Technische Universität München
Zentrum Mathematik

Adaptive grids and numerical fluid simulations for scrape-off layer plasmas

Hans-Joachim Klingshirn

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

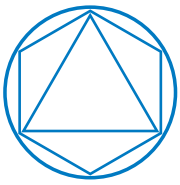
Vorsitzender:	Univ.-Prof. Dr. O. Junge
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. P. Rentrop
2.	Hon.-Prof. Dr. S. Günter
3.	Univ.-Prof. Dr. A. Rieder Universität Karlsruhe (TH)

Die Dissertation wurde am 07.04.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 21.07.2010 angenommen.

Adaptive grids and numerical fluid simulations for scrape-off layer plasmas

Dissertation
von
Hans-Joachim Klingshirn

durchgeführt am
Max-Planck-Institut für Plasmaphysik



Technische Universität München
Zentrum Mathematik

Zusammenfassung

In Kernfusionsexperimenten mit magnetischem Einschluss werden Plasmen mit lokalen Temperaturen von mehr als 100 Millionen Kelvin erzeugt. Die Plasmarandschicht, die in direktem Kontakt mit der Gefäßwand steht, ist hierbei von zentraler Bedeutung für die Qualität des Plasmaeinschlusses und die Haltbarkeit der Oberflächenmaterialien. Um ihr Verhalten zu untersuchen, werden neben Experimenten auch numerische Simulationen genutzt.

Diese Arbeit behandelt den Einsatz adaptiver Rechengitter und hierfür geeigneter numerischer Verfahren für Randschichtsimulationen. Die resultierenden Algorithmen ermöglichen die dynamische Adaption von am Magnetfeld ausgerichteten Gittern zur präzisen Beschreibung des stark anisotropen Energie- und Teilchentransports im Plasma. Die entwickelten Verfahren werden schrittweise in den Multifluid-Plasmacode B2 integriert, mit dem Ziel, die Rechenzeit der Simulationen zu verkürzen und den Anwendungsbereich des Codes zu erweitern.

Abstract

Magnetic confinement nuclear fusion experiments create plasmas with local temperatures in excess of 100 million Kelvin. In these experiments the scrape-off layer, which is the plasma region in direct contact with the device wall, is of central importance both for the quality of the energy confinement and the wall material lifetime. To study the behaviour of the scrape-off layer, in addition to experiments, numerical simulations are used.

This work investigates the use of adaptive discretizations of space and compatible numerical methods for scrape-off layer simulations. The resulting algorithms allow dynamic adaptation of computational grids aligned to the magnetic fields to precisely capture the strongly anisotropic energy and particle transport in the plasma. The methods are applied to the multi-fluid plasma code B2, with the goal of reducing the runtime of simulations and extending the applicability of the code.

Contents

1. Introduction

1.1. Nuclear fusion

Nuclear fusion is the energy source of the stars. In the core of our sun, every second 564 million tons of hydrogen are fused to 560 million tons of helium. The mass defect occurs owing to the higher binding energy of the helium nuclei and results in the release of energy. Since the middle of the 20th century, fusion research has been pursued to establish this process as a viable energy source on earth. It is particularly attractive because the needed resources (hydrogen and lithium) are available in abundance and no CO_2 is emitted in the process [67].

To initiate the fusion process, nuclei have to come close enough together for the strong nuclear force to overcome the repulsive Coulomb force due to their positive charge. The best candidate for such a reaction is fusion of the hydrogen isotopes deuterium and tritium

$$\mathrm{D}^2 + T^3 \rightarrow \mathrm{He}^4 + \mathrm{n}^1 + 17.6\mathrm{MeV}. \tag{1.1}$$

A gas consisting of these isotopes has a maximum reaction rate at a temperature of several 10s of keV[1], i.e. more than 100 million degrees Celsius.

At such high temperatures electrons and nuclei separate and form a plasma. Because Coulomb collisions between nuclei are much more likely to occur than fusion reactions, a high number of collisions is required to achieve a sufficient energy yield. Therefore a good confinement of the plasma is necessary. The condition for a self-sustaining D-T thermonuclear fusion plasma is given by [120]

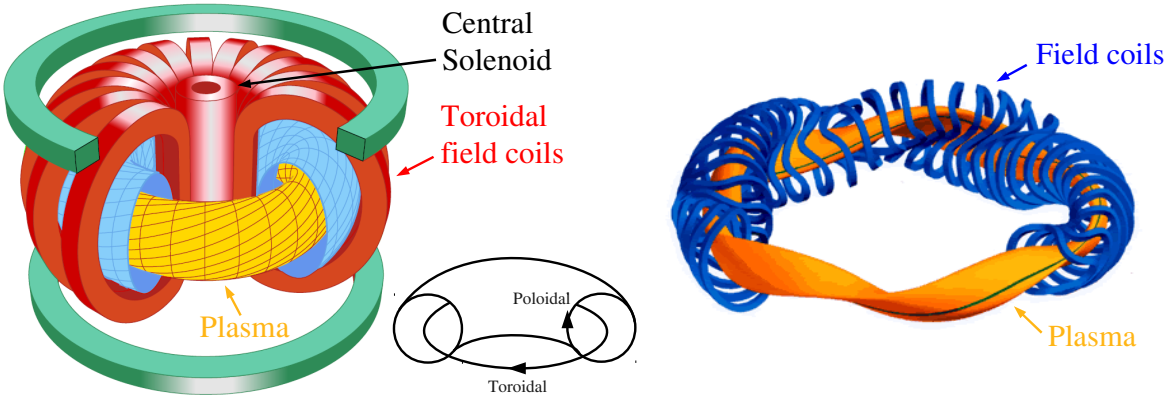
$$n_e T_e \tau_E \geq 10^{21} \frac{\mathrm{keV\ s}}{\mathrm{m}^3}. \tag{1.2}$$

The triple product contains the electron density n_e, electron temperature T_e and confinement time $\tau_E = W/P_l$ (which measures how fast the energy content W of the plasma is lost at a energy loss rate P_l).

[1] Plasma temperatures are commonly given in electron volts (eV). Conversion to Kelvin (K) is done using the Boltzmann constant $k_B = 8.617343 \cdot 10^5 \frac{\mathrm{eV}}{\mathrm{K}}$: $1\mathrm{eV}/k_B = 11604.505\mathrm{K}$.

1.2. Magnetic confinement fusion

Because gravitational confinement as present in the core of stars is technically impossible in experiments, other ways to confine the plasma have to be used. Magnetic confinement exploits the fact that the plasma species (ions and electrons) can be influenced by electric and magnetic fields. The Lorenz force $\vec{F} = q(\vec{E} + \vec{v} \times \vec{B})$ causes charged particles to follow a gyration motion along magnetic field lines. This allows them to move freely along the field line, but movement perpendicular to the field is suppressed. By constructing a magnetic field configuration with closed field lines the plasma can be confined in a magnetic "cage". The challenge is to find a configuration with good confinement properties that is not susceptible to instabilities.



(a) Geometry of a tokamak device. Image: [2]

(b) Geometry of the Wendelstein 7-X stellarator. Image: [1]

Figure 1.1.: Magnetic confinement experiment configurations

The two most advanced magnetic confinement concepts are the toroidal tokamak (figure 1.1a) and stellarator (figure 1.1b) configurations. In a stellarator device the entire magnetic field is formed by external coils. This allows operation in steady-state, but to obtain good confinement properties a complex three-dimensional coil geometry is required. A tokamak device generates only the toroidal component of the magnetic field with external coils. A poloidal component is added by inducing a toroidal current in the plasma using a current ramp-up in the central solenoid of the device, resulting in a combined field with helical field lines winding around the torus. The current ramp-up brings the disadvantage of a pulsed operation mode, making steady-state operation challenging. However, due to their simpler technical design, development of the tokamak experiment line outpaced their stellarator counterparts.

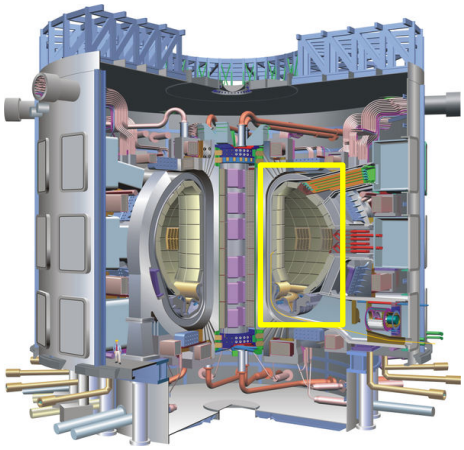
Current tokamak experiments like ASDEX Upgrade [63] and JET [89] operate at a core density of $n_e \approx 10^{20}$ particles/m^3, temperatures of $T_e \approx 20$keV and a confinement time of $\tau_E \approx 0.1$s (ASDEX Upgrade) up to $\approx 1s$ (JET). The goal is to achieve a plasma with an energy gain factor

$$Q = \frac{P_{fusion}}{P_{heating}} = \frac{\text{Fusion energy released}}{\text{Heating energy required}} > 1. \tag{1.3}$$

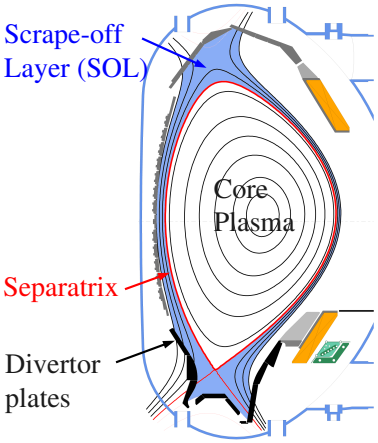
The first experiment which is expected to achieve this is the international experiment ITER [92, 66, 6] (figure 1.2a), located near Cadarache, France and expected to commence operation in 2019. Its goal is to reach $Q \geq 10$ and $P_{fusion} = 500\text{MW}$. It will provide the physics basis for the first real fusion power plant, DEMO [77], which will then demonstrate the feasibility of magnetic confinement fusion as an industrial-scale power source.

1.3. Edge physics

The confinement of the plasma cannot be perfect. Due to collisions and turbulent processes (so-called "anomalous transport") hot plasma is lost from the confinement region and hits the walls of the plasma vessel, where the power has to be dissipated. Acceptable power fluxes to the wall are in the range of 5MW/m^2 for steady-state operations. Higher power loads can damage the plasma facing components due to erosion, leading to the release of impurities which cool the plasma. Controlling power exhaust is therefore of critical importance for confinement quality and machine lifetime.



(a) Design sketch of the ITER tokamak [6]. The position of the toroidal plasma vessel is highlighted.



(b) Geometry of a poloidal cut of the divertor tokamak ASDEX Upgrade [15].

Figure 1.2.: Geometry of a divertor tokamak

An advanced design which avoids direct contact between device wall and core plasma is the divertor configuration (figure 1.2b). The magnetic field is modified by control coils to form a separatrix which separates the core region (with closed field lines) from the scrape-off layer (SOL, the region of open field lines intersecting material surfaces). Plasma passing through the separatrix is diverted towards the target plates at the bottom of the machine where it is neutralized. This moves the region where plasma-wall-interaction is occurring away from the core plasma, resulting in better impurity control and the possibility to reduce the heat flux to the target plates using radiation losses.

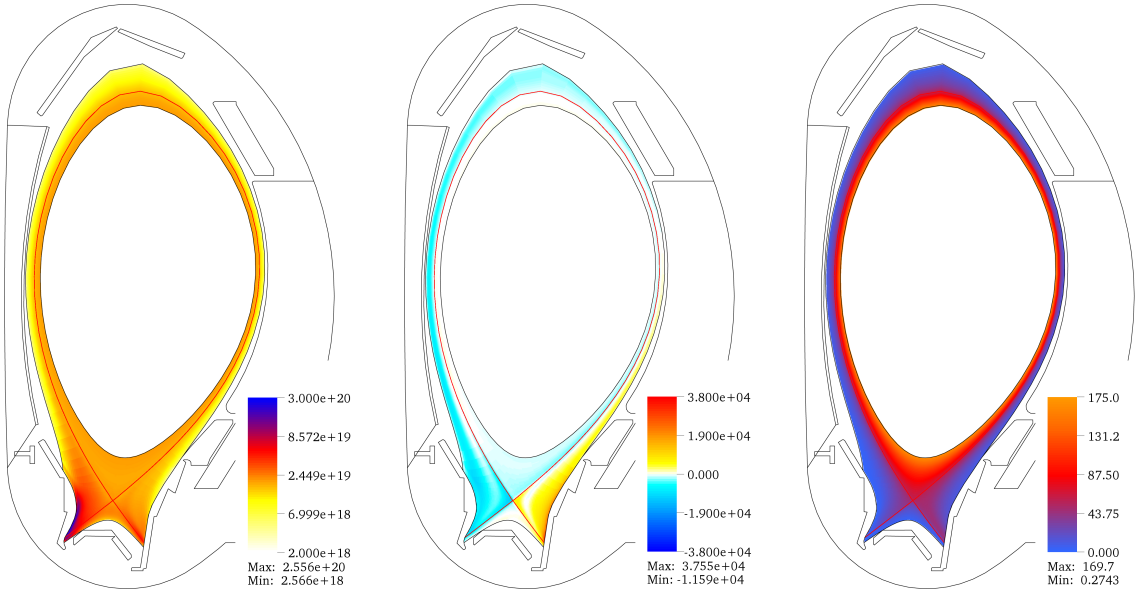
The physics in this edge region is characterized by a complex interplay between plasma and neutrals transport, atomic processes and plasma-wall interaction [106, 101]. Studying this highly nonlinear system in its entirety requires sophisticated tools. This triggered the development of comprehensive computational models during the last decades. At the time this thesis is written, state of the art for SOL physics modeling in tokamaks are two-dimensional models that exploit the toroidal symmetry of the devices. Collisionality in the SOL is usually high enough to justify the use of a fluid model for the ionized plasma species, while the neutral particles are commonly described with a kinetic model. An approach often adopted is to couple a computational fluid dynamics (CFD) code with a kinetic Monte-Carlo code. Codes in this category are B2-EIRENE (aka SOLPS) [102], EDGE2D-NIMBUS [94] and UEDGE [98]. Three-dimensional codes of this type (EMC3/EIRENE [52], BoRiS [30], FINDIF [80]) are used to model stellarators and three-dimensional effects in tokamaks.

1.4. Motivation for this thesis

The SOLPS (*Scrape-Off Layer Plasma Simulation*) suite of codes combines the multi-fluid 2d plasma code B2 [32] with the Monte-Carlo neutrals code EIRENE [96, 95]. Its development is driven by an international collaboration, between the main partners IPP Garching, IEF-4 (Plasmaphysik) FZ Jülich, LIMHP-CNRS Université Paris 13 and St. Petersburg State Technical University. It is used in modeling efforts for several experiments around the world, including design studies for ITER [72, 71, 70].

Recent work on SOLPS has focused on extending the physics model (e.g. inclusion of drift physics [103, 100], improved treatment of kinetic effects in the fluid models [38], wall materials modeling [28, 46]) and use of the code for increasingly complex applications (e.g. modeling of high-Z impurities with a large number (≈ 100) of ion species [28], detachment front physics requiring high spatial resolution [121], time-dependent ELM cycle modeling [44]). Common to all these advances is a continuous increase of computational complexity, causing a growing demand on computational resources and increasing the wall-clock time of simulations (for some applications to an extent that they become infeasible). This can be compensated by using more resources (i.e. parallelization of the code) or by using more efficient numerical methods.

This work focuses on the fluid code B2 in its current version B2.5. A typical plasma solution computed by B2 covering the SOL and a small part of the core is shown in figure 1.3. The fast transport along field lines leads to an equilibration of density, temperature and momentum on closed flux surfaces, resulting in an essentially one-dimensional situation in the core (in fact, one-dimensional models are the de-facto standard for global plasma transport simulations of the core region). It is mainly in the divertor region where, due to atomic physic and plasma-wall interaction, the solution develops a pronounced two-dimensional structure. It would therefore be highly attractive to be able to adapt the spatial discretization to the solution, resulting in a reduction of the degrees of freedom needed to solve a given problem. This is even more the case for the simulation of transient phenomena requiring high localized spatial



(a) Ion density $n_i(\text{particles}/m^3)$ (b) Parallel Ion velocity $u_{\|i}(\text{m/s})$ (c) Electron temperature $T_e(\text{eV})$

Figure 1.3.: Example results for a steady-state simulation performed with B2. The simulation domain covers the SOL and an annulus of the core. Electron density n_e and electron temperature T_e show an essentially one-dimensional structure along the field lines in the core and parts of the SOL and a pronounced two-dimensional structure in the divertor

resolution in changing regions of the domain. For the B2 code this is particularly attractive because its implicit discretization requires the solution of sparse linear systems, the dimension of which is given directly by the number of cells in the domain.

Magnetic confinement exploits the strong anisotropy of particle and energy transport in magnetized plasma, with ratios of transport parallel vs. perpendicular to the field lines reaching values of up to 10^8. This is particularly challenging in numerical simulations because approximation errors can easily dominate effects of the model. In B2 this problem is solved by using field-aligned grids to exactly separate the parallel and perpendicular transport direction. The field-aligned computational grid is created in a pre-processing step. The data structure of B2.5 only supports structured grids, resulting in a global coupling of grid resolution and limiting the possibilities to customize grids for an anticipated solution. During the simulation the grid is fixed, making it impossible for the code to react to changes in the solution.

The focus of this thesis is therefore to

- enable the B2 code to use unstructured field-aligned grids,
- provide mechanisms to adapt the grid to the solution during the simulations and
- reduce the computation time needed to achieve a given solution accuracy.

1.5. Previous work

Adaptive algorithms for discretizations in space and time have been studied extensively in theory and are nowadays in widespread use in many fields of computational science and engineering. Especially in the area of computational gas and fluid dynamics schemes using unstructured grids are steadily progressing since the 1980ies, resulting in the availability of robust adaptive solvers for complex geometries (cf. [53, 7] and references therein).

In the area of plasma edge fluid codes a number of interesting approaches to adaptivity have been developed and tested [109, 122, 21]. However, none of these projects resulted in a code with a physics model as comprehensive as the existing edge modeling packages. An attempt to introduce grid adaptation into SOLPS was started with the SOLPS6 project [27]. The work in this thesis revisits the base grid/working grid approach developed in SOLPS6 and combines it with a modern data structure, more flexible adaptation algorithms and numerical schemes developed for unstructured grid CFD codes.

1.6. Thesis outline

The thesis is organized as follows. Chapter 2 gives a short overview of the B2 multi-species plasma fluid model. The main part of the thesis is then structured into the following three topics.

- **Grids and data structures**. Chapter 3 describes the base grid/working grid approach and the new unstructured grid data structure. Adaptation algorithms for efficient manipulation of the grids are derived. An overview of adaptation criteria and feature detectors is given. The chapter closes with a demonstration of applying the data structure and algorithms to tokamak grids.
- **Numerical methods**. Chapter 4 gives an introduction to the finite volume method and its specific implementation used in the B2.5 code for structured grids. A high-resolution method designed for use on unstructured grids is described in detail, including a short overview of time discretizations. A comprehensive benchmark of the schemes for steady and unsteady situations is presented in chapter 5.
- **Implementation of the B2.6 code**. The last two chapters cover the work done to implement the components described in the previous chapters into the B2 code. Chapter 6 gives an overview of the B2 algorithm and describes the steps necessary for the transition to the unstructured grid. Chapter 7 describes a benchmark of an intermediate step in this transition, the B2.6-structured code.

The thesis is closed by a summary and an outline of planned future work. The appendix contains additional details concerning the numerical schemes, implementation and benchmark cases.

2. The B2 scrape-off layer plasma fluid model

The physics of scrape-off layer plasmas is characterized by a complex interplay of different non-linear physical processes: transport of ionized and neutral particles, ion-ion and ion-neutral interaction, radiation transport and plasma-wall interaction. Exhaustive overviews of the field can be found in [106] and [101].

This complexity necessarily has to be reflected in comprehensive models of the scrape-off layer and their numerical implementations. The SOLPS (*Scrape-off layer plasma simulation*) code package (aka B2-EIRENE) [102] is an example for such a model. It combines the Monte-Carlo neutrals transport code EIRENE [95] with the multi-species plasma fluid code B2 [32]. Typical applications of such a model include modeling of experiments based on diagnostics measurements (mostly for steady-state analysis, but also for time-dependent phenomena), qualitative studies of aspects of the physical system as well as predictive analysis for experiment planning and engineering studies. An example of the latter is use of the SOLPS4 code version to support the ITER design phase [72, 71, 70].

This thesis concentrates on the B2 code in its current version B2.5. In this chapter an overview of the B2 plasma fluid model is given and the geometry of the model is established to motivate the grid generation techniques described in chapter 3.

2.1. Model geometry and coordinate systems

2.1.1. Tokamak Equilibria

The magnetic field $\vec{B}$ in a Tokamak is formed by the superposition of an external field generated by magnetic field coils (which determine the general plasma shape) and a field component originating from currents in the plasma (which are driven mainly by induction using a current ramp-up in the central solenoid). The resulting helical magnetic field lines winding around the torus (cf. figure 1.1a) form nested *magnetic* or *flux surfaces* (cf. figure 2.1).

In the axially symmetric tokamak case the equilibrium magnetic field configuration balancing magnetic force and plasma pressure is found as the solution of the Grad-Shafranov equation

$$-\left(R\frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial\Psi}{\partial R}\right)+\frac{\partial^2\Psi}{\partial z^2}\right)=\mu_0(2\pi R)^2\frac{dp}{d\Psi}+\mu_0^2 I_{pol}\frac{dI_{pol}}{d\Psi}. \tag{2.1}$$

The flux function Ψ denotes the poloidal flux lying inside a magnetic surface. Equilibrium reconstruction based on (2.1) is routinely performed for tokamak experiments using interpretive codes (e.g. CLISTE [79]), with machine parameters (e.g. the coil layout) and various measurements from diagnostics as input and solution constraints.

2.1.2. Coordinate Systems

Due to the strong field alignment of the plasma transport, models for energy and particle transport are best formulated in appropriate field-aligned coordinate systems. Figure 2.1 shows the systems relevant for the B2 model.

① **Cylindrical system** (R, ϕ, z). R is the major radius, ϕ the toroidal direction/angle, z is the height.
② **Parallel system** $(\parallel, \perp, r)$. The parallel direction $\parallel$ is in the direction of the magnetic field $\vec{B}$, $\perp$ is perpendicular to $\vec{B}$ in the flux surface (diamagnetic direction), r outward normal to the flux surface.
③ **Poloidal system** (θ, r, ϕ). θ is tangential to the magnetic surface in the poloidal plane, r is normal to the flux surface in the poloidal plane, ϕ is the angle in the toroidal direction.

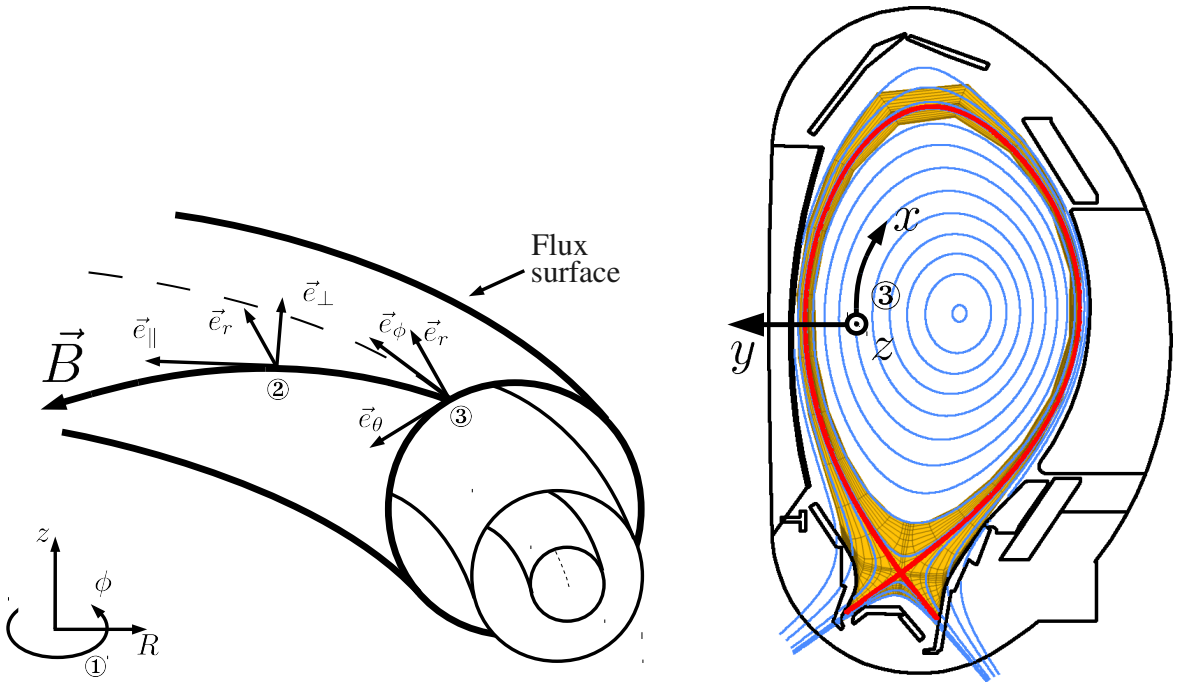
Device geometry data is usually given in the global cylindrical system ①. The transport model uses the field-aligned curvilinear coordinates ② and ③. Note that while figure 2.1a only shows circular surfaces, the basis vectors of the poloidal and parallel systems depend only on the local geometry and are thus defined for arbitrarily shaped surfaces, including geometry singularities like X- and O-points.

The B2 model is derived in the poloidal system, which from now on will be relabeled as $(\theta, r, \phi) = (x, y, z)$ with orientations as shown in figure 2.1b. The coordinate mapping is derived from the magnetic field $\vec{B}$ as obtained from the equilibrium reconstruction. $\vec{B} = (B_x, 0, B_z)^T$ is the magnetic field vector in the poloidal system, the unit vector in the field direction is

$$\vec{b} = \begin{pmatrix} b_x \\ 0 \\ b_y \end{pmatrix} = \begin{pmatrix} B_x/B \\ 0 \\ B_z/B \end{pmatrix}, \quad B = \|\vec{B}\|. \tag{2.2}$$

Vector quantities given along the parallel direction (like the parallel velocities $u_{\parallel a}$) are projected on the poloidal direction by $u_x = \frac{1}{b_x} u_\parallel$ (b_x is the *magnetic field pitch*). $\vec{B}$ is assumed to be fixed (i.e. time-independent).

Relevant properties of curvilinear coordinates are outlined in appendix A.1. The metric coefficients are $h_x = 1/\|\nabla x\|, h_y = 1/\|\nabla y\|$. Due to the rotational symmetry in the toroidal direction $h_z = 2\pi R$ can be used with R being the major device radius (i.e. the distance from the torus axis). All quantities are given in physical units, for vector quantities the physical



(a) ① cylindrical, ② parallel, ③ poloidal

(b) B2 system convention

Figure 2.1.: Global and (local) field-aligned coordinate systems. The B2 model is formulated in the poloidal system with coordinate directions given in sub-figure b, with a typical simulation domain covered by the computational grid marked in orange.

components (i.e. the projection on on the unit basis vectors) are used instead of the co- or contra-variant components.

2.1.3. Model geometry

The model is mainly applied to tokamak devices for which rotational symmetry in the toroidal direction can be assumed. The simulation domain starts on a closed flux surface in the core and usually extends radially outward up to the first limiter structure of the main chamber vessel wall. In the poloidal direction the domain ends at the target plates. The toroidal direction includes the entire torus. A typical simulation domain is shown in the poloidal cut in figure 2.1b, the full three-dimensional domain is shown in figure 2.2. It can be extended to include more of the core region. The cutoff is usually chosen as required for a robust boundary condition definition.

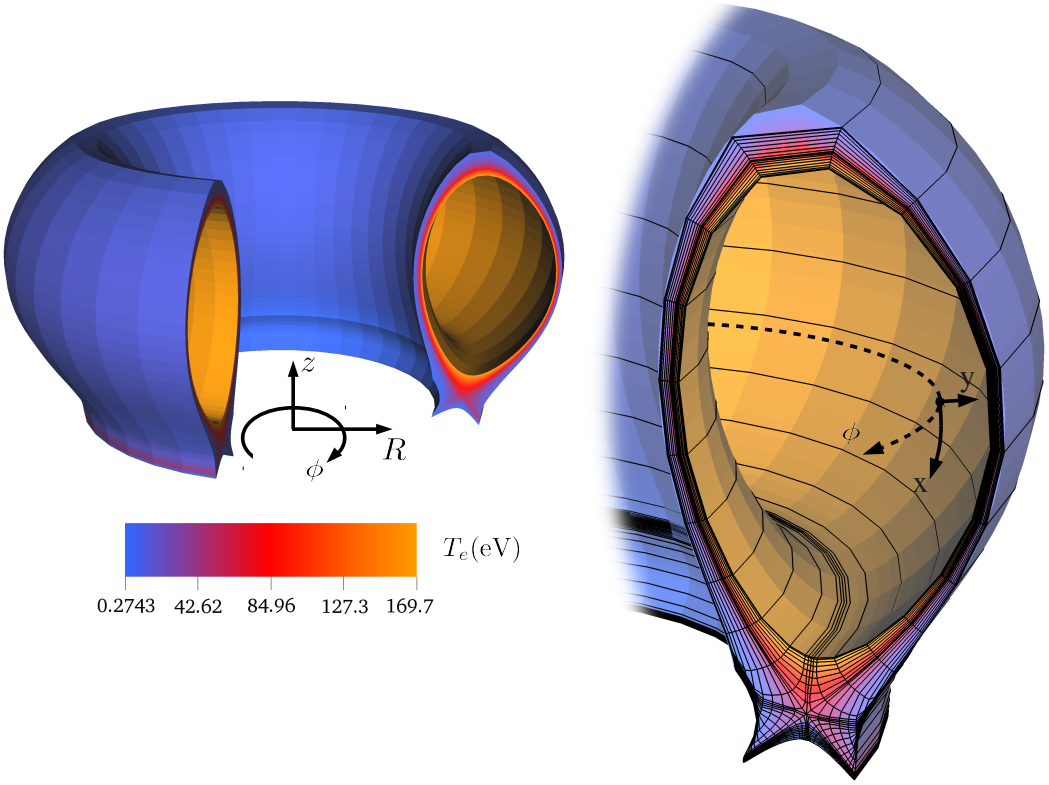


Figure 2.2.: Model geometry of the B2 code. The simulation domain covers an annulus of the core and extends up to the first limiter structure of the wall in the radial direction, in the toroidal direction it is periodic. A grid with 48×18 cells is shown in the plot. Plotted is the electron temperature from the benchmark case in chapter 7.

2.2. B2 multi-fluid model

Under the assumption of high collisionality, a fluid model can be used to describe the plasma in the scrape-off layer. The B2 fluid equations are derived from the Braginskii equations [34], which in turn are obtained by forming moments of the kinetic transport equations. Starting from the original B2 formulation in [31], major development steps of the equation system were the incorporation of electric fields [16] and inclusion of drifts [100, 103, 33].

The multi-fluid model treats every ion species in the plasma as a single fluid phase. Due to the alignment of transport along the magnetic field lines, only the velocity in the parallel direction is kept as a dependent variable. A common temperature for all ions and a separate electron temperature is assumed. The primary dependent plasma state variables are then the particle density $n_a[1/\mathrm{m}^3]$ and parallel velocity $u_{\parallel a}[\mathrm{m/s}]$ of every species a, the ion temperature $T_i[\mathrm{eV}]$, electron temperature $T_e[\mathrm{eV}]$ and the potential $\phi[\mathrm{V}]$.

The equation system consists of coupled nonlinear parabolic partial differential equations of advection-diffusion-reaction type. The general form is similar to the Navier-Stokes equations.

Mass and momentum conservation is stated separately for the individual plasma species. Inertia of electrons is neglected, and only an elliptic current continuity equation is kept. Conservation of energy is stated in separate ion and electron energy equations. For reasons of clarity the equations are given here in a rather general form. For detailed write-ups for different versions of the equations, the full forms of the drift terms, coefficients and source term models the reader is referred to [16, 33, 100, 103, 102, 47][1]. Expansion of the equations in curvilinear coordinates is also not covered here, a detailed treatment of this technical step is given in [16].

B2 model equations

The *continuity equation* for species a is

$$\frac{\partial n_a}{\partial t} + \nabla \cdot \vec{\Gamma}_a = S_a^n \tag{2.3}$$

with the particle flux density

$$\vec{\Gamma}_a = n_a u_{\parallel a} \vec{b} + n_a \vec{V}_a - D_a \nabla n_a - D_a^p \nabla p_a. \tag{2.4}$$

where $\vec{V}_a$ contains drift and anomalous velocities, $p_a = n_a(T_i + Z_a T_e)$ is the partial pressure and D_a, D_a^p are anomalous diffusion coefficients. S_a^n is the particle source term.

The *parallel momentum equation* for species a (with ion mass m_a) is

$$\frac{\partial}{\partial t}(m_a n_a u_{\parallel a}) + \nabla \cdot \vec{\Gamma}_a^m = -\vec{b} \cdot \nabla p_a + \sum_b R_{ab\parallel} + R_{ea\parallel} + F_E + S_a^m. \tag{2.5}$$

On the right-hand side are the pressure term, ion-ion friction $R_{ab\parallel}$, ion-electron friction $R_{ea\parallel}$, force due to the electric field F_E and momentum source terms S_a^m. The viscosity tensor is simplified by omitting cross-field derivatives. The remaining viscosity coefficients η_a are included in the parallel momentum flux density

$$\vec{\Gamma}_a^m = m_a \vec{\Gamma}_a u_{\parallel a} - \eta_a \nabla u_{\parallel a}. \tag{2.6}$$

The *current continuity equation* is

$$\nabla \cdot \vec{j} = 0 \tag{2.7}$$

with the effective current density

$$\vec{j} = \nabla \cdot (\sigma \hat{\vec{E}} - \alpha_e \nabla T_e) + \vec{j}_d \tag{2.8}$$

where σ is the conductivity, $\hat{\vec{E}} = \frac{1}{e n_e} \nabla (n_e T_e) - \nabla \phi$ the modified electric field (e is the electron charge) and α_e the thermo-electric coefficient of the thermal force. $\vec{j}_d$ includes additional drift and viscosity terms.

[1] And the B2.5 source code.

The *Electron energy equation* is

$$\frac{\partial}{\partial t}\left(\frac{3}{2}n_e T_e\right) + \nabla\cdot\vec{Q}_e = -n_e T_e \nabla\cdot(u_{\parallel e}\vec{b}) + \vec{J}\cdot\hat{\vec{E}} - Q_{ei} + S_e^H \tag{2.9}$$

with the electron heat flux density

$$\vec{Q}_e = \left[\frac{3}{2},\frac{5}{2}\right]\vec{\Gamma}_e T_e - n_e\kappa_e\nabla T_e + \alpha_e T_e\hat{\vec{E}}. \tag{2.10}$$

The coefficient κ_e is the electron heat conductivity. The electron flux density and parallel electron velocity are

$$\vec{\Gamma}_e = \sum_a Z_a\vec{\Gamma}_a - \frac{1}{e}\vec{j}, \qquad u_{\parallel e} = \sum_a Z_a\frac{u_{\parallel a}n_a}{n_e} - \frac{1}{en_e}\vec{b}\cdot\hat{\vec{j}} \tag{2.11}$$

where Z_a is the effective charge of species a, $n_e = \sum_a Z_a n_a$ the electron density and $\hat{\vec{j}}$ the current density with the $\vec{j}_d$ term omitted. The square brackets express that the advective part of the electron flux has a prefactor of $\frac{3}{2}$ while the diffusive part has prefactor $\frac{5}{2}$. The terms on the right hand side of (2.9) include electron velocity divergence, Joule heating, classical electron-ion heat exhange term Q_{ei} and heat source S_e^H.

The *ion energy equation* is ($n_i = \sum_a n_a$)

$$\begin{aligned}\frac{\partial}{\partial t}\left(\frac{3}{2}n_i T_i\right) + \nabla\cdot\vec{Q}_i = &- n_i T_i\nabla\cdot(u_{\parallel a}\vec{b}) + \sum_a \eta_{\parallel a}(\nabla_\parallel u_{\parallel a})^2 \\ &+ \sum_{ab} F_{ab}(u_{\parallel b} - u_{\parallel a}) + Q_{ei} + S_i^H\end{aligned} \tag{2.12}$$

with the ion energy flux density ($\Gamma_i = \sum_a \Gamma_a$)

$$\vec{Q}_i = \left[\frac{3}{2},\frac{5}{2}\right]\vec{\Gamma}_i T_i - n_i\kappa_i\nabla T_i. \tag{2.13}$$

As in the electron heat flux (2.10), the advective and diffusive components have different prefactors. κ_i is the ion heat conductivity. The right-hand side includes terms for the ion velocity divergence, friction heating terms including inter-species friction, the heat exchange term and general heat sources.

The most important direct coupling mechanism are particle, heat and momentum exchange due to atomic processes (which are included in the respective source terms, see below), pressure, inter-species friction and the $\nabla u_\parallel$ heating terms. Influence of the electric field is especially important in simulation including drift physics. Proper treatment of these coupling mechanisms is of critical importance in the numerical solution of the system.

Ion species

The number of ion species (and therefore the number of continuity and momentum equations) depends on the situation being modeled. It ranges from 2 species (pure hydrogen plasmas,

e.g. D, D^+) to typical cases including impurity species (carbon, neon, argon, ...) with up to 30 charge states. The situation becomes more challenging for cases including impurities with a high atomic number like tungsten ($Z_a = 74$), which is an attractive material for plasma facing walls due to its high melting point [86, 58]. The code is reported to converge for such cases. However, simulation times increase significantly. Recent work has concentrated on introducing charge-state bundling to reduce the number effective species [28]. Per-species parallelization as outlined in appendix D.2 can also efficiently address this problem.

Transport coefficients

For the exact form of the classical transport coefficients the reader is referred to [34, 17, 120]. In general the coefficients have a (in some cases strongly) nonlinear dependence on the plasma state. The drift terms are covered extensively in [103, 102]. Some kinetic corrections are required for the viscosity η and the parallel heat conductivity $\kappa_{\|}$. Currently this is addressed by the introduction of flux limiters, a more realistic kinetic model is currently under development [43, 40]. In the radial direction experiments show significantly higher transport levels than predicted by classical theory, which are attributed to complex turbulent behavior of magnetized plasmas [120]. However, accurate turbulence models have to resolve a 5-dimensional phase space and require massively parallel codes to achieve sensible simulation times. Transport models like B2 therefore use an anomalous transport ansatz with heuristic models based on coefficients derived from turbulence simulations and experimental data fits. A detailed discussion of this can be found in [68].

Source terms

The source terms S_a (particles), S_a^m (momentum), S_e^H (electron heat) and S_i^H (ion heat) contain nonlinear models for the atomic processes ionization, recombination and charge exchange. They are also used to couple the code to neutrals transport codes (either a simple internal fluid neutrals model or the Monte-Carlo code EIRENE with a comprehensive physics model). Furthermore problem-dependent sources like plasma fueling, external heating and impurity generation can be included.

Boundary conditions

A wide range of physical boundary conditions is used to provide flexibility in the modeling. At the target plates the dynamics of the plasma is governed by the formation of an electrostatic sheath, which accelerates the ions to their local sound speed. At the core boundary particle and energy densities and/or fluxes are prescribed to match conditions for given core plasma parameters, sometimes combined with feedback control to fix specific plasma quantities. At the boundary towards the vacuum region between the simulation domain and the wall the same approach can be taken, or decay lengths can be prescribed.

Boundaries of the simulation domain that do not end at a material surface often require ad-hoc assumptions and pose big challenges for the modeler. Recent efforts aim at coupling the B2 code to specialized transport models for the core and the vacuum region to address the intrinsic uncertainties of this approach [108]. A general framework for this kind of code-code coupling is currently being developed in the context of the Integrated Tokamak Modeling (ITM) EFDA Task Force [5].

3. Grids and data structures

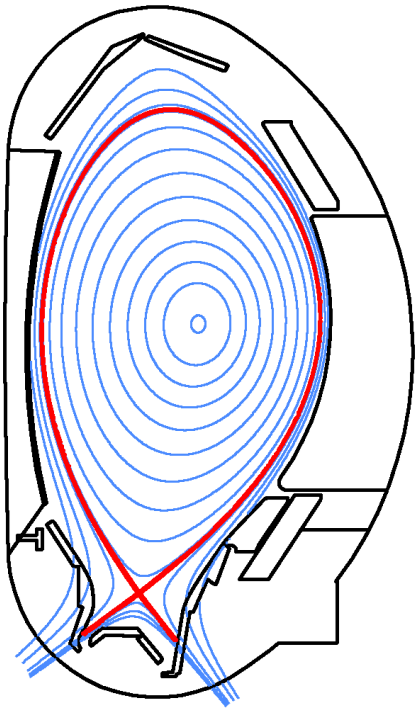
The B2 model equations given in section 2.2 are expanded in the poloidal field-aligned coordinate system to obtain a clean separation of strong parallel and weak radial transport. This allows a robust numerical treatment of the strong transport anisotropy. The disadvantage is that orthogonal field-aligned grids are required for the spatial discretization, making the task of grid generation challenging.

This chapter begins with a description of the grids used by B2 and the grid generation chain currently used to create them. Following this an approach to grid adaptation that maintains field-alignment is presented. The data structure and algorithms needed for its implementation and application are described, including examples to demonstrate their capabilities.

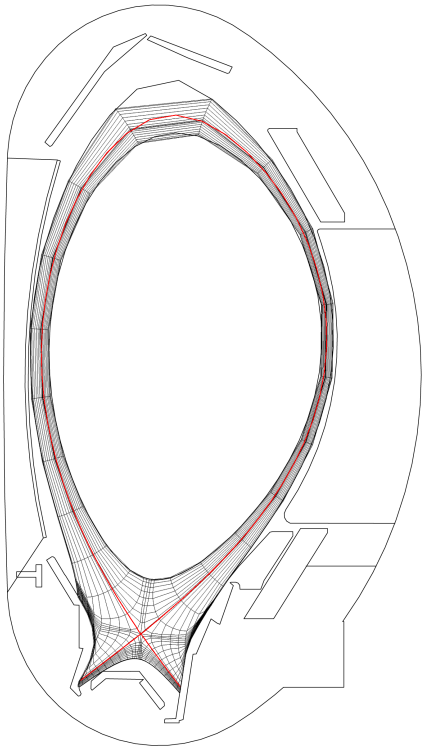
3.1. Grid generation and data structures in B2.5

3.1.1. Field-aligned grids

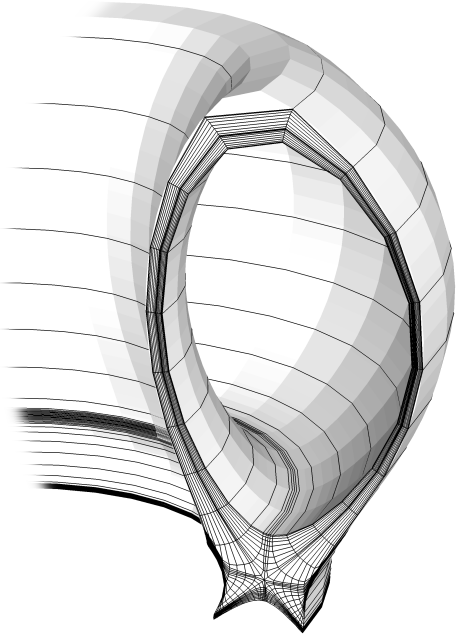
The spatial discretization of the B2 code exploits the toroidal axial symmetry of tokamak devices to reduce the three-dimensional problem to two dimensions. The cells of the grid extend around the torus in the toroidal direction and close on themselves. Projected on a poloidal cut through the torus the cells are quadrilaterals with cell faces either aligned to the magnetic flux surfaces or orthogonal to them. Accurate representation of the topology is ensured by explicit placement of faces and vertices on the separatrix and the x-point. An example grid for the benchmark geometry used in chapter 7 is shown in figure 3.1.



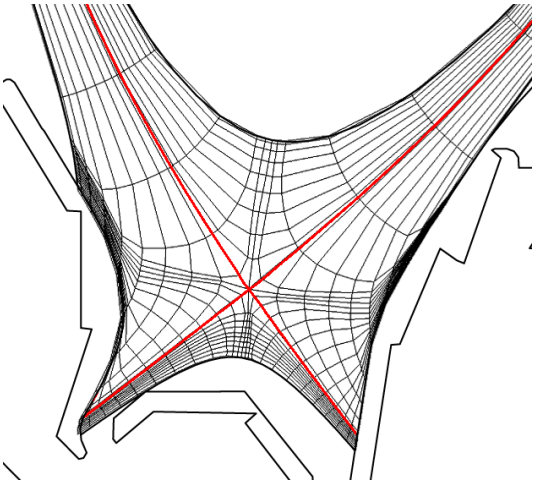
(a) Flux surface structure projected to the poloidal plane. The separatrix is shown in red.



(b) Grid projected on the poloidal plane (48x18 cells)



(c) The actual grid cells extend periodically around the torus



(d) Grid in the divertor region. The x-point and separatrix are resolved explicitly.

Figure 3.1.: Example grid for the ASDEX Upgrade discharge #16151 benchmark geometry.

3.1.2. The B2.5 grid data structure

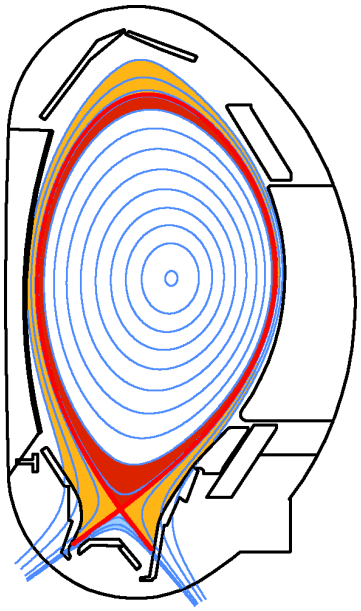
The quadrilateral grid cells in physical space are mapped to unit squares on a Cartesian grid in computational space using local field aligned coordinate mappings.

Depending on the magnetic field configuration the simulation domain is divided in regions. Every region forms a contiguous block of grid cells in computational space. To simplify the data structure, these blocks in computational space are required to be rectangular. Figure 3.2 illustrates the mapping for a so-called "single-null" topology. Similar mappings have been devised for more complex topologies [47, 78].

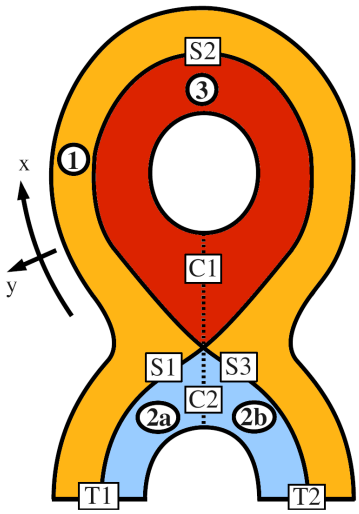
The mapping allows information associated with cells to be stored efficiently in 2-dimensional arrays, identifying every cell $\Omega_{i,j}$ with its position (i, j) in the array. Inside every region the neighbors of a cell can be implicitly identified as $\Omega_{i\pm1,j\pm1}$. However, the x-point singularity leads to non-trivial neighborhood connectivity between cells at region boundaries, expressed by "geometry cuts" in computational space. To simplify support for such nontrivial geometries, the B2.5 code explicitly stores next-neighbor connectivity information for the individual cells.

Classification of B2.5 grids

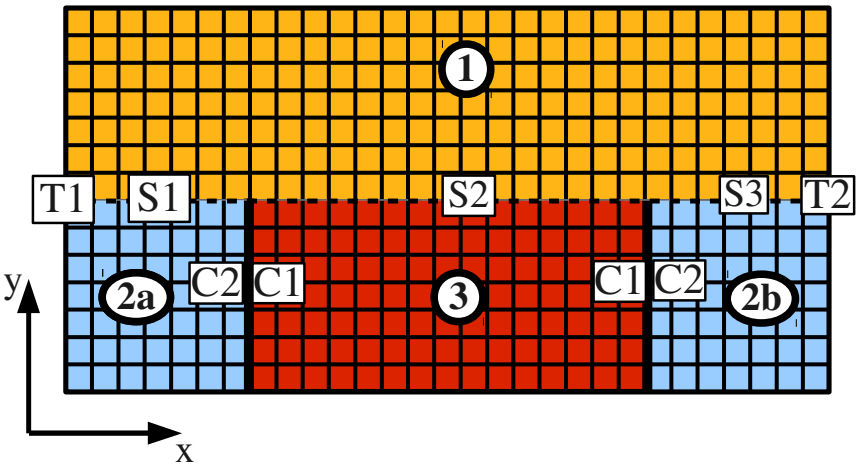
To put the B2 grid in context to existing literature [119, 75], it can be classified as block-structured with a globally unstructured topology. The individual blocks are structured quadrilateral curvilinear grids in physical space that are coordinate-mapped to simple rectangular Cartesian meshes in computational space. The field-alignment can be understood as an extension of boundary-fitting.



(a) Physical space (poloidal cut, real geometry)



(b) Physical space (poloidal cut, simplified geometry)



(c) Grid domain mapped into computational space

Figure 3.2.: Simulation domains in the poloidal cut of a single-null configuration (with one x-point) and its mapping into computational space. The regions are 1. scrape-off layer (SOL), 2. private flux region (PFR) and 3. core. The line segments S1-3 mark the separatrix, T1 and T2 are the inner (left) and outer (right) divertor target plates. C1-2 are "geometry cuts" necessary for the mapping into computational space due to the topology change of the flux surfaces. C1 marks the periodicity boundary in the core, C2 connects the two halves of the PFR.

3.1.3. SOLPS grid generation workflow

For every magnetic field configuration, a specific grid has to be generated. The SOLPS package provides a partially automated grid generation workflow (cf. figure 3.3) for this task. A discretized equilibrium magnetic field configuration is obtained by import tools depending on the data acquisition infrastructure of the experiment. A GUI editor (DG) is used to combine the equilibrium and a model of the physical structure of he experiment vessel and to set up input files for the grid generator. The actual grid generation is done by the CARRE code [78]. When a satisfactory grid is obtained, the grid description is converted to the input format expected by the B2.5 code.

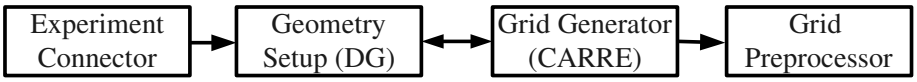
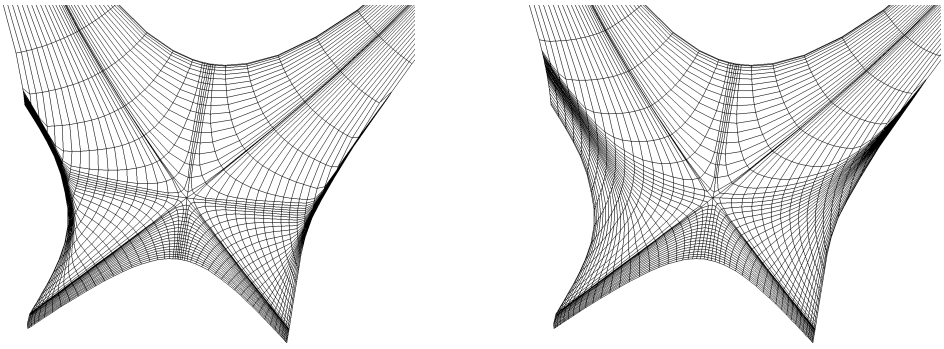


Figure 3.3.: SOLPS grid generation workflow

The grid generation itself can be quite challenging. The generator must automatically identify the topology of the field lines and implement the field-alignment and orthogonality constraints. In addition to this, the limitation of the B2.5 data structure to rectangular blocks of grid cells in computational space leads to problems at the target plates. In general, magnetic field lines do not intersect the target plates at right angles (steep inclinations of the plates are specifically used to increase the plasma contact area in the scrape-off layer). Some field lines therefore terminate earlier than others, but the number of grid points along a line is effectively fixed by the data structure. The grid generator has to relax the orthogonality constraint close to the target plate controlled by user-defined parameters. For a given geometry therefore different grids can be created. Two possible divertor solutions for the #16151 benchmark geometry are shown in figure 3.4.



(a) Strict orthogonality (b) Relaxed orthogonality

Figure 3.4.: Two possible solutions for the grid in the divertor region. Grid maintains orthogonality but exhibits strong "bunching" of grid points close to the target plates. Grid b relaxes orthogonality to partially avoid this.

3.1.4. Approximation of geometric quantities

The geometry of the grid is defined by the position of the four vertices of every cell in physical space and the magnetic field at these points (see figure 3.5). Approximations to all geometric quantities required in the code are derived from this information.

Face areas and cell volume are computed taking the toroidal symmetry into account. The scale factors are approximated as the face midpoint distances

$$h_x = \|\overrightarrow{x_W x_E}\|, \quad h_y = \|\overrightarrow{x_S x_N}\|.$$

The cell volume in the toroidally symmetric case is given by

$$|\Omega_i| = 2\pi \; c_R \; A_{\Omega_i},$$

where c_R is the radial distance of the cell centroid C to the torus axis and A_{Ω_i} the area of the quadrilateral $x_1x_2x_3x_4$. Efficient formulas of geometric quantities for quadrilaterals are described in [119]. Additionally, correction factors are used in the code when computing lengths in the radial direction to account for grid non-orthogonality.

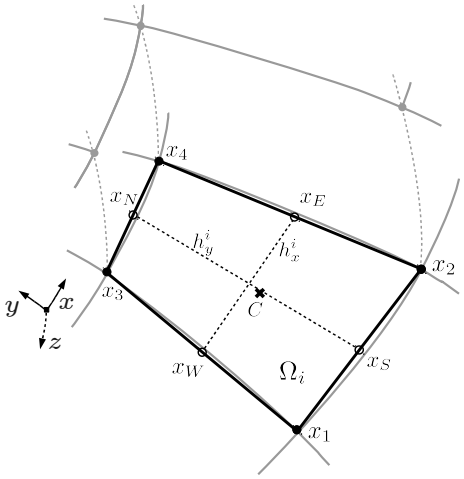


Figure 3.5.: Approximation of geometric quantities in coordinate-mapped grid cells.

3.2. Adaptive field-aligned grids

As outlined in the previous section, grid generation for B2 is a complex task usually requiring user intervention, especially to judge and optimize grid quality. While the grid workflow is very advanced with respect to support of varying experiment geometries, it is not well suited to specifically adapt grids to individual solutions. Because grid generator and solver are separate codes, this currently requires an iterative approach driven manually by the user, with only a very limited number of parameters to influence the grid generator. Therefore an alternative approach to grid adaptation is required.

3.2.1. The base grid / working grid method

The "base grid / working grid" method [27] reuses the existing grid generation workflow. For a given geometry, a high-resolution base grid is generated. The actual simulation is then performed on a working grid (adaptively) constructed at runtime by combining the cells of the base grid into composite cells. An example is shown in figure 3.6.

A composite cell Ω_i is defined by its *stencil* S_i, which enumerates all cells in the base grid it consists of. The restriction is imposed that every face of a composite cell is aligned along a coordinate line, simplifying the shape of stencils to rectangles in the computational domain[1]. A stencil can thus be written compactly as

$$S_i = ((o_x^i, o_y^i)(d_x^i, d_y^i)). \tag{3.1}$$

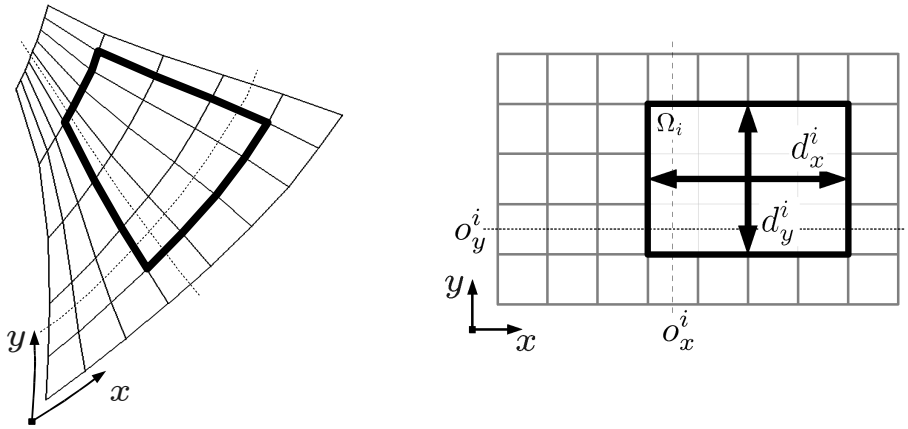
Grids of this type are also called "logically rectangular".

The approach offers two distinct advantages. First, the existing grid workflow (including input/output routines and base grid data structure) can be reused. Second, field-alignment of the working grid is automatically maintained. The disadvantage is that solver and grid generator are still decoupled, making changes to the base grid impossible. This means that the maximum possible resolution is fixed to the base grid level. Furthermore, any grid defects present in the base grid directly carry over to the working grid. The maximum accuracy obtainable is therefore still defined by the quality of the grids provided by the grid generator.

3.2.2. Working grid data structure

While the B2.5 data structure can still be used to store the base grid, its limitation to structured grids is prohibitive to using it for storing adaptive grids. For the working grid a more flexible data structure is required. With the restriction to logically rectangular grids, construction of the working grid can be considered in the context of Cartesian grid adaptation, a topic which has been covered extensively in the literature.

[1] This restriction is to reduce the complexity of data structure and adaptation algorithms. The finite volume methods described in chapter 4 can handle arbitrarily shaped cells.



(a) Physical space (b) Computational space with stencil

Figure 3.6.: Composite cell Ω_i in physical space represented by stencil S_i in computational space

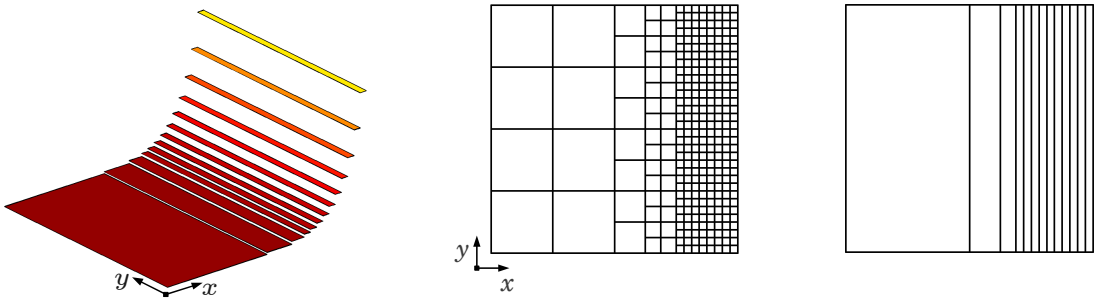
Design goals

Recalling the motivations given in 1.4, a number of key requirements on the working grid data structure emerge which can be summarized under the aspects *flexibility* and *efficiency*.

1. **Flexibility**
   a) The multi-scale character of the physics requires rapid local variations in resolution, spanning a wide range of resolution levels in one working grid.
   b) The strongly anisotropic transport leads to pronounced differences in the solution structure in radial and poloidal direction, requiring anisotropic grid adaptation.
2. **Efficiency**
   a) Residual computation and system assembly in the finite volume solver require efficient access to neighbor connectivity.
   b) Local modifications of the grid should be relatively cheap.

Anisotropic grid adaptation

Anisotropic adaptation [22, 55, 84, 115] allows changes of the cell aspect ratio and thus decoupling of grid resolution in the different coordinate directions, which is of key importance to effectively reduce the number of grid cells when the dynamics of a system change depending on coordinate direction. A well-known example for this is boundary layer formation in fluid dynamics problems. Figure 3.7 visualizes the problem: the restriction to isotropic grids inherently couples the grid resolution between spatial dimensions.



(a) Solution (b) Isotropic grid (c) Anisotropic grid

Figure 3.7.: Grids adapted to a boundary-layer type solution. The isotropic grid (with fixed cell aspect ratio) inherently couples grid resolution between coordinate directions.

3.2.3. Data structure choice

Two suitable data structure designs are found in the literature. Specifically designed for logically rectangular grids is the so-called AMR (Adaptive Mesh Refinement) method [24]. It uses nested patches of block-structured grids which are organized in a tree data structure. Due to the simple structure of the individual patches AMR can be used to construct very efficient explicit solvers. A disadvantage is that deducing connectivity information at patch boundaries requires traversal of the hierarchy tree or explicit storage. Furthermore application of AMR to anisotropic adaptation is nontrivial because for cells with varying aspect ratio no unique grid hierarchy can be defined.

A more general approach is to assume that the working grid is completely unstructured, as is e.g. commonly done for triangular grids. The individual components of the grid (cells, faces, vertices) are identified explicitly using a numbering convention. Geometry and connectivity information for all objects is stored in a data structure consisting of essentially one-dimensional lists. This approach is abbreviated as “UG” (unstructured grid) data structure.

For this work the UG structure was chosen. The main point is trivial availability of the face-cell connectivity information, which is very important due to the concentration of the used finite volume discretization on face fluxes.

In terms of software development effort, while the UG structure is simpler in design than AMR, atomic grid operations and the global adaptation algorithms are more complex (reflecting the higher flexibility of the structure). Local modifications to the grid also are cheaper in AMR, as changes are ideally contained to subtrees of the grid, while a change of the UG structure affects all connected grid objects. However, in the B2 code design the overhead imposed by grid adaptation is negligible compared to the time spent in the solver (especially for the steady-state case).

The use of complex data structures introduces performance penalties when compared to

implementations using simple structured grids. This has to be compensated by advantages of the adaptive discretization. Because the B2.5 code already explicitly stores connectivity information to support complex magnetic topologies (c.f. section 3.1.2), in this case the UG data structure does not add additional overhead (but may offer possibilities to optimize memory access patterns, c.f. appendix B.4).

3.2.4. Unstructured grid data structure description

The grid $\mathcal{T} = (\Omega_i, e_j, V_k)$ consists of cells Ω_i, faces e_j and vertices V_k. Objects are identified by an index, which is subsequently used to store and access data associated with the objects. The information stored for each object is split into the categories *stencil*, *connectivity* and *geometry*.

Stencil

For every cell Ω_i, the stencil $S_i = ((o_x^i, o_y^i)(d_x^i, d_y^i))$ as described in section 3.2.1 is stored. It is important to note that when the stencil data is known, all connectivity and geometry data can be recovered from the underlying base grid. This allows the remaining structure to be reduced to the data necessary for the formulation of the adaptation algorithms and efficient implementation of the solver.

Connectivity

For every grid object, lists are stored containing the indices of connected objects. For reasons of efficiency the size of these lists is fixed, limiting the range of grids that can be represented in the UG structure. Every cell is defined by its four corner vertices. Cell faces on each side can be split into two sub-faces, i.e. a cell can have four to eight faces and face-neighbors. An overview of the stored connectivity information is given in figure 3.8 and table 3.1.

Object	Connectivity data
Cell Ω_i:	Indices of faces ($e_{\Omega,i}^{\mathrm{Top},1}$ - $e_{\Omega,i}^{\mathrm{Left},2}$) and vertices ($V_{\Omega,i}^{\mathrm{A}}$ - $V_{\Omega,i}^{\mathrm{D}}$).
Face e_j:	Indices of the cells in increasing ($\Omega_{F,j}^{\mathrm{inc}}$) and decreasing ($\Omega_{F,j}^{\mathrm{dec}}$) coordinate direction, indices of start ($V_{F,j}^{\mathrm{From}}$) and end ($V_{F,j}^{\mathrm{To}}$) vertices.
Vertex V_k:	Indices of up to four incident faces ($e_{V,k}^{\mathrm{Top}} \dots e_{V,k}^{\mathrm{Left}}$) and cells ($\Omega_{V,k}^{\mathrm{A}} \dots \Omega_{V,k}^{\mathrm{D}}$).

Table 3.1.: Connectivity information stored in the UG data structure for individual objects

Individual entries of the cell, face and vertex connectivity lists can stay undefined depending on the grid structure. In cases where connectivity information can be stored in more than one

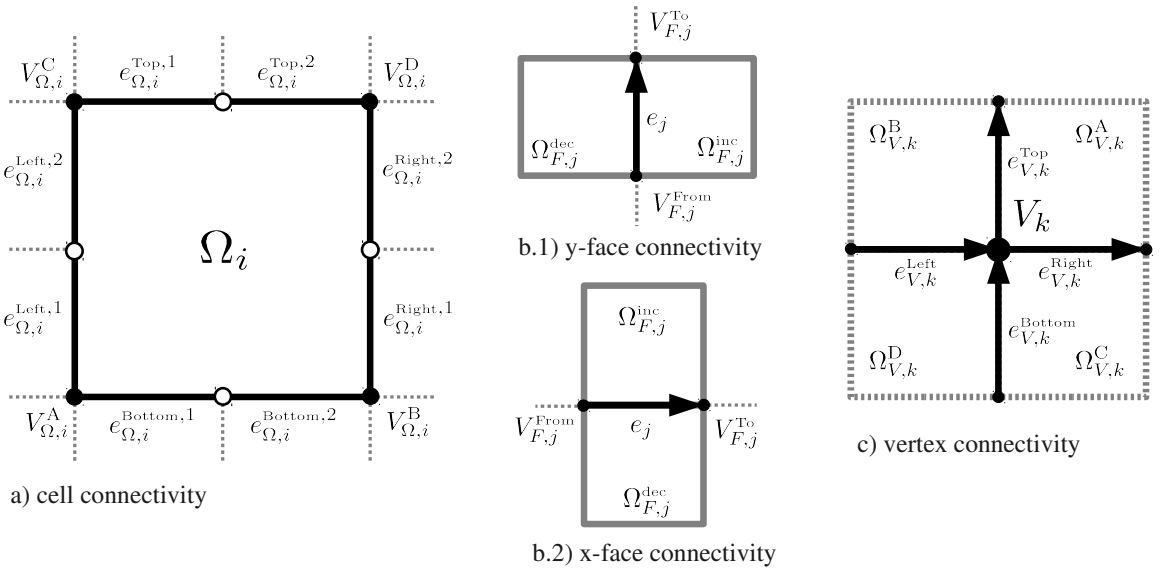


Figure 3.8.: Object connectivity stored in the UG data structure

list entry, to avoid ambiguities only the entry for the object closer to the coordinate origin is set.

Geometry

The main geometric quantities are given in table 3.2. Face areas and cell volumes for the composite objects are computed by simply summing up the base grid quantities. The scale factors are computed as a harmonic average

$$h^c_x = \frac{1}{d^c_x}\sqrt{|\Omega_c|\left(\sum_{j=o^c_y}^{o^c_y+d^c_y-1}\sum_{i=o^c_x}^{o^c_x+d^c_x-1}\left(h^2_{x,(i,j)}/|\Omega_{(i,j)}|\right)^{-1}\right)^{-1}}. \tag{3.2}$$

All other distance measures are then derived from the scale factors and the stencil sizes. All geometric quantities are stored in physical units, vectors are given in physical units relative to the field-aligned unit base vectors. For the faces a simplification is introduced exploiting the field alignment. Instead of the normal vector of the face an orientation flag is stored which indicates whether the face is aligned along a x(poloidal)- or y(radial)-coordinate line. A face aligned to an x-coordinate line is called x-face, a face aligned to an y-coordinate line is called y-face.

In addition to the main connectivity and geometry information, extended properties of grid objects are stored. These include ghost cell and boundary flags, persistence flags (marking a face irremovable) and region numberings to identify sub-blocks of the grid.

Object	Geometry data
Cell Ω_i:	Volume $\lvert\Omega_i\rvert$, scale factors h_x^i, h_y^i, magnetic field vector $\vec{B}_i$
Face e_i:	Area $\lvert e_i\rvert$, Orientation flag (x or y aligned), vector from face midpoint to centroid of cell in increasing ($\vec{m}_i^{\text{inc}}$) and decreasing ($\vec{m}_i^{\text{dec}}$) coordinate direction normal to face, vectors connecting centroids of incident cells $\vec{c}_i$

Table 3.2.: Geometry information stored in the UG data structure for individual objects

3.2.5. Atomic grid operations

Definition 1 (**Atomic grid modifications**). *A* grid modification *consists of a set of* atomic grid operations*. An atomic operation is either a* cell split/face insertion *or a* cell merge/face deletion*. A set of cells that coalesces into one parent cell is called a* merge group*, the resulting cells of a split operation are called a* split group*.*

An atomic grid operation affects several grid objects simultaneously. The range of possible atomic operations determines how complex the change done to the grid in one modification can be and therefore how many modifications are necessary to reach a given grid configuration.

The atomic operations possible for the UG structure are summarized in figure 3.9. Both isotropic operations a,d and anisotropic operations c,e can be combined in the same grid modification. Operation b was included to allow more efficient coarsening for anisotropic unstructured grids[2].

[2]While every isotropic operation can in principle be performed by a succession of anisotropic operations, such an approach requires successive grid modifications to execute one isotropic grid operation, severely complicating algorithm implementation.

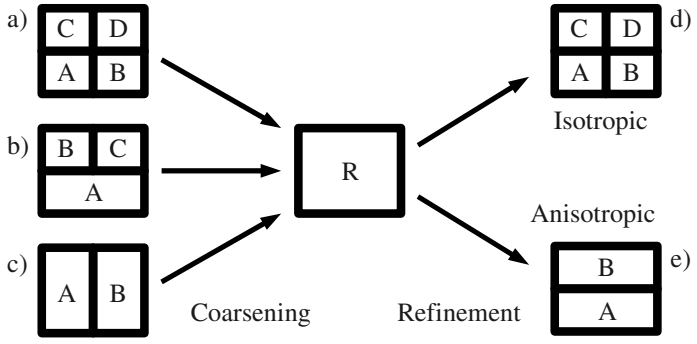


Figure 3.9.: Possible atomic grid operations for the UG data structure. The modifications are presented qualitatively, all operations can be performed for their rotated equivalents and cells with arbitrary aspect ratios

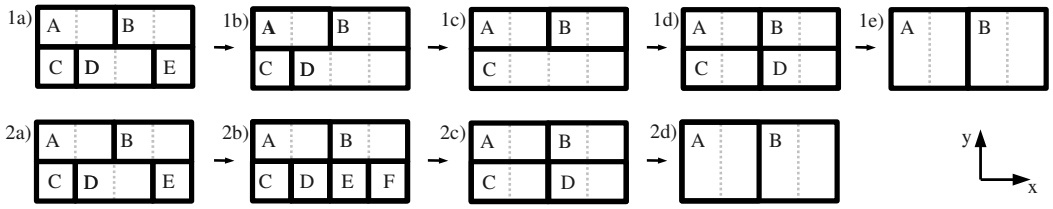


Figure 3.10.: Example of a locked (non-admissible) grid

3.2.6. Representable and admissible grids

The fundamental limitations of the UG data structure are summarized in the following definition.

Definition 2 (**Representable grid**). *A grid is called* representable *(i.e. can be represented in the UG data structure) if it satisfies the following rules.*

1. Cell shape rule*:*
   Cells are logically rectangular with a minimum stencil size of $(d_x^i, d_y^i) = (1, 1)$.
2. Sub-face rule*:*
   Every cell has a maximum of two neighbors / sub-faces in every direction.

However, this definition allows grids incompatible with the design of efficient and robust grid adaptation algorithms. Figure 3.10 demonstrates the problem. Grid 1a is representable, but merging cells in the y direction leads to non-rectangular cells and therefore direct coarsening in the y-direction is not possible. To reach grid state 1e with atomic operations, a combination of coarsening and refinement operations is necessary. Unless the adaptation algorithm can plan several grid modifications at once (making it excessively complex), the grid is stuck in this state.

Grid hierarchies

Problematic situation like the one in figure 3.10 can occur whenever the stencils of connected cells do not align on at least one boundary perpendicular to their common face. This can be avoided by imposing an additional global cell alignment rule. A simple but effective rule is given by the following definition.

Definition 3 (**Admissible stencils and grids**). *Let* $S_i = ((o_x^i, o_y^i), (d_x^i, d_y^i))$ *be the stencil of cell* Ω_i*. Both stencil and cell are called* admissible *if the following statements hold:*

1. *The stencil sizes are powers of two, i.e.*

$$(d_x^i, d_y^i) = (2^a, 2^b), \qquad \text{with } a, b \in \mathbb{N}.$$

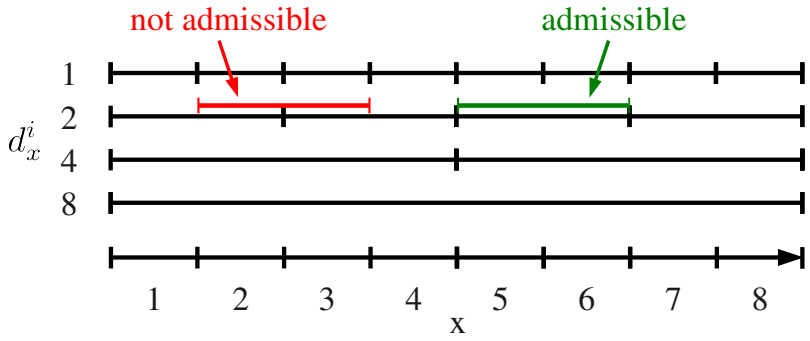


Figure 3.11.: Grid spacing hierarchy in one dimension defining admissible and non-admissible cell stencils.

2. *The stencil origin in a dimension is a multiple of the stencil size in this dimension, i.e.*

$$(o_x^i - 1, o_y^i - 1) = (\alpha d_x^i, \beta d_y^i), \qquad \text{with } \alpha, \beta \in \mathbb{N}.$$

Otherwise it is called non-admissible*. If the stencils of all cells Ω_i in a grid are admissible, the grid is called* admissible*. Furthermore, atomic operations and grid modifications resulting in an admissible grid are likewise called admissible.*

Admissible grids effectively impose a hierarchy of nested grid spacings in every coordinate direction with a resolution ratio of 2:1 between consecutive hierarchy levels (see figure 3.11) by prescribing the possible alignments for a cell with a given stencil size.

Refining/splitting a cell in one direction creates two child cells with half the stencil size in that dimension. For a split in the y direction as sketched in figure 3.9e, splitting cell R in cells A+B, this can be written as

$$S_R = ((o_x^R, o_y^R), (d_x^R, d_y^R)) \xrightarrow{\text{y split}} \begin{cases} S_A = ((o_x^R, o_y^R), (d_x^R, d_y^R/2)) \\ S_B = ((o_x^R, o_y^R + d_y^R/2), (d_x^R, d_y^R/2)) \end{cases}$$

The stencils for anisotropic splits in the x direction and isotropic splits are derived likewise.

For a set of cells $\Omega_1 \dots \Omega_n$ being merged, the stencil of the parent cell Ω_R is given by

$$\Omega_R((o_x^R, o_y^R), (d_x^R, d_y^R)) = \left(\left(\left\lfloor \frac{o_x^i}{2d_x^{\min}} \right\rfloor \cdot 2d_x^{\min} + 1, \left\lfloor \frac{o_y^i}{2d_y^{\min}} \right\rfloor \cdot 2d_y^{\min} + 1 \right), \left(2d_x^{\min}, 2d_y^{\min}\right) \right). \quad (3.3)$$

where $\lfloor \cdot \rfloor$ is the floor function and $d_{x/y}^{\min}$ is the minimum stencil size involved in the respective dimension, defined by

$$d_x^{\min} = \min_{i\,=\,1\dots n} d_x^i, \quad d_y^{\min} = \min_{i\,=\,1\dots n} d_y^i.$$

A merge operation can only be performed if for all involved cells (3.3) resolves to a common parent cell (this fact is actually used later in the adaptation algorithm to identify merge

groups). Adhering to this rule imposes the power-of-two hierarchy on the working grid in every coordinate direction. Choosing $d_{x/y}^{\min}$ to define the hierarchy level makes sure that only cells with the same stencil size in the merge direction are combined. The rule works for all three merge cases a)-c) in figure 3.9.

3.3. Grid adaptation algorithms

The grid adaptation process is controlled by *adaptation criteria* (denoted by $\mathcal{C}$) which are designed to eventually produce a grid that is optimized for a certain purpose. At the same time, only modifications to the grid that a) can be cast into atomic operations and b) result in an admissible grid can be performed, which severely restricts the range of possible modifications. It is the task of the grid adaptation algorithm to find a grid modification that matches the intention of the criteria. The resulting grid modification is summarized in a list of *action flags* $\mathcal{M} = (a^i_{\mathrm{cv}}, a^j_{\mathrm{fc}})$, where

$$a^i_{\mathrm{cv}} = \begin{cases} \textit{empty} \\ \textit{x-split} \\ \textit{y-split} \\ \textit{x+y-split} \end{cases} \quad \forall \text{ cells } \Omega_i \quad \text{and} \quad a^j_{\mathrm{f}} = \begin{cases} \textit{keep} \\ \textit{remove} \end{cases} \quad \forall \text{ faces } e_j. \tag{3.4}$$

The general procedure for a grid adaptation cycle is given by algorithm 1.

Algorithm 1: Grid adaptation cycle

Input: Grid $\mathcal{T}_{\mathrm{in}}$, Criteria values $\mathcal{C}$
Output: Grid $\mathcal{T}_{\mathrm{out}}$, grid modification $\mathcal{M}$

$\mathcal{M}$ = Evaluate action thresholds($\mathcal{C}$, strategy)
$\mathcal{M}$ = **ExtendCellSplits** ($\mathcal{T}_{\mathrm{in}}$, $\mathcal{M}$, strategy)
$\mathcal{M}$ = **ExtendFaceRemovals** ($\mathcal{T}_{\mathrm{in}}$, $\mathcal{M}$, strategy)
$\mathcal{T}_{\mathrm{out}}$ = Apply modification($\mathcal{T}_{\mathrm{in}}$, $\mathcal{M}$)

3.3.1. Criteria values

Criteria are conceptually understood to measure a local error that can be reduced by locally increasing the grid resolution, i.e. spitting cells. The criteria evaluation assigns criteria values to the cells and faces, producing a criteria value list $\mathcal{C} = (c^i_{\mathrm{c},x/y}, c^i_{\mathrm{f}})$.

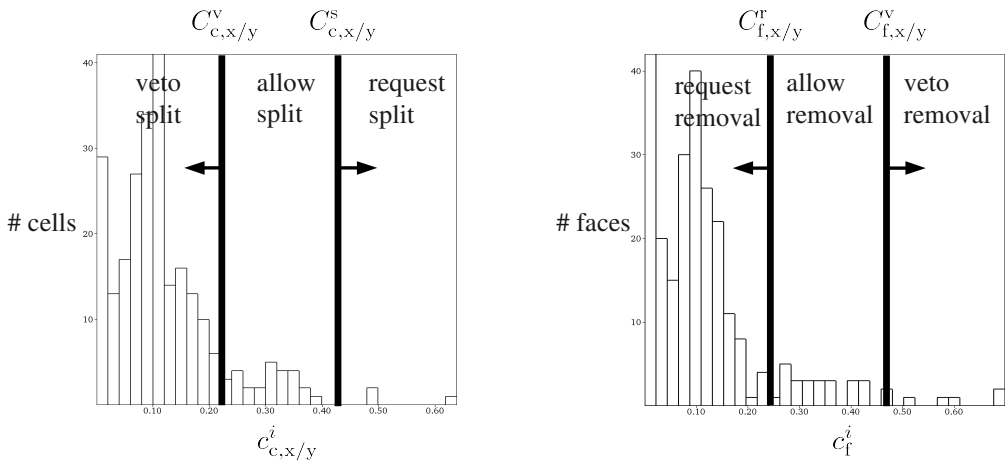
For cells, one value is defined per coordinate direction (designated $c^i_{\mathrm{c},x}$ and $c^i_{\mathrm{c},y}$ for cell Ω_i in the x- and y-direction respectively). For faces, only one value (designated c^i_{f} for face e_i) is given.

Grid modification actions are derived from the criteria using thresholds. *Action thresholds* define for what value ranges an action is required: cells with a high "error" are marked to be split, faces with a low "error" are marked to be removed. Additionally a *veto threshold* can be used to veto the same actions: cells with a low "error" must not be split, faces with a high "error" must not be removed. For objects with criteria values between the action and veto thresholds, the adaptation algorithm can choose to apply actions if necessary. The

Object	Rule	Action
Cell	$c^i_{\mathrm{c},x} \geq C^{\mathrm{s}}_{\mathrm{c},x}$	mark for split in x direction
	$c^i_{\mathrm{c},y} \geq C^{\mathrm{s}}_{\mathrm{c},y}$	mark for split in y direction
	$C^{\mathrm{v}}_{\mathrm{c},x} \leq c^i_{\mathrm{c},x} < C^{\mathrm{s}}_{\mathrm{c},x}$	allow split in x direction
	$C^{\mathrm{v}}_{\mathrm{c},y} \leq c^i_{\mathrm{c},y} < C^{\mathrm{s}}_{\mathrm{c},y}$	allow split in y direction
	$c^i_{\mathrm{c},x} < C^{\mathrm{v}}_{\mathrm{c},x}$	veto split in x direction
	$c^i_{\mathrm{c},y} < C^{\mathrm{v}}_{\mathrm{c},y}$	veto split in y direction
x-Face	$c^i_{\mathrm{f}} \leq C^{\mathrm{r}}_{\mathrm{f,x}}$	mark x-face for removal
	$C^{\mathrm{r}}_{\mathrm{f,x}} < c^i_{\mathrm{f}} \leq C^{\mathrm{v}}_{\mathrm{f,x}}$	allow removal of x-face
	$c^i_{\mathrm{f}} > C^{\mathrm{v}}_{\mathrm{f,x}}$	veto removal of x-face
y-Face	$c^i_{\mathrm{f}} \leq C^{\mathrm{r}}_{\mathrm{f,y}}$	mark y-face for removal
	$C^{\mathrm{r}}_{\mathrm{f,y}} < c^i_{\mathrm{f}} \leq C^{\mathrm{v}}_{\mathrm{f,y}}$	allow removal of y-face
	$c^i_{\mathrm{f}} > C^{\mathrm{v}}_{\mathrm{f,y}}$	veto removal of y-face

Table 3.3.: Object action rule definitions. A x-face is aligned along a x-coordinate line, a y-face is aligned along a y-coordinate line.

detailed threshold rules are summarized in table 3.3, but are clearer to understand when looking at histograms of criteria values as shown in figure 3.12. Obviously $C^{\mathrm{v}}_{\mathrm{c},x/y} \leq C^{\mathrm{s}}_{\mathrm{c},x/y}$ and $C^{\mathrm{r}}_{\mathrm{f,x/y}} \leq C^{\mathrm{v}}_{\mathrm{f,x/y}}$ have to be satisfied. Furthermore, to give meaningful results, the intention of face and cell criteria must not contradict. For closely related face and cell criteria sharing a common normalization, this means $C^{\mathrm{r}}_{\mathrm{f,x/y}} < C^{\mathrm{s}}_{\mathrm{c},x/y}$.



(a) Cell criterion histogram with x/y split thresholds

(b) Face criterion histograms with face removal thresholds

Figure 3.12.: Criteria histograms with action thresholds. Some remarks on the dynamics of error histograms following a sequence of grid modifications and their implications for the design of adaptation strategies can be found in [8].

How split requests for cells are set can be influenced by choosing a *cell split strategy*. The possible choices are:

- **Anisotropic cell split strategy**: the rules in table 3.3 are followed, splits in x and y direction are requested separately.
- **Isotropic cell split strategy**: an isotropic split (in both x and y direction) is requested already if a cell exceeds the split threshold in one direction only.

3.3.2. Adaptation algorithm

An action resulting from evaluation of the action thresholds has to be tested, anticipating the changes it causes to see whether it can be performed by atomic operations and results in an admissible grid. If not, depending on a given strategy the adaptation algorithm can add actions to the modification in a *modification extension* step to make it admissible. If this fails the action is rejected.

To simplify the algorithm, analysis of cell and face actions is done in separate steps. The order of these steps is important, because only the analysis done in the second step can take the results of the first step into account. The choice made here is that cell splits take precedence over face removals. This is motivated by the fact that (assuming that high grid resolution correlates with low errors) ensuring sufficiently high grid resolution takes precedence over reducing the number of grid objects (and thus the computational complexity of the solver).

In the following algorithms a shorthand notation is used to identify groups of objects defined by the local connectivity. Some examples are given in table 3.4. Some repeating and non-essential parts of the algorithms are omitted for brevity.

Shorthand	Meaning
$\mathcal{F}_{\Omega,i}$	All faces connected to cell Ω_i
$\mathcal{F}^{\text{Top}}_{\Omega,i}$	All top faces connected to cell Ω_i
$\mathcal{F}^{\text{x-aligned}}_{\Omega,i} = \partial\Omega_j$	All x-aligned faces connected to cell Ω_i
$\mathcal{C}_{V,k}$	All cells connected to vertex V_k
$\mathcal{V}_{\Omega,i}$	All vertices connected to cell Ω_i

Table 3.4.: Examples for shorthands used in the adaptation algorithm listings

Extending cell splits

Splitting a cell into sub-cells can lead to connected cells with more than 2 sub-faces/neighbors on one side (figure 3.13a), requiring a split of these cells. Two *cell split extension strategies* (not to be confused with the *cell split strategy*) are possible:

- **Isotropic split extension**: only isotropic splits are performed (figure 3.13b).

- **Anisotropic split extension**: splits of connected cells are performed only in the same coordinate direction as the split causing them (figure 3.13c).

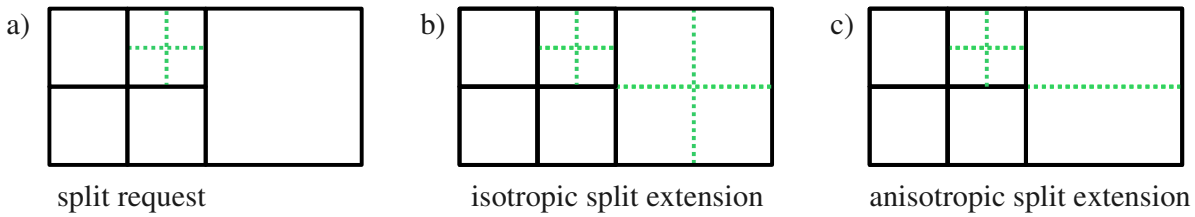


Figure 3.13.: Cell split extension strategies. a) non-admissible modification creating a cell with three sub-faces on one side. b) modification made admissible through additional isotropic split. c) modification made admissible through additional anisotropic split.

The additional splits introduced are possibly non-admissible themselves, leading to the recursive algorithm **ExtendCellSplits** (algorithm 2) to resolve the global consequences of a split. The entry point is a linear search for all cells marked to be split. The *cell split strategy* is again of importance here. In the isotropic case, for a requested x+y-split the splits in both directions must succeed for the combined action to be possible, whereas for the anisotropic cell split strategy a failed split in one direction does not affect the other direction, allowing separate consideration.

Algorithm 2: **ExtendCellSplits**

Input: Modification $\mathcal{M}$, Criteria $\mathcal{C}$
Output: Modification $\mathcal{M}$

for $i = 1 \dots N_{\text{cv}}$ **do**
  if (a^i_{cv} =x+y) **and** (Isotropic split strategy) **then**
    `# Isotropic split strategy`
    possible = **AnalyzeCellSplit**(Ω_i, $\mathcal{M}$, x+y split)
    $\mathcal{M}$ = **PropagateCellSplitDecision**(Ω_i, $\mathcal{M}$, *possible*)
  else
    `# Anisotropic split strategy`
    for (every split direction d in a^i_{cv}) **do**
      possible = **AnalyzeCellSplit**(Ω_i, $\mathcal{M}$, d)
      $\mathcal{M}$ = **PropagateCellSplitDecision**(Ω_i, $\mathcal{M}$, *possible*)

The analysis step is done in **AnalyzeCellSplit** (algorithm 3). It does not directly update $\mathcal{M}$ as the final decision whether the action is possible can only be made after the recursion is completed and all consequences have been evaluated. Instead, preliminary "considered" or "impossible" action flags (which are also used to avoid loops and redundant tests in the recursion) are recorded and a boolean possibility result is returned. At any point in **AnalyzeCellSplit** when a split is decided to be impossible, any further analysis can be skipped (the corresponding pseudo-code has been omitted for reasons of clarity).

Algorithm 3: **AnalyzeCellSplit**

Input: Modification M, split directions d
Output: Modification M, boolean *possible* flag

```
# Step 1:  what directions have to be tested
```
test x/y split = (x/y direction is in d)
if (*test x/y split*) **and** (direction has already been tested or is considered) **then**
  `# Skip tests and reuse existing result`
  test x/y split = false; *x/y split possible* = existing result

```
# Step 2:  perform necessary tests
```
if (*test x split*) **then**
  Mark Ω_i as considered for x split
  `# Stencil limit?  Criterion above veto threshold?`
  x split possible = ($d_x^i > 1$) **and not** ($c_{\mathrm{c},x}^i < C_{\mathrm{c},x}^{\mathrm{v}}$)
  `# Is direct split possible?  First do top/north direction`
  if ($|\mathcal{F}_{\mathrm{Top}}^{\Omega_i}| = 2$) **then**
    `# Top face already split`
    x split possible = *true*
  else
    `# Check neighbor/sub-face rule for top neighbor` Ω_j
    Get top neighbor Ω_j
    if ($|\mathcal{F}_{\mathrm{Bottom}}^{\Omega_j}| = 1$) **then**
      `# Sub-face rule not violated for` Ω_j `by x split of` Ω_i
      x split possible = *true*
    else
      `# Sub-face rule will be violated for` Ω_j`.  Try to split it.`
      if (Isotropic split extension strategy) **then**
        x split possible = **AnalyzeCellSplit**(Ω_j, $\mathcal{M}$, x+y split)
      else if (Anisotropic split extension strategy) **then**
        x split possible = **AnalyzeCellSplit**(Ω_j, $\mathcal{M}$, x split)
  `# Do the same test for the bottom/south direction (`$\mathcal{F}_{\mathtt{Bottom}}^{\Omega_i}$`)`
  ⋮
  if not (*x split possible*) **then** Preliminary mark x split impossible for Ω_i
if (*test y split*) **then**
  Do the equivalent tests/recursion for y direction
  ⋮

```
# Step 3:  combine test results
```
if (Isotropic split strategy) **then**
  `# All the directions have to be splittable`
  possible = (*x split possible*) **and** (*y split possible*)
  if (**not** *possible*) **then** Preliminary mark x+y split impossible for Ω_i
else if (Anisotropic split strategy) **then**
  `# The requested direction has to be splittable`
  possible = *x split possible* **or** *y split possible*

After completing **AnalyzeCellSplit**, **PropagateCellSplitDecision** travels the recursion again and transforms the preliminary flags into final decisions depending on the result. The exact description of **PropagateCellSplitDecision** is omitted here, the basic idea is very similar to **AnalyzeCellSplit**.

Extending face removals

Like a cell split, a face removal can lead to violation of the sub-face rule. Additionally, if anisotropic cells are allowed, cells that are not logically rectangular can occur. Face removals are more complicated to analyze than splits, as depending on the chosen removal strategy a face often cannot be removed individually but only as part of a *face group* which is removed simultaneously to form an atomic operation. While the general design of the face removal analysis algorithm is similar to the split extension, one recursion level now consists of two steps: face group analysis (**AnalyzeFaceGroupRemove**) and face analysis (**AnalyzeFaceRemove**).

The starting point of the face removal analysis (algorithm 4) is again a linear search over all faces. For faces marked for removal the corresponding face group is analyzed.

Algorithm 4: **ExtendFaceRemoval**

Input: Modification $\mathcal{M}$, Criteria $\mathcal{C}$
Output: Modification $\mathcal{M}$

for $i = 1 \dots N_{\text{fc}}$ **do**
  if $(a^i_{\text{fc}} = \text{remove})$ **then**
    possible = **AnalyzeFaceGroupRemove**($\mathcal{M}, \mathcal{C}, e_i$)
    PropagateFaceRemoveDecision($\mathcal{M}$, *possible*)

Face group analysis

The first task in a face group analysis is finding the face group for the input face e_i, the definition of which depends on the *coarsening strategy*. Possible choices are:

- **Isotropic coarsening strategy**: allow only coarsening steps resulting in isotropic cells (aspect ratio 1:1).
- **Anisotropic coarsening strategy**: allow all admissible coarsening steps.

Figure 3.14 shows the possible face group configurations (up to symmetry and omitting similar cases). In all cases, the cells that are to be merged must resolve to a common parent cell in the next coarser hierarchy of the grid, which can be tested by evaluating equation (3.3). For the isotropic merge strategy (figure 3.14a), this is already enough to find the correct group,

Algorithm 5: **AnalyzeFaceGroupRemove**

Input: Modification $\mathcal{M}$, Criteria $\mathcal{C}$, face e_i
Output: Modification $\mathcal{M}$, boolean *groupPossible* flag

if (Face e_i already tested) **then** return existing *groupPossible* result

```
# Find and analyze faces in group
```
$\mathcal{G}_i$ = **GetFaceGroup(**e_i**)**
if ($|\mathcal{G}_i| = 0$) **then** *groupPossible* = false

forall (Face e_k in $\mathcal{G}_i$) **do**
- *facePossible* = **AnalyzeFaceRemove(**$\mathcal{M}$, e_k**)**
- `# If a face fails, the group fails.`
- `# What this means for the individual faces depends on the group type.`
- **if** (**not** *facePossible*) **then**
  - **if** (Isotropic coarsening strategy) **then**
    - `# If one face fails, none can be removed`
    - Mark all faces in $\mathcal{G}_i$ preliminary impossible
  - **else if** (Anisotropic coarsening strategy) **then**
    - `# Effect depends on face group type`
    - **switch** $|\mathcal{G}_i|$ **do**
      - **case** 1
        - `# Single face` $e_i = e_k$`, case figure 3.14a`
        - Mark e_i as preliminary impossible
      - **case** 3
        - `# Three face (T) case, case figure 3.14b`
        - Mark failed face e_k and input face e_i preliminary impossible
        - Mark all faces in $\mathcal{G}_i$ that are sub-faces preliminary impossible
      - **case** 4
        - `# Four faces, case figure 3.14c`
        - Mark input face e_i preliminary impossible (leave others untouched)
  - Return *groupPossible* = false

and with this strategy removal of any face in the group will lead to the removal of the entire group unless the action is vetoed for one of the faces.

In the anisotropic case (figure 3.14b), the situation is more complicated because simultaneous removal of two faces with differing alignment from an isotropic face group (which are by itself admissible) leads to L-shaped cells (figure 3.15). In the case of a "T"-shaped face group (figure 3.14b 4,5) this can happen if a sub-face is removed (in this context a *sub-face* is a face that does not cover the entire side of one of the cells it is connected to). This has to be avoided by extending the face group accordingly. If a face cannot be removed due to grid hierarchy alignment rules, the face group for this face is empty (as e.g. shown in figure 3.14a 1,2, b 1,2). Also note that due to the grid cell alignment rule, face group selection is never ambiguous.

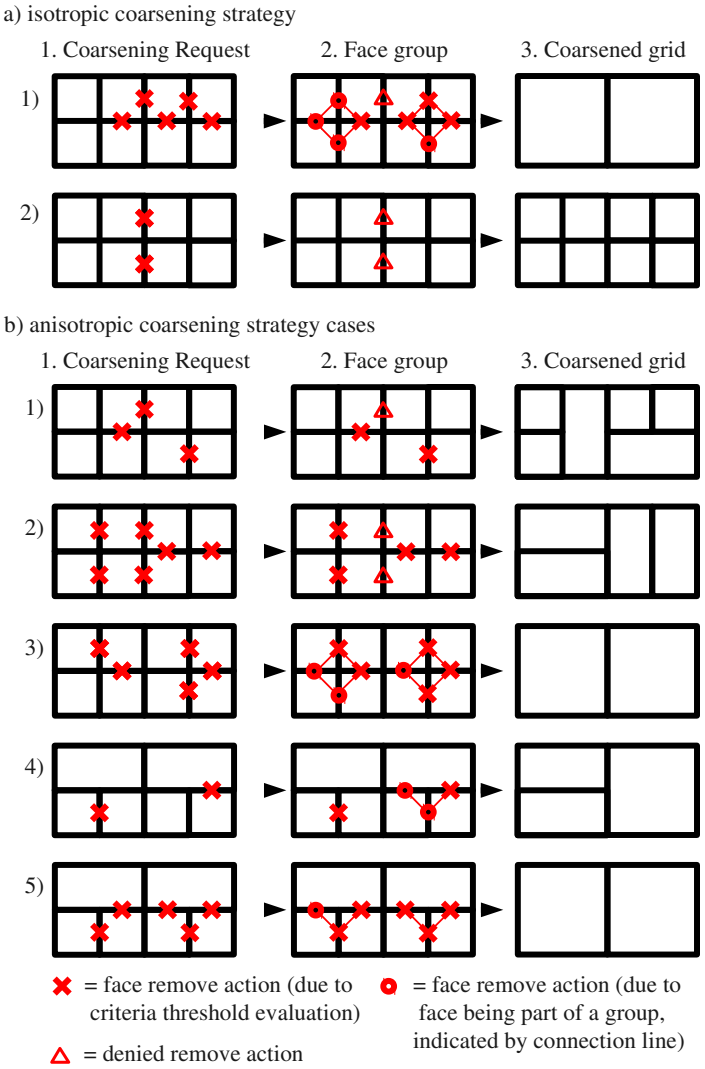


Figure 3.14.: Face group case differentiation for isotropic and anisotropic coarsening strategy.

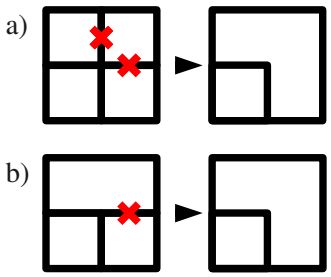


Figure 3.15.: L-shaped cells with anisotropic coarsening. The face group has to be extended to result in an admissible atomic operation.

Algorithm 6: **GetFaceGroup**

Input: Modification $\mathcal{M}$, face e_i
Output: Face group $\mathcal{G}_i$

if (Isotropic coarsening strategy) **then**
  `# Case figure 3.14a. First look at start vertex of face` $V_{F,i}^{\mathtt{From}}$
  Get cells $\mathcal{C}_{V_{F,i}^{\mathrm{From}}}$ connected to start vertex $V_{F,i}^{\mathrm{From}}$ of face e_i
  if (All cells in $\mathcal{C}_{V_{F,i}^{\mathrm{From}}}$ resolve to same parent cell) **then**
    Return group $\mathcal{G}_i = \mathcal{F}_{V_{F,i}^{\mathrm{From}}}$ of faces connected to $V_{F,i}^{\mathrm{From}}$
  `# Do the same for the end vertex` $V_{F,i}^{\mathtt{To}}$
  Get cells $\mathcal{C}_{V_{F,i}^{\mathrm{To}}}$
  if (all cells in $\mathcal{C}_{V_{F,i}^{\mathrm{To}}}$ resolve to same parent cell) **then**
    Return group $\mathcal{G}_i = \mathcal{F}_{V_{F,i}^{\mathrm{To}}}^{\mathrm{Top}}$
else if (Anisotropic coarsening strategy) **then**
  `# Case figure 3.14b`
  if (e_i is a sub-face) **then**
    Find (T-shaped) face group (either $\mathcal{F}_{V_{F,i}^{\mathrm{From}}}$ or $\mathcal{F}_{V_{F,i}^{\mathrm{To}}}$)
      by testing for common parent cell
    if ($\mathcal{F}_{V_{F,i}^{\mathrm{From}}}$ or $\mathcal{F}_{V_{F,i}^{\mathrm{To}}}$ is face group) **then**
      Return the group
    else Return empty group $\mathcal{G}_i = \emptyset$
  else if (e_i is not a sub-face) **then**
    if (cells $\Omega_{F,i}^{\mathrm{dec}}$ and $\Omega_{F,i}^{\mathrm{inc}}$ connected to face e_i resolve to same parent cell) **then**
      `# Cells can be merged. Have to extended group? (figure 3.14b 3-5)`
      if (cells $\mathcal{C}_{V_{F,i}^{\mathrm{From}}}$ coarsen to same parent cell) **then**
        if (No face in $\mathcal{F}_{V_{F,i}^{\mathrm{From}}}$ is marked as impossible to remove) **and** (A face in $\mathcal{F}_{V_{F,i}^{\mathrm{From}}}$ with different orientation than e_i is marked for remove) **then**
          Return $\mathcal{G}_i = \mathcal{F}_{V_{F,i}^{\mathrm{From}}}$
      Do the same test for cells $\mathcal{C}_{V_{F,i}^{\mathrm{To}}}$ and faces $\mathcal{F}_{V_{F,i}^{\mathrm{To}}}$ connected to $V_{F,i}^{\mathrm{To}}$
      ⋮
      `# If no group found yet, no group extension necessary`
      $\mathcal{G}_i = \{e_i\}$
`# If no group was found, face` e_i `cannot be removed. Return an empty group.`
$\mathcal{G}_i = \emptyset$

Face analysis

If the face group found for a face is not empty, the algorithm tests for every face in the group whether removing it is possible using the function **AnalyzeFaceRemove**. A group can only be removed if all individual faces can be removed. However, the reverse argument that if a group fails, none of the individual faces can be removed, is not true in general and depends on the chosen coarsening strategy.

The central concern in **AnalyzeFaceRemove** is violation of the sub-face rule in the parent cell resulting from the merge. If the rule is violated, **AnalyzeFaceRemove** identifies coarsening actions necessary to avoid it, finds the faces that have to be removed to perform them and tests the possibility of by calling **AnalyzeGroupRemove** for these faces.

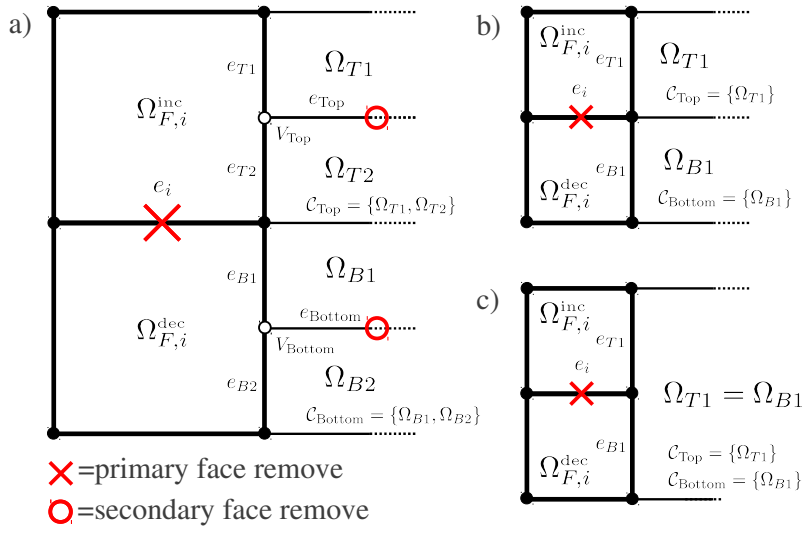


Figure 3.16.: Possible cases for extending removal of face e_i towards the right neighbours (see algorithm 7)

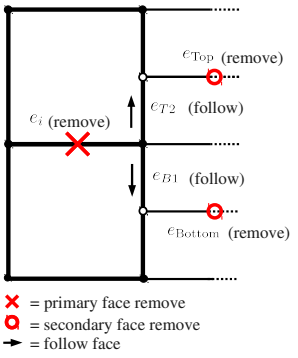


Figure 3.17.: "Follow" flags for face recursion

As in the cell split recursion, loops and redundant tests are avoided by recording preliminary result flags. Again, the detailed pseudo-code for these tests and the **PropagateDecision** algorithm are mostly omitted for brevity, as they do not contribute new essential parts to the algorithm. The only complications to be mentioned for the face recursion is that in order to follow the entire path of faces involved in a global face remove extension, some faces that are not marked for removal have to be followed. They are marked using a "follow" flag, as shown in figure 3.17.

3.3.3. Criteria threshold and strategy choices

Summarizing the parameters controlling the adaptation algorithms, the user has to specify the action and veto thresholds and three strategy settings (either *isotropic* or *anisotropic*): the *cell split strategy* that defines how the action thresholds are interpreted, the *cell split*

Algorithm 7: **AnalyzeFaceRemove**

Input: Modification $\mathcal{M}$, Face e_i, Face group $\mathcal{G}_i$
Output: Modification $\mathcal{M}$, boolean *facePossible* flag

if (e_i is persistent) **then** return *facePossible*=false
`# Evaluate veto threshold (depending on face orientation)`
if ($c_f^i > C_{\mathrm{f,x/y}}^{\mathrm{v}}$) **then** return *facePossible*=false

`# The parent cell is also in tested GetFaceGroup, repeated for clarity`
Get cells $\Omega_k = \Omega_{F,i}^{\mathrm{dec}}$, $\Omega_l = \Omega_{F,i}^{\mathrm{inc}}$ connected to e_i
if not (Ω_k, Ω_l coarsen to same parent) **then** return *facePossible*=false
`# Is a connected cell marked to be split (split precedence)?`
if ($a_{cv}^k \neq$ none) **or** ($a_{cv}^l \neq$ none) **then** return *facePossible*=false

if (e_i is x-aligned) **then**
  if (e_i is rightmost face in group $\mathcal{G}_i$) **then**
    `# Check sub-face rule on the right side.`
    Collect right neighbors $\mathcal{C}_{\mathrm{Top}}, \mathcal{C}_{\mathrm{Bottom}}$ (figure 3.16)
    `# Case figure 3.16c:  only one neighbor cell`
    if ($\Omega_{T1} = \Omega_{B1}$) **then** return *facePossible* = true
    `# Case figure 3.16a,b:  check for simultaneous splits`
    if (a cell in $\mathcal{C}_{\mathrm{Top}}$ or $\mathcal{C}_{\mathrm{Bottom}}$ is marked for split in y-direction) **then**
      Return *facePossible* = false
    if ($|\mathcal{C}_{\mathrm{Top}}| = 2$) **then**
      `# Case figure 3.16a`
      Find face e_{Top} connecting Ω_{T1} and Ω_{T2}
      facePossible = **AnalyzeFaceGroupRemove**($\mathcal{M}$, $\mathcal{C}$, e_{Top})
    if ($|\mathcal{C}_{\mathrm{Bottom}}| = 2$) **then**
      Find e_{Bottom} connecting Ω_{B1} and Ω_{B2}
      facePossible = **AnalyzeFaceGroupRemove**($\mathcal{M}$, $\mathcal{C}$, e_{Bottom})
  else if (e_i is leftmost face in group $\mathcal{G}_i$) **then**
    Do the equivalent test with the neighbor cells and faces on the left side.
    ⋮
else if (e_i is y-aligned) **then**
  Do the equivalent tests with neighbor cells and faces on top and bottom side.
  ⋮

extension strategy controlling necessary grid refinement actions and the *coarsening strategy* governing face removals.

The exact choice of criteria thresholds obviously depends mainly on how the adaptation criteria are defined and normalized, for which some examples are described in the next section. Independently of this, some general observations can be made.

Global vs. local extensions

The presented recursive action extension algorithms enable local grid modifications to cause global changes. This is usually intended, and unwanted excessive coarsening can be contained using the veto thresholds. When pushing the veto mechanism to its logical extreme by setting the veto threshold equal to the corresponding action threshold

$$C^s_{c,x/y} = C^v_{c,x/y}, \quad C^r_{f,x/y} = C^v_{f,x/y}, \tag{3.5}$$

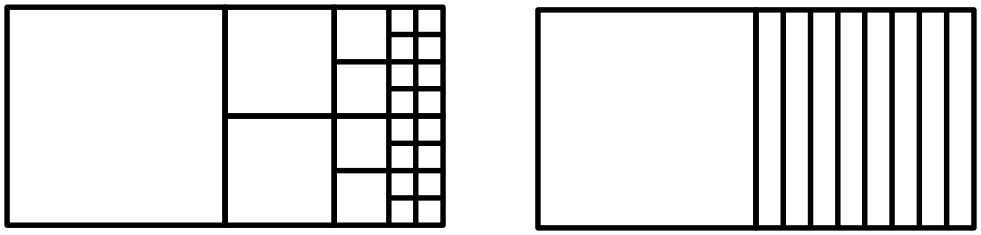
only actions will be performed that are explicitly requested by the criteria. This changes the character of the algorithm from global to local, reducing it to a grid structure sanity check. While such a configuration makes no sense for complex dynamical criteria, it can be used for strict enforcement of static criteria like fixed grid resolution patterns.

Threshold choice

In non-time-dependent simulations, the grid is expected to converge to a steady state together with the solution. In this situation sensitive criteria can cause values for an object to jump directly from the refinement to the coarsening range, resulting in the grid alternating between two states. This can be partially avoided by choosing the action thresholds $C^s_{c,x/y}, C^r_{f,x/y}$ sufficiently far apart. A more effective fix is to choose an appropriate (possibly nonlinear) normalization of the criteria.

Fully isotropic vs. fully anisotropic strategy

While any combination of strategies is supported and might make sense in specific situations, the most common scenarios are fully isotropic or fully anisotropic adaptation. Anisotropic adaptation offers higher flexibility and thus potentially allows grids with lower cell and face counts. The disadvantage is that cells with very high aspect ratios and steep changes in grid resolution can occur (c.f. figure 3.18), which must be taken into account when designing criteria. For isotropically adapted grid the cell aspect ratio is fixed, and the coupling between dimension due to the sub-face rule limits the resolution change between neighbor cells to 2:1.



(a) Isotropic strategy (b) Anisotropic strategy

Figure 3.18.: Grid resolution jumps for different adaptation strategies.

3.4. Adaptation criteria

Adaptation criteria are the defining element driving grid evolution. Their goal can be described concisely to optimize a grid w.r.t. the numerical approximation error.

- For a given global error bound, find the grid with the lowest cell count, or
- for a given cell count, find the grid delivering the lowest possible global error.

Despite the clear definition of the objective, deriving criteria to meet it is not straightforward. The main obstacle is that determining the approximation error of a numerical method (in this case the finite volume method (FVM)) is a hard task already for simple model equations like the advection-diffusion equation, let alone nonlinear equation systems like the B2 equations. But even when reliable error estimates for individual quantities are available, efficient use of this information depends on the problem. For example, approaches that try to equidistribute the error can be inefficient when the interest is an optimal approximation of a specific output functional. Overall, criteria design is highly problem specific.

3.4.1. Error estimators

Due to the high interest in efficient schemes, *a-posteriori* estimation of discretization errors is a topic covered extensively in the literature, and various approaches have been developed. Starting from the derivation of the numerical scheme, estimates for the local truncation error (LTE) (4.22) can be given, and depending on the model equations can be used directly to estimate the global error (4.20). A general approach to determine the LTE is Richardson or τ-extrapolation. Two approximations with differing accuracy (either computed using different grid resolutions or using two numerical schemes with different consistency order) are solved and the solutions are combined to estimate the LTE. While this can be done straightforwardly for rectangular grids forming a grid hierarchy as studied in this work, problems occur for unstructured grids where the local truncation error depends strongly on cell geometry, potentially suffering from inaccuracies near refinement interfaces [8]. This effect is observed in the benchmarks discussed in chapter 5.

Another common theme is exploitation of the strong theoretical foundation of the Finite Element Method (FEM). The FVM and FEM are linked by the close relationship of high-order FVMs to Discontinuous Galerkin (DG) methods in Petrov-Galerkin form, allowing application of ideas and results from FEM error analysis to the FVM. This resulted in the derivation of a number of error estimators [111, 18, 62, 37] for the global error, which for B2 can be applied directly to the individual linearized equations.

More advanced schemes consider user-defined output functionals (e.g. lift and drag for aerodynamic simulations) and relate the sensitivity of these quantities to the solution via an adjoint problem [18, 85]. Combined with an approximation of the global error, this relation can then be used to directly correct the output functional and drive the grid adaptation to specifically optimize the discretization for the functional of interest. As most modeling tasks using B2 revolve around accurate predictions for certain key quantities (e.g. power flux density on divertor and wall), the adjoint-based approach holds great promise.

While criteria that are aware of the global error are ultimately most desirable, their theory can be complex and they often only work reliably for smooth grids and solutions. For unstructured grids and solutions exhibiting strong features like shocks, they can become hard to use and often have to be combined with heuristic criteria to stabilize the adaptive algorithm.

3.4.2. Feature detectors

A general class of heuristic criteria are *feature detectors*. They are based on the observation that the LTE is usually high in regions with high data variation (like steep gradients or shocks) and overall solution accuracy can be improved by adapting the grid to accurately resolve them. Criteria of this type have been applied with good results in various fields and are still in widespread use due to their simplicity and robustness. Also, an advantage that should not be underestimated is easy accessibility, as their straightforward connection of cause and effect enables the modeler to customize the feature detectors to generate grids based on his intuition and experience. However, careless use of feature detectors can be dangerous [118]. The fundamental limitations of this approach must be kept in mind at all times.

Some simple feature detectors

Some general design guidelines for feature detectors are detailed in [8]. The main point is that criteria should mimic the qualitative behavior of the local truncation error, i.e. $c_{child} < (1/2)^p c_{parent}$ after a split for a p-th order scheme. Therefore approximations of derivatives (that ideally do not change when modifying the grid resolution) are unsuited as criteria.

In the following examples, two types of feature detectors are used. Type 1 simply measures variation of a given grid function ϕ_i at every face e_k by computing a simple absolute undivided

difference

$$c_f^k = |\phi_i - \phi_d| \quad \text{with } \Omega_i = \Omega_{F,k}^{\text{inc}}, \Omega_d = \Omega_{F,k}^{\text{dec}} \tag{3.6}$$

and therefore triggers refinement in regions of steep gradients and coarsening for flat regions. Type 2 measures the variation of the gradient in the direction normal to the face (e.g. from least-squares reconstruction)

$$c_f^k = |(\nabla\phi)_i - (\nabla\phi)_d|, \tag{3.7}$$

resulting in coarsening where the solution varies linearly and refinement in regions where it changes slope. Unlike the criteria in [8], both criteria behave like the LTE of a first-order scheme.

A simple normalization is to rescale the criteria to the interval $[0, 1]$:

$$c_f^{k,*} = \frac{c_f^k}{\max_{i=1,N_{fc}} c_k^f}. \tag{3.8}$$

Note that this normalization changes at each grid modification cycle. For a steady-state solution, the grid will nevertheless converge to a fixed state, but the resulting grid resolution and cell count for a specific threshold choice depend on the solution and are therefore hard to predict, requiring some initial tests and adjustments. Adaptation parameter scans can then be done by scaling the thresholds. More predictable results can be achieved by choosing a fixed global criterion normalization at the beginning of the simulation.

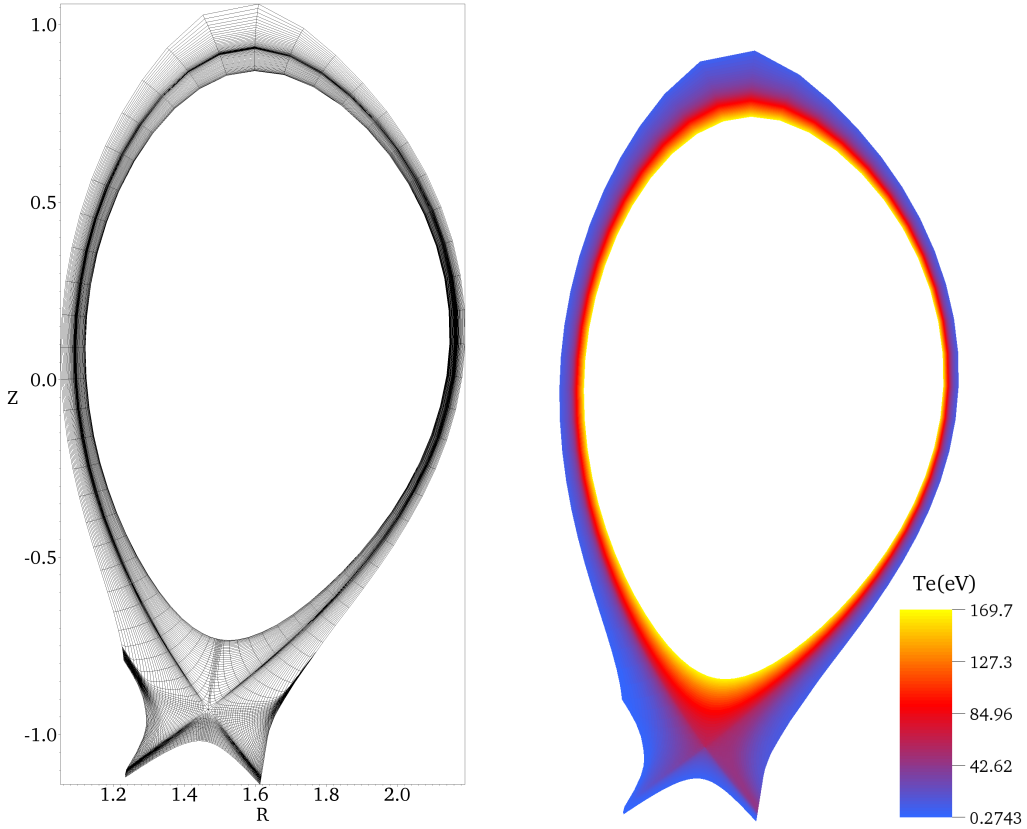
Cell criteria in every coordinate direction are derived from the face criteria by averaging the values of the faces normal to the respective coordinate direction

$$c_{c,x}^i = \frac{\sum_{e_j \in \mathcal{F}_{\Omega,i}^{\text{y-aligned}}} c_f^j}{|\mathcal{F}_{\Omega,i}^{\text{y-aligned}}|}, \qquad c_{c,y}^i = \frac{\sum_{e_j \in \mathcal{F}_{\Omega,i}^{\text{x-aligned}}} c_f^j}{|\mathcal{F}_{\Omega,i}^{\text{x-aligned}}|}. \tag{3.9}$$

Similarly, face criteria can be derived from cell criteria using maximum or averaging operators.

3.5. Grid adaptation examples

The following examples serve the purpose of demonstrating the capabilities and flexibility of the adaptation algorithm and highlighting some of the issues mentioned so far. In this section only steady-state situations are shown, a time-dependent benchmark using adaptive grids is discussed in section 5.2. It must be stressed that in the following examples the model equations are not solved, the focus is entirely to test the adaptation algorithms using given converged B2.5 solutions.



(a) Base grid (128x64)

(b) T_e reference

Figure 3.19.: Base grid and T_e reference solution in physical space. An enlarged view of the grid in the divertor region is shown in figure 3.24a.

3.5.1. ASDEX Upgrade discharge #16151 benchmark solution

This example is based on the B2 steady-state standalone benchmark case for ASDEX Upgrade discharge #16151 as used in chapter 7. A base grid with a resolution of 128x64 cells[3] in the simulation domain is used. The grid is adapted to a fixed converged electron temperature T_e solution (see physical domain plot in figure 3.19b) using two feature detectors:

- Criterion C1: data variation (3.6) with global normalization (3.8).
- Criterion C2: gradient variation (3.7) with global normalization (3.8).

The goal is to obtain grids that offer a good solution representation while reducing the number of cells and respectively the grid resolution. Initial action thresholds are chosen as listed in table 3.5 and scaled linearly with a common factor to produce a series of threshold sets that result in grids with varying resolution. For every threshold set, starting from the finest possible (base) grid, adaptation cycles with both fully isotropic and fully anisotropic adaptation strategies are performed. Depending on parameters, 3-10 iterations are necessary

[3] 8192 cells in the simulation domain, 8580 cells including ghost cells.

	$C^{\mathrm{v}}_{\mathrm{c},x}$	$C^{\mathrm{s}}_{\mathrm{c},x}$	$C^{\mathrm{v}}_{\mathrm{c},y}$	$C^{\mathrm{s}}_{\mathrm{c},y}$	$C^{\mathrm{r}}_{\mathrm{f,x}}$	$C^{\mathrm{v}}_{\mathrm{f,x}}$	$C^{\mathrm{r}}_{\mathrm{f,y}}$	$C^{\mathrm{v}}_{\mathrm{f,y}}$
C1	0.00	0.70	0.00	0.70	0.10	0.30	0.20	0.30
C2	0.00	0.35	0.00	0.40	0.10	0.20	0.15	0.40

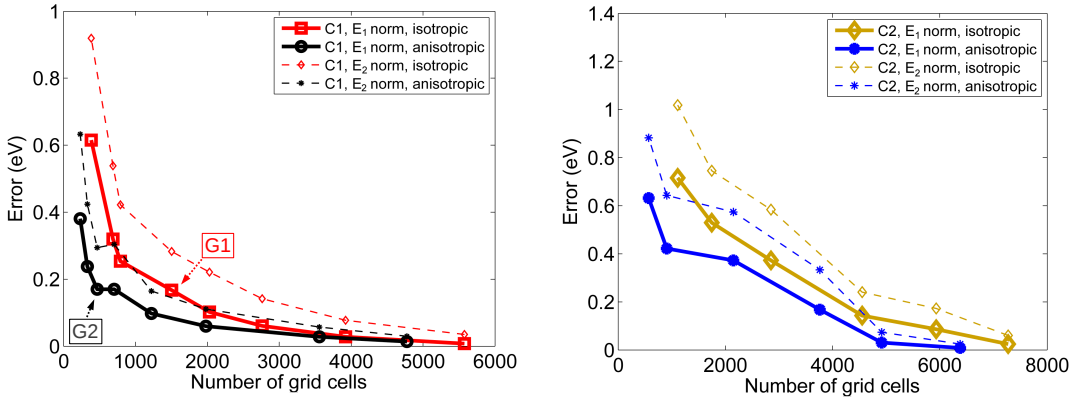
Table 3.5.: Cell and face action thresholds for the #16151 benchmark grid adaptation algorithm test. They are rescaled with a factor $\lambda \in [0.1, 1.4]$ to obtain a range of grids.

until the working grid reaches a fixed state. While in principle this number is bounded by the range of resolution levels in the grid, in these examples it also depends on how fast the normalization of the criteria settles. In some cases oscillation around the final state is observed. This effect is strongest for anisotropic grids with high cell numbers and can be avoided with a more sophisticated normalization. As the problem only affects a very small number of cells in all cases, this is omitted here for reasons of simplicity.

After every adaptation cycle the T_e solution is transferred from the old to the new grid using linear reconstruction as described in section 4.4.2 for all interpolation steps. The error of the solution representation on the final working grid is determined by transferring the solution back to the base grid and computing the deviation from the original solution. For the error computation the volume-averaged measure (4.26) denoted E_p is used.

Comparison of criteria and strategies

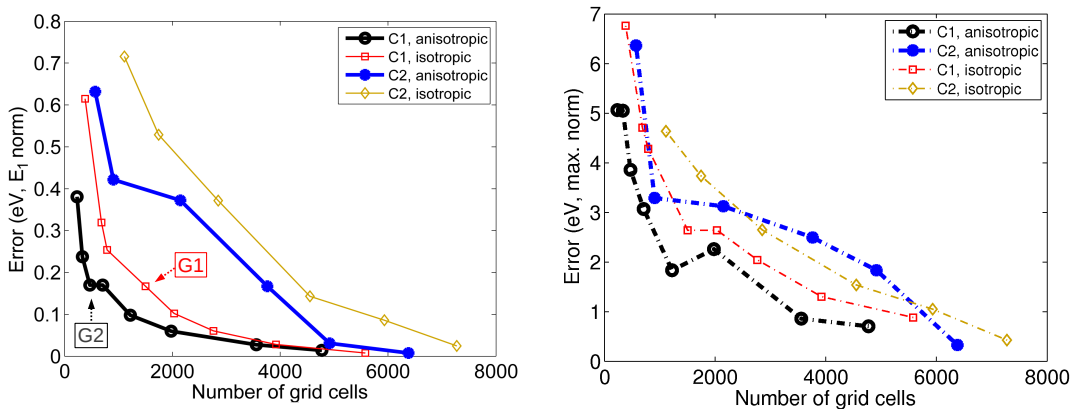
The error vs. cell count relations obtained by this resolution scan are first plotted individually for the two criteria in figures 3.20a and 3.20b. Every data point represents one threshold set and the resulting final grid. As desired, for both criteria the error drops with increasing grid resolution.



(a) Criterion C1

(b) Criterion C2

Figure 3.20.: E_1 and E_2 error of T_e for varying grid resolution and strategies



(a) C1 vs. C2, E_1 error

(b) C1 vs. C2, E_∞ error

Figure 3.21.: Error comparison criteria C1 vs. C2, isotropic and anisotropic strategies. C1 (data variation) clearly outperforms C2 (gradient variation).

The anisotropic strategy consistently requires less grid cells to achieve a given accuracy, especially for coarse grids. This is illustrated by comparing the grids labeled G1 and G2 in figure 3.20a. Both achieve a E_1 error of $\approx 0.17eV$. G1 is an isotropic grid with 1505 cells, G2 is anisotropic with 470 cells. Plots in the computational domain are presented in figures 3.22 and 3.23, the physical view of the grids in the divertor region is shown in figures 3.24b and 3.24c. The criterion forces the grid to high radial(y) resolution in the core region where at the same time the solution is constant in the poloidal(x) direction. The fixed cell aspect ratio enforced by the isotropic strategy leads to an excessively high poloidal resolution in this region, significantly increasing the total cell number. With the anisotropic strategy, the grid effectively reduces to a one-dimensional discretization in the core.

To compare the performance of the criteria, the E_1 and E_∞ errors of both criteria are plotted in figures 3.21a and 3.21b. In this example criterion C1 based on data variation clearly outperforms criterion C2 based on gradient variation. What also becomes clear from the E_∞ and E_2 errors is that the behavior of isotropic grids is more consistent in decreasing the error, while grids resulting from anisotropic adaptation exhibit local increases in error when increasing the grid resolution. Looking at the grid in figure 3.23 shows that the aspect ratio of cells in anisotropic grids obtained in this resolution scan can be very high, with much more rapid variation of the grid spacings especially for coarse grids. In this situation the criteria can react much stronger to grid modifications than for grids which have smooth transitions of resolution (as is the case for the isotropic grids), resulting in such non-smooth behavior. A solution to this is to limit the allowed aspect ratio of cells (both in computational and physical space, possibly depending on local parameters or grid region), forcing the algorithm to produce smoother grids.

To provide more insight into the performance of criterion C1, the absolute and relative errors of the solution representation on grid G2 with respect to the reference solution are shown in figures 3.25 and 3.26. While the absolute error is distributed equally over the whole domain, the relative error is localized to the low-temperature regions close to the walls and targets.

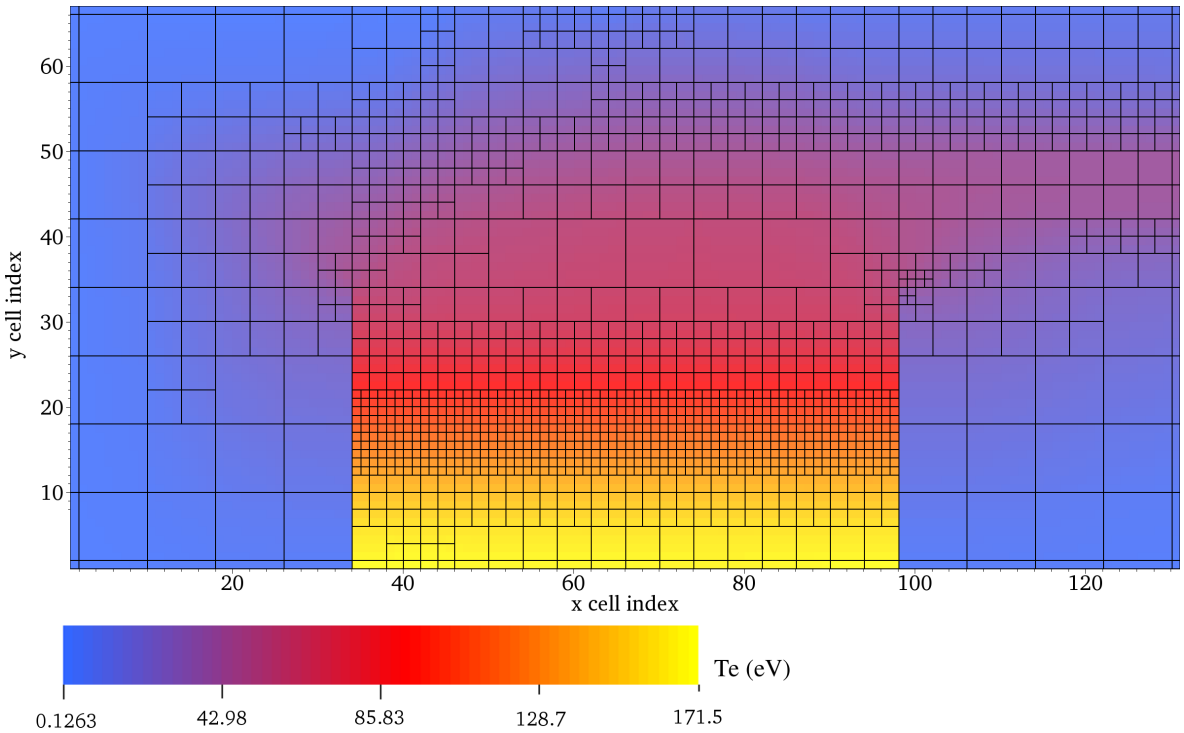


Figure 3.22.: Grid G1 (computational space plot, cf. figure 3.2): Criterion C1, isotropic strategy, 1505 cells, adapted to T_e (The scales in figures 3.22 and 3.23 deviates slightly from figure 3.19b because plots in computational space include ghost cells). The isotropic strategy leads to excessively high grid resolution in the core.

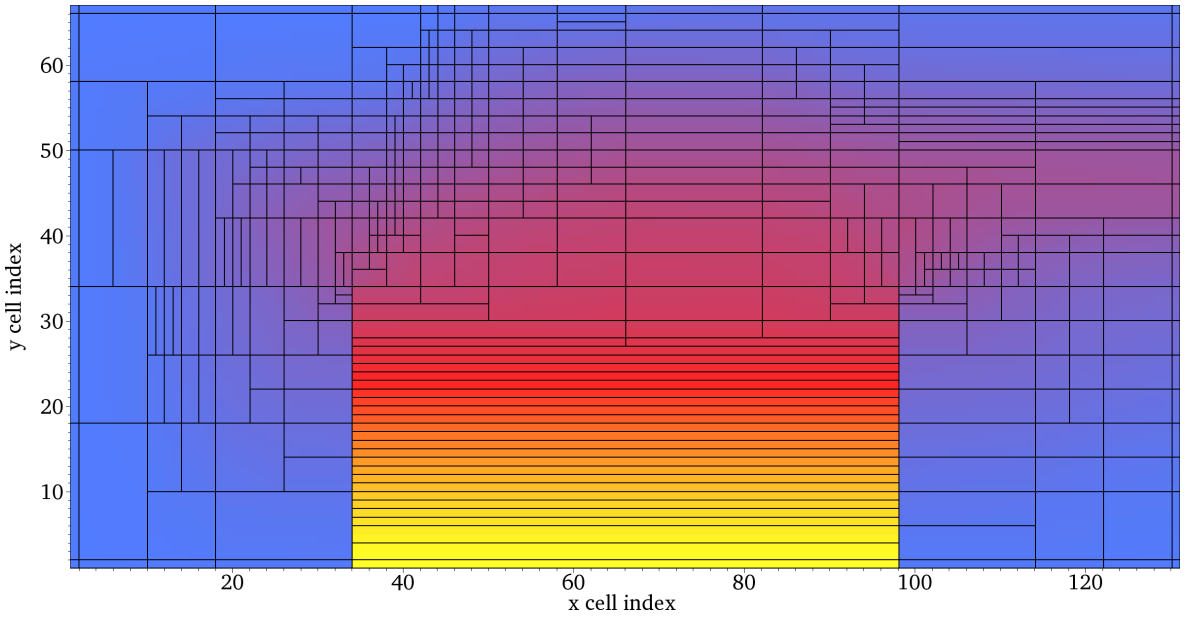
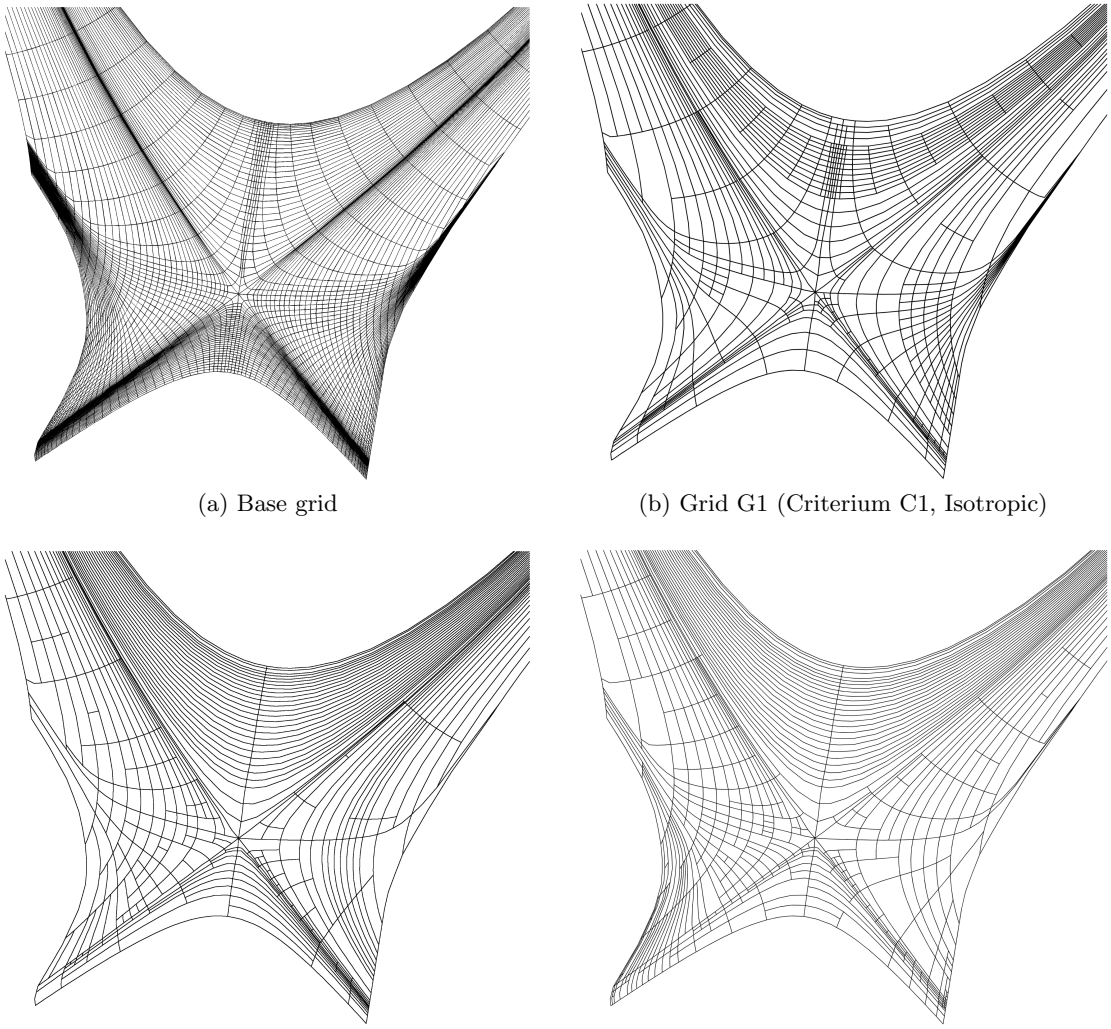


Figure 3.23.: Grid G2 (computational space plot, cf. figure 3.2): Criterion C1, anisotropic strategy, 470 cells, adapted to T_e. The one-dimensional situation in the core is correctly resolved.



(a) Base grid

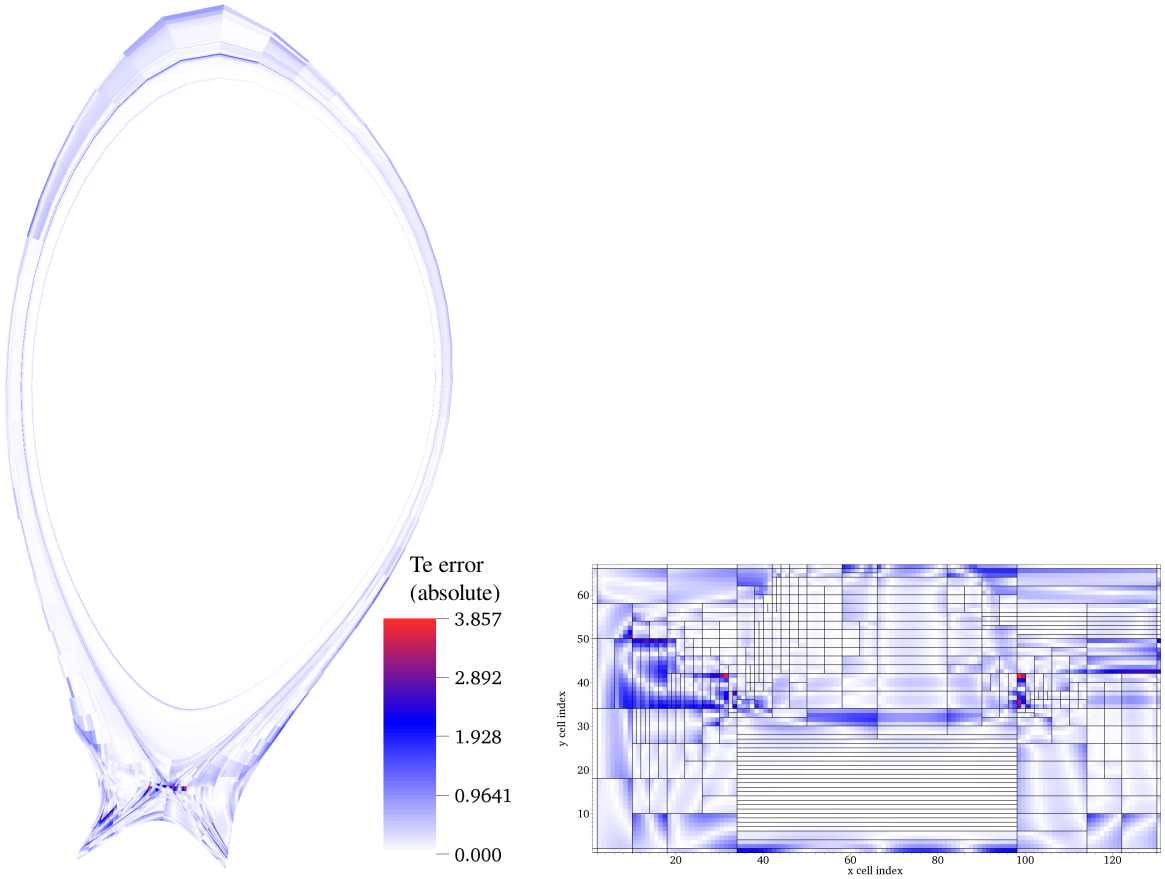
(b) Grid G1 (Criterium C1, Isotropic)

(c) Grid G2 (Criterium C1, anisotropic)

(d) Grid G3 (Criterium C3, anisotropic)

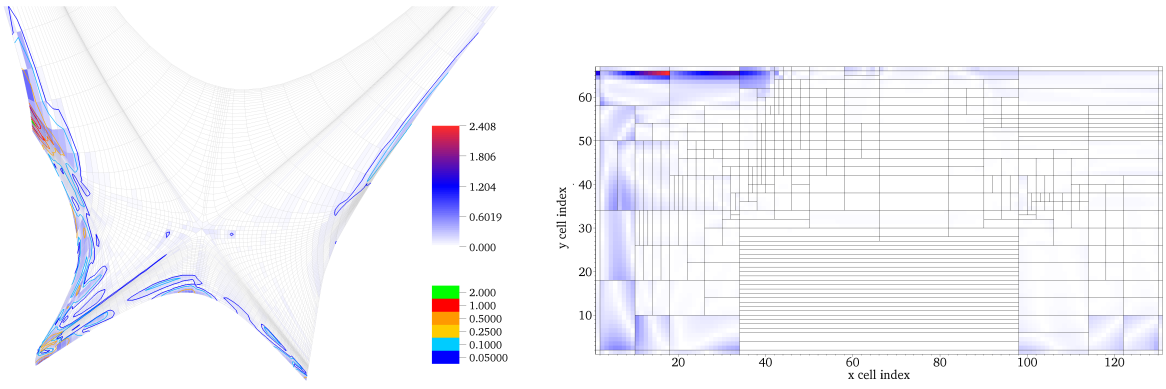
Figure 3.24.: Plots in physical space of grids in the divertor region. Criterium C3 increases the resolution at the target plates.

This is to be expected: criterion C1 only considers absolute data variation, and the global normalization weighs all deviations equally.



(a) Physical domain plot (b) Computational domain plot

Figure 3.25.: Absolute error of the T_e approximation on grid G2 compared to the reference solution. Criterium C1 leads to similar absolute deviations in the whole domain.



(a) Physical domain plot (b) Computational domain plot

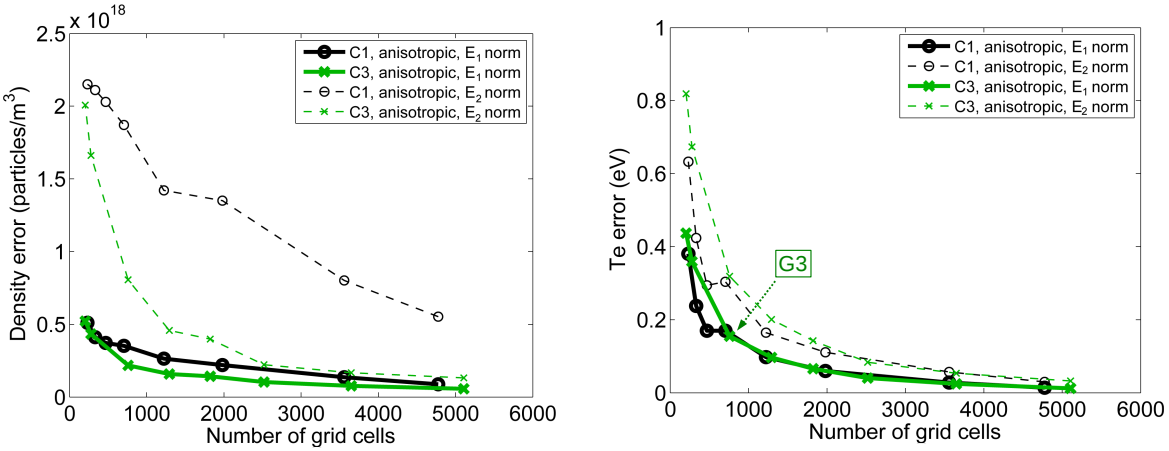
Figure 3.26.: Relative error of the T_e approximation on grid G2 with respect to the reference solution. High relative errors are located in areas with low temperature.

Combining criteria

The example above only considers one state variable (T_e). Usually a code stores and solves multiple quantities on the working grid, and in general these exhibit differing qualitative behavior. To take all relevant quantities into account, either multiple criteria or an appropriate proxy quantity have to be used to drive grid adaptation. A straightforward approach is to compute separate criteria with comparable normalizations and combine them into a single criterion which is then passed to the adaptation algorithm.

This approach is demonstrated for the same benchmark case used before. Figure 3.27a shows again the grid resolution scan, this time plotting the error introduced for the electron density n_e on the the anisotropic grids created with the C1 criterion. The C1 criterion does not consider n_e, and while the E_1 error behaves satisfactory, the E_2 error reveals a high localized error that only improves slowly when increasing the resolution. A new criterion C3 is defined by applying a criterion of type C1 also to n_e and combining the criterion values resulting from T_e and n_e using the max operator. The combined criterion reduces the n_e error (also plotted in figure 3.27a) at the same rate previously observed for T_e. The solution approximation of T_e (figure 3.27b) is mainly unaffected. Compared to criterion C1, only the cell number for coarse grids is increased, but the effect vanishes quickly with increasing cell count. To illustrate the difference, figure 3.28 shows a plot in computational space of the grid marked G3, which like G1 and G2 has an E_1 error for T_e of ≈ 0.17. The grid shows how additional cells are used to resolve the density feature in the inboard and outboard divertor. The corresponding divertor plot is shown in figure 3.24d.

It must be kept in mind that the aim of this example was to study the performance of the adaptation algorithms. They were used to find grids that offer a favorable balance between cell count and approximation quality for a given solution grid function, but the grids were not used to solve for these solutions. In fact, whether the grids obtained using the feature detectors are particularly suited for the numerical solution of the B2 model equations is not obvious and has to be studied separately.



(a) (b)

Figure 3.27.: Comparison criterion C1 (only T_e) vs. criterion C3 (combined T_e and n_e)

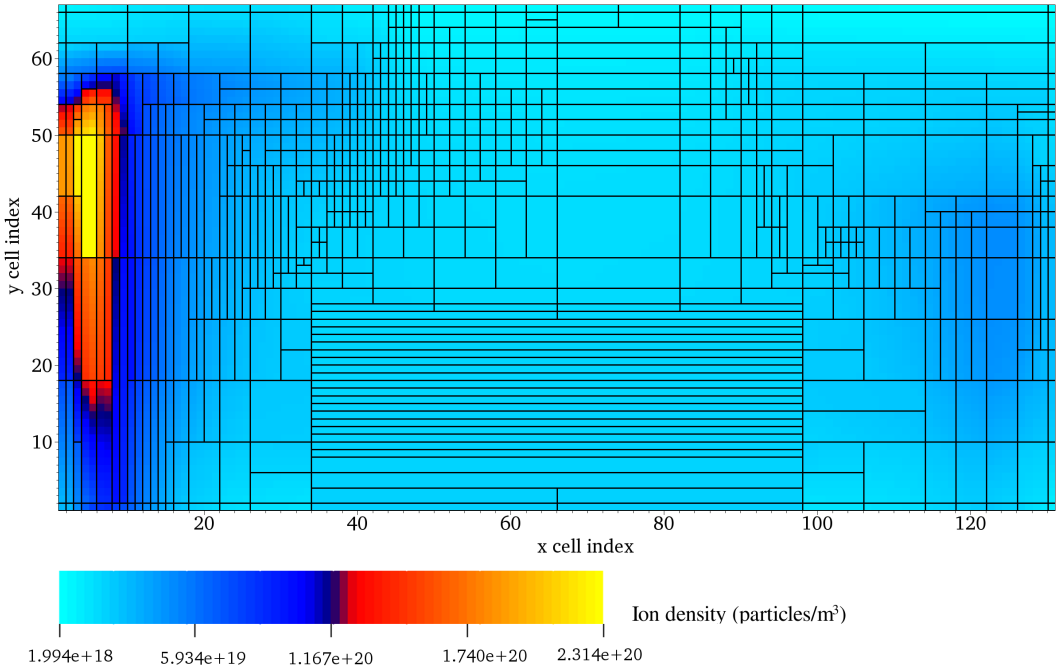


Figure 3.28.: Grid G3 (760 cells), adapted to T_e and n_e with criterion C3 (anisotropic strategy), plotted in computational space. The resolution at the target plates is increased to match the electron density. The corresponding plot of the divertor region in physical space is shown in figure 3.24d.

3.5.2. Grid hierarchy generation

Next to solution-driven grid adaptation, the adaptation algorithm can be used to force the grid resolution to a preset configuration by defining artificial criteria. An application for this is the creation of grid hierarchies needed for multigrid algorithms. This example uses a base grid created for the ASDEX Upgrade discharge #15320 with disconnected double null topology. Starting from a working grid adapted to a n_e solution (figures 3.29a and 3.30a) a hierarchy of nested grids with a stencil aspect ratio of 4 : 1 is created by forcing coarsening of cells smaller than the resolution of the current hierarchy level. In this case the stencil size was used as a criterion, but the physical cell size could be used as well. The process stops when the coarsest possible grid is reached, with every sub-block of the grid being covered by one cell only (figures 3.29e and 3.30e). Note that the sub-blocks of the base grid used don't have cell dimensions that are powers of two, demonstrating the algorithm's capability to use sub-block hierarchies and achieving maximum possible coarsening (see appendix A.3.2).

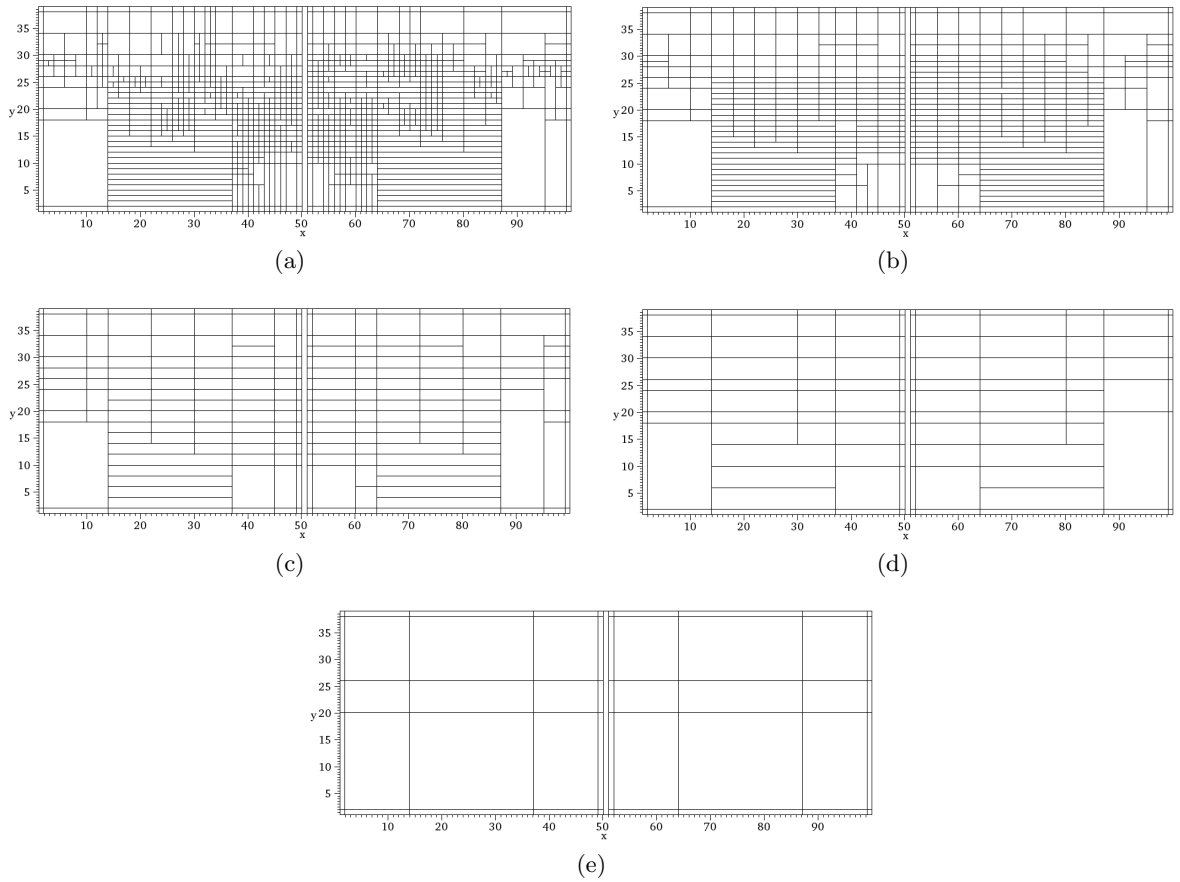


Figure 3.29.: Grid hierarchy for a double-null grid (plotted in computational space). The highest-resolution grid (a) is adapted to the electron density. Subsequent grids in the hierarchy are created by forcing the adaptation algorithm to remove cells above a certain refinement level. Grid (e) is the coarsest possible grid.

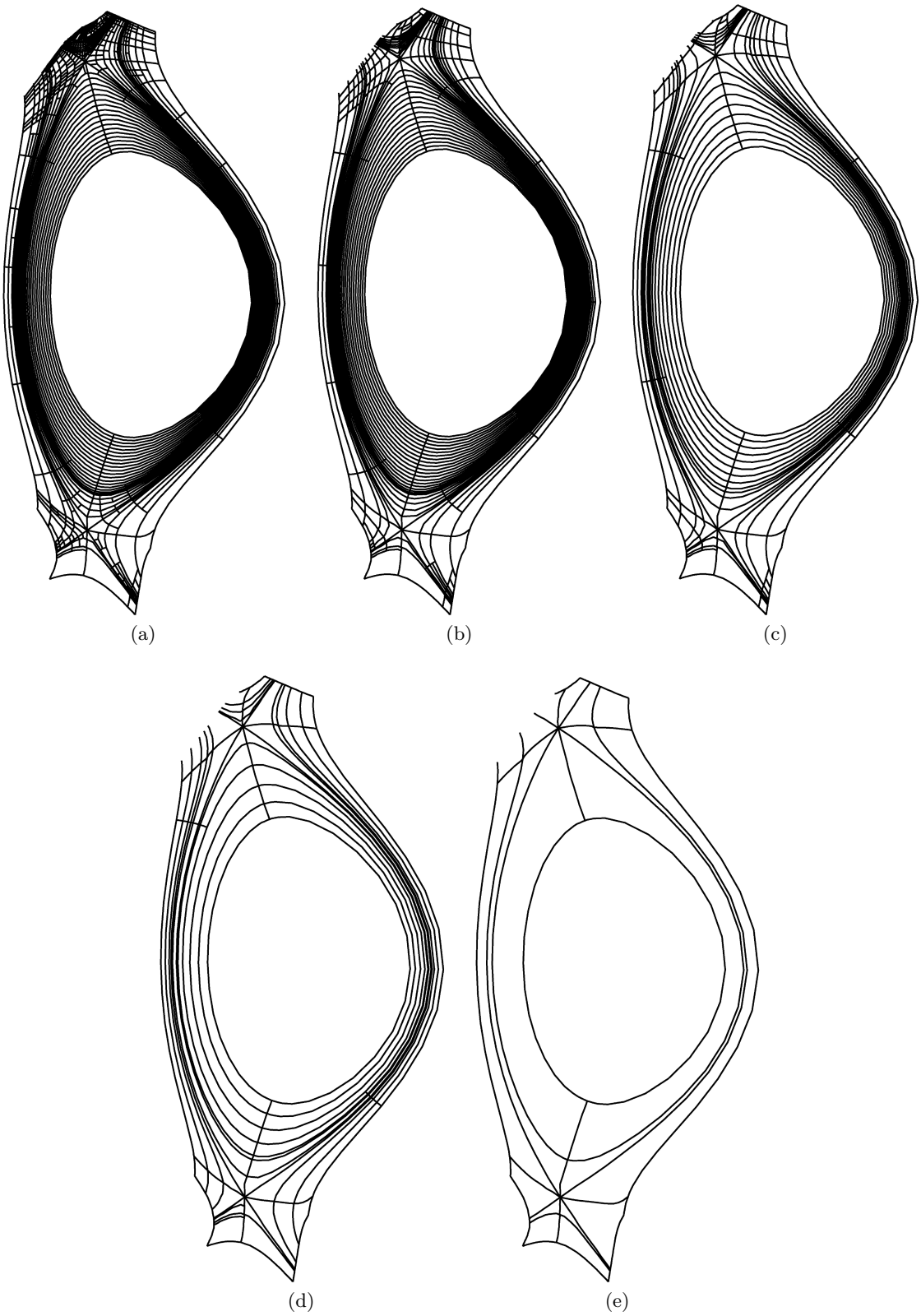


Figure 3.30.: Grid hierarchy for a double-null grid (plotted in physical space)

4. Numerical methods

B2.5 solves the nonlinear coupled model equations given in section 2.2 by sequential solution of linearized equations and relaxation of the plasma state (cf. chapter 6). The linear equations thus obtained are advection-diffusion equations with source terms. This chapter discusses finite-volume methods for their solution on structured and unstructured grids.

4.1. The advection-diffusion-equation

Let $\phi : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}, \phi = \phi(\vec{x}, t)$ be a conservative scalar quantity, e.g. energy, mass density or momentum. In a fixed volume $\Omega \subset \mathbb{R}^n$ then holds the conservation law for ϕ

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_\Omega \phi d\Omega + \int_{\partial\Omega} \vec{f} \cdot \vec{n} \; d\partial\Omega = \int_\Omega s d\Omega. \tag{4.1}$$

The flux vector $\vec{f} : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n, \vec{f} = \vec{f}(\vec{x}, t)$ gives the rate of transport (or *flux density*), $\vec{n}$ is the outward normal vector and $s : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}, s = s(\vec{x}, t)$ denotes a source term. The conservation law states that the change of the quantity ϕ in Ω is accounted for by the flux through the volume boundary $\partial\Omega$ and sources.

In the context of fluid modeling the total flux is a result of advection (synonymously called convection) and diffusion. The advective flux

$$\vec{f_a} = \vec{u}\phi \tag{4.2}$$

describes transport of the quantity ϕ due to a velocity field $\vec{u} : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}^n, u = u(\vec{x}, t)$. The diffusive flux

$$\vec{f_d} = -D\nabla\phi \tag{4.3}$$

is driven by the gradient of the quantity ϕ, resulting in transport from regions of high concentration to regions of low concentration. $D : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}_0^{+,n\times n}, D = D(\vec{x}, t)$ is a matrix of positive diffusion coefficients. The combined advection-diffusion flux is then

$$\vec{f_{ad}} = \vec{u}\phi - D\nabla\phi. \tag{4.4}$$

Applying the divergence theorem (assuming the required smoothness for $\vec{f}$, i.e. $f \in C^1(\mathbb{R}^n \times \mathbb{R}^+)$) to (4.1) and observing that the equation holds for any volume $\Omega \subset \mathbb{R}$, one can reformulate the integral equation into the differential equation

$$\frac{\partial\phi}{\partial t} + \nabla \cdot \vec{f_{ad}} = \frac{\partial\phi}{\partial t} + \nabla \cdot (\vec{u}\phi - D\nabla\phi) = s. \tag{4.5}$$

If ϕ is time-dependent and $D \neq 0$, (4.5) is of parabolic type. In the limit case of no advection ($\vec{u} = 0$), the *diffusion* or *heat equation* is obtained. In the other limiting case of no diffusion ($D = 0$), the *advection equation* in conservative or divergence form is obtained, which is of hyperbolic type. This change of type is referred to as a *singular perturbation*. In the steady-state case ($\frac{\partial \phi}{\partial t} = 0$), (4.5) is of elliptic type.

The problem for the unsteady situation is defined as follows.

Definition 4 (Initial-value problem). *Find $\phi : \Omega \times [0, T] \to \mathbb{R}$ so that*

$$\begin{aligned} \frac{\partial \phi}{\partial t} + L\phi &= s && \text{in } \Omega \times (0, T], \\ B\phi &= g && \text{on } \partial\Omega \times (0, T], \\ \phi(\vec{x}, 0) &= \phi_0(\vec{x}) \end{aligned} \tag{4.6}$$

where the operator $L = \nabla \cdot \vec{f}_{ad}$ is the flux divergence. B and g describe the boundary conditions. It is completed by specifying an initial value ϕ_0 and suitable boundary conditions.

Likewise, the steady-state boundary value problem is defined by

Definition 5 (Boundary-value problem). *Find $\phi : \Omega \to \mathbb{R}$ so that*

$$\begin{aligned} L\phi &= s && \text{in } \Omega, \\ B\phi &= g && \text{on } \partial\Omega. \end{aligned} \tag{4.7}$$

Existence and uniqueness of solutions to problems (4.6) and (4.7) are studied in the context of general first- and second-order partial differential equations [50].

4.1.1. Boundary conditions

To obtain a well-posed problem, a consistent set of boundary conditions has to be defined. For the parabolic and elliptic case we consider the conditions (with $\vec{n}$ the outward normal vector of Ω)

$$\phi = g_d \text{ on } \partial\Omega_d \quad \text{Dirichlet condition, and} \tag{4.8}$$

$$\vec{n} \cdot \nabla\phi = g_n \text{ on } \partial\Omega_n \quad \text{Neumann condition.} \tag{4.9}$$

On every part of the boundary one condition must be specified

$$\partial\Omega_d \cup \partial\Omega_n = \partial\Omega, \quad \partial\Omega_d \cap \partial\Omega_n = \emptyset. \tag{4.10}$$

If only Neumann conditions are specified, the compatibility condition

$$\int_{\partial\Omega} f_{ad} d\partial\Omega = -\int_{\Omega} s d\Omega \tag{4.11}$$

has to be fulfilled, and the solution is not unique.

In the hyperbolic case, the definition of inflow and outflow conditions has to be consistent with the characteristic curves of the problem. Dirichlet conditions are set on inflow boundaries, and appropriate Neumann conditions on outflows.

4.1.2. Expansion in curvilinear coordinates

In the presence of a time-independent coordinate mapping $T : \Omega_\xi \to \Omega_x, \vec{x} = \vec{x}(\vec{\xi})$, (4.5) can be expanded to

$$\frac{\partial \phi}{\partial t} + \frac{1}{\sqrt{g}} \left(\sum_{i=1}^{n} \frac{\partial}{\partial \xi_i} (\sqrt{g} u^i \phi(\vec{x}(\vec{\xi}))) - \sum_{j=1}^{d} \frac{\partial}{\partial \xi_j} \left(\sqrt{g}\ d^{jj}\ \frac{\partial \phi(\vec{x}(\vec{\xi}))}{\partial \xi_j} \right) \right) = s. \tag{4.12}$$

where the transport coefficients are used in contra-variant form u^i, d^{ii}. Properties of the coordinate mapping and the metric coefficient $\sqrt{g}$ are detailed in appendix A.1. In an orthogonal system (4.12) simplifies to

$$\begin{aligned} \frac{\partial \phi}{\partial t} + \nabla \cdot \vec{f} &= \frac{\partial \phi}{\partial t} + \frac{1}{\sqrt{g}} \sum_i \frac{\partial}{\partial \xi_i} \sqrt{g} \left(u^i \phi - d^{ii} \frac{\partial \phi}{\partial \xi_i} \right) \\ &= \frac{\partial \phi}{\partial t} + \frac{1}{\sqrt{g}} \sum_i \frac{\partial}{\partial \xi_i} \sqrt{g} \left(\frac{1}{h_i} \tilde{u}^i \phi - \frac{1}{h_i^2} \tilde{d}^{ii} \frac{\partial \phi}{\partial \xi_i} \right) = s \end{aligned} \tag{4.13}$$

with $\tilde{u}^i\,[\mathrm{m/s}], \tilde{d}^{ii}\,[\mathrm{m^2/s}]$ being the transport coefficients relative to the covariant basis in physical units.

4.1.3. The Péclet number

The character of the transport is described by the dimensionless Péclet number, which is defined as the ratio of advective vs. diffusive transport

$$\mathrm{Pe} = \frac{UL}{D}, \tag{4.14}$$

where $L[\mathrm{m}]$ is a typical length scale of the flow, $U[\mathrm{m/s}]$ the velocity and $D[\mathrm{m^2/s}]$ the diffusion strength. At a point $\vec{x}$, for a flow along the direction given by the unit vector $\vec{n}$, the Péclet number can be defined as

$$P_{\vec{n}} = \frac{\vec{n} \cdot \vec{u}(\vec{x})\ L}{\|D(\vec{x})\vec{n}\|}. \tag{4.15}$$

For advection-dominated flow $|\mathrm{Pe}| \to \infty$, for purely diffusive flow $\mathrm{Pe} = 0$.

4.2. Finite volume discretization

A vast number of schemes is found in the literature for the numerical solution of conservation laws of type (4.1). One class of schemes which exactly maintains conservation is the *finite volume method* (FVM) [51, 20, 74]. This property and its relatively straightforward implementation made it very popular in the field of computational fluid dynamics.

4.2.1. General finite volume scheme

The first basic idea in the derivation of the FVM is to discretize the domain Ω into cells (or *control volumes*) Ω_j (cf. figure 4.1). For every cell, the cell average of the solution ϕ

$$\phi_j(t) = A_h\phi|_{\Omega_j} \equiv \frac{1}{|\Omega_j|}\int_{\Omega_j} \phi(\vec{x},t)d\Omega \tag{4.16}$$

is defined. $A_h : L^2 \to V_h^0, \phi_h = A_h\phi$ is called the averaging operator. The solution approximation $\phi_h \in V_h^0$ is then a step function (or "grid function"), and

$$V_h^0 = \left\{v \mid v|_{\Omega_j} \in \chi(\Omega_j)\ \forall\Omega_j \in \mathcal{T}\right\} \tag{4.17}$$

is the space of piecewise constant functions with $\chi(\Omega_j)$ being the set of characteristic functions in the control volume Ω_j. $\mathcal{T}$ denotes the space discretization (cf. section 3.2.4).

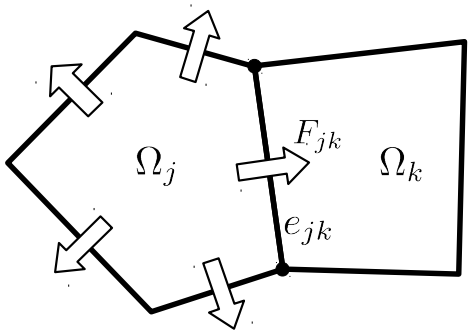


Figure 4.1.: Central elements of a finite volume discretization: cells (control volumes) Ω_j, Ω_k, face e_{jk}, face flux F_{jk}

Starting from the integral equation (4.1), the second step to derive a semi-discrete form is approximation of the boundary fluxes through the cell faces e_{jk} with numerical fluxes $F_{jk} : V_h^0 \to \mathbb{R}, F_{jk} = F_{jk}(\phi_h)$

$$\int_{\partial\Omega} \vec{f}\cdot\vec{n}\ d\partial\Omega \approx \sum_{\forall e_{jk}\in\partial\Omega_j} F_{jk}.$$

The integral face flux F_{jk} is oriented along the outward normal of Ω_j, i.e. a positive flux is from Ω_j to Ω_k. Combined with an approximation to the source term volume integral

$$S_j \approx \int_{\Omega_j} s(\vec{x},t)d\Omega_j \tag{4.18}$$

the semi-discrete scheme

$$\frac{d}{dt}\phi_j(t) + \underbrace{\frac{1}{|\Omega_j|}\sum_{\forall e_{jk}\in\partial\Omega_j} F_{jk}}_{=:L_h(\phi_h)} = \frac{1}{|\Omega_j|}S_j \tag{4.19}$$

is obtained, where L_h is the discrete approximation to the operator $L = \nabla\cdot\vec{f}$. The resulting system of ordinary differential equations can be marched forward in time using the vertical method of lines in combination with an appropriate time-integration scheme.

4.2.2. Global and local errors

A spatial or temporal discretization introduces approximation errors. This section concentrates on the errors in the steady-state situation. Definitions of time discretization errors are given in appendix B.2. Of central interest is the *(global) spatial discretization error.*

Definition 6 (**Global spatial discretization error**). *For a given grid $\mathcal{T}$ let $\phi_h^* \equiv A_h\phi \in V_h^0$ be the grid function of the exact solution ϕ and $\phi_h \in V_h^0$ a numerical approximation to it. The global error is then defined as*

$$\epsilon_h = \phi_h^* - \phi_h. \tag{4.20}$$

The (local) spatial truncation error of an operator is defined as the difference between the exact value and the result when substituting the exact grid solution ϕ_h^* into the numerical approximation of the operator.

Definition 7 (**Spatial truncation errors**). *The spatial truncation error τ_{jk}^F of the flux approximation F_{jk} at face e_{jk} is defined as*

$$|e_{jk}|\ \tau_{jk}^F = \int_{e_{jk}} \vec{f}\ \vec{n} dS - F_{jk}. \tag{4.21}$$

The spatial truncation error τ_j^L of the discrete approximation L_h to the operator L in the cell Ω_j is defined as

$$|\Omega_j|\ \tau_j^L = \int_{\Omega_j} L\phi d\Omega - |\Omega_j| L_h\phi_j^* = \int_{\Omega_j} L\phi d\Omega - \sum_{\forall e_{jk}\in\partial\Omega_j} F_{jk}(\phi_h^*). \tag{4.22}$$

The definition of τ_j^L reflects the fact that the finite volume method does not evaluate L pointwise but as a volume integral. The idea is that $L\phi = L_h\phi_h^* + \tau_h^L$. For a linear equation, the relation between ϵ_h and τ_h^L can be shown to be [53]

$$L_h\epsilon_h = -\tau_h^L. \tag{4.23}$$

The local truncation error acts as a source for the discretization error, which in this case is advected and diffused by L_h. For non-linear equations, the relation is not as simple. If the error is small enough, a local linearization around the solution is feasible, allowing to apply the same concept. Knowledge about the truncation error allows the estimation of the global error via equation (4.23).

Definition 8 (**Consistency and convergence order**). *A scheme is called consistent of order q if*

$$\|\tau_h^L\| = \mathcal{O}(h^q) \quad \text{for } h \to 0$$

where h is a measure of the grid spacing, e.g. $h = \max_{\forall\Omega_j}(\Delta x, \Delta y)$. A scheme is convergent of order p if

$$\|\epsilon_h\| = \mathcal{O}(h^p) \quad \text{for } h \to 0.$$

The coefficients p, q are derived by substituting a Taylor expansion of ϕ into the scheme and determining the error terms directly, or alternatively by numerical grid refinement experiments. On coarse grids the high-order error terms can dominate the error, and thus the convergence rates suggested by the order of the scheme are only realized on sufficiently fine grids where monotonic convergence behavior is observed.

4.2.3. Error measures

When quantifying errors and residuals, the choice of norms is important. A common approach is to use an average of the absolute values to some power p

$$\|\epsilon_h\|_p = \left(\frac{1}{N} \sum_j |\epsilon_j^p| \right)^{1/p} \tag{4.24}$$

where mostly $p = 1, 2, \infty$ are considered. Additional care has to be taken on nonuniform grids. Consider a solution-adaptive algorithm that increases the grid resolution in parts of the domain with a high error measure (e.g. steep gradients or shocks) where $\mathcal{O}(1)$ errors occur. This localizes the error in (possibly many) small cells, but as all cells have the same weight in eq. (4.24), the error measure might not decrease. Therefore, as pointed out in [110], some sort of volume weighting must be applied. Possible choices are the analogues to function-space norms given in [74]

$$\|\epsilon_h\|_p = \left(\sum_j |\Omega_j| |\epsilon_j|^p \right)^{1/p} \tag{4.25}$$

or the similar volume-averaged measures presented in [110]

$$\|\epsilon_h\|_p = \left(\frac{\sum_j |\Omega_j| |\epsilon_j|^p}{\sum_j |\Omega_j|} \right)^{1/p}. \tag{4.26}$$

4.2.4. Numerical flux functions

Central to the design of a consistent finite volume scheme is the approximation of face fluxes. Because the scheme only keeps track of the cell averages ϕ_j, the solution is not continuous but multi-valued at cell faces. In general, the flux density f is replaced by a numerical flux function $F : \mathbb{R} \times \mathbb{R} \to \mathbb{R},\ F = F(\phi^-, \phi^+)$, which computes the flux density depending on the solution state $\phi^\pm$ to both sides of a face (in the case of higher-order schemes, the state might also include further quantities like the derivatives, e.g. $F = F(\phi^-, \phi^+, (\nabla\phi)^-, (\nabla\phi)^+)$.

The total flux F_{jk} through face e_{jk} is then computed using a numerical quadrature rule

$$F_{jk} = \sum_{q=1}^{Q} \omega_q F\left(\phi_{jk}^-(x_q), \phi_{jk}^+(x_q) \right) \tag{4.27}$$

where $x_q \in e_{jk}$ are the quadrature points and ω_q the associated quadrature weights. The accuracy of the scheme obviously depends on the order of the quadrature rule. Its choice is strongly influenced by the approximation accuracy of solution states at the quadrature points and the geometry description [36].

Central requirements for the numerical fluxes are *consistency* and *conservativeness*.

Definition 9 (**Consistent and conservative numerical flux**). *A numerical flux is called* consistent *if*

$$F_{jk} = \int_{e_{jk}} \vec{f}(\phi) \cdot \vec{n}_{jk} dS \quad \text{for constant } \phi. \tag{4.28}$$

It is conservative *if*

$$F_{jk} = -F_{kj}. \tag{4.29}$$

Consistency states that if the solution state on both sides of the face is identical, the original flux $\vec{f}$ is obtained. This guarantees that the scheme correctly approximates the original flux f. Conservation states that the contribution of F_{jk} to the flux balance equations of Ω_j and Ω_k cancels exactly, ensuring exact conservation of the quantity ϕ over the whole domain Ω.

4.2.5. Positive schemes

Consistency and conservation of the numerical flux is not enough to ensure convergence of the numerical solution ϕ_h. A common problem observed in the discretization of conservation laws is the occurrence of unphysical solution oscillations even in steady-state computations. This is not a sign of instability of the scheme (as the concept of stability relates boundedness of the solution to perturbations in the input data) but shows that that the discretization does not reproduce maximum principles present in the original equation. This is especially problematic for strictly positive quantities like densities and temperatures. Therefore schemes are designed that reproduce these maximum principles in discrete form. Such schemes are often called *monotone* or of *positive type*. This section sums up the central results establishing positivity for the finite volume methods discussed in section 4.3 and 4.4.

Theorem 1 (**Maximum principle for the advection-diffusion equation** [93]). *Consider the stationary advection-diffusion equation (cf. (4.5))*

$$\sum_i u_i \frac{\partial \phi}{\partial x_i} - \sum_{i,j} d_{ij} \frac{\partial \phi}{\partial x_i \partial x_j} = s, \quad \vec{x} \in \Omega \subset \mathbb{R}^n \tag{4.30}$$

with $s \leq 0$. Let $\partial\Omega$ be smooth and for all outward derivatives on $\partial\Omega' \subset \partial\Omega$ hold $\partial\phi/\partial\vec{n} \leq 0$. Then either local maxima occur only on $\partial\Omega \backslash \partial\Omega'$ or $\phi = const$ in Ω.

Proof. See [93]. □

An equivalent minimum principle is obtained by reversing the sign of ϕ. A discrete equivalent (in one dimension to simplify notation) is stated by the following theorem.

Theorem 2 (**Discrete maximum principle** [119]). *Let c_i and $x_i \in \mathbb{R}, i = 1, \ldots, N$ satisfy*

$$\sum_{i=1}^{N} c_i = 0, \quad c_i < 0 \ \text{ for } i > 1 \quad \text{and} \quad \sum_{i}^{N} c_i x_i \leq 0.$$

Then

$$x_i = x_1 \ \text{ for } \ i = 1, \ldots, N \quad \text{or} \quad x_1 < \max\{x_i : i > 1\}.$$

Proof. See [119]. □

Equivalent statements hold for the time-dependent case [119]. The following theorem establishes the maximum principle for scheme (4.19) combined with forward Euler time-stepping.

Theorem 3 (**Local extremum diminishing (LED) scheme** [18]). *Consider the fully discrete scheme*

$$\phi_j^{n+1} = \phi_j^n + \frac{\Delta t}{|\Omega_j|} \sum_{\forall e_{jk} \in \partial\Omega_j} c_{jk}(\phi_h^n)(\phi_k^n - \phi_j^n). \tag{4.31}$$

If the coefficients c_{jk} satisfy

$$c_{jk}(\phi_h^n) \geq 0 \quad \forall e_{jk} \in \partial\Omega_j \tag{4.32}$$

and the time step is restricted by

$$1 - \frac{\Delta t}{|\Omega_j|} \sum_{\forall e_{jk} \in \partial\Omega_j} c_{jk}(\phi_h^n) \geq 0, \tag{4.33}$$

the scheme exhibits the local space-time discrete maximum principle on the time interval $[t^n, t^{n+1}]$

$$\min_{\forall e_{jk} \in \partial\Omega_j} (\phi_j^n, \phi_k^n) \leq \phi_j^{n+1} \leq \max_{\forall e_{jk} \in \partial\Omega_j} (\phi_j^n, \phi_k^n). \tag{4.34}$$

Proof. See [18]. □

Theorem 3 establishes the necessary conditions for global boundedness of the solution in the time-dependent case and a discrete maximum principle in the steady-state case. Requirement (4.32) can alternatively be formulated by writing the scheme in the form

$$L_h \phi_h = A \cdot \phi_h = \sum_i \sum_j a_{ij}(\phi_h)\phi_j, \quad A \in \mathbb{R}^{N \times N}. \tag{4.35}$$

The operator L_h is said to be of *positive type* if

$$a_{ij} < 0 \quad \text{for} \quad i \neq j, \quad \sum_j a_{ij} = 0 \tag{4.36}$$

holds. Particularly attractive due to their simple implementation also for implicit discretization are schemes that result in a linear operator L_h. However, the following negative result bounds the accuracy of linear discretizations of the advection equation.

Theorem 4 (**Godunov order barrier** [56])**.** *Linear one-step second-order accurate numerical schemes for the advection equation*

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = 0, \quad t > 0, x \in \mathbb{R}, c \in \mathbb{R} \tag{4.37}$$

cannot be monotonicity preserving, unless $|c|\frac{\Delta t}{\Delta x} \in \mathbb{N}$.

Proof. See [56]. □

The stated restriction of the Courant number is not practical for general applications. This result extends to multi-step schemes and multiple dimensions. To overcome it, nonlinear schemes have to be used. *Total variation diminishing (TVD)* schemes have been designed for this purpose, however they are limited to first-order accuracy in the multi-dimensional case [57]. A class of schemes that can realize high-order accuracy in the multi-dimensional case and generalizes to unstructured grids are nonlinear positive coefficient schemes satisfying the conditions in theorem 3. The limited linear-reconstruction high-resolution scheme described in section 4.4 falls in this category.

Theorem 5. (Maximum principle for linear reconstruction FVM on unstructured grids [18]) *Let* ϕ_j^{min} *be the minimum and* ϕ_j^{max} *be the maximum solution cell averages for cell* Ω_j *and all adjacent cells, i.e.*

$$\phi_j^{min} \equiv \min_{\forall e_{jk} \in \partial\Omega_j} (\phi_j, \phi_k) \quad \text{and} \quad \phi_j^{max} \equiv \max_{\forall e_{jk} \in \partial\Omega_j} (\phi_j, \phi_k). \tag{4.38}$$

The fully discrete finite volume scheme

$$\phi_j^{n+1} = \phi_j^n + \frac{\Delta t}{|\Omega_j|} \sum_{\forall e_{jk} \in \partial\Omega_j} F_{jk} \quad \forall \Omega_j \in \mathcal{T} \tag{4.39}$$

with F_{jk} *given by equation (4.27) with nonnegative quadrature weights, Lipschitz continuous flux function* F *and linear reconstruction to quadrature points*

$$\phi_{jk}^-(\vec{x}) \equiv \lim_{\epsilon \downarrow 0} R_1^0(x - \epsilon \vec{n}_{jk}(\vec{x}); \phi_h), \quad \vec{x} \in e_{jk}, \quad \phi_h \in V_h^0 \tag{4.40}$$

$$\phi_{jk}^+(\vec{x}) \equiv \lim_{\epsilon \downarrow 0} R_1^0(x + \epsilon \vec{n}_{jk}(\vec{x}); \phi_h), \quad \vec{x} \in e_{jk}, \quad \phi_h \in V_h^0 \tag{4.41}$$

(where the reconstruction operator R *is given by definition 10 and* $\vec{n}_{jk}$ *is the outward normal of* e_{jk}*) exhibits for every* $\Omega_j \in \mathcal{T}$ *the local space-time maximum principle*

$$\min_{\forall e_{jk} \in \partial\Omega_j} (\phi_j^n, \phi_k^n) \leq \phi_j^{n+1} \leq \max_{\forall e_{jk} \in \partial\Omega_j} (\phi_j^n, \phi_k^n). \tag{4.42}$$

if the linear reconstruction satisfies for the flux quadrature points $\vec{x}_q \in e_{jk}, q = 1, \ldots, Q, e_{jk} \in \partial\Omega_j$

$$\max(\phi_j^{\min,n}, \phi_k^{\min,n}) \leq \phi_{jk}^{-,n}(x_q) \leq \min(\phi_j^{\max,n}, \phi_k^{\max,n}). \tag{4.43}$$

and the time step is restricted by

$$1 - \frac{\Delta t}{|\Omega_j|}\Gamma^{geom} \sum_{\substack{\forall e_{jk} \in \partial\Omega_j \\ 1 \leq q \leq Q}} \sup_{\substack{\phi^1 \in [\phi_j^{\min,n}, \phi_j^{\max,n}] \\ \phi^2 \in [\phi_j^{\min,n}, \phi_j^{\max,n}]}} \left| \frac{\partial F_{jk}}{\partial \phi^+}(\vec{n}_{jk}(\vec{x}_q); \phi^1, \phi^2) \right| \geq 0 \tag{4.44}$$

with Γ^{geom} *a factor depending on cell geometry (*$\Gamma^{geom} = 2$ *for rectangles). In steady-state under the same assumptions the local spatial maximum principle*

$$\min_{\forall e_{jk} \in \partial\Omega_j} \phi_k \leq \phi_j \leq \max_{\forall e_{jk} \in \partial\Omega_j} \phi_k \tag{4.45}$$

holds.

Proof. See [18]. □

4.3. Classical finite volume schemes

Current versions of the B2.5 use a variant of the linear finite volume scheme described in [90]. It continues to form the basis discretization in B2.6-structured (cf. chapter 6) and is used as a low-order approximation in defect correction procedure used in the HR FVM described in section 4.4.

4.3.1. The Patankar hybrid scheme

The Patankar scheme [90] uses a central difference approximation for the diffusion flux (4.3). The advection flux (4.2) approximation combines the central (CDS) and upwind (UDS) difference scheme depending on the local numerical Péclet number (details on the CDS and UDS scheme can be found in appendix B.1). The goal is to avoid the unfavorable first-order accuracy of the upwind scheme where possible while using its positivity properties where needed to satisfy the positive coefficient condition of theorem 3.

To derive the scheme, the dimensionless flux density

$$f^* = P\phi - \frac{\partial \phi}{\partial x}$$

is considered in the simplified geometry given in figure 4.2. P is the Péclet number for the flux along $\vec{n}$ given by (4.15). The solution value ϕ at the face is assumed to be a linear combination of ϕ_d and ϕ_i, while the gradient is a multiple of $\phi_i - \phi_d$. Thus

$$f^* \approx B(P)\phi_d - A(P)\phi_i$$

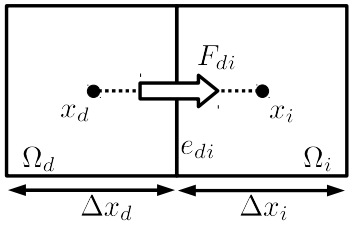


Figure 4.2.: Geometry for 1d flux (4.46)

with A, B dimensionless coefficients depending on P. Using a symmetry argument and exploiting the fact that $f = \vec{u}\phi_d = \vec{u}\phi_i$ for a constant solution $\phi_d = \phi_i$, the coefficients can be simplified to

$$A(P) = A(|P|) + [\![-P, 0]\!], \quad B(P) = A(-P).$$

with $[\![\cdot, \cdot]\!] = \max(\cdot, \cdot)$. The combined numerical advection-diffusion flux is then

$$F_{di} = |e_{di}| \left[(D_{di} A(|P_{di}|) + [\![F^a_{di}, 0]\!])\phi_d - (D_{di} A(|P_{di}|) + [\![-F^a_{di}, 0]\!])\phi_i \right]. \tag{4.46}$$

$F^a_{di} = \|\vec{u}_{di}\vec{n}\|$ is the advection flux and D_{di} the diffusion coefficient at the face. The numerical Péclet number at the face e_{di} is

$$P_{di} = \frac{\vec{u}_{di}\vec{n}L_{di}}{D_{di}} \quad \text{with} \quad L_{di} = \frac{\Delta x_d + \Delta x_i}{2} \tag{4.47}$$

Choice of function $A(|P|)$

To complete the scheme $A(|P|)$ has to be defined. A choice of functions is presented in table 4.1 and figure 4.3. UDS and CDS (cf. appendix B.1) are included as special cases. The exponential scheme recovers the exact solution for (4.5) in the one-dimensional homogeneous case (for constant u, D and Dirichlet boundary conditions). The hybrid schemes are motivated by a search for computationally less expensive replacements for it. The B2.5 hybrid scheme defines the discretization currently used in the B2.5 code. In practice the schemes perform very similar.

Scheme	$A(\|P\|)$
Upwind difference (UDS)	1
Central difference (CDS)	$1 - 0.5\|P\|$
Hybrid	$[\![0, 1 - 0.5\|P\|]\!]$
B2.5 hybrid	$\sqrt{1 + \frac{1}{4}\|P\|^2} - \frac{1}{2}\|P\|$
Exponential	$\|P\|/(\exp(\|P\|) - 1)$

Table 4.1.: Choices of switch functions $A(|P|)$ for the Patankar flux (4.46)

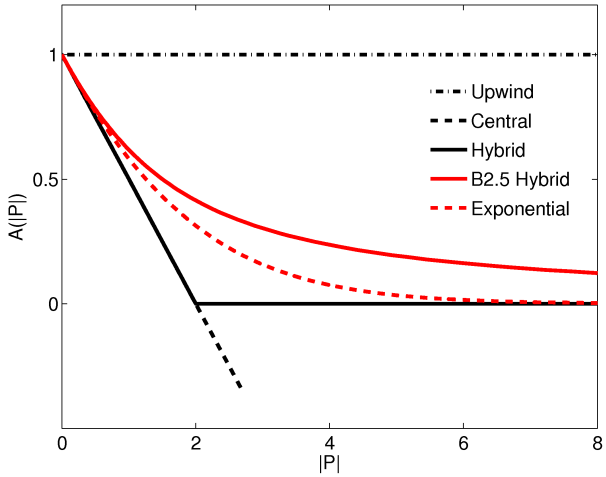


Figure 4.3.: $A(|P|)$ functions for the Patankar flux (4.46) given in table 4.1

Application to multidimensional domains

The linear flux function (4.46) is derived for the one-dimensional situation shown in figure 4.2. For certain grids the generalization to the multi-dimensional case is straightforward, as long as the geometry assumptions made in the derivation are not violated.

First, the central difference approximation assumes a linear profile of the solution along the line $\overline{x_j x_k}$ connecting the cell centroids. This line has to be orthogonal to the face, i.e. $\overline{x_j x_k} \perp e_{jk}$. Second, midpoint quadrature is assumed. Therefore the intersection point $x_{jk} = \overline{x_j x_k} \cap e_{jk}$ has to coincide with the face midpoint.

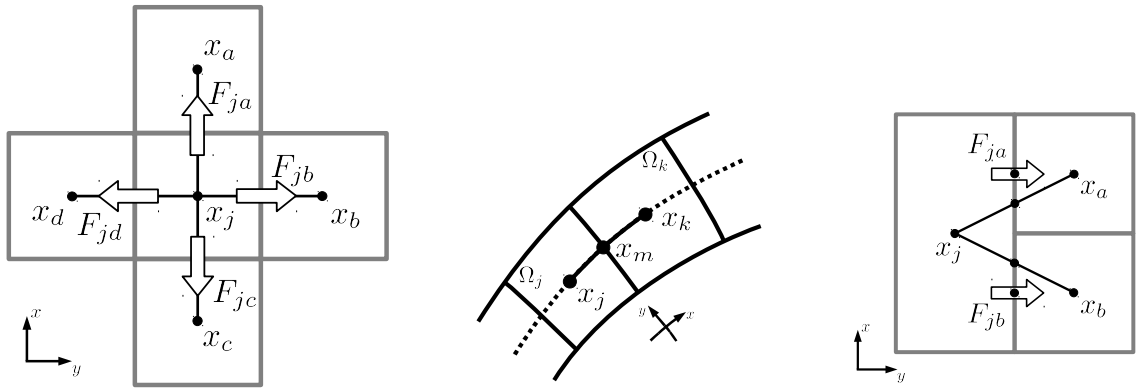
Structured Cartesian (figure 4.4a) and logically rectangular coordinate-mapped grids (figure 4.4b, as used in B2) satisfy these constraints. For unstructured grids with more complex cell geometry (figure 4.4c) this is not the case. Violation of the assumptions degrades the accuracy of the approximation. This might be acceptable within some tolerances. The effect is studied in the numerical benchmark presented in chapter 5.

4.3.2. Flux balance equations

Assembling the flux balance for the control volume Ω_j by summing the contribution from the individual face contributions yields the algebraic equation

$$\int_{\partial\Omega_j} \vec{f} \cdot \vec{n} d\Omega \approx \sum_{e_{jk} \in \partial\Omega_j} F_{jk} = \sum_{e_{jk} \in \partial\Omega_j} a_{jk,j}\phi_j + a_{jk,k}\phi_k = \sum_{i \in \mathcal{S}_j} a_i \phi_i = S_j \tag{4.48}$$

where S_j is an approximation of the source volume integral $\int_{\Omega_j} S d\Omega$ and $\mathcal{S}_j$ is the the *stencil* of control volume j, which is a set of indices $\{j, k | e_{jk} \in \partial\Omega_j\}$ of Ω_j and its neighbors.



(a) Structured Cartesian grid (b) Curvilinear logically rectangular grid (c) Unstructured grid

Figure 4.4.: Flux balance for multidimensional situation: grid geometries

The linear coefficients a_i are prescribed by the choice of numerical flux function. For the Patankar flux (4.46) they are for cell Ω_j

$$a_k = -\left(D_{jk}A(|P_{jk}|) + [\![-F^a_{jk}, 0]\!]\right) \quad \forall k \in \mathcal{S}_j, k \neq j \tag{4.49}$$

$$\begin{aligned} a_j &= \sum_{k\in\mathcal{S}_j, k\neq j} \left(D_{jk}A(|P_{jk}|) + [\![\ F^a_{jk}, 0]\!]\right) \\ &= \sum_{k\in\mathcal{S}_j, k\neq j} (-a_k) \quad - \sum_{k\in\mathcal{S}_j, k\neq j} F^a_{jk} \end{aligned} \tag{4.50}$$

where the identity $[\![-F, 0]\!] = [\![F, 0]\!] - F$ was used.

The linear equation system

Stating the linear flux balance equation 4.48 for all n control volumes $\Omega_j \in \mathcal{T}$ results in a linear system

$$A\vec{\phi}_h = \vec{S}_h, \quad A \in \mathbb{R}^{n\times n}. \tag{4.51}$$

The matrix A is sparse with $\mathcal{O}(n)$ nonzero entries depending on the stencil size. Its structure is given by the connectivity of the grid $\mathcal{T}$. For structured grids with lexicographical ordering this leads to band matrices which can be stored efficiently. This simple structure is lost when using nontrivial connectivity or unstructured grids (see appendix B.4 for examples).

Solution methods for linear systems are not in the central focus of this thesis. Some notes on the solver used in the implementation are given in appendix B.3.

4.3.3. Boundary conditions

Boundary conditions for scheme (4.50) can be imposed using vanishingly small *ghost cells* at the outside boundary of the simulation domain Ω. This simplifies algorithms by allowing to treat boundary faces in most cases exactly like interior faces. Dirichlet conditions are implemented by prescribing the solution value d in the ghost cell, Neumann conditions by fixing the central difference approximation between ghost and interior cells to the prescribed value g. For the simplified geometry in figure 4.2 this means (with Ω_i being the interior, Ω_d the ghost cell)

$$\phi_d = d \quad \text{(Dirichlet)} \quad \text{and} \quad \frac{2(\phi_i - \phi_d)}{\Delta x_d + \Delta x_i} = g \quad \text{(Neumann)}. \tag{4.52}$$

In general, the flux balance equation for ghost cells differs from interior cells and requires special treatment[1] when assembling system (4.51). Furthermore, due to the vanishingly small volume of the ghost cells, (4.52) is equivalent to using one-sided differences with an $\mathcal{O}(\Delta x)$ truncation error. This does not affect the performance of second-order schemes [53, 119].

4.4. High-resolution finite volume schemes

The Patankar scheme presented in the previous section has two drawbacks. First, as a linear scheme it is limited to first order accuracy for advection-dominated flows. (cf. theorem 4). Second, the central diffusion approximation breaks down for general geometries like unstructured grids. The solution to the first problem is to introduce nonlinear schemes, which are often referred to as *high-order* or *high-resolution (HR)* finite volume schemes. The second problem calls for a truly multidimensional approach.

4.4.1. Reconstruction schemes

The first central property of the general finite volume scheme outlined in section 4.2.1 is the solution representation using cell averages defined by equation (4.16), or written with the averaging operator $A_h : L^2 \to V_h^0, \phi_h = A_h\phi$ as

$$\phi_j = A_h\phi|_{\Omega_j} = \frac{1}{|\Omega_j|}\int_{\Omega_j} \phi(\vec{x}, t)d\Omega.$$

Starting from this cell average representation, a general approach to the design of high-resolution schemes is solution reconstruction. The idea was first introduced as an extension to the Godunov scheme in [114], leading to the MUSCL scheme for structured grids. The idea was then extended to unstructured grids [19]. The following section mainly follows [18].

[1] In iterative solvers like the B2.5 algorithm this is avoided by prescribing modified linearized source terms for the ghost cells ("strongly conditioned method"). However, this approach is not applicable for general solvers.

Definition 10 (**Reconstruction operator** [18]). *A reconstruction operator*

$$R_p^0 : V_h^0 \to V_h^p, \quad R_p^0 = R_p^0(\vec{x}, \phi_h) \tag{4.53}$$

recovers a piecewise p-th order polynomial in every cell from the cell-average solution representation ϕ_h. It is called p-exact *if it recovers polynomials of order p exactly ($R_p^0 A_h|_{\mathcal{P}_p} = I$, $\mathcal{P}_p$ being the space of polynomials of degree at most p). Furthermore the mean value in the cell has to be conserved, i.e.*

$$\frac{1}{|\Omega_j|} \int_{\Omega_j} R_p^0(\vec{x}, \phi_h) d\Omega = \phi_j. \tag{4.54}$$

or $A_h R_p^0 = I$.

In general, the reconstruction is discontinuous at cell faces. R_p^0 acts as an approximate inverse to the cell-average operator A_h and can be used to reconstruct the solution state at quadrature points to increases the accuracy of the flux approximations used in (4.19).

Various reconstruction methods are discussed in the literature. The *Essentially Non-Oscillatory (ENO)* method in its weighted or unweighted form [87, 104] selects an optimal stencil with the aim of reducing oscillations in the reconstruction to the level of the local discretization error. *Green-Gauss integral path* reconstructions form a face path around the reconstruction point and use the divergence theorem to find an approximation of the gradient [11, 42]. Least-square-methods construct an optimal fit for the coefficients of the multidimensional polynomial by solving a least-square problem for every control volume.

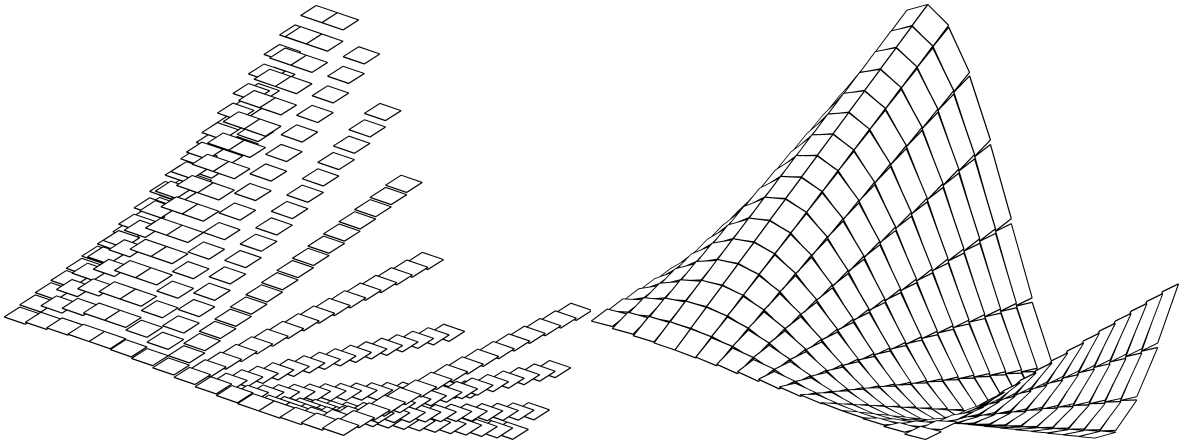
An important property for all these schemes is *compactness*, i.e. minimizing stencil size due to accuracy, stability and efficiency considerations. A p-th order polynomial in n dimensions has $\binom{p+n}{n}$ degrees of freedom, requiring a minimal set of neighbor cells of this size in the reconstruction stencil. With growing stencil size the data used for the reconstruction comes from increasingly distant cells. This decreases the robustness of the scheme, necessitating some form of data weighting [11, 12]. Furthermore, efficiency starts to suffer. Especially on unstructured grids, finding neighbor cells beyond the next-neighbor connectivity is either time-consuming or requires some sort of pre-processing and associated stencil storage.

4.4.2. Linear reconstruction on unstructured grids

The simplest reconstruction operator R_1^0 recovers a piecewise linear function

$$R_1^0(\vec{x}; \phi_h)|_{\Omega_j} = \phi_j + (\nabla \phi_h)_{\Omega_j} \cdot (\vec{x} - \vec{x}_j) \tag{4.55}$$

in every control volume Ω_j. An example for the reconstruction is shown in figure 4.5. Combining R_1^0 with the flux functions given in section 4.4.3 results in a scheme with a truncation error of $\mathcal{O}(h^2)$, where h is a representative grid spacing. This is verified by numerical experiments in section 5.1.



(a) Cell averages (b) Linear reconstruction

Figure 4.5.: Solution representations for the benchmark problem given in section 5.1

Compared to more accurate higher-order reconstructions ($p > 1$), linear reconstruction offers two advantages. First, only one neighbor cell per dimension is required and therefore next-neighbor connectivity information is sufficient for stencil selection. Second, when choosing the centroid $\vec{x}_j = \vec{x}_j^c$ of the cell as support point for the linear function[2], constraint (4.54) is trivially fulfilled. This property also extends to coordinate mapped grids for a proper choice of $\vec{x}_j^c$. For higher-order reconstructions ($p > 1$) this has to be enforced explicitly (see e.g. [88]).

Least-square gradient reconstruction

Linear reconstruction reduces to *gradient reconstruction* at the cell centroid. An approximation of $(\nabla\phi_h)_{\Omega_j}$ has to be computed in the centroid $\vec{x}_j^c$ of every cell Ω_j. This work focuses on the least-square method for gradient reconstruction. It can easily be applied to arbitrary grids while maintaining favorable robustness and accuracy properties. It is also easily extended to recover higher-order polynomials [88].

The least-squares system for control volume Ω_j is assembled by projecting the gradient $(\nabla\phi_h)_{\Omega_j}$ onto the vector connecting the centroids to obtain

$$(\nabla\phi_h)_{\Omega_j} \cdot (\vec{x}_k - \vec{x}_j) = \phi_k - \phi_j \quad \forall e_{jk} \in \partial\Omega_j$$

for all neighbor cells of Ω_j. Renumbering the cell indices locally (see figure 4.6) so that $\Omega_j = \Omega_0$ and $\Omega_i, i = 1, \ldots, N$ its neighbors and defining $(\Delta x_i, \Delta y_i) = \Delta\vec{x}_{0i} = \vec{x}_j - \vec{x}_0$, this gives the system

$$\begin{pmatrix} w_1\Delta x_1 & w_1\Delta y_1 \\ \vdots & \vdots \\ w_N\Delta x_N & w_N\Delta y_N \end{pmatrix} (\nabla\phi_h)_{\Omega_0} = \begin{pmatrix} w_1(\phi_1 - \phi_0) \\ \vdots \\ w_N(\phi_N - \phi_0) \end{pmatrix}. \tag{4.56}$$

[2]This can be understood as a *cell-centered finite volume method.*

or with shorthands

$$\left[\vec{L}_1\ \vec{L}_2\right](\nabla\phi_h) = \vec{d}. \tag{4.57}$$

Note that $\vec{L}_{1,2}$ only contain geometry information and can be precomputed. Additionally, the individual equations are multiplied with weights w_i for stability reasons. A common prescription is $w_i = 1/\|\Delta\vec{x}_{0i}\|$, giving a higher weight to data at points closer to $\vec{x}_0$ [12, 11].

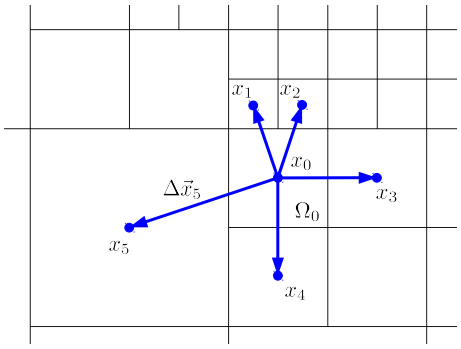


Figure 4.6.: Local geometry of the stencil used for least-square gradient reconstruction in cell Ω_0

Except in special cases (e.g. boundary cells), the linear system (4.57) is overdetermined and has to be treated as a linear least-squares fit problem. This problem is in general ill-conditioned, especially for non-smooth solutions where the contributions from different neighbors diverge. Instead of using the direct solution approach via the normal equations, the more robust method of Householder transformations and QR factorization is advocated [18, 88].

In the simple case of gradient recovery, the explicit solution

$$\nabla\phi = \frac{1}{l_{11}l_{22} - l_{12}^2}\begin{pmatrix} l_{22}(\vec{L}_1\cdot\vec{d}) - l_{12}(\vec{L}_2\cdot\vec{d}) \\ l_{11}(\vec{L}_2\cdot\vec{d}) - l_{12}(\vec{L}_2\cdot\vec{d}) \end{pmatrix} \tag{4.58}$$

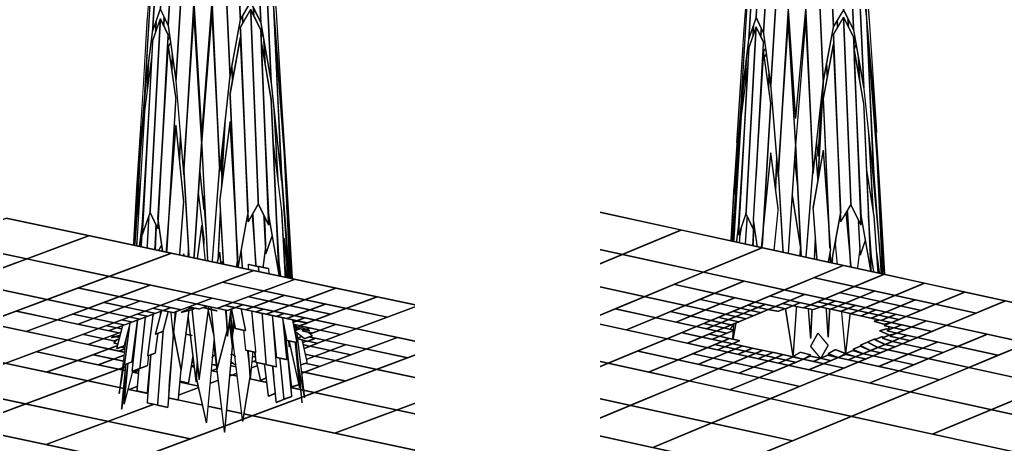
to (4.57) can be given [18], where $l_{ij} = \vec{L}_i\cdot\vec{L}_j$.

Face-based implementation

As noted in [18], when using the stencil given by the face-neighbors of a cell, (4.58) can be implemented efficiently using an algorithm iterating over the faces of the grid. Algorithms of this type benefit from the face-based UG data structure described in section 3.2.2. Returning to the global face numbering, for every space dimension two coefficients per face e_{jk} have to be computed, representing the contribution of the face difference $\Delta\phi_{jk} = \phi_k - \phi_j$ to the gradients of cells Ω_j and Ω_k. For $N = 2$ dimensions, the coefficients are

$$c_{jk}^{j,x} = \lambda_j w_{jk}^2\left(l_{22}^j\Delta x_{jk} - l_{12}\Delta y_{jk}\right) \qquad c_{jk}^{k,x} = \lambda_k w_{jk}^2\left(l_{22}^k\Delta x_{jk} - l_{12}\Delta y_{jk}\right) \tag{4.59}$$

$$c_{jk}^{j,x} = \lambda_j w_{jk}^2\left(l_{22}^j\Delta x_{jk} - l_{12}\Delta y_{jk}\right) \qquad c_{jk}^{k,x} = \lambda_k w_{jk}^2\left(l_{22}^k\Delta x_{jk} - l_{12}\Delta y_{jk}\right), \tag{4.60}$$



(a) Unlimited reconstruction exceeding solution minimum at cone base

(b) Limited reconstruction

Figure 4.7.: Limiting for the time-dependent benchmark in section 5.2

with coefficients

$$\lambda_i = \frac{1}{l_{i,11} l_{i,22} - l_{i,12}^2} \quad \text{and} \quad \begin{pmatrix} \Delta x_{jk} \\ \Delta y_{jk} \end{pmatrix} = \begin{pmatrix} x_k - x_j \\ y_k - y_j \end{pmatrix}. \tag{4.61}$$

After pre-computing the coefficients, the reconstruction step can then be implemented as a linear loop over the faces which only needs the face $\rightarrow$ cell connectivity, as shown in algorithm 8. Note that the sign skip $\Delta\vec{x}_{jk} = -\Delta\vec{x}_{kj}$ is canceled by $\Delta\phi_{jk} = -\Delta\phi_{kj}$.

Algorithm 8: Face-based least squares gradient reconstruction

Pre-compute $c_{jk}^{j,x}, c_{jk}^{j,y}, c_{jk}^{k,x}, c_{jk}^{k,y}$

foreach *cell* Ω_j **do**

  $(\nabla\phi)_j = \vec{0}$

foreach *face* e_{jk} **do**

  $\Delta\phi_{jk} = \phi_k - \phi_j$

  $(\nabla\phi)_j = (\nabla\phi)_j + \begin{pmatrix} c_{jk}^{j,x} \\ c_{jk}^{j,y} \end{pmatrix} \Delta\phi_{jk}, \quad (\nabla\phi)_k = (\nabla\phi)_k + \begin{pmatrix} c_{jk}^{k,x} \\ c_{jk}^{k,y} \end{pmatrix} \Delta\phi_{jk}$

Limiting

The solution reconstruction R_0^1 can introduce unwanted artificial extrema as e.g. shown in figures 4.7 and 4.8. This violates condition (4.43) of theorem 5 and results in a non-positive scheme. To avoid this the reconstruction is modified using *limiters*. A general theory of *flux*

limiters exists for one-dimensional schemes on structured grids [112]. A diffusive first-order flux like the hybrid flux (4.46) is combined with a higher-order flux to form

$$F_{i+1/2} = F^{low}_{i+1/2} + \psi(r) F^{high}_{i+1/2}, \quad r = \frac{\phi_i - \phi_{i-1}}{\phi_{i+1} - \phi_i}. \tag{4.62}$$

If the flux limiter function $\psi(r)$ satisfies conditions commonly summarized in the *Sweby diagram* [112], the resulting scheme is total-variation diminishing (TVD). More natural for reconstruction schemes are *slope limiters* that do not modify the resulting flux but the reconstruction directly (cf. figures 4.7, 4.8). The modified reconstruction is

$$\tilde{R}^0_1(\vec{x}; \phi_h)|_{\Omega_j} = \phi_j + \lambda_{\Omega_j} (\nabla \phi_h)_{\Omega_j} \cdot (\vec{x} - \vec{x}_j) \tag{4.63}$$

with λ_{Ω_j} a scalar gradient limiter in cell Ω_j. On structured grids, a direct relation between flux and slope limiters exists [105], allowing reuse of known limiters. However, the formulation (4.62) using forward and backward differences breaks down on nonuniform and unstructured grids [23], requiring a more general approach. A well-known general limiter for linear reconstruction schemes on unstructured grids is the *Barth-Jespersen* limiter.

Definition 11 (**Barth-Jespersen (BJ) Limiter [19]**).

$$\lambda_{\Omega_j} = \min_{x_q \in X_L} \begin{cases} \frac{\phi_j^{\max} - \phi_j}{R^0_1(\vec{x}_q; \phi_h)|_{\Omega_j} - \phi_j} & \text{if } R^0_1(\phi_q; \phi_h)|_{\Omega_j} > \phi_j^{\max} \\ \frac{\phi_j^{\min} - \phi_j}{R^0_1(\vec{x}_q; \phi_h)|_{\Omega_j} - \phi_j} & \text{if } R^0_1(\phi_q; \phi_h)|_{\Omega_j} < \phi_j^{\min} \end{cases} \tag{4.64}$$

with the local minima and maxima

$$\phi_j^{\min} = \min\left(\phi_j, \min_{F_{jk} \in \partial\Omega_j} \phi_k\right) \quad \phi_j^{\max} = \max\left(\phi_j, \max_{F_{jk} \in \partial\Omega_j} \phi_k\right)$$

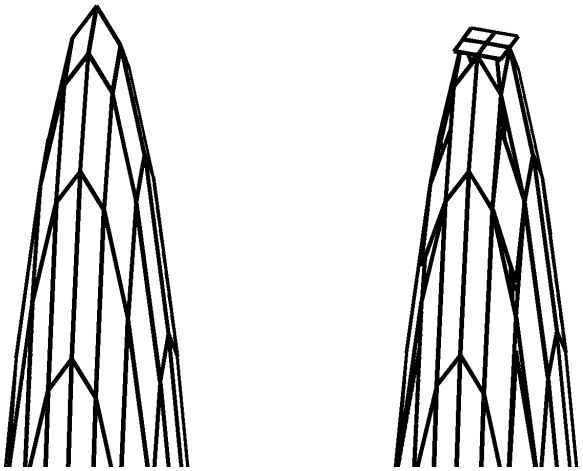
and $X_L \subset \partial\Omega_j$ *a set of limiting points*

For a positive scheme it is sufficient to choose X_L to contain the flux quadrature points. To obtain a positive reconstruction in the whole cell, the limiter is applied to the cell vertices $X_\mathrm{L} = \mathcal{V}_{\Omega,j}$.

The limiting function λ is necessarily non-linear. For explicit schemes, implementation of the resulting nonlinear flux is straightforward. Implicit schemes using limited reconstructions result in nonlinear equation systems, requiring iterative solution. Also, as is the case for the BJ limiter, the limiter function can be non-smooth, possibly causing problems with convergence in iterative schemes. To improve BJ-type limiters in this respect, smooth modifications have been proposed [82, 83, 116, 117].

4.4.3. Flux functions

The reconstruction polynomial is used to recover more accurate solution and derivative approximations at flux quadrature points. In the case of linear reconstruction, for a face e_{jk},



(a) Unlimited reconstruction (b) After limiting

Figure 4.8.: Effect of limiting at local extrema. The unlimited reconstruction a) exceeds the maximum cell averages at the cone tip. The limiter sets the gradient in the cells at the tip to zero to avoid this.

let $\vec{d_j} = \vec{x}_q - \vec{x}_j$ be the vector from the cell centroid x_k to the to the quadrature point x_q (see figure 4.9). The reconstructed values are then

$$\phi^- = \phi_j + \vec{d_j} \cdot (\nabla\phi)_j, \qquad (\nabla\phi)^- = (\nabla\phi)_j, \tag{4.65}$$

$$\phi^+ = \phi_k + \vec{d_k} \cdot (\nabla\phi)_k, \qquad (\nabla\phi)^+ = (\nabla\phi)_k. \tag{4.66}$$

For the advection term the upwind flux (cf. the UDS scheme in appendix B.1)

$$F^a_{jk} = \frac{\vec{u}\cdot\vec{n}}{2}(\phi^- + \phi^+) + \frac{|\vec{u}\cdot\vec{n}|}{2}(\phi^- - \phi^+) \; = \begin{cases} \vec{u}\cdot\vec{n}\phi^- & \text{for } \vec{u}\cdot\vec{n} \geq 0 \\ \vec{u}\cdot\vec{n}\phi^+ & \text{for } \vec{u}\cdot\vec{n} < 0 \end{cases} \tag{4.67}$$

is used with the reconstructed values instead of the cell averages. For the diffusion flux

$$F^d_{jk} = D_q(\nabla\phi)_q \cdot \vec{n}_q \tag{4.68}$$

an approximation of the gradient $(\nabla\phi)_q$ at the quadrature point is required. The following two schemes are considered:

- **Averaged gradient with jump correction**. The reconstructed gradient on both sides is averaged with weights $\alpha_\pm$. To avoid inconsistency in the presence of limiting, an additional term correcting for the discontinuity of the reconstructed solution at the face is introduced. Δl is the gradient length at the face, here $\Delta l = \|\vec{d_j}\| + \|\vec{d_k}\|$ is used. The complete gradient approximation is then

$$(\nabla\phi)_q = \left(\alpha_-(\nabla\phi)^- + \alpha_+(\nabla\phi)^+\right) + \frac{\phi^+ - \phi^-}{\Delta l}. \tag{4.69}$$

The weights are set using a parameter $\lambda \in [0,1], \alpha_- = \lambda, \alpha_+ = (1-\lambda)$. The simplest choice is arithmetic averaging, i.e. $\lambda = 0.5$. Setting $\lambda = \|\vec{d}_j\|/(\|\vec{d}_j\| + \|\vec{d}_k\|)$ (linear interpolation) recovers the central difference scheme on structured grids with nonuniform grid spacing. A similar scheme (without jump correction) is used in [88]. The jump correction is similar to the diffusive fluxes with penalty terms used in the *Discontinuous Galerkin methods* [14, 41].

- **Orthogonal reconstruction with central differencing**. Following [53], the solution is reconstructed to points on a line $\overline{x_- x_+}$ orthogonal to the face and passing through the quadrature point (marked red in figure 4.9). The gradient is then obtained as a central difference

$$(\nabla\phi)_q = \frac{R(x_+) - R(x_-)}{\|\vec{x}_+ - \vec{x}_-\|}. \tag{4.70}$$

A comparison of the performance of the different diffusion flux choices is presented in section 5.1.3.

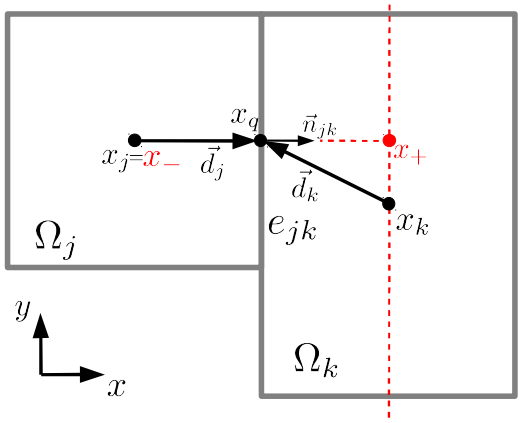


Figure 4.9.: Geometry for solution reconstruction at face flux quadrature points. Objects in red color are used for the orthogonal face gradient reconstruction (4.70).

4.4.4. Iterative solution procedure

The flux balance is assembled for every cell as done for the linear FVM in (4.48). For steady-state solutions and implicit time integration schemes for the HR scheme, instead of a linear system a nonlinear system

$$\mathcal{A}\vec{\phi}_h = \vec{S}_h. \tag{4.71}$$

is obtained. It has to be solved using an iterative algorithm. The Newton method can be used, however, the Jacobian has to derived manually for every flux/limiter combination or has to be approximated using finite differences. Furthermore, compared to the system for the linear scheme (4.51), the Jacobian matrix contains a high number of nonzero entries[3]. Therefore a simpler *defect correction (DC)* approach is chosen.

[3]Linear reconstruction leads to the flux balance of a cell to depend on second-neighbor information, resulting in stencil sizes from 13 on structured rectangular grids up to 29 on unstructured grids (cf. appendix B.4)

Defect correction

Defect correction (DC) [107, 91, 53] is a method to improve the accuracy of a numerical scheme. Assume that for a given boundary value problem two discretizations

$$L_l \phi_l = S_l \quad \text{and} \quad L_h \phi_h = S_h$$

are given, with L_h offering higher accuracy than L_l and L_l having lower computational cost and/or better stability properties. The discretizations can use different numerical schemes or represent the same scheme on grids with different resolutions. The accuracy of the discretization L_l can then be improved using the iterative procedure

$$\begin{aligned} L_l \phi_h^{(0)} &= S_l \\ L_l \phi_h^{(n)} &= S_l - L_h \phi_h^{(n-1)} + L_l \phi_h^{n-1}. \end{aligned} \tag{4.72}$$

The discretization L_h is only computed for known intermediate solutions to apply a correction to the right hand side. If L_l has consistency order $\mathcal{O}(\Delta x)$ and L_h has $\mathcal{O}(\Delta x^2)$, on structured grids already after the first iteration ϕ_h^1 has second order accuracy [119]. The method also works for nonlinear discretizations and integrates naturally into multi-grid schemes [59].

In the solver used for the benchmarks in chapter 5 the Patankar scheme (4.50) is combined with the HR flux functions from section 4.4.3 (cf. algorithm 9). The algorithm is then similar to the simplified Newton method (or "Chord-Newton-Process") using the flux linearization resulting from the Patankar scheme as an approximate Jacobian.

4.4.5. Implementation of boundary conditions

Some care is required when implementing boundary conditions for the HR scheme. The solution values in ghost cells (cf. section 4.1.1) may or may not have influence on reconstruction and limiting depending on the boundary conditions. Introducing systematic errors at this stage can severely affect the accuracy, especially for in- and outflow conditions. Here, the approach is adopted that values of ghost cells (i.e. the boundary conditions) have no influence on interior cells in the solution reconstruction step. If required by the boundary condition, the reconstruction is modified in a separate step. In the limiting step the ghost cell values are included again[4], therefore at this stage they must be initialized properly. The full solution procedure for the HR scheme with boundary treatment is shown in algorithm 9. Another option is to include the boundary conditions treatment directly in the reconstruction step [25].

[4] This is due to a technical limitation. Limiting involves reconstructing to cell vertices, for which it cannot be decided easily whether they are on a boundary

Algorithm 9: Defect correction (with boundary conditions)

Assemble linear system L_l (including boundary conditions)
Compute low-order solution:

$$L_l \phi_h^{(0)} = S_l$$

for $i = 0, 1, \ldots$ **do**

begin Reconstruction and limiting

Compute reconstruction R_0^1 (algorithm 8)
Modify reconstruction to match boundary conditions
Set ghost cell according to boundary conditions
Apply limiter (4.64) to reconstruction

end

Compute correction:

$$C_{\Omega_k}^i = \sum_{e_{kl} \in \partial\Omega_k} F_{jk}^{\mathrm{l}}(\phi^{(i)}) - F_{jk}^{\mathrm{h}}(\phi^{(i)})$$

Solve for corrected RHS:

$$L_l \phi_h^{(i+1)} = S_h + C^i$$

4.5. Time discretizations

After performing the discretization in space, the semi-discrete scheme (4.19) can be considered as an initial value problem for a system of ordinary differential equations (ODEs)

$$\phi_h(0) = \phi_{h,0} \tag{4.73}$$
$$\phi_h'(t) = F(t, \phi_h(t)) \tag{4.74}$$

with $t \in (0, T]$ and initial value $\phi_0 \in \mathbb{R}^N$. The right hand side $F : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$ contains the spatial discretization and source terms. To obtain a fully discrete scheme, a suitable time discretization has to be chosen. A vast number of schemes exists, see e.g.[60, 61]. This section describes some schemes commonly used in fluid dynamics and outlines their properties. The emphasis is on methods of low order, matching the spatial discretization. More details are given in appendix B.2.

Linear one-step methods

One-step methods use only information from the last time-step ϕ_n to compute the next time-step ϕ_{n+1}. We consider Runge-Kutta methods of the form

$$\phi_{n+1} = \phi_n + \Delta t \sum_{i+1}^{s} b_i F(t_n + c_i \Delta t, \phi_{n,i}), \tag{4.75}$$

$$\phi_{n,i} = \phi_n + \Delta t \sum_{j=1}^{s} \alpha_{ij} F(t_n + c_j \Delta t, \phi_{n,j}), \quad i = 1, \ldots, s. \tag{4.76}$$

The methods evaluate F at intermediate time-points and compute a linear combination with coefficients α_{ij}, b_i and $c_i = \sum_{j=1}^{s} \alpha_{ij}$. Order conditions for the coefficients can be derived, resulting in an entire class of schemes. Explicit methods result if $\alpha_{ij} = 0$ for $j \geq i$. The simplest example is the explicit or forward first-order Euler method

$$\phi_{n+1} = \phi_n + \Delta t F(t_n, \phi_n). \tag{4.77}$$

An explicit method of second order is the one-step midpoint rule

$$\phi_{n+1} = \phi_n + \Delta t F(t_n + \frac{1}{2}\Delta t, t_n + \frac{1}{2}\Delta t F(t_n, \phi_n)). \tag{4.78}$$

An important implicit scheme is the implicit or backward Euler method

$$\phi_{n+1} = \phi_n + \Delta t F(t_{n+1}, \phi_{n+1}). \tag{4.79}$$

Linear multi-step methods

Multi-step methods not only use data from the current time-step ϕ_n, but also from previous time-steps. A linear k-step methods has the general form

$$\sum_{j=0}^{k} \alpha_j \phi_{n+j} = \Delta t \sum_{j=0}^{k} \beta_j F(t_{n+j}, \phi_{n+j}), \quad n = 0, 1, \ldots. \tag{4.80}$$

The advantage over single step methods is that only one evaluation of F is required per time level. If $\beta_k = 0$, the method is explicit, otherwise it is implicit. A class of schemes often used in computational fluid dynamics are BDF (*backward differentiation*) schemes. The two-level BDF-1 scheme is the same as the implicit Euler method (4.79). The general three-level BDF method is

$$\frac{3}{2}\phi_{n+2} - 2\phi_{n+1} + \frac{1}{2}\phi_n = \theta \Delta t F_{n+2} 2(1-\theta)\Delta t F_{n+1} - (1-\theta)\Delta t F_n \tag{4.81}$$

Choosing $\theta = 1$ yields an implicit scheme (commonly called *BDF-2*), $\theta = 0$ the related explicit scheme (*extrapolated BDF-2*). An approach that mixes the explicit and implicit variant by locally varying θ is described in [64].

In general, multi-step methods require special attention during the first $k-1$ time-steps. For this startup phase an alternative scheme has to be used. To avoid reduction of the order of the multi-step scheme, the startup values have to be approximated with the same approximation order.

4.5.1. Discussion of time-stepping schemes

The efficiency of the time integration schemes depends critically on the step size allowed when taking stability and positivity considerations into account. For pure advection problems, explicit schemes commonly perform better as they are faster to compute than their implicit counterparts, which require solution of large linear systems. However, for diffusive problems the step size restriction due to the inverse square relation to the grid size in equation (B.16) is much more severe. Thus for diffusive and stiff problems implicit methods are preferred. A central problem to all schemes is positivity, which can impose a time step restriction also for implicit schemes.

The B2 code currently employs the implicit Euler scheme, which has optimal stability and positivity properties, but only first-order accuracy. The implicit BDF-2 method is an attractive way to improve on this. The time step restriction it introduces can be acceptable depending on timescales present in the physics.

5. Numerical benchmarks

This chapter covers three topics. The first is to verify the implementation of the high-resolution finite volume method presented in chapter 4 using a steady-state benchmark problem. The same problem is then used to study the behaviour of the scheme on unstructured grids. Finally, a time-dependent benchmark problem is solved to gain insight into combining the Patankar hybrid and HR spatial discretization with the timestepping schemes listed in section 4.5.

5.1. Steady-state solver verification

5.1.1. Verification procedure

The benchmark case considered here is derived using the method of manufactured solution. The exact solution considered is $\phi : \mathbb{R}^2 \to \mathbb{R}$,

$$\phi(x, y) = y \sin(2\pi x). \tag{5.1}$$

Plots of the solution on the unit square are shown in figure (5.1). It is substituted into the advection-diffusion equation (5.2) to obtain the source term (5.3). The benchmark problem is then:

Benchmark problem 1 (**General steady-state advection-diffusion problem**). *Solve*

$$\nabla \cdot (\vec{u}\phi - D\nabla\phi) = s. \tag{5.2}$$

on the unit square $\Omega = [0, 1] \times [0, 1]$ *with* $\vec{u} = \left(\begin{smallmatrix} u_1 \\ u_2 \end{smallmatrix}\right), D = \left(\begin{smallmatrix} d_{11} & 0 \\ 0 & d_{22} \end{smallmatrix}\right)$ *and the source term*

$$s(x, y) = 2u_1\pi y \cos(2\pi x) + u_2 \sin(2\pi x) + 4d_{11}\pi^2 y \sin(2\pi x). \tag{5.3}$$

Dirichlet, in- or outflow conditions are prescribed on $\partial\Omega$ *depending on* $\vec{u}$ *and* D.

From (5.1), (5.2) and (5.3) the exact expression for the cell averages ϕ_i and source integrals S_i in cell Ω_i and the total flux F_{jk} through face e_{jk} are derived (see appendix C.1). Three cases with the following transport coefficients and boundary conditions are considered.

- **Case 1: Isotropic diffusion**.

$$\vec{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{5.4}$$

  Dirichlet conditions set to ϕ on all boundaries.
- **Case 2: Diagonal advection**

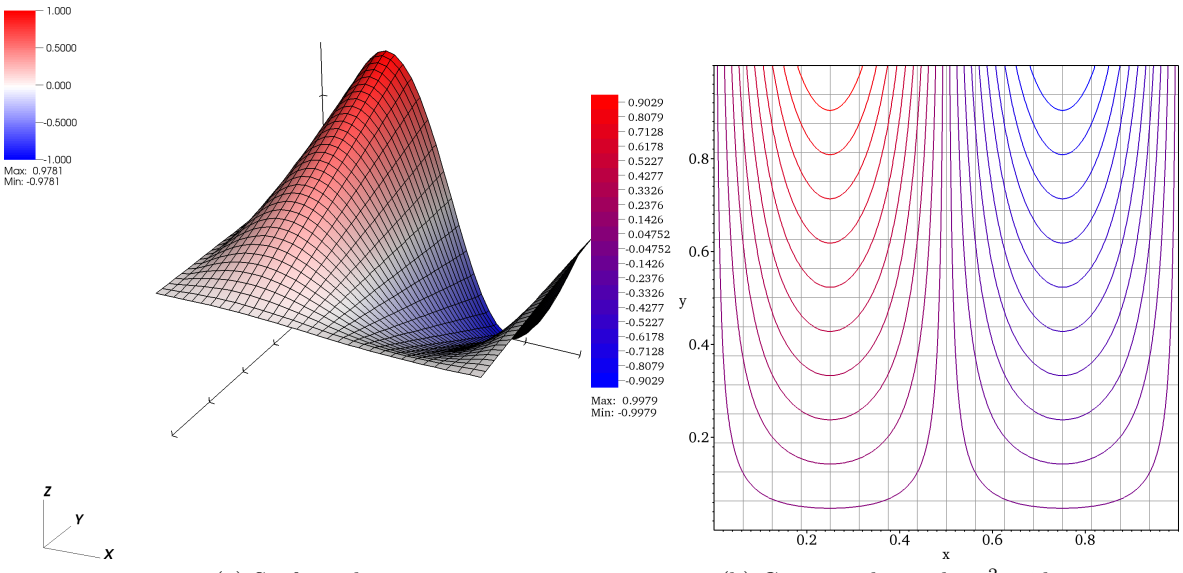
$$\vec{u} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{5.5}$$

  Inflow conditions set to ϕ on west and south, outflow condition on east and north boundaries.
- **Case 3: Anisotropic transport**

$$\vec{u} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad D = \begin{pmatrix} 0 & 0 \\ 0 & 10^{-8} \end{pmatrix} \tag{5.6}$$

  Inflow condition set to ϕ on west, outflow on east boundary. Dirichlet conditions on south and north boundaries.

Case 1 and 2 are used to separately study the performance for purely diffusive and purely advective transport. Case 3 reproduces the strongly anisotropic coordinate-aligned transport commonly encountered in the B2.5 code.



(a) Surface plot

(b) Contour plot with 16^2 grid

Figure 5.1.: Plots of the reference solution (5.1). ϕ is a sinus wave along the x and linear along the y coordinate.

The exact solution data is used in two steps to verify the solver. First, for a given grid $\mathcal{T}$ the exact cell averages ϕ_h^* are prescribed and the flux and operator approximations are computed. The truncation errors of the scheme for the operator τ_h^L (4.22) and flux τ_h^F (4.21) are then obtained by comparison with the exact values.

In the second step, the exact source term S_h^* and boundary conditions are prescribed and solved for. The global error ϵ_h can then be computed directly by (4.20).

The benchmark is performed for two spatial discretizations:

- The linear **Patankar hybrid scheme** as described in section 4.3, which reduces to first-order upwind for the pure advection case and second-order central differencing for the pure diffusion cases. The switch function $A(|P|) = |P|/(\exp(|P|) - 1)$ resulting in the exponential scheme is used.
- The **high-resolution (HR)** scheme using limited linear reconstruction as described in section 4.4. Limiting is performed using the scalar Barth-Jespersen (BJ) limiter (4.64). The nonlinear scheme is solved with defect correction (algorithm 9), using the Patankar hybrid scheme as low-order approximation.

A note on error measurement

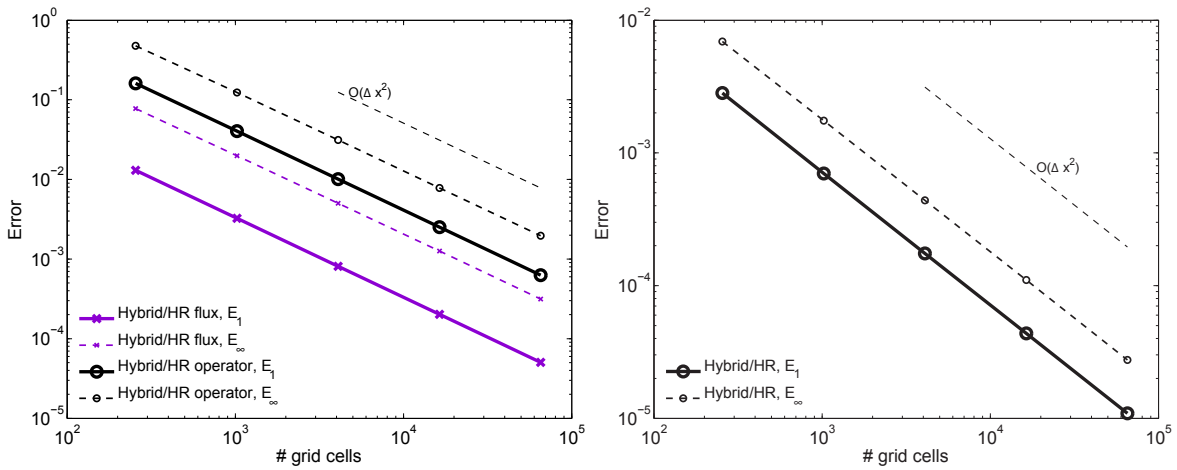
In this chapter, all errors are computed using the volume-weighted average formula (4.26), for which the notation E_p is introduced (with $p = 1$ or ∞ (maximum norm)). As weight the cell area $|\Omega_j|$ is used. The exception to this are the flux density truncation errors, where as weight the face area $|e_j|$ is used. Ghost cells are not included in the error measurements. Most error plots contain reference lines for $\mathcal{O}(h)$ and $\mathcal{O}(h^2)$ error reduction for easier comparison. In this context, h is an equivalent grid spacing for a two-dimensional structured grid with the given number of cells.

5.1.2. Structured grids

First, the convergence behaviour of the Hybrid and HR scheme is verified using a series of structured grids. Starting with a grid that covers the unit square with 16^2 square cells ($h = \Delta x = \Delta y = 1/16$, indicated in figure 5.1b), the resolution is increased by isotropic refinement, resulting in the grid series $16^2, 32^2, 64^2, 128^2, 256^2$. All grids are structured (every cell has one neighbour per side). On every resolution level the truncation and global errors are measured as described above.

Case 1: Isotropic diffusion

In the pure diffusion case, on uniform structured grids the hybrid scheme and all diffusion fluxes for the HR scheme described in section 4.4.3 reduce to central differences (CDS). The operator and flux errors are shown in figure 5.2a, as expected they show $\mathcal{O}(\Delta h^2)$ behaviour. The same holds for the global error shown in figure 5.2b. Both the hybrid and HR scheme give identical results. In contrast to the advection flux studied in the next section, the diffusion flux approximations are by design unaffected by the limiting, and no defect correction iterations are necessary for the HR scheme.



(a) Flux and operator truncation errors

(b) Global error

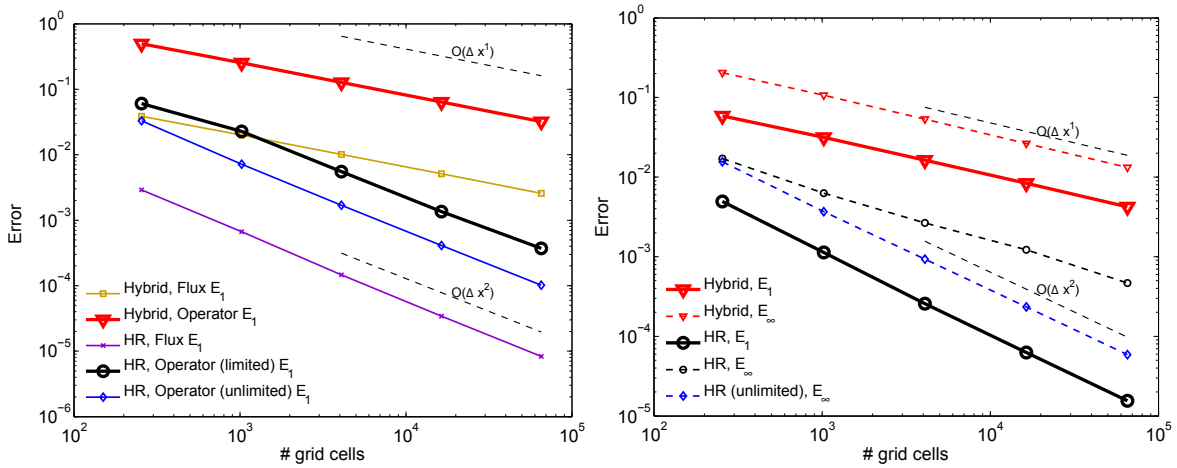
Figure 5.2.: Test case 1 (isotropic diffusion), structured refinement $16^2 \rightarrow 256^2$. Every point represents one grid resolution, i.e. one resolution level.

Case 2: Diagonal advection

In the pure advection case the hybrid scheme reduces to the first order upwind scheme. Truncation and global error drop as expected with $\mathcal{O}(\Delta h)$ (figures 5.3a,b).

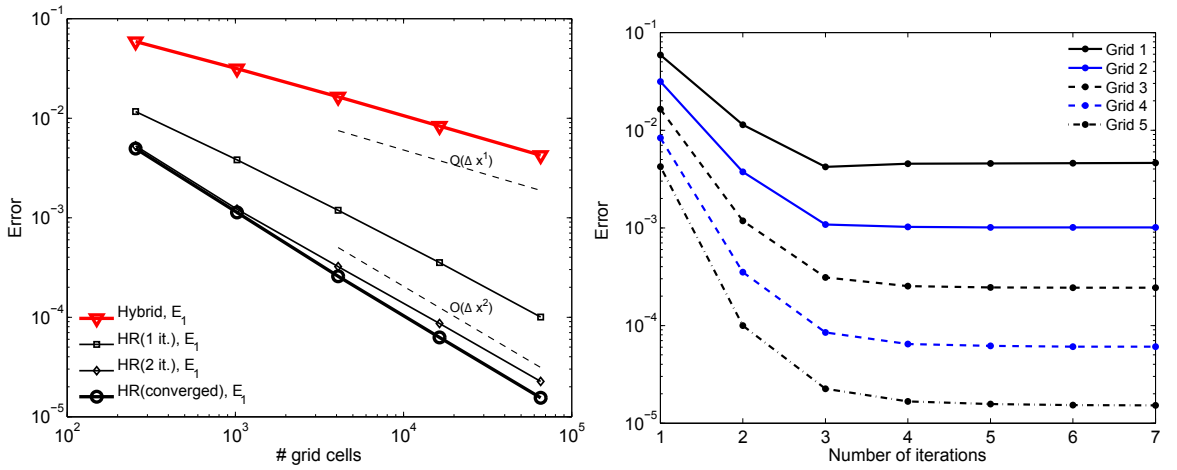
For the HR scheme, the E_1 measures of truncation and global error show the expected $\mathcal{O}(\Delta h^2)$ behaviour. The effect of limiting on accuracy is noticeable already in this example. Figures 5.3a and 5.3b include additional error data for the HR scheme without limiting. Limiting increases the operator truncation error. This is reflected in the maximum norm of the global error, which is degraded to $\mathcal{O}(h)$. In this example the limiter has the biggest effect on the north boundary due to the boundary condition implementation with ghost cells, which causes the flux truncation error locally to drop to roughly $\mathcal{O}(h)$. The global E_1 error of the HR scheme is practically unaffected by this. Both the limited and unlimited scheme produce nearly identical results (only the E_1 global error for the limited scheme is plotted in figure 5.3b).

Convergence of the defect correction iterations for the HR scheme is shown in figure 5.4a. The first iteration is equivalent to the solution of the hybrid scheme. Second order convergence of the global error is obtained already after the second iteration. Figure 5.4b shows the convergence history for the individual grids (grid 1 is 16^2, grid 5 is 256^2). The number of iterations needed increases with resolution, with a maximum of 4 iterations needed until convergence.



(a) Truncation errors (b) Global error

Figure 5.3.: Test case 2 (diagonal advection), structured refinement $16^2 \rightarrow 256^2$: truncation and global error for the Hybrid and HR scheme. The plots include data for both the limited and unlimited variant of the HR scheme to demonstrate the effect of limiting.

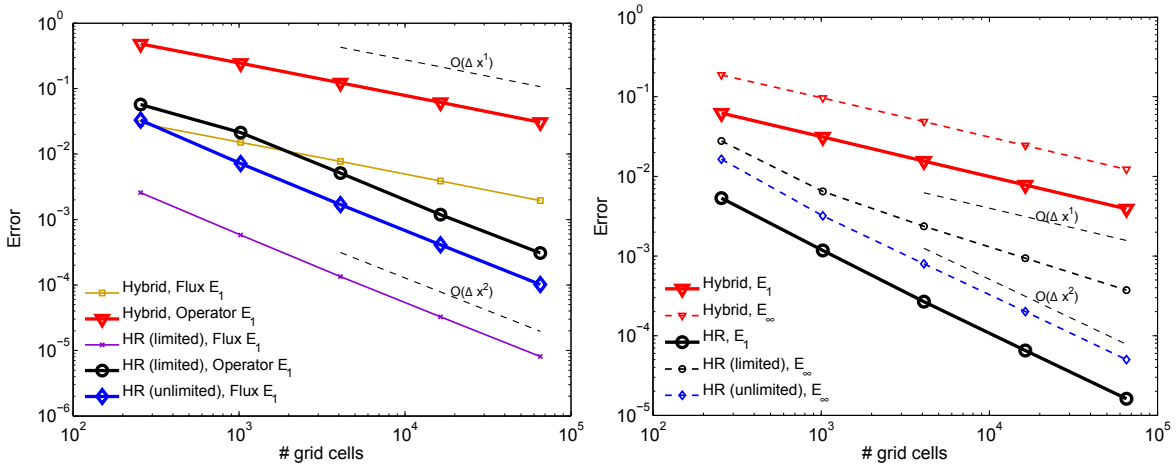


(a) Global E_1 error for the initial iterations and the converged state (b) Convergence behaviour for the individual grids. Grid 1=16^2, Grid 5 = 256^2

Figure 5.4.: Test case 2 (diagonal advection), structured refinement $16^2 \rightarrow 256^2$: convergence of defect correction iterations for the HR scheme.

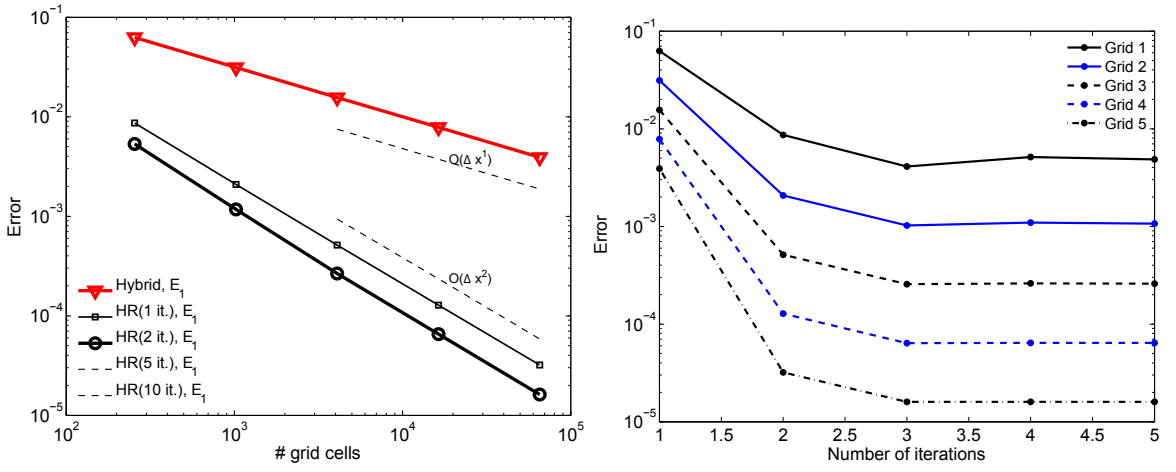
Case 3: Advection + Diffusion

In the anisotropic transport case the grid-aligned transport results in a situation of weakly coupled one-dimensional advection problems. The obtained error behaviour (figure 5.5) is nearly identical to the pure advection case. Convergence of the iteration scheme (figure 5.6) is faster, requiring up to 3 iterations. Second order accuracy is again obtained after 1 iteration.



(a) Truncation errors (b) Global error

Figure 5.5.: Test case 3 (anisotropic transport), structured refinement $16^2 \rightarrow 256^2$: truncation and global errors



(a) HR scheme iteration convergence (b) HR scheme iteration on individual grids

Figure 5.6.: Test case 3 (anisotropic transport), structured refinement $16^2 \rightarrow 256^2$: convergence of defect correction iterations

5.1.3. Unstructured grids

After establishing the behaviour of the schemes on structured grids, the same benchmark cases are solved on unstructured grids. Starting again from a 16^2 grid covering the unit square, the grid is refined four times by randomly selecting 25% of the cells for isotropic refinement. Then the grid is driven to the maximum resolution of 256^2 by refining all cells, for which a maximum of four refinement steps are required.

A detail view of a representative sequence of unstructured grids obtained by this process is shown in figure 5.7. Figures 5.7a-d show the grid state after the four random refinement steps. Figure 5.7e shows the first grid for which all cells were refined. The following grids 5.7e,f show that this quickly results in a grid with simple structured connectivity. The data plotted on these grids is explained in the discussion of case 2.

The error plots for all three cases below were generated for the same grid sequence (this "reference random grid sequence" shown in figure 5.7 is obtained by prescribing the random number generator seed). To make sure that the results are robust, all cases were run on further random grids series obtained by varying the refinement probability between 0.1 and 0.3. Overall, ten random grid series with up to 9 intermediate grids are created. The errors measured for these grid scans are shown in figures 5.10b, 5.13b and 5.15.

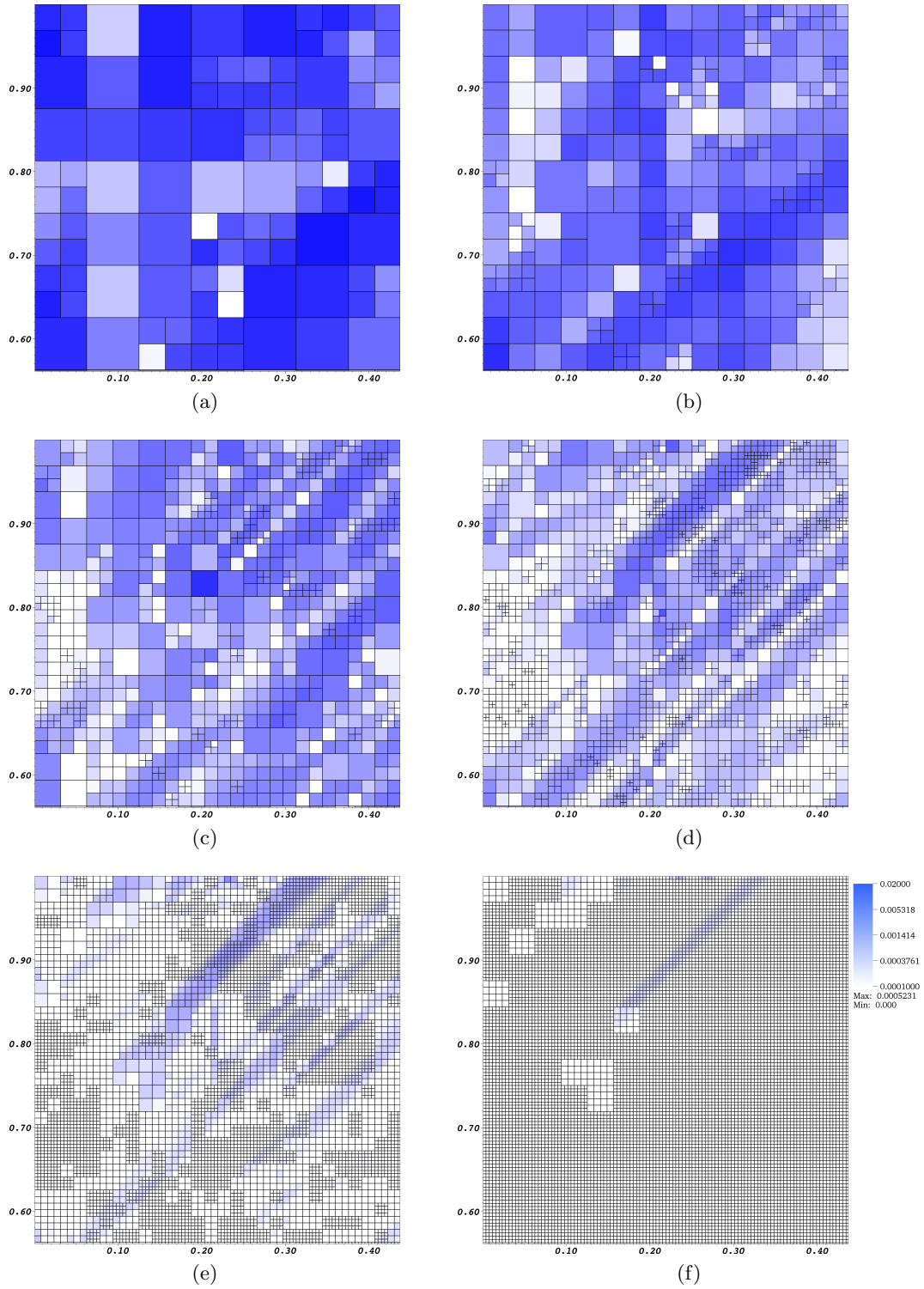


Figure 5.7.: "reference random grid sequence". Plotted data is the absolute error for benchmark case 2 (diagonal advection), showing clear patterns of global errors originating at refinement interfaces.

Case 1: Isotropic diffusion

On unstructured grids the gradient approximations at the faces given in section 4.4.3 differ in approximation accuracy. In a first step their performance is compared. The gradient average (4.69) is abbreviated as AJP for $\lambda = 0.5$ (arithmetic average) and AJC for $\lambda = \|\vec{d}_j\|/(\|\vec{d}_j\| + \|\vec{d}_k\|)$ (linear interpolation). The orthogonal reconstruction (4.70) is abbreviated as REC. Figure 5.8 shows the E_1 and E_∞ error of the converged HR scheme solution for the three diffusion fluxes computed on the reference grid series. While all three show similar behaviour and roughly second order convergence, the AJP scheme consistently produces the lowest error. It is therefore adopted as the reference diffusion flux in the following benchmark. The hybrid scheme reduces again to central differences and is affected by the problems described in section 4.3.1.

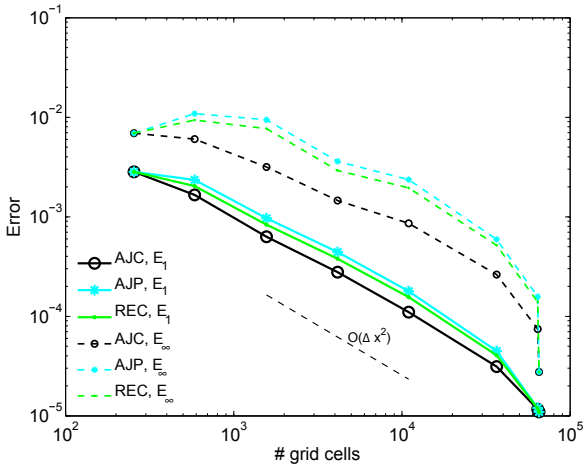
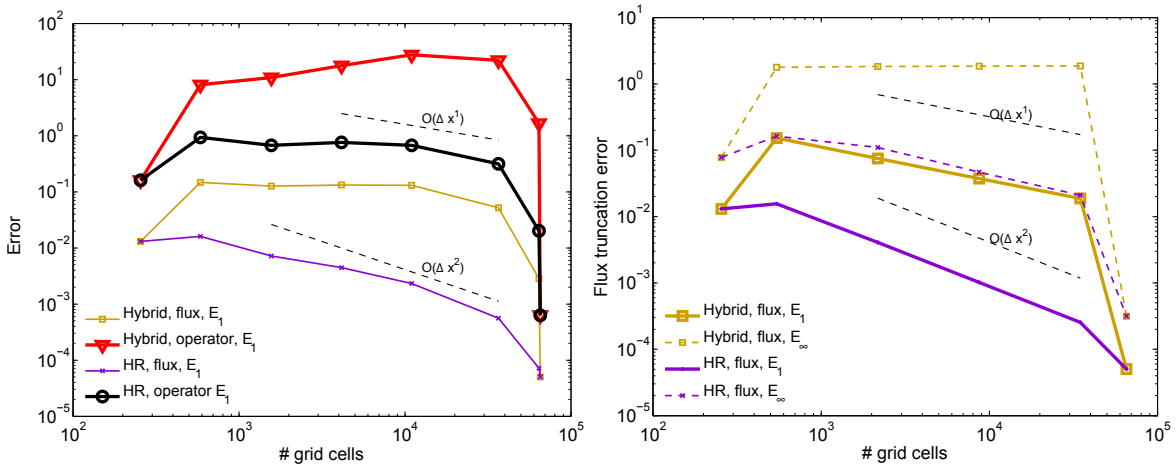


Figure 5.8.: Comparison of different diffusion flux approximations

Figure 5.9a shows the development of the local truncation errors. Especially the operator errors (which are even increasing with higher grid resolution) shown in this plot have to be interpreted carefully. They reflect the fact that when the local geometry of a cell changes from essentially structured (one neighbour per side) to unstructured (more than one neighbour per side), the truncation error increases abruptly. With increasing cell count also the relative number of such irregular cells increases. They occur at resolution jumps, thus when refining all cells in the grid their number grows linearly while the overall cell count grows quadratically.

To obtain a clearer picture on the truncation errors, plot 5.9b show the diffusion flux truncation error for a variation of the random refinement grid sequence. Instead of four only one random refinement step is performed, after which the resolution is again increased isotropically for all cells. The hybrid/central difference scheme clearly shows no dependence on grid spacing (i.e. $\mathcal{O}(1)$) for the flux approximation error and $\mathcal{O}(h)$ reduction of the E_1 flux error. For the HR scheme $\mathcal{O}(h)$ and $\mathcal{O}(h^2)$ is observed respectively.

However, despite the negative results for the truncation errors, the global error shows a better picture. Figure 5.10 shows the global error for both schemes, 5.10a on the reference grid series



(a) Truncation errors for the reference grid series

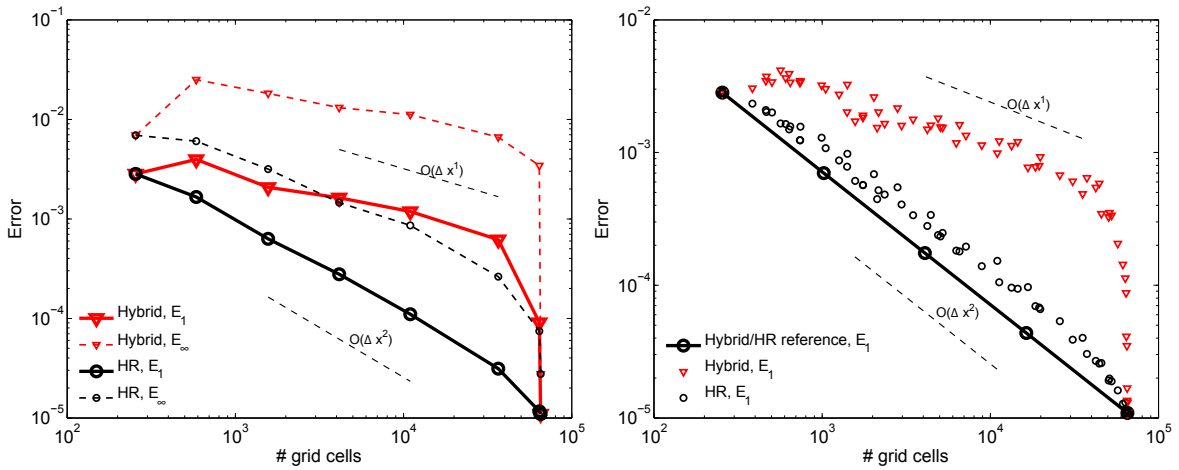
(b) Flux truncation errors for alternate grid series

Figure 5.9.: Test case 1 (isotropic diffusion), unstructured refinement $16^2 \rightarrow 256^2$: operator and flux truncation errors

and 5.10b on the grid scan obtained by varying the refinement probability. Furthermore, figure 5.10b shows the reference convergence behaviour obtained for both schemes on the structured refined grids presented before, which bounds the optimal convergence rate. The hybrid scheme (which in this case is equivalent to central differencing) even increases the error when deviating from a structured grid, which is to be expected due to the drop in consistency order observed for the local truncation errors.

Overall, on the unstructured grids the hybrid (central difference) scheme shows roughly $\mathcal{O}(h)$ error reduction. As the grid becomes increasingly structured again, the error is reduced very fast until at 256^2 it again coincides with the HR scheme.

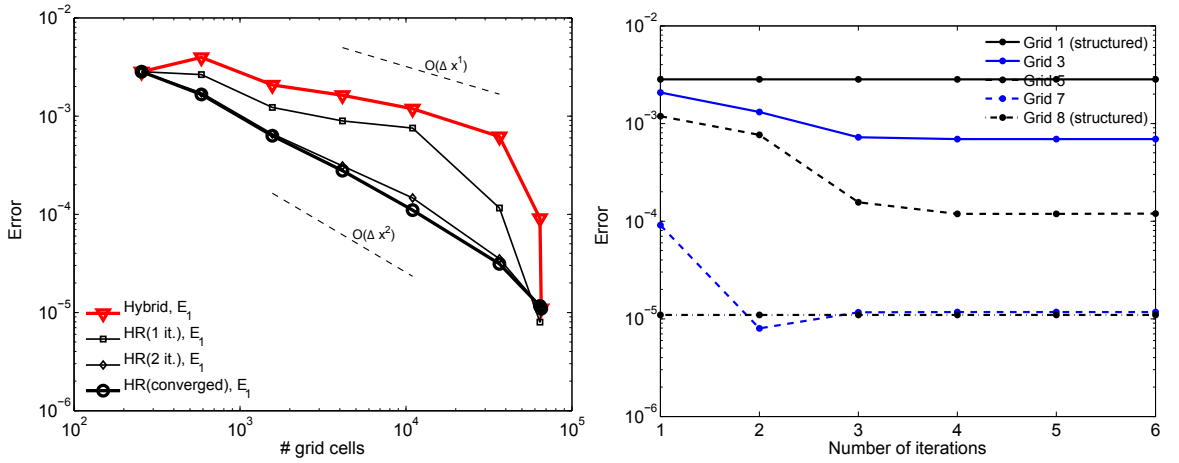
Behaviour of the HR scheme is significantly more robust, with error reduction close to $\mathcal{O}(h^2)$. Because the diffusive flux (4.69) is less sensitive to limiting due to the jump correction, the convergence behaviour of the DC iterations (see figure 5.11) is robust, with good convergence already after 2 iterations.



(a) Global error on reference grid series

(b) Global error for grid scan. The solid line gives the convergence obtained on structured grids.

Figure 5.10.: Test case 1 (isotropic diffusion), unstructured refinement $16^2 \rightarrow 256^2$: global errors



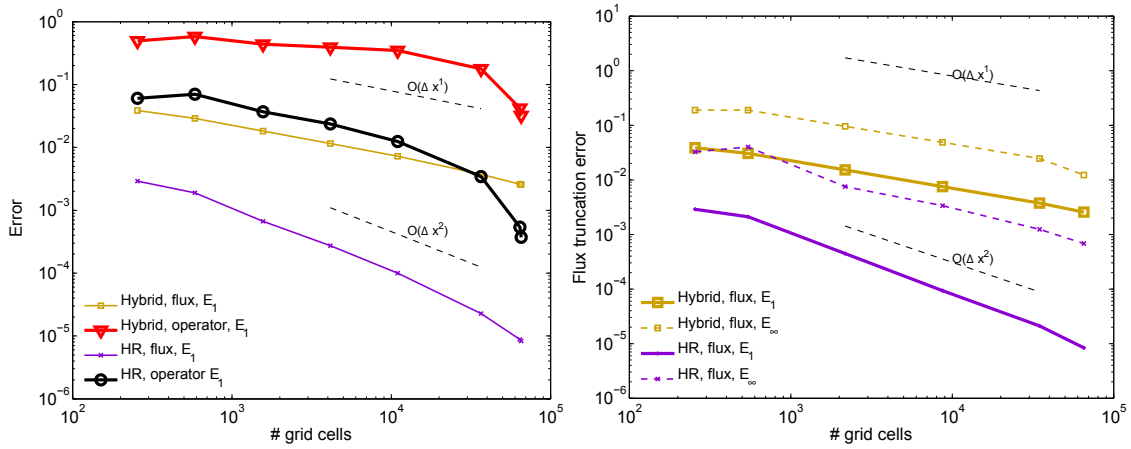
(a) HR scheme iteration convergence

(b) HR scheme convergence on individual grids

Figure 5.11.: Test case 1 (isotropic diffusion), unstructured refinement $16^2 \rightarrow 256^2$: convergence of DC iterations

Case 2: Diagonal advection

For the diagonal advection case, when studying the local truncation error (figure 5.12) the same limitations apply as mentioned in the pure diffusion case 1. However, as the flux truncation errors for the grid series with only one step of random refinement plotted in figure 5.12b show, the advection flux approximations behave much more robustly ($\mathcal{O}(h)$ in E_1 and E_∞ norm for the hybrid/upwind scheme, $\mathcal{O}(h^2)$ in E_1 and $\mathcal{O}(h)$ (due to the limiting) in the E_∞ norm for the HR scheme).



(a) Truncation errors

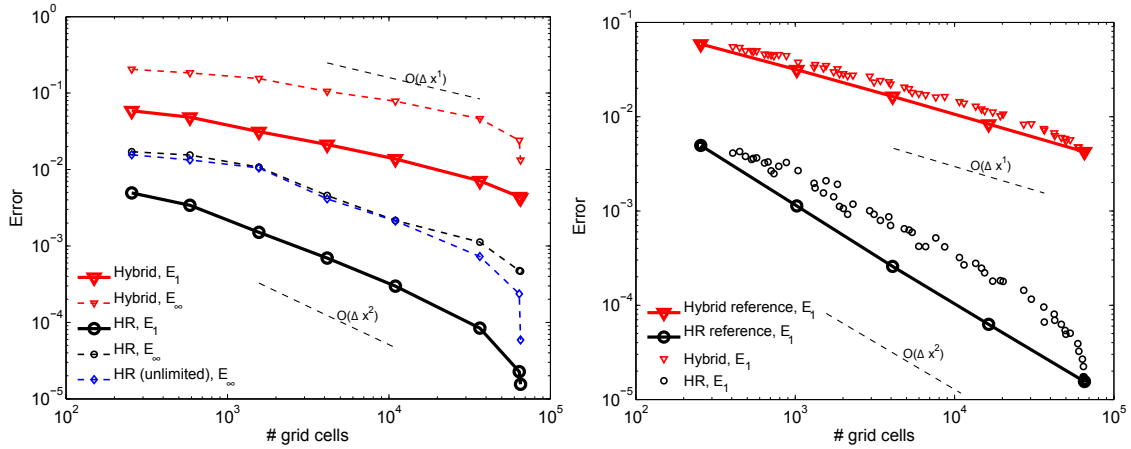
(b) Flux truncation errors for alternate grid series

Figure 5.12.: Test case 2 (diagonal advection), unstructured refinement $16^2 \rightarrow 256^2$: local truncation errors

Considering the global error shown in figure 5.13a, the first-order upwind scheme shows its robustness and does not deviate far from the reference behaviour obtained with structured refinement. The plot contains again the reference convergence behaviour previously obtained for structured refinement. The HR scheme again maintains close to $\mathcal{O}(h^2)$ convergence, with faster error reduction as the grid becomes more structured again. The effect of limiting is barely noticeable in the E_1 norm of the global error, but can be seen in the maximum norm show in figure 5.13a. With limiting the maximum error stays at a high level even on the structured 256^2 grid.

Where this case deviates from the observations made so far is in the convergence of the defect correction iterations. With increasing grid resolution convergence slows down significantly. It is slowest (in excess of 20 iterations necessary) for the second-to-last grid, which contains mostly cells with structured connectivity (one neighbour per side). This is not surprising, as the rate at which the iteration propagates the corrections throughout the grid depends on the eigenvalues of the iteration matrix, which in turn depend on the grid spacing. Convergence on the final 256^2 grid is then fast again, accounting for the fact that with the transition from unstructured to structured grid high local truncation errors caused by non-structured cells vanish.

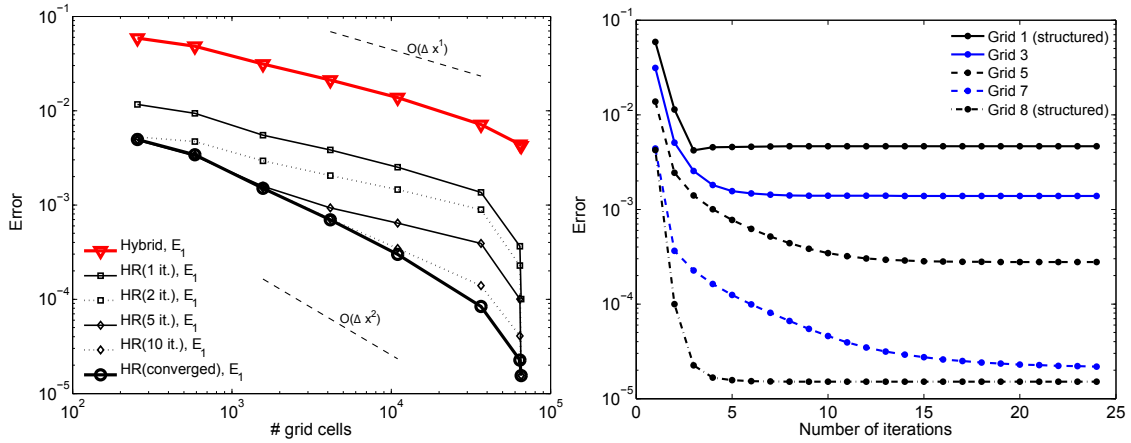
Figure 5.7 shows the global error for the converged solution of the HR scheme overlaid on the reference grid series. The propagation of high local truncation errors originating at resolution jumps along the diagonal convection direction is clearly visible. This demonstrates the error propagation behaviour predicted by (4.23).



(a) Global error on reference grid series

(b) Global error, grid scan. The solid lines represent the reference convergence on structured grids.

Figure 5.13.: Test case 2 (diagonal advection), unstructured refinement $16^2 \rightarrow 256^2$: global errors



(a) HR scheme iteration convergence

(b) HR scheme convergence on individual grids

Figure 5.14.: Test case 2 (diagonal advection), unstructured refinement $16^2 \rightarrow 256^2$: convergence of DC iterations. Convergence slows down significantly on unstructured grids with small cells.

Case 3: Advection + Diffusion

Performance of the schemes in the anisotropic transport case is again very similar to the pure advection case, therefore plots of the local truncation error are omitted. The main difference is that for this simpler (essentially one-dimensional for the advective flux) situation the DC iterations converge much faster than in the diagonal advection case, with good results after 6 iterations for all grids. The global errors obtained in the grid scan and the iteration performance on the reference grid series is shown in figure 5.15 and 5.16.

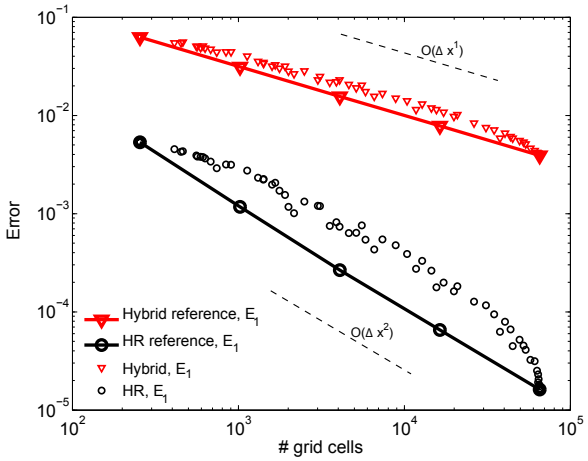
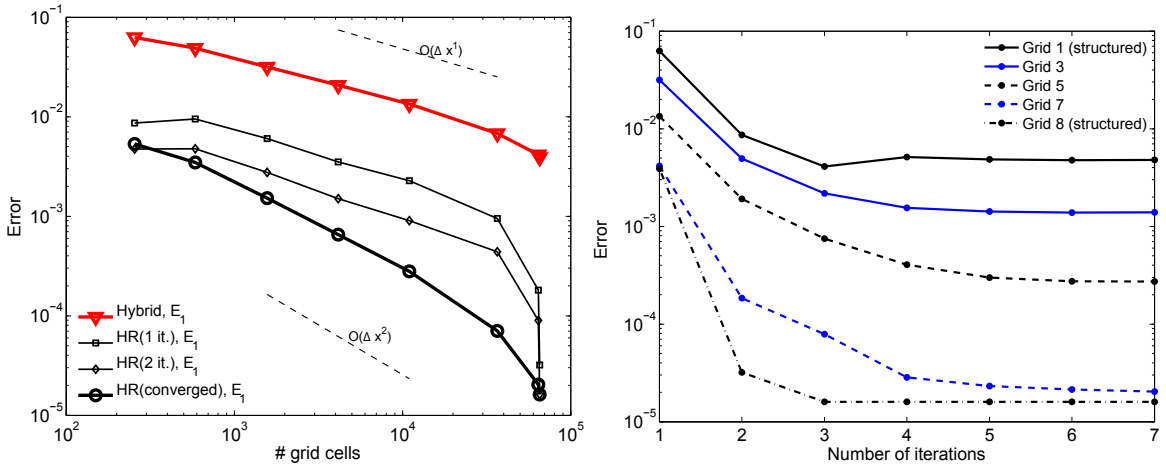


Figure 5.15.: Test case 3 (diagonal advection), unstructured refinement $16^2 \rightarrow 256^2$. Global error, grid scan



(a) HR scheme iteration convergence

(b) HR scheme convergence on individual grids

Figure 5.16.: Test case 3 (diagonal advection), unstructured refinement $16^2 \rightarrow 256^2$: convergence of DC iterations. Compared to case 2, convergence is fast on all grids.

5.1.4. Coordinate-mapped grids

An equivalent verification was performed in the presence of an orthogonal curvilinear mapping. The benchmark grid (cf. figure 5.18) is given by the conformal mapping [54]

$$x(\vec{\xi}) = \sqrt{\frac{\sqrt{\xi_1^2 + \xi_2^2} + \xi_1}{2}}, \quad y(\vec{\xi}) = \sqrt{\frac{\sqrt{\xi_1^2 + \xi_2^2} - \xi_1}{2}}. \tag{5.7}$$

The solution $\phi(\vec{\xi}) = \xi_1 + \xi_2$ is prescribed, which is linear in $\vec{\xi}$ and therefore nonlinear in $\vec{x}$. For this situation, analytic derivation of exact source terms and fluxes is significantly more complicated. The reference quantities as described in section 5.1.1 are therefore computed to a sufficient accuracy using adaptive numerical quadrature[1].

Figure 5.18 shows the reference solution and an unstructured curvilinear grid. Error reduction behaviour for benchmark case 1 (isotropic diffusion) is shown in figure 5.17 (cf. figure 5.11a for the non-mapped case). The results obtained on the mapped grids are very similar to those for non-mapped grids. The main differences are slight variations in the errors due to additional geometry approximation effects. Detailed analysis of the individual cases is therefore omitted.

[1] Routines from the NAG library [4] are used.

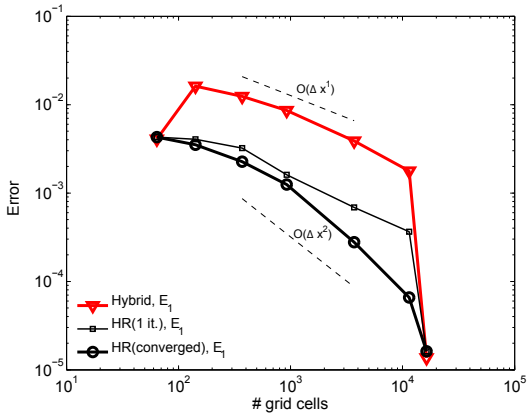


Figure 5.17.: Test case 1 (isotropic diffusion) on coordinate-mapped grid, unstructured refinement $8^2 \rightarrow 128^2$: global E_1 errors of DC iterations.

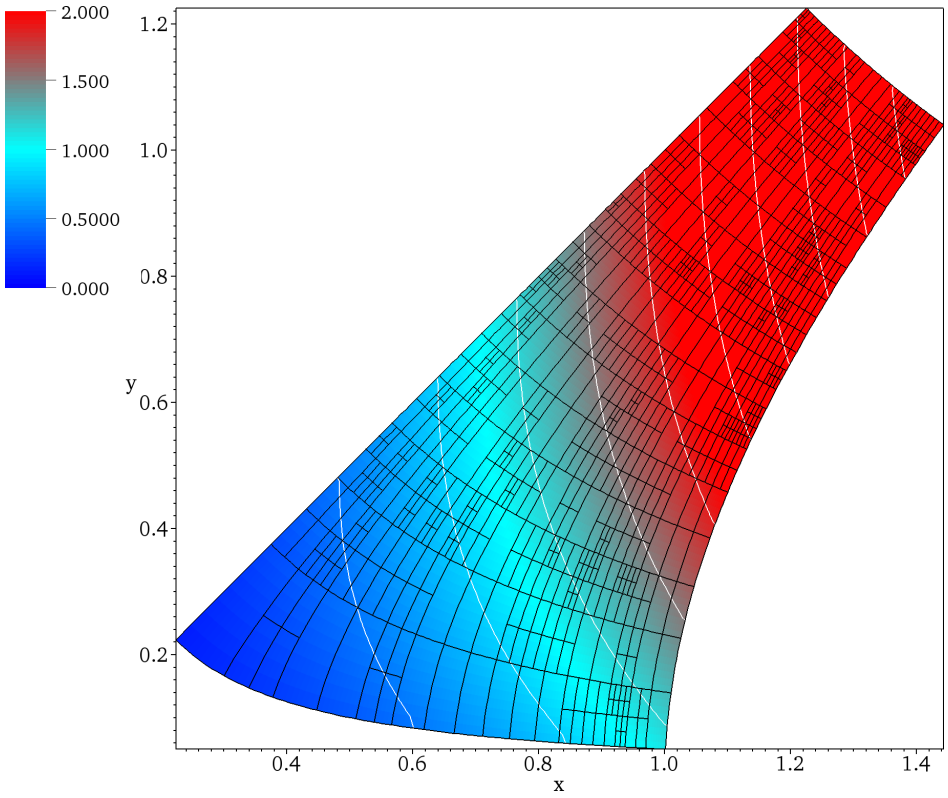


Figure 5.18.: Reference solution for verification on mapped grids. White lines are countours of ϕ.

5.2. Time-dependent benchmark

The purpose of this benchmark is to a) study combination of the hybrid and HR FVM with the time stepping schemes given in section 4.5 and b) apply the adaptation algorithms to a dynamic situation. The problem is taken from [104] and was originally stated in [48].

Benchmark problem 2 (**Unsteady circular advection**). *Solve the advection equation*

$$\frac{\partial \phi}{\partial t} + \vec{u} \cdot \nabla \phi = 0 \tag{5.8}$$

on the unit square $\Omega = [0,1] \times [0,1]$ *and* $t \in (0, \pi]$ *for the circular velocity field*

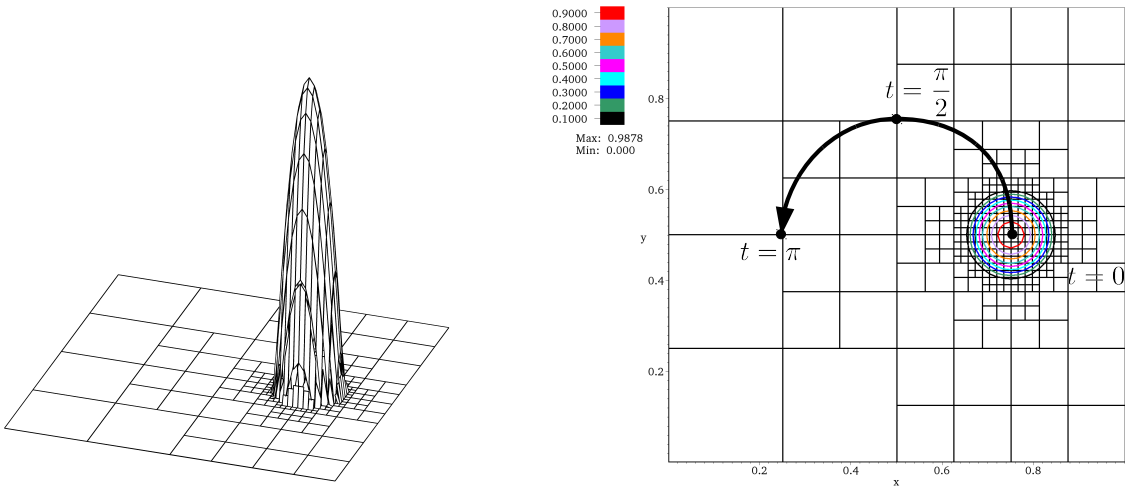
$$\vec{u}(x,y) = \begin{pmatrix} \frac{1}{2} - x \\ y - \frac{1}{2} \end{pmatrix} \tag{5.9}$$

and starting value

$$\phi(\vec{x}, 0) = \begin{cases} -\frac{1}{0.01}((x-\frac{3}{4})^2 + (y-\frac{1}{2})^2) + 1 & \text{for } \frac{1}{0.01}((x-\frac{3}{4})^2 + (y-\frac{1}{2})^2) + 1 < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.10}$$

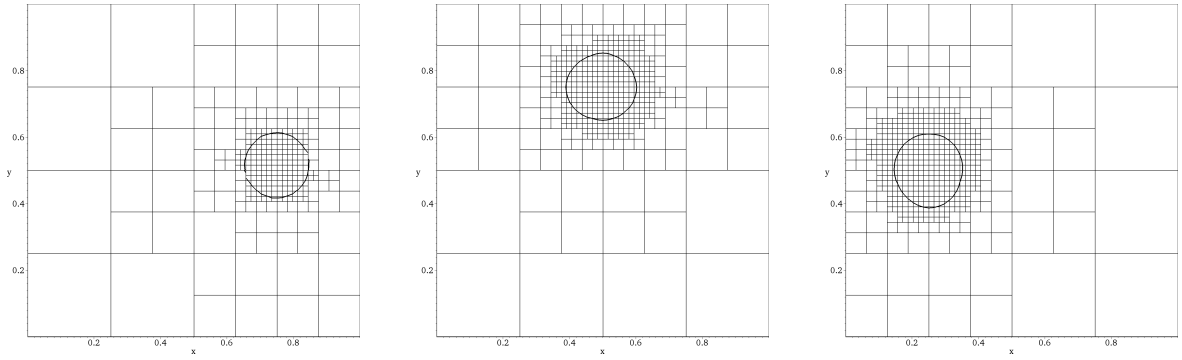
Inflow with $\phi = 0$ *and outflow conditions are imposed according to the velocity field.*

The starting value is a cone centered at $(0.75, 0.5)$ with diameter 0.1 and height 1 (see figure 5.19a). It is rotated by 180° (figure 5.19b). The problem is solved on an adaptive grid with a maximum resolution of 128^2 cells, i.e. $\Delta x = \Delta y = 1/128$. The numerical solution is compared to the exact solution at $t = \pi$ using the same error measures as in section 5.1.



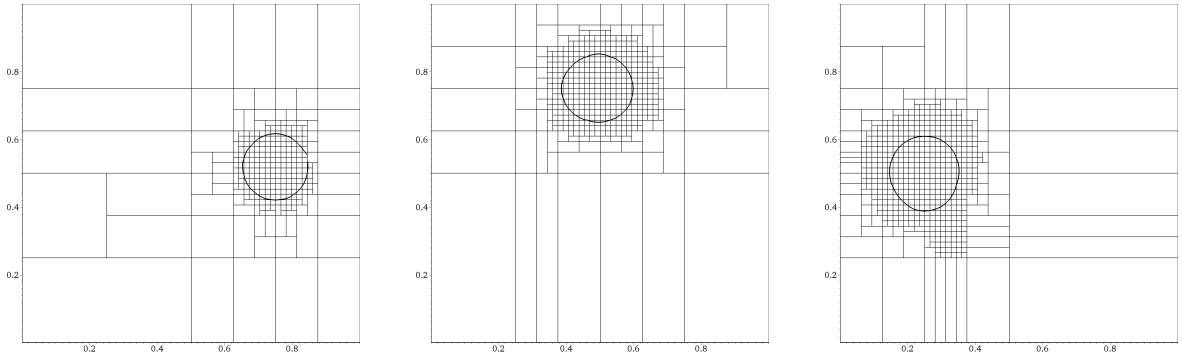
(a) (b)

Figure 5.19.: Benchmark problem 2: initial value with adapted grid (isotropic strategy)



(a) Shortly after start (b) $t = \pi/2$ (c) $t = \pi$

Figure 5.20.: Intermediate grids for the time-dependent benchmark case (isotropic adaptation strategy). The contour line marks $\phi = 0.1$.



(a) Shortly after start (b) $t = \pi/2$ (c) $t = \pi$

Figure 5.21.: Intermediate grids for the time-dependent benchmark case (anisotropic adaptation strategy). The larger cells cause faster smearing at the fringe of the cone compared the isotropic strategy (figure 5.20). This leads to a larger area covered by fine cells at $t = \pi$.

5.2.1. Grid adaptation

Grid adaptation is performed after every timestep using a simple feature detection criterion. Cells with $\phi > 0.001$ are marked for refinement, everywhere else all faces are marked for removal. This forces the grid adaptation algorithm to keep the resolution as high as possible at the position of the cone, while keeping the resolution as coarse as possible everywhere else. The grid at $t = 0$ is indicated in figures 5.19a and 5.19b.

Intermediate grids from the simulations are shown in figure 5.20 (isotropic strategy) and figure 5.21 (anisotropic strategy). The high-resolution region follows the cone and the adaptation algorithm efficiently removes the fine cells after it passed. As the cone spreads due to numerical diffusion in the spatial discretization, the refinement region grows. This is more pronounced for the anisotropic case due to the larger cells present in the grid.

5.2.2. Comparison of spatial and temporal discretizations

A note on timesteps

The behaviour of the time discretization error was studied by scanning a range of timesteps. For all schemes the expected behaviour was observed when exceeding the stability or positivity boundaries (i.e. solution blow-up when exceeding the Courant limit and negative values when exceeding the positivity limit). For the second-order temporal discretizations the timestep has a negligible influence, the error is dominated by the spatial discretization. For the first-order schemes, the influence of the timestep is slightly more pronounced but of low overall importance. All simulation results below were obtained (if not otherwise stated) for a timestep of $\Delta t = 1/128$, which is below the Courant and positivity limit of all involved schemes.

First-order upwind spatial discretizaton

As detailed in appendix B.1, the first-order upwind difference scheme introduces strong numerical diffusion. This is clearly visible in figure 5.22. Table 5.1 summarizes the minimum and maximum values observed for $\phi(t = \pi)$ and the E_1 error w.r.t. the exact solution. All schemes maintain positivity exactly and show nearly idential maximum cone heights and errors. The slightly better performance of the explicit Euler is explained by cancellation of temporal and spatial error (this is known as the *Courant-Isaacson-Rees*-scheme)[65].

Scheme	Maximum	Minimum	E_1 Error
Explicit Euler	0.28608	0.0	0.018646
Implicit Euler	0.26040	0.0	0.019611
Midpoint RK	0.27212	0.0	0.019158
Explicit BDF-2	0.27207	0.0	0.019159
Implicit BDF-2	0.27222	0.0	0.019156

Table 5.1.: Hybrid spatial discrization (first-order upwind) combined with various time discretizations. Listed are the maximum solution value (cone height), minimum value and E_1 error at $t = \pi$.

High-resolution spatial discretizaton

In combination with the HR spatial discretization, all second-order time discretizations produce very similar results. Both cone height (table 5.2) and shape (figure 5.23b) are maintained significantly better than for the hybrid scheme. Behaviour of the first-order time discretizations differs. In the case of the implicit Euler scheme, the lower first-order accuracy is obvious (5.23a).

The explicit Euler method exhibits an effect known as "compression", i.e. a steepening of the

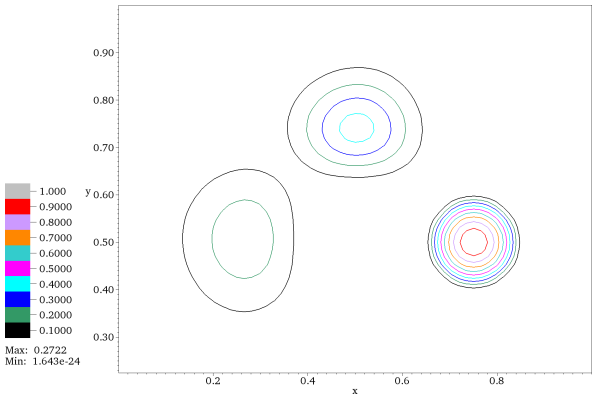


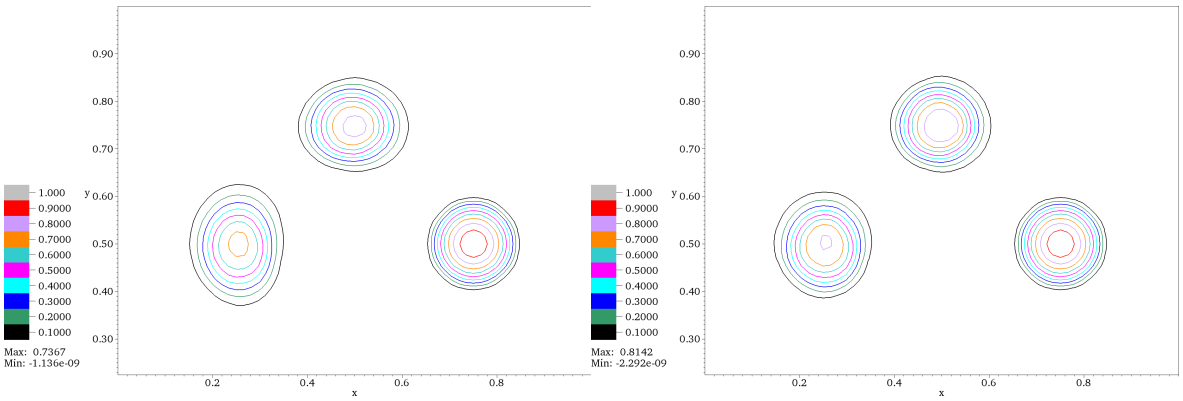
Figure 5.22.: Hybrid spatial discretization (first-order upwind, UDS) + implicit BDF2. The other time discretizations show equivalent behaviour. The cone deformation is caused by the strong numerical diffusion of the UDS scheme.

cone which is clearly visible in figure 5.24. It originates from interaction between the time discretization and the nonlinear limiter and can distort the solution as shown in figure 5.24a. This combination is therefore not recommended.

The most important observation here is the positivity violation of the implicit schemes. The negative values indicated in table 5.2 occur because the defect correction algorithm applied in each timestep did not solve the nonlinear system (4.71) exactly. Tests show that by increasing the number of defect correction iterations (algorithm 9) the negative values can be brought in the range of machine accuracy, but cannot be fully avoided. This has to be kept in mind when designing algorithms that rely on exact positivity.

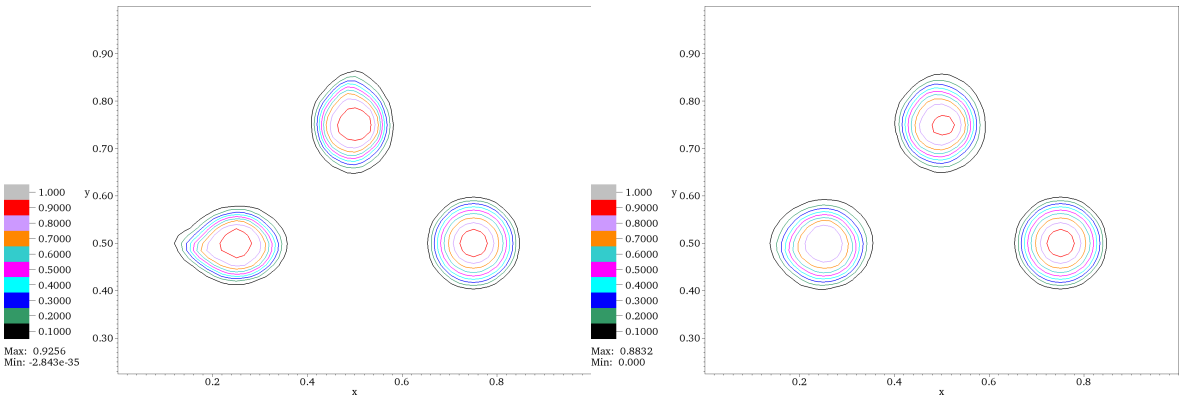
Scheme	Maximum	Minimum	E_1 Error
Explicit Euler	0.88318	0.0	0.0027219
Implicit Euler	0.73674	-1.1362e-09	0.0067140
Midpoint RK	0.81369	0.0	0.0045211
Explicit BDF-2	0.81327	0.0	0.0045754
Implicit BDF-2	0.81420	-2.2921e-09	0.0044634

Table 5.2.: High-resolution discretization combined with the listed time-stepping schemes (cf. table 5.1).



(a) Implicit Euler ($\mathcal{O}(\Delta t)$ accuracy) (b) Implicit BDF-2 ($\mathcal{O}(\Delta t^2)$ accuracy)

Figure 5.23.: HR spatial discretization + implicit timestepping methods.



(a) $\Delta t = 1/64$ (b) $\Delta t = 1/128$

Figure 5.24.: HR spatial discretization + explicit Euler timstepping. Steepening of the cone occurs due to interaction of the nonlinear slope limiter with the time discretization and possibly leads to unwanted deformations.

6. The B2.6 code

The numerical methods, algorithms and data structures discussed so far were designed and tested to form the basis of an adaptive version of the current B2.5 code, subsequently called B2.6. This chapter first starts with an overview of the B2.5 algorithm and then details the steps necessary to integrate the new components. The final B2.6 code then consists of two new solvers representing different stages of the code conversion process: B2.6-structured and B2.6-unstructured.

6.1. Overview of the B2.5 code

B2.5 is a compressible multi-fluid code. Its spatial discretization is based on the cell-centered linear finite volume scheme described in section 4.3 and field-aligned block-structured grids as described in section 3.1.2. All plasma state variables are stored colocated[1], the resulting risk of checkerboard pressure oscillations is controlled using Rhie-Chow velocity interpolation [97, 53]. The discretization is fully implicit, as time discretization the forward Euler scheme (4.79) is used.

6.1.1. Iterative solution algorithm

The nonlinear coupled system of fluid equations given in section 2.2 is solved by iterative sequential relaxation of linearized equations. One time step consists of three nested iteration loops separating atomic rate coefficient, source term and transport coefficient updates. An outline of the core solver structure is given in algorithm 10.

6.1.2. Inner iteration

In this context the term "inner iteration" does not refer to one iteration of an iterative linear system solver, but to a complete set of sequential relaxations of all equations as outlined in algorithm 11. The individual equations are solved in turn in linearized form. Couplings between equations are implemented by simultaneous consistent updates to the plasma state.

[1] I.e. velocities are stored on cells and not on a staggered grid centered on faces.

Algorithm 10: B2.5 algorithm: time step and iteration loops

Solver initialization
while $t \leq T$ **do**
  Outer iteration
    Compute atomic rate coefficients
    Intermediate iteration
      Compute source terms
      (*EIRENE coupling*)
      Inner iteration
        → algorithm 11
  $t = t + \Delta t$
Solver finalization

Algorithm 11: B2.5 inner iteration

Compute transport coefficients
Momentum block
  foreach species a **do**
    Solve momentum equation (2.5) → $\Delta u_{\parallel a}$
  Solve total momentum equation → $\Delta u_{\parallel t}$
  Relax $u_{\parallel a}$ for all species
Continuity block
  foreach species a **do**
    Solve pressure correction equation → Δp_a
    Relax $u_{\parallel a}, n_a$
Potential block
  Update potential following n_e change
  while $R_\phi > \epsilon_p$ **do**
    Solve potential equation (2.7) → $\Delta\phi$
    Relax ϕ
Energy block
  Solve ion energy equation (2.12) → ΔT_i
  Solve electron energy equation (2.9) → ΔT_e
  Solve total energy equation → ΔT_t
  Relax T_e, T_i, ϕ and $u_{\parallel a}$ for all species

General solution approach

For the solution of individual equations all coefficients and plasma quantities except the quantity under consideration $\varphi(\vec{x}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ are kept fixed and the corresponding

equation is solved for a correction $\Delta\varphi$. For this the advection-diffusion form

$$\frac{\partial}{\partial t}(\rho\varphi) + \nabla \cdot \underbrace{(\rho\varphi\vec{u} - D\ \nabla\varphi)}_{\vec{f}(\varphi)} = \tilde{S}(\varphi) \tag{6.1}$$

for $\varphi(\vec{x}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ (explicitly including the density $\rho(\vec{x}, t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$) is rewritten as

$$D(\varphi) := \nabla \cdot \vec{f}(\varphi) - \underbrace{\left(\tilde{S}(\varphi) - \frac{\partial}{\partial t}(\rho\varphi)\right)}_{:=S(\varphi)}. \tag{6.2}$$

The flux density $\vec{f}$ contains all terms that can be written as a flux divergence, all other terms are included in the source term S. For the solution φ then holds $D(\varphi) = 0$. The current approximation φ^* yields the residual $D(\varphi^*) = R^*$. A correction $\Delta\varphi$ is wanted so that $D(\varphi^* + \Delta\varphi) = 0$. For the source $S(\varphi)$ a linearization of the form

$$S(\varphi) \doteq S_0 + S_1\varphi + S_2\rho + S_3\rho\varphi \tag{6.3}$$

with constant coefficients $S_{0,1,2,3}$ is used. Linearization of (6.2) around φ^* then yields the correction equation

$$\nabla \cdot \vec{f}(\Delta\varphi) - (S_1 + S_3\rho)\Delta\varphi = -R^*. \tag{6.4}$$

It is solved using the hybrid linear finite volume scheme given in section 4.3. The solution is then updated to

$$\varphi = \varphi^* + \alpha_\varphi \Delta\varphi. \tag{6.5}$$

with an under-relaxation factor $0 < \alpha_\varphi < 1$.

Time discretization

The implicit Euler time discretization (4.79)

$$\frac{\partial}{\partial t}(\rho\varphi) \approx \frac{\rho\varphi^{n+1} - \rho\varphi^n}{\Delta t}, \tag{6.6}$$

is included in the source term S by modifying the source coefficients (6.3) to

$$S_0' = S_0 + \frac{\rho\varphi^{n+1}}{\Delta t}, \qquad S_3' = S_3 - \frac{1}{\Delta t}. \tag{6.7}$$

The time-dependent term is also included in steady-state simulations. The time step then acts as an additional relaxation factor.

Coupling terms

The sequential decoupled solution approach requires (in contrast to Newton-like simultaneous solution of the entire system) explicit treatment of the coupling terms. Coupling due to atomic processes is included in the source linearization. To improve convergence and enforce exact cancellation of the friction terms in the momentum equations an additional total momentum equation is solved. The same approach is used for the energy exchange terms of the energy equations, requiring the solution of a total energy equation.

Coupling of the velocity to the pressure gradient is taken into account by recasting the continuity equations into pressure-correction equations. The pressure corrections are used to update the densities via the equation of state for the partial pressure $p_a = n_a(T_i + Z_a T_e)$. Velocity corrections are constructed by exploiting the linear discretization of the momentum equation following the SIMPLE algorithm [90, 53]. A similar approach is used to update the velocities following the temperature correction. Additional potential updates are applied following changes in electron density and temperature. More details on the coupling algorithm are given in appendix A.2.

6.2. Development of the B2.6 code

6.2.1. Implementation challenges

The proposed changes (replacement of the data structure and the spatial discretization) have two common properties: they are pervasive and at the same time invisible from the outside. Their implementation touches virtually all parts of the code, but the new code B2.6 is expected to be able to exactly reproduce the results of B2.5. The only immediate change visible to the user are additional options for the numerical solver that control the grid adaptation algorithm.

Besides the new data structure and numerical methods required for the adaptive solver, from a software development point of view the following challenges present themselves.

Code size. The entire B2.5 code[2] currently amounts to roughly 150000 lines of Fortran code[3]. The core solver which includes all relevant components of the numerical algorithm (excluding initialization/finalization) is estimated at 35000 lines. In addition to this, the SOLPS package contains an extensive amount of pre- and post-processing infrastructure[4] which is often tied tightly to the internal structure of B2.5.

Physics model content. A central feature of the B2.5 model is the comprehensive physics model it represents, which was verified and validated extensively both in code-code and code-experiment studies [45, 40, 29, 39]. This has to be maintained by a new code version to be attractive to users, and is a strong reason why an evolutionary approach appears to be best despite the additional technical challenges it poses.

Regression testing. Structural changes to the code that should not alter its behavior have to be checked against the original code using regression tests to avoid errors and diverging code behaviour. Failing to do so is a recipe for disaster and effectively requires a complete re-verification of the new code.

User interface. Existing data and input definitions for simulations performed with the B2.5 code are expected to work with the new code without changes. The alternative would be to provide a conversion tool for input files, which is a task of considerable complexity and relatively error-prone.

To keep the transition manageable, the code modification was split into two blocks: data structure replacement and spatial discretization implementation. The resulting code B2.6 offers three choices ("code paths") for the core numerical solver: the original *B2.5-standard* solver, an adaptive solver restricted to structured grids (*B2.6-structured*) and a solver for

[2] Including I/O routines, some code-related pre- and post-processing programs and the b2plot visualization tool, but no external codes like CARRE or EIRENE.

[3] Mostly Fortran 77-style fixed source form with use of some Fortran 90 features.

[4] This includes a wide range of tools, ranging from shell scripts to an MDSPlus database for simulation results.

fully unstructured grids (*B2.6-unstructured*).

6.2.2. The B2.6-structured code

The central change in B2.6-structured is the replacement of the old B2.5 data structure with the UG data structure (cf. section 3.2.2). The following section describes the main aspects of this conversion.

Required new infrastructure components

Grid initialization. To read the base grid the existing B2.5 input routines and data structure are re-used. The working grid is then assembled in the UG data structure from this data. If the code is to be started with a previous working grid configuration, the associated stencil has to be supplied and is used to directly assemble the specified grid.

Boundary data structure. The boundary and wall property definitions of B2.5 are reused, i.e. boundary areas are defined by identifying the associated ghost cell blocks on the base grid. A separate boundary data structure for the working grid is introduced which stores face lists of boundary segments and can be modified to follow grid changes.

Plasma state data structure. All primary and derived plasma state variables are stored in one-dimensional lists, using the face and cell ordering of the grid. Input/output of the plasma state is done via the base grid using the existing routines.

The data structures are implemented using Fortran derived data types with dynamically allocatable components. This allows the code to easily work with multiple copies of the central data structured when required (cf. appendix A.4).

Input/Output

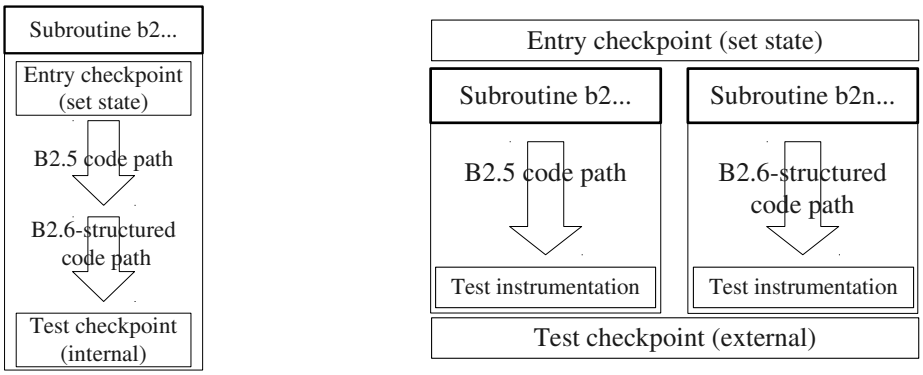
Plasma state, model parameters and further code configuration parameters are read and written with the existing B2.5 routines, using the B2.5 data structure as a transfer buffer. The same approach is taken for runtime diagnostics (residuals, traces, restart checkpoints). This allows to immediately reuse the rich pre- and post-processing tool-chain of B2.5 and the SOLPS package and completely avoids changes in the simulation input files. The downside is a significantly increased memory demand for B2.6. This can be avoided by only allocating the B2.5 data structure for solver initialization and finalization and selectively replacing the I/O routines used for runtime diagnostics.

Conversion of solver components

With the required data structures in place, the biggest block of work is to convert the central solver components (computation of coefficients, sources and fluxes, linear system assembly and solution) to use the new grid and plasma state data structure. This results in different "code paths" which effectively do the same computation but use different infrastructure and partially differing algorithms. A two-step approach is used in the conversion as outlined in figure 6.1.

First step: conversion. The new B2.6-structured code path using the UG structure is included in the same source files as the original B2.5-standard code path. At runtime the code then effectively performs all computations twice, both on the B2.5 and B2.6 data structure. This enables very detailed comparisons of the results of both code paths (using the *internal* variant of the testing framework described in section 6.3) and allows rapid development iterations during code conversion.

Second step: separation. The B2.5 and B2.6 code paths included in the same source file are split again, both retaining any previously shared control code. The B2.5 source files contain the same code as before the B2.6-structured conversion, plus additional instrumentation for the testing framework. After the split the two code paths are not executed within the same scope anymore. Therefore instead of the internal now the *external* testing framework has to be used for regression testing.



(a) Step 1: conversion

(b) Step 2: separation

Figure 6.1.: Conceptual two-step code conversion process. a) Duplication of code inside an execution unit (e.g. a subroutine) where extremely detailed tests between the code paths are possible. b) Separation of the source code into standalone units containing instrumentation for regression testing. The actual test is then deferred to a test checkpoint external to both units.

The control flow of the B2.5-standard and B2.6-structured code paths are essentially identical. The biggest change in the algorithms is switching from loops over cells (computing contributions for left and bottom faces) to loops over faces (computing contributions to the connected cells). Larger changes are introduced at the level of linear system assembly to achieve a stronger separation of numerics and physics. Also the boundary condition imple-

mentation requires major changes due to the more complex boundary data structure. Direct testing of individual intermediate quantities in these code regions is still possible and done within the external testing framework.

Merging changes from the B2.5 development line

Separating the code paths has one additional benefit. The source files containing the B2.5 code path are nearly identical to the main B2.5 development trunk, except for additional statements due to the test instrumentation. The modifications are small enough to allow automatic merging of changes from the B2.5 trunk into the B2.5 source files contained within the B2.6 code branch. Modifications done due to ongoing extensions of the main B2.5 development line can therefore be included without manual intervention. The necessary updates to the B2.6 code paths are then done manually with support of the testing framework.

Physics model

In the first step the "classical" part of the B2 model was converted. Drift terms and some specialized transport coefficients, extended wall models and boundary conditions were omitted due to time constraints. However, the conversion approach described above allows fast and reliable conversion of missing pieces. In case of the drift terms, B2.6-structured should take recent progress into account which resulted in a more robust numerical treatment [99].

Spatial discretization

B2.6-structured keeps the existing hybrid finite volume spatial discretization of B2.5. As shown by the negative results in the numerical benchmarks, the performance of this scheme degrades significantly on unstructured grids. The code is therefore artificially limited to structured grids (i.e. no cells with complex connectivity are allowed, only one neighbor per side). This is enforced during grid adaptation by averaging adaptation criteria in the poloidal and radial direction over the entire domain. An additional advantage of this is that existing interpolation schemes can be reused, which significantly speeds up code conversion.

Grid adaptation mechanism

B2.6-structured already includes all components necessary to enable grid modifications at runtime, resulting in algorithm 12. Adaptation is currently triggered at preset adaptation time steps. Criteria can be computed or collected at any point during the computation. The effective atomic grid operations are also used to modify the boundary data structure. Transfer of the plasma state to the new grid is done by first transferring it to the base grid

(using conservative interpolation based on linear reconstruction as described in 4.4.2) and then computing cell averages for the new grid.

Algorithm 12: B2.6 algorithm including grid adaptation

```
Solver initialization
Initialize working grid 𝒯^0
Initialize plasma state 𝒫^0
Initialize boundary structure ℬ^0
while t ≤ T do
    Outer iteration
        ⋮
        (equivalent to iterations in algorithm 10)
        ⋮
    t = t + Δt
    if (adaptation time-step) then
        Grid adaptation
            Evaluate criteria 𝒞 on grid 𝒯^i
            Create new grid 𝒯^{i+1} (→ algorithm 1)
            Create new plasma state 𝒫^{n+1} and boundary structure ℬ^{n+1}
            Destroy old structures 𝒯^i, 𝒫^i, ℬ^i
Solver finalization
```

Status of the B2.6-structured code

The regression testing framework enabled fast code conversion while establishing agreement between the B2.5 and B2.6-structured code paths to within very tight tolerances (the default relative error threshold for the tests is 10^{-10}) when running them on the same grid. At this point the B2.6-structured code path can replace the B2.5-structured solver completely, as demonstrated in chapter 7. Some parts of the B2.5 physics model are missing but can be added efficiently as needed. The separation of the code paths is only partially completed, meaning that in some sections both are executed (with the results of B2.5-structured being ignored). Furthermore no serial optimization of B2.6-structured was done yet, making direct performance comparisons difficult.

The entire grid adaptation mechanism is in place, and the code is demonstrated to perform flawlessly when running with grid adaptation in the benchmark given in chapter 7. However, due to known limitations of the hybrid spatial discretization it is restricted to structured grids (hence the name B2.6-structured). Convergence behavior of B2.6-structured on adapted structured grids is equivalent to B2.5 (see figure 6.2 for an example residual trace).

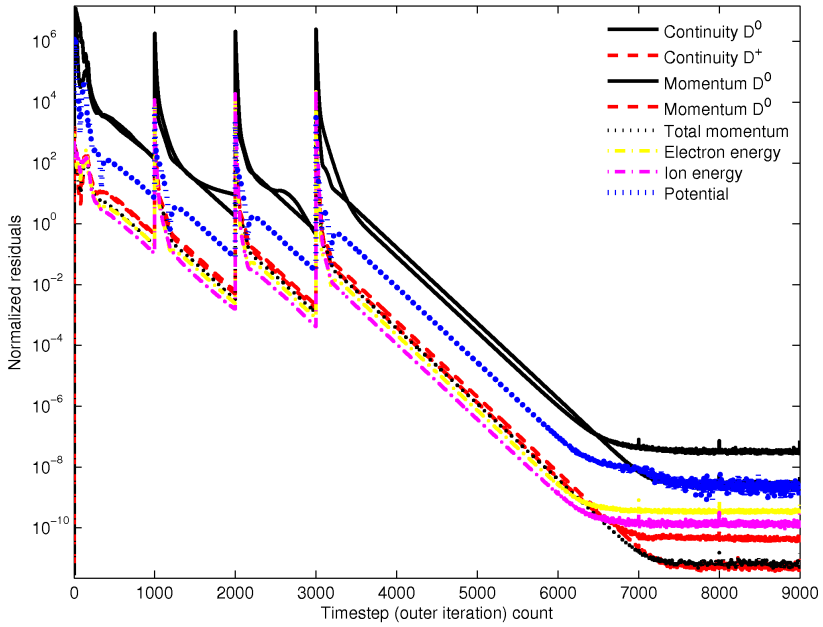


Figure 6.2.: Residual time trace of B2.6-structured running with grid adaptation enabled. The code starts from a flat (unconverged) initial solution. Every 1000 time-steps a grid adaptation cycle is performed, marked by jumps in the residuals. At time-step 3000 the grid settles into its final state.

6.2.3. The B2.6-unstructured code

The B2.6-unstructured code path is created by replacing the numerical scheme in the B2.6-structured path with the high-resolution FVM presented in section 4.4. In contrast to B2.6-structured, which mainly keeps the control flow and spatial discretization of B2.5, the changes required for B2.6 are of a different and more complex nature.

Spatial discretization

Replacement of the spatial discretization requires modifications in the following areas.

Coefficient computation. In B2.5, the transport coefficients of the model are computed using two approaches: computation on cells and use of generalized interpolation routines to obtain approximations at the cell faces, or use of ad-hoc interpolation at various places in the code. Most of these interpolations directly include geometric coefficients, and most of them break down for the more complex geometry of unstructured grids. This situation is similar to the problems described for the hybrid spatial discretization in section 4.3.1. At the same time the choice of interpolations can have a strong impact on both accuracy and convergence behavior. The combination of these issues requires some attention and makes conversion of the coefficient computation time-consuming.

Flux computations. The flux computations required for the residual in (6.4) are replaced with HR fluxes based on linear reconstruction. The flux linearization is assembled using the hybrid scheme. Using two spatial discretizations side-by-side requires changes to the transport coefficient definition, which must drop some geometric quantities previously included.

Defect correction iteration. Solution of the linear correction equations (6.4) is changed to use the defect correction algorithm 9.

Boundary conditions. For the hybrid scheme the existing boundary condition code can be re-used (the algorithm change to use the new boundary data structure was already done in B2.6-structured). The high-resolution fluxes require additional boundary condition specifications as described in section 4.4.5.

Conversion process

All necessary data structures and related infrastructure were already completed in the B2.6-structured conversion and are used in unchanged form for B2.6-unstructured. Implementation of the B2.6-unstructured starts by extending the B2.6-structured code path with the new HR solver components. The general control flow remains unchanged. The definition of some variables is changed w.r.t. the inclusion of geometric coefficients. This is taken into account in the testing framework by performing appropriate conversions.

Because both B2.6 code paths use the same data structures, internal in-scope comparisons are not possible. Instead, the external testing framework is used directly to compare B2.6-unstructured to B2.5-standard. Due to the significant changes in the numerical scheme, these comparisons are very different to the comparison done for B2.6-structured. Systematic differences in the approximations have to be taken into account, leading to much higher tolerances required for the tests. However, as the different codes still approximate the same physical quantities, the approach is still feasible, even for verification of B2.6-unstructured running on unstructured grids.

Status of the B2.6-unstructured code

Due to time constraints, the B2.6-unstructured code is only partially completed and currently cannot be used as a stand-alone solver. The explicit flux computations are largely converted to the HR scheme. The biggest block currently being worked on are the coefficient computations, where more general interpolation routines based on linear reconstruction are being developed. The implementation of boundary conditions is currently missing completely. Unfortunately the internal dependencies of quantities inside the code (e.g. heat fluxes build on particle fluxes, which in turn include a range of coefficients) prohibit separation of the implementation into work packages for the individual equations.

However, the choice to introduce separate code B2.6-structured and B2.6-unstructured code paths allows the continued development of aspects of the code benefiting from the new

data structures in the scope of B2.6-structured. Due to the shared infrastructure, B2.6-unstructured directly benefits from these changes. Most of the planned modifications listed in section 8.1 can therefore be developed and exploited while the work on B2.6-unstructured is still in progress.

6.3. Internal regression testing framework

During the code conversion, testing only global output quantities and relying on standard debugging techniques (e.g. run-time symbolic debuggers and selective instrumentation) to fix differences is not time-efficient considering the size of the code. Regression tests between the different code paths necessarily have to be done at a very fine-grained level. The regression testing framework described here was designed to allow fast code conversion from B2.5 to B2.6.

6.3.1. Internal testing

For the conversion to B2.6-structured calls to internal test routines are inserted to directly compare variables within the same scope (usually a subroutine). This test instrumentation is permanent (i.e. once put in, the testing code stays in place and can be centrally disabled). When running with tests enabled, on entering a code block to be tested the B2.5-standard state is transferred to B2.6 to establish the same input for both code paths. Inside the code block test calls comparing the output of the two code paths are placed as needed. When running the code, failing tests are recorded and stop the code at pre-specified checkpoints. Additionally, the diverging data sets are written out for analysis and visualization. The approach proved to be very efficient in the identification of coding errors during the B2.6-structured conversion. As default error criterion a maximum relative deviation of 10^{-10} is allowed. Complications arise for quantities passing through zero, for which specific absolute error thresholds have to be used.

6.3.2. External testing

Separation of the code paths removes the possibility of in-scope testing. The test calls are therefore changed to output data to an external testing framework. The standard approach used for this is to stream individual objects (mostly arrays) at test checkpoints to a file and tagging them with an unique identifier and meta-data (e.g. acceptable error tolerances). An external tool then automatically compares the output from different code paths and reports differences. The same data stream files are used to set the state at entry checkpoints (as done in the internal tests). The approach is sketched in figure 6.3. It enables direct comparison between B2.5-standard and either B2.6-structured or B2.6-unstructured. This external test

mechanism can easily be integrated into a build framework to provide automated regression testing.

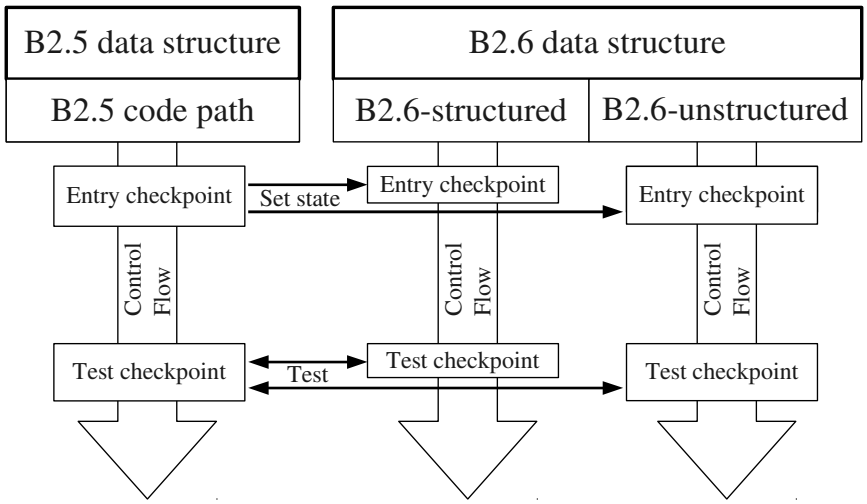


Figure 6.3.: Conceptual structure of the external testing framework. Entry checkpoints are used to read the state provided by the B2.6-standard code path into the B2.6 data structures. The results of a code block are then written out at test checkpoints and compared to the B2.6-standard results with external tools.

7. B2.6-structured adaptation benchmark

The B2.6-structured solver supports grid modifications at runtime, enabling the use of solution-driven grid adaptation under the restriction of structured grid connectivity. The following benchmark demonstrates adaptive simulations using feature detectors as adaptation criteria. A detailed comparison of key quantities to a reference solution gives some insight in the accuracy of the adaptive solution method.

7.1. Benchmark procedure

This steady-state benchmark case is a 2-species (pure deuterium plasma) scenario which is based on the ASDEX Upgrade L-mode discharge #16151. The same base grid as in the adaptation algorithm tests in section 3.5 is used (128×64 cells, shown in figure 3.19a).

Simulations are performed for two series of grids with varying resolution. The first set of grids represent a resolution scan to evaluate what errors can be expected when changing the grid resolution without taking the solution state into account. From these simulations, six "best case" grids with the best overall performance in terms of error vs. cell count are selected for comparison with the adaptive runs. A detailed overview including errors of the primary plasma quantities for this grid series is given in appendix C.2.1. The second grid series is generated using automatic adaptation as described in the next section.

Lacking an exact analytical solution, the converged solution on the base grid is used as the reference solution ϕ_{ref}. To compute the deviation of the solution obtained a working grid from the reference solution, the plasma solution ϕ_{bg} on the base grid is reconstructed from the solution ϕ_{wg} on each working grid using the linear reconstruction method described in section 4.4.2. The relative deviation is then computed as

$$\epsilon_{\mathrm{rel}} = \left| \frac{\phi_{\mathrm{bg}} - \phi_{\mathrm{ref}}}{\phi_{\mathrm{ref}}} \right| \tag{7.1}$$

and the measure (4.26) denoted E_p is computed (c.f. section 5.1.1), using the cell volume as weight. The only exception is the parallel ion velocity $u_{\parallel i}$, where instead of (7.1) the deviation relative to the local ion sound velocity is used

$$\epsilon_{\mathrm{rel},u_{\parallel i}} = \left| \frac{u_{\parallel i,\mathrm{bg}} - u_{\parallel i,\mathrm{ref}}}{c_s} \right|, \quad \text{with } c_s = \sqrt{\frac{kT_i}{m_i}}. \tag{7.2}$$

7.1.1. Adaptive simulations

The second set of grids is obtained using solution-driven grid adaptation. The adaptation runs start with the converged reference plasma state on a working grid with base grid resolution. One grid adaptation cycle is performed every 50 outer iterations (including the first iteration). The adaptation cycles stops after the fifth adaptation step (i.e. after 200 time steps), which is sufficient for the grids to settle into their final state. The solution is then converged on the resulting final grid.

Adaptation criteria

An adaptation criterion combining feature detectors for the primary plasma quantities ion density n_i, parallel ion velocity $u_{\|i}$, ion and electron temperature T_i, T_e and potential ϕ is used. For every quantity the data variation (3.6) at faces is computed, normalized by (3.8) and transferred to control volumes using the average (3.9). The criteria are combined to

$$c = \max\left(0.3c_{n_i}, 0.8c_{u_{\|i}}, 0.4c_{T_i}, 0.65c_{T_e}, 0.75c_{\phi}\right). \tag{7.3}$$

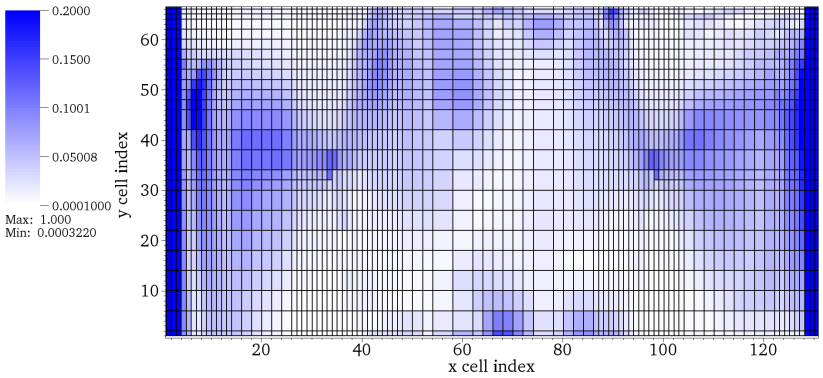
The additional weighting coefficients are chosen to balance the individual criteria.

To force the grid adaptation cycles to only produce grids with essentially structured connectivity, the combined criteria are averaged along coordinate lines. Cell criteria $c_{c,y}$ (controlling splits in the y direction) and face criteria c_f for x-aligned faces are averaged along the poloidal direction. Likewise, cell criteria $c_{c,x}$ (controlling splits in the x direction) and face criteria c_f for y-aligned faces are averaged along the radial direction. An example for the combined cell criterion value $c_{c,x}$ before and after averaging is shown in figure 7.1. As in the resolution scan, the grid resolution at the target plates is forced to the highest level.

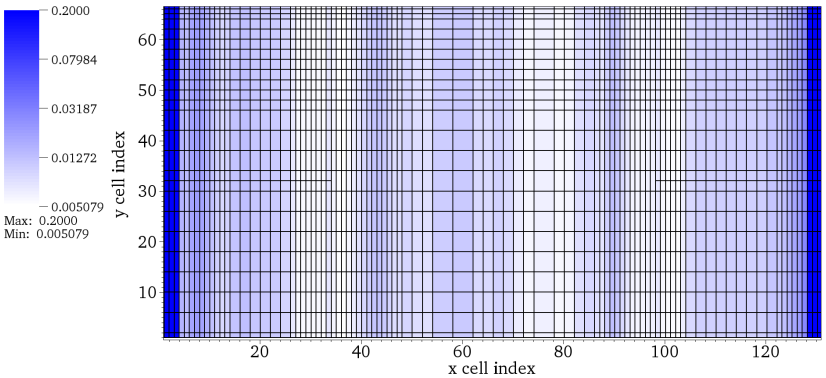
The action thresholds used for the benchmark are listed in table 7.1. As in the grid adaptation algorithm examples (section 3.5), the thresholds are multiplied with a linear factor (ranging from $[0.6 \ldots 5]$) to obtain a range of grids with varying solution. An example grid obtained in the adaptive runs is shown in figures 7.2-7.4. Detailed error plots for this specific grid are included in the following sections.

$C^{\mathrm{v}}_{\mathrm{c},x}$	$C^{\mathrm{s}}_{\mathrm{c},x}$	$C^{\mathrm{v}}_{\mathrm{c},y}$	$C^{\mathrm{s}}_{\mathrm{c},y}$	$C^{\mathrm{r}}_{\mathrm{f,x}}$	$C^{\mathrm{v}}_{\mathrm{f,x}}$	$C^{\mathrm{r}}_{\mathrm{f,y}}$	$C^{\mathrm{v}}_{\mathrm{f,y}}$
0.000	0.030	0.000	0.025	0.004	0.180	0.009	0.180

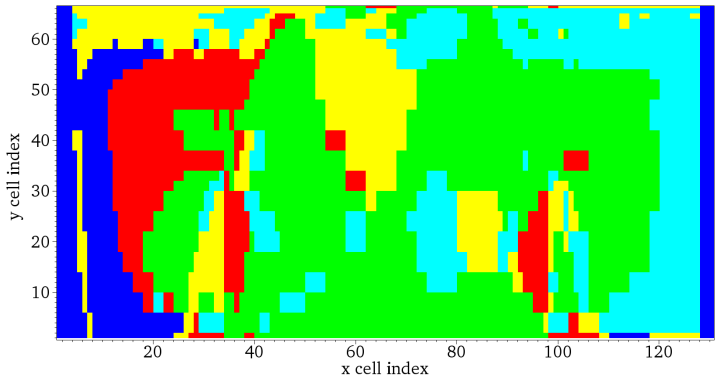
Table 7.1.: Cell and face action thresholds for the B2.6-structured adaptation benchmark. They are rescaled with a factor $\lambda \in [0.6, 5]$ to obtain a series of adapted grids with varying resolution.



(a) Combined $c_{c,x}$ criterion value without averaging. At the target plates the criterion is fixed to force refinement.



(b) Combined $c_{c,x}$ criterion value after averaging along the radial direction.



(c) Regions marked by active criterion.
Blue = n_i, cyan = $u_{\|i}$, T_i = green, T_e = red, ϕ = yellow

Figure 7.1.: Combined criterion for structured adaptation. Feature detectors are evaluated for the primary plasma quantities. The combined criterion (figure a) is constructed by selecting the maximum value computed by the feature detectors for the individual criteria. The averaged combined criterion is shown in figure b. Figure c shows which criterion produced the maximum $c_{c,x}$ value in individual grid cells.

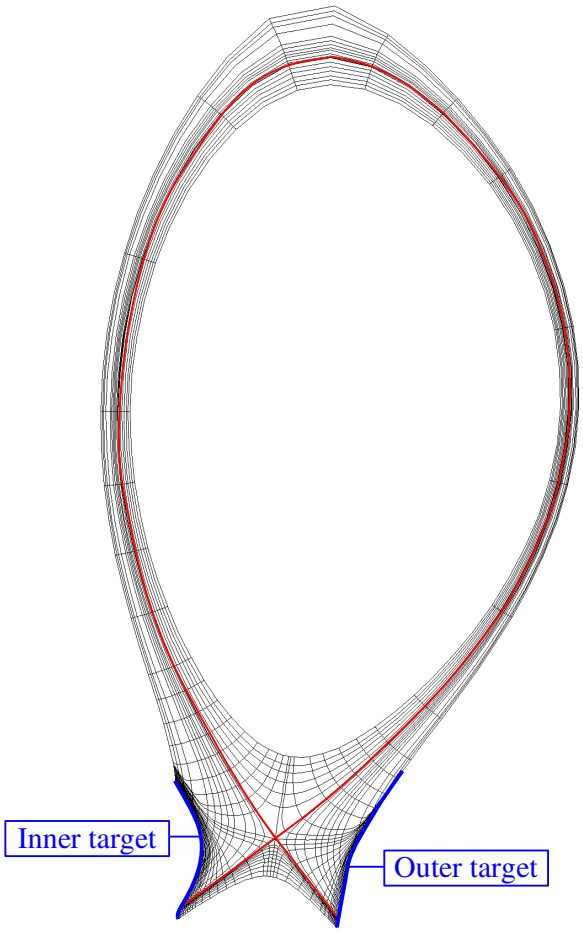
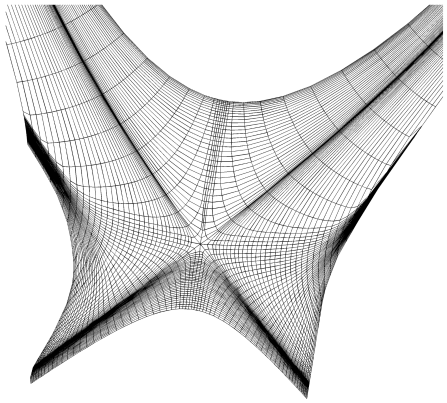
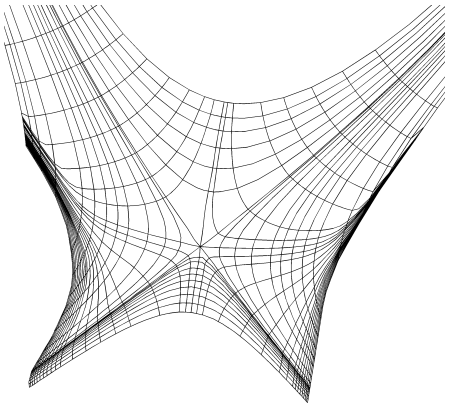


Figure 7.2.: Full grid (adapted, 1476 cells)



(a) Divertor region
(base grid, 8192 cells)



(b) Divertor region
(adapted grid, 1476 cells)

Figure 7.3.: Divertor plots

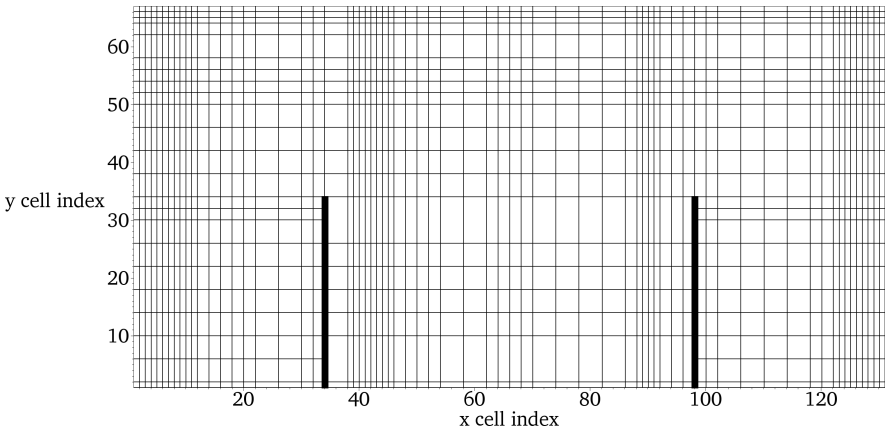


Figure 7.4.: Adapted grid with 1476 cells (+196 ghost cells), plotted in computational space (cf. figure 3.2 for the mapping). Plots of this grid in physical space are shown in figures 7.2 and 7.3b.

7.2. Results

7.2.1. Errors in primary plasma quantities

The average relative deviation of the primary plasma quantities for the two grid series is shown in figure 7.5. They are well below 10% already for low cell counts, showing that for this benchmark example low-resolution grids are already sufficient to obtain a good solution approximation. The grids created by the adaptation process a) manage to steadily decrease the error with increasing cell count and b) match or outperform the best-case grid selected from the resolution scan. An exception to this are the $u_{\|i}$ and T_i deviations on coarse grids (< 1000 cells).

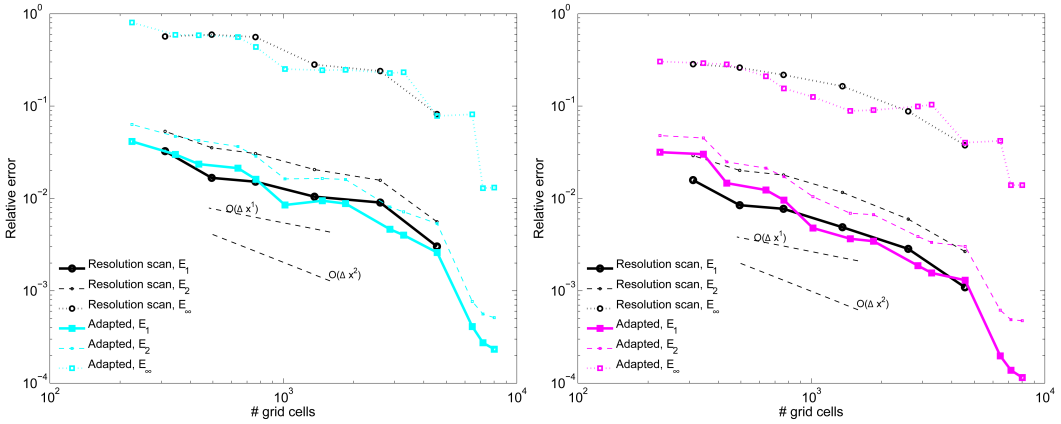
Two-dimensional plots of the relative error in the physical domain as obtained on an adapted grid with 1476 cells are shown in figures 7.6, 7.7 and 7.8. The temperatures and the potential exhibit similar error patterns, with regions of high relative deviation concentrated in the private flux region and around the x-point. For all quantities, it holds that regions with relative deviations of more than 10% are very localized and usually located in areas with high grid deformations (i.e. near the x-point and at the target plates). The deviations in the core are low.

7.2.2. Errors in target plate profiles

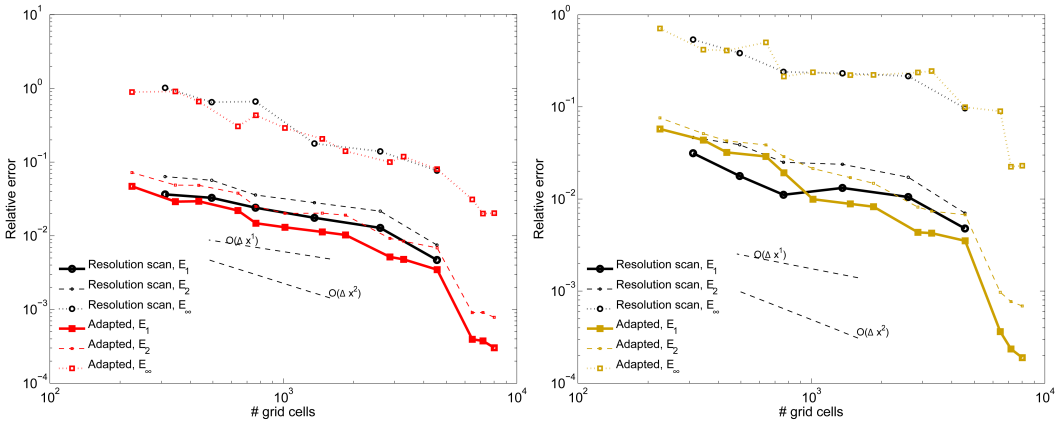
Also of interest is the approximation of solution profiles of some key quantities at the target plates (their position is shown in figure 7.2). Plots are given for the total energy flux on inner and outer target (figure 7.9), the electron temperature (figure 7.10) and the electron density (figure 7.11). The profile plots include a) the profile of the reference solution, b) the profile obtained on the example adapted grid with 1476 cells (figure 7.2) and c) for comparison also a profile on a grid from the resolution scan grid with 1188 cells (+172 ghost cells, effective cell stencil size is $(2, 4)$).

The relative errors observed for the profiles are in general in a similar range as the errors of the primary plasma quantities. Approximation of the qualitative shape of the profiles is robust even on coarse grids. Again, the adapted solution matches or outperforms the best-case solutions obtained in the grid scan.

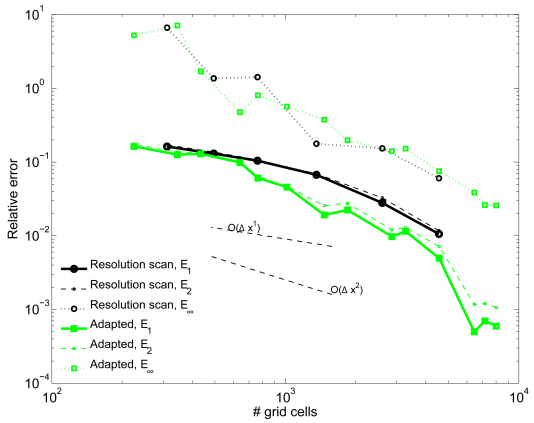
Figure 7.9 also includes data on the error of the total energy flux integral over the entire target plate area. The approximation of this quantity is very robust, to within 1% for grids exceeding 1000 cells.



(a) Ion density n_i (b) Parallel ion velocity $u_{\parallel i}$

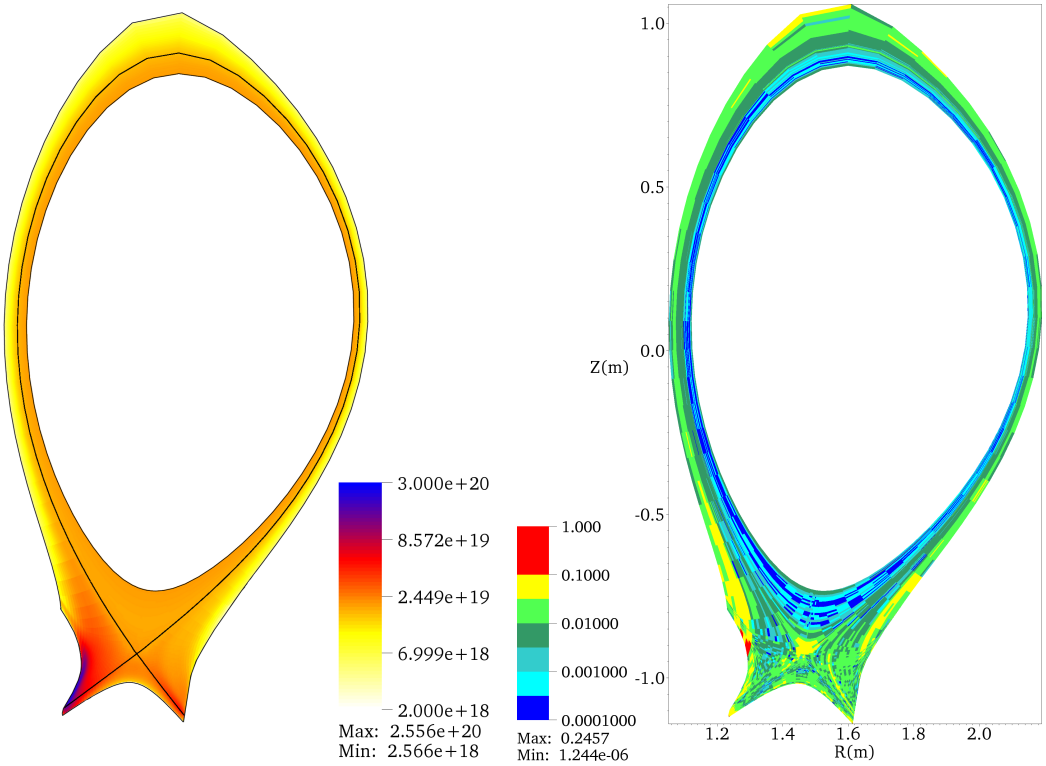


(c) Electron temperature T_e (d) Ion temperature T_i

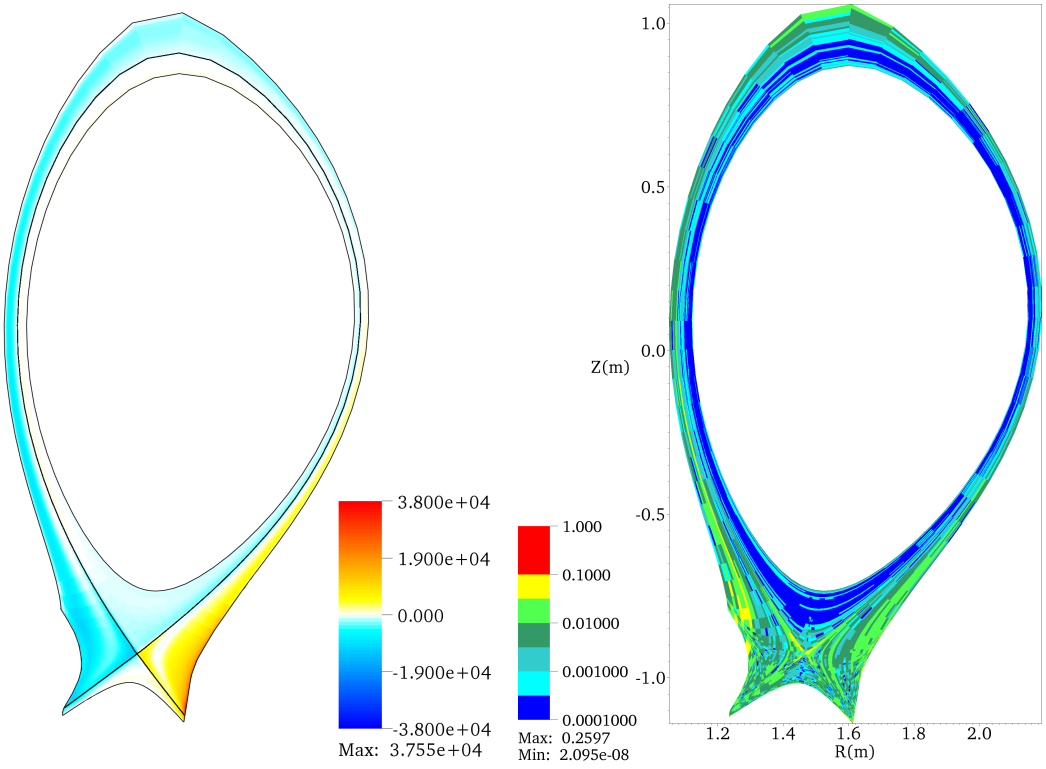


(e) Potential ϕ

Figure 7.5.: Relative errors of the primary plasma variables for the converged solutions on both grid series. Data from adapted grids is shown in color, data from the optimal resolution scan grids is shown in black.

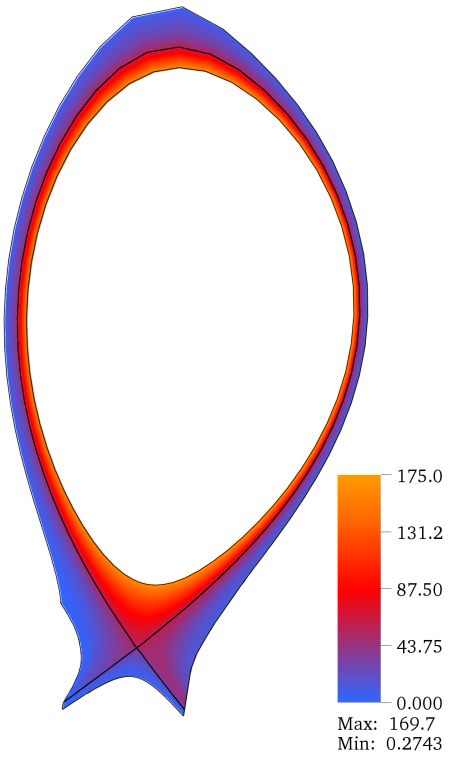


(a) Reference ion density $n_i(1/\mathrm{m}^3)$ (b) Relative n_i error

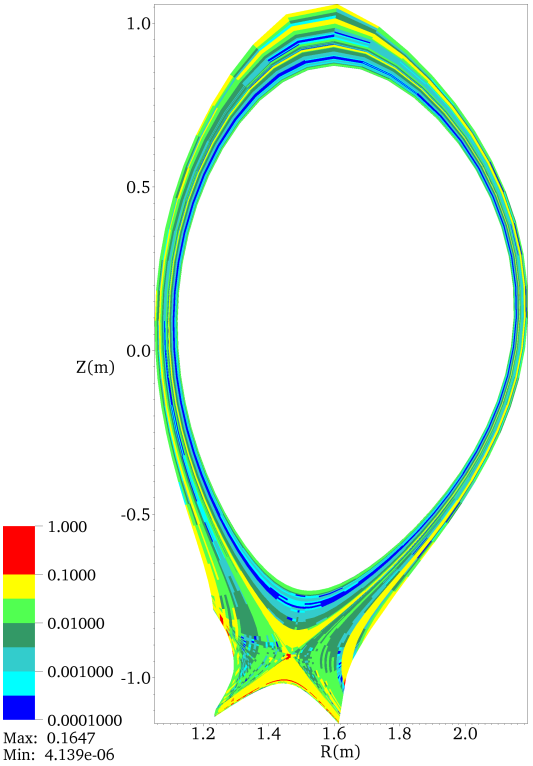


(c) Reference parallel ion velocity $u_{\parallel,i}(\mathrm{m/s})$ (d) Relative $u_{\parallel i}$ error

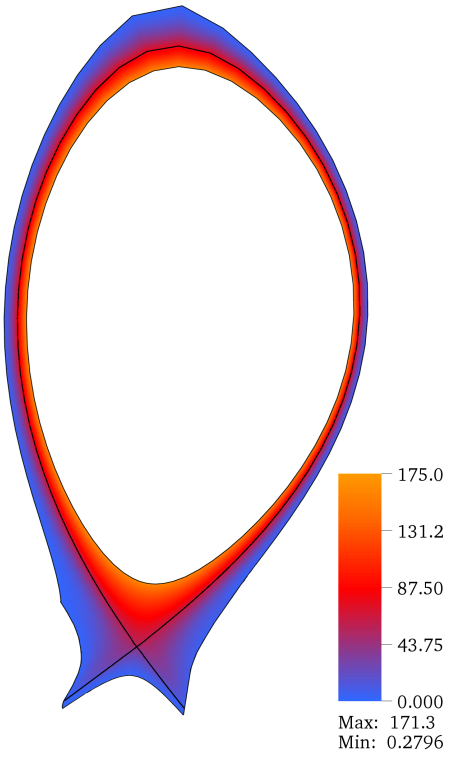
Figure 7.6.: Reference n_i, $u_{\parallel i}$ solutions and relative errors on grid shown in figure 7.2.



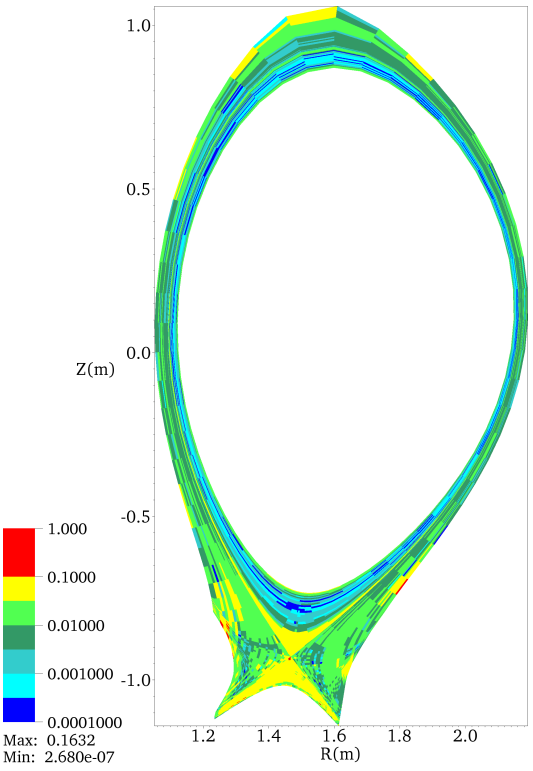
(a) Reference electron temperature T_e(eV)



(b) Relative T_e error

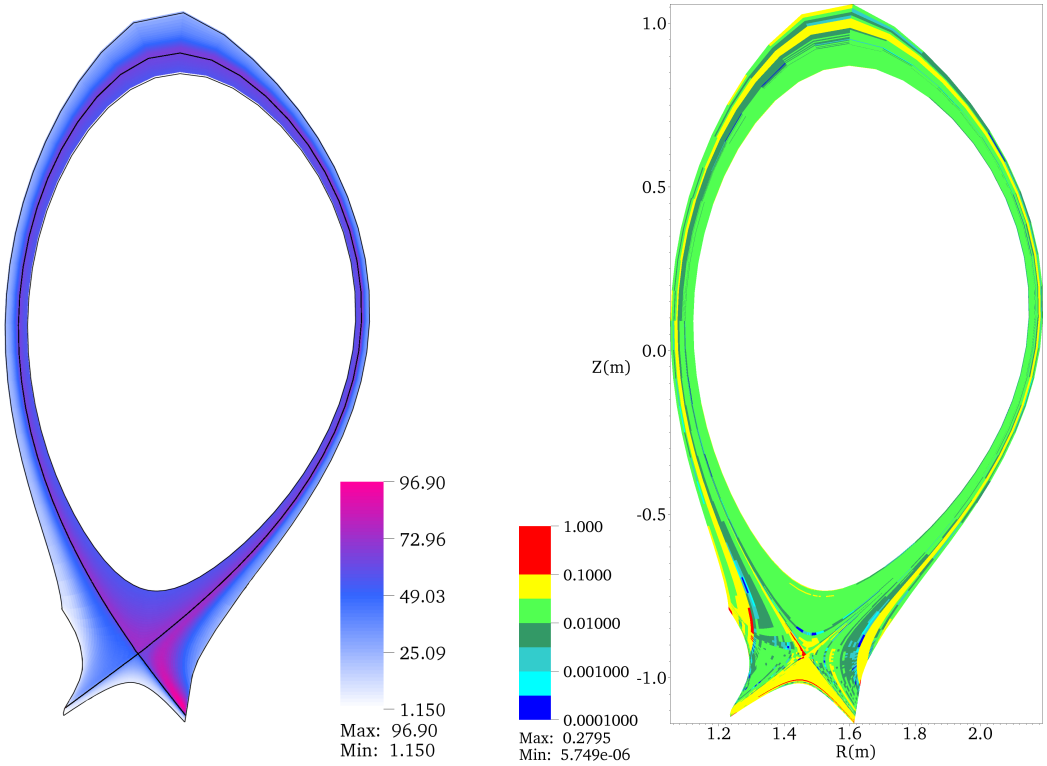


(c) Reference ion temperature T_i(eV)



(d) Relative T_i error

Figure 7.7.: Reference T_e, T_i solutions and relative errors on grid shown in figure 7.2.



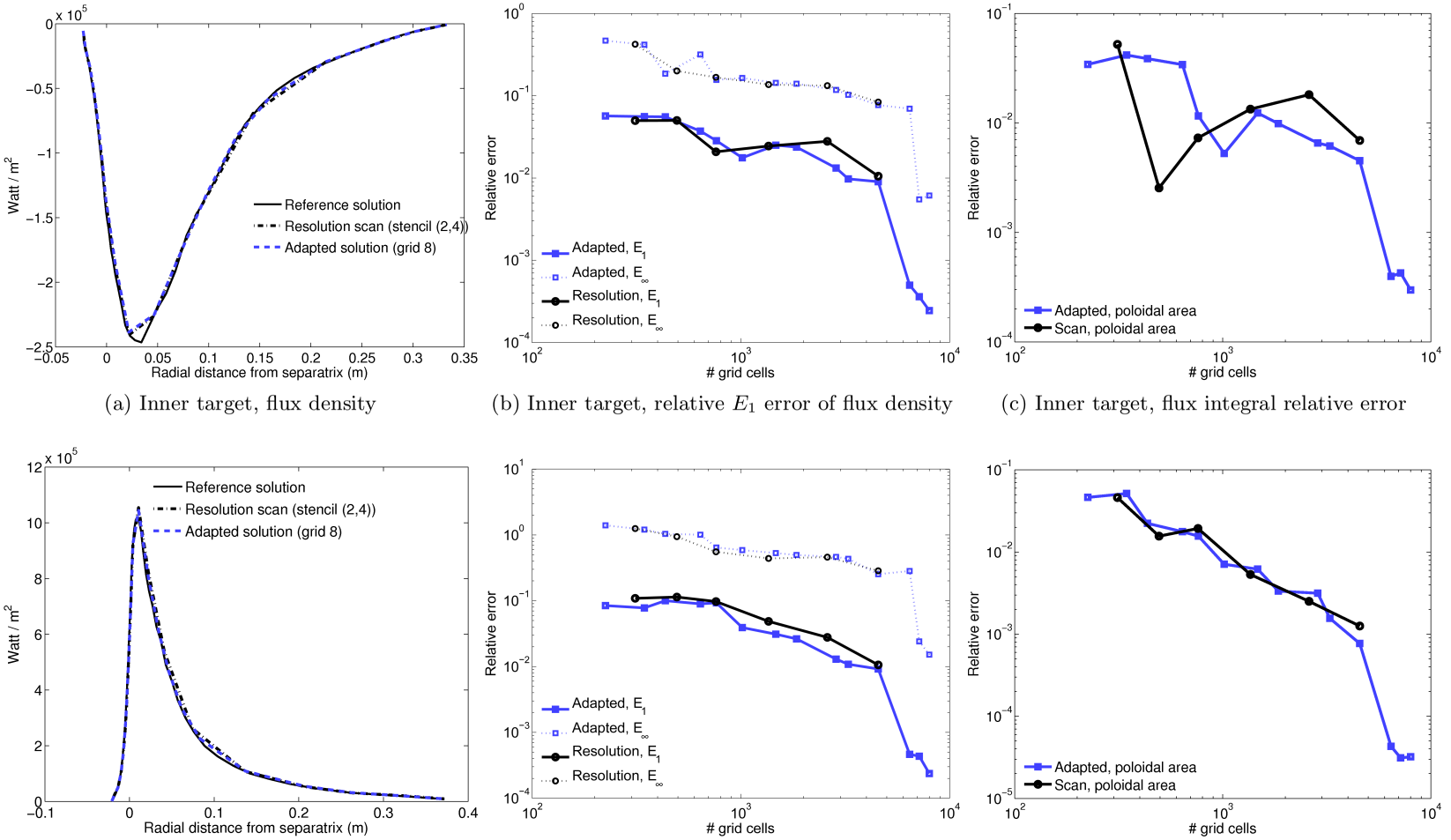
(a) Potential ϕ(V) (b) Relative ϕ error

Figure 7.8.: Reference ϕ solution and relative error on grid shown in figure 7.2.

7.3. Conclusions

The behavior of B2.6-structured when running on adapted grids coarser than the base grid resolution is very robust. The feature detection criterion, while being of rather simple form, delivers surprisingly good results. The grids it produces match the results of hand-picked grids with fixed cell-aspect ratios at all resolution levels, without the need to perform an extensive resolution scan first. Already in this early development stage (with the capabilities of B2.6-structured still limited to structured grids) this kind of solution-driven grid adaptation can be a useful tool for production-level simulations. The capability of the code to automatically create grids with different resolutions proved to be very useful for performing the grid resolution scan simulations. Scans of this type (which previously required manual grid conversion) are routinely possible with B2.6-structured.

Both B2.5 and B2.6-structured use the same solution approach and exhibit identical convergence behavior. However, grid adaptation cycles lead to large jumps in the residuals that in general do not decay quickly to the residual level before the adaptation step (see figure 6.2). In general, the flux balances contain contributions from very large individual fluxes, and even slight changes due to changing geometric coefficients or interpolation errors immediately cause large imbalances.



(a) Inner target, flux density

(b) Inner target, relative E_1 error of flux density

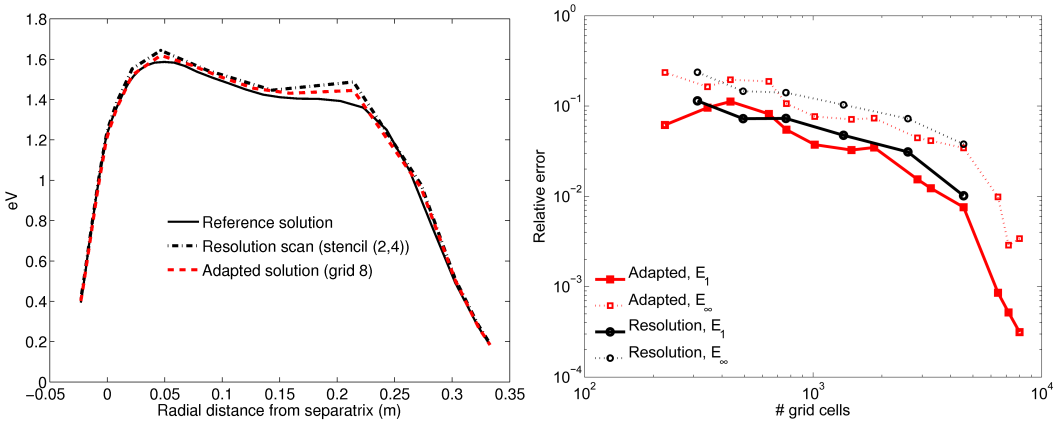
(c) Inner target, flux integral relative error

(d) Outer target, flux density

(e) Outer target, relative E_1 error of flux density

(f) Outer target, flux integral, relative error

Figure 7.9.: Total energy flux on inner and outer target. Plots a, d show the poloidal flux density profiles along the target plates on three individual grids (reference, resolution scan, adapted figure 7.2). Plots b and e show the relative error of the profiles for the two grid series. Plots c and f show the absolute deviation of the integral poloidal energy flux(in plot c the integral deviation passes through zero).



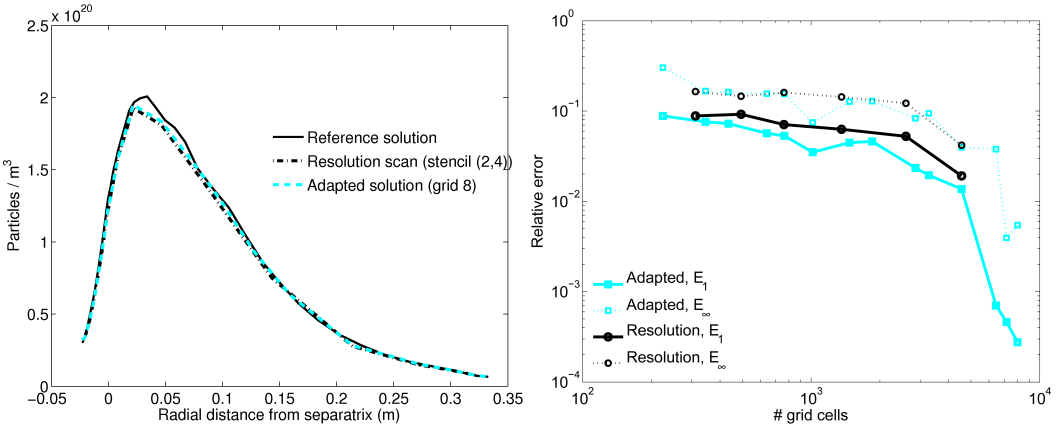
(a) T_e, inner target (b) T_e, inner target, relative E_1 error

(c) T_e, outer target (d) T_e, outer target, relative E_1 error

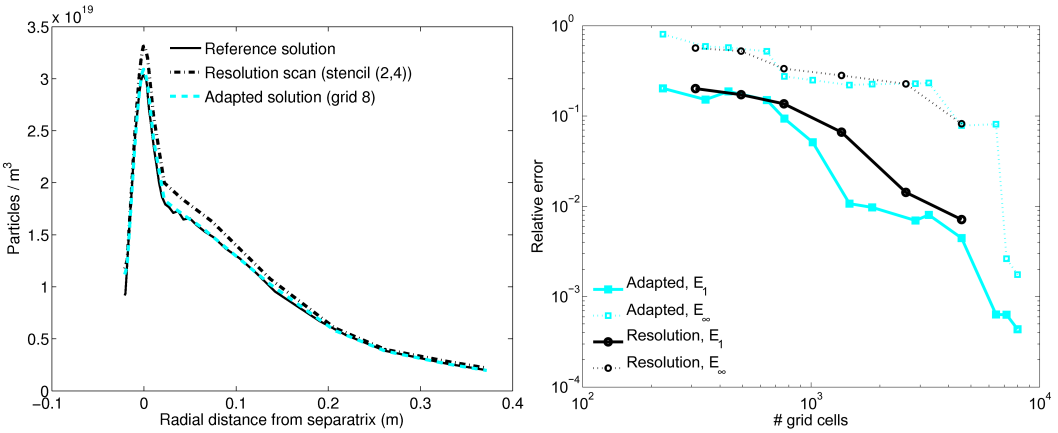
Figure 7.10.: Electron temperature T_e on inner and outer target. Plots a and c show profiles for individual grids, plots b and d show the profile deviation for the two grid series.

While the time required to complete an inner iteration drops as expected with decreasing grid cell counts, the relation between grid resolution and convergence speed is more complicated. In general, convergence speed of the runs performed for this steady-state benchmark case remains largely unaffected, with roughly 5000-6000 iterations required to reach convergence (not accounting for differing residuals after the adaptation phase). Some extremely coarse grids (< 400 cells) exhibit slower convergence and require up to 12000 iterations to converge. This is most likely related to the extreme difference in spatial resolution caused by forcing the grid to high resolution at the target plates. To control this kind of effect, a better understanding of the influence of grid resolution on the properties of the nonlinear iteration scheme is required, especially for the use with multigrid methods. The general topic of convergence behavior and acceleration, optimal choice of relaxation factors and optimization of the iteration scheme was not studied in this work. B2.6 can be used for systematic studies in this area.

For the moment, a good adaptation strategy is to adapt the grid during the starting phase



(a) n_e, inner target (b) n_e, inner target, relative E_1 error



(c) n_e, outer target (d) n_e, outer target, relative E_1 error

Figure 7.11.: Electron density n_e on inner and outer target. Plots a and c show profiles for individual grids, plots b and d show the profile deviation for the two grid series.

of the iteration, during which the solution structure starts to emerge but the residuals are still high. Later in the run, when the residuals dropped below a certain limit, adaptation is suspended to avoid increasing the residuals again.

The code separation between B2.6-structured and B2.5 is currently not complete. This and the fact that no serial optimization for B2.6-structured was done yet makes a clean comparison of code timings impossible. However, some observations can be made. Matrix assembly (coefficient and flux computation) scales linearly with cell count. The scaling of the solution time for the linear system obviously depends on the solver. Measurements for the MUMPS solver used in the benchmarks are presented in appendix B.3. For the relatively low-resolution grids used in B2 it scales roughly linearly with cell count. This behavior is expected to carry over to B2.6-structured and was approximately observed in the benchmark runs.

8. Summary and outlook

The increased complexity of scrape-off layer simulations performed for the modeling of magnetic confinement fusion devices calls for efficient numerical tools. This motivated the extension of the multi-species plasma fluid code B2 to support adaptive grids presented in this work.

Data structure and adaptation algorithms

The main challenge in the grid generation process for the B2 code is the strong anisotropy of particle and heat transport in the magnetized plasma, requiring field-aligned grids. To exploit the existing grid-generation workflow, adapted grids are built using a base grid / working grid approach. Based on the requirements of the B2 code, a flexible grid data structure was designed. It supports unstructured logically rectangular coordinate-mapped grids in 2 dimensions and is optimized for use with implicit finite volume spatial discretizations.

To enable efficient grid modifications the data structure is complemented by a global grid refinement and coarsening algorithm. It supports isotropic and anisotropic adaption strategies and is controlled through user defined adaptation criteria and action thresholds. Flexibility and robustness of the algorithms was demonstrated by adapting grids to converged B2 solutions using feature detectors as criteria. Efficient adaptation (and especially coarsening) in dynamic situations was shown for a time-dependent benchmark case.

Finite volume solver

Moving from structured to unstructured grids results in a more complicated cell geometry. This has to be reflected in the choice of spatial discretization. A high-resolution (HR) finite volume method based on limited linear solution reconstruction was considered as a replacement of the current B2.5 hybrid scheme. Both schemes were implemented in a solver for the advection-diffusion equation using the new data structure. Their performance on unstructured grids was compared using a steady-state benchmark case. As expected, the performance of the hybrid scheme degrades on unstructured grids while the HR scheme maintains its properties. For computations on unstructured grids upgrading B2 to the HR scheme is mandatory and has the additional benefit of providing second order accuracy for all flow conditions.

	B2.5	B2.6-structured	B2.6-unstructured
Hybrid FVM	✓	✓	-
High-resolution FVM	-	●	●
Grid adaptation	-	✓(structured)	✓(unstructured)
Standard physics model	✓	✓	●
Drift physics model	✓	×	×

Table 8.1.: Implementation status of the B2 code family. × = not started, ●= partially completed, ✓ = completed, - = not possible with this version.

Adaptive B2.6 code

The grid data structure, adaptation algorithms and finite volume solver form a self-contained Fortran 90 library that was implemented and verified separately. The new components were then introduced step-by-step into the B2.5 code, starting with conversion of the code to the new data structure while maintaining the hybrid finite volume discretization. The resulting B2.6-structured code contains all necessary components for criteria-controlled grid adaptation, greatly extending its technical capabilities (including simultaneous handling of multiple grids and plasma states). Owing to the continued use of the hybrid scheme, it is limited to structured grids and so far not the complete physics model been converted (drift physics are currently not included). A built-in regression testing framework is provided for verification against the B2.5 code, establishing exact agreement of the results when running on the same grid and allowing for easy implementation of the missing physics modules. The user interface of the B2.5 code is retained, allowing a direct migration to the new code version.

A benchmark case based on an ASDEX upgrade discharge was used to study the capabilities of grid adaptation in B2.6-structured using feature-detection criteria. The accuracy of solutions obtained on solution-adapted grids with respect to a reference solution was compared against solutions from a grid resolution scan. The adapted grids were shown to be as good or better than the best solutions obtained in the resolution scan for any given grid cell count. Good solution accuracy already for low cell counts and robust convergence was observed, resulting in an overall reduction of computation time.

Furthermore, in the scope of this thesis development of the extended code version B2.6-unstructured was started. It includes the HR finite volume method studied in this thesis. Based on the experience gained with the numerical benchmarks and B2.6-structured, due to the transition to a fully unstructured grid a further significant reduction of the grid cell count and computation time for a given accuracy is expected. The implementation status of the B2 code family is summarized in table 8.1.

Already in its present form B2.6-structured is a useful tool to improve the efficiency of scrape-off layer simulations. B2.6-unstructured will further extend the capabilities of the adaptive codes. Overall, the modernized software architecture of the B2.6 codes greatly extends the possibilities for further advances.

8.1. Future work

Further code development. The implementation of the HR scheme in B2.6-unstructured has to be completed and verification against B2.5 (as done for B2.6-structured) has to be performed. The physics model of both versions has to be extended to match the capabilities of B2.5. Inclusion of the drifts treatment should take the latest progress in the field into account.

Introduction into production use. The new code version has to be introduced into the main SOLPS distribution and offered to users as an option for production runs, including proper support and documentation. The focus of further improvements has to be based on experience collected from use of the code on real problems.

Adaptation criteria. An immediate necessity for a production-quality B2.6 code is to design and test set of robust "standard" adaptation criteria based on the feature detectors which enable users to take advantage of the new features with minimal effort. A challenging longer-term project is the derivation of adaptation criteria based on error estimators and grid optimization for output functionals.

Grid orthogonality. Due to limitations of the current B2.5 data structure, the grid generator has to relax the grid orthogonality constraint close to the target plates depending on the device geometry. This has possibly adverse effects on the solution accuracy and requires special attention in the discretization. An approach to avoid this problem entirely with the new data structure is presented in appendix D.1.

Multigrid. The capability of the B2.6 code generation to handle multiple grids and plasma states combined with easy access to grid hierarchies due to efficient coarsening (as demonstrated in section 3.5.2) allows the implementation of geometric multigrid algorithms for convergence acceleration with relatively little effort. Of particular interest is the so-called full approximation scheme (FAS) for nonlinear problems, with the iterative relaxation of the coupled equation system acting as a smoother. The anisotropy in the problem can be addressed with semi-coarsening, i.e. changing the resolution depending on the coordinate direction. This is now possible with the anisotropic adaptation strategy. The multigrid approach can be combined directly with grid adaptation and the defect correction technique [59, 113, 69].

Parallelization. With the neutrals code EIRENE becoming available in a parallelized version, parallelization of the B2 codes becomes important. Due to the structure of the sequential relaxation algorithm and use of relatively low-resolution 2d grids, parallelization on the level of the linear solver and use of domain-decomposition techniques is unfeasible. More promising is parallelization on a per-equation basis. A first step is to distribute the solution computation of continuity and momentum equations to single cores. Some notes on this can be found in appendix D.2.

Appendices

A. Notes on the B2 model and codes

A.1. Properties of curvilinear coordinates

This section outlines some basic properties of curvilinear coordinate mappings. A comprehensive reference (especially in the context of magnetic plasma confinement) is [49].

A coordinate mapping $T : \Omega_\xi \to \Omega_x, \vec{x} = \vec{x}(\vec{\xi})$ assigns to every point $\vec{\xi}$ in a set $\Omega_\xi \subseteq \mathbb{R}^3$ (defined by the position vector $\vec{\xi} = (\xi^1, \xi^2, \xi^3)$ in the ξ-coordinate system) a corresponding point $\vec{x}$ in another set $\Omega_x \subseteq \mathbb{R}^3$ (defined by the position vector $\vec{x} = (x^1, x^2, x^3)$ in the x-coordinate system). The Jacobian of the mapping is

$$J \equiv J\left(\vec{x}\left(\vec{\xi}\right)\right) \equiv \frac{\partial \vec{x}\left(\vec{\xi}\right)}{\partial \xi^1} \cdot \left(\frac{\partial \vec{x}\left(\vec{\xi}\right)}{\partial \xi^2} \times \frac{\partial \vec{x}\left(\vec{\xi}\right)}{\partial \xi^3} \right) = \begin{vmatrix} \frac{\partial x^1}{\partial \xi^1} & \frac{\partial x^1}{\partial \xi^2} & \frac{\partial x^1}{\partial \xi^3} \\ \frac{\partial x^2}{\partial \xi^1} & \frac{\partial x^2}{\partial \xi^2} & \frac{\partial x^2}{\partial \xi^3} \\ \frac{\partial x^3}{\partial \xi^1} & \frac{\partial x^3}{\partial \xi^2} & \frac{\partial x^3}{\partial \xi^3} \end{vmatrix} \tag{A.1}$$

For the mapping to be invertible, $J \neq 0$ must hold in Ω_x. If this is the case, the inverse mapping is $T' : \Omega_x \to \Omega_\xi, \vec{\xi} = \vec{\xi}(\vec{x})$, and every point in the sets Ω_x and Ω_ξ can be identified uniquely with its coordinates in the x- or ξ-system.

The coordinate curves $\{\vec{x}(\vec{\xi}) : \xi^\alpha, \xi^\beta = const\}$ for a system are obtained by holding two coordinates fixed and varying the remaining coordinate. The coordinate surfaces $\{\vec{x}(\vec{\xi}) : \xi^\alpha = const\}$ are obtained if one coordinate is fixed and the remaining two are varied. In the simplest case of a *rectilinear* grid, the coordinate curves are straight lines. In the general case of arbitrary formed coordinate curves the coordinate system is called *curvilinear*. If the coordinate curves intersect in every point at right angles, the curvilinear system is called *orthogonal*.

Co- and contra-variant basis vectors

At every point $x \in \Omega_x$ two sets of basis vectors are defined. The *covariant* basis vectors (written with superscript indices α) are defined as

$$\vec{e}_{(i)} \equiv \frac{\partial \vec{x}\left(\vec{\xi}\right)}{\partial \xi^i}, \quad i = 1 \ldots 3. \tag{A.2}$$

They are *tangential* to the ξ^i coordinate curves. The second set are the *contra-variant* basis vectors (written with subscript indices i)

$$\vec{e}^{(i)} \equiv \nabla \xi^i, \quad i = 1 \ldots 3. \tag{A.3}$$

They are *perpendicular* to the coordinate surfaces. Co- and contra-variant basis vectors are closely related: the vector sets are *reciprocal*, meaning $\vec{e}^{(i)} \cdot \vec{e}_{(j)} = \delta_i^j$, where δ is the Kronecker symbol. The bases are *local*, they depend on position.

Any vector $\vec{u} = (\tilde{u}_1, \tilde{u}_2, \tilde{u}_3) \in \mathbb{R}^3$ can then be expressed as

$$\vec{u} = \sum_{i=1}^{3} u_i \vec{e}^{(i)} = \sum_{i=1}^{3} u^i \vec{e}_{(i)}. \tag{A.4}$$

$u_{(i)} = \vec{u} \cdot \vec{e}^{(i)}$ are the *covariant components* of the vector (written with subscripts like the covariant basis vectors) and $u^i = \vec{u} \cdot \vec{e}_{(i)}$ the *contra-variant components* (written with superscripts like the contra-variant basis vectors). When working with physical units, it should be noted that $\vec{e}_{(i)}$ and $\vec{e}^{(i)}$ are not unit vectors and the co- and contra-variant components have different units than $\tilde{u}_i$.

Metric coefficients

The metric coefficients are of central importance for working with curvilinear coordinates. The covariant coefficients g_{ij} and contra-variant coefficients g^{ij} are defined as vector products of the basis vectors

$$g_{ij} \equiv \vec{e}_{(i)} \cdot \vec{e}_{(j)}, \quad g^{ij} \equiv \vec{e}^{(i)} \cdot \vec{e}^{(j)} \tag{A.5}$$

and form a symmetric matrix. They are related by

$$\delta_i^k = \sum_{j=1}^{3} g^{kj} g_{ji}. \tag{A.6}$$

The determinant g of the covariant coefficient matrix $[g_{ij}]$ is related to the Jacobian J by

$$g \equiv \det[g_{ij}] = \begin{vmatrix} \partial\frac{x^1}{\xi^1} & \partial\frac{x^1}{\xi^2} & \partial\frac{x^1}{\xi^3} \\ \partial\frac{x^2}{\xi^1} & \partial\frac{x^2}{\xi^2} & \partial\frac{x^2}{\xi^3} \\ \partial\frac{x^3}{\xi^1} & \partial\frac{x^3}{\xi^2} & \partial\frac{x^3}{\xi^3} \end{vmatrix} \cdot \begin{vmatrix} \partial\frac{x^1}{\xi^1} & \partial\frac{x^2}{\xi^1} & \partial\frac{x^3}{\xi^1} \\ \partial\frac{x^1}{\xi^2} & \partial\frac{x^2}{\xi^2} & \partial\frac{x^3}{\xi^2} \\ \partial\frac{x^1}{\xi^3} & \partial\frac{x^2}{\xi^3} & \partial\frac{x^3}{\xi^3} \end{vmatrix} = (J)^2 \tag{A.7}$$

or $J = \sqrt{g}$.

The scale factors $h_i = |\vec{e}_{(i)}|$ are defined as the length of the tangential basis vectors. With

$$\tilde{\vec{e}}_{(i)} = \vec{e}_{(i)}/h_i \tag{A.8}$$

being the tangential unit basis vectors, the metric coefficients can be rewritten as

$$g_{ij} = h_i h_j \tilde{\vec{e}}_{(i)} \cdot \tilde{\vec{e}}_{(j)} = h_i h_j \cos \sphericalangle(\vec{e}_{(i)}, \vec{e}_{(j)}). \tag{A.9}$$

For the diagonal entries this gives $g_{ii} = h_i^2$ or $h_i = \sqrt{g_{ii}}$. Conversion between covariant and contra-variant components is then possible by

$$u_i = \sum_{j=1}^{3} g_{ij} u^j, \quad u^i = \sum_{j=1}^{3} g^{ij} u_j. \tag{A.10}$$

Orthogonal curvilinear coordinates

For an orthogonal system the off-diagonal entries $g_{ij}, g^{ij}, i \neq j$ are zero. The expression for the Jacobian then simplifies to

$$J = \sqrt{(g)} = h_1 h_2 h_3. \tag{A.11}$$

Furthermore the relation $g_{ii} = 1/g^{ii}$ holds.

As already noted, the basis vectors are in general not unit vectors and therefore the co- and contra-variant vector components have units. The vector components in the covariant (tangential) direction are given in physical units (i.e. the real length of the projection on unit basis vectors) by

$$\tilde{u}^i = h_i u_i = \frac{u^i}{h_i}. \tag{A.12}$$

A.2. Sequential coupled relaxation in B2.5/B2.6

This section describes the explicit implementation of the sequential coupled relaxation algorithm in the inner iteration (cf. algorithm 11) of B2.5 and B2.6.

Momentum equations and pressure coupling

The individual momentum equations are solved separately for the velocity updates $\delta u_{\parallel a}$. Exact cancellation of the inter-species friction terms $R_{ab\parallel}$ is enforced by solving an additional total momentum equation (obtained as the sum of the individual momentum equations) for the total velocity update $\delta u_{\parallel t}$. The momentum update is then

$$u_{\parallel a} = u^*_{\parallel a} + \alpha_{u_\parallel} \mathrm{med}(\delta u_{\parallel a}, \delta u_{\parallel t}, \delta u_{\parallel a} + \delta u_{\parallel t}) \tag{A.13}$$

with $u^*_{\parallel a}$ being the previous approximation, med the median value and $\alpha_{u_\parallel}$ the momentum under-relaxation factor.

For compressible flow, both density n_a and velocity $u_{\parallel a}$ occurring in the particle flux Γ_a are primary variables and must be updated consistently to correct the residual of the continuity equation. To increase robustness in the presence of (effectively incompressible) low Mach number flows a pressure coupling approach is used. A change in partial pressure δp_a causes the (linearized) corrections

$$p_a = p^*_a + \alpha_p \delta p_a, \quad n_a = k\alpha_p \delta p_a, \quad u_{\parallel a} = u^*_{\parallel a} - \alpha_p c b_x \frac{1}{h_x}\frac{\partial p_a}{\partial x}. \tag{A.14}$$

The coefficient $k = 1/(T_i + Z_a T_e)$ is given by the equation of state $p_a = n_a(T_i + Z_a T_e)$ with fixed T_e, T_i. The velocity update has to be chosen so that the momentum flux divergence and the pressure gradient term in the momentum equation (2.5) cancel. As in the SIMPLE algorithm [90, 53], the linear FVM discretization of the momentum equation is exploited directly by choosing $c_i = 1/a^{m,a}_{ii}$, where $a^{m,a}_{ii}$ is the diagonal entry for cell Ω_i of the corresponding linear system (4.51). Substitution of the corrections (A.14) (without the α_p factor) into the continuity equation (2.3) and linearization yields the pressure-correction equation (expanded in the poloidal coordinate system)

$$\frac{\partial}{\partial x}\left[\frac{\sqrt{g}}{h_x}\left((u^*_{\parallel a} + V_{a,x})k\delta p_a - (kD_{a,x} + D^p_{a,x} + n^*_a c b_x^2)\frac{1}{h_x}\frac{\partial \delta p_a}{\partial x}\right)\right] \\ \frac{\partial}{\partial y}\left[\frac{\sqrt{g}}{h_y}\left(V_{a,y}k\delta p - (kd_{a,y} + D^p_a)\frac{1}{h_y}\frac{\partial \delta p}{\partial y}\right)\right] = -R_{n_a}. \tag{A.15}$$

R_{n_a} is the residual of the continuity equation for species a. Due to compressibility, (A.15) is in in advection-diffusion form and the pressure field is fixed (in the incompressible case only the relative pressure is relevant, and A.15 would be a Poisson equation). Analysis of this pressure coupling is best done in a distributive iteration framework [119].

Potential equation iteration

The potential ϕ is modified to follow the change of the electron density by

$$\phi^0 = \phi^* + \frac{T_e}{e}\frac{2(n_e - n_e^*)}{n_e + n_e^*}, \tag{A.16}$$

where n_e^* is the value before n_e after the density update. When the drift physics model is used, an accurate solution of the potential equation (2.7) is required. It is therefore solved iteratively for $\delta\phi$ to update $\phi^{i+1} = \phi^i + \alpha_\phi \delta\phi$, with a re-computation of the current after every update. The iteration stops when the potential residual drops below a preset value.

Heat equation coupling

The electron and ion energy equations (2.9),(2.12) are solved for the temperature updates δT_e, δT_i. The strong coupling of the energy equations through the exchange term Q_{ei} is treated by solving a total energy equation (obtained by summing (2.12) and (2.9)) for the total correction δT_t. The temperature updates are then

$$T_e = T_e^* + \alpha_T \underbrace{\frac{\delta T_e + \delta T_t}{2}}_{\delta T_e'}, \quad T_i = T_i^* + \alpha_T \underbrace{\frac{\delta T_i + \delta T_t}{2}}_{\delta T_i'}. \tag{A.17}$$

Furthermore the potential and velocities are updated to

$$\phi = \phi^* + \alpha_T \frac{\delta T_e'}{e}, \quad u_{\parallel a} = u_{\parallel a}^* - \alpha_T c_T b_x \frac{1}{h_x}\frac{\partial(n_e \delta T_e' + n_i \delta T_i')}{\partial x}. \tag{A.18}$$

The velocity updates are chosen to compensate for the pressure change due to the temperature update. The coefficient c_T is derived from the linearization of the total momentum equation in a similar way as done for the pressure-correction equation.

A.2.1. General implementation notes

A number of techniques are used to stabilize the algorithm, with an emphasis on robust convergence behavior in the presence of strongly nonlinear behavior of the model. Typical values for the under-relaxation factor α in the code are $0.1 \ldots 0.5$. An additional equation-specific under-relaxation is used that rescales the diagonal coefficient of the linear system (4.51) with the local absolute residual.

Transport coefficients are mainly computed for cells and are then interpolated to faces using varying interpolation methods (linear interpolation, volume-weighting, harmonic averages where required for stability on coarse grids).

Source terms are linearized to fit into (6.3), when necessary the linearization is modified to maintain the positive coefficient condition (4.32). Source integrals are computed using the midpoint rule. Boundary condition implementation is done by modification of the source linearization in the ghost cells.

A.3. Notes on B2 grids

This section contains some notes on special extension of the UG data structure (cf. section 3.2.2) required to support B2 grids.

A.3.1. Special vertices

Special points like an x-point (shown in figure A.1) can have more than four incident faces and cells, exceeding the capacity of the F_V and Ω_C index lists. These vertices can be included in the UG data structure by leaving the lists empty (this convention is also used to identify these vertices). Because the vertex-to-face connectivity information is only used in the adaptation algorithms, (and special points by definition cannot be removed), this poses no special problems.

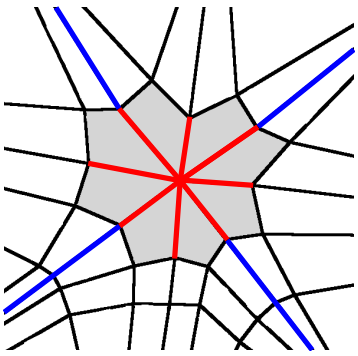


Figure A.1.: X-point with more than four incident faces. Points like this can be included in the grid but require special treatment in some parts of the algorithms.

A.3.2. Non-aligned grid blocks

Rule (3.3) aligns grid cells along a hierarchy in which the stencil origin of the biggest possible composite cell covering the entire base grid is at $(o_x, o_y) = (1, 1)$. For block-structured grids the sub-blocks don't necessarily have dimensions that are powers of two and therefore do not align with the hierarchy. This is the case for most existing B2 grids. Faces at the boundary of a block are persistent and cannot be removed. This effectively stops coarsening of cells at

a certain level. Further coarsening of such grids (up to the coarsest possible grid with one cell covering every block) can be enabled by

- using separate grid hierarchies within every sub-block (aligned to the origin cell of the block) and
- partial relaxation of the conditions on cell stencils in a merge group at the boundary of a block. This can be implemented by assuming cells to extend outside the block boundary to match the stencil size of their interior neighbors. The same logic is used to determine cell sizes when performing cell splits at the boundary.

An example where this extension is applied is shown in section 3.5.2 for a double-null geometry.

A.4. Data structure implementation

The data structure implementation uses the allocatable array extension of Fortran 95 (ISO/IEC TR 15581, now included in the Fortran 2003 standard). It allows derived data types with allocatable arrays as components. An example is the following reduced grid data structure used in the solver (omitted lines are marked by dots).

```
type, public :: G2DEStripped
        ! Pointer to extended grid structure
        type(Grid2dEdge), pointer :: master => null()
        ! Pointers to stencil and base grid
        type(G2DEStencil), pointer :: st => null()
        type(Grid2dBase), pointer :: bg => null()
                    :
        integer(iKind) :: nCv = 0  ! ncv: number of control volumes in grid
                    :
        ! Cv-face connectivity. Dimensions:
        !   1: Decreasing (IDEC=1) or increasing (IINC=2) direction
        !   2: Face index ( 1 : nFc )
        integer(iIndex), dimension(:,:), allocatable :: fcCv
                    :
        ! Faces connected to cv, in same ordering as in Grid2dEdge,
            ControlVolume2d
        ! Dimensions: 1 - local face index ( 1 : G2DE_CV_NFACE )
        !             2 - cv index ( 1 : nCv )
        integer(iIndex), dimension(:,:), allocatable :: cvFc
                    :
        ! Area of face in physical space
        real(rKind), dimension(:), allocatable :: aFc ! (nFc)
        ! Volume of cv in physical space
        real(rKind), dimension(:), allocatable :: vCv ! (nCv)
                    :
end type G2DEStripped
```

Allocatable components can be allocated inside a subroutine. For every data structure associated constructor and destructor routines are provided.

```
subroutine createG2DEStripped( g2de, grid )
  type(Grid2dEdge), intent(in), target :: g2de
  type(G2DEStripped), intent(out) :: grid
                    :
  ! Transfer statistics
  grid % nCv = g2de % nCv
                    :
  ! link grid
  grid % master => g2de
  grid % st => g2de % stencil
  grid % bg => g2de % bGrid
```

```
                    ⋮
    ! allocate arrays
    allocate( grid % cvFc( G2DE_CV_NFACE, grid % nCv ) )
    allocate( grid % fcCv( 2, grid % nFc ) )

    allocate( grid % aFc( grid % nFc ) )
    allocate( grid % vCv( grid % nCv ) )
                    ⋮
  end subroutine createG2DEStripped
```

```
  subroutine destroyG2DEStripped( grid )
    type(G2DEStripped), intent(inout) :: grid
                    ⋮
    grid % nCv = GRID_UNDEFINED
                    ⋮
    nullify( grid % master )
    nullify( grid % st )
    nullify( grid % bg )
                    ⋮
    ! deallocate arrays
    deallocate( grid % aFc )
    deallocate( grid % vCv )
                    ⋮
  end subroutine destroyG2DEStripped
```

State variables are stored in one-dimensional lists. Timing tests with varying compilers showed that using allocatable components has an negligible effect on array access speed.

```
  ! Primary plasma state variables
  type B2Plasma
          integer(iKind) :: ns ! number of species
          ! density and velocity (second index is species)
          real(rKind), dimension(:,:), allocatable :: na, ua
          ! potential, electron temperature, ion temperature
          real(rKind), dimension(:), allocatable :: po, te, ti
  end type B2Plasma
```

Using this form of derived types a) significantly reduces the length of subroutine parameter lists, b) dramatically improves code readability and maintainability and c) allows use of multiple instances of the data structures in the code.

B. Notes on numerical methods

B.1. Classical finite volume schemes

For the discretization of the advection term in (4.5), the hybrid finite volume described in section 4.3 switches between the central and the upwind flux approximations depending on the local Péclet number. To provide a clearer picture of the hybrid discretization, the following section discusses these schemes separately. Good references for this are [90, 119].

B.1.1. Central differences

Assume that the line $\overrightarrow{x_d x_i}$ between the centroids of the two control volumes Ω_d and Ω_i connected by the face e_{di} passes through the midpoint x_m of the face (see figure B.1). In this case a linear profile of the solution between the centroids can be assumed to derive (in combination with the midpoint integration rule) the flux approximation

$$F^a_{di} = |e_{di}| u_{di}((1-\lambda_{di})\phi_d + \lambda\phi_i) \tag{B.1}$$

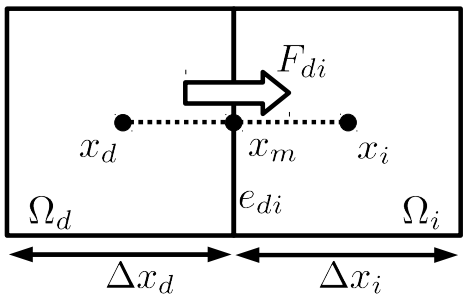


Figure B.1.: Simplified flux geometry

for the advective flux and

$$F^d_{di} = |e_{di}| D_{di} \frac{\phi_i - \phi_d}{\frac{1}{2}(\Delta x_d + \Delta x_i)} \tag{B.2}$$

for the diffusive flux. u_{di} is the velocity and D_{di} the diffusivity at the face. The coefficient $\lambda = \Delta x_d/(\Delta x_d + \Delta x_i)$ defines the weights for linear interpolation. Due to the equivalence of (B.2) and (B.1) to a central difference approximation of the first derivative, these fluxes are often referred to as the *central difference scheme (CDS)*.

Combining the two fluxes gives

$$F^{ad}_{di} = |e_{di}| \left(\underbrace{\frac{u_{di}\Delta x_i - 2D_{di}}{\Delta x_d + \Delta x_i}}_{a_d} \phi_d + \underbrace{\frac{u_{di}\Delta x_d + 2D_{di}}{\Delta x_d + \Delta x_i}}_{a_i} \phi_i \right) \tag{B.3}$$

Analyzing the scheme in the one-dimensional form (B.2) shows a discretization error of $\mathcal{O}(\Delta x^2)$, which also holds for non-smooth structured grids [119]. This behavior is observed in the numerical benchmarks on structured grids presented in section 5.1.

Considering how this flux contributes to the flux balance equation for control volume Ω_d, the coefficient a_i is positive as long as the numerical Péclet number is smaller than 2. Therefore, according to theorem 3, the scheme does not maintain the maximum principle for all flow conditions and is prone to develop unphysical solution oscillations.

A direct solution to this problem is to decrease the effective numerical Péclet number. This can be done by either reducing the grid spacing Δx (refining the grid in regions with a high P) or artificially increasing the diffusion.

B.1.2. The first-order upwind scheme

Flux B.2 can be modified to give a scheme of positive type by changing the discretization of the advective term to

$$F_{di}^{a} = \frac{u_{di}}{2}(\phi_d + \phi_i) + \frac{|u_{di}|}{2}(\phi_d - \phi_i) \; = \begin{cases} u_{di}\phi_d & \text{for } u_{di} \geq 0 \\ u_{di}\phi_i & \text{for } u_{di} < 0 \end{cases} , \tag{B.4}$$

which chooses the value of ϕ at the face coming from the upwind side with respect to the advective flow direction. It is often referred to as the *upwind difference scheme (UDS).* Combining this with the central diffusion flux B.2 results in the flux approximation (again given for the simplified situation of figure B.1)

$$F_{di}^{ad} = |e_{di}| \left[\underbrace{\left(\frac{u_{di} + |u_{di}|}{2} - \frac{D_{di}}{\frac{1}{2}(\Delta x_d + \Delta x_i)} \right)}_{a_d} \phi_d + \underbrace{\left(\frac{u_{di} - |u_{di}|}{2} + \frac{D_{di}}{\frac{1}{2}(\Delta x_d + \Delta x_i)} \right)}_{a_i} \phi_i . \right] \tag{B.5}$$

A negative contribution of the advective flux to the coefficient a_i is prevented, yielding a scheme of positive type for any Péclet number. The price for this favorable property is indicated by theorem 4. Analysis of the one-dimensional form (B.5) shows an truncation error of $\mathcal{O}(h)$ for purely advective flows [119].

For the advection equation, a clearer view on the error is obtained by deriving the *modified equation.* A Taylor expansion is substituted into the numerical approximation and higher-order terms of the truncation error are written out. For the UDS flux (B.4) this gives (developing around the position of the face e_{di}, assuming the upwind value is taken at the

center the upwind control volume)

$$\left.\begin{array}{ll} F_{di} = |e_{di}|u_{di}(\phi(x_{di}) - \frac{\Delta x}{2}\phi_x(x_{di}) + O(\Delta x^2)) & \text{for } u_x > 0 \\ F_{di} = |e_{di}|u_{di}(\phi(x_{di}) + \frac{\Delta x}{2}\phi_x(x_{di}) + O(\Delta x^2)) & \text{for } u_x < 0 \end{array}\right\}$$
$$= u_x\phi_{di} - |u_x|\frac{h}{2}\phi_x(x_{di}) + O(\Delta x^2) \quad \text{(B.6)}$$

This shows that the upwind flux is a second-order approximation to an advection-diffusion flux with a diffusion coefficient depending on the grid spacing. This artificial increase of the local Péclet number maintains the positivity of the scheme.

The $\mathcal{O}(\Delta x)$ truncation error of the upwind scheme necessitates the use of very fine grids in order to achieve reasonable accuracy. This significantly increases runtime and memory requirements of simulations. Therefore the use of at least second-order accurate schemes is advised [73].

B.1.3. Role of the continuity equation

A common property of the linear flux balance equation (4.48) of a conservative scheme is

$$a_j + \sum_{k\in\mathcal{S}_j, k\neq j} a_k = 0 \quad \forall\Omega_j \in \mathcal{T} \quad \text{(B.7)}$$

i.e. the central coefficient a_j exactly cancels the other coefficients of the flux linearization [90]. Ignoring complications due to boundary conditions, this ensures that a constant solution ϕ solves system (4.51). This condition is obviously not fulfilled by the coefficients given in (4.50):

$$a_j + \sum_{k\in\mathcal{S}_j, k\neq j} a_k = \underbrace{-\sum_{k\in\mathcal{S}_j, k\neq j} F^a_{jk}}_{①} \quad \text{(B.8)}$$

Term ① contains only advective fluxes and corresponds to the discretization of $\nabla\cdot\vec{u}$, i.e. the continuity equation. It disappears for a divergence-free velocity field $\vec{u}$. This has consequences depending on the type of flow under consideration.

- For incompressible flows, $\nabla\vec{u} = 0$ must hold. Eliminating ① amounts to explicitly enforcing this constraint and makes the scheme more robust in the presence of non divergence-free velocity fields (as might occur for intermediate steps of iterative pressure-correction schemes). (B.7) is then satisfied exactly.
- For compressible flows term ① must be kept or treated explicitly.

B.2. Properties of time-stepping schemes

This section summarizes key properties of time-stepping schemes, following [65].

B.2.1. Consistency and convergence order

Definition 12. ***Consistency order.*** *Let $\phi(t)$ be the exact solution and ϕ_{n+1} an approximate solution at time-step $n+1$ after performing one time-step starting from $\phi_n = \phi(t_n)$. A time-stepping scheme is said to have consistency order p if for the* local error *in one time-step holds*

$$\phi_{n+1} - \phi(t_{n+1}) = \mathcal{O}(\Delta t^{p+1}). \tag{B.9}$$

Integrating over the time interval $[0, T]$ with a time-step Δt, the local errors originating from the $T/\Delta t$ time-steps accumulate into the global error*, which is then $\mathcal{O}(\Delta t^p)$. Assuming stability and a sufficiently smooth function F, the scheme is then said to be* convergent of order *p.*

B.2.2. A-Stability

The concept of stability describes the effect of perturbations in the input data on the solution. It is commonly studied by considering the linear test equation

$$\phi'(t) = \lambda\phi(t), \quad \lambda \in \mathbb{C}. \tag{B.10}$$

Stability of one-step methods

Definition 13. ***A-Stability region for one-step methods.*** *Let $z = \lambda\Delta t$. Applying a one-step scheme (4.76) to (B.10) yields the recursion $\phi_{n+1} = R(z)\phi_n$. $R(z)$ is the* stability function *of the method. The set*

$$\mathcal{S} = \{z \in \mathbb{C}, |R(z)| \leq 1\} \tag{B.11}$$

is the stability region of the method*. If $\mathcal{S}$ contains the left complex half-plane $\mathbb{C}^-$, the method is called* A-stable*. If additionally $|R(\infty)| < 1$ holds, it is called* strongly A-stable*, and* L-stable *if also $|R(\infty)| = 0$.*

A-Stability of multi-step methods

Let again $z = \lambda \Delta t$. Applying the multi-step method (4.80) to equation (B.10) yields the recursion

$$\sum_{j=0}^{k} (\alpha_j - z\beta_j) w_{n+j} = 0, \quad n = 0, 1, \ldots. \tag{B.12}$$

The stability region $\mathcal{S} \subset \mathbb{C}$ is the set of all $z \in \mathbb{C}$ for which the recursion (B.12) is bounded for all starting sequences $w_0, \ldots, w_{k-1}$. If $0 \in \mathcal{S}$, the method is called *zero-stable*. The definitions for A- and L-stability from the one-step methods also apply directly to the multi-step methods.

The stability region of a multi-step method is related to its characteristic polynomial. The requirement for zero-stability limits the consistency order of explicit multi-step methods to $p = k$ and for implicit schemes to $p = 2\lfloor (k+2)/2 \rfloor$ (first Dahlquist barrier). The second Dahlquist barrier states that A-stable linear multi-step methods are at most of consistency order $p = 2$. A relaxed criterion called A(α)-stability can be used to obtain methods of order up to six (in practice four).

When using separate discretizations in space and in time, stability requires that $z = \Delta t \lambda \in \mathcal{S}$ for all eigenvectors λ of the spatial discretization, possibly imposing a limit on the time step Δt. This limit is commonly expressed using the Courant number.

Definition 14. ***Courant Number***. *The Courant number for the 1d advection equation*

$$\phi_t + a\phi_x = 0, \quad a > 0 \tag{B.13}$$

is defined as

$$\nu = \frac{\Delta t \ a}{\Delta x} \tag{B.14}$$

with Δt the time-step and Δx the spatial grid resolution. For the 1d diffusion problem

$$\phi_t - d\phi_{xx} = 0, \quad d > 0 \tag{B.15}$$

a similar number is defined by

$$\mu = \frac{\Delta t \ d}{\Delta x^2}. \tag{B.16}$$

In the multi-dimensional case, the absolute Courant numbers of the individual dimensions are added to

$$\nu_{2d} = \frac{\Delta t \ |a_x|}{\Delta x} + \frac{\Delta t \ |a_y|}{\Delta y}. \tag{B.17}$$

Following [65], for the classical schemes in section B.1 the eigenvalue criteria $z = \Delta t \ \lambda \in \mathcal{S}$ are given in table B.1.

Of the methods listed in section 4.5, the Courant time-step limit only applies to the explicit ones. The maximum Courant number for a given time-stepping scheme can be determined

$z_{a,\mathrm{UDS}} = \nu(e^{-2i\omega} - 1)$	First order upwind advection
$z_{a,\mathrm{CDS}} = \frac{1}{2}\nu(e^{-2i\omega} - e^{2i\omega})$	Second order central advection
$z_{d,\mathrm{CDS}} = \mu(e^{-2i\omega} - 2 + e^{2i\omega})$	Second order central diffusion

Table B.1.: Eigenvalue criteria for classical schemes: $z = \Delta t\ \lambda \in \mathcal{S}$ with $\omega \in [-1/2\pi, 1/2\pi]$.

Spatial discretization	Expl. Euler	One-step midpoint	Expl. BDF-2
UDS Advection	1.0	1.0	0.66
CDS Advection	0	0	0
CDS Diffusion	0.5	0.5	0.33

Table B.2.: Courant limits for combinations of schemes when applied to (B.13) and (B.15)

experimentally (see [65] for detailed tables for common single- and multi-step methods). The values for the methods considered here are summarized in table B.2.

Both the implicit backward Euler and the implicit BDF-2 scheme are A-stable, and are therefore stable for arbitrarily large time-steps.

B.2.3. Positivity and Monotonicity

The concept of positivity was already discussed for the spatial discretizations. Positivity can be violated during time integration even for positive spatial schemes, requiring special attention when choosing the discretization in time. In general, requiring positivity can impose a time step limit even for implicit methods.

Positivity for linear systems

Consider a linear ODE system of the form $\phi' = A\phi(t)$, with the matrix $A \in \mathbb{R}^{n\times n}$ satisfying

$$a_{ij} \geq 0 \text{ for } i \neq j \quad \text{ and } a_{ii} \geq -\alpha \quad \forall i = 1, ..., n. \tag{B.18}$$

The parameter $\alpha > 0$ is problem-specific and A has no eigenvalues on the positive real axis. Using again the stability function R of a one-step scheme, the approximation at time step $n + 1$ is

$$\phi_{n+1} = R(A\Delta t)\phi_n.$$

The threshold factor γ_R is the largest real number so that $R(x) > 0$ holds for $x \in [-\gamma, 0]$. For the homogeneous linear system the scheme will then be positive if Δt is limited so that $\alpha\Delta t \leq \gamma_R$.

Of the single-step methods given in section 4.5, only the implicit Euler method maintains positivity for arbitrary step sizes. For explicit Runge-Kutta methods with $s = p \leq 4$ the threshold is $\gamma_R \leq 1$. A general result is that the consistency order of a scheme that is positive for arbitrary time steps is at most one [26]. For multi-step methods, the startup phase also

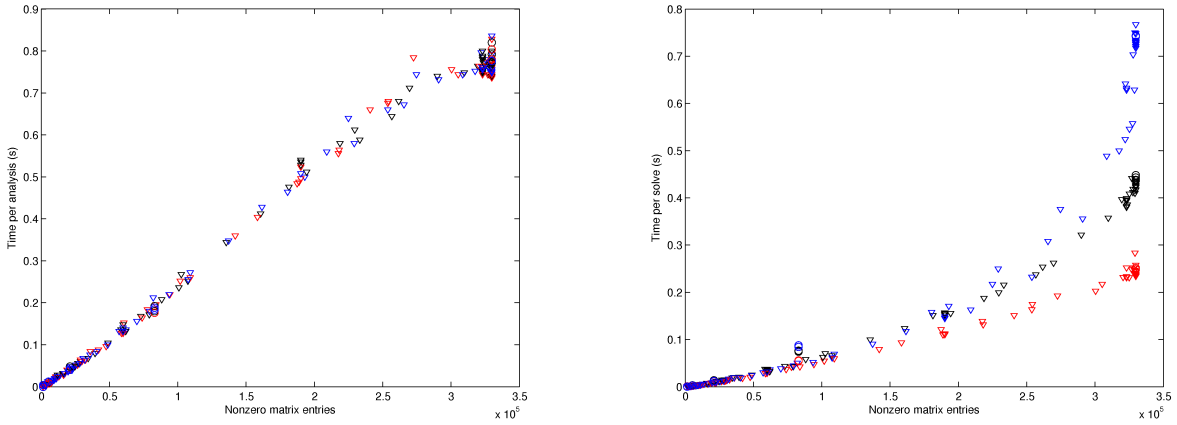
has to be considered. For the BDF2 method $\gamma_R = \frac{1}{2}$ is obtained, a detailed analysis can be found in [64]. Results for the inhomogeneous and nonlinear case be found in [65].

B.3. Solution of linear systems

The implicit discretizations discussed in chapter 4 require the solution of sparse linear equation systems. The matrices are square, have real entries and are in general unsymmetrical. The dimensions N of the system is given by the number of cells in the spatial grid. Common values for the benchmark discussed in this thesis range up to $N = 10^5$. The sparse matrices are stored using a modified CSR-type data structure (derived from the SMLIB library [81]) which allows direct modification of diagonal entries in the assembled matrix.

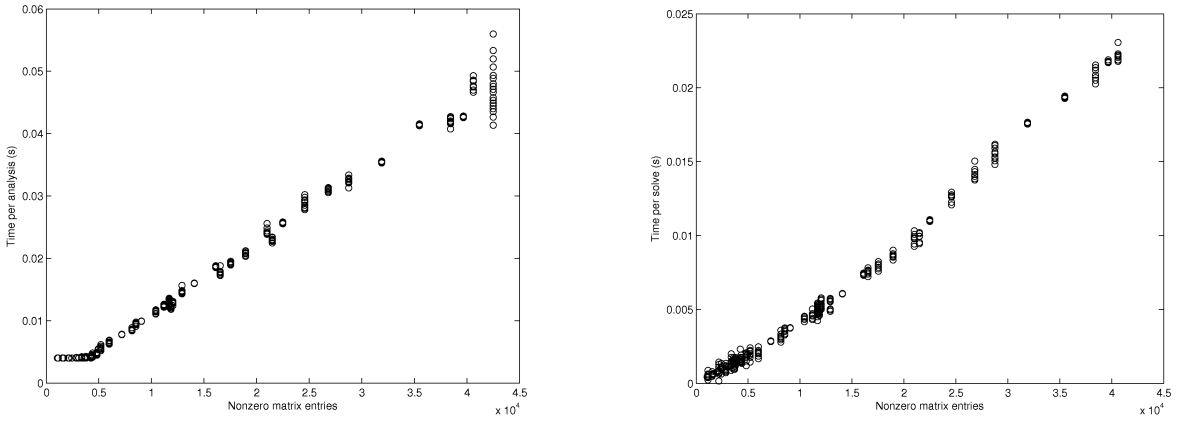
The direct multi-frontal sparse matrix solver MUMPS [13, 3] running in serial mode was used. It uses separate analysis, factorization and solution steps, allowing reuse of previous analysis results to speed up repeated inversion of the same matrix. This makes it well suited for use with the defect-correction method discussed in section 4.4.4.

To gain some understanding on the increase of computation time with increasing grid cell count, timing measurements for the matrix analysis and factorization+substitution phase were performed when running the benchmark cases. The results are presented in figures B.2 and B.3. Because different compute nodes were used for the verification problem and the B2 benchmarks, the absolute timing between figures B.2 and B.3 cannot be compared. Within the plots the data is directly comparable. Of interest are not the absolute values but the scaling trends.



(a) Analysis (b) Substitution

Figure B.2.: Timings for the MUMPS solver as measured for the steady-state benchmark problem in section 5.1. Every dot represents one grid. Colors indicate which benchmark case (cf. section 5.1.1) was run: black = case 1 (diffusion), red = case 2 (advection), blue = case 3 (anisotropic transport). For every dot the timing was sampled at least 30 times.



(a) Analysis (b) Substitution

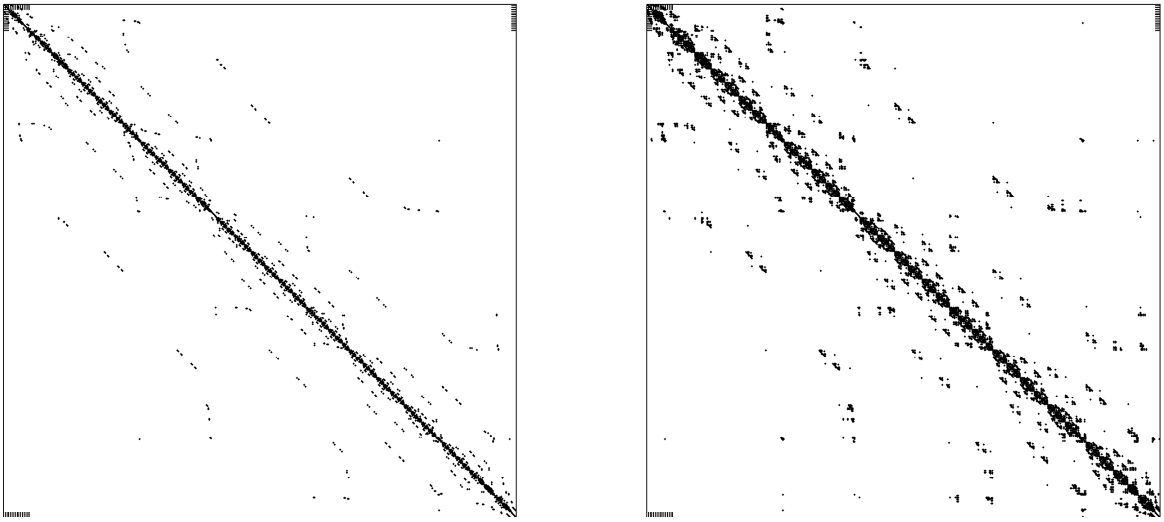
Figure B.3.: Timings for the MUMPS solver as measured for the B2.6-structured adaptation benchmark in chapter 7. Every dot represents one grid. For every grid the timing was sampled at least several hundred times. Exceptions are the points with the highest resolution (to the far right), here no sampling was done and therefore the measurements are not reliable.

B.4. Grid object orderings

Atomic grid modifications can be implemented in different ways, and while despite their conceptual simplicity this can be quite involved due to the multitude of connections between the objects, the details are not of interest here. What is relevant is the effect of object insertions and removals on the ordering of the object lists. The ordering and grid connectivity is reflected in the structure of the flux linearization matrices. It therefore has a direct impact on the performance of the code, both in the matrix assembly and solution phase [35, 76].

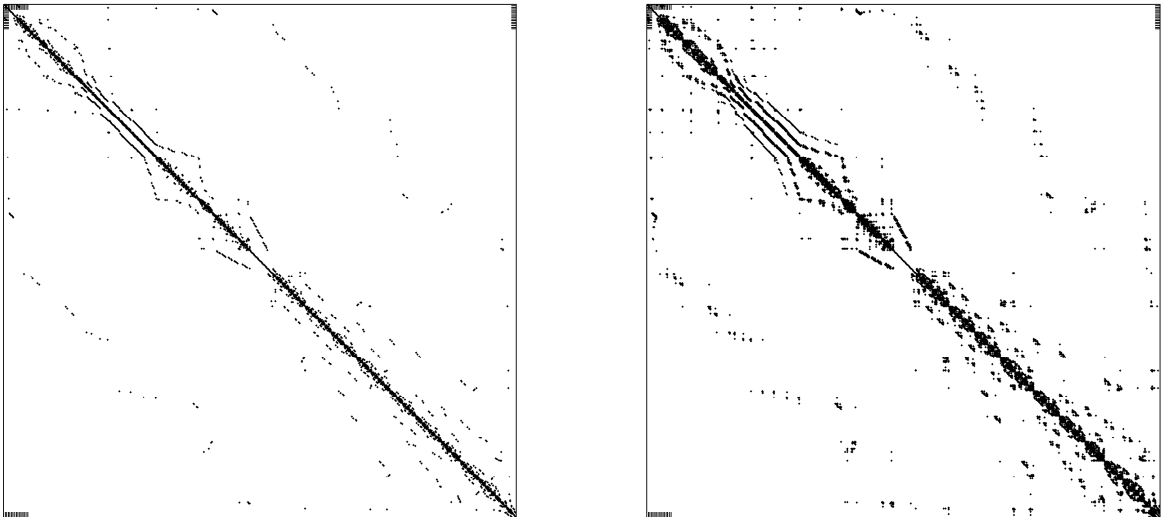
The key concern here is data locality: data from objects that are close together in the simulation domain should be located close together in memory to maximize CPU cache reuse when performing operations that loop over object lists. The strategy chosen in the present implementation places newly inserted child objects directly next to their parent objects. This maintains data locality to some degree. However, in unstructured grids non-locality unavoidably occurs. This effect can be visualized by plotting the structure of sparse matrices resulting from the different benchmark problems. Examples are shown in figures B.4, B.6, B.5 and B.7. The matrix structures for next-neighbor connectivity (equivalent to a five-point-stencil) and second-neighbor connectivity (equivalent to the stencil used for linear reconstruction) are shown.

A straightforward solution to increase locality is object reordering, an approach that has been discussed widely in the literature. The grid object lists are reordered to optimize their memory layout with respect to the most common access patterns. Standard packages are available for this task (some are used by the MUMPS solver). In the scope of this work no serial optimization of this form was done, but care was taken that the design decision do not inhibit later application of these techniques.



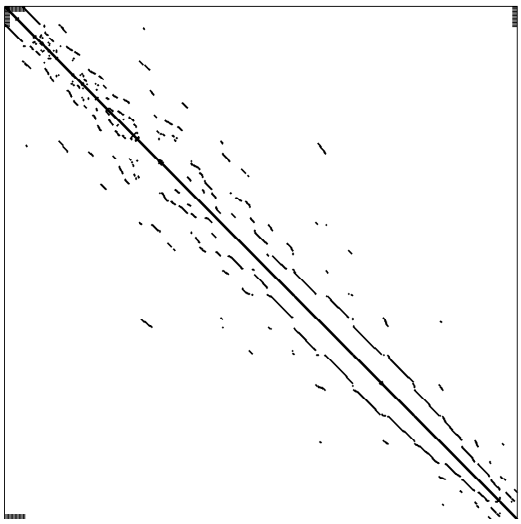
(a) Next neighbor connectivity (b) Second neighbor connectivity

Figure B.4.: Matrix structure of the grid obtained at the end of the time-dependent advection test with isotropic adaptation as shown in figure 5.20c.

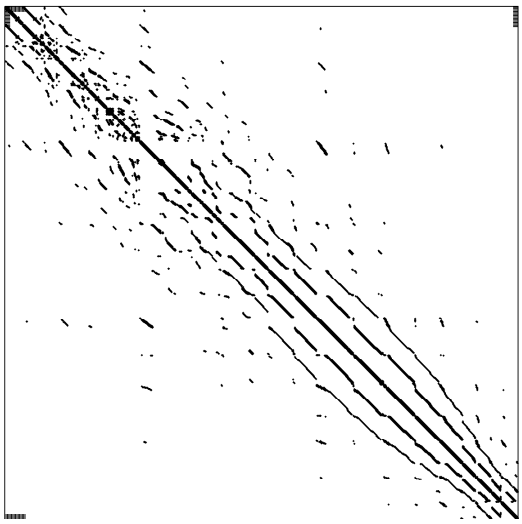


(a) Next neighbor connectivity (b) Second neighbor connectivity

Figure B.5.: Matrix structure of the grid obtained at the end of the time-dependent advection test with anisotropic adaptation as shown in figure 5.21c.

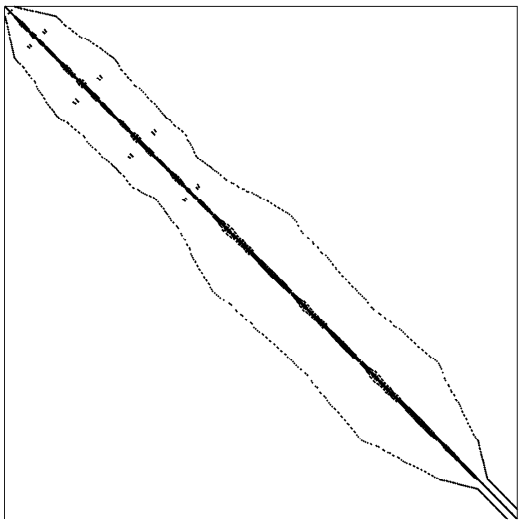


(a) Next neighbor connectivity

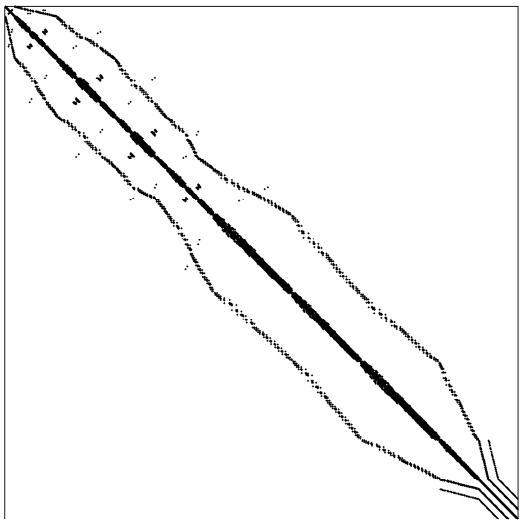


(b) Second neighbor connectivity

Figure B.6.: Matrix structure of the B2 grid obtained using anisotropic adaptation as shown in figure 3.28.



(a) Next neighbor connectivity



(b) Second neighbor connectivity

Figure B.7.: Matrix structure of the grid obtained in the B2.6-structured benchmark as shown in figure 7.4.

C. Benchmark details

C.1. Steady-state advection-diffusion benchmark details

In this section the analytical prescriptions of the quantities used for the solver verification in section 5.1 are stated. The exact solution considered is $\phi : \mathbb{R}^2 \to \mathbb{R}$,

$$\phi(x, y) = y \sin(2\pi x).$$

Substituting it into 4.5 with $\vec{u} = \left(\begin{smallmatrix} u_1 \\ u_2 \end{smallmatrix}\right), D = \left(\begin{smallmatrix} d_{11} & 0 \\ 0 & d_{22} \end{smallmatrix}\right)$ yields the source term

$$s(x, y) = 2u_1\pi y \cos(2\pi x) + u_2 \sin(2\pi x) + 4d_{11}\pi^2 y \sin(2\pi x).$$

Let $\Omega_i = \{(x, y)|x \in [x_1, x_2], y \in [y_1, y_2]\} \subset \mathbb{R}^2$ be a rectangular cell, $e_j^{(y)} = \{(x_0, y)|y \in [y_1, y_2]\} \subset \mathbb{R}^2$ a face aligned to the y-coordinate direction and $e_l^{(x)} = \{(x, y_0)|x \in [x_1, x_2]\} \subset \mathbb{R}^2$ a face aligned to the x-coordinate direction for given $x_{0,1,2}, y_{0,1,2} \in \mathbb{R}$. From this the quantities required for the solver verification can be derived.

Cell average of solution $\phi(x, y)$ in cell Ω_i:

$$\phi_i = \frac{1}{|\Omega_i|} \int_{y_1}^{y_2} \int_{x_1}^{x_2} f(x, y) dx dy = -\frac{\left(y_1^2 - y_2^2\right) \left(\cos(2\pi x_1) - \cos(2\pi x_2)\right)}{4\pi(x_2 - x_1)(y_2 - y_1)}$$

Source integral in cell Ω_i:

$$S_i = \int_{y_1}^{y_2} \int_{x_1}^{x_2} s(x, y) dx dy = \frac{1}{\pi}((y_1 - y_2) \sin(\pi(x_1 - x_2))(u_1\pi(y_1 + y_2) \cos(\pi(x_1 + x_2)) + \left(u_2 + 2d_1\pi^2(y_1 + y_2)\right) \sin(\pi(x_1 + x_2)))) \quad \text{(C.1)}$$

Integral of the advection-diffusion flux $\vec{f}_{ad}$ (eq. 4.4) over faces $e_j^{(y)}$ (flux in x-direction)

$$F_j^x = \int_{y_1}^{y_2} \vec{f}(x, y) \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} dy = d_1\pi \cos(2\pi x) y_1^2 - \frac{1}{2} u_1 \sin(2\pi x) y_1^2 - d_1\pi y_2^2 \cos(2\pi x) + \frac{1}{2} u_1 y_2^2 \sin(2\pi x) \quad \text{(C.2)}$$

and over face $e_k^{(x)}$ (flux in y-direction)

$$F_k^y = \int_{x_1}^{x_2} \vec{f}(x,y) \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} dy = -\frac{(d_2 - u_2 y)(\cos(2\pi x_1) - \cos(2\pi x_2))}{2\pi}. \tag{C.3}$$

C.2. B2.6-structured ASDEX Upgrade #16151 benchmark details

C.2.1. Grid resolution scan

To put the accuracy of the solutions obtained on the adapted grids in chapter 7 into perspective, a grid resolution scan is performed. A series of structured grids is created consisting of composite cells with stencil size $(d_x, d_y) = (x, y), x, y \in [1, 2, 4, 8]$, resulting in a total of 15 grids. To reduce errors resulting from cell $\rightarrow$ face interpolation close to the targets, the grid at the target plates and vacuum region boundary is kept at the highest resolution. This proved to be important for accuracy of the target profile plots. A representative mesh for this approach is shown in figure C.1.

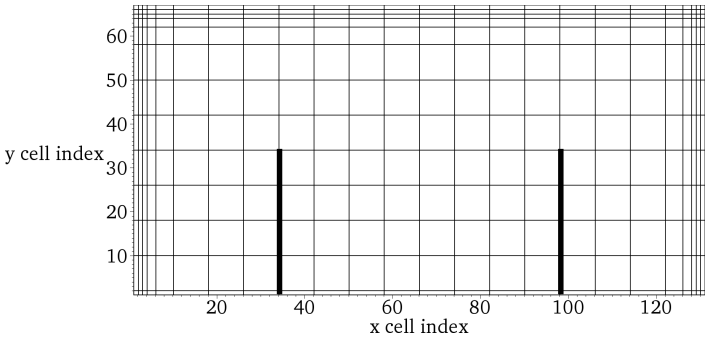
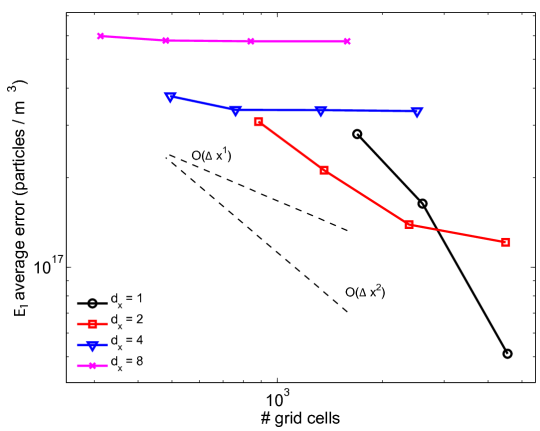


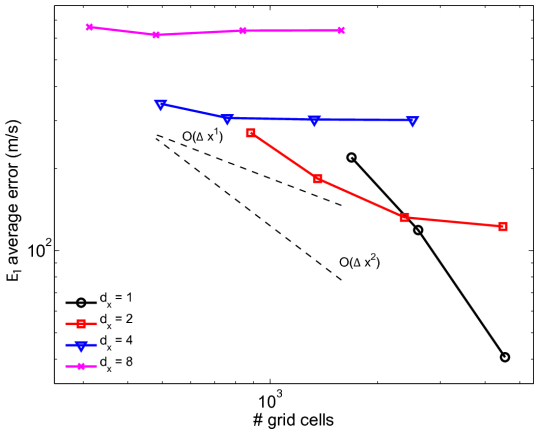
Figure C.1.: Example grid used in the resolution scan. The base grid consists of 128×64 cells. The working grid cells have a stencil of $(d_x, d_y) = (8, 8)$. At the west, east and north boundary the grid is forced to the highest resolution.

On every grid, the benchmark problem ... is solved to convergence. The following plots show the average absolute L_1 deviation (computed by equation (4.26)) with respect to the reference solution for the quantities $n_i, u_{\|i}$ for the ion species and T_i, T_e, ϕ. To simplify interpretation, the data is shown in three different ways:

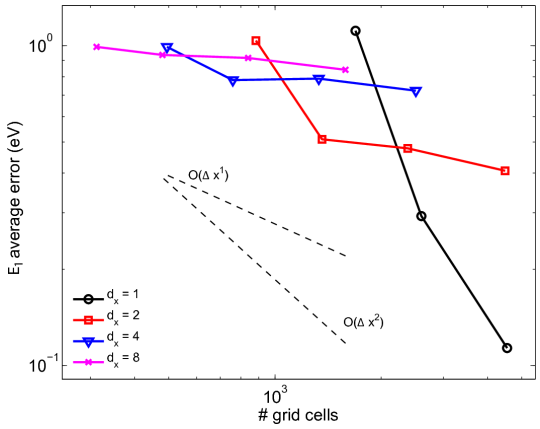
- **Fixed x resolution**: the lines connect points for grids with a fixed x-resolution $d_x =$ const, while the y-resolution $d_y \in [1, 2, 4, 8]$ is varied.
- **Fixed y resolution**: the lines connect points for grids with a fixed y-resolution $d_y =$ const, while the x-resolution $d_x \in [1, 2, 4, 8]$ is varied.
- **Fixed resolution ratio**: the lines connect points for grids with a fixed resolution relation $d_x/d_y =$ const.



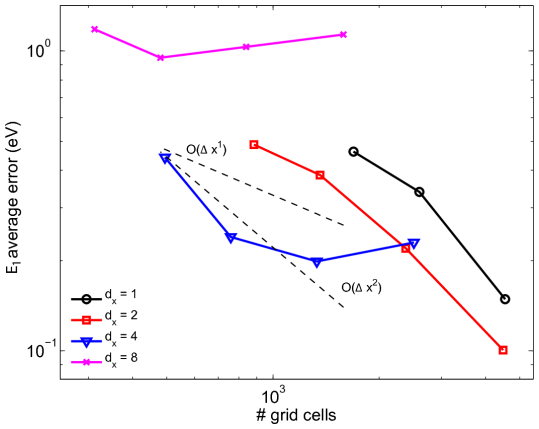
(a) Ion density n_i



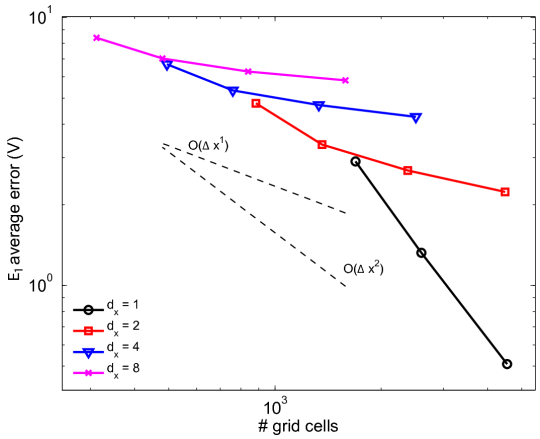
(b) Parallel ion velocity $u_{\parallel i}$



(c) Electron temperature T_e

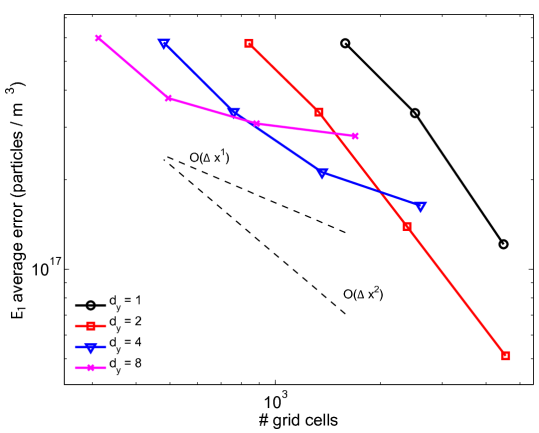


(d) Ion temperature T_i

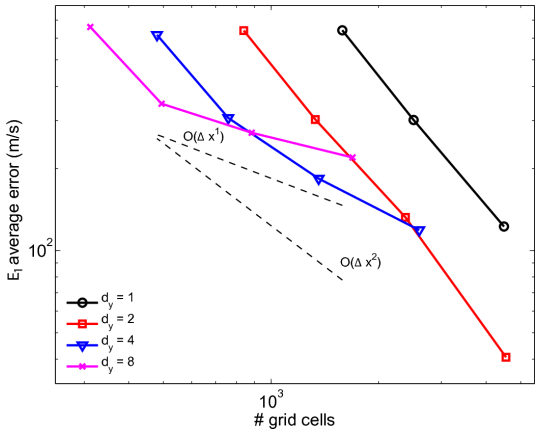


(e) Potential ϕ

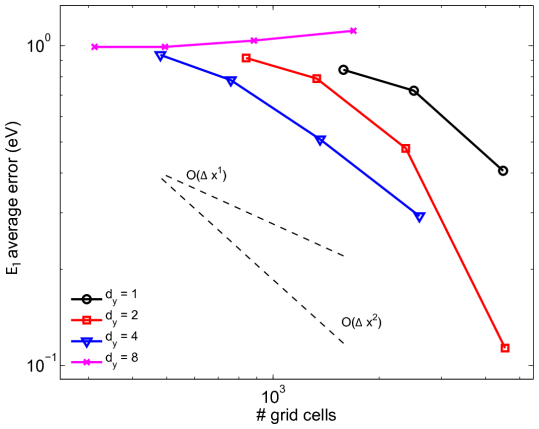
Figure C.2.: Absolute errors of main plasma quantities for the structured grid resolution scan on a log-log plot. Lines connect data points with fixed grid stencil size/resolution along poloidal/x direction. Data for the reference grid (stencil size $(1, 1)$) is not plotted.



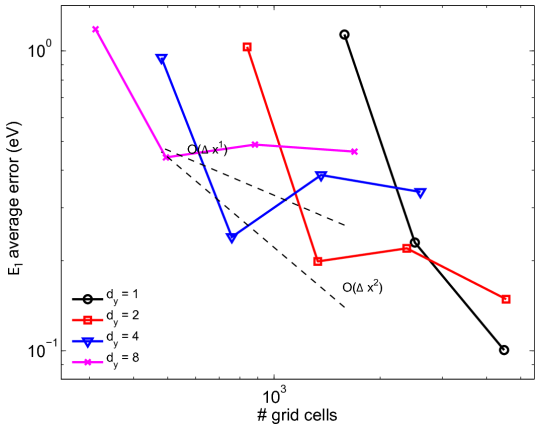
(a) Ion density n_i



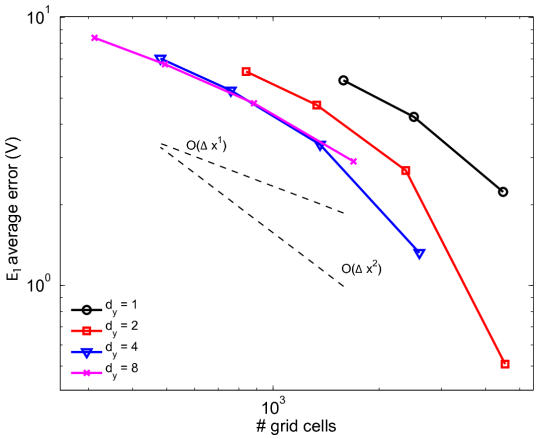
(b) Parallel ion velocity $u_{\parallel i}$



(c) Electron temperature T_e

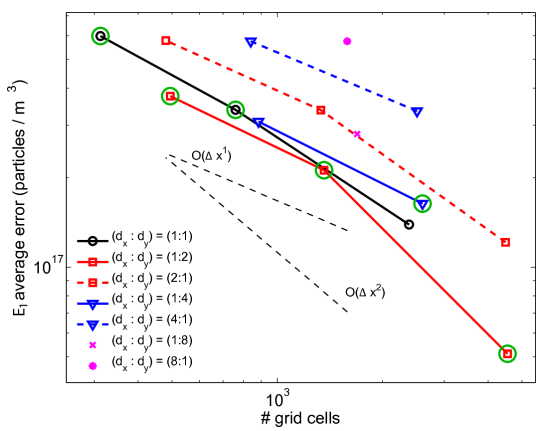


(d) Ion temperature T_i

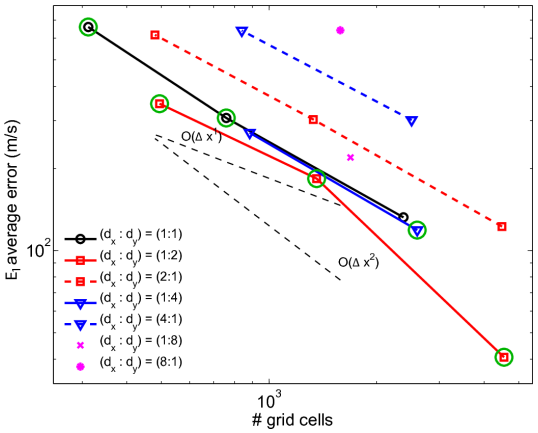


(e) Potential ϕ

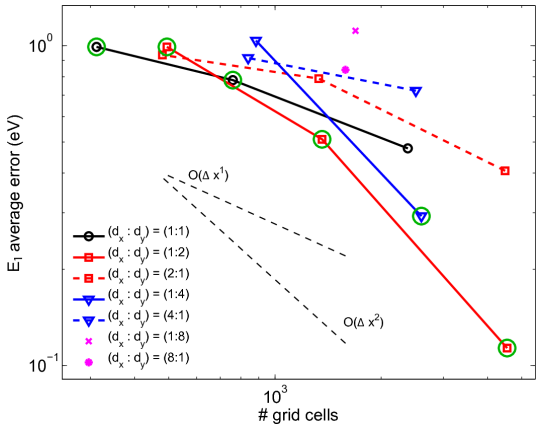
Figure C.3.: Absolute errors of main plasma quantities for the structured grid resolution scan on a log-log plot. Lines connect data points with fixed grid stencil size/resolution along poloidal/y direction. Data for the reference grid (stencil size $(1, 1)$) is not plotted.



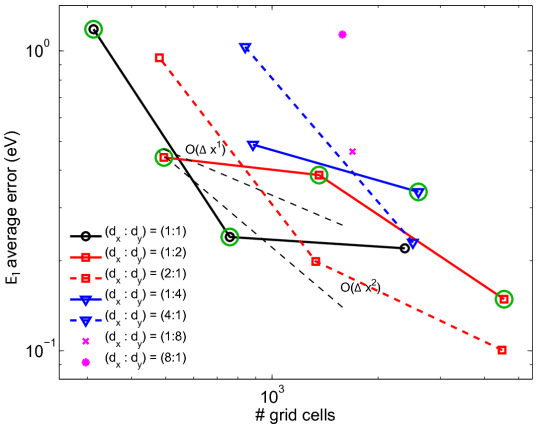
(a) Ion density n_i



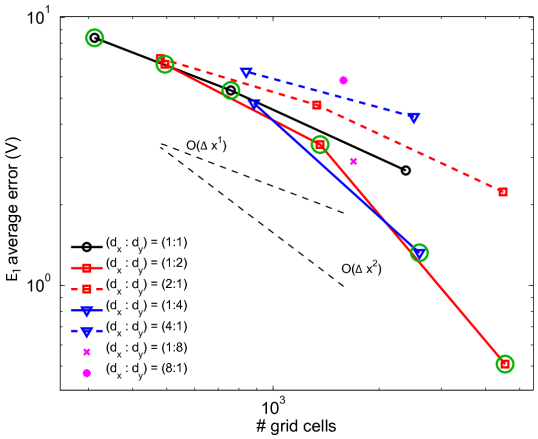
(b) Parallel ion velocity $u_{\parallel i}$



(c) Electron temperature T_e



(d) Ion temperature T_i



(e) Potential ϕ

Figure C.4.: Absolute errors of main plasma quantities for the structured grid resolution scan on a log-log plot. Lines connect data points with fixed grid stencil size relation. Points circled in green are included in the "best case" reference grid set used in chapter 7.

Discussion

The resolution scan gives insight what behavior can be expected from the code when solving the #16151 benchmark example on varying grid resolutions. The wide error variation observed for similar grid cell counts underlines the importance of a sensible grid resolution choice.

Figure C.2 shows that error reduction slows down significantly after the radial/y stencil size reaches $d_y = 4$. As shown in figure C.3, grids with this radial resolution show the best overall error reduction. On the other hand, for grids with fixed d_y (figure C.3) increasing the resolution in the poloidal direction leads to continuous reduction of the error. Obviously for this benchmark poloidal resolution is more important than radial resolution, and consequently grids with $d_x \leq d_y$ (represented with solid lines in figure C.4) perform best.

These statements hold for the quantities n_i, $u_{\parallel i}$, T_e and ϕ. Behavior of the T_i error deviates from the other quantities. For grids with $d_y > 1$, error reduction slows down significantly for $d_x < 4$. (for $d_y = 4$ and $d_x < 4$ the error even increases with increasing cell count). A relatively low error is already achieved on the rather coarse grid with stencil size $(4, 4)$.

The plots contain reference lines for convergence orders $\mathcal{O}(\Delta x)$ and $\mathcal{O}(\Delta x^2)$, with Δx being an assumed reference grid spacing for both radial and poloidal direction. When comparing error convergence rates to these reference values, it has to be kept in mind that in figures C.2 and C.3 the resolution is only varied in one coordinate direction along each line, resulting in a linear increase of the cell count. A direct comparison is possible for figure C.4, where along every line the resolution is simultaneously increased in both coordinate directions. Convergence towards the reference solution with first to second order is observed, meeting the expectation on the hybrid scheme. However, it has to be kept in mind that a) the reference solution is not exact and b) especially the low resolution grids might not be in the asymptotic convergence range.

As a reference "best case" set for comparison with the adaptation runs, the grids with stencils $(1, 2), (1, 4), (2, 4), (4, 4), (4, 8)$, and $(8, 8)$ are chosen. The corresponding points in figure C.4 are marked with green circles. This grid series is near-optimal for all quantities but T_i. The optimal grids for T_i are $(2, 1), (4, 2), (4, 4), (4, 8)$ and $(8, 8)$. However, to maintain a consistent set of data, the reference set is also used for T_i.

D. Notes on further development

D.1. Addressing grid non-orthogonality in B2.6

The problem of grid non-orthogonality close to the target plates as described in section 3.1.1 is ultimately caused by technical restrictions of the B2.5 grid data structure. Introduction of the UG data structure in B2.6 lifts these restrictions. It trivially supports varying numbers of grid cells in the radial or poloidal direction and allows a proper treatment of the nonorthogonality problem. A further benefit would be to allow extension of the B2 grid towards the plasma vessel wall.

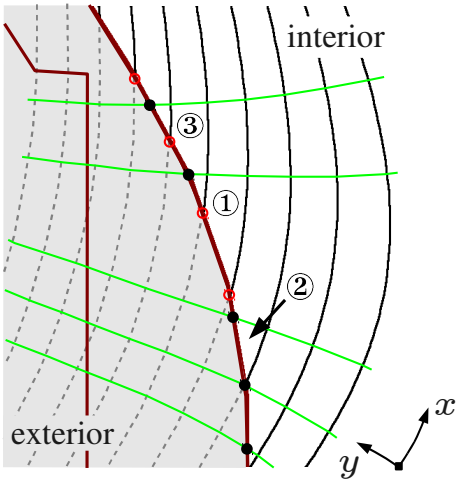


Figure D.1.: Possible cut-cell configurations for smooth boundaries

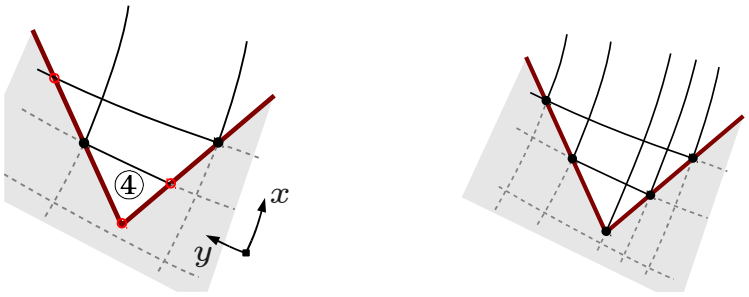
An approach established widely in the literature to combine Cartesian grids with complex geometries is the cut-cell approach [10, 9]. In the B2 context it can be applied by using the existing grid generators to create grids that are fully orthogonal, ignoring the bounding surfaces of the device geometry and the requirement to align grid faces to match them. This creates cells at the boundary which are "cut" by a material surface, resulting in a more complicated cell geometry.

Figure D.1 illustrates the situation for a smooth boundary, where cells are cut by only one boundary segment. Assuming a given spacing of grid lines in the y direction, placing grid points in the x direction without awareness for the boundary geometry leads to problematic grid cell shapes. Three cases have to be considered.

- Quadrilateral cells of type ① with a non-field-aligned boundary face can be treated exactly like in B2.5 by distinguishing between poloidal/parallel and wall contact area.
- Cells of type ② which degenerate into triangles can be handled straightforward in the UG data structures as quadrilaterals with no neighbor in one direction. However, the cell volume and face areas of these cells can become very small relative to the rest of the grid even for uncomplicated geometries, with negative effects on the numerical solver.
- Cells of type ③ have 5 faces, which is supported in B2.6-unstructured by treating the face incident to the boundary as an y-face (aligned to a y coordinate line). This is not possible in B2.5-structured.

Combining type ② and ③ cells into a composite type ① cell using grid adaptation is not possible, as in general they do not coarsen into a common parent cell. These problems can be avoided altogether by placing intersection points of x- and y-coordinate on the boundary (points marked as red circles in figure D.1). This prevents creation of type ③ cells and alleviates the problem of excessively small type ② cells. A grid constructed following this rule is shown in figure D.3. Only type ① and ② cells remain. The grid plot in computational space shows the ragged region of interior cells. Note that the existing base grid data structure can still be used, cells exterior to the simulation are stored but marked as inactive.

The situation is more complex for non-smooth boundary surfaces where grid cells can be cut by more than two boundary segments (figure D.2a). This results situations like cell ④ where the grid-spacing in the y-direction under-resolves the geometry. The resulting cells are very hard to fit into the UG data structure. This can be avoided by modifying the grid spacing to match features of the geometry (cf. figure D.2b). Note that additional x-coordinate lines are needed to match points where the y-coordinate lines intersect the targets a second time (this can be anticipated easily).



(a) Geometry not resolved

(b) Geometry properly resolved

Figure D.2.: Non-smooth boundary case. Geometry-aware grid point placement in the y coordinate direction is required to resolve the geometry.

D.1.1. Consequences for the grid generators

It is clear that while the cut-cell approach removes complexity from the grid generator (as there is no need anymore to relax the orthogonality constraint in parts of the domain), more care is required at choosing radial and poloidal grid spacings.

For geometries with smooth targets, selecting appropriate spacings in the poloidal direction is straightforward. Grid points at the target plates are given by the intersection of the chosen x-coordinate lines, but additional care has to be taken to ensure sufficient poloidal resolution. Note that inserting additional y-coordinate lines also requires insertion of additional x-coordinate lines. This can already be taken into account when selecting the radial grid spacing.

If more complex geometries with non-smooth target geometries are to be used an additional

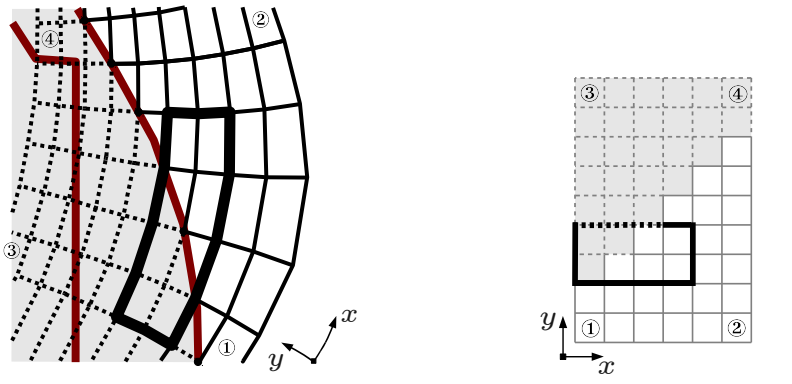
preprocessing step is required to identify features in the boundary and adjust the radial grid spacings to match them.

Overall, some algorithmic changes are required in the grid generators which should be localized to sections of the code that determine grid spacings. An additional advantage of such an extension is improved awareness of the local boundary geometry. What also has to be considered is that the grid generator output might have to be extended to contain additional information about a) the cut cell geometry and b) which cells fall outside the simulation domain.

D.1.2. Consequences for the adaptation algorithm

Dropping the restriction that base grid blocks have to be of rectangular shape in the computational domain requires revisiting the UG data structure and adaptation algorithm implementation, which is restricted to logically rectangular composite cells due to the cell stencil definition (cf. section 3.2.1). While coarsening to logically rectangular grid cells is possible to some degree even in the presence of a ragged boundary in computational space, it still is very constrictive. In some regions the grid resolution is effectively pinned to the highest possible (base grid) resolution, which is annoying due to the global effects for B2.6-structured and prohibitive for the implementation of multi-grid type algorithms.

This problem can be overcome while maintaining the general design by allowing stencils to include inactive cells outside the computational domain, as demonstrated in the composite cell indicated in figure D.3. The main change required in the algorithms to support this is to enable the stencils of cut cells to grow by adding inactive cells and to include special treatment when deriving the geometry approximation of composite cut cells. These extensions do not pose serious problems.



(a) Physical space (b) Computational space

Figure D.3.: A composite cut cell at the boundary in physical and computational space. The stencil includes both active interior (white) and inactive exterior (gray) cells. Cut cells are considered interior. Fluxes over dotted faces are not discretized.

Another technical point here is the construction of ghost cells. At the moment, ghost cells are generated by the grid generator and passed to B2 , where (except in the adaptation algorithms) they are generally treated like normal cells. Likewise, the concept of composite cut cells can be extended to ghost cells. However, as the actual geometry of the ghost cells does not matter by definition, it might be easier to generate them inside B2.6 instead of relying on the grid generator to do this.

D.2. Notes on parallelization for B2

Due to the relatively low grid resolutions used in typical simulations (common grids are in the range of 10^5 cells), standard approaches for parallelization of partial-differential equation solvers like domain decomposition or parallelization of linear solvers are not feasible.

Due to the strong coupling between the equations there is also no obvious way how to parallelize the iterative solution algorithm (cf. section 6.1) itself. However, what can be done rather easily is a partial parallelization by splitting the the continuity and momentum equation blocks.. Inside these blocks the equations for the individual species can be solved completely independent. This was verified by implementing deferred updates for these blocks in the serial code. While this approach obviously will not result in good scaling, simulations with a large number of species should still benefit significantly.

Considering the structure of the code, a sensible approach is to use OpenMP for the implementation and aim for typical cluster compute nodes with 8-16 cores. The main immediate challenge is to find a suitable linear solver that can be used in such a setup. For further work the task parallelism features recently added in the OpenMP v3.0 standard could be exploited to try a further parallelization of the iterations.

Bibliography

[1] Kernfusion - Berichte aus der Forschung. Technical report, Max-Planck Institut für Plasmaphysik, Garching, 2002.

[2] Summer University for Plasma Physics (Lecture notes). Technical report, Max-Planck Institut für Plasmaphysik, 2004.

[3] MUMPS website. http://mumps.enseeiht.fr/, 2008.

[4] Numerical Algorithms Group Website. http://www.nag.co.uk/, 2008.

[5] EFDA Integrated Tokamak Modeling Task Force (ITM-TF) Website. https://portal.efda-itm.eu/portal/, 2009.

[6] Official ITER Website. http://www.iter.org/, 2009.

[7] M. Aftosmis. Solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries. *VKI Lecture Series*, 2:1997, 1997.

[8] M. Aftosmis and M. Berger. Multilevel error estimation and adaptive h-refinement for cartesian meshes with embedded boundaries. *AIAA paper*, (2002–863), 2002.

[9] M. Aftosmis, M. Berger, and G. Adomavicius. A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries. *AIAA paper*, (2000–808), 2000.

[10] M. Aftosmis, M. Berger, and J.E. Melton. Robust and efficient Cartesian mesh generation for component-based geometry. *AIAA Journal*, 36(6):952–960, 1998.

[11] M. Aftosmis, D. Gaitonde, and T.S. Tavares. The Behavior of Linear Reconstruction Techniques on Unstructured Meshes. Technical Report AD-A282 951, Wright Laboratory, 1994.

[12] M. Aftosmis, D. Gaitonde, T.S. Tavares, et al. Behavior of linear reconstruction techniques on unstructured meshes. *AIAA Journal*, 33(11):2038–2049, 1995.

[13] P.R. Amestoy, I.S. Duff, J.Y. L'Excellent, and J. Koster. A fully asynchronous multi-

frontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2002.

[14] D.N. Arnold, F. Brezzi, B. Cockburn, and L.D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[15] ASDEX Upgrade Team. ASDEX Upgrade Machine Documentation. https://www.aug.ipp.mpg.de/aug/local/aug_only/AUG_Aufbau/Drawing_Gallery/, 2007.

[16] M. Baelmans. *Code improvements and applications of a two-dimensional edge plasma model for toroidal devices.* PhD thesis, Katholieke Universiteit Leuven, 1994.

[17] R. Balescu. *Transport processes in plasmas.* North-Holland, 1988.

[18] T.J. Barth. Numerical methods and error estimation for conservation laws on structured and unstructured meshes. *Lecture notes, von Karman Institute for Fluid Dynamics, Series*, 4, 2003.

[19] T.J. Barth and D.C. Jespersen. The design and application of upwind schemes on unstructured meshes. *AIAA paper*, (1989–0366), 1989.

[20] T.J. Barth and M. Ohlberger. Finite volume methods: foundation and analysis. *Encyclopedia of computational mechanics*, 1:439–470, 2004.

[21] O. Batishchev, A. Batishcheva, and A.S. Kholodov. Unstructured adaptive grid and grid-free methods for edge plasma fluid simulations. *Journal of Plasma Physics*, 61(05):701–722, 1999.

[22] M. Berger and M. Aftosmis. Aspects (and aspect ratios) of cartesian mesh methods. *Lecture notes in Physics*, pages 1–12, 1998.

[23] M. Berger, M. Aftosmis, and S.M. Murman. Analysis of slope limiters on irregular grids. *AIAA Paper*, (2005–490), 2005.

[24] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[25] E. Bertolazzi and G. Manzini. A unified treatment of boundary conditions in least-square based finite-volume methods. *Computers & Mathematics with Applications*, 49(11-12):1755–1765, 2005.

[26] C. Bolley and M. Crouzeix. Conservation de la positivité lors de la discrétisation des problemes dévolution paraboliques. *RAIRO Analyse numérique*, 12(3):237–245, 1978.

[27] X. Bonnin, H. Bürbaumer, R. Schneider, D.P. Coster, F. Aumayr, and H.P. Winter. The two-mesh grid refinement method for the B2 code. *Contributions to Plasma Physics*, 42(2-4):175–180, 2002.

[28] X. Bonnin, D.P. Coster, M. Warrier, A. Rai, and R. Schneider. Integrated modelling of plasma-wall interactions in tokamaks with B2.5: mixed materials, layers and coatings, bundled charge states, and hydrogen inventory. In *Europhysics Conference Abstracts (CD-ROM, Proc. of the 34th EPS Conference on Plasma Physics, Warsaw, 2007)*, Geneva, 2007. EPS.

[29] X. Bonnin, A. Kukushkin, and D.P. Coster. Code development for ITER edge modelling-SOLPS5. 1. *Journal of Nuclear Materials*, 390:274–277, 2009.

[30] M. Borchardt, J. Riemann, R. Schneider, and X. Bonnin. W7-X edge modelling with the 3D SOL fluid code BoRiS. *Journal of Nuclear Materials*, 290:546–550, 2001.

[31] B.J. Braams. *Computational studies in tokamak equilibrium and transport.* PhD thesis, Rijksuniversiteit, Utrecht, Nederland, 1986.

[32] B.J. Braams. A multi-fluid code for simulation of the edge plasma in tokamaks. Technical report, 1987.

[33] B.J. Braams. Radiative Divertor Modelling for ITER and TPX. *Contributions to Plasma Physics*, 36:276–281, 1996.

[34] S. Braginskii. Transport processes in a plasma. *Reviews of Plasma Physics*, 1:205, 1965.

[35] D. Burgess and M.B. Giles. Renumbering unstructured grids to improve the performance of codes on hierarchical memory machines. *Advances in Engineering Software*, 28(3):189–201, 1997.

[36] D. Calhoun and R.J. LeVeque. An accuracy study of mesh refinement on mapped grids. *Adaptive Mesh Refinement-Theory and Applications*, pages 91–101, 2005.

[37] C. Carstensen, R. Lazarov, and S. Tomov. Explicit and averaging a posteriori error estimates for adaptive finite volume methods. *SIAM Journal on Numerical Analysis*, 42(6):2496–2521, 2005.

[38] A.V. Chankin. Development of a kinetic module for parallel transport for solps. Talk at IPP Theory Meeting 2009, Sellin, November 2009.

[39] A.V. Chankin, DP Coster, R. Dux, C. Fuchs, G. Haas, A. Herrmann, LD Horton, A. Kallenbach, M. Kaufmann, C. Konz, et al. SOLPS modelling of ASDEX upgrade H-mode plasma. *Plasma physics and controlled fusion*, 48:839–868, 2006.

[40] A.V. Chankin, D.P. Coster, R. Dux, C. Fuchs, G. Haas, A. Herrmann, L.D. Horton,

A. Kallenbach, M. Kaufmann, C. Konz, et al. Critical issues identified by the comparison between experiment and SOLPS modelling on ASDEX Upgrade. In *Proc. 21st Int. Conf on Fusion Energy 2006*. IAEA, 2007.

[41] B. Cockburn. Discontinuous Galerkin methods. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 83(11):731–754, 2003.

[42] W.J. Coirier. *An Adaptively-Refined, Cartesian, Cell-Based Scheme for the Euler and Navier-Stokes Equations*. PhD thesis, The University of Michigan, 1994.

[43] D.P. Coster. *Tokamak divertor modeling with fluid and kinetic codes*. PhD thesis, Princeton University, 1993.

[44] D.P. Coster. Whole device ELM simulations. *Journal of Nuclear Materials*, 390:826–829, 2009.

[45] D.P. Coster, X. Bonnin, G. Corrigan, G.S. Kirnev, G. Matthews, J. Spence, et al. Benchmarking Tokamak edge modelling codes. *Journal of Nuclear Materials*, 337:366–370, 2005.

[46] D.P. Coster, X. Bonnin, and M. Warrier. Extensions to the SOLPS edge plasma simulation code to include additional surface interaction posibilities. *Physica Scripta*, 124:9–12, 2006.

[47] D.P. Coster et al. Solps manual. Technical report, Max-Planck Institut für Plasmaphysik, Garching, 2009.

[48] W. Crowley. Numerical advection experiments. *Monthly Weather Review*, 96(1):1–11, 1968.

[49] W.D. D'haeseleer, W.N.G. Hitchon, J.D. Callen, and J.L. Shohet. *Flux coordinates and magnetic field structure*. Springer, 1991.

[50] L.C. Evans. *Partial Differential Equations*. Graduate Studies in Mathematics. American Mathematical Society, 2008.

[51] R. Eymard, T. Gallouët, and R. Herbin. Finite volume methods. *Handbook of Numerical Analysis*, 7:713–1018, 2000.

[52] Y. Feng, F. Sardei, J. Kisslinger, and P. Grigull. A 3D Monte Carlo code for plasma transport in island divertors. *Journal of Nuclear Materials*, 241:930–934, 1997.

[53] J.H. Ferziger and Peric. *Computational methods for fluid dynamics*. Springer, 2002.

[54] R. P. Feynman, R.B. Leighton, and M. Sands. *The Feynman Lectures on Physics, Vol. 2.* Addison-Wesley, Redwood City, CA, 1989.

[55] L. Formaggia, S. Micheletti, and S. Perotto. Anisotropic mesh adaptation in computational fluid dynamics: Application to the advection-diffusion-reaction and the Stokes problems* 1. *Applied Numerical Mathematics*, 51(4):511–533, 2004.

[56] S.K. Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306, 1959.

[57] J.B. Goodman and R.J. LeVeque. On the accuracy of stable schemes for 2d scalar conservation laws. *Mathematics of computation*, 45(171):15–21, 1985.

[58] O. Gruber, A.C.C. Sips, R. Dux, T. Eich, C. Fuchs, A. Herrmann, A. Kallenbach, C.F. Maggi, R. Neu, T. Pütterich, et al. Compatibility of ITER scenarios with full tungsten wall in ASDEX Upgrade. *Nuclear Fusion*, 49:115014, 2009.

[59] W. Hackbusch. *Multi-grid methods and applications.* Springer, 1985.

[60] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations I: nonstiff problems.* Springer, 1993.

[61] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving ordinary differential equations: Stiff and differential-algebraic problems.* Springer, 1993.

[62] R. Herbin and M. Ohlberger. A posteriori error estimate for finite volume approximations of convection diffusion problems. In *Proc. 3rd Int. Symp. on Finite Volumes for Complex Applications-Problems and Perspectives*, pages 753–760, 2002.

[63] A. Herrmann and O. Gruber. ASDEX Upgrade - introduction and overview. *Fusion Science and Technology*, 44(3):569–577, 2003.

[64] W. Hundsdorfer. Partially implicit bdf2 blends for convection dominated flows. *SIAM Journal on Numerical Analysis*, pages 1763–1783, 2001.

[65] W. Hundsdorfer and J.G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations.* Springer, 2003.

[66] K. Ikeda. Progress in the ITER physics basis. *Nuclear Fusion*, 47, 2007.

[67] M. Kaufmann. *Plasmaphysik und Fusionsforschung.* Vieweg+ Teubner Verlag, 2003.

[68] J.W. Kim. *An analysis of the anomalous transport of the plasma edge in ASDEX Upgrade.* PhD thesis, Technische Universität München, 2002.

[69] B. Koren. Multigrid and defect correction for the steady Navier-Stokes equations. *Journal of Computational Physics*, 87(1):25–46, 1990.

[70] V. Kotov, D. Reiter, and A. Kukushkin. Numerical study of the ITER divertor plasma with the B2-EIRENE code package. Technical Report Jül-4257, Forschungszentrum Jülich, 2007.

[71] A. Kukushkin and H.D. Pacher. Divertor Modelling. *Plasma Physics and Controlled Fusion*, 44:931–943, 2002.

[72] A. Kukushkin, H.D. Pacher, V. Kotov, D. Reiter, D. Coster, and G.W. Pacher. Effect of conditions for gas recirculation on divertor operation in ITER. *Nuclear fusion*, 47:698–705, 2007.

[73] B.P. Leonard and J. Drummond. Why you should not use'Hybrid','Power-Law' or related exponential schemes for convective modelling: there are much better alternatives. *International journal for numerical methods in fluids*, 20(6):421–442, 1995.

[74] R.J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press, 2002.

[75] V.D. Liseikin. *Grid generation methods*. Springer, 1999.

[76] R. Löhner. Some useful renumbering strategies for unstructured grids. *International Journal for Numerical Methods in Engineering*, 36(19):3259–3270, 2005.

[77] D. Maisonnier. European DEMO design and maintenance strategy. *Fusion Engineering and Design*, 83(7-9):858–864, 2008.

[78] R. Marchand and M. Dumberry. Carre: a quasi-orthogonal mesh generator for 2d edge plasma modelling. *Computer Physics Communications*, 96(2-3):232–246, 1996.

[79] P.J. McCarthy, P. Martin, and W. Schneider. The cliste interpretive equilibrium code. Technical report, Max-Planck Institut für Plasmaphysik, Garching, 1999.

[80] N. McTaggart, R. Zagórski, X. Bonnin, A. Runov, R. Schneider, T. Kaiser, T. Rognlien, and M. Umansky. 3D edge energy transport in stellarator configurations. *Journal of Nuclear Materials*, 337:221–226, 2005.

[81] E.A. Meese. SMLIB v1.1 - A Fortran 90 Library for Sparse Matrix Calculations. Technical report, Department of Applied Mechanics, Norwegian University of Science and Technology, Trondheim, 1998.

[82] K. Michalak and C.F. Ollivier-Gooch. Differentiability of slope limiters on unstructured grids. In *Proceedings of the Fourteenth Annual Conference of the Computational Fluid Dynamics Society of Canada, CFD Society of Canada*, 2006.

[83] K. Michalak and C.F. Ollivier-Gooch. Limiters for unstructured higherorder accurate solutions of the euler equations. In *Proceedings of the AIAA Forty-Sixth Aerospace Sciences Meeting*, 2008.

[84] E. Morano, D.J. Mavriplis, and V. Venkatakrishnan. Coarsening strategies for unstructured multigrid techniques with application to anisotropic problems. *SIAM Journal of Scientific Computing*, 20(2):393–415, 1998.

[85] M. Nemec, M. Aftosmis, and M. Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. *AIAA Paper*, (2008–725), 2008.

[86] R. Neu, R. Dux, A. Kallenbach, T. Pütterich, M. Balden, C. Fuchs, A. Herrmann, C.F. Maggi, M. O'mullane, R. Pugno, et al. Tungsten: an option for divertor and main chamber plasma facing components in future fusion devices. *Nuclear Fusion*, 45(3):209–218, 2005.

[87] C.F. Ollivier-Gooch. Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction. *Journal of Computational Physics*, 133(1):6–17, 1997.

[88] C.F. Ollivier-Gooch and M. Van Altena. A high-order-accurate unstructured mesh finite-volume scheme for the advection-diffusion equation. *Journal of Computational Physics*, 181(2):729–752, 2002.

[89] J. Pamela and J. Ongena. Overview of JET results. *Nuclear Fusion*, 45:S63–S85, 2005.

[90] S.V. Patankar. *Numerical heat transfer and fluid flow*. Hemisphere Pub, 1980.

[91] V. Pereyra. On improving an approximate solution of a functional equation by deferred corrections. *Numerische Mathematik*, 8(4):376–391, 1966.

[92] FW Perkins, P. Barabaschi, G. Bosia, et al. ITER physics basis. *Nuclear Fusion*, 39:12, 1999.

[93] M.H. Protter and H.F. Weinberger. *Maximum principles in differential equations*. Prentice-Hall, 1984.

[94] G.J. Radford, A.V. Chankin, G. Corrigan, R. Simonini, J. Spence, and A. Taroni. The particle and heat drift fluxes and their implementation into the EDGE2D transport code. *Contributions to Plasma Physics*, 36(2-3):187–191, 2006.

[95] D. Reiter. The EIRENE Code User Manual. Technical report, Forschungszentrum Jülich, November 2005.

[96] D. Reiter, M. Baelmans, and P. Börner. The EIRENE and B2-EIRENE codes. *Fusion Science and Technology*, 47(2):172–186, 2005.

[97] C.M. Rhie and W. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21:1525–1532, 1983.

[98] T. Rognlien, J.L. Milovich, M.E. Rensink, and G.D. Porter. A fully implicit, time dependent 2-D fluid code for modeling tokamak edge plasmas. *Journal of Nuclear Materials*, 196:347–351, 1992.

[99] V. Rozhansky, E. Kaveeva, P. Molchanov, I. Veselova, S. Voskoboynikov, D.P. Coster, G. Counsell, A. Kirk, and S. Lisgo. New B2-SOLPS5.2 transport code for H-mode regimes in tokamaks. *Nuclear Fusion*, 49:025007, 2009.

[100] V. Rozhansky, S. Voskoboynikov, E. Kaveeva, D.P. Coster, and R. Schneider. Simulation of tokamak edge plasma including self-consistent electric fields. *Nuclear Fusion*, 41:387–401, 2001.

[101] R. Schneider. Plasma edge physics for tokamaks. Technical report, Max-Planck Institut für Plasmaphysik, Garching, February 2001.

[102] R. Schneider, X. Bonnin, et al. Plasma Edge Physics with B2-Eirene. *Contributions to Plasma Physics*, 46(1-2):3–191, 2006.

[103] R. Schneider, D.P. Coster, B.J. Braams, P. Xantopoulos, V. Rozhansky, S. Voskoboynikov, L. Kovaltsova, and H. Bürbaumer. B 2-solps 5.0: Sol transport code with drifts and currents. *Contributions to Plasma Physics*, 40(3-4):328–333, 2000.

[104] T. Sonar. *Mehrdimensionale ENO-Verfahren.* Teubner Stuttgart, 1997.

[105] S. Spekreijse. Multigrid solution of monotone second-order discretizations of hyperbolic conservation laws. *Mathematics of Computation*, 49(179):135–155, 1987.

[106] P.C. Stangeby et al. *The plasma boundary of magnetic fusion devices.* Institute of Physics Publishing, 1999.

[107] H.J. Stetter. The defect correction principle and discretization methods. *Numerische Mathematik*, 29(4):425–443, 1978.

[108] F. Subba, X. Bonnin, D.P. Coster, and R. Zanino. 2D fluid modeling of the ASDEX upgrade scrape-off layer up to the first wall. *Computer Physics Communications*, 2008.

[109] F. Subba and R. Zanino. A 2d fluid model of the scrape-off layer (sol) using adaptive unstructured finite volumes. *Journal of Nuclear Materials*, 290:743–747, 2001.

[110] M. Sun and K. Takayama. Error localization in solution-adaptive grid methods. *Journal of Computational Physics*, 190:346–350, 2003.

[111] S. Tomov. *Adaptive methods for finite volume approximations*. PhD thesis, Texas A&M University, 2002.

[112] E.F. Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer, 2009.

[113] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[114] B. Van Leer. Towards the ultimate conservative difference scheme i. the quest of monotonicity. *Lecture notes in Physics*, 18(163-168):7, 1973.

[115] D.A. Venditti and D.L. Darmofal. Anisotropic grid adaptation for functional outputs: application to two-dimensional viscous flows. *Journal of Computational Physics*, 187(1):22–46, 2003.

[116] V. Venkatakrishnan. On the accuracy of limiters and convergence to steady state solutions. *AIAA paper*, (1993–0880), 1993.

[117] V. Venkatakrishnan. Convergence to steady state solutions of the Euler equations on unstructured grids with limiters. *Journal of Computational Physics*, 118(1):120–130, 1995.

[118] G.P. Warren, W.K. Anderson, J.L. Thomas, and S.L. Krist. Grid convergence for adaptive methods. *Numerical methods for fluid dynamics 4*, page 317, 1993.

[119] P. Wesseling. *Principles of Computational Fluid Dynamics*. Springer, 2001.

[120] J. Wesson and D. Campbell. *Tokamaks*. Oxford University Press, USA, 3rd edition, 2004.

[121] M. Wischmeier, M. Groth, A. Kallenbach, A.V. Chankin, D.P. Coster, R. Dux, A. Herrmann, H. W. Müller, R. Pugno, D. Reiter, A. Scarabosio, J. G. Watkins, DIII-D team, and ASDEX Upgrade Team. Current understanding of divertor detachment: Experiments and modelling. *Journal of Nuclear Materials*, 390–391:250–254, 2009.

[122] R. Zanino and F. Subba. Adaptive Anisotropic Finite Elements for Edge Plasma Modeling. *Contributions to Plasma Physics*, 38(1-2):355–360, 2006.

Acknowledgments

I would like to thank everybody who supported me in my work on this dissertation:

Dr. David Coster for offering this challenging task and his continued support and encouragement. With his profound knowledge of physics and computer science he made sure I didn't get lost in this vast field.

Prof. Dr. Peter Rentrop for his substantial expertise and excellent mentoring.

Dr. Xavier Bonnin for helping decipher the secrets of B2, restoring my hope and hosting my visits to the LIMHP Université Paris 13.

My colleagues at the Lehrstuhl für Numerische Mathematik M2 (TU München) for making me feel welcome.

My colleagues at the Max-Planck-Institut für Plasmaphysik for making the institute a great place to work.

My family for their continued support.

And Anna for being there, all the time.