this can be written in the following way. The true solution $\boldsymbol{u} \in V$ is approximated by an approximated solution $\boldsymbol{v}_h \in V_h$. The discretization error for our finite element discretization is of order

$$\|\boldsymbol{u} - \boldsymbol{u}_h\| = O(h^2). \tag{4.17}$$

This means that a feasible stopping criterion is one where the increase in accuracy from one multigrid iteration $(k-1)$ to the next multigrid iteration (k) is smaller than this discretization accuracy, i.e.,

$$\|\boldsymbol{u}_h^{(k)} - \boldsymbol{u}_h^{(k-1)}\| < \|\boldsymbol{u} - \boldsymbol{u}_h\| = O(h^2). \tag{4.18}$$

One possible choice is

$$\|\boldsymbol{u}_h^{(k)} - \boldsymbol{u}_h^{(k-1)}\| < \beta h^2 \tag{4.19}$$

where β can be used to adjust the level of accuracy needed.

Because it has been observed by our group that it is bad in some situations to set a too high β ANTARES uses a rather conservative value. That way we reach an accuracy better than discretization accuracy but still far above machine precision.

A safe implementation should be reducing the first residual $\|\boldsymbol{r}\|^{(0)}$ by 6 orders of magnitude.

One question that remains to be investigated is the statement by Happenhofer, (2014) that the error tolerance of the CG solver on the coarsest grid should be one order of magnitude better than that of the multigrid solver. This is what we investigate next.

4.1.6.1 Testing the Influence of ϵ_{coarse}

The Setting

Here, we perform simulations for values of $\epsilon_{\text{coarse}} \in \{0.1 \cdot \epsilon_{\text{fine}}, 0.01 \cdot \epsilon_{\text{fine}}, 0.001 \cdot \epsilon_{\text{fine}}\}$ to determine the influence of the coarse grid solver accuracy. We use fine grid resolutions of $N_{\text{f}} = 100^3$ (corresponds to 3 grid levels and $N_{\text{c}} = 25$), $N_{\text{f}} = 200^3$ (corresponds to 4 grid levels and $N_{\text{c}} = 25$) and $N_{\text{f}} = 204^3$ (corresponds to 3 grid levels and $N_{\text{c}} = 51$). This allows us to determine if the number of grid levels and coarse grid points changes the behavior in any way. We use W-cycling because it has proven to be more robust and faster in the previous section. The number of pre- and post-smoothing steps is set to 1.

The numerical results can be found in tables A.31 - A.39. The summarized results are visualized in figures 4.14 and 4.15.

The Results

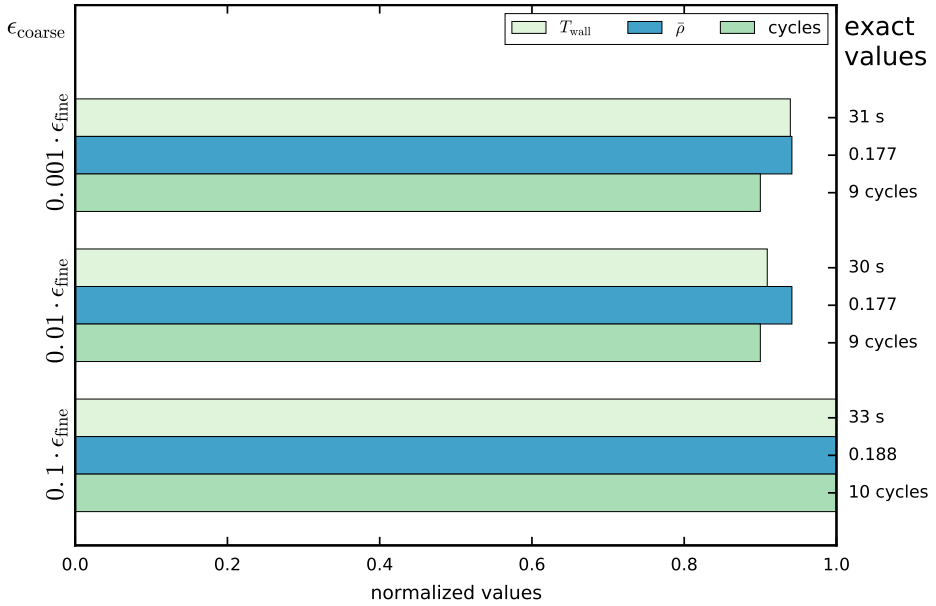
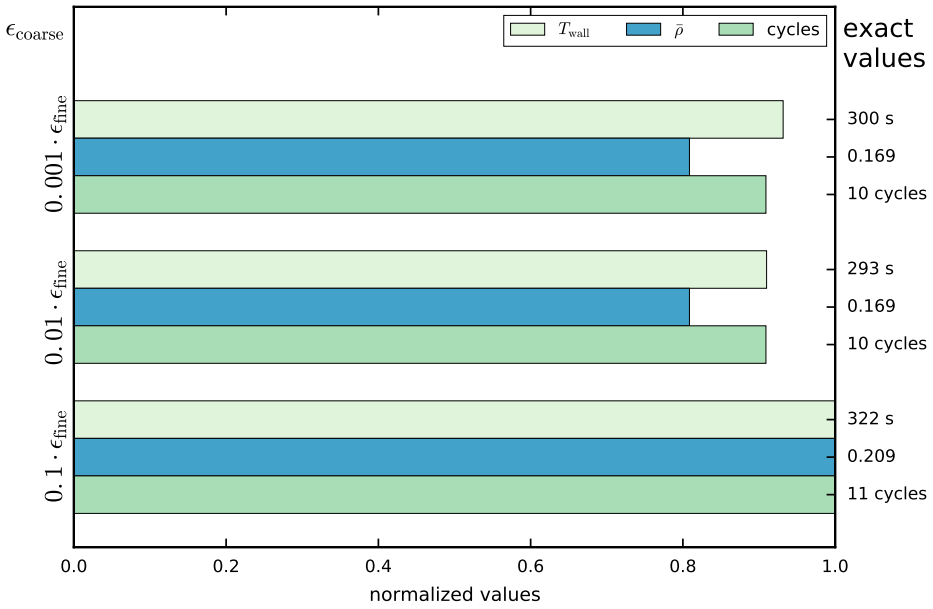


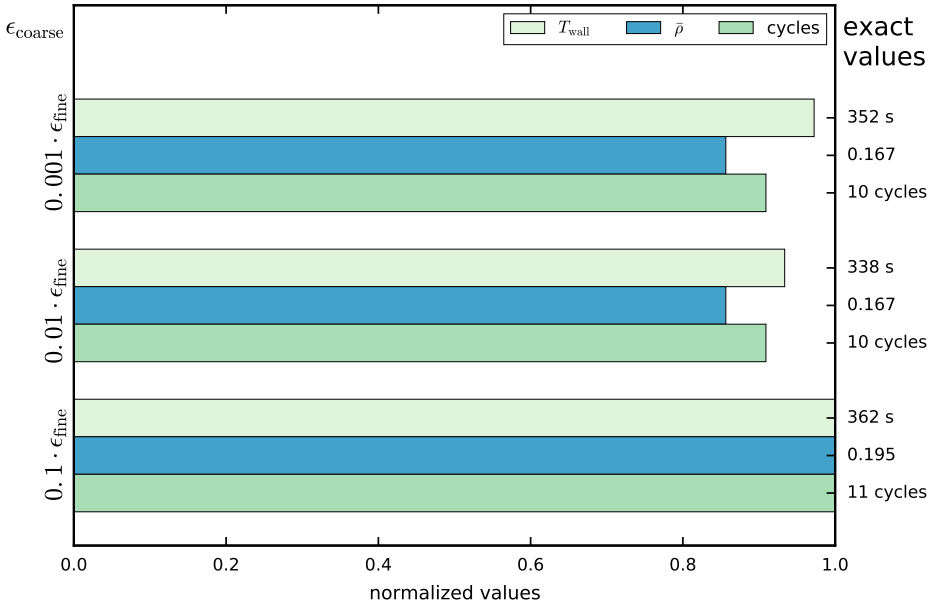
Figure 4.14: Wall clock times, average smoothing rates and MG-cycles needed to achieve convergence.

We see that for each simulation setting $\epsilon_{\text{coarse}} = 0.1 \cdot \epsilon_{\text{fine}}$ is not sufficient in terms of efficiency for the tested parameters in test problem 1. Two orders of magnitude yield better result in terms of wall clock time, average smoothing rate and needed multigrid cycles. Three orders of magnitude result in slightly higher wall clock times in all three tested resolutions. Accordingly, the standard 3D setting for the coarse grid accuracy has been implemented as

$$\epsilon_{\text{coarse}} = 0.01 \cdot \epsilon_{\text{fine}}. \tag{4.20}$$



(a) $N_{\text{f}} = 200^3$



(b) $N_{\text{f}} = 204^3$

Figure 4.15: Wall clock times, average smoothing rates and MG-cycles needed to achieve convergence for $N_{\text{f}} = 204^3$.

4.1.7 Summary of Best Parameters for Test Case 1

Here, we summarize the results which we have obtained for the parameters when handling test case 1. Again, we point out that each problem treated with the solver will likely have a different set of best parameters values. They will have to be adjusted by each user separately for the kind of problem they want to tackle. This section served the purpose of demonstrating the impact that the multigrid components have on the wall clock times, convergence rates and multigrid cycles needed to achieve convergence. For the purpose of solving (4.3), the following parameter yield the best results in terms of wall clock times:

- number of pre– and post–smoothing steps: $\nu_1 = \nu_2 = 1$
- grid cycle traversal: W-cycling for more than 2 levels
- coarse grid accuracy: $\epsilon_{\text{coarse}} = 0.01\epsilon_{\text{fine}}$

Next, we verify the accuracy of the solver for different boundary conditions, starting with Neumann boundaries.

4.2 Test Case 2 — Linear Equation with Neumann B.C.

To test the behavior of the solver for Neumann boundary conditions we treat the same equation as in test case 1, albeit with different boundary conditions. The problem reads

$$-\nabla\cdot(x\nabla u(\boldsymbol{x}))+xu(\boldsymbol{x}) = 4x\sin(x)\sin(y)\sin(z)-\cos(x)\sin(y)\sin(z). \tag{4.21}$$

with the boundary conditions

$$\begin{aligned}\frac{\partial u}{\partial x}(0,y,z) &= \sin(y)\sin(z),\\ \frac{\partial u}{\partial x}(2\pi,y,z) &= \sin(y)\sin(z),\\ u(x,0,z) &= u(x,2\pi,z),\\ u(x,y,0) &= u(x,y,2\pi).\end{aligned} \tag{4.22}$$

The exact solution is

$$u(x,y,z) = \sin(x)\sin(y)\sin(z). \tag{4.23}$$

It should be noted here that one needs to pay attention when implementing the Neumann boundaries: the general Neumann problem reads
find u such that

$$\begin{cases} -\nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))+\xi(\boldsymbol{x})u(\boldsymbol{x}) = f(\boldsymbol{x}) & \text{in } \Omega\\ \nabla u\cdot\boldsymbol{n} = g_N & \text{on } \partial\Omega. \end{cases} \tag{4.24}$$

So the boundary conditions are specified by

$$\nabla u\cdot\boldsymbol{n} = g_N. \tag{4.25}$$

This means that g_N for this problem is given by

$$\begin{aligned} g_{N,\text{lower boundary}} &= -\sin(y)\sin(z) \\ g_{N,\text{upper boundary}} &= \sin(y)\sin(z) \end{aligned} \tag{4.26}$$

because of the direction of $\boldsymbol{n}$.

In contrast to test case 1, only the accuracy of the solver is tested here. It would serve no purpose to repeat the parameter investigations for every test problem because the best parameters need to be found out by every user of the solver for each individual problem anyway. We use the parameters which we have found out to be best for test case 1, i.e.

- number of pre– and post–smoothing steps: $\nu_1 = \nu_2 = 1$
- grid cycle traversal: W-cycling for more than 2 levels
- coarse grid accuracy: $\epsilon_{\text{coarse}} = 0.01\,\epsilon_{\text{fine}}$

The numerical results can be found in tables A.40 and A.41.

4.2.1 Calculating the Order of Accuracy

By the same process as in section 4.1.1 we calculate the order of accuracy of the solver. We derive the order p with the values of tables A.40 and A.41 and obtain

$$\begin{aligned} p \approx& \frac{\log(e(h_1)/e(h_2))}{\log(h_1/h_2)} \\ =& \frac{\log\left(5.44\cdot 10^{-3}/3.32\cdot 10^{-4}\right)}{\log\left(1.26\cdot 10^{-1}/3.14\cdot 10^{-2}\right)} \\ =& 2.01. \end{aligned} \tag{4.27}$$

The accuracy is in accord with what was expected, i.e., it is the same as the discretization accuracy of the finite elements method. We conclude that the Neumann boundaries are implemented correctly.

4.3 Test Case 3 — Linear Equation with Nonhom. Dirichlet B. C.

The problem we used to test the implementation of non–homogeneous Dirichlet boundary conditions is

$$-\nabla\cdot((x+1)\nabla u(\boldsymbol{x})) + 10000\,u(\boldsymbol{x}) = 2(x+1)\cos(z)\sin(y) + 10000\cos(z)\sin(y) \tag{4.28}$$

with the boundary conditions

$$\begin{aligned} u(0,y,z) &= \cos(z)\sin(y), \\ u(2\pi,y,z) &= \cos(z)\sin(y), \\ u(x,0,z) &= u(x,2\pi,z), \\ u(x,y,0) &= u(x,y,2\pi). \end{aligned} \tag{4.29}$$

The exact solution is

$$u(x,y,z) = \sin(y)\cos(z). \tag{4.30}$$

Again, the goal is to verify that the solver works with non-homogeneous Dirichlet boundary conditions, so, as in test case 2, only the order of accuracy is investigated. We use the same settings and multigrid parameters which have been used for test case 2, i.e.,

- number of pre– and post–smoothing steps: $\nu_1 = \nu_2 = 1$,
- grid cycle traversal: W-cycling,
- coarse grid accuracy: $\epsilon_{\text{coarse}} = 0.01\,\epsilon_{\text{fine}}$.

The numerical results can be found in tables A.42 and A.43.

4.3.1 Calculating the Order of Accuracy

We obtain for the order p

$$\begin{aligned} p &\approx \frac{\log(e(h_1)/e(h_2))}{\log(h_1/h_2)} \\ &= \frac{\log\left(1.56\cdot 10^{-6}/8.19\cdot 10^{-7}\right)}{\log\left(3.14\cdot 10^{-2}/2.53\cdot 10^{-2}\right)} \\ &= 2.98. \end{aligned} \tag{4.31}$$

While this is unexpected for a second order accurate discretization the equation

$$p \geq \text{ discretization accuracy} \tag{4.32}$$

still holds.

Having investigated the order of the method in serial runs we now turn to the investigation of the behavior of the parallel multigrid solver in the next section.

4.4 Test Case 4 — Parallel Linear Multigrid

4.4.1 Introduction

Not much has been said about the parallelization concept of ANTARES in this thesis. For more details on that topic, see Obertscheider, (2007) for a description of the parallelization concept used in ANTARES and Happenhofer, (2014) for a thorough treatment of parallelization issues when

using multigrid methods in conjunction with ANTARES. A summary on the implementation details for the parallel multigrid solver can be found in Blies, F. Kupka, and H. J. Muthsam, (2015). For a thorough introduction to the concept of parallel scalability, see Hager and Wellein, (2010).

In order to verify the correct implementation of the parallel three–dimensional multigrid solver we have decided to perform two investigations. One of the strong scaling behavior in section 4.4.2 and one of the weak scaling behavior in section 4.4.3.

To test the parallelization properties of the linear solver we use the same problem as in test case 1, i.e.,

$$-\nabla\cdot(x\nabla u(\boldsymbol{x}))+x\,u(\boldsymbol{x}) = 4x\sin(x)\sin(y)\sin(z)-\cos(x)\sin(y)\sin(z) \tag{4.33}$$

with the boundary conditions

$$\begin{aligned} u(0,y,z) &= u(2\pi,y,z) = 0,\\ u(x,0,z) &= u(x,2\pi,z),\\ u(x,y,0) &= u(x,y,2\pi) \end{aligned} \tag{4.34}$$

and the exact solution

$$u(x,y,z) = \sin(x)\sin(y)\sin(z). \tag{4.35}$$

The parallelization tests have been performed on a different system than the serial tests. This means that the wall clock times from this section cannot be compared with those from previous tests.

4.4.2 Investigation of the Strong Scaling Behavior

Strong scaling is defined as how the wall clock time changes with the number of processors used for a fixed total problem size. To determine the strong scaling behavior of the developed multigrid solver we have investigated the wall clock times for solving (4.33) with the settings outlined in table 4.1 and 4.2. The detailed numerical results can be found in tables A.44- A.50.

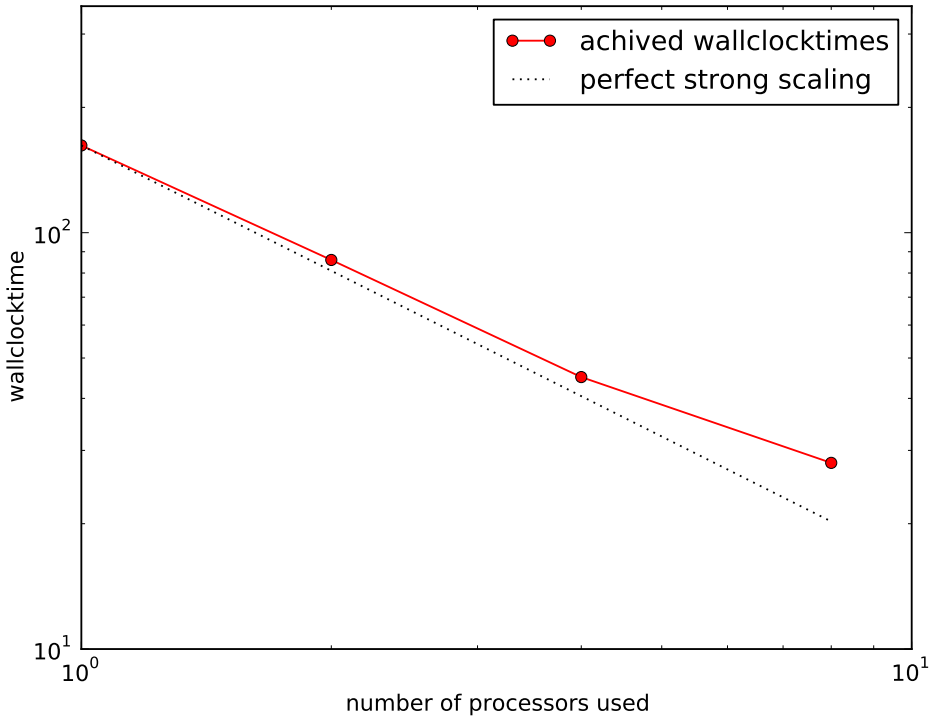
simulation parameters			
levels: 4W, $N_c = 31$, $N_f = 248$, $h = 2.53\cdot10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$			
$\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/100$, system: Sedna			
resolution	number of processors	domain decomposition	wall clock time [s]
$248\times248\times248$	1 (serial run)	n/a	162
$248\times248\times248$	2	2 1 1	86
$248\times248\times248$	4	2 2 1	45
$248\times248\times248$	8	2 2 2	28

Table 4.1: Parameters and wall clock times for the strong scaling experiments for 1-8 processors.

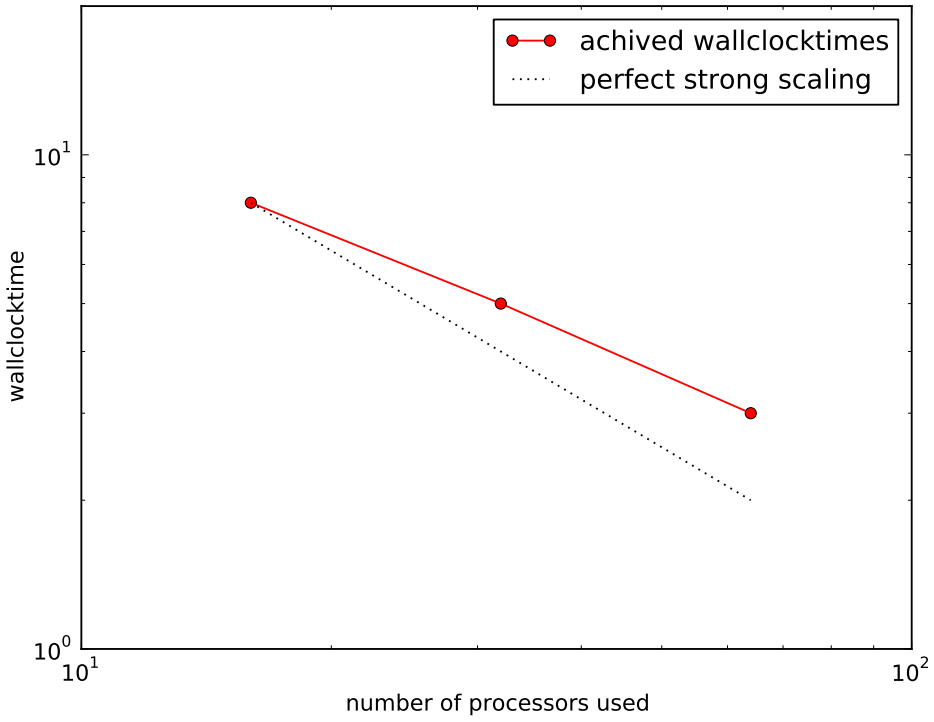
simulation parameters			
levels: 4W, $N_c = 31$, $N_f = 248$, $h = 2.53 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$			
$\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/100$, system: VSC2			
resolution	number of processors	domain decomposition	wall clock time [s]
$248 \times 248 \times 248$	16	4 2 2	8
$248 \times 248 \times 248$	32	4 4 2	5
$248 \times 248 \times 248$	64	4 4 4	3

Table 4.2: Parameters and wall clock times for the strong scaling experiments for 16-64 processors.

The results are visualized in figures 4.16(a) and 4.16(b). Observing these figures we see that the solver has (for this problem with the chosen parameters) a very good strong scaling behavior for a small number of processors and a good strong scaling behavior for a medium number of processors. Note, however, that the wall clock times for many processors are very low, so that a fluctuation of one second has a much higher impact on the results that when the wall clock time is 162 s as in the serial run.



(a) Small number of processors. The simulations have been performed on the system "Sedna".



(b) Medium number of processors. Simulations have been performed on the system "VSC2".

Figure 4.16: Wall clock times vs. number of processors for small and medium number of processors. The dashed line represents perfect strong scaling.

4.4.3 Investigation of the Weak Scaling Behavior

Weak scaling is defined as how the wall clock time changes with the number of processors for a fixed problem size per processor. To investigate the weak scaling behavior of our solver, we have run simulations with the resolutions shown in table 4.3. For each simulation setting, the number of grid points in one specific direction was doubled, as was the number of processors in that direction, so that the amount of work per processor was constant. The wall clock time was measured for each experiment. We repeated the measurement five times for each setting because of the small wall clock times involved.

simulation parameters			
$\nu_1 = 1, \qquad \nu_2 = 1$			
$\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13}), \qquad \epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100, \qquad$ system: VSC2			
resolution	no of processors	domain decomp.	wall clock times [s]
$216 \times 108 \times 108$	16	4 2 2	$(3.21 + 3.18 + 3.17 + 3.19 + 3.24)/5 = 3.20$
$216 \times 216 \times 108$	32	4 4 2	$(4.85 + 4.83 + 4.85 + 4.83 + 4.84)/5 = 4.84$
$216 \times 216 \times 216$	64	4 4 4	$(3.09 + 3.09 + 3.09 + 3.14 + 3.16)/5 = 3.11$
$432 \times 216 \times 216$	128	8 4 4	$(4.99 + 5.10 + 5.05 + 5.06 + 5.14)/5 = 5.07$
$432 \times 432 \times 216$	256	8 8 4	$(7.68 + 7.58 + 7.42 + 7.64 + 7.40)/5 = 7.54$
$432 \times 432 \times 432$	512	8 8 8	$(5.86 + 5.69 + 5.87 + 5.72 + 5.75)/5 = 5.78$

Table 4.3: Parameters and wall clock times for the weak scaling experiments.

Because we repeated each experiment five times, we do not show the exact numerical values in the appendix. Instead, we only show the summary of the results in the table above.

One sees that the wall clock times change only marginally when increasing the resolution and the processor count at the same time, which is evidence of a good weak scaling behavior. The worst weak scaling seems to occur for 256 processors while the wall clock time for 64 processors are even better than the wall clock times for 16 processors. The simulations with 256 processors are the only simulations where the quotient of $T_i/T_{16} > 2$. For all other simulations, this quotient is smaller than 2 which is evidence for the good weak scaling behavior of the solver. The different wall clock times are not caused by the solver exclusively: another factor is the underlying architecture of the VSC2 and the different inter- and intranode communication speeds. Also, this investigation is only viable for problem (4.33). An investigation of how good the scaling behavior is for other problems can be done in the future.

4.5 Test Case 5 — Nonlinear Multigrid with Dirichlet Boundaries

We now turn to the tests of the nonlinear solver. The problem that we solve is

$$\begin{aligned}-\nabla\cdot((5+u(\boldsymbol{x}))\nabla u(\boldsymbol{x}))+u(\boldsymbol{x}) = &-(\cos(z)^2\sin(1+x)^2\sin(y)^2)+\sin(1+x)\sin(y)\sin(z)\\ &-\cos(y)^2\sin(1+x)^2\sin(z)^2-\cos(1+x)^2\sin(y)^2\sin(z)^2\\ &+3\sin(1+x)\sin(y)\sin(z)\cdot(5+\sin(1+x)\cdot\sin(y)\cdot\sin(z))\end{aligned} \tag{4.36}$$

with the boundary conditions

$$\begin{aligned}u(0,y,z) =& u(2\pi,y,z) = \sin(1)\sin(y)\sin(z),\\ u(x,0,z) =& u(x,2\pi,z),\\ u(x,y,0) =& u(x,y,2\pi)\end{aligned} \tag{4.37}$$

and the exact solution

$$u(x,y,z) = \sin(1+x)\,\sin(y)\,\sin(z). \tag{4.38}$$

For the nonlinear solver, only its accuracy is tested here. We use the following parameters which are combined from what we have found out to be best for test case 1 and the number of pre- and post-smoothing steps used by Happenhofer, (2014):

- $\nu_1 = \nu_2 = 5$,
- W-cycling.

4.5.1 Calculating the Order of Accuracy

By the same calculation as in section 4.1.1 we determine the order of accuracy of the nonlinear solver. We derive the order p with the values of tables A.51 and A.52 and obtain

$$\begin{aligned}p \approx& \frac{\log(e(h_1)/e(h_2))}{\log(h_1/h_2)}\\ =& \frac{\log\left(3.94\cdot10^{-4}/6.36\cdot10^{-4}\right)}{\log\left(3.14\cdot10^{-2}/3.83\cdot10^{-2}\right)}\\ =& 2.4.\end{aligned} \tag{4.39}$$

The accuracy is again in accord with what was expected, i.e., it is the same or better as the discretization accuracy of the finite elements method. We conclude that the nonlinear solver with non-homogeneous Dirichlet boundaries is implemented correctly.

One point to note is the wall clock time of the simulations. The run with a resolution of $164\times164\times164$ grid points and three grid levels took 350 s to complete. The run with $200\times200\times200$ grid points and four grid levels only needed 315 s in spite of the fact that there were 1.8 times as

many grid points in the higher refined simulation. This is likely due to the fact that the coarse grid equation has to be solved on a grid with a lower resolution if one uses more grid levels.

This shows again that one should experiment with the multigrid parameters before making a definite parameter choice for a long simulation.

4.6 Test Case 6 — Nonlinear Multigrid with Neumann Boundaries

We now turn to the tests of the nonlinear solver with non-homogeneous Neumann boundaries. The problem that we are solving is

$$\begin{aligned} -\nabla\cdot((10+u(\boldsymbol{x}))\nabla u(\boldsymbol{x}))+(1000+u(\boldsymbol{x})) =& 1000-\sin^2(x)\sin^2(y)\cos^2(z)+\sin(x)\sin(y)\sin(z) \\ &-\sin^2(x)\cos^2(y)\sin^2(z)-\cos^2(x)\sin^2(y)\sin^2(z) \\ &+3\sin(x)\sin(y)\sin(z)(\sin(x)\sin(y)\sin(z)+10) \end{aligned} \tag{4.40}$$

with the boundary conditions

$$\begin{aligned} \frac{\partial u}{\partial x}(0,y,z) &= \frac{\partial u}{\partial x}(2\pi,y,z)=\sin(y)\sin(z), \\ u(x,0,z) &= u(x,2\pi,z), \\ u(x,y,0) &= u(x,y,2\pi) \end{aligned} \tag{4.41}$$

and the exact solution

$$u(x,y,z)=\sin(x)\sin(y)\sin(z). \tag{4.42}$$

It should be noted here that one needs to pay attention when implementing the Neumann boundaries: the general Neumann problem reads
find u such that

$$\begin{cases} -\nabla\cdot(\kappa(u)\nabla u(\boldsymbol{x}))+\xi(u)=f(\boldsymbol{x}) & \text{in } \Omega \\ \nabla u\cdot\boldsymbol{n}=g_N & \text{on } \partial\Omega. \end{cases} \tag{4.43}$$

So the boundary conditions are

$$\nabla u\cdot\boldsymbol{n}=g_N. \tag{4.44}$$

This means that g_N for this problem is given by

$$\begin{aligned} g_{N,\text{lower boundary}} &= -\sin(y)\sin(z) \\ g_{N,\text{upper boundary}} &= \sin(y)\sin(z) \end{aligned} \tag{4.45}$$

because of the direction of $\boldsymbol{n}$.

4.6.1 Calculating the Order of Accuracy

By the same calculation as in section 4.1.1 we determine the order of accuracy of the nonlinear solver. We derive the order p with the values of tables A.53 and A.54 and obtain

$$\begin{aligned} p \approx & \frac{\log(e(h_1)/e(h_2))}{\log(h_1/h_2)} \\ = & \frac{\log\left(5.85 \cdot 10^{-4}/2.32 \cdot 10^{-3}\right)}{\log\left(3.74 \cdot 10^{-2}/7.48 \cdot 10^{-2}\right)} \\ = & 1.99. \end{aligned} \tag{4.46}$$

The accuracy is again in accord with what was expected, i.e., it is the same or better as the discretization accuracy of the finite elements method and we conclude that the nonlinear solver with Neumann boundaries is implemented correctly.

4.7 Test Case 7 — Parallel Nonlinear Multigrid

To test the correctness of the parallelization of the nonlinear multigrid solver, we repeat test case 6 with a domain composition of $2 \times 2 \times 2$, i.e., on 8 processors, so the problem has been stated in 4.40. While it would be interesting to investigate the scaling properties of the nonlinear solver as well, this has to be postponed

to a later time. We only verify the correctness of the implementation and for that, we move to the calculation of the order of accuracy of the method.

4.7.1 Calculating the Order of Accuracy

By the same calculation as in section 4.1.1 we determine the order of accuracy of the nonlinear solver. We derive the order p with the values of tables A.55 and A.56 and obtain

$$\begin{aligned} p \approx & \frac{\log(e(h_1)/e(h_2))}{\log(h_1/h_2)} \\ = & \frac{\log\left(5.89 \cdot 10^{-4}/2.33 \cdot 10^{-3}\right)}{\log\left(3.74 \cdot 10^{-2}/7.48 \cdot 10^{-2}\right)} \\ = & 1.98. \end{aligned} \tag{4.47}$$

The accuracy is again in accord with what was expected, i.e., it is the same or better as the discretization accuracy of the finite elements method and we conclude that the nonlinear solver with Neumann boundaries is implemented correctly.

4.8 Summary

This concludes our tests for the multigrid solver. We have verified the order of accuracy of the solver for linear and nonlinear test problems for which the analytical solutions are known. We

have tested the solver for different boundary conditions and have also tested the correctness of parallelization for both the linear and nonlinear solver.

Every test resulted in an order about equal or better than the discretization accuracy so the conclusion we draw is that the solver has been correctly derived and implemented into the ANTARES framework.

Furthermore, for the test problem with homogeneous Dirichlet boundary conditions, we have tested the influence of various parameters of the multigrid solver on the convergence rates and wall clock times.

For the linear parallel solver, we have done some simple scaling tests to get a hint at the strong and weak scaling properties of the solver.

CHAPTER 5

SUMMARY OF PART I

The problem we started from was the following one: depending on which approach we choose to solve the equations of fluid dynamics with ANTARES, we arrive at a three-dimensional Helmholtz equation with varying coefficients either in linear form,

$$-\nabla\cdot(\kappa(\boldsymbol{x})\nabla u(\boldsymbol{x}))+\xi(\boldsymbol{x})u(\boldsymbol{x})=f(\boldsymbol{x}), \tag{5.1}$$

or in nonlinear form

$$-\nabla\cdot(\kappa(u)\nabla u(\boldsymbol{x}))+\xi(u)=f(\boldsymbol{x}) \tag{5.2}$$

with $u:\mathbb{R}^3\to\mathbb{R}$, $\boldsymbol{x}\in\mathbb{R}^3$, $\kappa:\mathbb{R}^3\to\mathbb{R}^+$, $\xi:\mathbb{R}^3\to\mathbb{R}^+$ and $f:\mathbb{R}^3\to\mathbb{R}$. The boundary conditions can either be of Dirichlet type

$$u=g_D \quad \text{on } \Gamma_D \tag{5.3}$$

or of Neumann type

$$\nabla u\cdot\boldsymbol{n}=\frac{\partial u}{\partial n}=g_N \quad \text{on } \Gamma_N. \tag{5.4}$$

The equations are to be solved over a rectangular domain $\Omega\subset\mathbb{R}^3$.

Since ANTARES was not able to handle this kind of equation in three dimensions prior to this thesis, deriving and implementing a solver for this equation was the topic of this part. The part was subdivided into three sections: discretization of the equation with the finite elements method, the derivation of the multigrid components and extensive tests of the implemented solver.

Discretization

We started from the discretization of the linear and nonlinear equations with the finite elements method in chapter 2 in order to obtain a (non-)linear system of equations. The choices we have made for the finite elements, denoted by $(K,\mathcal{Q},\mathcal{N})$, are

- for the element domain $K \subseteq \mathbb{R}^n$: since our simulations are run in a rectangular domain with straight boundaries we can use one of the easiest choices for K: rectangular cuboids;

- for the finite–dimensional space of shape functions $\mathscr{Q}$ on K: since our domain Ω is perfectly rectangular, a choice of $k = 1$ leads to a sufficiently accurate approximation, so we use trilinear polynomials for our finite element discretization:

$$\mathscr{P} = \mathscr{Q}_1(\mathbb{R}^3) = \text{span}\{1, x, y, z, xy, xz, yz, xyz\}; \tag{5.5}$$

- for the set of $n = \dim(\mathscr{P})$ nodal variables $\mathscr{N} = \{N_1, N_2, ..., N_k\}$ which define the degrees of freedom: a standard choice for Lagrangian degrees of freedom on $\mathscr{P} = \mathscr{Q}_1(\mathbb{R}^3)$ are the 8 corners of the cubicle.

Solving the Discretized System with Multigrid Methods

To solve the (non-)linear system of equations that resulted from the discretization we have used the multigrid approach. To that end we have derived the necessary multigrid components for both the linear and the nonlinear system of equations and have implemented the algorithm into the ANTARES framework. The choices we have made for the multigrid components are

- smoothing routine: one of the best iterative schemes for multigrid smoothing is the Gauss–Seidel method because of its superior smoothing properties. It is implemented in ANTARES in the lexicographic variant only.

- prolongation operator: for Finite Element Methods the prolongation is defined through the natural inclusion of the coarse grid subspace V_{2h} into the fine grid subspace V_h. This results in the canonical trilinear prolongation operator.

- restriction operator: the restriction operator is canonically defined for a discretization with the finite elements method. It coincides with the three–dimensional full–weighting approach.

- the coarse grid operator: it was obtained via the Galerkin coarse grid approximation where it is constructed with the help of the prolongation and the restriction operators.

- the coarse grid solver: we use a Conjugate Gradient method on the coarsest grid to solve the system of coarse-grid equations.

- for the nonlinear system of equations we have linearized the system with the Newton scheme.

Testing the Implemented Solver

After having implemented the multigrid solver into ANTARES we have verified the order of accuracy of the solver for linear and nonlinear test problems for which the analytical solutions are known. We have tested the solver for different boundary conditions and have also tested the correctness of parallelization for both the linear and nonlinear solver. Furthermore, we have done a preliminary study of the strong and weak scaling properties of the solver with very satisfactory results.

In addition, for the test problem with homogeneous Dirichlet boundary conditions, we have investigated the influence of various parameters of the multigrid solver on the convergence rates and wall clock times and have arrived at the following conclusions: for the purpose of solving (4.3), the following parameters yield the best results in terms of wall clock times:

- number of pre– and post–smoothing steps: $\nu_1 = \nu_2 = 1$
- grid cycle traversal: W-cycling for more than 2 levels
- coarse grid accuracy: $\epsilon_{\text{coarse}} = 0.01\epsilon_{\text{fine}}$

We point out that each problem treated with the solver will likely have a different set of best parameters values. They will have to be adjusted by each user separately for the kind of problem they want to tackle.

Conclusion

We have successfully derived, developed and implemented a multigrid solver for the three-dimensional, (non-)linear Helmholtz equation with varying coefficients and thereby considerably extended the possible areas of application of ANTARES.

Part II

Implicit–Explicit Runge–Kutta Methods for Incompressible Flows

CHAPTER 6

THEORY AND DERIVATION OF THE METHOD

6.1 Introduction

As we have mentioned in the introduction, the time scales of the different physical processes are of major importance when trying to conduct efficient numerical simulations of these systems. We are looking at the Navier–Stokes equations in the Boussinesq approximation which have been presented in section 1.2.2. We reprint them in their 3D, non–dimensionalized form for the reader's convenience:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{6.1}$$

$$\frac{\partial u}{\partial t} + \boldsymbol{u} \cdot \nabla u = \mathrm{Pr}\left[-\frac{\partial p_{\mathrm{eff}}}{\partial x} + \mathrm{Ra_T}\, T + \mathrm{Ra_S}\, S + \nabla^2 u \right] \tag{6.2}$$

$$\frac{\partial v}{\partial t} + \boldsymbol{u} \cdot \nabla v = \mathrm{Pr}\left[-\frac{\partial p_{\mathrm{eff}}}{\partial y} + \nabla^2 v \right] \tag{6.3}$$

$$\frac{\partial w}{\partial t} + \boldsymbol{u} \cdot \nabla w = \mathrm{Pr}\left[-\frac{\partial p_{\mathrm{eff}}}{\partial z} + \nabla^2 w \right] \tag{6.4}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \nabla^2 T \tag{6.5}$$

$$\frac{\partial S}{\partial t} + \boldsymbol{u} \cdot \nabla S = \mathrm{Le} \nabla^2 S \tag{6.6}$$

with

$$\mathrm{Pr} = \frac{\nu}{\kappa_T}, \quad \mathrm{Le} = \frac{\kappa_S}{\kappa_T}, \quad Ra_T = \frac{\alpha L^3 \Delta T g}{\kappa_T \nu}, \quad Ra_S = \frac{\beta L^3 \Delta S g}{\kappa_T \nu}. \tag{6.7}$$

We also define

$$\mathrm{R}_\rho = \frac{\mathrm{Ra_T}}{\mathrm{Ra_S}}. \tag{6.8}$$

According to F. Kupka et al., (2012), the following time step restriction occur due to the physical processes which the equations model:

$$\text{advection:} \qquad \Delta t_{\text{ad}} = c_{\text{ad}} \frac{\min(\Delta x, \Delta y, \Delta z)}{|\boldsymbol{u}|}, \tag{6.9}$$

$$\text{diffusion of heat:} \qquad \Delta t_T = c_{\text{diff}} \min(\Delta x^2, \Delta y^2, \Delta z^2), \tag{6.10}$$

$$\text{diffusion of solute:} \qquad \Delta t_S = c_{\text{diff}} \frac{\min(\Delta x^2, \Delta y^2, \Delta z^2)}{\text{Le}}, \tag{6.11}$$

$$\text{viscous stresses:} \qquad \Delta t_{\text{visc}} = c_{\text{diff}} \frac{\min(\Delta x^2, \Delta y^2, \Delta z^2)}{\text{Pr}}. \tag{6.12}$$

Here, c_{ad} and c_{diff} are the advective and diffusive Courant numbers, respectively. When using explicit time integration, in order for the numerical scheme to be stable, all processes must be temporally resolved and hence the overall applicable time step is

$$\Delta t_{\text{explicit}} = \min(\Delta t_{\text{ad}}, \Delta t_{\text{T}}, \Delta t_{\text{S}}, \Delta t_{\text{visc}}). \tag{6.13}$$

In other words, the maximal possible timestep reduces whenever the Lewis number is large, the Prandtl number is large or the fluid velocity is high. The dependence of Δt on the diffusivity of heat does not appear here in explicit form due to the non-dimensionalization (t is measured in units of t_{T}. Usually, one is interested in solutions which change according to the advective time scale but there are many cases where other time scales are smaller. For these flows, various methods have been invented to alleviate the time step restrictions by the other processes.

Our methods of choice are SSP IMEX schemes. When treating the diffusive parts of the equations implicitly, depending on the method the diffusive time scale plays no part in limiting the time step any more except for cases where the solution indeed changes on such a time scale) and thus the overall time step is given by

$$\Delta t_{\text{diffusive implicit}} = \min(\Delta t_{\text{ad}}, \Delta t_{\text{visc}}). \tag{6.14}$$

When treating the diffusive and viscous parts of the equations implicitly the theoretically achievable maximum time step increases even further to

$$\Delta t_{\text{diffusive and viscous implicit}} = \Delta t_{\text{ad}}. \tag{6.15}$$

Again, here we assume the most common astrophysical case where significant (large relative) changes of the solution do not occur for $\Delta t < \Delta t_{\text{ad}}$. So the time step is only limited by the advective terms, i.e., we have eliminated every time step restriction that was introduced by the parabolic nature of diffusive and viscous processes.

One important consideration to keep in mind when comparing time step restrictions of different time integration schemes is that each scheme has its own limit on both the advective and the diffusive Courant numbers because of different stability regions. More stable schemes permit higher Courant numbers – and thus time steps – than schemes with smaller stability

domains. See Happenhofer, (2014) and F. Kupka et al., (2012) and references therein for details on the stability regions of the IMEX schemes. We give both advective and diffusive Courant numbers in the numerical experiments to come.

Prior to this thesis, ANTARES had no implicit scheme implemented for the Boussinesq approximation. Although a variety of explicit methods are available, implicit methods had not been implemented and so the maximal time step for simulations in the Boussinesq approximation has been

$$\Delta t_{\text{Boussinesq, before thesis}} = \min(\Delta t_{\text{ad}}, \Delta t_{\text{T}}, \Delta t_{\text{S}}, \Delta t_{\text{visc}}), \tag{6.16}$$

even for stiff equations.

The case is different for the fully compressible Navier–Stokes equations. For these equations, F. Kupka et al., (2012) have introduced total-variation-diminishing implicit-explicit Runge–Kutta methods for flows which are limited by diffusive processes. Happenhofer, (2014) has implemented these into ANTARES. That means that for compressible equations, the smallest time step with these new methods is given by

$$\Delta t_{\text{compressible}} = \min(\Delta t_{\text{ad}}, \Delta t_{\text{visc}}). \tag{6.17}$$

In this part of the thesis we closely follow F. Kupka et al., (2012) in deriving the method for the Boussinesq approximation and Happenhofer, (2014) for its implementation into ANTARES. This chapter focuses on the theoretical side of the method: in section 6.2 we present the derivation of the Runge–Kutta equations for the method, section 6.3 gives a few implementation details. Chapter 7 covers numerical experiments with the method and a discussion of the results.

6.2 Deriving the SSP IMEX RK Equations for the Boussinesq Approximation

6.2.1 Runge–Kutta Methods for Stiff Equations

In ANTARES the method of lines approach is used to transform the Boussinesq equations (6.1) - (6.6), which are a system of partial differential equations, to the ordinary differential equation initial value problem

$$\begin{aligned} \frac{\partial y(t)}{\partial t} &= F(y(t)), \\ y(0) &= y_0. \end{aligned} \tag{6.18}$$

If F can be partitioned into a stiff and non-stiff part, this can be written as

$$\begin{aligned} \frac{\partial y(t)}{\partial t} &= F(y(t)) + G(y(t)), \\ y(0) &= y_0 \end{aligned} \tag{6.19}$$

where $F(y(t))$ and $G(y(t))$ have different stiffness properties (for a short introduction to stiff equations, see section 1.3, for a thorough treatment see (Hairer, 1993)).

Partitioned Runge–Kutta methods are a popular choice for treating problems with the structure of (6.19). With these methods, F is integrated explicitly and G is integrated implicitly. In general, an s-stage partitioned Runge–Kutta method characterized by coefficient matrices $A = (a_{ij})$ and $\tilde{A} = (\tilde{a}_{ij})$ defines one step $y_{old} \rightarrow y_{new}$ by

$$\begin{aligned} y_i &= y_{\text{old}} + \Delta t \sum_{j=1}^{s} a_{ij} F(y_j) + \Delta t \sum_{j=1}^{s} \tilde{a}_{ij} G(y_j), \quad i = 1, ..., s \\ y_{\text{new}} &= y_{\text{old}} + \Delta t \sum_{j=1}^{s} b_j F(y_j) + \Delta t \sum_{j=1}^{s} \tilde{b}_j G(y_j). \end{aligned} \tag{6.20}$$

The coefficient matrices are represented by Butcher tableaus in the following way. a_{ij} and b_{ij} are given by

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline A & b_1 & b_2 & \cdots & b_s \end{array} \tag{6.21}$$

and $\tilde{a}_{ij}$ and $\tilde{b}_{ij}$ are given by

$$\begin{array}{c|cccc} \tilde{c}_1 & \tilde{a}_{11} & \tilde{a}_{12} & \cdots & \tilde{a}_{1s} \\ \tilde{c}_2 & \tilde{a}_{21} & \tilde{a}_{22} & \cdots & \tilde{a}_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{c}_s & \tilde{a}_{s1} & \tilde{a}_{s2} & \cdots & \tilde{a}_{ss} \\ \hline \tilde{A} & \tilde{b}_1 & \tilde{b}_2 & \cdots & \tilde{b}_s \end{array} . \tag{6.22}$$

The coefficients c_i and $\tilde{c}_i$ are just the sums of the corresponding rows such that $c_i = \sum_{j=1}^{s} a_{ij}$, $i = 1, 2, ..., s$ and $\tilde{c}_i = \sum_{j=1}^{s} \tilde{a}_{ij}$, $i = 1, 2, ..., s$.

If $a_{ij} = 0$ for $j \geq i$, the method is referred to as an *implicit-explicit* (IMEX) method. The tableaus determine the stability properties of the method. Many different methods exist in the literature with different tableaus.

The strong-stability preserving (SSP) property of an IMEX method is used to suppress spurious oscillations in the spatial discretization and has been demonstrated to be necessary for stable integration by Gottlieb, Ketcheson, and Shu, (2009). We follow the standard convention in nomenclature for SSP IMEX methods: an SSP IMEX method is referred to as SSP$k(s, \sigma, p)$ when is has the following properties: k is the order of the method in the stiff limit, i.e., when the non-stiff part F is negligible. s is the number of stages in the implicit scheme and σ the number of stages in the explicit scheme. p is the global order of the resulting combined method.

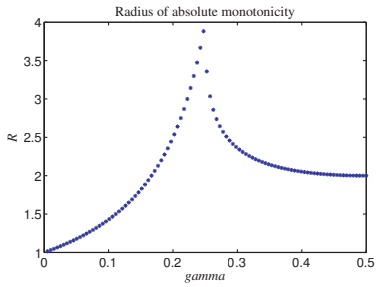


Figure 6.1: Radius of absolute monotonicity $\mathscr{R}(\tilde{A})$ as a function of γ for (6.23) (taken from F. Kupka et al., 2012)

6.2.2 The IMEX SSP2(2,2,2) Method

In this thesis, the SSP2$(2,2,2)$ method of Pareschi and Russo, (2005) with adjustments made by F. Kupka et al., (2012) is implemented. It is characterized by the Butcher tableaus

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline A & 1/2 & 1/2 \end{array} \quad \text{and} \quad \begin{array}{c|cc} \gamma & \gamma & 0 \\ 1-\gamma & 1-2\gamma & \gamma \\ \hline \tilde{A} & 1/2 & 1/2 \end{array} . \tag{6.23}$$

where Pareschi and Russo, (2005) used $\gamma = 1 - 1/\sqrt{2}$. F. Kupka et al., (2012) have adapted the value of γ according to the resulting stability, accuracy and dissipativity properties. E.g., fig. 6.1 shows the relationship that γ has on the radius of absolute monotonicity of the method. F. Kupka et al., (2012) suggested to optimize γ by taking it as large as necessary to avoid any linear stability restrictions and as small as possible to minimize the stability constant. They have found that the choice $\gamma = 0.24$ yielded the most efficient time integrator for the applications they were interested in and hence this is the method which is used in this thesis. The Butcher tableaus of this method read

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline A & 1/2 & 1/2 \end{array} \quad \text{and} \quad \begin{array}{c|cc} 0.24 & 0.24 & 0 \\ 0.76 & 0.52 & 0.24 \\ \hline \tilde{A} & 1/2 & 1/2 \end{array} . \tag{6.24}$$

6.2.3 Application to the Boussinesq Equations

Writing the Boussinesq equations (1.29)-(1.32) in the form of (6.19) yields

$$\frac{\partial}{\partial t}\underbrace{\begin{pmatrix} 0 \\ u \\ v \\ w \\ T \\ S \end{pmatrix}}_{y(t)} = \underbrace{\begin{pmatrix} \nabla\cdot\boldsymbol{u} \\ -\boldsymbol{u}\cdot\nabla u + \mathrm{Pr}\left(-\partial_x p_{\mathrm{eff}} + \mathrm{Ra_T}\,T + \mathrm{Ra_S}\,S + \nabla^2 u\right) \\ -\boldsymbol{u}\cdot\nabla v + \mathrm{Pr}(-\partial_y p_{\mathrm{eff}} + \nabla^2 v) \\ -\boldsymbol{u}\cdot\nabla w + \mathrm{Pr}(-\partial_z p_{\mathrm{eff}} + \nabla^2 w) \\ -\boldsymbol{u}\cdot\nabla T + \nabla^2 T \\ -\boldsymbol{u}\cdot\nabla S + \mathrm{Le}\nabla^2 S \end{pmatrix}}_{F(y(t))} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{G(y(t))} \tag{6.25}$$

In this form, no part of the equations has been declared as stiff, so the usual explicit methods can be applied. It now depends on the dimensionless numbers Pr and Le which time scale is the most restrictive one and depending on those numbers, the system (6.25) is either written as

$$\frac{\partial}{\partial t}\underbrace{\begin{pmatrix} 0 \\ u \\ v \\ w \\ T \\ S \end{pmatrix}}_{y(t)} = \underbrace{\begin{pmatrix} \nabla\cdot\boldsymbol{u} \\ -\boldsymbol{u}\cdot\nabla u + \mathrm{Pr}\left(-\partial_x p_{\mathrm{eff}} + \mathrm{Ra_T}\,T + \mathrm{Ra_S}\,S + \nabla^2 u\right) \\ -\boldsymbol{u}\cdot\nabla v + \mathrm{Pr}(-\partial_y p_{\mathrm{eff}} + \nabla^2 v) \\ -\boldsymbol{u}\cdot\nabla w + \mathrm{Pr}(-\partial_z p_{\mathrm{eff}} + \nabla^2 w) \\ -\boldsymbol{u}\cdot\nabla T \\ -\boldsymbol{u}\cdot\nabla S \end{pmatrix}}_{F(y(t))} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \nabla^2 T \\ \mathrm{Le}\nabla^2 S \end{pmatrix}}_{G(y(t))} \tag{6.26}$$

if the diffusive time scale of concentration – governed by Le – is smallest and as

$$\frac{\partial}{\partial t}\underbrace{\begin{pmatrix} 0 \\ u \\ v \\ w \\ T \\ S \end{pmatrix}}_{y(t)} = \underbrace{\begin{pmatrix} \nabla\cdot\boldsymbol{u} \\ -\boldsymbol{u}\cdot\nabla u + \mathrm{Pr}\left(-\partial_x p_{\mathrm{eff}} + \mathrm{Ra_T}\,T + \mathrm{Ra_S}\,S\right) \\ -\boldsymbol{u}\cdot\nabla v + \mathrm{Pr}(-\partial_y p_{\mathrm{eff}}) \\ -\boldsymbol{u}\cdot\nabla w + \mathrm{Pr}(-\partial_z p_{\mathrm{eff}}) \\ -\boldsymbol{u}\cdot\nabla T \\ -\boldsymbol{u}\cdot\nabla S \end{pmatrix}}_{F(y(t))} + \underbrace{\begin{pmatrix} 0 \\ \mathrm{Pr}\nabla^2 u \\ \mathrm{Pr}\nabla^2 v \\ \mathrm{Pr}\nabla^2 w \\ \nabla^2 T \\ \mathrm{Le}\nabla^2 S \end{pmatrix}}_{G(y(t))} \tag{6.27}$$

if both the diffusive and viscous time scales are the limiting terms.

One remark on the pressure: when using the Boussinesq equations in ANTARES, the pressure is calculated with a pressure correction method (see (F. Zaussinger, 2010)) which means that the pressure is calculated implicitly as well. This is done independently from this splitting, however, so that we declare it as belonging to the explicit part $F(y(t))$. This has no influence on our Runge–Kutta algorithm here.

We derive the equations to be solved for the case of using implicit integration for the diffusive terms only (system (6.26)) first and then proceed to the derivation of the equations for the case of implicit integration of diffusive and viscous terms (system (6.27)).

6.2.4 Implicit Integration of Diffusive Terms Only

From (6.20) we see that the i-th implicit stage of a diagonal IMEX-Runge-Kutta scheme is given by

$$y_i = y^* + \Delta t \tilde{a}_{i,i} G(y_i), \tag{6.28}$$

where y^* is known from previous stages. Applying this to (6.26) leads to

$$\begin{aligned}
u_i &= u^*, && (6.29)\\
v_i &= v^*, && (6.30)\\
w_i &= w^*, && (6.31)\\
T_i &= T_i^* + \Delta t \tilde{a}_{i,i} \nabla^2 T_i, && (6.32)\\
S_i &= S_i^* + \Delta t \tilde{a}_{i,i} \mathrm{Le} \nabla^2 S_i. && (6.33)
\end{aligned}$$

Rewriting eq. (6.32) and (6.33) yields

$$-\nabla^2 T_i + \frac{T_i}{\Delta t \tilde{a}_{i,i}} = \frac{T^*}{\Delta t \tilde{a}_{i,i}}, \tag{6.34}$$

$$-\nabla^2 S_i + \frac{S_i}{\mathrm{Le}\,\Delta t \tilde{a}_{i,i}} = \frac{S^*}{\mathrm{Le}\,\Delta t \tilde{a}_{i,i}}. \tag{6.35}$$

These are generalized Helmholtz equations of the form

$$-\nabla \cdot (\kappa(\boldsymbol{x}) \nabla \phi(\boldsymbol{x})) + \xi(\boldsymbol{x}) \phi(\boldsymbol{x}) = f(\boldsymbol{x}) \tag{6.36}$$

with

$$\begin{aligned}
\kappa(\boldsymbol{x}) &= 1,\\
\xi(\boldsymbol{x}) &= \frac{1}{\Delta t \tilde{a}_{i,i}},\\
f(\boldsymbol{x}) &= \frac{T^*}{\Delta t \tilde{a}_{i,i}},\\
\phi(\boldsymbol{x}) &= T_i
\end{aligned} \tag{6.37}$$

for the temperature equation and

$$\begin{aligned}
\kappa(\boldsymbol{x}) &= 1,\\
\xi(\boldsymbol{x}) &= \frac{1}{\mathrm{Le}\Delta t \tilde{a}_{i,i}},\\
f(\boldsymbol{x}) &= \frac{S^*}{\mathrm{Le}\Delta t \tilde{a}_{i,i}},\\
\phi(\boldsymbol{x}) &= S_i
\end{aligned} \tag{6.38}$$

for the salinity equation. In ANTARES, we can use the multigrid solver from Happenhofer, (2014) to solve these equations for up to two dimensions and the multigrid solver from part I of this thesis to solve the equations in three dimensions.

6.2.5 Implicit Integration of Diffusive and Viscous Terms

Repeating this process for the case of diffusively and viscously limited flows, i.e., using (6.27) as our starting point, gives

$$u_i = u^* + \Delta t \tilde{a}_{i,i} \mathrm{Pr} \nabla^2 u_i, \tag{6.39}$$

$$v_i = v^* + \Delta t \tilde{a}_{i,i} \mathrm{Pr} \nabla^2 v_i, \tag{6.40}$$

$$w_i = w^* + \Delta t \tilde{a}_{i,i} \mathrm{Pr} \nabla^2 w_i, \tag{6.41}$$

$$T_i = T_i^* + \Delta t \tilde{a}_{i,i} \nabla^2 T_i, \tag{6.42}$$

$$S_i = S_i^* + \Delta t \tilde{a}_{i,i} \mathrm{Le} \nabla^2 S_i. \tag{6.43}$$

Rewriting the equations gives

$$-\nabla^2 u_i + \frac{u_i}{\mathrm{Pr}\,\Delta t \tilde{a}_{i,i}} = \frac{u^*}{\mathrm{Pr}\,\Delta t \tilde{a}_{i,i}}, \tag{6.44}$$

$$-\nabla^2 v_i + \frac{v_i}{\mathrm{Pr}\,\Delta t \tilde{a}_{i,i}} = \frac{v^*}{\mathrm{Pr}\,\Delta t \tilde{a}_{i,i}}, \tag{6.45}$$

$$-\nabla^2 w_i + \frac{w_i}{\mathrm{Pr}\,\Delta t \tilde{a}_{i,i}} = \frac{w^*}{\mathrm{Pr}\,\Delta t \tilde{a}_{i,i}}, \tag{6.46}$$

$$-\nabla^2 T_i + \frac{T_i}{\Delta t \tilde{a}_{i,i}} = \frac{T^*}{\Delta t \tilde{a}_{i,i}}, \tag{6.47}$$

$$-\nabla^2 S_i + \frac{S_i}{\mathrm{Le}\,\Delta t \tilde{a}_{i,i}} = \frac{S^*}{\mathrm{Le}\,\Delta t \tilde{a}_{i,i}}. \tag{6.48}$$

All of these are generalized Helmholtz equations of the form

$$-\nabla \cdot (\kappa(\boldsymbol{x}) \nabla \phi(\boldsymbol{x})) + \xi(\boldsymbol{x}) \phi(\boldsymbol{x}) = f(\boldsymbol{x}) \tag{6.49}$$

with

$$\begin{aligned} \kappa(\boldsymbol{x}) &= 1, \\ \xi(\boldsymbol{x}) &= \frac{1}{\Delta t \tilde{a}_{i,i}}, \\ f(\boldsymbol{x}) &= \frac{T^*}{\Delta t \tilde{a}_{i,i}}, \\ \phi(\boldsymbol{x}) &= T_i \end{aligned} \tag{6.50}$$

for the temperature equation,

$$\begin{aligned} \kappa(\boldsymbol{x}) &= 1, \\ \xi(\boldsymbol{x}) &= \frac{1}{\mathrm{Le}\,\Delta t \tilde{a}_{i,i}}, \\ f(\boldsymbol{x}) &= \frac{S^*}{\mathrm{Le}\,\Delta t \tilde{a}_{i,i}}, \\ \phi(\boldsymbol{x}) &= S_i \end{aligned} \tag{6.51}$$

for the salinity equation and

$$\begin{aligned}\kappa(\boldsymbol{x}) &= 1,\\ \xi(\boldsymbol{x}) &= \frac{1}{\mathrm{Pr}\,\Delta t\tilde{a}_{i,i}},\\ f(\boldsymbol{x}) &= \frac{q^*}{\mathrm{Pr}\,\Delta t\tilde{a}_{i,i}},\\ \phi(\boldsymbol{x}) &= q_i\end{aligned} \tag{6.52}$$

for the velocity component equations, q being placeholder for u, v or w.

Again, we can use the multigrid solver from Happenhofer, (2014) to solve the two–dimensional case and the multigrid solver from part I of this thesis to solve the three–dimensional case. Note that in the case of implicit integration of both viscous and diffusive terms there are five elliptic equations to be solved at every stage of the Runge–Kutta scheme. In the case of two-stage Runge–Kutta methods that is ten elliptic equations per time step (not counting the elliptic equations which must be solved for the pressure correction). Although the time step can be chosen to be larger if using implicit methods, it remains to be seen if the overall computational efficiency can be increased at all by this approach because ten calls per time step to the elliptical solver are computationally expensive. We conduct some numerical experiments in chapter 7 to determine if and when the IMEX method should be used.

6.3 Implementation Details of the IMEX Method

6.3.1 Time Step Control

Similar to Happenhofer, (2014), we have used a criterion based on observed two-point instabilities to determine the maximal time step. If using implicit integration for the diffusive as well as the viscous terms, there is no time step limit left except the advective one. In the diffusive phase, the fluid velocity is minimal, however, so the maximum time step for which the numerical scheme is still stable is huge compared to Δt_{ad}. Accuracy considerations have to help to make sure that the time step does not become to large. For that matter, the variables u, v, w, T, S are observed and their difference between adjacent grid cells is calculated every time step by calculating

$$\begin{aligned}d_1 &= q_{i,j-1,k} - q_{i,j-2,k}\\ d_2 &= q_{i,j,k} - q_{i,j-1,k}\\ d_3 &= q_{i,j+1,k} - q_{i,j,k}\\ d_4 &= q_{i,j+2,k} - q_{i,j+1,k}\end{aligned} \tag{6.53}$$

and

$$\begin{aligned} d_5 &= q_{i,j,k-1} - q_{i,j,k-2} \\ d_6 &= q_{i,j,k} - q_{i,j,k-1} \\ d_7 &= q_{i,j,k+1} - q_{i,j,k} \\ d_8 &= q_{i,j,k+2} - q_{i,j,k+1} \end{aligned} \tag{6.54}$$

where q is one of the conserved variables u, v, w, T, S and $1 < i < n_x$, $1 < j < n_y$ and $1 < k < n_z$ for a computational grid of $n_x \times n_y \times n_y$ grid points. If the sign pattern of (d_1, d_2, d_3, d_4) or (d_5, d_6, d_7, d_8) corresponds to $(+,-,+,\pm)$, $(-,+,-,\pm)$, $(\pm,+,-,+)$ or $(\pm,-,+,-)$ a two-point instability has been found. This is an indication for a time step that is too large. Happenhofer, (2014) has chosen a limit of $n_y \times 0.1$ two-point instabilities for the two-dimensional implementation of the IMEX routines. We use the same limit in three dimensions, i.e., the limit for two-point instabilities is set to $n_y \times n_z \times 0.1$. If this limit is exceeded, the calculation of the current time step is repeated with a time step of $\Delta t_{\text{new}} = 2/3 \times \Delta t_{\text{old}}$. Contrary to Happenhofer, (2014), we do not allow the system to readjust for 15 time steps but reduce the time step until the number of two-point instabilities stays below the limit. Numerical tests during the development phase have shown that letting the system readjust for 15 time steps produces visual artifacts in the case of implicit integration of viscous and diffusive terms. This is a clear sign of a time step which is too big.

6.3.2 The Algorithm

The algorithm, on the other hand, is exactly the same as Happenhofer, (2014). In the following summary s denotes an arbitrary stage:

1. At the start, $y_{\text{old}}^{(s)} = [u^*, v^*, w^*, T^*, S^*]^{\text{T}}$ is given.
2. Evaluation of the implicit stage (6.28):
   - for DIFF_IMEX, (6.34) and (6.35) are calculated with one of the elliptical solvers in ANTARES. This results in $y_{\text{imp}}^{(s)} = [u^*, v^*, w^*, T_{\text{imp}}, S_{\text{imp}}]^{\text{T}}$
   - for VISC_IMEX, (6.44) to (6.48) are calculated with one of the elliptical solvers in ANTARES. This results in $y_{\text{imp}}^{(s)} = [u_{\text{imp}}, v_{\text{imp}}, w_{\text{imp}}, T_{\text{imp}}, S_{\text{imp}}]^{\text{T}}$
3. Explicit sweep:

$$y_{\text{exp}}^{(s)} = y_{\text{old}}^{(s)} + \Delta t F(y_{\text{imp}}^{(s)}) \tag{6.55}$$

4. Assembly of the final stage values:
   The computed values $y_{\text{exp}}^{(s)}$ and $y_{\text{imp}}^{(s)}$ must be assembled to obtain the final values for the stage considered. To this end (6.20) gives for the stage s:

$$y_{\text{sum}}^{(s)} = y_{\text{old}}^{(s)} + b_i (y_{\text{exp}}^{(s)} - y_{\text{old}}^{(s)}) + \frac{\tilde{b}_i}{\tilde{a}_{ii}} (y_{\text{imp}}^{(s)} - y_{\text{old}}^{(s)}) \tag{6.56}$$

5. Assembly of the new $y_{\text{old}}^{(s+1)}$:
   If s equals the total number of stages S, $y_{\text{new}} = y_{\text{sum}}^{(S)}$ and a new step may be started. If s does not equal the total number of stages, $y_{\text{old}}^{(s+1)}$ must be assembled s.t. according to definition (6.20)
   $$y_{\text{imp}}^{(s+1)} = y_{\text{old}}^{(s+1)} + \Delta t \tilde{a}_{s+1,s+1} G(y_{\text{imp}}^{(s+1)}) \tag{6.57}$$

6. Repeat 1.-5 for each time step in the simulation.

A remark on the explicit sweep in step 3: in the Boussinesq approximation, the pressure is also treated implicitly. That means that the explicit sweep (represented by $F(\cdot)$) is only explicit in the sense of the Runge–Kutta time integration scheme presented here but not explicit in the general ANTARES scheme. For details on how the Boussinesq approximation is implemented in ANTARES, see F. Zaussinger, (2010).

As was the case with the multigrid solver in part I the amount of programming necessary to implement the solver into the ANTARES framework to the point that it worked flawlessly cannot be represented in this written thesis, but the programming work has been a significant part of this Ph.D. thesis.

After having implemented the solver we are now ready to use the method with ANTARES to perform numerical experiments.

CHAPTER 7

NUMERICAL EXPERIMENTS WITH IMEX SSP2(2,2,2)

7.1 Outline

The experimental part of part II is structured the following way: first, we describe the experimental setup used for the numerical tests in this chapter in section 7.2. Then, we perform simulations with different time integration routines first in two dimensions (section 7.3) and then in three dimensions (section 7.5). A discussion of the results follows in section 7.6 and we conclude with a summary in 7.7.

7.2 The Experimental Setup

Again, we refer to F. Kupka et al., (2012) in order to define the problem which we use to investigate the IMEX methods for the Boussinesq approximation. In summary, the test problem is the following: we specify a hydrostatic configuration with is unstable against convection. The mean molecular weight is linearly and stably stratified. As time evolves, we expect convection to set in and mix the zone completely. For the purposes of investigating the IMEX methods, we are only interested in the diffusive phase, i.e., the phase before the onset of convection: once convection sets in, the fluid velocity increases and (6.9) is the limit on the time step so that implicit methods yield no benefits over explicit schemes. Each experiment was conducted with an explicit method and with the newly developed implicit methods. As explicit method, we have chosen to use the explicit SSPRK(3,2) method, which was first presented by Kraaijevanger, (1991) and is considered to be the optimal second-order scheme with three stages. Its diffusive Courant number is 0.6726, we use a safety factor of 0.5 which leads to an effective diffusive Courant number of 0.336. The advective Courant number was chosen to be 0.3. As implicit methods, we used the IMEX SSP2$(2,2,2)$, $\gamma = 0.24$ method derived in the last chapter, once in the form where

only the diffusive terms are integrated implicitly, called DIFF_IMEX from here on and once in the version where the diffusive and viscous terms are integrated implicitly, known as VISC_IMEX henceforth. The diffusive Courant number of this method is 0.375. With the same safety factor of 0.5, this gives an effective diffusive Courant number of 0.225. The advective Courant number was chosen to be 0.25.

We see that the explicit method has two advantages in terms of computational work already: first, the time step restrictions are inherently not as strict due to higher Courant numbers of the used explicit method. However, this is partially compensated by SSP(3,2) requiring 3 stages per time step. Secondly, the explicit methods do not need to use the elliptical solver (except for the pressure terms in the Boussinesq approximation which are not considered here) which saves a significant amount of computational work. We see if these two initial disadvantages can be overcome by the larger time step which are allowed by the implicit part of the methods.

The runtime of most of our simulations has been limited to 0.03 thermal diffusion time scales. After this time convection has already set in for the choice of our simulation parameters. Since we are interested in the possible gain through IMEX methods which only have an effect in the diffusive phase, the convective phase is of no interest for us, so we neglect it and stop the simulation when convection has set in. For all our simulations with parameters from table 7.1, this has already happened at 0.03 thermal diffusion time scales.

All simulations are performed with the same values for $\mathrm{Le}, \mathrm{R}_\rho$ and Ra_T^* and different values of the Prandtl number because Pr determines the viscous time step limit. We are interested in the time step actually taken as a function of the simulation runtime and investigate for what values of the Prandtl number the use of the IMEX method makes sense.

7.3 Two-Dimensional Experiments

The simulations in the following section have been performed on the Vienna Scientific Cluster 2, VSC2, with a resolution of 800×800 grid points on 256 cores (16×16). A summary of the performed experiments and the corresponding figures with the results is shown in table 7.1.

7.3.1 The Reference Run: $\mathrm{Pr} = 1, \mathbf{Le} = 0.1, \mathbf{R}_\rho = 0.1, \mathbf{Ra_T} = 5 \times 10^5$.

We have chosen this parameter setting as our point of reference because the limit on the time step of the viscous and diffusive terms is the same due to the Prandtl number of 1. This allows us to directly compare the efficiencies in terms of computational costs (measured by the wall clock time) and necessary steps to reach 0.03 thermal diffusion time scales.

7.3.1.1 Explicit Time Integration

We see from figure 7.1 that the time step actually taken is limited by viscosity and diffusion. They pose the same limit on the time step. What is noteworthy is that even in the advective

Pr	Le	R_ρ	Ra_T	Ra_T^*	RK scheme	result in figure
0.01	0.1	0.1	5.00×10^7	5.00×10^5	SSP32	7.9
0.01	0.1	0.1	5.00×10^7	5.00×10^5	DIFF_IMEX	7.10(a)
0.01	0.1	0.1	5.00×10^7	5.00×10^5	VISC_IMEX	7.10(b)
0.01	0.1	0.1	5.00×10^7	5.00×10^5	DIFF_IMEX, $\gamma = 0.245$	7.15(b)
0.01	0.1	0.1	5.00×10^7	5.00×10^5	VISC_IMEX, $\gamma = 0.245$	7.16(b)
0.01	0.1	0.1	5.00×10^7	5.00×10^5	DIFF_IMEX, $\gamma = 0.2475$	7.15(c)
0.01	0.1	0.1	5.00×10^7	5.00×10^5	VISC_IMEX, $\gamma = 0.2475$	7.16(c)
0.03	0.1	0.1	1.70×10^7	5.00×10^5	SSP32	7.7
0.03	0.1	0.1	1.70×10^7	5.00×10^5	DIFF_IMEX	7.8(a)
0.03	0.1	0.1	1.70×10^7	5.00×10^5	VISC_IMEX	7.8(b)
0.1	0.1	0.1	5.00×10^6	5.00×10^5	SSP32	7.5
0.1	0.1	0.1	5.00×10^6	5.00×10^5	DIFF_IMEX	7.6(a)
0.1	0.1	0.1	5.00×10^6	5.00×10^5	VISC_IMEX	7.6(b)
0.5	0.1	0.1	1.00×10^6	5.00×10^5	SSP32	7.3
0.5	0.1	0.1	1.00×10^6	5.00×10^5	DIFF_IMEX	7.4(a)
0.5	0.1	0.1	1.00×10^6	5.00×10^5	VISC_IMEX	7.4(b)
1	0.1	0.1	5.00×10^5	5.00×10^5	SSP32	7.1
1	0.1	0.1	5.00×10^5	5.00×10^5	DIFF_IMEX	7.2(a)
1	0.1	0.1	5.00×10^5	5.00×10^5	VISC_IMEX	7.2(b)
1	0.1	0.1	5.00×10^5	5.00×10^5	VISC_IMEX, $\gamma = 0.245$	7.17(b)
1	0.1	0.1	5.00×10^5	5.00×10^5	VISC_IMEX, $\gamma = 0.2475$	7.17(c)
2	0.1	0.1	2.50×10^5	5.00×10^5	SSP32	7.11
2	0.1	0.1	2.50×10^5	5.00×10^5	VISC_IMEX	7.12
3	0.1	0.1	1.67×10^5	5.01×10^5	SSP32	7.13
3	0.1	0.1	1.67×10^5	5.01×10^5	VISC_IMEX	7.14
7	0.1	0.1	7.14×10^4	5.00×10^5	SSP32	7.15(a)
7	0.1	0.1	7.14×10^4	5.00×10^5	VISC_IMEX	7.15(b)
7	0.1	0.1	7.14×10^4	5.00×10^5	VISC_IMEX, $\gamma = 0.245$	7.18(c)
7	0.1	0.1	7.14×10^4	5.00×10^5	VISC_IMEX, $\gamma = 0.2475$	7.18(c)

Table 7.1: Simulation parameters for the IMEX experiments in two dimensions. The γ factor for the IMEX methods is 0.24 unless stated otherwise. The resolution was 800×800. The simulations have been performed on 16×16 cores on the VSC2.

phase the time step is restricted by diffusion and viscosity. To reach 0.03 diffusion time scales, ANTARES needed 57144 steps and completed the simulation after 18 hours and 45 minutes. We now investigate the effect of using implicit-explicit Runge–Kutta methods for this parameter setting.

7.3.1.2 IMEX Time Integration

Using implicit integration for diffusive terms only does not bring any benefits for this parameter setting (figure 7.2(a)). This is immediately understandable because while the diffusive time step

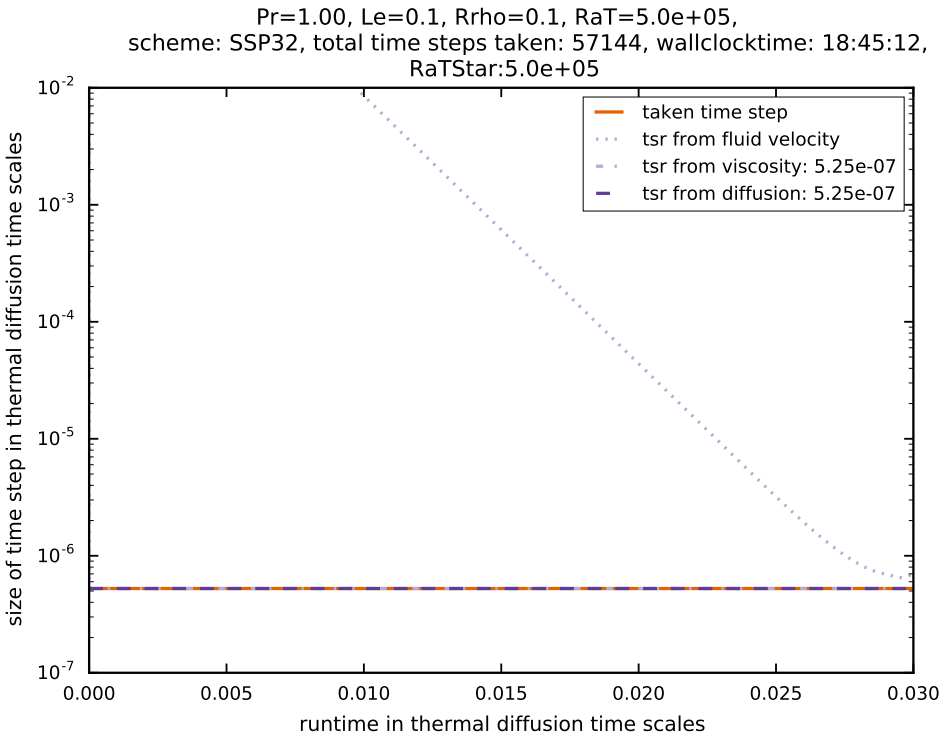
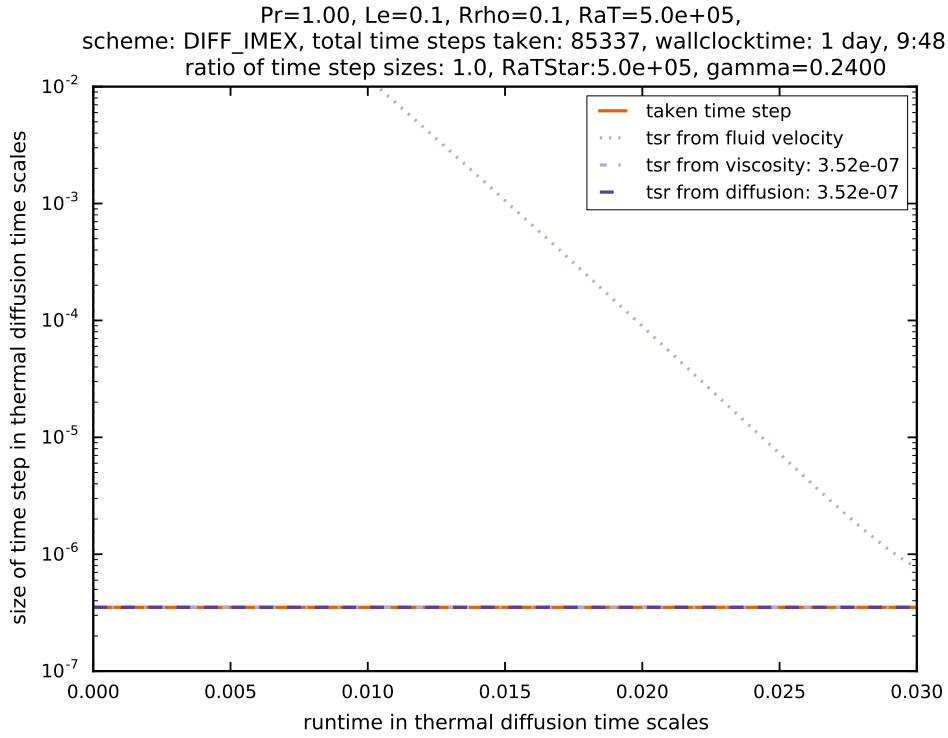


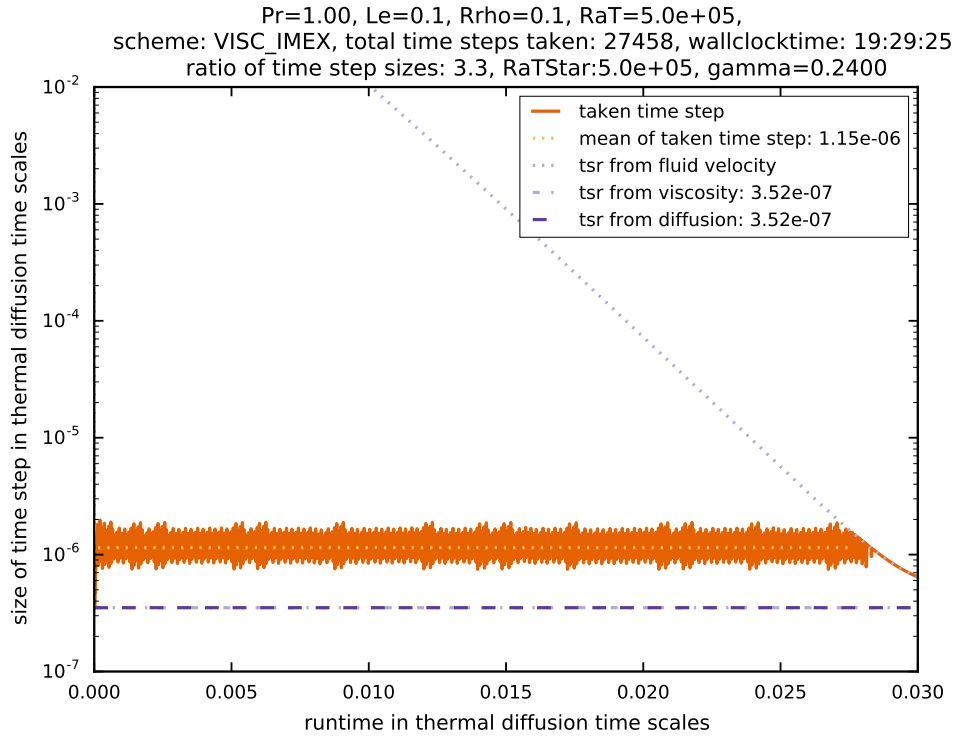
Figure 7.1: The reference run: $\mathrm{Pr} = 1, \mathrm{Le} = 0.1, \mathrm{R}_\rho = 0.1, \mathrm{Ra}_\mathrm{T} = 5 \times 10^5$. Time integration was performed with SSPRK(3,2).

limit is lifted, the viscous one is still in place and so the actually used time step cannot increase above this limit. One also sees the clear advantage of using the explicit SSPRK(3,2) scheme here: its higher diffusive Courant number permits much larger time steps than the diffusive Courant number of the IMEX SSP2(2,2,2) scheme does. This is why the simulation with DIFF_IMEX needs 85337 time steps to reach the same simulation time. This becomes apparent in the wall clock time, too, which is 33 hours and 48 minutes — 1.8 times longer than the simulation with SSPRK(3,2). This is a direct result from the number of time steps but is also due to the fact that 2 elliptic equations must be solved per stage of the Runge–Kutta method (in addition to the one implied by the Boussinesq approximation) which needs more computational time than using explicit integration.

However, using implicit integration for diffusive and viscous terms does lead to a larger time step (figure 7.2(b)). Both the diffusive and the viscous time step limits are lifted and the used time step in the diffusive phase only depends on the occurrence of two-point instabilities (see section 6.3.1). This also explains the high variance in the actually used time step. The algorithm constantly checks the number of two-point instabilities against the set threshold and adjusts the time step accordingly. Taking the mean of the used time step yields an average time step which is 3.3 times higher than the limit of either diffusive and viscous processes. However, the wall



(a) Implicit integration for diffusive terms only.



(b) Implicit integration for diffusive and viscous terms.

Figure 7.2: Evolution of time step size for $\mathrm{Pr} = 1.0, \mathrm{Le} = 0.1, \mathrm{R}_\rho = 0.1, \mathrm{Ra}_\mathrm{T} = 5 \times 10^5$. Time integration was performed with IMEX methods.

clock time is with 28 hours and 53 minutes still 1.5 times higher than the wall clock time of the simulation with the explicit SSPRK(3,2). Having to solve 4 elliptic equations per stage of the Runge–Kutta scheme (in addition to the one implied by the Boussinesq approximation) definitely takes its toll on the computational work that is required per step. Also, every time the number of two-point instabilities rises above the set threshold the time integration step must be repeated. This explains the comparably high number of 27458 time steps in spite of a time step 3.3 times larger.

7.3.2 Reducing the Prandtl Number: Pr = 0.5

Next, we look at the results of the experiments with reduced Prandtl numbers: Theoretically, reducing the Prandtl number should increase the viscous time step limit while the diffusive one stays the same. This should cause DIFF_IMEX to be of more use than in the reference run. Let us look at each parameter setting from table 7.1 where Pr = 0.5 in turn to see if the theoretically predicted outcome can be verified by experiment.

7.3.2.1 Explicit Time Integration

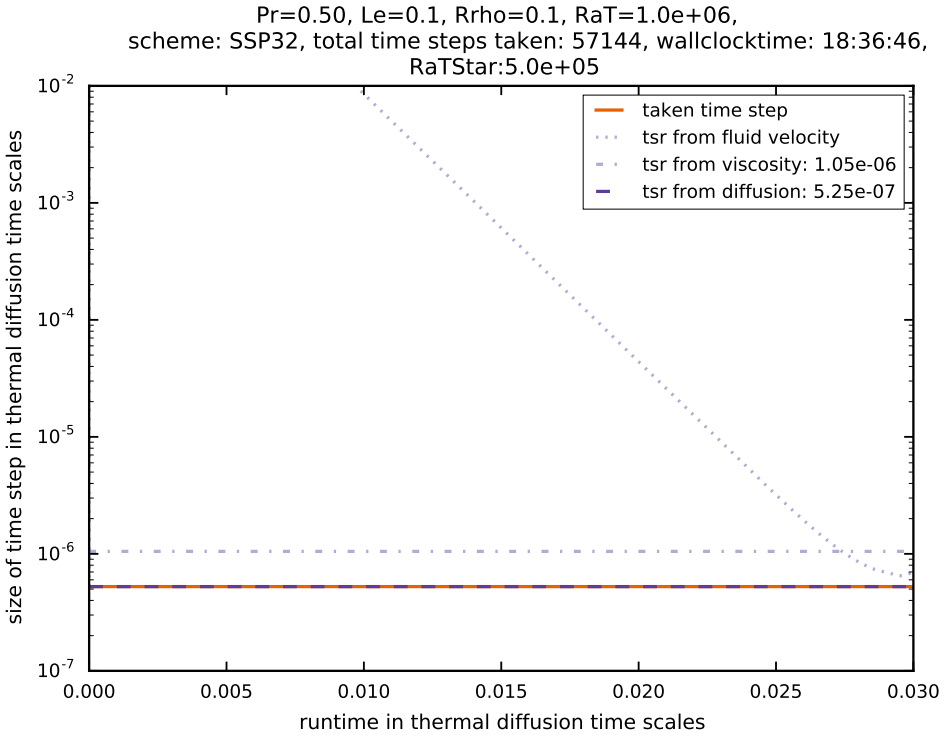


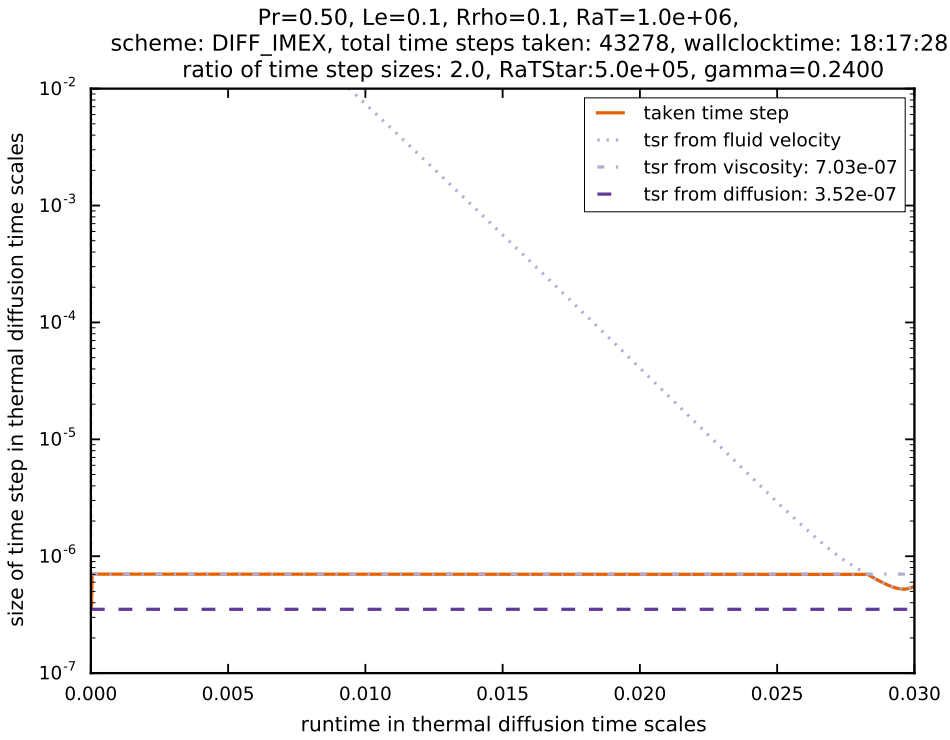
Figure 7.3: Evolution of time step size for Pr = 0.5. Time integration was performed with SSPRK(3,2).

We see from figure 7.3 that the time step actually taken is limited by diffusion in this parameter setting. Similar to the reference setting, even in the advective phase, the time step is restricted by diffusion and not by the fluid velocity. To reach 0.03 diffusion time scales, ANTARES needed 57144 steps which is exactly the same number of time steps as was needed in the reference run. This also demonstrates that Ra_T^* (which is $\mathrm{Ra}_\mathrm{T} \times \mathrm{Pr}$) is the dimensionless number which determines the onset of convection, not Ra_T alone. The simulation was completed after 18 hours and 36 minutes. This is 9 minutes faster than the reference run but this could also be due to the workload on VSC2 or the specific nodes that have been used for the calculation. Reducing the Prandtl number does not seem to make a difference for the number of time steps when using explicit methods as long as the diffusion time scale is the limiting one. We now investigate the effect of using implicit-explicit Runge–Kutta methods for this parameter setting.

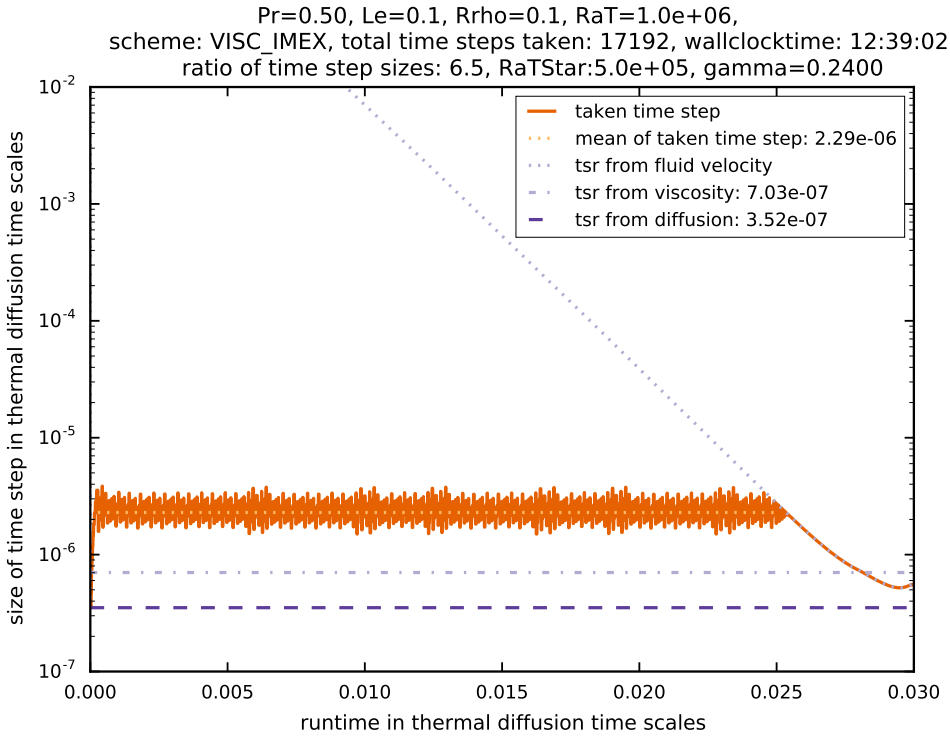
7.3.2.2 IMEX Time Integration

The simulation shown in figure 7.4(a) clearly shows the advantage of using implicit integration for the diffusive term. The actually used time step is increased from the diffusion time step limit to the viscous time step limit when using DIFF_IMEX. As soon as the fluid becomes fast enough the fluid velocity determines the maximally applicable time step and implicit integration has no further benefits. The number of time steps needed for DIFF_IMEX is 43278, the wall clock time is 18 hours and 17 minutes which is 98% of the wall clock time with SSPRK(3,2). This is a very small acceleration, bearing in mind that the actually taken time step is twice the diffusion limit. But again, the superior diffusive Courant number of SSPRK(3,2) makes the DIFF_IMEX simulation comparable in terms of wall clock times.

Using implicit integration for viscous and diffusive terms gives us a clear advantage for this parameter setting as we see in figure 7.4(b). Not only is the time step actually taken larger than the diffusive step restriction, but also higher than the viscous one. This leads to a time step 6.5 times the size of the one taken for explicit integration. In the end, we only needed 17192 time steps and managed to do so in only 12 hours and 39 minutes which is 68% of the time needed with explicit integration with SSPRK(3,2).



(a) Implicit integration for diffusive terms only.



(b) Implicit integration for diffusive and viscous terms.

Figure 7.4: Evolution of time step size for $\mathrm{Pr} = 0.5$. Time integration was performed with IMEX methods.

7.3.3 Reducing the Prandtl Number: Pr = 0.1

7.3.3.1 Explicit Time Integration

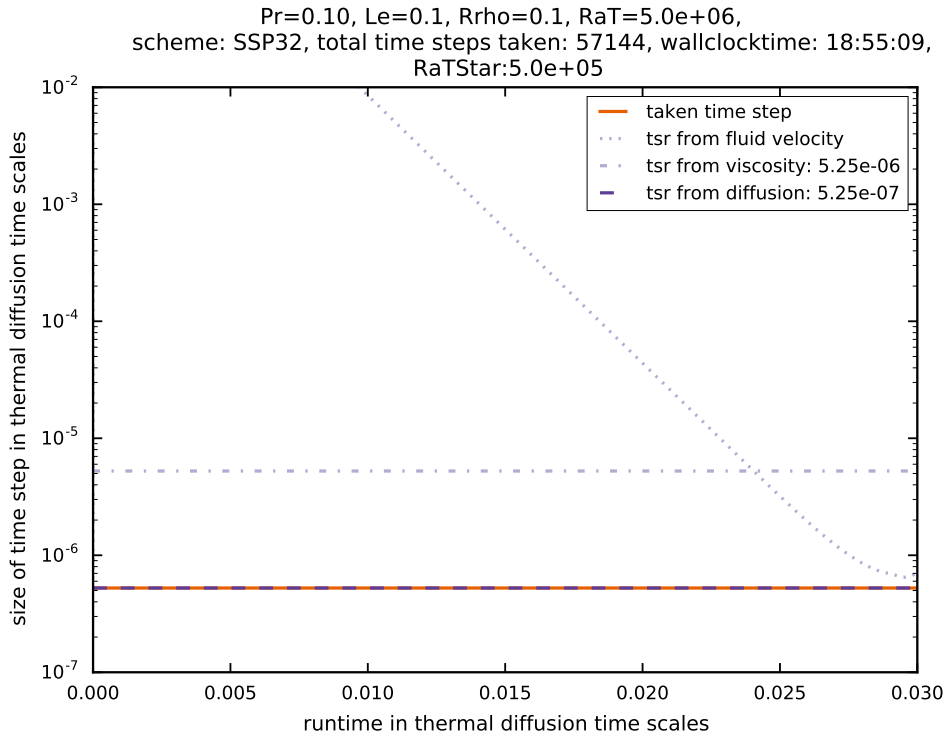


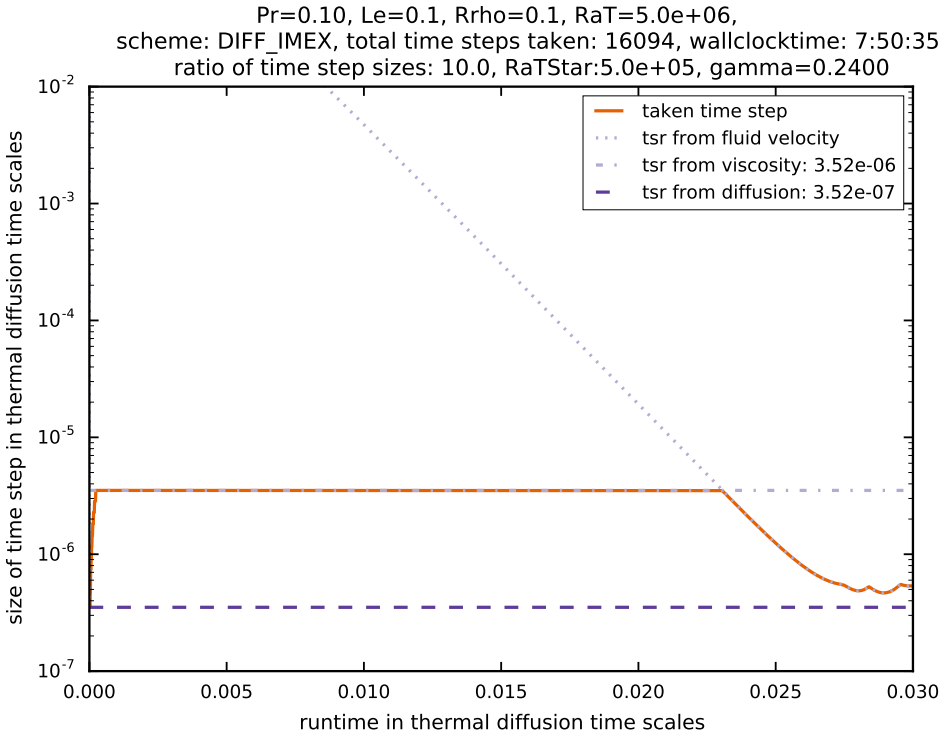
Figure 7.5: Time step evolution for Pr = 0.1. Time integration was performed with SSPRK(3,2).

We see from figure 7.5 that the time step actually taken is limited by diffusion in this parameter setting. Similar to the reference setting and the simulation with Pr = 0.5 the time step continues to be restricted by diffusion in the advective part of the simulation. To reach 0.03 diffusion time scales, ANTARES needed again 57144 steps which is exactly the same number of time steps as was needed in the reference run and for Pr = 0.5. The simulation was completed after 18 hours and 55 minutes. This is again in the same range as the simulations with Pr = 1 and Pr = 0.5. This makes sense because by adjusting the Prandtl number, only the viscous time limit is influenced. This poses no limit for these simulations, however, because they are diffusively limited.

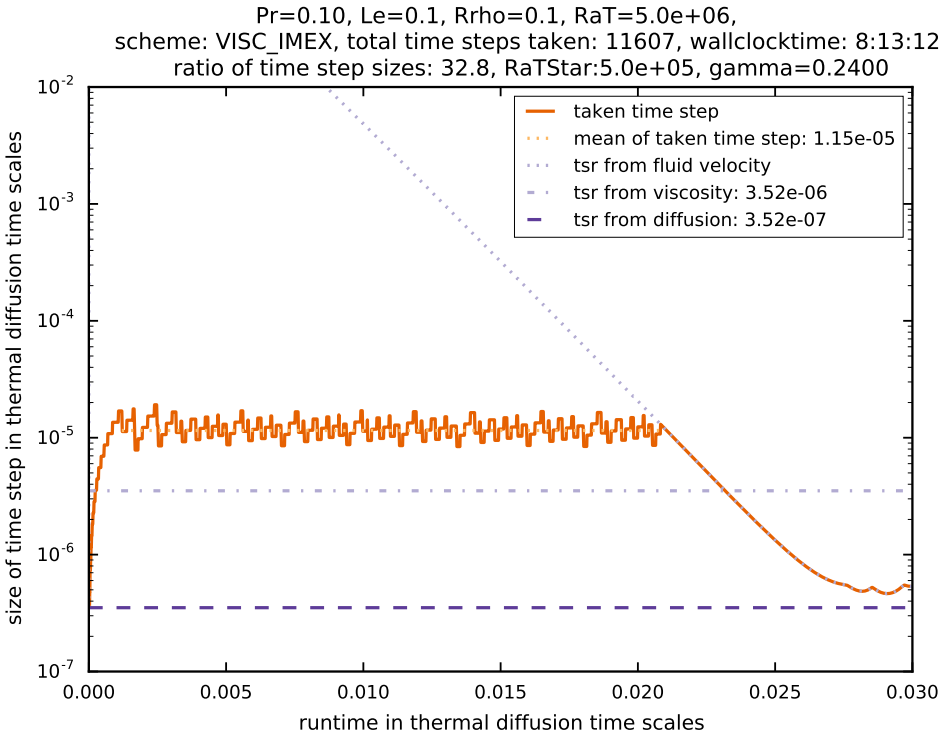
Next, we investigate the effect of using implicit-explicit Runge–Kutta methods for this parameter setting.

7.3.3.2 IMEX Time Integration

The simulation shown in figure 7.6(a) clearly shows the advantage of using implicit integration for the diffusive term. The actually used time step is increased from the diffusion limit to the



(a) Implicit integration for diffusive terms only.



(b) Implicit integration for diffusive and viscous terms.

Figure 7.6: Evolution of time step size for $\mathrm{Pr} = 0.1$. Time integration was performed with IMEX methods.

viscous limit when using DIFF_IMEX. As soon as the fluid becomes fast enough the fluid velocity determines the maximally applicable time step and implicit integration has no further benefits. The number of time steps needed for DIFF_IMEX is 16094, the wall clock time is 7 hours and 50 minutes which is an acceleration of 2.4 compared with SSPRK(3,2). This is a very good acceleration. The time step actually taken is limited by the viscous time scale which is 10 times as large as the diffusive one. That means that for this parameter setting, the IMEX method does not reach its limit yet because there are no two-point instabilities visible which means that the time step could be increased further.

Using implicit integration for viscous and diffusive terms further magnifies the used time step to be on average 39 times that of the originally most restrictive time scale, i.e., the diffusive one (7.6(b)). In the end, we only needed 11609 time steps and managed to do so in 8 hours and 13 minutes. This is 43% of the time needed with explicit integration with SSPRK(3,2). But this is a little slower than using implicit integration for diffusive terms only. This is likely due to the increased workload when using implicit integration for 4 equations per stage instead of 2 equations per stage (in addition to the one implied by the Boussinesq approximation).

Since we have not yet reached the limit for the IMEX method, we further decrease the Prandtl number to Pr = 0.01.

7.3.4 Reducing the Prandtl Number: Pr = 0.03

Here, we perform a simulation with a reduction of the Prandtl number to Pr = 0.03.

7.3.4.1 Explicit Time Integration

For explicit integration, not much changes from the prior simulations: we see from figure 7.7 that the time step actually taken is limited by diffusion again because it is the most restrictive limit on the time step. Similar to the settings before, the time step continues to be restricted by diffusion in the advective part of the simulation. To reach 0.03 diffusion time scales, ANTARES needed again 57144 steps which is exactly the same number of time steps that was needed in runs before. The simulation was completed after 18 hours and 41 minutes. This is again in the same range as the simulations with Pr = 1, Pr = 0.5 and Pr = 0.1. This makes sense because by adjusting the Prandtl number, only the viscous time limit is influenced. This poses no limit for these simulations, however, because they are diffusively limited. So for explicit integration, it makes no difference if the Prandtl number is set to 1 or 0.03. The time step is restricted by diffusion in both cases.

Next, we investigate the effect of using implicit-explicit Runge–Kutta methods for this parameter setting.

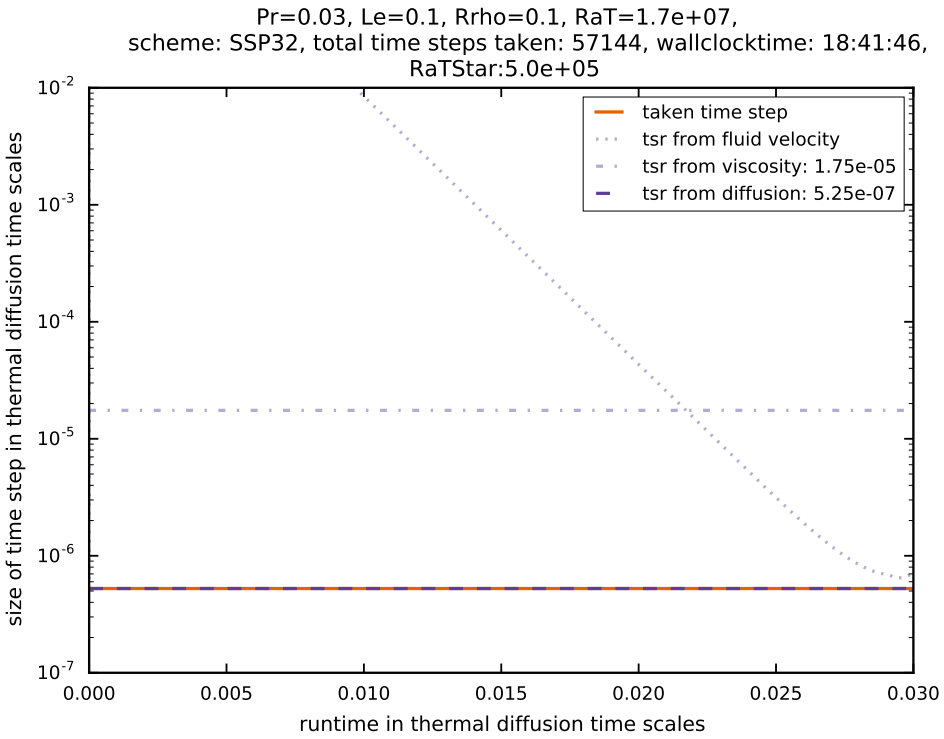
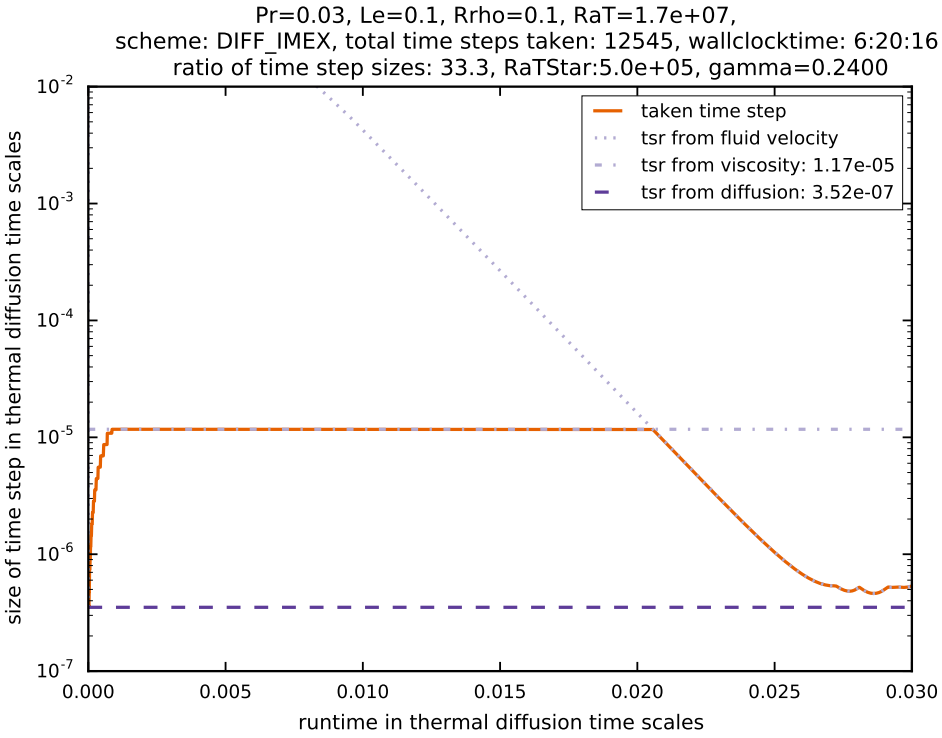


Figure 7.7: Time step evolution for Pr = 0.03. Time integration was performed with SSPRK(3,2).

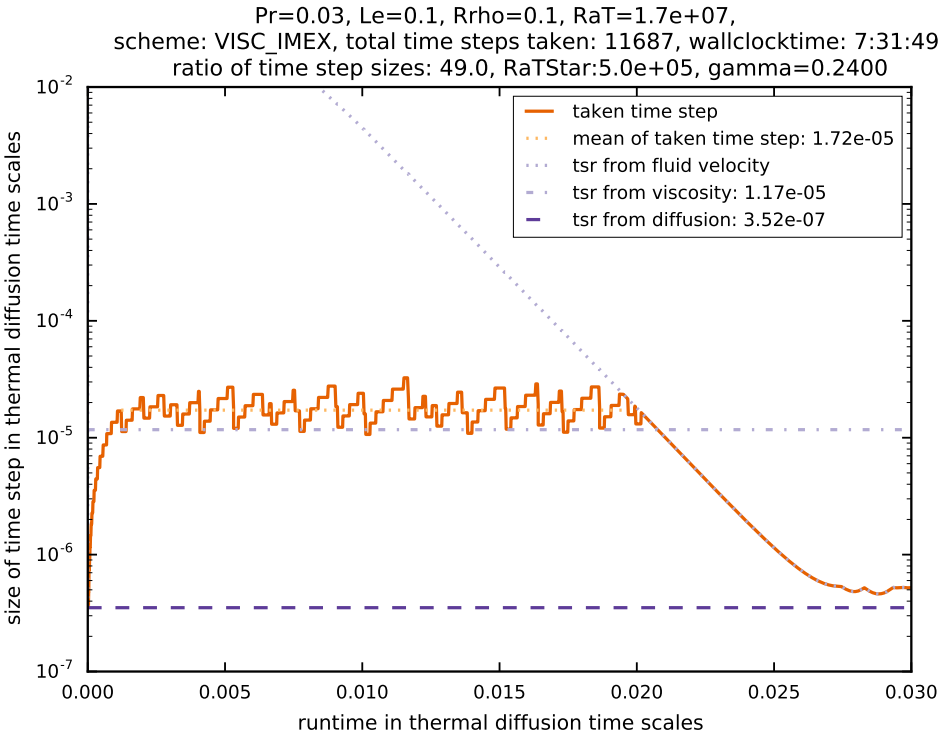
7.3.4.2 IMEX Time Integration

The simulation shown in figure 7.8(a) clearly shows the advantage of using implicit integration for the diffusive term. The actually used time step is increased from the diffusion limit to the viscous limit when using DIFF_IMEX. As soon as the fluid becomes fast enough the fluid velocity determines the maximally applicable time step and implicit integration has no further benefits. The number of time steps needed for DIFF_IMEX is 12545, the wall clock time is 6 hours and 20 minutes which is an acceleration of 2.95 compared to SSPRK(3,2). This is a very good acceleration. The time step actually taken is limited by the viscous time scale which is 33.2 times as large as the diffusive one. That means that for this parameter setting, the IMEX method does still not reach its stability limit because there are still no two-point instabilities visible which means that the time step could be increased further.

Using implicit integration for viscous and diffusive terms further magnifies the used time step to be on average 39 times that of the originally most restrictive time scale, i.e., the diffusive one (7.6(b)). In the end, we only needed 11687 time steps and managed to do so in 7 hours and 31 minutes. This is an acceleration of 2.8 compared to explicit integration with SSPRK(3,2). But this is a little slower than using implicit integration for diffusive terms only. This is likely due to the increased workload when using implicit integration for 4 equations per stage instead of 2 equations per stage (in addition to the one implied by the Boussinesq approximation).



(a) Implicit integration for diffusive terms only.



(b) Implicit integration for diffusive and viscous terms.

Figure 7.8: Evolution of time step size for $\mathrm{Pr} = 0.03$. Time integration was performed with IMEX methods.

Since we have not yet reached the limit for the IMEX method, we further decrease the Prandtl number to Pr = 0.01.

7.3.5 Reducing the Prandtl Number: Pr = 0.01

We performed one more simulation with a reduction of the Prandtl number to Pr = 0.01.

7.3.5.1 Explicit Time Integration

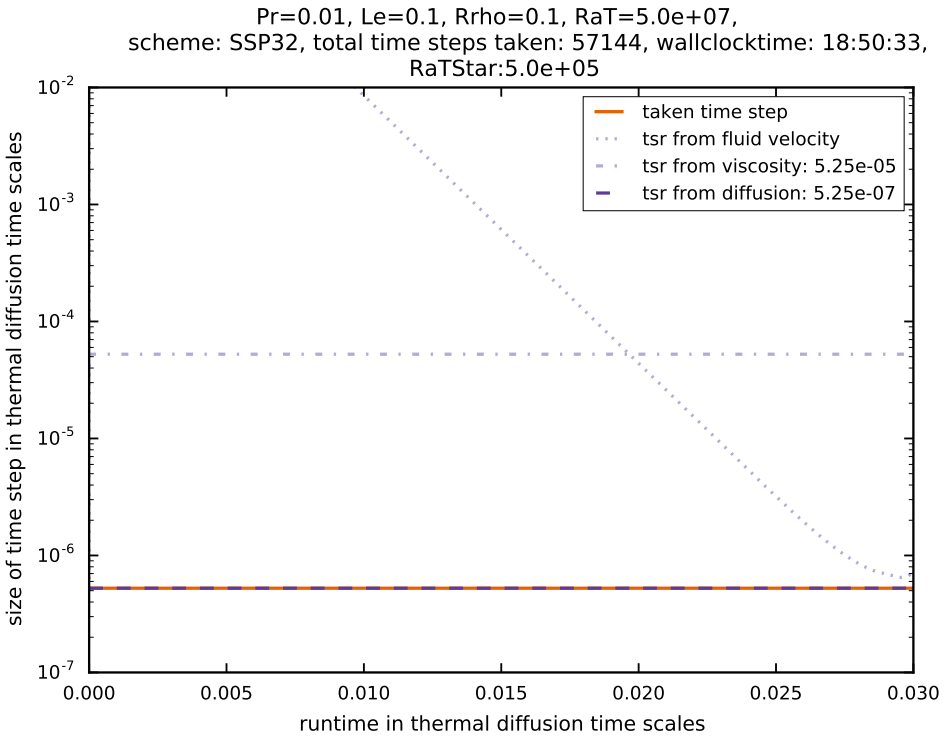


Figure 7.9: Time step evolution for Pr = 0.01. Time integration was performed with SSPRK(3,2).

For explicit integration, not much changes from the two prior simulations: we see from figure 7.9 that the time step actually taken is limited by diffusion again because it is the most restrictive limit on the time step. Similar to the settings before, the time step continues to be restricted by diffusion in the advective part of the simulation. To reach 0.03 diffusion time scales, ANTARES needed again 57144 steps which is exactly the same number of time steps that was needed in runs before. The simulation was completed after 18 hours and 50 minutes. This is again in the same range as the simulations with Pr = 1, Pr = 0.5 and Pr = 0.1. This makes sense because by adjusting the Prandtl number, only the viscous time limit is influenced. This poses no limit for these simulations, however, because they are diffusively limited. So for explicit integration,

it makes no difference if the Prandtl number is set to 1 or 0.01. The time step is restricted by diffusion in both cases.

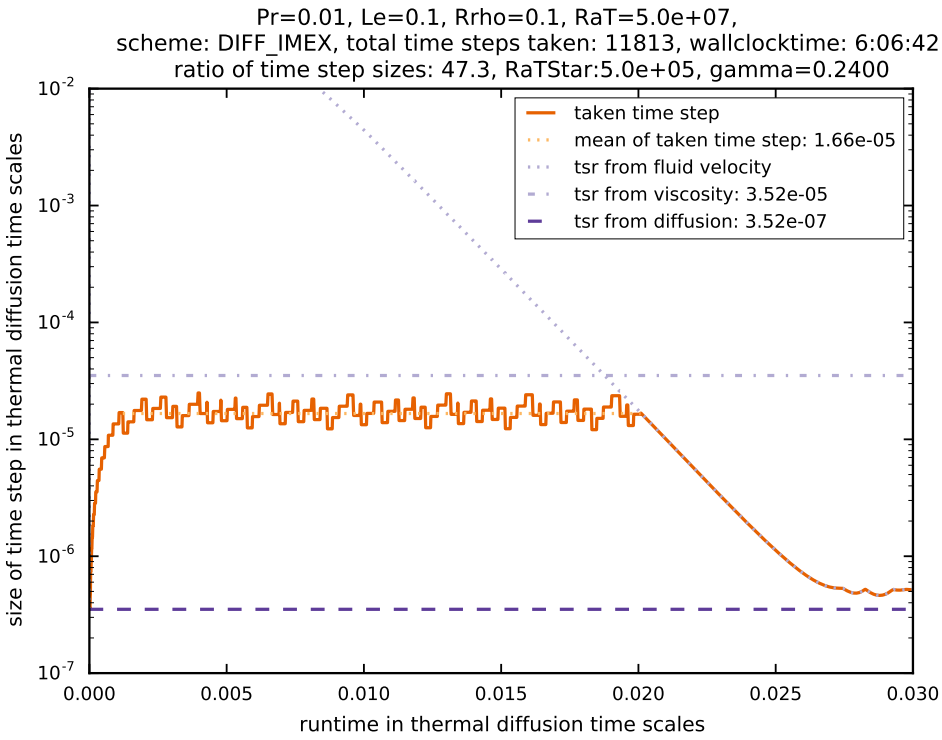
Next, we investigate the effect of using implicit-explicit Runge–Kutta methods for this parameter setting.

7.3.5.2 IMEX Time Integration

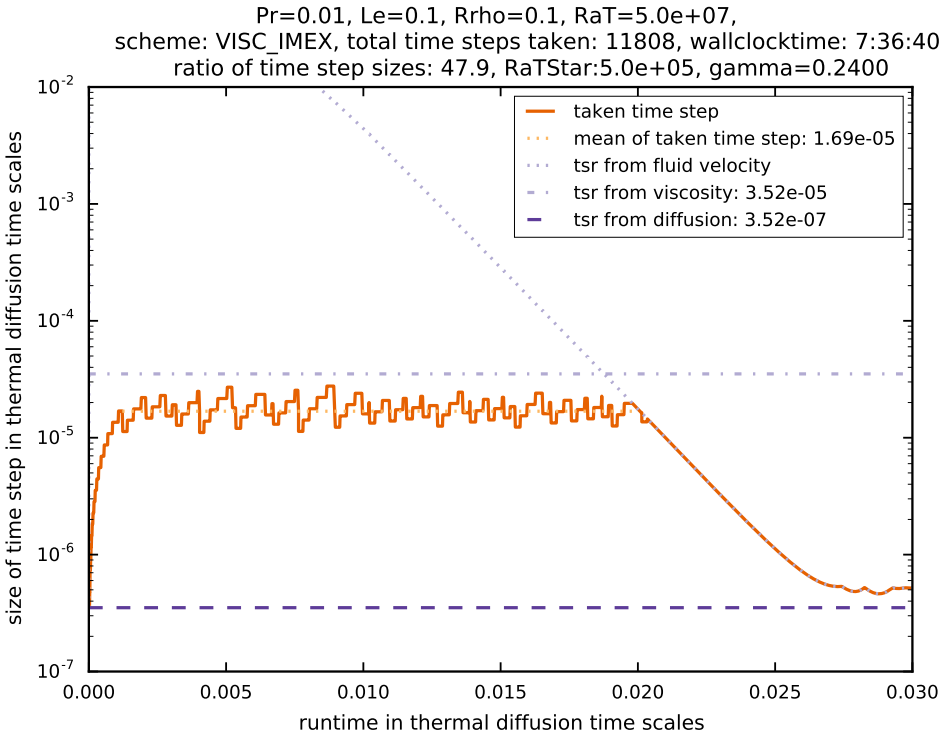
The simulation shown in figure 7.10(a) again clearly shows the advantage of using implicit integration for the diffusive term. The actually used time step is increased from the diffusion limit. However, for Pr = 0.01 DIFF_IMEX reaches its stability limit as is apparent by the activation of the time step control mechanism. This is the first parameter setting where this occurs and it does so for a time step size 38.7 times larger than the diffusive limit. The number of time steps needed for DIFF_IMEX is 11813, the wall clock time is 6 hours and 6 minutes which is 32% of the wall clock time with SSPRK(3,2). This is an extremely good acceleration: integrating the diffusive terms implicitly leads to an acceleration of about 3. The time step actually taken is not limited by the viscous time scale in this scenario because the Prandtl number is so low that the viscous time step limit is too high to have an impact. Because we have reached the stability limit of IMEX SSP2$(2,2,2),\gamma = 0.24$ with this parameter setting, it would be of interest to see if an increase of γ leads to a more stable IMEX scheme. This will be investigated in section 7.4.

Using VISC_IMEX has no significant effect on the size of the possible time step. This is immediately clear because the viscous time step limit is never reached in this parameter setting so the implicit integration of viscous terms makes no sense here. That is why there is no further acceleration from DIFF_IMEX visible here. The actual time step is still on average 39 times that of the originally most restrictive time scale (which is the diffusive one, see figure 7.10(b)). In the end, we needed 11808 time steps and managed to do so in 7 hours and 36 minutes. While this is 40% of the time needed with explicit integration with SSPRK(3,2) it is longer than the wall clock time for implicit integration of the diffusive terms only. This is again likely due to the increased workload when using implicit integration for 4 equations per stage instead of 2 equations per stage (in addition to the one implied by the Boussinesq approximation).

Since we have reached the stability limit for the IMEX SSP2(2,2,2) method, a further reduction of the Prandtl number does not yield more insight. Instead, we turn to the simulations where we have increased the Prandtl number.



(a) Implicit integration for diffusive terms only.



(b) Implicit integration for diffusive and viscous terms.

Figure 7.10: Evolution of time step size for Pr = 0.01. Time integration was performed with IMEX methods.

7.3.6 Increasing the Prandtl Number: Pr = 2

Increasing the Prandtl number leads to a more restrictive viscous time step limit. We expect to see a clearer advantage when using VISC_IMEX. Using implicit integration for the diffusive terms only does not make any sense for values of the Prandtl number greater than one because the time step is restricted by viscosity in this regime. This means that implicit integration of just the diffusive terms is of no use. That is why we perform the time integration in this section with 2 methods only: SSPRK(3,2) and VISC_IMEX.

7.3.6.1 Explicit Time Integration

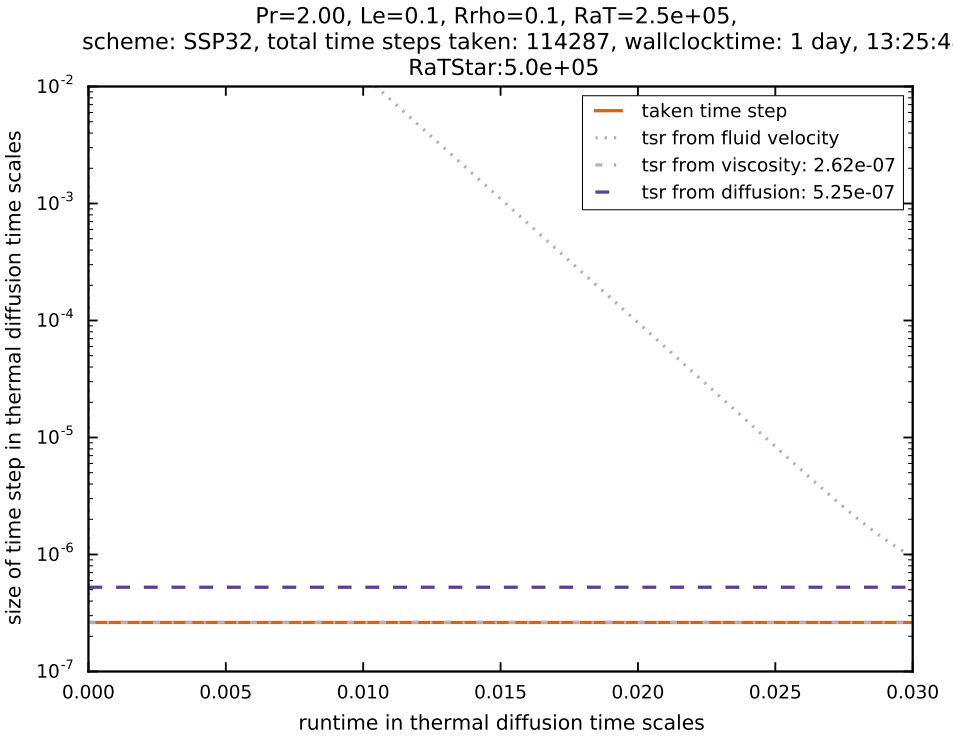


Figure 7.11: Time step evolution for Pr = 2. Time integration was performed with SSPRK(3,2).

The results for explicit integration can be found in figure 7.11 . The time step actually taken is limited by viscosity because it is the most restrictive limit on the time step. A difference to the simulations with Pr ≤ 1 is that we do not quite reach the advective phase in this setting. This might be due to Ra_T reaching lower and lower values for increasing Pr in order for Ra_T^* to be constant. But the qualitative statements that we are about to make about this parameter setting remain accurate.

To reach 0.03 diffusion time scales, ANTARES needed 113287 steps. This is almost twice of what it needed in the simulations with lower Prandtl number. This is because the viscous

time step limit decreases with increasing Pr. Since the viscous limit is the one controlling the actually used time step, this leads to a smaller overall time step and as a result also to a longer simulation: the wall clock time was 37 hours and 25 minutes.

Next, we investigate the effect of using implicit-explicit Runge–Kutta methods for this parameter setting.

7.3.6.2 IMEX Time Integration

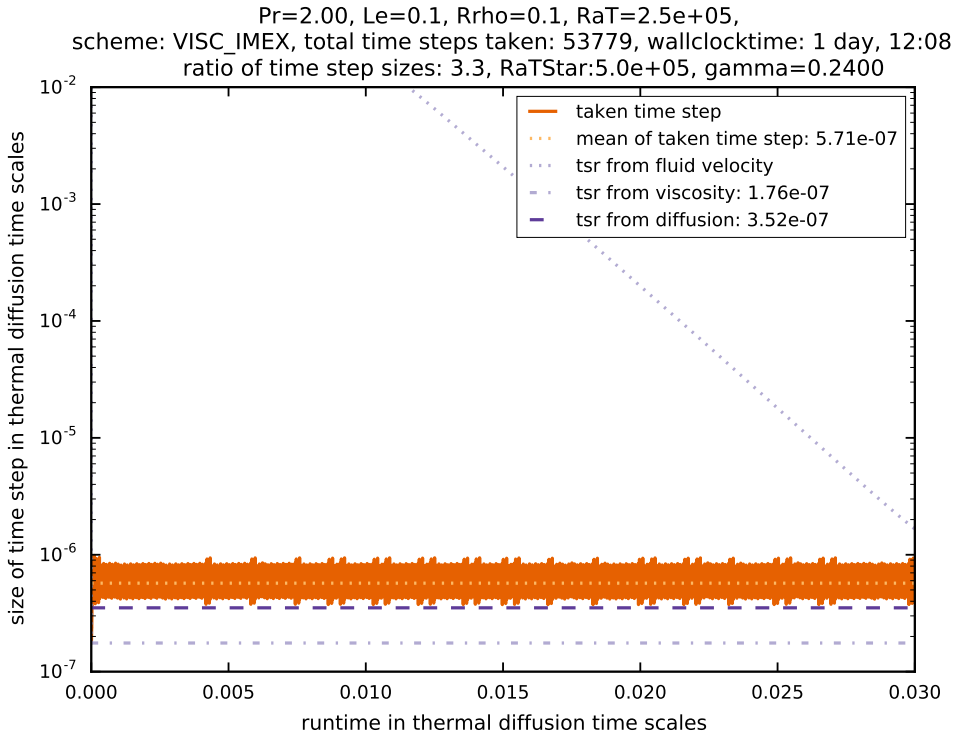


Figure 7.12: Evolution of time step size for Pr = 2. Time integration was performed with implicit integration of diffusive and viscous terms.

While the simulation shown in figure 7.12 shows a slight advantage of using implicit integration for diffusive and viscous terms, the increase in the time step is much smaller than was hoped for. The number of time steps needed with IMEX was 53779 steps, which is only 47% of time steps that were needed for the explicit integration. But the wall clock time was with 36 hours and 8 minutes almost the same as with the explicit integration. The likely reason for this can be seen from figure 7.12: the time step control mechanism had much more work to do than in the simulations with lower Prandtl number. This means that there have been many occurrences of two-point instabilities and the time step had to be repeated a significant number of times. This is a lot of wasted computational effort and explains why the accepted time steps needed such a long computational time.

We now further increase the Prandtl number to investigate if this behavior persists.

7.3.7 Increasing the Prandtl Number: Pr = 3

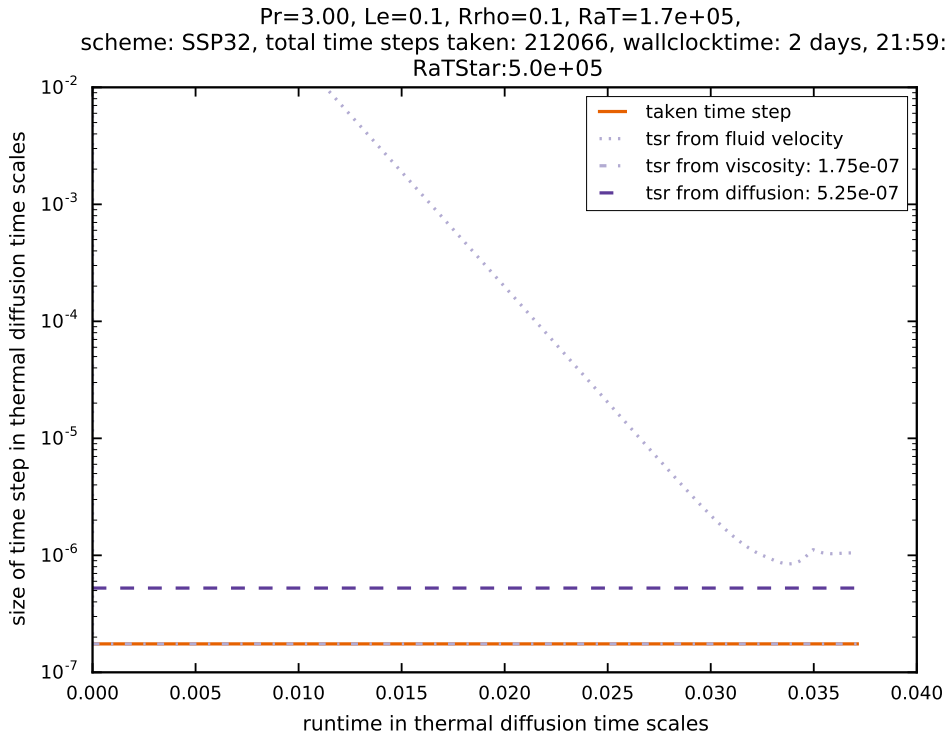


Figure 7.13: Time step evolution for Pr = 3. Time integration was performed with SSPRK(3,2). It was aborted after 70 hours of runtime. At that point it had reached 0.0371 thermal diffusion time scales.

The results can be found in figures 7.13 and 7.14 . This is the first simulation setting where the threshold of 70 hours which we have set for the wall clock time was reached and thus the simulation was aborted prematurely. Here, primarily the overall simulation time in terms of thermal diffusion times lets us decide on which time integration scheme performed better. The reason why the simulations take longer for increased Prandtl numbers is that we are *decreasing* the maximally allowed time step (6.12), i.e.,

$$\Delta t_{\text{visc}} = c_{\text{diff}} \frac{\min(\Delta x^2, \Delta y^2, \Delta z^2)}{\text{Pr}},$$

which leads to smaller time step limits, the larger the Prandtl number is chosen.

As a matter of fact, both the viscous as well as the diffusive time step limits are far below the advective time step limit that is reached when convection sets in. Because the simulation was stopped after a runtime of 70 hours, we cannot compare the wall clock times, the total number of steps and so on as we did for the simulations above. Instead, we focus on the time in thermal

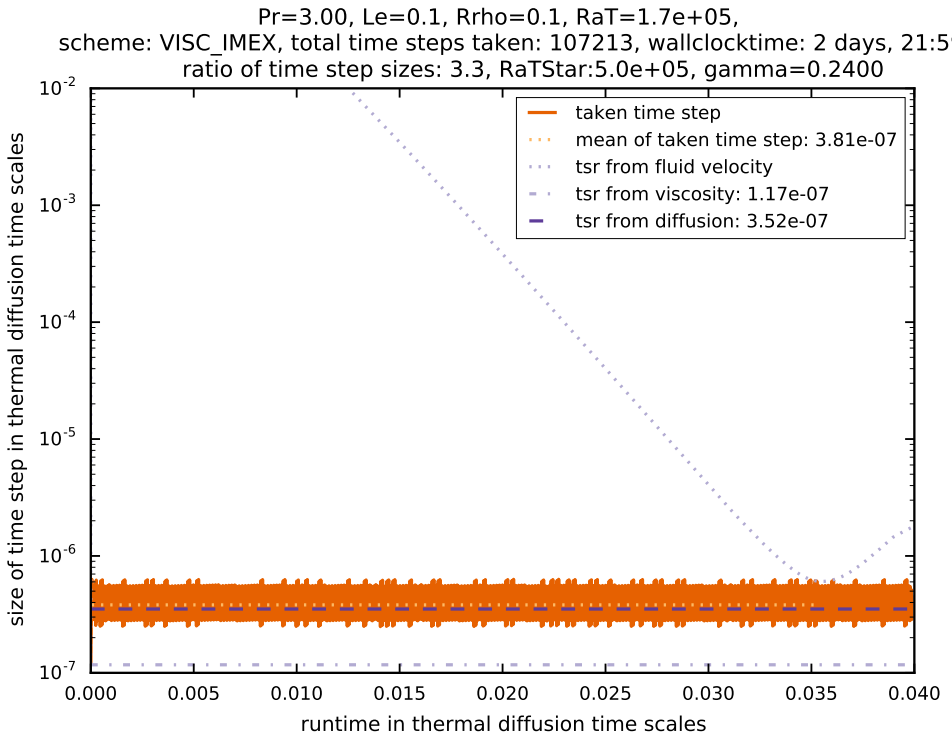


Figure 7.14: Evolution of time step size for Pr = 3. Time integration was performed with implicit integration of diffusive and viscous terms. It was aborted after 70 hours of runtime. At that point it had reached 0.0399 thermal diffusion time scales.

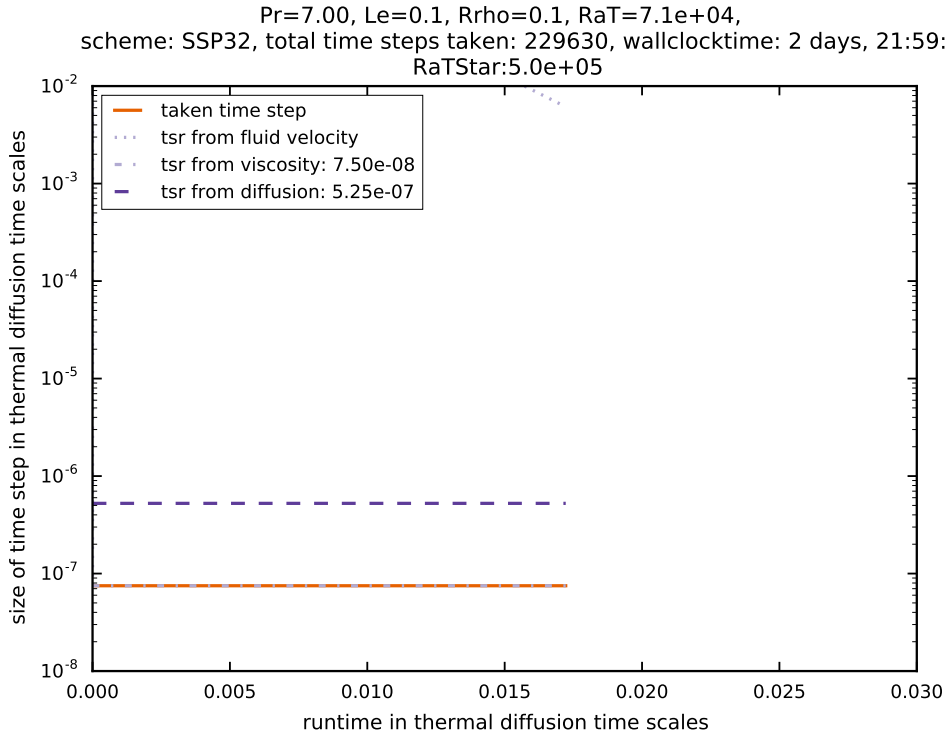
diffusion time scales that has been reached during the wall clock time of 70 hours: 0.0371 for the explicit SSPRK(3,2) scheme and 0.0399 for IMEX SSP2(2,2,2). This means that using IMEX does have an advantage in simulations with high Prandtl number. It was able to achieve 1.07 times more thermal diffusion time scales in 70 hours than the explicit scheme.

The mean of the actually used time step is again 3.3 times higher than the viscous time step limit which solidifies our assumption that this limit is set by the specific choice of the IMEX method.

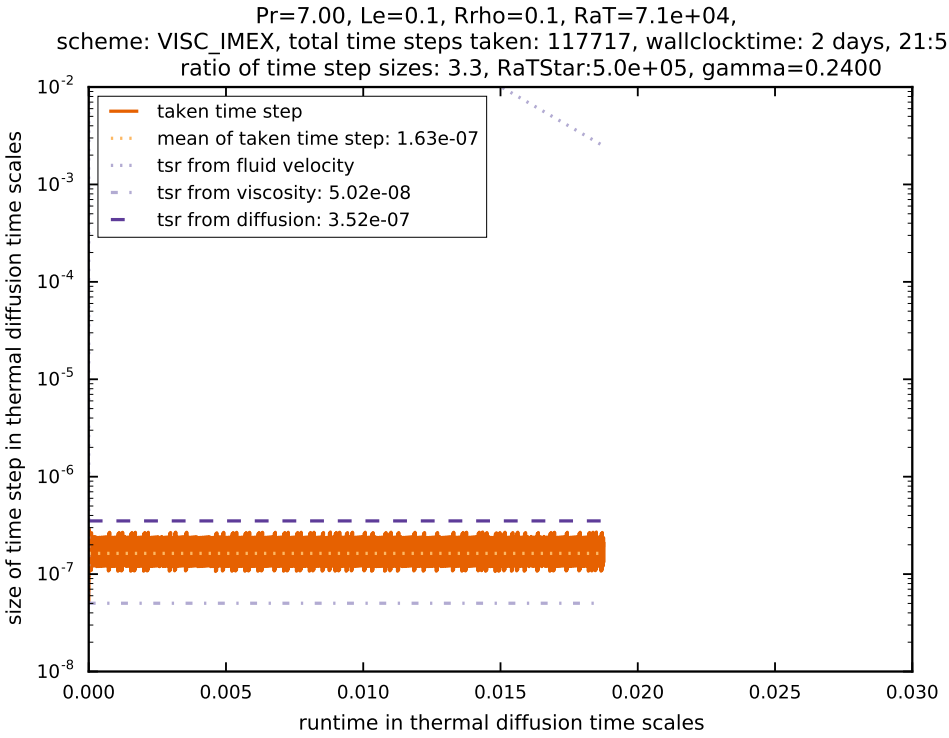
We perform one more simulation with Pr = 7, the approximate Prandtl number of water.

7.3.8 Increasing the Prandtl Number: Pr = 7

This is the last simulation in our set of highly-resolved simulations. The results can be found in figures 7.15(a) and 7.15(b). As could be expected from the simulation with Pr = 3 the wall clock time threshold of 70 hours leads to an abortion of these simulations as well because of the further reduced time step limit. Again, we compare the simulations by comparing the simulated time in thermal diffusion time scales: explicit integration reached 0.0172 thermal diffusion time scales and VISC_IMEX reached 0.0188 thermal diffusion time scales. This time, integration with IMEX



(a) Time integration performed with SSPRK(3,2).



(b) Time integration was performed with implicit integration of diffusive and viscous terms.

Figure 7.15: Evolution of time step size for Pr = 7. The simulations have been aborted after 70 hours of runtime.

was able to simulate a 1.09 times larger interval in time simulated time than explicit integration.

The mean of the actually used time step is again 3.3 times higher than the viscous time step limit which further solidifies our assumption that this limit is set by the specific choice of the IMEX method.

Pr	method	ratio of time steps	wall clock time
0.01	DIFF_IMEX, $\gamma = 0.24$	47.3	06:06:42
0.01	DIFF_IMEX, $\gamma = 0.245$	95.9	05:55:23
0.01	DIFF_IMEX, $\gamma = 0.2475$	100	05:57:47
0.01	VISC_IMEX, $\gamma = 0.24$	47.9	07:36:40
0.01	VISC_IMEX, $\gamma = 0.245$	106.6	07:18:25
0.01	VISC_IMEX, $\gamma = 0.2475$	199.8	07:20:39

Table 7.2: Results from the simulations with increased γ for Pr = 0.01.

7.4 Increasing the γ Factor of the SSP IMEX Scheme

It has been mentioned in section 6.2.2 that the value of γ has been chosen because F. Kupka et al., (2012) found it to be the most efficient time integrator for their applications. In our group, for simulations of semiconvection, $\gamma = 0.24$ yielded the best integrator. The situation is different when looking at simulations of Cepheids. Here, a larger value of γ (and with that a larger stability region) was more suitable (see Happenhofer, 2014).

To determine if increasing γ is of any use in our simulations, we repeat the simulations for Prandtl numbers 0.01, 1 and 7 for higher values of γ, namely $\gamma = 0.245$ and $\gamma = 0.2475$ and investigate if the time steps obtained by the prior experiments can be enhanced by the use of a higher γ.

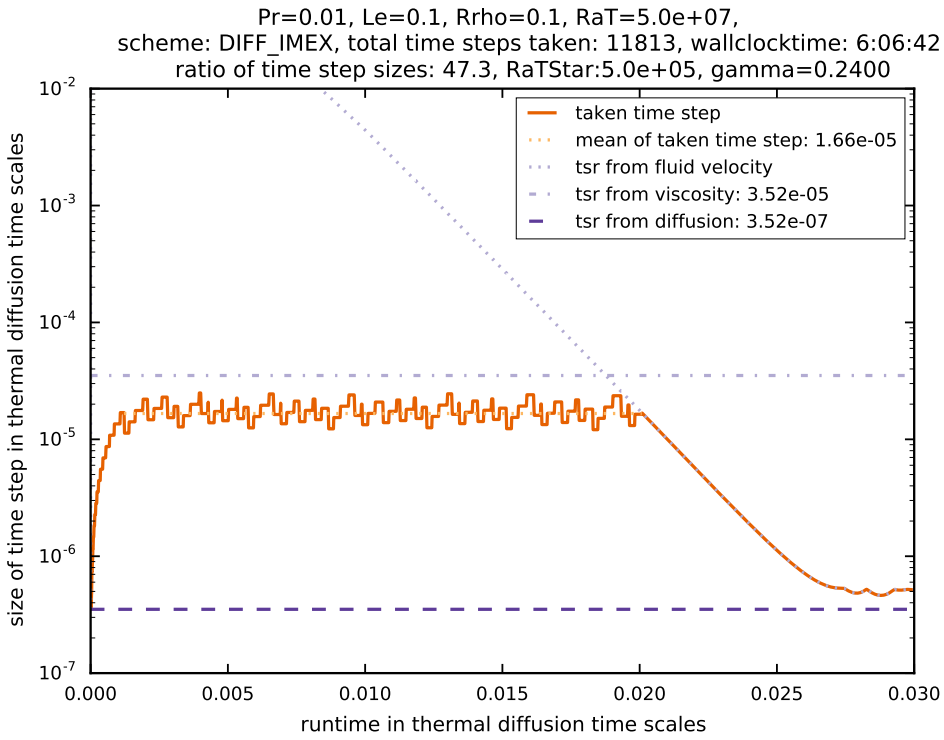
7.4.1 Increasing γ for Pr = 0.01

We repeat the simulations from section 7.3.5 with $\gamma = 0.245$ and $\gamma = 0.2475$. The results for DIFF_IMEX are shown in figure 7.16, the results for VISC_IMEX are shown in figure 7.17 and table 7.2.

7.4.1.1 Effect on the Size of the Time Step Actually Taken

We see that an increase of γ leads to a significant increase of the actual time step. While for $\gamma = 0.24$ DIFF_IMEX reached its stability limit (fig 7.16(a)) with a ratio of the average taken time step over the time step limit from diffusion of 47.3, for $\gamma = 0.245$ the ratio increased to 95.9 which is almost as good as the viscous limit allows for this parameter setting. A further increase of γ to 0.2475 yields an actual time step which is limited by viscosity and no two-point instabilities occur at all.

For VISC_IMEX the effect is even more pronounced because there is no upper limit from the viscosity time scale. For $\gamma = 0.24$ VISC_IMEX was limited by the stability of DIFF_IMEX (fig 7.17(a)). Because the stability region of DIFF_IMEX increases when increasing γ, we see a further increase of the time step when using VISC_IMEX with $\gamma = 0.245$ (fig 7.16(c)). The time step with IMEX methods is 106.6 times as large as the time step with explicit methods.

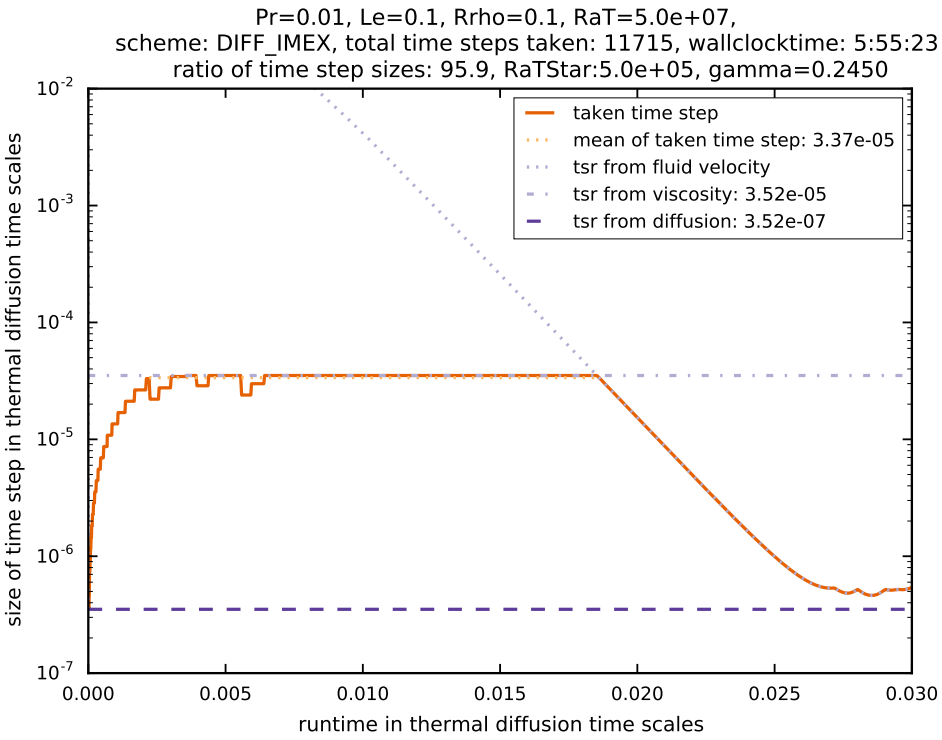


(a) $\gamma = 0.24$

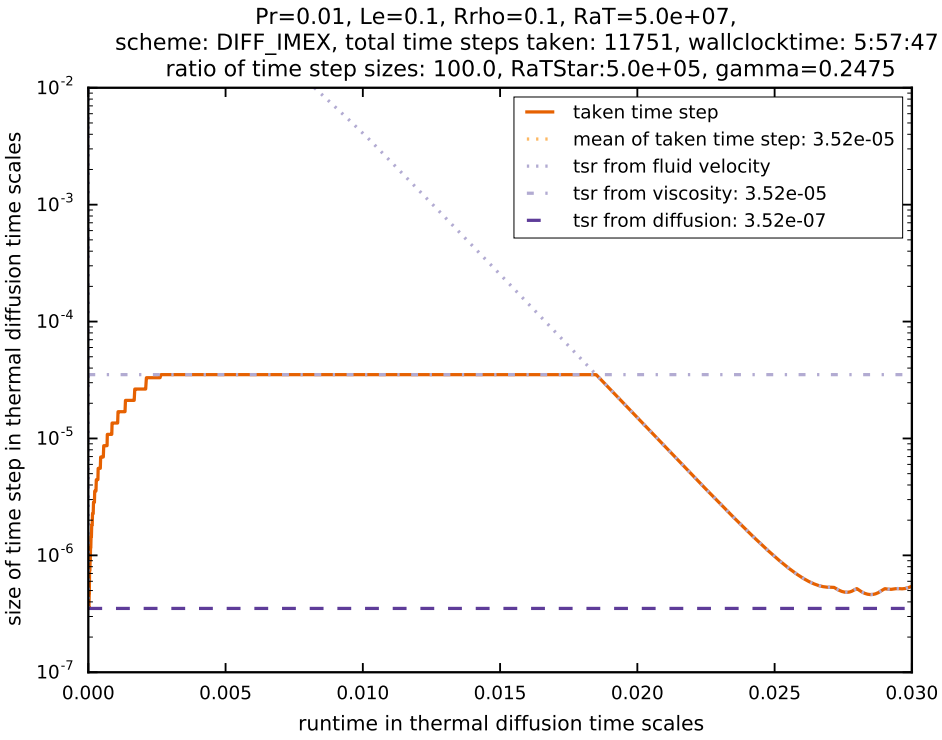
A further increase of γ to the value 0.2475 (fig 7.17) even leads to a time step which is 199.8 times as large as the time step employed with the explicit method. It can be assumed that the time step would increase even further if the fluid time scale would not inhibit any further increase.

7.4.1.2 Effect on the Wall Clock Time

The effects that increasing γ has on the wall clock time are more ambivalent. For both DIFF_IMEX and VISC_IMEX there is a slight decrease of the wall clock time when using $\gamma = 0.245$ but then a tiny increase in wall clock time when increasing γ further to 0.2475. The reason for why increasing γ (and thus the time step actually taken) leads to surprisingly small changes in wall clock time is discussed in section 7.6.3.

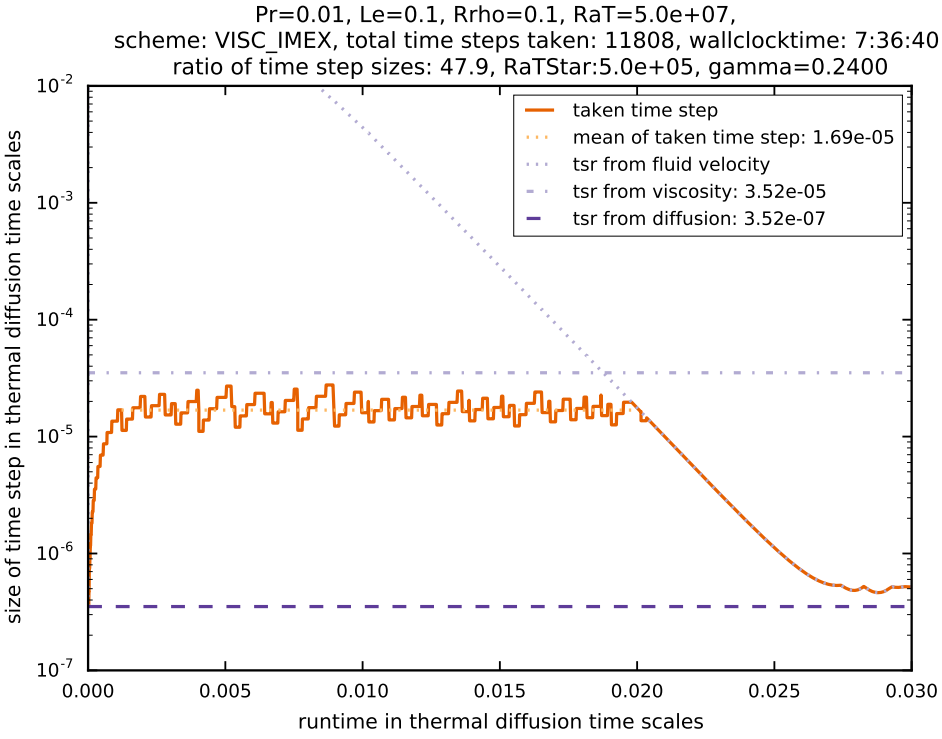


(b) $\gamma = 0.245$

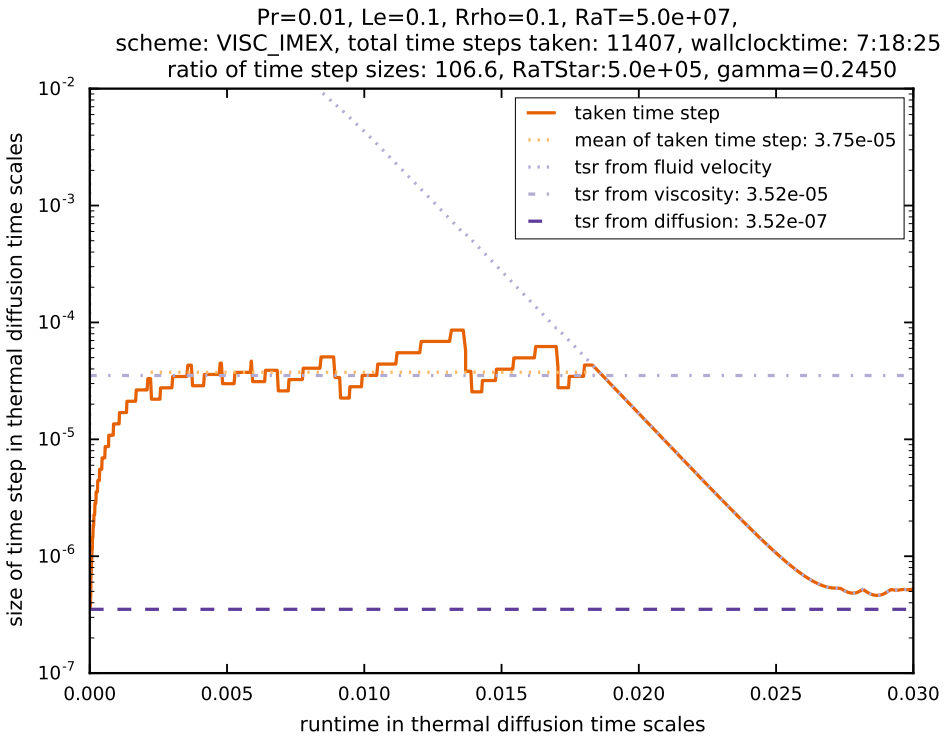


(c) $\gamma = 0.2475$

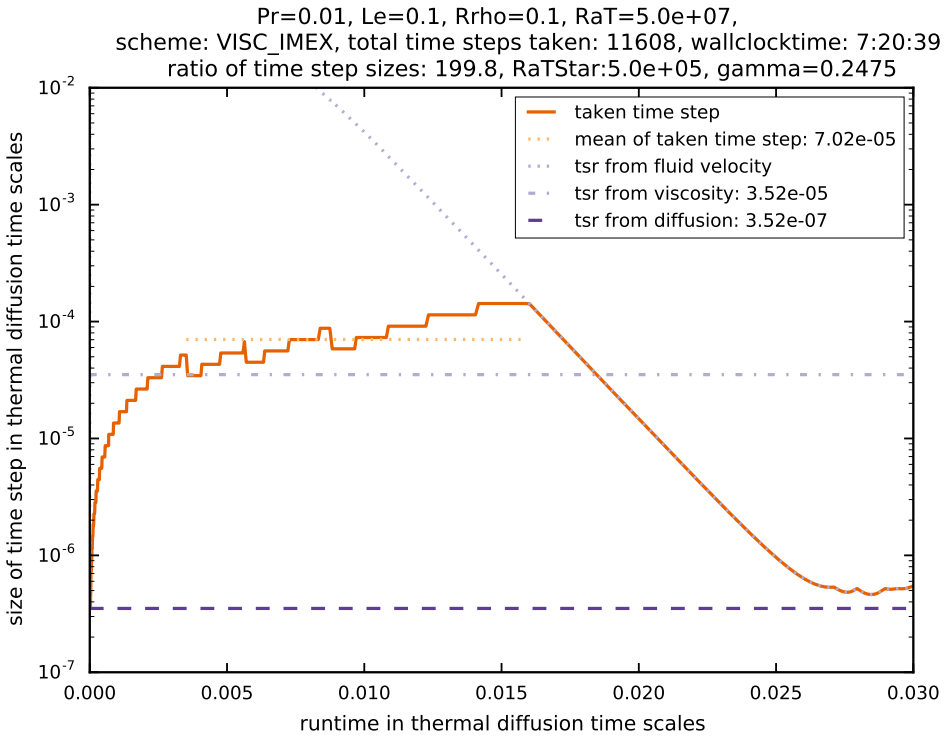
Figure 7.16: Comparison of a modification of γ on the size of the time step for $\mathrm{Pr} = 0.01$. Time integration was performed with DIFF_IMEX.


Pr=0.01, Le=0.1, Rrho=0.1, RaT=5.0e+07,
scheme: VISC_IMEX, total time steps taken: 11808, wallclocktime: 7:36:40
ratio of time step sizes: 47.9, RaTStar:5.0e+05, gamma=0.2400
taken time step
mean of taken time step: 1.69e-05
tsr from fluid velocity
tsr from viscosity: 3.52e-05
tsr from diffusion: 3.52e-07
size of time step in thermal diffusion time scales
runtime in thermal diffusion time scales
10^-2
10^-3
10^-4
10^-5
10^-6
10^-7
0.000
0.005
0.010
0.015
0.020
0.025
0.030

(a) $\gamma = 0.24$



(b) $\gamma = 0.245$

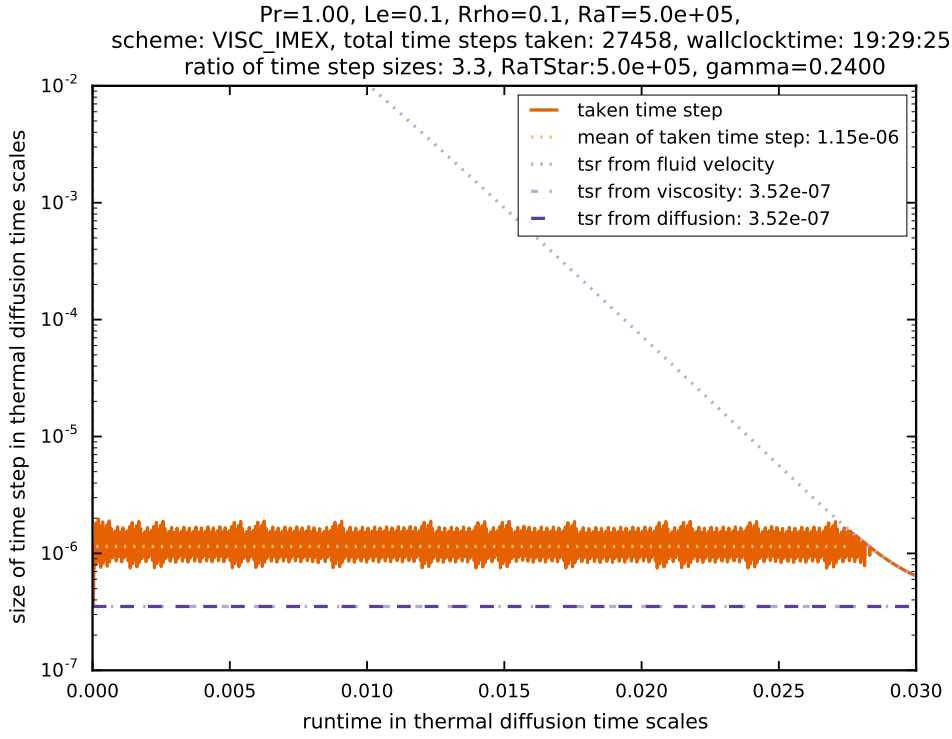


(c) $\gamma = 0.2475$

Figure 7.17: Comparison of a modification of γ on the size of the time step for $\mathrm{Pr} = 0.01$. Time integration was performed with VISC_IMEX.

7.4.2 Increasing γ for Pr = 1

Here, we repeat the SSPRK(3,2) and VISC_IMEX simulations from section 7.3.1 with $\gamma = 0.245$ and $\gamma = 0.2475$. The results are shown in figure 7.18.



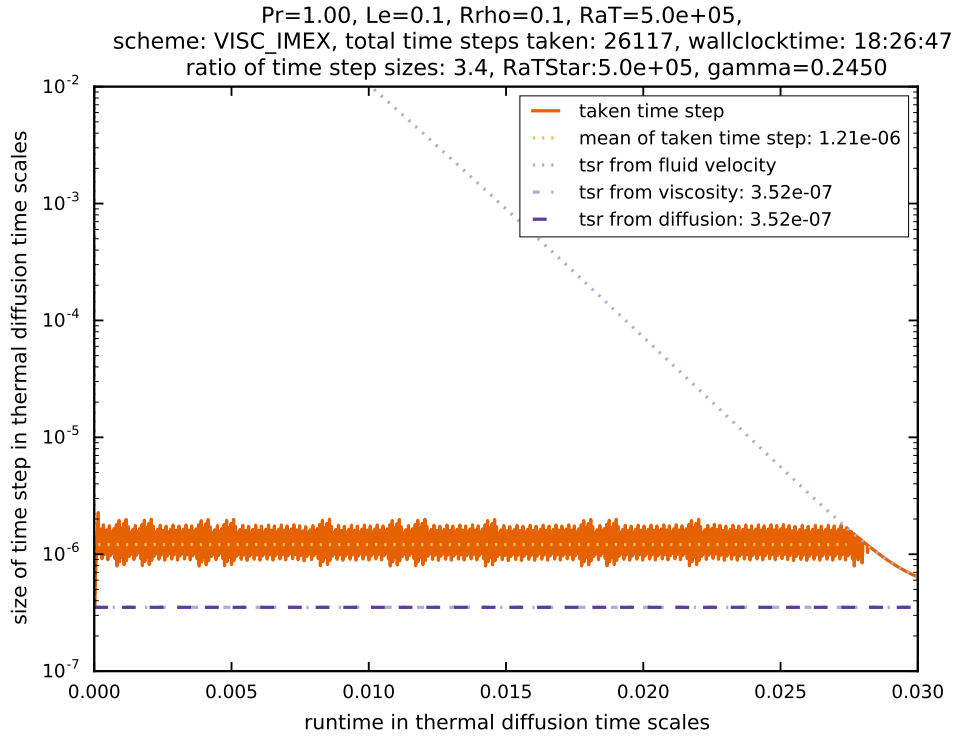
(a) $\gamma = 0.24$

7.4.2.1 Effect on the Size of the Time Step Actually Taken

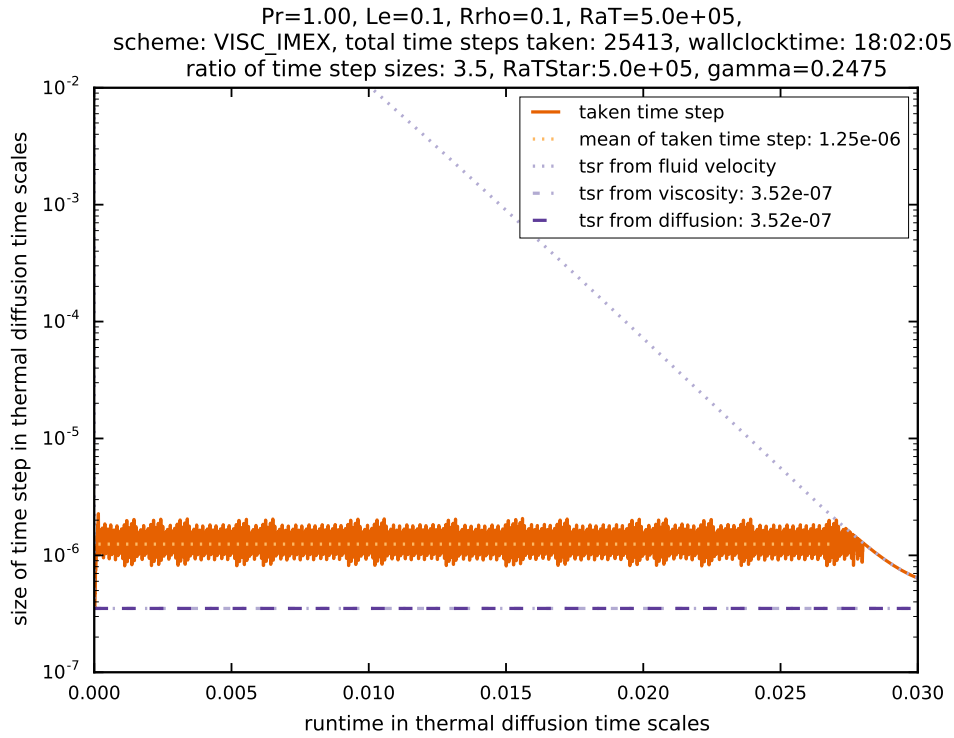
We see that an increase of γ only has a minuscule effect on the average time step actually taken. For each value of γ, there is a large amount of two-point instabilities. No value of γ was able to increase the stability of the method in a way that it did in the simulations with Pr = 0.01. For $\gamma = 0.24$ VISC_IMEX reached its stability limit (fig 7.18(a)) with a ratio of the average taken time step over the time step limit from diffusion of 3.3, for $\gamma = 0.245$ the ratio increased to 3.4 and a further increase of γ to 0.2475 led to a time step ratio of 3.5. This is in stark contrast with Pr = 0.01 where the time step could be increased by a factor of 100. We discuss the likely reason for that in section 7.6.3.

7.4.2.2 Effect on the Wall Clock Time

The effect on the wall clock time is more pronounced than for Pr = 0.01. While the simulation with $\gamma = 0.24$ took 19 hours and 29 minutes to complete, the simulation with $\gamma = 0.2475$ only



(b) $\gamma = 0.245$



(c) $\gamma = 0.2475$

Figure 7.18: Comparison of a modification of γ on the size of the time step for Pr = 1. Time integration was performed with VISC_IMEX.

Pr	method	ratio of time steps	wall clock time
1	VISC_IMEX, $\gamma = 0.24$	3.3	19:29:25
1	VISC_IMEX, $\gamma = 0.245$	3.4	18:26:47
1	VISC_IMEX, $\gamma = 0.2475$	3.5	18:02:05

Table 7.3: Results from the simulations with increased γ for Pr = 1.

needed 18 hours and 2 minutes. While this is not a drastic increase it is sufficient to warrant the use of a higher value of γ for simulations with a Prandtl number of 1. A possible reason for this is that a higher stability of the method leads to fewer two-point oscillations. This in turn leads to fewer time step rejections.

7.4.3 Increasing γ for Pr = 7

Here, we repeat the SSPRK(3,2) and VISC_IMEX simulations from section 7.3.8 with $\gamma = 0.245$ and $\gamma = 0.2475$. The results are shown in figure 7.19 and table 7.4

7.4.3.1 Effect on the Size of the Time Step Actually Taken

We see that an increase of γ only has a minuscule effect on the average time step actually taken. For each value of γ, there is a large amount of two-point instabilities. No value of γ was able to increase the stability of the method in a way that it did in the simulations with Pr = 0.01. For $\gamma = 0.24$ VISC_IMEX reached its stability limit (fig 7.19(a)) with a ratio of the average taken time step over the time step limit from diffusion of 3.3, for $\gamma = 0.245$ the ratio increased to 3.4 and a further increase of γ to 0.2475 led to a time step ratio of 3.5. These are exactly the same values as for the simulations with Pr = 1.

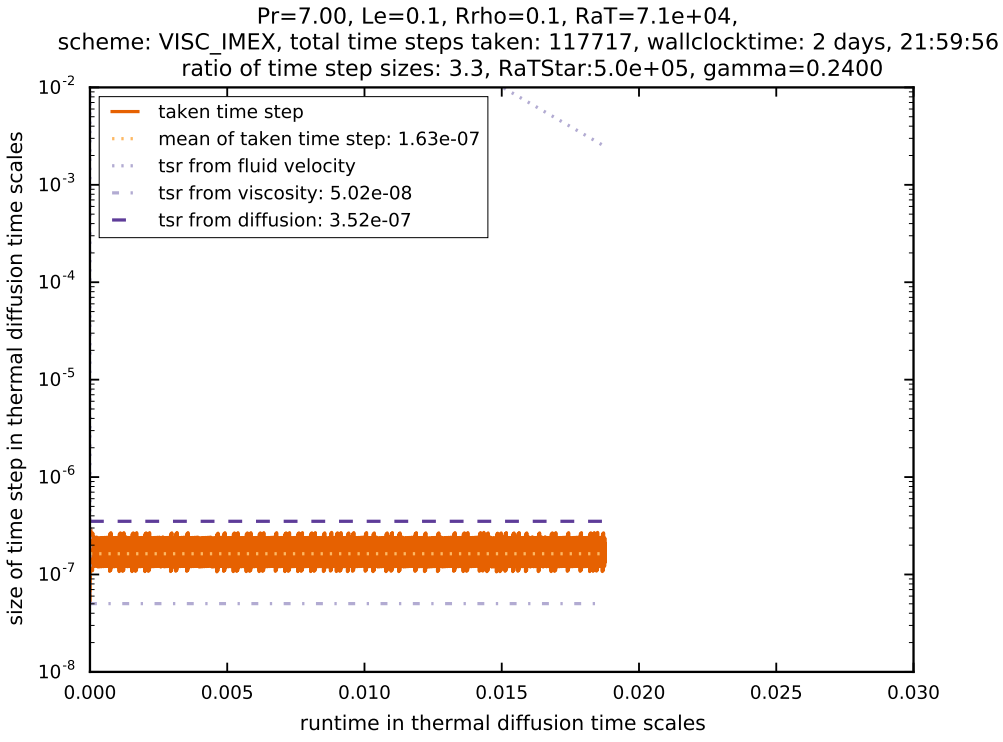
7.4.3.2 Effect on the Wall Clock Time

For the simulation with Pr = 7 we cannot measure the wall clock time because the simulations have been aborted after a runtime of 72 hours. Instead, we measure the effect of increasing γ with the simulated time that has been reached in a wall clock time of 72 hours. We see that similarly to the simulation with Pr = 1 that the highest γ yields the most efficient simulation for Pr = 7. This is likely again due to fewer time step rejections as a consequence of an enhanced stability due to a larger value of γ.

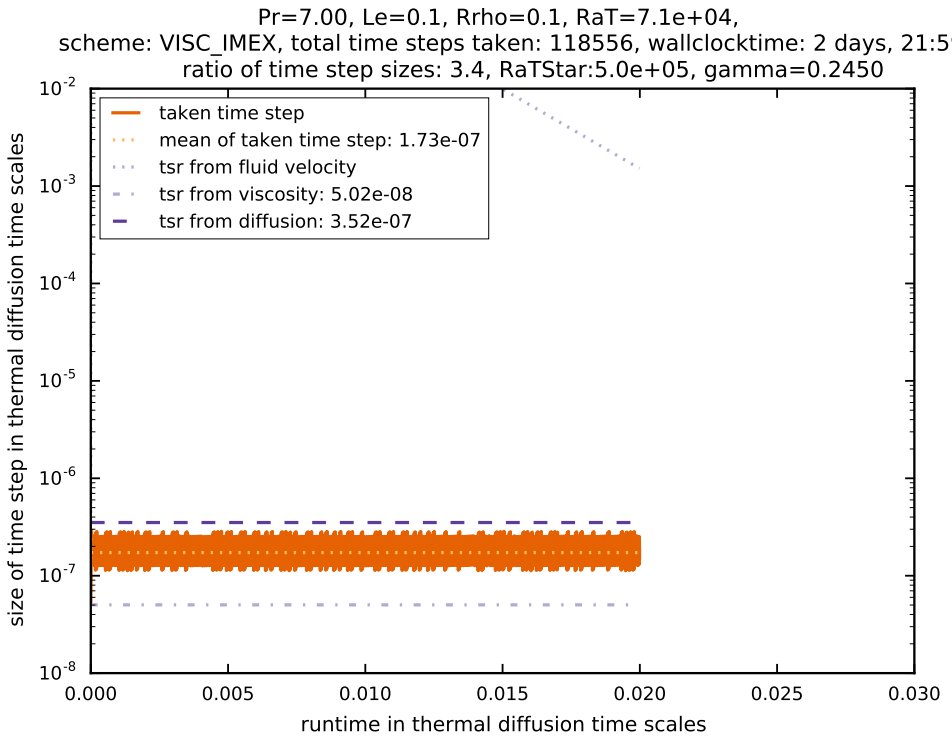
Next, we investigate the behavior of the IMEX routines for three dimensions. Before we can do that, we reduce the resolution quite a bit, however.

Pr	method	ratio of time steps	simulated time [thermal diff. time scales]
7	VISC_IMEX, $\gamma = 0.24$	3.3	1.876e-2
7	VISC_IMEX, $\gamma = 0.245$	3.4	1.999e-2
7	VISC_IMEX, $\gamma = 0.2475$	3.5	2.023e-2

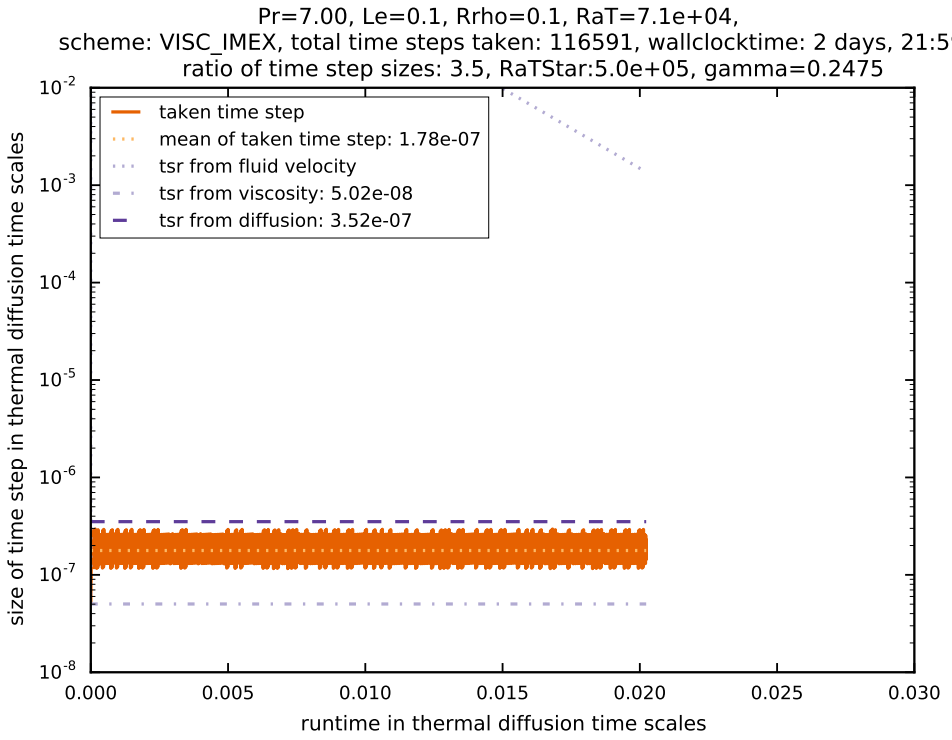
Table 7.4: Results from the simulations with increased γ for Pr = 7.



(a) $\gamma = 0.24$



(b) $\gamma = 0.245$



(c) $\gamma = 0.2475$

Figure 7.19: Comparison of a modification of γ on the size of the time step for Pr = 7. Time integration was performed with VISC_IMEX.

7.5 Three-Dimensional Experiments

7.5.1 Reducing the Resolution to Prepare for 3D Simulations

Simply adding a third dimension to our 2D grid of $800 \times 800 = 6.40 \times 10^5$ grid points and thereby creating a grid consisting of $800 \times 800 \times 800 = 5.12 \times 10^8$ grid points is not possible with our limited resources. We have to reduce the number of grid points in order to get a manageable amount of computational work. The clearest advantage of using implicit integration for diffusive and viscous terms has been observed in the parameter setting with $\mathrm{Pr} = 0.1$. The achieved acceleration was 2.3. We run a 2D simulation with a much reduced resolution now to investigate whether the results so far hold with a decreased resolution.

In the simulations with 800×800 grid points, we had 20 points in the thermal boundary layer and 6 points in the helium boundary layer in x-direction. We now relax this condition and only require 10 points to be in the thermal boundary layer. This results in 3 points in the helium boundary layer in x-direction. The resolution in y-direction is half of that because of the rectangular structure of the grid (see F. Zaussinger, 2010, for an explanation as how to determine the number of points in a boundary layer in ANTARES). We now run a simulation with 272×272 grid points on 8×8 processors, once for each time integration routine.

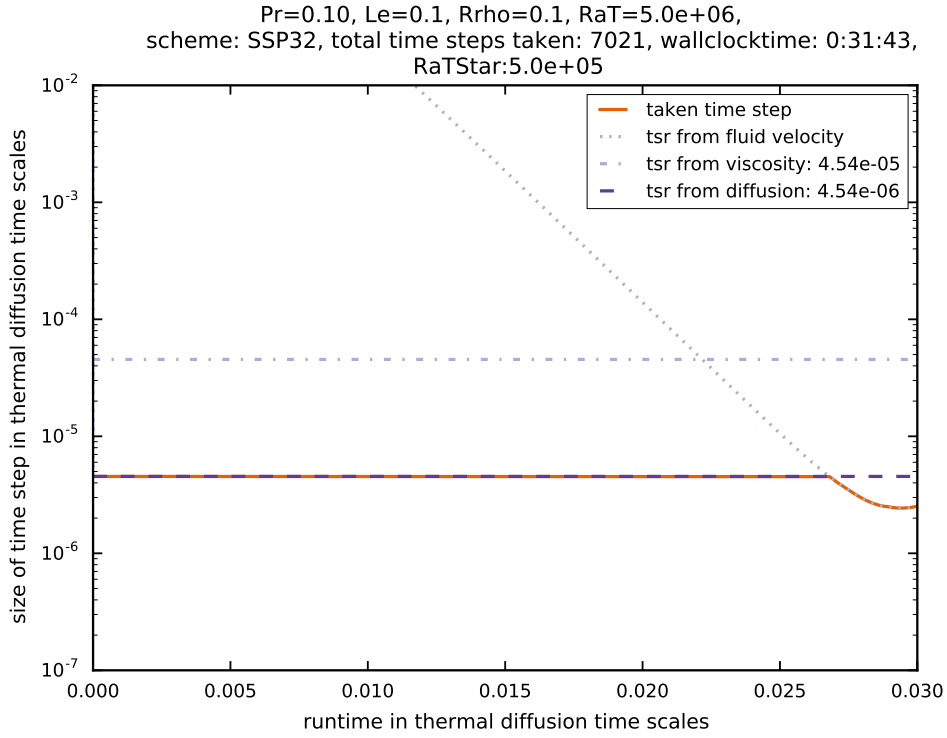
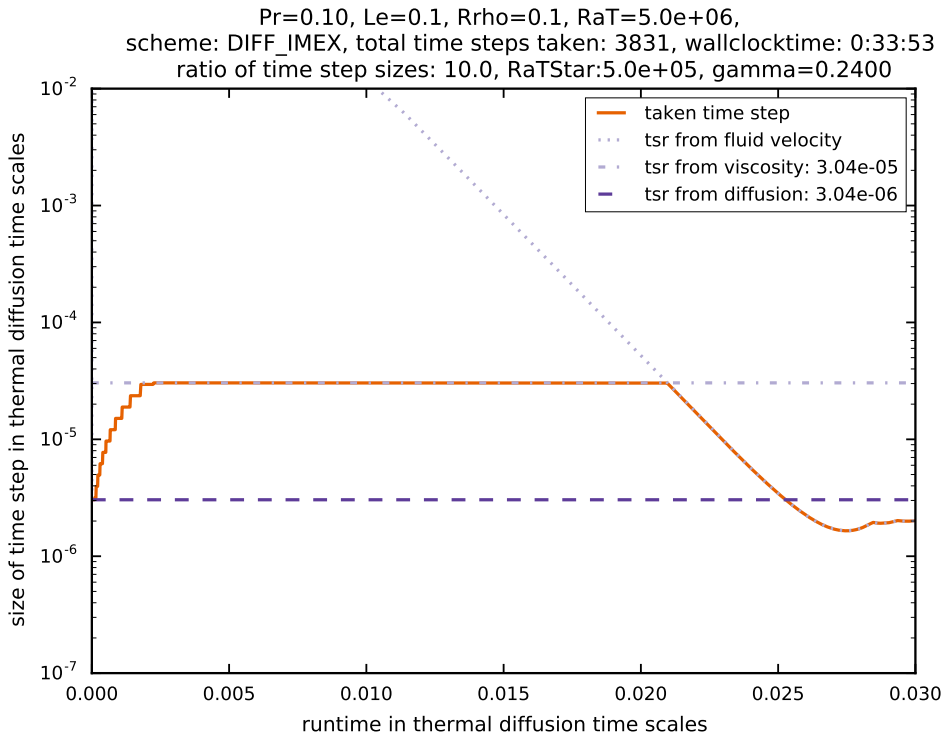


Figure 7.20: Time step evolution for $\mathrm{Pr} = 0.1$ and a reduced resolution of 272×272 grid points. Time integration was performed with SSPRK(3,2).

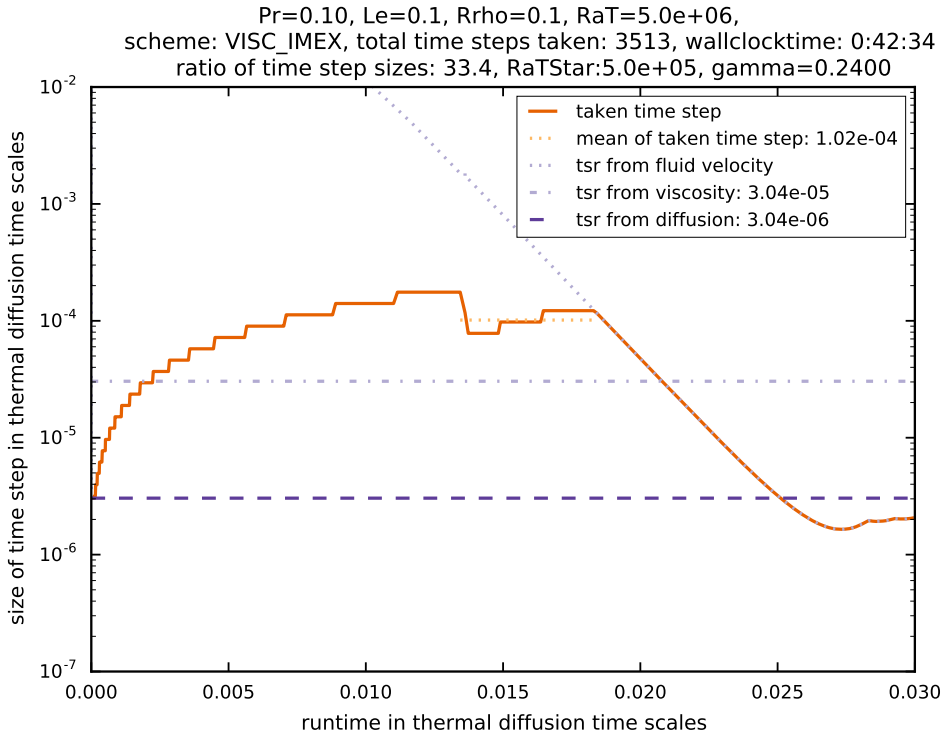
The time step evolution for the explicit SSPRK(3,2) can be found in figure 7.20 . We see that the result is very similar to the highly resolved simulations with the same parameters. The time step actually taken is limited by diffusion because that is the most restrictive limit on the time step. The reason for the time step limits being larger than in the highly resolved cases is that the spatial resolution is one of the factors entering the equations (6.9) - (6.12). So a low spatial resolution also permits a low temporal resolution. This accelerates the simulation: instead of needing almost 19 hours on 256 cores, this simulation only needed half an hour on 64 cores. This is exactly what we want for the three-dimensional simulations and the reason why we reduced the resolution in the first place. It also demonstrates a good scaling (in the weak sense) of the entire computational approach.

Figure 7.21 shows the results for DIFF_IMEX and VISC_IMEX. Qualitatively, the results are the same as in the highly resolved simulation: using implicit integration just for diffusion leads to an increased time step, limited by viscosity. Using implicit integration for diffusive and viscous terms yields a larger increase of the actually used time step until the time step control detects two-point instabilities and reduces the time step again.

Both methods lead to stable simulations so we think that a resolution of 272 points per dimension suffices for three dimensional experiments.



(a) Implicit integration for diffusive terms only.



(b) Implicit integration for diffusive and viscous terms.

Figure 7.21: Time step evolution for $\mathrm{Pr} = 0.1$ and a reduced resolution of 272×272 grid points. Time integration was performed with DIFF_IMEX and VISC_IMEX.

7.5.2 The Results for Three Dimensions

We use the same parameters for the 3D experiments as we did for the 2D experiment with reduced resolution. The parameters can be found in table 7.5 .

Pr	Le	R_ρ	Ra_T	Ra_T^*
0.1	0.1	0.1	5.00E+06	5.00E+05

Table 7.5: Simulation parameters for the 3D experiments. The simulations have been performed with SSP32, DIFF_IMEX and VISC_IMEX. The resolution was $272 \times 272 \times 272$ in each case. The simulations have been performed on $8 \times 8 \times 8$ cores on the VSC2.

7.5.2.1 Explicit Time Integration

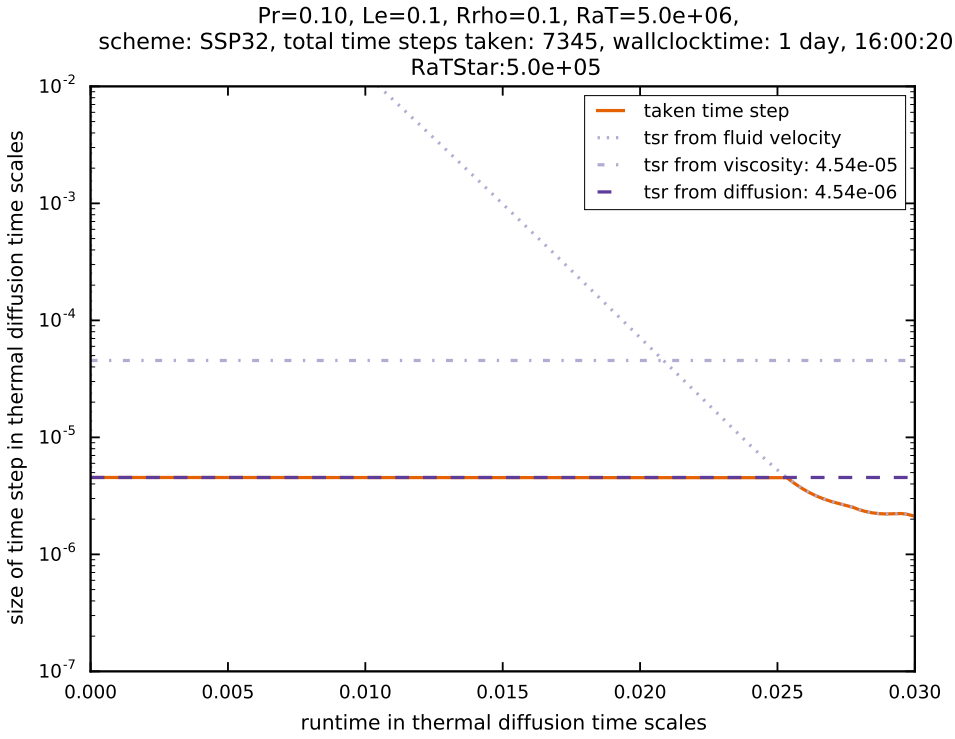


Figure 7.22: Time step evolution for $Pr = 0.1$ in three dimensions with a resolution of $272 \times 272 \times 272$. Time integration was performed with SSPRK(3,2).

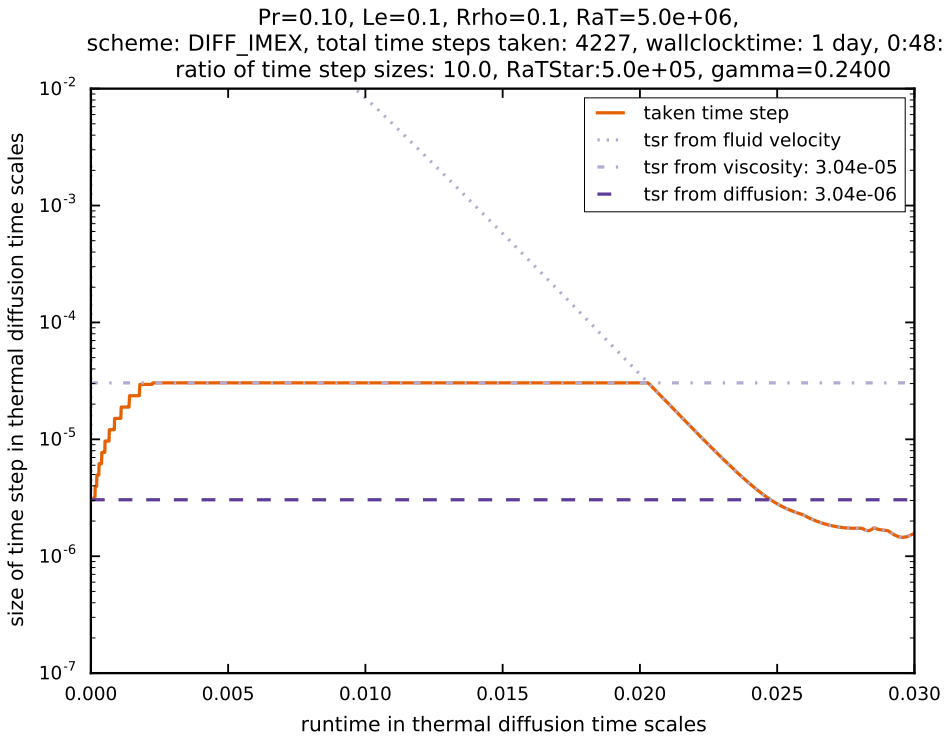
We see from figure 7.22 that the result for the three dimensional experiments is very close to the result of the two dimensional experiments with reduced resolution in section 7.5.1.

To reach 0.03 diffusion time scales, ANTARES needed 7345 steps which is 324 steps more than in the 2D case. The simulation was completed after 40 hours. In contrast, the 2D simulation with the same parameters was completed after 32 minutes.

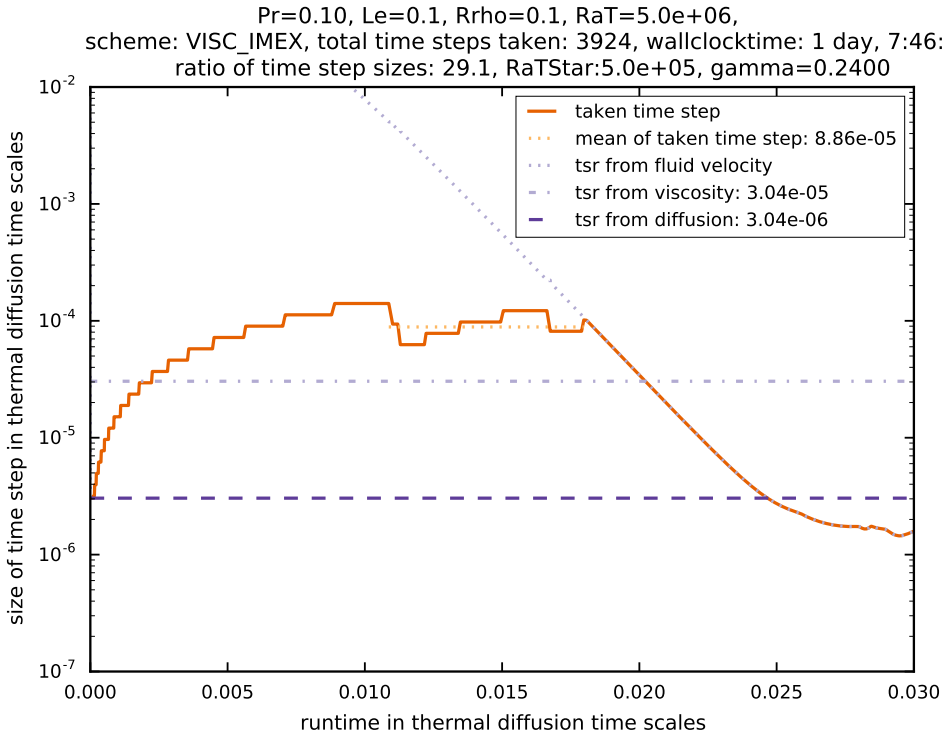
Next, we investigate the benefit of using implicit-explicit Runge–Kutta methods in three dimensions.

7.5.2.2 IMEX Time Integration

The results for the IMEX runs are shown in figure 7.6. One can notice at once that the qualitative evolution of the time step looks almost exactly as in the two dimensional run with reduced resolution. This is a very good result because it shows that the IMEX schemes are correctly implemented in three dimensions. It also demonstrates the correctness of the multigrid solver that has been developed as part of this thesis because that solver was used in these simulations to solve the elliptic equation arising from the implicit integration. Because the results are so similar to the 2D simulations, almost the same can be said in summary: using DIFF_IMEX leads to an increased time step, limited by viscosity. Using VISC_IMEX yields a larger increase of the actually used time step until the time step control detects two-point instabilities and reduces the time step again.



(a) Implicit integration for diffusive terms only.



(b) Implicit integration for diffusive and viscous terms.

Figure 7.23: Evolution of time step size for $\mathrm{Pr} = 0.1$ in three dimensions with a resolution of $272 \times 272 \times 272$. Time integration was performed with IMEX methods.

7.5.3 A Few Words on the Scaling Properties of the 3D Helmholtz Solver

Taking a short detour from our investigation of the IMEX methods, we can use the simulations performed to investigate the scaling properties of the solver that was developed in part I. The 2D simulations used the solver by Happenhofer, (2014) while the 3D simulations used our solver. This gives us a good opportunity to estimate the overhead that is incurred by the 3D solver compared to the 2D solver.

To obtain a baseline for comparison we first calculate the workload per processor W, i.e., the number of grid points divided by the number of processors. For the two- and three-dimensional case we have, respectively,

$$\begin{aligned} W_{2d} &= \frac{\text{number of grid points in 2D}}{\text{number of processors in 2D}} = \frac{276^2}{64} = \frac{73984}{64} = 1156 \\ W_{3d} &= \frac{\text{number of grid points in 3D}}{\text{number of processors in 3D}} = \frac{276^3}{512} = \frac{20123648}{512} = 39304. \end{aligned} \tag{7.1}$$

We see that $W_{3d}/W_{2d} = 34$ which means that – neglecting overheads like communication – performing simulations in three dimensions should be 34 times slower than performing the same simulations in two dimensions. From looking at the wall clock times actually achieved in table 7.6 we see that 3D simulation with explicit time integration is 75.6 times slower than the 2D simulation with explicit time integration which suggests an overhead of the explicit solver (plus one implicit equation for the pressure correction in the Boussinesq approximation) of about 120%. Now, to make the statement "our developed Helmholtz solver scales well", we would need to achieve an overhead in the IMEX simulations with is lesser or equal to 120%. As we see from table 7.6 this is indeed the case: the 3D simulations with IMEX are only a factor of about 44 times slower than the 2D simulations. This is much less than in the explicit case when the Helmholtz solvers are not used (which is also subject to overhead due to communication and the Poission equation for the pressure which is solved each time step).

This leads us to solidify our conclusion in section 4.4 that the scaling of the 3D Helmholtz

Method and Dimensionality	Actual Wall Clock Times	Normalized WCT	Overhead to 2D
2D explicit	00:31:43	1	
3D explicit	40:00:20	75.7	2.2
2D DIFF_IMEX	00:33:53	1	
3D DIFF_IMEX	24:48:22	43.9	1.3
2D VISC_IMEX	00:42:34	1	
3D VISC_IMEX	31:46:14	44.8	1.3

Table 7.6: An overview of wall clock times for 2D and 3D simulations and different time integration routines. Grid points in 2 dimensions: 276^2, grid points in 3 dimensions: 276^3, MPI processes in 2 dimensions: 64, MPI processes in 3 dimensions: 512 .

solver is very good, even when using it for real world applications instead of analytical toy problems.

7.6 Discussion of the Results

7.6.1 Comparing the Size of the Time Steps

In order to investigate the mathematical properties of the IMEX methods we now take a look at the size of the actual time steps that have been achieved with each method. To this end, we first look at the size of the time steps when using the explicit SSPRK(3,2) method.

7.6.1.1 Time Steps with explicit SSPRK(3,2)

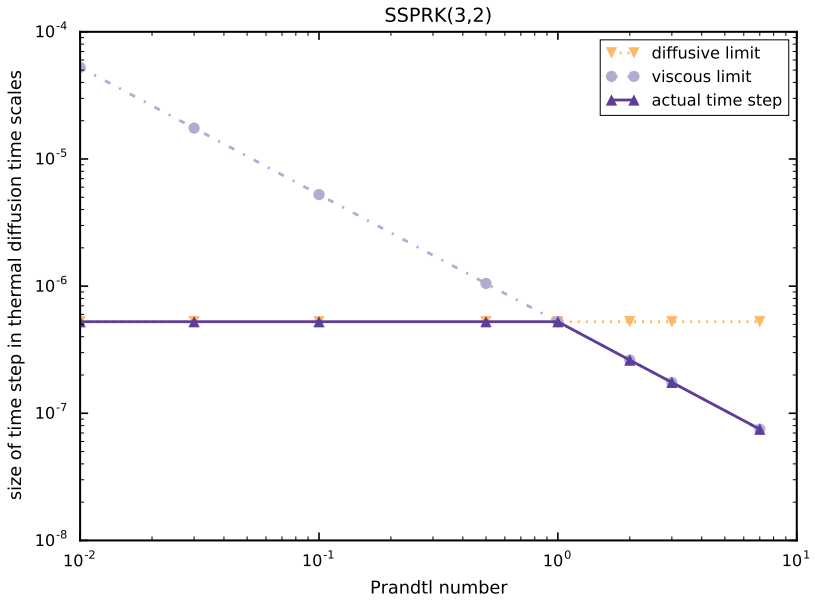


Figure 7.24: A plot of the size of time steps in dependence of the Prandtl number when using SSPRK(3,2). Also shown are the diffusive and viscous limits for the corresponding Prandtl numbers. It turns out that the actual time step is the minimum of the diffusive and the viscous limit.

We can see in in figure 7.24 that the time step actually taken when using explicit integration is indeed the minimum of the diffusive and the viscous limits. For Prandtl numbers $\mathrm{Pr} < 1$ the diffusive time scale is the most stringent one and for $\mathrm{Pr} > 1$ the viscous time scale is the most stringent one. This is in accordance to theory: in explicit methods, the smallest time scales determine the time step actually taken.

Next, we investigate which impact the use of DIFF_IMEX has on the time step actually taken.

7.6.1.2 Time Steps with DIFF_IMEX

The situation when using implicit integration for the diffusive terms can be seen in figure 7.25. We observe that the time step actually taken when using implicit integration for diffusive terms

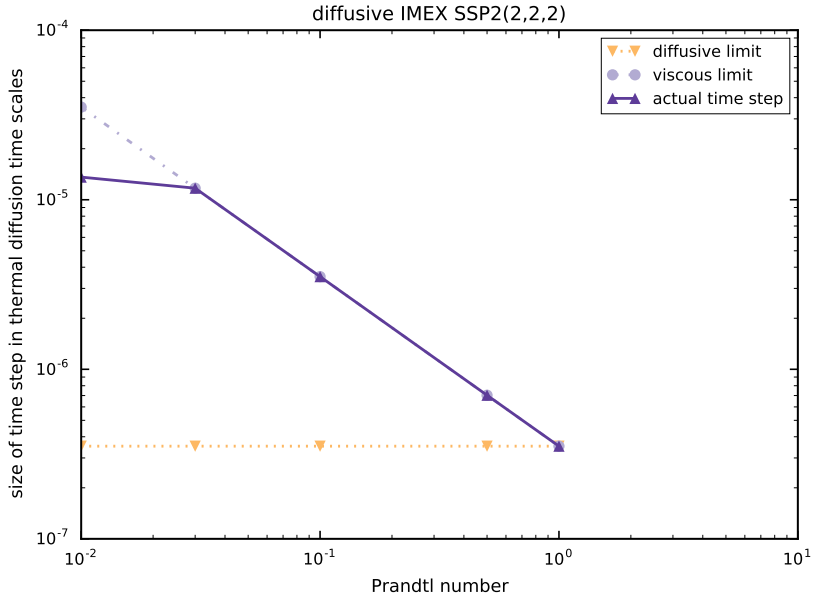


Figure 7.25: A plot of the size of time steps in dependence of the Prandtl number when using implicit integration for diffusive terms only and a γ of 0.24. Also shown are the diffusive and viscous limits for the corresponding Prandtl numbers. The actual time step is larger than the diffusive limit for every Prandtl number. The size of the time step is limited by the viscous time scale for Prandtl numbers up to 0.1. Somewhere between 0.01 and 0.03, the stability of DIFF_IMEX puts a limit on the achievable time step.

is above the limit posed by diffusive processes for every Prandtl number we used diffusive IMEX for. For small Prandtl numbers ($Pr < 0.1$) the actual time step is limited by the intrinsic stability of the IMEX SSP2$(2,2,2),\gamma = 0.24$ scheme, i.e., two-point oscillations occur and limit a further increase of the time step actually taken. As we have seen in section 7.4, if we increase γ from 0.24 to 0.245 or 0.2475 the size of the time step increases even more because the stability region is enlarged by increasing γ and hence, no two-point instabilities (and thus the stability of the method) limit the time step but the viscous limit.

When increasing the Prandtl number, one reaches the point where the viscous time step limit becomes important. This happens at around $Pr = 0.03$ in figure 7.25. From this point on, an increase of the Prandtl number leads to a decrease in the time step actually taken because the viscous limit becomes more stringent with increasing Prandtl number.

7.6.1.3 Time Steps with VISC_IMEX

The relationship of the time step actually taken and the Prandtl number when using implicit integration for diffusive and viscous terms is shown in figure 7.26. For VISC_IMEX, there is only the advective time scale as an externally imposed limit on the time step. What this means is that the maximally possible time step for VISC_IMEX stems purely from the properties of

the underlying Runge–Kutta scheme. This allows us to investigate the behavior of the scheme directly.

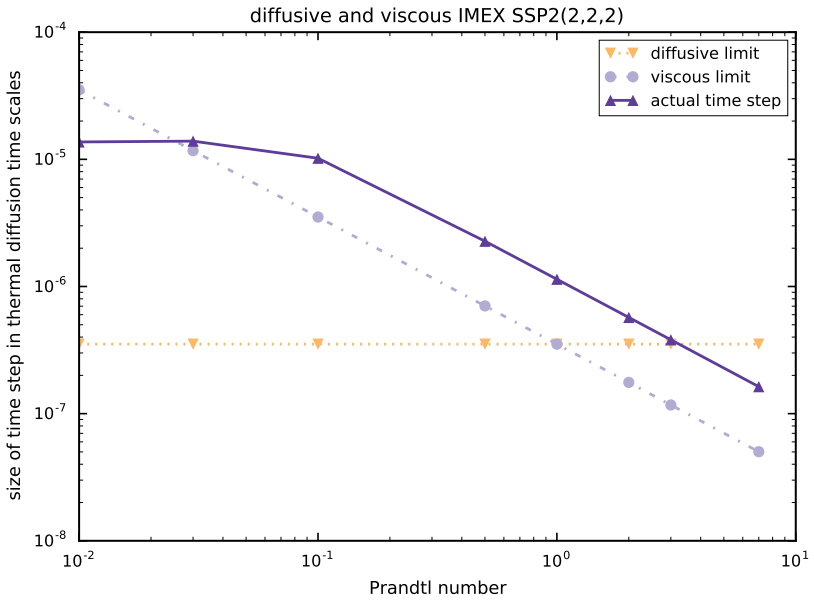


Figure 7.26: A plot of the size of time steps in dependence of the Prandtl number when using implicit integration for both diffusive and viscous terms. The actual time step is above the diffusive limit for every Prandtl number.

For Prandtl numbers $\mathrm{Pr} \in \{0.01, 0.03\}$ the method shows the same behavior as when using DIFF_IMEX, i.e, the time step actually used is limited by the overall stability of the IMEX SSP2(2,2,2) method.

For Prandtl numbers larger than about 0.1, the viscous IMEX method permits a consistent improvement of time steps about 3 times larger for each Prandtl number. This behavior is an indicator for the occurrence of an intrinsic limit of VISC_IMEX SSP2(2,2,2). A possible explanation is an insufficient damping property of the IMEX method which becomes apparent when the time step actually taken is about 3 times as large as the viscous limit allows.

7.6.2 Comparing the Achieved Acceleration

After having investigated the mathematical properties of the method, we now want to answer the question that is most relevant to users of ANTARES: based on these experiments, which time integration scheme should be used when performing simulations with a certain Prandtl number? To answer that question, we have plotted the acceleration in terms of wall clock times vs. the Prandtl number for each time integration scheme in figure 7.27. We only have a very limited number of data points available and the connecting lines between the data points are only there

to help visualizing the relationship between acceleration and Prandtl number. The numerical

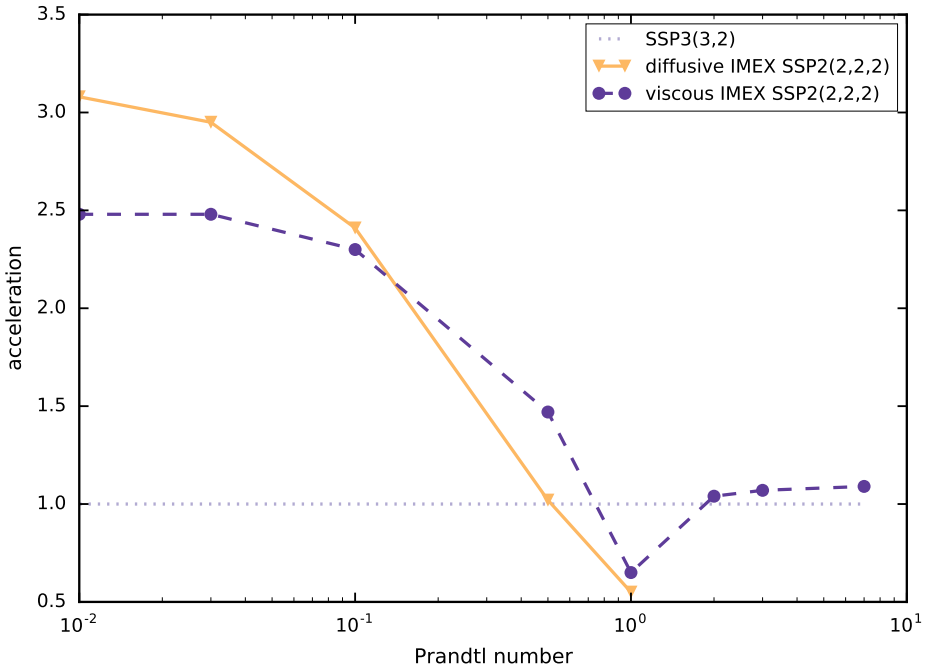


Figure 7.27: Plot of the achieved accelerations in two dimensions in dependence of the Prandtl number for the used time integration methods. Simulation parameters are $\mathrm{Le} = 0.1, \mathrm{R}_\rho = 0.1, \mathrm{Ra}_\mathrm{T}^* = 5 \times 10^5$. The resolution is 800×800 grid points. This is a plot of table 7.7

values for the accelerations can be found in table 7.7.

	acceleration for	
Pr	DIFF_IMEX	VISC_IMEX
0.01	3.18	2.48
0.03	2.95	2.48
0.1	2.41	2.30
0.5	1.02	1.47
1	0.55	0.65
2	n/a	1.04
3	n/a	1.07
7	n/a	1.09

Table 7.7: Achieved accelerations in two dimensions for different Prandtl numbers. Simulation parameters are $\mathrm{Le} = 0.1, \mathrm{R}_\rho = 0.1, \mathrm{Ra}_\mathrm{T}^* = 5 \times 10^5$. The resolution is 800×800 grid points. These are the numerical results for figure 7.27.

7.6.2.1 Low Prandtl Numbers

We observe the following for the acceleration in dependence of the Prandtl number: for low Prandtl numbers, using the IMEX method yields a significant boost in terms of wall clock times. VISC_IMEX gives an acceleration of about 2.5 whereas DIFF_IMEX even gives us an acceleration of about 3. This makes sense because the limiting time scale for Pr = 0.01 is the diffusive time scale and both DIFF_IMEX as well as VISC_IMEX are limited by the diffusive time scale (see figures 7.10(a) and 7.10(b)) due to the finite stability region of IMEX2$(2,2,2)$ at $\gamma = 0.24$. In this case, VISC_IMEX does not have any advantages over DIFF_IMEX. The higher computational cost of the method leads to an inferior acceleration for low Prandtl numbers in comparison to DIFF_IMEX. For Pr = 0.03 the situation is similar

For Pr = 0.1 the situation is a little bit more complicated. While initially, the diffusive time scale is limiting the time step, DIFF_IMEX alleviates this restriction. However, when the time step is increased tenfold, the viscous time limit prevents further acceleration. That is why the acceleration for DIFF_IMEX is not as large for the Pr = 0.1 simulation as it is for the Pr = 0.01 simulation. For VISC_IMEX, the viscous time step limit only sets in once the stability limit of the IMEX2$(2,2,2)$ method at $\gamma = 0.24$ is reached which is not the case for this problem with Pr = 0.1. So the actual time step is further increased. In terms of acceleration, when using a Prandtl number of Pr $\in [0.01, 0.1]$ VISC_IMEX is not able to beat DIFF_IMEX, however. The necessary solution of 2 additional elliptic equations in each stage of the Runge–Kutta scheme is too expensive.

7.6.2.2 Medium Prandtl Numbers

Once the viscous time step limit becomes small enough (i.e., the Prandtl number sufficiently high), VISC_IMEX shows an advantage over DIFF_IMEX. For Pr = 0.5, the latter has no significant advantage over the explicit SSPRK(3,2) scheme whereas VISC_IMEX leads to an acceleration of about 1.5. Somewhere between Pr = 0.5 and Pr = 1, the acceleration of VISC_IMEX drops below 1, too, and for a Prandtl number of 1, the explicit SSPRK(3,2) is superior to both IMEX methods.

7.6.2.3 High Prandtl Numbers

For Prandtl numbers above 2, VISC_IMEX yields a small improvement over the explicit scheme. It is rather negligible, however, especially for simulations of semiconvection, where one is usually interested in the advective part of the simulations, i.e., the time where convection has already set in. In these parts, SSPRK(3,2) has a significant advantage over the IMEX schemes because of its higher advective Courant numbers and lower computational cost.

Of course, these results are only a small step towards a thorough investigation with many more numerical experiments in order to generate more data and to solidify the first results we have found. But despite the relative lack of data, we can observe a trend for the acceleration in

dependence of the Prandtl number. And since this trend is in accordance with theory, we have no reason not to believe the results.

An investigation of achievable acceleration as a function of other parameters as e.g. the Lewis number of Ra_T^* can be the subject of future research.

7.6.3 Discussion on the Influence of γ

Pr	method	ratio of time steps	wall clock time OR runtime in dts
0.01	DIFF_IMEX, $\gamma = 0.24$	47.3	06:06:42
0.01	DIFF_IMEX, $\gamma = 0.245$	95.9	05:55:23
0.01	DIFF_IMEX, $\gamma = 0.2475$	100	05:57:47
0.01	VISC_IMEX, $\gamma = 0.24$	47.9	07:36:40
0.01	VISC_IMEX, $\gamma = 0.245$	106.6	07:18:25
0.01	VISC_IMEX, $\gamma = 0.2475$	199.8	07:20:39
1	VISC_IMEX, $\gamma = 0.24$	3.3	19:29:25
1	VISC_IMEX, $\gamma = 0.245$	3.4	18:26:47
1	VISC_IMEX, $\gamma = 0.2475$	3.5	18:02:05
7	VISC_IMEX, $\gamma = 0.24$	3.3	1.876e-2
7	VISC_IMEX, $\gamma = 0.245$	3.4	1.999e-2
7	VISC_IMEX, $\gamma = 0.2475$	3.5	2.023e-2

Table 7.8: Overview of the effect that γ has on the ratio of the average time step actually used in IMEX simulations to one used in simulations with explicit time integration. For simulations with Pr = 7 the simulations have been aborted at a wall clock time of 72 hours. For that reason, the runtime in thermal diffusion time scales (dts) is given instead of the wall clock times.

7.6.3.1 The Influence of γ on the Size of the Time Step

For simulations with a low Prandtl number, we have seen a clear effect of γ on the stability of the IMEX scheme and hence the achievable time step. The average time step actually used with Pr = 0.01 and $\gamma = 0.2475$ was 4.2 times as large as the one used with $\gamma = 0.24$.

This effect becomes much less pronounced when increasing the Prandtl number, however. For both the simulations with Pr = 1 and Pr = 7 the time step ratio for $\gamma = 0.2475$ was only 1.06 times as large as the time step ratio when using $\gamma = 0.24$.

This solidifies our conclusion about the IMEX SSP2(2,2,2) method:

- for low values of the Prandtl number, the method yields good results and the achievable time step is limited by the stability region of the method. Increasing the stability region (by choosing a larger γ) yields larger time steps.

- for high values of the Prandtl number, the method yields only small improvements of the average time step actually taken. A possible explanation is that the method might have insufficient damping properties.

One thing to take note of is that the ratio of the the time step actually used to the *diffusive* time step limit can be significantly increased by changing γ.

7.6.3.2 The Influence of γ on the Wall Clock Times

For productions runs, when simulations can take several months to complete, the value one tries to minimize is the wall clock time. Our results show that for complex applications with complex numerical codes, it is often not the method with the most favorable mathematical property that emerges as the optimal method. One must rather account for many inter-dependencies which are present in a numerical code.

Seeing the increase in the size of the time step for the simulations with a small Prandtl number one could assume that this would lead to significant improvements in the wall clock times, too. However, the wall clock times hardly differ for these different methods. How can this be explained? The reason for this is probably twofold:

- the larger the time step actually taken is, the worse the initial guess for the multigrid correction is, i.e., the multigrid solver needs more iterations to converge and takes a longer time to run.

- The larger the time step actually taken is, the smaller $\xi(\boldsymbol{x})$ in (6.36) becomes. $\xi(\boldsymbol{x})$ has major implications for the numerical efficiency of the multigrid solver, however: the larger $\xi(\boldsymbol{x})$ is the faster the solver converges.

The efficiency of the solver for the elliptic equations is of crucial importance in these applications: it is called 8 times per Runge–Kutta time step for 2D VISC_IMEX simulations. The simulation with Pr = 7 and $\gamma = 0.2475$ performed 116591 time steps, which gives 932728 calls to the multigrid solver. It is understandable that even a small improvement in its efficiency can have a large impact on the overall performance.

7.7 Summary

In this section, we have compared the performance of the explicit SSPRK(3,2) method with two versions of the implicit-explicit SSP2$(2,2,2),\gamma = 0.24$ method: one version where we have used implicit integration for diffusive terms only (named “DIFF_IMEX”) and one version where we have used implicit integration for diffusive and viscous terms (named “VISC_IMEX”). The performance measures have been the acceleration (i.e., the ratio of wall clock times of the

computations) and the size of time steps taken. The latter performance measure is of theoretical interest only.

While it is mathematically interesting to find out that VISC_IMEX consistently gives us a time step about three times larger than the viscous time step limit for Prandtl numbers above ≈ 0.1, it is more important to know which time integration scheme leads to the best performance when using ANTARES. This is measured by the ratio of wall clock times. Based on the measurements depicted in figure 7.27, we propose to use the following time integration method in dependence of the Prandtl number:

- for Prandtl numbers $\lesssim 0.1$: use DIFF_IMEX. Since the time step is increased all the way up to the stability limit of the SSP2$(2,2,2),\gamma = 0.24$ method, there is no further increase in acceleration possible by using VISC_IMEX.
- for Prandtl number $0.1 \lesssim \mathrm{Pr} \lesssim 0.8$: use VISC_IMEX. In this parameter regime, the viscous time step limit is sufficiently low for VISC_IMEX to have a significant effect on the acceleration. The time step is augmented sufficiently to offset the increased computational cost of VISC_IMEX compared to DIFF_IMEX.
- for Prandtl numbers $\gtrsim 0.8$: use explicit SSPRK(3,2). While VISC_IMEX does show a small acceleration, it is slower than the explicit scheme once the simulation reaches the advective phase. Since simulations with ANTARES are primarily concerned with the advective phase, it can be assumed that the advective phase will cover a significant part of the overall simulation runtime. The very small acceleration that IMEX leads to in these parameters regime does not warrant a use of it in the overall simulation.

Of course, these results should be checked for the parameter setting in question when planning to use IMEX in a long simulation. It could very well be the case that different values of $\mathrm{Le}, \mathrm{R}_\rho$ or Ra_T change the results that we have obtained here. A thorough investigation of different parameter regimes can be the subject of further research.

We have also investigated the impact of an increase of γ for a small subset of the simulations with the following results:

- for low values of the Prandtl number, increasing γ leads to significantly larger time steps which led us to conclude that the time step actually taken when using IMEX SSP2(2,2,2) is limited by its stability region for low values of Pr. The wall clock time could not be reduced by a substantial amount, however, because too large time steps lead to unfavorable numerical properties for the multigrid solver
- For high values of the Prandtl number, increasing γ yields only small improvements of the average time step actually taken. A possible explanation is that the method might have insufficient damping properties for large values of Pr. The wall clock time, however, could be reduced by a small amount when choosing a large value for γ.

This concludes part II. We have successfully developed and tested a strong-stability preserving implicit-explicit Runge–Kutta method for viscous flows. We will summarize this part of the thesis in the next chapter.

CHAPTER 8

SUMMARY OF PART II

In this part we have started from the following problem statement: each time scale of physical processes must be resolved when using explicit time integration methods for the numerical solution of the governing equations for physical systems. The scales of the various processes can differ by substantial amounts, sometimes several orders of magnitude. This poses stringent restrictions on the time step which can be taken to solve the equations numerically. Over time, different methods to deal with these restrictions have been implemented into ANTARES, one example being implicit-explicit (IMEX) Runge–Kutta methods for diffusively limited compressible flows. When integrating the governing equations with these IMEX methods, the processes with the most stringent time step restrictions are integrated implicitly, whereby the restriction on the overall time step is alleviated and much larger time steps can be taken. Prior to this thesis, there was the possibility in ANTARES to use IMEX methods with diffusive processes in compressible flows. We extended the potential use of these diffusive IMEX methods to incompressible flows and developed new methods for flows where the viscous time scale is the most stringent one.

For the new viscous methods, we haven chosen the IMEX SSP2$(2,2,2),\gamma = 0.24$ method for diffusive flows of F. Kupka et al., (2012) as our starting point and applied their formalism to the equations of fluid dynamics in the Boussinesq approximation. This resulted in two new IMEX methods for ANTARES:

- an IMEX SSP2$(2,2,2),\gamma = 0.24$ Runge–Kutta method for incompressible flows in the Boussinesq approximation, where the diffusive terms of the Navier–Stokes equations are integrated implicitly and the viscous terms are integrated explicitly (called DIFF_IMEX in this thesis)

- and an IMEX SSP2$(2,2,2),\gamma = 0.24$ Runge–Kutta method for incompressible flows in the Boussinesq approximation, where both the diffusive and viscous terms of the Navier–Stokes

equations are integrated implicitly (called VISC_IMEX in this thesis).

After having implemented these methods into the ANTARES framework we have run simulations for Prandtl numbers 0.01, 0.03, 0.1, 0.5, 1, 2, 3 and 7 with the explicit SSPRK(3,2) method and with the two new versions of the implicit-explicit SSP2$(2,2,2),\gamma = 0.24$ method to establish whether the time step restrictions can be alleviated effectively and how the new methods affect the wall clock times of the simulations for different regimes of the Prandtl number. The performance measures have been the acceleration (i.e., the ratio of wall clock times of the computations) and the size of time steps taken. As a result we came up with the following suggestions for the use of IMEX methods in the Boussinesq approximation:

- for Prandtl numbers $\lesssim 0.1$: use DIFF_IMEX. Since the time step is increased all the way up to the stability limit of the SSP2$(2,2,2),\gamma = 0.24$ method, there is no further increase in acceleration possible by using VISC_IMEX.

- for Prandtl numbers $0.1 \lesssim \mathrm{Pr} \lesssim 0.8$: use VISC_IMEX. In this parameter regime, the viscous time step limit is sufficiently low for VISC_IMEX to have a significant effect on the acceleration. The time step is augmented sufficiently to offset the increased computational cost of VISC_IMEX compared to DIFF_IMEX.

- for Prandtl numbers $\gtrsim 0.8$: use explicit SSPRK(3,2). While VISC_IMEX does show a small acceleration, it is slower than the explicit scheme once the simulation reaches the advective phase. Since simulations with ANTARES are primarily concerned with the advective phase, it can be assumed that the advective phase will cover a significant part of the overall simulation runtime. The very small acceleration that IMEX leads to in these parameters regime does not warrant a use of it in the overall simulation.

Of course, these results only hold for the parameter setting tested here. It could very well be the case that different values of $\mathrm{Le}, \mathrm{R}_\rho$ or Ra_T change the results that we have obtained. A thorough investigation of different parameter regimes can be the subject of further research.

Additionally, we have repeated a subset of the simulations with a different γ factor of the IMEX schemes (and thus an increased stability region) to figure out if the use of a more stable method of time integration enables larger time steps and improved wall clock times. These are the results:

- for low values of the Prandtl number, increasing γ leads to significantly larger time steps which led us to conclude that the time step actually taken when using IMEX SSP2(2,2,2) is limited by its stability region for low values of Pr. The wall clock time could not be reduced by a substantial amount, however, because too large time steps lead to unfavorable numerical properties for the multigrid solver.

- for high values of the Prandtl number, increasing γ yields only small improvements of the average time step actually taken. A possible explanation is that the method might have insufficient damping properties for large values of Pr. The wall clock time, however, could be reduced by a small amount when choosing a large value for γ.

Although most tests for the time integration routines have been performed in 2D, we have also made comparisons with the 3D case for which the solver developed in part I was used. Our tests indicate that the performance of the time integration methods is independent from this and that the solver developed in part I also has excellent scaling and performnace properties.

In summary, by implementing the SSP implicit-explicit Runge–Kutta method for the Boussinesq approximation we could accelerate the simulations of incompressible flows with ANTARES, especially in simulations where $\mathrm{Pr} \leq 1$. In simulations with $\mathrm{Pr} > 1$ the number two-point instabilities became too large and the time step was repeated too often to yield a significant acceleration. One idea to counter that is the development of three-stage IMEX methods, as e.g. IMEX SSP2(3,3,2) LPUM, which has proven to have superior properties to similar time integration schemes in Happenhofer, (2014) and F. Kupka et al., (2012). This can be the topic of future research.

Part III

Appendix

APPENDIX A

RESULTS OF MULTIGRID SOLVER TESTS

In this chapter, we present the exact numerical results of the simulations performed to verify the accuracy of the multigrid solver. The tables include the following parameters:

- levels: the number of grid levels in the multigrid algorithm,
- N_f: the number of points on the finest grid. Usually, we have the same number of grid points per direction so that $N_\mathrm{f} = 50^3$ means a resolution on the fine grid of $50 \times 50 \times 50$. The resolution is specified whenever there are a different number of points per direction.
- N_c: the number of points on the coarsest grid,
- h: the grid spacing in x-direction (usually the same as the grid spacing in y- and z-direction),
- ν_1: the number of pre-smoothing steps,
- ν_2: the number of post-smoothing steps,
- ϵ_fine: the stopping criterion for the multigrid solver,
- ϵ_coarse: the stopping criterion for the conjugate gradient solver on the coarsest grid.

The following entries are present in the tables:

- k: the iteration of the multigrid algorithm,
- $\|\boldsymbol{e}\|_2^{(k)}$: the 2-norm of the error at iteration k, i.e., the difference between the calculated solution at iteration k, $\boldsymbol{u}^k$, and the exact solution $\boldsymbol{u}_\mathrm{exact}$,
- $\|\boldsymbol{r}\|_2^{(k)}$: the 2-norm of the residual at iteration k where the residual is given by $\boldsymbol{r} = \boldsymbol{f} - A\boldsymbol{u}$ for the linear equations and by $\boldsymbol{r} = \boldsymbol{f} - A(\boldsymbol{u})$ for the nonlinear equations,

- $\|\boldsymbol{r}_c\|_2^{(k)}$: the 2-norm of the coarse grid residual at iteration k,
- the other entries are combinations of the errors or residuals at different iterations.

A.1 Linear Multigrid

A.1.1 Test Case 1 — Linear Equation with Homogeneous Dirichlet B.C.

These are the results which are discussed in section 4.1.
The problem considered is

$$-\nabla\cdot(x\nabla u(\boldsymbol{x}))+x\,u(\boldsymbol{x})=4x\sin(x)\sin(y)\sin(z)-\cos(x)\sin(y)\sin(z) \tag{A.1}$$

with the homogeneous Dirichlet boundary conditions

$$\begin{aligned} u(0,y,z)&=u(2\pi,y,z)=0,\\ u(x,0,z)&=u(x,2\pi,z),\\ u(x,y,0)&=u(x,y,2\pi). \end{aligned} \tag{A.2}$$

The exact solution is

$$u(x,y,z)=\sin(x)\sin(y)\sin(z). \tag{A.3}$$

Table A.1: Test case 1 — Results of section 4.1.2.1

	levels: 2, $N_c = 25$, $N_f = 50^3$, $h = 1.26 \cdot 10^{-1}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.592E-01				
1	1.520E-01	1.050E-01	1.505E-01	5.416E+00	0.0273	0.6595
2	4.790E-03	2.038E-03	1.545E-01	1.472E-01	0.0315	0.0194
3	5.436E-03	5.606E-05	4.376E-03	6.457E-04	1.1348	0.0275
4	5.317E-03	1.925E-06	1.160E-04	1.187E-04	0.9782	0.0343
5	5.316E-03	2.047E-07	4.364E-06	1.513E-06	0.9997	0.1063
6	5.315E-03	3.101E-08	3.707E-07	4.338E-07	0.9999	0.1515
7	5.315E-03	4.833E-09	5.290E-08	5.822E-08	1.0000	0.1559
8	5.315E-03	7.553E-10	8.133E-09	8.806E-09	1.0000	0.1563
9	5.315E-03	1.183E-10	1.266E-09	1.312E-09	1.0000	0.1566
10	5.315E-03	1.857E-11	1.975E-10	1.946E-10	1.0000	0.1570
11	5.315E-03	2.922E-12	3.086E-11	2.875E-11	1.0000	0.1574
convergence after 11 multigrid cycles mean convergence rate: 0.128 wall clock time: 7 sec						

Table A.2: Test case 1 — Results of section 4.1.2.1

	levels: 3, $N_c = 25$ $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.990E-02				
1	3.030E-01	5.527E-02	1.793E-02	5.265E+00	0.0544	2.7778
2	1.547E-01	3.957E-03	3.731E-02	1.483E-01	0.5105	0.0716
3	2.179E-02	1.681E-03	2.139E-02	1.329E-01	0.1408	0.4247
4	4.231E-03	2.324E-04	2.700E-03	1.756E-02	0.1942	0.1383
5	2.139E-03	5.968E-05	7.560E-04	2.092E-03	0.5056	0.2568
6	1.188E-03	1.044E-05	1.286E-04	9.509E-04	0.5554	0.1749
7	1.357E-03	2.292E-06	2.910E-05	1.688E-04	1.1421	0.2196
8	1.326E-03	4.367E-07	5.514E-06	3.125E-05	0.9770	0.1905
9	1.331E-03	9.016E-08	1.157E-06	5.720E-06	1.0043	0.2064
10	1.331E-03	1.776E-08	2.277E-07	9.224E-07	0.9993	0.1970
11	1.331E-03	3.588E-09	4.640E-08	1.640E-07	1.0001	0.2020
12	1.331E-03	7.123E-10	9.232E-09	2.832E-08	1.0000	0.1985
13	1.331E-03	1.424E-10	1.856E-09	3.850E-09	1.0000	0.2000
14	1.331E-03	2.831E-11	3.703E-10	6.074E-10	1.0000	0.1988
15	1.331E-03	5.687E-12	7.409E-11	1.206E-10	1.0000	0.2009
16	1.331E-03	1.128E-12	1.471E-11	1.009E-11	1.0000	0.1984
	convergence after 16 multigrid cycles					
	mean convergence rate: 0.234					
	wall clock time: 98 sec					

Table A.3: Test case 1 — Results of section 4.1.2.1

levels: 4, $N_c = 25$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.487E-03				
1	4.705E-01	2.854E-02	2.183E-03	5.098E+00	0.0845	11.4744
2	4.912E-01	4.375E-03	6.908E-03	2.078E-02	1.0442	0.1533
3	2.535E-01	3.591E-03	7.910E-03	2.377E-01	0.5161	0.8208
4	1.016E-01	1.693E-03	4.244E-03	1.520E-01	0.4006	0.4715
5	5.303E-02	7.390E-04	1.681E-03	4.853E-02	0.5222	0.4365
6	2.400E-02	3.731E-04	8.989E-04	2.903E-02	0.4525	0.5048
7	1.178E-02	1.736E-04	4.195E-04	1.222E-02	0.4908	0.4654
8	5.256E-03	8.247E-05	1.998E-04	6.521E-03	0.4463	0.4750
9	2.832E-03	3.925E-05	9.627E-05	2.424E-03	0.5388	0.4759
10	1.043E-03	1.853E-05	4.570E-05	1.790E-03	0.3681	0.4723
11	8.233E-04	8.779E-06	2.180E-05	2.193E-04	0.7897	0.4737
12	3.009E-04	4.151E-06	1.037E-05	5.224E-04	0.3654	0.4728
13	4.110E-04	1.961E-06	4.927E-06	1.101E-04	1.3661	0.4723
14	3.133E-04	9.256E-07	2.339E-06	9.767E-05	0.7623	0.4721
15	3.446E-04	4.365E-07	1.109E-06	3.132E-05	1.1000	0.4716
16	3.295E-04	2.057E-07	5.251E-07	1.511E-05	0.9561	0.4713
17	3.350E-04	9.690E-08	2.484E-07	5.455E-06	1.0166	0.4710
18	3.327E-04	4.561E-08	1.174E-07	2.242E-06	0.9933	0.4707
19	3.335E-04	2.145E-08	5.542E-08	8.136E-07	1.0024	0.4704
20	3.332E-04	1.009E-08	2.614E-08	3.029E-07	0.9991	0.4701
21	3.333E-04	4.740E-09	1.231E-08	1.049E-07	1.0003	0.4699
22	3.333E-04	2.227E-09	5.796E-09	3.533E-08	0.9999	0.4698
23	3.333E-04	1.046E-09	2.725E-09	1.094E-08	1.0000	0.4697
24	3.333E-04	4.914E-10	1.280E-09	3.185E-09	1.0000	0.4697
25	3.333E-04	2.309E-10	6.011E-10	8.663E-10	1.0000	0.4698
26	3.333E-04	1.085E-10	2.820E-10	2.429E-10	1.0000	0.4699
27	3.333E-04	5.103E-11	1.322E-10	9.544E-11	1.0000	0.4704
28	3.333E-04	2.405E-11	6.192E-11	6.720E-11	1.0000	0.4712
29	3.333E-04	1.136E-11	2.900E-11	5.286E-11	1.0000	0.4723
30	3.333E-04	5.379E-12	1.358E-11	3.894E-11	1.0000	0.4736
31	3.333E-04	2.543E-12	6.361E-12	2.622E-11	1.0000	0.4727

Continued on next page

Table A.3 – *Continued from previous page*

levels: 4, $N_c = 25$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$
$\epsilon_{\text{fine}} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
32	3.333E-04	1.185E-12	2.982E-12	1.804E-11	1.0000	0.4662
33	3.333E-04	6.233E-13	1.403E-12	4.654E-12	1.0000	0.5258
34	3.333E-04	6.989E-13	6.920E-13	4.822E-12	1.0000	1.1214

convergence after 34 multigrid cycles
mean convergence rate: 0.492
wall clock time: 1624 sec

Table A.4: Test case 1 — Results of section 4.1.2.1

levels: 2, $N_c = 51$, $N_f = 102^3$, $h = 6.16 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.875E-02				
1	1.687E-01	5.289E-02	1.850E-02	5.400E+00	0.0303	2.8210
2	4.567E-03	1.066E-03	1.907E-02	1.642E-01	0.0271	0.0202
3	1.482E-03	3.602E-05	6.229E-04	3.084E-03	0.3246	0.0338
4	1.283E-03	1.020E-06	1.817E-05	1.997E-04	0.8653	0.0283
5	1.280E-03	6.182E-08	8.077E-07	2.539E-06	0.9980	0.0606
6	1.279E-03	9.630E-09	6.089E-08	9.760E-07	0.9992	0.1558
7	1.279E-03	1.812E-09	9.752E-09	1.496E-07	0.9999	0.1882
8	1.279E-03	3.395E-10	1.784E-09	2.814E-08	1.0000	0.1873
9	1.279E-03	6.369E-11	3.336E-10	5.217E-09	1.0000	0.1876
10	1.279E-03	1.195E-11	6.255E-11	9.723E-10	1.0000	0.1876
11	1.279E-03	2.246E-12	1.173E-11	1.781E-10	1.0000	0.1879
12	1.279E-03	4.352E-13	2.204E-12	2.799E-11	1.0000	0.1938
convergence after 12 multigrid cycles mean convergence rate: 0.136 wall clock time: 101 sec						

Table A.5: Test case 1 — Results of section 4.1.2.1

levels: 3, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.344E-03				
1	3.394E-01	2.735E-02	2.286E-03	5.229E+00	0.0610	11.6701
2	1.925E-01	2.211E-03	4.783E-03	1.469E-01	0.5672	0.0808
3	2.839E-02	1.018E-03	2.797E-03	1.642E-01	0.1474	0.4606
4	7.846E-03	1.590E-04	4.069E-04	2.054E-02	0.2764	0.1562
5	1.845E-03	4.475E-05	1.226E-04	6.001E-03	0.2352	0.2814
6	1.972E-04	8.808E-06	2.364E-05	1.648E-03	0.1069	0.1968
7	3.948E-04	2.136E-06	5.890E-06	1.975E-04	2.0016	0.2425
8	3.038E-04	4.556E-07	1.259E-06	9.102E-05	0.7694	0.2133
9	3.237E-04	1.047E-07	2.942E-07	1.994E-05	1.0657	0.2298
10	3.196E-04	2.314E-08	6.521E-08	4.073E-06	0.9874	0.2211
11	3.206E-04	5.196E-09	1.487E-08	9.430E-07	1.0030	0.2245
12	3.204E-04	1.153E-09	3.334E-09	2.041E-07	0.9994	0.2219
13	3.204E-04	2.612E-10	7.555E-10	3.780E-08	1.0001	0.2265
14	3.204E-04	5.766E-11	1.696E-10	9.494E-09	1.0000	0.2207
15	3.204E-04	1.291E-11	3.837E-11	2.160E-09	1.0000	0.2239
16	3.204E-04	3.017E-12	8.650E-12	3.156E-10	1.0000	0.2337
17	3.204E-04	6.891E-13	1.939E-12	1.019E-10	1.0000	0.2284
18	3.204E-04	3.460E-13	4.764E-13	9.653E-12	1.0000	0.5020
19	3.204E-04	4.481E-13	2.321E-13	3.831E-11	1.0000	1.2952
convergence after 19 multigrid cycles mean convergence rate: 0.265 wall clock time: 1061 sec						

Table A.6: Test case 1 — Results of section 4.1.2.1

	levels: 2, $N_c = 99$, $N_f = 198^3$, $h = 3.17 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.563E-03				
1	1.734E-01	2.737E-02	2.554E-03	5.395E+00	0.0311	10.6753
2	5.133E-03	5.533E-04	2.634E-03	1.683E-01	0.0296	0.0202
3	5.466E-04	1.964E-05	9.235E-05	4.586E-03	0.1065	0.0355
4	3.440E-04	5.479E-07	2.682E-06	2.026E-04	0.6293	0.0279
5	3.413E-04	2.310E-08	1.424E-07	2.683E-06	0.9922	0.0422
6	3.403E-04	2.864E-09	9.433E-09	1.034E-06	0.9970	0.1240
7	3.401E-04	5.978E-10	1.471E-09	1.603E-07	0.9995	0.2088
8	3.401E-04	1.235E-10	2.896E-10	3.311E-08	0.9999	0.2065
9	3.401E-04	2.561E-11	5.972E-11	6.788E-09	1.0000	0.2074
10	3.401E-04	5.317E-12	1.237E-11	1.388E-09	1.0000	0.2076
11	3.401E-04	1.140E-12	2.567E-12	2.565E-10	1.0000	0.2144
12	3.401E-04	3.213E-13	5.478E-13	4.501E-11	1.0000	0.2819
13	3.401E-04	1.746E-13	1.727E-13	5.130E-11	1.0000	0.5433
	convergence after 13 multigrid cycles					
	mean convergence rate: 0.145					
	wall clock time: 1452 sec					

A.1.1.1 Second Series of Experiments — The Influence of the Number of Smoothing Steps

Table A.7: Test case 1 — Results of section 4.1.2.2

	levels: 2, $N_c = 25$, $N_f = 50^3$, $h = 1.26 \cdot 10^{-1}$, $\nu_1 = 2$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.592E-01				
1	1.782E-01	3.549E-01	1.505E-01	5.390E+00	0.0320	2.2295
2	8.762E-03	1.934E-02	1.546E-01	1.695E-01	0.0492	0.0545
3	5.621E-03	8.242E-04	4.525E-03	3.141E-03	0.6415	0.0426
4	5.338E-03	3.219E-05	1.241E-04	2.833E-04	0.9496	0.0391
5	5.319E-03	2.189E-06	1.055E-05	1.905E-05	0.9964	0.0680
6	5.316E-03	3.490E-07	1.408E-06	2.863E-06	0.9995	0.1594
7	5.315E-03	7.360E-08	2.353E-07	5.356E-07	0.9999	0.2109
8	5.315E-03	1.565E-08	4.565E-08	1.090E-07	1.0000	0.2127
9	5.315E-03	3.361E-09	9.480E-09	2.251E-08	1.0000	0.2147
10	5.315E-03	7.234E-10	2.016E-09	4.640E-09	1.0000	0.2152
11	5.315E-03	1.561E-10	4.319E-10	9.510E-10	1.0000	0.2157
12	5.315E-03	3.374E-11	9.286E-11	1.942E-10	1.0000	0.2162
13	5.315E-03	7.313E-12	2.001E-11	3.890E-11	1.0000	0.2167
convergence after 13 multigrid cycles mean convergence rate: 0.160 wall clock time: 6 sec						

Table A.8: Test case 1 — Results of section 4.1.2.2

levels: 3, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.990E-02				
1	3.240E-01	1.866E-01	1.793E-02	5.244E+00	0.0582	9.3776
2	1.621E-01	2.128E-02	3.735E-02	1.619E-01	0.5003	0.1140
3	2.466E-02	6.672E-03	2.150E-02	1.374E-01	0.1521	0.3136
4	4.812E-03	1.098E-03	2.809E-03	1.985E-02	0.1952	0.1645
5	2.257E-03	2.601E-04	7.982E-04	2.556E-03	0.4689	0.2370
6	1.189E-03	4.927E-05	1.398E-04	1.068E-03	0.5269	0.1894
7	1.363E-03	1.070E-05	3.222E-05	1.740E-04	1.1463	0.2172
8	1.325E-03	2.124E-06	6.268E-06	3.835E-05	0.9719	0.1985
9	1.331E-03	4.434E-07	1.346E-06	6.845E-06	1.0052	0.2088
10	1.330E-03	9.089E-08	2.721E-07	9.553E-07	0.9993	0.2050
11	1.331E-03	1.860E-08	5.664E-08	2.418E-07	1.0002	0.2047
12	1.331E-03	3.799E-09	1.160E-08	3.888E-08	1.0000	0.2042
13	1.331E-03	8.029E-10	2.405E-09	8.032E-10	1.0000	0.2114
14	1.331E-03	1.607E-10	4.874E-10	1.571E-09	1.0000	0.2001
15	1.331E-03	3.410E-11	1.033E-10	2.849E-10	1.0000	0.2122
16	1.331E-03	7.978E-12	2.148E-11	8.152E-11	1.0000	0.2340
17	1.331E-03	1.511E-12	4.190E-12	1.918E-11	1.0000	0.1894
convergence after 17 multigrid cycles mean convergence rate: 0.235 wall clock time: 75 sec						

Table A.9: Test case 1 — Results of section 4.1.2.2

levels: 4, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.487E-03				
1	4.898E-01	9.625E-02	2.183E-03	5.079E+00	0.0880	38.6980
2	5.126E-01	1.917E-02	6.919E-03	2.281E-02	1.0466	0.1992
3	2.726E-01	1.344E-02	7.954E-03	2.400E-01	0.5318	0.7008
4	1.129E-01	6.670E-03	4.327E-03	1.597E-01	0.4141	0.4964
5	5.961E-02	3.014E-03	1.772E-03	5.326E-02	0.5281	0.4519
6	2.784E-02	1.538E-03	9.678E-04	3.177E-02	0.4670	0.5103
7	1.405E-02	7.388E-04	4.636E-04	1.379E-02	0.5046	0.4804
8	6.474E-03	3.594E-04	2.265E-04	7.575E-03	0.4608	0.4864
9	3.479E-03	1.753E-04	1.120E-04	2.995E-03	0.5374	0.4877
10	1.393E-03	8.498E-05	5.459E-05	2.086E-03	0.4004	0.4848
11	9.992E-04	4.131E-05	2.674E-05	3.939E-04	0.7173	0.4861
12	3.520E-04	2.005E-05	1.307E-05	6.473E-04	0.3522	0.4855
13	4.463E-04	9.726E-06	6.380E-06	9.438E-05	1.2681	0.4850
14	3.105E-04	4.717E-06	3.113E-06	1.358E-04	0.6957	0.4850
15	3.498E-04	2.287E-06	1.517E-06	3.927E-05	1.1265	0.4848
16	3.281E-04	1.108E-06	7.391E-07	2.165E-05	0.9381	0.4845
17	3.358E-04	5.366E-07	3.597E-07	7.621E-06	1.0232	0.4843
18	3.325E-04	2.598E-07	1.749E-07	3.249E-06	0.9903	0.4841
19	3.337E-04	1.257E-07	8.498E-08	1.148E-06	1.0035	0.4840
20	3.332E-04	6.084E-08	4.126E-08	4.507E-07	0.9986	0.4839
21	3.334E-04	2.943E-08	2.001E-08	1.566E-07	1.0005	0.4838
22	3.333E-04	1.424E-08	9.699E-09	5.045E-08	0.9998	0.4837
23	3.333E-04	6.887E-09	4.697E-09	1.526E-08	1.0000	0.4837
24	3.333E-04	3.332E-09	2.273E-09	4.917E-09	1.0000	0.4838
25	3.333E-04	1.612E-09	1.099E-09	1.278E-09	1.0000	0.4840
26	3.333E-04	7.808E-10	5.307E-10	2.958E-10	1.0000	0.4842
27	3.333E-04	3.782E-10	2.562E-10	1.967E-10	1.0000	0.4844
28	3.333E-04	1.834E-10	1.236E-10	1.736E-10	1.0000	0.4848
29	3.333E-04	8.919E-11	5.957E-11	1.248E-10	1.0000	0.4864
30	3.333E-04	4.351E-11	2.871E-11	9.075E-11	1.0000	0.4878
31	3.333E-04	2.134E-11	1.384E-11	6.396E-11	1.0000	0.4905

Continued on next page

Table A.9 – *Continued from previous page*

levels: 4, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
32	3.333E-04	1.047E-11	6.677E-12	3.917E-11	1.0000	0.4906
33	3.333E-04	5.030E-12	3.228E-12	2.379E-11	1.0000	0.4804
34	3.333E-04	2.482E-12	1.565E-12	1.012E-11	1.0000	0.4934
35	3.333E-04	2.230E-12	7.790E-13	6.646E-12	1.0000	0.8984
36	3.333E-04	1.994E-12	4.558E-13	1.690E-11	1.0000	0.8942
37	3.333E-04	2.260E-12	3.581E-13	2.310E-11	1.0000	1.1333
convergence after 37 multigrid cycles mean convergence rate: 0.519 wall clock time: 1356 sec						

Table A.10: Test case 1 — Results of section 4.1.2.2

	levels: 2, $N_c = 51^3$, $N_f = 102^3$, $h = 6.16 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.875E-02				
1	1.921E-01	1.783E-01	1.850E-02	5.376E+00	0.0345	9.5117
2	6.649E-03	1.052E-02	1.907E-02	1.854E-01	0.0346	0.0590
3	1.664E-03	4.840E-04	6.436E-04	4.985E-03	0.2503	0.0460
4	1.309E-03	1.996E-05	2.061E-05	3.555E-04	0.7864	0.0412
5	1.284E-03	9.093E-07	2.057E-06	2.471E-05	0.9811	0.0456
6	1.280E-03	8.851E-08	2.538E-07	3.832E-06	0.9970	0.0973
7	1.279E-03	2.099E-08	3.963E-08	7.510E-07	0.9994	0.2371
8	1.279E-03	5.063E-09	7.843E-09	1.731E-07	0.9999	0.2412
9	1.279E-03	1.239E-09	1.796E-09	4.152E-08	1.0000	0.2447
10	1.279E-03	3.031E-10	4.322E-10	1.006E-08	1.0000	0.2446
11	1.279E-03	7.418E-11	1.053E-10	2.445E-09	1.0000	0.2448
12	1.279E-03	1.816E-11	2.575E-11	5.908E-10	1.0000	0.2449
13	1.279E-03	4.457E-12	6.303E-12	1.403E-10	1.0000	0.2454
14	1.279E-03	1.136E-12	1.547E-12	2.941E-11	1.0000	0.2549
	convergence after 14 multigrid cycles					
	mean convergence rate: 0.166					
	wall clock time: 99 sec					

Table A.11: Test case 1 — Results of section 4.1.2.2

levels: 3, $N_c = 51$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.344E-03				
1	3.542E-01	9.213E-02	2.286E-03	5.214E+00	0.0636	39.3075
2	1.947E-01	1.147E-02	4.784E-03	1.595E-01	0.5498	0.1245
3	2.955E-02	3.886E-03	2.801E-03	1.652E-01	0.1517	0.3388
4	8.036E-03	7.028E-04	4.139E-04	2.151E-02	0.2719	0.1809
5	1.903E-03	1.810E-04	1.249E-04	6.133E-03	0.2368	0.2576
6	2.677E-04	3.772E-05	2.430E-05	1.635E-03	0.1407	0.2083
7	4.014E-04	8.913E-06	6.074E-06	1.337E-04	1.4992	0.2363
8	3.031E-04	1.944E-06	1.309E-06	9.821E-05	0.7553	0.2181
9	3.236E-04	4.452E-07	3.077E-07	2.044E-05	1.0674	0.2290
10	3.195E-04	9.979E-08	6.875E-08	4.129E-06	0.9872	0.2241
11	3.206E-04	2.234E-08	1.576E-08	1.117E-06	1.0035	0.2239
12	3.204E-04	5.026E-09	3.555E-09	1.984E-07	0.9994	0.2250
13	3.204E-04	1.157E-09	8.205E-10	3.263E-08	1.0001	0.2301
14	3.204E-04	2.567E-10	1.854E-10	1.508E-08	1.0000	0.2220
15	3.204E-04	5.743E-11	4.178E-11	1.561E-09	1.0000	0.2237
16	3.204E-04	1.444E-11	1.034E-11	5.358E-11	1.0000	0.2515
17	3.204E-04	3.857E-12	2.389E-12	2.777E-10	1.0000	0.2671
18	3.204E-04	1.616E-12	5.238E-13	4.823E-11	1.0000	0.4190
19	3.204E-04	1.122E-12	2.911E-13	1.915E-11	1.0000	0.6940
20	3.204E-04	1.249E-12	2.415E-13	5.556E-11	1.0000	1.1139
convergence after 20 multigrid cycles mean convergence rate: 0.278 wall clock time: 849 sec						

Table A.12: Test case 1 — Results of section 4.1.2.2

levels: 2, $N_c = 99^3$, $N_f = 198^3$, $h = 3.17 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.563E-03				
1	1.960E-01	9.211E-02	2.554E-03	5.372E+00	0.0352	35.9337
2	6.441E-03	5.534E-03	2.634E-03	1.895E-01	0.0329	0.0601
3	6.788E-04	2.596E-04	9.771E-05	5.762E-03	0.1054	0.0469
4	3.692E-04	1.097E-05	3.566E-06	3.097E-04	0.5438	0.0422
5	3.446E-04	4.609E-07	4.076E-07	2.460E-05	0.9334	0.0420
6	3.410E-04	2.697E-08	4.820E-08	3.602E-06	0.9895	0.0585
7	3.403E-04	6.088E-09	6.741E-09	6.684E-07	0.9980	0.2258
8	3.401E-04	1.563E-09	1.209E-09	1.580E-07	0.9995	0.2567
9	3.401E-04	4.117E-10	2.774E-10	4.024E-08	0.9999	0.2634
10	3.401E-04	1.083E-10	7.053E-11	1.048E-08	1.0000	0.2629
11	3.401E-04	2.849E-11	1.842E-11	2.734E-09	1.0000	0.2632
12	3.401E-04	7.540E-12	4.838E-12	6.988E-10	1.0000	0.2646
13	3.401E-04	2.102E-12	1.279E-12	1.676E-10	1.0000	0.2788
14	3.401E-04	8.842E-13	3.576E-13	4.008E-11	1.0000	0.4206
15	3.401E-04	5.251E-13	1.615E-13	3.656E-11	1.0000	0.5938
16	3.401E-04	4.608E-13	1.441E-13	4.255E-12	1.0000	0.8777
17	3.401E-04	3.804E-13	1.385E-13	1.929E-12	1.0000	0.8256
18	3.401E-04	3.797E-13	1.410E-13	6.689E-12	1.0000	0.9981
convergence after 18 multigrid cycles mean convergence rate: 0.232 wall clock time: 1567 sec						

Table A.13: Test case 1 — Results of section 4.1.2.2

levels: 2, $N_C = 25$, $N_\mathrm{f} = 50^3$, $h = 1.26 \cdot 10^{-1}$, $\nu_1 = 1$, $\nu_2 = 2$ $\epsilon_\mathrm{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_\mathrm{coarse} = \max(10^{-12}\|\boldsymbol{r}_\mathrm{c}\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_\mathrm{c}\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.592E-01				
1	1.554E-01	1.072E-01	1.538E-01	5.413E+00	0.0279	0.6736
2	4.905E-03	6.362E-03	1.580E-01	1.505E-01	0.0316	0.0593
3	5.428E-03	2.769E-04	4.640E-03	5.229E-04	1.1066	0.0435
4	5.321E-03	1.128E-05	1.298E-04	1.073E-04	0.9802	0.0407
5	5.316E-03	7.396E-07	1.083E-05	4.377E-06	0.9992	0.0656
6	5.315E-03	1.026E-07	1.279E-06	8.418E-07	0.9998	0.1387
7	5.315E-03	1.986E-08	1.780E-07	1.601E-07	1.0000	0.1936
8	5.315E-03	4.109E-09	2.990E-08	3.277E-08	1.0000	0.2069
9	5.315E-03	8.747E-10	5.811E-09	6.753E-09	1.0000	0.2129
10	5.315E-03	1.879E-10	1.208E-09	1.387E-09	1.0000	0.2148
11	5.315E-03	4.055E-11	2.570E-10	2.832E-10	1.0000	0.2158
12	5.315E-03	8.777E-12	5.515E-11	5.773E-11	1.0000	0.2164
convergence after 12 multigrid cycles mean convergence rate: 0.152 wall clock time: 6 sec						

Table A.14: Test case 1 — Results of section 4.1.2.2

	levels: 3, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.990E-02				
1	3.112E-01	5.562E-02	1.874E-02	5.257E+00	0.0559	2.7952
2	1.622E-01	6.471E-03	3.904E-02	1.491E-01	0.5210	0.1164
3	2.307E-02	2.020E-03	2.247E-02	1.391E-01	0.1423	0.3121
4	4.718E-03	3.351E-04	2.934E-03	1.835E-02	0.2045	0.1659
5	2.234E-03	7.931E-05	8.329E-04	2.484E-03	0.4735	0.2367
6	1.171E-03	1.510E-05	1.456E-04	1.063E-03	0.5240	0.1904
7	1.361E-03	3.274E-06	3.354E-05	1.907E-04	1.1629	0.2168
8	1.325E-03	6.542E-07	6.517E-06	3.659E-05	0.9731	0.1998
9	1.332E-03	1.368E-07	1.398E-06	6.701E-06	1.0051	0.2091
10	1.330E-03	2.793E-08	2.817E-07	1.116E-06	0.9992	0.2042
11	1.331E-03	5.758E-09	5.860E-08	2.085E-07	1.0002	0.2061
12	1.331E-03	1.176E-09	1.193E-08	3.463E-08	1.0000	0.2042
13	1.331E-03	2.422E-10	2.459E-09	4.532E-09	1.0000	0.2060
14	1.331E-03	4.933E-11	5.006E-10	9.461E-10	1.0000	0.2036
15	1.331E-03	1.020E-11	1.033E-10	1.592E-10	1.0000	0.2069
16	1.331E-03	2.108E-12	2.099E-11	5.143E-13	1.0000	0.2065
17	1.331E-03	4.405E-13	4.260E-12	6.725E-12	1.0000	0.2090
	convergence after 17 multigrid cycles mean convergence rate: 0.233 wall clock time: 74 sec					

Table A.15: Test case 1 — Results of section 4.1.2.2

levels: 4, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.487E-03				
1	4.822E-01	2.862E-02	2.310E-03	5.086E+00	0.0866	11.5072
2	5.143E-01	5.785E-03	7.324E-03	3.206E-02	1.0665	0.2021
3	2.723E-01	4.017E-03	8.420E-03	2.420E-01	0.5294	0.6943
4	1.117E-01	1.998E-03	4.580E-03	1.606E-01	0.4101	0.4975
5	5.960E-02	9.032E-04	1.875E-03	5.206E-02	0.5338	0.4520
6	2.782E-02	4.606E-04	1.023E-03	3.179E-02	0.4667	0.5100
7	1.392E-02	2.213E-04	4.894E-04	1.389E-02	0.5006	0.4804
8	6.433E-03	1.076E-04	2.390E-04	7.490E-03	0.4620	0.4861
9	3.460E-03	5.245E-05	1.180E-04	2.973E-03	0.5379	0.4876
10	1.379E-03	2.542E-05	5.744E-05	2.081E-03	0.3986	0.4847
11	9.927E-04	1.235E-05	2.809E-05	3.867E-04	0.7197	0.4857
12	3.481E-04	5.988E-06	1.371E-05	6.447E-04	0.3506	0.4850
13	4.450E-04	2.901E-06	6.678E-06	9.695E-05	1.2785	0.4845
14	3.096E-04	1.405E-06	3.251E-06	1.354E-04	0.6957	0.4843
15	3.499E-04	6.801E-07	1.581E-06	4.027E-05	1.1301	0.4840
16	3.281E-04	3.289E-07	7.679E-07	2.176E-05	0.9378	0.4837
17	3.358E-04	1.590E-07	3.726E-07	7.656E-06	1.0233	0.4833
18	3.325E-04	7.680E-08	1.806E-07	3.302E-06	0.9902	0.4830
19	3.337E-04	3.708E-08	8.748E-08	1.193E-06	1.0036	0.4828
20	3.332E-04	1.789E-08	4.232E-08	4.599E-07	0.9986	0.4825
21	3.334E-04	8.629E-09	2.046E-08	1.601E-07	1.0005	0.4823
22	3.333E-04	4.161E-09	9.878E-09	5.450E-08	0.9998	0.4822
23	3.333E-04	2.006E-09	4.765E-09	1.692E-08	1.0001	0.4821
24	3.333E-04	9.672E-10	2.297E-09	5.046E-09	1.0000	0.4821
25	3.333E-04	4.664E-10	1.106E-09	1.366E-09	1.0000	0.4822
26	3.333E-04	2.250E-10	5.322E-10	3.845E-10	1.0000	0.4824
27	3.333E-04	1.086E-10	2.559E-10	1.747E-10	1.0000	0.4826
28	3.333E-04	5.244E-11	1.229E-10	1.317E-10	1.0000	0.4830
29	3.333E-04	2.542E-11	5.904E-11	1.057E-10	1.0000	0.4848
30	3.333E-04	1.236E-11	2.834E-11	7.871E-11	1.0000	0.4862
31	3.333E-04	6.040E-12	1.361E-11	5.378E-11	1.0000	0.4887

Continued on next page

Table A.15 – *Continued from previous page*

levels: 4, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 2$
$\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
32	3.333E-04	2.954E-12	6.545E-12	3.332E-11	1.0000	0.4891
33	3.333E-04	1.402E-12	3.152E-12	2.166E-11	1.0000	0.4746
34	3.333E-04	6.764E-13	1.524E-12	9.816E-12	1.0000	0.4825
35	3.333E-04	6.255E-13	7.512E-13	3.582E-12	1.0000	0.9247

convergence after 35 multigrid cycles
mean convergence rate: 0.500
wall clock time: 1282 sec

Table A.16: Test case 1 — Results of section 4.1.2.2

levels: 2, $N_c = 51^3$, $N_f = 102^3$, $h = 6.16 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.875E-02				
1	1.701E-01	5.316E-02	1.860E-02	5.398E+00	0.0305	2.8354
2	4.089E-03	3.415E-03	1.917E-02	1.660E-01	0.0240	0.0642
3	1.461E-03	1.602E-04	6.485E-04	2.628E-03	0.3573	0.0469
4	1.289E-03	6.680E-06	2.096E-05	1.726E-04	0.8819	0.0417
5	1.281E-03	2.939E-07	2.080E-06	7.357E-06	0.9943	0.0440
6	1.279E-03	2.249E-08	2.419E-07	1.665E-06	0.9987	0.0765
7	1.279E-03	4.627E-09	3.261E-08	3.482E-07	0.9997	0.2058
8	1.279E-03	1.079E-09	5.383E-09	8.251E-08	0.9999	0.2333
9	1.279E-03	2.622E-10	1.117E-09	1.991E-08	1.0000	0.2430
10	1.279E-03	6.403E-11	2.604E-10	4.834E-09	1.0000	0.2442
11	1.279E-03	1.567E-11	6.295E-11	1.174E-09	1.0000	0.2447
12	1.279E-03	3.838E-12	1.536E-11	2.820E-10	1.0000	0.2449
13	1.279E-03	9.470E-13	3.759E-12	6.407E-11	1.0000	0.2468
convergence after 13 multigrid cycles mean convergence rate: 0.154 wall clock time: 92 sec						

Table A.17: Test case 1 — Results of section 4.1.2.2

levels: 3, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.344E-03				
1	3.424E-01	2.740E-02	2.310E-03	5.226E+00	0.0615	11.6882
2	1.946E-01	3.456E-03	4.835E-03	1.478E-01	0.5684	0.1261
3	2.873E-02	1.171E-03	2.831E-03	1.659E-01	0.1477	0.3388
4	8.011E-03	2.129E-04	4.182E-04	2.072E-02	0.2788	0.1819
5	1.892E-03	5.474E-05	1.261E-04	6.120E-03	0.2361	0.2571
6	2.021E-04	1.143E-05	2.447E-05	1.689E-03	0.1068	0.2088
7	3.981E-04	2.696E-06	6.120E-06	1.960E-04	1.9702	0.2359
8	3.032E-04	5.899E-07	1.319E-06	9.487E-05	0.7617	0.2188
9	3.239E-04	1.349E-07	3.098E-07	2.061E-05	1.0680	0.2287
10	3.196E-04	3.017E-08	6.906E-08	4.302E-06	0.9867	0.2237
11	3.206E-04	6.787E-09	1.584E-08	1.021E-06	1.0032	0.2250
12	3.204E-04	1.523E-09	3.571E-09	2.089E-07	0.9993	0.2244
13	3.204E-04	3.466E-10	8.170E-10	3.973E-08	1.0001	0.2276
14	3.204E-04	7.690E-11	1.843E-10	1.144E-08	1.0000	0.2219
15	3.204E-04	1.733E-11	4.182E-11	2.047E-09	1.0000	0.2254
16	3.204E-04	4.107E-12	9.694E-12	2.822E-10	1.0000	0.2370
17	3.204E-04	9.559E-13	2.179E-12	1.723E-10	1.0000	0.2328
18	3.204E-04	4.423E-13	5.157E-13	9.899E-12	1.0000	0.4627
19	3.204E-04	3.971E-13	2.662E-13	5.796E-11	1.0000	0.8979
20	3.204E-04	4.331E-13	2.305E-13	6.538E-11	1.0000	1.0906
convergence after 20 multigrid cycles mean convergence rate: 0.280 wall clock time: 829 sec						

Table A.18: Test case 1 — Results of section 4.1.2.2

levels: 2, $N_c = 99^3$, $N_f = 198^3$, $h = 3.17 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.563E-03				
1	1.739E-01	2.740E-02	2.558E-03	5.394E+00	0.0312	10.6899
2	4.554E-03	1.792E-03	2.638E-03	1.694E-01	0.0262	0.0654
3	5.198E-04	8.575E-05	9.801E-05	4.034E-03	0.1141	0.0478
4	3.500E-04	3.649E-06	3.597E-06	1.698E-04	0.6733	0.0426
5	3.422E-04	1.508E-07	4.102E-07	7.780E-06	0.9778	0.0413
6	3.405E-04	7.007E-09	4.738E-08	1.671E-06	0.9951	0.0465
7	3.402E-04	1.182E-09	6.072E-09	3.432E-07	0.9990	0.1687
8	3.401E-04	2.901E-10	9.075E-10	8.529E-08	0.9997	0.2453
9	3.401E-04	7.619E-11	1.795E-10	2.203E-08	0.9999	0.2627
10	3.401E-04	2.001E-11	4.326E-11	5.758E-09	1.0000	0.2626
11	3.401E-04	5.269E-12	1.115E-11	1.494E-09	1.0000	0.2634
12	3.401E-04	1.413E-12	2.923E-12	3.677E-10	1.0000	0.2681
13	3.401E-04	4.475E-13	7.801E-13	7.996E-11	1.0000	0.3167
14	3.401E-04	2.283E-13	2.479E-13	6.107E-11	1.0000	0.5102
convergence after 14 multigrid cycles mean convergence rate: 0.158 wall clock time: 1378 sec						

Table A.19: Test case 1 — Results of section 4.1.2.2

	levels: 2, $N_c = 25^3$, $N_f = 50^3$, $h = 1.26 \cdot 10^{-1}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.592E-01				
1	1.813E-01	3.624E-01	1.538E-01	5.387E+00	0.0326	2.2770
2	1.097E-02	2.657E-02	1.580E-01	1.703E-01	0.0605	0.0733
3	5.889E-03	2.836E-03	4.622E-03	5.086E-03	0.5366	0.1067
4	5.394E-03	3.142E-04	2.346E-04	4.954E-04	0.9159	0.1108
5	5.330E-03	3.832E-05	4.997E-05	6.316E-05	0.9883	0.1220
6	5.319E-03	6.040E-06	1.156E-05	1.143E-05	0.9979	0.1576
7	5.316E-03	1.295E-06	2.747E-06	2.737E-06	0.9995	0.2144
8	5.316E-03	3.263E-07	6.671E-07	7.772E-07	0.9999	0.2520
9	5.315E-03	8.829E-08	1.657E-07	2.375E-07	1.0000	0.2705
10	5.315E-03	2.506E-08	4.234E-08	7.439E-08	1.0000	0.2839
11	5.315E-03	7.411E-09	1.122E-08	2.344E-08	1.0000	0.2957
12	5.315E-03	2.265E-09	3.103E-09	7.383E-09	1.0000	0.3057
13	5.315E-03	7.100E-10	8.962E-10	2.317E-09	1.0000	0.3134
14	5.315E-03	2.264E-10	2.691E-10	7.246E-10	1.0000	0.3189
15	5.315E-03	7.308E-11	8.329E-11	2.257E-10	1.0000	0.3228
16	5.315E-03	2.378E-11	2.634E-11	6.956E-11	1.0000	0.3254
17	5.315E-03	7.785E-12	8.449E-12	2.112E-11	1.0000	0.3273
	convergence after 17 multigrid cycles					
	mean convergence rate: 0.249					
	wall clock time: 6 sec					

Table A.20: Test case 1 — Results of section 4.1.2.2

levels: 3, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.990E-02				
1	3.317E-01	1.878E-01	1.874E-02	5.237E+00	0.0596	9.4368
2	1.650E-01	1.028E-02	3.907E-02	1.666E-01	0.4976	0.0547
3	2.347E-02	6.426E-03	2.254E-02	1.415E-01	0.1422	0.6252
4	4.527E-03	5.256E-04	3.032E-03	1.894E-02	0.1929	0.0818
5	2.153E-03	2.015E-04	8.523E-04	2.374E-03	0.4756	0.3835
6	1.213E-03	2.650E-05	1.473E-04	9.398E-04	0.5635	0.1315
7	1.359E-03	7.068E-06	3.373E-05	1.455E-04	1.1199	0.2667
8	1.326E-03	1.206E-06	6.424E-06	3.297E-05	0.9757	0.1707
9	1.331E-03	2.741E-07	1.365E-06	4.981E-06	1.0038	0.2272
10	1.330E-03	5.625E-08	2.677E-07	5.409E-07	0.9996	0.2052
11	1.331E-03	1.432E-08	5.454E-08	2.805E-07	1.0002	0.2546
12	1.331E-03	2.873E-09	1.075E-08	2.180E-09	1.0000	0.2007
13	1.331E-03	8.365E-10	2.476E-09	5.683E-09	1.0000	0.2911
14	1.331E-03	3.456E-10	5.121E-10	5.155E-09	1.0000	0.4132
15	1.331E-03	6.588E-11	9.487E-11	4.550E-10	1.0000	0.1906
16	1.331E-03	2.528E-11	4.768E-11	3.891E-10	1.0000	0.3837
17	1.331E-03	1.341E-11	1.230E-11	1.998E-10	1.0000	0.5307
18	1.331E-03	2.808E-12	1.909E-12	3.287E-11	1.0000	0.2094
19	1.331E-03	9.144E-13	1.781E-12	1.450E-11	1.0000	0.3256
convergence after 19 multigrid cycles mean convergence rate: 0.274 wall clock time: 56 sec						

Table A.21: Test case 1 — Results of section 4.1.2.2

levels: 4, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.487E-03				
1	5.012E-01	9.654E-02	2.310E-03	5.067E+00	0.0900	38.8116
2	5.258E-01	9.884E-03	7.332E-03	2.455E-02	1.0490	0.1024
3	2.769E-01	1.259E-02	8.451E-03	2.489E-01	0.5266	1.2733
4	1.098E-01	5.303E-03	4.631E-03	1.671E-01	0.3966	0.4214
5	5.794E-02	2.366E-03	1.920E-03	5.187E-02	0.5277	0.4461
6	2.622E-02	1.229E-03	1.041E-03	3.172E-02	0.4525	0.5196
7	1.308E-02	5.647E-04	4.931E-04	1.314E-02	0.4987	0.4594
8	5.804E-03	2.717E-04	2.381E-04	7.271E-03	0.4439	0.4811
9	3.076E-03	1.294E-04	1.161E-04	2.728E-03	0.5299	0.4764
10	1.201E-03	6.115E-05	5.573E-05	1.875E-03	0.3906	0.4724
11	8.724E-04	2.903E-05	2.687E-05	3.291E-04	0.7261	0.4748
12	3.237E-04	1.369E-05	1.289E-05	5.486E-04	0.3711	0.4717
13	4.139E-04	6.449E-06	6.153E-06	9.014E-05	1.2784	0.4709
14	3.176E-04	3.036E-06	2.932E-06	9.625E-05	0.7675	0.4708
15	3.445E-04	1.426E-06	1.394E-06	2.687E-05	1.0846	0.4697
16	3.298E-04	6.682E-07	6.616E-07	1.476E-05	0.9572	0.4685
17	3.346E-04	3.126E-07	3.132E-07	4.836E-06	1.0147	0.4678
18	3.329E-04	1.462E-07	1.478E-07	1.704E-06	0.9949	0.4675
19	3.335E-04	6.821E-08	6.959E-08	6.593E-07	1.0020	0.4667
20	3.333E-04	3.182E-08	3.269E-08	2.764E-07	0.9992	0.4665
21	3.333E-04	1.483E-08	1.533E-08	5.261E-08	1.0002	0.4659
22	3.333E-04	6.909E-09	7.178E-09	1.089E-08	1.0000	0.4660
23	3.333E-04	3.222E-09	3.355E-09	1.245E-08	1.0000	0.4663
24	3.333E-04	1.501E-09	1.566E-09	2.207E-09	1.0000	0.4659
25	3.333E-04	7.017E-10	7.296E-10	1.938E-09	1.0000	0.4675
26	3.333E-04	3.271E-10	3.397E-10	3.912E-10	1.0000	0.4661
27	3.333E-04	1.536E-10	1.582E-10	6.744E-10	1.0000	0.4695
28	3.333E-04	7.269E-11	7.367E-11	8.825E-11	1.0000	0.4734
29	3.333E-04	3.439E-11	3.438E-11	1.778E-11	1.0000	0.4731
30	3.333E-04	1.646E-11	1.608E-11	1.359E-10	1.0000	0.4786
31	3.333E-04	8.043E-12	7.567E-12	3.865E-11	1.0000	0.4887

Continued on next page

Table A.21 – *Continued from previous page*

levels: 4, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
32	3.333E-04	3.797E-12	3.589E-12	2.182E-13	1.0000	0.4720
33	3.333E-04	2.068E-12	1.715E-12	3.402E-11	1.0000	0.5446
34	3.333E-04	1.929E-12	8.601E-13	4.367E-13	1.0000	0.9329
convergence after 34 multigrid cycles mean convergence rate: 0.498 wall clock time: 846 sec						

Table A.22: Test case 1 — Results of section 4.1.2.2

levels: 2, $N_c = 51^3$, $N_f = 102^3$, $h = 6.16 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$
$\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.875E-02				
1	1.933E-01	1.793E-01	1.860E-02	5.375E+00	0.0347	9.5605
2	6.764E-03	1.252E-02	1.916E-02	1.866E-01	0.0350	0.0698
3	1.912E-03	1.371E-03	6.453E-04	4.853E-03	0.2826	0.1095
4	1.375E-03	1.504E-04	4.292E-05	5.372E-04	0.7190	0.1097
5	1.298E-03	1.695E-05	9.470E-06	7.672E-05	0.9442	0.1127
6	1.284E-03	2.078E-06	2.231E-06	1.404E-05	0.9892	0.1226
7	1.280E-03	3.291E-07	5.396E-07	3.322E-06	0.9974	0.1584
8	1.280E-03	7.369E-08	1.327E-07	9.659E-07	0.9992	0.2239
9	1.279E-03	2.018E-08	3.318E-08	3.139E-07	0.9998	0.2739
10	1.279E-03	6.077E-09	8.480E-09	1.069E-07	0.9999	0.3011
11	1.279E-03	1.942E-09	2.239E-09	3.707E-08	1.0000	0.3196
12	1.279E-03	6.461E-10	6.192E-10	1.293E-08	1.0000	0.3327
13	1.279E-03	2.204E-10	1.814E-10	4.516E-09	1.0000	0.3412
14	1.279E-03	7.631E-11	5.631E-11	1.577E-09	1.0000	0.3462
15	1.279E-03	2.663E-11	1.834E-11	5.490E-10	1.0000	0.3490
16	1.279E-03	9.344E-12	6.180E-12	1.894E-10	1.0000	0.3508
17	1.279E-03	3.302E-12	2.127E-12	6.346E-11	1.0000	0.3533
18	1.279E-03	1.189E-12	7.513E-13	2.152E-11	1.0000	0.3600

convergence after 18 multigrid cycles

mean convergence rate: 0.252

wall clock time: 101 sec

Table A.23: Test case 1 — Results of section 4.1.2.2

levels: 3, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.344E-03				
1	3.570E-01	9.227E-02	2.310E-03	5.211E+00	0.0641	39.3691
2	1.918E-01	4.803E-03	4.835E-03	1.652E-01	0.5373	0.0520
3	2.729E-02	3.585E-03	2.831E-03	1.645E-01	0.1423	0.7464
4	7.181E-03	2.996E-04	4.281E-04	2.011E-02	0.2632	0.0836
5	1.673E-03	1.313E-04	1.251E-04	5.508E-03	0.2330	0.4383
6	2.317E-04	1.832E-05	2.326E-05	1.442E-03	0.1384	0.1395
7	3.864E-04	5.387E-06	5.813E-06	1.547E-04	1.6679	0.2940
8	3.081E-04	9.600E-07	1.209E-06	7.828E-05	0.7974	0.1782
9	3.224E-04	2.428E-07	2.774E-07	1.435E-05	1.0466	0.2529
10	3.196E-04	4.829E-08	5.945E-08	2.889E-06	0.9910	0.1989
11	3.205E-04	1.166E-08	1.347E-08	8.965E-07	1.0028	0.2415
12	3.204E-04	2.713E-09	2.884E-09	6.145E-08	0.9998	0.2326
13	3.204E-04	5.746E-10	6.758E-10	2.267E-08	1.0001	0.2118
14	3.204E-04	2.004E-10	1.651E-10	1.750E-08	0.9999	0.3488
15	3.204E-04	7.621E-11	3.461E-11	3.105E-09	1.0000	0.3802
16	3.204E-04	1.002E-11	8.990E-12	3.124E-10	1.0000	0.1315
17	3.204E-04	7.713E-12	4.923E-12	6.183E-10	1.0000	0.7699
18	3.204E-04	4.200E-12	1.227E-12	2.031E-10	1.0000	0.5445
19	3.204E-04	1.614E-12	2.817E-13	1.667E-10	1.0000	0.3844
20	3.204E-04	1.099E-12	3.201E-13	9.436E-11	1.0000	0.6807
21	3.204E-04	1.290E-12	2.292E-13	9.703E-11	1.0000	1.1734
convergence after 21 multigrid cycles mean convergence rate: 0.311 wall clock time: 615 sec						

Table A.24: Test case 1 — Results of section 4.1.2.2

levels: 2, $N_c = 99^3$, $N_f = 198^3$, $h = 3.17 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.563E-03				
1	1.965E-01	9.224E-02	2.558E-03	5.372E+00	0.0353	35.9826
2	5.649E-03	6.384E-03	2.637E-03	1.908E-01	0.0288	0.0692
3	9.273E-04	7.057E-04	9.960E-05	4.722E-03	0.1641	0.1105
4	4.326E-04	7.744E-05	8.461E-06	4.947E-04	0.4665	0.1097
5	3.579E-04	8.606E-06	1.877E-06	7.467E-05	0.8274	0.1111
6	3.443E-04	9.751E-07	4.445E-07	1.360E-05	0.9620	0.1133
7	3.413E-04	1.207E-07	1.079E-07	3.013E-06	0.9913	0.1237
8	3.405E-04	2.006E-08	2.657E-08	8.316E-07	0.9976	0.1662
9	3.402E-04	5.050E-09	6.618E-09	2.680E-07	0.9992	0.2518
10	3.401E-04	1.588E-09	1.673E-09	9.331E-08	0.9997	0.3144
11	3.401E-04	5.427E-10	4.325E-10	3.356E-08	0.9999	0.3418
12	3.401E-04	1.925E-10	1.160E-10	1.222E-08	1.0000	0.3548
13	3.401E-04	6.956E-11	3.290E-11	4.462E-09	1.0000	0.3613
14	3.401E-04	2.538E-11	1.001E-11	1.621E-09	1.0000	0.3649
15	3.401E-04	9.344E-12	3.262E-12	5.801E-10	1.0000	0.3681
16	3.401E-04	3.527E-12	1.126E-12	2.055E-10	1.0000	0.3774
17	3.401E-04	1.415E-12	4.216E-13	8.577E-11	1.0000	0.4011
18	3.401E-04	6.227E-13	2.032E-13	5.201E-11	1.0000	0.4402
19	3.401E-04	4.305E-13	1.483E-13	3.764E-11	1.0000	0.6913
20	3.401E-04	4.747E-13	1.375E-13	4.256E-11	1.0000	1.1028
convergence after 20 multigrid cycles mean convergence rate: 0.272 wall clock time: 1613 sec						

A.1.1.2 Third Series of Experiments — The Influence of the Grid Traversal

Table A.25: Test case 1 — Results of section 4.1.3

levels: 3W, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.990E-02				
1	1.651E-01	5.390E-02	1.838E-02	5.403E+00	0.0297	2.7090
2	1.618E-03	1.020E-03	1.892E-02	1.635E-01	0.0098	0.0189
3	1.330E-03	8.001E-06	5.463E-04	2.876E-04	0.8222	0.0078
4	1.337E-03	7.873E-07	4.872E-06	6.955E-06	1.0052	0.0984
5	1.331E-03	4.230E-08	3.066E-07	6.129E-06	0.9954	0.0537
6	1.331E-03	6.513E-09	1.596E-08	5.295E-07	0.9996	0.1540
7	1.331E-03	1.148E-09	1.881E-09	9.658E-08	0.9999	0.1763
8	1.331E-03	2.060E-10	3.268E-10	1.702E-08	1.0000	0.1793
9	1.331E-03	3.690E-11	5.879E-11	3.028E-09	1.0000	0.1792
10	1.331E-03	6.601E-12	1.055E-11	5.402E-10	1.0000	0.1789
11	1.331E-03	1.179E-12	1.889E-12	1.050E-10	1.0000	0.1786
convergence after 11 multigrid cycles mean convergence rate: 0.123 wall clock time: 70 sec						

Table A.26: Test case 1 — Results of section 4.1.3

levels: 3W, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$
$\epsilon_{\text{fine}} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	1.990E-02				
1	1.903E-01	1.827E-01	1.925E-02	5.378E+00	0.0342	9.1839
2	7.131E-03	1.292E-02	1.980E-02	1.831E-01	0.0375	0.0707
3	1.625E-03	1.338E-03	5.434E-04	5.506E-03	0.2278	0.1036
4	1.391E-03	1.499E-04	1.739E-05	2.337E-04	0.8561	0.1120
5	1.343E-03	1.689E-05	1.263E-06	4.823E-05	0.9653	0.1127
6	1.334E-03	2.075E-06	1.688E-07	8.904E-06	0.9934	0.1228
7	1.332E-03	3.259E-07	3.176E-08	2.171E-06	0.9984	0.1571
8	1.331E-03	7.033E-08	7.134E-09	6.464E-07	0.9995	0.2158
9	1.331E-03	1.810E-08	1.931E-09	2.101E-07	0.9998	0.2574
10	1.331E-03	5.045E-09	5.957E-10	7.053E-08	0.9999	0.2787
11	1.331E-03	1.487E-09	1.950E-10	2.394E-08	1.0000	0.2948
12	1.331E-03	4.600E-10	6.552E-11	8.152E-09	1.0000	0.3093
13	1.331E-03	1.478E-10	2.229E-11	2.779E-09	1.0000	0.3214
14	1.331E-03	4.878E-11	7.624E-12	9.512E-10	1.0000	0.3300
15	1.331E-03	1.638E-11	2.616E-12	3.250E-10	1.0000	0.3358
16	1.331E-03	5.554E-12	9.041E-13	1.112E-10	1.0000	0.3391
17	1.331E-03	1.840E-12	3.402E-13	4.184E-11	1.0000	0.3313

convergence after 17 multigrid cycles

mean convergence rate: 0.237

wall clock time: 56 sec

Table A.27: Test case 1 — Results of section 4.1.2.2

levels: 4W, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.487E-03				
1	1.717E-01	2.708E-02	2.300E-03	5.397E+00	0.0308	10.8888
2	3.369E-03	5.312E-04	2.371E-03	1.683E-01	0.0196	0.0196
3	4.046E-04	1.100E-05	7.201E-05	2.964E-03	0.1201	0.0207
4	3.379E-04	1.848E-07	1.435E-06	6.671E-05	0.8351	0.0168
5	3.339E-04	1.032E-08	2.482E-08	4.021E-06	0.9881	0.0558
6	3.334E-04	1.681E-09	8.309E-10	4.871E-07	0.9985	0.1629
7	3.333E-04	3.227E-10	9.427E-11	8.996E-08	0.9997	0.1920
8	3.333E-04	6.190E-11	1.858E-11	1.692E-08	0.9999	0.1918
9	3.333E-04	1.204E-11	3.524E-12	3.211E-09	1.0000	0.1945
10	3.333E-04	2.394E-12	7.133E-13	5.783E-10	1.0000	0.1988
11	3.333E-04	5.745E-13	2.153E-13	1.469E-10	1.0000	0.2400
12	3.333E-04	3.554E-13	1.290E-13	8.015E-11	1.0000	0.6187
13	3.333E-04	4.263E-13	1.235E-13	4.401E-12	1.0000	1.1995
convergence after 13 multigrid cycles mean convergence rate: 0.147 wall clock time: 749 sec						

Table A.28: Test case 1 — Results of section 4.1.3

levels: 4W, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.487E-03				
1	1.932E-01	9.126E-02	2.442E-03	5.375E+00	0.0347	36.6898
2	6.772E-03	6.412E-03	2.514E-03	1.864E-01	0.0351	0.0703
3	7.570E-04	6.565E-04	7.021E-05	6.015E-03	0.1118	0.1024
4	3.954E-04	7.474E-05	2.276E-06	3.616E-04	0.5223	0.1138
5	3.445E-04	8.301E-06	9.741E-08	5.088E-05	0.8713	0.1111
6	3.360E-04	9.385E-07	4.561E-09	8.470E-06	0.9754	0.1131
7	3.341E-04	1.157E-07	6.327E-10	1.914E-06	0.9943	0.1232
8	3.336E-04	1.850E-08	2.122E-10	5.436E-07	0.9984	0.1599
9	3.334E-04	4.253E-09	7.387E-11	1.758E-07	0.9995	0.2300
10	3.334E-04	1.208E-09	2.634E-11	6.032E-08	0.9998	0.2840
11	3.333E-04	3.774E-10	9.370E-12	2.118E-08	0.9999	0.3125
12	3.333E-04	1.246E-10	3.342E-12	7.485E-09	1.0000	0.3302
13	3.333E-04	4.270E-11	1.197E-12	2.616E-09	1.0000	0.3426
14	3.333E-04	1.522E-11	4.663E-13	8.844E-10	1.0000	0.3564
15	3.333E-04	5.174E-12	2.361E-13	3.990E-10	1.0000	0.3399
16	3.333E-04	2.010E-12	1.539E-13	2.123E-10	1.0000	0.3884
17	3.333E-04	1.574E-12	1.607E-13	7.675E-11	1.0000	0.7830
18	3.333E-04	1.332E-12	1.838E-13	4.803E-11	1.0000	0.8464
19	3.333E-04	9.730E-13	1.671E-13	1.615E-11	1.0000	0.7305
20	3.333E-04	1.239E-12	1.751E-13	4.057E-11	1.0000	1.2733
convergence after 20 multigrid cycles mean convergence rate: 0.285 wall clock time: 570 sec						

Table A.29: Test case 1 — Results of section 4.1.3

levels: 3W, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 2$, $\nu_2 = 2$ $\epsilon_{fine} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.344E-03				
1	1.685E-01	2.654E-02	2.355E-03	5.400E+00	0.0303	11.3227
2	9.537E-04	4.891E-04	2.424E-03	1.675E-01	0.0057	0.0184
3	2.318E-04	7.435E-06	6.955E-05	7.219E-04	0.2430	0.0152
4	3.279E-04	6.950E-07	4.847E-07	9.610E-05	1.4146	0.0935
5	3.209E-04	1.481E-08	7.591E-08	7.004E-06	0.9786	0.0213
6	3.205E-04	1.328E-09	2.359E-09	4.119E-07	0.9987	0.0897
7	3.204E-04	2.363E-10	2.016E-10	6.524E-08	0.9998	0.1779
8	3.204E-04	4.471E-11	3.343E-11	1.304E-08	1.0000	0.1892
9	3.204E-04	8.718E-12	6.317E-12	2.533E-09	1.0000	0.1950
10	3.204E-04	1.681E-12	1.249E-12	5.217E-10	1.0000	0.1929
11	3.204E-04	3.289E-13	2.747E-13	1.122E-10	1.0000	0.1956
12	3.204E-04	2.374E-13	1.570E-13	2.858E-11	1.0000	0.7218
13	3.204E-04	2.771E-13	1.479E-13	2.384E-11	1.0000	1.1671
convergence after 13 multigrid cycles mean convergence rate: 0.142 wall clock time: 810 sec						

Table A.30: Test case 1 — Results of section 4.1.3

levels: 3W, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-10}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \max(10^{-12}\|\boldsymbol{r}_c\|_2^{(k)}, 10^{-13})$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{r}_c\|_2^{(k)}$	$\|\boldsymbol{e}^{(k)} - \boldsymbol{e}^{(k-1)}\|_2$	$\|\boldsymbol{e}\|_2^{(k-1)}/\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.568E+00	2.344E-03				
1	1.927E-01	8.945E-02	2.381E-03	5.376E+00	0.0346	38.1641
2	6.796E-03	6.303E-03	2.450E-03	1.859E-01	0.0353	0.0705
3	6.632E-04	6.368E-04	6.691E-05	6.132E-03	0.0976	0.1010
4	3.719E-04	7.281E-05	2.835E-06	2.912E-04	0.5609	0.1143
5	3.301E-04	8.086E-06	2.216E-07	4.186E-05	0.8875	0.1111
6	3.227E-04	9.129E-07	2.981E-08	7.398E-06	0.9776	0.1129
7	3.211E-04	1.116E-07	5.287E-09	1.607E-06	0.9950	0.1223
8	3.206E-04	1.742E-08	1.099E-09	4.589E-07	0.9986	0.1560
9	3.205E-04	3.878E-09	2.601E-10	1.501E-07	0.9995	0.2227
10	3.204E-04	1.071E-09	7.228E-11	5.181E-08	0.9998	0.2762
11	3.204E-04	3.266E-10	2.270E-11	1.821E-08	0.9999	0.3049
12	3.204E-04	1.057E-10	7.620E-12	6.466E-09	1.0000	0.3236
13	3.204E-04	3.556E-11	2.645E-12	2.321E-09	1.0000	0.3365
14	3.204E-04	1.226E-11	9.422E-13	8.287E-10	1.0000	0.3447
15	3.204E-04	3.979E-12	3.610E-13	3.149E-10	1.0000	0.3246
16	3.204E-04	1.629E-12	2.151E-13	4.135E-11	1.0000	0.4093
17	3.204E-04	1.276E-12	1.586E-13	2.579E-11	1.0000	0.7833
18	3.204E-04	1.341E-12	1.628E-13	7.496E-12	1.0000	1.0513
convergence after 18 multigrid cycles mean convergence rate: 0.248 wall clock time: 604 sec						

A.1.1.3 Adjusting ϵ_{coarse}

For the tables in this section different values are shown in the tables. Instead of focusing on the errors and residuals of each iteration step the focus now is on the computed value $\boldsymbol{u}^{(k)}$ itself because information on the error is not available in production simulations. Instead, the difference $\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$ is a good measure of whether the desired accuracy is achieved.

Table A.31: Test case 1 — Results of section 4.1.6

levels: 3W, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/10$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.9898E-02	0.0000			
1	1.9027E-01	1.8274E-01	5.7335	5.734E+000	0.0000	9.1839
2	7.1311E-03	1.2921E-02	5.5722	1.851E-001	1.0290	0.0707
3	1.6246E-03	1.3383E-03	5.5694	5.822E-003	1.0005	0.1036
4	1.3909E-03	1.4994E-04	5.5696	7.273E-004	1.0000	0.1120
5	1.3427E-03	1.6894E-05	5.5696	2.126E-004	1.0000	0.1127
6	1.3337E-03	2.0737E-06	5.5696	7.033E-005	1.0000	0.1227
7	1.3314E-03	3.2532E-07	5.5696	2.363E-005	1.0000	0.1569
8	1.3306E-03	6.8203E-08	5.5696	7.740E-006	1.0000	0.2096
9	1.3307E-03	2.0487E-08	5.5696	2.729E-006	1.0000	0.3004
10	1.3307E-03	8.8251E-09	5.5696	1.247E-006	1.0000	0.4308
convergence after 10 multigrid cycles mean convergence rate: 0.188 wall clock time: 33 sec						

Table A.32: Test case 1 — Results of section 4.1.6

levels: 4W, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/10$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	2.4873E-03	0.0000			
1	1.9316E-01	9.1258E-02	5.7375	5.7375E+00	0.0000	36.6898
2	6.7720E-03	6.4116E-03	5.5719	1.8841E-01	1.0297	0.0703
3	7.5699E-04	6.5653E-04	5.5684	6.1770E-03	1.0006	0.1024
4	3.9539E-04	7.4737E-05	5.5686	5.8410E-04	1.0000	0.1138
5	3.4451E-04	8.3015E-06	5.5686	1.2480E-04	1.0000	0.1111
6	3.3623E-04	9.3950E-07	5.5686	3.9291E-05	1.0000	0.1132
7	3.3437E-04	1.1584E-07	5.5686	1.4101E-05	1.0000	0.1233
8	3.3365E-04	1.8718E-08	5.5686	4.9492E-06	1.0000	0.1616
9	3.3325E-04	5.4447E-09	5.5686	1.6751E-06	1.0000	0.2909
10	3.3319E-04	4.8534E-09	5.5686	1.0966E-06	1.0000	0.8914
11	3.3329E-04	2.4129E-09	5.5686	7.3772E-07	1.0000	0.4972
convergence after 11 multigrid cycles mean convergence rate: 0.209 wall clock time: 322 sec						

Table A.33: Test case 1 — Results of section 4.1.6

	levels: 3W, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/10$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	2.3438E-03	0.0000			
1	1.9268E-01	8.9450E-02	5.7370	5.7370E+00	0.0000	38.1641
2	6.7955E-03	6.3031E-03	5.5725	1.8738E-01	1.0295	0.0705
3	6.6316E-04	6.3683E-04	5.5684	6.3170E-03	1.0007	0.1010
4	3.7194E-04	7.2814E-05	5.5686	5.3900E-04	1.0000	0.1143
5	3.3011E-04	8.0870E-06	5.5686	1.0931E-04	1.0000	0.1111
6	3.2259E-04	9.1210E-07	5.5686	3.5622E-05	1.0000	0.1128
7	3.2092E-04	1.1119E-07	5.5686	1.2109E-05	1.0000	0.1219
8	3.2049E-04	1.6882E-08	5.5686	4.0072E-06	1.0000	0.1518
9	3.2051E-04	4.9481E-09	5.5686	1.4946E-06	1.0000	0.2931
10	3.2053E-04	2.5955E-09	5.5686	6.9826E-07	1.0000	0.5246
11	3.2034E-04	1.7606E-09	5.5686	3.9833E-07	1.0000	0.6783
	convergence after 11 multigrid cycles					
	mean convergence rate: 0.195					
	wall clock time: 362 sec					

Table A.34: Test case 1 — Results of section 4.1.6

	levels: 3W, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.9898E-02	0.0000			
1	1.9027E-01	1.8274E-01	5.7335	5.7335E+00	0.0000	9.1839
2	7.1311E-03	1.2921E-02	5.5722	1.8507E-01	1.0290	0.0707
3	1.6246E-03	1.3383E-03	5.5694	5.8218E-03	1.0005	0.1036
4	1.3909E-03	1.4994E-04	5.5696	7.2727E-04	1.0000	0.1120
5	1.3427E-03	1.6893E-05	5.5696	2.1259E-04	1.0000	0.1127
6	1.3338E-03	2.0750E-06	5.5696	7.0319E-05	1.0000	0.1228
7	1.3316E-03	3.2587E-07	5.5696	2.3659E-05	1.0000	0.1570
8	1.3310E-03	7.0253E-08	5.5696	8.1045E-06	1.0000	0.2156
9	1.3307E-03	1.8064E-08	5.5696	2.7846E-06	1.0000	0.2571
	convergence after 9 multigrid cycles					
	mean convergence rate: 0.177					
	wall clock time: 30 sec					

Table A.35: Test case 1 — Results of section 4.1.6

	levels: 4W, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	2.4873E-03	0.0000			
1	1.9316E-01	9.1258E-02	5.7375	5.7375E+00	0.0000	36.6898
2	6.7720E-03	6.4116E-03	5.5719	1.8841E-01	1.0297	0.0703
3	7.5699E-04	6.5653E-04	5.5684	6.1770E-03	1.0006	0.1024
4	3.9540E-04	7.4737E-05	5.5686	5.8410E-04	1.0000	0.1138
5	3.4452E-04	8.3013E-06	5.5686	1.2477E-04	1.0000	0.1111
6	3.3604E-04	9.3838E-07	5.5686	3.9277E-05	1.0000	0.1130
7	3.3414E-04	1.1574E-07	5.5686	1.4032E-05	1.0000	0.1233
8	3.3361E-04	1.8539E-08	5.5686	5.0009E-06	1.0000	0.1602
9	3.3345E-04	4.3598E-09	5.5686	1.7914E-06	1.0000	0.2352
10	3.3337E-04	1.1914E-09	5.5686	6.2695E-07	1.0000	0.2733
	convergence after 10 multigrid cycles					
	mean convergence rate: 0.169					
	wall clock time: 293 sec					

Table A.36: Test case 1 — Results of section 4.1.6

	levels: 3W, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	2.3438E-03	0.0000			
1	1.9268E-01	8.9450E-02	5.7370	5.7370E+00	0.0000	38.1641
2	6.7955E-03	6.3031E-03	5.5725	1.8738E-01	1.0295	0.0705
3	6.6316E-04	6.3683E-04	5.5684	6.3170E-03	1.0007	0.1010
4	3.7194E-04	7.2814E-05	5.5686	5.3900E-04	1.0000	0.1143
5	3.3009E-04	8.0864E-06	5.5686	1.0934E-04	1.0000	0.1111
6	3.2270E-04	9.1298E-07	5.5686	3.5620E-05	1.0000	0.1129
7	3.2108E-04	1.1155E-07	5.5686	1.2141E-05	1.0000	0.1222
8	3.2061E-04	1.7350E-08	5.5686	4.2696E-06	1.0000	0.1555
9	3.2046E-04	3.8616E-09	5.5686	1.5219E-06	1.0000	0.2226
10	3.2041E-04	1.0249E-09	5.5686	5.2799E-07	1.0000	0.2654
	convergence after 10 multigrid cycles mean convergence rate: 0.167 wall clock time: 338 sec					

Table A.37: Test case 1 — Results of section 4.1.6

	levels: 3W, $N_c = 25^3$, $N_f = 100^3$, $h = 6.28 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/1000$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.9898E-02	0.0000			
1	1.9027E-01	1.8274E-01	5.7335	5.7335E+00	0.0000	9.1839
2	7.1311E-03	1.2921E-02	5.5722	1.8507E-01	1.0290	0.0707
3	1.6246E-03	1.3383E-03	5.5694	5.8218E-03	1.0005	0.1036
4	1.3909E-03	1.4994E-04	5.5696	7.2728E-04	1.0000	0.1120
5	1.3427E-03	1.6893E-05	5.5696	2.1259E-04	1.0000	0.1127
6	1.3338E-03	2.0747E-06	5.5696	7.0319E-05	1.0000	0.1228
7	1.3316E-03	3.2591E-07	5.5696	2.3659E-05	1.0000	0.1571
8	1.3310E-03	7.0329E-08	5.5696	8.1054E-06	1.0000	0.2158
9	1.3308E-03	1.8098E-08	5.5696	2.7868E-06	1.0000	0.2573
convergence after 9 multigrid cycles mean convergence rate: 0.177 wall clock time: 31 sec						

Table A.38: Test case 1 — Results of section 4.1.6

	levels: 4W, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/1000$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	2.4873E-03	0.0000			
1	1.9316E-01	9.1258E-02	5.7375	5.7375E+00	0.0000	36.6898
2	6.7720E-03	6.4116E-03	5.5719	1.8841E-01	1.0297	0.0703
3	7.5699E-04	6.5653E-04	5.5684	6.1770E-03	1.0006	0.1024
4	3.9540E-04	7.4737E-05	5.5686	5.8410E-04	1.0000	0.1138
5	3.4452E-04	8.3013E-06	5.5686	1.2477E-04	1.0000	0.1111
6	3.3604E-04	9.3849E-07	5.5686	3.9278E-05	1.0000	0.1131
7	3.3413E-04	1.1565E-07	5.5686	1.4032E-05	1.0000	0.1232
8	3.3359E-04	1.8503E-08	5.5686	4.9992E-06	1.0000	0.1600
9	3.3341E-04	4.2577E-09	5.5686	1.7822E-06	1.0000	0.2301
10	3.3335E-04	1.2108E-09	5.5686	6.3525E-07	1.0000	0.2844
	convergence after 10 multigrid cycles mean convergence rate: 0.169 wall clock time: 300 sec					

Table A.39: Test case 1 — Results of section 4.1.6

	levels: 3W, $N_c = 51^3$, $N_f = 204^3$, $h = 3.08 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/1000$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	2.3438E-03	0.0000			
1	1.9268E-01	8.9450E-02	5.7370	5.7370E+00	0.0000	38.1641
2	6.7955E-03	6.3031E-03	5.5725	1.8738E-01	1.0295	0.0705
3	6.6316E-04	6.3683E-04	5.5684	6.3170E-03	1.0007	0.1010
4	3.7194E-04	7.2814E-05	5.5686	5.3900E-04	1.0000	0.1143
5	3.3009E-04	8.0864E-06	5.5686	1.0934E-04	1.0000	0.1111
6	3.2269E-04	9.1287E-07	5.5686	3.5619E-05	1.0000	0.1129
7	3.2108E-04	1.1161E-07	5.5686	1.2142E-05	1.0000	0.1223
8	3.2063E-04	1.7414E-08	5.5686	4.2709E-06	1.0000	0.1560
9	3.2047E-04	3.8757E-09	5.5686	1.5240E-06	1.0000	0.2226
10	3.2042E-04	1.0696E-09	5.5686	5.4367E-07	1.0000	0.2760
convergence after 10 multigrid cycles mean convergence rate: 0.167 wall clock time: 352 sec						

A.2 Test Case 2 — Linear Equation with Neumann B.C.

The problem considered is

$$-\nabla\cdot(x\nabla u(\boldsymbol{x}))+xu(\boldsymbol{x})=4x\sin(x)\sin(y)\sin(z)-\cos(x)\sin(y)\sin(z). \tag{A.4}$$

with the boundary conditions

$$\begin{aligned}\frac{\partial u}{\partial x}(0,y,z)&=\sin(y)\sin(z),\\ \frac{\partial u}{\partial x}(2\pi,y,z)&=\sin(y)\sin(z),\\ u(x,0,z)&=u(x,2\pi,z),\\ u(x,y,0)&=u(x,y,2\pi).\end{aligned} \tag{A.5}$$

The exact solution is

$$u(x,y,z)=\sin(x)\sin(y)\sin(z). \tag{A.6}$$

Table A.40: Test case 2 — Results of section 4.2.1

levels: 2W, $N_c=25^3$, $N_f=50^3$, $h=1.26\cdot10^{-1}$, $\nu_1=1$, $\nu_2=1$ $\epsilon_{fine}=\max(10^{-6}\|\boldsymbol{r}\|_2^{(0)},10^{-13})$, $\epsilon_{coarse}=\epsilon_{fine}/100$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)}-\boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.9314E-01	0.0000			
1	1.7789E-01	3.6237E-01	5.7150	5.7150E+00	0.0000	1.8762
2	1.1458E-02	2.6814E-02	5.5721	1.7192E-01	1.0257	0.0740
3	6.1280E-03	2.8631E-03	5.5727	7.2686E-03	0.9999	0.1068
4	5.5673E-03	3.1277E-04	5.5728	1.2703E-03	1.0000	0.1092
5	5.4689E-03	3.6067E-05	5.5728	2.8763E-04	1.0000	0.1153
6	5.4459E-03	4.8515E-06	5.5728	7.1058E-05	1.0000	0.1345
7	5.4402E-03	8.5041E-07	5.5728	1.8032E-05	1.0000	0.1753
8	5.4387E-03	1.8345E-07	5.5728	4.6231E-06	1.0000	0.2157
convergence after 8 multigrid cycles						
mean convergence rate: 0.178						
wall clock time: 2 sec						

Table A.41: Test case 2 — Results of section 4.2.1

	levels: 4W, $N_c = 25^3$, $N_f = 200^3$, $h = 3.14 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	4.2486E-03	0.0000			
1	1.9280E-01	9.1250E-02	5.7369	5.7369E+00	0.0000	21.4776
2	7.4879E-03	6.4145E-03	5.5712	1.8861E-01	1.0297	0.0703
3	1.2414E-03	6.5972E-04	5.5683	6.5477E-03	1.0005	0.1028
4	5.5234E-04	7.5085E-05	5.5686	8.2513E-04	1.0000	0.1138
5	3.8410E-04	8.3376E-06	5.5686	2.3721E-04	1.0000	0.1110
6	3.4594E-04	9.4254E-07	5.5686	7.9163E-05	1.0000	0.1130
7	3.3607E-04	1.1618E-07	5.5686	2.7512E-05	1.0000	0.1233
8	3.3312E-04	1.9644E-08	5.5686	9.5827E-06	1.0000	0.1691
9	3.3216E-04	5.1842E-09	5.5686	3.3540E-06	1.0000	0.2639
10	3.3180E-04	1.5685E-09	5.5686	1.1634E-06	1.0000	0.3026
	convergence after 10 multigrid cycles					
	mean convergence rate: 0.173					
	wall clock time: 81 sec					

A.3 Test Case 3 — Linear Equation with Nonhom. Dirichlet B. C.

The problem considered is

$$-\nabla\cdot((x+1)\nabla u(\boldsymbol{x}))+10000\,u(\boldsymbol{x})=2(x+1)\cos(z)\sin(y)+10000\cos(z)\sin(y) \tag{A.7}$$

with the boundary conditions

$$\begin{aligned} u(0,y,z)&=\cos(z)\sin(y),\\ u(2\pi,y,z)&=\cos(z)\sin(y),\\ u(x,0,z)&=u(x,2\pi,z),\\ u(x,y,0)&=u(x,y,2\pi). \end{aligned} \tag{A.8}$$

The exact solution is

$$u(x,y,z)=\sin(y)\cos(z). \tag{A.9}$$

Table A.42: Test case 3 — Results of section 4.3.1

levels: 4W, $N_c=25^3$, $N_f=200^3$, $h=3.14\cdot10^{-2}$, $\nu_1=1$, $\nu_2=1$ $\epsilon_{fine}=\max(10^{-6}\|\boldsymbol{r}\|_2^{(0)},10^{-13})$, $\epsilon_{coarse}=\epsilon_{fine}/100$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)}-\boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	7.8551E+00	2.4512E+00	0.0000			
1	1.0146E-01	1.0338E-01	7.9251	7.9251E+00	0.0000	0.0422
2	5.1133E-03	6.3412E-03	7.8541	9.9895E-02	1.0090	0.0613
3	6.1867E-04	6.2557E-04	7.8551	4.5349E-03	0.9999	0.0987
4	8.0520E-05	6.9436E-05	7.8551	5.4018E-04	1.0000	0.1110
5	1.0768E-05	8.5729E-06	7.8551	6.9935E-05	1.0000	0.1235
6	1.5571E-06	1.1216E-06	7.8551	9.3241E-06	1.0000	0.1308
convergence after 6 multigrid cycles mean convergence rate: 0.192 wall clock time: 52 sec						

Table A.43: Test case 3 — Results of section 4.3.1

	levels: 4W, $N_c = 31$, $N_f = 248$, $h = 2.53 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{coarse} = \epsilon_{fine}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	7.8589E+00	1.2891E+00	0.0000			
1	1.3275E-01	9.4888E-02	7.9564	7.9564E+00	0.0000	0.0736
2	7.1758E-03	6.3565E-03	7.8573	1.3117E-01	1.0126	0.0670
3	1.0371E-03	7.2060E-04	7.8589	6.2233E-03	0.9998	0.1134
4	1.6422E-04	9.5073E-05	7.8589	8.7703E-04	1.0000	0.1319
5	2.6754E-05	1.4371E-05	7.8589	1.3786E-04	1.0000	0.1512
6	4.4213E-06	2.3126E-06	7.8589	2.2390E-05	1.0000	0.1609
7	8.1870E-07	3.8139E-07	7.8589	3.6845E-06	1.0000	0.1649
convergence after 7 multigrid cycles						
mean convergence rate: 0.205						
wall clock time: 106 sec						

A.4 Test Case 4 — Parallel Multigrid

The problem considered is

$$-\nabla\cdot(x\nabla u(\boldsymbol{x}))+x\,u(\boldsymbol{x}) = 4x\,\sin(x)\,\sin(y)\,\sin(z) - \cos(x)\,\sin(y)\,\sin(z) \tag{A.10}$$

with the boundary conditions

$$\begin{aligned} u(0,y,z) &= u(2\pi,y,z) = 0,\\ u(x,0,z) &= u(x,2\pi,z),\\ u(x,y,0) &= u(x,y,2\pi) \end{aligned} \tag{A.11}$$

and the exact solution

$$u(x,y,z) = \sin(x)\,\sin(y)\,\sin(z). \tag{A.12}$$

Table A.44: Test Case 4 — results of section 4.4.2

levels: 4W, $N = 248$, $h = 2.53\cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$ domain decomp.: 1 1 1						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)}-\boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.3046E-03	0.0000			
1	1.9424E-01	7.3550E-02	5.7389	5.7389E+00	0.0000	56.3789
2	7.0483E-03	5.1631E-03	5.5710	1.9037E-01	1.0301	0.0702
3	6.1498E-04	5.3040E-04	5.5684	6.5965E-03	1.0005	0.1027
4	2.9081E-04	6.0364E-05	5.5685	5.1783E-04	1.0000	0.1138
5	2.2772E-04	6.6974E-06	5.5685	1.2341E-04	1.0000	0.1109
6	2.1961E-04	7.5139E-07	5.5685	3.2222E-05	1.0000	0.1122
7	2.1770E-04	8.9810E-08	5.5685	1.2275E-05	1.0000	0.1195
8	2.1719E-04	1.3163E-08	5.5685	4.3057E-06	1.0000	0.1466
9	2.1703E-04	2.9151E-09	5.5685	1.5747E-06	1.0000	0.2215
10	2.1697E-04	7.9827E-10	5.5685	5.5823E-07	1.0000	0.2738
convergence after 10 multigrid cycles mean convergence rate: 0.166 wall clock time: 162 sec						

Table A.45: Test case 4

levels: 4W, $N = 248$, $h = 2.53 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$ domain decomp.: 2 1 1						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.3046E-03	0.0000			
1	1.9426E-01	7.3548E-02	5.7389	5.7389E+00	0.0000	56.3779
2	7.0482E-03	5.1633E-03	5.5710	1.9041E-01	1.0301	0.0702
3	6.1587E-04	5.3031E-04	5.5684	6.5960E-03	1.0005	0.1027
4	2.9089E-04	6.0365E-05	5.5685	5.1824E-04	1.0000	0.1138
5	2.2773E-04	6.6999E-06	5.5685	1.2356E-04	1.0000	0.1110
6	2.1961E-04	7.5330E-07	5.5685	3.2243E-05	1.0000	0.1124
7	2.1770E-04	9.0866E-08	5.5685	1.2280E-05	1.0000	0.1206
8	2.1719E-04	1.3629E-08	5.5685	4.3066E-06	1.0000	0.1500
9	2.1703E-04	3.0960E-09	5.5685	1.5754E-06	1.0000	0.2272
10	2.1697E-04	1.0177E-09	5.5685	5.6430E-07	1.0000	0.3287
convergence after 10 multigrid cycles mean convergence rate: 0.167 wall clock time: 86 sec						

Table A.46: Test case 4

levels: 4W, $N = 248$, $h = 2.53 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$ domain decomp.: 2 2 1						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.3046E-03	0.0000			
1	1.9426E-01	7.3549E-02	5.7389	5.7389E+00	0.0000	56.3784
2	7.0480E-03	5.1634E-03	5.5710	1.9041E-01	1.0301	0.0702
3	6.1601E-04	5.3035E-04	5.5684	6.5959E-03	1.0005	0.1027
4	2.9088E-04	6.0366E-05	5.5685	5.1842E-04	1.0000	0.1138
5	2.2773E-04	6.6999E-06	5.5685	1.2355E-04	1.0000	0.1110
6	2.1961E-04	7.5335E-07	5.5685	3.2242E-05	1.0000	0.1124
7	2.1770E-04	9.0884E-08	5.5685	1.2280E-05	1.0000	0.1206
8	2.1719E-04	1.3632E-08	5.5685	4.3065E-06	1.0000	0.1500
9	2.1703E-04	3.0965E-09	5.5685	1.5753E-06	1.0000	0.2271
10	2.1697E-04	1.0164E-09	5.5685	5.6397E-07	1.0000	0.3283
convergence after 10 multigrid cycles mean convergence rate: 0.167 wall clock time: 45 sec						

Table A.47: Test case 4

levels: 4W, $N = 248$, $h = 2.53 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$ domain decomp.: 2 2 2						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.3046E-03	0.0000			
1	2.0786E-01	7.3727E-02	5.7530	5.7530E+00	0.0000	56.5148
2	3.0999E-02	4.8953E-03	5.5558	2.2233E-01	1.0355	0.0664
3	4.5299E-03	7.7013E-04	5.5697	3.5115E-02	0.9975	0.1573
4	1.3238E-03	7.1434E-05	5.5683	5.7507E-03	1.0003	0.0928
5	3.3376E-04	1.8791E-05	5.5686	1.5490E-03	1.0000	0.2630
6	2.2203E-04	2.1236E-06	5.5685	2.7395E-04	1.0000	0.1130
7	2.1836E-04	6.0521E-07	5.5685	6.1095E-05	1.0000	0.2850
8	2.1711E-04	8.6666E-08	5.5685	1.1595E-05	1.0000	0.1432
9	2.1702E-04	2.0984E-08	5.5685	2.7621E-06	1.0000	0.2421
10	2.1695E-04	3.4956E-09	5.5685	6.9488E-07	1.0000	0.1666
11	2.1693E-04	9.4854E-10	5.5685	2.2742E-07	1.0000	0.2714
convergence after 11 multigrid cycles mean convergence rate: 0.208 wall clock time: 28 sec						

Table A.48: Test case 4

levels: 4W, $N = 216$, $h = 2.91 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$ domain decomp.: 4 2 2						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.9745E-03	0.0000			
1	2.0444E-01	8.4356E-02	5.7486	5.7486E+00	0.0000	42.7228
2	2.6601E-02	5.7789E-03	5.5566	2.1645E-01	1.0346	0.0685
3	3.3539E-03	9.1014E-04	5.5695	2.8993E-02	0.9977	0.1575
4	9.3065E-04	1.6571E-04	5.5684	3.8890E-03	1.0002	0.1821
5	4.0545E-04	1.3315E-04	5.5686	9.8317E-04	1.0000	0.8035
6	3.6065E-04	1.3114E-04	5.5686	1.7469E-04	1.0000	0.9849
convergence after 6 multigrid cycles mean convergence rate: 0.390 wall clock time: 8 sec						

Table A.49: Test case 4

levels: 4W, $N = 216$, $h = 2.91 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$ domain decomp.: 4 4 2						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.9745E-03	0.0000			
1	2.0605E-01	8.6168E-02	5.7492	5.7492E+00	0.0000	43.6408
2	2.6520E-02	6.2464E-03	5.5570	2.1755E-01	1.0346	0.0725
3	3.2955E-03	9.8878E-04	5.5695	2.8773E-02	0.9977	0.1583
4	8.8975E-04	1.7512E-04	5.5684	3.7528E-03	1.0002	0.1771
5	4.0388E-04	1.3349E-04	5.5686	9.2414E-04	1.0000	0.7623
6	3.6055E-04	1.3115E-04	5.5686	1.6384E-04	1.0000	0.9824
convergence after 6 multigrid cycles mean convergence rate: 0.382 wall clock time: 5 sec						

Table A.50: Test case 4

levels: 4W, $N = 216$, $h = 2.91 \cdot 10^{-2}$, $\nu_1 = 1$, $\nu_2 = 1$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$ domain decomp.: 4 4 4						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.9745E-03	0.0000			
1	2.3440E-01	8.8340E-02	5.7752	5.7752E+00	0.0000	44.7405
2	5.1605E-02	6.4008E-03	5.5403	2.6950E-01	1.0424	0.0725
3	1.1495E-02	1.0970E-03	5.5585	4.2246E-02	0.9967	0.1714
4	3.8395E-03	3.7924E-04	5.5712	1.5102E-02	0.9977	0.3457
5	5.7701E-04	1.3626E-04	5.5688	3.4778E-03	1.0004	0.3593
6	3.3257E-04	1.3126E-04	5.5685	3.2722E-04	1.0001	0.9633
convergence after 6 multigrid cycles mean convergence rate: 0.382 wall clock time: 3 sec						

A.5 Test Case 5 — Nonlinear Equation with Dirichlet Boundaries

The problem considered is

$$\begin{aligned}-\nabla\cdot((5+u(\boldsymbol{x}))\nabla u(\boldsymbol{x}))+u(\boldsymbol{x}) = &-(\cos(z)^2\sin(1+x)^2\sin(y)^2)+\sin(1+x)\sin(y)\sin(z)\\ &-\cos(y)^2\sin(1+x)^2\sin(z)^2-\cos(1+x)^2\sin(y)^2\sin(z)^2\\ &+3\sin(1+x)\sin(y)\sin(z)\cdot(5+\sin(1+x)\cdot\sin(y)\cdot\sin(z))\end{aligned} \tag{A.13}$$

with the boundary conditions

$$\begin{aligned}u(0,y,z) =& u(2\pi,y,z)=\sin(1)\sin(y)\sin(z),\\ u(x,0,z) =& u(x,2\pi,z),\\ u(x,y,0) =& u(x,y,2\pi)\end{aligned} \tag{A.14}$$

and the exact solution

$$u(x,y,z)=\sin(1+x)\,\sin(y)\,\sin(z). \tag{A.15}$$

Table A.51: Test case 5

levels: 3W, $N_\mathrm{f}=164\times164\times164$, $h_x=3.83\cdot10^{-2}$, $\nu_1=5$, $\nu_2=5$ $\epsilon_\mathrm{fine}=\max(10^{-6}\|\boldsymbol{r}\|_2^{(0)},10^{-13})$, $\epsilon_\mathrm{coarse}=\epsilon_\mathrm{fine}/100$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)}-\boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5442E+00	1.4120E-01	0.0000			
1	3.6619E-01	9.0518E-03	8.7689	5.7074E+00	0.7534	0.0641
2	8.4098E-03	1.5190E-04	8.6533	3.7398E-01	1.0134	0.0168
3	1.1237E-03	3.2693E-06	8.6559	9.2550E-03	0.9997	0.0215
4	6.3614E-04	2.0268E-07	8.6558	5.6506E-04	1.0000	0.0620
convergence after 5 multigrid cycles wall clock time: 350 sec						

Table A.52: Test case 5

levels: 4W, $N_f = 200 \times 200 \times 200$, $h_x = 3.14 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-13})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5486E+00	1.0466E-01	0.0000			
1	3.6757E-01	6.7288E-03	8.7709	5.7161E+00	0.7536	0.0643
2	8.1434E-03	1.1899E-04	8.6529	3.7522E-01	1.0136	0.0177
3	6.5818E-04	2.4923E-06	8.6556	8.5784E-03	0.9997	0.0209
4	3.9355E-04	1.1310E-07	8.6557	4.3983E-04	1.0000	0.0454
convergence after 5 multigrid cycles wall clock time: 315 sec						

A.6 Test Case 6 — Nonlinear Equation with Neumann Boundaries

The problem that we are solving is

$$\begin{aligned}-\nabla\cdot((10+u(\boldsymbol{x}))\nabla u(\boldsymbol{x}))+(1000+u(\boldsymbol{x})) = 1000 &- \sin^2(x)\sin^2(y)\cos^2(z)+\sin(x)\sin(y)\sin(z)\\ &-\sin^2(x)\cos^2(y)\sin^2(z)-\cos^2(x)\sin^2(y)\sin^2(z)\\ &+3\sin(x)\sin(y)\sin(z)(\sin(x)\sin(y)\sin(z)+10)\end{aligned} \tag{A.16}$$

with the boundary conditions

$$\begin{aligned}\frac{\partial u}{\partial x}(0,y,z) &= \frac{\partial u}{\partial x}(2\pi,y,z) = \sin(y)\sin(z),\\ u(x,0,z) &= u(x,2\pi,z),\\ u(x,y,0) &= u(x,y,2\pi)\end{aligned} \tag{A.17}$$

and the exact solution

$$u(x,y,z) = \sin(x)\,\sin(y)\,\sin(z). \tag{A.18}$$

Table A.53: Test case 6 — Results of section 4.6

levels: 3W, $N_\mathrm{f} = 84\times84\times84$, $h_x = 7.48\cdot10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$ $\epsilon_\mathrm{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_\mathrm{coarse} = \epsilon_\mathrm{fine}/100$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)}-\boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	9.9278E-02	0.0000			
1	2.4521E-01	8.3576E-03	5.6799	5.6799E+00	0.0000000	0.0842
2	9.1901E-02	1.7318E-03	5.5395	2.2236E-01	1.0253588	0.2072
3	5.8943E-02	1.1081E-03	5.5528	3.3674E-02	0.9976006	0.6399
4	3.8369E-02	7.1319E-04	5.5589	2.0845E-02	0.9989097	0.6436
5	2.5290E-02	4.5925E-04	5.5629	1.3434E-02	0.9992779	0.6439
6	1.7080E-02	2.9584E-04	5.5655	8.6554E-03	0.9995316	0.6442
7	1.2026E-02	1.9068E-04	5.5672	5.5837E-03	0.9996962	0.6445
8	8.9949E-03	1.2303E-04	5.5683	3.6113E-03	0.9998029	0.6452
9	7.2116E-03	7.9525E-05	5.5690	2.3480E-03	0.9998719	0.6464
10	6.1493E-03	5.1592E-05	5.5694	1.5427E-03	0.9999166	0.6487
11	5.4771E-03	3.3704E-05	5.5698	1.0338E-03	0.9999454	0.6533
12	5.0091E-03	2.2308E-05	5.5700	7.1651E-04	0.9999641	0.6619
13	4.6507E-03	1.5112E-05	5.5701	5.2219E-04	0.9999762	0.6774
14	4.3562E-03	1.0629E-05	5.5702	4.0454E-04	0.9999840	0.7033
15	4.1040E-03	7.8765E-06	5.5702	3.3234E-04	0.9999892	0.7410

Continued on next page

Table A.53 – *Continued from previous page*

levels: 3W, $N_f = 84 \times 84 \times 84$, $h_x = 7.48 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$						
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
16	3.8836E-03	6.1937E-06	5.5703	2.8557E-04	0.9999925	0.7863
17	3.6893E-03	5.1405E-06	5.5703	2.5252E-04	0.9999947	0.8300
18	3.5175E-03	4.4402E-06	5.5703	2.2691E-04	0.9999961	0.8638
19	3.3659E-03	3.9332E-06	5.5703	2.0561E-04	0.9999971	0.8858
20	3.2322E-03	3.5343E-06	5.5704	1.8709E-04	0.9999978	0.8986
21	3.1148E-03	3.2004E-06	5.5704	1.7057E-04	0.9999982	0.9055
22	3.0118E-03	2.9095E-06	5.5704	1.5564E-04	0.9999986	0.9091
23	2.9217E-03	2.6505E-06	5.5704	1.4207E-04	0.9999988	0.9110
24	2.8431E-03	2.4170E-06	5.5704	1.2970E-04	0.9999990	0.9119
25	2.7747E-03	2.2053E-06	5.5704	1.1842E-04	0.9999991	0.9124
26	2.7151E-03	2.0126E-06	5.5704	1.0811E-04	0.9999992	0.9126
27	2.6634E-03	1.8370E-06	5.5704	9.8712E-05	0.9999993	0.9128
28	2.6186E-03	1.6768E-06	5.5704	9.0110E-05	0.9999994	0.9128
29	2.5797E-03	1.5307E-06	5.5704	8.2273E-05	0.9999994	0.9129
30	2.5460E-03	1.3973E-06	5.5704	7.5109E-05	0.9999995	0.9129
31	2.5169E-03	1.2756E-06	5.5704	6.8557E-05	0.9999995	0.9129
32	2.4915E-03	1.1645E-06	5.5704	6.2594E-05	0.9999996	0.9129
33	2.4696E-03	1.0630E-06	5.5704	5.7150E-05	0.9999996	0.9129
34	2.4506E-03	9.7043E-07	5.5704	5.2161E-05	0.9999997	0.9129
35	2.4340E-03	8.8590E-07	5.5704	4.7624E-05	0.9999997	0.9129
36	2.4197E-03	8.0872E-07	5.5704	4.3465E-05	0.9999997	0.9129
37	2.4071E-03	7.3828E-07	5.5704	3.9686E-05	0.9999997	0.9129
38	2.3962E-03	6.7398E-07	5.5704	3.6235E-05	0.9999998	0.9129
39	2.3867E-03	6.1527E-07	5.5704	3.3064E-05	0.9999998	0.9129
40	2.3784E-03	5.6168E-07	5.5704	3.0193E-05	0.9999998	0.9129
41	2.3711E-03	5.1277E-07	5.5704	2.7565E-05	0.9999998	0.9129
42	2.3647E-03	4.6811E-07	5.5704	2.5162E-05	0.9999998	0.9129
43	2.3591E-03	4.2734E-07	5.5704	2.2963E-05	0.9999999	0.9129
44	2.3542E-03	3.9015E-07	5.5704	2.0978E-05	0.9999999	0.9130
45	2.3498E-03	3.5620E-07	5.5704	1.9143E-05	0.9999999	0.9130
46	2.3459E-03	3.2521E-07	5.5704	1.7473E-05	0.9999999	0.9130
47	2.3425E-03	2.9695E-07	5.5704	1.5949E-05	0.9999999	0.9131
48	2.3395E-03	2.7118E-07	5.5704	1.4551E-05	0.9999999	0.9132

Continued on next page

Table A.53 – *Continued from previous page*

levels: 3W, $N_\mathrm{f} = 84 \times 84 \times 84$, $h_x = 7.48 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_\mathrm{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_\mathrm{coarse} = \epsilon_\mathrm{fine}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
49	2.3369E-03	2.4759E-07	5.5704	1.3253E-05	0.9999999	0.9130
50	2.3345E-03	2.2603E-07	5.5704	1.2081E-05	0.9999999	0.9129
51	2.3324E-03	2.0632E-07	5.5704	1.1014E-05	0.9999999	0.9128
52	2.3305E-03	1.8827E-07	5.5704	1.0037E-05	0.9999999	0.9125
53	2.3289E-03	1.7181E-07	5.5704	9.1550E-06	0.9999999	0.9126
54	2.3274E-03	1.5676E-07	5.5704	8.3492E-06	0.9999999	0.9124
55	2.3260E-03	1.4302E-07	5.5704	7.6120E-06	1.0000000	0.9124
56	2.3248E-03	1.3048E-07	5.5704	6.9395E-06	1.0000000	0.9123
57	2.3238E-03	1.1906E-07	5.5704	6.3373E-06	1.0000000	0.9125
58	2.3228E-03	1.0858E-07	5.5704	5.7667E-06	1.0000000	0.9120

convergence after 59 Newton–multigrid cycles
mean convergence rate: 0.844
wall clock time: 682 sec

Table A.54: Test case 6 — Results of section 4.6

	levels: 4W, $N_f = 168 \times 168 \times 168$, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.5041E-02	0.0000			
1	2.3015E-01	3.2749E-03	5.7109	5.7109E+00	0.0000000	0.2177
2	6.6450E-02	2.5299E-04	5.5447	2.3354E-01	1.0299780	0.0773
3	5.2334E-02	1.7550E-04	5.5532	1.5413E-02	0.9984710	0.6937
4	4.2121E-02	1.4084E-04	5.5561	1.0257E-02	0.9994787	0.8025
5	3.3909E-02	1.1312E-04	5.5586	8.2634E-03	0.9995580	0.8031
6	2.7331E-02	9.0856E-05	5.5606	6.6361E-03	0.9996440	0.8032
7	2.2065E-02	7.2984E-05	5.5621	5.3305E-03	0.9997130	0.8033
8	1.7855E-02	5.8633E-05	5.5634	4.2823E-03	0.9997687	0.8034
9	1.4495E-02	4.7108E-05	5.5645	3.4405E-03	0.9998136	0.8034
10	1.1821E-02	3.7853E-05	5.5653	2.7647E-03	0.9998498	0.8035
11	9.6994E-03	3.0419E-05	5.5660	2.2221E-03	0.9998790	0.8036
12	8.0229E-03	2.4448E-05	5.5665	1.7865E-03	0.9999026	0.8037
13	6.7050E-03	1.9652E-05	5.5670	1.4370E-03	0.9999215	0.8038
14	5.6736E-03	1.5800E-05	5.5673	1.1566E-03	0.9999367	0.8040
15	4.8711E-03	1.2707E-05	5.5676	9.3168E-04	0.9999490	0.8042
16	4.2489E-03	1.0223E-05	5.5678	7.5149E-04	0.9999588	0.8045
17	3.7672E-03	8.2293E-06	5.5680	6.0725E-04	0.9999668	0.8050
18	3.3929E-03	6.6292E-06	5.5682	4.9195E-04	0.9999732	0.8056
19	3.0995E-03	5.3456E-06	5.5683	3.9996E-04	0.9999783	0.8064
20	2.8661E-03	4.3166E-06	5.5684	3.2675E-04	0.9999824	0.8075
21	2.6766E-03	3.4925E-06	5.5685	2.6866E-04	0.9999858	0.8091
22	2.5192E-03	2.8333E-06	5.5685	2.2275E-04	0.9999885	0.8113
23	2.3851E-03	2.3069E-06	5.5686	1.8664E-04	0.9999906	0.8142
24	2.2681E-03	1.8874E-06	5.5686	1.5834E-04	0.9999924	0.8182
25	2.1640E-03	1.5541E-06	5.5687	1.3627E-04	0.9999938	0.8234
26	2.0696E-03	1.2901E-06	5.5687	1.1909E-04	0.9999949	0.8301
27	1.9829E-03	1.0818E-06	5.5687	1.0570E-04	0.9999958	0.8385
28	1.9024E-03	9.1808E-07	5.5687	9.5217E-05	0.9999965	0.8487
29	1.8272E-03	7.8983E-07	5.5687	8.6906E-05	0.9999971	0.8603
30	1.7564E-03	6.8951E-07	5.5688	8.0201E-05	0.9999976	0.8730
31	1.6896E-03	6.1094E-07	5.5688	7.4706E-05	0.9999980	0.8860

Continued on next page

Table A.54 – *Continued from previous page*

	levels: 4W, $N_f = 168 \times 168 \times 168$, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
32	1.6264E-03	5.4906E-07	5.5688	7.0059E-05	0.9999983	0.8987
33	1.5665E-03	4.9984E-07	5.5688	6.6061E-05	0.9999986	0.9103
34	1.5097E-03	4.6009E-07	5.5688	6.2548E-05	0.9999988	0.9205
35	1.4558E-03	4.2737E-07	5.5688	5.9368E-05	0.9999990	0.9289
36	1.4046E-03	3.9985E-07	5.5688	5.6476E-05	0.9999991	0.9356
37	1.3561E-03	3.7619E-07	5.5688	5.3801E-05	0.9999992	0.9408
38	1.3101E-03	3.5542E-07	5.5688	5.1304E-05	0.9999993	0.9448
39	1.2664E-03	3.3684E-07	5.5688	4.8955E-05	0.9999994	0.9477
40	1.2250E-03	3.1996E-07	5.5688	4.6736E-05	0.9999995	0.9499
41	1.1859E-03	3.0442E-07	5.5688	4.4634E-05	0.9999995	0.9515
42	1.1488E-03	2.8999E-07	5.5688	4.2621E-05	0.9999996	0.9526
43	1.1137E-03	2.7648E-07	5.5688	4.0739E-05	0.9999996	0.9534
44	1.0805E-03	2.6375E-07	5.5688	3.8899E-05	0.9999997	0.9540
45	1.0491E-03	2.5172E-07	5.5688	3.7178E-05	0.9999997	0.9544
46	1.0195E-03	2.4031E-07	5.5688	3.5520E-05	0.9999997	0.9547
47	9.9160E-04	2.2947E-07	5.5688	3.3942E-05	0.9999997	0.9549
48	9.6527E-04	2.1916E-07	5.5688	3.2430E-05	0.9999998	0.9550
49	9.4046E-04	2.0933E-07	5.5688	3.0987E-05	0.9999998	0.9552
50	9.1710E-04	1.9996E-07	5.5688	2.9618E-05	0.9999998	0.9552
51	8.9513E-04	1.9101E-07	5.5688	2.8278E-05	0.9999998	0.9553
52	8.7446E-04	1.8248E-07	5.5688	2.7041E-05	0.9999998	0.9553
53	8.5505E-04	1.7433E-07	5.5688	2.5828E-05	0.9999998	0.9553
54	8.3683E-04	1.6655E-07	5.5688	2.4671E-05	0.9999998	0.9554
55	8.1974E-04	1.5912E-07	5.5688	2.3577E-05	0.9999998	0.9554
56	8.0371E-04	1.5202E-07	5.5688	2.2524E-05	0.9999998	0.9554
57	7.8869E-04	1.4524E-07	5.5688	2.1523E-05	0.9999999	0.9554
58	7.7462E-04	1.3876E-07	5.5688	2.0574E-05	0.9999999	0.9554
59	7.6146E-04	1.3257E-07	5.5688	1.9647E-05	0.9999999	0.9554
60	7.4915E-04	1.2666E-07	5.5688	1.8768E-05	0.9999999	0.9554
61	7.3763E-04	1.2101E-07	5.5688	1.7945E-05	0.9999999	0.9554
62	7.2688E-04	1.1562E-07	5.5688	1.7132E-05	0.9999999	0.9554
63	7.1684E-04	1.1046E-07	5.5688	1.6372E-05	0.9999999	0.9554
64	7.0747E-04	1.0554E-07	5.5688	1.5649E-05	0.9999999	0.9554

Continued on next page

Table A.54 – *Continued from previous page*

	levels: 4W, $N_f = 168 \times 168 \times 168$, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$ $\epsilon_{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{coarse} = \epsilon_{fine}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
65	6.9872E-04	1.0083E-07	5.5688	1.4942E-05	0.9999999	0.9554
66	6.9057E-04	9.6331E-08	5.5688	1.4276E-05	0.9999999	0.9554
67	6.8296E-04	9.2037E-08	5.5688	1.3646E-05	0.9999999	0.9554
68	6.7587E-04	8.7931E-08	5.5688	1.3033E-05	0.9999999	0.9554
69	6.6926E-04	8.4012E-08	5.5688	1.2460E-05	0.9999999	0.9554
70	6.6311E-04	8.0264E-08	5.5688	1.1895E-05	0.9999999	0.9554
71	6.5737E-04	7.6687E-08	5.5688	1.1375E-05	0.9999999	0.9554
72	6.5203E-04	7.3265E-08	5.5688	1.0859E-05	0.9999999	0.9554
73	6.4706E-04	6.9996E-08	5.5688	1.0372E-05	0.9999999	0.9554
74	6.4243E-04	6.6876E-08	5.5688	9.9196E-06	0.9999999	0.9554
75	6.3812E-04	6.3892E-08	5.5688	9.4710E-06	0.9999999	0.9554
76	6.3410E-04	6.1043E-08	5.5688	9.0524E-06	0.9999999	0.9554
77	6.3037E-04	5.8318E-08	5.5688	8.6440E-06	0.9999999	0.9554
78	6.2688E-04	5.5717E-08	5.5688	8.2643E-06	0.9999999	0.9554
79	6.2364E-04	5.3230E-08	5.5688	7.8927E-06	1.0000000	0.9554
80	6.2062E-04	5.0855E-08	5.5688	7.5458E-06	1.0000000	0.9554
81	6.1781E-04	4.8584E-08	5.5688	7.2067E-06	1.0000000	0.9553
82	6.1519E-04	4.6409E-08	5.5688	6.8749E-06	1.0000000	0.9552
83	6.1275E-04	4.4338E-08	5.5688	6.5845E-06	1.0000000	0.9554
84	6.1047E-04	4.2353E-08	5.5688	6.2849E-06	1.0000000	0.9552
85	6.0835E-04	4.0456E-08	5.5689	6.0045E-06	1.0000000	0.9552
86	6.0636E-04	3.8640E-08	5.5689	5.7384E-06	1.0000000	0.9551
87	6.0451E-04	3.6908E-08	5.5689	5.5073E-06	1.0000000	0.9552
88	6.0276E-04	3.5252E-08	5.5689	5.2904E-06	1.0000000	0.9551
89	6.0110E-04	3.3685E-08	5.5689	5.1336E-06	1.0000000	0.9556
90	5.9950E-04	3.2226E-08	5.5689	5.0653E-06	1.0000000	0.9567
91	5.9790E-04	3.0931E-08	5.5689	5.1469E-06	1.0000000	0.9598
92	5.9642E-04	2.9700E-08	5.5689	4.9151E-06	1.0000000	0.9602
93	5.9523E-04	2.8300E-08	5.5689	4.0906E-06	1.0000000	0.9529
94	5.9408E-04	2.7052E-08	5.5689	4.0269E-06	1.0000000	0.9559
95	5.9302E-04	2.5832E-08	5.5689	3.8085E-06	1.0000000	0.9549
96	5.9203E-04	2.4677E-08	5.5689	3.6466E-06	1.0000000	0.9553
97	5.9110E-04	2.3569E-08	5.5689	3.4728E-06	1.0000000	0.9551

Continued on next page

Table A.54 – *Continued from previous page*

	levels: 4W, $N_f = 168 \times 168 \times 168$, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$ $\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$					
k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
98	5.9023E-04	2.2505E-08	5.5689	3.3231E-06	1.0000000	0.9549
99	5.8941E-04	2.1496E-08	5.5689	3.1699E-06	1.0000000	0.9551
100	5.8865E-04	2.0535E-08	5.5689	3.0346E-06	1.0000000	0.9553
101	5.8793E-04	1.9617E-08	5.5689	2.8972E-06	1.0000000	0.9553
102	5.8726E-04	1.8728E-08	5.5689	2.7617E-06	1.0000000	0.9547
103	5.8664E-04	1.7881E-08	5.5689	2.6295E-06	1.0000000	0.9548
104	5.8604E-04	1.7082E-08	5.5689	2.5247E-06	1.0000000	0.9553
105	5.8549E-04	1.6319E-08	5.5689	2.4103E-06	1.0000000	0.9553
106	5.8497E-04	1.5576E-08	5.5689	2.2965E-06	1.0000000	0.9545
convergence after 107 Newton–multigrid cycles mean convergence rate: 0.912 wall clock time: 10360 sec						

A.7 Test Case 7 — Parallel Nonlinear Multigrid

The problem that we are solving is

$$\begin{aligned} -\nabla\cdot((10+u(\boldsymbol{x}))\nabla u(\boldsymbol{x}))+(1000+u(\boldsymbol{x})) = 1000 &- \sin^2(x)\sin^2(y)\cos^2(z)+\sin(x)\sin(y)\sin(z) \\ &- \sin^2(x)\cos^2(y)\sin^2(z)-\cos^2(x)\sin^2(y)\sin^2(z) \\ &+ 3\sin(x)\sin(y)\sin(z)(\sin(x)\sin(y)\sin(z)+10) \end{aligned} \tag{A.19}$$

with the boundary conditions

$$\begin{aligned} \frac{\partial u}{\partial x}(0,y,z) &= \frac{\partial u}{\partial x}(2\pi,y,z) = \sin(y)\sin(z), \\ u(x,0,z) &= u(x,2\pi,z), \\ u(x,y,0) &= u(x,y,2\pi) \end{aligned} \tag{A.20}$$

and the exact solution

$$u(x,y,z) = \sin(x)\,\sin(y)\,\sin(z). \tag{A.21}$$

Table A.55: Test case 7 — Results of section 4.7

levels: 3W, $N_f = 84 \times 84 \times 84$ on $2 \times 2 \times 2$ procs, $h_x = 7.48 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	9.9278E-02	0.0000			
1	2.6423E-01	9.0165E-03	5.6339	5.6339E+00	0.0000000	0.0908
2	1.2749E-01	2.5011E-03	5.5135	2.1996E-01	1.0218306	0.2774
3	8.1870E-02	1.6150E-03	5.5357	4.6437E-02	0.9959903	0.6457
4	5.3478E-02	1.0399E-03	5.5478	2.8866E-02	0.9978342	0.6439
5	3.5488E-02	6.7003E-04	5.5556	1.8616E-02	0.9985818	0.6443
6	2.4278E-02	4.3185E-04	5.5607	1.1999E-02	0.9990810	0.6445
7	1.7467E-02	2.7849E-04	5.5641	7.7433E-03	0.9994047	0.6449
8	1.3450E-02	1.7975E-04	5.5662	5.0092E-03	0.9996143	0.6455
9	1.1113E-02	1.1623E-04	5.5676	3.2576E-03	0.9997498	0.6466
10	9.7126E-03	7.5436E-05	5.5685	2.1416E-03	0.9998373	0.6490
11	8.7973E-03	4.9317E-05	5.5691	1.4380E-03	0.9998939	0.6538
12	8.1270E-03	3.2703E-05	5.5695	1.0024E-03	0.9999305	0.6631
13	7.5845E-03	2.2260E-05	5.5697	7.3928E-04	0.9999541	0.6807
14	7.1153E-03	1.5817E-05	5.5699	5.8360E-04	0.9999694	0.7106
15	6.6943E-03	1.1929E-05	5.5700	4.9078E-04	0.9999793	0.7542
16	6.3102E-03	9.6104E-06	5.5701	4.3216E-04	0.9999858	0.8056
17	5.9571E-03	8.1953E-06	5.5702	3.9125E-04	0.9999900	0.8528
18	5.6318E-03	7.2696E-06	5.5702	3.5946E-04	0.9999928	0.8870
19	5.3320E-03	6.6007E-06	5.5702	3.3268E-04	0.9999946	0.9080
20	5.0560E-03	6.0687E-06	5.5703	3.0894E-04	0.9999959	0.9194
21	4.8022E-03	5.6153E-06	5.5703	2.8734E-04	0.9999967	0.9253
22	4.5692E-03	5.2121E-06	5.5703	2.6742E-04	0.9999973	0.9282
23	4.3557E-03	4.8452E-06	5.5703	2.4895E-04	0.9999977	0.9296
24	4.1604E-03	4.5076E-06	5.5703	2.3178E-04	0.9999980	0.9303
25	3.9820E-03	4.1950E-06	5.5703	2.1579E-04	0.9999983	0.9306
26	3.8193E-03	3.9047E-06	5.5703	2.0090E-04	0.9999984	0.9308
27	3.6713E-03	3.6349E-06	5.5703	1.8704E-04	0.9999986	0.9309
28	3.5367E-03	3.3838E-06	5.5703	1.7413E-04	0.9999987	0.9309
29	3.4147E-03	3.1501E-06	5.5703	1.6210E-04	0.9999988	0.9309
30	3.3041E-03	2.9327E-06	5.5704	1.5092E-04	0.9999989	0.9310
31	3.2042E-03	2.7302E-06	5.5704	1.4049E-04	0.9999990	0.9310

Continued on next page

Table A.55 – *Continued from previous page*

levels: 3W, $N_f = 84 \times 84 \times 84$ on $2 \times 2 \times 2$ procs, $h_x = 7.48 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
32	3.1140E-03	2.5417E-06	5.5704	1.3079E-04	0.9999991	0.9310
33	3.0326E-03	2.3662E-06	5.5704	1.2175E-04	0.9999991	0.9310
34	2.9594E-03	2.2028E-06	5.5704	1.1334E-04	0.9999992	0.9310
35	2.8936E-03	2.0507E-06	5.5704	1.0552E-04	0.9999992	0.9310
36	2.8344E-03	1.9091E-06	5.5704	9.8226E-05	0.9999993	0.9310
37	2.7814E-03	1.7773E-06	5.5704	9.1441E-05	0.9999993	0.9310
38	2.7338E-03	1.6546E-06	5.5704	8.5135E-05	0.9999994	0.9310
39	2.6911E-03	1.5404E-06	5.5704	7.9245E-05	0.9999994	0.9309
40	2.6530E-03	1.4340E-06	5.5704	7.3771E-05	0.9999995	0.9309
41	2.6188E-03	1.3350E-06	5.5704	6.8675E-05	0.9999995	0.9309
42	2.5882E-03	1.2428E-06	5.5704	6.3930E-05	0.9999995	0.9309
43	2.5609E-03	1.1570E-06	5.5704	5.9524E-05	0.9999996	0.9310
44	2.5364E-03	1.0771E-06	5.5704	5.5403E-05	0.9999996	0.9309
45	2.5145E-03	1.0027E-06	5.5704	5.1576E-05	0.9999996	0.9309
46	2.4949E-03	9.3346E-07	5.5704	4.8012E-05	0.9999997	0.9309
47	2.4774E-03	8.6903E-07	5.5704	4.4706E-05	0.9999997	0.9310
48	2.4616E-03	8.0900E-07	5.5704	4.1608E-05	0.9999997	0.9309
49	2.4475E-03	7.5312E-07	5.5704	3.8733E-05	0.9999997	0.9309
50	2.4349E-03	7.0114E-07	5.5704	3.6067E-05	0.9999997	0.9310
51	2.4236E-03	6.5270E-07	5.5704	3.3566E-05	0.9999998	0.9309
52	2.4134E-03	6.0765E-07	5.5704	3.1257E-05	0.9999998	0.9310
53	2.4042E-03	5.6567E-07	5.5704	2.9088E-05	0.9999998	0.9309
54	2.3960E-03	5.2663E-07	5.5704	2.7089E-05	0.9999998	0.9310
55	2.3886E-03	4.9024E-07	5.5704	2.5207E-05	0.9999998	0.9309
56	2.3819E-03	4.5640E-07	5.5704	2.3475E-05	0.9999998	0.9310
57	2.3759E-03	4.2486E-07	5.5704	2.1843E-05	0.9999998	0.9309
58	2.3704E-03	3.9553E-07	5.5704	2.0343E-05	0.9999999	0.9310
59	2.3655E-03	3.6823E-07	5.5704	1.8938E-05	0.9999999	0.9310
60	2.3610E-03	3.4276E-07	5.5704	1.7619E-05	0.9999999	0.9309
61	2.3570E-03	3.1911E-07	5.5704	1.6411E-05	0.9999999	0.9310
62	2.3533E-03	2.9707E-07	5.5704	1.5275E-05	0.9999999	0.9309
63	2.3500E-03	2.7655E-07	5.5704	1.4220E-05	0.9999999	0.9310
64	2.3470E-03	2.5746E-07	5.5704	1.3237E-05	0.9999999	0.9309

Continued on next page

Table A.55 – *Continued from previous page*

levels: 3W, $N_\mathrm{f} = 84 \times 84 \times 84$ on $2 \times 2 \times 2$ procs, $h_x = 7.48 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_\mathrm{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_\mathrm{coarse} = \epsilon_\mathrm{fine}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
65	2.3442E-03	2.3970E-07	5.5704	1.2330E-05	0.9999999	0.9310
66	2.3417E-03	2.2313E-07	5.5704	1.1470E-05	0.9999999	0.9309
67	2.3394E-03	2.0772E-07	5.5704	1.0676E-05	0.9999999	0.9309
68	2.3373E-03	1.9338E-07	5.5704	9.9446E-06	0.9999999	0.9310
69	2.3354E-03	1.8004E-07	5.5704	9.2586E-06	0.9999999	0.9310
70	2.3337E-03	1.6760E-07	5.5704	8.6144E-06	0.9999999	0.9309
71	2.3321E-03	1.5603E-07	5.5704	8.0209E-06	0.9999999	0.9310
72	2.3306E-03	1.4527E-07	5.5704	7.4705E-06	1.0000000	0.9310
73	2.3293E-03	1.3522E-07	5.5704	6.9454E-06	1.0000000	0.9308
74	2.3281E-03	1.2592E-07	5.5704	6.4787E-06	1.0000000	0.9312
75	2.3269E-03	1.1723E-07	5.5704	6.0251E-06	1.0000000	0.9310
76	2.3259E-03	1.0913E-07	5.5704	5.6048E-06	1.0000000	0.9309
77	2.3250E-03	1.0163E-07	5.5704	5.2223E-06	1.0000000	0.9313

convergence after 78 Newton–multigrid cycles

mean convergence rate: 0.875

wall clock time: 211 sec

Table A.56: Test case 7 — Results of section 4.1.2.1

levels: 4W, $N_f = 168 \times 168 \times 168$ on $2 \times 2 \times 2$ procs, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
0	5.5683E+00	1.5041E-02	0.0000			
1	2.3238E-01	3.2868E-03	5.6844	5.6844E+00	0.0000000	0.2185
2	9.2399E-02	3.4358E-04	5.5259	2.3436E-01	1.0286924	0.1045
3	7.2854E-02	2.5736E-04	5.5379	2.0610E-02	0.9978307	0.7491
4	5.8671E-02	2.0657E-04	5.5437	1.4257E-02	0.9989416	0.8026
5	4.7269E-02	1.6593E-04	5.5486	1.1493E-02	0.9991260	0.8033
6	3.8141E-02	1.3329E-04	5.5525	9.2309E-03	0.9992961	0.8033
7	3.0843E-02	1.0709E-04	5.5557	7.4158E-03	0.9994329	0.8034
8	2.5018E-02	8.6047E-05	5.5582	5.9582E-03	0.9995433	0.8035
9	2.0382E-02	6.9144E-05	5.5602	4.7876E-03	0.9996322	0.8036
10	1.6706E-02	5.5566E-05	5.5619	3.8475E-03	0.9997039	0.8036
11	1.3804E-02	4.4659E-05	5.5632	3.0927E-03	0.9997616	0.8037
12	1.1528E-02	3.5897E-05	5.5643	2.4866E-03	0.9998081	0.8038
13	9.7540E-03	2.8858E-05	5.5651	2.0000E-03	0.9998454	0.8039
14	8.3813E-03	2.3204E-05	5.5658	1.6097E-03	0.9998755	0.8041
15	7.3259E-03	1.8662E-05	5.5664	1.2966E-03	0.9998997	0.8043
16	6.5174E-03	1.5015E-05	5.5668	1.0457E-03	0.9999192	0.8046
17	5.8975E-03	1.2087E-05	5.5672	8.4493E-04	0.9999348	0.8050
18	5.4187E-03	9.7363E-06	5.5675	6.8449E-04	0.9999474	0.8056
19	5.0435E-03	7.8511E-06	5.5677	5.5656E-04	0.9999575	0.8064
20	4.7431E-03	6.3400E-06	5.5679	4.5488E-04	0.9999657	0.8075
21	4.4962E-03	5.1302E-06	5.5681	3.7438E-04	0.9999722	0.8092
22	4.2874E-03	4.1630E-06	5.5682	3.1098E-04	0.9999775	0.8115
23	4.1058E-03	3.3914E-06	5.5683	2.6134E-04	0.9999818	0.8147
24	3.9438E-03	2.7776E-06	5.5684	2.2273E-04	0.9999852	0.8190
25	3.7963E-03	2.2912E-06	5.5685	1.9291E-04	0.9999879	0.8249
26	3.6598E-03	1.9074E-06	5.5685	1.6997E-04	0.9999901	0.8325
27	3.5318E-03	1.6064E-06	5.5686	1.5234E-04	0.9999919	0.8422
28	3.4108E-03	1.3715E-06	5.5686	1.3873E-04	0.9999934	0.8538
29	3.2956E-03	1.1893E-06	5.5686	1.2810E-04	0.9999945	0.8671
30	3.1854E-03	1.0485E-06	5.5686	1.1967E-04	0.9999955	0.8816
31	3.0798E-03	9.3956E-07	5.5687	1.1279E-04	0.9999962	0.8961

Continued on next page

Table A.56 – *Continued from previous page*

levels: 4W, $N_f = 168 \times 168 \times 168$ on $2 \times 2 \times 2$ procs, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
32	2.9783E-03	8.5497E-07	5.5687	1.0706E-04	0.9999969	0.9100
33	2.8807E-03	7.8852E-07	5.5687	1.0210E-04	0.9999974	0.9223
34	2.7866E-03	7.3544E-07	5.5687	9.7755E-05	0.9999978	0.9327
35	2.6961E-03	6.9208E-07	5.5687	9.3806E-05	0.9999981	0.9410
36	2.6089E-03	6.5575E-07	5.5687	9.0187E-05	0.9999984	0.9475
37	2.5248E-03	6.2453E-07	5.5687	8.6813E-05	0.9999986	0.9524
38	2.4438E-03	5.9704E-07	5.5687	8.3647E-05	0.9999988	0.9560
39	2.3658E-03	5.7229E-07	5.5688	8.0617E-05	0.9999989	0.9586
40	2.2906E-03	5.4964E-07	5.5688	7.7736E-05	0.9999990	0.9604
41	2.2183E-03	5.2860E-07	5.5688	7.4976E-05	0.9999991	0.9617
42	2.1486E-03	5.0885E-07	5.5688	7.2325E-05	0.9999992	0.9626
43	2.0814E-03	4.9018E-07	5.5688	6.9788E-05	0.9999993	0.9633
44	2.0168E-03	4.7241E-07	5.5688	6.7319E-05	0.9999993	0.9637
45	1.9547E-03	4.5543E-07	5.5688	6.4951E-05	0.9999994	0.9641
46	1.8948E-03	4.3916E-07	5.5688	6.2667E-05	0.9999994	0.9643
47	1.8373E-03	4.2355E-07	5.5688	6.0464E-05	0.9999995	0.9644
48	1.7819E-03	4.0853E-07	5.5688	5.8337E-05	0.9999995	0.9645
49	1.7287E-03	3.9409E-07	5.5688	5.6300E-05	0.9999995	0.9646
50	1.6775E-03	3.8016E-07	5.5688	5.4308E-05	0.9999996	0.9647
51	1.6284E-03	3.6674E-07	5.5688	5.2397E-05	0.9999996	0.9647
52	1.5811E-03	3.5380E-07	5.5688	5.0554E-05	0.9999996	0.9647
53	1.5357E-03	3.4133E-07	5.5688	4.8775E-05	0.9999996	0.9647
54	1.4921E-03	3.2931E-07	5.5688	4.7072E-05	0.9999996	0.9648
55	1.4502E-03	3.1771E-07	5.5688	4.5403E-05	0.9999997	0.9648
56	1.4100E-03	3.0651E-07	5.5688	4.3805E-05	0.9999997	0.9648
57	1.3715E-03	2.9571E-07	5.5688	4.2262E-05	0.9999997	0.9648
58	1.3345E-03	2.8530E-07	5.5688	4.0775E-05	0.9999997	0.9648
59	1.2990E-03	2.7525E-07	5.5688	3.9339E-05	0.9999997	0.9648
60	1.2650E-03	2.6556E-07	5.5688	3.7966E-05	0.9999997	0.9648
61	1.2324E-03	2.5621E-07	5.5688	3.6618E-05	0.9999997	0.9648
62	1.2012E-03	2.4718E-07	5.5688	3.5329E-05	0.9999997	0.9648
63	1.1713E-03	2.3848E-07	5.5688	3.4083E-05	0.9999998	0.9648
64	1.1427E-03	2.3008E-07	5.5688	3.2883E-05	0.9999998	0.9648

Continued on next page

Table A.56 – *Continued from previous page*

levels: 4W, $N_f = 168 \times 168 \times 168$ on $2 \times 2 \times 2$ procs, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
65	1.1153E-03	2.2197E-07	5.5688	3.1724E-05	0.9999998	0.9648
66	1.0890E-03	2.1416E-07	5.5688	3.0620E-05	0.9999998	0.9648
67	1.0640E-03	2.0661E-07	5.5688	2.9529E-05	0.9999998	0.9648
68	1.0400E-03	1.9934E-07	5.5688	2.8490E-05	0.9999998	0.9648
69	1.0171E-03	1.9231E-07	5.5688	2.7486E-05	0.9999998	0.9648
70	9.9525E-04	1.8554E-07	5.5688	2.6517E-05	0.9999998	0.9648
71	9.7436E-04	1.7901E-07	5.5688	2.5596E-05	0.9999998	0.9648
72	9.5444E-04	1.7270E-07	5.5688	2.4682E-05	0.9999998	0.9647
73	9.3543E-04	1.6662E-07	5.5688	2.3813E-05	0.9999998	0.9648
74	9.1730E-04	1.6074E-07	5.5688	2.2973E-05	0.9999998	0.9648
75	9.0002E-04	1.5508E-07	5.5688	2.2164E-05	0.9999998	0.9648
76	8.8355E-04	1.4963E-07	5.5688	2.1396E-05	0.9999998	0.9648
77	8.6787E-04	1.4435E-07	5.5688	2.0631E-05	0.9999999	0.9647
78	8.5294E-04	1.3926E-07	5.5688	1.9903E-05	0.9999999	0.9648
79	8.3873E-04	1.3435E-07	5.5688	1.9201E-05	0.9999999	0.9648
80	8.2521E-04	1.2963E-07	5.5688	1.8539E-05	0.9999999	0.9648
81	8.1236E-04	1.2506E-07	5.5688	1.7873E-05	0.9999999	0.9647
82	8.0014E-04	1.2065E-07	5.5688	1.7244E-05	0.9999999	0.9648
83	7.8854E-04	1.1640E-07	5.5688	1.6635E-05	0.9999999	0.9647
84	7.7751E-04	1.1230E-07	5.5688	1.6062E-05	0.9999999	0.9648
85	7.6704E-04	1.0834E-07	5.5688	1.5486E-05	0.9999999	0.9647
86	7.5710E-04	1.0452E-07	5.5688	1.4938E-05	0.9999999	0.9647
87	7.4768E-04	1.0084E-07	5.5688	1.4411E-05	0.9999999	0.9647
88	7.3872E-04	9.7292E-08	5.5688	1.3918E-05	0.9999999	0.9649
89	7.3024E-04	9.3858E-08	5.5688	1.3415E-05	0.9999999	0.9647
90	7.2220E-04	9.0547E-08	5.5688	1.2941E-05	0.9999999	0.9647
91	7.1458E-04	8.7355E-08	5.5688	1.2487E-05	0.9999999	0.9647
92	7.0735E-04	8.4284E-08	5.5688	1.2058E-05	0.9999999	0.9648
93	7.0051E-04	8.1308E-08	5.5688	1.1621E-05	0.9999999	0.9647
94	6.9404E-04	7.8437E-08	5.5688	1.1209E-05	0.9999999	0.9647
95	6.8789E-04	7.5681E-08	5.5688	1.0829E-05	0.9999999	0.9649
96	6.8208E-04	7.3010E-08	5.5688	1.0438E-05	0.9999999	0.9647
97	6.7659E-04	7.0432E-08	5.5688	1.0066E-05	0.9999999	0.9647

Continued on next page

Table A.56 – *Continued from previous page*

levels: 4W, $N_f = 168 \times 168 \times 168$ on $2 \times 2 \times 2$ procs, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_{\text{fine}} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_{\text{coarse}} = \epsilon_{\text{fine}}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
98	6.7138E-04	6.7957E-08	5.5688	9.7252E-06	0.9999999	0.9649
99	6.6645E-04	6.5557E-08	5.5688	9.3712E-06	0.9999999	0.9647
100	6.6179E-04	6.3242E-08	5.5688	9.0387E-06	0.9999999	0.9647
101	6.5737E-04	6.1022E-08	5.5688	8.7367E-06	0.9999999	0.9649
102	6.5320E-04	5.8864E-08	5.5688	8.4129E-06	0.9999999	0.9646
103	6.4926E-04	5.6784E-08	5.5688	8.1158E-06	0.9999999	0.9647
104	6.4552E-04	5.4790E-08	5.5688	7.8438E-06	0.9999999	0.9649
105	6.4198E-04	5.2853E-08	5.5688	7.5565E-06	1.0000000	0.9647
106	6.3864E-04	5.0996E-08	5.5688	7.3009E-06	1.0000000	0.9649
107	6.3548E-04	4.9191E-08	5.5688	7.0306E-06	1.0000000	0.9646
108	6.3248E-04	4.7462E-08	5.5688	6.7957E-06	1.0000000	0.9649
109	6.2965E-04	4.5784E-08	5.5688	6.5468E-06	1.0000000	0.9646
110	6.2697E-04	4.4163E-08	5.5688	6.3112E-06	1.0000000	0.9646
111	6.2444E-04	4.2615E-08	5.5688	6.1058E-06	1.0000000	0.9649
112	6.2204E-04	4.1105E-08	5.5688	5.8759E-06	1.0000000	0.9646
113	6.1977E-04	3.9661E-08	5.5688	5.6810E-06	1.0000000	0.9649
114	6.1762E-04	3.8252E-08	5.5688	5.4642E-06	1.0000000	0.9645
115	6.1559E-04	3.6911E-08	5.5688	5.2893E-06	1.0000000	0.9649
116	6.1366E-04	3.5603E-08	5.5688	5.0911E-06	1.0000000	0.9646
117	6.1184E-04	3.4352E-08	5.5688	4.9214E-06	1.0000000	0.9649
118	6.1011E-04	3.3134E-08	5.5688	4.7382E-06	1.0000000	0.9646
119	6.0847E-04	3.1970E-08	5.5688	4.5813E-06	1.0000000	0.9649
120	6.0693E-04	3.0837E-08	5.5689	4.4101E-06	1.0000000	0.9646
121	6.0546E-04	2.9750E-08	5.5689	4.2597E-06	1.0000000	0.9647
122	6.0407E-04	2.8706E-08	5.5689	4.1168E-06	1.0000000	0.9649
123	6.0275E-04	2.7687E-08	5.5689	3.9603E-06	1.0000000	0.9645
124	6.0150E-04	2.6705E-08	5.5689	3.8183E-06	1.0000000	0.9645
125	6.0031E-04	2.5771E-08	5.5689	3.6993E-06	1.0000000	0.9650
126	5.9919E-04	2.4857E-08	5.5689	3.5570E-06	1.0000000	0.9645
127	5.9812E-04	2.3974E-08	5.5689	3.4286E-06	1.0000000	0.9645
128	5.9711E-04	2.3130E-08	5.5689	3.3141E-06	1.0000000	0.9648
129	5.9615E-04	2.2316E-08	5.5689	3.2006E-06	1.0000000	0.9648
130	5.9524E-04	2.1530E-08	5.5689	3.0884E-06	1.0000000	0.9648

Continued on next page

Table A.56 – *Continued from previous page*

levels: 4W, $N_\mathrm{f} = 168 \times 168 \times 168$ on $2 \times 2 \times 2$ procs, $h_x = 3.74 \cdot 10^{-2}$, $\nu_1 = 5$, $\nu_2 = 5$
$\epsilon_\mathrm{fine} = \max(10^{-6}\|\boldsymbol{r}\|_2^{(0)}, 10^{-12})$, $\epsilon_\mathrm{coarse} = \epsilon_\mathrm{fine}/100$

k	$\|\boldsymbol{e}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}$	$\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}\|_2$	$\|\boldsymbol{u}\|_2^{(k-1)}/\|\boldsymbol{u}\|_2^{(k)}$	$\|\boldsymbol{r}\|_2^{(k)}/\|\boldsymbol{r}\|_2^{(k-1)}$
131	5.9437E-04	2.0767E-08	5.5689	2.9726E-06	1.0000000	0.9645
132	5.9355E-04	2.0030E-08	5.5689	2.8657E-06	1.0000000	0.9645
133	5.9276E-04	1.9332E-08	5.5689	2.7800E-06	1.0000000	0.9652
134	5.9202E-04	1.8648E-08	5.5689	2.6725E-06	1.0000000	0.9646
135	5.9132E-04	1.7991E-08	5.5689	2.5806E-06	1.0000000	0.9648
136	5.9065E-04	1.7346E-08	5.5689	2.4764E-06	1.0000000	0.9642
137	5.9001E-04	1.6746E-08	5.5689	2.4130E-06	1.0000000	0.9654
138	5.8940E-04	1.6146E-08	5.5689	2.3069E-06	1.0000000	0.9641
139	5.8882E-04	1.5582E-08	5.5689	2.2402E-06	1.0000000	0.9651

convergence after 140 Newton–multigrid cycles
mean convergence rate: 0.930
wall clock time: 3891 sec

BIBLIOGRAPHY

Anderson, J. (1995). *Computational Fluid Dynamics. The basics with applications*. McGraw-Hill.

Arnold, D. and A. Logg (2014). "Periodic Table of the Finite Elements". In: *SIAM News* 47.

Blies, P. M., F. Kupka, and H. J. Muthsam (2015). "The ANTARES Code: New Developments". In: *Astronomical Society of the Pacific Conference Series*. Ed. by N. V. Pogorelov, E. Audit, and G. P. Zank. Vol. 498, p. 191.

Boussinesq, J. V. (1903). *Théorie analytique de la chaleur*. Vol. Vol 2. Paris: Gathier-Villars.

Brenner, S. C. and L. R. Scott (2008). *The Mathematical Theory of Finite Element Methods*. Ed. by J. Marsden, L. Sirovich, and S. Antman. Springer New York.

Briggs, W. L., S. F. McCormick, et al. (2000). *A multigrid tutorial*. Siam.

Cohen, I. and P. Kundu (2002). *Fluid Mechanics*. Elsevier Science.

Elman, H. C., D. J. Silvester, and A. J. Wathen (2014). *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press (UK).

Gear, C. W. (1971). *Numerical initial value problems in ordinary differential equations*. Prentice Hall PTR.

Gottlieb, S., D. I. Ketcheson, and C.-W. Shu (2009). "High order strong stability preserving time discretizations". In: *Journal of Scientific Computing* 38.3, pp. 251–289.

Grimm-Strele, H., F. Kupka, and H. Muthsam (2014). "Curvilinear grids for {WENO} methods in astrophysical simulations". In: *Computer Physics Communications* 185.3, pp. 764–776.

Grimm-Strele, H. (2010). "Numerical Solution of the Generalised Poisson Equation on Parallel Computers". Master's Thesis - University of Vienna. MA thesis. University of Vienna.

Grimm-Strele, H. (2014). "Numerical Grids for Spherical Shells and Other Complex Domains". PhD thesis. University of Vienna.

Hager, G. and G. Wellein (2010). *Introduction to high performance computing for scientists and engineers*. CRC Press.

Hairer, E. (1993). *Solving ordinary differential equations II*. Vol. 2. Berlin New York: Springer-Verlag.

Hanke-Bourgeois, M. (2009). *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*.

Happenhofer, N. (2014). "Efficient Time Integration of the Governing Equations in Astrophysical Hydrodynamics". PhD thesis. University of Vienna.

Happenhofer, N., H. Grimm-Strele, F. Kupka, B. Löw-Baselli, and H. Muthsam (2013). "A Low Mach Number Solver: Enhancing Applicability". In: *Journal of Computational Physics* 236, pp. 96–118. arXiv: 1112.3507v1 [physics.comp-ph].

Higueras, I. (2006). "Strong stability for additive Runge-Kutta methods". In: *SIAM Journal on Numerical Analysis* 44.4, pp. 1735–1758.

Higueras, I. (2009). "Characterizing strong stability preserving additive runge-kutta methods". In: *Journal of Scientific Computing* 39.1, pp. 115–128.

Higueras, I., N. Happenhofer, O. Koch, and F. Kupka (2014). "Optimized strong stability preserving {IMEX} Runge–Kutta methods". In: *Journal of Computational and Applied Mathematics* 272, pp. 116–140.

Hoffman, J. (2001). *Numerical methods for engineers and scientists*. New York: Marcel Dekker.

Köckler, N. (2012). *Mehrgittermethoden*. Vieweg+Teubner Verlag.

Kraaijevanger, J. F. B. M. (1991). "Contractivity of Runge-Kutta methods". In: *BIT Numerical Mathematics* 31.3, pp. 482–528.

Kupka, F., N. Happenhofer, I. Higueras, and O. Koch (2012). "Total-variation-diminishing implicit-explicit Runge-Kutta methods for the simulation of double-diffusive convection in astrophysics". In: *Journal of Computational Physics* 231.9, pp. 3561–3586.

Leconte, J. and G. Chabrier (2012). "A new vision of giant planet interiors: Impact of double diffusive convection". In: *Astronomy & Astrophysics* 540, A20, A20. arXiv: 1201.4483 [astro-ph.EP].

Leconte, J. and G. Chabrier (2013). "Layered convection as the origin of Saturn's luminosity anomaly". In: *Nature Geoscience* 6, pp. 347–350. arXiv: 1304.6184 [astro-ph.EP].

LeVeque, R. J. (2007). *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. Vol. 98. Siam.

Mortensen, H. (2011). *Computational Physics*. lecture notes. lexture notes.

Mundprecht, E., H. J. Muthsam, and F. Kupka (2013). "Multidimensional realistic modelling of Cepheid-like variables - I. Extensions of the ANTARES code". In: *Mon. Not. R. Astron. Soc.* 435, pp. 3191–3205. arXiv: 1209.2952 [astro-ph.SR].

Muthsam, H. J. (2011). "Numerical Simulations of Solar and Stellar Convection using the ANTARES code". In: *EAS Publications Series*. Ed. by H. Wozniak and G. Hensler. Vol. 44. EAS Publications Series, pp. 97–100.

Muthsam, H. J., F. Kupka, E. Mundprecht, F. Zaussinger, H. Grimm-Strele, and N. Happenhofer (2011). "Simulations of stellar convection, pulsation and semiconvection". In: *IAU Symposium*. Ed. by N. H. Brummell, A. S. Brun, M. S. Miesch, and Y. Ponty. Vol. 271. IAU Symposium, pp. 179–186. arXiv: 1009.2409 [astro-ph.SR].

Muthsam, H. J., B. Löw-Baselli, C. Obertscheider, M. Langer, P. Lenz, and F. Kupka (2007). "High-resolution models of solar granulation: the two-dimensional case". In: *Mon. Not. R. Astron. Soc.* 380, pp. 1335–1340. arXiv: 0706.3349.

Muthsam, H., F. Kupka, B. Low-Baselli, C. Obertscheider, M. Langer, and P. Lenz (2010). "ANTARES – A Numerical Tool for Astrophysical RESearch with applications to solar granulation". In: *New Astronomy* 15.5, pp. 460–475.

Obertscheider, C. (2007). "Modelling of Solar Granulation: Implementation and Comparison of Numerical Schemes". PhD thesis. University of Vienna.

Pareschi, L. and G. Russo (2005). "Implicit-explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxation". In: *Journal of Scientific computing* 25.1-2, pp. 129–155.

Quarteroni, A. (2009). *Numerical Models for Differential Problems*. Springer-Verlag Mailand.

Quarteroni, A. and A. Valli (1994). *Numerical Approximation of Partial Differential Equations*. Springer Berlin Heidelberg.

Spiegel, E. A. and G. Veronis (1960). "On the Boussinesq Approximation for a Compressible Fluid." In: *The Astrophysical Journal* 131, p. 442.

Trottenberg, U., C. Oosterlee, and A. Schüller (2001). *Multigrid*. Academic Press.

Wesseling, P. (1992). *An Introduction to Multigrid Methods*. John Wiley & Sons, Ltd., Chinester.

Zaussinger, F. and H. C. Spruit (2013). "Semiconvection: numerical simulations". In: *Astronomy & Astrophysics* 554, A119, A119. arXiv: 1303.4522 [astro-ph.SR].

Zaussinger, F. (2010). "Numerical simulation of double-diffusive convection". PhD thesis. University of Vienna.