



Dissertation

A Class of General Splitting Methods for Air Pollution Models: Theory and Practical Aspects

Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät II
Institut für Mathematik

von Martin Schlegel
geboren am 15.07.1981
in Halle (Saale)

Leipzig, 31.5.2011

Prepared at

LEIBNIZ INSTITUTE
FOR TROPOSPHERIC RESEARCH

Modeling Departement



Institute for Tropospheric Research
Permoserstraße 53
04318 Leipzig
Germany

Funded by

Deutsche
Forschungsgemeinschaft

DFG

*Simplicity is a great virtue but it requires hard work to achieve it
and education to appreciate it.
And to make matters worse: complexity sells better.*

Edsger W. Dijkstra

Contents

1	Introduction	3
1.1	Motivation	3
1.2	State of the art	6
2	Discretization in space and time	9
2.1	Advection-diffusion-reaction equations	9
2.2	Spatial discretizations	10
2.2.1	The method of lines approach	10
2.2.2	Flux limiting	16
2.2.3	Extension to higher dimensions	18
2.3	Runge–Kutta methods	21
2.3.1	Partitioned Runge–Kutta methods	24
3	A generic splitting scheme	29
3.1	Recursive Flux Splitting Multirate	29
3.1.1	General splitting	29
3.1.2	Construction of PRKs	31
3.1.3	Suitability of base methods	38
3.2	Spatial aspects of the splitting	42
3.2.1	Decomposition of the advection operator	42
3.2.2	Decompositions of advection-reaction systems	43
3.3	Order conditions	48
3.3.1	Order conditions up to order 2	49
3.3.2	Order conditions for order 3	50
3.3.3	Order conditions for recursive application	54
3.3.4	Order conditions for arbitrary inner methods	56
3.4	Extension to a mass preserving PRK	57
3.5	Stability for advection-reaction systems	62
4	Implementation issues	71
4.1	The multi scale atmospheric transport model	71
4.2	Data organization	71
4.3	Recursive implementation	74

4.4	Block coupling	77
4.4.1	Coupling of directly adjacent blocks	78
4.4.2	Coupling of diagonally adjacent blocks	81
4.4.3	Exchanges for a parallel implementation	88
4.5	Balancing	89
5	Numerical Tests	93
5.1	Comparison of splitting approaches	93
5.1.1	Setup	93
5.1.2	Results	93
5.1.3	Conclusions	97
5.2	Advection-reaction	97
5.2.1	Setup	97
5.2.2	Results	98
5.2.3	Conclusions	99
5.3	Fire spread	100
5.3.1	Setup and general properties	100
5.3.2	Discretization and implementation	100
5.3.3	Results	102
5.3.4	Conclusions	104
5.4	MUSCAT test cases	104
5.4.1	Academic test case	104
5.4.2	Realistic test case	106
6	Conclusions and outlook	111

Chapter 1

Introduction

1.1 Motivation

Numerical models can be employed to simulate a broad spectrum of physical phenomena, ranging from electronic circuits through dynamics of galaxies. Atmospheric models are of interest not only for the scientific community but also for the public by providing weather forecasts and climate models. We are concerned mainly with air pollution models which describe transport and chemical reactions of pollutants in the atmosphere. The latter can provide the public with air quality forecasts and thus help avoid exposure if high pollution levels are expected. Furthermore air pollution modeling may in the long run also help to improve weather models as the aerosol population strongly influences condensation processes in the atmosphere [16].

Chemical reactions are modeled as ordinary differential equations (ODE) in time. The transport processes are formulated as partial differential equations (PDEs) in space and time. Special challenges arise when these equations are to be solved numerically. Following the so called *method of lines* approach the model equations are first discretized in space resulting in a set of ODEs in time. Complexity in the spatial domain can simply be reduced by a coarser grid. The obvious downside of this approach is the loss of detail. Furthermore processes with a spatial scale below the cell size cannot be taken into account directly and must be approximated; a typical example for this in atmospheric models is turbulent mixing approximated by a diffusion operator. As a compromise the grid may be refined in regions of interest [6], [9]. Additionally in atmospheric models the vertical resolution generally is finer than the horizontal resolution in order to take important processes as convection and the dependence of the horizontal wind speed from height over ground into account.

The modeled physical phenomena exhibit a large spectrum of temporal and spatial scales (see Figure 1.1) and different characteristics. Diffusion and chemical reactions typically are *stiff*, whereas advection is non-stiff. Explicit time integration methods proved efficient for certain classes of equations as for instance the advection equation, whereas they are not suitable for stiff systems. Implicit methods on the other hand allow for larger time steps even for stiff systems, but they require repeated solving of a nonlinear

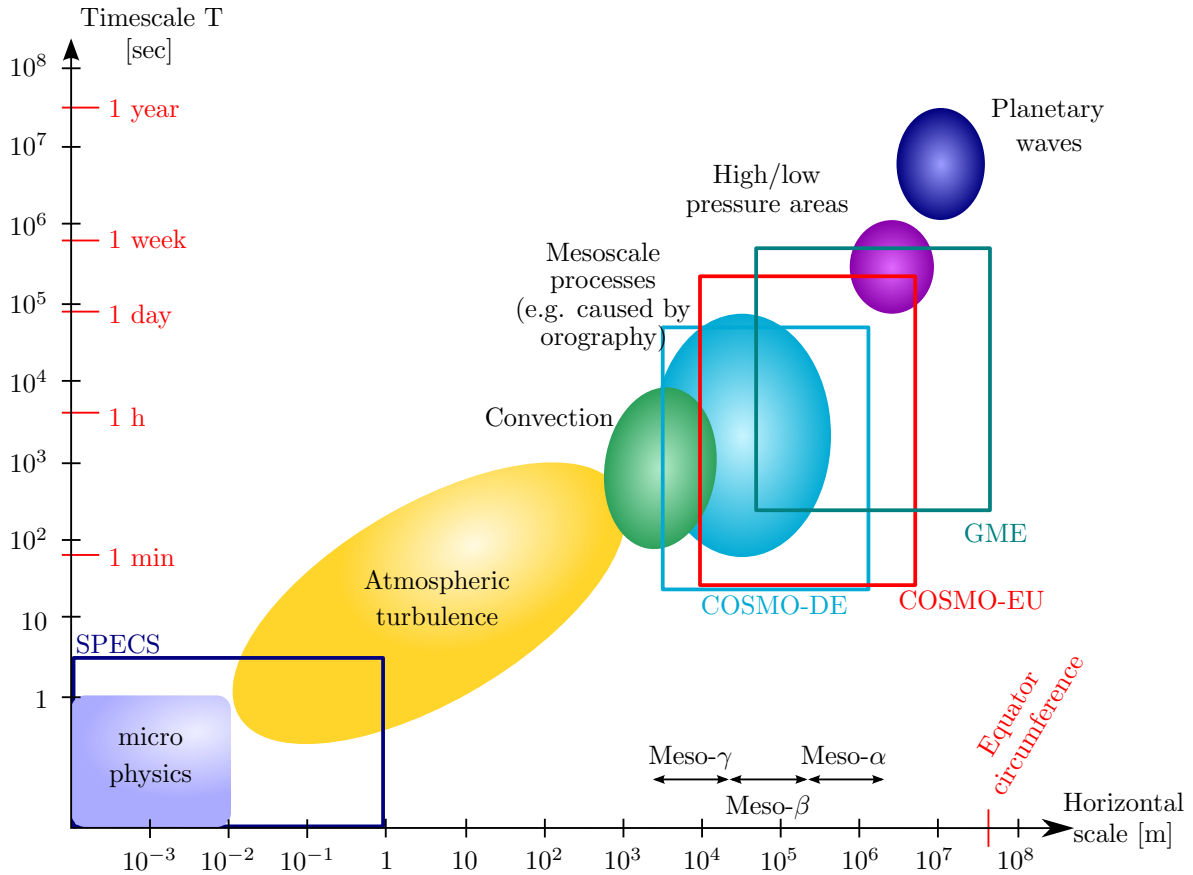


Figure 1.1: Temporal and spatial scales of meteorological processes and scales covered by different models. Courtesy of DWD (German Weather Service); with additions.

system of equations which reduces their efficiency for problems with many degrees of freedom. To combine the advantages of explicit and implicit solvers, implicit/explicit (IMEX) methods have been devised [18], [2]. This class of splitting methods has proven efficient for problems which can be decomposed into non-stiff (or mildly stiff) and stiff terms.

The systems of equations resulting from the semidiscretization of advection equations are challenging on their own. Stability requirements restrict the time step to be smaller than some value proportional to the cell size and inversely proportional to the wind speed. In combination with local grid refinement and spatially varying wind fields this means that the fastest cell dictates the global time step. Multirate methods have been developed which allow to compute the solution of selected components with different time step sizes so that stability requirements lead to a local restriction of the time step instead of a global one. Slower components thus can be integrated using fewer and larger time steps which leads to a reduction of the computational cost. The magnitude of this cost reduction strongly depends on the simulation setup and, as a rule of thumb, is more significant if fewer components require the smallest time step.

An important issue for the practical use of a numerical method is its implementation. A method that looks promising on paper does not necessarily translate to an efficient program. One of the reasons for this is overhead correlated to the algorithmic complexity. Furthermore high performance computing concentrates on distributing a simulation on multiple processors or nodes of a computer cluster. Computational power is mostly given in the form of *computer clusters* – as of November 2011, 82% of all systems listed in the top 500 list (<http://top500.org>) are clusters. Thus it is necessary to distribute the computational cost on multiple nodes of such a cluster. This is done by a decomposition of the spatial domain into blocks. Due to the interdependency of the different blocks, each block must also be provided with information on other blocks' cells in its vicinity. This can be thought of as block local boundary conditions. The correlated data is stored in so called *ghost cells* alias *halo cells* which must be updated periodically. A careful examination of both spatial discretization and time integration is necessary to implement an exchange strategy that provides all data needed while minimizing inter-process communication (i.e. inter block communication). This problem is common to all parallelized simulations of strongly coupled systems.

The aim of this thesis is the construction and analysis of a general splitting scheme which can be employed to construct explicit multirate schemes, IMEX schemes and combined multirate-IMEX schemes. Furthermore we shall detail the efficient implementation of this approach in a parallel atmospheric chemistry transport model. In Chapter 2 we shall introduce the central model equations as well as the necessary preliminaries of spatial semi-discretization and time integration.

Chapter 3 is dedicated to formal aspects of the novel *Recursive Flux Splitting Multirate* approach, a generic splitting scheme which constructs a partitioned time integration method from classical base methods. We shall discuss the general approach and detail the representation of the generated methods as partitioned Runge–Kutta methods. Spatial aspects of the splitting approach shall receive special attention as they are prerequisites for an efficient implementation. The order of the generated methods is analyzed in terms of the underlying base methods. To construct methods that preserve the linear invariants of the system we also consider the direct manipulation of the partitioned methods. The last section of the chapter contains a stability analysis for advection-reaction systems.

In Chapter 4 we shall detail the implementation of the formerly introduced splitting approach in MUSCAT [28], an air pollution model developed at the Leibniz Institute for Tropospheric Research. In particular we shall outline the employed data structures and present the parallel implementation of the time integration scheme. For two or more spatial dimensions the data exchange strategies are significantly more complicated than in the one dimensional case; they shall be discussed in detail. Finally we shall examine how parts of the simulation should be distributed on available processors so that idle times are minimized.

In Chapter 5 we present numerical tests that illustrate practical implications of splitting methods and allow to examine some questions arising in the previous chapters in more detail. We compare the alternative splitting approaches and examine two of the

earlier presented IMEX splittings. To demonstrate the potential of splitting methods we consider the purely explicit solution of a two dimensional diffusion-reaction equation. The performance of the implementation discussed in Chapter 4 shall also be examined.

1.2 State of the art

State of the art numerical models can be used to simulate a broad spectrum of atmospheric phenomena. These range from microphysical and chemical processes with characteristic times below a second and characteristic scales in the sub-millimeter range up to high/low pressure areas and planetary waves on scales about 1000 kilometers and characteristic times up to months. Additionally the simulation domain is often locally refined to allow for a more accurate examination of regions of interests [6],[9]. These may be densely populated areas in contrast to surrounding woodland or agricultural regions, or in a global model continents in contrast to oceans. More sophisticated approaches include dynamic adaptation of the spatial grid [8].

Atmospheric transport and reactions can be described using the advection-diffusion-reaction equation

$$\frac{\partial}{\partial t} c = \underbrace{-\frac{\partial}{\partial x}(uc) - \frac{\partial}{\partial y}(vc) - \frac{\partial}{\partial z}(wc)}_{\text{advection}} + \underbrace{\nabla \left(\rho D \nabla \frac{c}{\rho} \right)}_{\text{diffusion}} + \underbrace{f(x, y, z, t, c)}_{\text{reaction}} \quad (1.1)$$

with the concentration of a transported quantity c , wind vector $(u \ v \ w)^T$, density ρ , diffusion coefficient D and a reaction term f . The latter may also include sources and sinks. There are necessary requirements for the numerical solution of this equation. The preservation of mass is desirable as violation of this property means the introduction of unintended source terms or sink terms. Even more important is the requirement of positivity. Since the transported quantity describes a concentration, negative values are unphysical. Furthermore the reaction equation is usually nonlinear and even small negative concentrations may lead to positive eigenvalues of the Jacobian of f and thus to instability.

After spatial discretization the advection part of (1.1) can be solved efficiently using explicit Runge–Kutta (ERK) time integration methods [5]. These methods have in common that stability requirements limit the global time step to be smaller than some critical value proportional to the ratio of grid size and the magnitude of the wind speed for each cell [7], [23], [50]. Consequently the smallest cell, or more precisely the cell with the smallest characteristic time determines the global time step.

Multirate time integration methods can be employed to adapt the time step locally so that slower components take longer and fewer time steps, resulting in a moderate to substantial reduction of the computational cost, depending on the scenario to simulate [13]. Multirate approaches have been developed since the early 1980s. Osher and Sanders [35] presented a scheme allowing multirate Euler steps. More current approaches allow for a wide variety of multirate schemes by proposing generic multirate methods based on

classical ERKs. In 2005, Tang and Warnecke [45] proposed different multirate schemes which are however not mass preserving. The approach of Constantinescu and Sandu [7] yields multirate ERK schemes which are mass preserving and at most of second order accuracy. In 2007 Hundsdorfer et al. showed that multirate schemes which reduce to multiple applications of the underlying base method cannot be mass preserving and internally consistent at the same time [22]. Multirate schemes based on other classes of methods have also been devised, for instance Multirate explicit Adams methods [37] and Rosenbrock methods [38]. So called *Runge–Kutta–Chebyshev* methods have been introduced by Verwer [46]; these methods are explicit Runge–Kutta methods suitable for the solution of mildly stiff advection-diffusion-reaction equations.

The diffusion and reaction terms in (1.1) on the other hand are usually very stiff and consequently cannot be solved efficiently using explicit methods. Instead implicit methods must be employed. Splitting methods solving the non-stiff terms by explicit methods and the stiff terms by implicit methods are known as implicit/explicit (IMEX) methods [18], [2]. This class of splitting methods proved efficient for problems which can be split into non stiff (or mildly stiff) and stiff terms.

The speed up by use of splitting methods instead of monolithic time integration methods can be estimated by considering how many elements can be solved with which time step. Both the fraction of elements which require the smallest time step and the spectrum of relevant characteristic times is strongly problem dependent. For advective transport the maximum time step size still allowing for stable time integration is determined by the cell size and the wind speed [23]. For semidiscrete advection with one half of the cells stably integratable with a time step of Δt_{macro} and the other half of cells stably integratable with a time step of $\Delta t_{\text{macro}}/2$ only a speed up of $4/3$ can be obtained. If rigorous grid refinement is employed and/or the transport speed varies greatly in the simulated domain the potential benefit from using multirate methods is increased. In electronic circuits there are often comparatively few elements requiring a time step much smaller than the majority of elements and consequently computational cost can be reduced by a much larger amount, e.g. a factor of 6 in CPU time for the inverter chain problem examined in [38].

Splitting strategies can be roughly classified by the underlying methods used and by the coupling of the different subsystems. The implicit-explicit splitting strategy presented by Günther et al. in [15] avoids coupling of “active” and “latent” components during one macro time step; additionally the authors assume that the active components can be solved efficiently using explicit Runge–Kutta methods while the latent components may be solved by an implicit Runge–Kutta method. This is opposed to the splitting strategy presented in this thesis: if an implicit-explicit splitting is constructed as proposed in Section 3.1.1, only the fastest components are solved by an implicit method; additionally we assume full coupling of the subsystems. Savcenko et al. propose a multirate strategy based on Rosenbrock methods for stiff ordinary differential equations [38]. Instead of an a priori partitioning, the authors propose a dynamic partitioning based on a local error estimation. The multirate methods we shall present in this thesis are based on explicit Runge–Kutta methods. Like the methods presented in [45] and [7] they can be represented

as partitioned explicit Runge–Kutta methods. A distinctive feature of our method is that it is tailored to advection due to the use of a splitting by fluxes.

Atmospheric models have many degrees of freedom; the operative global weather model GME uses $\approx 39 \cdot 10^6$ grid points, its small scale counterpart COSMO-DE providing high resolution short term forecasts for Germany uses $\approx 9.7 \cdot 10^6$ grid points at a horizontal resolution of 2.8km (<http://www.dwd.de/modellierung>), where at each grid point a number of variables as density of air, wind speed, water content etc. are defined. Typical air pollution simulations are run with fewer grid cells but with more variables per grid cell, e.g. $\approx 10^5$ grid points and 298 pollutants considered in [20]. This problem size makes the use of modern high performance computers mandatory. To make use of the computational power of computer clusters the simulation must be parallelized, i. e. the model must be decomposed into blocks which then can be processed by different nodes of a cluster. Usually each block corresponds to a physical volume and holds additional information about adjacent blocks in a so called *halo*. This can be interpreted as block-wise boundary conditions. Coupling of neighboring blocks is implemented by data exchanges. While sophisticated time integration routines may serve to reduce the overall computational cost they are more complicated to implement. For a parallel implementation this means that special care must be taken on distribution of the blocks on the available computing elements. A sophisticated graph partitioning library often used for this task is Metis or ParMetis [26] (parallel version). Current versions of ParMetis support so called *multi constraint partitioning* [25] which makes it possible to take more complex program flows into account.

Current research is concerned with parallel implementations suitable to run on very large numbers of nodes [30]. In the latter case the research is correlated to the development of a detailed cloud model [16] which uses more than 200 variables per grid point in order to accurately describe microphysical processes. Another current field of research is general-purpose computing on graphics processing units (GPGPU) [33, 34], which allows for high efficiency in terms of floating point operations per second per watt. Data exchange between central processing units (CPUs) and graphics processing units (GPUs) is however very costly so that special care has to be taken on implementation.

Chapter 2

Discretization in space and time

2.1 Advection-diffusion-reaction equations

Let $c(x, y, z, t)$ be a concentration of a species or a vector of species depending on space coordinates x, y, z and time t . Most commonly used in air pollution models are the following partial differential equations (PDEs) [23]. In the context of air pollution modeling *mixing ratios* $\mu = c/\rho$, ρ being the density of the air, are more frequently used than just the concentrations c . All variables depend on space coordinates x, y, z and time t if not noted otherwise.

Advection

$$\begin{aligned}\frac{\partial}{\partial t}c + \frac{\partial}{\partial x}(uc) + \frac{\partial}{\partial y}(vc) + \frac{\partial}{\partial z}(wc) &= 0, \\ \frac{\partial}{\partial t}(\rho\mu) + \frac{\partial}{\partial x}(\rho u\mu) + \frac{\partial}{\partial y}(\rho v\mu) + \frac{\partial}{\partial z}(\rho w\mu) &= 0,\end{aligned}$$

describing transport of the species caused by a wind with given velocities u, v and w along the x, y and z axes respectively.

Diffusion

$$\begin{aligned}\frac{\partial}{\partial t}c &= \nabla \cdot \left(\rho D \nabla \frac{c}{\rho} \right), \\ \frac{\partial}{\partial t}(\rho\mu) &= \nabla \cdot (\rho D \nabla \mu),\end{aligned}$$

describing propagation of the species due to the mixing ratio gradient $\nabla\mu$, parameterized with a diffusion coefficient D .

Reaction

$$\begin{aligned}\frac{\partial}{\partial t}c &= f(x, y, z, t, c), \\ \frac{\partial}{\partial t}(\rho\mu) &= \tilde{f}(x, y, z, t, \mu, \rho),\end{aligned}$$

describing inter species reactions as well as sources and sinks.

All of these mechanisms can be treated simultaneously by composition to a single advection-diffusion-reaction equation:

$$\begin{aligned}\frac{\partial}{\partial t}c &= -\frac{\partial}{\partial x}(uc) - \frac{\partial}{\partial y}(vc) - \frac{\partial}{\partial z}(wc) + \nabla \left(\rho D \nabla \frac{c}{\rho} \right) + f(x, y, z, t, c), \\ \frac{\partial}{\partial t}(\rho\mu) &= -\frac{\partial}{\partial x}(\rho u \mu) - \frac{\partial}{\partial y}(\rho v \mu) - \frac{\partial}{\partial z}(\rho w \mu) + \nabla (\rho D \nabla \mu) + \tilde{f}(x, y, z, t, \mu, \rho).\end{aligned}$$

The boundary conditions may be periodic, e.g. for global models. Other models operating on a smaller spatial domain employ interpolated results of simulations on a larger scale. This approach is known as *nesting*. Initial conditions also vary between models. They may be given by the results of an earlier simulation, by measurements or by a combination of both as is the case for operative weather models. More academic setups employ an analytically obtained stable state of the atmosphere as initial condition.

If no mass is transferred across the boundary of the simulation domain and the reaction terms involve no sources or sinks these equations lead to a hyperbolic conservation law of the form

$$\int_x \int_y \int_z \sum_i k_i c_i dz dy dx = \text{const} \quad (2.1)$$

for some constant (k_i) which can be deduced from the modeled reaction equations.

If no chemical reactions are modeled the mass conservation law simplifies to read

$$\forall i : \int_x \int_y \int_z c_i dz dy dx = \text{const}$$

indicating that the mass of all contaminants is preserved.

2.2 Spatial discretizations

2.2.1 The method of lines approach

The idea leading to the method of lines (MOL) approach is a separation of spatial and temporal discretization. In the first step, the equation is discretized only in space, leading

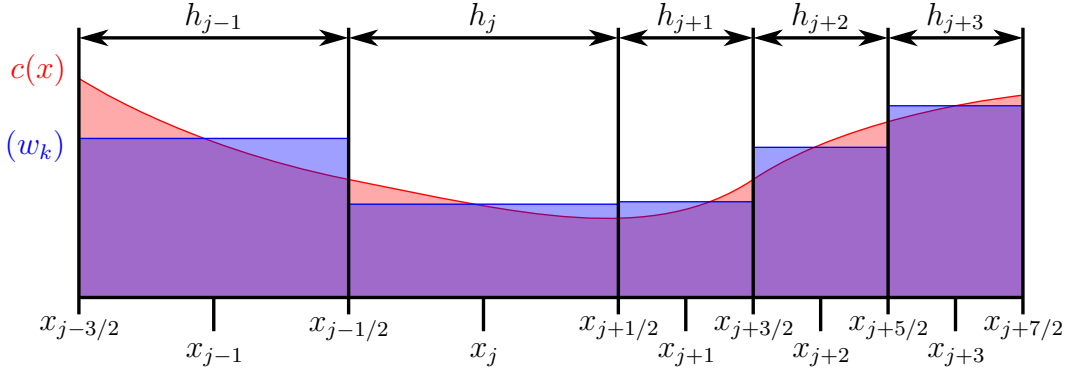


Figure 2.1: Finite volume discretization.

to a semidiscrete system. A partial differential equation (PDE) can be transformed to a semidiscrete ordinary differential equation (semidiscrete ODE). The spatially continuous variable c is approximated by a spatially discrete variable w .

Depending on the specific discretization each element of w may describe the value of c at a given fixed or variable point in space or the average value of c across a certain volume. We shall concentrate on the latter approach, known as *finite volumes*. The elements of the discretization in one spatial dimension read

$$\begin{aligned} w_j(t) &= \frac{1}{h_j} \int_{x_{j-1/2}}^{x_{j+1/2}} c(x, t) dx, \\ h_j &= x_{j+1/2} - x_{j-1/2}, \\ x_j &= \frac{x_{j-1/2} + x_{j+1/2}}{2}, \end{aligned}$$

with the cell size h_j and the cell edges $x_{i\pm 1/2}$. We shall denote the locations of the cell midpoints x_j with integer indexes, cf. Figure 2.1. Some quantities (e.g. density and concentrations) will be defined at the cell midpoints while other quantities (e.g. wind velocity) will be defined at the cell boundaries. This is a so called *staggered grid* or for two or more dimensions an *Arakawa-C* grid [1].

With these definitions we may rewrite the continuous conservation law (2.1) as

$$\sum_j \sum_i k_i w_j h_j = \text{const.}$$

In the context of air pollution modeling the grid often is nonequidistant. It results from an equidistant grid that is refined in regions of interest and/or coarsened in less important regions [6]. Thus the grid is piecewise equidistant, a so called *Shishkin grid* [43]. If not noted otherwise each cell may have its own individual size.

We first shall describe uniform advection in one dimension with constant wind speed $u = \text{const}$ and assume uniform unit density $\rho \equiv 1$:

$$\frac{\partial}{\partial t} c = -\frac{\partial}{\partial x} (uc). \quad (2.2)$$

With the above discretization the PDE (2.2) can be transformed to the semidiscrete *flux form* [23]

$$\dot{w}_j(t) = -\frac{1}{h_j} (f_{j+1/2}(w(t)) - f_{j-1/2}(w(t))), \quad (2.3)$$

with the fluxes f at the cell boundaries $x_{j+1/2}$

$$f_{j+1/2} = uw_{j+1/2}.$$

This can be interpreted so that the temporal evolutions \dot{w}_j are the differences of incoming and outgoing fluxes $f_{j-1/2}, f_{j+1/2}$, divided by the cell size h_j . The division by the cell volume is correlated to the transformation of a mass flux to a concentration evolution.

An obvious question is how to define $w_{j+1/2}$. As a first, crude approach we get the *first order upwind* scheme with

$$w_{j+1/2} = \begin{cases} w_j, & \text{if } u \geq 0, \\ w_{j+1}, & \text{if } u < 0, \end{cases}$$

leading to

$$\dot{w}_j(t) = \begin{cases} -\frac{1}{h_j} (uw_j - uw_{j-1}), & \text{if } u \geq 0, \\ -\frac{1}{h_j} (uw_{j+1} - uw_j), & \text{if } u < 0. \end{cases}$$

This means that a flux leaving a cell is directly proportional to the concentration in the cell. For varying wind speed u defined at the cell boundaries $x_{i\pm 1/2}$ and density ρ defined for each cell the first order upwind scheme reads

$$f_{j+1/2} = (\rho u)_{j+1/2} \left(\frac{w}{\rho} \right)_{j+1/2},$$

$$\left(\frac{w}{\rho} \right)_{j+1/2} = \begin{cases} (w_j/\rho_j), & \text{if } u_{j+1/2} \geq 0, \\ (w_{j+1}/\rho_{j+1}), & \text{if } u_{j+1/2} < 0. \end{cases}$$

It is a mandatory property of asymmetric approaches as (2.4), that the approximation distinguishes between different signs of the wind vector u , so that the discretizations are symmetric in the sense of

$$\dot{w}_j(u, (\dots, w_{j-1}, w_j, w_{j+1}, \dots)) = \dot{w}_j(-u, (\dots, w_{j+1}, w_j, w_{j-1}, \dots)), \quad (2.5a)$$

or in terms of the fluxes

$$f_{j+1/2}(u, (\dots, w_{j-1}, w_j, w_{j+1}, \dots)) = f_{j+1/2}(-u, (\dots, w_{j+2}, w_{j+1}, w_j, \dots)). \quad (2.5b)$$

Other, more accurate approaches can be constructed via interpolation of the values of some adjacent cells. We will examine two kinds of interpolation conditions: a cell averaged condition and a pointwise interpolation.

Choosing a linear function q for interpolation and requiring that the reconstructed mass per cell equals the mass as computed from the cell size and the average concentration we get the interpolation conditions

$$\begin{aligned} h_j w_j &= \int_{x_{j-1/2}}^{x_{j+1/2}} q(x - x_{j+1/2}) dx, \\ h_{j+1} w_{j+1} &= \int_{x_{j+1/2}}^{x_{j+3/2}} q(x - x_{j+1/2}) dx, \end{aligned}$$

which leads to the interpolating polynomial

$$q(x) = \frac{(-2(x - x_{j+1/2}) + h_{j+1}) w_j + (2(x - x_{j+1/2}) + h_j) w_{j+1}}{h_j + h_{j+1}}.$$

For this interpolant the same result can be obtained by pointwise interpolation through the cell midpoints:

$$\begin{aligned} w_j &= q(x_j - x_{j+1/2}) = q(-h_j/2), \\ w_{j+1} &= q(x_{j+1} - x_{j+1/2}) = q(h_{j+1}/2). \end{aligned}$$

Evaluating q at the cell boundary $x = x_{i+1/2}$ gives the approximation

$$w_{j+1/2} = q(0) = \frac{h_{j+1} w_j + h_j w_{j+1}}{h_j + h_{j+1}}, \quad (2.6)$$

called the *second order central scheme* [23].

Better approximations can be obtained by choosing higher order polynomials. If we choose a quadratic polynomial p as interpolant

$$p(x) = a_0 + a_1 (x - x_{j+1/2}) + a_2 (x - x_{j+1/2})^2$$

and require the reconstruction to be consistent with the finite volumes interpretation we obtain the conditions

$$\forall k \in \{j-1, j, j+1\} : \int_{x_{k-1/2}}^{x_{k+1/2}} p(x) dx = w_k h_k$$

or in full:

$$\begin{aligned} a_0 x + \frac{a_1}{2} (x - x_{j+1/2})^2 + \frac{a_2}{3} (x - x_{j+1/2})^3 \Big|_{x_{j-3/2}}^{x_{j-1/2}} &= w_{j-1} h_{j-1}, \\ a_0 x + \frac{a_1}{2} (x - x_{j+1/2})^2 + \frac{a_2}{3} (x - x_{j+1/2})^3 \Big|_{x_{j-1/2}}^{x_{j+1/2}} &= w_j h_j, \\ a_0 x + \frac{a_1}{2} (x - x_{j+1/2})^2 + \frac{a_2}{3} (x - x_{j+1/2})^3 \Big|_{x_{j+1/2}}^{x_{j+3/2}} &= w_{j+1} h_{j+1}. \end{aligned}$$

Using

$$x_{j+1/2} = x_{-1/2} + \sum_{k=1}^j h_k$$

we obtain for a_0, a_1, a_2 the system of equations

$$\begin{aligned} -a_0 h_{j-1} - \frac{a_1}{2} h_{j-1}^2 - \frac{a_2}{3} h_{j-1}^3 &= w_{j-1} h_{j-1}, \\ a_0 h_j + \frac{a_1}{2} h_j^2 + \frac{a_2}{3} h_j^3 &= w_j h_j, \\ a_0 h_{j+1} + \frac{a_1}{2} (2h_j h_{j+1} + h_{j+1}^2) + \frac{a_2}{3} ((h_j + h_{j+1})^3 - h_j^3) &= w_{j+1} h_{j+1}, \end{aligned}$$

resulting in the *third order upwind biased* scheme for non-negative advection speed [23]:

$$\begin{aligned} w_{j+1/2} = p(x_{j+1/2}) = a_0 &= w_j + \alpha_j w_{j-1} + \beta_j w_j + \gamma_j w_{j+1}, \\ \alpha_j &= -\frac{h_j h_{j+1}}{(h_{j-1} + h_j)(h_{j-1} + h_j + h_{j+1})}, \\ \beta_j &= -\frac{h_j ((h_{j-1} + h_j)^2 - h_j h_{j+1} - h_{j+1}^2)}{(h_{j-1} + h_j)(h_j + h_{j+1})(h_{j-1} + h_j + h_{j+1})}, \\ \gamma_j &= \frac{h_j (h_{j-1} + h_j)}{(h_j + h_{j+1})(h_{j-1} + h_j + h_{j+1})}. \end{aligned} \tag{2.7}$$

From (2.5) and (2.7) we obtain the approximation for negative advection speed

$$w_{j+1/2} = w_{j+1} + \alpha_j w_{j+2} + \beta_j w_{j+1} + \gamma_j w_j.$$

If we choose a quadratic interpolant \tilde{p} but interpolate pointwise through the midpoints:

$$\forall k \in \{j-1, j, j+1\} : \tilde{p}(x_k) = w_k,$$

we obtain

$$\begin{aligned} w_{j+1/2} = \tilde{p}(x_{j+1/2}) &= w_j + \tilde{\alpha}_j w_{j-1} + \tilde{\beta}_j w_j + \tilde{\gamma}_j w_{j+1}, \\ \tilde{\alpha}_j &= -\frac{h_j h_{j+1}}{(h_{j-1} + h_j)(h_{j-1} + 2h_j + h_{j+1})}, \\ \tilde{\beta}_j &= -\frac{h_j (h_{j-1} + h_j - h_{j+1})}{(h_{j-1} + h_j)(h_j + h_{j+1})}, \\ \tilde{\gamma}_j &= \frac{h_j (h_{j-1} + 2h_j)}{(h_{j-1} + h_j)(h_{j-1} + 2h_j + h_{j+1})}, \end{aligned} \tag{2.8}$$

for non-negative advection speed and by (2.5)

$$w_{j+1/2} = w_{j+1} + \tilde{\alpha}_j w_{j+2} + \tilde{\beta}_j w_{j+1} + \tilde{\gamma}_j w_j$$

for negative advection speed. For equidistant grids with

$$h_j \equiv h$$

Equations (2.6), (2.7) and (2.8) can be summarized in the κ -family [21]:

$$w_{j+1/2} = \begin{cases} w_j + \frac{1-\kappa}{4}(w_j - w_{j-1}) + \frac{1+\kappa}{4}(w_{j+1} - w_j), & \text{if } u_{j+1/2} \geq 0, \\ w_{j+1} + \frac{1-\kappa}{4}(w_{j+1} - w_{j+2}) + \frac{1+\kappa}{4}(w_j - w_{j+1}), & \text{if } u_{j+1/2} < 0, \end{cases} \quad (2.9)$$

with $\kappa = 1$ for (2.6), $\kappa = 1/3$ for (2.7) and $\kappa = 1/2$ for (2.8).

Further approximations can be obtained using higher order polynomials requiring a larger stencil or a completely different family of functions for interpolation or approximation of the $(w_{j+1/2}, x_{j+1/2})$. However there is no perfect spatial discretization, since every single one is a compromise between smoothness, in the ideal case leading to strict positivity, and accuracy [21]. Positivity means that no cell average concentration w_j at any time may be negative if the cell average concentration given by the initial condition are non-negative

$$\forall j : w_j(t=0) \geq 0 \quad \Rightarrow \quad \forall t : \forall j : w_j(t) \geq 0. \quad (2.10)$$

This is important if w models concentrations which must be positive to have any physical validity. A further requirement is that the discretization is *total variation diminishing* (TVD), i.e. the total variation semi-norm TV must decrease in time:

$$TV(t) = \sum_j |w_j(t) - w_{j-1}(t)|,$$

$$\forall t, \tau > 0 : \quad TV(t + \tau) < TV(t).$$

For instance the first order upwind scheme guarantees positivity while being not very accurate and very diffusive, i.e. details of the solution are lost. The higher order schemes on the other hand lead to improved accuracy and are far less diffusive. They do however introduce spurious oscillations leading to violation of positivity and to an increase of the total variation. Consider for instance application of the κ -family of semi-discretizations on an equidistant grid $\forall j : h_j = h$, uniform positive wind speed $\forall j : u_{j+1/2} = u > 0$ and the following profile:

$$\begin{aligned} w &= (\dots w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad \dots)^T \\ &= (\dots 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad \dots)^T, \\ \dot{w} &= (\dots 0 \quad \frac{u(\kappa+1)}{4h} \quad -\frac{u(\kappa-2)}{2h} \quad -\frac{u}{2h} \quad \frac{u(\kappa-2)}{2h} \quad -\frac{u(\kappa-1)}{4h} \quad 0 \quad \dots)^T. \end{aligned}$$

Obviously the discretization creates a negative value independent of the specific choice of κ since $w_4 = 0$ and $\dot{w}_4 = -\frac{u}{2h} < 0$. Further the total variation increases as for $\kappa \in [0, 1]$ we find

$$\frac{d}{dt}TV = |\dot{w}_2| + |\dot{w}_3 - \dot{w}_2| + |\dot{w}_4 - \dot{w}_3| + |\dot{w}_5 - \dot{w}_4| + |\dot{w}_6 - \dot{w}_5| + |\dot{w}_6| = \frac{u}{h} \left(\frac{9 - 5\kappa}{2} \right) > 0.$$

A requirement often found is mass conservation:

$$\forall t : \sum_j h_j w_j(t) = \text{const} = \sum_j h_j w_j(t=0)$$

resp.

$$\forall t : \sum_j h_j \dot{w}_j(t) = 0.$$

Obviously, the semidiscrete flux equation (2.3) satisfies this condition with any approximation of the intermediate values, since the fluxes leaving one cell always enter one of the neighboring cells, depending on the sign of the advection speed, so that

$$\sum_j h_j \dot{w}_j = \sum_j f_{j-1/2} - \sum_j f_{j+1/2} = 0.$$

2.2.2 Flux limiting

To guarantee positivity of the solution independent of the previous time step the flux can be limited. A possible interpretation of the higher order schemes is an additive correction of the first order upwind scheme [3], the oscillations caused by unlimited higher order schemes from that point of view becoming some kind of overcompensation.

We will rewrite the above discretizations in terms of the slope

$$w_j - w_{j-1},$$

indicating the approximated spatial gradient of the variable at $x_{j-1/2}$ and the slope ratio

$$r_{j+1/2} = \frac{w_{j+1} - w_j}{w_j - w_{j-1}},$$

which is a measure for the curvature of w at x_j . For locally smooth profiles we have $r_{j+1/2} \approx 1$, whereas at extrema the slope ratio becomes negative

$$w_{j-1} < w_j > w_{j+1} \quad \vee \quad w_{j-1} > w_j < w_{j+1} \quad \Rightarrow \quad r_{j+1/2} < 0.$$

In the *slope ratio formulation*, the κ -scheme (2.9) can be cast as

$$w_{j+1/2} = \begin{cases} w_j + \frac{1}{2}K(r_{j+1/2})(w_j - w_{j-1}), & \text{if } u_{j+1/2} \geq 0, \\ w_{j+1} + \frac{1}{2}K(1/r_{j+3/2})(w_{j+1} - w_{j+2}), & \text{if } u_{j+1/2} < 0, \end{cases} \quad (2.11)$$

with

$$K(r) = \frac{1 - \kappa}{2} + \frac{1 + \kappa}{2}r.$$

We will now replace the correction term $K(r_{j+1/2})$ by a limiter function $\phi_{j+1/2}$ so that for non-negative advection speed we have

$$w_{j+1/2} = w_j + \frac{1}{2}\phi_{j+1/2} \cdot (w_j - w_{j-1}). \quad (2.12)$$

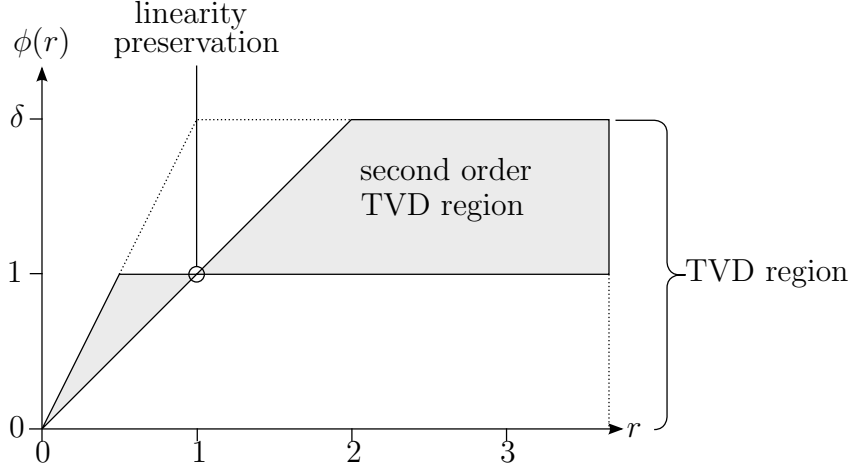


Figure 2.2: TVD region for limiters.

According to [21] a positive semidiscrete solution is guaranteed if the limiter function satisfies:

$$\begin{aligned}
 r_{j+1/2} \leq 0 &\Rightarrow \phi_{j+1/2} = 0, \\
 0 &\leq \phi_{j+1/2} \leq \delta, \\
 \phi_{j+1/2} &\leq 2r_{j+1/2}
 \end{aligned} \tag{2.13}$$

for some $\delta > 0$. Thus in particular the first order upwind scheme (2.4) that corresponds to (2.12) with $\delta \rightarrow 0$, is positive.

Zvan et al. [50] derived conditions for a semidiscretization to be total variation diminishing. These conditions exactly match the above conditions (2.13) for positivity. Additionally the authors introduced conditions to guarantee second order accuracy of a TVD scheme:

$$\begin{aligned}
 r_{j+1/2} \leq \phi(r_{j+1/2}) &\leq \max(2r_{j+1/2}, 1), \quad \text{if } 0 \leq r_{j+1/2} \leq 1, \\
 1 \leq \phi(r_{j+1/2}) &\leq \max(r_{j+1/2}, 2), \quad \text{if } 1 \leq r_{j+1/2}.
 \end{aligned} \tag{2.14}$$

Conditions (2.13) and (2.14) can be illustrated as shown in Figure 2.2. Defining

$$\phi_{j+1/2} = \phi(r_{j+1/2}) := \max\left(0, \min\left(2r_{j+1/2}, \delta, K(r_{j+1/2})\right)\right), \tag{2.15}$$

so that the interval in r , in which $\phi(r) = K(r)$ is maximized, and choosing $\delta = 2$, we obtain the limited, or positive, third order upwind biased scheme for positive advection speed and equidistant grids [23]:

$$w_{j+1/2} = w_j + \max\left(0, \min\left(r_{j+1/2}, 1, \frac{1}{6} + \frac{1}{3}r_{j+1/2}\right)\right) (w_j - w_{j-1}),$$

as well as the positive 1/2-scheme [23]:

$$w_{j+1/2} = w_j + \max\left(0, \min\left(r_{j+1/2}, 1, \frac{1}{4} + \frac{3}{4}r_{j+1/2}\right)\right) (w_j - w_{j-1}).$$

Obviously, the limiter (2.15) switches between the first order upwind scheme ($\phi(r) = 0$), the second order central scheme for equidistant grids ($\phi(r) = r$) and the according κ -scheme ($\phi(r) = K(r)$), using the latter whenever possible and reverting to the first order scheme only at extrema ($r \leq 0 \Rightarrow \phi(r) = 0$).

For non-equidistant grids the spatial discretization has to be adjusted. Berger et al. [3] derived conditions for slope limiters on irregular grids to get a TVD scheme. In the context of flux limiting these conditions can be expressed as

$$\min \left(2r_{j+1/2} \frac{h_j}{h_j + h_{j+1}}, \frac{h_j}{h_{j-1} + h_j} \right) \leq \phi(r_{j+1/2}) \leq \min (2, 2r_{j+1/2}). \quad (2.16)$$

Additionally, Berger et al. derived a linearity preserving condition:

$$\phi \left(\frac{h_j + h_{j+1}}{h_{j-1} + h_j} \right) = \frac{2h_j}{h_{j-1} + h_j}. \quad (2.17)$$

If this condition is met, data which is linear with respect to the spatial coordinate x :

$$\frac{w_j - w_{j-1}}{x_j - x_{j-1}} = \frac{w_{j+1} - w_j}{x_{j+1} - x_j}$$

will remain linear as the reconstructed value $(x_{j+1/2}, w_{j+1/2})$ is obtained via linear interpolation of (x_j, w_j) and (x_{j+1}, w_{j+1}) .

It can be shown that substituting the equidistant κ -scheme with its nonequidistant equivalent results in a scheme satisfying both (2.16) and (2.17).

We transform (2.7) to obtain the approximation $w_{j+1/2}$ in the slope ratio formulation:

$$\begin{aligned} w_{j+1/2} &= w_j + \alpha_j w_{j-1} + \beta_j w_j + \gamma_j w_{j+1} \\ &= w_j - (w_j - w_{j-1})\alpha_j + (w_{j+1} - w_j)\gamma_j + w_j \underbrace{(\alpha_j + \beta_j + \gamma_j)}_{\equiv 0} \\ &= w_j + \left(-\alpha_j + \gamma_j \frac{w_{j+1} - w_j}{w_j - w_{j-1}} \right) (w_j - w_{j-1}) \\ &= w_j + (-\alpha_j + \gamma_j r_{j+1/2}) (w_j - w_{j-1}), \\ \tilde{K}_j(r) &= 2(-\alpha_j + \gamma_j r) \end{aligned}$$

with α_j, γ_j as defined in (2.7) for the 1/3-scheme or (2.8) for the 1/2-scheme respectively.

The limiters resulting in the positive κ -schemes now read

$$\phi(r_{j+1/2}) = \max \left(0, \min \left(r_{j+1/2}, \delta, 2(-\alpha_j + \gamma_j r_{j+1/2}) \right) \right). \quad (2.18)$$

2.2.3 Extension to higher dimensions

Up to this point we discussed spatial discretizations in only one dimension. From a general point of view we approximated the fluxes across cell boundaries by linear combinations

of functions of known cell values:

$$\begin{aligned}\frac{\partial}{\partial t}c &= -\frac{\partial}{\partial x}f(c), \\ w_i &= \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} c \, dx, \\ f_{i+1/2} &= \sum_k \alpha_k f(w_{i+k}), \\ \dot{w}_j &= -\frac{f_{i+1/2} - f_{i-1/2}}{\Delta x},\end{aligned}$$

where the weights α define the spatial discretization. To extend this to higher dimensions, we employ a so called *split dimensional* approach. Consider a general two dimensional transport equation:

$$\frac{\partial}{\partial t}c = -\frac{\partial}{\partial x}f_x(c) - \frac{\partial}{\partial y}f_y(c).$$

Assuming the grid is regular we may employ the discretization in space on the x and the y axes separately:

$$\begin{aligned}\tilde{w}_i(y, t) &= \frac{1}{\Delta x_i} \int_{x_{i-1/2}}^{x_{i+1/2}} c(x, y, t) \, dx, \\ w_{i,j}(t) &= \frac{1}{\Delta y_j} \int_{y_{j-1/2}}^{y_{j+1/2}} \tilde{w}_i(y, t) \, dy, \\ &= \frac{1}{\Delta x_i \Delta y_j} \int_{y_{j-1/2}}^{y_{j+1/2}} \int_{x_{i-1/2}}^{x_{i+1/2}} c(x, y, t) \, dx \, dy \\ \Delta x_i &= x_{i+1/2} - x_{i-1/2}, \\ \Delta y_j &= y_{j+1/2} - y_{j-1/2}, \\ x_i &= \frac{x_{i-1/2} + x_{i+1/2}}{2}, \\ y_j &= \frac{y_{j-1/2} + y_{j+1/2}}{2}.\end{aligned}$$

The fluxes across cell boundaries are then approximated by linear combinations

$$\begin{aligned}f_{i+1/2,j} &= \sum_k \alpha_k^{(x)} f_x(w_{i+k,j}), \\ f_{i,j+1/2} &= \sum_k \alpha_k^{(y)} f_y(w_{i,j+k}).\end{aligned}$$

The superscripts of the α_k shall emphasize that different spatial discretizations may be employed along different main axes. The evolution tendency of the cell – or more precisely: the average cell concentration – now reads

$$\dot{w}_{i,j} = -\frac{f_{i+1/2,j} - f_{i-1/2,j}}{\Delta x} - \frac{f_{i,j+1/2} - f_{i,j-1/2}}{\Delta y}.$$

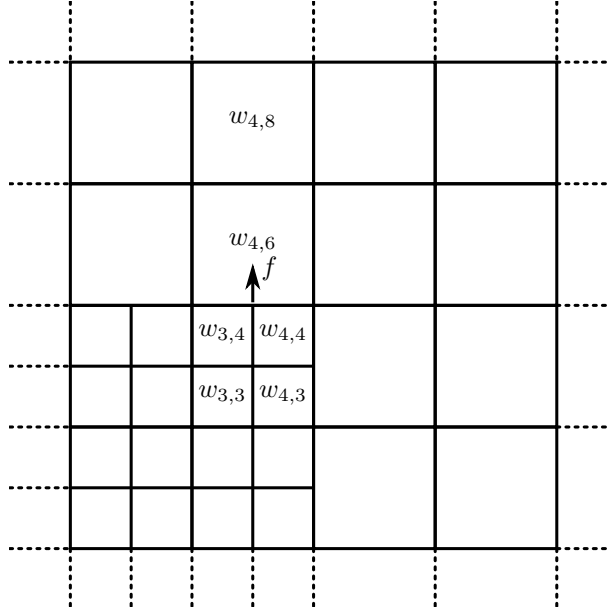


Figure 2.3: Illustration of a two dimensional, locally regular grid.

In two or more spatial dimensions local grid refinement becomes challenging. The originally regular grid then is only *locally* regular, see Figure 2.3. There are also cell faces which belong to one cell on one side and to two or more cells on the other side and which require special treatment. For instance the flux f in Figure 2.3 may be approximated in different ways by either computing two fluxes and averaging or by computing one flux from averaged cell values:

$$\begin{aligned}\tilde{f}^{(1)} &= (F(\dots, w_{3,3}, w_{3,4}, w_{4,6}, w_{4,8}, \dots) + F(\dots, w_{4,3}, w_{4,4}, w_{4,6}, w_{4,8}, \dots)) / 2, \\ \tilde{f}^{(2)} &= F\left(\dots, \frac{w_{3,3} + w_{4,3}}{2}, \frac{w_{3,4} + w_{4,4}}{2}, w_{4,6}, w_{4,8}, \dots\right).\end{aligned}$$

For linear functions F we have $\tilde{f}^{(1)} = \tilde{f}^{(2)}$, but for nonlinear functions as e.g. arising from (2.18) the results vary. Furthermore conservation of mass is guaranteed only if the flux contributing to the source term of $\dot{w}_{4,6}$ equals the flux contributing to the sink terms of $\dot{w}_{3,4}$ and $\dot{w}_{4,4}$.

There are other approaches for obtaining a good spatial discretization which depending on the context may mean high accuracy or small anisotropy. But these approaches require information on the time step size or even completely compound spatial and temporal discretization. As such they fall outside of the method of lines framework and shall not be considered here.

2.3 Runge–Kutta methods

Given an initial value problem

$$\dot{w} = F(t, w), \quad w(t_0) = w_0, \quad (2.19)$$

a Runge–Kutta method advances the state of the system from $w_n \approx w(t_n)$ to $w_{n+1} \approx w(t_{n+1})$ with $t_{n+1} = t_n + \Delta t$ by reformulation to

$$w(t + \Delta t) = w(t) + \int_t^{t+\Delta t} F(\tilde{t}, w(\tilde{t})) d\tilde{t},$$

and approximation of the integral by a suitable combination of discrete stages [23]:

$$w_{n+1} = w_n + \Delta t \sum_{j=1}^s b_j F(t_n + c_j \Delta t, W_{n,j}),$$

$$\forall i \in \{1, \dots, s\} : W_{n,i} = w_n + \Delta t \sum_{j=1}^s a_{ij} F(t_n + c_j \Delta t, W_{n,j}),$$

with stage vectors $W_{n,i} \approx w(t_n + \Delta t c_i)$, ($i = 1, \dots, s$). To keep the notation simple we omit the time step index for the stage vectors from now on, even though the dependency is still implicitly given.

The coefficients (a_{ij}) , (b_i) , (c_i) of a Runge–Kutta method are often arranged in a so-called Butcher tableaux [5] (see Table 2.1). The parameter s is called the number of *stages*, the $(c_i)_{i=1, \dots, s}$ are called the *nodes* of the method. The $(b_j)_{j=1, \dots, s}$ are sometimes called the *summation weights*. If the Runge–Kutta coefficients a_{ij} satisfy

$$\forall j \geq i : a_{ij} = 0,$$

the method is called *explicit*, which means that the W_j depend only on the previously calculated $W_{j'}$, $j' < j$. Usually the c_i satisfy $c_i = \sum_{j=1}^s a_{ij}$. If this row sum condition

general					for explicit methods				
c_1	a_{11}	a_{12}	\cdots	$a_{1,s}$	c_1				
c_2	a_{21}	a_{22}	\cdots	$a_{2,s}$	c_2	a_{21}			
\vdots	\vdots	\vdots	\ddots	\vdots	\vdots	\vdots	\ddots		
c_s	$a_{s,1}$	$a_{s,2}$	\cdots	$a_{s,s}$	c_s	$a_{s,1}$	\cdots	$a_{s,s-1}$	
	b_1	b_2	\cdots	b_s		b_1	\cdots	b_{s-1}	b_s

Table 2.1: Butcher tableaux.

$\begin{array}{c c} 0 & 0 \\ \hline & 1 \end{array}$ (RK1) Euler forward	$\begin{array}{c cc} 0 & & \\ \hline 1 & 1 & \\ & 1/2 & 1/2 \end{array}$ (RK2a) explicit trapezoidal rule	$\begin{array}{c cc} 0 & & \\ \hline 1/2 & 1/2 & \\ & 0 & 1 \end{array}$ (RK2b) Runge's method	$\begin{array}{c ccc} 0 & & & \\ \hline 1/2 & 1/2 & & \\ 1 & 1/2 & 1/2 & \\ & 1/3 & 1/3 & 1/3 \end{array}$ (RK32) Optimal 3-stage 2nd order method [14]
$\begin{array}{c ccc} 0 & & & \\ \hline 1/3 & 1/3 & & \\ 2/3 & 0 & 2/3 & \\ & 1/4 & 0 & 3/4 \end{array}$ (RK3a) Heun's third order method	$\begin{array}{c ccc} 0 & & & \\ \hline 1 & 1 & & \\ 1/2 & 1/4 & 1/4 & \\ & 1/6 & 1/6 & 2/3 \end{array}$ (RK3b)	$\begin{array}{c cccc} 0 & & & & \\ \hline 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$ (RK4) classical fourth order method	

Table 2.2: Explicit Runge–Kutta methods.

holds and $t' = 1$ then (2.19) simplifies to the autonomous system $w' = f(w)$, for which we actually try to approximate the Taylor series expansion [10]

$$w(t + \Delta t) = w(t) + \Delta t F(w(t)) + \dots + \frac{\Delta t^p}{p!} F^{(p)}(w(t)) + \mathcal{O}(\Delta t^{p+1}).$$

If all terms up to order p equal the Taylor series expansion of the exact solution, then the method is said to be *consistent of order p* or short of *order p* . The simplest explicit method is the so called *forward Euler* method (RK1), being first order accurate:

$$w_{n+1} = w_n + \Delta t F(t_n, w_n). \quad (2.20)$$

The parameters of some further explicit methods are listed in Table 2.2. The order of a Runge–Kutta method can be determined formally by examination of the Runge–Kutta coefficients $(a_{ij}), (b_i), (c_i)$ [4].

A means to examine the stability of time integration methods is the so called *von Neumann stability analysis*. Starting point is a linear initial value problem

$$\dot{y}(t) = \lambda y, \quad y(t = 0) = 1, \quad \lambda \in \mathbb{C}, \quad (2.21)$$

whose exact solution is given by

$$y(t) = e^{\lambda t}.$$

Applying a time integration method with a time step of Δt to (2.21) we obtain the *stability function* $R(\lambda \Delta t) \approx y(\Delta t)$ to the exact solution $y(\Delta t)$. The stability region \mathcal{S} is

then defined as the subset of the complex plane for which y is not amplified by numerical integration or in terms of R :

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}.$$

Stability for a linear system of equations can be examined by consideration of an initial value problem

$$\dot{Y} = A \cdot Y, \quad Y(t=0) = I \in \mathbb{R}^{N \times N}, \quad L \in \mathbb{R}^{N \times N}, \quad (2.22)$$

with I denoting the $N \times N$ identity matrix. Numerical solution of system (2.22) gives a matrix $\bar{R}(A\Delta t)$, the so called amplification matrix. If A is a normal matrix with eigenvalues $(\lambda_n)_{n=1, \dots, N}$ then [23]

$$\|\bar{R}(A\Delta t)\|^2 = \max_{1 \leq n \leq N} |R(\lambda_n \Delta t)|,$$

with $\|\cdot\|$ denoting the spectral norm.

We may now examine the case that A represents a linear advection operator with periodic boundary conditions. For this we consider one Fourier mode of the spatially discrete $(w_k)_{k=1, \dots, N}$:

$$w_k(t) = r_n e^{2\pi i n x_k},$$

or for an equidistant grid with N cells on the unit interval $x_k = k \cdot h = k/N$:

$$w_k(t) = r_n e^{2\pi i n h k}. \quad (2.23)$$

Inserting (2.23) into the first order upwind scheme (2.4) for uniform non-negative wind speed $u \geq 0$ we obtain the temporal evolution of the Fourier mode:

$$\begin{aligned} \dot{w}_k(t) &= -\frac{u}{h} (w_k(t) - w_{k-1}(t)), \\ \dot{r}_n e^{2\pi i n k/N} &= -\frac{u}{h} (r_n e^{2\pi i n k/N} - r_n e^{2\pi i n (k-1)/N}), \\ \dot{r}_n &= -r_n \frac{u}{h} (1 - e^{-2\pi i n/N}). \end{aligned}$$

Solving this system with a time integration scheme with stability function $R(z)$ we obtain the amplification of this Fourier mode

$$Amp_n(\Delta t) = \left| R\left(-\Delta t \frac{u}{h} (1 - e^{-2\pi i n/N})\right) \right|.$$

The factor $\Delta t u/h =: \nu$ is called the *Courant number*. The full discretization is stable if the amplification for all Fourier modes is smaller than or equal to one:

$$Amp = \max_{1 \leq n \leq N} |R(-\nu (1 - e^{-2\pi i n/N}))| \leq 1.$$

In this manner we can find a maximum Courant number for pairs of spatial discretization and time integration scheme. Some examples are given in Table 2.3. Note that RK2a and RK2b have the same stability function and thus behave identically for linear problems. The same holds for RK3a and RK3b.

	RK1	RK2a RK2b	RK32	RK3a RK3b	RK4
first order upwind (2.4)	1.00	1.00	2.00	1.26	1.39
second order central (2.6)	-	-	-	1.73	2.83
third order upwind (2.7)	-	0.87	1.26	1.63	1.75
second order upwind (2.8)	-	0.79	1.14	1.85	2.03

Table 2.3: Numerically determined maximum Courant numbers for some advection discretizations and explicit Runge–Kutta methods given in Table 2.2.

2.3.1 Partitioned Runge–Kutta methods

Partitioned Runge–Kutta (PRK) methods apply different Runge–Kutta methods on different partitions of the equation. This provides a theoretical framework for the examination of the partitioned schemes introduced later. To apply a partitioned method to an equation, the equation must be split. There are two splitting paradigms, leading to equivalent results: a *splitting by components* and a *splitting of the right hand side*. Consider an autonomous differential equation

$$\dot{w}_k = F_k(w),$$

with spatial index $k \in K$ for a set of indexes $K = \{1, \dots, m\}$. To obtain a bipartite splitting by components we define

$$\begin{aligned} w &= \begin{pmatrix} w^{(1)} \\ w^{(2)} \end{pmatrix}, \\ w^{(i)} &= \{w_k : k \in K_i\}, \\ \dot{w} &= \begin{pmatrix} \tilde{F}^{(1)}(w^{(1)}, w^{(2)}) \\ \tilde{F}^{(2)}(w^{(1)}, w^{(2)}) \end{pmatrix}, \\ \tilde{F}_k^{(l)}(w^{(1)}, w^{(2)}) &= \left\{ F_k \left(\begin{pmatrix} w^{(1)} \\ w^{(2)} \end{pmatrix} \right) : k \in K_l \right\}, \end{aligned}$$

with a partitioning of the indexes

$$K = K_1 \cup K_2; \quad K_1 \cap K_2 = \emptyset.$$

A partitioned Runge–Kutta method (PRK) consists of two or more Runge–Kutta methods with potentially different parameters (a, b, c) but equal number of stages s . Applying a

bipartite PRK on the partitioned system we obtain

$$\begin{aligned}
w_{n+1}^{(1)} &= w_n^{(1)} + \Delta t \sum_{j=1}^s b_j^{(1)} \tilde{F}^{(1)} \left(W_j^{(1)}, W_j^{(2)} \right), \\
\forall i \in \{1, \dots, s\} : W_i^{(1)} &= w_n^{(1)} + \Delta t \sum_{j=1}^s a_{i,j}^{(1)} \tilde{F}^{(1)} \left(W_j^{(1)}, W_j^{(2)} \right), \\
w_{n+1}^{(2)} &= w_n^{(2)} + \Delta t \sum_{j=1}^s b_j^{(2)} \tilde{F}^{(2)} \left(W_j^{(1)}, W_j^{(2)} \right), \\
\forall i \in \{1, \dots, s\} : W_i^{(2)} &= w_n^{(2)} + \Delta t \sum_{j=1}^s a_{i,j}^{(2)} \tilde{F}^{(2)} \left(W_j^{(1)}, W_j^{(2)} \right).
\end{aligned}$$

This splitting may equivalently be expressed as a splitting of the right hand side:

$$\begin{aligned}
\dot{w} &= \bar{F}^{(1)}(w) + \bar{F}^{(2)}(w), \\
\bar{F}_k^{(l)}(w) &= \begin{cases} F_k(w) & \text{if } k \in K_l, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

Again applying the general PRK we obtain

$$w_{n+1} = w_n + \Delta t \sum_{j=1}^s b_j^{(1)} \bar{F}^{(1)}(W_j) + \Delta t \sum_{j=1}^s b_j^{(2)} \bar{F}^{(2)}(W_j), \quad (2.24a)$$

$$\forall i \in \{1, \dots, s\} : W_i = w_n + \Delta t \sum_{j=1}^s a_{ij}^{(1)} \bar{F}^{(1)}(W_j) + \Delta t \sum_{j=1}^s a_{ij}^{(2)} \bar{F}^{(2)}(W_j). \quad (2.24b)$$

The equivalency of these splitting approaches can easily be shown using the following relation:

$$\begin{aligned}
\bar{F}^{(1)}(w) &= \begin{pmatrix} \tilde{F}^{(1)}(w^{(1)}, w^{(2)}) \\ 0 \end{pmatrix}, \\
\bar{F}^{(2)}(w) &= \begin{pmatrix} 0 \\ \tilde{F}^{(2)}(w^{(1)}, w^{(2)}) \end{pmatrix}.
\end{aligned}$$

A more formal examination of this equivalency is given by Ascher et al. in [2]. The difference of those splittings is that the $\bar{F}^{(i)}$ formally are defined on all indexes (though they may have finite supports), whereas the $\tilde{F}^{(i)}$ operate on disjoint index sets.

As for non-partitioned Runge–Kutta methods, formal order conditions can be derived [24]. These conditions consist of the classical order conditions for the individual Runge–Kutta methods and additional *coupling conditions*. The order conditions for bipartite PRKs are as follows

Order 1 - classical:

$$\sum_{i=1}^s b_i^{(1)} = \sum_{i=1}^s b_i^{(2)} = 1.$$

Order 2 - classical:

$$\sum_{i=1}^s b_i^{(1)} c_i^{(1)} = \sum_{i=1}^s b_i^{(2)} c_i^{(2)} = 1/2.$$

Order 2 - coupling:

$$\sum_{i=1}^s b_i^{(1)} c_i^{(2)} = \sum_{i=1}^s b_i^{(2)} c_i^{(1)} = 1/2.$$

Order 3 - classical:

$$\begin{aligned} \sum_{i=1}^s b_i^{(1)} \left(c_i^{(1)}\right)^2 &= \sum_{i=1}^s b_i^{(2)} \left(c_i^{(2)}\right)^2 = \frac{1}{3}, \\ \sum_{i=1}^s \sum_{j=1}^s b_i^{(1)} a_{i,j}^{(1)} c_j^{(1)} &= \sum_{i=1}^s \sum_{j=1}^s b_i^{(2)} a_{i,j}^{(2)} c_j^{(2)} = \frac{1}{6}. \end{aligned}$$

Order 3 - coupling:

$$\begin{aligned} \sum_{i=1}^s b_i^{(2)} \left(c_i^{(1)}\right)^2 &= \sum_{i=1}^s b_i^{(1)} \left(c_i^{(2)}\right)^2 = \frac{1}{3}, \\ \sum_{i=1}^s \sum_{j=1}^s b_i^{(2)} a_{i,j}^{(1)} c_j^{(1)} &= \sum_{i=1}^s \sum_{j=1}^s b_i^{(1)} a_{i,j}^{(2)} c_j^{(2)} = \frac{1}{6}, \\ \sum_{i=1}^s b_i^{(1)} \left(c_i^{(1)} - c_i^{(2)}\right)^2 &= \sum_{i=1}^s b_i^{(2)} \left(c_i^{(1)} - c_i^{(2)}\right)^2 = 0, \\ \sum_{i=1}^s b_i^{(1)} c_i^{(1)} \left(c_i^{(1)} - c_i^{(2)}\right) &= \sum_{i=1}^s b_i^{(2)} c_i^{(2)} \left(c_i^{(1)} - c_i^{(2)}\right) = 0, \\ \sum_{i=1}^s \sum_{j=1}^s b_i^{(1)} a_{i,j}^{(1)} \left(c_j^{(1)} - c_j^{(2)}\right) &= \sum_{i=1}^s \sum_{j=1}^s b_i^{(2)} a_{i,j}^{(1)} \left(c_j^{(1)} - c_j^{(2)}\right) = 0, \\ \sum_{i=1}^s \sum_{j=1}^s b_i^{(1)} a_{i,j}^{(2)} \left(c_j^{(1)} - c_j^{(2)}\right) &= \sum_{i=1}^s \sum_{j=1}^s b_i^{(2)} a_{i,j}^{(2)} \left(c_j^{(1)} - c_j^{(2)}\right) = 0. \end{aligned}$$

Both implicit/explicit (IMEX) methods and *multirate* methods based on Runge–Kutta methods can be cast as PRKs. An IMEX-PRK has the property that one partition is explicit while the other one is implicit:

$$\forall i : \forall j \geq i : a_{i,j}^{(1)} = 0, \quad (2.25a)$$

$$\exists i : \exists j \geq i : a_{i,j}^{(2)} \neq 0. \quad (2.25b)$$

This class of methods has been examined for instance by Ascher et al. in [2]. A multirate PRK on the other hand is characterized by the relative computational cost and the relative

stability of the methods. Ideally if method (1) requires $E_{RHS}^{(1)}$ evaluations of the right hand side and is stable for a given ODE and a time step of $\Delta t \leq \Delta t_{\max}^{(1)}$ then method (2) should be stable for a time step of $\Delta t \leq \Delta t_{\max}^{(2)} = m\Delta t_{\max}^{(1)}$ while requiring $E_{RHS}^{(2)} = m \cdot E_{RHS}^{(1)}$ evaluations of the right hand side for some value m . Multirate methods based on explicit Runge–Kutta methods have been presented for instance in [35] and [45]. A class of generic multirate methods have been presented by Constantinescu and Sandu in [7]. Following the nomenclature of Hundsdorfer [22] we call a multirate scheme a *true* multirate scheme if the methods reduce to multiple applications of a base method if employed separately.

In Section 3.2.2 we shall demonstrate how to construct parameters for PRKs arising from the generic splitting scheme. The order analysis for this class of methods shall be carried out in terms of these PRK parameters in Section 3.3.

Chapter 3

A generic splitting scheme

3.1 Recursive Flux Splitting Multirate

In this section we shall discuss formal properties of a generic splitting approach. The approach was first outlined in a paper by Knoth and Wolke [29]. The authors proposed an IMEX splitting similar to the general splitting to be discussed in Section 3.1.1 and for formal analysis assumed exact solution of the implicit part of the equation. In the course of my Diploma thesis we examined the use of explicit Runge–Kutta methods instead of the exact solution and thus constructed an explicit multirate scheme. As conservation of mass was a central requirement we chose to combine this class of methods with a splitting by fluxes as described in Section 3.2.1. The resulting class of methods was named the *Recursive Flux Splitting Multirate* scheme (RFSMR) [39]. In [40] we presented a more general splitting from which multirate-IMEX methods could be constructed.

3.1.1 General splitting

We shall now describe a generic splitting approach. Consider a differential equation

$$\dot{w} = F(w) + G(w).$$

Here G represents advection with comparatively low Courant numbers. The other term, F , may represent diffusion-reaction or advection with higher Courant numbers. For easier readability the terms associated with explicit methods will always be denoted by G , later with superscript indexes. The starting point is an explicit Runge–Kutta method (ERK) with s stages and parameters (a_{ij}, b_j, c_i) , $i \in 1, \dots, s$, $j \in 1, \dots, s$ in common notation. Subsequently we shall call this method the *outer* method. For this outer method it is a strict requirement that $\forall i : c_i = \sum_{j=1}^s a_{ij}$. We extend the notation by defining

$$a_{s+1,j} := b_j, \tag{3.1a}$$

$$c_{s+1} := \sum_{j=1}^s a_{s+1,j} = \sum_{j=1}^s b_j, \tag{3.1b}$$

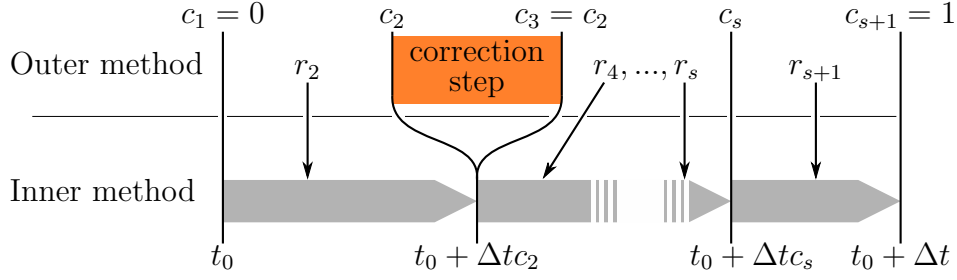


Figure 3.1: Illustration of one time step for generic method.

thus avoiding a separate treatment of the summation stage. The idea of the general splitting can be illustrated as shown in Figure 3.1. Note that we always have $c_1 = 0$ since the outer method is an explicit Runge–Kutta method; we have $c_{s+1} = 1$ for methods which are at least first order consistent. The equality $c_2 = c_3$ on the other hand was chosen for illustration purposes. We treat the stage values W_i as approximations to the real solution according to

$$W_i \approx w(t_0 + \Delta tc_i),$$

and integrate “from node to node” with a further method which we shall call *inner* method. If two successive nodes differ $c_{i-1} \neq c_i$, the outer method affects the inner method via a stage specific source term r_i . If on the other hand two successive nodes are equal the outer method affects the inner method via the initial condition of the next integration interval. We shall call this a *correction step*.

The algorithm for computing an approximate solution w_1 at time t_1 from a state w_0 at time t_0 can be cast as follows.

$$\begin{aligned} W_1 &= w_0, \\ r_i &= \sum_{j=1}^{i-1} (a_{ij} - a_{i-1,j}) G(W_j), \\ v_i(0) &= W_{i-1}, \end{aligned} \tag{3.2a}$$

$$\frac{dv_i}{d\tau} = \Delta tr_i + \underbrace{\Delta t (c_i - c_{i-1}) F(v_i)}_{(*)}, \quad \tau \in [0, 1], \quad i = 2, \dots, s+1, \tag{3.2b}$$

$$\begin{aligned} W_i &= v_i(1), \\ w_1 &= W_{s+1}. \end{aligned} \tag{3.2c}$$

For stages i with $c_i = c_{i-1}$ (i.e. for the correction steps) the term $(*)$ becomes zero so that (3.2) simplify to

$$W_i = W_{i-1} + \Delta t \sum_{j=1}^{i-1} (a_{ij} - a_{i-1,j}) G(W_j).$$

For $F \equiv 0$ the term $(*)$ is zero for all stages so that we obtain the classical ERK with the

parameters of the outer method approximating the solution of $\dot{w} = G(w)$:

$$\begin{aligned} W_1 &= w_0, \\ W_i &= W_{i-1} + \Delta t \sum_{j=1}^{i-1} (a_{ij} - a_{i-1,j}) G(W_j), \\ &= W_1 + \Delta t \sum_{j=1}^{i-1} a_{ij} G(W_j), \\ w_1 &= W_{s+1}. \end{aligned}$$

Generally equation (3.2b) may be solved using an arbitrary inner method. If an implicit Runge–Kutta method is employed we obtain an IMEX splitting in the sense of (2.25), employing an explicit Runge–Kutta method leads to an explicit multirate method. By recursive application a multirate method with more than two temporal refinement levels or a multirate-IMEX splitting can be constructed.

3.1.2 Construction of PRKs

We shall now examine the case that the inner method is a Runge–Kutta method. To allow for a distinction of the method parameters we denote the parameters of the inner method by a superscript “I” opposed to the parameters of the outer method denoted by a superscript “O”. The full method then reads:

$$W_1 = w_0, \tag{3.3a}$$

$$r_i = \sum_{j=1}^{i-1} (a_{ij}^O - a_{i-1,j}^O) G(W_j), \tag{3.3b}$$

$$V_{i,1} = W_{i-1}, \tag{3.3c}$$

$$V_{i,k} = V_{i,k-1} + \Delta t (c_k^I - c_{k-1}^I) r_i \tag{3.3d}$$

$$\begin{aligned} &+ \Delta t (c_i^O - c_{i-1}^O) \sum_{l=1}^{s^I} (a_{kl}^I - a_{k-1,l}^I) F(V_{i,l}), \\ &= V_{i,k-1} + \Delta t (c_k^I - c_{k-1}^I) \sum_{j=1}^{i-1} (a_{ij}^O - a_{i-1,j}^O) G(W_j) \tag{3.3e} \end{aligned}$$

$$+ \Delta t (c_i^O - c_{i-1}^O) \sum_{l=1}^{s^I} (a_{kl}^I - a_{k-1,l}^I) F(V_{i,l}),$$

$$i = 2, \dots, s^O + 1, \quad k = 2, \dots, s^I + 1, \tag{3.3f}$$

$$W_i = V_{i,s^I+1}, \tag{3.3g}$$

$$w_1 = W_{s^O+1}. \tag{3.3h}$$

While the inner method may be implicit it follows from (3.3c) and (3.3g) that

$$\forall i \in \{2, \dots, s^O + 1\} : V_{i,1} = W_{i-1} = V_{i-1,s^I+1},$$

$$\begin{array}{c|cc}
\frac{3+\sqrt{3}}{6} & \frac{3+\sqrt{3}}{6} & \\
\frac{3-\sqrt{3}}{6} & -\frac{1}{\sqrt{3}} & \frac{3+\sqrt{3}}{6} \\
\hline
& 1/2 & 1/2
\end{array}
\quad
\begin{array}{c|cc}
0 & 0 & \\
\frac{3+\sqrt{3}}{6} & 0 & \frac{3+\sqrt{3}}{6} \\
\frac{3-\sqrt{3}}{6} & 0 & -\frac{1}{\sqrt{3}} & \frac{3+\sqrt{3}}{6} \\
\hline
& 0 & 1/2 & 1/2
\end{array}$$

(SDIRK3)

Table 3.1: A *singly diagonal* 2-stage 3rd order implicit Runge–Kutta method (left) and an equivalent method with $c_1 = 0$ (right).

which is satisfied only if $c_1^I = c_{s^I+1}^I - 1 = 0$. This is no real restriction since for every inner Runge–Kutta method of at least first order accuracy we have $c_{s^I+1}^I = 1$. Furthermore for every Runge–Kutta method we can construct an equivalent Runge–Kutta method with $c_1 = 0$ by inserting a redundant stage, see Table 3.1 for an example.

In the context of multirate methods we shall call the sub method applied on G the *slow* method opposed to the method applied on F which shall be called the *fast* method. This corresponds to G describing transport with a lower Courant number than F . The complete partitioned method is a *true* multirate method in the sense that it reduces to multiple applications of the inner base method with a smaller time step if $G \equiv 0$. This point of view motivates the notion of the time step ratio R , that is the number of micro time steps per macro time step. Formally the time step ratio depends solely on the nodes c_i^O of the outer method. The inner method is executed s^O times with time step sizes of $\{\Delta t(c_2^O - c_1^O), \Delta t(c_3^O - c_2^O), \dots\}$. This is no severe limitation however as the inner method may be replaced by a composition of n steps of relative step size $1/n$, see Table 3.2 for an example.

To allow for a formal analysis of this class of multirate methods in a well-established framework we write algorithm (3.3) as a partitioned Runge–Kutta method.

$$\begin{aligned}
W_1 &= w_0, \\
W_i &= w_0 + \Delta t \sum_{j=1}^s a_{i,j}^{(1)} G(W_j) + \Delta t \sum_{j=1}^s a_{i,j}^{(2)} F(W_j)
\end{aligned} \tag{3.4a}$$

$$\begin{aligned}
&= W_{i-1} + \Delta t \sum_{j=1}^s \left(a_{i,j}^{(1)} - a_{i-1,j}^{(1)} \right) G(W_j) \\
&\quad + \Delta t \sum_{j=1}^s \left(a_{i,j}^{(2)} - a_{i-1,j}^{(2)} \right) F(W_j), \quad i = 2, \dots, s+1,
\end{aligned} \tag{3.4b}$$

$$w_1 = W_{s+1},$$

with suitable parameters $(a_{i,j}^{(1)})_{i,j=1,\dots,s}$ and $(a_{i,j}^{(2)})_{i,j=1,\dots,s}$. To construct the PRK parameters we employ a matrix notation as follows:

$$\begin{aligned}
A^m &= \begin{pmatrix} a_{1,1}^m & a_{1,2}^m & a_{1,3}^m & \cdots \\ a_{2,1}^m & a_{2,2}^m & a_{2,3}^m & \cdots \\ a_{3,1}^m & a_{3,2}^m & a_{3,3}^m & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \\
b^m &= (b_1^m \ b_2^m \ b_3^m \ \cdots), \\
c^m &= (c_1^m \ c_2^m \ c_3^m \ \cdots)^T,
\end{aligned}$$

for a superscript $m \in \{O, I, (1), (2)\}$ indicating a method. Additionally we define a transformation:

$$\begin{aligned}
A^m &= L \Delta A^m, & L &= \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 1 & 1 & 0 & \cdots \\ 1 & 1 & 1 & \ddots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}, \\
\Delta A^m &= L^{-1} A^m, & L^{-1} &= \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ & & & \ddots & \ddots \end{pmatrix}.
\end{aligned}$$

The matrices $A^{(1)}$ and $A^{(2)}$ correspond to the *classical* notation (3.4a), while the matrices $\Delta A^{(1)}$ and $\Delta A^{(2)}$ correspond to the *incremental* notation (3.4b). We shall construct the parameters in incremental form. Defining the $s^I \times s^I$ auxiliary matrices

$$\begin{aligned}
\Delta A^I &= L^{-1} A^I, \\
\Delta B^I &= \begin{pmatrix} b_1^I - a_{s^I,1}^I & \cdots & b_{s^I}^I - a_{s^I,s^I}^I \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}, \\
\Delta C^I &= \begin{pmatrix} c_1^I - c_0^I & 0 & \cdots \\ c_2^I - c_1^I & 0 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}, \\
\Delta M &= \begin{pmatrix} 1 & 0 & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix},
\end{aligned}$$

we may deduce from (3.3e) the parameters of the slow method

$$\Delta A^{(1)} = \begin{pmatrix} (a_{1,1}^O - a_{0,1}^O) \Delta M + (a_{2,1}^O - a_{1,1}^O) \Delta C^I & & & \\ (a_{2,1}^O - a_{1,1}^O) \Delta M + (a_{3,1}^O - a_{2,1}^O) \Delta C^I & (a_{2,2}^O - a_{1,2}^O) \Delta M + (a_{3,2}^O - a_{2,2}^O) \Delta C^I & & \\ \vdots & \vdots & & \ddots \end{pmatrix},$$

and the parameters of the fast method

$$\Delta A^{(2)} = \begin{pmatrix} (c_2^O - c_1^O)\Delta A^I & & & \\ (c_2^O - c_1^O)\Delta B^I & (c_3^O - c_2^O)\Delta A^I & & \\ & (c_3^O - c_2^O)\Delta B^I & \ddots & \\ & & & \ddots \end{pmatrix}.$$

To construct the parameters in classical notation $A^{\text{slow}} = L\Delta A^{\text{slow}}$ and $A^{\text{fast}} = L\Delta A^{\text{fast}}$ we first transform the auxiliary matrices:

$$\begin{aligned} A^I &= L\Delta A^I, \\ B^I &= \begin{pmatrix} b_1^I & \cdots & b_{s^I}^I \\ \vdots & \ddots & \vdots \\ b_1^I & \cdots & b_{s^I}^I \end{pmatrix}, \\ C^I &= L\Delta C^I, \\ M &= L\Delta M = \begin{pmatrix} 1 & 0 & \cdots \\ 1 & 0 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}, \end{aligned}$$

where B^I is defined differently from the other matrices so that

$$L \begin{pmatrix} \Delta A^I & & & \\ \Delta B^I & \Delta A^I & & \\ & \Delta B^I & \Delta A^I & \\ & & \ddots & \ddots \end{pmatrix} = \begin{pmatrix} A^I & & & \\ B^I & A^I & & \\ B^I & B^I & A^I & \\ \vdots & \vdots & \ddots & \ddots \end{pmatrix}.$$

We may now write the parameters of the method in classical notation, directly corresponding to Butcher tableaux:

$$\begin{aligned} A^{(1)} = L\Delta A^{(1)} &= \begin{pmatrix} a_{1,1}^O M + (a_{2,1}^O - a_{1,1}^O)C^I & & & \\ a_{2,1}^O M + (a_{3,1}^O - a_{2,1}^O)C^I & a_{2,2}^O M + (a_{3,2}^O - a_{2,2}^O)C^I & & \\ \vdots & \vdots & \ddots & \\ & & & \ddots \end{pmatrix}, \\ A^{(2)} = L\Delta A^{(2)} &= \begin{pmatrix} (c_2^O - c_1^O)A^I & & & \\ (c_2^O - c_1^O)B^I & (c_3^O - c_2^O)A^I & & \\ (c_2^O - c_1^O)B^I & (c_3^O - c_2^O)B^I & \ddots & \\ \vdots & \vdots & & \ddots \end{pmatrix}. \end{aligned}$$

This notation clearly shows once again that the first PRK sub method $A^{(1)}$ is equivalent to the outer method A^O with additional redundant stages while the second PRK sub method $A^{(2)}$ is equivalent to the inner base method A^I executed s^O times with relative step sizes of $\{c_2^O - c_1^O, \dots, c_{s^O+1}^O - c_{s^O}^O\}$. Employing the vectors

$$\mathbf{1} = (1, \dots, 1)^T,$$

$$\mathbf{e}_1 = (1, 0, \dots, 0),$$

we find the nodes of the constructed methods to read

$$\begin{aligned} c^{(1)} &= A^{(1)}\mathbf{e}_1 = \\ c^{(2)} &= A^{(2)}\mathbf{e}_1 = \begin{pmatrix} c_1^{\text{O}}\mathbf{1} + (c_2^{\text{O}} - c_1^{\text{O}})c^{\text{I}} \\ c_2^{\text{O}}\mathbf{1} + (c_3^{\text{O}} - c_2^{\text{O}})c^{\text{I}} \\ \vdots \end{pmatrix}. \end{aligned}$$

Using our extension of the common notation by $a_{s+1,j} = b_j$ we may extract the summation weights to read

$$\begin{aligned} b^{(1)} &= \mathbf{e}_1 (a_{s^{\text{O}}+1,1}^{\text{O}}M + (a_{s^{\text{O}}+2,1}^{\text{O}} - a_{s^{\text{O}}+1,1}^{\text{O}})C^{\text{I}} \quad a_{s^{\text{O}}+1,2}^{\text{O}}M + (a_{s^{\text{O}}+2,2}^{\text{O}} - a_{s^{\text{O}}+1,2}^{\text{O}})C^{\text{I}} \quad \dots) \\ &= (a_{s^{\text{O}}+1,1}^{\text{O}}\mathbf{e}_1 \quad a_{s^{\text{O}}+1,2}^{\text{O}}\mathbf{e}_1 \quad \dots) \\ &= (b_1^{\text{O}}\mathbf{e}_1 \quad b_2^{\text{O}}\mathbf{e}_1 \quad \dots), \\ b^{(2)} &= \mathbf{e}_1 ((c_2^{\text{O}} - c_1^{\text{O}})B^{\text{I}} \quad (c_3^{\text{O}} - c_2^{\text{O}})B^{\text{I}} \quad \dots) \\ &= ((c_2^{\text{O}} - c_1^{\text{O}})b^{\text{I}} \quad (c_3^{\text{O}} - c_2^{\text{O}})b^{\text{I}} \quad \dots). \end{aligned}$$

For an element-wise notation we employ tuple indexes with the first element of the tuple corresponding to the blocks in the matrix notation and to the stages of the outer method respectively:

$$a_{(i,k),(j,l)}^{(1)} = \begin{cases} a_{i,j}^{\text{O}} + (a_{i+1,j}^{\text{O}} - a_{i,j}^{\text{O}})c_k^{\text{I}} & \text{if } l = 1 \\ 0 & \text{if } l > 1 \end{cases}, \quad (3.5a)$$

$$b_{(j,l)}^{(1)} = \begin{cases} b_j^{\text{O}} & \text{if } l = 1 \\ 0 & \text{if } l > 1 \end{cases}, \quad (3.5b)$$

$$c_{(i,k)}^{(1)} = c_i^{\text{O}} + c_k^{\text{I}}(c_{i+1}^{\text{O}} - c_i^{\text{O}}), \quad (3.5c)$$

$$a_{(i,k),(j,l)}^{(2)} = \begin{cases} 0 & \text{if } j > i \\ (c_{j+1}^{\text{O}} - c_j^{\text{O}})a_{k,l}^{\text{I}} & \text{if } j = i \\ (c_{j+1}^{\text{O}} - c_j^{\text{O}})b_l^{\text{I}} & \text{if } j < i \end{cases}, \quad (3.5d)$$

$$b_{(j,l)}^{(2)} = (c_{j+1}^{\text{O}} - c_j^{\text{O}})b_l^{\text{I}}, \quad (3.5e)$$

$$c_{(i,k)}^{(2)} = c_i^{\text{O}} + c_k^{\text{I}}(c_{i+1}^{\text{O}} - c_i^{\text{O}}) = c_{(i,k)}^{(1)}. \quad (3.5f)$$

The resulting PRK is internally consistent (i.e. $c^{(1)} = c^{(2)}$), but it does not preserve all linear invariants, as the summation weights of the partitions differ (i.e. $b^{(1)} \neq b^{(2)}$). Trying to find base methods in order to satisfy $b^{(1)} = b^{(2)}$ leads to the restriction of the inner method to forward Euler and ultimately results in a (formally) partitioned method with $A^{(1)} = A^{(2)}$. The extension of the partitioned method to a mass preserving method will be sketched in Section 3.4. Note that though the constructed PRK formally has $s^{\text{O}}s^{\text{I}}$ stages, some of these stages may be redundant. This means that there is a partitioned method with fewer stages yielding equal results. Table 3.2 shows an explicit multirate

0	
1	1
	1/2 1/2

outer method
(RK2a)

0	
1/2	1/2
1/2	1/4 1/4
1	1/4 1/4 1/2
	1/4 1/4 1/4 1/4

inner method
(composition of two RK2a half steps)

0	
1/2	1/2
1/2	1/2 0
1	1 0 0
1	1 0 0 0
1	3/4 0 0 0 1/4
1	3/4 0 0 0 1/4 0
1	1/2 0 0 0 1/2 0 0
	1/2 0 0 0 1/2 0 0 0

slow method
as formally constructed

0	
1/2	1/2
1/2	1/4 1/4
1	1/4 1/4 1/2
1	1/4 1/4 1/4 1/4
1	1/4 1/4 1/4 1/4 0
1	1/4 1/4 1/4 1/4 0 0
1	1/4 1/4 1/4 1/4 0 0 0
	1/4 1/4 1/4 1/4 0 0 0 0

fast method
as formally constructed

0	
1/2	1/2
1/2	1/2 0
1	1 0 0
1	1 0 0 0
	1/2 0 0 0 1/2

slow method
without redundant stages

0	
1/2	1/2
1/2	1/4 1/4
1	1/4 1/4 1/2
1	1/4 1/4 1/4 1/4
	1/4 1/4 1/4 1/4 0

fast method
without redundant stages

Table 3.2: Example for a second order multirate method constructed with RFSMR.

0				
1/2	1/2			
1/2	-1/6	2/3		
1	1/3	-1/3	1	
	1/6	1/3	1/3	1/6

base method (outer and inner)
(RK43)

0										
1/4	1/4									
1/4	1/4	0								
1/2	1/2	0	0							
1/2	1/2	0	0	0						
1/2	-1/6	0	0	0	2/3					
3/4	1/12	0	0	0	1/6	1/2				
3/4	1/12	0	0	0	1/6	1/2	0			
1	1/3	0	0	0	-1/3	1	0	0		
1	1/3	0	0	0	-1/3	1	0	0	0	
	1/6	0	0	0	1/3	1/3	0	0	0	1/6

slow method

0										
1/4	1/4									
1/4	-1/12	1/3								
1/2	1/6	-1/6	1/2							
1/2	1/12	1/6	1/6	1/12						
1/2	1/12	1/6	1/6	1/12	0					
3/4	1/12	1/6	1/6	1/12	0	1/4				
3/4	1/12	1/6	1/6	1/12	0	-1/12	1/3			
1	1/12	1/6	1/6	1/12	0	1/6	-1/6	1/2		
1	1/12	1/6	1/6	1/12	0	1/12	1/6	1/6	1/12	
	1/12	1/6	1/6	1/12	0	1/12	1/6	1/6	1/12	0

fast method

Table 3.3: Example for a third order multirate method constructed with RFSMR.

method as formally constructed and an equivalent method without redundancies. The remaining Butcher tableaus of the PRKs shall be given in non-redundant form if not noted otherwise.

To construct a multirate PRK with an additional temporal refinement level we may employ one of the formerly constructed methods as outer method in a further construction step. Denoting the inner and outer method in this second construction step by a superscript “I2” and “O2” and following (3.5) we find the parameters of the new method to read

$$\begin{aligned} a_{(i,k),(j,l)}^{(3)} &= \begin{cases} 0 & \text{if } j > i \\ (c_{j+1}^{\text{O2}} - c_j^{\text{O2}}) a_{k,l}^{\text{I2}} & \text{if } j = i \\ (c_{j+1}^{\text{O2}} - c_j^{\text{O2}}) b_l^{\text{I2}} & \text{if } j < i \end{cases}, \\ b_{(j,l)}^{(3)} &= (c_{j+1}^{\text{O2}} - c_j^{\text{O2}}) b_l^{\text{I2}}, \\ c_{(i,k)}^{(3)} &= c_i^{\text{O2}} + c_k^{\text{I2}} (c_{i+1}^{\text{O2}} - c_i^{\text{O2}}). \end{aligned}$$

Since the new method $(A^{(3)}, b^{(3)}, c^{(3)})$ depends on the outer method only via its nodes c^{O2} and those nodes are equal for both outer methods in consideration (i.e. $c^{\text{O2}} = c^{(1)} = c^{(2)}$) we see that method $(A^{(3)}, b^{(3)}, c^{(3)})$ can be constructed equivalently via both paths. Note that this construction also involves construction of a method $(A^{(1a)}, b^{(1a)}, c^{(1a)})$ equivalent to $(A^{(1)}, b^{(1)}, c^{(1)})$ but with the nodes $c^{(1a)} = c^{(3)}$ and the construction of a method $(A^{(2a)}, b^{(2a)}, c^{(2a)})$ equivalent to $(A^{(2)}, b^{(2)}, c^{(2)})$ but with the nodes $c^{(2a)} = c^{(3)}$; an illustration of this is given in Figure 3.2. Further temporal refinement levels can be added analogously. The Butcher tableaus of methods constructed in the first stages of such a construction cascade are shown exemplarily in Table 3.4.

It can be shown that the partitioned methods constructed via this approach are second order accurate in time for second order base methods. Even third order accuracy in time can be obtained if the methods satisfy the classical third order conditions and one additional condition. An example of a third order multirate method is given in Table 3.3. For details refer to Section 3.3.

3.1.3 Suitability of base methods

In this section we shall discuss which base methods are suitable for the construction of multirate schemes with a specific time step ratio. We shall consider the general splitting as sketched in Section 3.1.1 and closely examine the nodes of the outer explicit method. The inner system (3.2) has to be solved for every stage i with $c_i \neq c_{i-1}$, so that the total interval integrated with the inner method is given by

$$\Delta t_{\text{inner}} = \Delta t \sum_{i=2}^{s+1} |c_i - c_{i-1}|.$$

Consequently outer methods with monotonically increasing nodes are computationally efficient in the sense that $\Delta t_{\text{inner}} = \Delta t$.

	"slow" equivalent to base method	"fast" equivalent to two base method half steps	"faster" equivalent to four base method quarter steps
base method	$\begin{array}{c c} 0 & \\ \hline 1/2 & 1/2 \\ \hline 0 & 1 \end{array}$		
first splitting level	$\begin{array}{c c} 0 & \\ \hline 1/4 & 1/4 \\ 2/4 & 2/4 & 0 \\ \hline 3/4 & 1/4 & 0 & 1/2 \\ \hline 0 & 0 & 1 & 0 \end{array}$	$\begin{array}{c c} 0 & \\ \hline 1/4 & 1/4 \\ 2/4 & 0 & 1/2 \\ \hline 3/4 & 0 & 1/2 & 1/4 \\ \hline 0 & 1/2 & 0 & 1/2 \end{array}$	
second splitting level	$\begin{array}{c c} 0 & \\ \hline 1/8 & 1/8 \\ 2/8 & 2/8 & 0 \\ \hline 3/8 & 3/8 & 0 & 0 \\ 4/8 & 4/8 & 0 & 0 & 1/4 \\ \hline 5/8 & 3/8 & 0 & 0 & 1/4 \\ 6/8 & 2/8 & 0 & 0 & 2/4 & 0 \\ \hline 7/8 & 1/8 & 0 & 0 & 3/4 & 0 & 0 \end{array}$	$\begin{array}{c c} 0 & \\ \hline 1/8 & 1/8 \\ 2/8 & 2/8 & 0 \\ \hline 3/8 & 1/8 & 0 & 1/4 \\ 4/8 & 0 & 0 & 1/2 & 0 \\ \hline 5/8 & 0 & 0 & 1/2 & 0 & 1/8 \\ 6/8 & 0 & 0 & 1/2 & 0 & 2/8 & 0 \\ \hline 7/8 & 0 & 0 & 1/2 & 0 & 1/8 & 0 & 1/4 \end{array}$	$\begin{array}{c c} 0 & \\ \hline 1/8 & 1/8 \\ 2/8 & 0 & 1/4 \\ \hline 3/8 & 0 & 1/4 & 1/8 \\ 4/8 & 0 & 1/4 & 0 & 1/4 \\ \hline 5/8 & 0 & 1/4 & 0 & 1/4 & 1/8 \\ 6/8 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 \\ \hline 7/8 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 1/8 \end{array}$
:	:	:	:
			...

Table 3.4: Example of a construction cascade; base method employed both as outer and inner method.

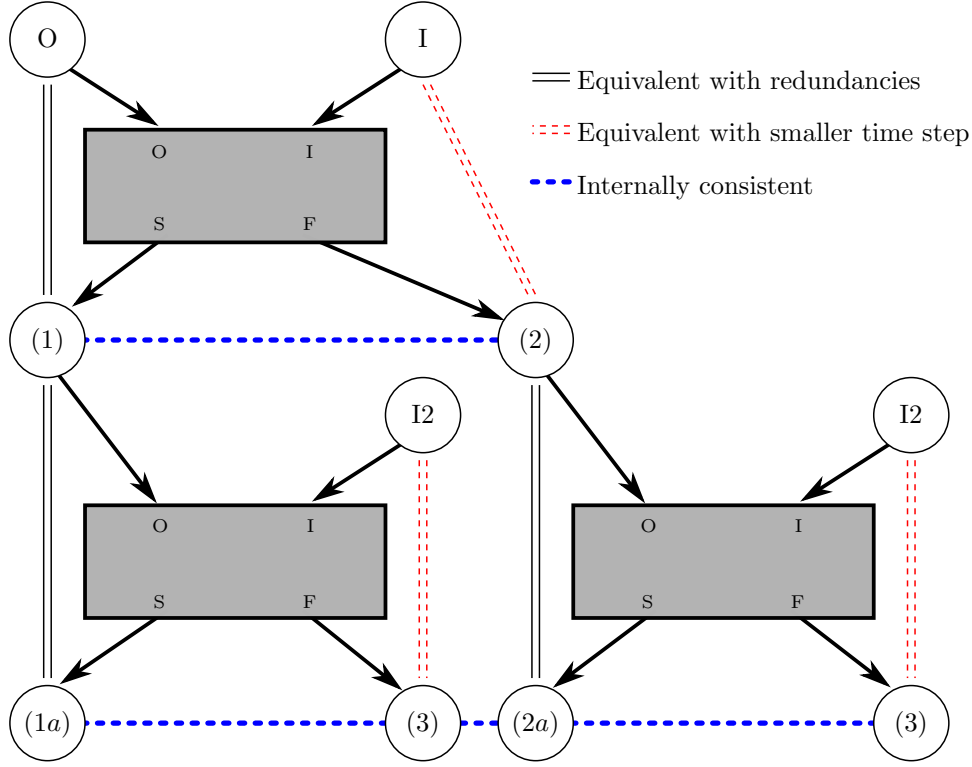


Figure 3.2: Top-down construction of a tripartite PRK. The grey boxes indicate the construction with inputs O and I for outer and inner method and outputs S and F for slow and fast method.

In the context of multirate methods we aim at a certain time step ratio R_{aim} . More precisely if we consider an ODE

$$\dot{w} = F(w) + G(w),$$

the system $\dot{w} = G(w)$ can be integrated with the outer base method and a time step smaller than $\Delta t_{\text{max}}^{\text{O}}$ while satisfying some stability constraints or accuracy demands. Similarly the inner system $\dot{w} = r + F(w)$ can be solved with the inner base method and a time step of $\Delta t_{\text{max}}^{\text{I}}$, so that the desired time step ratio is given by

$$R_{\text{aim}} = \frac{\Delta t_{\text{max}}^{\text{O}}}{\Delta t_{\text{max}}^{\text{I}}}.$$

For example in the context of multirate advection G may describe advection on a coarser grid while F describes advection on a finer grid. Assuming a uniform wind field with wind speed u the CFL condition leads to restrictions of the time step size

$$\begin{aligned} \Delta t_{\text{max}}^{\text{O}} &= S \cdot \nu_{\text{max}}^{\text{O}} \frac{\Delta x_{\text{coarse}}}{u}, \\ \Delta t_{\text{max}}^{\text{I}} &= S \cdot \nu_{\text{max}}^{\text{I}} \frac{\Delta x_{\text{fine}}}{u}, \end{aligned}$$

with ν_{\max}^O denoting the maximum Courant number for the employed advection operator and the outer method and ν_{\max}^I denoting the maximum Courant number for the inner method. In practice a safety factor $0 < S < 1$ is employed. If inner and outer method have the same maximum Courant number $\nu_{\max}^O = \nu_{\max}^I$, the time step ratio is determined solely by the ratio of the grid sizes:

$$R_{\text{aim}} = \frac{\Delta t_{\max}^O}{\Delta t_{\max}^I} = \frac{\Delta x_{\text{coarse}}}{\Delta x_{\text{fine}}}.$$

We generalized this splitting by allowing for the inner method applied once with a time step of Δt to be replaced by a composition of n steps of size $\Delta t/n$ each. If we choose this number of substeps individually for each stage of the outer method, the actual time step ratio R_i for stage i of the outer method and the time step ratio R for the complete method are given by

$$\begin{aligned} R_i &= \frac{n_i}{|c_i - c_{i-1}|}, \\ R &= \min_{i \in \{2, \dots, s+1\}} (R_i). \end{aligned}$$

Ensuring that $\forall i : R_i \geq R_{\text{aim}}$ the optimal choice for the n_i reads

$$n_i^{\text{opt}} = \lceil R_{\text{aim}} |c_i - c_{i-1}| \rceil.$$

For the practical implementation of the method we choose a time step ratio and compute values of n_i for each stage of the outer method, see Section 4.3. This also implies that the inner base method has to be applied $N = \sum_i n_i$ times in the course of one macro time step. We call a splitting *efficient in a general sense* if an increase of stability by a factor of R is involved with an increase of computational cost by the same factor. Expressed in the time step ratio R and the relative computational cost N :

$$\begin{aligned} R &= N, \\ \min_{i \in \{2, \dots, s+1\}} \left(\frac{n_i}{|c_i - c_{i-1}|} \right) &= \sum_{i=2}^{s+1} n_i. \end{aligned} \tag{3.6}$$

Aiming at a certain time step ratio R_{aim} , we want the actual time step ratio to be $R \geq R_{\text{aim}}$ while the computational cost shall be $N \leq R_{\text{aim}}$. Consequently we call a splitting *efficient with respect to a desired time step ratio* if

$$\begin{aligned} R_{\text{aim}} \leq R &= N \leq R_{\text{aim}}, \\ \min_{i \in \{2, \dots, s+1\}} \left(\frac{n_i^{\text{opt}}}{|c_i - c_{i-1}|} \right) &= R_{\text{aim}} = \sum_{i=2}^{s+1} n_i^{\text{opt}}, \\ \min_{i \in \{2, \dots, s+1\}} \left(\frac{\lceil R_{\text{aim}} |c_i - c_{i-1}| \rceil}{|c_i - c_{i-1}|} \right) &= R_{\text{aim}} = \sum_{i=2}^{s+1} \lceil R_{\text{aim}} |c_i - c_{i-1}| \rceil. \end{aligned} \tag{3.7}$$

If the nodes of the outer method are monotonically increasing, condition (3.7) simplifies to

$$\sum_{i=2}^{s+1} [R_{\text{aim}} |c_i - c_{i-1}|] = R_{\text{aim}},$$

which can equivalently be expressed as

$$\forall i \in \{2, \dots, s+1\} : R_{\text{aim}} |c_i - c_{i-1}| \in \mathbb{N}.$$

If on the other hand the nodes of the outer method are not monotonically increasing the method is inefficient since

$$\sum_{i=2}^{s+1} [R_{\text{aim}} |c_i - c_{i-1}|] > R_{\text{aim}}.$$

To illustrate and clarify we consider the general second order explicit Runge–Kutta method with two stages, whose Butcher tableau is given by

$$\begin{array}{c|cc} 0 & & \\ \hline c_2 & c_2 & \\ \hline & 1 - \frac{1}{2c_2} & \frac{1}{2c_2} \end{array}.$$

If we choose $c_2 = 1/3$, we obtain a method which is efficient in a general sense for $(n_2, n_3) = (2, 1)$, as

$$\min_{i \in \{2, \dots, s+1\}} \left(\frac{n_i}{|c_i - c_{i-1}|} \right) = \min \left\{ \frac{1}{1/3}, \frac{2}{2/3} \right\} = 3 = \sum_{i=2}^{s+1} n_i.$$

The method also is efficient with respect to desired time step ratios of $R_{\text{aim}} = 3k$, $k \in \mathbb{N}$, but it is not efficient with respect to a desired time step ratio of $R_{\text{aim}} = 2$ since the inner method has to be applied N times with

$$N = \sum_{i=2}^{s+1} [R_{\text{aim}} |c_i - c_{i-1}|] = 3 > R_{\text{aim}}.$$

Following the same rationale we find that the two stage second order method obtained with $c_2 = 1$ (RK2a) is efficient with respect to a desired time step ratio of $R_{\text{aim}} \in \mathbb{N}$, while the method constructed with $c_2 = 1/2$ (RK2b) is efficient with respect to a desired time step ratio of $R_{\text{aim}} = 2k$, $k \in \mathbb{N}$.

3.2 Spatial aspects of the splitting

3.2.1 Decomposition of the advection operator

In order to apply a multirate method to the advection equation, the advection operator must be partitioned. Opposed to the splitting paradigms discussed in Section 2.3.1 the

splitting approaches we shall describe now actually yield different results. Generally there are two alternative approaches, the so called *splitting by cells* and the less commonly employed *splitting by fluxes*, also called *splitting by faces* [22]. We consider a semidiscrete transport equation in flux formulation

$$\dot{w}_k = -\frac{f_{k+1/2}(w) - f_{k-1/2}(w)}{\Delta x_k}$$

and a partitioning of the indexes

$$K = K_1 \cup K_2; \quad K_1 \cap K_2 = \emptyset.$$

The basic idea of a splitting by cells is to assign each cell to a partition. This naturally follows from the splitting by components paradigm.

$$\begin{aligned} \dot{w} &= \bar{F}^{(\text{cell},1)}(w) + \bar{F}^{(\text{cell},2)}(w), \\ \bar{F}_k^{(\text{cell},i)}(w) &= \begin{cases} -\frac{f_{k+1/2}(w) - f_{k-1/2}(w)}{\Delta x_k} & k \in K_i, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

For a splitting by faces, each face (or more generally: each cell boundary) is assigned to a partition. Naively each face could be statically assigned. For a practical implementation however it is advantageous to dynamically assign the faces such that the fluxes leaving a statically defined set of cells are assigned to a partition:

$$\begin{aligned} \dot{w} &= \bar{F}^{(\text{face},1)}(w) + \bar{F}^{(\text{face},2)}(w), \\ \bar{F}_k^{(\text{face},i)}(w) &= -\frac{\bar{f}_{k+1/2}^{(\text{face},i)}(w) - \bar{f}_{k-1/2}^{(\text{face},i)}(w)}{\Delta x_k}, \\ \bar{f}_{k+1/2}^{(\text{face},i)}(w) &= \begin{cases} f_{k+1/2}(w) & k \in K_i \wedge f_{k+1/2} > 0 \vee k+1 \in K_i \wedge f_{k+1/2} < 0, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

In combination with upwind discretizations as (2.4) and (2.7) this makes the set of cells needed for the calculation of $\bar{F}^{(\text{face},i)}$ independent from the transport direction, see Figure 3.3.

A further advantage of a splitting by faces is that mass conservation is guaranteed when the partitioned system is solved with a PRK. However as shown by Hundsdorfer et al. [22], a splitting by fluxes cannot lead to a consistent discretization when combined with a *true* multirate scheme. Numerical tests in Section 5.1 match these theoretical predictions for high accuracies while no influence can be found for the large Courant numbers which are of special interest in the context of atmospheric simulations.

3.2.2 Decompositions of advection-reaction systems

We now consider a differential equation with a tripartite right hand side:

$$\dot{w} = G^{(1)}(w) + G^{(2)}(w) + F(w),$$

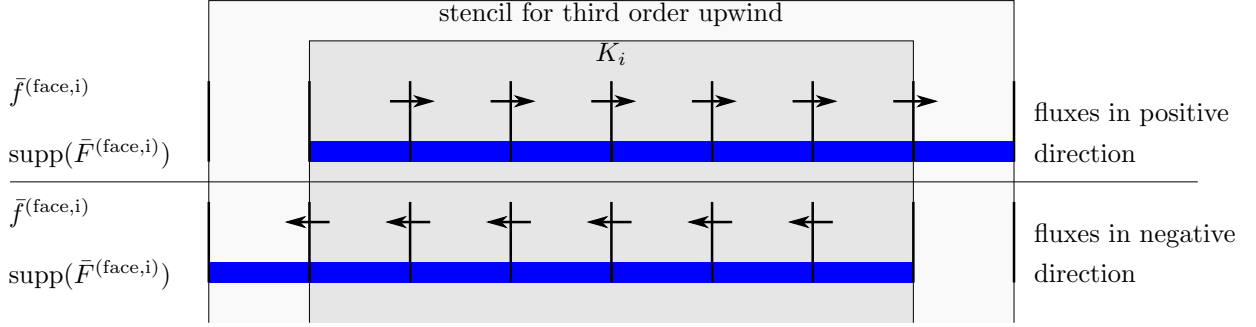


Figure 3.3: Illustration of stencil for a splitting by fluxes.

where $G^{(1)}(w)+G^{(2)}(w)$ represents advection split by fluxes and $F(w)$ represents a reaction term. We start by defining explicit integrators associated with $G^{(1)}$ and $G^{(2)}$. In terms of the inner and outer explicit base methods the resulting method then can be cast as

$$V_{1,s+1} = w_0, \quad (3.8a)$$

$$r_{i,k} = \sum_{j=1}^{i-1} (c_k^I - c_{k-1}^I) (a_{ij}^O - a_{i-1,j}^O) G^{(1)}(V_{j,s^I+1}) + \sum_{j=1}^{k-1} (c_i^O - c_{i-1}^O) (a_{kj}^I - a_{k-1,j}^I) G^{(2)}(V_{i,j}), \quad (3.8b)$$

$$v_{i,k}(0) = V_{i,k-1}, \quad (3.8c)$$

$$\frac{dv_{i,k}}{d\tau} = \Delta t r_{i,k} + \Delta t (c_i^O - c_{i-1}^O) (c_k^I - c_{k-1}^I) F(v_{i,k}), \quad \tau \in [0, 1], \quad (3.8d)$$

$$i = 2, \dots, s^O + 1, \quad k = 2, \dots, s^I + 1, \quad (3.8e)$$

$$V_{i,k} = v_{i,k}(1), \quad (3.8f)$$

$$w_1 = V_{s^O+1, s^I+1}. \quad (3.8g)$$

Note that the definition of the source term (3.8b) follows directly from the stage increment in the multirate case as defined in (3.3e). The algorithm can be illustrated as shown in Figure 3.5.

We shall now consider the case that $G^{(2)}$ is equal to zero. Source term and differential equation per stage then read

$$r_{i,k} = \sum_{j=1}^{i-1} (c_k^I - c_{k-1}^I) (a_{ij}^O - a_{i-1,j}^O) G^{(1)}(V_{j,s^I+1}),$$

$$v_{i,k}(0) = V_{i,k-1},$$

$$\frac{dv_{i,k}}{d\tau} = \Delta t r_{i,k} + \Delta t (c_i^O - c_{i-1}^O) (c_k^I - c_{k-1}^I) F(v_{i,k}), \quad \tau \in [0, 1],$$

$$V_{i,k} = v_{i,k}(1),$$

The only dependency of the source term $r_{i,k}$ from the parameters of the inner method is given by a factor of $(c_k^I - c_{k-1}^I)$. By transforming the above equations we obtain an equivalent formulation with a source term independent of the micro stage index k :

$$\begin{aligned} r_i &= \sum_{j=1}^{i-1} (a_{ij}^O - a_{i-1,j}^O) G^{(1)}(V_{j,s^{I+1}}), \\ v_{i,k}(c_{k-1}^I) &= V_{i,k-1}, \\ \frac{dv_{i,k}}{d\tau} &= \Delta t r_i + \Delta t (c_i^O - c_{i-1}^O) F(v_{i,k}), \tau \in [c_{k-1}^I, c_k^I], \\ V_{i,k} &= v_{i,k}(c_k^I), \end{aligned}$$

which can be further simplified by joining the integration intervals to read

$$v_i(c_1^I) = V_{i,1}, \tag{3.9a}$$

$$\frac{dv_i}{d\tau} = \Delta t r_i + \Delta t (c_i^O - c_{i-1}^O) F(v_i), \tau \in [c_1^I, c_{s^{I+1}}^I], \tag{3.9b}$$

$$V_{i,s^{I+1}} = v_i(c_{s^{I+1}}^I). \tag{3.9c}$$

If the inner method is first order consistent we have $c_1^I = 0$, $c_{s^{I+1}}^I = 1$, so that (3.9) is equivalent to the system in complete absence of a faster explicit integrator as given in (3.2). Note that this result only holds if we assume that (3.2b) and (3.9b) respectively are solved exactly. But even if a numerical integrator is employed considerations of the order of convergence still hold.

This observation is of importance if we consider advection-reaction problems. In this context $G^{(1)} + G^{(2)}$ denotes an advection term which is split by fluxes as described in Section 3.2.1, while F denotes a reaction term which has the property that the reaction inside of a cell is independent of other cells in the simulation domain; formally

$$F_n(w) = F_n(w_n),$$

with a spatial index n . The advection terms $G^{(1)}$ and $G^{(2)}$ have finite (though overlapping) supports so that for a subset N of cells we actually have

$$\forall n \in N : G_n^{(2)}(w) \equiv 0.$$

Thus it is straightforward to split $F = F^{(1)} + F^{(2)}$ by cells in such a way that

$$\begin{aligned} \text{supp}(F^{(2)}) &= \text{supp}(G^{(2)}), \\ \text{supp}(F^{(1)}) \cap \text{supp}(F^{(2)}) &= \emptyset, \end{aligned}$$

see also Figure 3.4.

Solving the semidiscretized system

$$\dot{w} = G^{(1)} + F^{(1)} + (G^{(2)} + F^{(2)}),$$

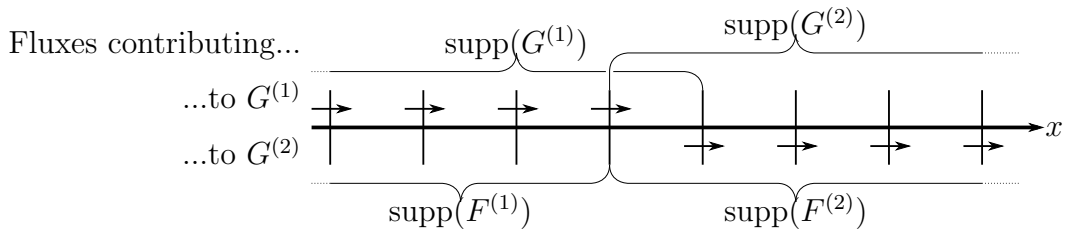


Figure 3.4: Illustration of supports near interface.

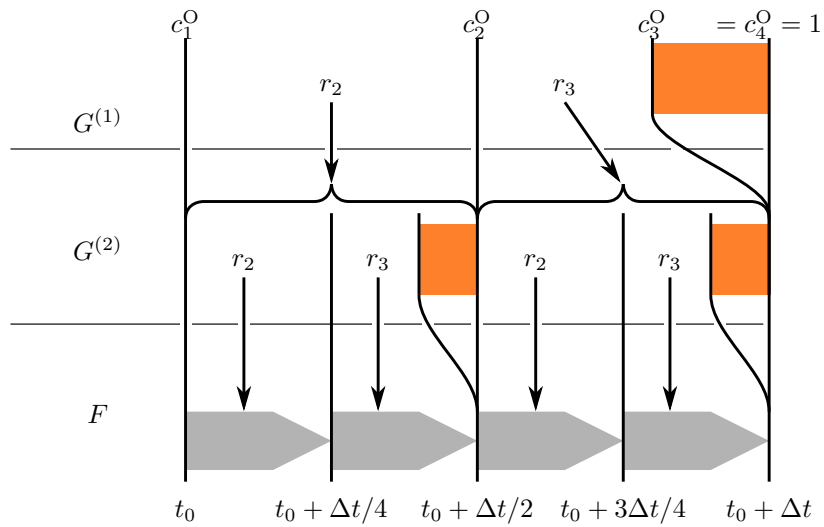


Figure 3.5: Illustration of a recursive time step with bipartite splitting, cf. (3.8). Explicit methods are (RK32). Analogous to Figure 3.1 correction steps are represented by orange boxes.

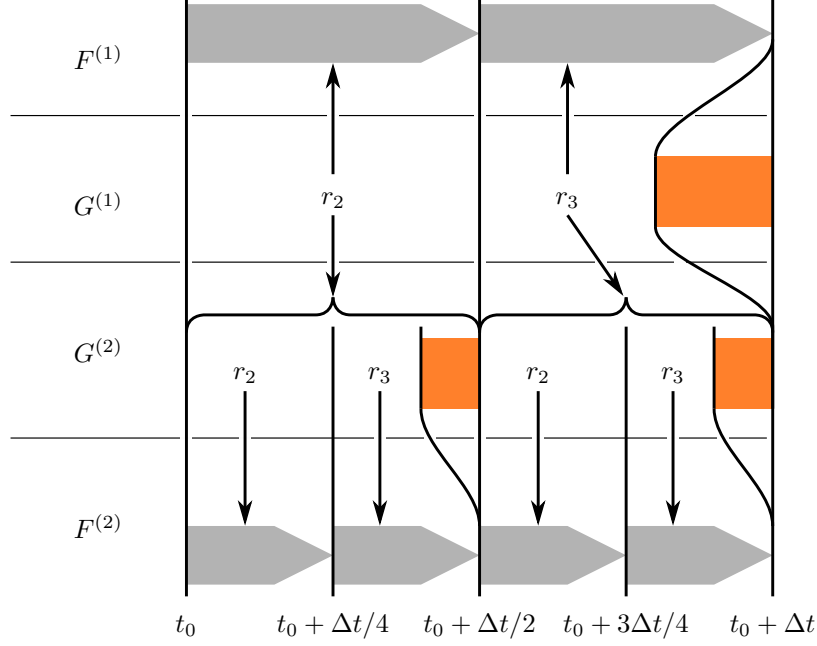


Figure 3.6: Illustration of a recursive time step with tripartite splitting, cf. (3.10). Explicit methods are (RK32).

we exploit the circumstance that $F^{(1)}$ is decoupled from $G^{(2)} + F^{(2)}$ and employ a tripartite splitting:

$$W_1 = \begin{pmatrix} W_1^{(1)} \\ W_1^{(2)} \end{pmatrix} = w_0, \quad (3.10a)$$

$$\begin{pmatrix} r_i^{(1)} \\ r_i^{(2)} \end{pmatrix} = \sum_{j=1}^{i-1} (a_{ij}^O - a_{i-1,j}^O) G^{(1)} \left(\begin{pmatrix} W_j^{(1)} \\ W_j^{(2)} \end{pmatrix} \right), \quad (3.10b)$$

$$\begin{pmatrix} v_i^{(1)}(0) \\ v_1^{(2)}(0) \end{pmatrix} = \begin{pmatrix} W_{i-1}^{(1)} \\ W_{i-1}^{(2)} \end{pmatrix}, \quad (3.10c)$$

$$\frac{dv_i^{(1)}}{d\tau} = \Delta t r_i^{(1)} + \Delta t (c_i^O - c_{i-1}^O) F^{(1)} \left(v_i^{(1)} \right), \quad (3.10d)$$

$$\frac{dv_i^{(2)}}{d\tau} = \Delta t r_i^{(2)} + \Delta t (c_i^O - c_{i-1}^O) \left(G^{(2)} \left(v_i^{(2)} \right) + F^{(2)} \left(v_i^{(2)} \right) \right), \quad (3.10e)$$

$$\tau \in [0, 1], i = 2, \dots, s+1,$$

$$\begin{pmatrix} W_i^{(1)} \\ W_i^{(2)} \end{pmatrix} = \begin{pmatrix} v_i^{(1)}(1) \\ v_1^{(2)}(1) \end{pmatrix}, \quad (3.10f)$$

$$w_1 = \begin{pmatrix} W_{s^O+1}^{(1)} \\ W_{s^O+1}^{(2)} \end{pmatrix}, \quad (3.10g)$$

An illustration of this algorithm is shown in Figure 3.6. Assuming that (3.10d) is solved exactly and (3.10e) is solved in an IMEX manner by recursive application of the same algorithm we obtain equal results as with the recursive bipartite splitting discussed above. Solving (3.10d) numerically yields differences between the bipartite and the tripartite approach due to the different step sizes with which the inner method is employed for cells belonging to the support of $F^{(1)}$.

As the tripartite formulation allows for a solution with fewer costly ODE solutions per cell and timestep, we implement this approach.

3.3 Order conditions

We will now examine the formal order of the generated partitioned method. General order conditions can be found in [17] or explicitly for partitioned Runge–Kutta methods in [24]. For our methods with $c_{(i,k)}^{\text{fast}} = c_{(i,k)}^{\text{slow}}$ most of the *coupling* conditions i.e. order conditions combining the parameters of the different methods are redundant. The only exception is one order three condition (3.11e). The conditions for first (3.11a), second (3.11b) and third order (3.11c),(3.11d),(3.11e) read

$$\sum_{j=1}^s b_j^{\text{slow}} = 1 = \sum_{j=1}^s b_j^{\text{fast}}, \quad (3.11a)$$

$$\sum_{j=1}^s b_j^{\text{slow}} c_j^{\text{slow}} = 1/2 = \sum_{j=1}^s b_j^{\text{fast}} c_j^{\text{fast}}, \quad (3.11b)$$

$$\sum_{j=1}^s b_j^{\text{slow}} (c_j^{\text{slow}})^2 = 1/3 = \sum_{j=1}^s b_j^{\text{fast}} (c_j^{\text{fast}})^2, \quad (3.11c)$$

$$\sum_{i=1}^s \sum_{j=1}^s b_i^{\text{slow}} a_{ij}^{\text{slow}} c_j^{\text{slow}} = 1/6 = \sum_{i=1}^s \sum_{j=1}^s b_i^{\text{fast}} a_{ij}^{\text{fast}} c_j^{\text{fast}}, \quad (3.11d)$$

$$\sum_{i=1}^s \sum_{j=1}^s b_i^{\text{slow}} a_{ij}^{\text{fast}} c_j^{\text{fast}} = 1/6 = \sum_{i=1}^s \sum_{j=1}^s b_i^{\text{fast}} a_{ij}^{\text{slow}} c_j^{\text{slow}}, \quad (3.11e)$$

with $s = s^0 s^1$.

We shall now examine the order of the generated partitioned methods assuming that the underlying base methods satisfy the classical order conditions, i.e. the order conditions for non-partitioned Runge–Kutta methods.

Initially we shall assume the outer method to be explicit, whereas no such assumption will be made for the inner method. First we shall give theorems for second order consistency of the constructed method. Examining the third order conditions we shall obtain an additional condition to be satisfied by the outer method. The thus arising question, if a third order accurate cascade of methods can be constructed shall be addressed in the following section. Concludingly we shall show that the examinations of bipartite PRKs also allow for statements about general m -part PRKs.

For the proofs the methods generated via the recursive flux splitting algorithm will be cast in PRK form as introduced in (3.5):

$$a_{is^{I+k},js^{I+l}}^{\text{slow}} = a_{(i,k),(j,l)}^{\text{slow}} = \begin{cases} a_{i,j}^{\text{O}} + (a_{i+1,j}^{\text{O}} - a_{i,j}^{\text{O}}) c_k^{\text{I}} & \text{if } l = 1 \\ 0 & \text{if } l > 1 \end{cases}, \quad (3.12)$$

$$b_{js^{I+l}}^{\text{slow}} = b_{(j,l)}^{\text{slow}} = \begin{cases} b_j^{\text{O}} & \text{if } l = 1 \\ 0 & \text{if } l > 1 \end{cases}, \quad (3.13)$$

$$c_{is^{I+k}}^{\text{slow}} = c_{(i,k)}^{\text{slow}} = c_i^{\text{O}} + c_k^{\text{I}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}), \quad (3.14)$$

$$a_{is^{I+k},js^{I+l}}^{\text{fast}} = a_{(i,k),(j,l)}^{\text{fast}} = \begin{cases} 0 & \text{if } j > i \\ (c_{j+1}^{\text{O}} - c_j^{\text{O}}) a_{k,l}^{\text{I}} & \text{if } j = i \\ (c_{j+1}^{\text{O}} - c_j^{\text{O}}) b_l^{\text{I}} & \text{if } j < i \end{cases}, \quad (3.15)$$

$$b_{js^{I+l}}^{\text{fast}} = b_{(j,l)}^{\text{fast}} = (c_{j+1}^{\text{O}} - c_j^{\text{O}}) b_l^{\text{I}}, \quad (3.16)$$

$$c_{is^{I+k}}^{\text{fast}} = c_{(i,k)}^{\text{fast}} = c_i^{\text{O}} + c_k^{\text{I}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) = c_{(i,k)}^{\text{slow}}. \quad (3.17)$$

3.3.1 Order conditions up to order 2

Theorem 3.3.1. *If the inner and the outer method satisfy the conditions for order $p = 1$, the methods generated via RFSMR satisfy the conditions for order $p = 1$ (3.11a).*

Proof.

$$\begin{aligned} \sum_{j=1}^{s^{\text{O}} s^{\text{I}}} b_j^{\text{slow}} &= \sum_{j=1}^{s^{\text{O}}} b_j^{\text{O}} = 1. \\ \sum_{j=1}^{s^{\text{O}} s^{\text{I}}} b_j^{\text{fast}} &= \sum_{j=1}^{s^{\text{O}}} \sum_{k=1}^{s^{\text{I}}} (c_{j+1}^{\text{O}} - c_j^{\text{O}}) b_k^{\text{I}} = \underbrace{\left(\sum_{j=1}^{s^{\text{O}}} (c_{j+1}^{\text{O}} - c_j^{\text{O}}) \right)}_{=c_{s^{\text{O}+1}^{\text{O}}}=1} \left(\sum_{k=1}^{s^{\text{I}}} b_k^{\text{I}} \right) \\ &= \underbrace{\left(\sum_{j=1}^{s^{\text{O}}} b_j^{\text{O}} \right)}_{=c_{s^{\text{O}+1}^{\text{O}}}=1} \left(\sum_{k=1}^{s^{\text{I}}} b_k^{\text{I}} \right) = 1. \end{aligned}$$

□

Theorem 3.3.2. *If the inner and the outer method have order of consistency $p = 2$, the methods generated via RFSMR satisfy the conditions for order $p = 2$ (3.11b).*

Proof. For the slow method:

$$\sum_{j=1}^{s^{\text{O}} s^{\text{I}}} b_j^{\text{slow}} c_j^{\text{slow}} = \sum_{j=1}^{s^{\text{O}}} b_j^{\text{O}} c_j^{\text{O}} = \frac{1}{2}.$$

For the fast method:

$$\begin{aligned}
\sum_{j=1}^{s^O s^I} b_j^{\text{fast}} c_j^{\text{fast}} &= \sum_{j=1}^{s^O} \sum_{k=1}^{s^I} (c_{j+1}^O - c_j^O) b_k^I \cdot (c_j^O + (c_{j+1}^O - c_j^O) c_k^I) \\
&= \sum_{j=1}^{s^O} \left(\sum_{k=1}^{s^I} c_j^O (c_{j+1}^O - c_j^O) b_k^I + (c_{j+1}^O - c_j^O)^2 \sum_{k=1}^{s^I} b_k^I c_k^I \right) \\
&= \sum_{j=1}^{s^O} \left(c_j^O (c_{j+1}^O - c_j^O) \underbrace{\sum_{k=1}^{s^I} b_k^I}_{=1} + (c_{j+1}^O - c_j^O)^2 \underbrace{\sum_{k=1}^{s^I} b_k^I c_k^I}_{=1/2} \right) \\
&= \sum_{j=1}^{s^O} \left((c_{j+1}^O c_j^O - (c_j^O)^2) + \frac{1}{2} \left((c_{j+1}^O)^2 - 2c_{j+1}^O c_j^O + (c_j^O)^2 \right) \right) \\
&= \frac{1}{2} \sum_{j=1}^{s^O} (c_{j+1}^O)^2 - (c_j^O)^2 = \frac{1}{2} (c_{s^O+1}^O)^2 = \frac{1}{2}.
\end{aligned}$$

□

3.3.2 Order conditions for order 3

We shall now examine the third order conditions (3.11c),(3.11d) for the constructed slow and fast methods under the assumption that both the inner and outer method have order of consistency $p = 3$.

Condition (3.11c) for the slow method:

$$\sum_{j=1}^{s^O s^I} b_j^{\text{slow}} (c_j^{\text{slow}})^2 = \sum_{j=1}^{s^O} b_j^O (c_j^O)^2 = \frac{1}{3}.$$

Condition (3.11c) for the fast method:

$$\begin{aligned}
\sum_{j=1}^{s^O s^I} b_j^{\text{fast}} (c_j^{\text{fast}})^2 &= \sum_{j=1}^{s^O} \sum_{k=1}^{s^I} (c_{j+1}^O - c_j^O) b_k^I \cdot (c_j^O + (c_{j+1}^O - c_j^O) c_k^I)^2 \\
&= \sum_{j=1}^{s^O} \sum_{k=1}^{s^I} (c_{j+1}^O - c_j^O) b_k^I \cdot \left((c_j^O)^2 + 2c_j^O (c_{j+1}^O - c_j^O) c_k^I + ((c_{j+1}^O - c_j^O) c_k^I)^2 \right) \\
&= \sum_{j=1}^{s^O} \sum_{k=1}^{s^I} \left((c_j^O)^2 (c_{j+1}^O - c_j^O) b_k^I + 2c_j^O (c_{j+1}^O - c_j^O)^2 c_k^I b_k^I \right. \\
&\quad \left. + (c_{j+1}^O - c_j^O)^3 (c_k^I)^2 b_k^I \right) \\
&= \sum_{j=1}^{s^O} \left((c_j^O)^2 (c_{j+1}^O - c_j^O) \underbrace{\sum_{k=1}^{s^I} b_k^I}_{=1} + 2c_j^O (c_{j+1}^O - c_j^O)^2 \underbrace{\sum_{k=1}^{s^I} c_k^I b_k^I}_{=1/2} \right. \\
&\quad \left. + (c_{j+1}^O - c_j^O)^3 \underbrace{\sum_{k=1}^{s^I} (c_k^I)^2 b_k^I}_{=1/3} \right) \\
&= \sum_{j=1}^{s^O} \left(\frac{1}{3} (c_{j+1}^O)^3 - \frac{1}{3} (c_j^O)^3 \right) = \frac{1}{3} \underbrace{(c_{s^O+1}^O)^3}_{=1} = \frac{1}{3}.
\end{aligned}$$

Condition (3.11d) for the slow method:

$$\begin{aligned}
\sum_{i=1}^{s^O s^I} \sum_{j=1}^{s^O s^I} b_i^{\text{slow}} a_{ij}^{\text{slow}} c_j^{\text{slow}} &= \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{j=1}^{s^O} \sum_{l=1}^{s^I} b_{(i,k)}^{\text{slow}} a_{(i,k),(j,l)}^{\text{slow}} c_{(j,l)}^{\text{slow}} \\
&= \sum_{i=1}^{s^O} \sum_{j=1}^{s^O} b_{(i,1)}^{\text{slow}} a_{(i,1),(j,1)}^{\text{slow}} c_{(j,1)}^{\text{slow}} \\
&= \sum_{i=1}^{s^O} \sum_{j=1}^{s^O} b_i^O a_{ij}^O c_j^O = 1/6.
\end{aligned}$$

Condition (3.11d) for the fast method; note that assuming the outer method to be explicit

we may split the above summation in two parts with $j \leq i$ and $j = i$ respectively:

$$\begin{aligned}
& \sum_{i=1}^{s^O s^I} \sum_{j=1}^{s^O s^I} b_i^{\text{fast}} a_{ij}^{\text{fast}} c_j^{\text{fast}} \\
&= \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{j=1}^{s^O} \sum_{l=1}^{s^I} b_{(i,k)}^{\text{fast}} a_{(i,k),(j,l)}^{\text{fast}} c_{(j,l)}^{\text{fast}} \\
&= \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{j=1}^{i-1} \sum_{l=1}^{s^I} (c_{i+1}^O - c_i^O) b_k^I (c_{j+1}^O - c_j^O) b_l^I (c_j^O + (c_{j+1}^O - c_j^O) c_l^I) \\
&\quad + \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{l=1}^{s^I} (c_{i+1}^O - c_i^O) b_k^I (c_{i+1}^O - c_i^O) a_{k,l}^I (c_i^O + (c_{i+1}^O - c_i^O) c_l^I) \\
&= \underbrace{\left(\sum_{k=1}^{s^I} b_k^I \right)}_{=1} \sum_{i=1}^{s^O} (c_{i+1}^O - c_i^O) \sum_{j=1}^{i-1} \sum_{l=1}^{s^I} (c_{j+1}^O - c_j^O) b_l^I (c_j^O + (c_{j+1}^O - c_j^O) c_l^I) \\
&\quad + \sum_{i=1}^{s^O} \left(c_i^O (c_{i+1}^O - c_i^O)^2 \underbrace{\sum_{k=1}^{s^I} \sum_{l=1}^{s^I} b_k^I a_{k,l}^I}_{=1/2} + (c_{i+1}^O - c_i^O)^3 \underbrace{\sum_{k=1}^{s^I} \sum_{l=1}^{s^I} b_k^I a_{k,l}^I c_l^I}_{=1/6} \right) \\
&= \sum_{i=1}^{s^O} (c_{i+1}^O - c_i^O) \sum_{j=1}^{i-1} \left(\underbrace{(c_{j+1}^O c_j^O - (c_j^O)^2)}_{=1} \sum_{l=1}^{s^I} b_l^I + \left((c_{j+1}^O)^2 - 2c_{j+1}^O c_j^O + (c_j^O)^2 \right) \underbrace{\sum_{l=1}^{s^I} b_l^I c_l^I}_{=1/2} \right) \\
&\quad + \sum_{i=1}^{s^O} \left(\frac{1}{3} (c_i^O)^3 + \frac{1}{6} (c_{i+1}^O)^3 - \frac{1}{2} c_{i+1}^O (c_i^O)^2 \right) \\
&= \sum_{i=1}^{s^O} (c_{i+1}^O - c_i^O) \sum_{j=1}^{i-1} \left(\frac{1}{2} (c_{j+1}^O)^2 - \frac{1}{2} (c_j^O)^2 \right) \\
&\quad + \sum_{i=1}^{s^O} \left(\frac{1}{3} (c_i^O)^3 + \frac{1}{6} (c_{i+1}^O)^3 - \frac{1}{2} c_{i+1}^O (c_i^O)^2 \right) \\
&= \sum_{i=1}^{s^O} \frac{1}{2} (c_{i+1}^O - c_i^O) (c_i^O)^2 + \sum_{i=1}^{s^O} \left(\frac{1}{3} (c_i^O)^3 + \frac{1}{6} (c_{i+1}^O)^3 - \frac{1}{2} c_{i+1}^O (c_i^O)^2 \right) \\
&= \sum_{i=1}^{s^O} \left(\frac{1}{6} (c_{i+1}^O)^3 - \frac{1}{6} (c_i^O)^3 \right) = \frac{1}{6} \underbrace{(c_{s^O+1}^O)^3}_{=1} = \frac{1}{6}.
\end{aligned}$$

Trying to solve the third order coupling conditions (3.11e) for the slow/fast partitioned method we obtain additional order conditions to be satisfied by the outer method. One of the coupling conditions is satisfied for any pair of third order base methods¹:

$$\begin{aligned}
& \sum_{i=1}^{s^O s^I} \sum_{j=1}^{s^O s^I} b_i^{\text{slow}} a_{ij}^{\text{fast}} c_j^{\text{fast}} \\
= & \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{j=1}^{s^O} \sum_{l=1}^{s^I} b_{(i,k)}^{\text{slow}} a_{(i,k),(j,l)}^{\text{fast}} c_{(j,l)}^{\text{fast}} \\
= & \left[\sum_{i=1}^{s^O} \sum_{j=1}^{i-1} \sum_{l=1}^{s^I} b_{(i,1)}^{\text{slow}} a_{(i,1),(j,l)}^{\text{fast}} c_{(j,l)}^{\text{fast}} \right] + \left[\sum_{i=1}^{s^O} \sum_{l=1}^{s^I} b_{(i,1)}^{\text{slow}} a_{(i,1),(i,l)}^{\text{fast}} c_{(i,l)}^{\text{fast}} \right] \\
= & \left[\sum_{i=1}^{s^O} \sum_{j=1}^{i-1} \sum_{l=1}^{s^I} b_i^O (c_{j+1}^O - c_j^O) b_l^I (c_j^O + c_l^I (c_{j+1}^O - c_j^O)) \right] \\
& + \left[\sum_{i=1}^{s^O} \sum_{l=1}^{s^I} b_i^O (c_{i+1}^O - c_i^O) \underbrace{a_{1,l}^I}_{=0} (c_i^O + c_l^I (c_{i+1}^O - c_i^O)) \right] \\
= & \sum_{i=1}^{s^O} b_i^O \left(\sum_{j=1}^{i-1} \underbrace{\left(\sum_{l=1}^{s^I} b_l^I \right)}_{=1} c_j^O (c_{j+1}^O - c_j^O) + \underbrace{\left(\sum_{l=1}^{s^I} b_l^I c_l^I \right)}_{=1/2} \left((c_{j+1}^O)^2 - 2c_{j+1}^O c_j^O + (c_j^O)^2 \right) \right) \\
= & \frac{1}{2} \sum_{i=1}^{s^O} b_i^O \sum_{j=1}^{i-1} \left((c_{j+1}^O)^2 - (c_j^O)^2 \right) \\
= & \frac{1}{2} \underbrace{\sum_{i=1}^{s^O} b_i^O (c_i^O)^2}_{=1/3} = \frac{1}{6}.
\end{aligned}$$

¹Actually for this calculation, only second order assumptions for the inner method are employed.

Solving for the other third order coupling condition we obtain

$$\begin{aligned}
& \sum_{i=1}^{s^O} \sum_{j=1}^{s^I} b_i^{\text{fast}} a_{ij}^{\text{slow}} c_j^{\text{slow}} \\
&= \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{j=1}^{s^O} \sum_{l=1}^{s^I} b_{(i,k)}^{\text{fast}} a_{(i,k),(j,l)}^{\text{slow}} c_{(j,l)}^{\text{slow}} \\
&= \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{j=1}^i b_{(i,k)}^{\text{fast}} a_{(i,k),(j,1)}^{\text{slow}} c_{(j,1)}^{\text{slow}} \\
&= \sum_{i=1}^{s^O} \sum_{k=1}^{s^I} \sum_{j=1}^i (c_{i+1}^O - c_i^O) b_k^I (a_{i,j}^O + (a_{i+1,j}^O - a_{i,j}^O) c_k^I) \left(c_j^O + \underbrace{c_1^I}_{=0} (c_{j+1}^O - c_j^O) \right) \\
&= \sum_{i=1}^{s^O} \sum_{j=1}^i a_{i,j}^O (c_{i+1}^O - c_i^O) c_j^O \underbrace{\left(\sum_{k=1}^{s^I} b_k^I \right)}_{=1} + (a_{i+1,j}^O - a_{i,j}^O) (c_{i+1}^O - c_i^O) c_j^O \underbrace{\left(\sum_{k=1}^{s^I} b_k^I c_k^I \right)}_{=1/2} \\
&= \sum_{i=1}^{s^O} \sum_{j=1}^i \left(a_{i,j}^O + \frac{1}{2} (a_{i+1,j}^O - a_{i,j}^O) \right) (c_{i+1}^O - c_i^O) c_j^O \\
&= \frac{1}{2} \sum_{i=1}^{s^O} (c_{i+1}^O - c_i^O) \sum_{j=1}^i (a_{i+1,j}^O + a_{i,j}^O) c_j^O = \frac{1}{6}. \tag{3.18}
\end{aligned}$$

This leads to the following theorem:

Theorem 3.3.3. *If the inner and the outer method have order of consistency $p = 3$ and the outer method additionally satisfies*

$$\sum_{i=1}^{s^O} (c_{i+1}^O - c_i^O) \sum_{j=1}^i (a_{i+1,j}^O + a_{i,j}^O) c_j^O = 1/3, \tag{3.19}$$

the methods generated via RFSMR satisfy the conditions for order $p = 3$ (3.11c), (3.11d).

Note that the additional condition (3.19) to be satisfied by the outer method is equivalent to the additional third order condition introduced by Knoth and Wolke in [29].

3.3.3 Order conditions for recursive application

Up to this point we considered the construction of *dualrate* methods, i.e. multirate methods with exactly two levels of refinement or IMEX methods if the inner method is an implicit Runge–Kutta method. Additional levels can be introduced by employing a formerly generated method as outer method and thus applying the splitting recursively. As

presented in Section 3.2.2 the new method can be constructed alternatively by applying either of the formally constructed methods as outer method. Consequently by showing that *either* one of the potential outer methods satisfies the conditions for outer methods occurring in Theorems 3.3.1, 3.3.2, 3.3.3 we show that *both* of the potential outer methods satisfy those conditions. We choose to examine the slow method. From Theorems 3.3.1 and 3.3.2 we know that a generated partitioned method $[(A^{\text{slow}}, b^{\text{slow}}, c^{\text{slow}}), (A^{\text{fast}}, b^{\text{fast}}, c^{\text{fast}})]$ is consistent of order $p \leq 2$ if both the outer method $(A^{\text{O}}, b^{\text{O}}, c^{\text{O}})$ and the inner method $(A^{\text{I}}, b^{\text{I}}, c^{\text{I}})$ employed for construction are consistent of order p . From Theorem 3.3.3 we know that the generated partitioned method is consistent of order $p = 3$ if both base methods employed for construction are third order consistent and the outer method $(A^{\text{O}}, b^{\text{O}}, c^{\text{O}})$ additionally satisfies (3.19).

To show third order consistency after the next construction step it remains to be shown that $(A^{\text{slow}}, b^{\text{slow}}, c^{\text{slow}})$ satisfies (3.19). Assuming that $(A^{\text{O}}, b^{\text{O}}, c^{\text{O}})$ satisfies (3.19) we find

$$\begin{aligned}
\frac{1}{3} &= \sum_{i=1}^{s^{\text{slow}}} (c_{i+1}^{\text{slow}} - c_i^{\text{slow}}) \sum_{j=1}^i (a_{i+1,j}^{\text{slow}} + a_{i,j}^{\text{slow}}) c_j^{\text{slow}} \\
&= \sum_{i=1}^{s^{\text{O}}} \sum_{k=1}^{s^{\text{I}}} (c_{(i,k+1)}^{\text{slow}} - c_{(i,k)}^{\text{slow}}) \sum_{j=1}^{s^{\text{O}}} \sum_{l=1}^{s^{\text{I}}} (a_{(i,k+1),(j,l)}^{\text{slow}} + a_{(i,k),(j,l)}^{\text{slow}}) c_{(j,l)}^{\text{slow}} \\
&= \sum_{i=1}^{s^{\text{O}}} \sum_{k=1}^{s^{\text{I}}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) (c_{k+1}^{\text{I}} - c_k^{\text{I}}) \sum_{j=1}^{s^{\text{O}}} (2a_{i,j}^{\text{O}} + (a_{i+1,j}^{\text{O}} - a_{i,j}^{\text{O}}) (c_{k+1}^{\text{I}} + c_k^{\text{I}})) c_j^{\text{O}} \\
&= \sum_{i=1}^{s^{\text{O}}} \sum_{k=1}^{s^{\text{I}}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) (c_{k+1}^{\text{I}} - c_k^{\text{I}}) \sum_{j=1}^{s^{\text{O}}} 2a_{i,j}^{\text{O}} c_j^{\text{O}} + \\
&\quad \sum_{i=1}^{s^{\text{O}}} \sum_{k=1}^{s^{\text{I}}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) (c_{k+1}^{\text{I}} - c_k^{\text{I}}) (c_{k+1}^{\text{I}} + c_k^{\text{I}}) \sum_{j=1}^{s^{\text{O}}} (a_{i+1,j}^{\text{O}} - a_{i,j}^{\text{O}}) c_j^{\text{O}} \\
&= \underbrace{\sum_{k=1}^{s^{\text{I}}} (c_{k+1}^{\text{I}} - c_k^{\text{I}})}_{=1} \sum_{i=1}^{s^{\text{O}}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) \sum_{j=1}^{s^{\text{O}}} 2a_{i,j}^{\text{O}} c_j^{\text{O}} + \\
&\quad \underbrace{\sum_{k=1}^{s^{\text{I}}} ((c_{k+1}^{\text{I}})^2 - (c_k^{\text{I}})^2)}_{=1} \sum_{i=1}^{s^{\text{O}}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) \sum_{j=1}^{s^{\text{O}}} (a_{i+1,j}^{\text{O}} - a_{i,j}^{\text{O}}) c_j^{\text{O}} \\
&= \sum_{i=1}^{s^{\text{O}}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) \sum_{j=1}^{s^{\text{O}}} (2a_{i,j}^{\text{O}} + a_{i+1,j}^{\text{O}} - a_{i,j}^{\text{O}}) c_j^{\text{O}} \\
&= \sum_{i=1}^{s^{\text{O}}} (c_{i+1}^{\text{O}} - c_i^{\text{O}}) \sum_{j=1}^{s^{\text{O}}} (a_{i+1,j}^{\text{O}} + a_{i,j}^{\text{O}}) c_j^{\text{O}} \stackrel{\text{by assumption 3}}{=} \frac{1}{3}.
\end{aligned}$$

Now we can formulate the theorem:

Theorem 3.3.4. *If the inner and the outer base method are of consistency order $p = 3$ and the outer method satisfies the additional third order condition (3.19) then both partitions of the generated method satisfy (3.19).*

Up to this point we have only examined partitioned methods consisting of exactly two elementary methods. For a general PRK of N methods the order conditions read

$$\begin{aligned} \text{Order 1:} \quad & \forall n \in \{1, \dots, N\} : \sum_j b_j^{(n)} = 1, \\ \text{Order 2:} \quad & \forall n_1, n_2 \in \{1, \dots, N\} : \sum_j b_j^{(n_1)} c_j^{(n_2)} = 1/2, \\ \text{Order 3:} \quad & \forall n_1, n_2, n_3 \in \{1, \dots, N\} : \sum_j b_j^{(n_1)} c_j^{(n_2)} c_j^{(n_3)} = 1/3, \\ & \forall n_1, n_2, n_3 \in \{1, \dots, N\} : \sum_i \sum_j b_i^{(n_1)} a_{ij}^{(n_2)} c_j^{(n_3)} = 1/6, \end{aligned}$$

with the superscripts denoting method indexes. Again we make use of the fact that the methods constructed using the Recursive Flux Splitting Multirate algorithm are internally consistent, so in this context we have $\forall n : c^{(n)} = c$ and consequently the order conditions for an N -part PRK simplify to

$$\begin{aligned} \text{Order 1:} \quad & \forall n : \sum_j b_j^{(n)} = 1, \\ \text{Order 2:} \quad & \forall n : \sum_j b_j^{(n)} c_j = 1/2, \\ \text{Order 3:} \quad & \forall n : \sum_j b_j^{(n)} c_j^2 = 1/3, \\ & \forall n_1, n_2 : \sum_i \sum_j b_i^{(n_1)} a_{ij}^{(n_2)} c_j = 1/6. \end{aligned}$$

We see that even for the third order conditions we only have to consider pairs of Runge–Kutta methods as we have done before. In combination with Theorem 3.3.4 this leads to:

Corollary 3.3.5. *Employing the Recursive Flux Splitting Multirate algorithm we can construct a partitioned Runge–Kutta method of third order accuracy and an arbitrary number of partitions by iteratively employing formerly constructed methods as outer method if the employed base methods satisfy the classical third order conditions (3.11a), (3.11b), (3.11c), (3.11d) and all base methods apart from the inner method employed for the last step satisfy the additional third order coupling condition (3.19).*

If all of the employed base methods satisfy (3.19), then the constructed PRK also satisfies (3.19).

3.3.4 Order conditions for arbitrary inner methods

In [29] Knoth and Wolke considered a general splitting as presented in Section 3.1.1 and derived order conditions up to order three. Specifically the authors examined a test equation

$$\dot{y} = f(y) + g(y),$$

and assumed that the inner system is solved exactly. However in their derivation of order conditions only the first two derivatives of the right hand side – corresponding to the first three derivatives of y – are employed so that it holds not only for an exact inner method but for any inner method of third order accuracy.

Consequently we may employ a third order partitioned method with an arbitrary number of partitions, also satisfying the additional third order condition (3.19), (see Corollary 3.3.5) and employ this method as outer method. The inner method then may be any time integration method of third order accuracy.

3.4 Extension to a mass preserving PRK

The time integration schemes examined up to this point are internally consistent but do not generally preserve the linear invariants of the system. The preservation of the linear invariants (or shorter mass preservation) is guaranteed for partitioned methods with equal summation stages, $\forall j : b_j^{\text{fast}} = b_j^{\text{slow}}$. Thus it is straightforward to modify a PRK constructed via the RFSMR approach by replacing the summation stage for all partitions so that the resulting method is mass preserving. The new summation weights $(\bar{b}_j)_{j=1,\dots,s}$ are a linear combination of the original summation weights

$$\forall j : \bar{b}_j = w b_j^{\text{fast}} + (1 - w) b_j^{\text{slow}}.$$

The method constructed in this manner satisfies the order conditions up to order 3 if (and only if) the original method is of order three. This is trivial since the summation weights occur only in linear form. As for the underlying method we have $c = c^{\text{fast}} = c^{\text{slow}}$ the

superscripts for the c_i are omitted below:

$$\begin{aligned}
\sum_{j=1}^s \bar{b}_j &= (1-w) \underbrace{\sum_{j=1}^s b_j^{\text{slow}}}_{=1} + w \underbrace{\sum_{j=1}^s b_j^{\text{fast}}}_{=1} = 1, \\
\sum_{j=1}^s \bar{b}_j c_j &= (1-w) \underbrace{\sum_{j=1}^s b_j^{\text{slow}} c_j}_{=1/2} + w \underbrace{\sum_{j=1}^s b_j^{\text{fast}} c_j}_{=1/2} = 1/2, \\
\sum_{j=1}^s \bar{b}_j c_j^2 &= (1-w) \underbrace{\sum_{j=1}^s b_j^{\text{slow}} c_j^2}_{=1/3} + w \underbrace{\sum_{j=1}^s b_j^{\text{fast}} c_j^2}_{=1/3} = 1/3, \\
\sum_{i=1}^s \sum_{j=1}^{i-1} \bar{b}_i a_{ij}^{\text{slow}} c_j &= (1-w) \underbrace{\sum_{i=1}^s \sum_{j=1}^{i-1} b_i^{\text{slow}} a_{ij}^{\text{slow}} c_j}_{=1/6} + w \underbrace{\sum_{i=1}^s \sum_{j=1}^{i-1} b_i^{\text{fast}} a_{ij}^{\text{slow}} c_j}_{=1/6} = 1/6, \\
\sum_{i=1}^s \sum_{j=1}^{i-1} \bar{b}_i a_{ij}^{\text{fast}} c_j &= (1-w) \underbrace{\sum_{i=1}^s \sum_{j=1}^{i-1} b_i^{\text{slow}} a_{ij}^{\text{fast}} c_j}_{=1/6} + w \underbrace{\sum_{i=1}^s \sum_{j=1}^{i-1} b_i^{\text{fast}} a_{ij}^{\text{fast}} c_j}_{=1/6} = 1/6.
\end{aligned}$$

Unfortunately, the advantageous property of mass preservation comes with a number of disadvantages. Most importantly the methods are not efficient anymore. For a Runge–Kutta method the right hand side must be evaluated at least once for each non-zero column of the Butcher tableau $\{j : \exists a_{i,j} \neq 0 \vee b_j \neq 0\}$. The slow methods generated via the RFSMR approach generally contain many zero columns but modifying the summation stage as proposed above leads to an increased number of non-zero columns and thus to increased computational cost. On the other hand one might choose $\bar{b} = b^{\text{slow}}$, so that the number of zero columns is maximized. This however leads to reduced stability for the fast method.

As an example consider the second order multirate method presented in Table 3.2. Listed in Table 3.5 you find the underlying method (a) and the method with modified summation stage (b). For this example the change of the summation stage makes the fourth stage redundant. The non-redundant method is given in Table 3.5c. For the slow method the stability function $R(z) = 1 + z + z^2/2$ is independent of the parameter w . We may however tune w in order to maximize stability for the fast method. The maximum Courant numbers for first order upwind and third order upwind are plotted against w in Figure 3.7. It shows that choosing $w \approx 0.739$ gives the best results for the third order upwind semidiscretization. A comparison with the underlying unextended method shows slightly improved efficiency (in terms of the stability to cost ratio) for the fast method applied on a third order upwind discretization while the efficiency of the slow method is reduced due to the increased number of evaluations of the right hand side, cf. Table 3.6.

0		0		0		0		0		0	
1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2
1/2	1/2	0		1/2	1/4	1/4	1/4	1/2		1/4	1/4
1	1	0	0	1	1/4	1/4	1/2	1	1/4	1/4	1/4
1	1	0	0	0	1/4	1/4	1/4	1	1/4	1/4	1/4
	1/2	0	0	0	1/2	1/4	1/4	1/4	1/4	1/4	0

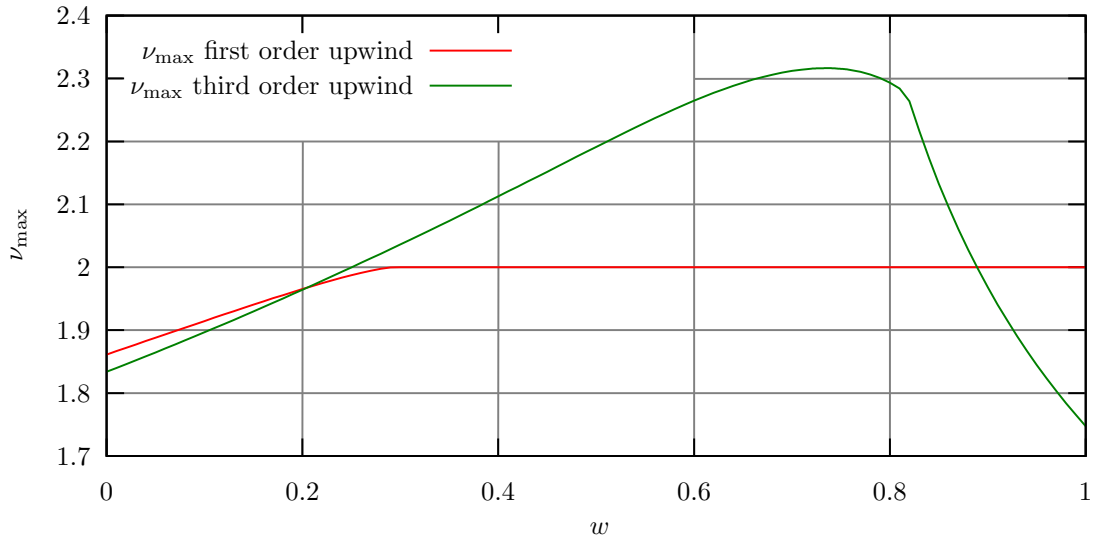


Figure 3.7: Maximum Courant number ν_{\max} of MPMR2a(fast) applied to advection operators as a function of w .

PRK	sub method	ν_{\max}	E_{RHS}	$\nu_{\max}/E_{\text{RHS}}$
RFSMR2a	slow	1.000 (0.874)	2	0.500 (0.437)
RFSMR2a	fast	2.000 (1.747)	4	0.500 (0.437)
MPMR2a($w = 0.739$)	slow	1.000 (0.874)	4	0.250 (0.219)
MPMR2a($w = 0.739$)	fast	2.000 (2.316)	4	0.500 (0.579)
MPMR2a($w = 0$)	slow	1.000 (0.874)	2	0.500 (0.437)
MPMR2a($w = 0$)	fast	1.861 (1.834)	4	0.465 (0.458)

Table 3.6: Maximum Courant numbers ν_{\max} , evaluations of the right hand side E_{RHS} and efficiency measure $\nu_{\max}/E_{\text{RHS}}$ for first order upwind (third order upwind) spatial discretization.

0					
1/2	1/2				
1/2	-1/6	2/3			
1	1/3	-1/3	1		
	1/6	1/3	1/3	1/6	

outer base method

0				0
γ	0	γ		
$1-\gamma$	0	$1-2\gamma$	γ	
	0	1/2	1/2	

inner base method

a) Base methods, $\gamma = \frac{3+\sqrt{3}}{6}$

0	0								
$\frac{\gamma}{2}$	$\frac{\gamma}{2}$	0							
$\frac{1-\gamma}{2}$	$\frac{1-\gamma}{2}$	0	0						
1/2	1/2	0	0	0					
1/2	-1/6	0	0	2/3	0				
$\frac{1+\gamma}{2}$	$\frac{-1+3\gamma}{2}$	0	0	$\frac{2}{3}-\gamma$	γ	0			
$\frac{-\gamma}{2}$	$\frac{2-3\gamma}{6}$	0	0	$-\frac{1}{3}+\gamma$	$1-\gamma$	0	0		
1	1/3	0	0	-1/3	1	0	0	0	
	0	1/4	1/4	0	0	1/4	1/4	0	MPIMEX3a
	1/6	0	0	1/3	1/3	0	0	1/6	MPIMEX3b

b) Explicit (slow) method, $\gamma = \frac{3+\sqrt{3}}{6}$

0	0								
$\frac{\gamma}{2}$	0	$\gamma/2$							
$\frac{1-\gamma}{2}$	0	$1/2-\gamma$	$\gamma/2$						
1/2	0	1/4	1/4	0					
1/2	0	1/4	1/4	0	0				
$\frac{1+\gamma}{2}$	0	1/4	1/4	0	0	$\gamma/2$			
$\frac{-\gamma}{2}$	0	1/4	1/4	0	0	$1/2-\gamma$	$\gamma/2$		
1	0	1/4	1/4	0	0	1/4	1/4	0	
	0	1/4	1/4	0	0	1/4	1/4	0	MPIMEX3a
	1/6	0	0	1/3	1/3	0	0	1/6	MPIMEX3b

c) Implicit (fast) method, $\gamma = \frac{3+\sqrt{3}}{6}$

Table 3.7: Example for a third order accurate mass preserving IMEX method. The two summation stages are alternatives we denote as MPIMEX3a and MPIMEX3b respectively.

cast as a PRK.

An example for a mass preserving third order IMEX method is given in Table 3.7. Note that a redundant stage was introduced in the inner base method to obtain a method with $c_1^I = 0$. This method leaves only little room for optimization. Considering

$$\bar{b} = ((1 - w)/6, w/4, w/4, (1 - w)/3, (1 - w)/3, w/4, w/4, (1 - w)/6)$$

we see that the implicit method is A-stable only if $w = 1$, as is the case for MPIMEX3a. For any other value of w not even A(α) stability is obtained. The stability of these methods applied on advection-reaction systems shall be examined in the following section.

3.5 Stability for advection-reaction systems

To examine the stability properties of the recursive flux splitting multirate approach applied on advection-reaction systems we consider an advection-reaction equation in matrix notation. Starting from an advection operator represented by a normal matrix A to be applied to each species individually and a reaction operator represented by a normal matrix B to be applied to each cell individually this reads

$$\dot{\hat{y}} = \underbrace{(A \otimes I^{(S)} + I^{(N)} \otimes B)}_{=R} \hat{y}, \quad (3.20a)$$

$$\hat{y}(0) = I^{(N)} \otimes I^{(S)}, \quad (3.20b)$$

$$A \in \mathbb{R}^{N \times N}, \quad I^{(N)} \in \mathbb{R}^{N \times N}, \quad (3.20c)$$

$$B \in \mathbb{R}^{S \times S}, \quad I^{(S)} \in \mathbb{R}^{S \times S}, \quad (3.20d)$$

with the number of cells N and the number of species S . The I denote identity matrices.

If the eigenvalues of A read $(\lambda_i)_{i=1, \dots, N}$ and the eigenvalues of B read $(\mu_j)_{j=1, \dots, S}$ there are eigendecompositions of A and B respectively:

$$\begin{aligned} \exists P \in \mathbb{R}^{N \times N} : P \Lambda P^{-1} &= A, \quad \Lambda = \text{diag} \left((\lambda_i)_{i=1, \dots, N} \right), \\ \exists Q \in \mathbb{R}^{S \times S} : Q M Q^{-1} &= B, \quad M = \text{diag} \left((\mu_j)_{j=1, \dots, S} \right). \end{aligned}$$

Based on these eigendecompositions we can construct an eigendecomposition of R :

$$\begin{aligned} R &= A \otimes I^{(S)} + I^{(N)} \otimes B \\ &= P \Lambda P^{-1} \otimes I^{(S)} + I^{(N)} \otimes Q M Q^{-1} \\ &= (P \otimes Q) \left[(P \otimes Q)^{-1} (P \Lambda P^{-1} \otimes I^{(S)} + I^{(N)} \otimes Q M Q^{-1}) (P \otimes Q) \right] (P \otimes Q)^{-1} \\ &= (P \otimes Q) \left[P^{-1} (P \Lambda P^{-1}) P \otimes Q^{-1} I^{(S)} Q + P^{-1} I^{(N)} P \otimes Q^{-1} (Q M Q^{-1}) Q \right] (P \otimes Q)^{-1} \\ &= (P \otimes Q) \left[(\Lambda \otimes I^{(S)}) + (I^{(N)} \otimes M) \right] (P \otimes Q)^{-1} \\ &= (P \otimes Q) \left[\text{diag} \left((\lambda_i + \mu_j)_{\substack{i=1, \dots, N \\ j=1, \dots, S}} \right) \right] (P \otimes Q)^{-1}. \end{aligned}$$

From this we see that the eigenvalues of R read

$$(\lambda_i + \mu_j)_{\substack{i=1,\dots,N \\ j=1,\dots,S}}$$

Consequently for advection-reaction systems it is sufficient to examine a linear test equation and consider the different eigenvalues of R individually. These considerations also hold for a *splitting by directions*, with N in (3.20) representing the number of cells along one spatial axis and S denoting the number of cells along a perpendicular spatial axis instead of species. This is of interest for a splitting into horizontal and vertical transport.

We now consider a linear test equation

$$\begin{aligned} \dot{y} &= \lambda y + \mu y, \\ y(0) &= 1, \end{aligned}$$

with λ representing the eigenvalues of the advection operator – either first order upwind (Λ_1) or third order upwind (Λ_3) – and μ representing the eigenvalues of the reaction terms. More specifically we define

$$\begin{aligned} \mu &\in \mathbb{R}^-, \\ \lambda &\in \Lambda_p(\nu), \\ \Lambda_1(\nu) &= \left\{ \nu \left(e^{2\pi i \frac{n}{N}} - 1 \right), n \in \{1, \dots, N\} \right\}, \\ \Lambda_3(\nu) &= \left\{ \nu \left(-\frac{1}{6} e^{4\pi i \frac{n}{N}} + e^{2\pi i \frac{n}{N}} - \frac{1}{2} - \frac{1}{3} e^{-2\pi i \frac{n}{N}} \right), n \in \{1, \dots, N\} \right\}, \end{aligned}$$

so that for a specified spatial discretization we may examine stability of the system in terms of the Courant number ν and the time constant of the reaction $|\mu|$. It shows that examining more than 16 wave numbers per advection operator only marginally improves the quality of the result. Thus we choose $N = 16$ in the definitions of Λ_1 and Λ_3 . For each eigenvalue λ of the advection operator the system is solved as follows

$$\begin{aligned} Y_1 &= 1, \\ r_i &= \sum_{j=1}^{i-1} (a_{ij} - a_{i-1,j}) \lambda Y_j, \\ v_i(0) &= Y_{i-1}, \\ \frac{d}{d\tau} v_i &= r_i + (c_i - c_{i-1}) \mu, \quad \tau \in [0, 1], \\ Y_i &= v_i(1), \\ y_{\text{step}}(\lambda) &= Y_{s+1}. \end{aligned}$$

The maximum amplification $Amp(\nu, \mu)$ for all eigenvalues of the advection operator after one time step reads

$$Amp(\nu, \mu) = \max_{\lambda \in \Lambda(\nu)} (|y_{\text{step}}(\lambda)|).$$

Note that Amp equals the spectral radius of the analogously approximated $\hat{y}(1)$ according to System (3.20).

Results for a few methods are shown in Figures 3.8 through 3.15. The stability region is shown in white. A blue shading indicates instability for first order upwind while a red shading indicates instability for third order upwind. The rationale for considering stability for third order upwind and first order upwind in a single plot is that we actually are interested in the third order upwind discretization *with limiter* as presented in Section 2.2.2. Since the nonlinear limiter (2.15) leads to a first order upwind discretization at extrema and to a third order upwind discretization for smooth profiles both linear discretizations are equally (and simultaneously) of interest.

As for the pure advection problem the IMEX method based on forward Euler as outer method and backward Euler as inner method (Figure 3.8) is unstable for third order upwind and small μ . Greater time constants of the reaction lead to an increased damping and thus to stability for larger Courant numbers.

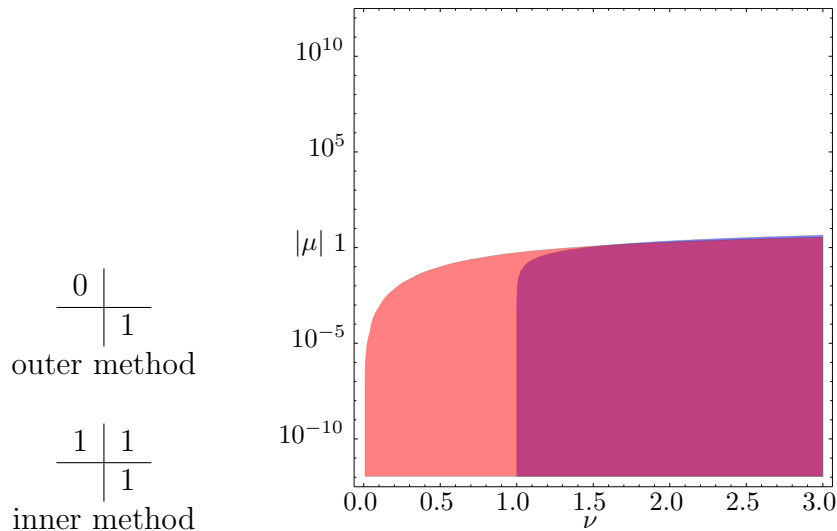


Figure 3.8: Stability of a first order IMEX method.

The second order IMEX method based on the explicit trapezoidal rule (Heun method) and the implicit trapezoidal rule is stable for a first order upwind discretization and $\nu \leq 1$, see Figure 3.9. Choosing the third order spatial discretization however leads to reduced stability for $|\mu| > 10$.

The three stage second order method (RK32) was recommended by Gerisch and Weiner in [14] for the use in Strang-type IMEX methods and it also proves advantageous for the splitting approach presented here. In combination with the implicit trapezoidal rule as inner method it shows good stability along the ν -axis for small $|\mu|$. However larger time constants of the reaction $|\mu| > 300$ again lead to reduced stability along the ν coordinate. This is correlated to the fact that the stability function of the implicit trapezoidal rule reads $R(z) = \frac{1+z/2}{1-z/2}$, so that for the stability limit we have $\lim_{z \rightarrow \infty} R(z) = -1$.

0	
1	1
	1/2 1/2
outer method	
0	0
1	1/2 1/2
	1/2 1/2
inner method	

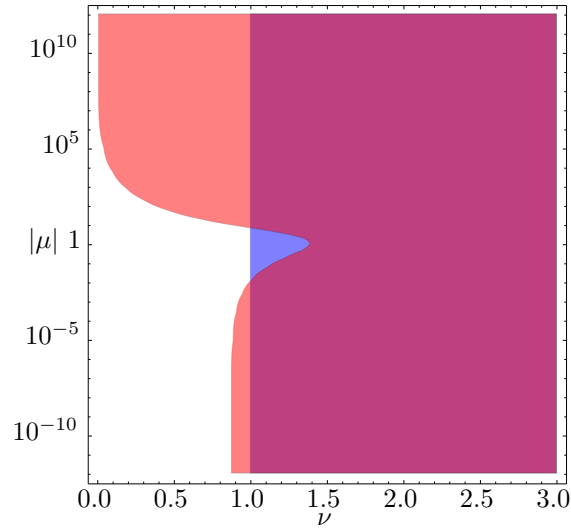


Figure 3.9: Stability of a second order IMEX method based on explicit and implicit trapezoidal rule.

0			
1/2	1/2		
1	1/2	1/2	
	1/3	1/3	1/3
outer method			
0	0		
1	1/2	1/2	
	1/2	1/2	
inner method			

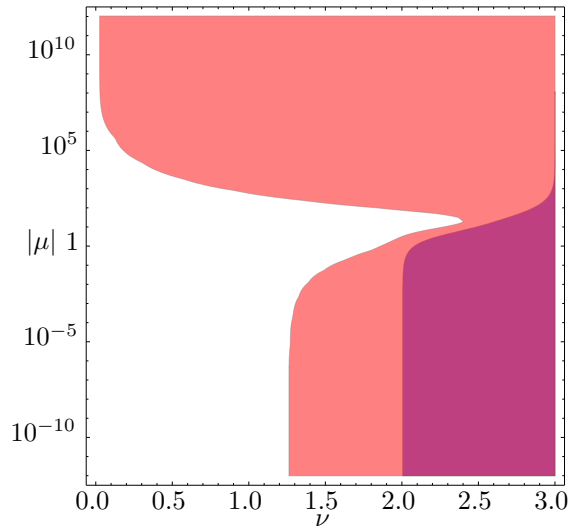


Figure 3.10: Stability of a second order IMEX method based on RK32.

Next we consider a third order IMEX method based on (RK43) and a two stage third order SDIRK method, see Figure 3.11. This method is stable for any $|\mu|$ and $\nu < 0.6$ ($\nu < 0.9$) for first (third) order upwind. Especially the reduced stability along the ν -axis for large reaction rates $|\mu|$ is unfavorable. We shall now construct an alternative third order method with four stages and better stability. We choose the nodes of the method to read $c_1 = 0$, $c_2 = c_3 = 1/2$, $c_4 = 1$. Solving for the order conditions up to order three including the additional coupling condition results in a family of ERKs with two free parameters, see Table 3.8. The remaining free parameters α , β can be tuned to optimize for stability.

0				
1/2	1/2			
1/2	-1/6	2/3		
1	1/3	-1/3	1	
	1/6	1/3	1/3	1/6
outer method				
$\frac{3+\sqrt{3}}{6}$	$\frac{3+\sqrt{3}}{6}$			
$\frac{3-\sqrt{3}}{6}$	$-\frac{1}{\sqrt{3}}$	$\frac{3+\sqrt{3}}{6}$		
	1/2	1/2		
inner method				

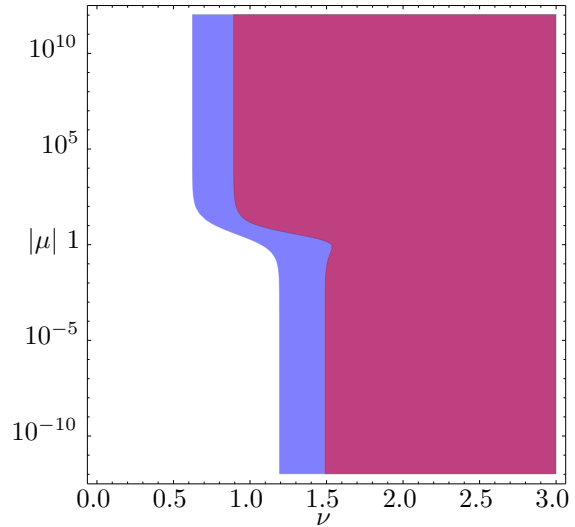


Figure 3.11: Stability of a third order IMEX method based on (RK43).

0				
1/2	1/2			
1/2	$\frac{1-2\alpha}{2}$	α		
1	$\frac{3\alpha-1}{3}$	$\frac{4-3\alpha-3\beta}{3}$	β	
	1/6	$\frac{9\alpha-2}{18\alpha}$	$\frac{3\alpha+2}{18\alpha}$	1/6

Table 3.8: A family of third order ERKs.

We employ the stability analysis as before and try to achieve stability for a Courant number of $\nu = 1.2$, reaction speeds of $|\mu| \in \{10^{-5}, 1, 10^5\}$ and the SDIRK method as inner integrator. The stability regions in terms of α and β for both spatial discretizations in consideration are shown in Figure 3.12. Choosing $\alpha = 1$ and $\beta = 1/4$ we obtain the method shown in Figure 3.13, subsequently denoted as (RK43b). In Section 5.2 we shall compare the IMEX methods based on (RK43) (Figure 3.11) and (RK43b) (Figure 3.13) applied to an advection-reaction problem numerically.

To take also a purely explicit splitting into consideration we examine a method constructed from (RK32) as outer method and the second order 8-stage Runge–Kutta–Chebyshev (RKC) method presented by Verwer in [46] as inner method, see Figure 3.14. Note that for this plot the $|\mu|$ axis is linear. The inner method is stable along the real axis down through $z = -42$ but due to the nodes of the outer method two steps of half size are employed, thus extending the stability along the real axis until $z = -84$. This is clearly represented in the stability plot. Interestingly the advection stability for first order upwind semidiscretization also reflects further properties of the inner method’s stability behavior as indicated by the horizontal lines between the plots. Employing RKC methods as inner methods is promising for a splitting into horizontal advection and vertical diffusion, even more so in meteorological models where the vertical grid size usually is

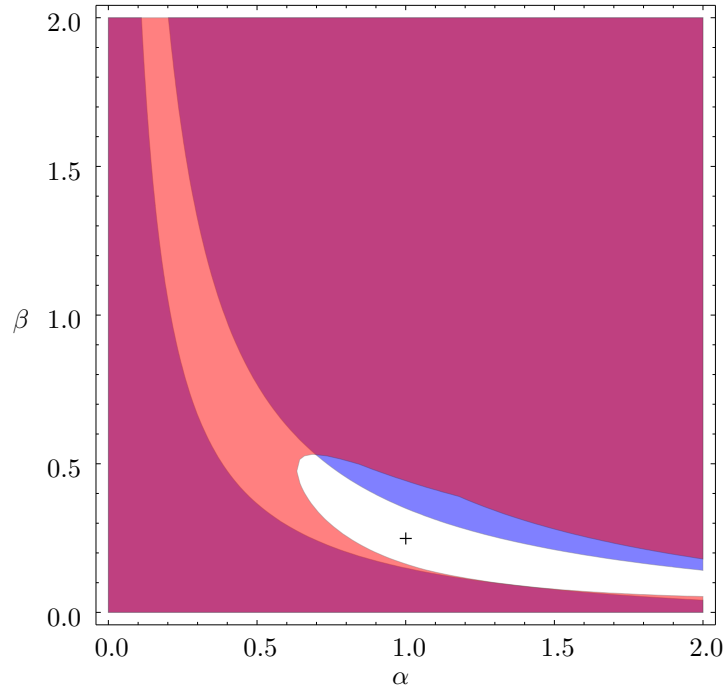


Figure 3.12: Stability for IMEX methods constructed from the family of third order ERKs (Table 3.8) as outer method and (SDIRK3) as inner method for $|\mu| \in \{10^{-5}, 1, 10^5\}$ and $\nu = 1.2$. The cross marks the point $(\alpha, \beta) = (1, 1/4)$.

much smaller than the horizontal grid size. If vertical transport also involves advection, the use of modified RKC methods as presented in [47] should be considered. The

First numerical tests showed promising results, but the combination with a further implicit method for the solution of advection-diffusion-reaction systems still has to be examined in detail. A particular question is if RKC methods or their variations presented in [47] can be modified in order to make them more suitable for the use as an outer method in the RFSMR approach or even *efficient with respect to a given time step ratio* (see Section 3.1.3) while maintaining their advantageous stability properties to a large extend.

The mass preserving IMEX methods listed in Table 3.7 are equivalent to the method shown in Figure 3.11 apart from the summation stages. Performing our stability analysis for the mass preserving methods shows that extending the method with the summation stage of the fast (or implicit) method results in reduced stability along the ν -axis, while extension with the summation stage of the slow (or explicit) method results in reduced stability along the $|\mu|$ -axis compared to the not mass preserving method, see Figure 3.15. Both methods are inferior to the underlying (not mass preserving) method in terms of stability.

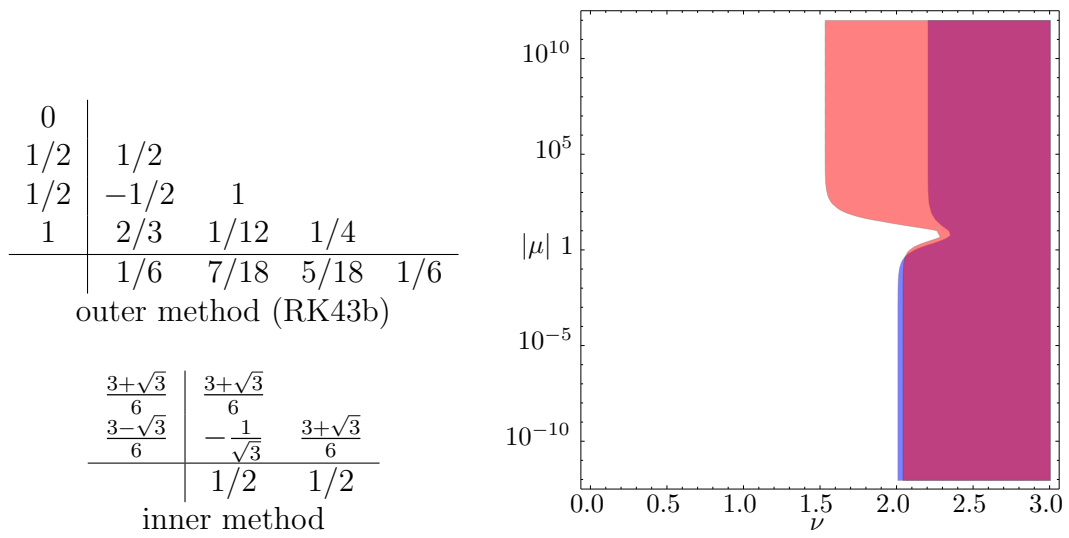


Figure 3.13: Stability of a third order IMEX method based on (RK43b).

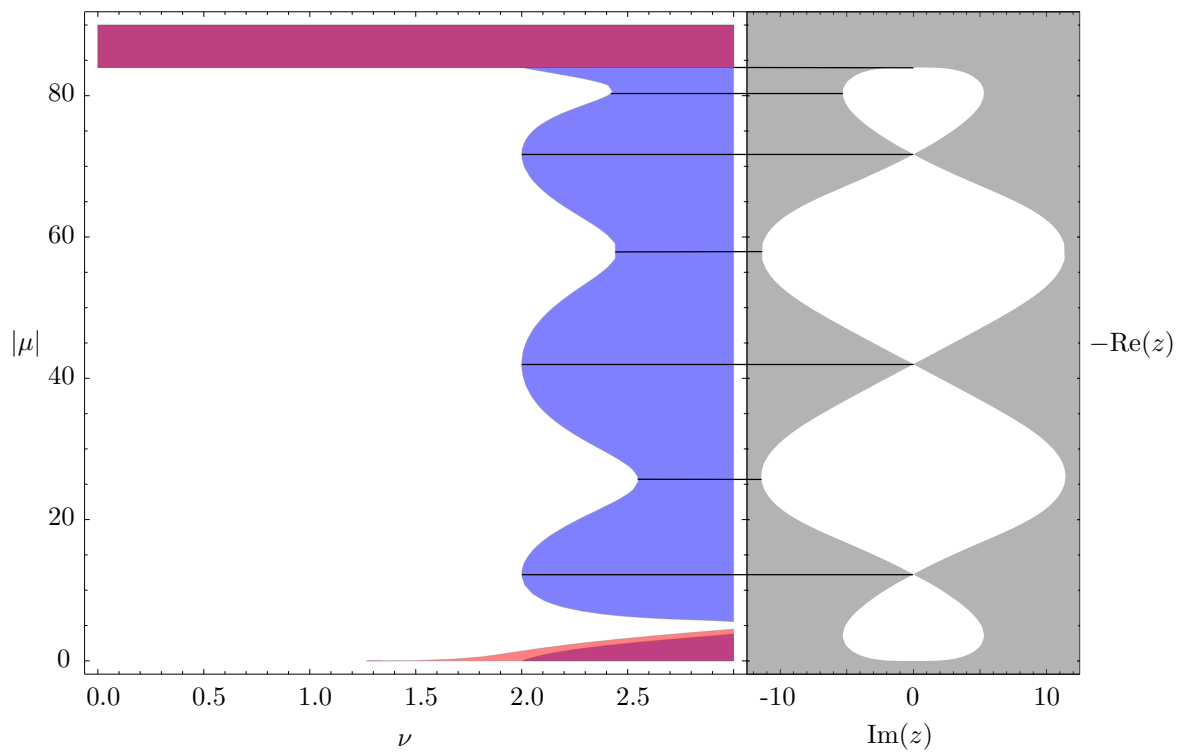


Figure 3.14: Stability of a second order explicit method based on (RK32) and the second order 8-stage Runge–Kutta–Chebyshev method (left) and the scaled classical stability region for the inner method (right).

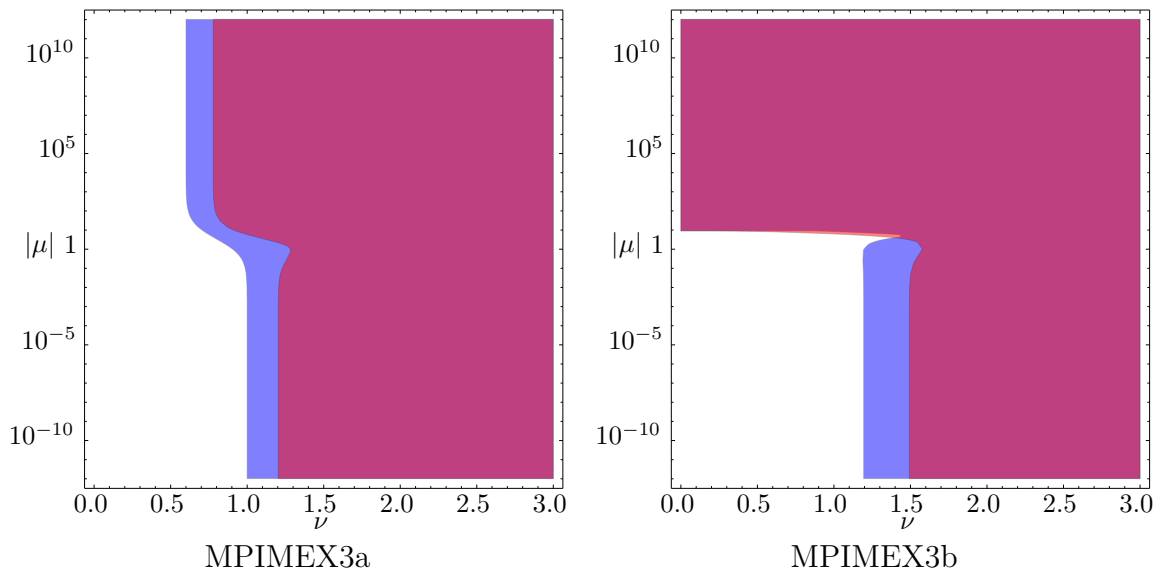


Figure 3.15: Stability of the mass preserving third order IMEX methods listed in Table 3.7.

Chapter 4

Implementation issues

4.1 The multi scale atmospheric transport model

The multi scale atmospheric transport model (MUSCAT) has been developed at the Leibniz Institute for Tropospheric Research since the early 1990s. It is written mainly in Fortran, with few additional modules written in C. The code enables parallel computing by employing MPI (Message passing interface [11, 12]). For sophisticated workload balancing the Metis/ParMetis libraries [26] are employed.

An early model was constructed to simulate chemical kinetics and diffusive transport inside of a vertical column [27]. This model was then extended to describe transport in three spatial dimensions with a terrain following grid and local grid refinement [28]. Coupling of horizontal advective transport and column processes was implemented using an IMEX approach. Later extensions and improvements of the code included improved workload balancing [49] and coupling to the COSMO model of the German Weather Service [41, 48, 31].

MUSCAT has been applied to a wide variety of simulation setups ranging from box models through fully three dimensional case studies concerning e.g. air pollution [20] or transport of Saharan dust [19].

4.2 Data organization

In MUSCAT data is organized hierarchically: the three dimensional computational domain is decomposed statically into rectangular blocks. This decomposition is applied in horizontal direction only so that every block ranges from ground level through the top of the domain to simulate. An important feature is that each block may have its own spatial refinement level. Thus selected regions may be examined in more detail. The cell size always is the macro cell size multiplied by an integer power of two. The spatial resolution of two directly adjacent blocks may differ by a factor of two maximum. The latter also holds for the temporal refinement level or short the *time level* of the block. Cell size, adjacency to other blocks, time level, etc. form the block meta-data. Even for

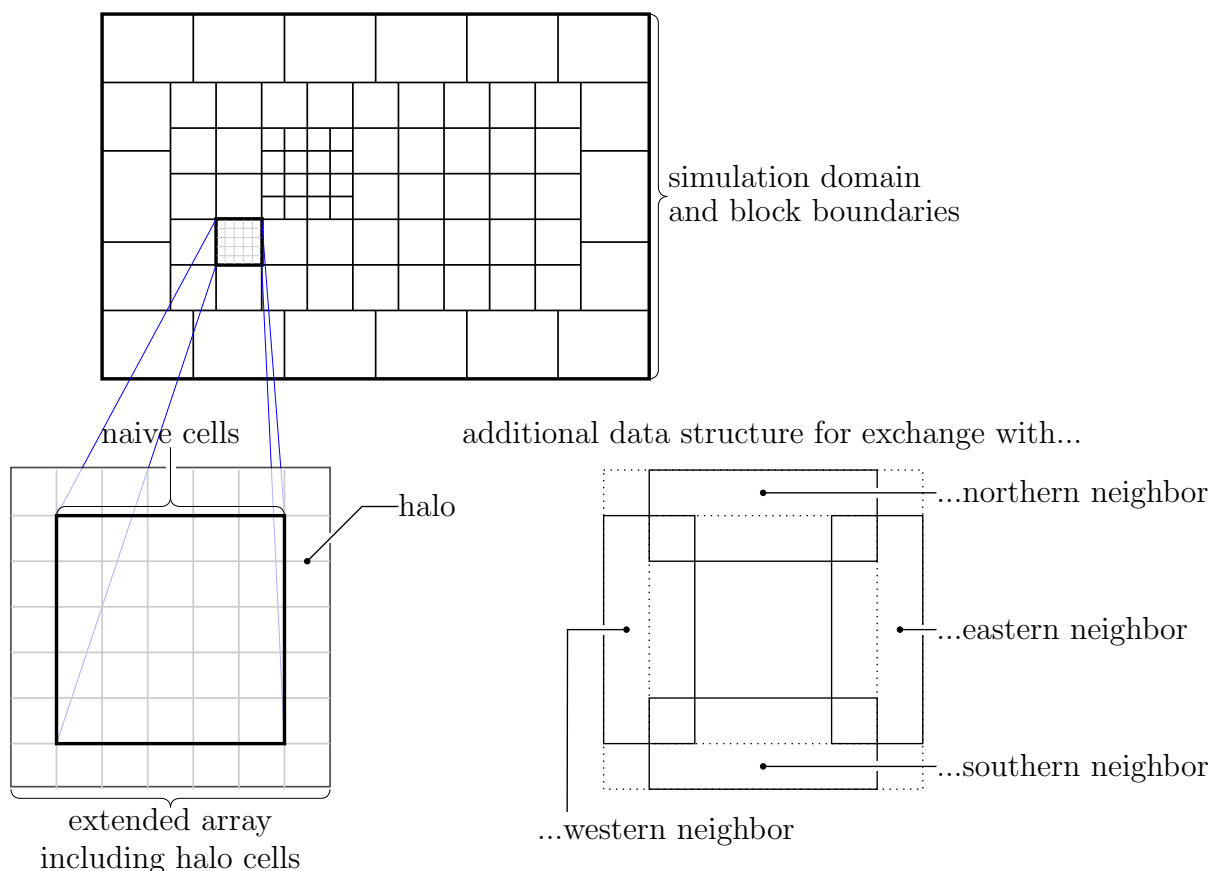


Figure 4.1: Data of one generic block.

parallel processing meta-data on all blocks is available at all processors. The main part of the data consists of a concentration array and a number of *difference* arrays associated with the stage vectors of the employed explicit Runge–Kutta method. Opposed to the meta-data, the main data of the block is present only on the processor the block is associated with. Additional variables include geometric data per cell (volume, extend per axis etc.) defined on initialization and meteorological data as wind speed or density of air. The latter may be provided by the COSMO model [44, 41] of the German weather service via online coupling or by a test driver.

The declaration of the concentration array is done in such a way that cell local values, i.e. the concentrations of the different tracers or species inside of a cell are directly adjacent in physical memory. The cell data in turn is organized so that cells within one column (i.e. cells at the same horizontal position) have minimal address distance. This layout is advantageous for the computations executed most frequently, i.e. cell local chemistry and column local (vertical) diffusion. For each advection step a number of fully coupled implicit chemistry-diffusion steps is needed which contribute significantly to the overall computational cost. All arrays have the same structure making vectorized operations possible.

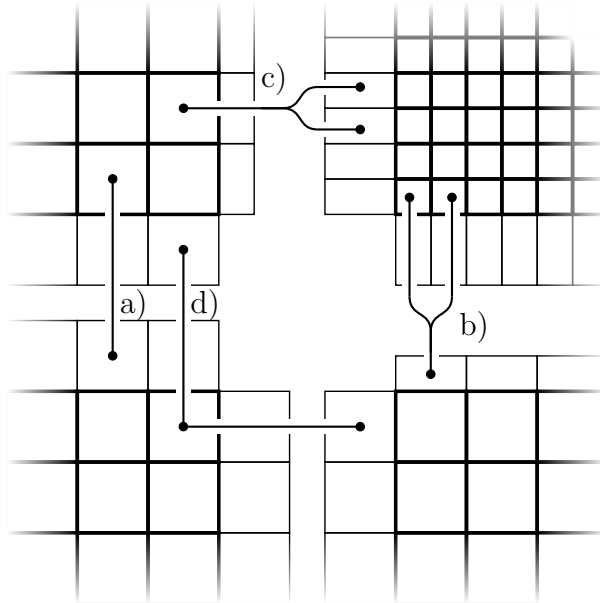


Figure 4.2: Geometrical cell structure of adjacent blocks. Halo cells depicted with thinner contours. Connectors between blocks indicate multiple representations of the same physical volume.

We employ an *extended array* declaration for the individual blocks where the extended array includes both the actual cells of the block and the surrounding halo or ghost cells which are needed for coupling with adjacent blocks. Thus all block local computations may be done on a logically Cartesian array. Additional data structures for the exchange of mass fluxes with neighboring blocks are needed along the respective boundaries, see Figure 4.1. These data structures correspond to the source term r . Employing a limited third order upwind spatial discretization [23] and a splitting by fluxes the halo has to be one cell wide (see also Section 3.2.1) while the additional data structures overlap with the halo cells and the outermost row of actual cells.

Though the grid of the block is logically Cartesian, its geometrical interpretation may be different. First of all the simulation domain usually represents a volume above a spherical surface. Furthermore perpendicularly to the interface the halo cells have the extent of the actual cells they overlap with, see Figure 4.2. Also marked in Figure 4.2 are the possible relations between actual cells and halo cells representing the same physical volume. The respective volume may be represented by:

- a) one inner cell and one halo cell,
- b) two inner cells and one halo cell of a coarser neighbor,
- c) one inner cell and two halo cells of a finer neighbor or
- d) inner cell(s) and halo cell(s) for each of two neighbors.

Case d) only occurs at block corners and may be complicated for neighboring blocks with different spatial refinement levels. These different possible relations have to be taken into account when data is exchanged between blocks as shall be discussed in Section 4.4.

Keep in mind that the domain is decomposed in the horizontal direction only, so that all of the above does not hold only for cells in one vertical layer but also for the columns, ranging from the bottom through the top of the simulation domain.

4.3 Recursive implementation

In this section we shall describe details of the implementation with focus on the time integration routines. The algorithm described formally in Section 3.1 naturally translates to an implementation as is suitable for a practical model. In the context of MUSCAT we want to solve an ODE

$$\dot{w} = F(w) + G(w),$$

with G representing horizontal advective fluxes and F representing vertical transport due to diffusion as well as chemical reactions. A flowchart of the time integration routine is given in Figure 4.3. The routine is called with a parameter L denoting the time level, a time step Δt and a point in time t_0 which is of interest for time dependent terms as e.g. time dependent source terms. Called from the main program with $L = 0$ the routine integrates all cells from a point in time t_0 to a point in time $t_0 + \Delta t$. Called recursively with $L > 0$ the routine integrates all cells associated with a time level equal or larger than L .

We shall now detail the various subroutines (represented as rectangles with double-struck vertical edges). To avoid unnecessary evaluations of the advection operator we store the advective fluxes $(G_i)_{i=2,\dots,s+1}$ computed for each stage of the Runge-Kutta method. This approach also means that only the most recent state vector W_i is employed for any calculation on stage $i + 1$ so that one variable W storing the latest intermediate result is sufficient. We denote block specific variables by a superscript B . To simplify the notation we introduce the set of blocks on time level L :

$$\bar{B}_L := \{B : L^B = L\},$$

with L^B denoting the time level of block B . To guarantee termination the maximum time level of all blocks is stored in a globally available variable $L_{\max} = \max_B \{L^B\}$.

The subroutine **Compute advective fluxes and source terms** updates the advective fluxes and source terms for those blocks B whose time level L^B equals the current time level L , given as parameter of the routine:

$$\begin{aligned} \forall B \in \bar{B}_L : G_i^B &:= G(W^B), \\ r^B &:= \sum_{j=1}^i (a_{ij} - a_{i-1,j}) G_i^B + r_{\text{slower}}^B. \end{aligned}$$

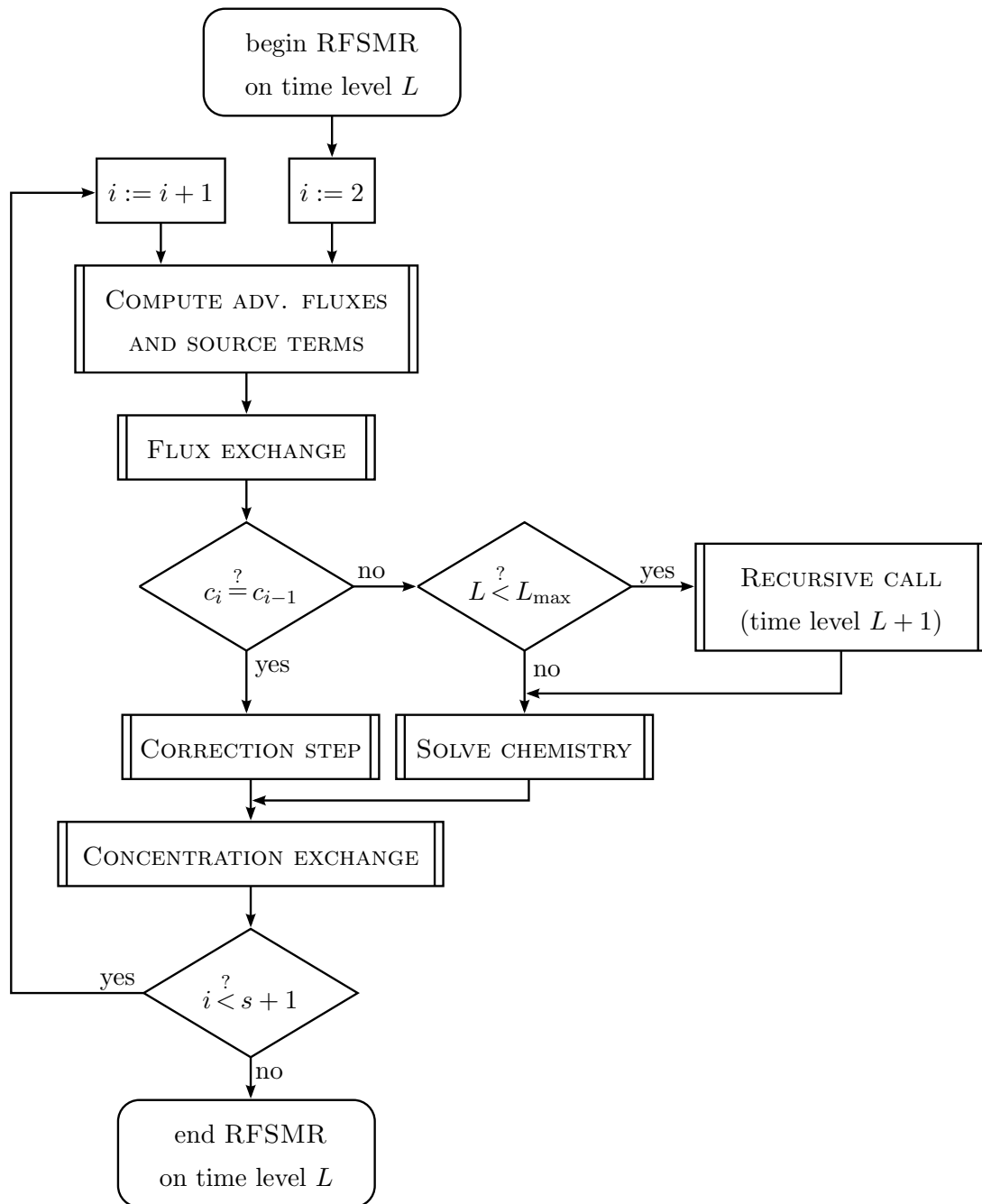


Figure 4.3: Flowchart of time integration routine on time level L ; s denotes the number of stages of the explicit base method, $(c_i)_{i=1,\dots,s+1}$ denote its nodes. The routine is called with $L = 0$ from the main program or recursively with $0 < L \leq L_{\max}$.

The term r_{slower}^B denotes source terms received from slower neighbor blocks as well as fluxes across the boundary of the simulation domain as defined by the boundary conditions.

Subroutine **Flux exchange** shall be detailed in Section 4.4. It modifies the value of $(r^B)_{B \in \bar{B}_L}$ so that it contains all advective fluxes influencing block B 's cells. By setting the value of $(r_{\text{slower}}^B)_{B \in \bar{B}_{L+1}}$ the coupling via source terms to faster neighbor blocks is also handled here.

The subroutine **Recursive call** computes the number of micro steps n to be taken and the micro time step Δt_μ to be employed on the next higher time level (see also Section 3.1.3)

$$\begin{aligned} n &:= \lceil 2(c_i - c_{i-1}) \rceil, \\ \Delta t_\mu &:= (c_i - c_{i-1})/n\Delta t, \end{aligned}$$

and calls the time integration routine n times recursively with parameters

$$\begin{aligned} L_{(\text{call } k)} &= L + 1, \\ \Delta t_{(\text{call } k)} &= \Delta t_\mu, \\ t_{0,(\text{call } k)} &= t_0 + k\Delta t_\mu, \quad k = 0, \dots, n - 1, \end{aligned}$$

where the subscript (call k) indicates the parameter as passed on the k -th call. This is equivalent to the solution of the inner system (3.10e) for the tripartite splitting discussed in Section 3.2.2. Note that if the explicit base method is efficient with respect to a time step ratio of $R = 2$ (cf. Section 3.1.3) we have $\Delta t_\mu = \Delta t/2$. By recursion the time step Δt_L associated with time level L is then found to read $\Delta t_L = \Delta t_{\text{macro}}/2^L$.

Subroutine **Solve chemistry** solves an initial value problem for an ordinary differential equation and stores its result in the concentration vector W according to

$$\begin{aligned} \forall B \in \bar{B}_L : \quad v^B(0) &= W^B, \\ \frac{d}{d\tau} v^B &= \Delta t r^B + \Delta t (c_i - c_{i-1}) F(v^B), \quad \tau \in [0, 1], \\ W^B &:= v^B(1). \end{aligned}$$

This is equivalent to the solution of the inner system (3.10d) for the tripartite splitting.

Subroutine **Correction step** updates the concentration vector W according to

$$\forall B \in \bar{B}_L : \quad W^B := W^B + \Delta t r^B.$$

The final routine **Concentration exchange** shall be discussed in Section 4.4. It modifies the value of the concentration vector $(W^B)_{B \in \bar{B}_L}$ to allow for coupling of neighboring blocks and also takes boundary conditions at the boundary of the simulation domain into account.

One significant advantage compared to a more general implementation allowing for the use of arbitrary partitioned methods is the reduced memory footprint. Depending on the base method's number of stages s we need $s + 1$ auxiliary vectors – one for each of the stage fluxes $(G_i)_{i=2, \dots, s+1}$ and one for the summation variable r – to store the advective fluxes plus additional data structures for data exchange. An equivalent representation as

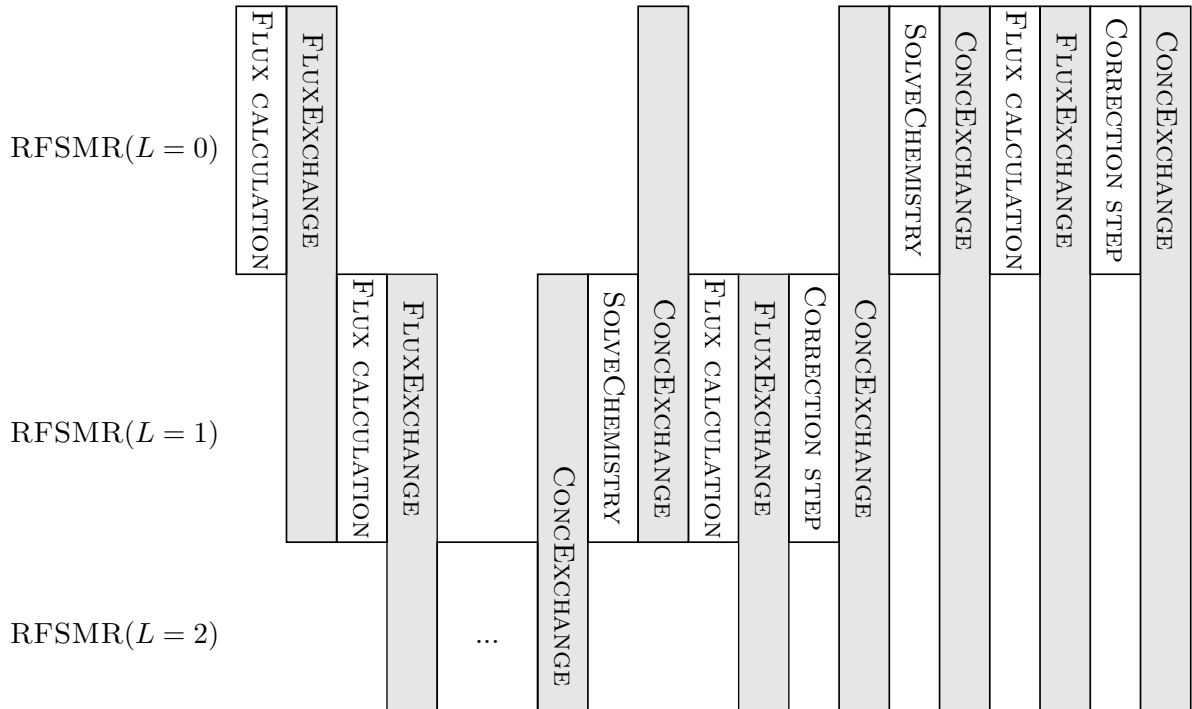


Figure 4.4: Illustration of exemplary program flow.

a partitioned method would involve significantly more stages depending on the number of time levels, see for instance Table 3.4.

Due to recursive calls and data exchanges the program flow is naturally structured in phases, see Figure 4.4. The resulting challenges for load balancing and their solution shall be discussed in Section 4.5.

4.4 Block coupling

The halo cells of a block providing information on other, adjacent blocks must be updated periodically. Exchange strategies aim at exchanging data as often as necessary and as rarely as possible. For a classical explicit Runge–Kutta method these criteria are met if the halo cell's concentrations are updated for each stage of the method. In the context of RFSMR exchanges may take the following forms:

Flux exchange

- (F1) between blocks on equal temporal refinement level, or
- (F2) from blocks on a coarser temporal refinement level to blocks on the next finer temporal refinement level.

Concentration exchange

- (C1) between blocks on equal temporal refinement level,
- (C2) from blocks on a finer temporal refinement level to blocks on the next coarser temporal refinement level, or
- (C3) from blocks on a coarser temporal refinement level to blocks on the next finer temporal refinement level.

Exchanges involving two blocks on the same temporal refinement level (F1,C1) are equivalent to the exchanges as needed for classical time integration; consequently they have to be executed for each Runge–Kutta stage of the respective time level. The flux exchange from a coarser to a finer temporal refinement level (F2) implements the coupling via the source term r and consequently has to take place before each recursive call. After each recursive call the updated concentrations must be passed back to adjacent blocks on a coarser refinement level (C2). The last of the listed exchanges (C3) is needed for correction steps to update both the halo cells and the outermost actual cells of the blocks on the next finer level.

Summarizing the above one can interpret all exchanges to be triggered by the slower of the involved blocks. Consequently the communication cost is reduced significantly compared to the naive strategy of updating the halo cells for each stage of the micro time step.

4.4.1 Coupling of directly adjacent blocks

From a geometrical point of view all exchanges are complicated by the fact that the same physical volume may be represented in different ways, see for instance Figure 4.5. The central requirement for multiple representations of a volume is that all of them represent the same mass m :

$$m^{(1)} = m^{(2)},$$

with the superscripts indicating different representations. All masses (and their temporal derivatives) are represented by concentrations c (and their temporal derivatives) in combination with the cell volumes V , so that the mass equivalency can be expressed as

$$\sum_{k=1}^{K^{(1)}} c_k^{(1)} V_k^{(1)} = \sum_{k=1}^{K^{(2)}} c_k^{(2)} V_k^{(2)}, \quad (4.1)$$

the subscript k representing a cell index. If both representations employ the same cell volumes, $\forall k : V_k^{(1)} = V_k^{(2)}$, then (4.1) is satisfied by:

$$\forall k : c_k^{(1)} = c_k^{(2)}.$$

At the interface between a block with a finer spatial resolution and a block with a coarser spatial resolution the same volume is represented by one coarser cell with volume $V_1^{(1)}$

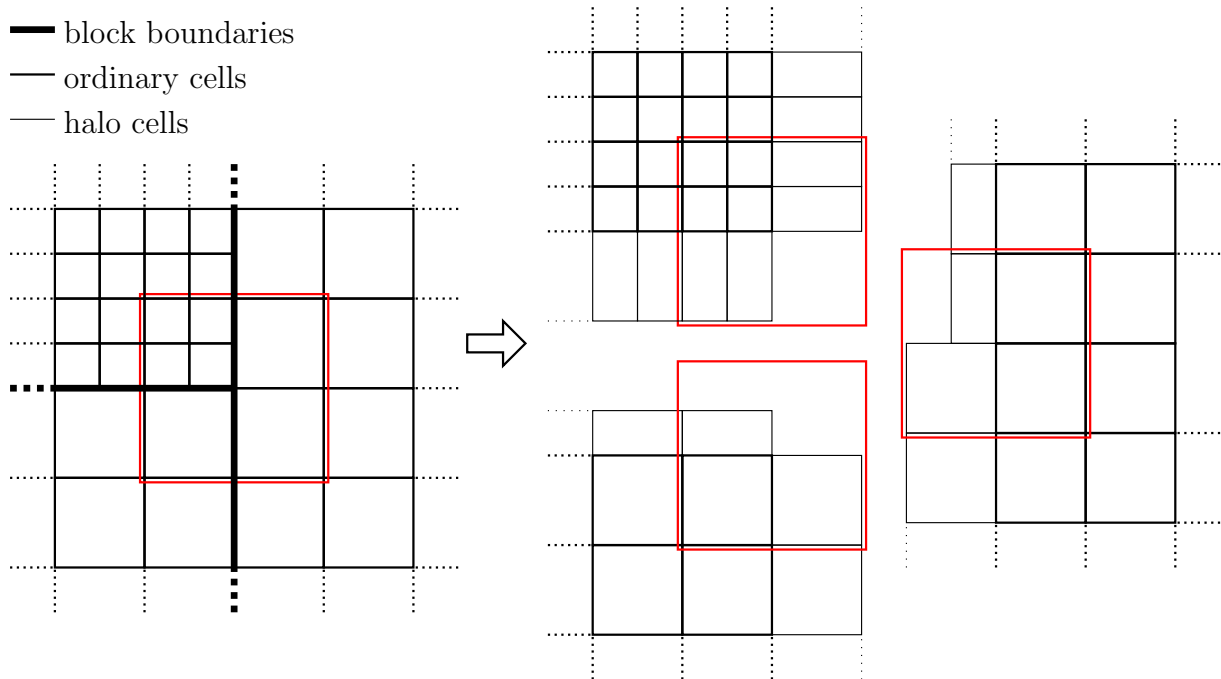


Figure 4.5: Multiple representations of the same physical volume. Left: connected grid with block boundaries. Right: decomposed grid, including halo cells. The red square marks the same physical volume in the different representations.

and by two finer cells with volumes $V_1^{(2)} = V_2^{(2)} = V_1^{(1)}/2$. Thus (4.1) is satisfied by

$$\begin{aligned} c_1^{(1)} V_1^{(1)} &= c_1^{(2)} V_1^{(2)} + c_2^{(2)} V_2^{(2)}, \\ c_1^{(1)} &= 1/2 \left(c_1^{(2)} + c_2^{(2)} \right), \end{aligned}$$

which is how the update of $c^{(1)}$ from $c^{(2)}$ is implemented. Data exchange in the opposite direction is implemented as

$$c_{1,2}^{(2)} := c_1^{(1)}.$$

The different kinds of exchanges work on different sets of cells. First we shall discuss the so called flux exchanges. Even though the term “flux exchange” is illustrative it is not exact. The computed mass fluxes are represented as partial source/sink terms associated with a given cell concentration. If the boundary fluxes are cast as $f_{i\pm 1/2,j}$ and $f_{i,j\pm 1/2}$ with spatial indexes i, j and cell volume $V_{i,j}$ then the temporal derivative of the concentration due to advection reads

$$\frac{d}{dt} c_{i,j} = - \frac{f_{i+1/2,j} - f_{i-1/2,j} + f_{i,j+1/2} - f_{i,j-1/2}}{V_{i,j}}.$$

Due to the definition of the splitting by fluxes as given in Section 3.2.1 each boundary flux is computed exactly once. Consequently a so called flux exchange always is an incremental

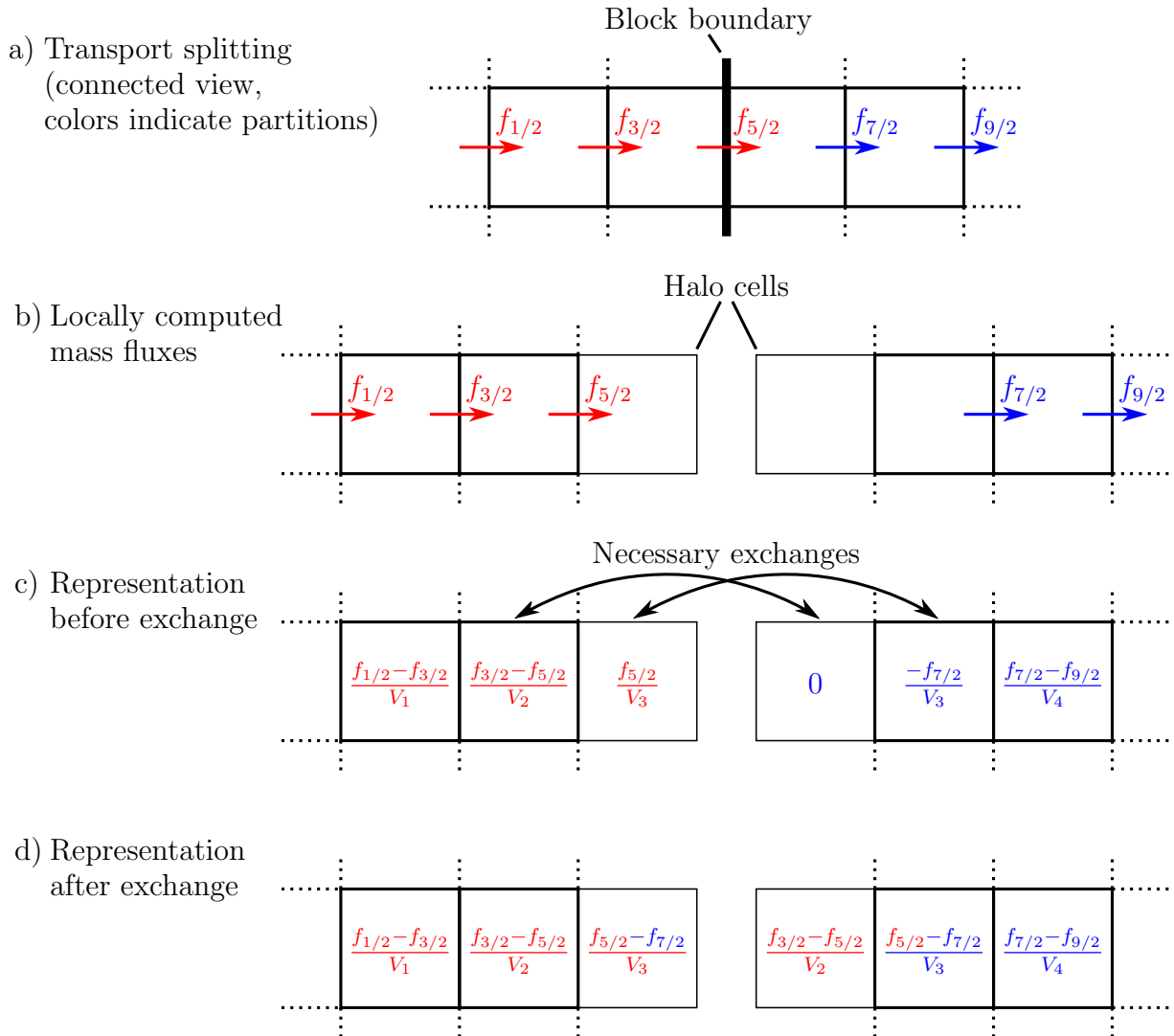


Figure 4.6: Illustration of computed fluxes, their representation and necessary exchanges.

exchange of concentration derivatives from the outermost column of actual cells to the adjacent block's halo cells and from the halo cells to the adjacent block's outermost actual cells, see Figure 4.6 for an illustration.

Since the exchange of concentration derivatives due to horizontal advection is incremental special care has to be taken on implementation. In the case of an exchange of kind (F2), the concentration tendencies are written to an additional array r_{slower} which can be interpreted to hold the boundary conditions for the associated block. Theoretically an exchange of kind (F1) would allow to directly access cells of the extended array. However this would require sophisticated scheduling. In particular an exchange from a block A to a block B would have to take into account whether an exchange from block B to block A occurred before. It is easier to handle if for the (F1) exchanges an additional data structure r_{equal} employed. Considering two blocks and denoting the block local variables

with superscripts the exchange in detail reads as follows. For better readability we mark the values of r before (pre) and after (post) the exchange.

$$\begin{aligned} r_{\text{equal}}^{(1)} &= r^{(2)}@pre, \\ r_{\text{equal}}^{(2)} &= r^{(1)}@pre. \end{aligned}$$

After this exchange the following assignments are done block locally:

$$\begin{aligned} r^{(1)}@post &= r^{(1)}@pre + r_{\text{equal}}^{(1)}, \\ r^{(2)}@post &= r^{(2)}@pre + r_{\text{equal}}^{(2)}, \end{aligned}$$

so that for a given cell index k where both $r^{(1)}$ and $r^{(2)}$ are defined we have

$$r_k^{(1)}@post = r_k^{(2)}@post = r_k^{(1)}@pre + r_k^{(2)}@pre.$$

This is represented by the `FLUXEXCHANGE` routine in the flow chart. An exchange of fluxes from a slower block to a faster block must also take the difference $c_i - c_{i-1}$ (with the nodes of the base method c and the current Runge–Kutta stage index of the slower block i) into account.

Concentration exchanges are more similar to classical exchanges in that they are direct copies (rather than increments) of the involved values. Consequently no additional data structures are necessary for their implementation. To determine how exactly concentration exchanges have to take place it is handy to examine the program flow. Concentration exchange between blocks on the same temporal refinement level (C1) take place at the final step of a Runge–Kutta stage. It consists of a halo cell update, as the concentration in the corresponding actual cell (or column in a three dimensional simulation) may have changed due to cell local chemistry or column local vertical diffusion.

A concentration exchange from a faster block to a slower adjacent block (C2) takes place after the recursive call associated with the faster block’s time level. It must involve an update of the slower neighbor’s halo cells as well as its outermost actual cells.

Finally, a concentration exchange from a slower block to a faster adjacent block (C3) takes place after a correction step on the time level of the slower block. To ensure correct coupling of adjacent blocks the exchange again involves an update of both halo cells and actual cells. An overview of all exchanges is given in Table 4.1.

4.4.2 Coupling of diagonally adjacent blocks

If the employed halo cells would serve solely as receivers for exchanges it would be sufficient to consider pairs of blocks who are directly adjacent. We call two blocks directly adjacent if the same physical volume is represented by an actual cell of one block and a halo cell of the other block. Equivalently direct adjacency can be defined by two blocks sharing a

	F1	F2	C1	C2	C3
actual to halo	X	X	X	X	X
halo to actual	X	X		X	X
incremental	X	X			

Table 4.1: Overview of data exchanges.

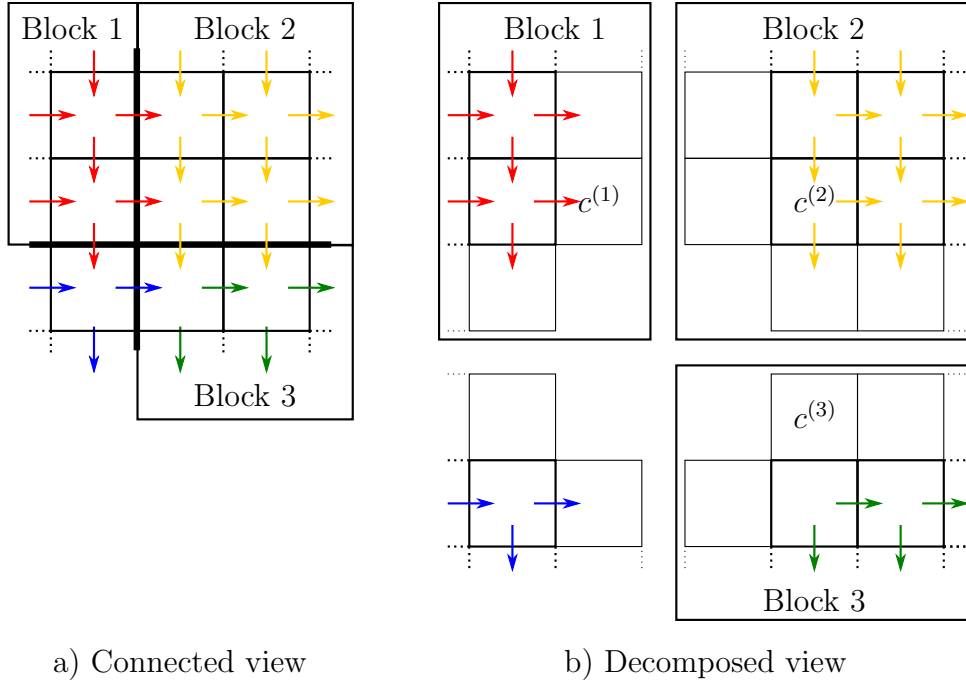


Figure 4.7: Illustration of a local grid decomposition. Blocks 1 and 3 are diagonally adjacent; the marked concentration and all other variables associated with this physical volume are represented three times.

face of a cell. Since the halo cells do not serve only as receivers for exchanges but also as sources, exchanges also have to include pairs of blocks whose halos overlap, i.e. blocks which are *diagonally adjacent*. This kind of exchange deserves special attention, as it may be geometrically more complicated. Additionally we shall demonstrate that it is necessary to take the temporal refinement level of a further block into account.

If two blocks are diagonally adjacent their respective halos overlap with each other while there is no overlap of their halo with the other block's actual cells. The physical volume represented by specific halo cells of both of the diagonally adjacent blocks is also represented by an actual cell of a third block. We call this third block the *mediator* between the first two. Consider a block setup as illustrated in Figure 4.7, with Block 2 being the mediator between the diagonally adjacent Blocks 1 and 3. We specifically consider the physical volume represented by a cell of each of those blocks and denote the fluxes computed by Block k with a superscript (k) for $k \in \{1, 2, 3\}$. We consider

two scenarios which employ the geometrical setup as depicted in Figure 4.7 but different relative temporal refinement levels of the involved blocks.

Scenario 1 - diagonal exchange required

If all blocks operate on the same temporal refinement level L , flux exchanges take place after the calculation of local fluxes on the same level L for a Runge–Kutta stage index i . Flux calculation on level L :

$$\begin{aligned} G_{i-1}^{(k)} &:= G(w^{(k)}); \quad k \in \{1, 2, 3\} \\ r^{(k)} &:= \sum_{j=1}^{i-1} (a_{i,j} - a_{i-1,j}) G_j^{(k)}; \quad k \in \{1, 2, 3\} \end{aligned}$$

After the exchange all blocks must hold the full information about the advective fluxes so that we need an exchange of the kind:

$$\begin{aligned} r_{\text{equal}}^{(1)} &:= r^{(2)} + r^{(3)}; \\ r_{\text{equal}}^{(2)} &:= r^{(1)} + r^{(3)}; \\ r_{\text{equal}}^{(3)} &:= r^{(1)} + r^{(2)}; \end{aligned}$$

The above exchanges may be executed in any order. In general imposing a specific order of exchanges will have a negative effect on performance as an additional synchronization of different processes is involved. Consequently the above exchange must be finalized by a block local assignment:

$$r^{(k)} := r^{(k)} + r_{\text{equal}}^{(k)}; \quad k \in \{1, 2, 3\}$$

It is evident that the values of r before (pre) and after (post) the exchange satisfy:

$$r^{(1)}@post = r^{(2)}@post = r^{(3)}@post = r^{(1)}@pre + r^{(2)}@pre + r^{(3)}@pre.$$

Scenario 2 - diagonal exchange not required

Now consider the case that Block 1 operates on temporal refinement level L whereas Blocks 2 and 3 operate on level $L+1$. We now also have to consider if the current Runge–Kutta stage on level L is involved with a recursive call (i.e. $c_i > c_{i-1}$) or with a simple correction step (i.e. $c_i = c_{i-1}$). In the latter case the flux exchange from level L to level $L+1$ is unnecessary. We now consider the case that $c_i > c_{i-1}$: Flux calculation on level L :

$$\begin{aligned} G_{i-1}^{(1)} &:= G(w^{(1)}); \\ r^{(1)} &:= \sum_{j=1}^{i-1} (a_{i,j} - a_{i-1,j}) G_j^{(1)}; \end{aligned}$$

Flux exchange from level L to level $L + 1$:

$$r_{\text{slower}}^{(2)} := r^{(1)} [2(c_i - c_{i-1})] / (c_i - c_{i-1});$$

Note that the additional factor is needed to compensate for the ratio of the time step size Δt on level L to the time step size Δt_μ on level $L+1$ with $\Delta t_\mu = (c_i - c_{i-1}) / [2(c_i - c_{i-1})] \Delta t$. Flux calculation on level $L + 1$:

$$\begin{aligned} G_{\bar{i}-1}^{(k)} &:= G(w^{(k)}) + r_{\text{slower}}^{(k)}; \quad k \in \{2, 3\} \\ r^{(k)} &:= \sum_{j=1}^{\bar{i}-1} (a_{\bar{i},j} - a_{\bar{i}-1,j}) G_j^{(k)}; \quad k \in \{2, 3\}, \end{aligned}$$

with \bar{i} indicating the Runge–Kutta stage index on level $L + 1$. At this point $r^{(2)}$ already holds all information on fluxes computed by Blocks 1 and 2.

Flux exchange on level $L + 1$:

$$\begin{aligned} r_{\text{equal}}^{(2)} &:= r^{(3)}; \\ r_{\text{equal}}^{(3)} &:= r^{(2)}; \end{aligned}$$

Block local finalization on level $L + 1$:

$$r^{(k)} := r^{(k)} + r_{\text{equal}}^{(k)}; \quad k \in \{2, 3\}$$

At this point we have:

$$\begin{aligned} r^{(2)} @ \text{post} = r^{(3)} @ \text{post} &= r_{\text{slower}}^{(2)} + r^{(2)} @ \text{pre} + r^{(3)} @ \text{pre} \\ &= r^{(1)} [2(c_i - c_{i-1})] / (c_i - c_{i-1}) + r^{(2)} @ \text{pre} + r^{(3)} @ \text{pre}. \end{aligned}$$

Concludingly we remark that in this case the mediator block actually serves as a mediator for the exchange. This is efficient as the flux exchange on level $L + 1$ is necessary anyway after flux calculation on this level. Furthermore applying a diagonal exchange in this scenario would lead to wrong results as the source term provided by Block 1 would falsely be applied twice.

Further scenarios can be examined by considering the program flow and the values of certain block local variables. Generally a diagonal flux exchange is unnecessary if and only if the geometrical mediator block serves as an exchange mediator. For flux exchanges this is the case whenever

- The receiving block is on a higher temporal refinement level than the sending block (i.e. an exchange of kind (F2) takes place) and
- the mediator block is on the equal temporal refinement level as the receiving block.

Similar considerations hold for diagonal concentration updates. Here a diagonal exchange can and must be omitted if the mediator's time level equals the receiving block's time level.

A constraint to the grid geometry is that the spatial resolution of two directly adjacent blocks may differ by a factor of two maximum. Consequently we must consider nine geometrical setups for diagonal adjacency; the mediating block may be finer, coarser or equally fine than/as the sending block and analogously the receiving block may be finer, coarser or equally fine than/as the mediator, see Figure 4.8. The adjacency indexes AI have been assigned for implementation reasons. If the roles of sender and receiver are swapped, the sign of the adjacency index is flipped. Consequently the symmetric setups $AI \in \{4, -4\}$ and $AI \in \{5, -5\}$ are assigned to pairs of indexes. We shall now shortly discuss how these different setups are treated. For most geometrical setups the diagonal exchange is straightforward. It is more complicated if the sending block has information on only half of the receiving cell's volume, which is the case for the bottom row cases depicted in Figure 4.8. We closely examine the case $AI = 1$. This case and its symmetric counterpart $AI = -1$ are very unlikely but nonetheless allowed setups. Denoting the different quantities with a superscript (S) for the representation in the sending block and a superscript (R) for the representation in the receiving block the flux exchange for $AI = 1$ must be implemented as follows. To abstract from the specific exchange we denote the concentration tendencies by \dot{c} . For a better understanding consider the volumes of the cells involved in this exchange

$$V_1^{(S)} = V_2^{(S)} = \frac{1}{4}V^{(R)},$$

and interpret the exchange as an exchange of mass fluxes. Then

$$\begin{aligned} \dot{c}^{(R)} &= \dot{m}^{(R)}/V^{(R)} \\ &= \left(\dot{m}_1^{(S)} + \dot{m}_2^{(S)} \right) / V^{(R)} \\ &= \left(\dot{c}_1^{(S)}V_1^{(S)} + \dot{c}_2^{(S)}V_2^{(S)} \right) / V^{(R)} \\ &= \frac{1}{4} \left(\dot{c}_1^{(S)} + \dot{c}_2^{(S)} \right). \end{aligned}$$

This exchange is valid because it actually is the exchange of a mass flux (though not implemented as such) across a cell boundary which is represented both in the sending block and the receiving block. The *concentration* exchange for $AI = 1$ on the other hand cannot be defined consistently, as the sending block has information on only half of the receiving cell's physical volume. The error is minimized implementing this exchange as

$$c^{(R)} := \frac{1}{2}c^{(R)} + \frac{1}{4} \left(c_1^{(S)} + c_2^{(S)} \right).$$

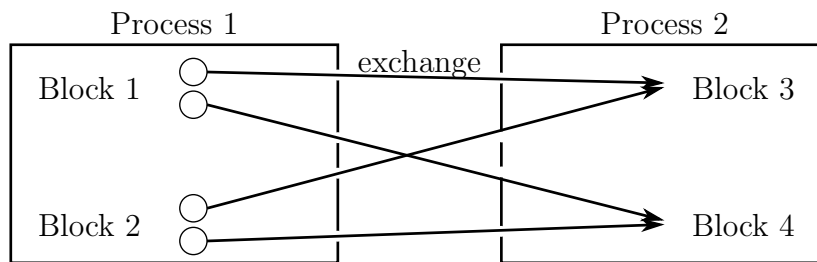
A summary of diagonal exchanges is given in Table 4.2.

	finer mediator	equally fine mediator	coarser mediator
finer receiver	 $AI = -1$	 $AI = -2$	 $AI \in \{4, -4\}$
equally fine receiver	 $AI = -3$	 $AI = 0$	 $AI = 2$
coarser receiver	 $AI \in \{5, -5\}$	 $AI = 3$	 $AI = 1$

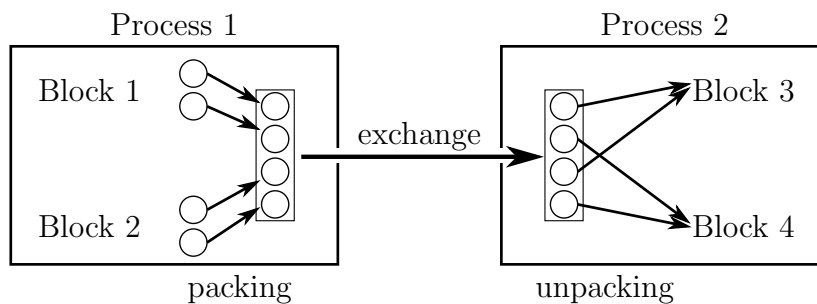
Figure 4.8: Possible geometries of diagonal adjacency with adjacency index AI . Upper left block is sending, lower right block is receiving. Mediators are called fine or coarse relative to the sender; receivers are called fine or coarse relative to the mediator.

$AI = 0$	$c^{(R)} := c^{(S)}$ $\dot{c}^{(R)} := \dot{c}^{(S)}$
$AI = 1$	$c^{(R)} := 1/2 c^{(R)} + 1/4 (c_1^{(S)} + c_2^{(S)})$ $\dot{c}^{(R)} := 1/4 (\dot{c}_1^{(S)} + \dot{c}_2^{(S)})$
$AI = -1$	$c_1^{(R)} := c^{(S)}$ $c_2^{(R)} := c^{(S)}$ $\dot{c}_1^{(R)} := \dot{c}^{(S)}$ $\dot{c}_2^{(R)} := \dot{c}^{(S)}$
$AI = 2$	$c^{(R)} := 1/2 (c_1^{(S)} + c_2^{(S)})$ $\dot{c}^{(R)} := 1/2 (c_1^{(S)} + c_2^{(S)})$
$AI = -2$	$c_1^{(R)} := c^{(S)}$ $c_2^{(R)} := c^{(S)}$ $\dot{c}_1^{(R)} := \dot{c}^{(S)}$ $\dot{c}_2^{(R)} := \dot{c}^{(S)}$
$AI = 3$	$c^{(R)} := 1/2 c^{(R)} + 1/2 c^{(S)}$ $\dot{c}^{(R)} := 1/2 c^{(S)}$
$AI = -3$	$c^{(R)} := c^{(S)}$ $\dot{c}^{(R)} := \dot{c}^{(S)}$
$AI \in \{4, -4\}$	$c_1^{(R)} := 1/2 (c_1^{(S)} + c_2^{(S)})$ $c_2^{(R)} := 1/2 (c_1^{(S)} + c_2^{(S)})$ $\dot{c}_1^{(R)} := 1/2 (\dot{c}_1^{(S)} + \dot{c}_2^{(S)})$ $\dot{c}_2^{(R)} := 1/2 (\dot{c}_1^{(S)} + \dot{c}_2^{(S)})$
$AI \in \{5, -5\}$	$c^{(R)} := 1/2 c^{(R)} + 1/2 c^{(S)}$ $\dot{c}^{(R)} := 1/2 c^{(S)}$

Table 4.2: Summary of diagonal exchanges' implementations.



a) Naive exchange (block wise communication)



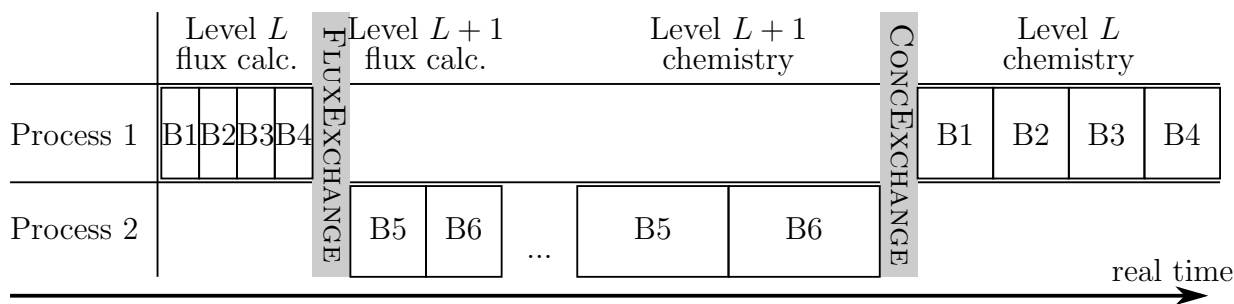
b) Exchange with packing/unpacking (process wise communication)

Figure 4.9: Illustration of block-wise communication versus process-wise communication. Circles symbolize data packages to be exchanged.

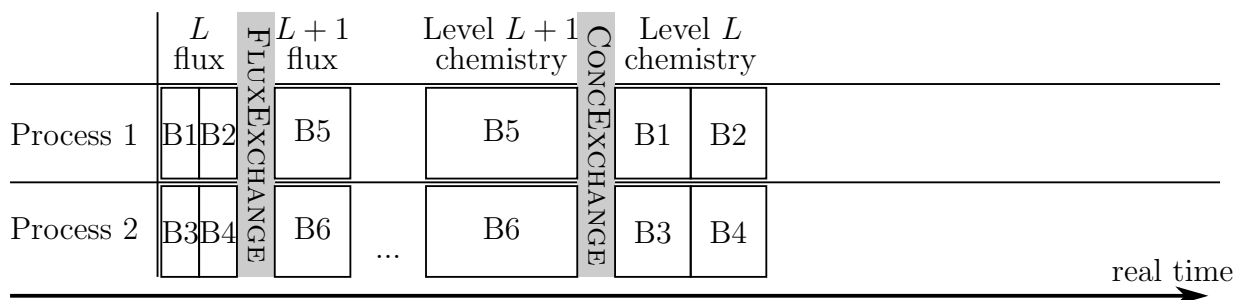
4.4.3 Exchanges for a parallel implementation

In order to optimize exchanges for parallel implementation one has to take into account how much data is exchanged and how often data is exchanged. The latter is important since several exchanges of given data volumes between exchange partners are more time consuming than a single exchange of the same cumulative volume. This can be exploited due to the fact that usually several blocks are associated with each of the employed processes. By first packing data, then sending/receiving data and finally unpacking, it is provided that each of the exchanges mentioned in the pseudo code can be implemented by one parallel exchange for each pair of processes instead of one parallel exchange for each pair of blocks, see Figure 4.9.

There is also potential for reducing the data volume whenever two directly adjacent blocks with different spatial resolutions exchange data. If data is sent from a coarser block to a more finely resolved block it is advantageous to send one scalar per sender's cell and apply it on the two receiving cells on unpacking. If on the other hand data is sent from a finer block to a coarser block it is better to average the values on packing so that one scalar per receiver's cell is exchanged.



a) Multi-process workflow for worst case block distribution.



b) Multi-process workflow for perfect block distribution.

Figure 4.10: Multi process workflow for different block-process mappings.

4.5 Balancing

If a simulation is to be distributed on multiple cores of one processor, multiple processors of a workstation or even multiple nodes of a computing cluster, it must be split in several parts. In the context of MUSCAT these parts are the different blocks. Those blocks have to be assigned to the available computing elements such that idle times are minimized. For classical time integration schemes this can be implemented by performing *workload balancing*. The Metis/ParMetis libraries [26] provide sophisticated balancing algorithms. Balancing problems are interpreted as a class of graph theoretical problems which may be subsumed as “minimize the edge cut without violating node balancing constraints”. Blocks of the decomposed simulation domain are mapped to nodes of the graph with one scalar weight per node proportional to the workload associated with the block. Two nodes are connected by an edge if the blocks are adjacent, the communication volume between those blocks is modeled by a scalar weight per edge. Consequently minimizing the edge cut is equivalent to minimizing the communication between partitions or processors while the node balancing constraints are to be understood as workload balancing constraints.

As illustrated in Figure 4.4 the program flow of MUSCAT is naturally structured into phases due to recursive calls. A direct effect of this multi-phase program flow is that simple workload balancing is not generally efficient to minimize idle times. Consider for instance a small set of blocks to be distributed on two processes and the resulting excerpt of the program flow as shown in Figure 4.10 and note that both cases shown correlate

to perfect naive workload balancing. The important difference is that in Figure 4.10b the blocks are mapped to the processes such that the workload *for each of the phases* is balanced. For the implementation we are discussing here this requirement is met when each of the temporal refinement levels is balanced.

The Metis/ParMetis libraries offer the possibility of *multi constraint balancing* [25]. Opposed to classical balancing (i.e. single constraint balancing) this considers a vector of weights for each node and aims at an even distribution for each vector dimension. For our algorithm that means that the weight vector w associated with a block reads

$$w \in \mathbb{R}^N, \quad w_k = \begin{cases} C & \text{if } k = L, \\ 0 & \text{otherwise,} \end{cases}$$

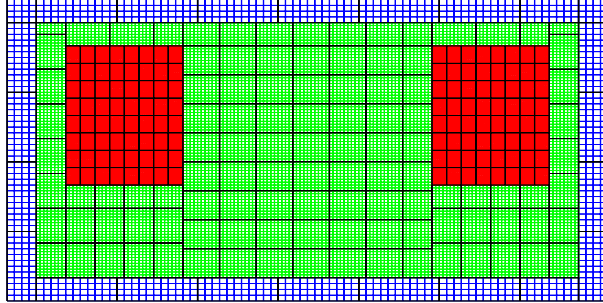
with $N = L_{\max} + 1$ denoting the number of time levels throughout the simulation, the block's time level L and the number of columns C within the block. Choosing this approach will probably not lead to satisfying results as the constraints leave only little margin for optimization. As a compromise we employ three constraints correlated to the highest temporal refinement level L_{\max} , the second highest temporal refinement level and the remaining levels. The rationale for this is that in most setups the two highest refinement levels will cause the bigger part of computational cost. Consequently balancing blocks on these levels will lead to an acceptable tradeoff between few constraints and little idle times. We define the three dimensional weight vector as follows:

$$w = \begin{cases} (C, 0, 0) & \text{if } L = L_{\max}, \\ (0, C, 0) & \text{if } L = L_{\max} - 1, \\ (0, 0, 2^L C) & \text{otherwise,} \end{cases}$$

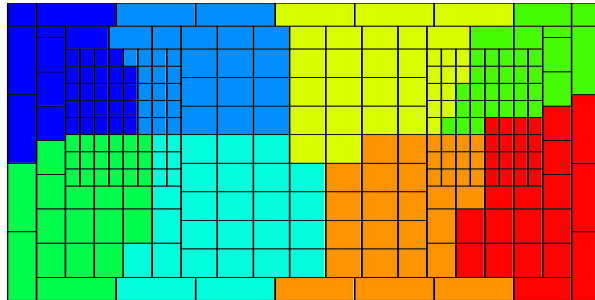
with L_{\max} denoting the maximum temporal refinement level. The factor 2^L in the third case is needed to account for the relative computational cost of blocks on potentially different time levels. It is not needed for the first two components of the vector, as their absolute weights are not taken into account by the ParMetis library. To prioritize balancing of the highest temporal refinement levels over the remaining levels, a smaller and thus stricter margin of tolerance is provided for the first component of the weight vector.

Computational tests with realistic scenarios show ambivalent results. An example is given in Figure 4.11. Single constraint balancing as shown in Figure 4.11b) leads to strong imbalances as the processor represented by green (top right) is associated with 25 of the blocks on the highest time level while the processor represented by cyan (bottom, half left) is associated with only 6 of the blocks on this level. Multi constraint balancing on the other hand leads to perfect distribution of blocks on the highest time level, see Figure 4.11c). However the sets of blocks associated with different processors are not connected anymore which leads to increased communication.

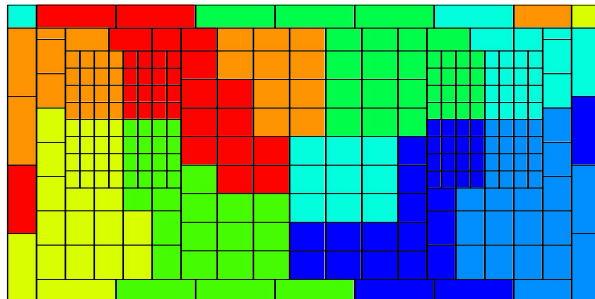
Summarizing we note that while multi constraint balancing is suitable to minimize idle times during computation it generally leads to higher communication cost, as the optimization of communication is hindered by more constraints. Thus it is generally



a) Grid structure; blue cells are associated with time level $L = 0$, green cells with $L = 1$, red cells with $L = 2$. All blocks on time level $L = 2$ have the same number of cells.



b) Distribution on 8 processors computed by single constraint balancing; colors indicate mapping to processors.



c) Distribution on 8 processors computed by multi constraint balancing; colors indicate mapping to processors.

Figure 4.11: Example for workload balancing.

recommendable for such simulations in which local computations take significantly more time than data exchange, e.g. simulations involving computationally expensive chemistry or microphysics. If in contrast to that a simulation is communication dominated, only little parallelization speedup can be expected even for an optimal distribution of blocks on different processors.

Chapter 5

Numerical Tests

In this chapter we present some numerical tests designed to examine different aspects of the splitting approach presented in Chapter 3. The programs employed in this chapter are prototypes which did not undergo thorough manual optimizations.

5.1 Comparison of splitting approaches

5.1.1 Setup

To examine the influence of the different splitting approaches described in Section 3.2.1 we consider advection with unit speed on the unit interval with periodic boundary conditions:

$$\begin{aligned}\frac{\partial}{\partial t}y &= -\frac{\partial}{\partial x}y, \\ y(x, 0) &= \sin^{16}(\pi x), \\ y(1, t) &= y(0, t), \quad (t \in [0, 1]).\end{aligned}$$

The system is spatially discretized using an equidistant grid and the third order upwind discretization (2.7). For time integration we employ the RFSMR approach with (RK43) as base method, cf. Table 3.3. The domain is split into four parts which are assigned alternately to the slow and the fast partition. We examine the errors obtained with a splitting by cells and a splitting by fluxes and compare the results obtained with the different splittings by means of the error ratio

$$ER = \frac{Err(y(t=1)_{\text{fluxsplit}})}{Err(y(t=1)_{\text{cellsplit}})} - 1,$$

where *Err* represents either the error in L_1 -norm or the error in L_∞ -norm.

5.1.2 Results

First we consider convergence with respect to the time step for a fixed grid size, see Figure 5.1. The reference solution is obtained by solving the system with a time step

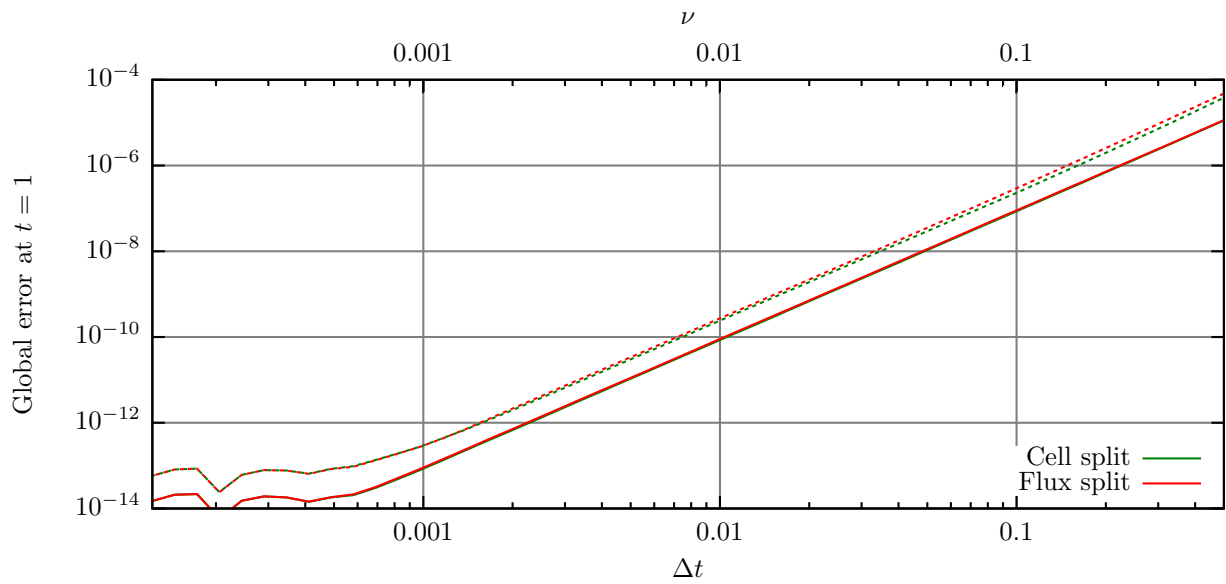


Figure 5.1: Error in L_1 -norm (solid) and L_∞ -norm (dashed) for a constant grid size of $\Delta x = 0.01$.

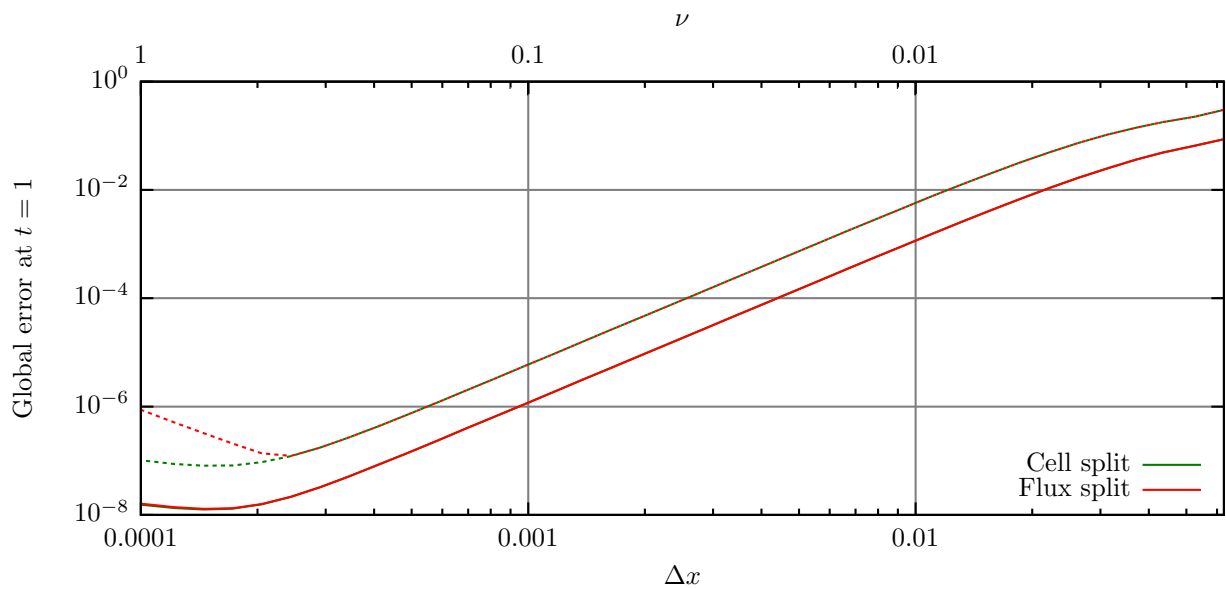


Figure 5.2: Error in L_1 -norm (solid) and L_∞ -norm (dashed) for a constant time step of $\Delta t = 10^{-5}$.

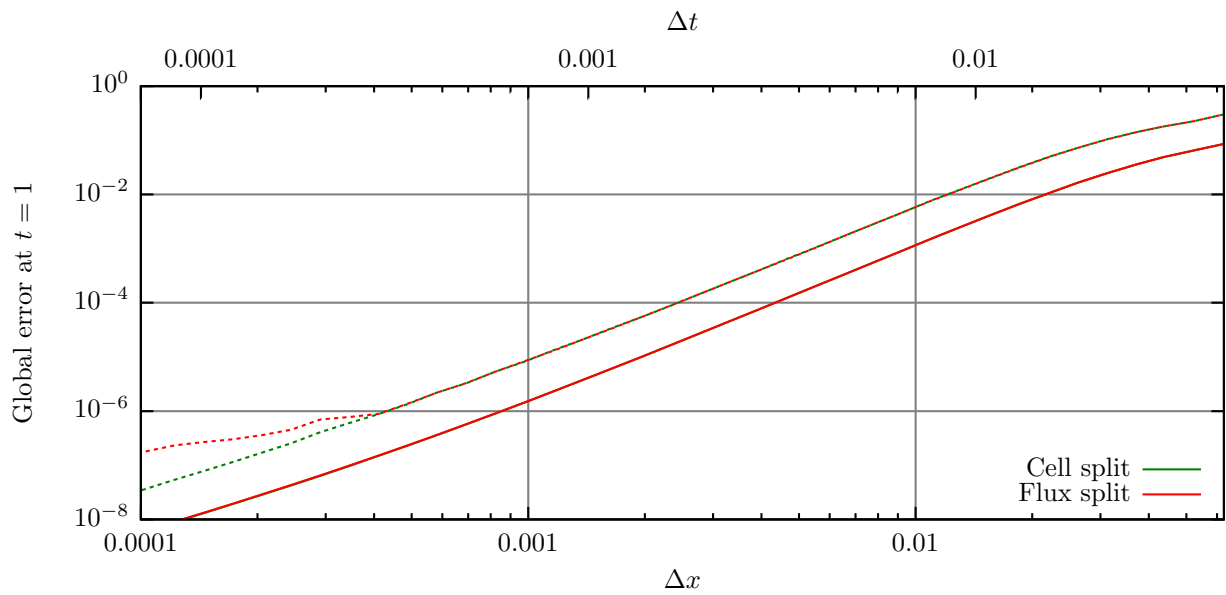


Figure 5.3: Error in L_1 -norm (solid) and L_∞ -norm (dashed) for a constant Courant number of $\nu = 0.7$.

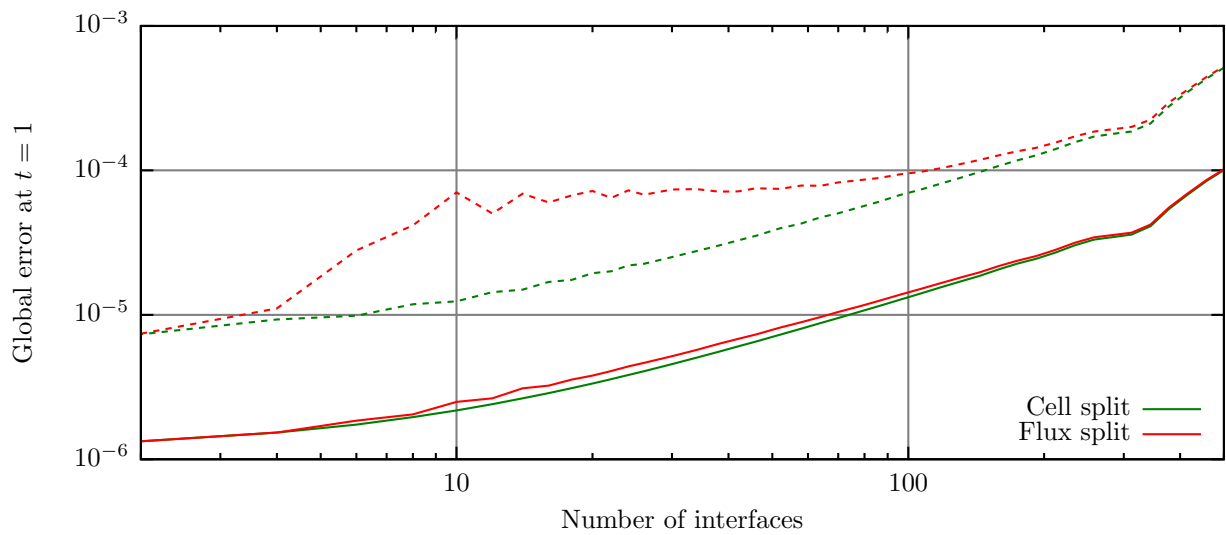


Figure 5.4: Error in L_1 -norm (solid) and L_∞ -norm (dashed) versus number of interfaces for $\nu = 0.7$; $\Delta x = 10^{-4}$.

base method	first order upwind			third order upwind		
	$\nu = 0.5$	$\nu = 0.75$	$\nu = 1$	$\nu = 0.5$	$\nu = 0.75$	$\nu = 1$
RK1	$-5.37 \cdot 10^{-4}$	$-8.13 \cdot 10^{-4}$	$-1.09 \cdot 10^{-3}$	—	—	—
RK2a	$-2.27 \cdot 10^{-8}$	$-2.63 \cdot 10^{-8}$	$7.34 \cdot 10^{-4}$	$-2.72 \cdot 10^{-4}$	$2.63 \cdot 10^{-4}$	$1.29 \cdot 10^{-3}$
RK2b	$9.79 \cdot 10^{-8}$	$3.29 \cdot 10^{-7}$	$2.00 \cdot 10^{-4}$	$-1.91 \cdot 10^{-4}$	$4.92 \cdot 10^{-4}$	$8.76 \cdot 10^{-4}$
RK32	$-2.63 \cdot 10^{-8}$	$-6.60 \cdot 10^{-8}$	$-9.56 \cdot 10^{-8}$	$-2.60 \cdot 10^{-4}$	$-3.21 \cdot 10^{-5}$	$6.16 \cdot 10^{-4}$
RK43	$6.49 \cdot 10^{-9}$	$2.19 \cdot 10^{-8}$	$5.19 \cdot 10^{-8}$	$2.18 \cdot 10^{-5}$	$5.33 \cdot 10^{-3}$	$2.58 \cdot 10^{-2}$

Table 5.1: L_1 -error ratios for different discretizations.

base method	first order upwind			third order upwind		
	$\nu = 0.5$	$\nu = 0.75$	$\nu = 1$	$\nu = 0.5$	$\nu = 0.75$	$\nu = 1$
RK1	$-3.7 \cdot 10^{-3}$	$-6.6 \cdot 10^{-3}$	$-1.0 \cdot 10^{-2}$	—	—	—
RK2a	$3.8 \cdot 10^{-5}$	$3.5 \cdot 10^{-5}$	$9.2 \cdot 10^{-4}$	$4.27 \cdot 10^{-11}$	$3.39 \cdot 10^{-12}$	$2.29 \cdot 10^{-10}$
RK2b	$-4.3 \cdot 10^{-5}$	$-7.8 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	$1.08 \cdot 10^{-10}$	$-1.66 \cdot 10^{-11}$	$8.42 \cdot 10^{-11}$
RK32	$3.4 \cdot 10^{-5}$	$4.3 \cdot 10^{-5}$	$1.0 \cdot 10^{-4}$	$-1.39 \cdot 10^{-8}$	$2.05 \cdot 10^{-11}$	$2.30 \cdot 10^{-12}$
RK43	$-1.0 \cdot 10^{-5}$	$-3.1 \cdot 10^{-5}$	$-5.6 \cdot 10^{-5}$	$4.80 \cdot 10^{-3}$	$2.17 \cdot 10^{-3}$	$-2.87 \cdot 10^{-2}$

Table 5.2: L_∞ -error ratios for different discretizations.

of $\Delta t = 10^{-5}$. As expected from the theoretical treatment, we can observe third order of convergence for larger time steps. For time steps smaller than 0.01 no further reduction of the error can be obtained due to round-off errors. The largest relative differences in the errors amount to 2% in the L_1 -norm and 25% in the L_∞ -norm, observed for a Courant number of $\nu = 1/2$.

Second we examine convergence with respect to the grid size for a fixed time step, see Figure 5.2. Now we consider the error in terms of the difference to the exact solution. For grid sizes $\Delta x > 8 \cdot 10^{-5}$, only negligible differences in the errors of the splitting approaches can be observed. Small grid sizes lead to large Courant numbers so that the error introduced by the time integration scheme becomes dominant. Clear differences can be seen for the ratio of the L_∞ -errors for small grid sizes.

Next we consider convergence for a fixed Courant number of $\nu = 0.7$, cf. Figure 5.3. Again we observe that the L_∞ -error of the flux splitting significantly exceeds the L_∞ -error of the cell splitting for small grid sizes, while the L_1 -errors differ by 2% maximum.

Tests with varying number of interfaces for a fixed grid size of $\Delta x = 10^{-4}$ and a fixed Courant number of $\nu = 0.7$ show the expected dependency of the splitting error on the number of interfaces, cf. Figure 5.4; more interfaces lead to an increased global error.

Motivated by the above results we examined a further test case with different spatial discretizations and time integration methods. We chose a grid size of $\Delta x = 10^{-3}$ and considered Courant numbers of $\nu \in \{0.5, 0.75, 1\}$, which are in the order of magnitude of Courant numbers employed for atmospheric simulations. Results in terms of the error ratios are shown in Tables 5.1 and 5.2; note that negative values indicate that the error obtained with a splitting by fluxes is lower than the error obtained with a splitting by cells. It shows that both alternatives lead to similar errors.

5.1.3 Conclusions

For high precision computations involving a small grid size and a small time step, a splitting by fluxes leads to larger errors than a splitting by cells. For computations with comparatively low accuracy demands both splittings perform equally well. For large Courant numbers both approaches yield similar results.

5.2 Advection-reaction

5.2.1 Setup

To demonstrate a true IMEX splitting and affirm the results of the stability analysis of Section 3.5 we consider uniform advection in one spatial dimension coupled with a reaction rate equation similar to Robertson's equation [36]:

$$\begin{aligned}\frac{\partial}{\partial t}c &= -\frac{\partial}{\partial x}c + R(c), \\ c(x=1, t) &= c(x=0, t) \\ c(x, t=0) &= \begin{pmatrix} \sin^{16}(\pi x) \\ 0 \\ 0 \end{pmatrix}, \\ R(c) &= \begin{pmatrix} -k_1c_1 + k_2c_2c_3 \\ k_3c_1 - k_4c_2c_3 - k_5c_2^2 \\ k_6c_2^2 \end{pmatrix}, \\ k_1 &= 10^{-4}k_3 = 0.4, \\ k_2 &= 10^{-4}k_4 = 0.1, \\ k_5 &= 10^{-3}k_6 = 3 \cdot 10^5.\end{aligned}$$

Note that the time constants k differ from those employed in the original article. Spatially the unit interval is discretized in 500 cells of size $h = 1/500$. The advection operator used is the positive third order upwind biased method. The outer method is either (RK43) or (RK43b). As inner integrator we employ the two stage third order SDIRK method (SDIRK3). To compute the stages of the implicit method we employ a Newton iteration and update the Jacobian for each iteration step. The analytic Jacobian is employed. The Newton iteration terminates as soon as the L_1 -norm of the difference of two successive iterates is lower than 10^{-10} or the number of steps exceeds 20. We take the maximum number of Newton steps employed for the solution of one system in the course of the integration as a measure for the stability and assume that the system is unstable if 20 or more iteration steps are employed.

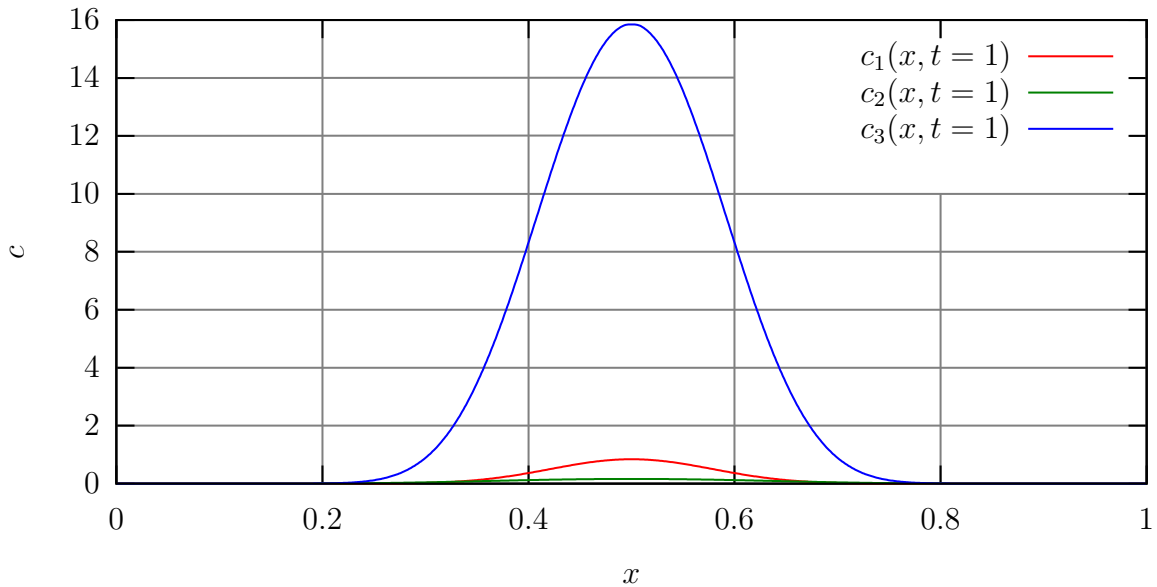


Figure 5.5: Reference solution for advection-reaction test at $t = 1$.

h	1/25	1/50	1/100	1/200	1/400	1/800	1/1600
$\nu_{\max}(\text{RK43})$	0.591	0.644	0.671	0.679	0.696	0.725	0.774
$\nu_{\max}(\text{RK43b})$	0.707	1.055	1.487	1.570	1.840	1.825	1.808

Table 5.3: Empiric stability limit depending on spatial resolution for positive third order upwind discretization.

5.2.2 Results

First we compute reference solutions at $t = 1$ for each of the ERKs in consideration with a time step of $\Delta t = 10^{-6}$, resulting in a Courant number of $\nu = 5 \cdot 10^{-5}$. The resulting profile is shown in Figure 5.5. To examine the convergence behavior we solve the system with different time steps and examine the weighted L_1 difference to the reference solution, see Figure 5.6. It shows that (RK43b) is more stable, while (RK43) yields smaller errors for Courant numbers $0.1 < \nu < 0.6$. For a Courant number of $\nu = 0.2$ the error obtained with (RK43b) is approximately six times as large as the error obtained with (RK43).

Finally we examine how the spatial resolution influences stability. We determined the stability limit precise down to three digits after the decimal point with a logarithmic search, see Table 5.3. For few (large) cells, both methods show similar stability limits, for

h	1/25	1/50	1/100	1/200	1/400	1/800	1/1600
$\nu_{\max}(\text{RK43})$	0.769	0.727	0.706	0.692	0.704	0.734	0.780
$\nu_{\max}(\text{RK43b})$	1.286	1.870	2.352	2.161	2.105	2.049	2.024

Table 5.4: Empiric stability limit depending on spatial resolution for first order upwind discretization.

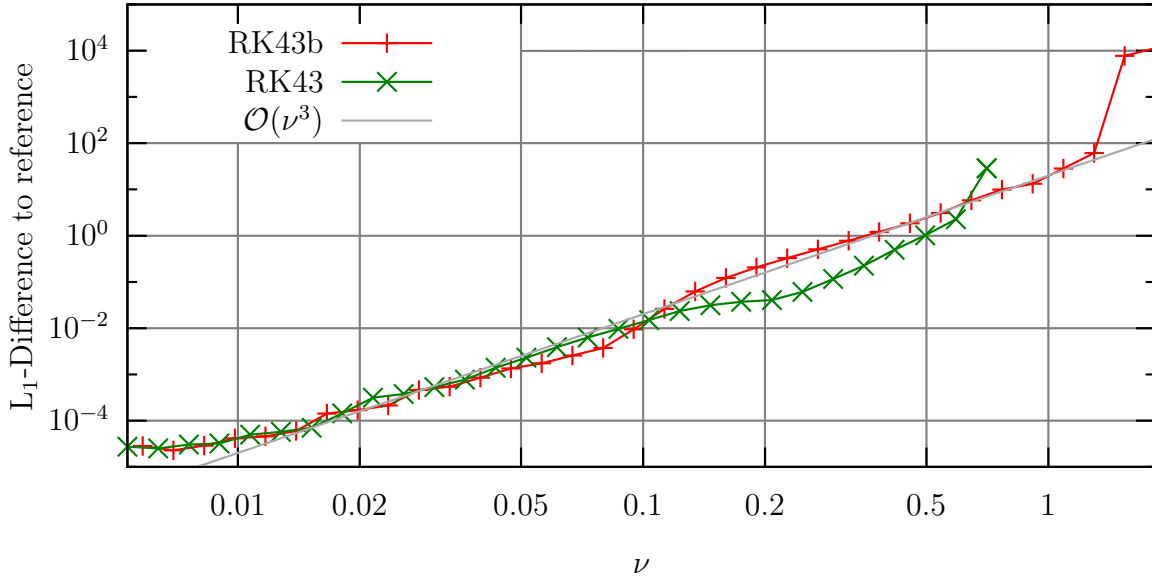


Figure 5.6: Error with respect to the Courant number for advection-reaction test.

more than 100 cells ($h < 1/100$) the stability limit approaches the theoretical prediction. To examine in how far this results from the nonlinearity of the spatial discretization we repeated the stability test with a first order upwind spatial discretization, see Table 5.4. Here the spatial resolution has a much smaller influence on the maximum Courant number. A further test with the third order upwind discretization (without limiter) was not possible as this method is not positivity preserving and thus is unstable for the test equation employed here.

5.2.3 Conclusions

The numerical results presented in this section affirm the theoretical results of the stability analysis presented in Section 3.5. On coarse spatial resolutions however the stability of the positive third order upwind discretization cannot be conservatively predicted based on the analysis of the third order upwind and first order upwind discretization. The four stage third order ERK optimized for stability (RK43b) shows the predicted stability in numerical tests.

5.3 Fire spread

5.3.1 Setup and general properties

We examine a diffusion-reaction equation describing a very simple model of fire spread:

$$\frac{\partial}{\partial t} \begin{pmatrix} \vartheta \\ c \end{pmatrix} = \underbrace{k_0 \vartheta c \begin{pmatrix} \vartheta \\ -c \end{pmatrix}}_{\text{reaction}} - \underbrace{k_1 \vartheta \begin{pmatrix} \vartheta \\ 0 \end{pmatrix}}_{\text{cooling}} + \underbrace{\nabla \left(D \nabla \begin{pmatrix} \vartheta \\ 0 \end{pmatrix} \right)}_{\text{diffusion}}$$

with the reaction speed $k_0 = 15$ the cooling speed $k_1 = 0.1$ and the heat diffusion coefficient $D = 10^{-4}$ on a square domain of $(x, y) \in [0, 1] \times [0, 1]$. We employ zero gradient boundary conditions. The initial condition is given by

$$\begin{aligned} c_0(x, y) &= y, \\ \theta_0(x, y) &= \begin{cases} 100 & \text{if } (x, y) \in [0, 0.01] \times [0, 0.01], \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The dynamics of the system are very interesting: after a brief initialization period during which the “ignition spark” is diffused, the system evolves slowly until reaching $t \approx 34$ (6.6% of total mass burned). Then peak temperature and speed increase significantly for a short period until $t \approx 39$ (91% of total mass burned), see also Figure 5.7. At all times the “fast” components are localized at the flame front, which makes this scenario an interesting candidate to solve with a multirate method, especially in the context of local time step adaptation.

5.3.2 Discretization and implementation

The domain is spatially discretized in 100 by 100 cells, the common second order central diffusion discretization was employed in a split-dimensional fashion. For time integration we employ a multirate scheme based on (RK32). This scheme is of second order accuracy, has good stability properties and has the advantage of conveniently providing a first order accurate solution as well, which is handy for automatic time step control as is employed for the chemistry solver. The reaction equation, including source terms representing the heat transport, is solved explicitly using an adaptive number of steps which is an individual integer for each cell. It proved efficient to first make one tentative step and then adapt the number of steps to be taken. Thus the overhead of a step size reduction is limited to the cost of a single time step. Tests of replacing the explicit chemistry solver by an implicit Runge–Kutta method of the same order showed poorer performance in terms of error compared to computational cost.

For the application of a multirate method on the diffusion equation we employ a splitting by components, which seems reasonable as the PDE does not have linear invariants whose conservation has to be ensured. If a multirate method is employed we choose the local time step to be proportional to a worst case estimation of the temperature gradient.

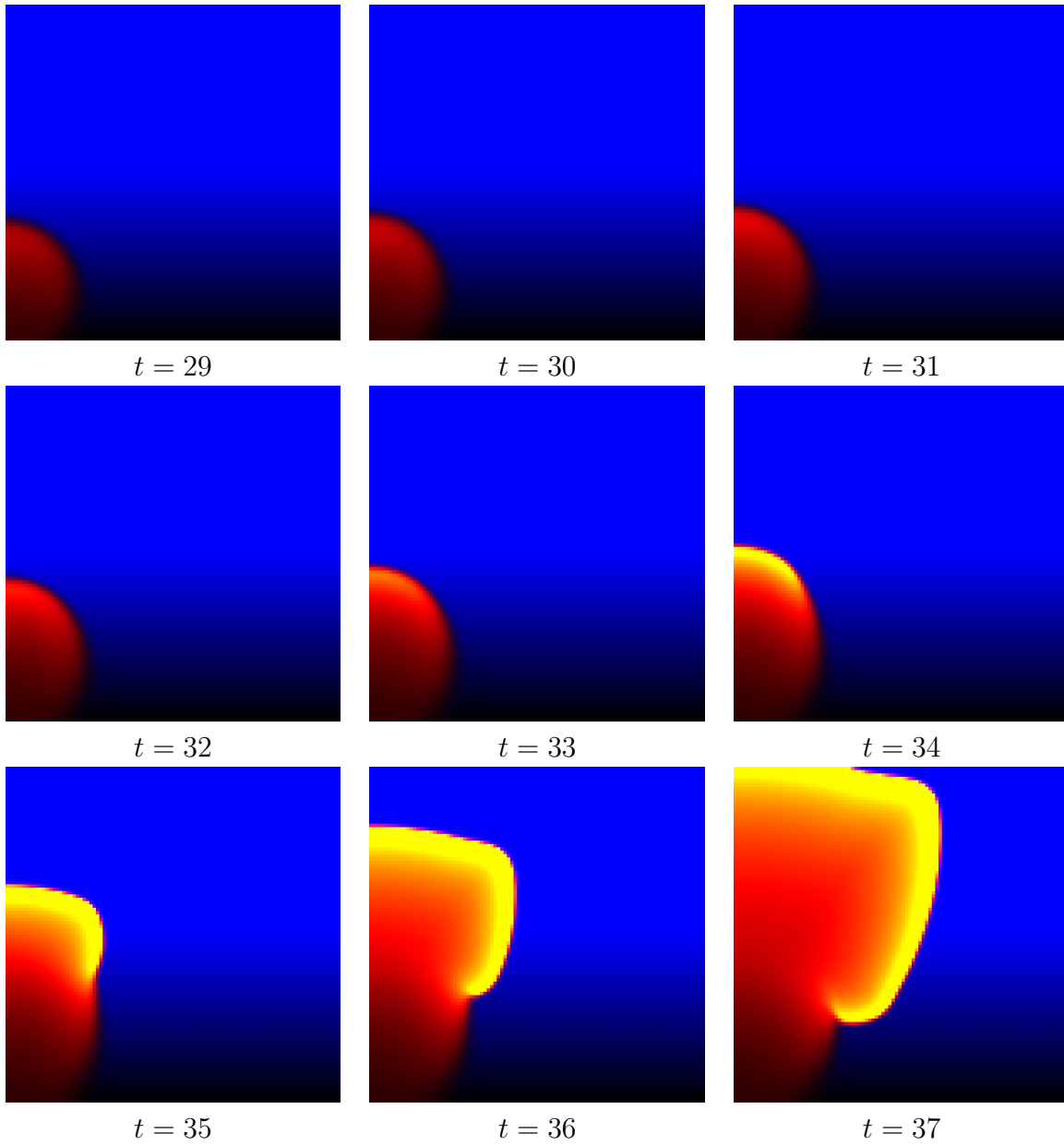


Figure 5.7: Qualitative behavior of fire spread test case. Concentration shown in black ($c = 0$) through blue ($c = 1$), temperature shown in black ($\vartheta = 0$) over red ($\vartheta \approx 6.4$) through yellow ($\vartheta > 53.6$).

Consequently the naive temporal refinement level $\tilde{L}_{i,j}$ for a cell at spatial indexes (i, j) is given by

$$\tilde{L}_{i,j} = \left\lceil \log_2 \left(\max \{ 1, |\vartheta_{i,j} - \vartheta_{i-1,j}|, |\vartheta_{i,j} - \vartheta_{i+1,j}|, |\vartheta_{i,j} - \vartheta_{i,j-1}|, |\vartheta_{i,j} - \vartheta_{i,j+1}| \} \right) \right\rceil.$$

These naive time levels are subsequently corrected to ensure that the time steps of adjacent cells differs by a factor of two maximum. Naively this can be implemented by initialization of the time levels $L_{i,j}$ according to

$$L_{i,j} := \tilde{L}_{i,j}$$

and repeated application of the following assignment

$$L_{i,j} := \max \{ L_{i,j}, \max \{ L_{i\pm 1,j}, L_{i,j\pm 1} \} - 1 \}.$$

This assignment ensures that none of the final time levels $L_{i,j}$ is smaller (i.e. coarser) than the corresponding naive time level $\tilde{L}_{i,j}$. The iteration is terminated when “nothing changes anymore” or more formally when the condition

$$\forall i, j : \max \{ |L_{i,j} - L_{i\pm 1,j}|, |L_{i,j} - L_{i,j\pm 1}| \} \leq 1,$$

is satisfied. This algorithm can easily be modified to take more complex adjacency relations into account. Note that if a cell has a faster neighbor, the reaction in this cell is solved on the next higher time level as motivated in Section 3.2.2.

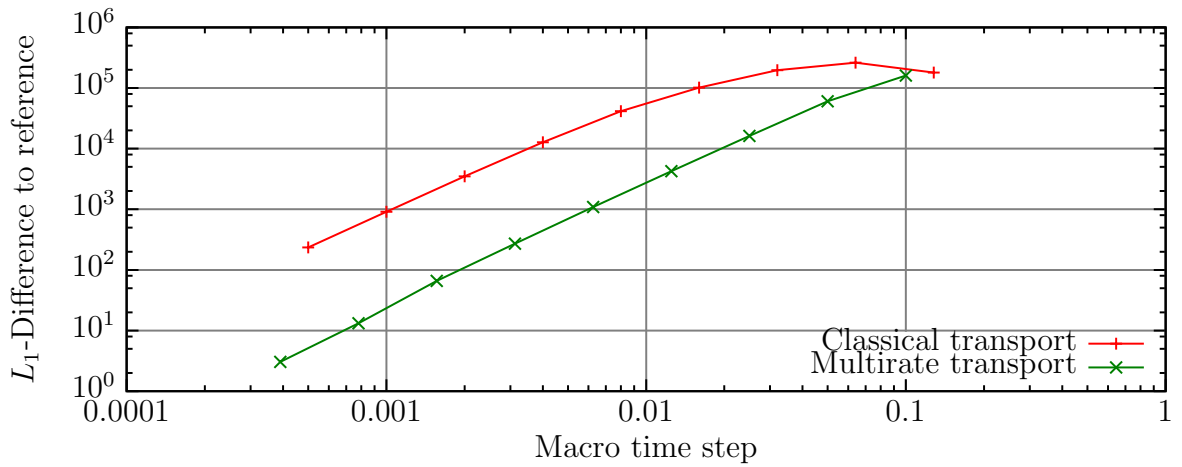
5.3.3 Results

We closely examine the results obtained for simulation time $t = 37$. At this point the major fraction of the initial concentration has reacted. To examine the error we compute a reference solution with the classical scheme, $\Delta t_{\text{macro}} = 10^{-5}$ and an absolute error tolerance for the reaction solver of 10^{-6} . The remaining computations were done with a tolerance of 10^{-5} .

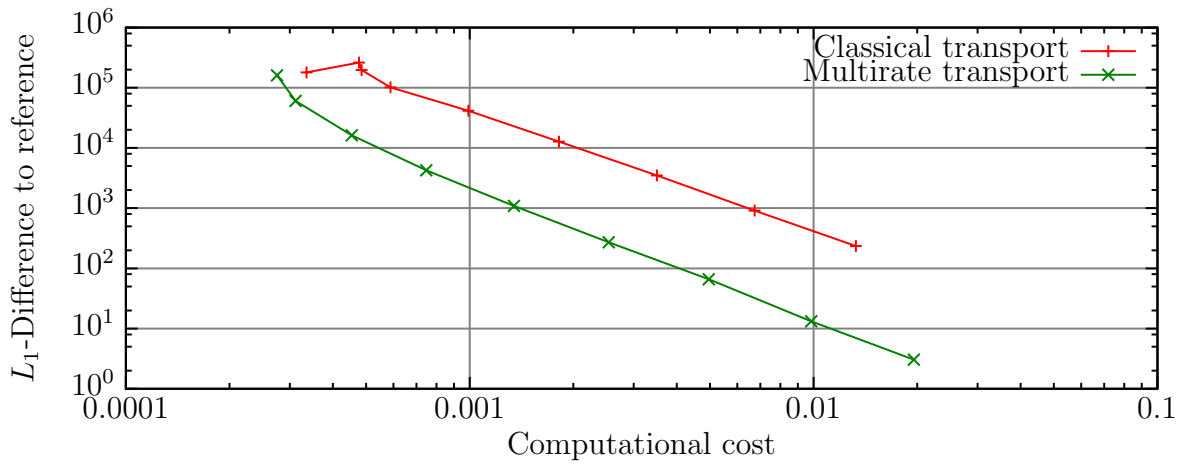
Both methods show the predicted second order of convergence, see Figure 5.8a. To examine the actual efficiency we also considered the computational cost, thus including effects of the increased algorithmic complexity of the multirate method and the overhead of choosing a cell specific time level for each time step, see Figure 5.8b. The computational cost is given in arbitrary units, as only the relative cost gives any information about the performance across different platforms.

To examine the evolution of the computational efficiency in course of the simulation more closely we compare two test runs which yield approximately equal errors at $t = 37$, that is $\Delta t_{\text{macro}} = 0.0035$ for the classical transport and $\Delta t_{\text{macro}} = 0.046$ for the multirate transport, see Figure 5.8c. Away from the “main event”, the computational cost per time step is inversely proportional to the macro time step for both methods. Only during the most intensive burning the computational cost of the multirate method exceeds the cost of the classical method. Around $t = 37$ nine temporal refinement levels were employed, which means that the smallest cell specific transport time step was

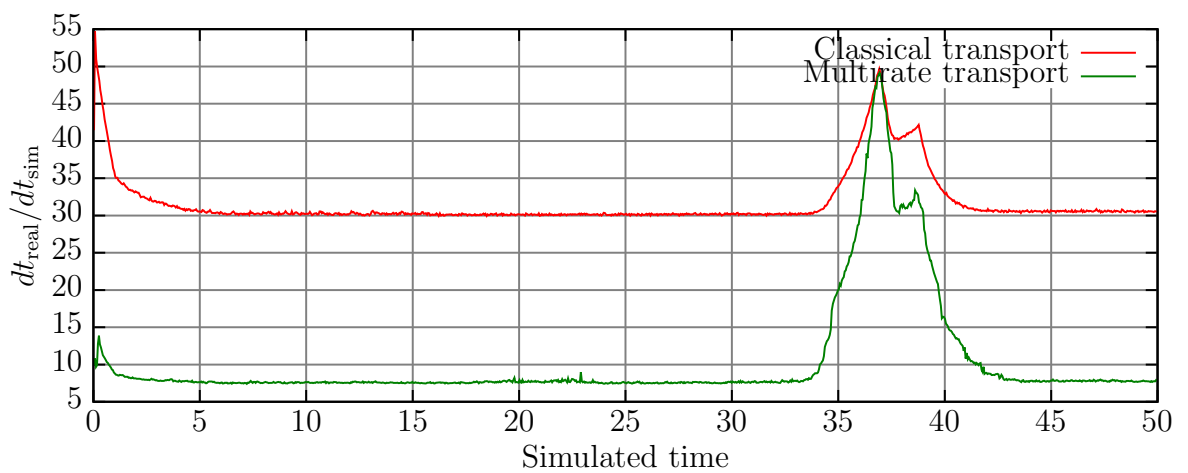
$$\Delta t_{\text{min}} = 2^{-9} \Delta t_{\text{macro}} \approx 6.84 \cdot 10^{-6}.$$



a) Global error at $t = 37$ as a function of the macro time step.



b) Global error at $t = 37$ as a function of the computational cost.



c) Computational cost in the course of one simulation as a function of the simulated time.

Figure 5.8: Fire spread test case results.

5.3.4 Conclusions

This test case is especially suitable to be solved with multirate methods as very few cells (those at the “flame front”) require a much smaller time step than the majority of cells. Even though the implementation of the multirate method was more complex it showed significantly higher efficiency in terms of computational cost for reaching a given precision.

5.4 MUSCAT test cases

The implementation described in Chapter 4 was tested with academic and realistic scenarios. Results show good agreement of the solutions obtained with the multirate splitting and classical time integration. We shall present two test cases. The first test is designed to make optimal use of the multirate approach by employing a grid with a small region of interest and a homogeneous, diagonal wind field. The other test case is taken from an earlier study, with a realistic wind field provided by the COSMO model [20]. Results of the latter case shall demonstrate the potential of multirate schemes for realistic scenarios as well as show remaining deficiencies. Both tests were run on an IBM p5-575 server with 16 POWER5 1.5GHz processors and 128GB of memory.

5.4.1 Academic test case

An important characteristic of parallel programs is the speedup¹ when distributing the problem on multiple processors. For the scenario discussed here we observed not quite an ideal (i.e. linear) speedup, but the overhead is small enough to justify parallel execution. Furthermore due to a sophisticated balancing approach making use of ParMetis’ multi constraint partitioning capabilities, the parallelization speedup is comparable to the one obtained for the much simpler case of single rate time integration.

The domain is quadratic in horizontal direction. A comparatively small region is refined, see Table 5.5 and Figure 5.9. Sources are located within the region most finely resolved.

horizontal cell size	relative area	number of columns	fraction of cells total
4km	≈69%	3584	≈18%
2km	≈20%	4096	≈21%
1km	≈10%	8192	≈41%
500m	≈1%	4096	≈21%

Table 5.5: Synopsis of spatial structure for academic test case.

¹Not to be confused with the reduction of computational cost due to application of a multirate scheme.

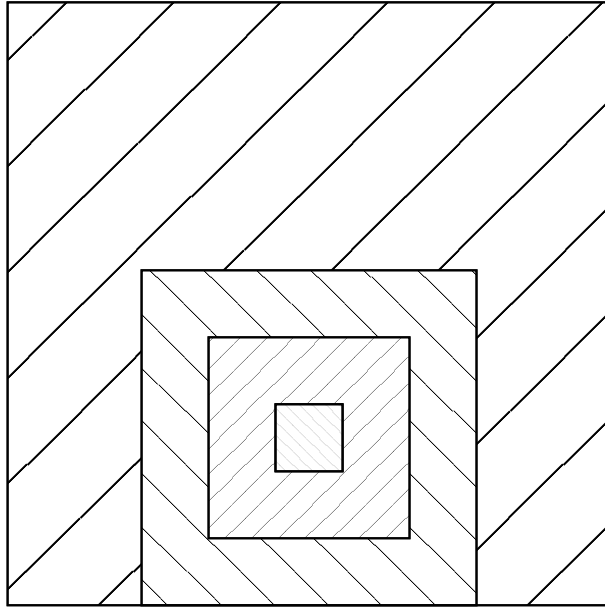


Figure 5.9: Illustration of the spatial structure for academic test run. Hatchings correspond to different grid sizes.

In this domain we tested two kinds of model equations: a pure advection with a uniform wind field and advection-diffusion-reaction with the same wind field, vertical diffusion and a chemistry model involving point sources in the finest region and 258 different chemical reactions of 98 reactants. The overall computational cost of the full system is about 100 times larger than that of the pure advection case. These tests are run on different numbers of processors each. We compare the computational cost of the multirate approach to the computational cost without temporal refinement, denoted *singlerate*. Results are shown in Figure 5.10.

As the time step is chosen to be directly proportional to the grid size, the naive reduction of computational cost is approximately 52%. For a pure advection problem we achieve an average cost reduction of 59%. Here the exceeding of the naive speedup can be explained by reduced communication. The behavior for the full advection-diffusion-reaction system however is not intuitive. In the latter case the multirate approach is applied only to the advection operator whose evaluation contributes about 1% to the total computational cost. However there still is a significant cost reduction of about 36%. The reason for this is the behavior of the term solved implicitly depending on the source term. Each update of this source term introduces a discontinuity in the right hand side of the equation. In combination with the error control of the second order implicit solver this causes smaller implicit steps or even makes expensive restarts necessary [29]. Fewer updates take place if the outer system is solved using a larger time step, thus indirectly improving the efficiency of the implicit solving.

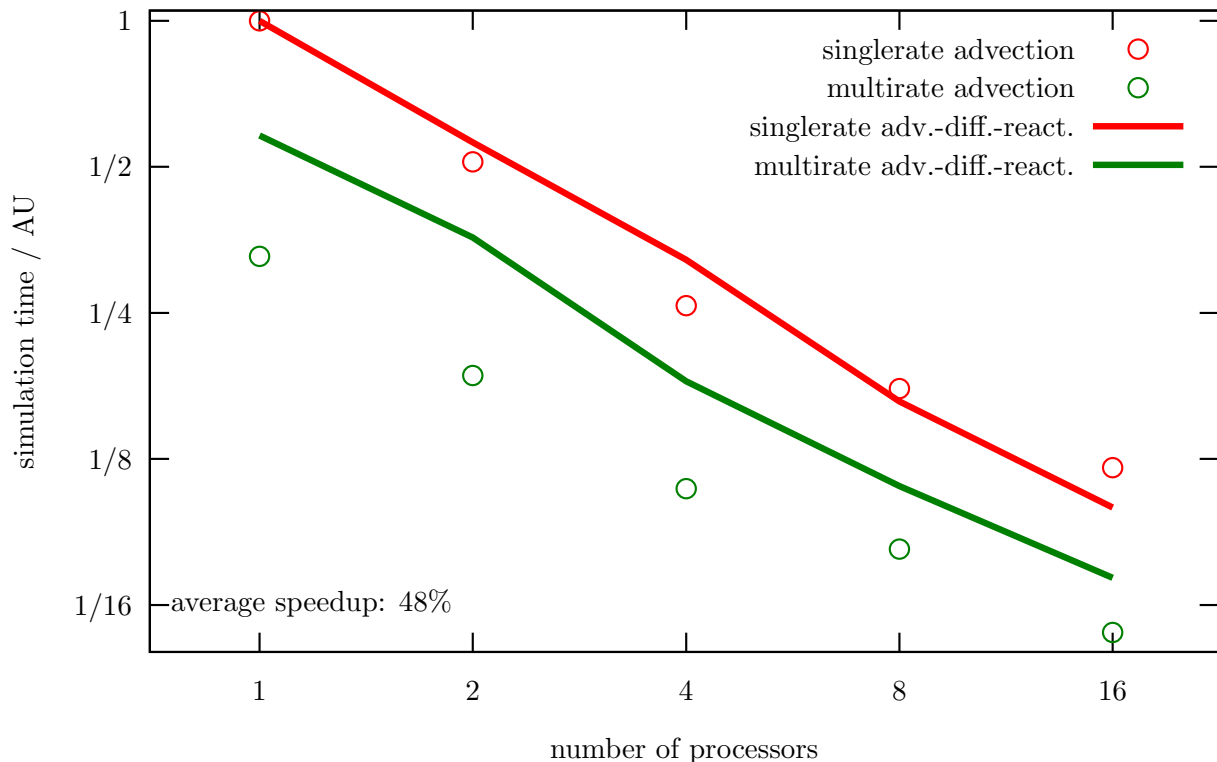


Figure 5.10: Computational cost for academic test case. The displayed simulation time was normalized such that the singlerate setup on one processor corresponds to one. Actual computational cost of the pure advection setup is about 1% of the full setup’s cost.

5.4.2 Realistic test case

The following test case is taken from a earlier study performed by Hinneburg et al- [20], examining the effects of emissions of two power plants in Germany. One plant is located near Lippendorf the other near Boxberg both in the federal country Saxony. Emissions are modeled by point sources which for the larger part represent the chimneys of the power plants, and area sources representing the emissions according to land usage. Meteorological data is provided by the COSMO model via online coupling.

horizontal cell size	relative area	number of columns	fraction of cells total
2.8km	≈41.0%	1748	≈7.3%
1.4km	≈41.0%	6992	≈29.2%
700m	≈16.6%	11312	≈47.2%
350m	≈1.4%	3904	≈16.3%

Table 5.6: Synopsis of spatial structure for realistic test case.

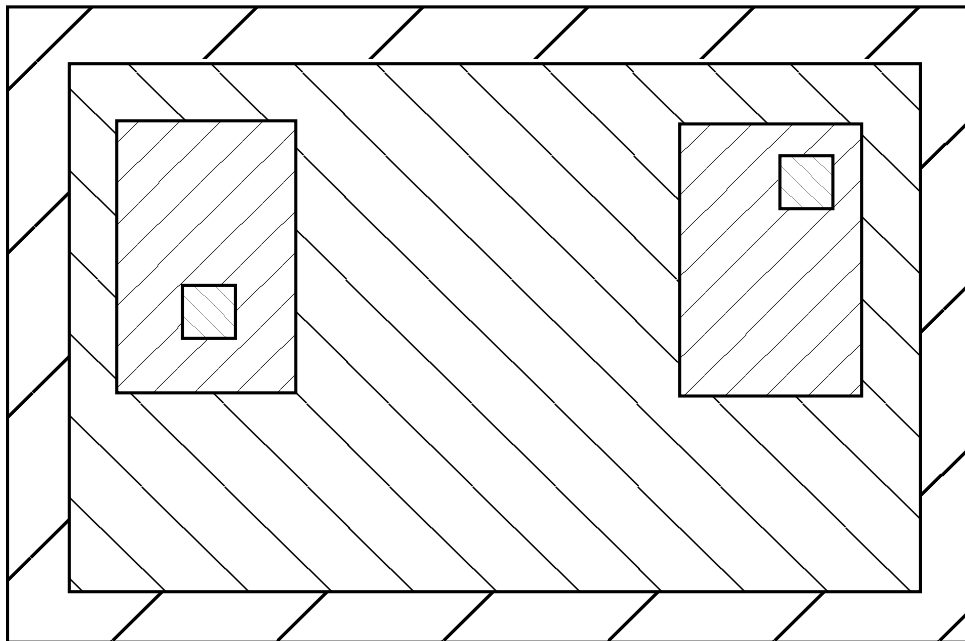


Figure 5.11: Illustration of the spatial structure for realistic test run. Hatchings correspond to different grid sizes.

Again we employ a grid with four levels of refinement with small, highly resolved regions around the power plants, totalling about 1% of the overall area, see Table 5.6 and Figure 5.11. The high resolution in this context is chosen to ensure an accurate description of the near field chemistry around the power plants by reducing numerical diffusion. Exactly equal parts of the total area are on the coarsest and second coarsest refinement level. Chemical reactions are modeled as in the more complex of the academic test cases with 258 different chemical reactions of 98 reactants.

Assuming a homogeneous wind field we would expect a slightly higher reduction of computational cost due to multirate time integration as for the academic test case with equal diffusion-reaction setup, as a smaller fraction of cells is on the finest refinement level. The actually obtained cost reduction is lower, see also Figure 5.12. This results from the fact that the wind fields provided by the COSMO model exhibit strong variability in all of the computational domain. A more sophisticated time step selection based on the characteristic times of the individual blocks rather than based solely on the spatial resolution can be constructed relatively easily. The time step selection employed in Section 5.3 could serve as a blue print for the this. Complications arise for parallel execution – if the temporal refinement level of a block is changed, a redistribution of the blocks is necessary. This holds for either of the balancing approaches discussed in Section 4.5.

A further defect concerns the parallelization speedup: while the problem scales well for up to eight processors, employing sixteen processors yields only very little improvement. At least in part this results from inhomogeneities due to the distribution of the point sources. The strongest point sources are inhomogeneously distributed on the blocks

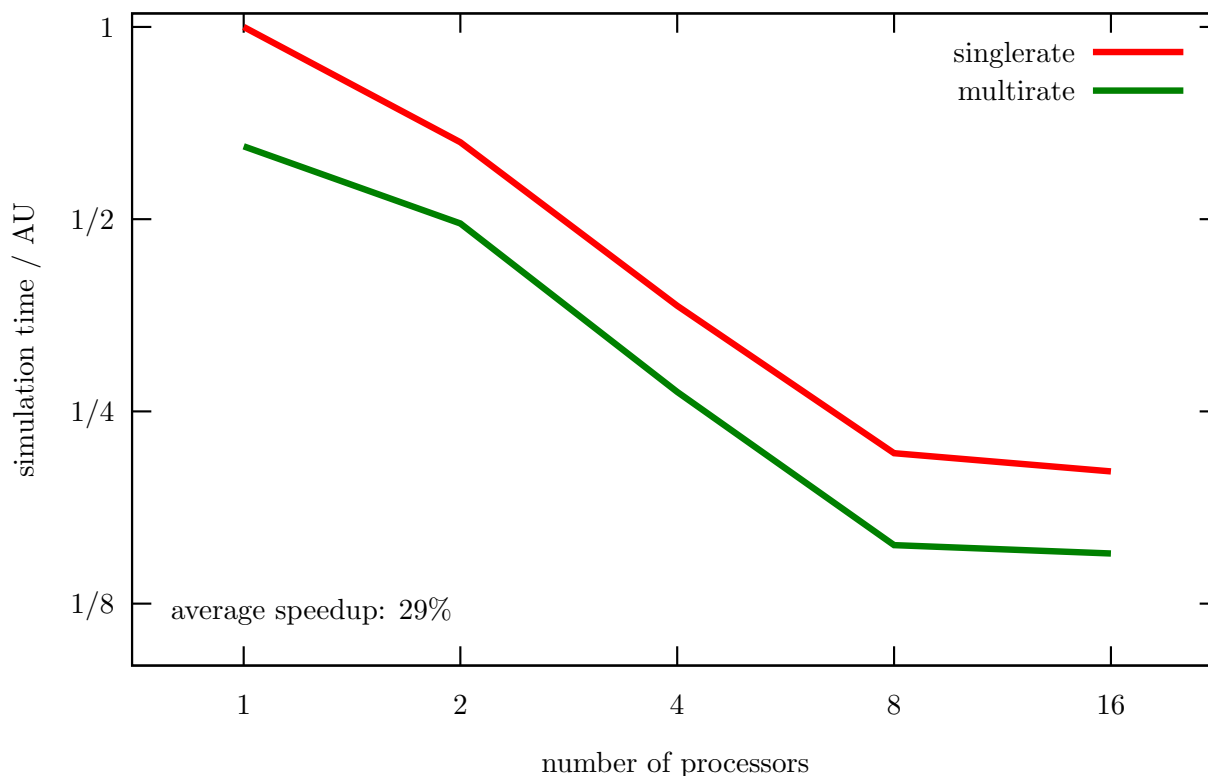


Figure 5.12: Computational cost for realistic test case. The displayed simulation time was normalized such that the singlerate setup on one processor corresponds one.

with the highest temporal and spatial refinement level. Each of the point sources induces significantly increased cost for the chemistry solver. For up to eight processors the blocks containing the point sources are distributed uniformly on all processors; for more processors this is not the case. This problem is even more complicated due to the temporally varying wind field. Due to higher concentrations the speed of chemical reactions inside of the plume is significantly higher than in free air. If the plume is transported into a previously empty cell, computational cost of this cell is increased for the next time step. This effect cannot easily be taken into consideration a priori. However employing dynamic repartitioning based on the measured workload per block seems to be a promising way to tackle this problem.

The correction of both of the mentioned defects involves the implementation of a complex repartitioning routine. The aim of workload balancing as discussed in Section 4.5 is a uniform distribution of time level specific workload on the available processors. Choosing the time level of a block based on the ratio of wind speed and cell size instead of cell size only makes dynamic time levels necessary for any simulation setups where the wind field varies with time. Changing the time level of a block leads to load imbalances and thus makes repartitioning necessary. However computing a new partitioning “from scratch” is not a good solution as the transfer of complete block data from one processor to another is expensive. Instead a sophisticated repartitioning routine must also take into

account, data on which blocks are present on which processors and ensure that as few blocks as possible are moved from one processing element to another. This also has to include a heuristic to determine if repartitioning is worthwhile.

Chapter 6

Conclusions and outlook

In this thesis we presented a general splitting approach called *Recursive Flux Splitting Multirate* (RFSMR) that can be employed to construct purely explicit multirate methods, implicit-explicit (IMEX) methods and multirate-IMEX methods combining the advantages of both splitting approaches. The approach is generic in the sense that a partitioned method is constructed from the parameters of an explicit Runge–Kutta method and further time integration methods which are not strictly limited to Runge–Kutta methods. Opposed to other generic multirate schemes the relative time step sizes for the different partitions cannot be controlled by a simple parameter. Instead the relative time step sizes are dictated by the parameters of the underlying Runge–Kutta methods. In a separate section we discussed how to obtain a desired ratio of time step sizes.

If all of the employed base methods are Runge–Kutta methods the generated methods can be written as partitioned Runge–Kutta (PRK) methods, which allows for formal examination in a well established framework. It shows that the generated methods are *internally consistent*, which means that the Runge–Kutta nodes of the different parts of the method are equal. This can be interpreted as intermediate results of the different parts being evaluated “at the same points in time”. Unfortunately the constructed methods do not preserve linear invariants of the system. In the context of split transport equations this corresponds to a mass defect in the size of the discretization error. To obtain a mass preserving full discretization for transport equations we split the advection operator by fluxes instead of the more commonly applied splitting by components. While the latter kind of splitting does not lead to a consistent full discretization in theory, numerical tests show that both splittings are on par for relevant simulation setups. For the solution of advection-reaction systems we showed that recursive application of a bipartite splitting can equivalently be expressed as a tripartite splitting which makes a more efficient implementation possible.

Formal analysis showed that the RFSMR approach leads to second order partitioned methods if second order base methods are employed. Third order partitioned methods can be constructed if the employed base methods satisfy the classical third order conditions as well as one additional order condition presented for a class of IMEX methods in [27]. Recently the third order method was applied successfully to the simulation of geophysical

flows using a discontinuous Galerkin approach, see Seny et al. [42]. The representation of the generated methods as PRKs makes it possible to modify the directly constructed methods so that they preserve the linear invariants. These modified methods are of the same order of accuracy as the directly constructed (unmodified) methods, but closer examination shows that they are less stable or computationally more expensive. The fact that those methods are internally consistent and mass preserving at the same time might however make them interesting candidates for certain applications.

Stability analysis was carried out for a class of advection-reaction problems and some IMEX methods constructed via the RFSMR approach and one instance of a modified, mass preserving IMEX method. A notable result from this analysis is the construction of a third order explicit method with improved stability properties which is suitable for the construction of a third order IMEX method.

A major part of this thesis is concerned with implementation issues for a three dimensional atmospheric chemistry and transport model (MUSCAT). While the time integration routine resulting from the RFSMR approach can easily be implemented, further details of this implementation pose challenges. The decomposition of the advection operator in two spatial dimensions, data exchanges and parallelization received special attention. The more complex program flow for a multirate-IMEX time integration scheme compared to the IMEX time integration scheme previously implemented in MUSCAT makes classical workload balancing unsuitable. Instead a so called *multi constraint* balancing approach has been discussed and implemented.

Numerical tests show that multirate schemes as well as multirate-IMEX schemes are more efficient than classical, monolithic time integration schemes. In particular the computational cost can be reduced while satisfying the same demands on stability or accuracy. For the multirate schemes based on the RFSMR approach we additionally find reduction of communication cost compared to classical time integration schemes.

Multirate time integration schemes are very promising methods for a wide range of applications. The combination with an IMEX splitting further increases their potential. Some simulation setups employing rigorous local grid refinement become feasible only when multirate methods are employed.

In this thesis we restricted ourselves to a multirate approach based on one step methods. A combination of RFSMR techniques with multistep methods may be advantageous for certain applications and further work will have to show how the general splitting approach presented in this thesis can be extended to the multistep case. Further development of MUSCAT will include a time step and time level selection that is not solely based on the blocks' spatial resolution but makes use of more detailed information. Additionally a repartitioning approach that takes the temporal refinement levels of the involved blocks into account should be implemented.

In atmospheric models, so called *Volume of fluid* (VOF) methods [32] may serve to reconstruct the boundary of clouds and thus allow both for a more accurate description of transport processes and for a better separation of dry phase chemistry in free air and wet phase chemistry inside of clouds. The combination of VOF methods with multirate time integration based on flux split transport seems to be promising.

Bibliography

- [1] A. Arakawa and V. R. Lamb. Computational design of the basic dynamical process of the ucla general circulation model. *Methods in Computational Physics*, 17:173–265, 1977.
- [2] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri. Implicit–explicit Runge–Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2–3):151–167, 1997.
- [3] M. Berger, M.J. Aftosmis, and S.M. Murman. Analysis of slope limiters on irregular grids. *AIAA paper 2005-0490*, in: *43rd AIAA Aerospace Sciences Meeting*, 2005.
- [4] J. Butcher. Coefficients for the study of Runge–Kutta integration processes. *J. Austral. Math. Soc.*, pages 185–201, 1963.
- [5] J. Butcher. Runge–Kutta methods. *Scholarpedia*, 2(8):3147, 2007.
- [6] E.M. Constantinescu and A. Sandu. *On Adaptive Mesh Refinement for Atmospheric Pollution Models*, volume 3515/2005 of *Lecture Notes in Computer Science*, pages 798–805. Springer Berlin / Heidelberg, 2005.
- [7] E.M. Constantinescu and A. Sandu. Multirate timestepping methods for hyperbolic conservation laws. *J. Sci. Comput.*, 33(3):239–278, 2007.
- [8] Emil Constantinescu, Adrian Sandu, and Gregory Carmichael. Modeling atmospheric chemistry and transport with dynamic adaptive resolution. *Computational Geosciences*, 12:133–151, 2008. 10.1007/s10596-007-9065-7.
- [9] W.F. Dabberdt, M.A. Carroll, D. Baumgardner, G. Charmichael, R. Cohen, T. Dye, J. Ellis, G. Grell, S. Grimmond, S. Hanna, J. Irwin, B. Lamb, S. Madronich, J. McQueen, J. Meagher, T. Odman, J. Pleim, H.P. Schmid, and D.L. Westphal. Meteorological research needs for improved air quality forecasting. *Bulletin- American Meteorological Society*, 85:563–586, 2004.
- [10] I.Th. Famelis, S.N. Papakostas, and Ch. Tsitouras. Symbolic derivation of Runge–Kutta order conditions. *J. Symb. Comput.*, 37(3):311–327, 2004.
- [11] Message Passing Interface Forum. *Mpi: A message-passing interface standard*, 1994.

- [12] Message Passing Interface Forum. *Mpi-2: Extensions to the message-passing interface*, 1997.
- [13] C.W. Gear and D.R. Wells. Multirate linear multistep methods. *BIT Num. Math.*, 24(4):484–502, 1984.
- [14] A. Gerisch and R. Weiner. On the positivity of low order explicit Runge–Kutta schemes applied in splitting methods. *Reports on Numerical Mathematics*, 1999.
- [15] M. Günther, A. Kvaerno, and P. Rentrop. Multirate partitioned runge-kutta methods. *BIT Numerical Methods*, pages 504–514, 2001.
- [16] V. Grützun, O. Knoth, and M. Simmel. Simulation of the influence of aerosol particle characteristics on clouds and precipitation with Im-specs: Model description and first results. *Atmos. Res.*, 90:233–242, 2008.
- [17] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations*, volume I. Springer Verlag, 1987.
- [18] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations*, volume II: Stiff and Differential-Algebraic Problems. Springer Verlag, 1991.
- [19] B. Heinold, I. Tegen, K. Schepanski, and O. Hellmuth. Dust radiative feedback on saharan boundary layer dynamics and dust mobilization. *Journal of Geophysical Research*, 112: D11204, 2008.
- [20] D. Hinneburg, E. Renner, and R. Wolke. Formation of secondary inorganic aerosols by power plant emissions exhausted through cooling towers in saxony. *Environ Sci Pollut Res*, 16:25–35, 2009.
- [21] W. Hundsdorfer, B. Koren, M. van Loon, and J. Verwer. A positive finite-difference advection scheme. *J. Comput. Phys.*, 117:35–46, 1995.
- [22] W. Hundsdorfer, A. Mozartova, and V. Savcenko. Analysis of explicit multirate and partitioned Runge–Kutta schemes for conservation laws. CWI Report MAS-E0715, Centrum voor Wiskunde en Informatica, Amsterdam, 2007.
- [23] W. Hundsdorfer and J. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion Reaction Equations*. Springer Series in Computational Mathematics. Springer, Berlin, Heidelberg, New York, 2003.
- [24] Z. Jackiewicz and R. Vermiglio. Order conditions for partitioned Runge–Kutta methods. *Applications of Mathematics*, 45(4):301–316, 1998.
- [25] G. Karypis. Multilevel algorithms for multi-constraint hypergraph partitioning. Technical Report 99–034, University of Minnesota, Department of Computer Science/Army HPC Research Center, 1999.

- [26] G. Karypis, K. Schloegel, and V. Kumar. *ParMetis - Parallel Graph Partitioning and Sparse Matrix Ordering Library, Version 3.1*, 2003.
- [27] O. Knoth and R. Wolke. A comparison of fast chemical kinetic solvers in a simple vertical diffusion model. *Air Pollution Modeling and its Application*, X:287–294, 1994.
- [28] O. Knoth and R. Wolke. An explicit-implicit numerical approach for atmospheric chemistry-transport modeling. *Atmospheric Environment*, 32(10):1785–1797, 1998.
- [29] O. Knoth and R. Wolke. Implicit-explicit Runge-Kutta methods for computing atmospheric reactive flows. *Appl. Numer. Math.*, 28(2-4):327–341, 1998.
- [30] M. Lieber, V. Grützun, R. Wolke, M. S. Müller, and W. E. Nagel. Highly scalable dynamic load balancing in the atmospheric modeling system cosmo-specs+fd4. In *PARA 2010, "Applied Parallel and Scientific Computing"*, volume 7133, pages 131–141, 2012.
- [31] M. Lieber and R. Wolke. Optimizing the coupling in parallel air quality model systems. *Environ. Modell. Softw.*, 23:235–243, 2008.
- [32] Len Margolin, Jon M. Reisner, and Piotr K. Smolarkiewitz. Application of the volume-of-fluid method to the advection–condensation problem. *Mon. Wea. Rev.*, 125:2265–2273, 1997.
- [33] Tomáš Oberhuber, Atsushi Suzuki, Jan Vacata, and Vítězslav Žabka. Image segmentation using CUDA implementations of the Runge-Kutta-Merson and GMRES methods. *Journal of Math-for-Industry*, 3 (2011C-10):73—79, 2011.
- [34] Tomáš Oberhuber, Atsushi Suzuki, and Vítězslav Žabka. The CUDA implementation of the method of lines for the curvature dependent flows. *Kybernetika*, 47(2):251–272, 2011.
- [35] S. Osher and R. Sanders. Numerical approximations to nonlinear conservation laws with locally varying time and space grids. *Math. Comput.*, 41:321–336, 1983.
- [36] H. H. Robertson. *Numerical Analysis: An Introduction*, chapter "The solution of a set of reaction rate equations", pages 178–182. Academic Press, 1966.
- [37] A. Sandu and E.M. Constantinescu. Multirate explicit adams methods for time integration of conservation laws. Technical Report TR-07-30 (989), Virginia Polytechnic Institute and State University, 2007.
- [38] V. Savcenko, W. Hundsdorfer, and J.G. Verwer. A multirate time stepping strategy for stiff ordinary differential equations. *BIT Numerical Mathematics*, (47):137–155, 2007.

- [39] Martin Schlegel, Oswald Knoth, Martin Arnold, and Ralf Wolke. Multirate Runge–Kutta schemes for advection equations. *JCAM*, 226:345–357, 2009.
- [40] Martin Schlegel, Oswald Knoth, Ralf Wolke, and Martin Arnold. Numerical Solution of Multiscale Problems in Atmospheric Modeling. 2012. accepted for Proc. of NUMDIFF-12, 2009.
- [41] U. Schättler, G. Doms, and C. Schraff. A description of the nonhydrostatic regional cosmo-model. part i: Dynamics and numerics. Technical report, Deutscher Wetterdienst, Offenbach, 2008. available from <http://www.cosmo-model.org>.
- [42] B. Seny, J. Lambrechts, R. Comblen, V. Legat, and J.-F. Remacle. Multirate time stepping for accelerating explicit discontinuous galerkin computations with application to geophysical flows. *Int. J. Numer. Meth. Fluids*, 2012. DOI: 10.1002/flid.3646.
- [43] G. I. Shishkin. Grid approximation of singularly perturbed boundary value problems with convective terms. *Sov. J. Numer. Anal. Math. Modelling*, 5:173–187, 1990.
- [44] J. Steppeler, G. Doms, U. Schättler, H.W. Bitzer, A. Gassmann, U. Damrath, and G. Gregoric. Meso-gamma scale forecasts using the nonhydrostatic model lm. *Meteorology and Atmospheric Physics*, 107:75–96, 2003.
- [45] H. Tang and G. Warnecke. A class of high resolution schemes for hyperbolic conservation laws and convection–diffusion equations with varying time and space grids. *SIAM J. Sci. Comput.*, 26(4):1415–1431, 2005.
- [46] J. G. Verwer. Explicit Runge–Kutta methods for parabolic partial differential equations. *Appl. Numer. Math.*, 22:359–379, 1996.
- [47] J. G. Verwer, B. P. Sommeijer, and W. Hundsdorfer. Rkc time-stepping for advection–diffusion–reaction problems. *J. Comput. Phys.*, 201:61–79, 2004.
- [48] R. Wolke, O. Knoth, O. Hellmuth, W. Schröder, and E. Renner. The parallel model system lm-muscat for chemistry-transport simulations: Coupling scheme, parallelization and applications. *Parallel Computing*, page 363–370, 2004.
- [49] R. Wolke, O. Knoth, and J. Weickert. Load balancing in the parallelization of the multiscale atmospheric chemistry-transport model muscat. In *16th IMACS World Congress*, 2000.
- [50] R. Zvan, P. Forsyth, and K. Vetzal. Robust numerical methods for PDE models of Asian options. *J. Comp. Fin.*, 1:39–78, 1998.

Statement of authorship

I hereby certify that this dissertation has been composed by myself, and describes my own work, unless otherwise acknowledged in the text. All references and verbatim extracts have been quoted, and all sources of information have been specifically acknowledged. It has not been accepted in any previous application for a degree.

Martin Schlegel
Leipzig, 31.5.2012

Eigenständigkeitserklärung

Hiermit erkläre ich dass ich diese Arbeit selbstständig verfaßt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich in jedem einzelnen Fall durch die Angabe der Quelle als Entlehnung kenntlich gemacht. Die vorliegende Arbeit wurde in keinem früheren Promotionsverfahren vorgelegt.

Martin Schlegel
Leipzig, 31.5.2012