

DISSERTATION

Titel der Dissertation

Efficient Time Integration of the Governing Equations in Astrophysical Hydrodynamics

Verfasser

Mag. rer. nat. Natalie Happenhofer

Angestrebter akademischer Grad

Doktorin der Naturwissenschaften (Dr. rer. nat.)

Wien, im November 2013

Studienkennzahl: A 791 405
Dissertationsgebiet: Mathematik
Betreuer: Univ.-Prof. Dr. Herbert J. Muthsam

ABSTRACT

Many numerical hydrodynamics codes are based on explicit time integration methods and thus severely limited in their efficiency since for stability reasons, explicit time integration methods often enforce very small timesteps. The present work aims to circumvent this difficulty by using *strong-stability-preserving (SSP) implicit-explicit (IMEX) Runge-Kutta methods* for different simulation settings in the field of astrophysical hydrodynamics. In this study, the hydrodynamics code ANTARES is used.

It is shown that SSP IMEX schemes proposed in literature already permit the employment of significantly higher timesteps than traditional explicit time integration methods. Nevertheless, the use of SSP IMEX methods which have been developed such as not only to have a large radius of absolute monotonicity but also possess properties deemed beneficial in the simulations within the focus of this thesis surpass their counterparts by far. Furthermore, the comparison of different SSP IMEX schemes allows to draw conclusions which properties are the crucial ones for inferring the most stability on the simulations.

It is another challenge to transform the higher timesteps into a real efficiency gain in terms of CPU-time. This difficulty is met by the development of a numerical solver based on the multigrid technique to obtain fast and accurate solutions of the implicit stage equations. Since ANTARES is capable of running on up to 4000 CPU-cores in parallel, the solver is likewise able to exploit the advantages of highly parallel computing environments.

In simulations of double-diffusive convection, the implicit stage entails the solution of two linear generalized Poisson equations. It is shown that the newly devised linear multigrid solver converges much faster than a comparative solver based on the Schur-Complement parallelized Conjugate Gradient algorithm. This efficiency is especially tangible if large timesteps are employed. Furthermore, it is deduced that SSP IMEX schemes possessing the properties of positivity of the amplification factor and uniform convergence convey significantly more stability to simulations of compressible double-diffusive convection than SSP IMEX methods lacking those. In this setting, the best performing SSP IMEX scheme allows the use of a timestep 20.7 times higher than the traditional two-stage explicit method. The corresponding CPU-time is 6 times lower than its explicit counterpart if 64 CPU-cores are employed.

Similarly, the SSP IMEX RK methods have been incorporated into the model of pulsating stars. The moving grid algorithm has been adapted for the use of SSP IMEX Runge-Kutta schemes, furthermore, spherical coordinates are employed. Since a realistic microphysics model is used in this model, the implicit stage amounts to one nonlinear generalized Poisson equation. To obtain a solution to the implicit stage equation, a nonlinear solver based on a Multigrid-Newton technique especially tailored towards the ANTARES framework has been devised. In one-dimensional cepheid models, the use of SSP-IMEX Runge-Kutta methods allows the use of a timestep about 98 times higher than explicit methods, resulting in wallclocktimes about 24 times lower.

First attempts in two-dimensional simulations of pulsating stars are reported, however, a few issues remain which have to be resolved in the future.

ZUSAMMENFASSUNG

Viele CFD-Softwarepakete basieren auf expliziten Zeitintegrationsmethoden. Diese Verfahren sind einfach und schnell zu implementieren, bergen jedoch den Nachteil, aus Stabilitätsgründen nur sehr kleine Zeitschritte zuzulassen. Diese können unter Umständen die Effizienz der Software erheblich herabsetzen. In der vorliegenden Arbeit wird versucht, dieses Problem mit Hilfe von strong-stability-preserving (SSP) implizit-expliziten (IMEX) Runge-Kutta Verfahren zu umgehen und damit die Effizienz des CFD-Codes ANTARES entscheidend zu verbessern.

Es zeigt sich, dass in verschiedenen Arbeiten vorgestellte SSP IMEX Runge-Kutta Methoden bereits eine stabile Zeitintegration mit deutlich größeren Zeitschrittweiten ermöglichen. Speziell entwickelte SSP IMEX Methoden, welche nicht nur einen großen Radius absoluter Monotonizität, sondern auch Eigenschaften wie Positivität und gleichmäßige Konvergenz aufweisen, ermöglichen jedoch eine stabile Zeitintegration unter Verwendung noch einmal deutlich größerer Zeitschritte als jene SSP IMEX Runge-Kutta Verfahren aus der Literatur. Durch den Einsatz verschiedener SSP IMEX Methoden mit unterschiedlichen Eigenschaften lassen sich auch Rückschlüsse auf den Einfluss verschiedener Eigenschaften auf die Stabilität der Simulationen ziehen. Um jedoch tatsächlich nennenswerte Geschwindigkeitsvorteile gegenüber traditionellen, expliziten Verfahren zu erzielen, müssen die hohen Zeitschritte auch in eine entsprechende Reduktion der CPU-Zeit übersetzt werden. Besonderes Augenmerk liegt dabei auf der schnellen Lösung der impliziten Runge-Kutta Stufe.

Die SSP IMEX Methoden wurden mit Hilfe von Simulationen doppelt-diffusiver Konvektion auf ihre Effizienz getestet. In diesem Modell besteht die implizite Runge-Kutta Stufe aus der Lösung von zwei linearen, verallgemeinerten Poisson-Gleichungen. Zur näherungsweisen Lösung dieser Gleichungen wurde ein Löser basierend auf der Multigrid-Technik entwickelt, welcher auch im parallelen Modus operieren kann. Es wird gezeigt, dass dieser Löser deutlich schneller konvergiert als ein vergleichbares Verfahren, welchem der Schur-Komplement parallelisierte Algorithmus der konjugierten Gradienten zu Grunde liegt. Speziell der Vorteil großer Zeitschritte wird durch den Multigrid-basierten Löser in Effizienz übersetzt, während der große Overhead des Schur-Komplement basierten-Lösers in diesem Fall besonders spürbar ist.

Durch den Einsatz verschiedener Verfahren in der Simulation doppelt-diffusiver Konvektion kann man schließen, dass die Eigenschaften der Positivität und der gleichmäßigen Konvergenz stabilisierend auf die Zeitintegration wirken und große Zeitschritte verwendet werden können. Das effizienteste SSP-IMEX Verfahren erlaubt den Einsatz eines Zeitschrittes, der um einen Faktor 20.7 größer ist als der Zeitschritt, den ein explizites Verfahren verwenden kann. Die Simulation kann dadurch 6x schneller durchgeführt werden als mit expliziten Methoden.

Die SSP IMEX-Methoden wurden auch in Simulationen pulsierender Sterne getestet. Da ein nennenswerter Teil des Sternes simuliert wird, ist die Verwendung von sphärischen Koordinaten unabdinglich. Außerdem erlaubt ANTARES die Verwendung eines Gitters, welches der radialen Pulsationsbewegung des Sternes folgt. Dieser Algorithmus wurde für die Verwendung von SSP IMEX Runge-Kutta Verfahren entsprechend adaptiert. Zur realistischen Modellierung der mikrophysikalischen Vorgänge wird die tabellierte OPAL Zustandsgleichung verwendet. Daher besteht die implizite Stufe aus einer nicht-linearen verallgemeinerten Poissongleichung. Ein Multigrid-Newton Löser wurde zur Lösung dieser Gleichung entwickelt. Um höchstmögliche Effizienz zu gewährleisten, wurde dieser Löser speziell an den ANTARES-Code angepasst.

Eindimensionale Cepheidenmodelle konnten mit Hilfe von IMEX-Runge-Kutta Methoden um einen Faktor 24 beschleunigt werden. Der dabei verwendete Zeitschritt ist circa 98x größer als jeder, mit dem explizite Methoden arbeiten.

Auch in zwei räumlichen Dimensionen sollen IMEX Runge-Kutta Methoden die Simulation von pulsierenden Sternen entscheidend effizienter machen. Erste Ergebnisse werden in dieser Arbeit präsentiert, jedoch sind in diesem Zusammenhang noch verschiedene Punkte offen.

Contents

1	Introduction	1
2	The Equations of Radiation Hydrodynamics	5
2.0.1	Frames of Reference	5
2.1	Conservation of Mass	6
2.2	Conservation of Momentum	7
2.3	Conservation of Energy	8
2.4	The Equation of State (EOS)	9
2.5	Summary	9
3	The ANTARES-Code	11
3.1	Methodology	11
3.2	Simulations	12
4	Implicit-Explicit (IMEX) Runge-Kutta Methods	13
4.1	Introduction	13
4.2	Runge-Kutta methods	14
4.2.1	Stability Analysis	15
4.2.2	The strong-stability-preserving (SSP) property	17
4.2.3	Analysis of the discretization schemes in the context of the NSE and the ANTARES-Code	18
4.3	SSP IMEX Runge-Kutta methods	19
4.4	Optimized IMEX Runge-Kutta methods	22
4.5	Summary	24
4.6	Application to the Navier-Stokes equations	24
4.6.1	Timestep Restrictions	24
4.6.2	Implementation issues	27
5	Multigrid Methods	29
5.1	Introduction	29
5.2	The Multigrid Idea	30
5.3	Multiple grids	31
5.4	Discretization Issues	33

CONTENTS

5.5	Linear Multigrid: The Correction Scheme (CS)	33
5.5.1	Coarse grid operators	36
5.5.2	Cycling Pattern and Grid Hierarchy	37
5.5.3	Intergrid operators	38
5.5.4	Smoothing Routines	40
5.5.5	Direct Solvers	42
5.5.6	Comparison between the linear multigrid solver and a solver based on the Conjugate Gradient algorithm	44
5.6	Multigrid methods for a non-linear problem	45
5.6.1	The Full Approximation Storage (FAS) scheme	45
5.6.2	The Multigrid-Newton method	48
5.6.3	Comparison of the MG-FAS and the MG-Newton algorithm in view of the ANTARES framework	48
5.6.4	Adaptation of the MG-Newton technique	50
5.7	Parallelization issues	53
5.7.1	The necessity of parallel computing	53
5.7.2	Parallel Multigrid	56
6	Application to Double-Diffusive Convection	63
6.1	The Model	63
6.1.1	The semiconvection problem	63
6.1.2	Extension of the Navier-Stokes equations to two-component flows	63
6.1.3	Model Assumptions	64
6.1.4	Initial Condition and Boundary Conditions	65
6.2	Numerical simulation of double-diffusive convection	66
6.2.1	Simulation of double-diffusive convection	67
6.2.2	A fractional step approach	70
6.2.3	Timestep restrictions in simulations of double-diffusive convection	71
6.2.4	The merit of SSP IMEX Runge-Kutta methods	71
6.2.5	Multigrid Setting	75
6.3	Results and Discussion	79
7	Application to simulations of Cepheids	97
7.1	Cepheids	97
7.2	Numerical Simulations of Pulsating Stars	97
7.2.1	The model	98
7.2.2	The grid structure	100
7.2.3	SSP IMEX RK-Methods for the Cepheid Model	101
7.2.4	Multigrid Setting	105
7.3	Results	108
7.3.1	1D-Cepheid Models	108

8	Outlook	115
8.1	Application of SSP IMEX RK-methods to 2D Cepheid Models	115
8.1.1	Simulation Setting	115
8.1.2	First results	116
8.2	The Eddington Approximation	122
9	Conclusions	123
A		127
A.1	Gauss’s Divergence Theorem	127
A.2	Comoving regions	127
B		131
B.1	A Finite Element (FEM) discretization	131
C		137
C.1	Stencil Notation	137
	Curriculum Vitae	147

Chapter 1

Introduction

Astrophysical fluid flows are characterized by a vast range of possible timescales but the solution changes typically on the hydrodynamical timescale. Nevertheless, some of the physically important processes such as radiative transfer, soundwaves, magnetohydrodynamic processes, etc. often operate on much shorter timescales. This poses a severe difficulty if explicit time integration methods are employed where, for stability reasons, the timestep is set by the fastest operating process, regardless whether it effects significant changes in the solution or not. These small timesteps greatly reduce the efficiency of the simulations and may even render the computation of certain settings unfeasible.

As long as soundwaves are energetically and dynamically unimportant, a numerical solution can be advanced with much larger timesteps by using a fractional step method ([16]). Similarly, if radiative transfer has the characteristics of a stiff problem, the use of semi-implicit time integration methods is desirable as well.

In the present work, the advantages of *strong stability preserving (SSP) implicit-explicit (IMEX) Runge-Kutta methods* in simulations of double-diffusive convection and simulations of cepheids are investigated. The calculations are performed within the ANTARES framework ([22]). In both settings, the explicit timestep size is severely limited by radiative diffusion processes and thus, an implicit treatment of the corresponding terms is preferable. Apart from that, however, the settings differ drastically from each other.

The most important special case of double-diffusive convection in astrophysics is *semiconvection*. Models of stellar structure and evolution predict settings where the heavier product of nuclear fusion provides stability to a zone which otherwise would be unstable to convective overturning because temperature sufficiently rapidly decreases against the direction of gravity. Such a zone would become convective if its composition were mixed. The question whether such a zone should be treated as if it were uniform has become known as the *semiconvection* problem (see [55, 56] and Chapter 6). Numerical simulations are supposed to answer this question by providing realistic estimations of stellar mixing rates. In this setting, only a small section of the star is modeled and the spherical geometry is considered negligible (box-in-a-star approach). Although the geometry of the computational domain is thus relatively simple, the numerical schemes used must be able to

1. INTRODUCTION

smoothly capture the transition between the diffusion-dominated and the advection-dominated phase of the simulation.

Cepheids, on the other hand, belong to the class of *pulsating stars*, which in turn are a subset of *intrinsic variable stars*. In pulsating stars, large-scale dynamical motions, usually including the entire star, are present (see [35, 37]). Thus, to capture the radial pulsation in a simulation, a major part of the star must be included in the computational domain and its spherical shape must be accounted for. Spatial and temporal scales at top and bottom of the considered domain are vastly different and pose a great challenge to the numerical schemes employed.

Some SSP IMEX methods deemed suitable for the considered purposes have been proposed in the literature and have been analysed with respect to their radius of absolute monotonicity, stability and dissipativity properties in [29]. First results within the semiconvection setting have been promising (c.f. [29]), thus, in [20] (resp. [19]) new SSP IMEX schemes have been devised which have been optimized with respect to properties believed important in “real-world applications”. Within this work, all of those schemes will be employed and the properties enhancing the stability of the time integration (and thus achieving high timesteps) will be singled out.

Whereas the analytic properties of the SSP IMEX Runge-Kutta schemes are certainly important, special attention must be paid to the efficient evaluation of the implicit stage. In the applications outlined above, the implicit stage amounts to (a) generalized Poisson equation(s). The (non-) linearity of this equation depends heavily on the underlying model assumptions. In simulations of double-diffusive convection, for example, the implicit stage amounts to two linear generalized Poisson equations. In [29], first simulations of double-diffusive convection using SSP IMEX methods from the literature have been conducted. The implicit stage equations were solved using the solver developed by [11], which is based on the preconditioned Conjugate Gradient algorithm and parallelized using the Schur Complement technique. Although significantly higher timesteps (factor ~ 15.5) have been achieved by the IMEX methods compared to the timesteps employed in explicit schemes, the gain in computational efficiency was only minimal ($\sim$ factor 1.12). Thus, it is one objective of this work to replace this solver by a more efficient one based on a multigrid method. Furthermore, an advantage of the multigrid technique is that it is easily adapted for the solution of nonlinear equations as well. This is important since in simulations of cepheids (and many applications which may be considered in the future) the implicit stage equation is nonlinear.

This PhD-thesis is organized in the following way: In Chapter 2, the governing equations of radiation hydrodynamics in both an Eulerian and a semi-Lagrangian framework are introduced. Chapter 3 is dedicated to the ANTARES code [22].

In Chapter 4, Runge-Kutta methods and the concepts of stability and dissipativity analysis are reviewed. IMEX Runge-Kutta methods are introduced and the findings of [29] concerning the stability and dissipativity analysis of the schemes are reported. Furthermore, the construction principles of the optimized IMEX schemes (see [20] (resp. [19])) are outlined and the properties of the newly devised schemes summarized. The incorporation of IMEX methods into the ANTARES

framework is briefly sketched.

The developement of the (non-) linear, parallelized multigrid solver especially tailored to the ANTARES framework is discussed in detail in Chapter 5. Its basic functionality is assessed in a number of tests, and first comparisons with the SCpCG solver developed by [11] are given.

In Chapter 6, the use of IMEX Runge-Kutta methods in simulations of double-diffusive convection is explained in detail. The simulations have been performed using explicit time integrators, SSP IMEX methods from the literature and optimized SSP IMEX methods. Furthermore, the implicit stage equations are solved with both the SCpCG solver and the newly developed solver based on the multigrid technique. The performance of the different simulations with respect to the time integration method as well as the performance of the different solvers tackling the (linear) implicit stage equations is discussed.

The incorporation of IMEX Runge-Kutta methods into the cepheid model is disclosed in Chapter 7. The performance of the different time integrators is evaluated on the basis of one-dimensional simulations of cepheids. First experiments using two-dimensional domains in this setting are reported in Chapter 8 and an outlook on future developments and applications is given. The last chapter of this thesis summarizes the results obtained.

Chapter 2

The Equations of Radiation Hydrodynamics

The equations modeling fluid flow are derived from the principles of mass, momentum, and energy conservation. Close to the surface of a star, radiation becomes an important process for energy transport. The equations of hydrodynamics have then to be coupled to the equations describing radiative transfer.

The meaning of the symbols is explained in Table 2.1. All functions considered are assumed to be sufficiently differentiable as to perform the necessary operations appearing in the derivations.

2.0.1 Frames of Reference

There are basically two different approaches of stating the governing equations (see [41]). The continuum density, velocity and energy fields can be formulated as *Eulerian fields*, in that they are indexed by the position $\vec{x}$ in an inertial frame. The alternative formulation as *Lagrangian fields* starts by considering points that move with the local fluid velocity; $\xi(t, \vec{x}_0)$ denotes the position at time t of the fluid particle that is located at $\vec{x}_0$ at the specified fixed reference time t_0 . Within the Lagrangian approach the fluid particle position $\xi(t, \vec{x}_0)$ is defined by

$$\frac{\partial \xi(t, \vec{x}_0)}{\partial t} = \vec{u}(\xi(t, \vec{x}_0), t) \quad (2.0.1)$$

subject to the initial condition

$$\xi(0, \vec{x}_0) = \vec{x}_0 \quad (2.0.2)$$

The equation expresses the fact that the fluid particle moves with the local fluid velocity. Lagrangian fields are defined in terms of their Eulerian counterparts,

$$\rho^L(t, \vec{x}_0) = \rho(\xi(t, \vec{x}_0), t) \quad (2.0.3)$$

For fixed x_0 , $\xi(t, \vec{x}_0)$ defines a trajectory that is the *fluid-particle path*. The partial derivative of

2. THE EQUATIONS OF RADIATION HYDRODYNAMICS

equation (2.0.3) is given by

$$\frac{\partial}{\partial t} \rho^L(t, \vec{x}_0) = \frac{\partial}{\partial t} \rho(\xi(t, \vec{x}_0), t) = \quad (2.0.4)$$

$$= \left(\frac{\partial}{\partial t} \rho(\vec{x}, t) \right)_{\vec{x}=\xi(t, \vec{x}_0)} + \frac{\partial}{\partial t} \xi(t, \vec{x}_0) \cdot (\nabla \rho(\vec{x}, t))_{\vec{x}=\xi(t, \vec{x}_0)} \quad (2.0.5)$$

$$= \left(\frac{\partial}{\partial t} \rho(\vec{x}, t) + \vec{u} \cdot \nabla \rho(\vec{x}, t) \right)_{\vec{x}=\xi(t, \vec{x}_0)} \quad (2.0.6)$$

$$= \left(\frac{D}{Dt} \rho(\vec{x}, t) \right)_{\vec{x}=\xi(t, \vec{x}_0)} \quad (2.0.7)$$

which gives rise to the definition of the *material* or *substantial derivative*,

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + \vec{u} \cdot \nabla \quad (2.0.8)$$

Thus, the rate of change of density following a fluid particle is given by the partial derivative of the Lagrangian field (equation (2.0.4)) and by the substantial derivative of the Eulerian field (equation (2.0.7)).

For some simulations within the focus of this thesis, a semi-Lagrangian approach is used. In this approach, the grid moves with the fluid at a certain grid velocity $\vec{u}_g$, which does not necessarily equal the fluid velocity $\vec{u}$. The governing equations are formulated within an Eulerian framework where the moving grid is considered as the reference frame. With respect to the grid, the fluid moves with velocity $\vec{u}_{\text{rel}} = \vec{u} - \vec{u}_g$. Therefore, equation (2.0.1) reads

$$\frac{\partial \xi(t, \vec{x}_0)}{\partial t} = \vec{u}_{\text{rel}}(\xi(t, \vec{x}_0), t) \quad (2.0.9)$$

This leads to a modified appearance of the substantial derivative as well as to theorems where the substantial derivative is used in the derivation, for example the Reynolds transport theorem. Note that in case of a fixed grid, equation (2.0.9) reduces correctly to equation (2.0.1) since in this case, $\vec{u}_g = 0$ and $\vec{u}_{\text{rel}} = \vec{u}$.

In the following, the governing equations are derived within the moving grid approach. For a detailed derivation, the reader is referred to [1], [21], and [41].

2.1 Conservation of Mass

The principle of mass conservation states that mass in a comoving volume does not change with time,

$$\frac{d}{dt} \int_V \rho \, dV = 0. \quad (2.1.1)$$

Using Theorem 1.2.2 of Appendix A.1 this expression reads

$$\int_V \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}_{\text{rel}}) \right) dV = 0 \quad (2.1.2)$$

and since V was arbitrarily chosen,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}_{\text{rel}}) = 0. \quad (2.1.3)$$

Equation (2.1.3) is also known as the **continuity equation**.

2.2 Conservation of Momentum

The momentum equation is based on Newton's second law. It relates the fluid particle acceleration to the surface forces S_V and body forces B_V experienced by the fluid in the considered volume V :

$$\frac{d}{dt} \int_V \rho \vec{u} dV = S_W + B_W. \quad (2.2.1)$$

The body force of interest is gravity,

$$B_W = \int_V \rho \vec{g} dV. \quad (2.2.2)$$

The surface forces, which are of molecular origin, are described by the stress tensor σ which, in the present context, can be split up as

$$\sigma_{ij} = -P\delta_{ij} + \tau_{ij} \quad (2.2.3)$$

where $-P\delta_{ij}$ denotes the symmetric and τ_{ij} the deviatoric part of the stress tensor.

Using the transport theorem of Appendix A.1, one obtains

$$S_W = \int_{\partial V} \sigma dS = \int_V \nabla \cdot \sigma dV = - \int_V \nabla P dV + \int_V \nabla \cdot \tau dV. \quad (2.2.4)$$

Furthermore, equation (2.2.1) can be simplified to read

$$\int_V \frac{\partial(\rho \vec{u})}{\partial t} + \nabla \cdot (\rho \vec{u}_{\text{rel}} \otimes \vec{u}) dV = - \int_V \nabla P dV + \int_V \nabla \cdot \tau dV + \int_V \rho \vec{g} dV. \quad (2.2.5)$$

Again, V was arbitrary chosen. Thus, the integrals may be dropped to arrive at

$$\frac{\partial(\rho \vec{u})}{\partial t} + \nabla \cdot (\rho \vec{u}_{\text{rel}} \otimes \vec{u}) = -\nabla P + \nabla \cdot \tau + \rho \vec{g}. \quad (2.2.6)$$

To include the effects of radiation, the contribution of radiative pressure has to be added to the momentum balance. Usually, this is done by writing

$$P = P_{\text{gas}} + P_{\text{rad}}, \quad (2.2.7)$$

where P_{gas} is the pressure obtained from the equation of state.

2.3 Conservation of Energy

Energy exists in many forms. Here the two most basic ones are considered: the specific internal energy $e(x, t)$ and the specific kinetic energy $e_{\text{kin}}(x, t)$. The specific total energy is defined as the sum of both, $e_{\text{tot}} = e + e_{\text{kin}}$. By purpose, the potential energy is excluded in the definition of this “total” energy.

To formulate the conservation of energy, the work the exterior applies to the fluid (i.e. the work done by body and surface forces) is taken into account. Naturally, the surface and body forces are the same as in the derivation of the momentum equation in the previous section. Furthermore, in a star, a nontrivial amount of energy is transported by heat conduction and radiation, and energy is released by nuclear processes. Therefore, it is convenient to formulate the total energy density balance as

$$\frac{\partial}{\partial t} \int_V \rho e_{\text{tot}} dV = \int_V \overbrace{\rho \vec{g} \cdot \vec{u}}^{\text{work done by body forces}} dV + \int_{\partial V} \overbrace{\vec{\sigma} \cdot \vec{u}}^{\text{work done by surface forces}} dS + \quad (2.3.1)$$

$$\int_{\partial V} F_{\text{rad}} dS + \int_{\partial V} F_{\text{cond}} dS + \int_{\partial V} F_{\text{nuc}} dS \quad (2.3.2)$$

Manipulating equation (2.3.2) by applying Gauss’ theorem (1.1.1), the transport theorem (1.2.2) and keeping in mind that V was arbitrarily chosen the energy equation reads

$$\frac{\partial(\rho e_{\text{tot}})}{\partial t} = -\nabla(\vec{u}_{\text{rel}}(\rho e_{\text{tot}}) + \vec{u}P) + \nabla \cdot (\vec{u} \cdot \tau) + \rho \vec{u}g + \nabla \cdot F_{\text{rad}} + \nabla \cdot F_{\text{cond}} + \nabla \cdot F_{\text{nuc}}. \quad (2.3.3)$$

In the simulations outlined in this thesis, the energy contribution stemming from nuclear processes can be considered as negligible, i.e.

$$F_{\text{nuc}} = 0. \quad (2.3.4)$$

Furthermore it is justified to assume that heat conduction is a purely diffusive process,

$$F_{\text{cond}} = -K_{\text{h}} \nabla T. \quad (2.3.5)$$

In stellar interiors, also radiative energy transport may be described by a diffusion process,

$$F_{\text{rad}} = -K_{\text{rad}} \nabla T. \quad (2.3.6)$$

K_{rad} denotes the radiative conductivity. Equation (2.3.6) is known as the *diffusion approximation of radiative transfer*.

Close to stellar surfaces, however, the diffusion approximation is highly inaccurate and F_{rad} has to be obtained solving a transport equation for the radiation field. Note that the divergence of

the radiative flux F_{rad} is known as the *radiative heating rate*,

$$Q_{\text{rad}} = \nabla \cdot F_{\text{rad}}. \quad (2.3.7)$$

2.4 The Equation of State (EOS)

The equation of state completes the equations of radiation hydrodynamics. If the model assumes the gas to be ideal, the equation of state reads

$$P = \frac{\rho RT}{\mu}. \quad (2.4.1)$$

If a realistic microphysics model is employed, the equation of state depends on the physical properties of the gas under study such as the content of hydrogen, helium and metals as well as on the degree of ionisation. For this case, the OPAL [42] equation of state and opacities are included. In practice, the quantities of interest are then not calculated by some formula but tables of values are provided which have to be looked up and interpolated to obtain the desired information. In ANTARES, cubic splines are used for the interpolation process. Although the coefficients of the interpolation polynomials are computed and stored in a setup step, a call to the EOS still involves a considerable computational effort.

For the equation of radiative transfer, the OPAL opacities are used, augmented by Rosseland opacities [2] in low temperature regions.

2.5 Summary

The set of the equations of radiation hydrodynamics in differential formal reads

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \vec{u}_{\text{rel}}) \quad (2.5.1)$$

$$\frac{\partial(\rho \vec{u})}{\partial t} = -\nabla \cdot (\rho \vec{u}_{\text{rel}} \otimes \vec{u}) - \nabla P + \nabla \cdot \tau + \rho g \quad (2.5.2)$$

$$\frac{\partial(\rho e_{\text{tot}})}{\partial t} = -\nabla \cdot (\vec{u}_{\text{rel}}(\rho e_{\text{tot}}) + \vec{u}P) + \nabla \cdot (\vec{u} \cdot \tau) + \rho \vec{u}g + \nabla \cdot F_{\text{rad}} \quad (2.5.3)$$

All appearing variables and parameters are collected in Table 2.1.

2. THE EQUATIONS OF RADIATION HYDRODYNAMICS

$\rho = \rho(x, y, z, t)$	gas density
$c = c(x, y, z, t)$	concentration of second species
$\vec{u} = \vec{u}(x, y, z, t) = (u, v, w)^T$	flow velocity
$\rho \vec{u}$	momentum density
$\vec{u} \otimes \vec{u}$	Kronecker product
$p = p(T, \rho)$	gas pressure
$\mu = (x, y, z, c, t)$	mean molecular weight
$\vec{g} = (g, 0, 0)^T$	gravitational acceleration
$\tau = \tau(x, y, z, t)$	viscous stress tensor for zero bulk velocity
η	dynamic viscosity (appears in the definition of τ)
ζ	bulk viscosity (appears in the definition of τ)
$(\rho e) = (\rho e)(x, y, z, t) = (\rho e_{\text{int}}) + (\rho e_{\text{kin}})$	total energy density
$T = T(x, y, z, t)$	temperature
$Q_{\text{rad}} = Q_{\text{rad}}(x, y, z, t)$	radiative source term
$\chi_\nu = \chi_\nu(T, \rho)$	(specific) opacity at frequency ν
$K_{\text{rad}} = K_{\text{rad}}(T, \rho)$	radiative conductivity
K_{h}	heat diffusion coefficient
$\kappa_{\text{c}} = \kappa(c, \rho)$	diffusion coefficient for species c
R	gas constant

Table 2.1: Variables and parameters in the model equations (2.1.3)–(2.4.1).

Chapter 3

The ANTARES-Code

3.1 Methodology

The ANTARES code (**A** Numerical **T**ool for **A**strophysical **R**ESearch, [22]) is designed to solve the system of partial differential equations presented in Chapter 2 numerically in one, two or three spatial dimensions on either a rectangular or a spherical, radially stretched grid. For the simulation of pulsating stars a semi-Lagrangian approach is implemented, allowing the grid to move with the horizontally averaged motion of the fluid (c.f. Chapter 7). Moreover, ANTARES supports the definition of up to three levels of grids which can be nested into each other to improve the spatial resolution.

Basically, the Navier-Stokes equations are discretized by the method-of-lines technique. In this approach, the partial differential equations are first discretized on a spatial grid and a system of ordinary differential equations in time is obtained. The approximate solution is then computed by a suitable time integration scheme.

For time integration, explicit total-variation-diminishing (TVD) Runge-Kutta methods [10] of second and third order are used. A recent branch also incorporates *singly diagonally implicit Runge-Kutta (SDIRK) methods* [26].

For spatial integration of the hyperbolic terms, discretization schemes of ENO type are used. These comprise classical *essentially non-oscillatory (ENO)* methods, *weighted essentially non-oscillatory (WENO)* methods (optionally in conjunction with Marquina flux splitting) and *convex non-oscillatory (CNO)* methods. The parabolic terms are discretized by dissipative finite difference schemes of fourth order (see [26]).

For radiation hydrodynamics, ANTARES additionally solves the equation of radiative transfer (see [32], [37], [39]) by the short characteristics method.

Apart from the basic functionality outlined above, ANTARES contains solvers especially tailored to the low Mach number regime. It is capable to solve the incompressible formulation of the NSE in the Boussinesq approach (see [56]). The compressible equations in the low Mach number regime are treated by a fractional step method outlined in [16]. At the core of both methods is a solver for the linear generalized Poisson equation. This solver is based on a finite element

3. THE ANTARES-CODE

discretization and employs the Conjugate Gradient algorithm to obtain a solution of the arising linear system (see [11] for a detailed description). It is one goal of this PhD-thesis to replace this solver by a faster one based on a multigrid approach.

ANTARES is fully parallelized by a domain decomposition approach. Each grid is split along the coordinate axes into subdomains with only a small overlap region to neighboring cubes (approx. 4-5 grid points). The variables attached to a certain portion of the grid are stored in the memory belonging to the dedicated CPU-core. In this way, each CPU-core is mapped to a specific geometrical volume. For distributed-memory architectures, this domain decomposition approach is implemented with the *message passing interface (MPI)* framework. With this approach, ANTARES has been run on up to 4000 CPU-cores. Its scaling properties are outlined in [16]. Furthermore, ANTARES offers a second concept of parallelization based on a shared-memory approach. It is implemented via OpenMP-directives and can be used along with or independent of the domain decomposition approach.

3.2 Simulations

Up to now, the ANTARES code has been employed in a wide variety of astrophysical simulations. The first goal of ANTARES was to perform realistic simulations of solar surface convection and granulation ([22]) as well as simulations of A- and F-stars ([28]) and white dwarfs ([33]). Furthermore, ANTARES is capable of simulating pulsating stars such as cepheids and RR-Lyrae stars ([35, 36]). A recent branch is also the simulation of the stellar interior where double-diffusive convection can take place ([55, 56]). However, in almost all applications of ANTARES but the originally envisaged simulations of solar convection, the small timestep required by the explicit time integrators has proven to be a bottleneck. One aim of this PhD-thesis is to incorporate and test novel time-integration methods to circumvent this difficulty and which are applicable to a larger number of cases than the aforementioned methods for low Mach number hydrodynamics.

In this spirit, parts of this PhD project were:

- Development of an efficient multigrid solver for the linear and the nonlinear generalized Poisson equation.
- Parallelisation of the multigrid solver via MPI (Message Passing Interface).
- Implementation of IMEX Runge-Kutta methods for the semiconvection problem.
- Implementation of IMEX Runge-Kutta methods for the cepheid model.

Chapter 4

Implicit-Explicit (IMEX) Runge-Kutta Methods

4.1 Introduction

Many of the simulation codes applied to astrophysical fluid dynamics are based on explicit time integration methods. The advantage of explicit time integration schemes is that they are easy to construct, parallelizable and more efficient if the flow is compressible and timestep sizes of the advective timescale are possible.

However, explicit time integration methods are not suited for *stiff partial differential equations*, e.g. differential equations where two or more very disparate timescales are important. The CFL-condition requires the timestep to be set by the shortest timescale (see [45]). This timescale may be set by a rather insignificant process, rendering the simulation slow and inefficient. A prominent example is solving the compressible Navier-Stokes equations in the low Mach number regime ($\text{Ma} < 0.3$). There, acoustic wave propagation is by far the fastest operating process. In spite of the fact that sound waves transport only a negligible amount of energy and therefore contribute only very little to the overall solution, the timestep must be chosen such as to resolve the fast motions of the sound waves.

Furthermore, in the very low Mach number regime ($\text{Ma} < 0.01$), explicit time integration methods fail entirely. This breakdown is well known in the literature and attributed to the fact that the pressure term is of order $1/\text{Ma}^2$ which introduces considerable inaccuracy as the Mach number approaches 0 (see [12, 16]).

On the other hand, most implicit time integration methods have the advantage of having no stability limit on the timestep. However, in terms of CPU time, implicit methods are computationally more expensive than explicit ones since the discretized system of partial differential equations usually results in a nonlinear system of equations which has to be solved for each timestep. The general procedure is to solve the arising system via a Newton method, which requires the determination of the corresponding Jacobian matrix. This approach is both time and memory consuming and it is not always clear whether the higher timestep pays off for this additional effort.

4. IMPLICIT-EXPLICIT (IMEX) RUNGE-KUTTA METHODS

In [50] the Navier Stokes equations are solved implicitly and simulations in the context of stellar astrophysics are performed. The authors report that in simulations of relatively hot A-type stars, the timestep is practically limited by the velocity of the fluid through the advective timescale, which determines the rate of change in the solution. In a simulation of a cold giant star, however, large CFL numbers are obtained.

In the simulations within the focus of this thesis, the features are changing on a timescale corresponding to the advective one. Therefore, an “operator splitting” approach similar to [57] is adopted, treating the advective (hyperbolic) term explicitly while solving the diffusive (parabolic) term with an implicit scheme. Contrary to [57], a unified approach using an additive Runge-Kutta scheme is preferred, which deals with the splitted operators in a consistent way. First, however, a short introduction to Runge-Kutta methods and their analysis is given.

4.2 Runge-Kutta methods

Due to the method-of-lines approach employed in discretizing the NSE within the ANTARES-framework, it suffices to consider the ODE initial value problem

$$\frac{\partial}{\partial t}y(t) = F(y(t)) \quad (4.2.1)$$

For this type of problems, *Runge-Kutta* schemes are popular.

A Runge-Kutta method is represented by the array of numbers (c.f. [5])

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} \quad (4.2.2)$$

where $A = (a_{ij})$ is called the *coefficient matrix*. The numbers $c_1, c_2, \dots, c_s$ are defined by $c_i = \sum_{j=1}^s a_{ij}$ for $i = 1, 2, \dots, s$ due to requirements of internal consistency (c.f. [5]).

Definition 4.2.1. An s-stage Runge-Kutta method characterized by coefficient matrix $A = (a_{ij})$ defines one step $y_{\text{old}} \rightarrow y_{\text{new}}$ by $(i = 1, \dots, s)$

$$\begin{aligned} y_i &= y_{\text{old}} + \Delta t \sum_{j=1}^s a_{ij} F(y_j) \\ y_{\text{new}} &= y_{\text{old}} + \Delta t \sum_{j=1}^s b_j F(y_j) \end{aligned}$$

If $a_{ij} = 0$ for $j \geq i$ the method is an *explicit* Runge-Kutta method, if $a_{ij} \neq 0$ for some $j \geq i$, the method is referred to as *implicit*.

For explicit Runge-Kutta methods, an alternative representation of the Runge-Kutta method can be advantageous. It is shown in [8] that the relations specifying one Runge-Kutta step can be rewritten in the form

$$y_i = \sum_{j=1}^{i-1} \lambda_{ij} y_j + \Delta t \mu_{ij} F(y_j) \quad i = 1, \dots, s \quad (4.2.3)$$

$$y_{\text{new}} = y_s. \quad (4.2.4)$$

The coefficients λ_{ij} and μ_{ij} are derived from the coefficient scheme (A, b) . This is known as the *Shu-Osher representation* of the explicit Runge-Kutta method (A, b) .

A thorough introduction to Runge-Kutta methods and their application to ordinary and partial differential equations can be found in [5], [15] and [46]. The subsequent sections summarize some of the concepts important within the context of this thesis.

4.2.1 Stability Analysis

Numerical solutions to differential equations should ideally resemble the analytical solution and display the same qualitative behavior. The assessment of qualitative properties of numerical solutions such as boundedness and convergence to zero (when these properties are possessed by the true solution) is part of the *stability analysis*.

Definition 4.2.2. The *stability function* of a Runge-Kutta scheme is defined by (c.f. [5])

$$R : \bar{\mathbb{C}} \times \bar{\mathbb{C}} \rightarrow \mathbb{C} \quad R(z) = 1 + zb^T(I - zA)^{-1}\hat{1} \quad (4.2.5)$$

where $\bar{\mathbb{C}} := \mathbb{C} \cup \infty$, $b = (b_1, \dots, b_s)^T$, $A = (a_{ij})$, $\hat{1} = (1, \dots, 1)^T$ and I is the identity matrix in $\mathbb{R}^{s \times s}$.

Definition 4.2.3. The *stability region* $\mathcal{S}$ of a Runge-Kutta method is defined by (c.f. [5, 15])

$$\mathcal{S} = \{z \in \bar{\mathbb{C}} \mid |R(z)| \leq 1\}. \quad (4.2.6)$$

A Runge-Kutta method is said to be *A-stable* if the stability region contains the non-positive complex half-plane $\mathbb{C}^- := \{z : \text{Re}(z) > 0\}$. It is referred to as *L-stable* if additionally $\lim_{|z| \rightarrow \infty} R(z) = 0$ holds.

For explicit Runge-Kutta methods, the stability function $R(z)$ is a polynomial, for implicit Runge-Kutta methods a rational function in $\bar{\mathbb{C}}$. Therefore, the stability region for explicit Runge-Kutta methods is always bounded, implying that explicit Runge-Kutta methods are never A-stable nor L-stable.

Linear Stability

Although the method-of-lines approach allows a strict separation of temporal and spatial discretization techniques, the interplay of those is of considerable importance when it comes to choosing suitable time-step sizes Δt as well as the spatial resolution h .

Linear stability is assessed by determining the *amplification function* $g(\mu, \theta) = R(-\mu h(\theta))$ with R denoting the stability function of the corresponding Runge-Kutta method and $h : [-\pi, \pi] \rightarrow \mathbb{R}$ depends on the spatial discretization (see also [29]). The variable μ denotes the *Courant-Friedrichs-Levy (CFL)* number.

A spatial discretization method combined with a temporal discretization is linearly stable if the amplification factor satisfies the condition

$$|g(\mu, \theta)| \leq 1 \quad \theta \in [0, 2\pi] \quad (4.2.7)$$

This stability condition imposes an upper bound on μ . The largest possible number $\mu > 0$ satisfying this condition is denoted by μ_a and is referred to as the *algebraic stability limit* in [34].

In [34], another, geometric stability condition is given. It relies on the definition of the spectrum of the spatial discretization scheme by $\mathcal{S}_L = \{-h(\theta) | 0 \leq \theta \leq 2\pi\}$ where $h(\theta)$ is the complex valued function dependent on the spatial discretization scheme also used in (4.2.7). Let $\mathcal{S}_t$ denote the stability region of the time-stepping scheme. Then the *geometric stability limit* denoted by μ_g is the largest number $\tilde{\mu}$ such that the rescaled spectrum $\tilde{\mu}\mathcal{S}_L$ lies inside the stability domain $\mathcal{S}_t$,

$$\tilde{\mu}\mathcal{S}_L \subseteq \mathcal{S}_t \quad (4.2.8)$$

Figure 4.1 illustrates the geometric stability condition.

Note that either approach gives the same results, i.e. $\mu_a = \mu_g$.

Dissipativity analysis

Some numerical schemes produce solutions that contain small oscillations. For many calculations, especially for nonlinear equations, such small, high frequency oscillations play a significant role in reducing the accuracy of the solution. Furthermore, in the context of parabolic equations, such high frequency oscillations are quickly damped by the analytical solution, a behavior, which should be reflected by the numerical solution. The decreasing of high frequency oscillations occurs due to *dissipation*.

Definition 4.2.4. A scheme is *dissipative of order $2r$* ([46]) if there exists a positive constant c such that each amplification factor $g(\mu, \theta)$ satisfies

$$|g(\mu, \theta)| \leq 1 - c \sin^{2r} \left(\frac{1}{2} \theta \right) \quad \forall \theta \neq 0 \quad (4.2.9)$$

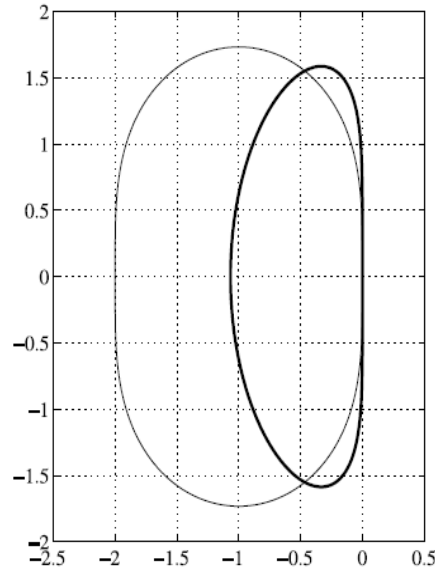


Figure 4.1: In this figure, the condition for linear stability is graphically illustrated. The thin line denotes the border of the stability region of the SSPRK(2,2) scheme whereas the spectrum of the linearized WENO method for the advection equation is denoted by the thick line. The stability condition demands that the spectrum of the spatial discretization scheme is contained in the stability region of the Runge-Kutta method. Therefore, the spectrum must be rescaled to satisfy this condition. The corresponding scaling factor is known as the CFL-number. This illustration is taken from [34].

However, in the analysis outlined below, this definition is altered in the sense that a scheme is considered dissipative if

$$|g(\mu, \theta)| < 1 \quad \forall \theta \neq 0 \quad (4.2.10)$$

$$|g(\mu, 0)| = 1. \quad (4.2.11)$$

Furthermore, it turns out in the simulations presented in Chapter 6 that a beneficial condition is to demand positivity of the amplification factor, i.e.

$$0 < g(\mu, \theta) < 1 \quad \forall \theta \neq 0. \quad (4.2.12)$$

4.2.2 The strong-stability-preserving (SSP) property

It is outlined in [9] that when dealing with discontinuous solutions of nonlinear hyperbolic partial differential equations, linear stability theory no longer guarantees convergence. The spatial discretization must be carefully chosen to be nonlinearly stable under Forward Euler integration – in the context of hyperbolic PDEs, the relevant nonlinear stability property takes the form of *total-variation diminishing (TVD)*, *total-variation-bounded (TVB)* or a non-oscillatory requirement. Within the ANTARES code, the spatial discretization is taken care of by an *essentially*

4. IMPLICIT-EXPLICIT (IMEX) RUNGE-KUTTA METHODS

non-oscillatory method, which satisfies the property of nonlinear stability.

However, for actual computations, higher order time discretization schemes than the Euler Forward method are used. High order *strong stability preserving (SSP)* methods guarantee that the nonlinear stability properties satisfied by the spatial discretization when coupled with forward Euler integration will be preserved when the same spatial discretization is coupled with these higher order time discretization methods.

Definition 4.2.5. Given a semi-discretization of the form (4.2.1) and a convex functional $\|\cdot\|$ it is assumed that

$$\|y_{\text{old}}(t) + \Delta t F(y_{\text{old}}(t))\| < \|y_{\text{old}}\| \quad (4.2.13)$$

for all $0 < \Delta t < \Delta t_{FE}$.

A numerical time integrator computing the next solution value y_{new} from previous values $y_i, i = 1, \dots, s$ is *strong stability preserving* or *monotonic*, if

$$\|y_{\text{new}}\| < \max_{i=1,s} \{\|y_i\|\} \quad (4.2.14)$$

holds for some timestep $\Delta t \leq c \Delta t_{FE}, c > 0$.

If the convex functional used is the *total variation norm* $\|\cdot\|_{TV}$, the property is commonly referred to as *total-variation-diminishing* (see [9]).

4.2.3 Analysis of the discretization schemes in the context of the NSE and the ANTARES-Code

Although the NSE contain advective as well as diffusive terms, it is justified in [46] that it is sufficient to consider only the diffusion term since the advective terms become negligible in the asymptotic limit. Therefore, the analysis is carried out considering the heat equation

$$\frac{d}{dt} u = b \frac{d^2 u}{(dx)^2}. \quad (4.2.15)$$

Within the ANTARES code, terms of second order are evaluated using the fourth order spatial discretization stencil

$$(L_{\Delta x} u^n)_j := \frac{-u_{j+2}^n + 16 u_{j+1}^n - 30 u_j^n + 16 u_{j-1}^n - u_{j-2}^n}{12 (\Delta x)^2}. \quad (4.2.16)$$

Furthermore, three different explicit Runge-Kutta methods are available within the ANTARES framework. The corresponding stability and dissipativity analysis have been performed in [26] and [29]. The findings are summarized below.

An SSPRK(2,2) method

In [10], the following second order scheme is identified as the optimal TVD Runge-Kutta method of second order in the sense of permitting a large CFL-coefficient,

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline A & \frac{1}{2} & \frac{1}{2} \end{array} \quad (4.2.17)$$

The stability region of this scheme is bounded and covers an elliptical region in the complex left half plane. It is tangential to the imaginary axis. The amplification factor is always positive and exceeds 1 at $\mu \approx 0.375$.

An SSPRK(3,2) method

This scheme has been stumbled across by observing the favorable properties of the IMEX SSP2(3,3,2) scheme presented in section 4.3 in the advection dominated regime of the double-diffusive convection scenarios (Chapter 6). It was first published by [27] and later declared the optimal second order scheme with three stages by various authors. The SSPRK(3,2) coefficient scheme reads

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} & 0 \\ \hline A & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \quad (4.2.18)$$

The corresponding stability region covers a bounded region in the negative halfplane. It is tangential to the imaginary axis without intersecting it. The first positive zero of the amplification factor is $\mu \approx 0.6726$ where the function changes its sign. It exceeds 1 at $\mu \approx 0.8474$.

4.3 SSP IMEX Runge-Kutta methods

Implicit-explicit (IMEX) Runge-Kutta methods belong to the class of *additive Runge-Kutta* methods. They are popular for the solution of ODE initial value problems of appearance

$$\frac{d y(t)}{dt} = F(y(t)) + G(y(t)) \quad (4.3.1)$$

where the vector fields $F(y(t))$ and $G(y(t))$ have different stiffness properties. Methods of this kind use different Runge-Kutta formulae for the treatment of the two vector fields.

Definition 4.3.1. An s-stage additive Runge-Kutta method characterized by coefficient matrices

4. IMPLICIT-EXPLICIT (IMEX) RUNGE-KUTTA METHODS

$A = (a_{ij})$ and $\tilde{A} = (\tilde{a}_{ij})$ defines one step $y_{\text{old}} \rightarrow y_{\text{new}}$ by

$$\begin{aligned} y_i &= y_{\text{old}} + \Delta t \sum_{j=1}^s a_{ij} F(y_j) + \Delta t \sum_{j=1}^s \tilde{a}_{ij} G(y_j) \quad i = 1, \dots, s \\ y_{\text{new}} &= y_{\text{old}} + \Delta t \sum_{j=1}^s b_j F(y_j) + \Delta t \sum_{j=1}^s \tilde{b}_j G(y_j) \end{aligned}$$

If $a_{ij} = 0$ but $\tilde{a}_{ij} \neq 0$ for some for $j \geq i$ the method is referred to as an *implicit-explicit Runge-Kutta* method.

In [17] and [18], a theory of strong stability preserving additive Runge-Kutta schemes is developed. An important role hereby plays the *region of absolute monotonicity*, which is defined by

$$\mathcal{R}(A, \tilde{A}) = \{(r, \tilde{r}) \in \mathbb{R}^2 \mid (A, \tilde{A}) \text{ is absolutely monotonous on } [-r, 0] \times [-\tilde{r}, 0]\} \quad (4.3.2)$$

and the absolute monotonicity at a given point $(r_0, \tilde{r}_0)$ is characterized by algebraic relations for the coefficient matrices A and $\tilde{A}$ (see [18] and [17]). The boundary of $\mathcal{R}(A, \tilde{A})$ in the first quadrant is called the *curve of absolute monotonicity*. The region of absolute monotonicity gives conditions on the timestep size Δt to ensure that the SSP-property is conserved. Naturally, if the curve of absolute monotonicity is 0, the method is not SSP. It is pointed out in [18] that necessary conditions for nontrivial curves of absolute monotonicity are the nonnegativity of the Runge-Kutta coefficients a_{ij} and $\tilde{a}_{ij}$ and the positivity of the coefficients b_j and $\tilde{b}_j$ ([18]). Both are handy criteria to determine whether a given IMEX scheme might be SSP or not.

In the following, some SSP IMEX schemes deemed suitable for the purposes of this thesis are presented. They have been analysed in [29] with respect to their stability regions, error constants and dissipativity properties. The properties relevant throughout this thesis are repeated below for the reader's convenience.

All strong stability preserving IMEX schemes considered have diagonally implicit Runge-Kutta methods as their implicit scheme. This structure ensures that the stages can be solved successively and the explicit part has to be evaluated only once in each stage.

As suggested in [16], the nomenclature $SSPk(s, \sigma, p)$ to distinguish different IMEX schemes is used. In this notation, k denotes the order of the method which is characterized by the coefficients for the explicit part, s the number of stages in the implicit scheme and σ the number of stages in the explicit scheme. p denotes the order of the combined method.

An IMEX SSP2(2,2,2) method

An IMEX SSP2(2,2,2) method mentioned in [40] is considered. It is characterized by the coefficient arrays

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline A & \frac{1}{2} & \frac{1}{2} \end{array} \quad \begin{array}{c|cc} \gamma & \gamma & 0 \\ 1 - \gamma & 1 - 2\gamma & \gamma \\ \hline \tilde{A} & \frac{1}{2} & \frac{1}{2} \end{array}. \quad (4.3.3)$$

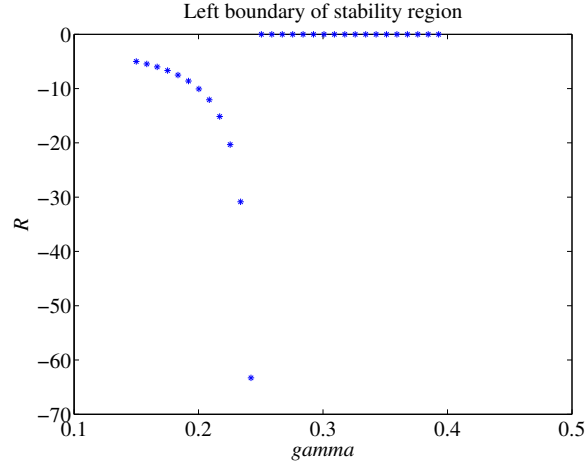


Figure 4.2: The left boundary of the stability region as a function of γ is depicted. Whereas for $\gamma < 0.25$, the stability region is bounded, it is unbounded for $\gamma \geq 0.25$. This picture is taken from [29].

where $\gamma = 1 - \frac{1}{\sqrt{2}} \approx 0.29289$.

The explicit scheme A is exactly the SSPRK(2,2) scheme discussed above.

The scheme $\tilde{A}$ is L-stable and dissipativity analysis shows that the first positive zero of the amplification factor is $\mu \approx 0.4526$ where the function changes its sign. The amplification factor never exceeds 1 (c.f. [29]).

The use of the parameter γ suggests its adaptation according to the resulting stability, accuracy and dissipativity properties (see [29]).

The stability regions for different values of γ cover bounded subdomains of the complex left half plane for $\gamma < 0.25$ while for $\gamma \geq 0.25$ the stability regions cover unbounded domains in the complex left half plane. However, even for $\gamma \geq 0.25$, the methods are not necessarily L-stable. The left boundary of the stability region is plotted in Figure 4.2 as a function of γ .

For $\gamma \geq 0.25$, the amplification factor for the spatial discretization given in (4.2.16) changes its sign but it never exceeds 1. Contrarily, for $\gamma < 0.25$, the amplification factor possesses no real roots but exceeds 1 at certain points. The roots and maxima of the amplification factor g as a function of γ are shown in Figure 4.3 .

An IMEX SSP2(3,3,2) method

Furthermore, an IMEX SSP2(3,3,2) method given by [17],

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} & 0 \\ \hline A & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \quad \begin{array}{c|ccc} \frac{1}{5} & \frac{1}{5} & 0 & 0 \\ \frac{3}{10} & \frac{1}{10} & \frac{1}{5} & 0 \\ 1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline \tilde{A} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \tag{4.3.4}$$

is regarded.

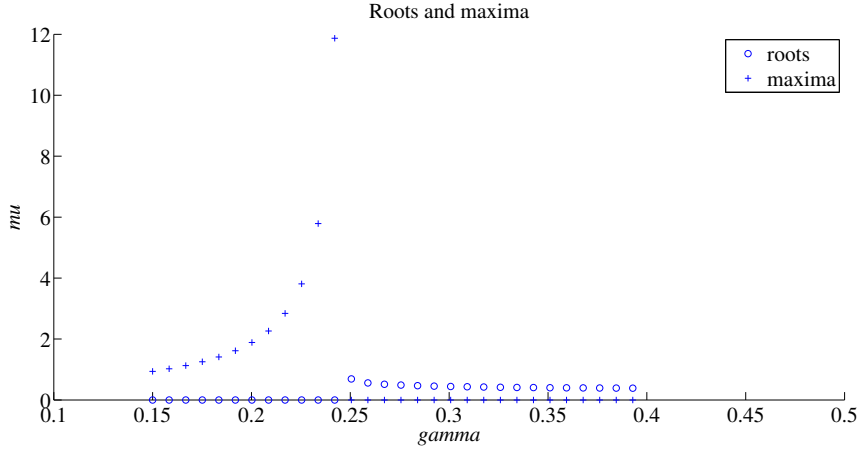


Figure 4.3: The roots and maxima of the amplification factor g are plotted as a function of γ . This picture is taken from [29].

The explicit scheme is exactly the SSPRK(3,2) scheme discussed above.

The implicit scheme $\tilde{A}$ is L-stable. The dissipativity analysis (c.f. [29]) for the implicit schemes shows that the first positive zero of the amplification factor is ≈ 0.6064 and g never exceeds 1.

4.4 Optimized IMEX Runge-Kutta methods

The SSP IMEX methods presented in the previous section have been constructed such as to optimize the region of absolute monotonicity which characterizes the step sizes admissible in order to ensure that the SSP-property is preserved throughout the time integration. However, the SSP-property is not the only property beneficial to astrophysical simulations. Therefore, in [20] (resp. [19]) new schemes have been devised with the aim of surpassing the original IMEX methods both in efficiency as well as in accuracy. Since the accuracy of simulations in the realm of astrophysics is generally limited by the spatial resolution (see [29]), it suffices to consider second-order accurate schemes.

The following guidelines have been assumed in the construction process:

- The IMEX scheme should be of second order.
- The IMEX scheme should have a large region of absolute monotonicity.
- The implicit scheme should be L-stable.
- For both schemes, the stability function should be nonnegative for a large interval of the negative real axis.
- The region of absolute stability of the IMEX scheme should be large.
- For a convenient and memory efficient implementation, the coefficients of the scheme should be rational numbers.

It is outlined in [20] (resp. [19]) that in order to fullfill all requirements listed above, the IMEX scheme must have at least three stages. Moreover, the weight vector $b = (b_1, b_2, \dots, b_n)$ appearing in the definition of a Runge-Kutta scheme is taken to be the same for both the explicit and the implicit subscheme of the IMEX method. This ensures that there are no extra coupling order conditions for the IMEX Runge-Kutta scheme; whence, if both schemes are second-order accurate, the IMEX RK scheme is second order accurate as well ([17]). Furthermore, the implicit scheme is a singly diagonally implicit (SDIRK) Runge-Kutta method which ensures that the stages can be solved successively. Details of the construction process are given in [20] (resp. [19]), in the following the most important schemes and their properties are summarized.

Nomenclature

Following the nomenclature of the previous section and constructing schemes fullfilling the requirements listed above, every scheme would be denominated IMEX SSP2(3,3,2). To distinguish the different schemes, additional letters are introduced which refer to the properties of the scheme:

- 'L': The method is L-stable.
- 'S': The stability region of the explicit scheme contains an interval on the imaginary axis.
- 'P': The amplification factor g is always positive.
- 'U': The IMEX method features uniform convergence. The property of *uniform convergence* basically ensures that the global error decreases quadratically even in the asymptotic limit of arbitrary stiffness of the ordinary differential equation to be solved.
- 'M': The IMEX method has a nontrivial region of absolute monotonicity.

Thus, the IMEX SSP2(3,3,2) method from the literature presented in section 4.3 is henceforth denominated as IMEX SSP2(3,3,2)-LUM.

An IMEX SSP2(3,3,2)-LSPUM method

The IMEX SSP2(3,3,2)-LSPUM method reads

$$\begin{array}{c|ccc}
 0 & 0 & 0 & 0 \\
 \frac{5}{6} & \frac{5}{6} & 0 & 0 \\
 \frac{11}{12} & \frac{11}{24} & \frac{11}{24} & 0 \\
 \hline
 A & \frac{24}{55} & \frac{1}{5} & \frac{4}{11}
 \end{array}
 \quad
 \begin{array}{c|ccc}
 \frac{2}{11} & \frac{2}{11} & 0 & 0 \\
 \frac{289}{462} & \frac{205}{462} & \frac{2}{11} & 0 \\
 \frac{751}{924} & \frac{2033}{4620} & \frac{21}{110} & \frac{2}{11} \\
 \hline
 \tilde{A} & \frac{24}{55} & \frac{1}{5} & \frac{4}{11}
 \end{array}
 \tag{4.4.1}$$

The stability region of the explicit scheme is a bounded region in the negative complex halfplane and covers a part of the imaginary axis. The amplification factor changes its sign at $\mu \approx 0.3409$ and exceeds 1 at $\mu \approx 0.5339$.

The implicit scheme is L-stable. Its amplification factor has no real roots and never exceeds 1. This IMEX method satisfies the condition for uniform convergence.

4. IMPLICIT-EXPLICIT (IMEX) RUNGE-KUTTA METHODS

An IMEX SSP2(3,3,2)-LPUM method

The IMEX SSP2(3,3,2)-LPUM method is characterized by the coefficient matrices

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} & 0 \\ \hline A & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \quad \begin{array}{c|ccc} \frac{2}{11} & \frac{2}{11} & 0 & 0 \\ \frac{69}{154} & \frac{41}{154} & \frac{2}{11} & 0 \\ \frac{67}{77} & \frac{289}{847} & \frac{42}{121} & \frac{2}{11} \\ \hline \tilde{A} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \quad (4.4.2)$$

The explicit scheme is the SSPRK(3,2) scheme discussed above.

The implicit scheme is L-stable. Its amplification factor has no real roots and never exceeds 1.

This IMEX method satisfies the condition for uniform convergence.

An IMEX SSP2(3,3,2)-LPM method

The IMEX SSP2(3,3,2) method is determined by the coefficient arrays

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} & 0 \\ \hline A & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \quad \begin{array}{c|ccc} \frac{2}{11} & \frac{2}{11} & 0 & 0 \\ \frac{5003}{13310} & \frac{2583}{13310} & \frac{2}{11} & 0 \\ \frac{6271}{6655} & \frac{39731}{139755} & \frac{10}{21} & \frac{2}{11} \\ \hline \tilde{A} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \quad (4.4.3)$$

The explicit scheme is the SSPRK(3,2) scheme discussed above.

The implicit scheme is L-stable. Its amplification factor has no real roots and never exceeds 1.

This IMEX method is not uniformly convergent.

4.5 Summary

Table 4.1 summarizes the findings of the stability and dissipativity analysis of the schemes discussed above.

4.6 Application to the Navier-Stokes equations

4.6.1 Timestep Restrictions

If the NSE are advanced in time by a Runge-Kutta method, restrictions on the timestep are imposed by heat diffusion τ_T , the viscosity τ_{visc} , the propagation of acoustic waves τ_{snd} and the velocity of the fluid τ_{fluid} . The actual time increment Δt is taken as the minimum of all restrictions,

$$\Delta t = \min(\tau_T, \tau_{\text{visc}}, \tau_{\text{snd}}, \tau_{\text{fluid}}) \quad (4.6.1)$$

	<i>exp</i> $g = 0$	<i>exp</i> $ g = 1$	<i>imp</i> $g = 0$	<i>imp</i> $ g = 1$
<i>explicit SSPRK-methods</i>				
SSPRK(2,2)	—	0.375	—	—
SSPRK(3,2)	0.6726	0.8474	—	—
<i>IMEX SSPRK-methods from the literature</i>				
IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$	—	0.375	0.452	—
IMEX SSP2(2,2,2), $\gamma = 0.24$	—	0.375	—	9.375
IMEX SSP2(3,3,2)-LUM	0.6726	0.8474	0.455	—
<i>optimized IMEX SSPRK-methods</i>				
IMEX SSP2(3,3,2)-LSPUM	0.3409	0.5339	—	—
IMEX SSP2(3,3,2)-LPUM	0.6726	0.8474	—	—
IMEX SSP2(3,3,2)-LPM	0.6726	0.8474	—	—

Table 4.1: Summary of the dissipativity analysis of SSP time integrators. The second and third column refer to the maximum Courant numbers admissible within the explicit (sub)schemes, whereas the fourth and fifth column list the maximum Courant numbers for the implicit sub-schemes.

and

$$\begin{aligned}
 \tau_T &= \frac{\mu_{\text{diff}}}{\kappa_T} \min((\Delta x)^2, (\Delta y)^2) \\
 \tau_{\text{visc}} &= \frac{\mu_{\text{diff}} \rho}{\eta} \min((\Delta x)^2, (\Delta y)^2) \\
 \tau_{\text{fluid}} &= \frac{\mu_{\text{adv}}}{\max(|\vec{u}|)} \min(\Delta x, \Delta y) \\
 \tau_{\text{snd}} &= \frac{\mu_{\text{adv}}}{\max(|\vec{u}| + c_{\text{snd}}, |\vec{u}| - c_{\text{snd}})} \min(\Delta x, \Delta y)
 \end{aligned}$$

where the sound speed is denoted by c_{snd} . Timestep restrictions stemming from the source term in equations (2.5.2) and (2.5.3) representing buoyancy forces acting on the flow can be neglected in the limit where $\max\{\Delta x, \Delta y\} \rightarrow 0$ since its contributions are of lower order ([46]). However, since the asymptotic limit is not reached in actual simulations, a discussion on the importance and magnitude of a buoyancy timestep restriction is found in [29].

The usage of different Courant numbers μ_{diff} and μ_{adv} requires an explanation. Basically, the *diffusive Courant number* μ_{diff} denotes the Courant number found by linear stability analysis of the fourth order discretization scheme (4.2.16) and the time integration scheme. It may be argued that the simplification to analyze only the heat equation instead of the NSE is justified since the advection term is of lower order and negligible in the asymptotic limit. However, actual simulations do not reach the asymptotic limit and therefore, the advective terms do have considerable influence. Since they are evaluated by a WENO scheme and not by the discretization stencil (4.2.16), the corresponding stability analysis is bound to yield a different CFL-number

4. IMPLICIT-EXPLICIT (IMEX) RUNGE-KUTTA METHODS

which is denoted by μ_{adv} .

A straightforward idea to obtain the *advective CFL number* μ_{adv} would be to perform a linear stability analysis considering the advection equation $u_t + u_x = 0$ discretized by the WENO-stencil and the different time integration methods. Such an analysis has been performed by [51]. The findings are deflating; the analysis shows that neither the SSPRK(2,2) nor the SSPRK(3,2) scheme are linearly stable for a $\mu > 0$. However, a similar study has been conducted by [34] highlighting an important difference to the study of [51]: only discrete values of the spectrum corresponding to the grid points instead of the entire (analytical) spectrum are considered in the analysis. (Note that this is actually what happens if a spatial discretization approximates the right hand side of the differential equation under consideration.) Due to this difference, small, but positive CFL-numbers are found such that the SSPRK(2,2) and the SSPRK(3,2) are linearly stable in conjunction with the WENO-stencil.

However, the following considerations prevent the direct adoption of these findings:

- The WENO-method employed is a non-linear discretization method. Both the studies of [34] and [51] rely on a linearization of the WENO-stencil, therefore, the results are only of limited meaning.
- The NSE do contain diffusive terms which “add stability” to the continuous and, possibly, to the spatially discretized operator since as a consequence of the inclusion of the diffusive terms, the eigenvalues may happen to be located in the negative complex half plane (and thus, possibly, in the stability region of the time integration scheme in question).

For this reason, the advective CFL number μ_{adv} used in the simulations are empirical values whereas for the values of μ_{diff} the findings of the linear stability analysis presented above are used. The maximum admissible values of μ_{diff} depend on the (IMEX) Runge-Kutta scheme and are given in Table 4.1. In case the amplification function changes its sign before it exceeds 1, the maximum CFL-number is taken as the value where $g(\mu, \theta)$ changes its sign. This is to avoid creating any additional, high frequency oscillations or complete suppression (through an amplification factor of 0) of the contribution by the diffusion processes at certain length scales. Both would be unphysical and reduce the stability of the numerical scheme.

IMEX methods treat the term $G(y(t))$ implicitly. Therefore, the corresponding timestep restrictions are much alleviated or can even be discarded completely if the implicit method is unconditionally stable. Thus, IMEX methods can provide the desired computational advantage.

4.6.2 Implementation issues

The equations for fluid flow as presented in Chapter 2 can be recast as

$$\underbrace{\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho \vec{u} \\ \rho e_{\text{tot}} \end{pmatrix}}_{y(t)} = \nabla \cdot \underbrace{\begin{pmatrix} \rho \vec{u} \\ \rho \vec{u} \otimes \vec{u} + P - \tau \\ \rho e_{\text{tot}} \vec{u} + P \vec{u} - \vec{u} \cdot \tau \end{pmatrix}}_{F(y(t))} + \underbrace{\begin{pmatrix} 0 \\ g \\ g \vec{u} \end{pmatrix}}_{G(y(t))} - \underbrace{\begin{pmatrix} 0 \\ 0 \\ Q_{\text{rad}} \end{pmatrix}}_{G(y(t))} \quad (4.6.2)$$

In the context of the problem (4.6.2), the i^{th} implicit stage of an IMEX Runge–Kutta method is typically of the form

$$y_i = y^* + \Delta t \tilde{a}_{ii} G(y_i), \quad (4.6.3)$$

where y^* is known from previous stages. This is a consequence of $\tilde{a}_{ij} = 0$ for $j > i$.

Due to model assumptions, which will be outlined for each application separately in Chapter 6, equation (4.6.3) amounts to either a linear generalized Poisson equation

$$-\nabla \cdot (\kappa(\vec{x}) \nabla u(\vec{x})) + c(\vec{x}) u(\vec{x}) = f(\vec{x}) \quad (4.6.4)$$

or its nonlinear counterpart

$$-\nabla \cdot (\kappa(u(\vec{x}), \vec{x}) \nabla u(\vec{x})) + c(u(\vec{x}), \vec{x}) = f(\vec{x}). \quad (4.6.5)$$

subject to suitable boundary conditions.

Naturally, the question of the existence of solutions to equations (4.6.4) and (4.6.5) arises. In this context, generally the notion of *weak solutions* is considered (see [7]). To ensure the existence (and, in the linear case, uniqueness) of weak solutions, it is required that the coefficient functions κ and c are strictly positive. It will be discussed for each application separately that these conditions are met. Furthermore, κ , c and the right hand side function f have to meet certain regularity properties. Since the underlying physics permit the assumption that every quantity arising in (4.6.2) is a continuously differentiable function in space and bounded, those requirements are fulfilled. An extensive analysis of the linear equation (4.6.4) is found in [7, 11, 15] and references therein. For theoretical treatment of the nonlinear equation (4.6.5) the reader is referred to [43].

Chapter 5

Multigrid Methods

5.1 Introduction

The efficiency of the implicit-explicit Runge-Kutta schemes presented in the previous section relies on the efficient solution of a boundary value problem. Generally, the numerical solution of boundary value problems is indispensable in almost all fields of physics. Furthermore, the wish for higher and higher resolution as well as the study of three-dimensional problems leads to systems of larger and larger numbers of equations. The growing need of computational power is met by the fast development of modern supercomputers. Naturally, also the demand of numerical schemes capable of optimally utilizing the given resources is great.

Multigrid methods range among the most efficient numerical schemes to solve boundary value problems. The first multigrid methods have been applied to Poisson's equation and show an efficiency similar to that of direct solvers, but differently from other numerical methods, the efficiency is not lost when other problems than Poisson's equation are considered. Furthermore, the multigrid technique is easily adapted to nonlinear problems.

The characteristic feature of the multigrid iteration is its fast convergence. Contrary to classical iterative methods, the convergence speed does not deteriorate when the discretization is refined ([47]). As a consequence, one obtains an acceptable approximation of the discrete problem at the expense of computational work proportional to the number of unknowns ([47], c.f. section 5.5.6).

However, there is no unique multigrid algorithm applying to any problem but a multigrid technique defining only the framework of the algorithm. The efficiency of the multigrid algorithm depends on the adjustment of its components to the problem in question.

In the following, the multigrid algorithm developed to solve the equations arising in the implicit stage of an IMEX Runge-Kutta method is developed.

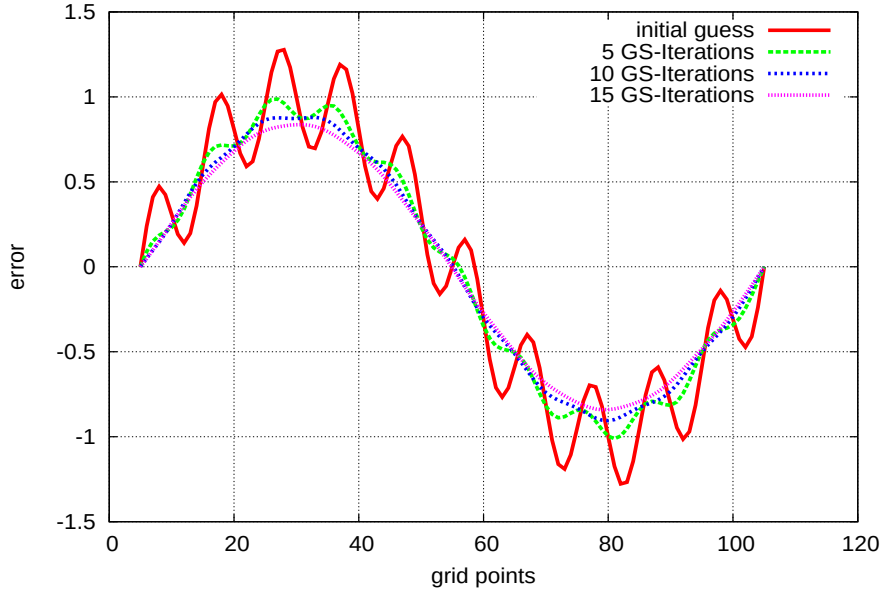


Figure 5.1: Error damping property of the Gauss-Seidel iteration: Whereas high frequency errors are efficiently damped after a few sweeps, the low frequency error is only minimally reduced.

5.2 The Multigrid Idea

The idea of using multiple grids for solving a system of equations originates in the observation of classical iterative schemes such as the Jacobi or the Gauss-Seidel iteration. A detailed analysis of the convergence behavior of such schemes (see [13] for the analysis of the convergence behavior of the damped Jacobi method) reveals that the methods in question are very fast to damp high frequency errors on the grid scale, but slow to reduce low frequency errors.

This fact is vividly demonstrated in the following example:

Example 5.2.1. The model problem

$$\nabla \cdot (\kappa(x) \nabla u(x)) + c(x) u(x) = f(x) \quad (5.2.1)$$

with $\kappa(x) = 1$, $c(x) = 1$ and $f(x) = 0$ is to be solved on a one-dimensional domain $I = [0, 2\pi]$ with homogeneous Dirichlet boundary conditions. Naturally, the analytical solution is $u = 0$. After discretization, the (lexicographic) Gauss-Seidel iterative method is employed to obtain an approximative solution. The solver is entered with the initial guess

$$u_{\text{init}}(x) = \sin x + 0.1 \sin 10x \quad (5.2.2)$$

Figure 5.1 shows the error damping property of the Gauss-Seidel method. While after 5 sweeps the high frequency error is significantly damped and, after 15 sweeps, completely removed from the residual error, the low frequency part is only minimally reduced, causing the overall scheme to converge very slowly.

Imagine now a sequence of nested grids, starting from the original one and coarsening successively

until a grid with a minimal number of grid points is produced. If now the high frequency component of the error is damped on the finest grid by performing a few relaxation sweeps and the residual error is transferred to the next coarser grid, the iterative method will be more effective in damping the low frequency error on the coarser grid. Furthermore, reducing the number of grid points reduces the computational effort of one iterative sweep. The obtained solution on the coarse grid – the so-called *coarse-grid correction* – is then interpolated to the fine grid and used to correct the solution. The alternate repetition of relaxation sweeps on the fine grid and the coarser grids leads to an iterative procedure which has become known as the *multigrid technique*.

5.3 Multiple grids

For multigrid methods, a hierarchy of nested grids has to be constructed. The finest grid is usually the one where the solution is to be obtained, coarser grids serve to accelerate convergence.

There are two basic multigrid approaches: geometric (GMG) and algebraic multigrid (AMG) methods. In a geometric multigrid approach, coarser grid levels as well as the various multigrid components are constructed from the given fine grid.

Contrarily, in an algebraic multigrid method, no information is used concerning the grid on which the governing partial differential equation is discretized. For the construction of coarse grid operators exclusively the information provided by the fine grid operator is used. This approach is especially suited for multigrid methods employed on highly unstructured grids. A thorough introduction to AMG methods is found in [31] and references therein.

Since the grids used in the ANTARES code are structured, the geometric multigrid approach is employed within this thesis. Two different types of grids are considered: an equidistantly spaced grid described by Cartesian coordinates and a radially stretched but horizontally equidistant grid where spherical coordinates are employed. The stretching ratio is defined by introducing the *stretching factor* q . The discrete radial coordinates of the grid points are then given by

$$r_i = r_{\text{top}} - \frac{(q^i - 1)}{q^n - 1}(r_{\text{top}} - r_{\text{bot}}) \quad (5.3.1)$$

where r_{top} and r_{bot} refer to the prescribed maximal (top) and minimal (bot) radius (see also [36]). Note that the nomenclature follows the ANTARES-convention of specifying the downward pointing axes as the x-axis (this stems from the physically intuitive approach to specify the x-direction as the direction pointing along the direction of gravity).

Coarser grids are readily defined by selecting grid points corresponding to the desired coarsening ratio. Figure 5.2 depicts the grids considered and the choice of coarser grid points in case a coarsening ratio $c_c = 2$ and a hierarchy of three grids is used. Either Dirichlet- or Neumann boundary conditions are employed in x-direction/r-direction whereas the solution is periodically continued in horizontal (y-/ ϕ -) direction. This choice of boundary conditions is reflected in the selection of the coarser grid points.

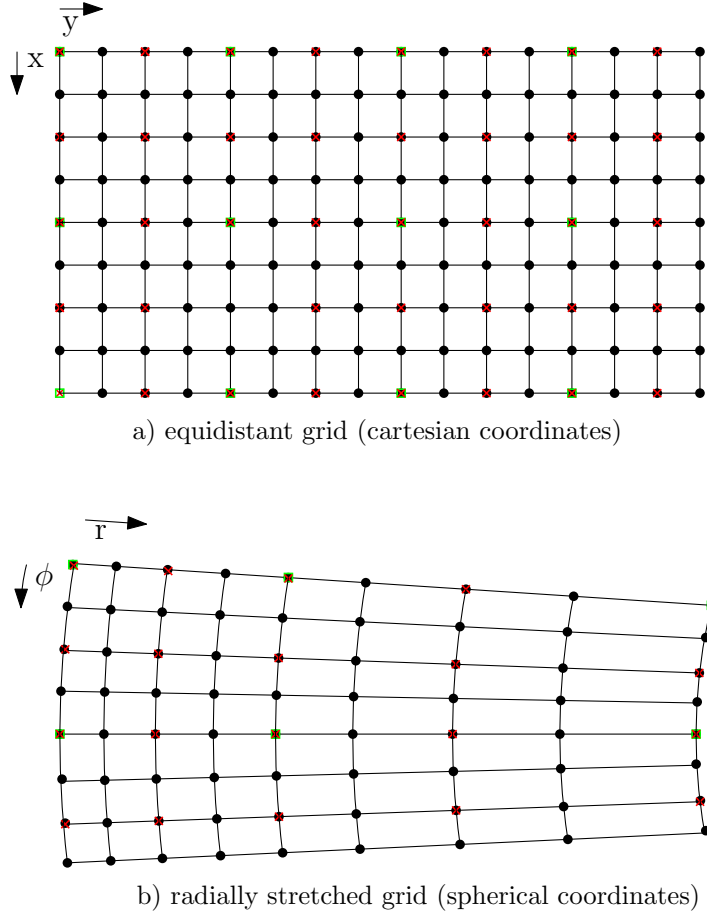


Figure 5.2: The grids used within the ANTARES code are depicted. Furthermore, the choice of coarser grid points in case a coarsening ratio of $c_c = 2$ and the employment of 3 grid levels is illustrated. The black discs denote the grid points of the fine grid G_1 , the red boxes denote the grid points of the next-coarser grid G_2 and another coarsening leads to the third grid G_3 indicated by green boxes. Note that in this choice already the boundary conditions (Dirichlet/Neumann in x -, respectively r -direction and periodic boundary conditions in y -, respectively ϕ -direction) are considered.

5.4 Discretization Issues

A suitable discretization is indispensable for the success of the multigrid method. In the present case, the finite element (FEM) technique is employed since it provides symmetric and positive definite matrices also if a stretched grid and spherical coordinates are used.

In the FEM approach, the idea is to approximate the solution in a finite-dimensional subspace of the function space that provides the abstract framework of the problem given. This subspace is spanned by *ansatz functions*. The choice of the ansatz functions determines the accuracy of the solution.

Following the suggestion of [3], the ansatz functions associated with the different grid levels are chosen such that

$$[\psi_{1\ 1}^{l_{\max}}, \dots, \psi_{n^{l_{\max}}\ m^{l_{\max}}}^{l_{\max}}] = S_{l_{\max}} \subset \dots \subset S_1 = [\psi_{1\ 1}^1, \dots, \psi_{n^1\ m^1}^1] \quad (5.4.1)$$

where $S_{l_{\max}}$ is the finite element space associated with the coarsest grid level and S_1 the one associated with the finest grid. In the present context, (bi)linear ansatz functions are chosen such that

$$\psi_{i,j}^l(x_{k,m}^l) = \delta_{ijkm} \quad (5.4.2)$$

where δ denotes the Kronecker symbol. For further details on the choice of ansatz functions, the assembly of the stiffness matrix and the treatment of boundary conditions, the reader is referred to Appendix B.1.

5.5 Linear Multigrid: The Correction Scheme (CS)

In this section, the multigrid algorithm for solving a system of linear equations

$$Au = f \quad A \in \mathbb{R}^{n \times n} \quad u, f \in \mathbb{R}^n \quad (5.5.1)$$

as outlined in [3, 4, 13] is presented.

The variables in the multigrid formulation carry 3 indices:

- l denotes the grid level
- k the number of the current multigrid cycle
- m the substep in a multigrid cycle

In the following algorithm, r and p denote the interpolation operators used to transfer the variables between the different grid levels. According to their function, the fine-to-coarse interpolation operator is called the *restriction* and denoted by r whereas the coarse-to-fine interpolation operator p is referred to as the *prolongation*. The choice of the interpolation operators as well as the relaxation procedure M and the coarse-grid operator A_{l+1} is discussed further below.

5. MULTIGRID METHODS

Multigrid-Iteration MGM^l

Let $x^{l,k,0}$ be an approximation in S_l .

First, ν_1 relaxation sweeps are executed,

$$x^{l,k,1} = M^{\nu_1} x^{l,k,0} \quad (5.5.2)$$

Then the residual $d_l = f - A_l x^{l,k,1}$ is restricted to S_{l+1} . Thus there is

$$A_{l+1} \hat{y}^{l+1} = r d_l \quad (5.5.3)$$

If $l+1 = l_{\max}$, a direct solver is used to compute $\hat{y}^{l+1} = y^{l+1}$

If $l+1 < l_{\max}$ an approximation y^{l+1} of $\hat{y}^{l+1}$ is computed by performing λ steps of MGM^{l+1} with the starting value $y^{l+1,0,0} = 0$.

Then, there is

$$x^{l,k,2} = x^{l,k,1} + p y^{l+1} \quad (5.5.4)$$

Finally, ν_2 relaxation sweeps are performed,

$$x^{l,k,3} = M^{\nu_2} x^{l,k,2} \quad (5.5.5)$$

and the new approximation $x^{l,k+1,0} = x^{l,k,3}$ is obtained.

Note that in this algorithm the residual d_l of the fine grid is restricted to the coarser grid level. The idea behind this is that the residual will, after a few relaxation sweeps, still be large due to the low frequency components of the residual error. However, the components associated with high frequency modes will have been successfully damped. Thus, solving the system at a coarser grid level will provide faster smoothing of the low frequency part of the error, since the number of unknowns and thus the computational time for one iterative sweep is lessened. The error is therefore damped at a coarser grid level, thus resulting in a better approximation. At the fine grid level, the solution u is then corrected by this approximation. Therefore, the linear multigrid algorithm is also known as the *correction scheme (CS)*. In the following, one traversal of the algorithm above will be referred to as one *CS-iteration*.

A close look at the algorithm MGM^l reveals that there are many components and parameters which can be adjusted to maximize the convergence speed of the multigrid algorithm. Apart from the appropriate choice of the coarsening and discretization strategy, the intergrid operators, the smoothing routine, the number of pre- and post-relaxation sweeps ν_1 and ν_2 as well as the cycling pattern determined by λ and the number of grid levels influence the performance of the multigrid algorithm. To assess the basic properties of the multigrid method developed in this thesis, the following test examples have been chosen:

Example 5.5.1. Using Cartesian coordinates, the choice of

$$\kappa(x, y) = 1 \quad (5.5.6)$$

$$c(x, y) = 1 \quad (5.5.7)$$

$$f(x, y) = 3 \sin x \cdot \sin y \quad (5.5.8)$$

leads to the solution $u(x, y) = \sin x \cdot \sin y$. Homogeneous Dirichlet boundary conditions are employed in x-, periodic boundary conditions in y-direction. If not specified differently, the equation is solved on the domain $\Omega = [0, 2\pi] \times [0, 2\pi]$ which is discretized with 401×400 equidistantly spaced grid points. A hierarchy of three grids is used and the multigrid iteration is terminated if the residual norm on the finest grid drops below an error tolerance of $\epsilon_{\text{fine}} = 10^{-6}$.

To validate the multigrid solver using a radially stretched grid and employing spherical coordinates, the coefficient functions and the discretization are chosen the following way:

Example 5.5.2. Using spherical coordinates, the choice of

$$\kappa(r, \phi) = 1 \quad (5.5.9)$$

$$c(x, y) = 1 \quad (5.5.10)$$

$$f(x, y) = \frac{\cos \phi}{4r^{3/2}} + \sqrt{r} \cos \phi \quad (5.5.11)$$

leads to the solution $u(r, \phi) = \sqrt{r} \cos \phi^1$. If not specified differently, the equation is solved on the domain $\Omega = [0, 2\pi] \times [0, 2\pi]$ which is discretized with 353×304 grid points. A stretching factor $q = 1.007$ is employed. Again, Dirichlet boundary conditions in r- and periodic boundary conditions in ϕ -direction are used. A hierarchy of three grids is used and the multigrid iteration is terminated if the residual norm on the finest grid drops below an error tolerance of $\epsilon_{\text{fine}} = 10^{-6}$.

To analyze the efficiency of the multigrid iteration, the *smoothing rate* $\bar{\rho}$ is introduced. The smoothing rate $\bar{\rho}$ is defined as the factor by which the residual norm decreases in a multigrid iteration,

$$\bar{\rho} = \frac{\|d_l^n\|}{\|d_l^{n-1}\|} \quad (5.5.12)$$

where d_l^n denotes the residual (on the finest grid) after the n^{th} multigrid cycle. Similarly, one can define an average smoothing rate over m multigrid cycles by

$$\bar{\rho}_{\text{av}} = \sqrt[m]{\frac{\|d_l^m\|}{\|d_l^1\|}}. \quad (5.5.13)$$

It is important to note that the multigrid convergence properties as well as the resulting wall-clocktimes are intricately intertwined. It is almost impossible to list all the used parameters for

¹Note that due to the employment of spherical coordinates, the ∇ -operator must be interpreted accordingly. Details on this are found in Appendix B.1.

5. MULTIGRID METHODS

every test presented below and maintain legibility for the reader. Thus, the different tests are such as to highlight the impact of certain features but without considering the overall convergence and wallclocktime (WCT). Instead, a *normed wallclocktime* (WCT_{norm}) is given which is computed the following way: One value of the testsuite considered is chosen as the reference value WCT_{ref} . The normed wallclocktimes are then computed as $WCT_{\text{norm}} = WCT/WCT_{\text{ref}}$. The overall performance of the multigrid solver especially in comparison with the solver developed by [11] will be discussed in Chapter 6.

5.5.1 Coarse grid operators

The coarse grid operator A_{l+1} should be a proper homogenization of the fine grid operator A_l . A straightforward way to obtain the coarse grid operator A_l in a geometric multigrid approach is to discretize equation (4.6.5) separately on coarser grids, without considering the different grid levels. This strategy known as *discretization coarse grid approximation* (*DCA*) has the advantage that all operators have the same structure (i.e. tridiagonal, pentadiagonal,...) and is recommended by [4] for smooth problems.

Another, widely used approach is to construct A_{l+1} from A_l using the restriction and prolongation operators,

$$A_{l+1} = rA_l p \quad (5.5.14)$$

This approach is called *Galerkin coarse grid approximation* (*GCA*). For the actual computation of the coarse grid operator A_{l+1} via equation (5.5.14), *stencil notation* is employed. A brief introduction to stencil notation is found in Appendix C.1.

The GCA approach entails the disadvantage that the structure of the operator A changes in higher dimensions, in two dimensions, for example, the fine grid operator has only five non-zero entries per line but the corresponding coarse grid operators have nine non-zero diagonals. It is therefore necessary to adapt the smoothing routine and the direct solver accordingly.

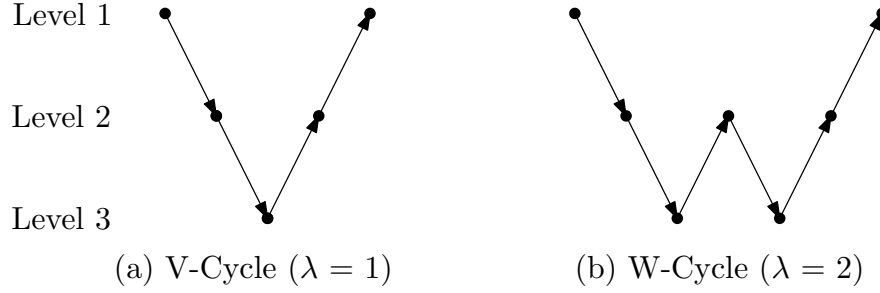
The GCA, though, ensures perfect coupling between the operators of the different grid levels. The test results reported in the Tables 5.1 and 5.2 show that whereas for the test example on the Cartesian grid both coarsening strategies render a very similar performance, for the test equation on the spherical grid the additional coupling of the GCA is absolutely necessary to obtain a reasonable multigrid performance.

coarsening method	mg-cycles	$\bar{\rho}_{\text{av}}$	WCT_{norm}
DCA	8	0.077	1
GCA	8	0.074	1.03

Table 5.1: Influence of the coarsening strategy in Example 5.5.1 (Cartesian grid).

coarsening method	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}
DCA	no convergence	-	-
GCA	8	0.1	1

Table 5.2: Influence of the coarsening strategy in Example 5.5.2 (stretched spherical grid).


 Figure 5.3: The grid traversal depending on different choices of λ is schematically depicted for a hierarchy of three grids. Level 1 denotes the finest grid.

5.5.2 Cycling Pattern and Grid Hierarchy

The cycling pattern of the multigrid method is determined by the parameter λ appearing in the formulation of the linear multigrid algorithm MGM¹. Common choices are $\lambda = 1$, resulting in the so-called *V-cycle* pattern, or $\lambda = 2$, leading to a *W-cycle*. Figure 5.3 depicts the grid traversal in each case. Furthermore, a combination of both (λ alternates between 1 and 2) is known as *F-cycling*.

It is said in [4] that the use of W-cycles is the safest way to traverse the grid hierarchy since the equation is solved to a much higher accuracy in one multigrid cycle than employing V-cycling. F-cycles are considered less expensive, but similarly efficient as W-cycles. V-cycles, however, have the advantage of saving a considerable fraction of the computational work. [4] recommends the use of V-cycling for problems where the (two-level) smoothing rate $\bar{\rho}_{av}$ is small and thus the multigrid method is efficient. This is actually the case in regular elliptic cases such as the present one.

The test on the cycling pattern reported in the Tables 5.3 and 5.4 indicate that indeed the V-cycle is the optimal strategy for traversing the grid hierarchy. The average smoothing rates $\bar{\rho}_{av}$ obtained by the use of W- and F-cycling are only minimally better than the V-cycling average smoothing rate. The wallclocktimes, though, increase drastically due to the frequent visits of the coarsest grid level.

cycling mode	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}
V-Cycle	8	0.074	1
W-Cycle	8	0.070	1.85
F-Cycle	8	0.070	1.40

Table 5.3: Influence of the cycling pattern in Example 5.5.1 (Cartesian grid).

5. MULTIGRID METHODS

cycling mode	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}
V-Cycle	8	0.10	1
W-Cycle	8	0.088	1.38
F-Cycle	9	0.114	1.37

Table 5.4: Influence of the cycling pattern in Example 5.5.2 (stretched spherical grid).

The number of grid levels has a severe impact on the multigrid convergence. In both test examples the multigrid smoothing rate drops significantly if more than three grid levels are employed (see Tables 5.5 and 5.6). This is due to the fact that if the resolution on the coarsest grid is too low to obtain a good approximation to the fine grid solution, more multigrid cycles are needed to obtain a solution on the finest grid. From this point of view, it is obvious that employing a hierarchy of only two grids leads to the highest smoothing rate, however, since the coarse grid solution has to be obtained on a quite well resolved mesh, the computational time taken to obtain a solution is also significant. Thus, the resolution of the coarse grid equation has been included in the Tables 5.5 and 5.6. The results permit the conclusion that optimal multigrid efficiency is obtained if the coarse grid equation is resolved with at least 50 but maximally 100 grid points in each direction. If the resolution is less than 25×25 grid points on the coarsest grid level, the multigrid smoothing rate increases remarkably and the computation of a sufficiently accurate approximative solution is slowed down significantly.

grid levels	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}	coarse grid resolution
2	8	0.066	1	201×200
3	8	0.073	0.39	101×100
4	13	0.196	0.39	51×50
5	20	0.356	0.55	26×26

Table 5.5: Influence of the number of grid levels in Example 5.5.1 (Cartesian grid).

grid levels	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}	coarse grid resolution
2	8	0.086	1	177×152
3	8	0.096	0.46	89×76
4	11	0.175	0.54	45×38
5	16	0.305	0.76	23×19

Table 5.6: Influence of the number of grid levels in Example 5.5.2 (stretched spherical grid).

5.5.3 Intergrid operators

An integral part of the multigrid iteration is the interpolation between the different grids. The fine-to-coarse interpolation operator is called *restriction* whereas its counterpart is referred to as

the *prolongation*. Many choices have been proposed by [4, 13], however, within the finite element discretization, both operators are canonically defined.

Let $S_{l_{\max}} \subset \dots \subset S_1$ be the finite element spaces associated with the grids in use, S_1 denoting the finest and $S_{l_{\max}}$ the coarsest grid level. Typically, the meshwidth in the grids follows $\frac{h_{i+1}}{h_i} = c_c$, $i = 1, \dots, l_{\max}$ and c_c denotes the coarsening ratio. In this thesis, $c_c = 2$ is used, however, any value might be employed.

In the following, the restriction operator from S_l to S_{l+1} is derived. The derivation follows the lines of [3].

Let S_{l+1} be a subset of S_l , $S_{l+1} \subset S_l$. Then, ψ_j^{l+1} can be written as a linear combination of the basis functions of S_l ,

$$S_l \supset S_{l+1} \ni \psi_j^{l+1} = \sum_{i=1}^n r_{ji} \psi_i^l \quad (5.5.15)$$

Let x_t^l denote the grid points on the l-level grid, $t = 1, \dots, n$, then

$$\psi_i^l(x_t^l) = \delta_{it} \quad (5.5.16)$$

holds. Be $v \in S_l$. Then v can be expressed as $\sum_{i=1}^n v(x_i^l) \psi_i^l$.

Since $\psi_j^{l+1} \in S_l$,

$$\psi_j^{l+1} = \sum_{i=1}^n \psi_j^{l+1}(x_i^l) \psi_i^l \quad (5.5.17)$$

Comparing the coefficients of equations (5.5.15) and (5.5.17), this leads to

$$r_{ji} = \psi_j^{l+1}(x_i^l) \quad (5.5.18)$$

In case of finite element equations, the canonical definition of the corresponding prolongation operator p is

$$p = r^* \quad (5.5.19)$$

where r^* is the *adjoint operator* of r . However, attention has to be paid to the scaling factors of the different scalar products defined in different function spaces. In the present case, the relation between p and r reads

$$p = r^* = \frac{h_x^{l+1}}{h_x^l} r^T \quad (5.5.20)$$

r^T denoting the *transpose* of r . Note that since both r and p are real matrices, the adjoint operator of r reduces to its transpose,

$$r^* = r^T \quad (5.5.21)$$

5. MULTIGRID METHODS

For the denotation of restriction and prolongation operators, stencil notation is used (see Appendix C.1).

The restriction operator of the equidistant grid reads

$$r_{1D} = \frac{1}{2} \begin{bmatrix} \frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix} \quad (5.5.22)$$

and

$$r_{2D} = \frac{1}{4} \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{q}{4} & \frac{q}{2} & \frac{q}{4} \end{bmatrix} \quad (5.5.23)$$

in one respectively two spatial dimensions. For the radially stretched grid, the restriction operator depends on the stretching factor q and reads

$$r_{1D} = \frac{1}{2} \begin{bmatrix} \frac{1}{1+q} & 1 & \frac{q}{1+q} \end{bmatrix} \quad (5.5.24)$$

respectively

$$r_{2D} = \frac{1}{4} \begin{bmatrix} \frac{1}{2+2q} & \frac{1}{1+q} & \frac{1}{2+2q} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{q}{2+2q} & \frac{q}{1+q} & \frac{q}{2+2q} \end{bmatrix}. \quad (5.5.25)$$

5.5.4 Smoothing Routines

The *smoothing* or *relaxation* procedure serves to damp the high frequency components of the residual error. Generally, any iterative method is suitable for this task. However, it is indicated in the literature that the Gauss-Seidel procedure is the most efficient choice of a smoother in the context of the multigrid iteration (c.f. [44], [24]).

The basic Gauss-Seidel method traverses the grid in a lexicographic order and uses all the previously computed values to update the value u_i ,

$$u_i^{new} = \frac{1}{\alpha_{ii}} \left(f_i - \sum_{j=1, j < i}^n \alpha_{ij} u_j^{new} - \sum_{j=1, j > i}^n \alpha_{ij} u_j^{old} \right) \quad (5.5.26)$$

where $A = (\alpha_{ij})$. Different sweeping patterns have been proposed (c.f. [47, p.29ff, p.132ff]). A very popular choice is to use *red-black*-Gauss-Seidel (GS-RB) smoothing where every other grid point is considered a red point whereas the remaining ones are colored black. In the first half-step of a GS-RB sweep, all the red points are updated simultaneously and in the second half-step, the new values on the black points are computed.

Especially for anisotropic problems, [47] recommends *line-smoothing*, where all entries belonging to a certain x- or y-line are updated simultaneously. *Zebra line-smoothing* (ZGS) is the analogon to the GS-RB sweep - in a first half-step, every other line is updated simultaneously, whereas in a second half-step, the remaining lines are treated. Figure 5.4 depicts different sweeping patterns.

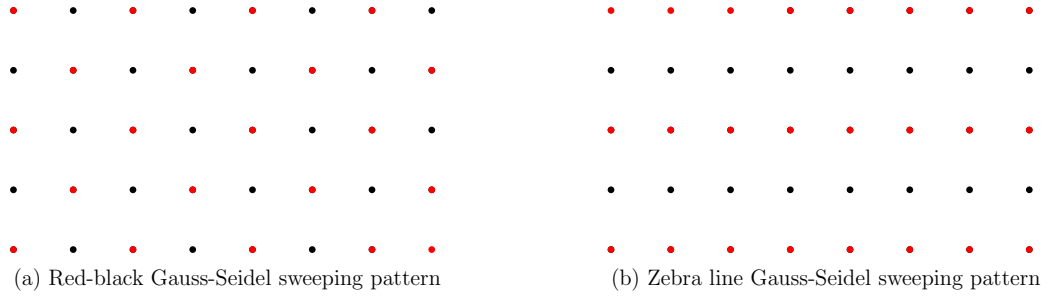


Figure 5.4: Schematic depiction of different sweeping patterns. In the first half-sweep, the red points are updated simultaneously, then, using the information obtained from the red grid points, the black ones are computed.

In the present context, the choice of the smoother is heavily guided by its parallelization properties. If the lexicographic GS-method is used, dependencies are present since the idea is to use the most recent values of u wherever possible. Therefore, the degree of parallelism of this scheme is very small.

In case of GS-RB or GS-Zebra-line smoothing, each step of the Gauss-Seidel relaxation consists of two half steps. In the first half-step, all red points/lines are treated simultaneously and independently of each other, in the second half step all black grid points/lines are treated, using the updated values in the red points/lines. Thus, in a domain decomposition approach, only little communication is required between the processors.

Another possibility is to use a few iterations of the Conjugate Gradient (CG) method employed as the direct solver on the coarsest grid. Details on the CG-Algorithm are found in the sections 5.5.5 and 5.7.

The impact of different smoothing routines on the multigrid solver differs drastically between the Cartesian and the stretched spherical grid. For the Cartesian grid, the results reported in Table 5.7 indicate the red-black Gauss-Seidel smoother is very effective, a result that agrees well with the findings reported in the literature (c.f. [47]). However, performing a few steps with the Conjugate Gradient algorithm provides even faster, for the test example considered even immediate, convergence.

For the stretched spherical grid, the multigrid smoothing rates are poor by multigrid standards. Similar to the results for Example 5.5.1, the red-black Gauss-Seidel smoother achieves the highest convergence rate of the Gauss-Seidel based smoothers. It is not surprising, though, that no convergence is achieved if Zebra y-line Gauss-Seidel smoothing is employed. The stretched grid inherits an anisotropy aligned with the x-direction. Therefore, strong coupling effects take place in this direction. A smoother ignoring this distinguished direction is bound to yield very poor convergence results (see [47]). However, using a few steps of the Conjugate Gradient algorithm as a smoother yields an also for multigrid standards acceptable convergence rate of 0.1.

The number of pre- and postsmoothing steps ν_1 and ν_2 also has a significant influence on the multigrid convergence. The test results reported in Tables 5.9 and 5.10 show that too little smoothing entails the need of many multigrid cycles, rendering the solution process computation-

5. MULTIGRID METHODS

ally expensive. Increasing the number of smoothing steps leads to a decrease of multigrid cycles and a better average smoothing rate $\bar{\rho}_{av}$. Too many pre-and postsmoothing steps, though, also increase the computational time without diminishing the smoothing rate substantially. Therefore, the aim is to find the configuration minimizing the computational time and having an acceptable smoothing rate. For Example 5.5.1, this point is achieved employing $\nu_1 = \nu_2 = 2$ smoothing steps (the differences in the computational times between $\nu_1 = \nu_2 = 2$ and $\nu_1 = \nu_2 = 5$ are too small to be significant). The results of Example 5.5.2 show that if a spherical, stretched grid is employed, the need for smoothing steps is much greater than in the case of a Cartesian grid. For Example 5.5.2, the choice of $\nu_1 = \nu_2 = 8$ appears to be optimal.

smoother	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}
GS-LEX	8	0.069	1
GS-RB	6	0.025	0.69
ZGSx	8	0.074	0.88
ZGSy	8	0.069	1.35
CG	immediate convergence	-	-

Table 5.7: Influence of the smoothing routine in Example 5.5.1 (Cartesian grid).

smoother	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}
GS-LEX	33	0.562	1
GS-RB	30	0.53	0.81
ZGSx	39	0.62	1.06
ZGSy	no convergence	-	-
CG	8	0.096	0.2

Table 5.8: Influence of the smoothing routine in Example 5.5.2 (stretched spherical grid).

ν_1	ν_2	mg-cycles	$\bar{\rho}_{av}$	WCT _{norm}
1	1	15	0.255	1
2	2	8	0.074	0.54
3	3	7	0.047	0.51
4	4	6	0.031	0.47
5	5	6	0.027	0.53

Table 5.9: Influence of the number of pre- and postsmoothing steps ν_1 and ν_2 in Example 5.5.1 (Cartesian grid).

5.5.5 Direct Solvers

The multigrid algorithm requires the solution of the coarsest grid equation. Any method for solving a system of linear equations is suitable for this task. In the present case the solver of [11] based on the *Conjugate Gradient (CG)* algorithm is employed.

ν_1	ν_2	mg-cycles	$\bar{\rho}_{\text{av}}$	WCT _{norm}
2	2	53	0.7	1
5	5	17	0.323	0.44
8	8	8	0.096	0.27
10	10	7	0.063	0.27
12	12	7	0.053	0.30

Table 5.10: Influence of the number of pre- and postsmoothing steps ν_1 and ν_2 in Example 5.5.2 (stretched spherical grid).

The CG-Algorithm is one of the best known iterative techniques for solving a symmetric and positive definite system of linear equations. The idea behind it is to find an approximate solution to the given system in some finite dimensional function space. This subspace is spanned by the residual vectors and thus enlarged each iteration until convergence is reached. For details on this technique, the reader is referred to [11, 44].

To ensure convergence of the CG-Algorithm, the operator A has to be symmetric and positive definite. Whereas on the fine grid this is ensured by the FEM discretization, it is easy to see that the operators obtained by the Galerkin coarsening approach also fulfill these requirements.

Lemma 5.5.3. *Let A be a symmetric and positive definite matrix. Then the matrix $\bar{A} = rAp$ with r and p denoting the restriction and prolongation operators of section 5.5.3 is also symmetric and positive definite.*

Proof. Symmetry:

$$\bar{A}^T = (rAp)^T = p^T A^T r^T \stackrel{\text{eq. (5.5.20)}}{=} \frac{h_x^{l+1}}{h_x^l} r A^T \frac{h_x^l}{h_x^{l+1}} p \stackrel{A^T=A}{=} r A p = \bar{A} \quad (5.5.27)$$

Positive definiteness:

It is to be shown that $\forall v \in \mathbb{R}^{n/2}, v \neq 0, v^T \bar{A} v > 0$.

Consider an arbitrary $v \in \mathbb{R}^{n/2}, v \neq 0$. It is easy to see that the prolongation² $p : \mathbb{R}^{n/2} \rightarrow \mathbb{R}^n$ is a matrix of full rank, $\text{rg}(p) = n/2$. It follows from

$$\frac{n}{2} = \dim(\mathbb{R}^{n/2}) = \text{rg}(p) + \dim \ker(p) = \frac{n}{2} + 0 \quad (5.5.28)$$

that $\ker p = \{0\}$. Thus, for $v \neq 0$, there is $pv \neq 0$.

²Actually, the prolongation operator p operates between the finite element spaces S_{l+1} and S_l . Since each element of S_{l+1} (resp. S_l) is uniquely represented by its coefficient array $(c_1, \dots, c_{n/2})$ (resp. $(c_1, \dots, c_n)$) of the corresponding linear combination of basis vectors, those spaces are isomorphic to $\mathbb{R}^{n/2}$ (resp. $\mathbb{R}^n$). Thus, it suffices to consider p as a linear map between $\mathbb{R}^{n/2}$ and $\mathbb{R}^n$.

5. MULTIGRID METHODS

Consider now $\tilde{v}^T := v^T r$. Then, there is $\tilde{v} = (v^T r)^T = r^T v = pv \neq 0$ and thus $\tilde{v} \neq 0$. It follows that

$$0 < \tilde{v}^T A \tilde{v} = v^T r A p v = v^T \bar{A} v \quad (5.5.29)$$

□

Furthermore, the influence of the multigrid iteration on the termination criterion of the direct solver has been tested. The results in Table 5.11 indicate that it is necessary to employ an error tolerance ϵ_{coarse} on the coarsest grid which is at least one magnitude more restrictive than ϵ_{fine} . However, prescribing an error tolerance too fine does not have an effect on the average smoothing rate $\bar{\rho}_{\text{av}}$ and only serves to slow down the computation since more iterations are necessary to obtain a sufficiently accurate coarse grid solution.

ϵ_{coarse}	mg-cycles	$\bar{\rho}_{\text{av}}$	WCT _{norm}
10^{-6}	-	-	-
10^{-7}	8	0.074	1
10^{-8}	8	0.074	0.94
10^{-9}	8	0.074	0.97
10^{-10}	8	0.074	1.09

Table 5.11: Influence of the error tolerance ϵ_{coarse} on the coarsest grid in Example 5.5.1 (Cartesian grid).

5.5.6 Comparison between the linear multigrid solver and a solver based on the Conjugate Gradient algorithm

Estimations on the computational work a solver requires have been conducted for different algorithms. It has turned out (c.f. [47, p. 14]) that the multigrid approach is the most efficient technique involving only $O(N \log \epsilon)$ (N being the number of unknowns, i.e. the number of grid points³) computations to solve the classical Poisson equation (i.e. $c = 0, \kappa = 1$) whereas traditional iterative solvers such as the Jacobi or the Gauss-Seidel method have a computational complexity of $O(N^2 \log \epsilon)$ ([47, p. 14]). Furthermore, an empirical study conducted in [24] demonstrates impressively that for the classical Poisson equation the solver based on a multigrid technique renders accurate results considerably faster (comparing the computational time) than solvers based on other algorithms. To show that the solver developed for the purposes of this thesis is similarly efficient, its performance is tested against a linear solver based on the Conjugate Gradient (CG) algorithm. This solver has been developed by H. Grimm-Strele (see [11]) and has been used in the first evaluation of the efficiency of IMEX Runge-Kutta methods in [29]. Note that the computational complexity of the CG-solver amounts to $O(N^{3/2} \log \epsilon)$ [47, p. 14].

³In the estimations of [47], it is assumed that the iterative solvers provide a solution in the range of the discretization accuracy. This is reflected by the $\log \epsilon$ -term.

In this testsuite, Example 5.5.1 has been solved with both the multigrid and the Conjugate Gradient solver on differently resolved computational domains. Table 5.12 gives the (normalized) wallclocktimes. Since the solver developed by [11] solely supports Cartesian geometries, only Example 5.5.1 is evaluated.

grid resolution	WTC_{norm} (WTC) of the MG-solver	WTC_{norm} (WTC) of the CG-solver
109×112	1 (0.294 s)	1 (0.577 s)
209×212	4.74 (1.396 s)	7.62 (4.344 s)
309×312	13.94 (4.099 s)	28.35 (16.357 s)
409×412	28.93 (8.504 s)	70.82 (40.865 s)
509×512	51.59 (15.167 s)	144.22 (83.222 s)

Table 5.12: Comparison of the efficiency of the multigrid solver and the solver based on the Conjugate Gradient algorithm ([11]) in Example 5.5.1. The normed wallclocktimes demonstrate the impact of a higher resolution on the computational time of the solver whereas the real wallclocktimes permit the comparison of the two different solvers. The simulations have been carried out on one single CPU-core.

The results show that even if the domain is coarsely resolved (109×112 grid points), the multigrid solver is about 1.96 times faster than the solver based on the Conjugate Gradient algorithm. As soon as the resolution increases, this factor grows significantly. At a resolution of 309×312 , the difference in wallclocktimes between the solvers is approximately a factor of 4, and using 509×512 grid points, the multigrid solver is about 5.5 times faster than the CG-solver.

5.6 Multigrid methods for a non-linear problem

It is a strength of the multigrid technique that it can be easily adapted to solve a system of nonlinear equations. The resulting iteration known as the *full approximation storage (FAS) scheme* allows the solution of a nonlinear system in a matrix-free manner, retaining the speed and accuracy inherent to the multigrid approach.

Alternatively, the solution of the nonlinear problem can be tackled by a *Multigrid-Newton*-procedure. The idea hereby is to linearize the nonlinear equations by a Newton-technique and to solve the arising linear system of equations employing the linear multigrid scheme.

In the following sections, both approaches are introduced and their advantages and disadvantages within the context of the ANTARES framework are discussed.

5.6.1 The Full Approximation Storage (FAS) scheme

Within the FAS-iteration, almost the same steps as in the linear Correction-Scheme (CS) iteration are performed; indeed, it is easy to see that if the FAS algorithm is applied to a linear problem, it reduces correctly to the CS-algorithm.

5. MULTIGRID METHODS

The FAS-Iteration FAS^l

Let $u^{l,k,0}$ be an approximation in S_l .

First, ν_1 relaxation sweeps are performed,

$$u^{l,k,1} = S^{\nu_1} u^{l,k,0} \quad (5.6.1)$$

For the coarse-grid correction, the residual $d_l = f_l - \mathcal{A}_l(u^{l,k,1})$ as well as the approximate solution $u^{l,k,1}$ is interpolated to the coarse grid,

$$d_{l+1} = r d_l \quad (5.6.2)$$

$$u^{l+1,k,0} = \tilde{r} u^{l,k,1} \quad (5.6.3)$$

The right hand side f of the coarse grid equation is defined as

$$f_{l+1} = \mathcal{A}_{l+1}(u^{l+1,k,0}) + r d_l \quad (5.6.4)$$

Let $\hat{v}_{l+1}$ be the solution of

$$\mathcal{A}_{l+1}(v) = f_{l+1} \quad (5.6.5)$$

If $l+1 = l_{\max}$, determine the exact solution $\hat{v}_{l+1}$ and set $\hat{v}_{l+1} = v^{l+1}$.

If $l+1 < l_{\max}$, determine an approximate solution v^{l+1} by performing λ steps with the algorithm FAS^{l+1} using the starting value $u^{l+1,0,0}$

Set

$$u^{l,k,2} = u^{l,k,1} + p(v^{l+1} - u^{l+1,k,0}) \quad (5.6.6)$$

Finally, ν_2 relaxation sweeps are performed,

$$u^{l,k,3} = S^{\nu_2} u^{l,k,2} \quad (5.6.7)$$

The restriction operator $\tilde{r}$ used in equation (5.6.3) does not necessarily equal r . It is outlined in [4] that in some cases it can be beneficial to use a restriction of higher order than r to transfer the approximate solution u to a coarser grid.

Although the FAS-algorithm treats the nonlinear system of equations as such, it still offers some possibilities of linearizing the system to solve. To apply a linear smoothing procedure, the nonlinear equation could be locally linearized on each grid. However, nonlinear smoothing procedures such as the nonlinear Gauss-Seidel or nonlinear Jacobi-algorithm (see [24]) rely on a Newton technique which involves the need of local linearization and thus of computing derivatives, although not the entire Jacobian matrix is required. To obtain a nonlinear and derivative-free solver, the standard choice for both the smoothing routine and the direct solver on the coarsest grid is the *fixed-point iteration*. Ensuring convergence of the fixed-point iteration, though, requires additional considerations.

The Fixed-Point Iteration

The fixed-point iteration is a method to determine solutions of a nonlinear (system of) equation(s) of appearance

$$\Phi(u) = u \quad (5.6.8)$$

To rearrange the nonlinear equation within the context of IMEX-Runge Kutta methods into the desired shape (5.6.8), definition (4.6.3) is invoked. The implicit stage of an IMEX Runge-Kutta method consists of solving the equation

$$y^* + \Delta t a_{ii} G(u) = u. \quad (5.6.9)$$

Consequently the prescription of equation (5.6.8) reads

$$\Phi(u) = y^* + \Delta t a_{ii} G(u) = u \quad (5.6.10)$$

The Banach Fixed-Point theorem (see for example [48, p. 12])⁴ states that the fixed-point iteration

$$u^{i+1} = \Phi(u^i) \quad (5.6.11)$$

converges if and only if the operator Φ is a *contraction*, i.e.

$$\exists q \in \mathbb{R}^n \ 0 < q < 1 \quad \forall u, u' \in \mathbb{R}^n \quad \|\Phi(u) - \Phi(u')\| \leq q \|u - u'\|. \quad (5.6.12)$$

To apply the Banach Fixed Point theorem to (5.6.10), it is assumed that the flux function G is *Lipschitz continuous*, that is

$$\exists L \in \mathbb{R} \quad \forall x \in \Omega : \|G(x) - G(x')\| \leq L \|x - x'\| \quad (5.6.13)$$

Therefore, there is

$$\|\Phi(u) - \Phi(u')\| = \Delta t a_{ii} \|G(u) - G(u')\| \quad (5.6.14)$$

$$\leq \Delta t a_{ii} L \|u - u'\| \quad (5.6.15)$$

which leads to the conclusion that Φ is a contraction if and only if

$$\Delta t a_{ii} L < 1. \quad (5.6.16)$$

Thus, to ensure convergence of the fixed point iteration, it is mandatory to ensure that the criterion $\Delta t \cdot a_{ii} \cdot L < 1$ is met. This, however, is equivalent to an additional timestep restriction.

⁴Note that the Banach Fixed Point theorem is usually formulated on general metric spaces. In the present case, however, the metric space considered is the $\mathbb{R}^n$ and the mapping is the 2-norm $\|\cdot\|$.

5.6.2 The Multigrid-Newton method

The Newton iteration transforms the nonlinear problem $\mathcal{A}(u) = f$ into a sequence of linear ones which are subsequently solved by the linear multigrid scheme.

The basic iteration is taken from [13].

The Multigrid-Newton Iteration

The iteration is started using a suitable initial guess $u^{l,k,0}$. In a first step, the Jacobian $A^{l,k,0}$ of the nonlinear operator $\mathcal{A}_l(u^{l,k,0})$,

$$A^{l,k,0} := D\mathcal{A}_l(u^{l,k,0}) \quad (5.6.17)$$

is determined.

Secondly, λ_{CS} iterations of the linear multigrid correction scheme to the system

$$A^{l,k,0}v_0^l = d^{l,k,0} \quad (5.6.18)$$

with the starting value $v_0^l = 0$ and $d^{l,k,0} = f^{l,k,0} - \mathcal{A}_l(u^{l,k,0})$ denoting the defect of the nonlinear equation are performed. This results in the value v_1^l .

Finally, the correction

$$u^{l,k+1,0} = u^{l,k,0} - v_1^l \quad (5.6.19)$$

is evaluated. The procedure is repeated until the prescribed error tolerance is met.

It is outlined in [13] that the coupling of a Newton technique with the multigrid iteration leads to a conflict concerning the convergence behavior; whereas the Newton scheme converges quadratically, the multigrid convergence is only linear. To obtain quadratic convergence of the combined algorithm, [13] suggests to perform $\lambda_{CS} > 1$ multigrid cycles in equation (5.6.18) instead of merely one. Unfortunately, it is not obvious how to guess the optimal value of λ_{CS} .

It is known that the success of the Newton algorithm heavily depends on the initial guess the iteration is started with. To enlarge the domain of admissible starting values, an under-relaxed (damped) correction

$$u^{l,k+1,0} = u^{l,k,0} - \omega v_1^l \quad 0 < \omega \leq 1 \quad (5.6.20)$$

may be used instead of equation (5.6.19). However, in the present context of time-dependent problems, it is assumed that the starting value given by the value of the previous timestep is sufficiently accurate to ensure convergence of the Newton technique even if equation (5.6.19) is used.

5.6.3 Comparison of the MG-FAS and the MG-Newton algorithm in view of the ANTARES framework

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 $\mathcal{A}$ entails a call to the EOS, which is, in case that a model employing realistic microphysics is considered, computationally expensive (see also section 2.4), this is considered a disadvantage. Furthermore, if the fixed-point iteration is used, the additional timestep restriction (5.6.16) has to be taken into account.

The obvious disadvantage of the Multigrid-Newton scheme is the need to assemble and store the global Jacobian matrix of the nonlinear system. However, in the present context, favorable circumstances are encountered.

The nonlinear equation to solve applying IMEX Runge-Kutta methods reads

$$-\nabla \cdot \left(\frac{K_{\text{rad}}(T)}{\rho} \left(\frac{\partial T}{\partial \varepsilon} \right)_\rho \nabla e_{\text{int } i} \right) + \frac{e_{\text{int } i}}{\Delta t a_{ii}} = \frac{e_{\text{int}}^*}{\Delta t a_{ii}} \quad (5.6.21)$$

(for a detailed derivation the reader is referred to Chapter 7). The unknown function to be determined is the temperature $u = T$. Let $(\mathbf{a}_k)$ denote the rows of the (discretized) nonlinear system of equations given by $\mathcal{A}(u)$. The Jacobian matrix of $\mathcal{A}(u)$ is then determined by

$$D\mathcal{A}(u) = \left(\frac{\partial \mathbf{a}_k}{\partial u_j} \right)_{kj} \quad (5.6.22)$$

Revising the discretization procedure outlined in Appendix B.1, it is easily seen that the derivatives needed to assemble $D\mathcal{A}(u)$ are $\frac{\partial K_{\text{rad } i}}{\partial u_j}$ and $\frac{\partial e_{\text{int } i}}{\partial u_j}$. In the models within the focus of this thesis, the radiative conductivity K_{rad} and the internal energy e_{int} are evaluated by the EOS. Luckily, the OPAL-EOS routine used in the ANTARES code automatically provides the necessary values to determine $\frac{\partial K_{\text{rad } i}}{\partial u_j}$ and $\frac{\partial e_{\text{int } i}}{\partial u_j}$. Thus, no additional computational effort apart from the call to the EOS has to be spent on the determination of the derivatives.

Another important question is whether the structure of Jacobian matrix operator differs from the original nonlinear one, which naturally has the same structure as the one arising from discretizing the linear general elliptic equation. In general, the structure of the Jacobian matrix differs, but in the present context, a thermodynamical principle proves to be very helpful: the EOS describes *local* coupling of thermodynamical quantities. Therefore, the radiative conductivity K_{rad} and the internal energy ϵ_i only depend on the temperature *at the same point*, i.e.

$$\frac{\partial K_{\text{rad } i}}{\partial u_j} = 0 \quad \text{if } i \neq j \quad (5.6.23)$$

$$\frac{\partial e_{\text{int } i}}{\partial u_j} = 0 \quad \text{if } i \neq j \quad (5.6.24)$$

The locality of the EOS as well as the use of bilinear finite elements in the discretization ensure that the Jacobian $D\mathcal{A}(u)$ effectively possesses the same structure as the original operator. Thus, for the solution of the linearized system, the linear multigrid correction scheme can be invoked

5. MULTIGRID METHODS

without any modifications. Furthermore, it is possible to reuse already allocated arrays in an implementation, and thus the memory requirements are reduced.

Contrarily to the MG-FAS scheme, the nonlinear operator $\mathcal{A}$ as well as the global Jacobian have to be determined only once per (outer) Newton-iteration.

It has already been mentioned that the call to the EOS is computationally expensive within the ANTARES framework. The following example estimates the impact of this call on the computational time of the nonlinear multigrid algorithms. All values assumed are an average of the different examples which have actually been tried. The computational times have been measured by using the profiling software *gprof* (available at <http://www.eclipse.org/linuxtools/projectPages/gprof/>).

Example 5.6.1. Assume that equation (5.6.21) is solved employing a hierarchy of 3 grids and the number of pre- and postsmoothing steps is $\nu_1 = \nu_2 = 5$. At the coarsest grid, 100 fixed-point iterations are necessary to determine a sufficiently accurate coarse grid solution and sufficient accuracy on the finest grid is reached after 3 multigrid cycles. The cost of calling the EOS and determine the nonlinear operator $\mathcal{A}$ is 0.1 s, the cost of determining the Jacobian (including the call to the EOS) also amounts to 0.1 s.

Thus, the computational time taken by the MG-FAS scheme for determining the nonlinear operator $\mathcal{A}$ is (note that the fixed-point iteration is assumed to be used as a smoother)

$$(3 \cdot 3 + 6 \cdot 5 + 3 \cdot 100) \cdot 0.1 \text{ s} = 33.9 \text{ s} \quad (5.6.25)$$

whereas the MG-Newton only spends

$$3 \cdot 3 \cdot 0.1 \text{ s} = 0.9 \text{ s} \quad (5.6.26)$$

setting up the relevant operator.

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 (5.6.21) arising in the context of IMEX Runge-Kutta methods. Table 5.13 summarizes the advantages and disadvantages of each approach.

5.6.4 Adaptation of the MG-Newton technique

In this section the basic functionality of the developed MG-Newton solver is tested. The test examples used are similar to those for the linear tests.

MG-FAS	MG-Newton
+ derivative free formulation	– need to calculate Jacobian + locality of EOS: same structure as corresponding linear operator
– timestep restriction: $\Delta t a_{ii} L < 1$	+ no additional timestep restriction
– determination of $\mathcal{A}$ on every grid level/in every iteration	+ determination of $\mathcal{A}$ once per outer Newton iteration

Table 5.13: Advantages and disadvantages of the MG-FAS and the MG-Newton algorithm.

Example 5.6.2. Using Cartesian coordinates, the choice of

$$\kappa(x, y) = 1.5 + u(x, y) \quad (5.6.27)$$

$$c(x, y) = u \quad (5.6.28)$$

$$f(x, y) = 3 \sin x \sin y + 2 \sin^2 x \sin y^2 - \sin y^2 \cos x \quad (5.6.29)$$

$$- \sin x^2 \cos y^2 + \sin x \sin y \quad (5.6.30)$$

leads to the solution $u(x, y) = \sin x \cdot \sin y$. Homogeneous Dirichlet boundary conditions are employed in x-, periodic boundary conditions in y-direction. If not specified differently, the equation is solved on the domain $\Omega = [0, 2\pi] \times [0, 2\pi]$ which is discretized with 409×412 equidistantly spaced grid points. A hierarchy of three grids is used and the multigrid iteration is terminated if the residual norm on the finest grid drops below an error tolerance of $\epsilon_{\text{fine}} = 10^{-6}$.

Example 5.6.3. Using spherical coordinates, the choice of

$$\kappa(r, \phi) = 5 + u(r, \phi) \quad (5.6.31)$$

$$c(r, \phi) = u(r, \phi) \quad (5.6.32)$$

$$f(r, \phi) = \frac{5 \cos \phi}{4r^{3/2}} - \frac{\sin u^2}{r} + \sqrt{r} \cdot \cos \phi \quad (5.6.33)$$

leads to the solution $u(r, \phi) = \sqrt{r} \cdot \cos \phi$. If not specified differently, the equation is solved on the domain $\Omega = [0, 2\pi] \times [0, 2\pi]$ which is discretized with 353×304 grid points. A stretching factor $q = 1.007$ is employed. Again, Dirichlet boundary conditions in x- and periodic boundary conditions in y-direction are prescribed. A hierarchy of three grids is used and the multigrid iteration is terminated if the residual norm on the finest grid drops below an error tolerance of $\epsilon_{\text{fine}} = 10^{-6}$.

In both test examples, the Newton-iteration is started using the initial guess $u_{\text{init}} = 0.5 u$. Since the positive definiteness of the linearized system cannot be ensured, it is not advisable to use the Conjugate Gradient algorithm within the linear multigrid iteration. Instead, the Zebra x-line Gauss-Seidel relaxation scheme is employed to solve the coarse grid equation.

λ_{CS}	Newton-iterations	$\bar{\rho}_{\text{av}}$	WCT _{norm}
1	9	0.097	1
2	6	0.030	1.94
3	5	0.015	2.92
4	5	0.015	3.38
5	4	0.006	3.72
6	4	0.006	3.89
7	4	0.006	3.95
8	4	0.006	4.09
9	4	0.006	4.09
10	4	0.006	4.12

Table 5.14: The number of Newton iterations as well as the resulting smoothing rate and the normed wallclocktime in Example 5.6.2 (Cartesian grid) is given in dependence of the number of CS-iterations λ_{CS} . The iteration is stopped as soon as the residual norm drops below the error tolerance $\epsilon_{\text{fine}} = 10^{-6}$.

The test reported in Table 5.14 and Table 5.15 examines the influence of the number of CS-iterations λ_{CS} employed to solve the linearized system. The struggle between the quadratic convergence of the Newton method and the linear convergence of the multigrid iteration mentioned by [13] is clearly visible. For $\lambda_{\text{CS}} = 1$ the average smoothing rates $\bar{\rho}_{\text{av}}$ are similar to the smoothing rates reported for the linear correction scheme (compare the results in Table 5.5 and 5.6, respectively). As λ_{CS} grows, the quadratic convergence behavior of the Newton method takes over, resulting in ever lower values for $\bar{\rho}_{\text{av}}$. Figure 5.5 depicts the average smoothing rate in dependence of the number of CS-sweeps per multigrid iteration.

The results indicate that although the convergence is accelerated if more CS-iterations are applied to the linear system, there exists an optimal value for λ_{CS} in the sense that this choice minimizes the mean smoothing rate. This optimal value is problem-dependent and amounts to $\lambda_{\text{CS}} = 5$ in Example 5.6.2 and to $\lambda_{\text{CS}} = 8$ in Example 5.6.3. Employing a higher λ_{CS} does not further accelerate the convergence but enhances the adverse effect of increasing the computational time. Having a look at the wallclocktimes reported in Table 5.14 and 5.15, an opposite trend is apparent. Whereas the mean smoothing rate decreases if more CS-iterations per Newton cycle are employed, the computational time the solver takes to compute an acceptable solution increases. This effect is especially pronounced if the Cartesian grid setting is considered, indicating that the Gauss-Seidel based coarse grid solver severely impacts the runtime. In this case, employing $\lambda_{\text{CS}} = 10$ leads to an increase of computational time of a factor of 4.12 compared to the setting with $\lambda_{\text{CS}} = 1$. Interestingly, on the spherical, stretched grid, the performance of the Gauss-Seidel procedure seems to be more efficient in procuring a coarse-grid solution. The increase in runtime using $\lambda_{\text{CS}} = 10$ only amounts to a factor of 1.59. A possible explanation is that the linearized system in the spherical, stretched geometry is much better conditioned than its Cartesian counterpart.

Figure 5.6 shows the evolution of the residual norm and the maximal value of the correction v_1^l

λ_{CS}	Newton-iterations	$\bar{\rho}_{\text{av}}$	WCT _{norm}
1	39	0.601	1
2	18	0.34	1.04
3	12	0.190	1.25
4	9	0.109	1.29
5	9	0.104	1.46
6	8	0.08	1.52
7	7	0.056	1.54
8	5	0.019	1.48
9	5	0.015	1.54
10	5	0.014	1.59

Table 5.15: The number of Newton iterations as well as the resulting smoothing rate and the normed wallclocktime in Example 5.6.3 (stretched, spherical grid) is given in dependence of the number of CS-iterations λ_{CS} . The iteration is stopped as soon as the residual norm drops below the error tolerance $\epsilon_{\text{fine}} = 10^{-6}$.

appearing in equation (5.6.19) of the MG-Newton algorithm for $\lambda_{\text{CS}} = 1$, $\lambda_{\text{CS}} = 5$ and $\lambda_{\text{CS}} = 10$. As the residual norm drops, also the corrections v_1^l become smaller. Thus, in case that it is not always possible or advisable to prescribe an absolute error tolerance, the maximum correction value could be used to terminate the MG-Newton iteration if it drops below a certain value. Although this kind of termination criterion requires some knowledge of the magnitude of the expected solution, it is more intuitive to use.

5.7 Parallelization issues

5.7.1 The necessity of parallel computing

For decades, new developments in computer science have been guided by Moore's Law which states that the number of transistors on a computer chip doubles every 24 months. The increase in the number of transistors has led to an equal increase in computer performance.

In recent years, however, it has become clear that, although Moore's Law is still valid, standard microprocessors are starting to hit the "heat barrier": Switching and leakage power of several-hundred-million transistor chips are so large that cooling becomes a primary engineering effort. On the other hand, the necessity of an ever increasing clock-frequency is driven by the insight that architectural advances and growing cache sizes alone will not be sufficient to keep up the one-to-one correspondence of Moore's law with application performance. A way out of this dilemma is the introduction of multicore processors and the assembly of many processors into supercomputers.

This completely new hardware design requires software to be programmed such as to take advantage of more than one processor working on the application. Previously, code has been processed *sequentially*, i.e., one instruction has been executed at a time. Multicore and multiprocessor

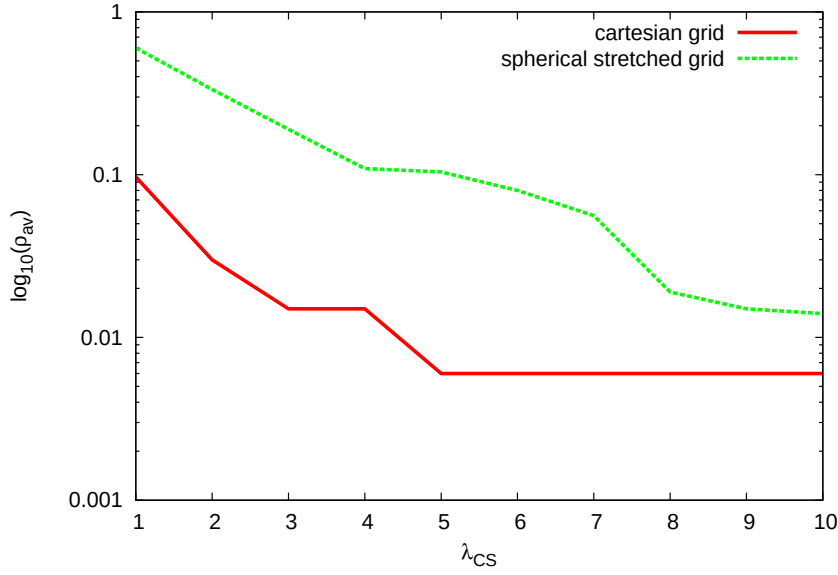


Figure 5.5: The influence of the number of CS-sweeps λ_{CS} per Newton iteration on the average smoothing rate $\bar{\rho}_{av}$.

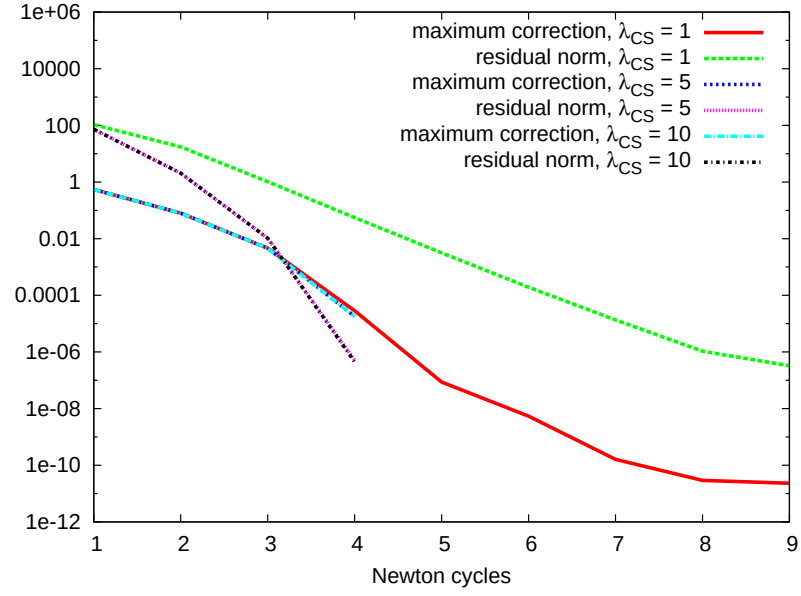
architectures now offer the possibility of executing several instructions at the same time – i.e. in *parallel*. To fully exploit the performance of a multicore processor, the application must be programmed such that several independent tasks can be executed simultaneously. This technique is called *parallelization*.

There are two fundamental variants of parallel computers: the shared-memory and the distributed-memory type. Both utilize networks for communication between processors, but in shared-memory architectures there are certain groups of CPUs sharing one memory space (thus eliminating the need to transfer data between the corresponding processes) whereas in the distributed-memory model it is assumed that each processor has its own private memory. Since most super-computer architectures are of the latter type, this thesis will focus on the distributed-memory approach.

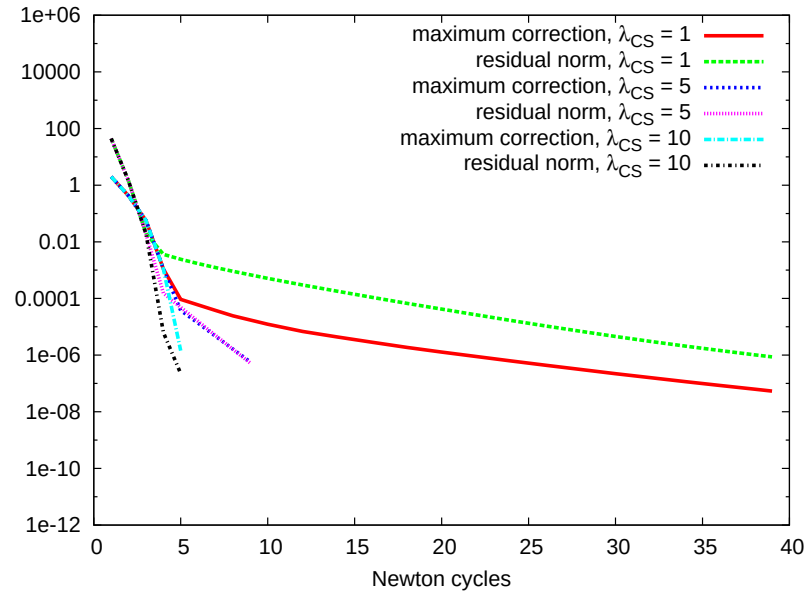
The Message Passing Interface (MPI)

Message passing is required if a parallel computer is of the distributed-memory type, i.e. if there is no way for one processor to directly access the memory of another. The use of explicit *message passing*, that is communication between processes, is surely the most tedious and complicated, but also the most flexible approach for parallelization. The *Message Passing Interface (MPI)* standard is the result of a joint standardization effort in the 1990s, with the aim to enable scientific users to write parallel programs that are easily portable. This goal has clearly been achieved: today, the MPI standard is supported by several free and commercial implementations and is considered as a necessary ingredient in any high performance computing system installation.

A thorough introduction to parallel computer systems and computing paradigms can be found



(a) Example 5.6.2 (Cartesian grid)



(b) Example 5.6.3 (stretched spherical grid)

Figure 5.6: The evolution of residual norm and the maximum value of the correction v_1^l after each Newton-iteration is plotted. Note that on the ordinate a logarithmic scale is used.

in [14].

5.7.2 Parallel Multigrid

A survey of parallel multigrid algorithms is found in [23]. In the following only a brief introduction on the strategies employed is given.

In the context of parallel multigrid algorithms, two main strategies have been pursued: The first approach involves no changes to the original multigrid algorithm. The calculations on the different grids are processed sequentially and parallelism is exploited by partitioning the grids to different processors. The parallel processing also influences the choice of the relaxation procedure (typically, in this context the Jacobi iteration or multicolor Gauss-Seidel methods are used) as well as the choice of the direct solver on the coarsest grid. In [23], this approach is referred to as *parallel computation within levels*.

The second approach is radically different and motivated by the desire to overcome the parallel limitation inherent in the sequential processing of different grids. Especially if the number of unknowns is of the same order of magnitude or less than the number of processors employed, the inherent sequentiality of the multigrid algorithm can lead to a large idle time of the processors. To resolve this problem, several algorithm modifications have been suggested. The resulting algorithms are referred to as *concurrent multigrid algorithms*.

Since in the applications within the focus of the ANTARES code (and therefore, of this thesis) the number of grid points will always be much larger than the number of processors employed, the first approach – parallel computation within levels – is employed. As outlined in Chapter 3, the ANTARES framework provides a domain decomposition strategy which can be used readily within the multigrid algorithm. When establishing a coarser grid as a subset of the original grid points, the adherence of the points to the processors is retained. Naturally, the number of grid levels must be such that even the coarsest grid still has enough grid points to keep the communication overhead low.

The assembly of the stiffness matrix as well as the intergrid operators are easily parallelized. Communication is kept low by using a certain amount of ghost cells. As a smoother, both the ZGS- and the GS-RB schemes have a high degree of parallelism such that only very little communication is necessary to perform a full relaxation sweep. The same is true for using a few CG-iterations as a smoother. Efficient parallelization of the direct solver on the coarsest grid has proven tricky, though.

Efficient Parallelization of the direct solver on the coarsest grid

It has been outlined in section 5.5.5 that the solver developed by Hannes Grimm-Strele [11] is used to compute an approximate solution on the coarsest grid. Although the CG-based solver is efficient and fast if employed on one processor, in every iteration entails at least two communication calls due to the necessity to evaluate the residual norm and the correct transfer of values for the periodic boundary conditions in horizontal direction. Whereas for a few iterations (such

as performed if the CG-algorithm is used as a smoother) this is negligible, the parallel efficiency of the direct solver (and thus of the overall multigrid iteration) is severely impacted if regularly a few hundred iterations are necessary to obtain a solution to the coarse grid system.

The original solver of [11] addresses this difficulty by employing the *Schur Complement algorithm*. In the following, the basic principles of the Schur-Complement algorithm are revised. For a detailed introduction to the Schur-Complement technique, the reader is referred to [11] or [44].

The Schur Complement Algorithm

The idea of the Schur Complement algorithm is to transform the given system of linear equations $Au = f$ into systems of smaller dimensions which can be solved independently of each other. To this end, A is rearranged to be of appearance

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \quad (5.7.1)$$

with

$$A_{1,1} = \begin{pmatrix} D_1 & & 0 \\ & D_2 & \\ & & \ddots \\ 0 & & & D_p \end{pmatrix}, \quad (5.7.2)$$

$A_{1,2} = (B_1, B_2, \dots, B_p)^T$ and $A_{2,1} = (C_1, C_2, \dots, C_p)$, p denoting the number of processors employed. It is assumed that $A_{1,1}$ is invertible.

The linear system $Au = f$ is now equivalent to

$$\begin{pmatrix} I & 0 \\ A_{2,1}A_{1,1}^{-1} & I \end{pmatrix} \underbrace{\begin{pmatrix} A_{1,1} & A_{1,2} \\ 0 & S_A \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}}_{=:(v_1, v_2)^T} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}, \quad (5.7.3)$$

$S_A = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$ denotes the *Schur Complement of the matrix A* .

Assuming that S_A is invertible and using the temporary variable $v = (v_1, v_2)^T$ as defined in equation (5.7.3), the Schur Complement algorithm entails the following steps:

- (i) Assembly of the right hand side v_2 of the Schur Complement equation,

$$v_2 = f_2 - A_{2,1}A_{1,1}^{-1}v_1 \quad (5.7.4)$$

$$v_1 = f_1 \quad (5.7.5)$$

- (ii) Solution of the Schur Complement equation

$$S_A u_2 = v_2 \quad (5.7.6)$$

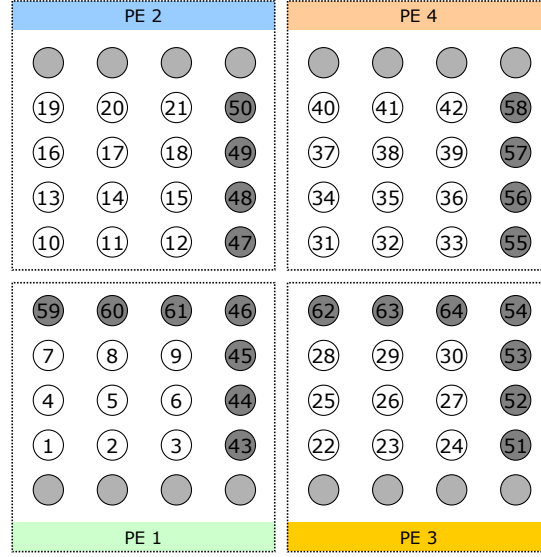


Figure 5.7: Schematic depiction of the reordering strategy if 4 processors are employed. The light grey nodes denote the physical boundary nodes, the dark grey ones are the chosen interface nodes. This figure is taken from [11].

(iii) Solving equation

$$A_{1,1}u_1 = v_1 - A_{1,2}u_2 \quad (5.7.7)$$

to obtain $u = (u_1, u_2)^T$.

Note that due to the block structure of $A_{1,1}$ and the information available from the solution of the Schur complement equations, equation (5.7.7) reduces to p problems which can be solved independently on each processor. To this end, the (serial) CG-algorithm is employed. To accelerate convergence, the system matrices are preconditioned by the Incomplete Cholesky Decomposition. The computationally expensive part of the Schur Complement algorithm lies in the assembly and solution of the Schur Complement system. To transform A into the desired structure (5.7.1), the nodes are suitable renumbered as depicted in Figure 5.7. This entails the identification of the *physical boundaries* as well as the *virtual* or *domain boundaries* arising due to the domain decomposition approach. The nodes on one side of the domain boundary are selected as interface nodes. Then, the nodes are renumbered, starting with the inner nodes (on all domains) and numbering the interface points last. The resulting matrix possesses the desired shape.

Remark 5.7.1. Note that this renumbering approach also allows a slightly different, albeit more intuitive interpretation: Due to the domain decomposition approach, the matrix A is split into different “portions”. All of these “portions” may be treated as if they were independent linear systems of equations. However, (virtual) boundary conditions have to be supplied. Thus, all the nodes being on domain boundaries (the nodes formerly referred to as “interface nodes”) are gathered into a single system of equations, the Schur-Complement system. After solving the

Schur-Complement equation, the necessary virtual boundary conditions are supplied to each domain and the corresponding linear systems can be solved.

The Schur-Complement equation is solved by the parallelized CG-algorithm. Here, all considerations made above concerning the communication overhead of a directly parallelized implementation of the CG-technique apply. However, the advantage of the Schur-Complement algorithm is that the system to be solved in parallel is greatly reduced in size compared to the original one.

It has been shown in [16] that the Schur Complement technique involves a certain overhead due to the assembly and solution of the Schur-Complement equation. However, it displays almost optimal strong scaling behavior and is particularly fast if many processors are employed. Thus, the technique has been adapted to solve the coarse grid system arising within the linear multigrid cycle.

Multigrid Scaling Tests

To assess the most efficient strategy to parallelize the direct solver on the coarsest grid, scaling tests have been conducted on the *Vienna Scientific Cluster 1 (VSC 1)*. In this case, the capabilities of the solver were not evaluated using a test example but on the *pressure-correction equation* appearing in the simulation of double-diffusive convection. Although the pressure-correction equation does not stem from the use of IMEX-Runge Kutta methods but arises in the context of a fractional step method, it will be seen in Chapter 6 that this is the most tedious equation to solve in the present context and therefore it is an equation for which optimal parallel efficiency is desired. The test has been conducted so as to perform one timestep of a double-diffusive simulation and to record the wallclocktimes spent in each routine by the use of the parallel profiling software TAU (available at <http://www.cs.uoregon.edu/research/tau/home.php>).

Table 5.17 reports the results of the multigrid scaling test where the equation considered is solved employing different numbers of processors and different numbers of grid points. The original Schur-Complement parallelized CG (SCpCG)-solver of [11] is tested against the multigrid solver employing the SCpCG-solver as coarse grid solver (MG-SCpCG), the multigrid solver employing the SCpCG algorithm on the coarsest grid but without preconditioning the system matrices in equation (5.7.7) (MG-SCCG) and the multigrid solver using the directly parallelized CG-algorithm on the coarsest grid (MG-parCG).

The results confirm the excellent strong scaling behavior of the SCpCG-solver, which has also been reported in [16]. In all of the test runs, a doubling of the number of processors leads to halving the computational time the SCpCG-solver takes to calculate a solution of the given problem. It has to be pointed out, though, that the setup of the Schur Complement algorithm incurs an overhead of a factor of 12 (see [16]).

The multigrid method features weak scaling at best, and only for sufficiently large systems of equations. However, this is expected due to its inherent sequentiality. The difference in scaling

5. MULTIGRID METHODS

grid points	SCpCG	MG-parCG
829×828	1 (0.198 s)	1 (0.106 s)
1629×1628	3.0 (0.594 s)	1.71 (0.181 s)
2429×2428	7.5 (1.475 s)	2.27 (0.241 s)
3229×3228	14.28 (2.827 s)	2.75 (0.291 s)

Table 5.16: Comparison of the computational efficiency if the solver is employed on 64 CPU-cores and the number of grid points is varied. The normed wallclocktime is given, the effective wallclocktime is given in brackets.

behavior is most impressive having a look at the results on 3229×3228 grid points: whereas the computational time for the SCpCG algorithm is reduced by a factor of 4 when employing four times as many processors, the MG-parCG solver seems almost indifferent to the number of processors used.

It is interesting to note that the efficiency of the multigrid algorithm increases as the number of grid points increases. Table 5.16 compares the wallclocktimes of the SCpCG solver and the MG-parCG solver when 64 CPU-cores and a varying number of grid points are used. Whereas the (approximate) quadrupling of the number of grid points results leads to a wallclocktime 14.28 times higher for the SCpCG algorithm, the multigrid algorithm takes only 2.75 times longer to solve the corresponding problem.

Comparing the different direct solvers employed in the multigrid algorithm, it is obvious that the MG-SCpCG solver incurs the most overhead. Leaving the preconditioning step out as it is done within the MG-SCCG solver, the computational time is significantly reduced compared to the time it takes the MG-SCpCG solver to provide a valid solution. This indicates that more time is spent setting up the preconditioning matrix than is saved by accelerating convergence in that way.

In terms of computational efficiency, the MG-parCG-solver is the most efficient one of the different multigrid solvers. This is surprising, since the parallelized CG-Algorithm is not known for its efficiency. However, this might be due to the fact that on the coarsest grid level, the linear system to be solved is already small and the setup costs of the Schur-Complement matrix are much more expensive than the non-efficient parallelization of the CG-algorithm. Furthermore, the MG-parCG solver always outperforms the SCpCG-solver despite its moderate scaling behavior. Concerning this effect, the authors of [23] remark

The most efficient algorithm is presumably the one that solves the problem in the shortest time. This means that questions about processor utilization, communication to computation ratios and other issues are secondary. While these questions are often very useful in analyzing the performance of an algorithm, their importance can be overstated. For example, an algorithm with little or no communication requirements and whose parallel performance is perfectly scalable may be, for a particular application and computer, much less efficient than an algorithm that does not parallelize well.

In this spirit, the MG-parCG solver is the method of choice throughout the remaining chapters

829 × 828 grid points				
CPU-cores	SCpCG	MG-SCpCG	MG-SCCG	MG-parCG
4	1 (2.584 s)	0.21 (0.546 s)	0.13 (0.342 s)	0.06 (0.155 s)
16	0.25 (0.657 s)	0.10 (0.264 s)	0.07 (0.183 s)	0.04 (0.105 s)
64	0.08 (0.198 s)	0.08 (0.206 s)	0.06 (0.148 s)	0.04 (0.106 s)
1629 × 1628 grid points				
CPU-cores	SCpCG	MG-SCpCG	MG-SCCG	MG-parCG
4	1 (9.543 s)	0.16 (1.519 s)	0.10 (0.925 s)	0.08 (0.774 s)
16	0.26 (2.416 s)	0.06 (0.545 s)	0.04 (0.349 s)	0.04 (0.336 s)
64	0.06 (0.594 s)	0.03 (0.319 s)	0.03 (0.239 s)	0.02 (0.181 s)
2429 × 2428 grid points				
CPU-cores	SCpCG	MG-SCpCG	MG-SCCG	MG-parCG
16	1 (5.738 s)	0.14 (0.775 s)	0.10 (0.574 s)	0.10 (0.553 s)
64	0.26 (1.475 s)	0.06 (0.330 s)	0.04 (0.226 s)	0.04 (0.241 s)
3229 × 3228 grid points				
CPU-cores	SCpCG	MG-SCpCG	MG-SCCG	MG-parCG
64	1 (2.827 s)	0.18 (0.497 s)	0.11 (0.319 s)	0.10 (0.291 s)
256	0.28 (0.791 s)	0.14 (0.408 s)	1.0 (2.828 s)	0.10 (0.294 s)

Table 5.17: Results of the Multigrid scaling test. The normed wallclocktime is given. Note that for every testsuite, a different reference value WTC_{ref} is assumed. To enable a comparison also between the distinct testsuites and thus to evaluate the effect of using more grid points on the same number of CPU-cores, the measured computational times are given in brackets.

of this thesis.

Chapter 6

Application to Double-Diffusive Convection

6.1 The Model

6.1.1 The semiconvection problem

Double-diffusive convection is a phenomenon encountered in chemically inhomogeneous flows where a significant molecular weight gradient is present.

Semiconvection is the most important special case of double-diffusive convection in astrophysics. [55] summarizes underlying mechanics of semiconvection and the consequences in the field of astrophysics in the following way:

Models of stellar structure and evolution predict settings where the heavier product of nuclear fusion provides stability to a zone which otherwise would be unstable to convective overturning since temperature decreases sufficiently rapidly against the direction of gravity. Such a zone would become convective if its composition were mixed and the question whether such a zone should be treated as if the material were mixed or not has become known as the semiconvection problem. The presence of a semiconvective zone, however, has only a minor effect on the thermal structure of the star, but the assumed amount of mixing of composition is critical because the evolution of the star is sensitive to the precise distribution of products of nuclear burning with depth in the star.

A detailed discussion on the physics of double-diffusive convection and its significance in astrophysics can be found in [21, 55, 54] and references therein.

6.1.2 Extension of the Navier-Stokes equations to two-component flows

The fluid considered in simulations of double-diffusive convection is a binary mixture. Therefore, an additional equation expressing partial mass conservation enters the model,

$$\frac{\partial(\rho c)}{\partial t} = -\nabla \cdot (\rho c \vec{u}_{\text{rel}}) + \nabla \cdot (\rho \kappa_c \nabla c) \quad (6.1.1)$$

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

where c denotes the mass fraction of the second species considered and κ_c is the corresponding diffusion coefficient. A detailed derivation of (6.1.1) is found in [56].

The set of equations describing the flow of a two-component fluid reads

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \vec{u}_{\text{rel}}) \quad (6.1.2)$$

$$\frac{\partial(\rho c)}{\partial t} = -\nabla \cdot (\rho c \vec{u}_{\text{rel}}) + \nabla \cdot (\rho \kappa_c \nabla c) \quad (6.1.3)$$

$$\frac{\partial(\rho \vec{u})}{\partial t} = -\nabla \cdot (\rho \vec{u}_{\text{rel}} \otimes \vec{u}) - \nabla P + \nabla \cdot \tau + \rho g \quad (6.1.4)$$

$$\frac{\partial(\rho e_{\text{tot}})}{\partial t} = -\nabla \cdot (\vec{u}_{\text{rel}}(\rho e_{\text{tot}}) + \vec{u}P) + \nabla \cdot (\vec{u} \cdot \tau) + \rho \vec{u}g + \nabla \cdot F_{\text{rad}} \quad (6.1.5)$$

Since the simulations of double-diffusive convection are carried out on a Cartesian grid fixed in space, $\vec{u}_g = 0$ and thus $\vec{u}_{\text{rel}} = \vec{u}$.

6.1.3 Model Assumptions

Semiconvection in a (main sequence) star in an advanced stage of evolution typically takes place in an interior region close to the core where helium is produced via nuclear reactions. Since this region is optically thick, it is justified to use the diffusion approximation to approximate the stationary limit of the radiative transfer equation,

$$Q_{\text{rad}} = \nabla \cdot F_{\text{rad}} = -\nabla \cdot (K_{\text{rad}} \nabla T) \quad (6.1.6)$$

Furthermore, the radiative conductivity K_{rad} , the diffusion coefficient of the second species κ_c as well as the kinematic viscosity ν (appearing in the definition of the deviatoric stress tensor τ) are assumed to be constant,

$$K_{\text{rad}} \equiv \text{const}, \quad (6.1.7)$$

$$\kappa_c \equiv \text{const}, \quad (6.1.8)$$

$$\nu \equiv \text{const}, \quad (6.1.9)$$

and the fluid mixture is treated as an ideal gas. Therefore, instead of using the OPAL EOS tabulated values, the equations (6.1.2-6.1.5) are closed by the equation of state for ideal gas,

$$P = \frac{R_{\text{gas}} \rho T}{m}, \quad (6.1.10)$$

and the inner energy is related to the pressure via

$$e_{\text{int}} = \frac{3}{2}P, \quad (6.1.11)$$

where m denotes the mean molecular weight of the compound. It is computed from the mass fraction c by $m = \frac{1}{c/\text{atm}_2 + (1-c)/\text{atm}_1}$. The variables atm_1 and atm_2 refer to the atomic weight of

the constituents of the fluid. In the present case, a mixture of hydrogen and helium is modeled (c denotes therefore the mass fraction of helium). Thus, $\text{atm}_1 = 1$ is defined as the atomic weight of hydrogen whereas $\text{atm}_2 = 4$ is the atomic weight of helium (in a non-ionized state).

6.1.4 Initial Condition and Boundary Conditions

The starting model

To set up a model of stellar convection, a hydrostatically layered fluid column which is unstable against convection as in [38] is considered. In terms of the Schwarzschild criterion of convective instability (see [52]) with the x -direction pointing along the direction of gravity, this reads

$$\frac{\partial T}{\partial x} > \left(\frac{\partial T}{\partial x} \right)_{\text{ad}}. \quad (6.1.12)$$

From the outset the gas is assumed to be an ideal gas with a γ -law where $\gamma = \frac{5}{3}$. The volume expansion coefficient is given by $\alpha = \frac{1}{\gamma-1}$ and the specific heat at constant pressure is $c_P = R_{\text{gas}}(1 + \alpha)$.

These data suffice to determine the adiabatic temperature gradient

$$\left(\frac{\partial T}{\partial x} \right)_{\text{ad}} = \frac{g}{c_P} \quad (6.1.13)$$

once the gravitational acceleration is g is specified.

The temperature is fixed at the top and a function $b(x)$ is defined to be the ratio of the actual temperature gradient to the adiabatic one:

$$\frac{\partial T}{\partial x} = b(x) \left(\frac{\partial T}{\partial x} \right)_{\text{ad}}. \quad (6.1.14)$$

Specifically, the function $b(x) = a$ is specified with $a > 1$ and equation (6.1.14) can be integrated straightforwardly. In this way the temperature – and therefore the energy – is specified for the entire domain.

Subsequently, the density is fixed at the top. Thus, the pressure P can be eliminated from the equation of hydrostatic equilibrium. Simple integration yields the density for the entire domain. The pressure P is obtained from the equation of state (6.1.10).

To complete the model, the kinematic viscosity ν and the thermal diffusion coefficient κ have to be set. This is achieved by prescribing the Rayleigh number Ra and the Prandtl number $Pr = c_P \nu \rho / K_{\text{rad}}$.

Up to this point, the setup is suitable to simulate convection of a homogeneous fluid. The setup for double-diffusive convection has been set on top of the setup for stellar convection by augmenting it with the specification of the Lewis Number $Le = c_P \rho \kappa_c / K_{\text{rad}}$, which relates the radiative conductivity K_{rad} to the diffusion coefficient κ_c , as well as with the stability parameter R_ρ . The stability parameter is defined as the ratio of the mean molecular weight gradient to the

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

superadiabatic gradient. In the astrophysical case, $Le < Pr < 1$ while $Ra^* = Ra \cdot Pr \gg 1$ and R_ρ is frequently of order unity. The nature of the flow sensitively depends on the value of R_ρ (see [55]).

To start dynamics away from equilibrium a random initial perturbation is applied.

Further details on the starting model are found in [56], a thorough study of the dependencies of the flow on the different dimensionless parameters Le , Pr , R_ρ , and Ra^* is found in [55].

Boundary conditions

Apart from the initial condition, suitable boundary conditions have to be specified which are invoked during each step. In this model, *slip* boundary conditions are specified for the velocity $\vec{u} = (u, v, w)^T$,

$$u(x_0, \cdot, \cdot) = 0, \quad u(x_N, \cdot, \cdot) = 0, \quad (6.1.15)$$

$$\frac{\partial v}{\partial x}|_{(x_0, \cdot, \cdot)} = 0, \quad \frac{\partial v}{\partial x}|_{(x_N, \cdot, \cdot)} = 0, \quad (6.1.16)$$

where x_0 denote the first and x_N the last grid point in vertical direction.

Temperature, density and the helium mass fraction are set to their initial values, i.e.

$$\rho(x_0, \cdot, \cdot) = \rho(x_0, \cdot, \cdot)^{\text{start}}, \quad \rho(x_N, \cdot, \cdot) = \rho(x_N, \cdot, \cdot)^{\text{start}}, \quad (6.1.17)$$

$$c(x_0, \cdot, \cdot) = c(x_0, \cdot, \cdot)^{\text{start}}, \quad c(x_N, \cdot, \cdot) = c(x_N, \cdot, \cdot)^{\text{start}}, \quad (6.1.18)$$

$$T(x_0, \cdot, \cdot) = T(x_0, \cdot, \cdot)^{\text{start}}, \quad T(x_N, \cdot, \cdot) = T(x_N, \cdot, \cdot)^{\text{start}}. \quad (6.1.19)$$

These boundary conditions act propulsive on the convection. Due to the unstable temperature stratification at the beginning, the temperature is higher at the bottom than at the top. Therefore, the fixed bottom boundary condition is a constant heat source whereas the top boundary condition has a cooling function.

In horizontal direction, periodic boundary conditions are applied.

6.2 Numerical simulation of double-diffusive convection

Numerical simulations in the double-diffusive regime as it occurs in stars have proven to be tricky. First of all, using realistic values for the Prandtl-, Lewis- and Rayleigh numbers would require a grid resolution far beyond the limit of feasibility. Furthermore, the double-diffusive convection takes place in regimes where the (maximum) Mach number $Ma = \vec{u}/c_{\text{snd}}$ is small (typically $Ma < 0.3$). In such regimes, acoustic waves propagate much faster than the fluid moves. Although it is often the case that these sound waves are energetically negligible, an explicit time integration method requires the temporal resolution to be such as to resolve even the fastest motions, in this case the propagation of sound waves. Thus, in order to obtain reliable and stable results, very small timesteps must be employed which slow down the computation.

Furthermore, explicit methods are known to render highly inaccurate results as the Mach number approaches 0 (see [12, 16, 25]).

In [56] and subsequently in [55], numerical simulations of double-diffusive convection have been performed and mixing rates as well as dependencies of the mixing rate on the intrinsic parameters have been determined computationally. To avoid the difficulties arising with the explicit time integration of the two-component NSE, the Boussinesq approximation has been employed. In the Boussinesq approximation, the fluid is assumed to be almost incompressible (density fluctuations are only accounted for in the buoyancy term). Furthermore, it is supposed that temperature and pressure fluctuations are small with respect to a given background stratification. Altering the NSE due to these assumptions, the problem of fast propagating sound waves is eliminated by solving a Poisson equation for the pressure P instead of obtaining it from the equation of state and the timestep can be chosen much larger. A limitation of this approximation is the assumption of only small pressure and temperature fluctuations, meaning that only a very small part of the star ($< 1 H_P$) can be simulated. The Boussinesq approximation breaks down at about $Ma > 0.3$. In [16], a fractional-step approach is used where the original (compressible) equations are not modified as in the Boussinesq approximation. In this approach, the stringent CFL-condition due to acoustic wave propagation is alleviated. Although the computational efficiency is not as high as when using the Boussinesq approximation, the method is not restricted to a certain Mach number range, allowing therefore a wider field of application. Furthermore, this approach provides the basis for the use of IMEX Runge-Kutta methods in simulations of double-diffusive convection.

6.2.1 Simulation of double-diffusive convection

In the following, the simulations of double-diffusive convection are based on the compressible set of Navier-Stokes equations. The fractional step method of [16] is employed.

Simulation Setting

For the simulation of double-diffusive convection, an equidistantly spaced grid and Cartesian coordinates (box-in-a-star approach) is used.

Due to the compressible nature of the flows modeled, the vertical extent of the simulation domain is specified in multiples of the pressure scale height $H_P = P/(\rho g)$. For the simulations presented the domain always covers $1 H_P$.

To determine the resolution required to resolve even the smallest structures in the simulations – namely, to perform a *direct numerical simulation* – mixing length theory is invoked. In [56], a correlation between the Rayleigh number and the diffusion coefficients is derived and the balance between advection by the interior flow and the diffusion across an interface is used to estimate

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

the boundary layer thickness δ_T as

$$\delta_T = \frac{H}{\sqrt[4]{Ra^*}} \quad (6.2.1)$$

where H denotes the (physical) height of the domain and $Ra^* = Ra \cdot Pr$ is the modified Rayleigh number. If H is translated into the number of grid points in vertical direction, δ_T gives the number of grid points resolving the thermal boundary layer.

Similarly, in case of double-diffusive convection, the boundary layer thickness of the second species δ_c follows from the definition of the Lewis number and scales with the thermal boundary layer thickness,

$$\delta_c = \sqrt{Le} \delta_T \quad (6.2.2)$$

Using the definition of the Prandtl number Pr , the same conclusion can be drawn for the viscous boundary layer,

$$\delta_{\text{visc}} = \sqrt{Pr} \delta_T. \quad (6.2.3)$$

In the present context it is useful to resolve the boundary layers with a minimum of 5 grid points which corresponds to the width of the stencil employed for spatial discretization.

The system of equations is discretized on a rectangular, equidistantly spaced grid. Simulation time is measured in units of *sound crossing times* (*scrt*). One sound crossing time is defined as the time it takes an acoustic wave to propagate from the bottom to the top of the simulation box.

General Outline

Figure 6.1 shows the typical stages of a simulation of double-diffusive convection. The setup of a simulation of double-diffusive convection is a hydrostatically stratified (pressure increases from top to bottom) but thermally unstable (temperature increases sufficiently fast from top to bottom) configuration. Therefore, hot fluid parcels at the bottom of the box tend to rise, which leads to the build-up of a velocity field. The first visible effects of this trend is the formation of gravity waves. The amplitude of those gravity waves grows until they break, introducing local turbulent mixing. The mixing quickly extends over the entire box, unsystematically at first, but due to the constant heating at the bottom and cooling at the top, convective rolls form. Once the convective rolls are formed, the statistically steady-state solution of the system is reached (see [55, 56]).

In Figure 6.2, the evolution of the timestep according to different criteria in the course of a simulation is depicted. The most stringent timestep criterion is the restriction due to the sound waves. However, it is closely followed by the restrictions due to diffusion processes. The timestep restriction posed by the fluid motion grows more severe due to the constant increase of the velocity and eventually becomes even more restrictive than the restriction due to diffusion processes. The timestep Δt actually used is defined as the minimum over all timestep restrictions. In the

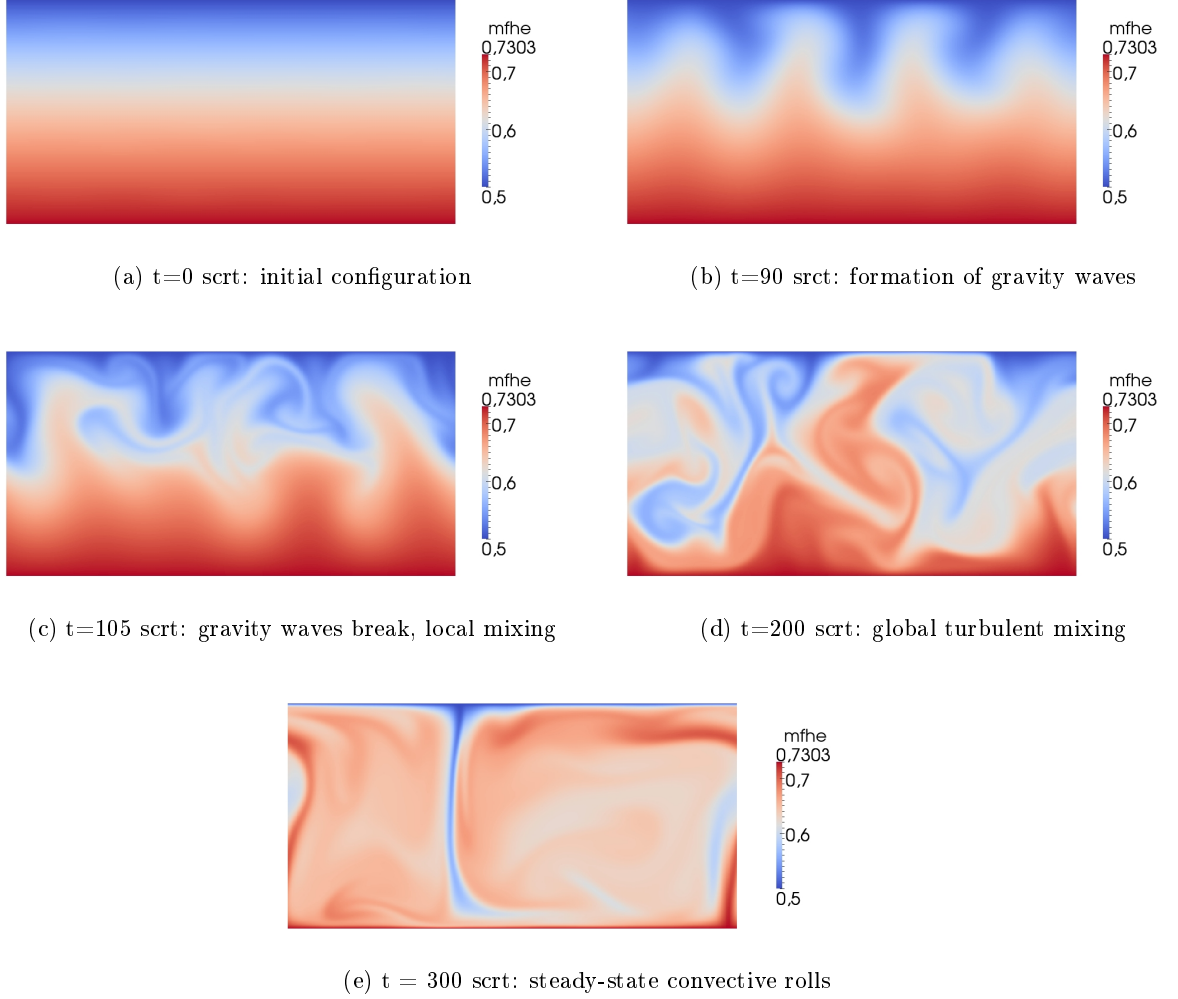


Figure 6.1: Typical stages of a simulation of double-diffusive convection. The helium massfraction c (in the pictures, it is denominated **mfhe**) is depicted.

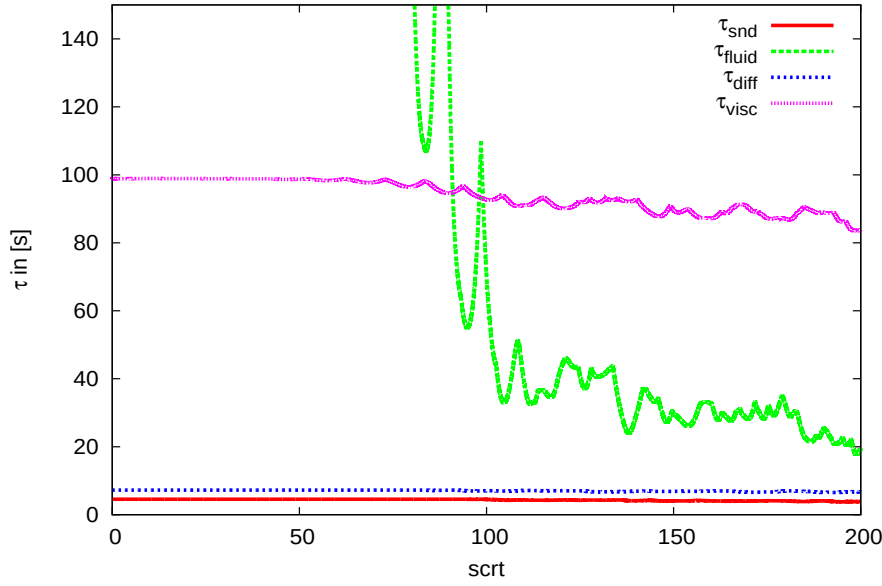


Figure 6.2: Timestep evolution in a typical simulation of double-diffusive convection. The restriction due to soundwaves, τ_{snd} proves the most restrictive criterion whereas τ_{fluid} becomes more restrictive as the fluid velocity grows.

following, the part of the simulation where $\tau_{\text{fluid}} > \tau_{\text{diff}}$ will be referred to as the *diffusive phase* whereas the second part where $\tau_{\text{diff}} > \tau_{\text{fluid}}$ is denominated the *advective phase* of the simulation. Since the general ansatz of this thesis is to accelerate simulations in the astrophysical regime not by solving the entire system of governing equations using an implicit time integration scheme but to follow an operator-splitting approach and to treat only terms corresponding to stiff processes implicitly, it is quite clear that real acceleration can only be achieved in the diffusive phase of the simulation. In the advective part, the simulation is governed by fluid motions which represent the main transport of energy. The timestep restriction posed by advection must therefore be adhered to. However, within the simulations of double-diffusive convection the point of interest is the evolution of the mixing process and the corresponding timescales, so the diffusive part in general composes a significant part of the simulation. Accelerating the computation in this regime leads to a considerable decrease of the overall computational time.

6.2.2 A fractional step approach

In [16], the original (compressible) set of equations (6.1.2)-(6.1.5) is integrated in time via a fractional step method. In this approach, all variables apart from the pressure terms in the momentum and energy equation are evaluated explicitly, resulting in density and partial density values at the new timestep and intermediate values for momentum and energy. The pressure is predicted at the new timestep via a *pressure correction equation* which takes the form of a linear generalized Poisson equation. The predicted pressure is used to update the momentum and energy values to the new timestep. To avoid decoupling of the pressure and the other variables, the predicted pressure is discarded after the momentum and energy update and the pressure at

the new timestep is evaluated using the equation of state. For a detailed outline of the algorithm, a number of tests examining the ability of the fractional step method to render accurate results in low and high Mach number regimes as well as convergence and scaling properties, the reader is referred to [16].

Note that the fractional step method alleviates the most stringent timestep restriction within simulations of double-diffusive convection, namely, the timestep restriction posed by sound waves. Thus, it provides the basis for the use of SSP IMEX Runge-Kutta methods in this context. In the remainder of this chapter it is therefore understood that “explicit evaluation of the NSE” means the use of the fractional step method although it entails the “implicit” pressure correction equation.

6.2.3 Timestep restrictions in simulations of double-diffusive convection

The fractional step method is designed to alleviate the stringent timestep restriction posed by acoustic wave propagation. Practically, this assumption results in the complete removal of the timestep criterion imposed by sound waves. The remaining time step criteria are due to diffusion processes, the fluid velocity and the viscosity. Note that additionally the timescale set by the diffusion of the second species must be taken into account. Therefore, the timestep Δt is defined as

$$\Delta t = \min(\tau_{\text{diff}}, \tau_{\text{visc}}, \tau_{\text{fluid}}) \quad (6.2.4)$$

where

$$\begin{aligned} \tau_c &= \frac{\mu_{\text{diff}}}{\kappa_c} \min((\Delta x)^2, (\Delta y)^2) \\ \tau_T &= \frac{\mu_{\text{diff}}}{\kappa_T} \min((\Delta x)^2, (\Delta y)^2) \\ \tau_{\text{visc}} &= \frac{\mu_{\text{diff}} \rho}{\eta} \min((\Delta x)^2, (\Delta y)^2) \\ \tau_{\text{fluid}} &= \frac{\mu_{\text{adv}}}{\max(|\vec{u}|)} \min(\Delta x, \Delta y) \\ \tau_{\text{diff}} &= \min(\tau_c, \tau_T) \end{aligned}$$

6.2.4 The merit of SSP IMEX Runge-Kutta methods

In simulations of stellar double-diffusive convection, most energy is transported by advection. By using the fractional step method, the timestep restriction due to sound waves is very much alleviated. However, it is now diffusion processes which enforce an unnecessarily small timestep. Through the use of IMEX RK methods, the timestep restriction due to diffusive processes can be effectively alleviated or even discarded, resulting in a much faster but equally accurate simulation.

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

The implicit stage equations in the double-diffusive convection model

Equations (6.1.2) - (6.1.5) can be recast as

$$\frac{\partial}{\partial t} \underbrace{\begin{pmatrix} \rho \\ \rho c_S \\ \rho \vec{u} \\ \rho e_{tot} \end{pmatrix}}_{y(t)} = \nabla \cdot \underbrace{\begin{pmatrix} \rho \vec{u} \\ \rho c_S \vec{u} \\ \rho \vec{u} \otimes \vec{u} + P - \tau \\ \rho e_{tot} \vec{u} + P \vec{u} - \vec{u} \cdot \tau \end{pmatrix}}_{F(y(t))} + \underbrace{\begin{pmatrix} 0 \\ 0 \\ g \\ g \vec{u} \end{pmatrix}}_{G(y(t))} + \underbrace{\begin{pmatrix} 0 \\ -\nabla \cdot (\rho \kappa_{c_S} \nabla c_S) \\ 0 \\ Q_{rad} \end{pmatrix}}_{G(y(t))} \quad (6.2.5)$$

where $F(y(t))$ and $G(y(t))$ are vector fields with different stiffness properties. As outlined in section 4.6.2, the implicit stage of an IMEX RK scheme is of the form

$$y_i = y^* + \Delta t \tilde{a}_{ii} G(y_i), \quad (6.2.6)$$

where y^* is known from previous stages. In view of the system (6.2.5), this translates to

$$\rho_i = \rho^*, \quad (6.2.7)$$

$$(\rho c)_i = (\rho c)^* - \Delta t \tilde{a}_{ij} \nabla \cdot (\rho_i \kappa_c \nabla c_i), \quad (6.2.8)$$

$$(\rho \vec{u})_i = (\rho \vec{u})^*, \quad (6.2.9)$$

$$e_i = e^* - \Delta t \tilde{a}_{ij} Q_{rad}. \quad (6.2.10)$$

Rearranging equation (6.2.8) leads to

$$\frac{\rho^*}{\Delta t \tilde{a}_{ii}} c_i - \nabla \cdot (\rho^* \kappa_c \nabla c_i) = \frac{\rho^* c^*}{\Delta t \tilde{a}_{ii}}. \quad (6.2.11)$$

which is a general linear elliptic equation in c_i of appearance (4.6.4).

Due to model assumptions, equation (6.2.10) can also be transformed to resemble a linear generalized Poisson equation. It is known that

$$e = e_{\text{int}} + e_{\text{kin}} = e_{\text{int}} + \frac{1}{2} \rho |\vec{u}|^2. \quad (6.2.12)$$

Using equation (6.2.9), equation (6.2.10) can be reformulated to

$$e_{\text{int } i} = e_{\text{int}}^* + \Delta t \tilde{a}_{ij} Q_{rad}. \quad (6.2.13)$$

In this model, $Q_{rad} = -\nabla \cdot (K_{rad} \nabla T)$. Invoking the EOS (6.1.10) and equation (6.1.11), the above expression becomes

$$-\nabla \cdot (K_{rad} \nabla T_i) + \frac{3}{2} \frac{R_{\text{gas}} \rho^*}{m_i \Delta t \tilde{a}_{ii}} T_i = \frac{e_{\text{int}}^*}{\Delta t \tilde{a}_{i,i}}. \quad (6.2.14)$$

In view of the global boundary conditions (6.1.18) - (6.1.19), it suggests itself to impose Dirichlet boundary conditions on both equations (6.2.11) and (6.2.14).

Note that the coefficient functions appearing in the linear generalized Poisson equations (6.2.11) and (6.2.14) are strictly positive. Thus, the existence of a unique solution is guaranteed (see [11]).

Timestep control

Due to the implicit treatment of the diffusive terms, the corresponding timestep restrictions are very much alleviated or even completely removed, if the implicit scheme is unconditionally stable. The remaining timestep criteria are those due to the fluid velocity, τ_{fluid} and of the viscous terms, τ_{visc} . In an implicit method, however, one has to introduce a timestep control strategy which is usually based on an empirical footing. In the literature, different control mechanisms have been invented (e.g. [50]).

One idea is to control the timestep Δt based on the convergence of the solver employed for the implicit equation by monitoring the number of iterations necessary to obtain an acceptable solution. If a certain number is exceeded, the timestep is decreased. If for a specific number of timesteps the number of iterations stays below the prescribed maximum iteration count, the timestep is increased.

Another approach is to compute the relative change in the solution between the timesteps m and $m + 1$. If the change is lower than a certain percentage, the timestep is increased, if it is larger than an acceptable percentage, the timestep is decreased.

The most effective criterion in regulating the timestep in simulations of double-diffusive convection, however, has proven to be the observation of two-point instabilities appearing in the conservative variables $(\rho, \rho c, \rho \vec{u}, Et)$. Due to the gravitational force operating vertically, such instabilities are prone to appear in the horizontal direction.

To detect the occurrence of such oscillations, the difference of the conservative variables between two grid cells is used to determine the sign of the corresponding gradient. In case of the density ρ this reads

$$\begin{aligned} d_1 &= \rho_{i,j-1} - \rho_{i,j-2} \\ d_2 &= \rho_{i,j} - \rho_{i,j-1} \\ d_3 &= \rho_{i,j+1} - \rho_{i,j} \\ d_4 &= \rho_{i,j+2} - \rho_{i,j+1} \end{aligned}$$

where $1 < i < n_x$ and $1 < j < n_y$, assuming the grid consists of $n_x \times n_y$ grid points.

If the sign pattern of (d_1, d_2, d_3, d_4) corresponds to $(+, -, +, \pm)$, $(-, +, -, \pm)$, $(\pm, +, -, +)$ or $(\pm, -, +, -)$, a two-point instability has been found. Since the timestep is initially chosen to be the one required for an explicit time integration method, such patterns are smoothed out rapidly

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

if present in the initial condition. Consequently, their later occurrence is a good indicator for an instability developing because of too large a timestep taken during the time integration.

The timestep control permits the occurrence of $n_y \cdot 0.1$ two-point instabilities for fixed i . If this limit is exceeded, the calculation is discarded and the actual timestep is repeated using a temporal increment of $\frac{2}{3}\Delta t$. To permit the system to readjust after reducing the temporal increment, no modifications to Δt are made for the next 15 timesteps regardless of the number of two-point instabilities. If the number of oscillations still exceeds the given limit after those 15 time steps, the temporal increment is diminished anew.

If no or very few two-point oscillations are encountered for 50 successive timesteps, the timestep is increased by a factor of $\frac{5}{4}$.

Furthermore, the timestep Δt must fulfill the following conditions:

- The time step restriction stemming from the viscous part provides an upper limit to Δt ,

$$\tau_{\text{visc}} \geq \Delta t$$

whereas τ_{diff} serves as a lower limit to the timestep,

$$\tau_{\text{diff}} \leq \Delta t$$

- In case of $\tau_{\text{diff}} \leq \tau_{\text{fluid}} \leq \tau_{\text{visc}}$, τ_{fluid} is an upper limit of Δt , whereas τ_{diff} provides a lower limit.

$$\tau_{\text{diff}} \leq \Delta t \leq \tau_{\text{fluid}}$$

- In case of the fluid velocity providing the most restrictive timestep $\tau_{\text{fluid}} \leq \tau_{\text{diff}}$, $\Delta t = \tau_{\text{fluid}}$ is chosen.

The Algorithm

In this paragraph, the integration of the equations of two-component fluid flow with an IMEX method as implemented in the ANTARES code is described step by step. The index s denotes an arbitrary stage.

(i) At the start, $y_{\text{old}}^{(s)} = (\rho^*, (c\rho)^*, (\rho u)^*, (\rho v)^*, (et)^*)$ is given.

(ii) Evaluation of the implicit stage:

The implicit stage is given formally by (6.2.6). Practically, this translates into an implicit equation for $(c\rho)$ given by equation (6.2.11) and an implicit equation for (et) , namely, equation (6.2.14). Note that (6.2.11) must be evaluated first, since its output is used to calculate m_i which enters the coefficient functions of (6.2.14). The results are gathered in $y_{\text{imp}}^{(s)} = (\rho^*, (c\rho)_{\text{imp}}, (\rho u)^*, (\rho v)^*, (et)_{\text{imp}})$.

(iii) Explicit sweep

Using $y_{\text{imp}}^{(s)}$, an explicit sweep

$$y_{\text{exp}}^{(s)} = y_{\text{old}}^{(s)} + \Delta t F(y_{\text{imp}}^{(s)}) \quad (6.2.15)$$

is performed. In context of the double-diffusive convection problem, $F(y_{\text{imp}}^{(s)})$ is evaluated using the fractional step method introduced in section (6.2.2).

(iv) 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, the prescription

$$y_{\text{sum}}^{(s)} = y_{\text{old}}^{(s)} + \Delta t b_i F(y_i^{(s)}) + \Delta t \tilde{b}_i G(y_i^{(s)}) \quad (6.2.16)$$

of definition (4.3.1) translates to

$$y_{\text{sum}}^{(s)} = y_{\text{old}}^{(s)} + b_i (y_{\text{exp}}^{(s)} - y_{\text{old}}^{(s)}) + \frac{\tilde{b}_i}{\tilde{a}_{ii}} (y_{\text{imp}}^{(s)} - y_{\text{old}}^{(s)}). \quad (6.2.17)$$

(v) Assembly of the “new” $y_{\text{old}}^{(s+1)}$

If s equals the total number of stages, $y_{\text{new}} = y_{\text{sum}}^{(s)}$ 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 such that according to definition (4.3.1)

$$y_{\text{imp}}^{(s+1)} = y_{\text{old}}^{(s+1)} + \Delta t \tilde{a}_{s+1,s+1} G(y_{\text{imp}}^{(s+1)}) \quad (6.2.18)$$

(vi) The procedure (i)-(v) is repeated.

6.2.5 Multigrid Setting

It has been discussed in Chapter 5 that many different parameters influence the convergence properties of the multigrid solver. The multigrid setting used to solve the implicit equations arising in the simulations of double-diffusive convection is outlined in the following.

To obtain the coarse grid operators, the Galerkin coarsening strategy is used. The number of grid points chosen in the simulations (429×428) allows a multigrid hierarchy consisting of three grid levels, which is traversed by V-cycling. The number of pre- and postsmoothing sweeps is set to $\nu_1 = \nu_2 = 2$.

How to choose the best smoothing procedure, however, is less obvious due to the similarity of the results reported in Table 5.7. Therefore, the semiconvective setting Simulation A which will be outlined in the next section has been evolved for 50 scrt using the smoothers ZGSx, GS-RB and a few steps of the CG-algorithm. Figure 6.3 shows the timestep evolution using different

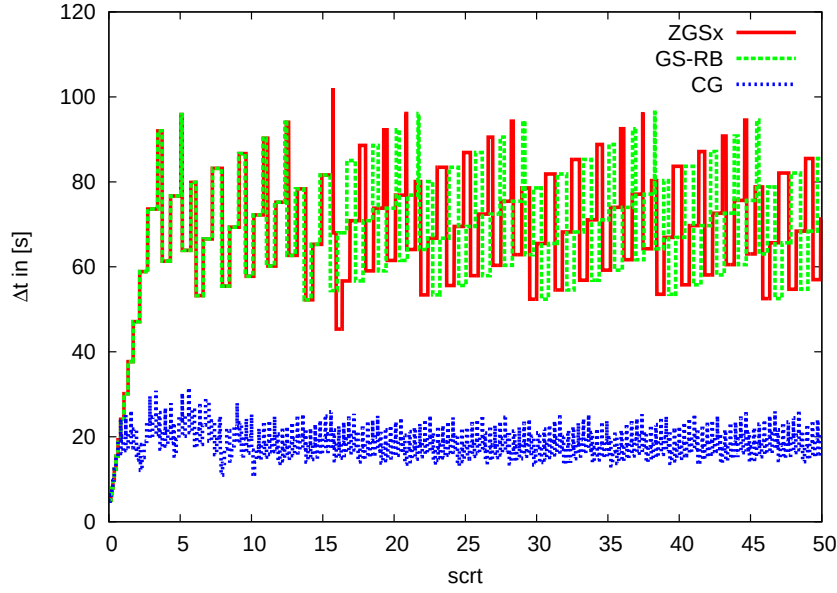


Figure 6.3: The timestep evolution of a Simulation A using different smoothers in the multigrid solver is depicted.

smoothing techniques. The timestep evolution indicates that performing a few steps of the CG-solver has very poor smoothing abilities. Instead of producing a smooth solution, the usage of the CG-smoother makes the overall solution more likely to contain two-point instabilities, which in turn are used to control the timestep (see section 6.2.4). Thus, this effect directly results in small timesteps. The GS-RB and the ZGSx-smoothers provide similar good smoothing properties such that nearly the same timesteps are achieved. The differences in the timestep after the first 15 scrt can be attributed to the evolution of turbulence (see [29]).

To determine whether the ZGSx or the GS-RB scheme performs better for simulations of double-diffusive convection, the number of multigrid cycles necessary to solve the the implicit equations for the mass fraction c , the temperature T and the pressure-correction equation arising in the fractional step method is assessed and depicted in Figure 6.4. Whereas for the implicit equation for the mass-fraction c and the implicit temperature equation T both schemes perform rather similarly and a slight advantage of the GS-RB smoother can be seen, this picture is reversed by comparing the number of multigrid-cycles necessary to solve the pressure-correction equation. In the latter case, the ZGSx smoother performs much better than the GS-RB smoother. This behavior can be attributed to the following issues: both the implicit mass fraction- and temperature equation arise due to the employment of IMEX methods. It is assumed that the diffusion of those quantities is a stiff process since both the mass fraction and the temperature change only little in a timestep. Therefore, also only a few iterations of the multigrid solver are necessary to solve the equations, especially so since both quantities are of order unity in the idealized semiconvective setting. The pressure, however, does change significantly in one timestep, thus more multigrid cycles are necessary for solving the equation up to the desired tolerance. Due to the influence of gravity, the x-direction is distinguished and line-smoothers are reported to have

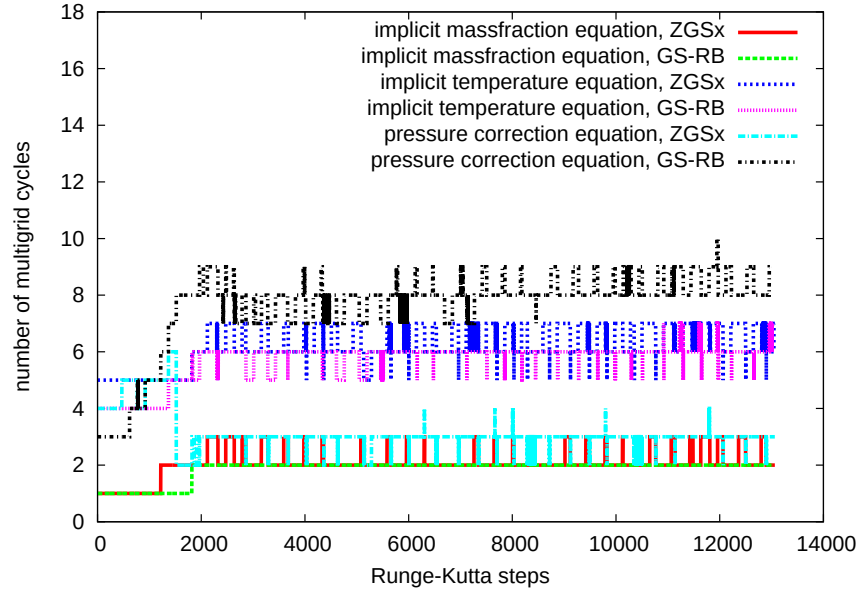


Figure 6.4: The number of multigrid cycles necessary to solve the arising implicit equations with the ZGSx and the GS-RB smoother is depicted.

favorable convergence properties especially in such a setting.

Thus, also the wallclocktimes reported in Table 6.1 are easily explained: due to the small timesteps achieved, the simulation employing a few steps of the CG-Algorithm as a smoother in the multigrid solver is slowest. Furthermore, due to the slightly higher number of multigrid iterations necessary in one step if the GS-RB smoother is used, its wallclocktime is higher than the one for the ZGSx-smoother. The best performing smoother is therefore the ZGSx-smoother, which will henceforth be the relaxation method of choice in simulations of double-diffusive convection.

smoother	WCT _{norm}
GS-RB	1
ZGSx	0.79
CG	1.80

Table 6.1: Normed wallclocktimes for the first 50 scrt in Simulation A employing different smoothers. The simulations have been performed on the Vienna Scientific Cluster 1 on 64 CPU-cores.

Full Multigrid (FMG)

The multigrid algorithm developed in Chapter 5 is easily used as a full multigrid (FMG) program. The main difference is that instead of starting with an arbitrary initial guess on u , the first approximation is obtained by an interpolation of a coarse-grid approximate solution. In an implementation, this is realised by calling to the MG-CS routine with $\nu_1 = \nu_2 = 0$. The idea is that the number of multigrid cycles necessary to obtain a solution is reduced by providing a good initial guess. A detailed introduction on FMG-techniques is found in [4].

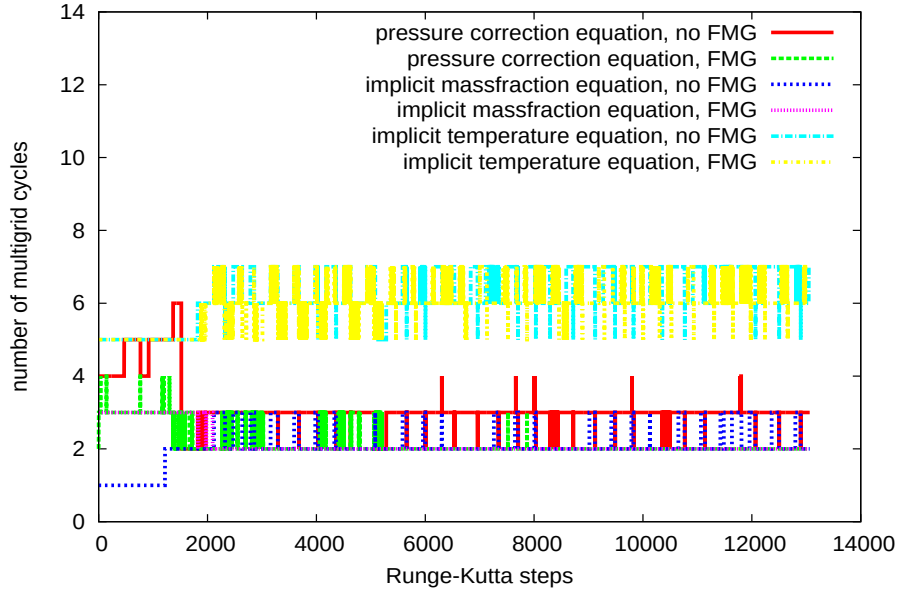


Figure 6.5: The influence of the FMG-technique on the number of multigrid cycles necessary to solve the implicit equations arising in one stage of a simulation of double-diffusive convection is depicted for the first 50 scrt.

In the following testsuite it is examined whether the initial guess stemming from the previous timestep or from an interpolation of a coarse grid solution is more advantageous in terms of CPU-time. Therefore, the double-diffusive convection setting later referred to as Simulation A has been evolved in time using the FMG-algorithm. Table 6.2 compares the computational efficiency of a simulation run with FMG and without, over the first 50 scrt and over the entire run of 200 scrt. The influence of the FMG-method on the number of multigrid iterations necessary to solve the implicit massfraction-, temperature- and pressure-correction equation is depicted in Figure 6.5.

	$WTC_{\text{norm}}, 0-50 \text{ scrt}$	$WTC_{\text{norm}}, 0-200 \text{ scrt}$
FMG	1	1
no FMG	1.01	0.92

Table 6.2: Comparison of the computational efficiency of a simulation of double-diffusive convection run with FMG and without.

Whereas the FMG-technique clearly decreases the number of multigrid cycles necessary to obtain an accurate solution, the computational time reflects this fact only in the first 50 scrt. Considering an entire simulation of double-diffusive convection, it becomes clear that as soon as the advective phase is reached and the timestep is dictated by the hydroinduced motions of the fluid, the advantage FMG offers vanishes and the additional effort of solving the coarse grid equation is reflected in the computational time. Thus, FMG is turned off in the simulations of double-diffusive convection.

6.3 Results and Discussion

Simulations of double-diffusive convection are carried out with different time integrators. For all simulations, an equidistantly spaced Cartesian grid discretized with 429×428 grid points is used. To test the efficiency especially of the optimized IMEX Runge-Kutta schemes, the parameters Pr , Le , Ra^* and R_ρ are varied. Three different settings of double-diffusive convection are considered, the characterizing dimensionless numbers are given in Table 6.3. Note that all the parameters are chosen such as to resolve the viscous and the diffusive boundary layer δ_{visc} and δ_c with approximately 5 grid points in vertical direction (compare equations (6.2.2) and (6.2.3), respectively). In the simulations considered, $1 \text{ sct} = 5125.5 \text{ s}$ and the simulation time is 200 sct.

	Pr	Le	R_ρ	Ra^*
Simulation A	0.1	0.1	1.1	160 000
Simulation B	0.05	0.05	1.1	160 000
Simulation C	0.5	0.1	1.1	160 000

Table 6.3: Parameters used in the simulations of double-diffusive convection.

In the interpretation of the results, special attention is paid to the following issues:

- Comparison of the performance of the explicit and the IMEX time integrators.
- Comparison of the performance of the IMEX methods from the literature and the optimized IMEX methods.
- Comparison of the performance of the SCpCG and the MG-parCG solver.

Tables 6.4, 6.5, and 6.6 summarize the results of the simulation series A, B, and C. In those tables, the μ_{adv} column gives the advective Courant number used, and Δt_{mn} denotes the mean timestep over the indicated time. The normalized wallclocktime WCT_{norm} as well as the measured wallclocktime WCT is given. It is important to bear in mind that the measured wallclocktime depends sensitively on the computer architecture, the compiler and the compiler options employed, furthermore, also the MPI-version linked to the code influences the runtime. In the present context, simulation series A has been performed on VSC 1 whereas the series B and C have been computed on VSC 2. Thus, the measured wallclocktimes are not comparable. The normalized wallclocktime, however, provides the means to compare the relative performance of the IMEX methods.

The Figures 6.6 and 6.7 show snapshots of simulation B integrated with different time integrators at $t=100 \text{ sct}$ and $t=200 \text{ sct}$, respectively. The very same “random” initial perturbation has been inferred to all the simulations of setting B.¹ At $t=100 \text{ sct}$, the simulation is just at the beginning of the turbulent phase and the differences between the simulations integrated with different time

¹Actually, all the simulations of one setting have been initialized using the same initial perturbation.

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

integrators are very small, indicating thus that also large timesteps during the diffusive phase yield acceptable accuracy. At $t=200$ scrt, however, the simulations have notably evolved away from each other due to turbulent mixing.

For a stable integration of the simulations in the settings B and C, the adaptive timestep control outlined in section 6.2.4 has been slightly modified. In simulation series A, the timestep was readjusted at the occurrence of $n_y * 0.1$ two-point oscillations (n_y denoting the number of grid points in the horizontal direction) and, to permit the system to readjust itself, no modifications on the timestep were allowed in the subsequent 15 timesteps. However, for simulation series B, this criterion was chosen to be more restrictive, namely, the timestep is diminished at the occurrence of $n_y * 0.05$ two-point instabilities and if the system has not adjusted itself after 8 timesteps, the timestep is further reduced.

Comparison of the explicit and the IMEX time integrators

In all three simulation series, the explicit three-stage second order SSPRK(3,2) method performs much better than its two-stage counterpart SSPRK(2,2). This is due to the fact that the maximum admissible diffusive Courant number of the SSPRK(3,2) method is about 1.8 times higher than the one of the SSPRK(2,2) scheme (see Table 4.1). However, it is not advisable to integrate at the stability limit which is the reason that in the simulations with SSPRK(2,2) a (diffusive) Courant number of 0.2 and in the simulations with SSPRK(3,2), a (diffusive) Courant number of 0.5 is used. The difference of a factor of 2.5 in the Courant numbers translate to an average runtime about 1.5 times faster. The difference between the ratio of Courant numbers and the efficiency gain is easily explained by pointing out that within the SSPRK(3,2) method, a third stage has to be computed each timestep.

Using IMEX RK-methods, a much higher timestep especially in the diffusive part of the simulation can be achieved. Since the diffusive terms are evaluated implicitly, only an upper limit of the diffusive Courant number to be applied in the evaluation of τ_T and τ_c is provided in case the implicit subscheme is not unconditionally stable. As an upper limit on μ_{diff} the findings of the stability analysis (see Table 4.1 in section 4.5) are used. Note, however, that the viscous terms are integrated explicitly, thus the maximum admissible diffusive Courant number of the explicit subscheme applies in the computation of the viscous timestep restriction τ_{visc} . The timestep in the diffusive regime of the simulation is controlled by the adaptive timestep control outlined in section 6.2.4.

Note that the necessity of solving three linear generalized Poisson equations each Runge-Kutta stage instead of only one as with the explicit time integration methods takes its toll: In all simulation series, the simulations carried out with the IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$ is computationally nearly as or even more expensive than the same simulation carried out with the explicit SSPRK(3,2) method in spite of the higher timestep employed.

It is interesting to observe the difference in the timestep of the IMEX SSP2(2,2,2) method simply

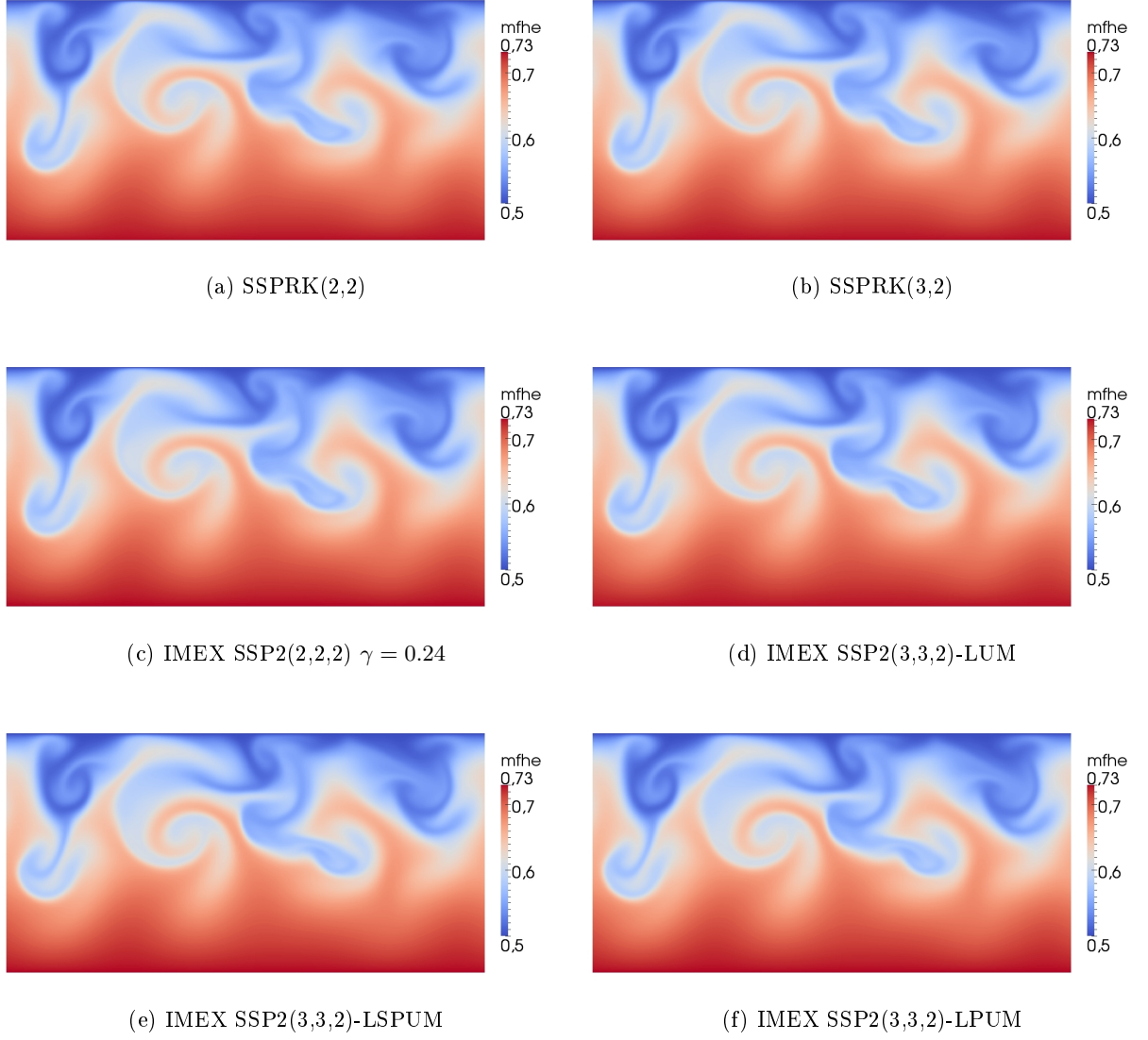
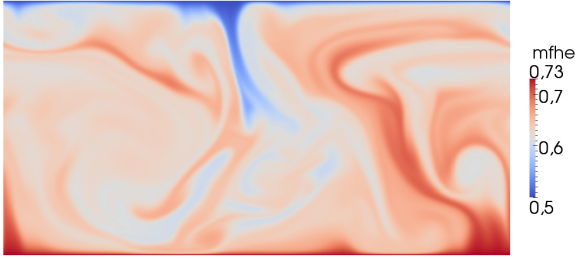
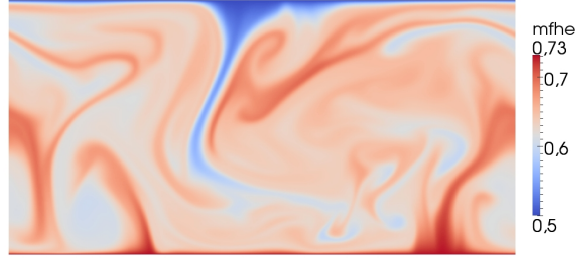


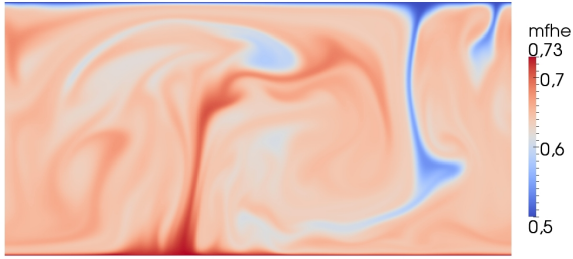
Figure 6.6: Snapshots of Simulation B at $t=100$ scrt integrated with different time integrators. The helium massfraction c (in the pictures, it is denominated **mfhe**) is depicted.



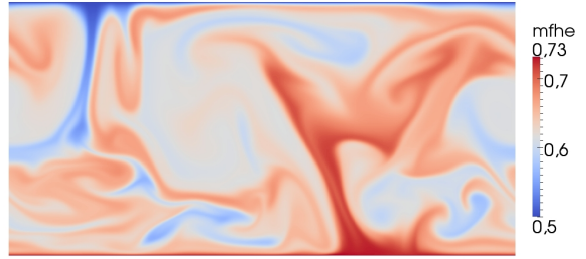
(a) SSPRK(2,2)



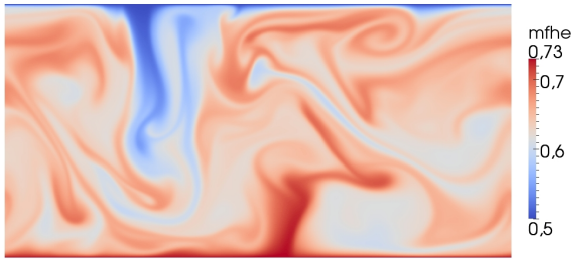
(b) SSPRK(3,2)



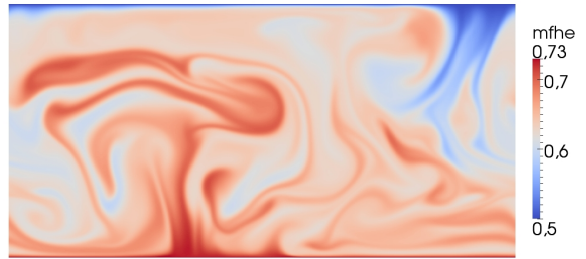
(c) IMEX SSP2(2,2,2) $\gamma = 0.24$



(d) IMEX SSP2(3,3,2)-LUM



(e) IMEX SSP2(3,3,2)-LSPUM



(f) IMEX SSP2(3,3,2)-LPUM

Figure 6.7: Snapshots of Simulation B at $t=200$ scrt integrated with different time integrators. The helium massfraction c (in the pictures, it is denominated **mfhe**) is depicted.

by varying the parameter γ . It seems that the positivity of the amplification factor g guaranteed for IMEX SSP2(2,2,2) methods with $\gamma < 0.25$ introduces more stability to the simulation than the property of L-stability possessed by IMEX SSP2(2,2,2) scheme with $\gamma = 1 - 1/\sqrt{2}$ (compare Figures 6.8, 6.9 and 6.10).

The three-stage second-order IMEX SSP2(3,3,2) methods permit a timestep about 2.6 times larger than the IMEX SSP2(2,2,2) $\gamma = 0.24$ scheme. This effect reminds of the better performance of the SSPRK(3,2) than the SSPRK(2,2) method and permits the conclusion that a third stage in a scheme of second order accuracy adds stability to the simulation. Figures 6.8 and 6.9 depict the timestep evolution with different time integrators in both the settings A and B.

The gain in computational time due to the employment of IMEX methods compared to the best performing explicit scheme SSPRK(3,2) ranges from a factor of 1.5 (IMEX SSP2(3,3,2)-LPM) to 2.1 (IMEX SSP2(3,3,2)-LUM) in simulation series A and from a factor of 1.26 (IMEX SSP2(2,2,2), $\gamma = 0.24$) to 2.0 (IMEX SSP2(3,3,2)-LPUM) in simulation series B if the entire simulation time of 200 scrt is considered. If only the diffusive phase of the simulation is taken into account, the differences in runtime are even higher, ranging from a factor of 1.24 (IMEX SSP2(2,2,2) $\gamma=0.24$) to 4.1 (IMEX SSP2(3,3,2)-LPUM) in simulation series A and from a factor of 1.34 (IMEX SSP2(2,2,2) $\gamma=0.24$) to 5.6 (IMEX SSP2(3,3,2)-LPUM) in simulation series B. In both settings, the IMEX SSP2(2,2,2) $\gamma = 1 - 1/\sqrt{2}$ schemes performs significantly worse than the SSPRK(3,2) method, since the gain in timestep is too small to pay off for the additional effort to solve the implicit stage equations. The percentages have been assessed by comparing the simulations carried out employing the MG-parCG solver.

The mean timestep in the best performing IMEX scheme, namely the IMEX SSP2(3,3,2)-LPUM method, is about 20.7 times (simulation series A) and even 26.7 times (simulation series B) higher than the mean timestep employed by the SSPRK(2,2) scheme. If the SSPRK(3,2) scheme is considered, a timestep 8.3 times (simulation series A) and 10.5 times (simulation series B) is achieved.

Since the upper limit set by τ_{visc} is not reached in the simulation series A and B, a resolution test to determine whether this discrepancy results from the inaccuracy of the spatial discretization has been performed for setting B and the IMEX SSP2(3,3,2)-LSPUM and IMEX SSP2(3,3,2)-LPUM time integrators. To this end, the first 20 scrt are run with a resolution of 229×228 , 429×428 , 829×828 and 1629×1628 grid points. Table 6.7 compares the observed effective diffusive Courant numbers. Indeed, the Courant numbers appear to display some trend as the resolution improves.

In simulation setting C, the viscous timestep limit is such as to provide an effective restriction to the timestep already at the given resolution of 429×428 grid points. Apart from the explicit schemes SSPRK(2,2) and SSPRK(3,2), which are restricted by τ_c , all the IMEX methods reach the maximum timestep set by τ_{visc} which in turn depends significantly on the maximum admissible diffusive Courant number of the explicit subscheme of the IMEX method. No security factor is employed, thus the integration takes place at the diffusive stability limit (see Table 4.1) of the explicit subscheme. Note that only the IMEX SSP2(3,3,2)-LSPUM method is ca-

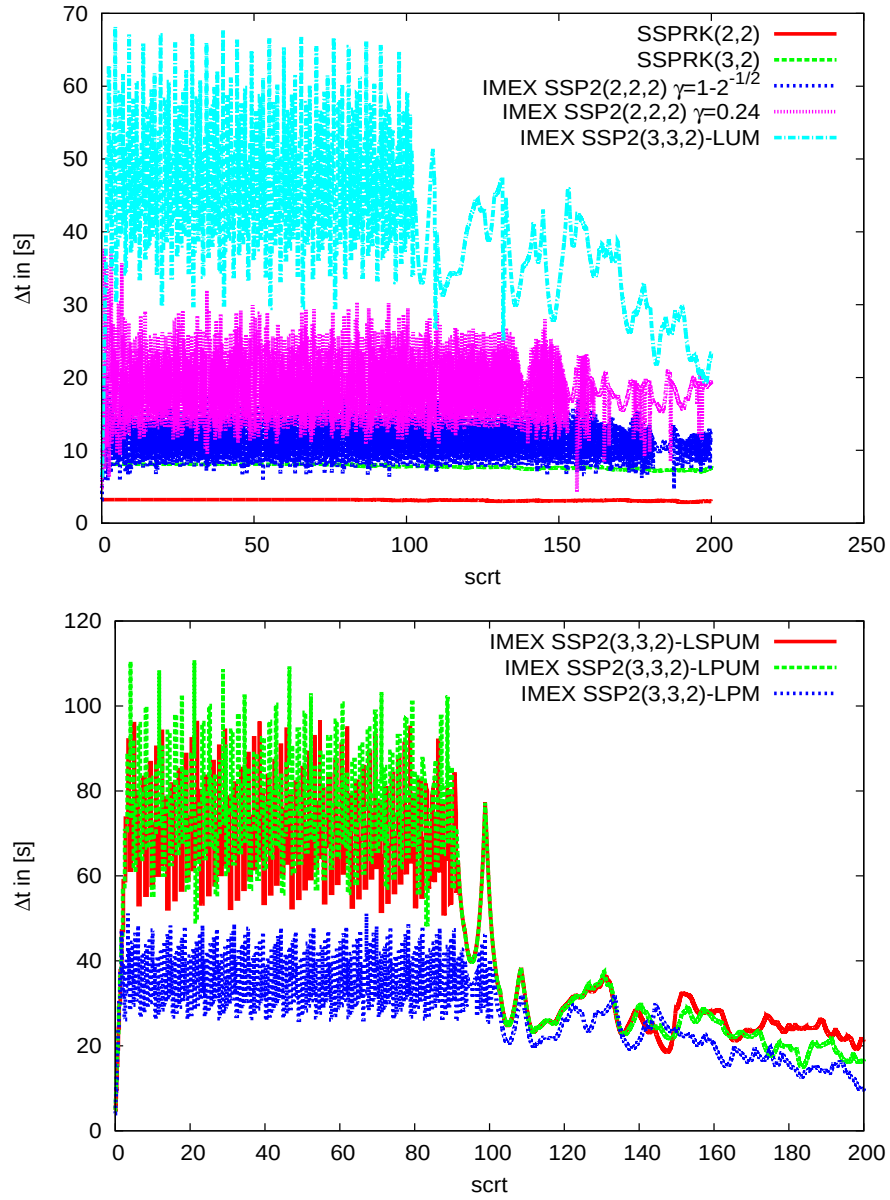


Figure 6.8: The timestep evolution using different time integrators in simulation series A.

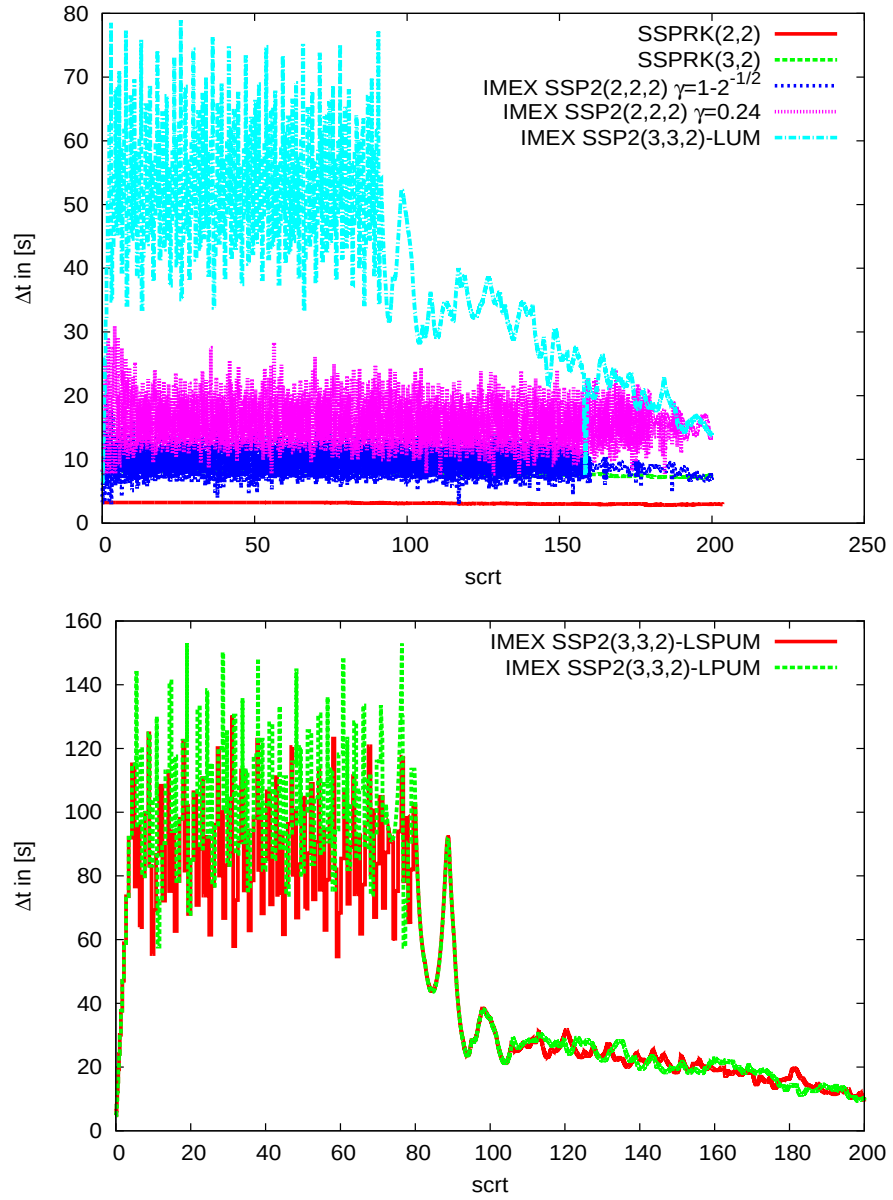


Figure 6.9: The timestep evolution using different time integrators in simulation series B.

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

pable of employing the limit timestep throughout the entire simulation. However, due to the significantly lower maximum diffusive Courant number of its explicit subscheme (c.f. Table 4.1), the IMEX SSP2(3,3,2) methods employing the SSPRK(3,2) method as the explicit subscheme achieve significantly better results in terms of the mean timestep Δt_{mn} and the computational time. Thus, in simulation series C, the best performing scheme is the IMEX SSP2(3,3,2)-LPUM scheme, which achieves a timestep 8.7 times higher than the explicit SSPRK(2,2) scheme and 3.5 times higher than the SSPRK(3,2) method. Considering the wallclocktime, this results in a runtime about 1.7 times faster compared to the SSPRK(2,2) scheme and 1.14 times faster if the SSPRK(3,2) scheme is considered. Note that in this setting, the explicit SSPRK(3,2) scheme is more efficient than the IMEX SSP2(3,3,2)-LSPUM and the IMEX SSP2(3,3,2)-LPM schemes. The timestep evolution of the different time integrators in simulation series C is depicted in Figure 6.10.

Comparison of the performance of the IMEX methods from the literature and the optimized IMEX methods

The optimized IMEX methods IMEX SSP2(3,3,2)-LSPUM and IMEX SSP2(3,3,2)-LPUM allow stable integration for timesteps 1.5 (simulation setting A) and 1.8 (simulation setting B) times larger than the best performing IMEX SSP scheme from the literature, i.e. IMEX SSP2(3,3,2)-LUM. In the diffusive regime (approximately the first 100 scrt in simulation A and the first 80 scrt in simulation B) this translates into a runtime nearly 1.52 times faster in simulation A and 1.6 times faster in simulation B. However, as soon as the advective phase is reached, the advantage of employing a higher advective Courant number (0.4 in the simulations performed with IMEX SSP2(3,3,2)-LUM and 0.3 with the optimized IMEX schemes) becomes notable. For the entire simulation time of 200 scrt, in simulation series A, the IMEX SSP2(3,3,2)-LUM method proves to be the fastest scheme considering the CPU time (02:51:59 compared to 02:59:38 of the IMEX SSP2(3,3,2)-LPUM scheme). In simulation B, the IMEX SSP2(3,3,2)-LPUM scheme wins over the IMEX SSP2(3,3,2)-LUM scheme (05:00:19 against a runtime of 05:02:34). However, it is foreseeable that if the simulations in setting B are continued, the simulation run with the IMEX SSP2(3,3,2)-LUM scheme will overtake the simulation integrated with the IMEX SSP2(3,3,2)-LPUM method due to its higher advective Courant number.

The schemes IMEX SSP2(3,3,2)-LSPUM and IMEX SSP2(3,3,2)-LPUM only differ in the property “S”, which corresponds to the inclusion of a part of the imaginary axis in the stability region of the explicit subscheme. Both schemes perform similarly and the IMEX SSP2(3,3,2)-LPUM scheme even allows a timestep slightly greater than the IMEX SSP2(3,3,2)-LSPUM method. Therefore, it is concluded that this property is of no importance in the simulations of double-diffusive convection. This conclusion is backed by the discussion of [34] and [51] whether the inclusion of parts of the imaginary axis in the stability region of an explicit scheme is necessary to ensure linear stability of the method in question when coupled to the WENO spatial discretization or not.

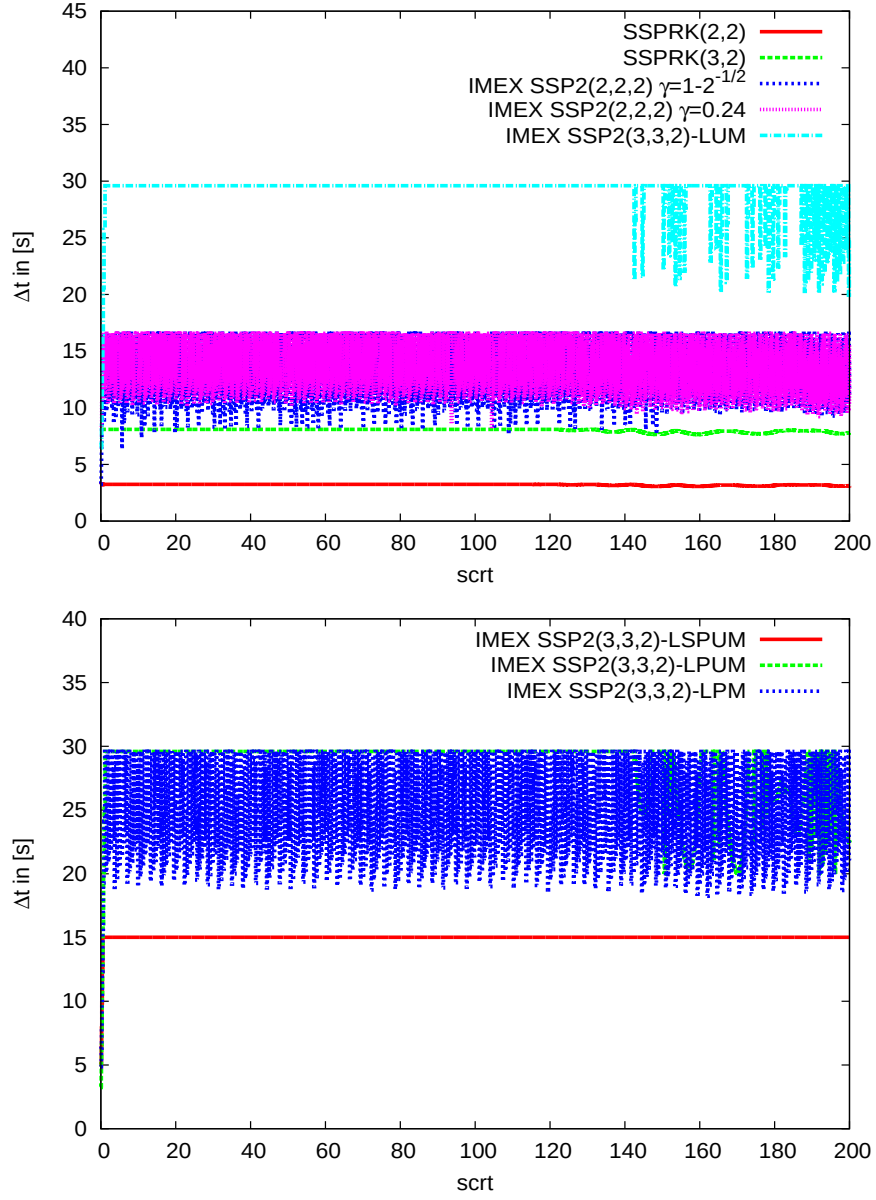


Figure 6.10: The evolution of the timestep with different time integrators in simulation series C.

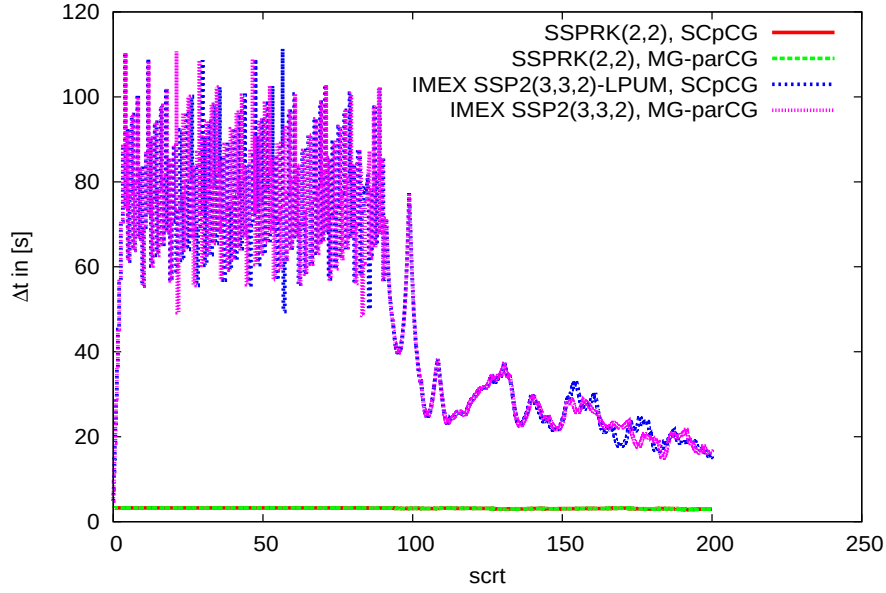


Figure 6.11: Dependency of the timestep on the linear solver in simulation setting A.

The IMEX SSP2(3,3,2)-LPM method, however, performs clearly worse than the IMEX SSP2(3,3,2)-LSPUM and the IMEX SSP2(3,3,2)-LPUM methods. The mean timestep employed by the IMEX SSP2(3,3,2)-LPM method is only half the timestep employed by the IMEX SSP2(3,3,2)-LPUM scheme. For simulation series B, no results of a simulation integrated with the IMEX SSP2(3,3,2)-LPM method are given in Table 6.5 since the integration with the IMEX SSP2(3,3,2)-LPM method conveys too little stability to the simulation to get over the transition of the diffusive to the advective regime of the simulation. Instead, the simulation breaks down. This indicates that the property of uniform convergence (“U”) is important for a stable integration of the simulations under consideration.

Comparison of the performance of the SCpCG and the MG-parCG solver

The results in Tables 6.4, 6.5 and 6.6 as well as the illustration in Figure 6.11 show that the mean timestep does not depend on the linear solver. The performance of the solvers, however, differ drastically which is reflected in the computational time. For the explicit SSPRK(2,2) and SSPRK(3,2) methods the SCpCG solver renders considerably faster accurate results for the pressure correction equation, and thus allows an integration about 1.1 (simulation series A) and 1.03 (simulation series B) faster than with the MG-parCG solver if the explicit time integrator SSPRK(2,2) is considered. If the SSPRK(3,2) scheme is used, the efficiency gain is about a factor of 1.01 (simulation series A) and 1.04 (simulation series B). As soon as a higher timestep is used, the picture changes. The reason for this behavior is found by having a look at the scaling and convergence properties of the two solvers. The scaling test in Table 5.17 shows that the SCpCG method scales almost optimally, whereas the parallelization of the multigrid iteration incurs a certain overhead. The Conjugate Gradient method, however, converges much slower than the multigrid technique, and the bottleneck of the Schur Complement parallelization technique is the

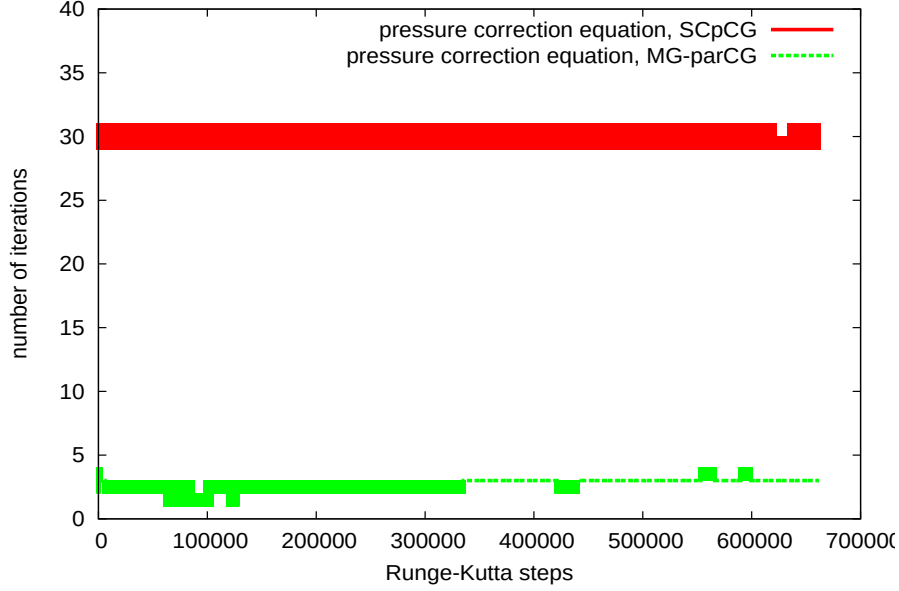
necessity of solving the Schur Complement equation by the directly parallelized CG-algorithm. The higher the timestep, the more the variables vary from one timestep to the next and more iterations are necessary to obtain a satisfying solution. Using the SCpCG solver, this is heavily reflected in the computational time. Employing the MG-parCG solver, however, the superior convergence properties of the multigrid iteration pay off and only a few additional iterations are necessary to obtain solutions to the implicit equations (see Figure 6.12).

The different performance of the SCpCG and the MG-parCG solver is most impressively quantified by comparing the results obtained in the diffusive part of the simulations in simulation series B: in the simulations performed with IMEX SSP2(3,3,2)-LSPUM and -LPUM, the employment of the MG-parCG solver instead of the SCpCG solver results in a runtime about 3.8 times faster. Thus, the efficiency gain obtained with the IMEX SSP2(3,3,2)-LSPUM and -LPUM schemes and the SCpCG solver, which amounts to a factor of 2.3 in the runtime compared to the explicit SSPRK(2,2) method and 1.4 in case the SSPRK(3,2) scheme is considered, is further enhanced by the factor of 3.8 if the MG-parCG solver is used.

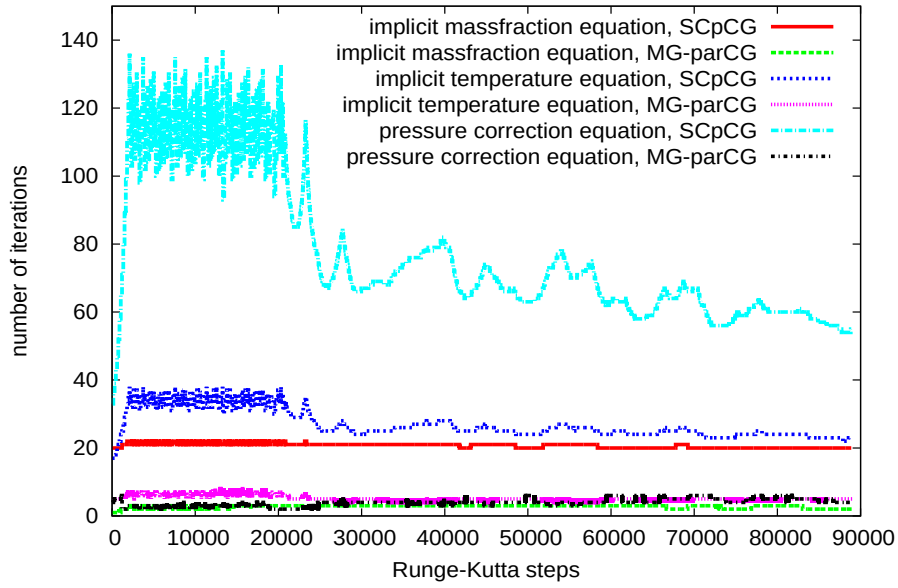
The acceleration due to the use of the multigrid technique can also be seen by comparing the first results of using SSP IMEX methods from the literature in simulations of double-diffusive convection, which have been published in [29]. (Note that simulation setup A corresponds to the setup used in the tests in [29].) As in the results found there, the highest timestep comparing the SSP IMEX methods from the literature is achieved by the IMEX SSP2(3,3,2)-LUM method, which bests the timestep achieved in the simulation with SSPRK(3,2) by a factor of 6.1. Since the SCpCG solver is used, the higher timestep translates only into a runtime about 1.05 times faster.² The results in Table 6.4 show that replacing the SCpCG-solver by the solver based on the multigrid technique results in an overall runtime about 2.2 times faster instead (in this comparison, the wallclocktimes of the entire runs, 200 scrt, are considered).

Having a look at the multigrid scaling tests in section 5.7.2 it becomes clear that employing 64 CPU-cores in a simulation resolved with 429×428 grid points means that a significant part of the computational time used by the multigrid solver is spent on communication. To lessen the communication overhead and determine its influence, the simulations carried out with the time integrators SSPRK(3,2) and IMEX SSP2(2,2,2) $\gamma = 0.24$ of simulation series A have been repeated employing 16 CPU-cores. The results are given in Table 6.8. Naturally, the mean timestep size Δt_{mn} has not changed, however, even using the explicit time integration method SSPRK(3,2) the application of the MG-parCG solver results in a decrease of computational time compared to the simulation carried out with the SCpCG solver. Note that the scaling properties encountered in this application correspond well to those found in the scaling tests as presented in Table 5.17. Whereas the time integration performed on 16 CPU-cores using the SCpCG solver takes on average 4.2 times longer than its counterpart on 64 CPU-cores, the difference employing the MG-parCG solver is less than a factor of 2. Thus, it has to be pointed out that in terms of CPU-time = number of CPU-cores $\times$ runtime, it is more efficient to use the MG-parCG solver

²In this comparison, the results published in [29] are used. Note, however, that the simulation there has a resolution of 400×400 grid points whereas the simulations conducted within this thesis are spatially resolved with 429×428 grid points. Thus, the comparison is only a qualitative one.



(a) SSPRK(2,2)



(b) IMEX SSP2(3,3,2)-LPUM

Figure 6.12: The number of iterations necessary to solve the Schur Complement equation as well as the number of multigrid cycles using different time integrators is depicted. The values have been obtained using the simulations of series A.

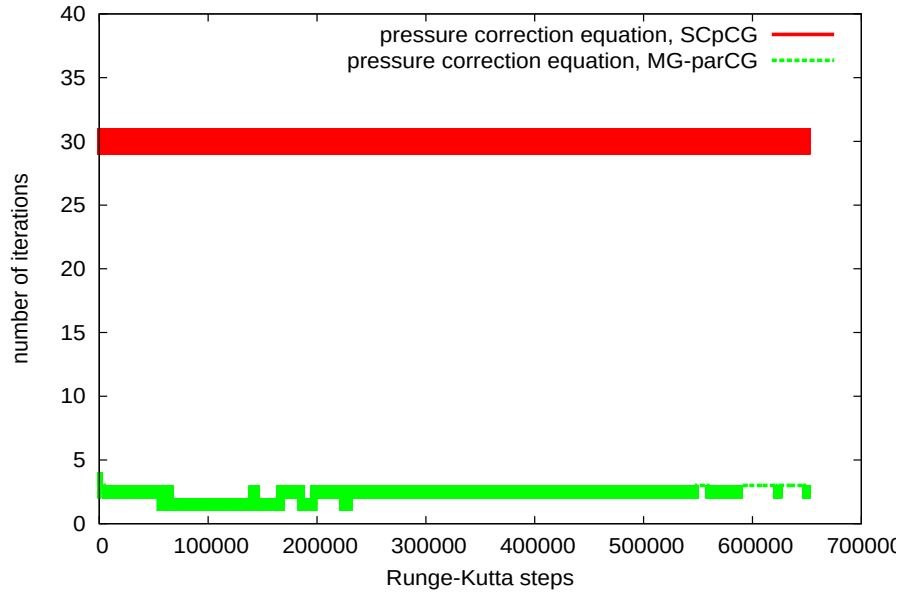


Figure 6.13: The number of iterations necessary to solve the Schur Complement equation as well as the number of multigrid cycles used to solve the pressure-correction equation in simulation setting C using the explicit time integrator SSPRK(2,2).

on 16 processors instead of employing 64 CPU-cores. However, a comparison of Table 6.4 and Table 6.8 shows that, in case the CPU-time is considered, it is also more efficient to employ the MG-parCG solver on 16 processors than running the same simulation on 64 CPU-cores using the SCpCG solver.

Note that in simulation series C, the simulations run with the explicit time integrators SSPRK(2,2) and SSPRK(3,2) are computationally more efficient if the MG-parCG solver is used. Whereas this is plausible for the simulations run with SSPRK(3,2) since the corresponding differences in simulation series A and B are also small, the quite significant difference in the runs with SSPRK(2,2) must be accounted for. To this end, the number of iterations necessary to solve the pressure-correction equation is depicted in Figure 6.13 and compared to Figure 6.12, which depicts the number of iterations performed to solve the pressure-correction equation in simulation setting A. However, no significant differences are detected and therefore, the differences in computational time do not stem from a significant increase of iterations within the SCpCG-solver. Thus, the measured runtime differences must be attributed to an unlucky assignment of computational nodes to the simulation run with SSPRK(2,2) and the SCpCG solver, namely nodes between which communication is slow due to the specific network structure of the VSC2. The same holds for the simulation of series C performed with the time integrator IMEX SSP2(3,3,2)-LPM and the MG-parCG solver: although a quite high timestep can be used, the gain in computational efficiency due to the employment of the multigrid solver is significantly worse than it is in a comparable solution.

Short Summary

- As soon as a significantly higher timestep is achieved, the MG-parCG solver renders accurate results significantly faster than the SCpCG solver. This difference amounts up to a factor 3.8 in runtime.
- The most efficient SSP IMEX method in simulations of double-diffusive convection is the IMEX SSP2(3,3,2)-LPUM scheme. It achieves significantly higher timesteps than traditional explicit time integration schemes and beats also the other SSP IMEX schemes. Using the IMEX SSP2(3,3,2)-LPUM scheme, the simulations are carried out in a fraction of the computational time other methods spend on the same problem.
- The properties “P” – positivity of the amplification factor – and “U” – uniform convergence – have a notable beneficial influence on the stability of the time integration.
- The property “S” – inclusion of a part of the imaginary axis in the stability region of the explicit subscheme – is not important in the context of simulations of double-diffusive convection.

Method	μ_{adv}	$\Delta t_{\text{mn}}, 100$ scrt	WCT _{norm} (WTC), 100 scrt	WCT _{norm} (WTC), 200 scrt
SSPRK(2,2), SCpCG	0.2	3.23 s	1 (04:23:24)	1 (08:57:08)
SSPRK(2,2), MG-parCG	0.2	3.23 s	1.07 (04:42:24)	1.12 (10:02:58)
SSPRK(3,2), SCpCG	0.5	8.06 s	0.71 (03:05:52)	0.71 (06:21:13)
SSPRK(3,2), MG-parCG	0.5	8.06 s	0.73 (03:13:13)	0.72 (06:24:35)
IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$, SCpCG	0.2	10.58 s	0.82 (03:35:37)	0.82 (07:16:42)
IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$, MG-parCG	0.2	10.57 s	0.73 (03:12:15)	0.75 (06:42:29)
IMEX SSP2(2,2,2), $\gamma = 0.24$, SCpCG	0.5	17.63 s	0.60 (02:35:37)	0.59 (05:12:08)
IMEX SSP2(2,2,2), $\gamma = 0.24$, MG-parCG	0.5	17.66 s	0.46 (02:01:50)	0.47 (04:11:58)
IMEX SSP2(3,3,2)-LUM, SCpCG	0.4	43.49 s	0.60 (02:36:33)	0.62 (05:31:09)
IMEX SSP2(3,3,2)-LUM, MG-parCG	0.4	43.35 s	0.27 (01:11:03)	0.32 (02:51:59)
IMEX SSP2(3,3,2)-LSPUM, SCpCG	0.3	62.12 s	0.53 (02:20:45)	0.62 (05:34:23)
IMEX SSP2(3,3,2)-LSPUM, MG-parCG	0.3	62.16 s	0.20 (00:52:38)	0.33 (02:59:38)
IMEX SSP2(3,3,2)-LPUM, SCpCG	0.3	66.85 s	0.52 (02:17:12)	0.62 (05:30:26)
IMEX SSP2(3,3,2)-LPUM, MG-parCG	0.3	66.92 s	0.18 (00:46:36)	0.35 (03:06:25)
IMEX SSP2(3,3,2)-LPM, SCpCG	0.25	33.63 s	0.63 (02:46:28)	0.71 (06:22:29)
IMEX SSP2(3,3,2)-LPM, MG-parCG	0.25	33.71 s	0.34 (01:30:18)	0.48 (04:16:51)

Table 6.4: Simulation Series A: The simulations have been performed on the VSC 1 using 64 CPU-cores. The effectively measured wallclocktime is given in the format (hh:mm:ss).

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

Method	μ_{adv}	$\Delta t_{\text{mn}}, 80$ scrt	WCT_{norm} (WTC), 80 scrt	WCT_{norm} (WCT), 200 scrt
SSPRK(2,2), SCpCG	0.2	3.23 s	1 (06:10:40)	1 (15:37:07)
SSPRK(2,2), MG-parCG	0.2	3.23 s	1.12 (06:55:33)	1.03 (16:09:40)
SSPRK(3,2), SCpCG	0.5	8.06 s	0.60 (03:41:30)	0.62 (09:43:24)
SSPRK(3,2), MG-parCG	0.5	8.06 s	0.64 (03:57:39)	0.65 (10:08:32)
IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$, SCpCG	0.2	8.91 s	0.90 (05:32:17)	0.91 (14:11:54)
IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$, MG-parCG	0.2	8.91 s	0.70 (04:20:00)	0.73 (11:25:51)
IMEX SSP2(2,2,2), $\gamma = 0.24$, SCpCG	0.5	15.09 s	0.60 (03:41:39)	0.63 (09:48:37)
IMEX SSP2(2,2,2), $\gamma = 0.24$, MG-parCG	0.5	15.09 s	0.48 (02:57:18)	0.51 (08:01:38)
IMEX SSP2(3,3,2)-LUM, SCpCG	0.4	48.40 s	0.50 (03:04:32)	0.58 (09:07:15)
IMEX SSP2(3,3,2)-LUM, MG-parCG	0.4	48.34 s	0.20 (01:14:11)	0.32 (05:02:34)
IMEX SSP2(3,3,2)-LSPUM, SCpCG	0.3	75.70 s	0.46 (02:52:12)	0.62 (09:38:21)
IMEX SSP2(3,3,2)-LSPUM, MG-parCG	0.3	75.45 s	0.12 (00:46:09)	0.35 (05:28:07)
IMEX SSP2(3,3,2)-LPUM, SCpCG	0.3	86.46 s	0.42 (02:34:30)	0.61 (09:34:53)
IMEX SSP2(3,3,2)-LPUM, MG-parCG	0.3	86.40 s	0.11 (00:42:31)	0.32 (05:00:19)

Table 6.5: Simulation Series B: The simulations have been performed on the VSC 2 using 64 CPU-cores. The effectively measured wallclocktime is given in the format (hh:mm:ss).

Method	μ_{adv}	$\Delta t_{\text{mn}}, 200$ scrt	WCT_{norm} (WCT), 200 scrt
SSPRK(2,2), SCpCG	0.2	3.23 s	1 (14:42:39)
SSPRK(2,2), MG-parCG	0.2	3.23 s	0.96 (14:05:58)
SSPRK(3,2), SCpCG	0.5	8.06 s	0.65 (09:31:29)
SSPRK(3,2), MG-parCG	0.5	8.06 s	0.64 (09:27:16)
IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$, SCpCG	0.2	12.52 s	0.82 (12:04:13)
IMEX SSP2(2,2,2), $\gamma = 1 - 1/\sqrt{2}$, MG-parCG	0.2	12.53 s	0.60 (08:45:51)
IMEX SSP2(2,2,2), $\gamma = 0.24$, SCpCG	0.5	13.21 s	0.78 (11:31:09)
IMEX SSP2(2,2,2), $\gamma = 0.24$, MG-parCG	0.5	13.21 s	0.58 (08:36:07)
IMEX SSP2(3,3,2)-LUM, SCpCG	0.4	28.91 s	0.64 (09:22:22)
IMEX SSP2(3,3,2)-LUM, MG-parCG	0.4	28.91 s	0.60 (08:52:24)
IMEX SSP2(3,3,2)-LSPUM, SCpCG	0.3	14.97 s	0.94 (13:54:03)
IMEX SSP2(3,3,2)-LSPUM, MG-parCG	0.3	14.97 s	0.76 (11:06:44)
IMEX SSP2(3,3,2)-LPUM, SCpCG	0.2	28.06 s	0.66 (09:40:30)
IMEX SSP2(3,3,2)-LPUM, MG-parCG	0.2	28.06 s	0.57 (08:19:33)
IMEX SSP2(3,3,2)-LPM, SCpCG	0.3	24.36 s	0.70 (10:16:26)
IMEX SSP2(3,3,2)-LPM, MG-parCG	0.3	24.36 s	0.75 (10:58:27)

Table 6.6: Simulation Series C: The simulations have been performed on the VSC 2 using 64 CPU-cores. The effectively measured wallclocktime is given in the format (hh:mm:ss).

6. APPLICATION TO DOUBLE-DIFFUSIVE CONVECTION

SSP2(3,3,2)–LSPUM			SSP2(3,3,2)–LPUM		
Resolution	CFL_{mn}	CFL_{max}	Resolution	CFL_{mn}	CFL_{max}
229x228	1.42	3.7	229x228	1.44	3.27
429x428	2.7	5.8	429x428	3.0	6.81
829x828	3.7	6.0	829x828	4.2	7.45
1629x1628	3.82	6.3	1629x1628	4.86	7.72

Table 6.7: Resolution Test for the simulation of double-diffusive convection, setting B. Left: IMEX SSP2(3,3,2)–LSPUM method. Right: IMEX SSP2(3,3,2)–LPUM method .

Method	μ_{adv}	Δt_{mn}	WCT (200 scrt)
SSPRK(3,2), SCpCG	0.5	8.06 s	24:22:22
SSPRK(3,2), pMG	0.5	8.06 s	17:02:10
IMEX SSP2(2,2,2), $\gamma = 0.24$, SCpCG	0.3	17.61 s	23:20:07
IMEX SSP2(2,2,2), $\gamma = 0.24$, pMG	0.3	17.61 s	08:14:03

Table 6.8: Simulations of simulation series A carried out on 16 CPU-cores. The simulations have been performed on VSC1. The effectively measured wallclocktime is given in the format (hh:mm:ss).

Chapter 7

Application to simulations of Cepheids

7.1 Cepheids

Cepheids belong to the class of pulsating stars, which in turn comprise a subset of intrinsic variable stars. These are stars whose variability arises from causes entirely within themselves, and not from geometric effects such as eclipses in binary stars; or to some external agent such as interaction with the interstellar medium or with circumstellar matter. Pulsating stars are stars in which large-scale dynamical motions, usually including the entire star, and usually more or less rhythmic, are present. The simplest kind of such a motion is a purely radial pulsation, in which the star maintains a spherical shape at all times, but changes its volume, as if it were breathing. ([37, p. 9]).

The pulsation of a variable star depends sensitively on the presence and location of its ionization zones. The κ -mechanism, which is the main motor for pulsation, does not work for completely ionized gas, neither can stable pulsation develop if the ionization zones are situated too far inside the star. Details on the κ -mechanism can be found in [37] and references therein.

On the importance of cepheids in astronomy, [37] remarks: *For the astronomer the importance of classical cepheids lies in their use as distance indicators. They are the most important tool for establishing the basic distance scale of the universe.*

Since stars other than the sun are far less accessible to direct observation, the understanding of the processes within those relies heavily on models of stellar evolution and numerical simulations. Such simulations provide quite a challenge for both the numerical schemes employed and the computational resources available.

7.2 Numerical Simulations of Pulsating Stars

Within the work of [35, 37] the ANTARES code has been extended to perform simulations of pulsating stars. The most important enhancement in this context was the introduction of a radially stretched grid to cope with the vast differences in length scales between the stellar surface and the interior. Additionally, it was necessary to employ spherical coordinates since

7. APPLICATION TO SIMULATIONS OF CEPHEIDS

a major part of a star is modeled. Furthermore, the grid should follow the radial pulsation of the star. The entailed modifications to the Navier-Stokes equations and the numerical schemes employed are discussed in detail in [35, 37]. In the following, only the most important issues influencing the use of IMEX Runge-Kutta methods in simulations of cepheids are revised.

7.2.1 The model

The processes within a pulsating star are mathematically described by the equations of hydrodynamics introduced in Chapter 2 ([35, 37]). In the context of simulations of cepheids, the fluid is assumed to be homogeneous. Due to the employment of spherical coordinates, geometric factors enter the equations,

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \vec{u}_{\text{rel}}) \quad (7.2.1)$$

$$\frac{\partial(\rho \vec{u})}{\partial t} = -\nabla \cdot (\rho \vec{u}_{\text{rel}} \otimes \vec{u}) - \nabla P + \nabla \cdot \tau + \rho g + \frac{S}{r} \quad (7.2.2)$$

$$\frac{\partial(\rho e_{\text{tot}})}{\partial t} = -\nabla \cdot (\vec{u}_{\text{rel}}(\rho e_{\text{tot}}) + \vec{u}P) + \nabla \cdot (\vec{u} \cdot \tau) + \rho \vec{u}g + \nabla \cdot F_{\text{rad}} \quad (7.2.3)$$

where

$$S = \begin{pmatrix} 2P + \frac{(\rho \vec{u})_\phi^2 + (\rho \vec{u})_\theta^2}{\rho} \\ -(\rho \vec{u})_r(\rho \vec{u})_\phi + (\rho \vec{u})_\phi^2(\rho \vec{u})_\theta \cot(\theta) \\ -P \cot(\theta) - \frac{(\rho \vec{u})_r(\rho \vec{u})_\theta + (\rho \vec{u})_\theta^2 \cot(\theta)}{\rho} \end{pmatrix} \quad (7.2.4)$$

Furthermore, also in the viscous stress tensor τ geometric terms turn up. For its appearance in two dimensions, the reader is referred to [37].

The gravity is determined via $g = \left(\frac{M(r)G}{r^2}, 0, 0 \right)^T$ where $G = 6.673 \cdot 10^{-8} \text{ cm}^3 \text{ g}^{-1} \text{ s}^{-1}$ denotes the gravitational constant and $M(r)$ is the amount of mass of the star contained inside the radius r . Since the grid is not fixed in space, in general there is $\vec{u}_{\text{rel}} \neq \vec{u}$.

The timestep restrictions outlined in Section 4.6.1 apply analogously.

The initial condition considered here is one of a cepheid of effective temperature $T = 5125 \text{ K}$, luminosity $L = 912.797 L_\odot$, mass $M = 5 M_\odot$, hydrogen content $X = 0.7$ and metallicity $Z = 0.01$. (Note that $L_\odot$ and $M_\odot$ refer to the solar luminosity and the solar mass.) It has been provided by G. Houdek¹ and incorporated into the ANTARES code by [37].

Within the ANTARES code, the height of the computational domain for simulations of cepheids is conveniently specified by the temperature range. The simulations carried out in [36, 37] cover almost 42% of the entire star, ranging from $T=4\,000 \text{ K}$ at the top to $T=320\,000 \text{ K}$ at the bottom. A realistic microphysics model using the OPAL equation of state is used. To obtain an accurate value for the radiative heating rate Q_{rad} , in optically thin regions the equation of

¹University of Aarhus, Denmark

radiative transfer [37, 32] is solved, elsewhere the diffusion approximation of radiative transfer (see equation (2.3.7) in Chapter 2) is assumed. To start dynamics away from equilibrium, an initial perturbation is applied.

Boundary Conditions

In simulations of cepheids, the vertical boundary conditions depend strongly on their physical location within the star. In the present models, the lower vertical boundary is defined at $T=320\,000$ K, inside a zone where energy is purely transported by radiation and no convective motions take place. Thus, it is justified to prescribe the initial value of the total energy flux entering at the bottom of the simulation domain of the starting model to F_{rad} ,

$$F_{\text{rad } r_{\text{bot}}} = F_{\text{rad } r_{\text{bot}}}^{\text{start}}. \quad (7.2.5)$$

Furthermore, *slip* boundary conditions are imposed on the velocity, where the component normal to the boundary (in this case, the radial velocity) is set to

$$u_r(r_{\text{bot}}) = 0 \quad (7.2.6)$$

whereas the gradients of the tangential components of the velocity vector are set to zero.

The boundary conditions along the upper vertical boundary of the domain depend on the domain itself. If a model ranging from $T=34\,000$ K to $T=320\,000$ K is used and thus no part of the stellar atmosphere is modeled, the starting value for the radiative flux may be prescribed at r_{top} ,

$$F_{\text{rad}}(r_{\text{top}}) = F_{\text{rad}}(r_{\text{top}})^{\text{start}} \quad (7.2.7)$$

If the computational domain reaches up to $T = 4\,000$ K, a part of the atmosphere of the star is contained in the simulation domain. In this case,

$$\nabla \cdot F_{\text{rad}}(r_{\text{top}}) = 0 \quad (7.2.8)$$

provides a physically meaningful boundary condition. Furthermore, the radial velocity is set to the computed grid velocity and Neumann boundary conditions on the density are prescribed,

$$u_r(r_{\text{top}}) = u_g \quad (7.2.9)$$

$$\frac{\partial \rho}{\partial r} = 0 \quad (7.2.10)$$

In horizontal direction, periodic boundary conditions are employed.

Computational Challenges

Numerical simulations of pulsating stars are both numerically and computationally demanding ([6, 36]). First of all, model parameters such as luminosity and metallicity must be chosen such

7. APPLICATION TO SIMULATIONS OF CEPHEIDS

as to reduce numerical demands. Possible choices of parameters can be assessed by performing a series of tests as in [36]. Furthermore, in the hydrogen ionisation zone close to the surface of the cepheid, steep gradients are likely to occur in the conservative variables. Thus, a high resolution is necessary to obtain accurate results, multiplying the computational effort for the timesteps performed. The required high resolution in turn influences another challenge: small timesteps are enforced due to the short timescale of radiation processes. Although the usage of the radiative transfer equation permits the application of a less restrictive timestep than equation (4.6.2), the minimum timestep is still considerably smaller than the one set by fluid motions. By treating the radiative heating term implicitly, SSP IMEX RK-methods promise a significant gain in computational efficiency compared to explicit time integration methods.

7.2.2 The grid structure

The computational domain is equipped with a 3D spherical grid which is used for 1D and 2D simulations such that one cell covers an entire shell at a given radius ([36, 37]). The radial range covers $[r_{\text{bot}}, r_{\text{top}}]$ where r_{bot} is fixed and r_{top} varies with time. In radial direction, the grid is stretched by a factor q . The mesh sizes follow the relation $\Delta r_{i+1} = q\Delta r_i$, thus the numerical grid becomes

$$r_i = r_{\text{top}} - (q^i - 1) \frac{r_{\text{top}} - r_{\text{bot}}}{q^n - 1} \quad (7.2.11)$$

and additionally, in 2D, in azimuthal direction the grid covers $\phi \in \left[-\frac{\phi_{\text{tot}}}{2}, \frac{\phi_{\text{tot}}}{2}\right]$,

$$\phi_j = -\frac{\phi_{\text{tot}}}{2} + j \Delta\phi \quad (7.2.12)$$

Since spherical, 3D coordinates are used, in a 2D simulation sectors in the equatorial plane ($\theta = 90^\circ$) are considered. In 1D, the simulation boxes are radial columns. The grid is depicted in Figure 5.2.

The moving grid

Each timestep n , the grid velocity u_{g}^{n+1} at r_{top} is determined as the horizontal average of the fluid velocity at r_{top} . At the bottom the grid is fixed, i.e. $u_{\text{g}}(r_{\text{bot}}) = 0$. In the next timestep, only the new position of outermost radial point r_{top} is computed by

$$r_{\text{top}}^{n+1} = r_{\text{top}}^n + \Delta t u_{\text{g}}^n. \quad (7.2.13)$$

All the other points r_i^{n+1} are computed to satisfy equation (7.2.11).

Although the model equations (7.2.1) -(7.2.3) are given in differential form, it is crucial to remember that this appearance was obtained assuming the integration volume V was arbitrarily chosen and did not change with time. In the moving grid approach, however, the latter assumption does not hold. It is therefore necessary to reflect these changes in the discretization. To this

end the integral formulation of the governing equations (2.1.2), (2.2.5) and (2.3.2) of Chapter 2 is invoked. In case of the continuity equation, the integral formulation reads

$$\int_{V(t)} \frac{\partial \rho}{\partial t} dV = - \int_{V(t)} \nabla \cdot (\rho \vec{u}_{\text{rel}}) dV \quad (7.2.14)$$

Discretizing the above expression by the Euler-forward method and the integrals by the midpoint rule, one arrives at

$$\rho^{n+1} V^{n+1} = \rho^n V^n - \Delta t \nabla \cdot (\rho \vec{u}_{\text{rel}})^n V^n \quad (7.2.15)$$

Similarly, if the Euler-backward scheme is used, equation (7.2.15) reads

$$\rho^{n+1} V^{n+1} = \rho^n V^n - \Delta t \nabla \cdot (\rho \vec{u}_{\text{rel}})^{n+1} V^{n+1} \quad (7.2.16)$$

Division by V^{n+1} leads to

$$\rho^{n+1} = \rho^n \frac{V^n}{V^{n+1}} - \Delta t \nabla \cdot (\rho \vec{u}_{\text{rel}})^n \frac{V^n}{V^{n+1}} \quad (7.2.17)$$

and

$$\rho^{n+1} = \rho^n \frac{V^n}{V^{n+1}} - \Delta t \nabla \cdot (\rho \vec{u}_{\text{rel}})^{n+1}, \quad (7.2.18)$$

respectively.

Thus, the time dependency of cell volumes is taken into account. Note that this approach applies to (IMEX-)Runge-Kutta schemes for which each Runge-Kutta substep consists of an Euler-forward/ Euler-backward step. This holds for all Runge-Kutta schemes considered in this thesis. The momentum and energy equations are discretized analogously.

7.2.3 SSP IMEX RK-Methods for the Cepheid Model

The implicit stage equation

In order to use IMEX RK-methods for integrating the model equations describing the fluid flow in pulsating stars, the equations are split such that the Q_{rad} term is considered the “stiff” term,

$$\underbrace{\frac{\partial}{\partial t} \begin{pmatrix} \rho \\ \rho \vec{u}_{\text{rel}} \\ \rho e_{\text{tot}} \end{pmatrix}}_{y(t)} = \nabla \cdot \underbrace{\begin{pmatrix} \rho \vec{u}_{\text{rel}} \\ \rho \vec{u}_{\text{rel}} \otimes \vec{u}_{\text{rel}} + P - \tau \\ \rho e_{\text{tot}} \vec{u}_{\text{rel}} + P \vec{u}_{\text{rel}} - \vec{u} \cdot \tau \end{pmatrix}}_{F(y(t))} + \begin{pmatrix} 0 \\ g + \frac{S}{\rho} \\ g \vec{u}_{\text{rel}} \end{pmatrix} - \underbrace{\begin{pmatrix} 0 \\ 0 \\ Q_{\text{rad}} \end{pmatrix}}_{G(y(t))}. \quad (7.2.19)$$

Thus, one implicit equation of appearance

$$y_i = y^* + \Delta t \tilde{a}_{ii} Q_{\text{rad}} \quad (7.2.20)$$

has to be solved each Runge-Kutta stage.

7. APPLICATION TO SIMULATIONS OF CEPHEIDS

To obtain an equation resembling a generalized Poisson equation, the diffusion approximation of radiative transfer (see equation (2.3.7) in Chapter 2) is used to model Q_{rad} . Note that this assumption is justified if an interior part of the cepheid is modeled. However, as soon as a part of the stellar atmosphere is considered, the equation of radiative transfer must be coupled to the model equations.

Hence, assuming the diffusion approximation equation (7.2.20) reads

$$-\nabla \cdot (K_{\text{rad}} \nabla T_i) + \frac{e_{\text{int } i}}{\Delta t \tilde{a}_{ii}} = \frac{e_{\text{int}}^*}{\Delta t \tilde{a}_{ii}}. \quad (7.2.21)$$

Since a realistic microphysics model is used, the radiative conductivity K_{rad} as well as the internal energy e_{int} depend nonlinearly on the temperature T .

The procedure to solve a nonlinear generalized Poisson equation employing a multigrid-Newton technique is outlined in Chapter 5. To assemble the Jacobian matrix operator, the derivatives $(\partial K_{\text{rad}}/\partial T)_\rho$ and $(\partial e_{\text{int}}/\partial T)_\rho$ have to be determined. To this end, the EOS routine comprised in the ANTARES code provides the values $(\partial T/\partial \epsilon)_\rho$ where $\epsilon = e_{\text{int}}/\rho$ (ϵ denotes the specific internal energy) as well as $(\partial \log(\kappa)/\partial \log(T))_\rho$ (κ denotes the thermal diffusion coefficient). Thus, the desired derivatives are computed as

$$(\partial e_{\text{int}}/\partial T)_\rho = \rho (\partial T/\partial \epsilon)_\rho^{-1} \quad (7.2.22)$$

$$(\partial K_{\text{rad}}/\partial T)_\rho = \frac{16\sigma_{\text{SB}}}{3\rho} \left(3T^2 \frac{1}{\kappa} - \frac{T^2}{\kappa} \left(\frac{\partial \log(\kappa)}{\partial T} \right)_\rho \right) \quad (7.2.23)$$

where $\sigma_{\text{SB}} = 5.67 \cdot 10^{-8} \frac{\text{W}}{\text{m}^2 \text{K}^4}$ denotes the *Stefan-Boltzmann constant*.

IMEX Methods within the Moving Grid Framework - the Algorithm

The time dependency of the grid must be taken into account when implementing IMEX Runge-Kutta schemes for the cepheid model. The grid movement is described by the differential equation

$$\frac{\partial r}{\partial t} = u_g. \quad (7.2.24)$$

The right hand side of the above expression enters the explicitly integrated term $F(y(t))$ of the governing equations. Thus, following the IMEX procedure used in simulations of double-diffusive convection (see section 6.2.4), the implicit stage is performed before the “new” grid is computed. The grid is updated subsequently and then the explicit stage is computed.

A close look at the procedure in section 6.2.4 reveals that the explicit update in equation (6.2.15) is always performed as an Euler forward step. The true timelevel given by the coefficients b_i in the coefficient matrix of the corresponding IMEX scheme is obtained by multiplication in the step gathering the results of the explicit and the implicit sweep, equation (6.2.17). Within the moving grid approach, however, the explicit flux function is no longer autonomous but must be updated to the time level indicated by the explicit Runge-Kutta subscheme of the IMEX method. To this

end, the Shu-Osher form of the explicit Runge-Kutta subscheme (see section 4.2) is invoked. The following algorithm illustrates the update process of the conservative variables from timelevel n to timelevel $n + 1$ within the moving grid approach. It applies to the two-stage second order method IMEX SSP2(2,2,2), however, the generalization to three- and more stage schemes is straightforward.

The explicit subscheme of the IMEX SSP2(2,2,2) method, namely, the explicit SSP(2,2) method, reads in Shu-Osher form

$$\begin{aligned} y_0 &= y_{\text{old}} \\ y_1 &= y_0 + \Delta t F(y_0) \\ y_2 &= \frac{1}{2}(y_0 + y_1 + \Delta t F(y_1)) \\ y_{\text{new}} &= y_2 \end{aligned}$$

(i) Stage 1:

At the beginning of stage 1, the value at timelevel n , $y_{\text{old}}^{(1)} = y^n$ as well as the (radial) grid velocity u_{g}^n is available.

(ii) The implicit substage of the first stage is evaluated via

$$y_{\text{imp}}^{(1)} = y_{\text{old}}^{(1)} + \gamma \Delta t G(y_{\text{imp}}^{(1)}) \quad (7.2.25)$$

(iii) Then, the grid is updated via

$$r_{\text{top}}^{n+1/2} = r_{\text{top}}^n + \Delta t u_{\text{g}}^n \quad (7.2.26)$$

and the r_i are computed according to prescription (7.2.11).

(iv) The explicit stage is performed on the grid at timelevel $n + 1/2$,

$$y_{\text{exp}}^{(1)} = y_{\text{old}}^{(1)} \frac{V^n}{V^{n+1/2}} + \Delta t F(y_{\text{imp}}^{(1)}) \quad (7.2.27)$$

(v) The new grid velocity $u_{\text{g}}^{n+1/2}$ is computed from $y_{\text{exp}}^{(1)}$.

(vi) For the final assembly of y^{n+1} at the new timelevel, an auxiliary array y_{sum}^{n+1} is used,

$$y_{\text{sum}}^{n+1} = y_{\text{old}}^{(1)} \frac{V^n}{V^{n+1/2}} + \frac{1}{2} \left(y_{\text{exp}}^{(1)} - y_{\text{old}}^{(1)} \frac{V^n}{V^{n+1/2}} \right) + \frac{1}{2\gamma} \left(y_{\text{imp}}^{(1)} - y_{\text{old}}^{(1)} \right) \frac{V^n}{V^{n+1/2}} \quad (7.2.28)$$

Note that this expression corresponds to $y_{\text{sum}}^{n+1} = 1/2(y_0 + y_1)$ in the Shu-Osher formulation of the explicit subscheme.

7. APPLICATION TO SIMULATIONS OF CEPHEIDS

(vii) Stage 2:

An auxiliary array for evaluation of the implicit stage is assembled,

$$y_{\text{old}}^2 = y_{\text{old}}^{(1)} \frac{V^n}{V^{n+1/2}} + \frac{1}{2} \left(y_{\text{exp}}^{(1)} - y_{\text{old}}^{(1)} \frac{V^n}{V^{n+1/2}} \right) + \frac{1-2\gamma}{\gamma} \left(y_{\text{imp}}^{(1)} - y_{\text{old}}^{(1)} \right) \frac{V^n}{V^{n+1/2}} \quad (7.2.29)$$

(viii) The implicit stage equation is solved,

$$y_{\text{imp}}^{(2)} = y_{\text{old}}^{(2)} + \gamma \Delta t G(y_{\text{imp}}^{(2)}) \quad (7.2.30)$$

(ix) The grid is updated via

$$r_{\text{top}}^{n+1} = r_{\text{top}}^{n+1/2} + \Delta t u_g^{n+1/2} \quad (7.2.31)$$

and the r_i are computed according to prescription (7.2.11).

(x) The explicit stage is performed on the new grid,

$$y_{\text{exp}}^{(2)} = y_{\text{sum}}^{n+1} \frac{V^{n+1/2}}{V^{n+1}} + \frac{\Delta t}{2} F(y_{\text{imp}}^{(2)}) \quad (7.2.32)$$

(xi) The new grid velocity u_g^{n+1} is computed from $y_{\text{exp}}^{(2)}$.

(xii) Due to the Shu-Osher formulation, the vector $y_{\text{exp}}^{(2)}$ already contains the density and momentum values at the new timelevel $n+1$, $y_{\text{exp}}^{(2)} = (\rho^{n+1}, (\rho \vec{u})^{n+1}, (et)^*)$. Only the energy value must be corrected due to the implicit treatment of the radiative heating rate,

$$(et)^{n+1} = (et)^* + \frac{1}{2\gamma} \left(y_{\text{imp}}^{(2)} - y_{\text{old}}^{(2)} \right) \frac{V^{n+1/2}}{V^{n+1}} \quad (7.2.33)$$

(xiii) Then, y^{n+1} is obtained via

$$y^{n+1} = y_{\text{exp}}^{(2)} = (\rho^{n+1}, (\rho \vec{u})^{n+1}, (et)^{n+1}) \quad (7.2.34)$$

Timestep Control

For simulations of cepheids, the timestep control based on the occurrence of two-point instabilities as used for simulations of double-diffusive convection has proven to be too restrictive. This is easily understood since in the double-diffusive convection setting, the fluid moves slowly compared to the speed of sound whereas in cepheids, the fluid reaches supersonic velocities which in turn introduce shock waves. Thus, patterns matching the criteria of two-point instabilities (see section 6.2.4) might be indicated as a two-point instability but naturally arise in that case from time to time due to the more turbulent nature of the flow.

Monitoring the rate of change of the implicitly obtained variable — in the present case, the temperature — provides too little control over the timestep. In simulations of cepheids, the

most effective timestep criterion has turned out to be the observation of the number of Newton-iterations. If more than 10 Newton-iterations are necessary to obtain an acceptable solution, in the following timestep, a temporal increment of $\frac{2}{3}\Delta t$ is used². To permit the system to readjust itself, no modifications on the Δt are made for the subsequent 15 timesteps. If less than 10 Newton-iterations are performed for 50 successive timesteps, the timestep Δt is augmented by a factor of $\frac{5}{4}$.

Boundary conditions

The boundary conditions applied to the implicit stage equation should be compatible with those of the explicit stage. The boundary conditions outlined above prescribe conditions on the radiative flux F_{rad} , which, since the diffusion approximation is employed, translate directly into conditions on ∇T ,

$$F_{\text{rad}} = K_{\text{rad}}\nabla T \Leftrightarrow \nabla T = \frac{F_{\text{rad}}}{K_{\text{rad}}}. \quad (7.2.35)$$

However, if condition (7.2.35) is prescribed, the temperature increases considerably at the upper boundary of the simulation domain. This increase cannot be accounted for by physical reasons, but indicates a violation of the conservation laws the governing equations are derived from. However, if Dirichlet boundary conditions prescribing fixed values for the top and bottom temperature are used in the implicit equation, no considerable unphysical increase of temperature is observed (see Figure 7.1). Thus, all the simulations presented in this chapter have been performed with Dirichlet boundary conditions imposed on the implicit temperature equation.

7.2.4 Multigrid Setting

For the one-dimensional simulations of cepheids, the one-dimensional counterpart of the nonlinear Multigrid-Newton solver presented in Chapter 5 is used. Spherical coordinates are employed, the relaxation method of choice is the lexicographic GS-method. Since the positive definiteness of the Jacobian matrix cannot be assured, the coarse grid equation is solved using the GS-scheme, which has been found to render fast and accurate results in this context. In every Newton-iteration, $\lambda_{\text{CS}} = 2$ multigrid sweeps are performed. The multigrid-Newton iteration is stopped as soon as the maximum value of the correction v (see section 5.6.2, equation (5.6.19)) inferred to the already obtained solution is less than 10^{-4} .

²In view of the test conducted in section 5.6.4 the limit of 10 Newton iterations might seem very restrictive, especially, if a low λ_{CS} is used. However, the reader has to bear in mind that the coefficients of the implicit temperature equation differs from the ones used in Example 5.6.3. The temperature is expected to vary only minimally from one timestep to the next and since the initial guess the solver is fed with is already quite close to the solution, convergence is achieved rapidly. In the context of simulations of cepheids, generally about 5 Newton iterations are performed until a satisfying solution to the implicit temperature equation is found. Thus, decreasing the timestep if twice as many iterations have been necessary to solve the implicit stage equation is a reasonable criterion.

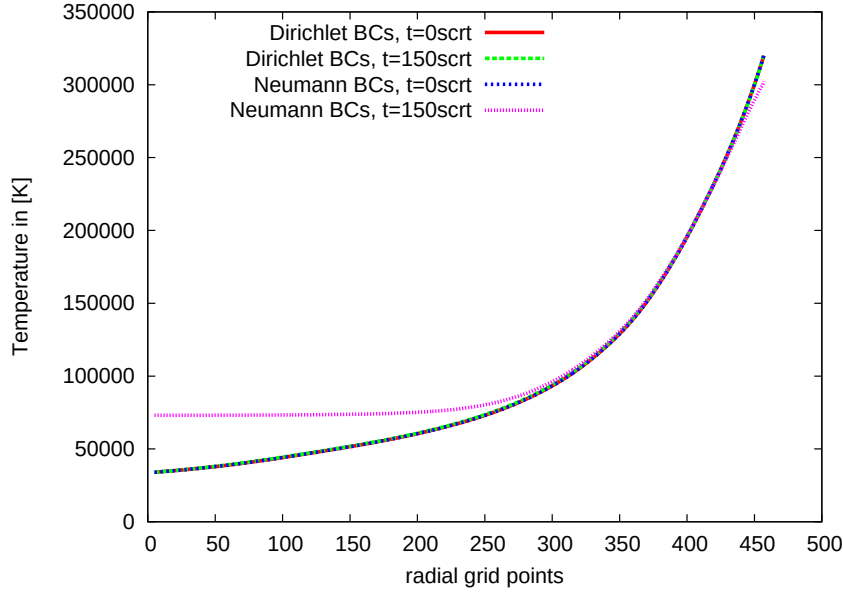


Figure 7.1: The initial temperature stratification is compared to the solution after 150 srt in the 1D deep cepheid model (see section 7.3.1) using Dirichlet and Neumann boundary conditions in the implicit temperature equation.

The modified trapezoidal rule

Within the FEM framework, the underlying geometry of the computational domain enters the integral formulation of the variational problem. If other than Cartesian ones are employed, the corresponding transformation must be reflected in the infinitesimal volume element $dV = dx dy dz$. In case of spherical coordinates, there is $dV = r^2 \sin \theta dr d\phi d\theta$, where expression $r^2 \sin \theta$ is the determinant of the corresponding transformation operator. Note that in the present context, essentially one- and two-dimensional simulation domains situated in the equatorial plane ($\theta = 90^\circ$) are considered, thus the transformed volume element reduces to $dV^{1D} = r^2 dr$ and $dV^{2D} = r^2 dr d\phi$, respectively.

The integral expressions of the variational formulation are discretized using the trapezoidal rule (see Appendix B.1). Since the ansatz functions employed are linear, in the Cartesian framework the appearing integrals are summations over linear and constant functions. It is easy to see that in this case, the trapezoidal rule does not provide approximative but exact values of the integrals. In case of spherical coordinates, however, the functional determinant r^2 enters the integrals. Therefore, the integrand is of second or third order in r . In this case, the numerical evaluation of the integral using the trapezoidal rule already introduces a certain error. Thus, the idea is to use weighting factors w_1 and w_2 in the trapezoidal rule such that functions of second and third order are integrated exactly.

It is the aim to find $w_1, w_2 \in \mathbb{R}$ in the trapezoidal rule

$$\int_{r_a}^{r_b} f(r) dr = w_1 f(r_a) + w_2 f(r_b) \quad (7.2.36)$$

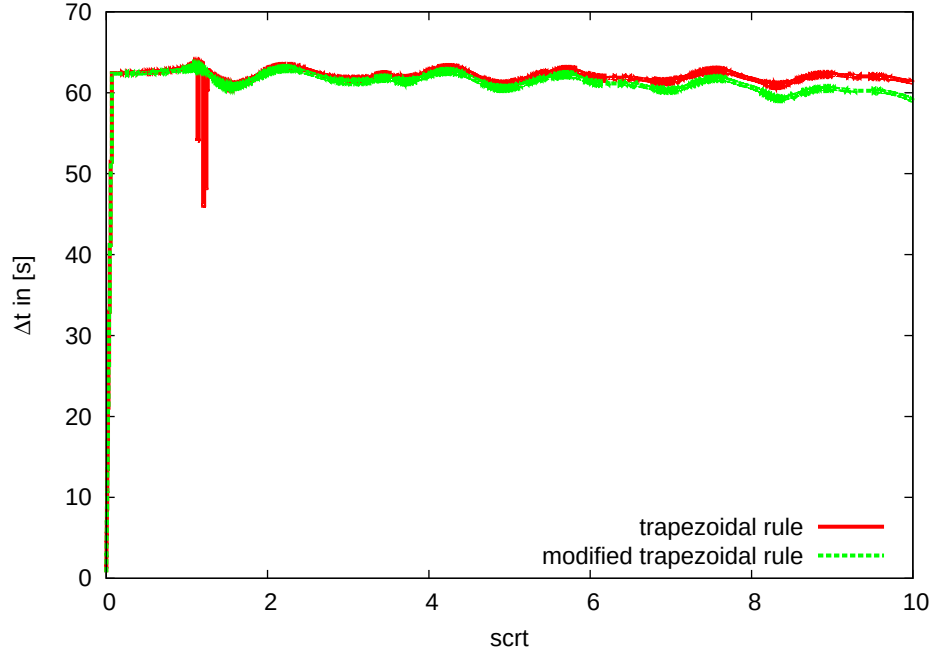


Figure 7.2: Timestep evolution in the 1D deep cepheid model using the original and the modified trapezoidal rule.

such that

$$\frac{r_b^3}{3} - \frac{r_a^3}{3} = \int_{r_a}^{r_b} r^2 dr = w_1 r_a^2 + w_2 r_b^2 \quad (7.2.37)$$

and

$$\frac{r_b^4}{4} - \frac{r_a^4}{4} = \int_{r_a}^{r_b} r^3 dr = w_1 r_a^3 + w_2 r_b^3. \quad (7.2.38)$$

Solving the arising linear system for w_1 and w_2 , one obtains

$$w_1 = (r_b - r_a) \frac{3r_a^2 + 2r_a r_b + r_b^2}{12r_a^2} \quad (7.2.39)$$

$$w_2 = (r_b - r_a) \frac{r_a^2 + 2r_a r_b + 3r_b^2}{12r_b^2} \quad (7.2.40)$$

The effect of the modified trapezoidal rule has been determined by running the 1D deep cepheid model (see section 7.3.1) using the classical trapezoidal rule and the modified trapezoidal rule. Figure 7.2 depicts the timestep evolution in both cases. It seems that whereas in the simulation performed with the classical trapezoidal rule, certain stability issues diminish the timestep after about 1.5 sqrt, the modified trapezoidal rule conveys more stability to the simulation so that also in this critical phase the time integration is performed using the maximum possible timestep. Therefore, the modified trapezoidal rule is employed in the simulation of pulsation stars presented below.

7.3 Results

7.3.1 1D-Cepheid Models

Modeling radial stellar pulsations has a long tradition in the astrophysical community. However, only recently the development of parallel supercomputers has provided the means to conduct simulations involving a respectable amount of spatial grid points and still some of the simulations envisaged are barely feasible (see [35]). Thus, first attempts of capturing the phenomenon of stellar pulsations were of a more theoretical nature and early simulations were naturally confined to one spatial dimension. Nevertheless, it has been found that one-dimensional models of stellar pulsation using accurate microphysics and opacities portray the overall mechanical aspects of stellar pulsation quite well and predict periods in good agreement with observations. A short history of modeling approaches and one-dimensional simulations of pulsating stars is provided by [37].

Furthermore, one-dimensional simulations are indispensable for setting up two-dimensional start models for simulations of pulsating stars (see [35, 37]).

In the following, the advantages of IMEX Runge-Kutta methods in one-dimensional simulations of cepheids are discussed. The simulation domain is aligned with the radial direction and covers a temperature range of 34 000 K to 320 000 K. In this region, the diffusion approximation provides a valid estimation of the radiative heating rate Q_{rad} . This model is also referred to as the *deep cepheid model* within this thesis.

The computational domain is discretized using 453 grid points and a stretching factor of $q = 1.007$. Simulation time is 10 scrt (in this setting, 1 scrt = 188 317 s). All simulations have been run on a single CPU core.

Efficiency of the SSP IMEX RK methods

Simulations of cepheids in one spatial dimension have been performed using different time integrators. Table 7.1 gives the Courant numbers and mean timesteps employed during the simulations as well as the wallclocktimes.

Note that depending on the time integration scheme used, different Courant numbers are important. The timestep used with the explicit SSPRK(2,2) and SSPRK(3,2) schemes is limited by radiative diffusion processes and thus, μ_{diff} is decisive for the actual value of Δt . The maximum admissible diffusive Courant numbers of the different time integrators have been determined in Chapter 4 (Table 4.1) by the means of stability analysis. However, it is not advisable to integrate at the stability limit which is the reason that the actual diffusive Courant number is obtained from $\mu_{\text{diff}}^{\text{act}} = s \cdot \mu_{\text{diff}}^{\text{max}}$ where $s \in (0, 1]$ denotes a security factor.

Using IMEX Runge-Kutta methods, the radiative heating rate Q_{rad} is integrated implicitly and thus, the timestep restriction due to radiative diffusion processes is very much alleviated or, in case the implicit subscheme is unconditionally stable, even completely removed. The second-shortest timescale in the setting considered is set by acoustic wave propagation. Thus, τ_{snd}

provides an upper limit on the timestep used. In the determination of τ_{adv} , the advective Courant number μ_{snd} is of importance.

Since the maximum admissible Courant number of the explicit SSPRK(3,2) scheme is much greater than the one of its two-stage second order counterpart SSPRK(2,2), the mean timestep employed in the simulation with SSPRK(3,2) is about 3.6 times higher than the one employed with SSPRK(2,2). However, it has to be pointed out that in the simulation with SSPRK(3,2), $\mu_{\text{diff}}^{\text{act}} = \mu_{\text{diff}}^{\text{max}}$ is chosen. Thus, this simulation is performed on the absolute stability limit. Due to the fact that the SSPRK(3,2) scheme entails three Runge-Kutta stages per timestep, the gain in the timestep translates only into a runtime about 2.6 times faster.

Method	μ_{adv}	$\mu_{\text{diff}}^{\text{act}}$	Δt_{mn}	WCT _{norm} (WTC)
SSPRK(2,2)	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	0.62 s	1 (06:55:07)
SSPRK(3,2)	0.5	$1.0 \cdot \mu_{\text{diff}}^{\text{max}}$	2.22 s	0.38 (02:40:37)
IMEX SSP2(2,2,2) $\gamma = 0.24$	0.5	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	26.56 s	0.07 (00:30:13)
IMEX SSP2(2,2,2) $\gamma = 1 - 1/\sqrt{2}$	0.5	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	60.34 s	0.03 (00:13:21)
IMEX SSP2(3,3,2)-LUM	-	-	-	-
IMEX SSP2(3,3,2)-LSPUM	0.5	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	60.73 s	0.04 (00:17:51)
IMEX SSP2(3,3,2)-LPUM	0.5	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	60.73 s	0.04 (00:17:00)
IMEX SSP2(3,3,2)-LPM	0.5	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	60.73 s	0.04 (00:17:40)

Table 7.1: Numerical results for 1D deep cepheid model. The simulations have been performed on the VSC1 using one single CPU-core.

The use of IMEX Runge-Kutta methods provides an immense increase in computational efficiency. All of the optimized IMEX Runge-Kutta methods, namely the IMEX SSP2(3,3,2)-LSPUM, the IMEX SSP2(3,3,2)-LPUM and the IMEX SSP2(3,3,2)-LPM scheme are able to stably employ the maximum timestep dictated by τ_{snd} (see Figure 7.3). Thus, the timestep employed by the optimized IMEX methods is almost 98 times higher than the one employed by the SSPRK(3,2) scheme and about 27.18 times larger than the timestep used with SSPRK(3,2). The wallclocktimes given in Table 7.1 show that the higher timestep translates into a runtime about 24.42 times faster than using the SSPRK(2,2) scheme and 9.4 times faster than obtained using the SSPRK(3,2) scheme. From the timestep difference and the efficiency gain of the three-stage IMEX methods compared to the SSPRK(3,2) scheme it can be deduced that the overhead of the multigrid solver is roughly a factor of 3 to 4. Note that fully implicit methods such as those used by [6] in a similar application bring about an overhead of a factor of 10 at least.

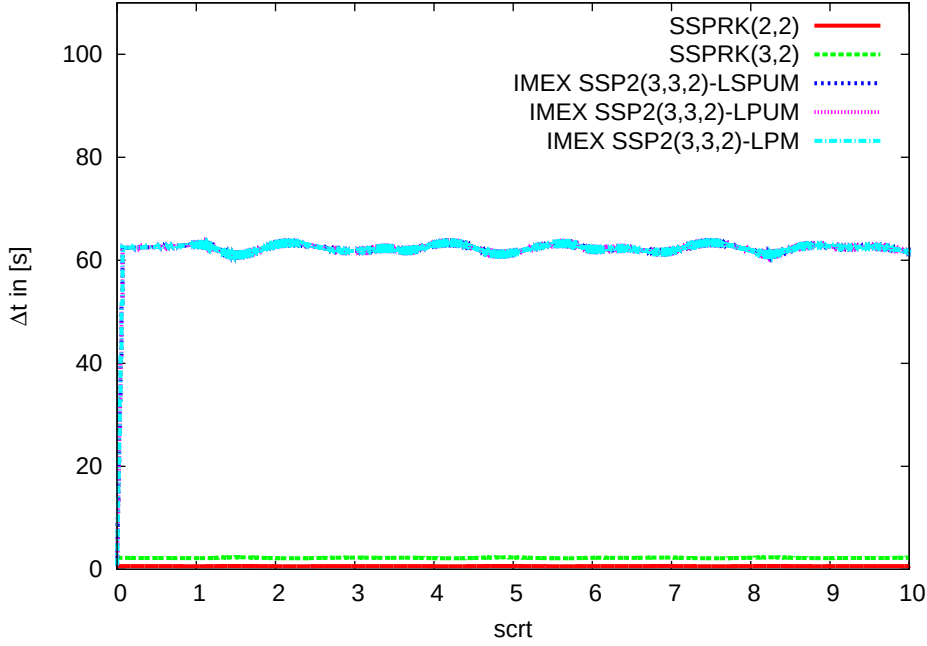


Figure 7.3: The timestep evolution with different time integrators in the 1D deep cepheid simulation.

The three-stage second order IMEX scheme from the literature, IMEX SSP2(3,3,2)-LUM proves to be too unstable to perform a simulation in the considered setting. The simulation breaks down at the first transition of the contraction of the star into an expansion. Therefore, no data regarding its numerical performance are given. Thus, it seems that the property “P” (positivity of the amplification factor) has a positive influence on the stability of the scheme.

The most efficient scheme in the present setting is the two-stage second order IMEX SSP2(2,2,2) scheme with $\gamma = 1 - 1/\sqrt{2}$. Since this scheme is able to employ the limit timestep set by τ_{snd} but entails the computation of only two stages per timestep, it is about 1.27 times faster than the best performing optimized IMEX Runge-Kutta scheme.

Figure 7.4 shows snapshots of the 1D deep cepheid simulation. The pulsation of the star is clearly seen in the evolution of the vertical momentum. At $t = 1$ scrt, the positive x-momentum indicates a contraction of the star (bear in mind that the positive x-direction is aligned with the direction of gravity within the ANTARES framework), whereas at $t = 2$ scrt, the expansion is at its maximum. One entire contraction-expansion cycle takes about 2 scrt. Both the simulations performed with SSPRK(3,2) and IMEX SSP2(3,3,2)-LPUM agree very well.³

³Note that the IMEX SSP2(3,3,2)-LPUM method is used representatively for all SSP MEX schemes considered in this thesis, i.e. the results of the other IMEX schemes tested agree similarly well.

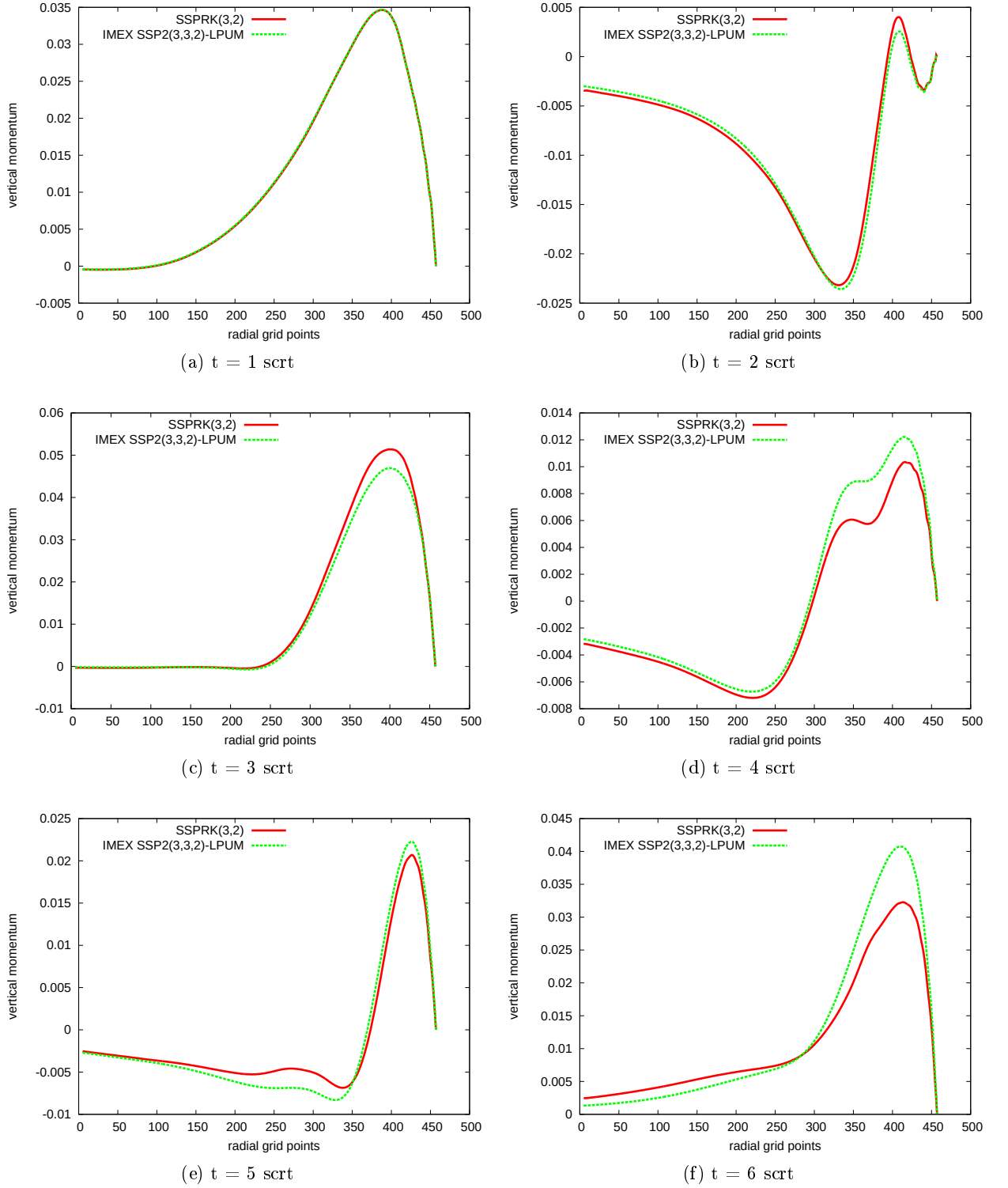


Figure 7.4: Snapshots of the 1D cepheid simulation performed with SSPRK(3,2) and IMEX SSP2(3,3,2)-LPUM.

Sensitivity of the IMEX SSP2(2,2,2) scheme on the parameter γ

The sensitivity of the IMEX SSP2(2,2,2) method on the parameter γ has been tested. Contrarily to the results in the double-diffusive convection model, where the use of $\gamma = 0.24$ has proven to be beneficial on the stability of the simulation, in the 1D deep cepheid model the IMEX SSP2(2,2,2) schemes employing a $\gamma \geq 0.2475$ perform significantly better. The mean timestep of the IMEX SSP2(2,2,2) scheme with $\gamma = 0.26$ is about 2.3 times higher than the one employed by the IMEX SSP2(2,2,2) scheme with $\gamma = 0.24$ and the higher timestep directly translates into a runtime 2.32 times smaller.

Figure 7.5 depicts the timestep evolution in the 1D deep cepheid model using the IMEX SSP2(2,2,2) method with different values for the parameter γ . It is clearly seen that the mean timestep increases as γ increases toward 0.25. An explanation for this behavior is easily found; it has been outlined in Chapter 4 that the implicit subschemes of the IMEX SSP2(2,2,) method with $\gamma < 0.25$ are not unconditionally stable. As depicted in Chapter 4, Figure 4.2, the stability region grows as γ increases, enabling thus a stable integration at larger timesteps. It may be deduced from the results in Table 7.2 that the stability region of the IMEX SSP2(2,2,2) scheme with $\gamma = 0.2475$, although still bounded, is already large enough to permit a mean timestep as large as the upper limit on Δt set by τ_{snd} .

Method	μ_{adv}	μ_{diff}	Δt_{mn}	WCT _{norm} (WTC)
IMEX SSP2(2,2,2) $\gamma = 0.24$	0.5	0.5	26.56 s	1 (00:30:13)
IMEX SSP2(2,2,2) $\gamma = 0.2425$	0.5	0.5	33.27 s	0.79 (00:23:59)
IMEX SSP2(2,2,2) $\gamma = 0.245$	0.5	0.5	44.11 s	0.61 (00:18:24)
IMEX SSP2(2,2,2) $\gamma = 0.2475$	0.5	0.5	60.57 s	0.45 (00:13:40)
IMEX SSP2(2,2,2) $\gamma = 0.25$	0.5	0.5	60.57 s	0.43 (00:13:05)
IMEX SSP2(2,2,2) $\gamma = 0.26$	0.5	0.5	60.61 s	0.43 (00:13:01)
IMEX SSP2(2,2,2) $\gamma = 1 - 1/\sqrt{2}$	0.5	0.5	60.34 s	0.44 (00:13:21)

Table 7.2: Sensitivity of the time integrator IMEX SSP2(2,2,2) on the parameter γ , tested on the one-dimensional deep cepheid model. The simulations have been performed on the VSC1 using a single CPU-core.

The advantage of the IMEX SSP2(2,2,2) methods with $\gamma < 0.25$ is the positivity of the amplification factor g , a property, which is not shared by the IMEX SSP2(2,2,2) schemes with $\gamma \geq 0.25$. In this case, however, no notable influence of the property “P” is observed.

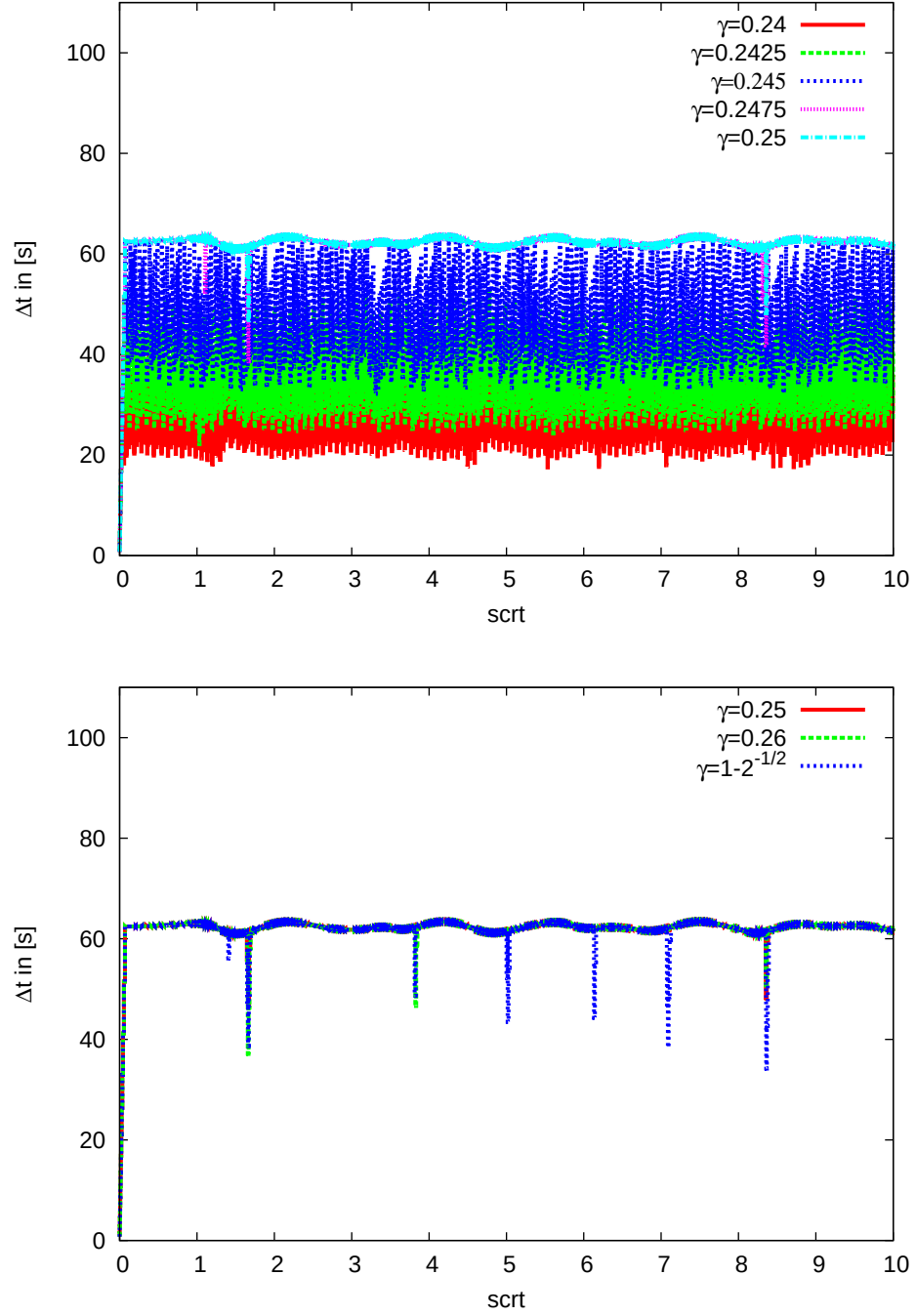


Figure 7.5: The timestep evolution in the simulations performed with IMEX SSP2(2,2,2) with varying γ is depicted.

Short Summary

- The most efficient SSP IMEX schemes in terms of computational time in one-dimensional simulations of pulsating stars is the IMEX SSP2(2,2,2) $\gamma = 1 - 1/\sqrt{2}$ scheme.
- All of the optimized SSP IMEX methods, i.e. the IMEX SSP2(3,3,2)-LSPUM, the IMEX SSP2(3,3,2)-LPUM and the IMEX SSP2(3,3,2)-LPM methods stably employ the limit timestep set by τ_{snd} throughout the entire simulation.
- The IMEX SSP2(3,3,2) scheme is not able to perform a stable time integration in the present setting. This might be due to the lack of the property “P” – positivity of the amplification factor.
- Neither the property “U” – uniform convergence – nor the property “S” – inclusion of a part of the imaginary axis in the stability region of the explicit subscheme – have a notable influence on the stability of the time integration.

Chapter 8

Outlook

8.1 Application of SSP IMEX RK-methods to 2D Cepheid Models

8.1.1 Simulation Setting

Due to the promising results in one dimension, it is aspired to use IMEX methods also in two-dimensional simulations of cepheids. However, the step from one to two spatial dimensions is a difficult one. Whereas in 1D, the only possible motions are oscillations, in 2D the build-up of convective rolls is possible and expected.

First experiments have been conducted using a 2D deep cepheid model. The initial condition is based on the one-dimensional cepheid model outlined in section 7.3.1, which has been advanced 150 scrt to obtain stable pulsation. The starting model for the 2D simulation is then constructed by homogeneously extending the 1D profile in horizontal direction. To start dynamics away from equilibrium, an initial perturbation is applied.

The 2D cepheid model is computed using a radially stretched, spherical grid with a resolution of 275×500 grid points. A stretching factor of $q = 1.11$ is employed. The two-dimensional model has already been advanced by E. Mundprecht¹ for approximately 13.82 scrt using the explicit SSPRK(2,2) scheme. Thus, the total runtime of the model considered amounts to 163.82 scrt. To test the use of IMEX RK-methods, the existing simulation has been continued for 2.2 scrt (in this setting, 1 scrt = 193 443 s). This simulation time corresponds approximately to one entire pulsation period.

The two-dimensional multigrid-Newton solver operating on a spherical, radially stretched grid is used to obtain a solution to the implicit temperature equation arising in the implicit stage of the IMEX methods. The grid resolution permits a multigrid hierarchy of 3 grids. The Zebra x-line Gauss-Seidel iteration is employed both as the smoothing routine and the coarse grid solver. For each Newton-iteration, $\lambda_{CS} = 1$ CS-iterations are applied to the linearized system. The multigrid-Newton iteration is stopped as soon as the maximum value of the correction v (see

¹E. Mundprecht, Institute for Mathematics, University of Vienna

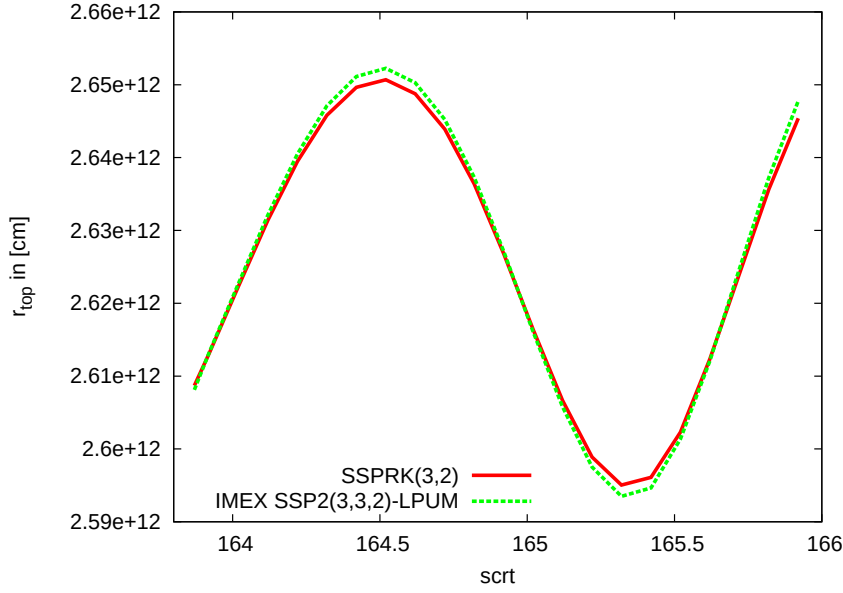


Figure 8.1: The evolution of the maximum radius r_{top} is depicted.

section 5.6.2, equation (5.6.19)), inferred to the already obtained solution is less than 10^{-6} . The timestep control as outlined in section 7.2.3 is used.

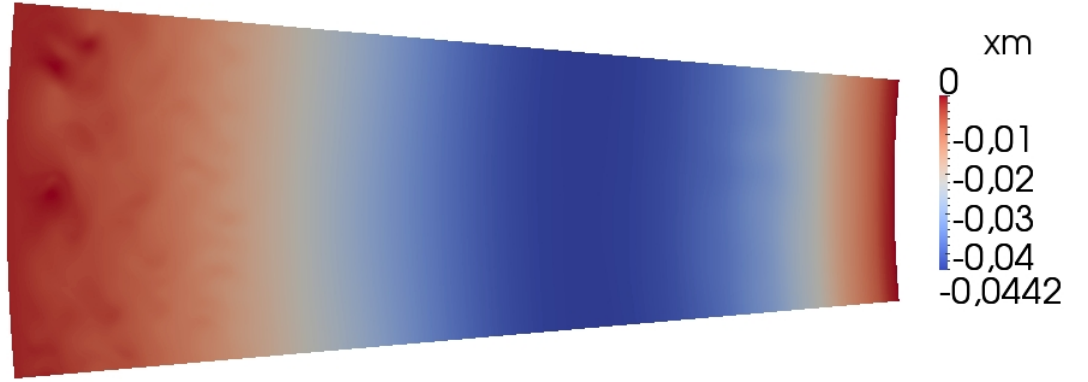
8.1.2 First results

Figure 8.1 shows that the SSP IMEX RK methods capture the pulsation of the star very well although the pulsation amplitude predicted by the IMEX SSP2(3,3,2)-LPUM method is slightly higher than the one calculated with SSPRK(3,2). This indicates that employing large timesteps, the numerical damping of the time integration scheme is less pronounced.

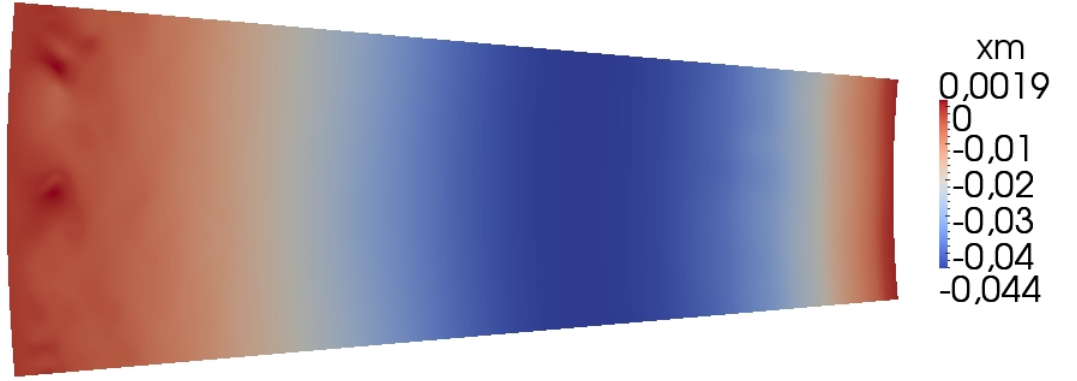
Figures 8.2, 8.3, and 8.4 present snapshots of the 2D cepheid simulations performed with SSPRK(3,2) and IMEX SSP2(3,3,2)-LPUM at $t=164.32$ scrt, $t=164.82$ scrt and $t=165.32$ scrt.² The vertical momentum density (in the Figures 8.2, 8.3, and 8.4 the vertical momentum density is denoted as x_m) is depicted. It is important to bear in mind that within the ANTARES-framework, the x-direction points along the direction of gravity. Thus, the negative momentum in Figure 8.2 and 8.4 denotes an expansive phase of the star whereas the positive momentum in Figure 8.3 indicates a contractive state.

A close look at Figure 8.2 reveals that almost no differences between the explicitly calculated solution and the semi-implicit one are discernible. Locations of strong movements (indicated by deeply red regions) form at similar positions, a tendency, which is continued in Figure 8.3. In the latter, however, it is already apparent that the developing features are much more pronounced in the simulation calculated with IMEX SSP2(3,3,2)-LPUM. Nevertheless, the corresponding legends assure that the magnitude of the x-momentum is still very similar.

²It is important to note that in the above discussion the IMEX SSP2(3,3,2) method is used representatively for all SSP IMEX schemes considered in this thesis, i.e. the discussed features appear analogously in all tested SSP IMEX schemes.



(a) SSPRK(3,2)



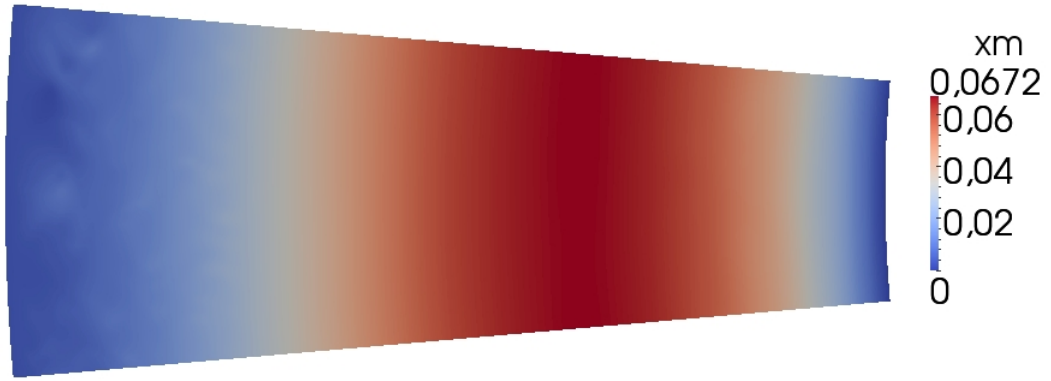
(b) IMEX SSP2(3,3,2)-LPUM

Figure 8.2: Snapshots of the 2D deep cepheid model at $t=164.32$ scrt, computed with SSPRK(3,2) and IMEX SSP2(3,3,2)-LPUM (representatively for all SSP IMEX schemes considered in this thesis). xm denotes the vertical momentum density. The simulations have been performed on VSC1 with 8 CPU-cores.

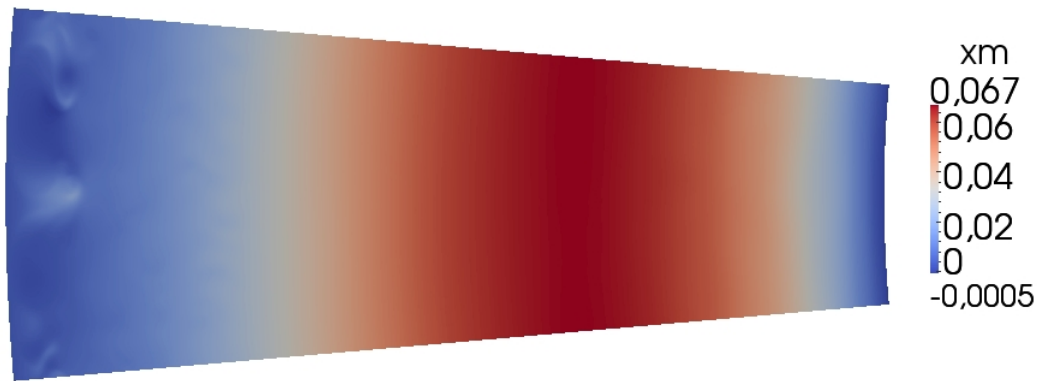
In Figure 8.4, however, it is obvious that the solutions have significantly drifted apart. A strong downdraft has developed in the semi-implicitly calculated solution, where the magnitude of the x -momentum is about 7 times higher than the one in the comparative, explicitly calculated solution. Although it is not possible to quantitatively verify the results of the explicit simulation, it is likely that the features appearing in the simulation calculated with IMEX SSP2(3,3,2)-LPUM stem from numerical issues. One possible reason is insufficient temporal coupling of the boundary conditions applied to the implicit and the explicit stages, an issue, which is to be pursued in the future.

Table 8.1 gives the wallclocktimes and mean timesteps obtained in the first 0.5 scrt of the simulations performed. The timestep evolution is depicted in Figure 8.1. Note that a very conservative advective Courant number $\mu_{adv} = 0.15$ has been used in the simulations.

As in the one-dimensional counterpart of these simulations, the best performing schemes are

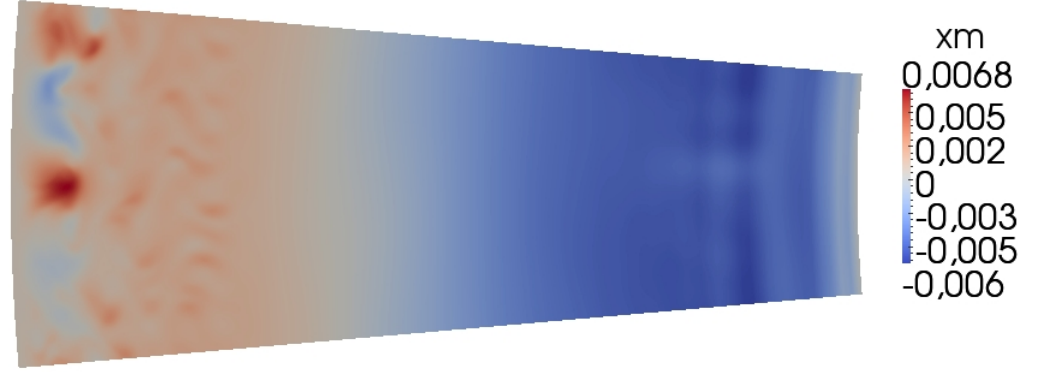


(a) SSPRK(3,2)



(b) IMEX SSP2(3,3,2)-LPUM

Figure 8.3: Snapshots of the 2D deep cepheid model at $t=164.82$ scrt, computed with SSPRK(3,2) and IMEX SSP2(3,3,2)-LPUM (representatively for all SSP IMEX schemes considered in this thesis). xm denotes the vertical momentum density. The simulations have been performed on VSC1 with 8 CPU-cores.



(a) SSPRK(3,2)



(b) IMEX SSP2(3,3,2)-LPUM

Figure 8.4: Snapshots of the 2D deep cepheid model at $t=165.32$ scrt, computed with SSPRK(3,2) and IMEX SSP2(3,3,2)-LPUM (representatively for all SSP IMEX schemes considered in this thesis). x_m denotes the vertical momentum density. The simulations have been performed on VSC1 with 8 CPU-cores.

8. OUTLOOK

the optimized IMEX SSP2(3,3,2)-LSPUM, -LPUM, and -LPM schemes as well as the two-stage IMEX SSP2(2,2,2), $\gamma = 0.24$ method. All of those schemes employ the timestep limit set by τ_{snd} throughout the entire simulation. It is interesting to note that the IMEX SSP2(2,2,2) scheme with $\gamma = 1 - 1/\sqrt{2}$ performs slightly worse than the IMEX SSP2(2,2,2) with $\gamma = 0.24$. This fact might be attributed to the lack of the property “P”, the positivity of the amplification factor. It has already been observed in section 7.3.1 that this property is capable of introducing a certain amount of stability to simulations of pulsating stars.

The drawback of the IMEX SSP2(2,2,2) schemes with $\gamma < 0.25$ is their bounded stability region already discussed in 7.3.1. However, in the present case, a rather conservative advective Courant number ($\mu_{\text{adv}} = 0.15$) is used. This affects the value of τ_{snd} and thus the timestep limit. Therefore, contrarily to the one-dimensional setting, the maximum achievable timestep lies within the stability region of the IMEX SSP2(2,2,2) $\gamma = 0.24$ scheme and can consequently be reached in the simulation.

It is interesting to note that the IMEX SSP2(3,3,2)-LUM method experiences severe stability problems which are indicated by a considerably lower mean timestep Δt_{mn} than the other SSP IMEX methods tested. Bearing in mind that for the one-dimensional deep cepheid model the IMEX SSP2(3,3,2)-LUM method broke down entirely (see section 7.3.1), it seems that the IMEX SSP2(3,3,2)-LUM scheme is for some reason especially unsuitable for the use within simulations of pulsating stars. Thus, the findings of the performance of the different time integration schemes observed in the one-dimensional simulations of cepheids (see section 7.3.1) carry over to the two-dimensional case. Nevertheless, due to the boundary issue, the results of Table 8.1 have to be interpreted with care.

Anyhow, the results in Table 8.1 definitely indicate that IMEX Runge-Kutta methods have the potential of significantly speeding up the computation of 2D cepheid models. It can be deduced from the results in Table 8.1 that the overhead of the nonlinear Multigrid-Newton solver amounts to approximately a factor of 3. This is even lower than the overhead in the one-dimensional simulations of cepheids (factor 4, c.f. section 7.3.1). However, the gain in timestep in the two-dimensional simulations is also much lower than in the one-dimensional simulations. Thus, it can be claimed that the Multigrid-Newton solver introduces almost the same overhead in one- and two-dimensional simulations of pulsating stars. This is an impressive demonstration of the superior convergence properties of the multigrid technique and the efficiency of the method even in a parallel computing environment.

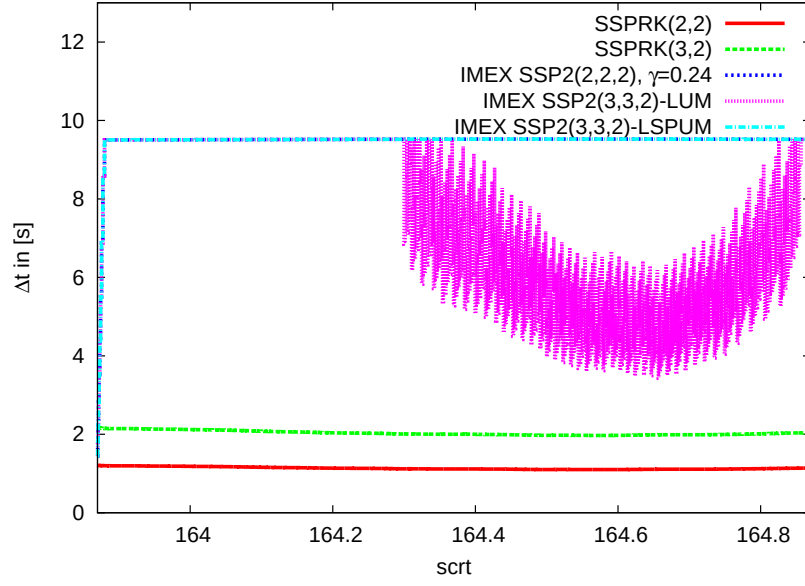


Figure 8.5: The timestep evolution in the 2D deep cepheid simulation is depicted. Due to the conservatively chosen value of the advective Courant number $\mu_{\text{adv}} = 0.15$, the timestep limitation due to the sound speed is small enough to effectively limit the timestep of the semi-implicit time integration methods.

Method	μ_{adv}	$\mu_{\text{diff}}^{\text{act}}$	Δt_{mn}	WCT _{norm} (WTC)
SSPRK(2,2)	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	1.12 s	1 (23:03:11)
SSPRK(3,2)	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	2.03 s	0.83 (19:10:51)
IMEX SSP2(2,2,2) $\gamma = 0.24$	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	9.40 s	0.35 (08:30:04)
IMEX SSP2(2,2,2) $\gamma = 1 - 1/\sqrt{2}$	0.5	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	8.91 s	0.40 (09:08:11)
IMEX SSP2(3,3,2)-LUM	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	6.46 s	(18:30:11)
IMEX SSP2(3,3,2)-LSPUM	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	9.40 s	0.54 (12:31:34)
IMEX SSP2(3,3,2)-LPUM	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	9.40 s	0.55 (12:36:41)
IMEX SSP2(3,3,2)-LPM	0.15	$0.5 \cdot \mu_{\text{diff}}^{\text{max}}$	9.40 s	0.55 (12:37:48)

Table 8.1: Numerical Results for 2D deep cepheid model. The simulations have been performed on the VSC1 on 8 CPU-cores.

8.2 The Eddington Approximation

In astrophysical hydrodynamics, radiative heat exchange plays an important role in determining the evolution of fluid motion. However, solving the general equation describing radiative transfer ([32, 39]) involves the solution of integro-differential equations which is a complicated and computationally costly venture. Thus, various simplifications of the radiative transfer equation have been introduced. Within this thesis, for instance, the diffusion approximation for radiative transfer (see equation (2.3.7) in Chapter 2) has been used to obtain an approximate value for the radiative heating rate Q_{rad} . The diffusion approximation, however, is only valid when the relevant scales of motion are optically thick ([32, 49]). This is a major limitation in simulations of astrophysical fluid flow.

The *Eddington approximation* provides a more general estimation of the radiative heating rate. It has been discussed in [49] that the Eddington approximation describes the optically thick limit practically perfectly and provides a reasonable accurate estimation of the optically thin limit. It is therefore suitable for the modeling of radiative processes also in stellar atmospheres.

Within the Eddington approach, the radiative heating rate Q_{rad} introduced in Chapter 2 is expressed by

$$Q_{\text{rad}} = \nabla \cdot \left(\frac{4\pi}{3\chi\rho} \nabla J \right) \quad (8.2.1)$$

where J is the mean intensity, χ denotes the opacity and ρ is the density.

The main advantage equation of (8.2.1) is that is of the same appearance as the diffusion approximation, where the radiative heating rate reads $Q_{\text{rad}}^{\text{DA}} = -\nabla \cdot (K_{\text{rad}} \nabla T)$ (see equation (2.3.7)). Having a look at the derivation of the implicit stage equation (c.f. equations (6.2.13), (6.2.14) and equations (7.2.20), (7.2.21)), it becomes clear that the divergence form of $Q_{\text{rad}}^{\text{DA}}$ is decisive for obtaining a generalized Poisson equation as an implicit stage equation. Since the radiative heating rate obtained using the Eddington approximation, equation (8.2.1), is also of divergence form it will similarly lead to a generalized Poisson equation as the implicit stage equation. Thus, the solver developed within this thesis can be used without modifications. In context of the Eddington approach, though, the arising nonlinear generalized Poisson equation is to be solved for the mean intensity J . The mean intensity J as well as the opacity χ , however, depend on the temperature T which is not yet determined in the implicit stage. Thus, an iterative algorithm must be devised which solves the nonlinear generalized Poisson equation for J and then uses the energy equation to obtain T for the computed approximative value for J .

The incorporation of the Eddington approximation would greatly enhance the range of applicability of the SSP IMEX Runge-Kutta methods within the ANTARES framework. It is hoped that simulations of cepheids including part of the atmosphere as well as simulation of A-type stars, which today are barely feasible due to the small timesteps enforced by explicit time integrations methods, are substantially increased in efficiency by the use of IMEX Runge-Kutta methods.

Chapter 9

Conclusions

It has been the objective of this work to alleviate unnecessarily small timestep restrictions enforced by traditional explicit time integration schemes by the use of strong stability preserving (SSP) implicit-explicit (IMEX) Runge-Kutta methods. Those methods have been incorporated into the ANTARES code and the efficiency of the schemes has been evaluated in simulations of double-diffusive convection and simulations of pulsating stars. Both applications differ considerably from each other and thus allow the determination of properties which are important for a stable time integration in very different settings.

To obtain a fast and accurate solution of the implicit stage equations, a (non-) linear solver based on the multigrid technique has been developed. It is shown in section 5.12 that this solver is capable of providing an accurate solution to the linear generalized Poisson equation much faster than the solver of [11] which is based on the Schur Complement parallelized preconditioned Conjugate Gradient algorithm, especially if the mesh is quite fine. Indeed, whereas at a resolution of 109×112 grid points, the difference in runtime amounts to a factor of 1.96, at a grid resolution of about 509×512 the multigrid solver provides an accurate solution about 5.5 times faster than the solver of [11].

Since the ANTARES code is highly scalable and easily runs on a few thousand CPU-cores, a considerable effort has been put into an efficient parallelization of the multigrid solver. However, the multigrid algorithm itself inherits a certain sequentiality which is the reason why in the literature (see e.g. [23]) it is reported that weak scaling is the best scaling behavior a multigrid solver may achieve. Within this work, different parallelization strategies have been tried for the coarse grid equation to determine the parallelization concept rendering the most efficient parallel multigrid setup. Contrarily, the Schur Complement parallelized preconditioned Conjugate Gradient (SCpCG) solver developed by H. Grimm-Strele [11] features strong scaling (c.f. Table 5.17 or [16]). Nevertheless, the scaling test in Table 5.17 shows that although the scaling behavior of the multigrid solver is far from optimal, its superior convergence properties render it much more efficient in terms of CPU time than the SCpCG-solver, even if a considerable number of CPU-cores is used. Thus, it is concluded that the property of strong scaling does not always indicate the most efficient method in terms of CPU-time.

9. CONCLUSIONS

Simulations of semiconvection in different parameter regimes have been run with the SSP IMEX Runge-Kutta methods in a highly parallel computing environment. Both the performance of the linear solvers as well as the performance of the semi-implicit time integration schemes compared to the explicit ones is discussed in section 6.3.

It becomes clear that the full potential of the SSP IMEX Runge-Kutta methods in the setting of double-diffusive convection is revealed in simulation series A and B (see Tables 6.4 and 6.5). The best performing scheme is the IMEX SSP2(3,3,2)-LPUM scheme which achieves timesteps about 26.7 times (compared to SSPRK(2,2)) and 10.7 times (compared to SSPRK(3,2)) higher than the explicit schemes (c.f. Table 6.5). Furthermore, the results show that SSP IMEX schemes possessing the property of uniform convergence convey much more stability to the simulation than the SSP IMEX methods lacking this feature. The comparison of the simulations run with the IMEX SSP2(2,2,2) schemes permit the conclusion that in the setting of double-diffusive convection, the property of a positive amplification factor (IMEX SSP2(2,2,2) with $\gamma = 0.24$) is more important than L-stability (IMEX SSP2(2,2,2) with $\gamma = 1 - 1/\sqrt{2}$) as far as stability and efficient computation is concerned (c.f. Tables 6.4, 6.5 and 6.6).

The comparison of the MG-parCG and the SCpCG-solvers in simulations of double-diffusive convection is an interesting one. It turns out that the usage of the MG-parCG solver in the simulations performed with the explicit time integrators SSPRK(2,2) and SSPRK(3,2) does not result in an efficiency gain, but contrarily, using the SCpCG solver in that setting is actually the more efficient choice in terms of computational time. The reason for this behavior is that due to the small timesteps employed, the solution of the pressure-correction equation (which is the only linear generalized Poisson equation to be solved if explicit time integration methods are employed in conjunction with the fractional step method of [16]) requires only a few iterations (see Figure 6.12). In this case, the strong-scaling properties of the SCpCG-solver come to pass and it delivers the solution of the pressure-correction equation slightly faster than the MG-parCG solver is able to due to its considerable communication overhead.

As soon as the SSP IMEX methods are employed and higher timesteps are achieved, this picture changes. The higher the timestep, the more the variables (note that for the fully compressible hydrodynamical equations for a two-component fluid IMEX methods require the solution of three linear generalized Poisson equations in each Runge-Kutta stage) change and more iterations are necessary to obtain a sufficiently accurate solution. Using the SCpCG solver, this is reflected heavily in the computational time. Employing the MG-parCG solver, however, the superior convergence properties of the multigrid iteration pay off and only a few additional iterations are necessary to obtain solutions to the implicit equations. The speedup of simulations of double-diffusive convection due to the use of the MG-parCG solver amounts up to a factor of 3.8 (c.f. simulations with IMEX SSP2(3,3,2)-LSPUM in the diffusive regime in simulation setting B, Table 6.5). It can therefore be claimed that an efficient solver such as the one based on the multigrid technique is indispensable to translate the higher timesteps achieved by the SSP IMEX methods into a real efficiency gain in terms of CPU-time.

Furthermore, SSP IMEX Runge-Kutta methods have been incorporated into the model of pul-

sating stars. To correctly capture the radial pulsation of the star, the existing moving grid algorithm has been adapted to the IMEX approach. Since this model features realistic micro-physics modeling, the implicit stage consists of a nonlinear generalized Poisson equation. This equation is tackled by a Multigrid-Newton technique.

Since in simulations of pulsating stars a major part of the star is modeled, spherical coordinates are employed. With the discretization of the multigrid solver, a modified trapezoidal rule is used which is shown to convey additional stability to the simulation (see Figure 7.2).

A severe challenge in this context is the correct implementation of boundary conditions for both the explicit and the implicit stage. It has been shown that only Dirichlet boundary conditions are capable of correctly retaining the temperature stratification in the star (see Figure 7.1).

In one-dimensional simulations of cepheids, the SSP IMEX methods surpass traditional explicit time-integration limitations by far. In this setting, all the optimized SSP IMEX methods perform similarly well, achieving the limit timestep set by τ_{snd} . This is about 98 times higher than the timestep achieved by SSPRK(2,2) and 27 times higher than the one of SSPRK(3,2), almost throughout the entire simulation (see Table 7.1 and Figure 7.3). However, the IMEX SSP2(3,3,2)-LUM scheme from the literature fails entirely to render stable results in this setting, indicating thus that the property “P” – positivity of the amplification factor – infers a certain amount of stability to the integration. The most efficient scheme in terms of CPU-time, though, is the IMEX SSP2(2,2,2)-scheme with $\gamma = 1 - 1/\sqrt{2}$. The mean timestep achieved by this time integrator is almost as high as the one achieved by the optimized SSP IMEX schemes and since only 2 Runge-Kutta stages (instead of 3) are performed each timestep, the computational time it takes to perform the simulation is about 31 times lower compared to SSPRK(2,2), 12 times lower compared to SSPRK(3,2) and even 1.3 times lower than the computational time taken by the optimized SSP IMEX methods (c.f. Table 7.1). The performance of the different time integrators in one-dimensional simulations of cepheids permit the conclusion that the property of positivity of the amplification factor may introduce a certain degree of stability to the simulation (compare the performance of the IMEX SSP2(3,3,2)-LPUM and the IMEX SSP2(3,3,2)-LUM scheme in Table 7.1). It is interesting to note, though, that in this setting this property “P” is not as decisive for the success of the time integration as it is in simulations of double-diffusive convection, where the IMEX SSP2(2,2,2) method with $\gamma = 1 - 1/\sqrt{2}$ achieves significantly smaller timesteps than the IMEX SSP2(2,2,2) scheme with $\gamma = 0.24$ (c.f. the Table 7.2 and Tables 6.4, 6.5, 6.6).

The overhead of solving the nonlinear equation within the implicit stage in one-dimensional simulations of pulsating stars amounts only to a factor of 4 (c.f. Table 7.1).

First attempts to use IMEX Runge-Kutta methods in two-dimensional simulations of cepheids are reported. It is shown in Figure 8.1 that the pulsation of the star is very well captured by the adapted moving grid algorithm. Furthermore, the superior efficiency of the IMEX methods is still given (c.f. Table 8.1). Timesteps about 8.4 (compared to SSPRK(2,2)) and 4.3 (compared to SSPRK(3,2)) times higher are achieved, resulting in a runtime 2.7 (compared to SSPRK(2,2)) and 2.3 (compared to SSPRK(3,2)) lower. The behavior of the different SSP IMEX methods is similar to the one observed in one-dimensional simulations of pulsating stars. It is important

9. CONCLUSIONS

to note that in this setting the efficiency of the SSP IMEX methods is limited by the use of a very conservative advective Courant number μ_{adv} , which affects the maximum achievable timestep. Furthermore, the results in Table 8.1 permit the conclusion that the overhead of the two-dimensional MG-Newton solver amounts approximately to a factor of 3.

In the future, the incorporation of the Eddington approximation (see section 8.2) would greatly enhance the range of application of IMEX methods within the ANTARES code. Simulations of A-type stars, for example, where a realistic microphysics model is used, are until today barely feasible due to the severe timestep restriction posed by radiation processes. Furthermore, the extension of the multigrid solver to three spatial dimensions would allow more realistic, three dimensional simulations in all the applications discussed before.

Appendix A

A.1 Gauss's Divergence Theorem

Theorem 1.1.1 (Gauss' Divergence Theorem). *Be W a region in space with boundary ∂W and F a continuously differentiable vector field. Then the volume integral of the divergence $\nabla \cdot F$ of F over W and the surface integral of F over the boundary ∂W of W are related by*

$$\int_W \nabla \cdot F dV = \int_{\partial W} F \cdot \vec{n} dS \quad (\text{A.1.1})$$

where $\vec{n}$ denotes the outward pointing unit normal field of the boundary ∂W .

Proof. A proof can be found in [30]. □

A.2 Comoving regions

Be $W(0)$ a subregion of the computational domain at time $t_0 = 0$ (see [1]). As time elapses, the particles in W travel. Their displacement is described by the function

$$\mathbf{x} \rightarrow \xi(\mathbf{x}, t)$$

Consequently, the region W at time t is defined as:

$$W(t) = \{\xi(\mathbf{x}, t) \mid \mathbf{x} \in W(0)\} =: \xi(W, t) \quad (\text{A.2.1})$$

Be $V(t)$ the volume of $W(t)$. It is obvious that $V(t)$ can be described as

$$V(t) = \int_{W(t)} d\xi$$

Integration by substitution yields

$$V(t) = \int_{W(0)} J(x, t) dx \quad (\text{A.2.2})$$

where $J(x, t) = \det \frac{\partial \xi_i}{\partial x_j}$ is the corresponding functional determinant.

Lemma 1.2.1. *The functional determinant $J(x, t)$ satisfies the following property:*

$$\frac{\partial}{\partial t} J(x, t) = J(x, t) (\nabla \cdot \vec{u}) \quad (\text{A.2.3})$$

Proof. The particle at the initial position $\mathbf{x}$ travel along the trajectory

$$\frac{\partial}{\partial t} \xi(\mathbf{x}, t) = \vec{u}(\xi(\mathbf{x}, t), t) \quad (\text{A.2.4})$$

The definition of the determinant gives

$$J(\mathbf{x}, t) = \det \frac{\partial \xi_i}{\partial x_j} = \quad (\text{A.2.5})$$

$$\sum_{\sigma \in \mathfrak{S}_n} \text{sign}(\sigma) \partial_{\sigma(1)} \xi_1 \cdots \partial_{\sigma(n)} \xi_n \quad (\text{A.2.6})$$

The sign of the permutation σ is defined as $\text{sign}(\sigma) = (-1)^r$, where r is the number of transpositions the permutation is composed of.

$\mathfrak{S}_n$ denotes the symmetric group of degree n .

The temporal derivative of $J(\mathbf{x}, t)$ now reads:

$$\frac{\partial}{\partial t} J(\mathbf{x}, t) = \sum_{i=1}^n \underbrace{\sum_{\sigma \in \mathfrak{S}_n} \text{sign}(\sigma) \partial_{\sigma(1)} \xi_1 \cdots \partial_t \partial_{\sigma(i)} \xi_i \cdots \partial_{\sigma(n)} \xi_n}_{K_i(\mathbf{x}, t)} \quad (\text{A.2.7})$$

By definition of the determinant it is easily seen that

$$K_i(\mathbf{x}, t) = \det \begin{pmatrix} \partial_1 \xi_1 & \cdots & \partial_t \partial_1 \xi_i & \cdots & \partial_1 \xi_n \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \partial_n \xi_1 & \vdots & \partial_t \partial_n \xi_i & \cdots & \partial_n \xi_n \end{pmatrix} \quad (\text{A.2.8})$$

Using equation (A.2.4) and the symmetry of the second derivatives, there is

$$\partial_t \partial_j \xi_j = \partial_j \partial_t \xi_i = \partial_j u_i(\xi(\mathbf{x}, t), t) = \sum_{k=0}^n \partial_k u_i \partial_j \xi_k \quad (\text{A.2.9})$$

Inserting this expression into (A.2.8)

$$K_i(\mathbf{x}, t) = \sum_{k=0}^n \partial_k u_i \det \begin{pmatrix} \partial_1 \xi_1 & \cdots & \partial_1 \xi_k & \cdots & \partial_1 \xi_n \\ \vdots & & \vdots & & \vdots \\ \partial_n \xi_1 & \vdots & \partial_n \xi_k & \cdots & \partial_n \xi_n \end{pmatrix} \quad (\text{A.2.10})$$

Contemplating the sum on the right hand side of equation (A.2.10), it is obvious that if the indices k and i differ, there are two equal rows in the matrix $K_i(\mathbf{x}, t)$. Consequently, the determinant

becomes 0, so equation (A.2.10) reduces to

$$\sum_{k=0}^n \partial_k u_i \det \begin{pmatrix} \partial_1 \xi_1 & \cdots & \partial_1 \xi_k & \cdots & \partial_1 \xi_n \\ \vdots & & \vdots & & \vdots \\ \partial_n \xi_1 & \vdots & \partial_n \xi_k & \cdots & \partial_n \xi_n \end{pmatrix} = \partial_i u_i J(\mathbf{x}, t) \quad (\text{A.2.11})$$

Combining equations (A.2.7) and (A.2.11) leads to

$$\partial_t J(\mathbf{x}, t) = \nabla \vec{u} \cdot J(\mathbf{x}, t). \quad (\text{A.2.12})$$

□

Theorem 1.2.2. Transport Theorem

Let W be an arbitrary region in space, $\rho(\mathbf{x}, t)$ the mass density as defined in Chapter 2, equation (2.0.8), and $f : W \times (0, \infty) \rightarrow \mathbb{R}$ denotes a continuously differentiable function. Then

$$\frac{d}{dt} \int_{W(t)} \rho(\xi, t) f(\xi, t) d\xi = \int_{W(t)} \rho \frac{Df}{Dt}(\xi, t) d\xi$$

holds.

Proof. Transformation onto the initial region $W(0)$ using integration by substitution leads to

$$\begin{aligned} & \frac{d}{dt} \int_{W(t)} \rho(\xi, t) f(\xi, t) d\xi = \\ & \frac{d}{dt} \int_{W(0)} \rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t) J(\mathbf{x}, t) d\mathbf{x} \end{aligned}$$

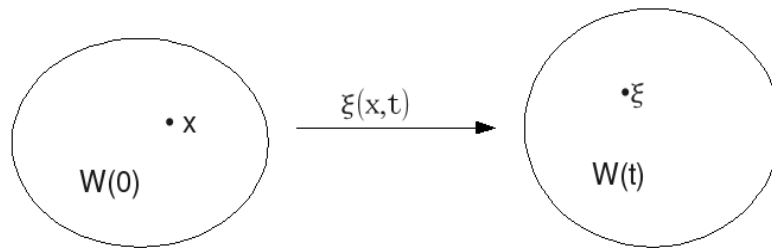


Figure A.1: Transformation of $W(0)$ to the comoving region $W(t)$

Since ρ and f are assumed to be continuous, it is justified to move the differential operator into the integral:

$$\begin{aligned}
& \frac{d}{dt} \int_{W(0)} \rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t) J(\mathbf{x}, t) dx = \\
& \int_{W(0)} \frac{d}{dt} (\rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t) J(\mathbf{x}, t)) dx = \\
& \int_{W(0)} \frac{d}{dt} (\rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t)) J(\mathbf{x}, t) + \rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t) \nabla \vec{u} J(\mathbf{x}, t) dx = \\
& \int_{W(0)} \left(\frac{d}{dt} (\rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t)) + \rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t) \nabla \vec{u} \right) J(\mathbf{x}, t) dx
\end{aligned}$$

Transforming back to $W(t)$ and using the material derivative of Chapter 2, equation (2.0.8) leads to

$$\begin{aligned}
& \int_{W(0)} \left(\frac{d}{dt} (\rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t)) + \rho(\xi(\mathbf{x}, t), t) f(\xi(\mathbf{x}, t), t) \nabla \vec{u} \right) J(\mathbf{x}, t) dx = \\
& \int_{W(t)} \frac{D}{Dt} (\rho(\xi, t) f(\xi, t)) + \rho(\xi, t) f(\xi, t) \nabla \vec{u} d\xi
\end{aligned}$$

Using the continuity equation of Chapter 2, equation (2.1.3):

$$0 = \frac{\partial \rho}{\partial t} + \operatorname{div}(\rho \vec{u}) = \frac{\partial \rho}{\partial t} + \rho \nabla \cdot \vec{u} + \vec{u} \cdot \nabla \rho = \frac{D\rho}{Dt} + \rho \nabla \cdot \vec{u}$$

Thus

$$\int_{W(t)} \frac{D}{Dt} (\rho(\xi, t) f(\xi, t)) + \rho(\xi, t) f(\xi, t) \nabla \vec{u} d\xi = \int_{W(t)} \rho \frac{Df}{Dt} d\xi$$

□

Theorem 1.2.3. *Be $f : W \times (0, \infty) \rightarrow \mathbb{R}$ continuously differentiable. Then*

$$\frac{d}{dt} \int_{W(t)} f(\xi, t) d\xi = \int_{W(t)} \left(\frac{\partial f(\xi, t)}{\partial t} + \nabla \cdot (f(\xi, t) \vec{u}(\xi, t)) \right) d\xi \quad (\text{A.2.13})$$

holds.

Proof. Employing integration by parts and Lemma 1.2.1 of this section gives

$$\begin{aligned}
& \frac{d}{dt} \int_{W(t)} f(\xi, t) d\xi = \frac{d}{dt} \int_{W(0)} f(\xi(\mathbf{x}, t), t) J(\mathbf{x}, t) d\mathbf{x} = \\
& \int_{W(0)} \left(\frac{Df(\xi(\mathbf{x}, t), t)}{Dt} J(\mathbf{x}, t) + f(\xi(\mathbf{x}, t), t) \frac{dJ(\mathbf{x}, t)}{dt} \right) d\mathbf{x} = \\
& \int_{W(0)} \left(\frac{Df(\xi(\mathbf{x}, t), t)}{Dt} + f(\xi(\mathbf{x}, t), t) \nabla \cdot \vec{u} \right) J(\mathbf{x}, t) d\mathbf{x} = \\
& \int_{W(t)} \left(\frac{\partial f(\xi, t)}{\partial t} + \nabla \cdot (f(\xi, t) \vec{u}) \right) d\xi
\end{aligned}$$

□

Appendix B

B.1 A Finite Element (FEM) discretization

In the FEM approach, the key idea is to approximate the solution u in a finite-dimensional subspace of the function space that provides the abstract framework of the problem given. The following derivation uses the nonlinear generalized Poisson equation, however, the procedure applies analogously to the linear generalized Poisson equation as well.

Be $S_1 = [\psi_1, \dots, \psi_n]$ the finite element space associated with the finest grid. The approximate solution u is sought in S_1 and is therefore denoted as

$$u(x) = \sum_{i=1}^n u_i \psi_i(x) \quad (\text{B.1.1})$$

For the finite element technique, it is necessary to reformulate equation (4.6.5) as a variational problem. Therefore, it is multiplied with the ansatz functions ψ_j (the ansatz functions possess all the necessary properties of test functions) and integrated over the computational domain Ω ,

$$-\int_{\Omega} \nabla \cdot (\kappa(u) \nabla u) \psi_j dV + \int_{\Omega} c(u) \psi_j dV = \int_{\Omega} f(x) \psi_j dV \quad j = 1, \dots, n \quad (\text{B.1.2})$$

Integration by parts leads to

$$\int_{\Omega} \kappa(u) \nabla u \nabla \psi_j dV + \int_{\Omega} c(u) \psi_j dV - \int_{\partial\Omega_N} \kappa(u) g(x) \psi_j(x) dS = \int_{\Omega} f(x) \psi_j dV \quad j = 1, \dots, n \quad (\text{B.1.3})$$

The boundary integral $-\int_{\partial\Omega_N} \kappa(u) g(x) \psi_j(x) dS$ stems from a Neumann boundary condition and vanishes if homogeneous Dirichlet boundary conditions are employed.

Inserting the definition of the approximate solution (B.1.1) leads to

$$\underbrace{\int_{\Omega} \kappa(u) u_i \nabla \psi_i \nabla \psi_j dV + \int_{\Omega} c(u) \psi_j dV - \int_{\partial\Omega_N} \kappa(u) g(x) \psi_j(x) dS}_{\mathcal{A}(u)=\mathcal{A}(u_1, \dots, u_n)} = \underbrace{\int_{\Omega} f(x) \psi_j dV}_b \quad j = 1, \dots, n \quad (\text{B.1.4})$$

which may be conveniently denoted by $\mathcal{A}(u) = b$. To compute the nonlinear operator, it can be

B.

split up into

$$\mathcal{A}(u) = \bar{\mathcal{A}} \cdot u \int_{\Omega} c(u) \psi_j dV - \int_{\partial\Omega_N} \kappa(u) g(x) \psi_j(x) dS \quad (\text{B.1.5})$$

where

$$\bar{\mathcal{A}} = \begin{pmatrix} a(\psi_1, \psi_1) & a(\psi_1, \psi_2) & a(\psi_1, \psi_3) & \cdots & a(\psi_1, \psi_n) \\ a(\psi_2, \psi_1) & a(\psi_2, \psi_2) & a(\psi_2, \psi_3) & \cdots & a(\psi_2, \psi_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a(\psi_n, \psi_1) & a(\psi_n, \psi_2) & \cdots & a(\psi_n, \psi_{n-1}) & a(\psi_n, \psi_n) \end{pmatrix} \quad (\text{B.1.6})$$

and the scalar product $a(\psi_i, \psi_j)$ is defined as

$$a(\psi_i, \psi_j) = \int_V \kappa(u) \nabla \psi_i \nabla \psi_j dV \quad (\text{B.1.7})$$

The integral expressions are evaluated using the trapezoidal rule.

Choice of the subspace(s) S_j

The choice of the ansatz functions ψ_i determines the overall order of the discretization. In 1D, piecewise linear (spline) functions are used. Thus, the discretization is of second order accuracy. The ansatz functions are defined as

$$\psi_i(x) = \begin{cases} a_i^1 x + b_i^1 & \text{if } x \leq x_i \\ a_i^2 x + b_i^2 & \text{if } x > x_i \end{cases} \quad (\text{B.1.8})$$

$$\psi_i(x_j) = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{else} \end{cases} \quad (\text{B.1.9})$$

In case the domain Ω is equidistantly spaced,

$$\Omega = \{x_i, i = 1, \dots, n\} \quad x_i = a + i \cdot h_x \quad (\text{B.1.10})$$

the above definition leads to

$$\psi_i(x) = \begin{cases} \frac{x-x_{i-1}}{h_x} & \text{if } x_{i-1} \leq x \leq x_i \\ \frac{x_i-x}{h_x} & \text{if } x_i \leq x \leq x_{i+1} \\ 0 & \text{else} \end{cases} \quad (\text{B.1.11})$$

Note that due to the bounded support of the ansatz functions ψ_i , the only nontrivial entries in the stiffness matrix $\mathcal{A}$ are the entries $a(\psi_i, \psi_{i-1})$, $a(\psi_i, \psi_i)$ and $a(\psi_i, \psi_{i+1})$, rendering a tridiagonal system of equations.

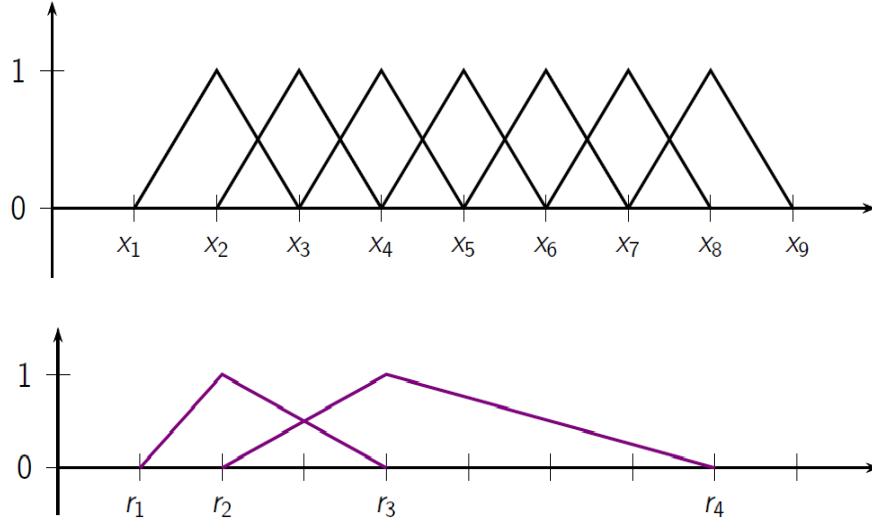


Figure B.1: In the upper picture, the one-dimensional ansatz functions are depicted on an equidistantly spaced grid. In case the grid is non-equidistantly spaced, the ansatz functions appear as in the lower picture.

Within the ANTARES code, not only equidistant but also stretched grids are employed. Based on a prescribed stretching factor q the corresponding grid points are computed via

$$r_i = r_{\text{top}} - (q^i - 1)h_r \quad (\text{B.1.12})$$

where h_r is a constant value reading

$$h_r = \frac{r_{\text{top}} - r_{\text{bot}}}{q^n - 1} \quad (\text{B.1.13})$$

(The notation r_i instead of x_i already indicates that in the present context, the stretched grid is used along with spherical coordinates.)

Consequently, the ansatz functions read

$$\psi_i(r) = \begin{cases} -\frac{q^{-i}(-h_r q + h_r q^i + q r_{\text{bot}})}{h_r(-1+q)} + \frac{q^{1-i}}{h_r(-1+q)}r & \text{if } r_{i-1} \leq r \leq r_i \\ \frac{q^{-i}(-h_r + h_r q^{1+i} + r_{\text{bot}})}{h_r(-1+q)} + \frac{q^{-i}}{h_r(1-q)}r & \text{if } r_i \leq r \leq r_{i+1} \\ 0 & \text{else} \end{cases} \quad (\text{B.1.14})$$

The one-dimensional ansatz functions are depicted in Figure B.1.

If spherical coordinates are employed, geometric factors enter the computation of the gradient and the divergence operator. Let F be a vector field and f denote a scalar function, then there is

$$\nabla \cdot F = \frac{1}{r^2} \frac{\partial}{\partial r} (r^2 F_r) + \frac{1}{r \sin \theta} \frac{\partial}{\partial \theta} (\sin \theta F_\theta) + \frac{1}{r \sin \theta} \frac{\partial}{\partial \phi} (F_\phi) \quad (\text{B.1.15})$$

$$\nabla f = \frac{\partial f}{\partial r} \hat{r} + \frac{1}{r \sin \theta} \frac{\partial f}{\partial \theta} \hat{\theta} + \frac{1}{r \sin \theta} \frac{\partial f}{\partial \phi} \hat{\phi} \quad (\text{B.1.16})$$

B.

where $\hat{r}$, $\hat{\phi}$ and $\hat{\theta}$ denote the unit vectors in the respective directions.

Also in the evaluation of the integral expressions appearing in the definition of the operator $\mathcal{A}$ the employment of spherical coordinates must be taken into account by using the functional determinant $dV = r^2 \sin \theta dV_{\text{pol}}$. Note that although the computation in this context is one-dimensional (the ansatz functions only depend on r), the full spherical coordinates are employed and it is assumed that whereas r varies, the angles ϕ and θ are set to constant values $\phi = 0^\circ$ and $\theta = 90^\circ$.

In two spatial dimensions, a decision on the shape of the finite elements the computational domain Ω is partitioned into has to be made. A popular choice is the use of triangular finite elements. An alternative is the employment of quadrilateral bilinear finite elements (see Figure B.2). In the present context, the quadrilateral elements are chosen since they offer several advantages over the triangular ones:

- In the multigrid technique considered, the coarse grid operators and the intergrid operators heavily depend on the underlying discretization. Contrarily to the triangular elements, the choice of quadrilateral finite elements leads to symmetric restriction and prolongation operators. This is considered advantageous especially in the context of solving nonlinear problems.
- Although both FEM triangulations lead to operators A (respectively $\mathcal{A}$ if the nonlinear case is considered) containing five non-zero entries per line, the discretization based on triangular finite elements introduces a dependency of a grid point (i,j) also on its diagonal neighbor $(i+1,j+1)$ instead of only its “direct” neighboring points indexed $(i+1,j)$, $(i-1,j)$, $(i,j+1)$, $(i,j-1)$ as do the quadrilateral elements. Whereas this is of no importance in case the linear multigrid scheme or the FAS method is employed, the global Jacobian required for the MG-Newton method reflects this dependency by containing seven non-zero entries per line instead of only five. Apart from the additional hard disk space this would require in an implementation, major modifications in the Galerkin coarsening approach as well as the smoothing procedure and the direct solver would be required to solve the linearized system properly.

The two-dimensional bilinear ansatz functions corresponding to the quadrilateral finite elements are defined as

$$\psi_{i,j}(x,y) = \begin{cases} a_{ji}^1 + b_{ji}^1 x + c_{ji}^1 y + d_{ji}^1 xy & \text{if } x \leq x_i, y \leq y_j \\ a_{ji}^2 + b_{ji}^2 x + c_{ji}^2 y + d_{ji}^2 xy & \text{if } x > x_i, y \leq y_j \\ a_{ji}^3 + b_{ji}^3 x + c_{ji}^3 y + d_{ji}^3 xy & \text{if } x > x_i, y > y_j \\ a_{ji}^4 + b_{ji}^4 x + c_{ji}^4 y + d_{ji}^4 xy & \text{if } x \leq x_i, y > y_j \end{cases} \quad (\text{B.1.17})$$

$$\psi_{i,j}(x_k, y_l) = \begin{cases} 1 & \text{if } (i,j) = (k,l) \\ 0 & \text{else} \end{cases} \quad (\text{B.1.18})$$

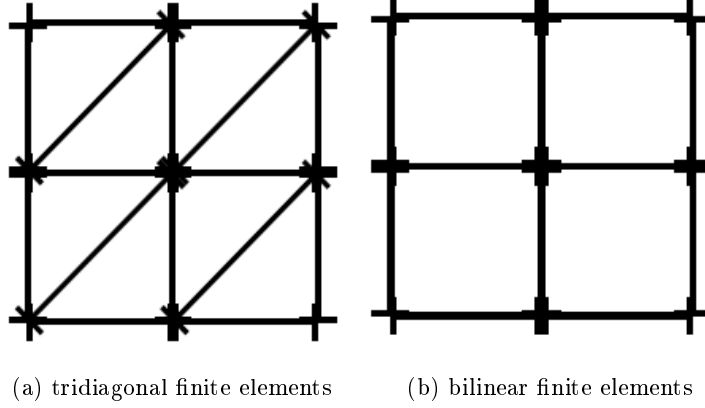


Figure B.2: In the left, the partitioning of the domain with triangular elements is depicted, the figure on the right hand side shows the use of bilinear finite elements.

or, in case of the spherical grid,

$$\psi_{i,j}(r, \phi) = \begin{cases} a_{ji}^1 + b_{ji}^1 r + c_{ji}^1 \phi + d_{ji}^1 r \phi & \text{if } r \leq r_i, \phi \leq \phi_j \\ a_{ji}^2 + b_{ji}^2 r + c_{ji}^2 \phi + d_{ji}^2 r \phi & \text{if } r > r_i, \phi \leq \phi_j \\ a_{ji}^3 + b_{ji}^3 r + c_{ji}^3 \phi + d_{ji}^3 r \phi & \text{if } r > r_i, \phi > \phi_j \\ a_{ji}^4 + b_{ji}^4 r + c_{ji}^4 \phi + d_{ji}^4 r \phi & \text{if } r \leq r_i, \phi > \phi_j \end{cases} \quad (\text{B.1.19})$$

$$\psi_{i,j}(r_k, \phi_l) = \begin{cases} 1 & \text{if } (i,j) = (k,l) \\ 0 & \text{else} \end{cases} \quad (\text{B.1.20})$$

Note that the ansatz (B.1.17) respectively (B.1.19) is only valid if the grid is aligned with the coordinate axes (see [3]), which is the case within the ANTARES framework.

The integral entries the operator $\mathcal{A}$ (resp. A) is evaluated using the trapezoidal rule.

Note that in two spatial dimensions, the operator $\mathcal{A}$ (resp. A) is a 4-dimensional structure in, $(\mathbb{R}^n \times \mathbb{R}^m) \times (\mathbb{R}^n \times \mathbb{R}^m)$ since it is obtained from expressions like $a(\psi_{ij}, \psi_{kl})$. Therefore, the bijective mapping $r : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, r(i, j) = (j - 1)n + i$ is used to transform $\mathcal{A}$ (resp. A) into a two-dimensional matrix operator. In this thesis, the referring to $\mathcal{A}$ (resp. A) is always to its transformed shape.

The resulting (transformed) operator is symmetric and positive definite, being of tridiagonal structure in 1D and containing 5 non-zero entries in the two-dimensional case.

Treatment of boundary conditions

The boundary conditions equation (4.6.5) is subjected to are typically of periodic type in horizontal direction and of Dirichlet- or Neumann type in the vertical direction.

B.

- Periodic boundary conditions

If periodic boundary conditions are employed, the boundary corresponds to the opposite boundary of the domain. Practically, there is no boundary.

- Neumann boundary conditions

By specifying Neumann boundary conditions, the gradient $g = \nabla u$ is fixed at the corresponding boundary. This boundary condition induces a non-trivial boundary integral $-\int_{\partial\Omega_N} \kappa(u)g(x)\psi_j(x)dS$ in the variational formulation (B.1.4) of the operator $\mathcal{A}$ (resp. A) which has to be discretized accordingly.

- Dirichlet boundary conditions

A Dirichlet boundary condition prescribes values for the unknown u at the boundary. Naturally, this type of boundary condition leads to a reduction of the number of equations in the (non-)linear system.

For the treatment of Dirichlet boundary conditions, it is customary to define a function $w(x)$ such that $u = u^0 + w$ and $u^0|_{\partial\Omega} = 0$ in order to arrive at a homogeneous problem for u^0 .

In case of a linear system of equations, it is customary to define w such that it is 0 everywhere but on the boundaries. In the nonlinear case, however, it is feared that the use of a discontinuous function introduces spurious oscillations in the solution. Therefore, $w(x)$ is defined as a linear function

$$w(x) = \frac{u_b - u_a}{x_b - x_a}x + u_a \quad (\text{B.1.21})$$

Replacing u by $u + w$ in the variational formulation (B.1.4), there is

$$\begin{aligned} \mathcal{A}^0(u^0) &= \mathcal{A}(u^0 + w) = \int_{\Omega} \kappa(\tilde{u}^0 + w) \nabla(\tilde{u}^0 + w) \nabla \psi_j dx + \int_{\Omega} c(\tilde{u}^0 + w) \psi_j dx \\ &= \int_{\Omega} \kappa(\tilde{u}^0 + w) \nabla u \nabla \psi_j dx + \int_{\Omega} \kappa(\tilde{u}^0 + w) \nabla w \nabla \psi_j dx + \int_{\Omega} c(\tilde{u}^0 + w) \psi_j dx \end{aligned}$$

and to obtain u^0 , the homogeneous system $\mathcal{A}^0 u^0 = f^0$ must be solved (see [43]).

Appendix C

C.1 Stencil Notation

This section is taken from [53].

In order to obtain a concise description of the transfer operators used in the multigrid algorithm, stencil notation is used.

Stencil notation for operators of type $\mathbb{R}^n \rightarrow \mathbb{R}^n$

Be $A : \mathbb{R}^n \rightarrow \mathbb{R}^n$ a linear operator. Then, using stencil notation, Au can be denoted by

$$(Au)_i = \sum_{j \in \mathbb{Z}^n} A(i, j) u_{i+j} \quad i \in G \quad (\text{C.1.1})$$

where G denotes the index set of the grid, i.e. $G = \{i_1, \dots, i_n\}$.

The set S_A defined by

$$S_A = \{j \in \mathbb{Z}^n : \exists i \in G \text{ with } A(i, j) \neq 0\} \quad (\text{C.1.2})$$

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 .

Stencil Notation for restriction operators

Be $r : \mathbb{R}^n \rightarrow \mathbb{R}^{n/2}$ a restriction operator. Using stencil notation, r can be represented by

$$(ru)_i = \sum_{j \in \mathbb{Z}^n} r(i, j) u_{2i+j} \quad i \in \bar{G} \quad (\text{C.1.3})$$

where $\bar{G}$ denotes the coarser grid.

C.

Stencil notation for prolongation operators

Be $p : \mathbb{R}^{n/2} \rightarrow \mathbb{R}^n$ a prolongation operator. Using stencil notation, p can be represented by

$$(pu)_i = \sum_{j \in \mathbb{Z}^n} p^*(j, i - 2j) \bar{u}_j \quad i \in G \quad (\text{C.1.4})$$

p^* denotes the adjoint of the prolongation operator. Hence, a convenient way to define the prolongation is by specifying its adjoint. In the context of a finite element discretization, the prolongation is known to be the adjoint of the restriction operator, therefore, the prescription (C.1.4) can readily be used.

Computation of the coarse grid operator with Galerkin approximation

With the use of stencil notation, simple formulae to compute the coarse grid operator $\bar{A}$ with GCA are obtained. Combining equation (C.1.1) with (C.1.4) leads to

$$(Ap\bar{u})_i = \sum_{k \in \mathbb{Z}^n} A(i, k) (p\bar{u})_{i+k} = \sum_{j \in \mathbb{Z}^n} \sum_{k \in \mathbb{Z}^n} A(i, k) p^*(j, i + k - 2j) \bar{u}_j \quad (\text{C.1.5})$$

Inserting the definition of the restriction in stencil notation, (C.1.3), the expression above reads

$$(rAp\bar{u})_i = \sum_{m \in \mathbb{Z}^n} r(i, m) (Ap\bar{u})_{2i+m} = \quad (\text{C.1.6})$$

$$= \sum_{m \in \mathbb{Z}^n} \sum_{k \in \mathbb{Z}^n} \sum_{j \in \mathbb{Z}^n} r(i, m) A(2i + m, k) p^*(j, 2i + m + k - 2j) \bar{u}_j \quad (\text{C.1.7})$$

With the change of variables $j = i + n$ this becomes

$$\sum_{m \in \mathbb{Z}^n} \sum_{k \in \mathbb{Z}^n} \sum_{n \in \mathbb{Z}^n} r(i, m) A(2i + m, k) p^*(i + n, m + k - 2n) \bar{u}_{i+n} \quad (\text{C.1.8})$$

from which it follows that

$$\bar{A}(i, n) = \sum_{m \in \mathbb{Z}^n} \sum_{k \in \mathbb{Z}^n} \sum_{n \in \mathbb{Z}^n} r(i, m) A(2i + m, k) p^*(i + n, m + k - 2n) \quad (\text{C.1.9})$$

Defining the structure S_r of the restriction operator by

$$S_r = \{j \in \mathbb{Z}^n : \exists i \in \bar{G} \text{ with } r(i, j) \neq 0\} \quad (\text{C.1.10})$$

and remembering S_A , equation (C.1.9) is equivalent to

$$\bar{A}(i, n) = \sum_{m \in S_r} \sum_{k \in S_A} \sum_{n \in \mathbb{Z}^n} r(i, m) A(2i + m, k) p^*(i + n, m + k - 2n) \quad (\text{C.1.11})$$

With this formula, the computation of $\bar{A}$ is straightforward.

Danksagung

Die vorliegende Arbeit wurde durch die Projekte P25229, P20762, P20973 und P18224 des österreichischen Fonds zur Förderung der wissenschaftlichen Forschung (FWF) finanziell gefördert. Im Besonderen möchte ich meinem Doktorvater Herbert Muthsam für die interessante Themenstellung und zahlreichen Anregungen danken. Spezieller Dank gilt auch Friedrich Kupka für seine ständige Gesprächsbereitschaft und große Geduld sowie Othmar Koch und Inmaculada Higuera, die dieser Arbeit die entscheidende fachliche Richtung gaben.

Außerdem danke ich meinen Kollegen Hannes Grimm-Strele, Florian Zaussinger, Eva Mundprecht, Bernhard Löw-Baselli und Patrick Blies für die angenehme und produktive Arbeitsatmosphäre.

Ein besonderer Dank gebührt auch meinem Verlobten Helmut Stadlmayer für seine nicht enden wollende Unterstützung und Motivation sowie meinen Eltern.

C.

Bibliography

- [1] A.J.Chorin and J.E.Marsden. *A Mathematical Introduction to Fluid Mechanics*. Springer Verlag, 2nd edition, 1990.
- [2] D.R. Alexander and J.W. Ferguson. Low-temperature Rosseland opacities. *The Astrophysical Journal*, 437:879–891, 1994.
- [3] D. Braess. *Finite Elemente*. Springer Verlag Berlin-Heidelberg, 4th edition, 2007.
- [4] Achi Brandt. *Multi-Grid Techniques: 1984 Guide with Applications to Fluid Dynamics*. Gesellschaft für Mathematik und Datenverarbeitung MBH Bonn, 1984.
- [5] J.C. Butcher. *The numerical analysis of ordinary differential equations*. John Wiley & Sons, 1987.
- [6] E. Dorfi and M.U. Feuchtinger. Nonlinear stellar pulsations 1. *Astronomy and Astrophysics*, 249:417–427, 1991.
- [7] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, 2002.
- [8] L. Ferracina and M.N. Spijker. An extension and analysis of the Shu-Osher representation of Runge-Kutta methods. *Mathematics of Computation*, 74, number 249:201–219, 2004.
- [9] S. Gottlieb, D. Ketcheson, and C.-W. Shu. High order strong stability preserving time discretizations. *J. Sci. Comput.*, 38:251–289, 2009.
- [10] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation*, 67(221):73–85, January 1998.
- [11] Hannes Grimm-Strele. Numerical solution of the generalised Poisson equation on parallel computers. Master’s thesis, Universität Wien, März 2010. available at <http://othes.univie.ac.at/9200/>.
- [12] H. Guillard and A. Murrone. On the behavior of upwind schemes in the low Mach number limit: 2. Godunov type schemes. *Computers and Fluids*, 33:655–675, 2004.
- [13] Wolfgang Hackbusch. *Multi-Grid Methods and Applications*. Springer Verlag Berlin-Heidelberg, 1985.

BIBLIOGRAPHY

- [14] G. Hager and G. Wellein. *Introduction to High Performance Computing for Scientists and Engineers*. Chapman & Hall/CRC, Computational Science Series, 2011.
- [15] Martin Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*. Teubner Verlag, 2nd edition, 2002.
- [16] N. Happenhofer, H. Grimm-Strele, F. Kupla, B. Loew-Baselli, and H. Muthsam. A low Mach number solver: Enhancing Applicability. *Journal of Computational Physics*, 236:96–118, 2013.
- [17] I. Higueras. Strong stability for additive Runge–Kutta methods. *SIAM J. Numer. Anal.*, 44:1735–1758, 2006.
- [18] I. Higueras. Characterizing strong stability preserving additive Runge–Kutta methods. *J. Sci. Comput.*, 39:115–128, 2009.
- [19] I. Higueras, N. Happenhofer, O. Koch, and F. Kupka. Optimized strong stability preserving IMEX Runge-Kutta methods. submitted to the Journal of Computational and Applied Mathematics.
- [20] I. Higueras, N. Happenhofer, O. Koch, and F. Kupka. Optimized IMEX Runge-Kutta methods for simulations in astrophysics: A detailed study. Technical report, Institute for Analysis and Scientific Computing, Vienna University of Technology, 2012. available at <http://www.asc.tuwien.ac.at/preprint/2012/asc14x1012.pdf>.
- [21] Wolfgang Hillebrandt and Friedrich Kupka, editors. *Lecture Notes in Physics 756: Interdisciplinary Aspects of Turbulence*, pages 107–155. Springer Verlag Berlin Heidelberg, 2009.
- [22] H.J.Muthsam, F.Kupka, B. Löw-Baselli, C.Obertscheider, M.Langer, and P.Lenz. ANTARES - a numerical tool for astrophysical research with applications to solar granulation. *New Astronomy*, 15:7460–475, 2010.
- [23] J.E. Jones and S.F. McCormick. *Parallel Multigrid Methods*, volume 4 of *ICASE/LaRC Interdisciplinary Series in Science and Engineering*, pages 203–224. Springer Netherlands, 1997.
- [24] M. Jung and U. Langer. *Methode der finiten Elemente für Ingenieure*. Teubner Verlag, 2001.
- [25] K.C. Karki and S.V. Patankar. Pressure based calculation procedure for viscous flows at all speeds in arbitrary configurations. *AIAA Journal*, 27:1167–1174, September 1989.
- [26] O. Koch, F. Kupka, B. Löw-Baselli, A. Mayrhofer, and F. Zaussinger. SDIRK methods for the ANTARES code. ASC Report 32/2010, Inst. for Anal. and Sci. Comput., Vienna Univ. of Technology, 2010. Available at <http://www.asc.tuwien.ac.at/preprint/2010/asc32x2010.pdf>.

-
- [27] J. Kraaijevanger. Contractivity of Runge-Kutta methods. *BIT*, 11:482–528, 1991.
 - [28] F. Kupka, J. Ballot, and H.J. Muthsam. Effects of resolution and Helium abundance in A-star surface convection simulations. *Comm. in Asteroseismology*, 160:30–63, 2009.
 - [29] F. Kupka, N. Happenhofer, I. Higuera, and O. Koch. Total-variation-diminishing implicit-explicit Runge-Kutta methods for the simulation of double-diffusive convection in astrophysics. *Journal of Computational Physics*, 231:3561–3586, 2012.
 - [30] J.E. Marsden and A. Tromba. *Vector Calculus*. Palgrave Macmillan, 5th edition, 2003.
 - [31] U. Meier Yang. Parallel algebraic multigrid methods- high performance preconditioners. Technical report, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory.
 - [32] D. Mihalas and B. Weibel Mihalas. *Foundations of radiation hydrodynamics*. Oxford University Press, 1984.
 - [33] M.H. Montgomery, D.E. Winget, A. Allen, R.E. Falcon, J. Gomez, J. Ellis, V. Havanur, K. Luecke, K. Melin, G. Miller, and C. Rendon. Exploring the universe with white dwarf stars: The first year of the freshman research initiative. In J.D. Stanford, L.M. Green, L. Hao, and Y. Mao, editors, *New Horizons in Astronomy: Frank N. Bash Symposium 2009*, volume 432 of *Lecture Notes in Computer Science*, pages 225–229. Springer Verlag, 2010.
 - [34] M. Motamed, C.B. Macdonald, and S.J. Ruuth. On the linear stability of the fifth-order WENO discretization. *Journal of Scientific Computing*, 47:127–149, 2011.
 - [35] E. Mundprecht and H. J. Muthsam. Modelling Stellar Convection and Pulsation in Multidimensions Using the ANTARES Code. In J. C. Suárez, R. Garrido, L. A. Balona, and J. Christensen-Dalsgaard, editors, *Stellar Pulsations: Impact of New Instrumentation and New Insights*, volume 31 of *Advances in Solid State Physics*, page 53, 2013.
 - [36] E. Mundprecht, H. J. Muthsam, and F. Kupka. Multidimensional realistic modelling of Cepheid-like variables. I: Extensions of the ANTARES code. *Monthly Notices of the Royal Astronomical Society*, 435(1):3191–3205, 2013.
 - [37] Eva Mundprecht. *Numerical modelling of pulsation and convection in cepheids*. PhD thesis, University of Vienna, 2011. available at <http://othes.univie.ac.at/14139/>.
 - [38] H.J. Muthsam, W. Göb, F. Kupka, and W. Liebich. Interacting convection zones. *New Astronomy*, 4:405–417, 1999.
 - [39] Christof Obertscheider. *Modelling of solar granulation-Implementation and Comparison of numerical schemes*. PhD thesis, University of Vienna, 2007.
 - [40] L. Pareschi and G. Russo. Implicit–explicit Runge–Kutta schemes and application to hyperbolic systems with relaxation. *J. Sci. Comput.*, 25:129–155, 2005.

BIBLIOGRAPHY

- [41] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.
- [42] F.J. Rogers, Swenson F.J., and C.A. Iglesias. OPAL equation-of-state tables for astrophysical applications. *The Astrophysical Journal*, 456:902–908, 1996.
- [43] Thomas Roubicek. *Nonlinear Partial Differential Equations with Applications*. Birkhäuser Verlag, Basel - Boston - Berlin, 2005.
- [44] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.
- [45] J. Stoer and R. Bulirsch. *Numerische Mathematik 2*. Springer-Verlag, Berlin-Heidelberg-New York, 3rd edition, 1990.
- [46] J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. SIAM, Philadelphia, PA, 2nd edition, 2004.
- [47] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.
- [48] Morrison T.S. *The Banach and Brouwer Fixed Point Theorems and Applications*. ProQuest, 2007.
- [49] W. Unno and E.A. Spiegel. The Eddington approximation in the radiative heat equation. *Publications of the Astronomical Society of Japan*, 18(2):85–85, 1966.
- [50] I. Viallet, M. and. Baraffe and R. Walder. Towards a new generation of multi-dimensional stellar evolution models: development of an implicit hydrodynamic code. *Astronomy & Astrophysics*, 531:A86, 2011.
- [51] R. Wang and R.J. Spiteri. Linear instability of the fifth-order WENO method. *SIAM Journal of Numerical Analysis*, 45 (5):1871–1901, 2007.
- [52] A. Weiss, W. Hillebrandt, H.-C. Thomas, and H. Ritter. *Cox & Giuli's Principles of Stellar Structure*. Cambridge Scientific Publishers Ltd, extended 2nd edition, 2004.
- [53] P. Wesseling. *An Introduction to Multigrid Methods*. R.T. Edwards, Inc, 2004.
- [54] F. Zaussinger, F. Kupka, and H. J. Muthsam. Semi-convection. In M. Goupil, K. Belkacem, C. Neiner, F. Lignières, and J. J. Green, editors, *Lecture Notes in Physics, Berlin Springer Verlag*, volume 865 of *Lecture Notes in Physics, Berlin Springer Verlag*, page 219, 2013.
- [55] F. Zaussinger and H.C. Spruit. Semiconvection: numerical simulations. *Astronomy & Astrophysics*, 554:A119, 2011.
- [56] Florian Zaussinger. *Numerical simulation of double-diffusive convection*. PhD thesis, University of Vienna, 2010. available at <http://othes.univie.ac.at/13172/>.

- [57] W. Zhang, L. Howell, A. Almgren, A. Burrows, and J. Bell. Castro: A new compressible astrophysical solver 2. Grey radiation hydrodynamics. *The Astrophysical Journal*, 715(2):1221–1238, 2011.

Curriculum Vitae

Name: Natalie Happenhofer
Foreign languages: English, Spanish
Programming languages: Fortran90, C++, MatLAB

Working experience

September 2004 - March 2008:	Employee at the supermarket BILLA (10h/week)
October 2008 - February 2008:	Tutor for “Analysis for Physicists” Institute for Mathematics, University of Vienna
March 2009 :	Carer at the exposition ‘Imaginary’ in Vienna organized by the Mathematical Institute of Oberwolfach, Germany
September 2009	Scientific researcher at the Max-Planck Institute for Astrophysics, Garching, Germany
October 2009 - February 2010	Tutor for ‘Introduction to linear algebra’ Institute for Mathematics, University of Vienna
February 2008-July2010	Working student at the research group ACORE Institute for Mathematics, University of Vienna Field of Activity: Visualization of scientific data, especially extending the open source software ParaView
August 2010 - Oktober2012	Scientific researcher at the research group ACORE Institute for Mathematics, University of Vienna
September 2012 - February 2013	Lecturer for “Uebungen zu Analysis fuer Physik”, Institute for Mathematics, University of Vienna
Since November 2012	Scientific researcher at the research group ACORE, part time Institute for Mathematics, University of Vienna
	Analysis Engineer at FACC AG, part time

Education

September 1993 - June 1997:	Primary school: Volksschule Weikersdorf in Baden bei Wien
September 1997 - June 2005:	High school: Bundesgymnasium Baden, Frauengasse 3-5 in Baden bei Wien
June 2003 :	Cambridge First Certificate of English
August 2003 - July 2004 :	Student exchange to Lima, Peru
<i>26.6.2005:</i>	<i>High school diploma with distinction</i>
October 2005 - October 2006:	Diploma studies in Physics
October 2005 - June 2010	Diploma studies in Mathematics
<i>24.6.2010</i>	<i>Master's degree with distinction</i>