INAUGURAL-DISSERTATION

zur Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität Heidelberg

vorgelegt von
Diplom-Meteorologin Teresa Beck
aus Stuttgart - Bad Cannstatt

Tag der mündlichen Prüfung:

In-Time Parallelization Of Atmospheric Chemical Kinetics

Betreuer :	Prof. Dr. Vincent Heuveline
Korreferent :	Prof. Dr. Christoph Kottmeier

Abstract

This work investigates the potential of an in-time parallelization of atmospheric chemical kinetics. Its numerical calculation is one time-consuming step within the numerical prediction of the air quality. The widely used parallelization strategies only allow a limited potential level of parallelism. A higher level of parallelism within the codes will be necessary to enable benefits from future exa-scale computing architectures. In air quality prediction codes, chemical kinetics is typically considered to react in isolated boxes over short splitting intervals. This allows their trivial parallelization in space, which however is limited by the number of grid entities. This work pursues a parallelization beyond this trivial potential and investigates a parallelization across time using the so called "parareal algorithm". The latter is an iterative prediction-correction scheme, whose efficiency strongly depends on the choice of the predictor. For that purpose, different options are being investigate and compared: Time-stepping schemes with fixed step size, adaptive time-stepping schemes and repro-models, functional representations, that map a given state to a later state in time. Only the choice of repro-models leads to a speed-up through parallelism, compared to the sequential reference for the scenarios considered here.

Zusammenfassung

Diese Arbeit untersucht das Potential einer Parallelisierung in der Zeit der atmosphärischen Reaktions-Kinetik. Deren Berechnung stellt einen der Rechenzeit-aufwändigsten Schritte bei der numerischen Vorhersage der Luftqualität dar. Die hierzu üblicherweise verwendeten Parallelisierungs-Ansätze ermöglichen nur ein beschränktes Potential an Parallelilität. Um von zukünftigen exa-scale Rechnerarchitekturen profitieren zu können, sind weitere Ebenen an Parallelität innerhalb der verwendeten Computer-Codes nötig. Üblicherweise wird die chemische Reaktionskinetik innerhalb solcher Luftqualitäts-Computerprogramme über kurze Zeitintervalle auf isolierten Boxen betrachtet. Dies erlaubt deren triviale Parallelisierung im Raum, die allerdings beschränkt ist durch die Anzahl der Gitter-Einheiten. Diese Arbeit strebt eine darüber hinausgehende Parallelisierung in der Zeit an unter Verwendung des sogenannten "pararealen Algorithmus". Letzterer beschreibt ein iteratives Prognose-Korrektur-Verfahren, dessen Effizienz stark von der Wahl des Prognose-Verfahrens abhängt. Zu diesem Zweck werden verschiedene Optionen untersucht und miteinander verglichen: Zeitschrittverfahren mit fester Schrittweite, adaptive Zeitschrittverfahren und Repro-Modelle, funktionale Darstellungen, die einem gegebenen Zustand einen späteren Zustand zuordnen. Für die hier betrachteten Szenarien führt nur die Wahl von Repro-Modellen zu parallelen Simulationen, die schneller sind als die sequentielle Referenz.

Mathematical Contribution

In this work, the potential of an in-time parallelization for the numerical approximation of stiff ordinary differential equations is being investigated. To this, the most prominent in-time parallel algorithm is being focused, the parareal algorithm, which is an iterative predictor-corrector scheme. Different options regarding the choice of the predictor and the decomposition into sub-intervals are investigated and compared. Only the usage of so called repro-models, functional representations of the input-output behavior of such chemical systems, allow for speed-ups in comparison to the sequential scheme.

To the best of the authors knowledge, no detailed investigation and comparison of parareal approaches for the approximation of stiff problems have been presented in literature yet. A combined application of the parareal algorithm and repro-modeling has not been presented in literature up to now and represents a novelty.

Acknowledgments

First and foremost I want to thank my advisor. Dear Prof. Heuveline, it has been a great honor for me to be your Ph.D. student at the Faculty of Mathematics and Computer Science at Heidelberg University. I want to express my deepest thankfulness for your support, your motivation and your patience. I have always appreciated your keen mind and your vision.

Great thanks are further dedicated to the IMK for funding and supporting me during the first 3 years of this thesis. In particular I want to thank Bernhard Vogel and his working group and my co-advisor Prof. Christoph Kottmeier for committed support and sound advice. Thanks are also devoted to the Faculty of Mathematics and Computer Science and the Interdisciplinary Center for Scientific Computing at Heidelberg University for their infrastructural and professional support. This work was partly funded by the Helmholtz project REKLIM and partly by the EU project Exa2Green. I want to thank all the partners within the projects for the collaboration. I enjoyed working in these interdisciplinary projects very much.

Tremendous credits go out to my colleagues. My dear fellows, your company made all the exertions of the last years tolerable. You made it possible to grow from and not to be ruined by the challenge Ph.D.. Countless times, I've been so thankful for your open ears, your good ideas, your great sense of humor and in particular for your continuous supply with cookies. Big thanks are further devoted to Mrs. Mehra for pulling the strings behind the scenes and not getting tired of my laments.

Finally, let me express my special thanks to those who supported and motivated me offstage: My family, my friends and in particular my partner. Super team.

Contents

Chapter 1

Introduction

Atmospheric air quality models[1] (AQMs) are complex computer programs, that simulate the physical and chemical processes of the reaction and dispersion of air pollutants in the atmosphere. Such models are important tools for the quality management of the air. In practice, they are typically employed to estimate the compliance of industrial facilities with ambient air quality standards. They also serve political decision-making, since they can be consulted for the identification of sources of air quality problems and for the determination of respective control requirements. Subsequent to new regulatory programs, AQMs can also be employed to forecast future pollutant concentrations.

The base of such AQMs is typically formed by chemical transport models (CTMs), that describe the temporal evolvement of chemical species due to advection, diffusion, chemical reaction and other processes. Mathematically, such models are described by sets of multi-scalar partial differential equations. Their numerical simulation requires discretizations in time and space. The resulting linear systems of equations comprise huge numbers of unknowns. Solving them is computationally demanding and requires the utilization of supercomputers. One of the key principles of high performance computing (HPC) is the distribution of computationally intensive tasks over multiple processors. This requires special attention in the choice and design of respective algorithms, since not all algorithms inhere parallelism by nature. For the solution of partial differential equations, one popular parallelization strategy is the so called domain decomposition: The original boundary value problem is decomposed into smaller chunks, that are being approximated in parallel, with an iterative update of interacting boundary values on neighboring chunks.

The domain decomposition method typically facilitates the first level of parallelization of a CTM. A second level of parallelization comes about as a natural byproduct of an operator splitting approach. Since CTMs inhere wide ranges of different time-scales, the individual processes like advection, diffusion, and reaction are being decoupled from another and nu-

[1]A little confusion exists around the term "model": In the field of Mathematics, a "model" is a description of a system by means of mathematical concepts and in a mathematical language. In applied sciences, the term "model" often constitutes a computer program, that numerically simulates chemical, physical or biological phenomena, captured in mathematical models. In this work, the term "model" will primarily be allotted with the Mathematical "model". Since the term "air quality model" is established and convenient, we add an exception here.

merically treated separately with individual time-stepping schemes. The consequence of the operator splitting on the numerical treatment of the chemical reaction is, that one, individual initial value problems is being solved per splitting interval and per grid entity. This allows for a trivial parallelization, with potential performance speed-ups up to the number of grid entities. If however more processors are available than grid entities, this potential can not be exploited any further.

Still, the simulation of the *chemical reaction kinetics*, the study of the rates of chemical processes, remains one computationally intensive and time-consuming subcomponent of an AQM. The chemical mechanisms, that are incorporated in AQMs describe processes on wide ranges of time-scales. Radical species, such as the hydroxyl radical, react very quickly, while others reveal atmospheric lifetimes of years (e.g. methane). Mathematically, atmospheric chemical kinetics are described by means of ordinary differential equations. For their numerical integration, adaptive techniques are inevitable, that quickly adapt the time-step size according to the reaction progress. Those schemes are typically iterative and inhere a very limited natural potential of parallelism. A further parallelization of these systems is far from obvious. Current research mainly focuses on a parallelization of the chemistry on an instruction line level [71]. Research has been done into an algorithmic reorganization by means of slight modifications of the solvers [20, 99, 129, 130]. The approaches presented in literature all show a limited parallelization potential.

Avowedly, the potential of a trivial parallelization is not yet fully realized at this point in time. In practice, many AQMs still sequentially compute individual boxes one after another on the same processor. A reason, may be a conservative code development policy, which in turns guarantees a high level of resilience. For operational models, this is one crucial characteristic. Climate, weather and air quality models are extensive code structures, which have been developed over years to decades. Porting hundreds of thousands of lines of code from Fortran 77 to the latest C++ standards is a challenging and time-consuming task. With increasing complexity of the available hardware, further the demands on the software design grew. A respective implementation requires trained man-power and a high degree of expertise. Typically, the developers are experts in their own field of subject and primarily interested in the results, their models produce. Interdisciplinary efforts are therefore necessary, to adapt the codes to optimally benefit from current and future hardware.

Decades of enhancement has left scientists with super-sophisticated, high-resolution models. With growing complexity and resolution, also the computational effort grew, which makes weather and climate prediction a time- and energy-consuming business. Especially high-resolution and long-term simulations are expensive both in the sense of wall clock time and energy consumption. Energy requirements of HPC systems are already and will become even more prohibitive. Following an overall hardware trend, the community recently investigates the porting of their codes to less energy-consumptive processors [24, 88]. Computationally intensive calculations are relocated on more energy-efficient processors, such as graphic processing units or advanced risk machines. Depending on the architecture, an efficient implementation will require higher levels of parallelism within the algorithms. In the prospect of exa-scale computers, the community is confronted with a crucial challenge: The currently used parallelization strategies inhere a limited level of parallelism and hence can only support a parallelization up to some limit.

The development of new parallelization strategies is not only important in the context of performance speed-ups and energy savings, but also in the context of the global balancing of work. This is in particular true for the calculation of the atmospheric chemistry, for which the computational effort strongly depends on the level of photolytic activity. Photolysis is dominated by the intensity of the sunlight, which shows a diurnal variation. Especially during transition times between night and day, the photolytic activity is high and the presence of fast processes makes the numerical approximation computationally intensive. On a global level, load-imbalances are the result: At one physical point in time, chemical boxes describing chemistry at sunrise, as well as at daytime, sunset or nighttime will have to be calculated simultaneously.

To pave the road towards exa-scale climate and weather prediction, scalable algorithms must be designed, investigated and finally also implemented. We therefore investigate the potential of an *in-time parallelization* of the numerical treatment of the atmospheric chemical kinetics. The basic principle of such methods is a simultaneous approximation of subsequent events.

Time-parallel algorithms for the solution of initial value problems look back at a long history [35]. In fact, they are as old as the first supercomputers. Most parallel-in-time methods catch up on the ideas of the domain decomposition method. Initial value problems are split into individual time-slabs, and solved in parallel. Through an outer iteration, initial values on each time-slab are updated and corrected by means of the final solutions on the precedent time-slabs. One of the essential advantages over other parallelization strategies for the solution of initial value problems is, that parallel-in-time algorithms allow for unlimited parallelization levels. Among a wide range of related algorithms, one prominent algorithm is the parareal algorithm.

The parareal algorithm is an iterative prediction-correctiom algorithm. Within the iteration, a coarse and cheap, sequential prediction is corrected by means of a fine, but computationally costly correction. While the coarse prediction has to be propagated sequentially, the costly computation of the correction can be parallelized. Independent of the choice of the coarse propagator, the algorithm converges to the solution of the fine propagator. For the latter, one chooses a respective numerical integrator with a desired step size or tolerance. For the coarse prediction, any scheme can be used: The same numerical scheme with a coarser time-step size, a different and less costly scheme or even a completely different model. The better the coarse prediction, the less iterations will be necessary until convergence. The faster the prediction, the higher the expected speed-up. To allow for an optimal speed-up, the choice of a coarse propagator has to be guided by finding a trade-off between high accuracy and low computational effort.

An application of the parareal algorithm to atmospheric chemical kinetics can be implemented in different ways: Internally, within one operator splitting interval or externally, across multiple operator splitting intervals. The latter case requires a consideration of external effects, such as advection, diffusion and changing photolysis rates at the interfaces between two intervals. In both cases, the application of the parareal algorithm is hindered by the multi-scale nature of the atmospheric chemical kinetics. The presence of wide ranges of time-scales in such systems forces the usage of adaptive time-stepping schemes as a fine propagator. Depending on the application, a decomposition of the simulation interval into equidistant time-slabs will then lead to load-imbalances, which can be diminished by using an adaptive decomposition. The multi-scale nature also affects the choice of a coarse propagator. Adaptive coarse integrators using multiple steps are too costly, while single-step integrators are not accurate enough. In that course, reduced models seem to be a promising alternative to balance accuracy and computational effort.

Traditionally, model reduction has been a crucial research topic in atmospheric chemical kinetics. The primary research purpose however is not to set up faster models, but to allow for a better understanding of the inherent dynamics. A reduced model is therefore not necessarily faster. Among a wide range of different approaches, we choose the class of repro-modeling approaches for the construction of fast, coarse propagators, that still provide a sufficient level of accuracy. Repro-models are functional representations, that map the solution from one point in time to another point in time and only valid for a fixed distance. This again requires a decomposition into equidistant time-slabs.

This thesis starts with an introduction into atmospheric chemical kinetics and its numerical treatment within compound air quality models. Current parallelization strategies are discussed. Chapter 3 outlines the principles of parallel-in-time integration techniques with a focus on the parareal algorithm. Special emphasis will be put on parareal techniques for the solution of multi-scale ordinary differential equations. Numerical results for first adaptive parareal tests will then be shown in Chapter 4. The results from that Chapter will motivate the search for reduced models to be applied as coarse propagators. An overview of model reduction approaches for atmospheric chemistry is then given in Chapter 5. In Chapter 6, we will thereupon present a new repro-model parareal approach, using functional representations as coarse propagators. Numerical results will be shown for three realistic atmospheric chemistry scenarios, that showcase the calculation of the chemical kinetics as it is incorporated in a real AQM by means of zero-dimensional box-models. Finally, Chapter 7 closes with conclusions and outlook.

Chapter 2

Atmospheric Chemical Kinetics

Computer programs to simulate the decomposition of the atmospheric air, conventionally denoted as air quality models (AQMs), base on chemical transport models (CTMs), that describe the temporal evolvement of chemical species due to advection, diffusion, chemical reaction and further atmospheric processes. For the numerical approximation of a CTM, one typically chooses an operator splitting approach, such that the individual processes are decoupled and solved in succession. One very time-consuming part within is the numerical solution of the chemical reaction kinetics. Due to the presence of a wide range of timescales within such systems, special integration techniques are required. This Chapter first outlines the basic principles of chemical transport modeling along with a depiction of a numerical solution approach. From then on, the focus will be put on chemical reaction kinetics. For illustration purposes, an exemplary atmospheric chemical mechanism is introduced, which will serve as a testbed throughout this work. Then, characteristics and challenges in solving chemical kinetics in atmospheric chemistry are focused, followed by a presentation of respective solvers. Finally, the Chapter is closed by considerations on the parallelization of the calculation of atmospheric chemical kinetics.

2.1 Air Quality Models

Atmospheric air quality models aspire a prediction of ambient concentrations of atmospheric pollutants to facilitate a monitoring of the air quality. To this, they numerically simulate the physical and chemical processes, ongoing in the atmosphere. Atmospheric pollutants may be chemically transformed, transported, diffused, mixed, diluted, washed-out by precipitation or removed through deposition. Chemical transport models (CTMs) provide the necessary mathematical framework, that allows an assessment of the integrated effects of these processes on the air quality.

Atmospheric chemistry is being affected by a wide range of processes, which all take place simultaneously. Some of the processes are directly influenced by the ambient weather and climate, such as wash-out by precipitation or chemical reaction by radiation. Vice versa, also the composition of the air influences weather and climate, for example aerosols affecting the radiation budget. Both meteorology and air quality are in fact linked.

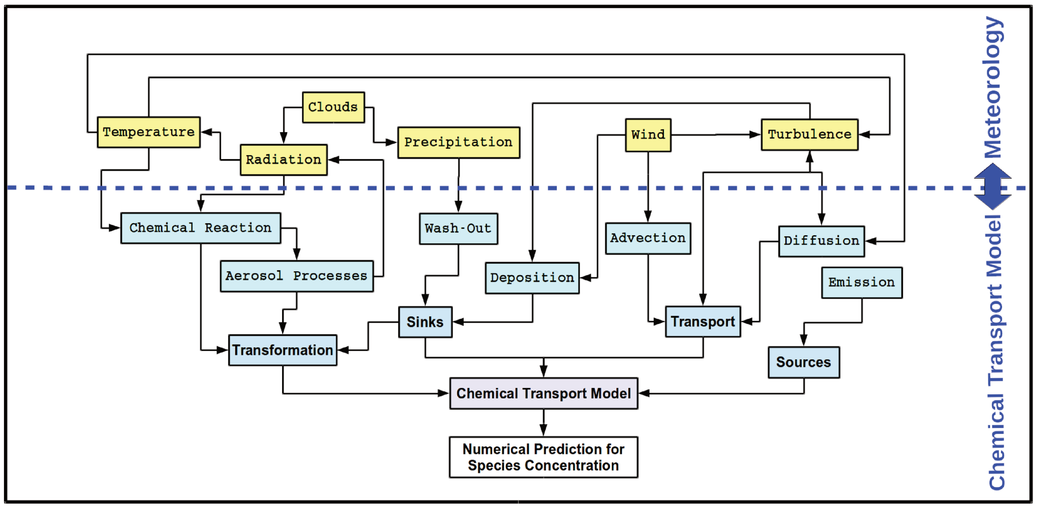


Figure 2.1: Schematized overview of an atmospheric air quality model.

Since a CTM only describes the temporal evolvement of chemical species, meteorological variables, however, are not modeled within the CTM itself. Typically, they enter the CTM as fixed values. These can be provided either from measured data, statistical models or an external meteorology model. Usually, an AQM is understood as a superordinate framework for the numerical solution of a CTM, coupled to a meteorological model. Models of meteorology, such as weather forecasting or climate models, describe the temporal evolvement of the atmospheric dynamics by means of prognostic or diagnostic equations for wind, temperature, pressure and density.

Historically, CTMs have been developed separately from meteorological models for decades [8]. In the current numerical praxis within AQMs, they therefore are only loosely coupled to each other: In the so called *offline coupled* AQMs, the CTMs are driven by a-priori calculated meteorological input data, calculated using a weather or climate forecasting model. Offline coupled AQMs do not account for effects of the air composition on meteorology. Recent developments adopt closer couplings of meteorological models and CTMs, by means of a so called *online coupling*, that allows for an incorporation of two-way interactions. Different than for aforementioned offline coupled models, data is exchanged in both directions at certain coupling time intervals. An extensive overview over current state of the art approaches used in regional European meteorology-chemistry models has been presented by Baklanov et al. [8].

An AQM typically represents each of the contributing processes by means of an individual component. A schematized overview of the main components within an AQM is drafted in Fig. 2.1. For a comprehensive overview, we refer to Seinfeld and Pandis [111] or Baklanov et al. [8]. Yellow boxes in Fig. 2.1 denote meteorological variables, which are not directly modeled within the CTM, but typically enter the CTM as external parameters. Green boxes represent the components, a CTM may be comprised of. These components describe the physical and chemical processes, atmospheric chemical species are exposed to in the atmosphere. Blue boxes signify greater process categories.

A chemical species may be transformed through chemical reaction in the gas-phase or the aqueous-phase of the atmosphere or through aerosol effects. It may further be influenced by sinks, such as wash-out by means of precipitation or deposition. Emission in contrast represents a source term. Species can further be transported by means of advection and diffusion. Diffusion describes the spreading of chemical species from highly concentrated areas to less concentrated areas. It is caused by a non-directional random movement of particles, induced by the presence of thermal energy. Different than advection, it is not related to a transporting medium, i.e. the wind field, but an autonomous movement. In fact, the denotation "transport" in this context is misleading, but conventional. In contrast, advective and turbulent transport relies on a transport of chemical species by means of the surrounding wind field.

The formulation of a CTM, as well as the formulation of a meteorological model, depends on the point of view of the modeler: In a *Lagrangian* framework one models a specific air parcel, that is being advected with the local wind. Respectively, one moves with the air parcel and its surroundings. Exchange of mass, momentum or energy across cells through transport is not considered. In a *Eulerian* framework one describes the processes in fixed grid cells. Mass, momentum or energy is allowed to enter and leave grid cells. A schematic overview of the Lagrangian and the Eulerian approach is presented in Fig. 2.2. Since the Eulerian approach is commonly used in practice within three-dimensional AQMs, the following equations will be presented in a Eulerian view.

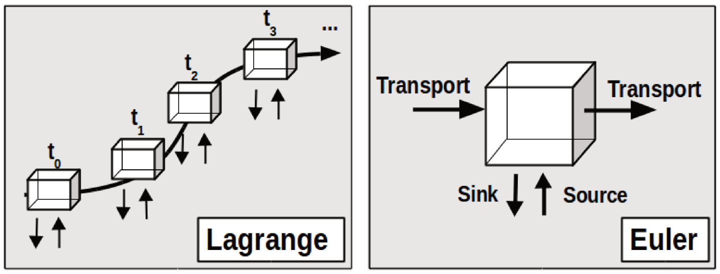


Figure 2.2: Lagrangian vs. Eulerian modeling framework.

2.1.1 Meteorological Model

As it has been outlined in the previous section, AQMs usually represent a CTM, coupled to a meteorological model. Potentially, the CTM could also be coupled to measurements or statistical models. For the sake of completeness, we briefly discuss the meteorological model here. The purpose of the meteorological model is to approximate the atmospheric dynamics. A wide variety of different models exist, each tailored to the intended purpose of usage. All of them base on the same basic sets of balance equations, which are known under the term *primitive equations*. These equations have first been formulated already in 1922 by Richardson [103]. Even today, they still formulate the so called dynamical core of most current atmospheric models.

Physically, the primitive equations can be derived from the principles of conservation of mass, conservation of momentum and conservation of energy, amended with the ideal gas equation. Two basic approximations are used: First, a hydrostatic approximation for the vertical momentum equation. Second, the Coriolis force is affected by horizontal wind components only. Their derivation can be found in most books on theoretical meteorology, c.f. [48], and shall not be focused here.

Equation system 2.1.1 shows a standard formulation of the primitive equations in cartesian coordinates. Since the equations are typically solved on a rotating sphere, it is common to formulate them in spherical coordinates. For the sake of simplicity, they are formulated in cartesian coordinates, here.

$$\begin{aligned}
\frac{\partial u_h}{\partial t} + u \cdot \nabla u_h + f(k \times u) &= -\frac{1}{\rho}\nabla_h p && \text{horizontal momentum equation}\\
\frac{1}{\rho}\frac{\partial p}{\partial z} &= -g && \text{hydrostatic equation}\\
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) &= 0 && \text{continuity equation}\\
c_p\left(\frac{\partial \Theta}{\partial t} + u \cdot \nabla \Theta\right) &= \left(\frac{p_0}{p}\right)^{\kappa} J && \text{thermodynamic energy equation}\\
p &= \rho R T && \text{ideal gas law}
\end{aligned} \tag{2.1.1}$$

whereat $u \in \mathbb{R}^3$ denotes the three-dimensional wind field $u = (u_x, u_y, u_z)$ and $u_h \in \mathbb{R}^2$ the horizontal components only $u_h = (u_x, u_y)$. Respectively ∇ and ∇_h denote the three-dimensional and horizontal gradient operators $\nabla = (\partial_x, \partial_y, \partial_z)$ and $\nabla_h = (\partial_x, \partial_y)$. The variable $f \in \mathbb{R}$ represents the Coriolis parameter, $k \in \mathbb{R}^3$ denotes the unit vector in vertical direction. Further, $p \in \mathbb{R}$ is pressure, $\rho \in \mathbb{R}$ density, $g \in \mathbb{R}$ gravitational acceleration, $c_p \in \mathbb{R}$ the specific heat at constant pressure, $\kappa = R/c_p \in \mathbb{R}$ and $R \in \mathbb{R}$ the gas constant for dry air. The term $J \in \mathbb{R}$ represents the diabatic heating per unit mass, such as radiation.

Above equation system is typically amended with further processes, that enter the equation system in a parametrized way. Typical parametrized processes are cloud micro-physics, radiation, precipitation, turbulence or convection. For the application within an AQM, above meteorological model system is further amended with a model for the chemical transport of atmospheric chemical species. In practice, the resulting equation system is not solved in a fully coupled fashion. Instead, the meteorological model is solved isolatedly from the CTM. Data is exchanged only at certain coupling intervals.

2.1.2 Chemical Transport Model

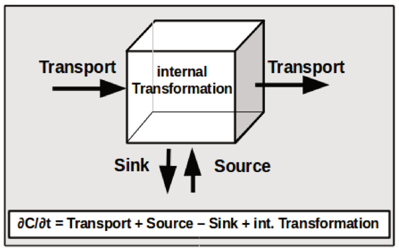


Figure 2.3: Mass balance within a fixed grid cell in the Eulerian view.

Physically, CTMs can be derived from mass balance equations for each chemical species at each grid cell. The conservation of mass of one chemical species $C = C(x,t) \in \mathbb{R}$ with $t \in \mathbb{R}$ and $x \in \mathbb{R}^3$ on some domain $\Omega \subset \mathbb{R}^3$ may be expressed in a differential form,

$$\frac{\partial C}{\partial t} + \nabla \cdot J = S, \tag{2.1.2}$$

whereat $J \in \mathbb{R}^3$ represents the vectorial mass flux into/out of the volume, S represents a sink and source term plus the rate of the internal transformation within the volume. Figure 2.3 visualizes the key principle: The rate of change of the concentration of a chemical species within one grid cell is defined by the net flux into/out of the cell (by means of advection or diffusion), the production or destruction rate within the cell through chemical or physical transformation processes and the rate of sinks (wash-out, deposition) and sources (emission).

The mass flux combines both effects of advection and diffusion, $J = J_D + J_A$. According to [111] and [3], atmospheric diffusion is assumed to follow Fick's law and hence can be modeled as proportional to the concentration gradient,

$$J_D = -K\nabla C,$$

with $K \in \mathbb{R}^{3\times 3}$ representing the diffusion coefficient, a diagonal matrix holding turbulent eddy diffusivities. The advective mass flux arises from linear advection and can be expressed as

$$J_A = Cu,$$

whereat $u = u(x,t) \in \mathbb{R}^3$ represents the three-dimensional wind velocity field. The total mass flux is then defined as $J = -K\nabla C + Cu$. Equating the latter in Eq. 2.1.2 yields a three-dimensional advection-diffusion equation,

$$\frac{\partial C}{\partial t} + \underbrace{\nabla \cdot (Cu)}_{\text{advection}} = \underbrace{\nabla (K \nabla C)}_{\text{diffusion}} + S.$$

In the following, we consider a vector $c = c(x,t) \in \mathbb{R}^s$ holding the concentrations of s chemical species. The temporal evolvement of the concentration of the s chemical species in a fixed air parcel can then be described by means of

$$\frac{\partial c}{\partial t} = - \underbrace{\nabla \cdot (cu)}_{\text{advection}} + \underbrace{\nabla (K \nabla c)}_{\text{diffusion}} + S.$$

Now let us specify the the source/sink and transformation term S. This term represents emission E as a source, sinks by means of wash-out and deposition, further the internal transformation processes caused by chemical reaction and aerosol,

$$S = E + \left(\frac{\partial c_i}{\partial t}\right)_{\text{wash-out}} + \left(\frac{\partial c_i}{\partial t}\right)_{\text{depos.}} + \left(\frac{\partial c}{\partial t}\right)_{\text{chem. reaction}} + \left(\frac{\partial c}{\partial t}\right)_{\text{aerosol}}.$$

Since we focus on atmospheric chemical kinetics in this work, we will only consider the transformation due to chemical reaction in the gas-phase chemistry and neglect further effects from now on. For details on the remaining terms, see for example Seinfeld and Pandis [111] or Arya [3]. Chemical reaction is represented by an ordinary differential equation of the form

$$\left(\frac{\partial c}{\partial t}\right)_{\text{chem. reaction}} = f(c,\ k,\ t), \tag{2.1.3}$$

whereat f describes the temporal change of the species concentrations as a function of concentration, reaction rates k and time. The reaction rates are a function of temperature, pressure and photolysis and will be discussed in more detail later. The resulting mass balance equation reads

$$\frac{\partial c}{\partial t} = -\underbrace{\nabla \cdot (cu)}_{\text{advection}} + \underbrace{\nabla \cdot (K \nabla c)}_{\text{diffusion}} + \underbrace{f(c,k,t)}_{\text{chem. reaction}} \quad . \tag{2.1.4}$$

As it has been depicted in the previous Section, CTMs are typically coupled with a meteorological model, that provides meteorological parameters, such as wind, temperature or pressure. For Eq. 2.1.4 this means, that the wind field u, the coefficient matrix K and the reaction rates k are not calculated by the CTM itself, but provided by or derived from meteorological input data. The problem then is linear with respect to advection and diffusion. In most cases, the reaction term f will be nonlinear.

2.1.2.1 Numerical Solution and Operator-Splitting

For the numerical solution of a CTM, as defined by Eq. (2.1.4), the problem typically is first discretized on a three-dimensional spatial grid $\Omega_h \subset \Omega \subset \mathbb{R}^3$. This leads to a semi-discrete system consisting of a huge number of ODEs. A survey over popular approaches to solve those systems can be found in Verwer et al. [128]. One of the most popular approaches is the so called operator splitting approach, cf. McRae, Goodin and Seinfeld, [87]. Since its introduction in the early 1980s it has found widespread use in chemical transport modeling. The basic idea is a decoupling of the different processes, to allow for using individual numerical time-stepping schemes for each of the components. To this end, the individual processes are solved in succession over split intervals $[t_n,\ t_{n+1}]$ with $t_{n+1} = t_n + \Delta t_{\text{split}}$ for $n = 0, 1, 2, ...$ An analysis of operator splitting and insight into the splitting error for advection-diffusion-reaction equations has for example been presented by Lanser and Verwer [61]. The key idea shall be outlined in the following.

Assume, we can describe the impact of the different processes comprising Eq. 2.1.4 on $c(t + \Delta t)$ by means of individual operators:

$$\begin{array}{lcll} \mathrm{A}(c,\ \Delta t) & := & -\nabla \cdot (uc) & \text{advection operator} \\ \mathrm{D}(c,\ \Delta t) & := & \nabla \cdot (K \nabla c) & \text{diffusion operator} \\ \mathrm{G}(c,\ \Delta t) & := & f(c,t) & \text{gas-phase chemistry} \end{array}$$

The concentration c at the next time-step $t + \Delta t$ can then be described by

$$c(x,\ t + \Delta t) = c(t) + [\mathrm{A{+}D{+}G}]\,(c(x,\ t),\ \Delta t). \tag{2.1.5}$$

We split up the difference $\Delta c = c(x,\ t) - c(x,\ t + \Delta t)$ into contributions from an isolated application of the individual processes, with Δc^{A} representing the difference arising from an isolated application of the advection operator, respectively Δc^{D} and Δc^{G}. A straight-forward, but inaccurate approach would then be to approximate the new solution at time $t + \Delta t$ as

$$c(x,\ t + \Delta t) \quad = \quad c(x,\ t) + \Delta c^{\mathrm{A}} + \Delta c^{\mathrm{D}} + \Delta c^{\mathrm{G}},$$

From a physical point of view, these processes take place at the same time and are not isolated. Applying the operators isolatedly in parallel is therefore not a good approximation to the real physical nature of Eq. 2.1.4. Other approaches base on a sequential application of the operators. One popular approach for the solution of CTMs is a symmetric splitting for advection and diffusion as proposed by McRae et al. [87],

$$\begin{aligned} c^1(x,\ t+\Delta t) &= \mathrm{A}(c(x,\ t),\ \frac{\Delta t}{2}) \\ c^2(x,\ t+\Delta t) &= \mathrm{D}(c^1(x,\ t),\ \frac{\Delta t}{2}) \\ c^3(x,\ t+\Delta t) &= \mathrm{G}(c^2(x,\ t),\ \Delta t) \\ c^4(x,\ t+\Delta t) &= \mathrm{D}(c^3(x,\ t),\ \frac{\Delta t}{2}) \\ c(x,\ t+\Delta t) &= \mathrm{A}(c^4(x,\ t),\ \frac{\Delta t}{2}), \end{aligned}$$

whereat advection and diffusion are applied twice during a splitting interval, while the chemistry operator is applied only once.

Within such an approach, the most time consuming part to solve the mass balance equation (2.1.4) within a splitting interval is the numerical integration of the chemical kinetics, cf. [142]. Given some spatial discretization $\Omega_h \subset \Omega \subset \mathbb{R}^3$ and under the assumption of the splitting of advection, diffusion and reaction, as introduced above, solving chemical kinetics means solving

$$\frac{\partial c}{\partial t} = f(c,\ k,\ t) \tag{2.1.6}$$

on each grid entity of Ω_h. Typically, one further assumes an autonomous mode, i.e. $f(c) \neq f(c,\ k,\ t)$ during one splitting interval. Rate constants then are not updated during the splitting interval itself, but at the end only. Chemistry thus is considered to take place in a constant environment with fixed photolysis conditions, pressure and temperature. After the integration time of $\Delta t_{\mathrm{split}}$, the isolation is canceled and species are advected and diffused and rate coefficients are updated to the current photolytic and thermodynamic state of the system. Computationally this means, that one isolated ODE system is being solved per grid entity, without any interaction between neighboring grid entities during $[t_n,\ t_n + \Delta t_{\mathrm{split}}]$. Section 2.3 will outline the major challenges and requirements arising from the split approach for the numerical treatment of the chemical kinetics. It further presents the most popular solvers for atmospheric chemical kinetics. In Sec. 2.4, aspects related to the parallelization of atmospheric chemical kinetics will be discussed. In advance of all that, the chemical mechanisms, that will later define the set of ODEs given in Eq. (2.1.6), are taken into account.

2.2 The Chemical Mechanism

The study of chemical kinetics in the atmosphere includes complex and sophisticated reactions between trace gases, such as ozone, nitrogen oxides, methane and hydrocarbons. These can't be analyzed without proposed chemical mechanisms, that step-by-step describe the occurrences in chemical reactions on molecular level before equilibrium is reached.

As a testbed for this work, we choose the chemical mechanism Regional Acid Decomposition Model version 2 (RADM2), which has been developed by Stockwell et al. [119]. Since its development in 1990, RADM2 has found wide usage in atmospheric AQMs to predict concentrations of air pollutants and oxidants. The RADM2 mechanism represents tropospheric chemistry by means of 63 species that interact in 158 reactions. Inorganic chemistry comprises 4 radical and 17 stable species, whereat N_2, O_2, H_2O are abundantly stable. Organic chemistry is represented by 26 stable and 16 radical species or species groups. For the numerical treatment of the hundreds of volatile organic compounds (VOCs) in the atmosphere, the RADM2 mechanism follows a molecular lumping approach proposed by Middleton [89]. Depending on their emission rates and their reactivity with the hydroxyl radical HO, VOCs are aggregated into 15 species groups (e.g. HC_3, HC_5, HC_8, HC_3, OL2, OLT,...). For details on the classification, see [89]. A comprehensive description of the RADM2 mechanism is given in Stockwell et al. [119]. Lists of the chemical species, the elementary reactions and the reaction rates are also presented in the Appendix.

2.2.1 Mathematical Model

In a homogeneous reactor, a chemical mechanism comprised of R elementary reactions of S species $\mathrm{A}_{\mathrm{s}\in\mathrm{S}}$ can be denoted by

$$\sum_{s\in S} \alpha_{r,s}\mathrm{A}_s \quad \xrightarrow{\mathrm{k_r}} \quad \sum_{s\in S} \beta_{r,s}\mathrm{A}_\mathrm{s} \quad \forall r \in R,$$

with the stoichiometrical coefficients $\alpha_{r,i}$ and $\beta_{r,i}$ and the rate coefficients $\mathrm{k_r}$. The reaction system can then be described by a set of ODEs in terms of a vector of concentrations c, with c_i holding the concentration of species A_i,

$$\left(\frac{\partial c_i}{\partial t}\right) \quad = \quad \sum_{r=1}^{R} \mathrm{k_r}\left(\beta_{r,i} - \alpha_{r,i}\right) \prod_{s\in S} c_s^{\alpha_{r,s}}, \qquad \text{for } i = 1, ..., S,$$

see for example Warnatz et al. [136].

2.2.2 Example Six-Variable Tropospheric Mechanism

For the sake of clearness, the methodology developed in this work will be depicted by means of a much simpler, six-dimensional model problem, introduced by Tomlin et al. [121]. This model is a small subset of the Master Chemical Mechanism (MCM), developed by Jenkins et al. [51], containing 8 reactions and 6 variable species. The model assumes fixed photolysis conditions with a clear sky at mid-day, a solar declination of 23.79° and a zenith angle of 16.2 at a latitude of 40°. Table 2.1 shows the reactions along with the rate coefficients. The initial concentrations are presented in Tab. 2.2. For more details on the model, see Tomlin et al. [121].

Table 2.1: Reaction mechanism.

	Reaction			Reaction rate	
1	O^1D	$\xrightarrow{k_{O_2}}$	O^3P	$2.101 \cdot 10^{8}$	s^{-1}
2	O^1D	$\xrightarrow{k_{N_2}}$	O^3P	$5.060 \cdot 10^{8}$	s^{-1}
3	O^1D	$\xrightarrow{k_{H_2O}}$	$2HO$	$5.412 \cdot 10^{7}$	s^{-1}
4	$HO + CO$	$\xrightarrow{k_2}$	$HO_2 + \text{products}$	$2.384 \cdot 10^{-13}$	$\frac{1}{\text{molecule}\cdot cm^3 \cdot s}$
5	$HO_2 + NO$	$\xrightarrow{k_3}$	$HO + NO_2$	$8.941 \cdot 10^{-12}$	$\frac{1}{\text{molecule}\cdot cm^3 \cdot s}$
6	$HO + NO_2$	$\xrightarrow{k_4}$	HNO_3	$1.408 \cdot 10^{-11}$	$\frac{1}{\text{molecule}\cdot cm^3 \cdot s}$
7	$HO_2 + HO_2$	$\xrightarrow{k_5}$	$H_2O_2 + \text{products}$	$2.67 \cdot 10^{-12}$	$\frac{1}{\text{molecule}\cdot cm^3 \cdot s}$
8	O_3	$\xrightarrow{j_1}$	$O^1D + \text{products}$	$2.11 \cdot 10^{-5}$	s^{-1}

From the reactions presented in Tab. 2.1 we derive the following rate equations for the reaction educts:

$$\frac{d\,[O^1D]}{dt} = j_1\,[O_3] - k_1'\left[O^1D\right], \tag{2.2.1}$$

$$\frac{d\,[HO]}{dt} = 2k_{H_2O}\left[O^1D\right] - k_2'\,[HO] + k_3\,[HO_2]\,[NO] - k_4\,[HO]\,[NO_2], \tag{2.2.2}$$

$$\frac{d\,[HO_2]}{dt} = k_2'\,[HO] - k_3\,[HO_2]\,[NO] - 2k_5\,[HO_2]^2, \tag{2.2.3}$$

$$\frac{d\,[NO_2]}{dt} = k_3\,[HO_2]\,[NO] - k_4\,[HO]\,[NO_2], \tag{2.2.4}$$

$$\frac{d\,[NO]}{dt} = -k_3\,[HO_2]\,[NO], \tag{2.2.5}$$

$$\frac{d\,[O_3]}{dt} = -j_1\,[O_3]. \tag{2.2.6}$$

Eqs. (2.2.1 -2.2.6), represent a set of nonlinear, stiff ordinary differential equations (ODEs). Initial conditions are given in Tab. 2.2.

2.3 Chemical Kinetics Solvers

Solving chemical kinetics accounts for most of the CPU time to solve the mass balance equations, Eq. (2.1.4): According to Zhang et al. [142] the percentage may account up to 95% of the total CPU time. For an efficient solution of atmospheric gas-phase chemistry, special integration techniques are required. This section outlines the major challenges and requirements such integrators have to cope with. For comprehensive discussions, see for example Zhang et al. [142] or Verwer et al. [128]. We start with some basic definitions for one-step methods.

2.3.1 Convergence, Order and Stability

Solving chemical kinetics means solving an autonomous ODE system of the form

$$\frac{\partial u}{\partial t} = f(u), \quad \text{with } u(0) = u_0. \tag{2.3.1}$$

Table 2.2: Initial concentrations.

	Species	Description	Initial Value $[\text{molecules}\cdot\text{cm}^{-3}]$
1	O^1D	Excited state oxygen atom	$3.060 \cdot 10^5$
2	HO	Hydroxy radical	$5.660 \cdot 10^6$
3	HO_2	Hydroperoxy radical	$5.570 \cdot 10^8$
4	O_3	Ozone	$7.380 \cdot 10^{11}$
5	NO	Nitric oxide	$1.000 \cdot 10^6$
6	NO_2	Nitrogen dioxide	$5.000 \cdot 10^6$
fix	CO	Carbon monoxide	$2.458 \cdot 10^{12}$
fix	O_2	Oxygen	implicitly defined by k_1'
fix	N_2	Nitrogen	implicitly defined by k_1'
fix	H_2O	Water	implicitly defined by k_1'

	Rate coefficients in condensed form	
	$k_1' := k_{O_2}[O_2]_{\text{fix}} + k_{N_2}[N_2]_{\text{fix}} + k_{H_2O}[H_2O]_{\text{fix}}$	$7.70 \cdot 10^8$
	$k_2' := k_2 [CO]_{\text{fix}}$	$5.87 \cdot 10^{-1}$

A numerical method approximates solutions U to Eq. (2.3.1) at discrete times t_n with $U_n \approx u(t_n)$ and $t_n = t_0 + n\Delta t$, where Δt is the time-step size of the method. Such a method can either be a one- or a multi-step method. A one-step method calculates a solution using only information from the precedent time-step, $U_{n+1} = F(U_n)$, while multi-step methods involve information from the previous s steps to calculate a solution $U_{n+1} = F(U_n,\ U_{n-1},\ ....,\ U_{n+1-s})$. Further, one can distinguish between explicit and implicit methods. Explicit methods calculate the state at time t_n explicitly from that at previous time, $U_{n+1} = F(U_n)$, while implicit methods involve also the current state, $G(U_n,\ U_{n+1}) = 0$.

For a given numerical method, important properties from a mathematical point of view are: *Convergence, order* and *stability [26, 45, 46].*

Definition (Convergence)**.** A numerical method is *convergent*, if the numerical solution U_n at time $t = t_n$ to any ODE of the form Eq. (2.3.1) with a Lipschitz function f approaches the exact solution $u(t_n)$ for $\Delta t \to 0$, i.e.

$$\lim_{\Delta t \to 0} \max_{n=0,1...,\tau/\Delta t} \|U_n - u(t_n)\| = 0, \quad \forall \tau > 0.$$

Definition (Convergence Order)**.** A numerical method is further *convergent of order p*, if for any ODE of the form Eq. (2.3.1) with a Lipschitz function f there exists a $C \in \mathbb{R}$, such that

$$\|U_n - u(t_n)\| \leq C\Delta t^{p+1}$$

holds for all n.

For some *stiff* problems, standard numerical methods may exhibit instabilities in the solution. The stability of a numerical method describes, in which way numerical errors, that are generated during the solution are being magnified. In 1963, Dahlquist introduced a concept to rate the stability of a numerical method by means of its behavior when solving a stiff test equation [27],

$$u' = \lambda u, \quad u_0 = 1, \quad \lambda \in \mathbb{C}. \tag{2.3.2}$$

Given a numerical method, its corresponding numerical solution to Eq. (2.3.2) after one time-step of size Δt defines the stability function $R(z)$ with $z := \lambda \Delta t$. The set

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

is further called the method's stability domain, cf. [46]. The test equation Eq. (2.3.2) has an analytic solution, $u(t) = \exp(\lambda t)$, which approaches zero as $t \to \infty$ for all $\lambda \in \mathbb{C}^-$ with $\mathbb{C}^- : \ \{\lambda \in \mathbb{C};\ \mathrm{Re}(\lambda) < 0\}$. A numerical method, that exhibits the same behavior, is said to be stable. Various definitions of stability exist. We introduce the two most important stability definitions for one-step methods. For a comprehensive overview of further stability definitions, we refer to Hairer et al. [46].

Definition (A-Stability [46])**.** A numerical method is *A-stable*, if its stability domains satisfies $S \supset \mathbb{C}^-$.

An A-stable method will return a monotonic decreasing series U_n with $n = 1, 2, ..., \infty$ of approximations for all $\lambda \in \mathbb{C}^-$ and for any fixed time-step size $\Delta t \in \mathbb{R}$ with $\Delta t > 0$. The series U_n obviously approaches zero as $n \to \infty$. A-stable methods are therefore stable for any $z \in \mathbb{C}^-$, which exactly equates the stability domain of the analytic solution.

Definition (L-Stability [32])**.** A method is *L-stable*, if it is A-stable and if in addition $\lim\limits_{z \to -\infty} R(z) = 0$.

The solution of an L-stable method also approaches zero, but already after one single time-step with $\Delta t \to \infty$. L-stability implies, that the method is stable for any $z \in \mathbb{C}$.

2.3.2 Requirements on a Chemical Kinetics Solver

Accuracy The integration of chemical kinetics is typically embedded in a chemical transport model or a bigger AQM. Such models inhere multiple sources of inaccuracies, such as data and modeling errors, which are hard to quantify but always present. Demanding high accuracy in the numerical integration of the chemical kinetics would be superfluous, as its basic assumptions are inaccurate already: initial conditions, rate coefficients, incomplete chemical or physical mechanisms, for example. The overall aim is to achieve extremely fast solutions at low accuracies; Zhang et al. [142] demand relative errors below 0.1%, Verwer et al. [128] claim relative errors below 1.0% to be sufficient. The numerical error in the final solution is determined by the choice of the time-step sizes and by the order of accuracy of the numerical algorithm. During integration time, one typically controls the error by adjusting the time-step size. This requires a robust error control mechanism on the base of respective error estimators.

Stiffness The term *stiffness* can be encountered in most literature on chemical kinetics of the atmosphere. Stiffness is a vague concept. Curtiss and Hirschfelder first introduced the term in 1951 to describe a type of differential equation, that is "*exceedingly difficult to solve by ordinary numerical procedures*" [25]. Since then, various attempts towards a clear mathematical definition have been made. For a detailed depiction, see Spijker [114]. The most common definition found in literature (see e.g. [2, 28, 40, 60, 112]) defines the occurrence of stiffness as a situation, when the largest step size to guarantee numerical stability is much smaller than the largest step size for which the discretization error is still sufficiently small. In search of a definition, characteristics and criteria have been formulated to at least *describe* stiffness. One such characteristics is the requirement of implicit integration techniques to solve stiff problems, cf. [25, 46]. According to Lambert [60], stiffness is present, if the solution to the differential equations inheres some components, which decay much more rapidly than others. Or in a formal way, if

$$\frac{\max_{i=1,\ldots,s}|\mathrm{Re}(\lambda_i)|}{\min_{i=1,\ldots,s}|\mathrm{Re}(\lambda_i)|} \gg 1, \tag{2.3.3}$$

with λ_i denoting the $i-th$ eigenvalue of the Jacobian of the right hand side of Eq. (2.3.1).

The ODEs arising from atmospheric chemical kinetics certainly meet the stiffness criterion introduced by Eq. (2.3.3): Atmospheric chemistry is a multi-scalar phenomenon, as the involved species interact on scales from milliseconds (radicals such as the hydroxyl radical HO) to years (e.g. methane CH_4).

For reasons of accuracy and stability, very small time-steps are necessary, whenever stiffness occurs. In the context of chemical transport models, this is especially true for the very first milliseconds: Due to the splitting approach, transient (i.e. instable, short-lived) species will be present at the beginning of every split interval. Thus very small time-steps are required at initial time. After the transients have equilibrated, the time-step size can be increased remarkably. Quickly adjusting time-step sizes therefore are crucial for the efficiency of a solver. To solve stiff problems, unconditionally stable methods are preferable, i.e. the size of the time-step size should not be controlled by stability reasons, but by accuracy considerations only. Numerical methods for solving atmospheric chemical kinetics are desired to be L-stable, or at least A-stable. Such methods are typically implicit. For a nonlinear right-hand-side function $f(u)$ implicit methods are computationally expensive, as in general, implicit equations are solved by iteration.

Preserving natural solution properties The solution to Eq. (2.1.6) in general is required to be mass conserving and positive. Integrators are requested to hand these properties down to the numerical solution [104]. Not all integrators naturally cope with these requirements.

The conservation of mass is intrinsic to most chemistry solvers, as mass is a linear invariant of the systems being solved. Some popular solvers, e.g. QSSA [47], do not take the analytic Jacobian into account. This leads to an artificial production of mass, hence these solvers are not mass-conserving.

The conservation of positivity is not a direct constraint of Eq. (2.1.6), but a constraint of physical relevance. Most methods of order one preserve positivity unconditionally [15].

Higher order schemes do not a priori maintain positivity, some higher order schemes preserve positivity for very small time-steps only. For stability reasons, an artificial preservation of positivity is advantageous. The most simplest approach to force positivity bases on clipping, i.e. setting negative concentration values to zero. However, this approach is not mass-conserving. More elaborate techniques ensure positivity through additional post-processing steps, such as a projection onto a non-negative simplex. Methods favoring positivity (as introduced by Sandu [108]) present computationally less costly alternatives.

2.3.3 Rosenbrock Methods

As it has been depicted in the previous section, a numerical solver for Eq. (2.1.6) has to fulfill certain requirements:

– low to modest accuracy (typically $0.1 - 1\%$ is sufficient),

– stability in the presence of stiffness,

– preservation of mass and positivity.

Within the past decades, great efforts have been made to search for *efficient* stiff solvers for atmospheric chemistry problems, that fulfill above mentioned criteria (cf. [109, 110, 127, 142]). The *efficien*cy of a solver can thereby roughly be quantified by the ratio of a target accuracy to wall clock time (WCT). An overview of suitable approaches (e.g. Quasi-Steady-State-Approximation (QSSA) [47], Backward Differentiation Formulas (BDF) [46], implicit Runge-Kutta [46] and Rosenbrock [46] methods) can be found in Zhang et al. [142]. With its high efficiency at moderate accuracy requirements the class of Rosenbrock methods has become very popular among them. Compared to most other implicit solvers, they present a computationally light concept: They avoid nonlinear systems by replacing them by a sequence of linear systems. It has to be emphasized though, that there is no universally *good* or *bad* choice - it always depends on the specific problem and on the individual tuning of a solver.

Rosenbrock methods can be put down to Runge-Kutta methods: Instead of applying multiple Newton iterations at each stage, they can be interpreted to apply one Newton iteration per step only. In literature, they therefore are often encountered as *linearly implicit Runge-Kutta methods* (cf. [46]). A general $s-$stage Rosenbrock method [46, 105] applied to an autonomous system with $f = f(u) \neq f(u,\ t)$ can be given by

$$\begin{aligned} k_i &= h_n\, f\left(U_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j\right) + \Delta t\ J \sum_{j=1}^{i} \gamma_{ij} k_j, && (2.3.4) \\ U_{n+1} &= U_n + \sum_{j=1}^{\sigma} b_j k_j && (2.3.5) \end{aligned}$$

for $i = 1, ..., \sigma$ and with the Jacobian $J = \partial_{y_n} f(U_n)$, the determining coefficients α_{ij}, b_j and γ_{ij}. Each of the σ stages consists of solving one system of linear equations with k_i unknowns and the matrix $\mathbb{I} - h_n\ \gamma_{ii} J$. For performance purposes, methods for which $\gamma_{11} = \gamma_{22} =$

... $= \gamma_{rr} = \gamma$ are advantageous, as then, the LU-factorization for $(\mathbb{I} - h_n\, \gamma_{ii} J)$ has to be solved once only. In this case, the total computational effort for one time-step consists of one evaluation of the Jacobian, one real $d \times d$ LU-factorization (for $u \in \mathbb{R}^d$), σ forward and backward substitutions and σ function evaluations.

To control the step size selection local error estimators are employed. Those are typically constructed by means of a lower order solution $\hat{U}_{n+1}$, which can be obtained using the same increment vectors k_j but different weights $\hat{b}_j$ as in Eq. (2.3.5),

$$\hat{U}_{n+1} \;=\; U_n + \sum_{j=1}^{\sigma} \hat{b}_j k_j .$$

The weights $\hat{b}_j$ are chosen, such that the order of consistency $\hat{p}$ of $\hat{U}_{n+1}$ is one below that of U_n. Local error estimators incorporate the difference between $\hat{U}_{n+1}$ and U_{n+1} in a scaled norm, while considering user defined error tolerances,

$$\begin{aligned} \text{Err} &= \left\| \hat{U}_{n+1} - U_{n+1} \right\| \\ &= \sqrt{\frac{1}{d} \sum_{j=1}^{d} \left(\frac{\left(\hat{U}_{n+1} - U_{n+1}\right)_j}{\text{RelTol}_j \cdot \max\left\{|U_{n,j}|, |U_{n+1,j}|\right\} + \text{AbsTol}_j} \right)^2 } . \end{aligned} \tag{2.3.6}$$

A step is accepted only if Err ≤ 1. Rejected steps are repeated with smaller time-step sizes. The new step size is predicted through an asymptotic formula,

$$h_{\text{new}} \;=\; h_{\text{old}} \cdot \left[\min\left(\phi_{\max},\; \max\left(\phi_{\min},\; \phi_{\text{safe}} \cdot \text{Err}^{-1/(\hat{p}+1)} \right) \right) \right],$$

with the upper and lower bounds $\phi_{\max}$ and $\phi_{\min}$ for $\phi = h_{\text{new}}/h_{\text{old}}$, a safety factor ϕ_{safe} and the limitation that $h_{\min} \leq h_{\text{new}} \leq h_{\max}$ and $h(t = t_0) = h_{\text{start}}$.

2.4 Global Efficiency and Parallelization

We define the *efficiency* of the adaptive Rosenbrock solver as the ratio of a target accuracy to WCT per solver call. As a thumb-rule, the WCT for its serial execution can roughly be approximated as a linear function of the number of accepted time-steps within a split interval, multiplied with the computing time spent per step.[1] The number of time-steps is strongly influenced by the time-stepping control mechanism. The control mechanism can be tuned individually e.g. defining the smallest and largest step size, the size ratio between subsequent steps or the maximum number of step sizes. The computing time per step can further be reduced for example by using sparse linear algebra implementations.

In application within an AQM, the overall efficiency of the solver does not only depend on the efficiency of a single solver call, but also on its global implementation into the AQM it

[1]For decreasing relative tolerances, the number of rejected time-steps increases nonlinearily, which leads to an additional computational effort, that is not considered in this approximation. As the number of rejected time-steps typically is very small compared to the number of accepted time-steps, the computational effort for the rejected time-steps is neglected in the following.

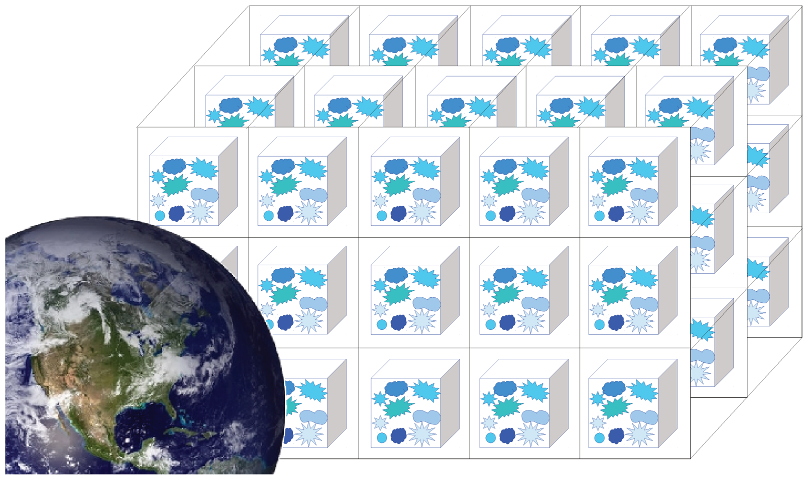


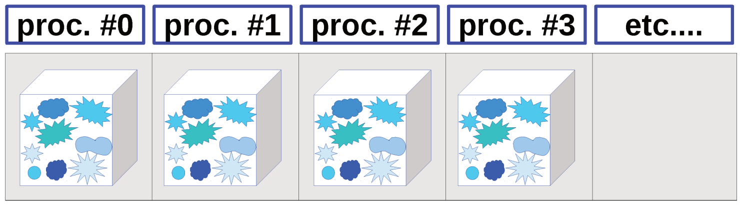
Figure 2.1: Simulation of chemical kinetics in the atmosphere at time t_n. The atmosphere is discretized into a three-dimensional computing grid. Due to the splitting of chemical reaction and other processes, one individual chemical system (depicted as a box with reactions) has to be solved per grid entity. In total, a plurality of solver calls have to be carried out for every splitting time-step.

is embedded in. Due to the splitting of chemical reaction and other processes, one chemical system has to be solved per grid entity and per splitting time-step. In total, a plurality of solver calls have to be carried out for every splitting time-step, see Fig. 2.1. In this context, we define a *global efficiency*, as the ratio of a target accuracy to the WCT for the execution of the plurality of individual solver calls per splitting time-step $[t_n,\ t_n + \Delta t_{\text{split}}]$. In an AQM one solver call has to be carried out per grid entity on Ω_h and per splitting time-step $[t_n,\ t_n + \Delta t_{\text{split}}]$. The key influence factor on the global efficiency is parallelization. We identify two tiers of parallelization: A coarse-grained tier for the parallel execution of multiple solver calls, and a fine-grained tier for the parallel execution of one single solver call, see Figs. 2.2a and 2.2b. Both tiers will be explained in the following.

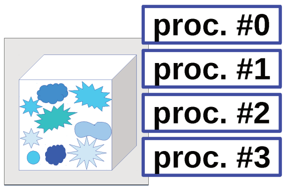
2.4.1 Coarse-Grained Parallelization

The splitting approach outlined in Sec. 2.1 allows a trivial parallelization of the chemical kinetics on each grid entity on Ω_h. As chemical reaction is considered to be isolated during one split interval, data level parallelism exists over the entire grid. On a discrete computing mesh $\Omega_h \in \Omega$ (e.g. a cubic mesh with mesh widths $\Delta x, \Delta y$ and Δz, $\Omega_h := \{(x, y, z) \in \Omega;\ x = k\Delta x,\ y = l\Delta y,\ z = m\Delta z \quad \text{mit } k, l, m \in \mathbb{Z}\}$), potentially $P = k \cdot l \cdot m$ independent initial value problems can be solved over $[t_n,\ t_n + \Delta t_{\text{split}}]$ in parallel, see also Fig. 2.2a.

On the coarsest level, data-level parallelism (DLP) can be utilized for a parallelization across multiple processors. Typically, this level is supplied in current days AQMs. A coarse-grained parallelization of the overall AQMs is achieved through *domain decomposition*, a



(a) The chemical system for each grid entity is associated with one processor each.



(b) The chemical kinetics on each grid entity is associated with multiple processors.

Figure 2.2: Coarse-grained parallelization (left) vs. fine-grained parallelization (right).

decomposition of the discrete computing mesh Ω_h into smaller domains. Hence, the chemical kinetics on each grid entity belonging to one domain are solved in succession by the same processor. For most atmospheric AQMs there is still a high potential for a broader exploitation of DLP across multiple processors.

DLP can also be exploited within an single-instruction-multiple-data (SIMD) style processor, i.e. instruction lines within the chemical algorithm simultaneously act on different data. Traditionally, this approach stems from the vector supercomputers of the 1980s and early 1990s. Currently, the idea experiences a revival: Modern accelerator architectures such as graphical processing units (GPUs) inhere SIMD features.

Both approaches towards a coarse-grained parallelization using DLP are promising on current multi-core architectures. To optimally benefit from this potential, the implementation has to be tailored to the computing architecture. Pioneering work into that direction has been presented by Linford et al. [69–73], who a.o. implemented and tested a three-stage Rosenbrock solver on different multi-core platforms [71]: a homogeneous multi-core CPU chipset Intel Quad-Core Xeon, a NVIDIA GPU and a heterogeneous multi-core Cell Broadband Engine Architecture (CBEA). On the Intel Quad-Core Xeon, a parallelization had been achieved through a straightforward one cell-per thread decomposition, which was directly implemented with OpenMP. An almost linear speed-up for up to eight cores was the result for both single and double precision with a maximum of 7.5×. Also for the NVIDIA implementation, a one-cell-per-thread approach was adopted. In a first tentative port, the entire Rosenbrock solver had been implemented with CUDA as one single kernel. The authors then dismissed this approach because of poor performance results. In a second approach, they only ported lower level instructions to the GPU in form of one kernel per operation (e.g. function evaluation or vector operations), while the more sophisticated time- and error-control logic was computed on a CPU. This approach showed the major drawback, that all independent initial value problems are forced to use the same time-step size and number of time-steps. To reduce the overall computing effort, time-step size and errors were thereupon stored separately for each grid entity, which allowed converged initial value problems to relinquish the GPU cores. Due to memory limitations, the maximum speed-up was 8.5× in single precision. For the port on the CBEA, a master-worker approach was chosen: The Power Processor Unit (PPU) as a master prepares and manages data for its slaves, the Synergistic Processing Units (SPUs). The SPUs operate on 128-bit vectors, which allows to benefit from a SIMD parallelization. On each SPU data from two or four independent initial value problems (depending on the precision) could be processed at the same time, resulting in a four-cell-per-thread decomposition in single precision and a two-cell-per-thread decomposition in double precision. A maximum speed-up of 41.1× was achieved. Building on the success of the implementations suggested in [71], Linford et al. subsequently released a software tool, that automatically generates code to simulate chemical kinetic system on multi-core platforms: the accelerated Kinetic PreProcessor KPPA [72].

As chemical reaction is embarrassingly parallel on a fixed grid (due to its potential for a parallelization over the grid entities) , a coarse-grained parallelization based on data-level parallelism potentially allows a speed-up of a factor of $P = k \cdot l \cdot m$ - given the computing architecture supports P independent, but concurrently working threads. Now assume Q threads are available with $Q \gg P$. No further benefit can then be achieved from coarse-grained parallelism on a fixed mesh Ω_h. By the time the number of threads exceeds the number of individual initial value problems, further parallelization approaches therefore have to be considered.

2.4.2 Fine-Grained Parallelization

As the potential of a coarse-grained parallelization is limited to the number of grid entities in Ω_h, we examine the potential of a fine-grained parallelization within one solver call. The starting point is, that each grid entity has its own $Q/P > 1$ threads available, as depicted in Fig. 2.2b.

On a data level, the potential of a parallelization is limited and restricted to lower-level linear algebra routines. This approach is mainly limited by the problem sizes within the solver, or in other words, the number of chemical species involved in a reaction. For a typical application in an atmospheric AQM, the number of chemical species ranges between 40 and 100. For the mechanism considered here, RADM2, 63 species are involved, leading to problem sizes of 63. Using parallel linear algebra techniques is of course possible. The success however is disputable, since the performance gains achieved from the parallelization of such small-sized problems will possibly not even balance the parallelization overhead. As it will be pointed out later (Chapter 5), the number of chemical species specified above, does not arise from chemical or physical reasons, but from computational convenience. From a chemical point of view, thousands of species may be involved. If they are explicitly incorporated into the mechanism, parallel linear algebra techniques become highly promising. With the mechanisms considered in popular atmospheric chemistry models, however, parallel linear algebra techniques do not qualify for significant performance gains.

Within one solver call, the main availability of parallelism is on an instruction level. A fine-grained tier of parallelization can therefore mainly be accomplished through a splitting of the integration algorithm into independent instruction lines. These can be run in parallel on different threads, but acting on the same data. Rosenbrock methods in their original form have limited potential for a parallelization. To identify potential instruction level parallelism, we consider the pseudo code for a three stage Rosenbrock scheme, as presented in Algorithm 2.1. Per each time-step, one $d \times d$ LU decomposition and each $\sigma = 3$ function evaluations, forward and backward substitutions are necessary. In it's classical form, a straightforward parallelization of single instruction lines is very limited (e.g. a parallelization of lines 1 and 2, also of line 4 and 5 in Alg. 2.1), since the calculation of one stage depends on the precedent stages.

Algorithm 2.1 Pseudo code for a three-stage Rosenbrock solver with $\gamma_{ij} = \gamma$.

```
1: Initialize rate constants $R_{CONST}(t,\ u)$ from species vector $U$ and meteorology $(\rho,\ q,\ p,\ t)$
2: Initialize time variables: $t = T_0$, $h = h_0$
3: while $t \leq T_{end}$ do
4:     $F_0 = f(t,\ u)$
5:     $JAC_0 = jac(t,\ u)$
6:     repeat
7:         $G =$ LU_DECOMP $(\frac{1}{h\gamma} - JAC_0)$
8:         for $\sigma = 1$ to 3 do
9:             Compute stage $K_\sigma$ from $F$ and stages $K_{1,\dots,\sigma-1}$
10:            Implicitly solve for $K_\sigma$ using $G$
11:            Update $F$ from $K_{1,\dots,\sigma}$
12:        end for
13:        Compute new $U$ from stages $K_{1,\dots,\sigma}$
14:        Compute error $Err$
15:        if Err$> 1$ then
16:            Reduce time-step size $h$
17:        end if
18:    until Err$\leq 1$
19: end while
```

Within the last 20 years, modifications to the original Rosenbrock scheme have been proposed, that allow higher levels of a parallelization. In 1995, Lirong and Degui [75] presented a modified parallel Rosenbrock method: Their basic idea was to parallelize the calculation of the increment vectors k_i with $i = 1, ..., \sigma$ (line 9 in Algorithm 2.1) into σ independent operations. This can be achieved by using the stages from previous time-steps. The construction of $k_i(t_n)$ then does not depend on the preceding stages $k_{j<i}$ at time t_n anymore, but on the stages from the previous time-step $k_{1:\sigma}(t_{n-1})$ (see Eq. 2.3.4), which allows their parallel computation. The same idea was also adopted by Cao et al. [20], who introduced slight modifications to the original algorithm proposed in [75] to improve convergence and stability properties of the algorithm. Results were presented for $p = \sigma = 2$ and $p = \sigma = 3$. Ponalagusamy and Ponnammal [99] also adopted the same approach and proposed a parallel four-stage fourth-order Rosenbrock method, that has been shown to outperform the algorithm proposed by Cao et al. [20] in terms of accuracy and performance. The authors claim, an (approximately) perfect speed-up is achievable with $p = \sigma = 4$ compared to a serial execution of the algorithm. In principle, the method can be generalized for any $p = 2^n$ independent threads with any n. However, this requires the existence of an adequate 2^n-stage Rosenbrock

scheme, which may require further studies to find optimal parameters for such Rosenbrock schemes.

Voss and Kahliq [129, 130] proposed modified Rosenbrock methods, with σ external linearly implicit stages, which each contain p additional linearly implicit internal stages. The internal stages can be computed in parallel. Communication is needed only after the completion of the σ external stages. The results for numerical tests with $p = 2$ and a three-stage fourth-order Rosenbrock scheme with a third order embedded error estimator indicate, that the approach is competitive in terms of accuracy to a classic three-step fourth-order Rosenbrock scheme for low relative tolerances. The authors conclude, that a parallelization with $p > 2$ in general is possible, but will require further studies to find optimal parameters.

All of these attempts depicted very limited parallelization levels. In the best case [99], a parallelization into $p = 4$ processes was possible, with an approximately perfect speed-up, compared to a serial execution of the parallel algorithm. In the following section, a completely different approach will be presented, that allows unlimited parallelization levels: Time-parallel methods.

Chapter 3

Parallel-in-Time Integration

In the precedent Chapter, we have been discussing the potential of a parallelization of atmospheric chemical kinetics both on a coarse- and a fine-grained level. A coarse-grained parallelization could easily be realized, as the ODEs describing chemical kinetics in a compound atmospheric AQM are embarrassingly parallel. We have seen the potential of a coarse-grained parallelization being limited by the number of grid entities in the discrete computing mesh Ω_h. Beyond that limit, the parallelization level can be increased by exploiting fine-grained parallelism. The approaches presented so far - modified parallel Rosenbrock methods - however allow only a limited level of parallelization (typically ≤ 4). In the best case, perfect speed-up could be achieved, compared to a serial execution. In principle, higher levels of parallelization are possible for some of the approaches presented, but require further parameter studies. In this Chapter, an algorithmic approach is presented, that allows additional parallelization potential. First, an overview over parallel methods for the solution of ODEs is given. Following, its most promising representative, the *parareal algorithm*, is discussed in Sec. 3.2, covering also aspects related to convergence, stability, complexity and parallel efficiency. A discussion of its applicability to chemical kinetics is given in Sec. 3.3, along with an overview of potential solution strategies from literature in Sec. a3.4.

3.1 Parallel Methods for the Solution of Ordinary Differential Equations

High performance computing has become an indispensable tool to many fields in science and engineering. In fact, computer-based numerical simulation meanwhile is one of the standard tools for many research areas. The use of HPC systems allows to solve problems, that formerly had been too computationally challenging to solve. This technology enabled scientists to conduct virtual instead of real experiments, that would be too costly or time-consuming, or in some cases like weather prediction simply not possible.

One crucial role in this development played the rapid increase in computing power of the fast hardware development within the last decades. The other essential role plays the development of highly-parallel algorithms. Already in 1964, around the same time of the emergence of the first "supercomputers", Nievergelt had foreseen the need for parallelism in numerical algorithms [97]: "For the last 20 years, one has tried to speed up numerical computation mainly by providing ever faster computers. Today, as it appears that one is getting closer to the maximal speed of electronic components, emphasis is put on allowing operations to be performed in parallel. In the near future, much of numerical analysis will have to be recast in a more *parallel* form." In the following year, Moore objected Nievergelt's first observation with his well-known "Moore's Law" [93]: The observation that the number of transistors in a dense integrated circuit doubles approximately every two years. Even though, Nievergelt was totally wrong with his first observation, the resulting considerations have been highly relevant: The need for parallel algorithms.

Since then, the development of parallel algorithms has been one active research topic to various communities in applied mathematics. In one of the typical applications, HPC systems are applied to solve huge systems of partial differential equations, as they appear for example in computational fluid dynamics. One of the classical algorithmic approach for the parallel solution of such equations adopts a parallelization in space, *domain decomposition*: The boundary value problem is split up into a series of smaller boundary values. The independency of the decomposed domains allows their parallel computation, interactions between adjacent domains are accounted for through an outer iteration. One major problem within this approach is the limitedness of the speed-up: The higher the granularity of the domain decomposition, the more iterations are needed until convergence. Further, also the communication overhead grows with increasing granularity.

As the potential of a parallelization in space is limited, further parallelization strategies have to be considered. A domain decomposition in the temporal domain is one obvious possibility, which is also highly relevant for the solution of initial value problems, where a spatial decomposition is invalid. However, a straightforward parallelization in time of such problems is counterintuitive: Such equation systems describe states of e.g. physical, chemical or biological systems with individual processes taking place one after another. Sequentially. The system's state at some time t depends on all earlier, states $u(\sigma)$ with $\sigma \leq t$. Standard numerical methods for the approximation of the solution $u(t, x)$ for $t > T_0$ (with given $u_o = u(T_0)$ for the initial time T_0) approximate the solution at successive time-steps. If this procedure is to be parallelized, the solution at some later time will have to be known in preface. Apparently, such schemes need an iterative outer loop to correct the initial guesses on each sub-interval.

The first tentative example of such a parallel-in-time solution scheme for initial value problems was proposed by Nievergelt in 1964 [97]. Nievergelt's idea was to split the integration period $t = [T_0,\ T_{\text{end}}]$ into N subsequent intervals $[t_0 = T_0,\ t_1]$, $[t_1,\ t_2]$, ..., $[t_{N-1},\ t_N = T_{\text{end}}]$. Using a coarse prediction scheme, initial predicted values are found on each of the sub-intervals. On the first interval, the real (approximate) solution at time $x_1 = x(t = t_1)$ is calculated. On each of the following intervals, M_n with $n = 1, ..., N-1$ different solution branches are calculated with respective initial values y_n^j with $j = 1, ..., M_n$, centered around one coarsely predicted initial value per interval. These solution branches $[\chi_n,\ \chi_{n+1}]$ all can be computed in parallel. Following, the branches are connected by an interpolation: The end value of the first branch χ_1 is interpolated with the M_1 solution branches of the second interval $[\chi_1,\ \chi_2]$. The resulting solution χ_2 is again interpolated with all the M_2 solution branches of the third interval $[\chi_2,\ \chi_3]$, etc. until final time is reached. Later, the algorithm was extended into a parallel multiple shooting technique by Khalaf and Hutchinson [54] and Kiehl [55].

In 1967, Miranker and Lininger [92] proposed a completely different approach to parallelize the numerical procedure for solving ODEs: A parallelization across the method using a predictor-corrector scheme. Miranker and Lininger gave a general formulation of a class of parallel integration methods of the linear multistep type and presented a comprehensive investigation of different pairs of predictors and correctors. In the retrospective, their ideas can also be interpreted as one very early representative of the class of multigrid methods, that were introduced later in the middle of the 1980s by Hackbusch [44].

Parallel methods for ODEs are *not* restricted to a parallelization in time. Many different approaches exist. According to Burrage [19], they can be classified into three categories: parallelism across the method, parallelism across the system and parallelism in time.

1. **Parallelism across the method:** The principle of a parallelism across the method is the parallelization of independent function evaluations of the solution scheme. This is in general possible with multistage and multistep methods. Efficient methods of this category are typically indirect methods, like prediction-correction techniques as presented in Miranker and Liniger [92]. The authors further proposed parallel Runge-Kutta methods. This can be achieved with individual stages computed on dedicated processors, c.f. van der Houwen and Sommeijer [125]. One such example is the multi-implicit Runge–Kutta solver ParSODES by Bendtsen [12], which yielded a speed-up between 3 and 5 compared to a state-of-the-art sequential code. The potential level of parallelism across the method is of course limited to the number of computing stages. Unfortunately, the speed-up of the computing stages does not compare well with those of Jacobian computing and the communication takes more time than the computing of the stages, c.f. Guibert and Tromeur-Dervout [43].

2. **Parallelism across the system:** A parallelization across the system can be achieved through a partitioning of the right-hand-side function itself. The problem is decomposed into subproblems, that are solved by different step-size strategies in parallel. To allow a synchronization between the decoupled computations, the largest step-size used must be an integral multiple of all smaller step-sizes. The most popular schemes of this class have been presented under the name *waveform relaxation.* Those methods have been invented for circuit simulation [63], where the right-hand-side function can easily be split according to sub-circuits. Such methods in practice are difficult to use, since they require a high level of knowledge of the system to reorder and split the equations.

3. **Parallelism in time:** Methods of this category all adopt a partitioning of the integration interval into consecutive sub-intervals, which are solved concurrently. One essential issue within is to find respective seed values on each of the sub-intervals. Most of those methods are related to multiple shooting techniques (e.g. [54, 55]) and multigrid ap-

proaches (c.f. Hackbusch [44]). Also, the waveform relaxation can be interpreted as a parallelization across time, as time-parallelism is found in the quadrature method.

One crucial restriction in the applicability of the first two categories is the fact, that time-stepping algorithms are *fundamentally non-scalable* [126]: For one moment in time, all processors are active on the same time level. Using all possible ways of a parallelization across the method and the system, the maximal speed-up through parallelization achievable during an interval is limited. Truly scalable parallel algorithms therefore must also account for a parallelization in time.

In 2001, Lions, Maday and Turinici proposed a new parallel-in-time algorithm for the solution of partial differential equations (PDEs): The *parareal algorithm* (c.f. [7, 10, 74]). The global time evolution problem is broken into a series of independent evolution problems on smaller time-slabs nested in an outer loop. Initial inaccurate predictions serve as seeding values on each time-slab. Iteratively, those initial values are update using a predictor-corrector scheme with a fine, but parallel predictor and a sequential, but coarse predictor. The parareal algorithm can both be interpreted as a multiple shooting technique and a kind of multigrid approach (see Gander and Vandewalle [37]).

Within the last years, the algorithm has been tested to a wide range of problems: linear and non-linear parabolic problems [117], non-differential equations [10], stochastic ODEs [9], the hyperbolic acoustic-advection equation [107], the incompressible Navier-Stokes equations [34], and many more. For an overview, see for example Nielsen [96]. Its stability and convergence have been examined extensively for example by Bal and Maday [10], Gander and Vandewalle [37, 38] and Maday et al. [84]. One of its advantages over most other parallel-in-time algorithms is the straightforward implementation. Further, it enables a high level of flexibility in the choice of the coarse and fine integration scheme. In the course of this work, it allows to employ reduced models as coarse integration schemes.

3.2 The Parareal Algorithm

Consider a general time depending problem of the form

$$\frac{\partial u}{\partial t} \quad + \quad Au = 0 \tag{3.2.1}$$

with an operator A from one Hilbert space to another and $t \in [T_0,\ T_{\text{end}}]$ and $u(T_0) = u_0$. Assume a numerical operator $\mathcal{F}_{\Delta t}$, that operates on some given initial state and approximates the solution to Eq. (3.2.1) at some later time $t_{n+1} = t_n + \Delta t$. This operator could for example be an Euler or a Runge-Kutta scheme, that internally may require smaller sub-time-steps of size $\delta t < \Delta t$ for reasons of accuracy or stability. From now on, this operator will be denoted as the *fine propagator*. In order to apply this fine propagator to solve Eq. (3.2.1), we decompose the time interval $[T_0,\ T_{\text{end}}]$ into N sequential sub-intervals, that is

$$t_0 = T_0 < t_1 < \cdots < t_n = T_0 + n\Delta t < t_{n+1} < t_N = T_{\text{end}}.$$

The numerical solution $\hat{U}_n$ to Eq. (3.2.1) at time t_n using the fine propagator is then defined as

$$\hat{U}_n := \mathcal{F}_{\Delta t}\left(t_{n-1},\ \hat{U}_{n-1}\right),$$

with $\hat{U}_0 = u_0$. Analogically, we define a second, c*oarse propagator* $\mathcal{G}_{\Delta t}$, that also maps a given state of the solution at time t_n to the solution of Eq. (3.2.1) at some later time t_{n+1}, but uses a coarser internal time-step δT than $\mathcal{F}_{\Delta t}$, i.e. $\delta t < \delta T \leq \Delta t$. The numerical solution $\tilde{U}_n$ at time t_n using the coarse propagator hence is

$$\tilde{U}_n := \mathcal{G}_{\Delta t}(t_{n-1},\ \tilde{U}_{n-1}).$$

Applying the coarse propagator instead of the fine propagator to solve Eq. (3.2.1) over N sequential sub-intervals will return a less accurate solution $\tilde{U}_n$ than $\hat{U}_n$, but at significantly less computational efforts. Sequentially stepping through all N sub-intervals and solving

$$\tilde{U}_n = \mathcal{G}_{\Delta t}(t_{n-1},\ \tilde{U}_{n-1}),$$

with $\tilde{U}_0 = u_0$ will return a less accurate trajectory $\tilde{U}_{n=0:N}$ than $\hat{U}_{n=0:N}$. The key idea of the parareal algorithm is to correct this coarse trajectory by means of a feedback mechanism, which is defined as

$$U_n = \mathcal{G}_{\Delta t}(t_{n-1},\ U_{n-1}) + \mathcal{F}_{\Delta t}(t_{n-1},\ \tilde{U}_{n-1}) - \mathcal{G}_{\Delta t}(t_{n-1},\ \tilde{U}_{n-1}).$$

Since the correction does not lead to an overall accurate solution trajectory, the feedback process is repeated iteratively over $k = 1, \ldots, k_{\max}$. The resulting algorithm is the parareal algorithm,

$$U_n^k = \underbrace{\mathcal{G}_{\Delta t}(t_{n-1},\ U_{n-1}^k)}_{\text{sequential prediction}} + \underbrace{\mathcal{F}_{\Delta t}(t_{n-1},\ U_{n-1}^{k-1}) - \mathcal{G}_{\Delta t}(t_{n-1},\ U_{n-1}^{k-1})}_{\text{parallel correction}}, \tag{3.2.2}$$

with $U_0^0 = u_0$ and $U_n^0 = \mathcal{G}_{\Delta t}(U_{n-1}^0)$ and for all $n = 1, \ldots, N$. The prediction step sequentially steps through all time-slabs. Since both terms of the correction step do only depend on the solution at the previous iteration stage, they can be performed in parallel. A short draft of the parareal algorithm in its most simple form is presented in Algorithm 3.1.

One essential ingredient of the parareal algorithm is the fact, that the costly computations of the fine propagator can be parallelized, as only the first term in Eq. (3.2.2), the prediction step, has to be computed sequentially. The iteration is repeated until the solution U_n^k has converged to the solution of the fine propagator, as it would have resulted from a purely sequential application. The stopping criterion can for example be chosen as that the difference in the solution at final time T_{end} between two consecutive iterations falls below a threshold value. If one is interested in an accurate solution over the whole integration period, other stopping criteria are of course preferable. Besides the choice of the stopping criterion, a number of other parameters determine the efficiency of the parareal algorithm: the choice of the fine and the coarse propagators and the number of time-steps N. The fine propagator defines the reference solution, to which the parareal solution finally converges to. Its choice strongly depends on the scenario and is not universal. The choice of the coarse propagator is considerably free and only limited by physical and stability considerations: It can be for example be a different time-stepping scheme than $\mathcal{F}$, the same time-stepping scheme, but with different parameters, or even a completely different model. However, the choice strongly affects convergence rates and stability, consequently also the number of iterations $k_{\max}$ of the algorithm. This issue will be substantial in the following sections. The number of time-steps N defines the number of processes, that will be computed in parallel. The choice of this parameter is on one hand limited by the number of processors available. Further, it is also limited by the choice of the fine $\mathcal{F}_{\Delta t}$ and coarse $\mathcal{G}_{\Delta t}$ propagators, as their internally used time-step sizes δt and δT must satisfy $\delta t < \delta T \leq \Delta t = (T_{\text{end}} - T_0)/N$.

Algorithm 3.1 Pseudo code for the parareal algorithm.

```
for $i = 1$ to $N$ do
    ${U_n}^0 = \mathcal{G}_\Delta t(U^0_{n-1})$                ▷ Initialization (sequential)
end for
$k = 1$
repeat
    for $i = 1$ to $N$ do
        $D_i = \mathcal{F}_{\Delta t}({U_{n-1}}^k) - \mathcal{G}_{\Delta t}({U_{n-1}}^k)$     ▷ Correction (parallel)
    end for
    ${U_0}^{k+1} = u_0$
    for $i = 1$ to $N$ do
        $U_i^{k+1} = \mathcal{G}_{\Delta t}({U_{n-1}}^{k+1}) + D_i$      ▷ Prediction (sequential)
    end for
    $k = k + 1$
until convergence
```

Example For the sake of clarity, this section presents a short visualization of the parareal algorithm, applied to solve the following initial value problem:

$$\frac{\partial u}{\partial t} = \exp(2.5 \cdot t) \cdot (2.5 - 10 \cdot \sin(10 \cdot t)) =: f(t) \tag{3.2.3}$$

with $u(0) = 1$ and $t \in [0,\ 1.0]$. We decompose the time interval into $N = 10$ equidistant, sequential time-slabs of size $\Delta t = 0.1$, i.e. $t_n = n \cdot 0.1$ with $n \in \{0, 1, ..., 10\}$. For both the fine and the coarse propagator a Crank-Nicolson-scheme is adopted, with internally used time-step sizes $\delta t = 0.01$ for the fine and $\delta T = \Delta t = 0.1$ for the coarse integrator.

$$\begin{aligned}
\tilde{U}_n := \mathcal{F}_{\Delta t}(t_{n-1},\ U_{n-1}) &= U_{n-1} + \sum_{j=0}^{9} \left(\delta t \cdot \frac{f(t_{n-1} + j \cdot \delta t) + f(t_{n-1} + (j+1) \cdot \delta t)}{2} \right), \\
\hat{U}_n := \mathcal{G}_{\Delta t}(t_{n-1},\ U_{n-1}) &= U_{n-1} + \delta T \cdot \frac{f(t_{n-1}) + f(t_{n-1} + \delta T)}{2},
\end{aligned}$$

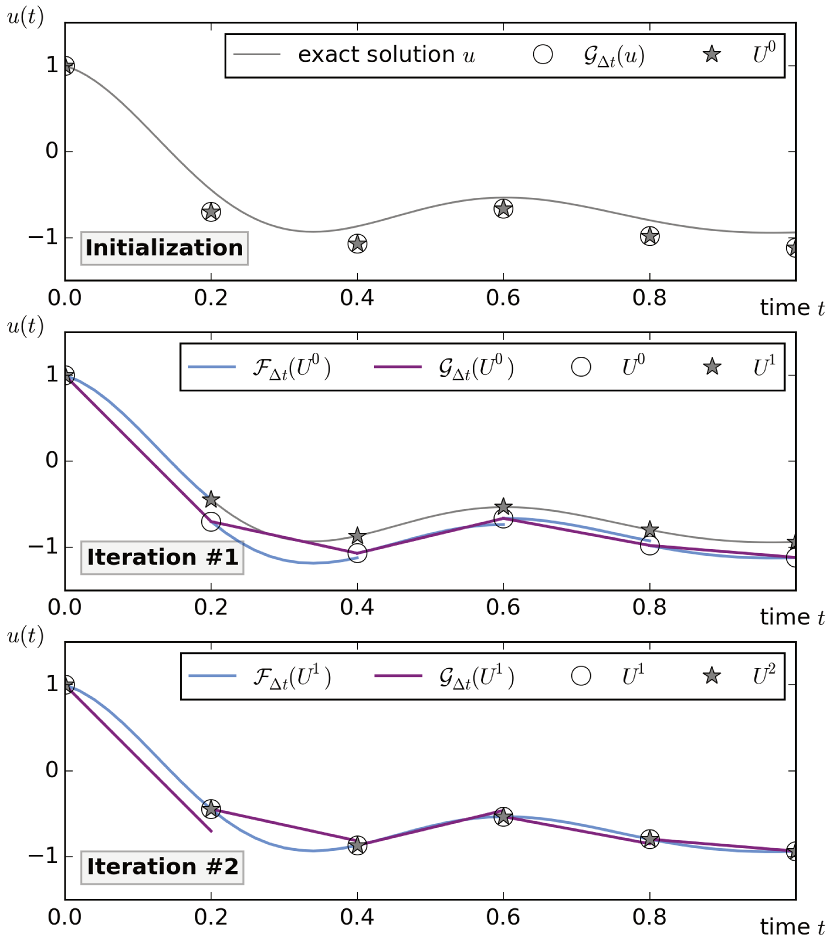


Figure 3.1: Parareal iterations in the solution of the initial value problem Eq. (3.2.3). Circles denote the parareal solution of the previous iteration. Stars denote the actual parareal solution at this iteration stage.

As an iteration breaking criterion we take the l_2-norm of the absolute difference in the solution trajectory between two subsequent iterations,

$$\text{Err} = \sqrt{\sum_{n=1:N} \left(U_n^k - U_n^{k-1}\right)^2} = \left\|U^k - U^{k-1}\right\|_{l_2}.$$

For simplicity, we will omit the explicitly written down time-dependency in the notation of the fine and the coarse propagator and write $\mathcal{F}_{\Delta t}(U_{n-1})$ instead of $\mathcal{F}_{\Delta t}(t_{n-1},\, U_{n-1})$. Figure 3.1 shows the initialization process and two iterations of the parareal algorithm. For clarity, the pure sequential solution has been added in gray in all pictures. The first figure shows the initialization process, whereat only the coarse propagator is applied in a purely sequential fashion from one to the next time-step t_n for all $n = 1, ..., 10$. The solutions $U^0_{0:N} = \mathcal{G}_{\Delta t}(u)$ define the initial values on each of the time slabs in the first iteration. The second plot shows the parareal solution after the first iteration U^1, marked as gray stars. Starting from its respective initial value $U^0_{0:N}$, both a fine (marked blue) and a coarse (marked purple) propagation are performed on each of the time-slabs in parallel. The difference between fine and coarse prediction, $\mathcal{F}_{\Delta t}(U_n^0) - \mathcal{G}_{\Delta t}(U_n^0)$, defines the parallel prediction. Following, the predictions on each time-slab are corrected by means of a sequential coarse update, $U_{n+1}^1 = \mathcal{G}_{\Delta t}(U_n^1) + \mathcal{F}_{\Delta t}(U_n^0) - \mathcal{G}_{\Delta t}(U_n^0)$ for all $n \in [1, N]$. The resulting parareal solution after the first iteration is marked as a gray star again. Notice, that the parareal solution at the end of the first time-slab now equates the solution of the fine propagator, since $U_0^k = u(0)$ for all k, hence $U_1^1 = \mathcal{G}_{\Delta t}(U_0^1) + \mathcal{F}_{\Delta t}(U_0^0) - \mathcal{G}_{\Delta t}(U_0^0) = \mathcal{F}_{\Delta t}(U_0^0)$. The second iteration, depicted in the last plot, starts with the initial values $U_{0:N}^1$ on each time slab. Again, fine and coarse predictions are performed, followed by a sequential coarse update. Now, the difference between the result of the second iteration U^2 and the one after the first iteration is very small, the iteration breaks and the parareal solution is found.

3.2.1 Convergence

The parareal algorithm is an iterative scheme, that converges towards the solution of a purely sequential application of the fine propagator $\mathcal{F}_{\Delta t}$. After $k = N$ iterations, the solution at final time $U_N^{k=N}$ equates the solution of a purely sequential application of $\mathcal{F}_{\Delta t}$. A proof can be found in the Appendix .

A very first convergence analysis has been provided in the introductive parareal paper by Lions et al. [74] in 2001. The authors analyzed the accuracy of the approximate solution U_n^k for a parareal scheme with an explicit Euler scheme as a coarse propagator $\mathcal{G}_{\Delta t}$, a sufficiently fine propagator $\mathcal{F}_{\Delta t}$ (i.e. the solution is a sufficiently accurate approximation of the exact solution u), applied to a linear, scalar model problem

$$\frac{\partial u}{\partial t} = \lambda u, \quad u(0) = u_0, \quad u \in \mathbb{R}, \quad t \in [0,\ T], \quad \lambda \in \mathbb{C}, \tag{3.2.4}$$

in the $k-$th iteration with varying time-step sizes Δt.

Theorem 1. *Let $\Delta t = \frac{T}{N}$, $t_n = n\Delta t$ for $n = 0, 1..., N$. Let $\mathcal{F}_{\Delta t}(U_n^k)$ be the exact solution at time t_n of the model problem (3.2.4) with $\lambda \in \mathbb{R}$. Let $\mathcal{G}_{\Delta t}$ be a backward Euler approximation with time-step size Δt. Then,*

$$\max_{1 \leq n \leq N} |u(t_n) - U_n^k| \leq C_k \Delta t^{k+1}.$$

For a proof, see Lions et al. [74].

Apparently, in the special case of using a backward Euler scheme as a coarse propagator, the algorithm behaves like $\mathcal{O}(\Delta t^k)$ for a fixed iteration step k and varying time-step sizes Δt. This is remarkable, since the backward Euler scheme itself is only of first order.

Later, the estimate was extended to more general integration schemes with arbitrary coarse propagators, e.g. by Bal and Maday [10]:

Theorem 2. *Assume a bounded function u in $t \in [0,\ T]$, a coarse propagator $\mathcal{G}_{\Delta t}$, which is of order m and Lipschitz, and a fine propagator $\mathcal{F}_{\Delta t}$, that is sufficiently accurate, so that we* *can replace the fine propagator $\mathcal{F}_{\Delta t}$ by the analytic operator. Then, the order of accuracy of the parareal algorithm applied to a linear model problem at iteration k is $(m+1)k$,*

$$|u(t_n) - U_n^k| \leq C_k n^k (\lambda \Delta t)^{(m+1)k}$$

For a proof, we refer to Bal and Maday [10].

Again, the parareal algorithm turns any coarse propagator of order m into a higher order, $\mathcal{O}(\Delta t^{(m+1)k-1})$ method. Notice, that the constant C_k both in Theorem 1 and 2 varies with k. Both theorems therefore do not cover the convergence of the algorithm for an increasing number of iterations and a fixed time-step size Δt. Gander and Vandewalle [39] later extended the convergence analysis of the linear scalar problem to fixed Δt and varying k. They showed superlinear convergence in k when using parareal on bounded time intervals and linear convergence for unbounded intervals. Gander and Hairer [36] further carried out a nonlinear convergence analysis, with the result, that the parareal algorithm converges superlinearly on any bounded time interval for nonlinear problems.

3.2.2 Stability

We have met stability already in the context of ODE solvers, see Sec. 2.3.1. A stable, numerical scheme does not magnify numerical errors, that are generated during the solution process. In this section, we present results for a stability analysis of the parareal algorithm as presented by Maday et al. [84] for an application to a linear system of ODEs with constant coefficients. In the context of Sec. 2.3.1, we had defined the stability functions $R(z)$ as the numerical solution to Dahlquist's test equation $y' = \lambda y, \quad y_0 = 1, \; z = \Delta t \lambda$ after one step. We assume, the stability function of the fine propagator $\mathcal{F}_{\Delta t}$ with internal step size δt is $r := r(\lambda \delta t)$ and define $\overline{r} := r(\lambda \delta t)^{\frac{\Delta t}{\delta t}}$. The stability function of the coarse propagator scheme $\mathcal{G}_{\Delta t}$ with internal step size δT is $R := R(\lambda \delta T)$, further, we define $\overline{R} := R(\lambda \delta t)^{\frac{\Delta t}{\delta T}}$.

Theorem 3 (Stability for $\lambda \in \mathbb{C}$: The general case)**.** *The stability of the parareal algorithm (3.2.2) applied to the autonomous ODE*

$$\frac{\partial y}{\partial t} = \lambda y, \quad y(0) = y_0, \quad \textit{with } \lambda \in \mathbb{C}$$

is guaranteed for a fixed number of iterations as long as $|\overline{r}| \leq 1$ and $|\overline{R}| \leq 1$.

For a proof, see Maday et al. [84].

This stability condition however is not very restrictive. The serial solution will be obtained after a maximum number of iterations with $k = N$, independent of instabilities arising in the parareal algorithm. The limit $k = N$ of course is not interesting in practice, as then no computational gain is possible. We adopt the concept of strong stability introduced by Maday et al. [84] and introduce two more restrictive stability theorems: One for a real and negative eigenvalue and one for a complex and negative eigenvalue. Proofs of both theorems can be found in Maday et al. [84].

Definition. The parareal algorithm inheres *strong stability*, if it is stable for all possible N and all numbers of iterations $1 \leq k \leq N$.

Theorem 4 (Strong stability for real, negative eigenvalues)**.** *The stability of the parareal algorithm (3.2.2) applied to the autonomous ODE*

$$\frac{\partial y}{\partial t} = \lambda y, \quad y(0) = y_0, \quad \textit{with } \lambda \in \mathbb{R} \textit{ and } \lambda < 0,$$

is guaranteed if $-1 \leq \overline{r}\ , \overline{R} \leq 1$. *The algorithm (3.2.2) further provides strong stability, if*

$$\frac{\overline{r}-1}{2} \quad \leq \quad \overline{R} \leq \frac{\overline{r}+1}{2}.$$

For a proof, see Maday et al. [84].

From theorem 4 it can be seen, that given an exact fine propagator with $\overline{r} \to 0$, the parareal scheme will be stable for any $|R| \leq \frac{1}{2}$, independent of the stiffness of the system. All L-stable schemes fulfill this requirement.

Theorem 5 (Strong stability for complex eigenvalues with negative real parts)**.** *The strong stability of the parareal algorithm (3.2.2) applied to the autonomous ODE*

$$\frac{\partial y}{\partial t} = \lambda y, \quad y(0) = y_0, \quad \textit{with } \lambda \in \mathbb{C} \textit{ and } Re(\lambda) < 0,$$

is guaranteed if

$$e^{x_R} \leq \frac{1}{2}\frac{1 - e^{2x_{\overline{r}}}}{1 - e^{x_{\overline{r}}}\cos\varepsilon},$$

with the complex stability functions $\overline{R} = e^{x_R}e^{i(\theta+\varepsilon)}$ *and* $\overline{r} = e^{x_{\overline{r}}}e^{i\theta}$, *with the real parts* $x_{\overline{r}}$ *and* $x_{\overline{R}}$ *and the imaginary part* θ *with a phase difference* ε *between* $\overline{r}$ *and* $\overline{R}$.

For a proof, see Maday et al. [84].

One important question in designing a parareal algorithm is the question, of how to choose the fine and the coarse propagator in order to guarantee a stable scheme. A general answer to this question can not be given. No parareal scheme has yet been found, that guarantees stability for all possible eigenvalues, all possible number of time-slabs N and all numbers of iterations k. Especially in the case of purely imaginary eigenvalues, stability is hard to guarantee. The parareal solution of hyperbolic problems and convection-dominated convection-diffusion problems, therefore is an ongoing challenge.

3.2.3 Complexity, Speed-Up and Efficiency

In this section, we will investigate the computational complexity of the parareal algorithm with an implementation as proposed in Alg. 3.1. To this end, we will compare the summed computing time over all processors and the absolute execution time of the parareal algorithm to those of a sequential implementation. The parareal algorithm is an iterative approach, designed in the attempt to parallelize parts of the solution procedure. The overall computational effort is always higher than for a sequential algorithm. Its benefit over a purely sequential algorithm is, that some parts of the code can be performed in parallel, leading to smaller absolute WCTs.

Regardless of their implementation, the computational complexity for both the fine and the coarse propagator depends on the number of time-steps N being used. We denote the computing times $\mathrm{CT}_{\mathcal{F}}$ and $\mathrm{CT}_{\mathcal{G}}$, as the times each of the integrator needs to perform one single time-step with time-step size δt or δT. The total computing time to propagate the solution from time t_n to $t_{n+1} = t_n + \Delta t$ using the fine propagator with time-steps δt therefore is given by $\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}}$, respectively $\frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}$ for the coarse integrator. We assume, that k_{conv} iterations are necessary, until the parareal algorithm has converged. The computational time for the sequential initialization using the coarse integrator is $\mathrm{CT}_{\mathrm{init}} = N\frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}$. As this is a sequential step, the respective WCT equates the computing time, $\mathrm{WCT}_{\mathrm{init}} = \mathrm{CT}_{\mathrm{init}}$. During each of the $k = 1, ..., k_{\mathrm{conv}}$ iterations, first the parallel correction step with an overall computing time of $\mathrm{CT}_{\mathrm{cor}} = N\left(\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}} + \frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}\right)$ has to be performed. This part can be performed in parallel distributed over N processors. Hence the total WCT for this step is $\mathrm{WCT}_{\mathrm{cor}} = \max\left(\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}},\ \frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}\right)$. Subsequently, a sequential prediction step using the coarse propagator has to be calculated. Both the computing time and the WCT again are $\mathrm{CT}_{\mathrm{pred}} = \mathrm{WCT}_{\mathrm{pred}} = N\frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}$. Finally, the accumulated computing time summed over all processor is,

$$\begin{aligned}\mathrm{CT}_{\mathrm{par}} &= \mathrm{CT}_{\mathrm{init}} + k_{\mathrm{conv}}\left(\mathrm{CT}_{\mathrm{corr}} + \mathrm{CT}_{\mathrm{pred}}\right)\\ &= N\frac{\Delta t}{\delta T}\left(1 + 2k_{\mathrm{conv}}\right)\mathrm{CT}_{\mathcal{G}} + k_{\mathrm{conv}}N\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}},\end{aligned}$$

while the overall parallel execution time is

$$\begin{aligned}\mathrm{WCT}_{\mathrm{par}} &= \mathrm{WCT}_{\mathrm{init}} + k_{\mathrm{conv}}\left(\mathrm{WCT}_{\mathrm{corr}} + \mathrm{WCT}_{\mathrm{pred}}\right)\\ &= N\frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}} + k_{\mathrm{conv}}\left(\max\left(\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}},\ \frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}\right) + N\frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}\right)\\ &= N\frac{\Delta t}{\delta T}\left(1 + k_{\mathrm{conv}}\right)\mathrm{CT}_{\mathcal{G}} + k_{\mathrm{conv}}\max\left(\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}},\ \frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}\right).\end{aligned} \tag{3.2.5}$$

We assume the computing time of the fine propagation scheme during t_n and t_{n+1} to take longer than the one for the coarse scheme, $\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}} > \frac{\Delta t}{\delta T}\mathrm{CT}_{\mathcal{G}}$. Hence we can estimate the parallel WCT as

$$\mathrm{WCT}_{\mathrm{par}} = N\frac{\Delta t}{\delta T}\left(1 + k_{\mathrm{conv}}\right)\mathrm{CT}_{\mathcal{G}} + k_{\mathrm{conv}}\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}}. \tag{3.2.6}$$

In order to estimate the potential speed-up compared to a purely sequential algorithm, we investigate the computing time using the fine propagator algorithm with time-step size δt: $\mathrm{WCT}_{\mathrm{seq}} = N\frac{\Delta t}{\delta t}\mathrm{WCT}_{\mathcal{F}} = \frac{T}{\delta t}\mathrm{WCT}_{\mathcal{F}}$. Communication between processors is further assumed to be negligible. The speed-up of the parallel algorithm over the sequential algorithm can then be estimated by the ratio of the sequential WCT time to the parallel WCT,

$$\begin{aligned} S &= \frac{\mathrm{WCT}_{\mathrm{seq}}}{\mathrm{WCT}_{\mathrm{par}}} \\ &= \frac{N\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}}}{N\frac{\Delta t}{\delta T}\left(1+k_{\mathrm{conv}}\right)\mathrm{CT}_{\mathcal{G}} + k_{\mathrm{conv}}\frac{\Delta t}{\delta t}\mathrm{CT}_{\mathcal{F}}} \\ &= \frac{N}{N\frac{\delta t}{\delta T}\frac{\mathrm{CT}_{\mathcal{G}}}{\mathrm{CT}_{\mathcal{F}}}\left(1+k_{\mathrm{conv}}\right) + k_{\mathrm{conv}}}. \end{aligned} \tag{3.2.7}$$

From this relation, we see, that in the limit of $\frac{\delta t}{\delta T}\frac{\mathrm{CT}_{\mathcal{G}}}{\mathrm{CT}_{\mathcal{F}}} \to 0$, i.e. the coarse propagator is a lot faster than the fine one, we get a speed-up of

$$S = \frac{N}{k_{\mathrm{conv}}}.$$

The number of parallel processes N limit the maximum potential speed-up. Increasing numbers of iterations instead deteriorate speed-up. In the hypothetic case of $N < k_{\mathrm{conv}}$, that parareal algorithm will be slower than a sequential algorithm. In practice, this case is irrelevant, since the correct solution is found at the latest at $N = k_{\mathrm{conv}}$, as will be shown in Sec. 3.2.1. In case of $k_{\mathrm{conv}} = 1$, the parallel algorithm distributed over N processors leads to a perfect speed-up. Further, Eq. (3.2.7) shows, that the speed-up on one processor

$$S_1 := \frac{1}{\frac{\delta t}{\delta T}\frac{\mathrm{CT}_{\mathcal{G}}}{\mathrm{CT}_{\mathcal{F}}}\left(1+k_{\mathrm{conv}}\right) + k_{\mathrm{conv}}} \leq 1$$

is always smaller equal one, since it must be $k_{\mathrm{conv}} \geq 1$. The parareal algorithm therefore makes no sense, if applied sequentially.

The parallel efficiency can be estimated as the ratio of speed-up S_p for p processors to the number of processors, assuming $N = p$,

$$E := \frac{S_p}{p} = \frac{1}{p\frac{\delta t}{\delta T}\frac{\mathrm{CT}_{\mathcal{G}}}{\mathrm{CT}_{\mathcal{F}}}\left(1+k_{\mathrm{conv}}\right) + k_{\mathrm{conv}}}.$$

Again in the limit of $\frac{\delta t}{\delta T}\frac{\mathrm{CT}_{\mathcal{G}}}{\mathrm{CT}_{\mathcal{F}}} \to 0$, we see, that k_{conv} poses an upper bound on the parallel efficiency, as $\lim_{\frac{\delta t}{\delta T}\frac{\mathrm{CT}_{\mathcal{G}}}{\mathrm{CT}_{\mathcal{F}}} \to 0} E = \frac{1}{k_{\mathrm{conv}}}$.

From the results of this section, we see, that given a fixed number of iterations k_{conv}, the best speed-up can be achieved, when using a very fast coarse propagator compared to the fine propagator. In the following, we will however see, that the choice of the coarse propagator strongly affects convergence and stability. In terms, a fast and less accurate solution may induce needs for higher numbers of iterations. Then of course, the speed-up decreases again. Finding a suitable coarse propagator in practice turns out to be the most challenging task in designing an efficient parareal scheme.

3.2.4 Distribution of Parallel Work

A potential gain in speed-up and efficiency can also be achieved from an asynchronous distribution of the parallel work. Figure 3.2 visualizes a simple implementation of the parareal algorithm as proposed in the pseudo code Alg. 3.1. At initial time, a sequential coarse prediction is calculated over all N time-slabs. Following, in each iteration N independent initial value problems are solved in parallel, followed again by a sequential, coarse prediction over all N time-slabs. As it can be seen in Fig. 3.2, some parts of the algorithm are purely sequential. We therefore have a natural limit for the potential speed-up, that can be achieved, as it has been depicted in the previous section.

It is obvious, that a fine propagation $\mathcal{F}_{\Delta t} U_n^k$ can start, as soon as its respective initial value, the parareal solution U_n^k, has been found. This enables a simple work scheduling approach. Several authors recognized the potential of a distribution of work to improve the performance of the parareal algorithm: Minion [90] presented a *pipelined parareal* method, where a processor computes a step in the algorithm as soon as possible. A similar approach has also been proposed by Berry et al. [13] under the name *event-driven distribution*. Aubanel [6] further presented a detailed study of the scheduling of tasks in the parareal algorithm. He proposed two implementations: One manager-worker paradigm with overlapping sequential and parallel phases and one task-based, completely distributed parareal implementation. Both implementations resulted in a significant improvement of the potential speed-up. The manager-worker paradigm however turned out to be extremely memory-consumptive when using many parallel processes.

We briefly outline the fully distributed parareal implementation as proposed by Aubanel [6]. A timing chart for the scheduling of work within the first three iterations can be seen in Fig. 3.3. As soon as the coarse, sequential predictor $\mathcal{G}(U_{n-1}^k)$ has propagated the solution over $[t_{n-1},\ t_n]$ an initial value for the succeeding time-slab at time $[t_n,\ t_{n+1}]$ is found and the processor associated with this time-slab starts the calculation of the correction $\mathcal{F}(U_n^k) - \mathcal{G}(U_n^k)$ over $[t_n,\ t_{n+1}]$.

The speed-up for a distributed parareal implementation as proposed by Aubanel [6] can be given as

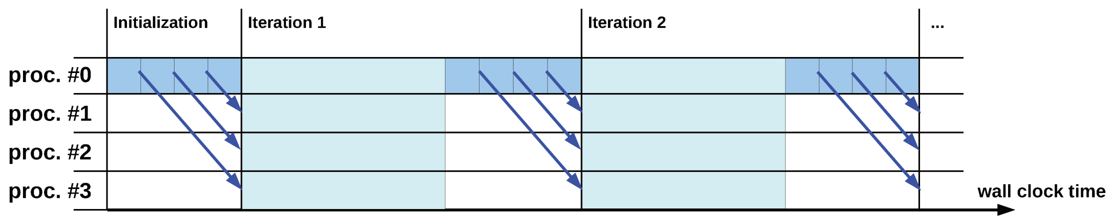


Figure 3.2: Timing chart for the first two iterations for a simple parareal implementation as proposed in Alg. 3.1. The x-axis denotes the WCT. Blue bars indicate computing effort due to sequential coarse prediction steps using $\mathcal{G}$. Turquoise bars indicate computing effort due to parallel correction steps using $\mathcal{F} - \mathcal{G}$. White space indicates idle time. Blue arrows mark the communication of initial conditions for each fine propagator call.

$$S_{\text{dist}} = \frac{N}{N \frac{\delta t}{\delta T} \frac{\text{CT}_{\mathcal{G}}}{\text{CT}_{\mathcal{F}}} + k_{\text{conv}} \left(\frac{\delta t}{\delta T} \frac{\text{CT}_{\mathcal{G}}}{\text{CT}_{\mathcal{F}}} + 1 \right)}. \tag{3.2.8}$$

whereat communication and correction time are neglected again. We recall the speed-up for a simple parareal implementation as it has been introduced in Eq. (3.2.7),

$$S_{\text{simple}} = \frac{N}{N \frac{\delta t}{\delta T} (1 + k_{\text{conv}}) \frac{\text{CT}_{\mathcal{G}}}{\text{CT}_{\mathcal{F}}} + k_{\text{conv}}}. \tag{3.2.9}$$

We consider the limit of $N \to \infty$, for both speed-up estimates,

$$\begin{aligned} \lim_{N \to \infty} S_{\text{simple}} &= \frac{\delta T}{\delta t} \frac{\text{CT}_{\mathcal{F}}}{\text{CT}_{\mathcal{G}}} \frac{1}{(1 + k_{\text{conv}})}, \\ \lim_{N \to \infty} S_{\text{dist}} &= \frac{\delta T}{\delta t} \frac{\text{CT}_{\mathcal{F}}}{\text{CT}_{\mathcal{G}}}. \end{aligned}$$

It can easily be seen, that the speed-up of the distributed parareal is only limited by the relation between the fine and the coarse propagator speed, whereas it is also limited by the number of iterations in the simple parareal implementation. For an infinite number of parallel processes N, we can therefore expect a speed-up gain of a factor of $1/(1 + k_{\text{conv}})$.

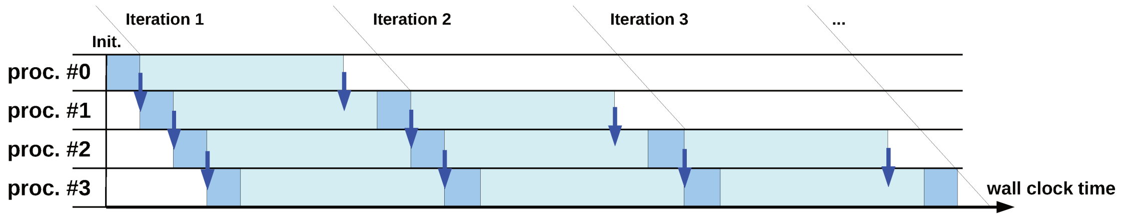


Figure 3.3: Timing chart for the first three iterations for a task-based, fully distributed parareal implementation as proposed by Aubanel [6]. The x-axis denotes the WCT. Blue bars indicate computing effort due to sequential coarse prediction steps using $\mathcal{G}$. Turquoise bars indicate computing effort due to parallel correction steps using $\mathcal{F} - \mathcal{G}$. White space indicates idle time. Blue arrows mark the communication of initial conditions for each fine propagator call.

3.2.5 Properties of the Parareal Algorithm

In the previous sections, we have highlighted the parareal algorithm under the different aspects of its computational complexity, speed-up, parallel efficiency, convergence and stability. We have seen some fundamental properties:

1. **Complexity, Speed-Up and Efficiency**
   - The overall complexity of the parareal algorithm is higher than for the sequential algorithm, hence for the speed-up of a sequential execution holds $S_1 \leq 1$.
   - For a very fast coarse propagator, the parallel efficiency scales as $\frac{1}{k_{\text{conv}}}$. A perfect speed-up is then achieved for $k_{\text{conv}} = 1$.
2. **Convergence**
   - The parareal algorithm converges with the order $m(k+1)$, with k the number of iterations and m the order of the coarse propagator.
   - The parareal solution converges towards the solution of the fine propagator.
   - The solution of the parareal algorithm equates the solution of the fine propagator at a maximum number of iterations, equal to the number of intervals N.
3. **Stability**
   - No parareal scheme is known, that guarantees stability for all eigenvalues.
   - Stability depends on the choice of the fine and the coarse propagator:
     - For real eigenvalues, stability is guaranteed, for any choice of stable coarse and fine propagators.
     - For real, negative eigenvalues, stability is guaranteed, if both the coarse and the fine propagator are L-stable.
     - For complex, negative eigenvalues, stability is hard to guarantee.

For an efficient parareal implementation, the choices of the following parameters have to be weighed up carefully:

- An **iteration stopping criterion**.
- The **number of time-steps** N, which defines the level of parallelization.
- The **fine propagator** $\mathcal{F}$, which defines the solution the algorithm finally converges.
- The **coarse propagator** $\mathcal{G}$, which affects convergence and stability, but not the accuracy of the end result. An ideal speed-up for a fixed number of iterations can be achieved with a very fast coarse propagator compared to the fine propagator. A very fast, but too coarse propagator will in terms result in an increase in the number of iterations, which deteriorates speed-up. The coarse propagator therefore is the **key ingredient of an efficient parareal scheme.**

3.3 Application to Chemical Transport Models

So far, we have not taken into account the different potential fields of application for a parallel-in-time calculation of the atmospheric chemical kinetics within a CTM. We reconsider the operator splitting approach, as introduced in Sec. 2.1.2.1. Its basic idea was a decoupling of advection, diffusion, reaction and further processes. For the chemistry this means, that during a splitting interval Δt_{split} chemistry is reacting in an isolated box, since no external disturbances caused by advection, diffusion or other processes have to be taken into account. Photolysis is typically updated only at the beginning of each splitting interval and then considered to be constant. Consequently, we can implement a parallelization in time either *internally* (within each splitting interval) or *externally* (across multiple splitting intervals). A confrontation of both approaches is depicted in Fig. 3.1.

An internal parallelization equates a parallelization within one single splitting interval. For this, a splitting interval $[t_n,\ t_n + \Delta t_{\text{split}}]$ is decomposed into N sub-intervals, that are being solved in parallel. The choice of the number of parallel intervals N is hereby arbitrary.

An external parallelization equates a parallel solution of N subsequent initial value problems over $[t_n,\ t_n+\Delta t_{\text{split}}]$ and $n = 1, ..., N$ in a parareal fashion. At the interfaces between two splitting intervals, we need to consider external effects, such as changing photolysis, advection and diffusion.

Now let us further consider a CTM, which is embedded in a compound AQM, that also accounts for a meteorological model. Then, we also need to take into account the coupling between CTM and meteorology. Typically, both are calculated independently from each other over certain coupling time intervals $\Delta t_{\text{coupl.}}$. Data is exchanged only at the end of these intervals. Depending on the level of coupling, data is either exchanged in both directions (*online coupling*) or from meteorology to chemistry only (*offline coupling*). We assume, that the size of the coupling intervals of the CTM and the meteorological model is a multiple of the size of the operator splitting time interval, that is being used internally within the CTM: $\Delta t_{\text{coupl.}} = M \cdot \Delta t_{\text{split}}$, whereat M again depends on the level of coupling between meteorology and chemistry.

For both online and offline coupled AQMs, the internal parallelization allows for a straightforward application. The application of an external parallelization within an online coupled AQM is hindered by the circumstance, that chemistry and meteorology show a two-way interaction. An iterative approximation procedure for the chemistry across L subsequent coupling intervals in fact also requires an iterative approximation of the meteorology. This in turn means, that one would either have to parallelize the whole dynamics of the meteorological

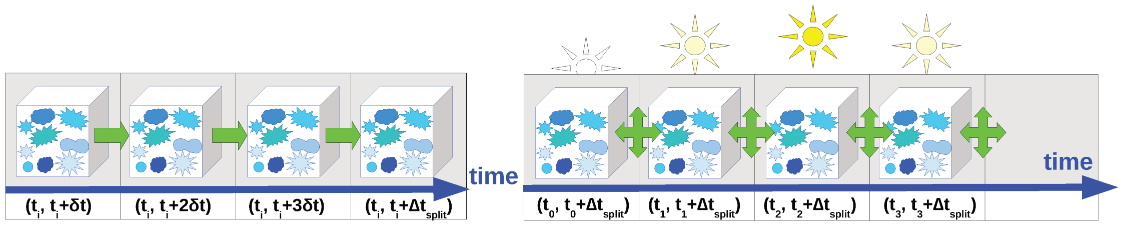

time
$(t_i,\ t_i+\delta t)$
$(t_i,\ t_i+2\delta t)$
$(t_i,\ t_i+3\delta t)$
$(t_i,\ t_i+\Delta t_{split})$
time
$(t_0,\ t_0+\Delta t_{split})$
$(t_1,\ t_1+\Delta t_{split})$
$(t_2,\ t_2+\Delta t_{split})$
$(t_3,\ t_3+\Delta t_{split})$

Figure 3.1: Internal vs. external parallelization.

model within the AQM in a parareal fashion or to decrease the level of couplings between meteorology and chemistry. An alternative would be to restrict the in-time-parallelization to a parallelization within one coupling interval $\Delta t_{\text{coupl.}}$ itself. Instead of sequentially solving M splitting intervals of size Δt_{split}, one could solve them in parallel over bigger intervals of size $M \cdot \Delta t_{\text{split.}}$ with $M > 1$, while adding an outer iteration. In this context, the degree of coupling between meteorology and chemistry would be unaffected.

From the parareal perspective, the question is, if we parallelize one single stiff problem in time or multiple subsequent stiff problems. In practice, this poses different requirements and restrictions on the decomposition into time-slabs and the choice of a suitable coarse propagator.

3.4 Parareal for Stiff Problems

Applying the parareal algorithm to chemical kinetics theoretically is straightforward: We choose a fine and a coarse propagator which both are stable for the problem we want to solve, decompose time into N time-slabs, define a stopping criterion and there we go. Unfortunately, things are a lot more complicated in practice. The reason again is called: Stiffness. Over the last years, different approaches have developed to facilitate parareal approaches to solve stiff problems. The most popular ones will be outlined in this section.

In advance, we start with some naive considerations to highlight the key challenges. In Sec. 3.2.2 we have seen, that a stable parareal scheme can be constructed even for very stiff systems, as long as both the fine and the coarse propagator are L-stable. The original parareal method has been designed to use classical single-step numerical methods with fixed time-step sizes for both the fine and the coarse propagator. As described in Sec. 2.3, adaptive time-stepping schemes should be used for stiff problems for reasons of stability. Now let us consider a parareal implementation with a partitioning of the time domain $[0,\ T]$ into N equidistant time-slabs, but using adaptive time-stepping schemes $\mathcal{F}$ and $\mathcal{G}$. An adaptive time-stepping scheme typically starts with an initial time-step size, say $h_0 > 0$. During the solution procedure, its time-step size is being adapted on the base of local error estimators. For many adaptive schemes, the adaptation process is restricted by upper and lower limits, $h_{\min} \leq h_{\text{new}} \leq h_{\max}$, and/or a distinct step size increase ratio $\phi_{\min} \leq h_{\text{new}}/h_{\text{old}} \leq \phi_{\max}$, see for example Sec. 2.3.3. Using an adaptive time-stepping scheme then poses two major challenges:

1. **Computational overhead in the sequential initialization:** The sequential initialization process consists of one coarse propagator call per time-slab, each starting with an initial time-step size h_0. In total, N coarse propagator starts are necessary on a time domain $[0,\ T]$ decomposed into N equidistant time-slabs. A small initial time-step size will be necessary on time-slabs where the solution is stiff. In non-stiff regions, the same initial time-step size is unnecessarily small. This could be omitted when using different time-step sizes on each interval. This requires an a priori knowledge of the solution's behavior, which is not available in general. To reach the end time on each interval $[t_n,\ t_{n+1}]$, the integrator will further have to perform additional adaptation steps. The

cumulative number of time-steps will therefore be higher, than when directly integrating from $[0,\ T]$ without restarting every t_n with $n = 0,\ ...,\ N$.

2. **Load-Imbalances during the parallel prediction:** During the parallel prediction step, both the fine and the coarse integrator will automatically adapt their internally used time-step size h to the solution's behavior. If the problem has stiff characteristics, the time-step size will decrease. On a time-slab, on which the problem is stiff, a lot more time-steps will therefore be needed to approximate the solution, both using $\mathcal{F}$ and $\mathcal{G}$. This will lead to load-imbalances over all time-slabs during the parallel prediction process. On a non-stiff time-slab, the integrator will start with the small initial time-step size $h = h_0$ again. In the next few steps, the time-step size will iteratively be increased until the step-size is just small enough to allow the time-stepping scheme to give a good approximation to the solution. All the aforegoing iterations may have been cut down on starting with a bigger initial time-step size. However, there may be cases again, where the update step has disturbed the initial solution on one time-slab, so that stiffness is really present. Then, small initial values will be inevitable again.

One possible solution to constrain the computational overhead in the sequential initialization is a direct integration from $[0,\ T]$, followed by an interpolation of initial values on every time-slab using the solution values at the internally adapted time-steps during the direct integration. Especially for nonlinear problems, this is an additional source of error. A load-balancing can potentially be achieved by a non-uniform decomposition of the time domain. If this decomposition is further suggested by the coarse integrator, i.e. defined by the internally used time-step sizes, the sequential coarse initialization can be performed directly. An interpolation will then not be necessary anymore. This way, $N - 1$ integrator restarts can be omitted.

In the following section, we see further variants, that have been designed to cope with stiff ODEs: reduced model and micro/macro parareal, hybrid parareal, extrapolated parareal and IMEX parareal. It has to be noted, that up to now, no general consensus exists on efficient parareal implementations for the solution of stiff ODEs.

3.4.1 Adaptive Parareal

In 2007, Guibert and Tromeur-Dervout [43] introduced an adaptive parareal approach, similar to the one explained above, for the solution of stiff systems of ODEs and differential algebraic equations (DAEs). The authors introduce adaptivity within the parareal algorithm for the time-stepping, the number of time-slabs and the time decomposition. Both as a coarse and a fine integrator, an adaptive Rosenbrock of order three with an embedded second order error estimator, was chosen. For both integrators, different tolerances were used. The number of time-slabs and the decomposition of the time interval was based on the time-steps chosen by the coarse integrator at the sequential initialization process. Notice, that the time-slabs now were not equidistant anymore. For the Lotka-Volterra problem, a pair of moderately stiff nonlinear ODEs, their approach turned out to be promising: For a decomposition of the total integration period $[0,\ T]$ into 1168 non-equidistant time-slabs, convergence was reached after between 2 and 7 iterations, depending on the choice of the relative tolerances for $\mathcal{F}$

and $\mathcal{G}$. However, for this number of time-slabs, a comparable convergence speed was also achieved on a decomposition into 1168 equidistant time-slabs, again using adaptive $\mathcal{F}$ and $\mathcal{G}$. For this example, no benefit can be seen from an adaptation of the time decomposition into non-equidistant time-slabs.

In 2012, Nielsen [96] further applied an adaptive parareal algorithm to another nonlinear, stiff ODE system: The Van der Pol oscillator. On each of the N equidistant time-slabs of size Δt, an adaptive time-stepping scheme was used as both the fine and the coarse propagator. Both schemes are an ESDIRK23 scheme, a diagonally implicit Runge-Kutta scheme of order two with an embedded error estimator of order three. The only difference between $\mathcal{F}_{\Delta t}$ and $\mathcal{G}_{\Delta t}$ again existed in the choice of the relative tolerances. The results showed, that the convergence behavior of the parareal solution for growing numbers of iterations is less smooth than for non-stiff applications. For too coarse tolerances for $\mathcal{G}$, the error in the final parareal solution was even shown to grow with increasing numbers of iterations in some cases. Also the predicted efficiency and speed-up follow a less smooth behavior. The author concluded a speed-up to be limited to a factor of 4. This was blamed to the decrease of the size of the time-slabs Δt with increasing number of time-slabs N. A decrease in Δt was assumed to hinder the efficiency of the adaptive time-stepping control, whose time-stepping mechanism then is forced to use an upper limit of Δt.

3.4.2 Reduced Model Parareal

The first combined approach of model reduction and parareal has been presented by Maday in 2007 [83] for the solution of linear chemical kinetic systems. As a coarse propagator, Maday chose a reduced model to the full kinetics, that governs the main features of the evolution at equilibrium. The reduced model equates the full model, with fast timescales assumed to be in a quasi-stationary state. Fast timescales are associated with very large, negative eigenvalues. As they are assumed to equilibrate rapidly, they are not necessary to describe the long-term behavior of the system. Instead, the full system is propagated over only a very short interval to account for the fast timescales. Then, they are removed from the system of ODEs and expressed through algebraic relations of the slow variables. This way, both the system's stiffness and its size decrease, which facilitates using bigger time-steps. Fast species are recovered at the end of the integration period. For the application to a stiff and linear kinetic system, he has shown his approach to maintain the convergence performance as for a classical parareal approach, but with a significant reduction in the total CPU time. Its success however strongly depends on the availability of a reduced model, which is on hand close enough to the fine approximation of the solution, but on the other hand a lot faster than the fine propagator.

A similar approach has later been adopted by Blouza et al. [14] and applied to a nonlinear ozone production system. The authors utilized the well-known quasi-steady state method (QSS), that will be described in more detail in Sec. 5.3. The basic idea again is a decoupling of fast and slow timescales, whereat fast timescales are eliminated from the ODE system and expressed through algebraic relations. The coarse solver again started with the full system. After 2.75 h physical time, fast species were eliminated from the system, leading to a non-stiff

ODE. The parareal computation on 20 processors was 4 times faster than the sequential fine integration. Again, fixed time-stepping methods are used for both the coarse and the fine integrator.

Recently, a new approach for the parareal solution of singularly perturbed ODEs has been proposed by Samaey et al. [62], *a micro-macro parareal algorithm.* Its key idea again is to use different models on microscopic and macroscopic scales, with a reduced model constructed on the base of an elimination of fast timescales. Different than in [83] and [14], the macroscopic coarse model does not contain all degrees of freedom. As their dimensions do not coincide anymore, special care has to be taken in the coupling of the macroscopic and the microscopic model. To this end, the authors of [62] proposed two new algorithms and contrasted them with the coupling of the microscopic and macroscopic levels in [83] and [14]. They showed those differences to affect the computational complexity, their potential of a generalization and the convergence behavior.

3.4.3 Other Approaches

Extrapolated Parareal For very stiff problems, such as the Oregonator problem, Guibert and Tromeur-Dervout [43] showed their adaptive approach to fail. Thereupon, they proposed an adaptive parallel extrapolation method: Assume a decomposition of $[0,\ T]$ into N time-slabs. On the first time-slab, an extrapolation operator is constructed, with its coefficients calculated from different grid levels on some control point values within the first time-slab. The extrapolation operator then is broadcast to the other subdomains, where extrapolations are performed. Following, on each time-slab the fine solvers are initialized with the extrapolated solutions. In the subsequent $k-$th iteration with $k = 2, ..., \ k_{\max}$, an improved extrapolation operator is constructed using the results from the 1st until the $(k-1)$-th time-slabs, which are now exact in the $(k-1)$-th iteration. This approach has shown to reduce the computational costs for the coarse initialization on the time-slabs. A similar approach has later also been proposed by Wu and Huang [139] for the solution of (non-stiff) nonlinear ODEs and PDEs under the name *Parareal-Richardson* algorithm and as a combination of the original parareal algorithm with the Richardson extrapolation. The authors showed, that the accuracy of the numerical solution of the new algorithm was higher than that of the original parareal algorithm.

Hybrid Parareal Spectral Deferred Corrections Method In 2008, Minion and Williams [91] proposed a parareal approach with decreased computational costs associated with the fine propagator. They introduced a new class of iterative time parallel methods for the solution of ODEs, based on a parallel variation of the method of Spectral Deferred Corrections (SDC, c.f. [31]). Deferred correction methods iteratively construct high-order, stiffly-stable time integrators for ODEs on the base of low-order time-stepping schemes. The basic principle is to increase the accuracy of the low order scheme by performing a series of correction sweeps. SDC methods use spectral integration on Gaussian quadrature nodes for the construction of the corrections. Minion and Williams [91] showed, that SDC has strong similarities with the parareal algorithm. Following, a hybrid strategy was constructed, that combines both features of parareal and the SDC method. Instead of the costly solving of the subproblems on each time-slab during each iteration with the fine propagator, an iterative ODE method is used based on deferred corrections. In a subsequent investigation by Minion [90], the *hybrid parareal spectral deffered corrections method*, was examined in more detail. One benefit over the classic parareal method is, that it makes use of previously computed solutions within each interval. The computational cost associated with the fine propagator becomes much cheaper. The hybrid parareal SDC approach has shown to significantly reduce the computational costs for each iteration compared to the classic parareal approach. Further, it has been shown, that the cost per iteration then is dominated by the sequential, coarse prediction. For the solution of PDEs this offers the attractive opportunity to use coarser spatial discretization for the coarse propagator. However, SDC methods applied to stiff ODEs have shown order reduction phenomena for too big time-step sizes [49]. In 2014, Bu and Lee [17] proposed an enhanced algorithm for stiff systems. Instead of SDC, the authors used Krylov Deferred Correction (KDC) as a fine propagator. The algorithm was tested for the nonlinear, stiff ODE system of the Van der Pol oscillator with increasing stiffness. Analysis and numerical results showed, that the new algorithm converges in less iterations by increasing the order of the parareal method.

Implicit-Explicit Parareal Another approach to decrease the computational costs associated both with the fine and the coarse propagator was suggested by Wang and Wu [135] in 2014. They proposed an implicit-explicit version of the parareal algorithm, implemented with different pairs of implicit-explicit Runge-Kutta methods (IMEX RK, c.f. [4]). Although not exactly derived for stiff problems, the approach is especially suitable, when the right hand side function f can be split into stiff and non-stiff parts $f = f_{\text{stiff}} + f_{\text{non-stiff}}$[1]. Then, the stiff part is assigned with the costly implicit component of the IMEX RK method in order to satisfy stability requirements, while the non-stiff part is assigned with its explicit, less costly component. Different combinations of parareal IMEX RK pairs were tested by means of the Gray-Scott reaction-diffusion model. Best results were achieved, when assigning a fully implicit-explicit Euler scheme to the coarse integrator.

[1]Unfortunately, this is typically not given in atmospheric chemical kinetics. For the sake of completeness, IMEX parareal is presented here though.

Chapter 4

Numerical Experiments - Six-Variable Mechanism

In the precedent Chapter, we have been discussing two potential implementations of an in-time parallelization of the chemical kinetics in compound air quality models: An internal parallelization within one splitting interval or an external parallelization across multiple splitting intervals. We have further been discussing different techniques for the parareal solution of stiff ODEs arising in chemical kinetics. In this Chapter, we will focus on an internal parallelization and showcase adaptive parareal approaches by means of the example six-variable tropospheric mechanism, a stiff and nonlinear system of ODEs, as introduced in Sec. 2.2.2. In all cases, the total integration interval is set to one full hour, $T_0 = 0$ sec and $T_{\text{end}} = 3600$ sec. We consider a simple parareal implementation as described by means of Alg. 3.1. Since the parareal algorithm is an iterative scheme, we need to define a stopping criterion. For all experiments presented in the following, we define the stopping criterion such that the parareal scheme exits, as soon as the relative change in the approximated solution at final time T_{end}, $U_N \approx u(T_N)$,

$$\left\| \frac{U_N^k - U_N^{k-1}}{U_N^k} \right\|_{L^2} = \sqrt{\sum_{i=1}^{n_{\text{spec}}} \left(\frac{U_{i,N}^k - U_{i,N}^{k+1}}{U_{i,N}^k} \right)^2}, \tag{4.0.1}$$

falls below a threshold of $\text{tol}_{\text{stop}} = 10^{-2}$.

4.1 Classical Parareal (CP)

In a first experiment, we investigate the classic parareal approach as introduced by Lions et al. [74] using fixed time-steps for both the coarse and the fine integrator with different time-step sizes. The time-step size for the coarse propagator is set to the maximum time-step size that still guarantees stability, which is $\delta T = 10^{-3}$ sec. For the fine propagator, we set $\delta t = 10^{-5}$ sec.

In a sequential simulation of the stiff, nonlinear system (2.2.1-2.2.6) using a fixed time-step implicit Euler scheme with $\delta t = 10^{-5}$ sec, $N = 360\ 000\ 000$ time-steps are necessary to propagate the solution from $T_0 = 0$ sec to $T_{\text{end}} = 3\ 600$ sec. The total WCT needed is $\text{WCT}_{\mathcal{F}} = 45\ 552.34$ sec. For the following investigations, we assume, that the computing time per time-step is constant and independent of the actual size of a time-step. Then, we can approximate the computing time per step as one time-step $\text{CT}_{\text{step}} \approx \text{CT}_{\mathcal{G}} \approx \text{CT}_{\mathcal{F}} = \frac{\text{WCT}_{\mathcal{F}}}{N} \approx 0.135 \cdot 10^{-3}$ sec.

An estimate for the parallel WCT has already been presented by means of Eq. (3.2.6). With above introduced constant effort per time-step, it is being simplified to

$$\begin{aligned} \text{WCT}_{\text{par}} &= \text{CT}_{\text{step}} \left(\underbrace{N \frac{\Delta t}{\delta T}}_{N_{\text{init}}} + k_{\text{conv}} \underbrace{\left(N \frac{\Delta t}{\delta T} + \frac{\Delta t}{\delta t} \right)}_{N_k} \right) \\ &= \text{CT}_{\text{step}} \left(N_{\text{init}} + k_{\text{conv}} \cdot N_k \right) \\ &= 0.135 \cdot 10^{-3} \left(\frac{3\ 600}{10^{-3}} + k_{\text{conv}} \left(\frac{3\ 600}{10^{-3}} + \frac{3\ 600}{N \cdot 10^{-5}} \right) \right) \text{ sec} \\ &= 486 + k_{\text{conv}} \left(486 + \frac{10^2}{N} \right). \end{aligned} \tag{4.1.1}$$

From Eq. (4.1.1), that the parallel WCT for the classic parareal approach is below the WCT of the serial integration $\text{WCT}_{\mathcal{F}}$ for all choices of N and for all iteration counts $k_{\text{conv}} < 76$ - in practice, typically 1 to 5 iterations are needed until convergence. The smallest parallel WCT is then given for $N \to \infty$. Let us assume, 5 iterations are necessary for $N \to \infty$. Then a parareal scheme on $p \to \infty$ processors will find a solution in $\approx 3\ 000$ sec, that is approx. 6.6% of the WCT needed for a sequential integration with $\mathcal{F}$. Compared to a sequential integration using an implicit Euler scheme, the parareal integration will be significantly faster. The situation changes dramatically, when comparing the parareal approach to a sequential solver, specialized for stiff ODEs, such as RODAS(3)4 [46], an L-stable, adaptive Rosenbrock-solver of order 4 with a third order embedded local error estimator. A RODAS(3)4 solver initiated with a relative tolerance of $rtol = 10^{-4}$ finds a solution in 102 adapted time-steps and a total WCT of approx. 0.02 sec. For this scenario, the classic parareal scheme obviously is not competitive with the specialized, adaptive time-stepping scheme!

4.2 Adaptive Parareal I (AP-I)

In a next experiment, we test the adaptive parareal approach, as presented by Nielsen [96]. For both $\mathcal{F}$ and $\mathcal{G}$ we use the same time-stepping scheme, RODAS(3)4, but with different tolerances. The relative tolerance for the fine propagator is set to $\text{rtol}_{\mathcal{F}} = 10^{-4}$, the tolerance for the coarse solver is $\text{rtol}_{\mathcal{G}} = 10^{-2}$. The total integration period $[T_0, T_{\text{end}}]$ is split into N equidistant time-slabs of size Δt.

Figure 4.1 shows the adapted time-steps for a sequential integration over the full time interval $[T_0,\ T_{\text{end}}]$ decomposed into N equidistant sub-intervals with varying N. On each of the sub-intervals $[t_n,\ t_n + \Delta t]$, the coarse propagator $\mathcal{G}$ with $\text{rtol}_{\mathcal{G}} = 10^{-2}$ is re-initiated with an initial time-step size of $h_0 = 10^{-9}$ sec and the initial conditions U_n^0 equal to the solution at final time of the previous time-slab, that is $U_n^0 = \mathcal{G}(\Delta t,\ U_{n-1}^0)$. The number of accumulated time-steps over all sub-intervals grows with the number of time-slabs N. Figure 4.2 further shows, that the accumulated number of time-steps in the sequential propagation over all N sub-intervals using an adaptive coarse scheme $\mathcal{G}$ scales approx. exponentially with the number of time-slabs. This is due to the fact that the time-step size is initialized with $h_0 = 10^{-9}$ sec on each of the time-slabs.

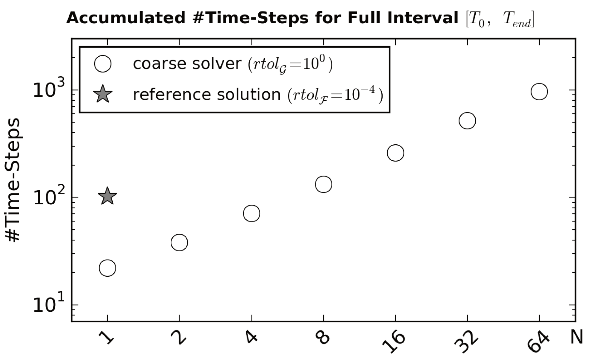


Figure 4.2: Accumulated number of time-steps for the sequential prediction from $[T_0, T_{end}]$ over N equidistant time-slabs of size $\Delta t = (T_{end} - T_0)/N$ using the coarse propagator $\mathcal{G}$ with $rtol_{\mathcal{G}} = 10^0$.

Within the parareal algorithm, a sequential coarse propagation over the full interval is executed $1 + k_{\text{conv}}$ times: First of all, in the sequential initialization phase; in the following once per iteration within the parareal correction step. From Fig. 4.2 it can be seen, that for more than four time-slabs, i.e. $N > 4$, the accumulated number of time-steps over all sub-intervals already exceeds the number of time-steps, that are necessary for a sequential integration from $[T_0,\ T_{\text{end}}]$ using the fine propagator $\mathcal{F}$. Already in the initialization phase, the cost for a purely sequential (non-parareal) integration using $\mathcal{F}$ is smaller than in the adaptive parareal case. An adaptive coarse propagator, that is being re-initiated with h_0 at every sub-interval, does therefore not present a promising choice with regard to a potential speed-up and an increase in efficiency.

In the following, we discuss the load balancing during the parallel correction step. Figure 4.4 shows the internally adapted time-step sizes of the fine propagator $\mathcal{F}$ on each time-slab for a

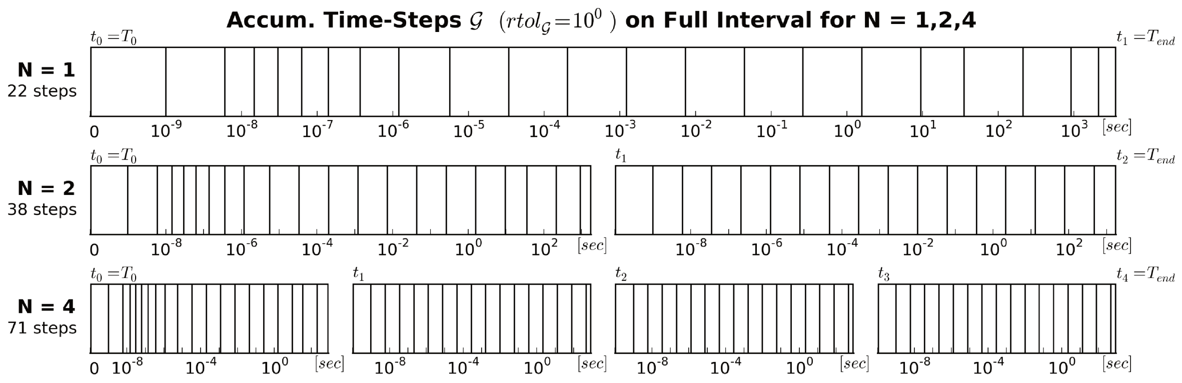


Figure 4.1: Adapted time-steps on the decomposed interval $[T_0,\ T_{\text{end}}]$ into N equidistant sub-intervals $[t_0,\ t_1]$, $[t_1,\ t_2]$, ..., $[t_{N-1},\ t_N]$ for $N = 1, 2, 4$. On each of the N sub-intervals, the coarse propagator $\mathcal{G}$ is started with $\text{rtol}_{\mathcal{G}} = 10^0$ and an initial time-step size of $h_0 = 10^{-9}$ sec. The accumulated number of time-steps on the full interval displayed on the very left increases with growing N.

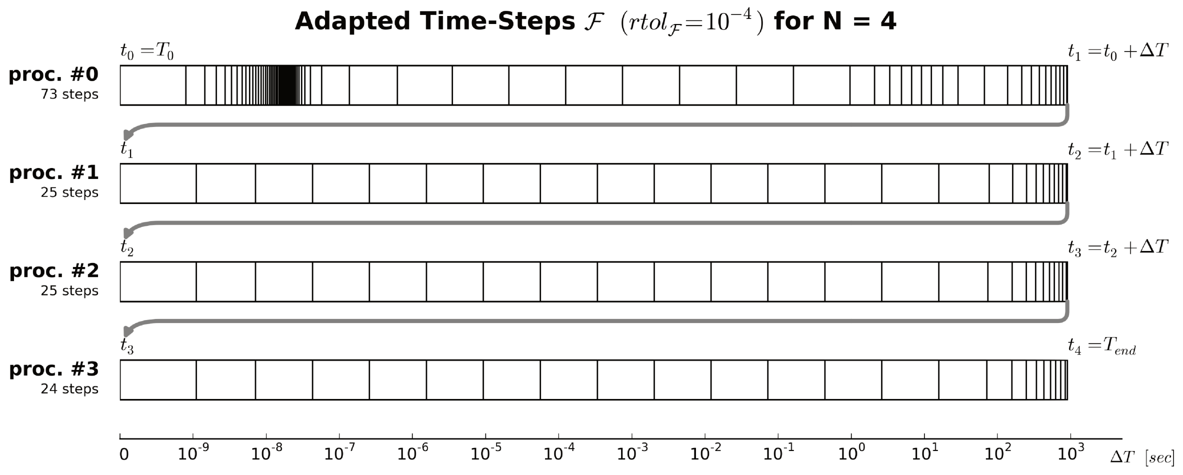


Figure 4.4: Adapted time-step sizes for the parallel fine computation using $\mathcal{F}$ with $rtol_{\mathcal{F}} = 10^{-4}$ in the first iteration. The computational effort to be afforded by the first processor equates approximately three times the effort of the other processors.

decomposition into 4 sub-intervals. Initial conditions for all sub-intervals have been computed from a sequential, coarse initialization. The initial step size used by $\mathcal{F}$ is $h_0 = 10^{-9}$ sec on all time-slabs.

Due to the fact, that the initial solution at time T_0 is out of balance and leads to a stiff initial value problem, the first processor (proc. #0) has to resolve very small timescales, leading to a high number of total time-steps on this interval. All other processors rapidly increase their internally used time-step sizes h, leading to a comparably small number of time-steps, as can also be seen in Fig. 4.3. A decomposition into equidistant time-slabs will therefore lead to load imbalances between the processors during the parallel prediction step: For a decomposition into 4 sub-intervals, the computational effort to be required by the first processor equates approximately three times the effort of the other processors. A global imbalance of work is the result.

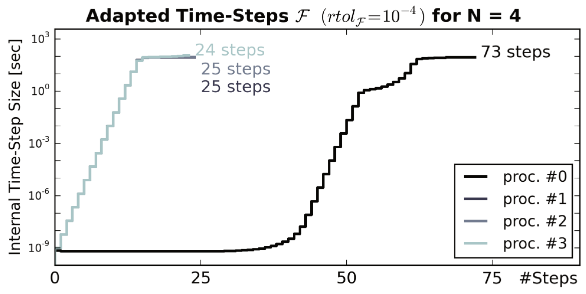


Figure 4.3: Adapted time-step sizes for AP-I and $N = 4$ during parallel correction using $\mathcal{F}$ with $rtol_{\mathcal{F}} = 10^{-4}$ in the first iteration. The computational effort to be afforded by processor #0 equates approximately three times the effort of the other processors.

4.3 Adaptive Parareal II (AP-II)

Now, we test the adaptive parareal approach, as presented in Guibert and Tromeur-Dervout [43]. Different, than in the previous test, we now also introduce adaptivity in the decomposition of the time interval and the number of time-slabs. Again for both $\mathcal{F}$ and $\mathcal{G}$ the RODAS(3)4 solver with $\text{rtol}_{\mathcal{F}} = 10^{-4}$ and $\text{rtol}_{\mathcal{G}} = 10^{-2}$ is used during the iteration.

The decomposition of the time domain is now suggested by the coarse integration during the initialization, using N internally adapted time-steps. The result is a non-uniform temporal mesh with N sub-intervals, whereat N can be controlled by changing $\text{rtol}_{\mathcal{G}}$, i.e. $N = N(\text{rtol}_{\mathcal{G}})$.

Figure 4.1 shows the quality of the initial solution, estimated by the relative error of the parareal solution at final integration time T_{end} compared to a fine sequential reference solution $u(T_{\text{end}})$. The reference solution has been calculated a priori using a sequential RODAS(3)4 solver with $\text{rtol} = 10^{-6}$. The relative error at time T_{end} is calculated as

$$\left\| \frac{u(T_{\text{end}}) - U_N^k}{u(T_{\text{end}})} \right\|_{L^2} = \sqrt{\sum_{i=1}^{s} \left(\frac{u_i(T_{\text{end}}) - U_{iN}^k}{u_i(T_{\text{end}})} \right)^2}. \tag{4.3.1}$$

For AP-I, the quality of the initial solution improves with an increasing number of sub-intervals N, although the relative tolerance for the coarse propagator remains the same. In terms, the internally used number of time-steps also increases with growing N, as it had been depicted in Fig. 4.2, leading to an actual quality, higher than defined in $\text{rtol}_{\mathcal{G}}$. For AP-II, the number of sub-intervals N is not a parameter, but defined by the choice of the relative tolerance. Respectively, also the quality of the solution depends on the choice of the relative tolerance $\text{rtol}_{\mathcal{G}}$. As a result, the number of iterations needed until convergence is higher than for AP-I with the same number of N, as can be seen in Tab. 4.1.

In the following, we consider the parallel WCT of AP-II. The computational complexity of $\mathcal{F}$ and $\mathcal{G}$ (the RODAS(3)4 solver) is defined by the number of internal function evaluations, which depends on the number of accepted and rejected steps. For increasing tolerances, the number of rejected time-steps increases nonlinearly. Typically, the number of rejected time-steps is a lot smaller than the number of accepted time-steps, so rejected time-steps are neglected in the following estimate. Again, we denote the computational time for one time-step by CT_{step} and assume it to be constant and equivalent for both $\mathcal{F}$ and $\mathcal{G}$. We can then define the computational complexity for an integration over N internal time-steps as $\text{CT}_{\text{step}} \cdot N$. Hence, the WCT for the initialization, using the coarse solver $\mathcal{G}$ is given by

$$\text{WCT}_{\text{init}} = \text{CT}_{\text{step}} \cdot N_{\text{init}} = \text{CT}_{\text{step}} \cdot N.$$

The WCT per iteration is defined by

$$\text{WCT}_k = \text{CT}_{\text{step}} \cdot \underbrace{\left(\max_{i=0,N-1} (N_{i,\mathcal{F}}^k, N_{i,\mathcal{G}}^k) + \sum_{i=0}^{N-1} \tilde{N}_{i,\mathcal{G}}^k \right)}_{N_k},$$

whereat $N_{i,\mathcal{F}}^k$ denotes the number of internally adapted time-steps during the parallel correction step on the $i-th$ time-slab and in the $k-$th iteration using the fine propagator $\mathcal{F}$. Respectively, we define $N_{i,\mathcal{G}}^k$. Per design, we can assume the coarse propagator to require less time-steps than the fine propagator, i.e. $\max_{i=0,N-1}(N_{i,\mathcal{F}}^k, N_{i,\mathcal{G}}^k) = \max_{i=0,N-1}(N_{i,\mathcal{F}}^k)$. The variables $\tilde{N}_{i,\mathcal{G}}^k$ denote the number of internally adapted time-steps during the sequential

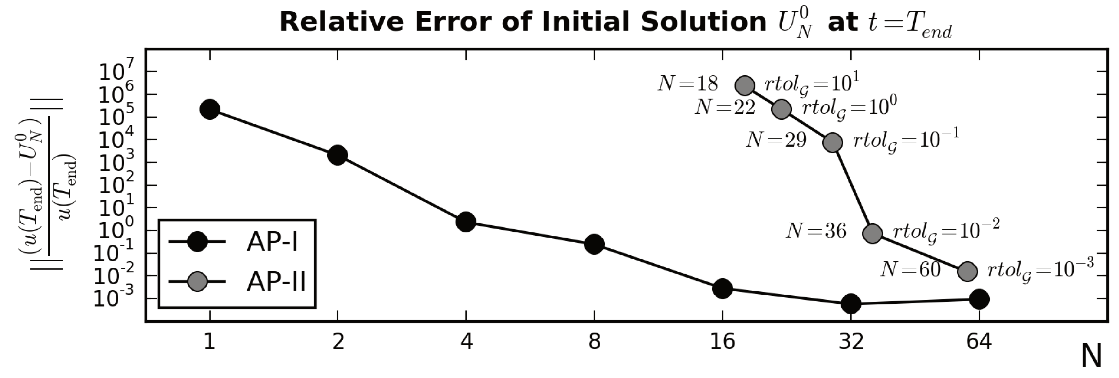


Figure 4.1: Relative error of coarse, sequential initial solution at time $t = T_n$ to fine, sequential reference solution $y(T_N)$. For AP-I, the quality of the initial solution improves with an increasing number of sub-intervals N, while $rtol_\mathcal{G} = 10^0$ in all cases. For AP-II, the quality of the reference solution only depends on the choice of the relative tolerance, independent of the number of sub-intervals. The reference solution was calculated using a relative tolerance of $rtol = 10^{-6}$.

parareal prediction step on the $i-th$ time-slab and in the $k-$th iteration using the coarse propagator $\mathcal{G}$. This leads to a parallel WCT of

$$\begin{aligned}\mathrm{WCT}_{\rm par} &= \mathrm{WCT}_{\rm init} + \sum_{k=1}^{k_{\rm conv}} k \cdot \mathrm{WCT}_k \\ &= \mathrm{CT}_{\rm step} \cdot \underbrace{\left(N + \sum_{k=1}^{k_{\rm conv}} k \cdot N_k \right)}_{N_{\rm par}}. \end{aligned} \tag{4.3.2}$$

We define the number of non-parallelizable time-steps, $N_{\rm par}$, that is the number of time-steps, that have to be calculated consequently in the iterative parareal approach,

$$N_{\rm par} = N + \sum_{k=1}^{k_{\rm conv}} k \cdot N_k = N + \sum_{k=1}^{k_{\rm conv}} k \cdot \left(\max_{i=0,N-1}(N_{i,\mathcal{F}}^k) + \sum_{i=0}^{N-1} \tilde{N}_{i,\mathcal{G}}^k \right). \tag{4.3.3}$$

From Eqs. (4.3.2) and (4.3.3) we can directly see, that the parallel WCT for a given number N is dominated by three factors: a) the maximum number of internal time-steps over all time-slabs during the parallel prediction per iteration, $\max_{i=0,N}(N_{i,\mathcal{F}}^k)$, b) the summed coarse update steps during the sequential correction per iteration, $\sum_{i=0}^{N} \tilde{N}_{i,\mathcal{G}}^k$, and c) the number of iterations $k_{\rm conv}$ until convergence, that we have already seen to be influenced by the quality of the initial solution.

The first influencing factor, the maximum number of internal time-steps over all N time-slabs, is given by $\max_{i=0,N-1}(N_{i,\mathcal{F}}^k)$. From Sec. 4.2 and the experiments within we have seen, that it is advisable, to target a decomposition in a way, that all $N_{i,\mathcal{F}}$ are balanced over all time-slabs. In Fig. 4.2, we see the number of internally adapted time-steps per time-slab/processor during the parallel fine propagation within the parallel correction using $\mathcal{F}$ with

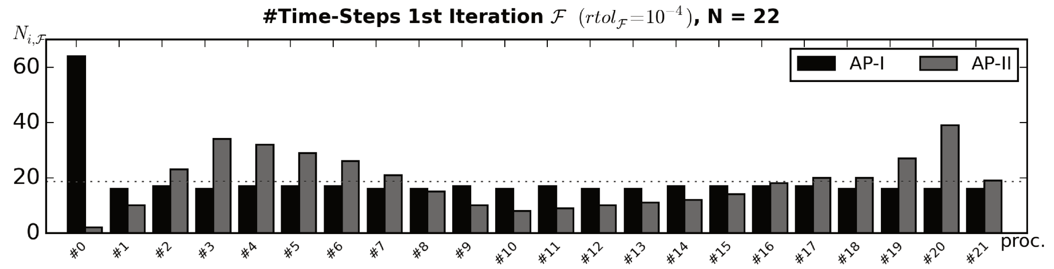


Figure 4.2: Internally adapted number of time-steps during the parallel correction by means of $\mathcal{F}$ with $rtol_{\mathcal{F}} = 10^{-4}$ in the first iteration. For AP-I the decomposition is uniform, i.e. all 20 time-slabs have the same size. For AP-II the decomposition was chosen as suggested by the sequential, coarse integration using $\mathcal{G}$ with $rtol_{\mathcal{G}} = 10^0$. The dotted horizontal line shows the average number of time-steps both for AP-I and AP-II, the average values approx. coincide here. The maximal deviation from the average is smaller for AP-II, hence the work-load is more balanced than for AP-I.

$\text{rtol}_{\mathcal{F}} = 10^{-4}$ in the first iteration. The dotted horizontal lines shows the average number of time-steps for both AP-I and AP-II, whereat the averages approx. coincide. For AP-II the decomposition is suggested by a coarse initialization, whereat varying the tolerance leads to a different decomposition. For $\text{rtol}_{\mathcal{G}} = 10^0$, a decomposition into $N = 22$ time-slabs is the result. Respectively, we set up AP-I with a decomposition into 22 equidistant time-slabs. For AP-I, 64 internally adapted time-steps are needed on the first time-slab, while 16-17 time-steps are necessary on the following 21 time-slabs. The consequence again is, that the global work during the parallel, fine prediction is not balanced over all processors, but mainly afforded by processor #0. For AP-II, the maximal deviation from the average is smaller than for AP-I. The computational effort per processor is equilibrated better amongst all processors.

The second factor of influence is the number of coarse update steps $\sum_{i=0}^{N-1} \tilde{N}_{i,\mathcal{G}}^k$. Same as for AP-I, we restart the coarse integrator with an initial time-step size of $h_0 = 10^{-9}$ also for AP-II. In Sec. 4.2 we have already seen, that an adaptive coarse propagator restarted with very small time-step sizes during the sequential prediction leads to an exponentially growing accumulated number of steps for the coarse prediction, see Fig. 4.2. In practice, it turns out, that both for AP-I and AP-II the number of coarse update steps dominates the overall WCT, as can also be seen in Tab. 4.1. In principle, it is possible to lower that portion by re-initiating $\mathcal{G}$ with bigger time-step sizes during the sequential prediction step. A consequence will however be a loss of quality of the coarsely predicted solution. Then, higher numbers of iterations will be needed until convergence. In the following Sec. 4.4, we will showcase the influence of using a very coarse prediction (a one-step, non-adaptive integrator) on the number of iterations until convergence and the parallel WCT.

Theoretically, it is also possible, to re-initiate fine and coarse propagators with bigger step sizes during the parallel correction step. In practice, this approach turns out to be trappy. The initial value problem initialized with the actual solution $u(t_n)$ at time t_n may already be balanced, so that no fast modes are present. Then, bigger time-step sizes can be chosen. However, this is not necessarily true for the approximated solution U_n^k, which may still be disturbed by fast modes, that did not yet reach equilibrium in the iterative parareal approach. In this case, very small time-steps will be needed to approximate the next iterate of the parareal solution at time t_n. A reinitialization with bigger time-step sizes may then lead to a sequence of decreasing time-steps and hence a higher number of rejected time-steps.

4.4 Adaptive Parareal III (AP-III)

As we have seen in the previous tests, using an adaptive solver as a coarse propagator during the parareal iteration leads to an overhead in the sequential coarse prediction, since the adaptive solver scales approximately exponentially with the number of time-slabs N, see Fig. 4.2. Therefore, we now consider a coarse propagator, that uses one single time-step only.

Same as for AP-II, the initial decomposition of the time domain is suggested by a first, coarse integration using the RODAS(3)4 solver. The number of time-slabs N is controlled by changing tolerances. During the parareal iteration, the coarse solver is now forced to use exactly one time-step, i.e. the solution is propagated from the beginning of one time-slab to its end without any intermediate steps. The total computational cost for the sequential coarse updates then scales linearly with the number of time-slabs.

Figure 4.1 shows the convergence rates of AP-I, AP-II and AP-III for different $N = 10,\ 20,\ 30,\ 40$. The x-axis denotes the number of iterations, whereat 0 equates the initialization and 1 the first parareal iteration. The logarithmic y-axis denotes the relative error in the $k-th$ iteration for the solution U^k at final integration time T_{end}, calculated from Eq. (4.3.1),

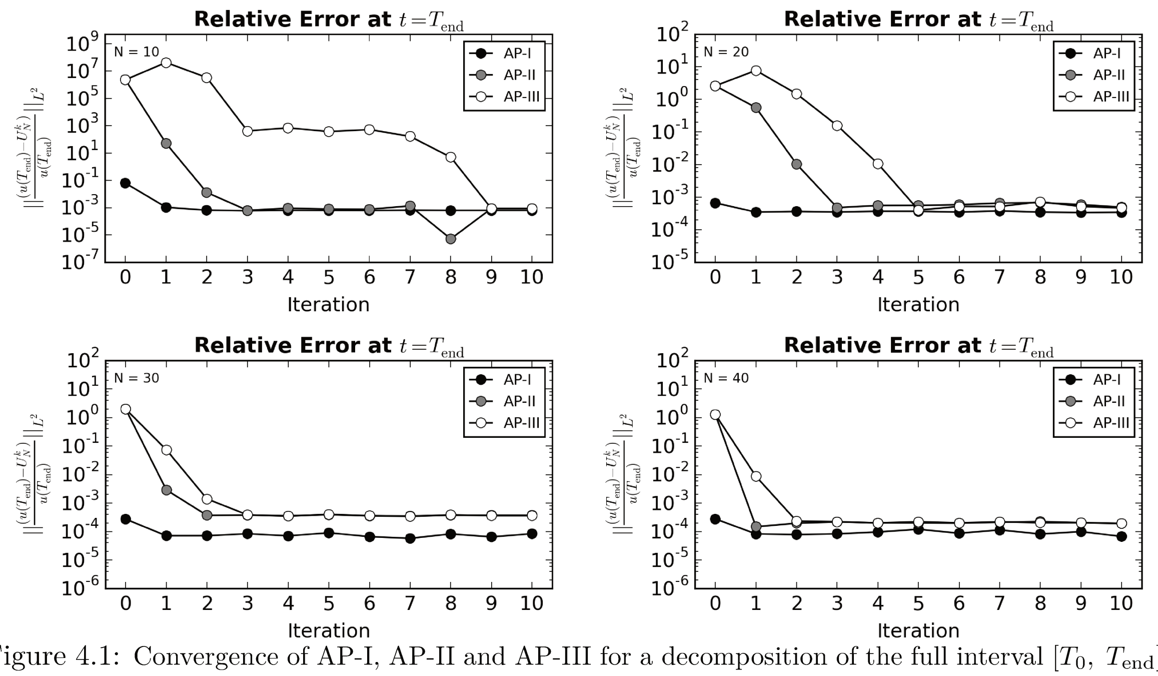


Figure 4.1: Convergence of AP-I, AP-II and AP-III for a decomposition of the full interval $[T_0,\ T_{\text{end}}]$ into $N = 10,\ 20,\ 30,\ 40$ time-slabs. For AP-I the time-slabs are equidistant and of size $(T_0 - T_{\text{end}})/N$. For AP-II and AP-III the decompositions coincide. The size of their time-slabs has been suggested by the initial coarse prediction using $\mathcal{G}$, leading to non-uniform decompositions with varying time-slab sizes.

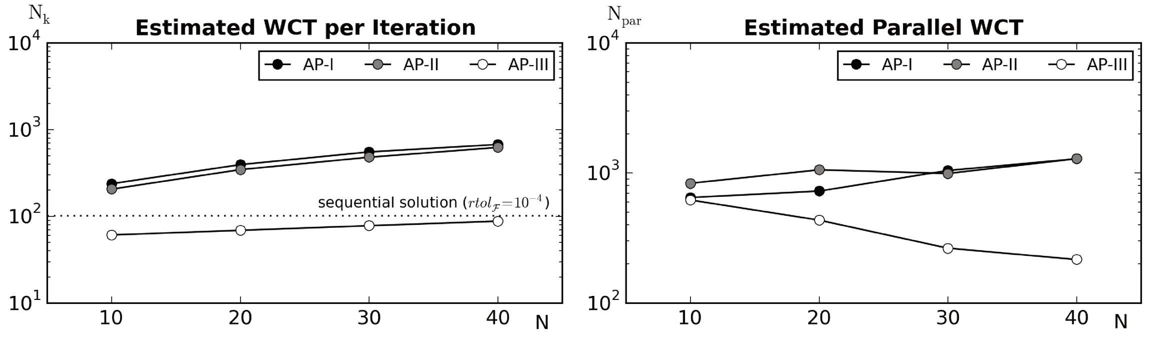


Figure 4.2: Estimated WCT per iteration (left) and estimated parallel WCT (right) for AP-I, AP-II and AP-III. The x-axis denotes the number of time-slabs, i.e. the number of parallel processes. The y-axis denotes the total number of sequentially executed time-steps N_k and N_{par} respectively. The WCT is a linear function of the number of steps multiplied with the (approx.) constant computing time per step CT_{step}, $\text{WCT} = \text{CT}_{\text{step}} \cdot N$.

with the reference solution as described in Sec. 4.3.

It can easily be seen, that AP-I starts with significantly smaller initial relative errors than AP-II and AP-III for all choices of N. We have been discussing this behavior already by means of Fig. 4.1. Since the initial solution for AP-I is already quite accurate, the convergence is best for AP-I for all choices of N.

The initial decomposition into time-slabs are equivalent for AP-II and AP-III. Consequently, AP-II and AP-III start with the same relative error for all choices of N. In the following iterations, the relative error decreases faster for AP-II. This can be blamed on the quality of the coarse, sequential prediction: While AP-III uses only one time-step with a fixed time-step size to propagate the solution from the beginning to the end of each time-slab, AP-II uses an adaptive solver during the coarse correction step. It is therefore no surprise, that AP-III converges slower than AP-II.

In Fig. 4.2 we see the estimated parallel WCT per iteration (left) and in total (right) for AP-I, AP-II and AP-III. In both plots, the WCT is depicted not in terms of seconds but in terms of the number of non-parallelizable time-steps N_{par}. The parallel WCT is a linear function of N_{par} multiplied with the (approximately) constant computing time per time-step CT_{step}, $\text{WCT}_{\text{par}} = \text{CT}_{\text{step}} \cdot N_{\text{par}}$. The respective numbers of iterations, the number of time-steps in the initialization phase, the averaged number of time-steps per iteration and the total number of non-parallelizable steps N_{par} for varying N are also given in table 4.1.

The number of steps per iteration is highest for AP-I. The small difference in steps per iteration between AP-I and AP-II showcases the effect of the load-balancing achieved from a decomposition into non-uniform time-slabs, see also column 4 and 5, Tab. 4.1. The values for $\max_{i=0,N-1}(N_{i,\mathcal{F}})$ coincide for AP-II and AP-III, while the accumulated number of steps for the sequential prediction $\sum_{i=0}^{N} \tilde{N}_{i,\mathcal{G}}$ is significantly smaller for AP-III. In terms, we have seen, that AP-III converges slower, than AP-I and AP-II. Still, the parallel WCT is lowest for AP-III. The dashed line in this plot denotes the number of time-steps in a purely sequential execution of $\mathcal{F}$. Only for AP-III the number of non-parallelizable steps per iteration is below

				N_k averaged over k_conv		
	N	k_conv	N_init	$\max_{i=0,N1}(N_{i,\mathcal{F}})$	$\sum_{i=0}^{N-1} \tilde{N}_{i,\mathcal{G}}$	N_par
AP-I	10	2	172	66	172	648
	20	1	331	64	331	726
	30	1	491	63	491	1045
	40	1	612	63	612	1287
AP-II	10	4	10	51	155	834
	20	3	20	49	297	1058
	30	2	30	48	433	992
	40	2	40	48	576	1288
AP-III	10	10	10	51	10	620
	20	6	20	49	20	434
	30	3	30	48	30	264
	40	2	40	48	40	216
RODAS(3)4	**-**	**-**	**-**	**-**	**-**	**102**

Table 4.1: Parallel computational effort until convergence, estimated by the total number of non-parallelizable time-steps N_par, i.e. the accumulated number of sequentially executed time-steps in the parareal algorithm. The parallel WCT can be approximated as $\mathrm{WCT}_\mathrm{par} = N_\mathrm{par} \cdot \mathrm{CT}_\mathrm{par}$.

the number of steps required for a purely sequential integration using $\mathcal{F}$ (102 steps). In practice however, AP-III needs 2-10 iterations until convergence. The parallel WCT for AP-III is therefore more than 100% higher than the WCT of a sequential integration with a respectively set-up RODAS(3)4 solver. All parallel algorithms presented so far are therefore slower than the purely sequential execution of the RODAS(3)4 solver.

4.5 Conclusions

In this Chapter, we showed numerical results for an internal parallelization of an example six-variable tropospheric mechanism. The classical parareal [CP] algorithm was designed for a decomposition in time with fixed, equidistant time-slab sizes. We showed, that even for an infinite number of parallel processes, the classical parareal scheme is not competitive in terms of WCT to a sequential, adaptive time-stepping scheme, such as the Rosenbrock solver RODAS(3)4 [46]. When using an adaptive solver as the fine propagator in combination with a decomposition into equidistant time-slabs [AP-I], global load-imbalances have shown to result. Those imbalances are a direct result of the stiffness within the system, whose presence forces the adaptive solver to use very small time-steps to guarantee stability. If the problem is stiff, more time-steps will be required, leading to a higher computational demand. A non-equidistant decomposition in time [AP-II] has been shown to improve the global load-imbalances. As an initial suggestion for a non-equidistant decomposition, the same adaptive solver can be utilized as during the iteration, but with a coarser tolerance. Different than in the classical approach, the number of time-slabs and respectively the number of parallel processes can't be defined directly by the user, but only implicitly by tuning the tolerance of the coarse initialization solver call. Special care has to be taken, when employing an adaptive time-stepping scheme also as the coarse solver during the sequential prediction step within an iteration. From a performance point of view, the sequential prediction should be very cheap, since it is the only sequential part of the algorithm. When using an adaptive scheme, very small time-step sizes are typically necessary at the beginning of each time-slab. This makes the overall sequential prediction very costly. To decrease the computational cost, the coarse solver was then forced to use only one time-step [AP-III]. Since this restriction deteriorates the quality of the approximated solution, more iterations were necessary until convergence.

All adaptive parareal approaches have shown to converge, but not to outperform the purely sequential integrator in terms of WCT. In the following, we shall therefore consider alternative candidates for cost-efficient, but still accurate coarse propagators: reduced models. In the next Chapter, we focus on the derivation of reduced models for atmospheric chemical kinetics. We examine the qualification of different reduced models for the application as a coarse propagator within the parareal algorithm. A suitable reduced model should be significantly faster than the full model, without a substantial degradation in accuracy.

Chapter 5

Model Reduction

In the previous Chapter, we had identified needs for a fast, but still relatively accurate propagator to be applied as a coarse propagator within the parareal algorithm. In this section we investigate the qualification of different reduced models for this purpose. To this end, an overview of existing approaches and their application to atmospheric chemical kinetics is presented. The two most popular approaches within (lumping methods and QSSA) will be outlined first. Following, two in this context more promising approaches, ILDM and repromodeling, will then be discussed in more detail.

5.1 Overview

A detailed reaction mechanism for atmospheric gas-phase chemistry may include hundreds of chemical species, that interact in hundreds to thousands of reactions. One such example is the comprehensive Leeds Master Chemical Mechanism (MCM) [52], which - in its current version MCMv3 - describes tropospheric chemistry by means of 5 900 species and 13 500 reactions. The numerical simulation of such systems is in general possible, but can lead to considerable computational effort. In practice, chemistry calculations have to be carried out repeatedly at millions of grid-points, as they typically are embedded in chemical transport models, that may be part of large-scale AQMs. But both processor's memory and computing capacity are limited, so a direct numerical integration of the full system for each grid-point is hardly feasible. Simplifications of the complex chemical systems are crucial towards their efficient incorporation into a regional modeling framework.

A straightforward way to reduce both memory requirements and CPU-time is to decrease the number of species and/or the number of reactions. This can be accomplished by a removal of redundant species and/or reactions within the mechanism. The result is a “skeletal” scheme, that is designed to comprise less species and reaction steps, but still captures the main features of the full scheme. Many different methods to identify redundant species and/or reactions exist, starting with heuristic approaches such as trial and error methods or comparing reaction rates. More sophisticated methods base on timescale and sensitivity analysis, investigations of reaction graphs, entropy production and optimization methods. For an overview of species and reaction removal methods, see for example Turányi and Tomlin [123].

Reducing the number of species and/or reactions in a mechanism is not necessarily related to a direct removal of species/reactions. Lumping approaches present an alternative approach, that refrains from removing individual species or reactions. Instead, sets of chemical species with similar characteristics are replaced by aggregated species, that can either be real or surrogate species. Lumping approaches have been a very popular way to decrease the number of unknowns for decades, see for example Atkinson [5], Li and Rabitz [65, 66], Middleton et al. [89], Wang et al. [132] or Whitehouse et al. [137].

The CPU-time does not only depend on the number of unknowns, but also strongly on the presence of a wide range of timescales within such a chemical system. In literature, such multi-scale initial value problems are encountered under the term stiff problems. Stiffness has been shown as the major constraint on the choice of the integration scheme and the size of the time-steps for the numerical integration, see Section 2.3.2.

The operator splitting approach, as introduced in Sec. 2.1 and as it is typically applied to solve the mass balance equations of chemical species Eq. (2.1.4), directly enforces the stiffness of the initial value problems (2.1.6) to be solved for the chemical kinetics: As a result of the splitting of advection, diffusion and reaction, and due to abrupt photolysis updates when considering chemical kinetics in an autonomous mode, transient species will be present at the beginning of every splitting interval. To resolve such fast processes, very small time-steps are required during the numerical integration at initial time. This characteristics is being addressed by a class of model reduction methods basing on a separation of timescales.

A famous approach for model reduction is the Quasi-Steady State Approximation (QSSA) (c.f. Hesstvedt et al. [47]), where timescales are related to chemical species. Fast species are assumed to react instantaneously and locally equilibrate with respect to slow species. Then, their concentration can be determined as algebraic functions of the slow ones and the size of the ODE system to be solved reduces to the number of slow species. From a theoretical point of view, the QSS assumption has one severe drawback: not all species can sharply be classified into either fast or slow, since some species interact both in fast and slow processes at the same time. More sophisticated approaches, coming from the existence of *slow invariant manifolds* (SIM) (c.f. Fenichel [33]), overcome this drawback, by associating timescales with the system's modes, i.e. the eigenvectors and eigenvalues. The modes, with large, negative eigenvalues are assumed to equilibrate quickly. The solution then is said to have collapsed onto a lower-dimensional manifold. One such example is the Intrinsic Low Dimensional Manifold (ILDM) method (c.f. Maas and Pope [81]), in which fast modes are completely eliminated from the system during integration time. This does not necessarily lead to a decrease in the number of unknowns, but to a decrease in stiffness and hence to larger time-steps that can be used during integration time. Finally, the fast modes are incorporated in a parametrized fashion depending on the slow modes at final time. The outcome of such methods hence is not a reduced or skeletal mechanism, but a completely new set of ODEs, defined in the state space of the slow modes, along with some algebraic restrictions. However, such methods are tricky: Besides the identification of a lower dimensional manifold, onto which the system collapses eventually, and the selection of the parameterizing variables, one essential challenge is the calculation of the parametrization itself. For a nonlinear ODE, the calculation of one parametrization point equates solving one nonlinear problem, which can make the method computationally demanding. Performance gains can be achieved by calculating the parametrization a priori and storing them in look-up tables. Due to memory restrictions, such look-up tables are not feasible for atmospheric chemistry, as shown by Lowe and Tomlin [77].

In many cases, above introduced model reduction methods do not yet lead to reduced models, that can be incorporated in a satisfactory way as a coarse propagator, since their numerical solution is either not accurate enough or not significantly faster than the solution of the full model. A possible further step is the parametrization of the reaction mechanisms itself, which is often encountered under the term repro-modeling (c.f. [121, 122]). The idea behind such techniques is to describe chemical kinetics by means of explicit algebraic functions, which can be obtained by numerical fitting. A number of approaches exist, among them the parametrization via orthonormal polynomials as introduced by Turányi [122] or using spline functions as proposed by Buki et al. [18]. Within the last years the method of High Dimensional Model Representation (HDMR, c.f. Rabitz [102]), has found wide usage, e.g. by Shorter et al. [113], Wang et al. [134], or Boutahar and Sportisse [16]. Hereby, chemical kinetics is parametrized through an expansion of correlated functions. Successful applications of the HDMR method for atmospheric chemistry have been presented by Wang et al. [131, 133]. Fully Equivalent Operational Models (FEOM) based on the HDMR method were constructed, that were up to 1 000 times faster than the original box-model, while maintaining accuracy comparable to the original model over wide ranges of initial concentrations.

A different class of methods is related to the statistical behavior of the system, the so called class of Proper Orthogonal Decomposition methods (POD), as introduced by Kunisch and Volkwein [57, 58]. A reduced model is obtained as a projection of the full model into the reduced basis, with the reduced basis being constructed from preprocessed trajectories (snapshot method). Investigations of the potential of POD for chemical kinetics have been made by Sportisse and Djouad [116] and Boutahar and Sportisse [16]. The results for a box-model theoretically show great potential, as Sportisse and Djouad [116] could identify reduced models with 2 to 3 local degrees of freedom for a species such as ozone. The original model contained 31 degrees of freedom. Due to the high numerical efforts to optimally choose, calculate, store and evaluate the snapshots, those methods have not gained popularity in atmospheric chemical kinetics in the three-dimensional practice. Those methods therefore are not addressed in this work.

Typically, the chemistry models applied in operational air quality models are already highly reduced models of more comprehensive mechanisms using various of the above mentioned model reduction methods. Starting from a comprehensive mechanism, the first step towards a computationally feasible, smaller model is to eliminate redundant species from the mechanism. Following, a further reduction is achieved by lumping approaches, which are often applied in combination with QSS approximations for special species. One example of a combined application of different model reduction techniques to MCMv2 (3 478 species and 10 763 reactions) along with a detailed discussion has been presented by Whitehouse et al. [137]. Combining timescale analysis, sensitivity analysis and a timescale aware lumping approach led to a reduced mechanism, “that contains 35% of the number of species and 40%

of the number of reactions compared to the full mechanism."

The chemical mechanism, that is being considered in this work, RADM2 [119] (see also Sec. 2.2), is already a highly condensed mechanism. The mechanism builds up from a simplified compound of various state-of-the-art mechanisms, such as the explicit mechanism of Leone and Seinfeld [64], the mechanisms of Lurmann et al. [78], the carbon bond mechanism of Whitten et al. [138] and the master mechanism of Kerr and Calvert [53]. According to Stockwell et al., "The main source of complexity in the explicit mechanisms is the organic chemistry. Treating the organic chemistry of the troposphere explicitly would require thousands of chemical species." Instead, the hundreds of volatile organic compounds (VOCs), are aggregated using a molecular lumping approach. The RADM2 mechanism is comprised of only 63 chemical species and 158 reactions.

This section has been motivated by the search of a reduced model, that can be applied as a coarse propagator within the parareal algorithm. Since RADM2 is already a highly condensed mechanism, we in fact seek a reduced model of a reduced model. In the following, we discuss the two most popular approaches (lumping methods and QSSA). Subsequently, two in this context more promising approaches (ILDM and repro-modeling), will then be discussed in more detail.

5.2 Lumping Methods

Traditionally, lumping methods have been the predominant model reduction approach in atmospheric chemistry. The idea behind it is quite simple and straightforward: A group of species is being substituted in the mechanism by a new model variable, denoted as a *lumped* variable. The lumped variable can either be an actual or a hypothetical species, which depends linearly or non-linearly on the original species. This way, the original set of equations is reduced to a lower dimensional lumped set.

The motivation to use lumping in atmospheric chemical kinetics stems from the hundreds of volatile organic compounds (VOCs) in the hydrocarbon chemistry of the troposphere. Since the late 1970s, it has been common use to group VOCs into smaller numbers of lumped species. One of the first applications was the Dodge mechanism from 1977, in which the total reactive hydrocarbon concentration is represented by two surrogate species only: n-butene and propene.

Within the last decades, several reduced chemical mechanisms have been developed for use in urban and regional AQMs, among them the series of Regional Acid Decomposition Mechanisms (RADM [119]), the Statewide Air Pollution Research Center mechanism family (SAPRC [21, 22]) at University of California, and the Carbon Bond Mechanism series (CBM [41]). Most mechanisms have been derived on base of the same data for reaction rates and products. Besides varieties in the treatment of unknown parameters, an essential difference exists, in how they aggregate organic chemistry into smaller sets of lumped species. Over the years, a plurality of lumping methods have been developed. The most influential class is given by **molecular lumping**: VOCs with similar chemical character (e.g. alkanes, alkanes, aromatics, etc.) are grouped together into lumped variables. A lumped variable then represents the group of VOCs in the mechanism on a molecule-for-molecule basis. The reactivity and rate constants for the lumped variable can be derived by analyzing kinetic and mechanistic data under typical conditions, with the data taken either from literature or from smog chamber experiments. Differences in parameter values can be accounted for by means of reactivity or molar weighting, as it will be outlined later in this section. Molecular lumping has found wide usage in AQM. Most of the chemical mechanisms currently being used in popular air quality models, have been derived through this approach: RADM2 [119], RACM [118], SAPRC-99 [21, 22], SAPRC-07 [23]. An alternative to molecular lumping is presented by the approach of **structural lumping**, where a lumped variable comprises species with similar structure, reactivity and bond type. This concept has found application in the Carbon Bond Mechanism CBM-Ex [41], where carbon atoms are grouped according to their bonding: single bonded, fast doubly bonded, slow doubly bonded, etc..

Both above introduced methods are severely limited: They present frameworks, to derive highly tailored reduced models, designed to fit the concentration profiles of certain species of interest only, but not necessarily of all species. Even though lumping has been used in atmospheric chemical kinetics for decades, criticizers have been despising it as a semi-empirical method, especially as according to Li an Rabitz [65], "There is no known *a priori* way to determine the lumping scheme.". In response to this criticism, recent developments adopt a **kinetic lumping** approach, whereat species are aggregated in a timescale aware fashion. Representative methods are the DCAL method by Li and Rabitz [67, 68] and the APLA algorithm by Djouad and Sportisse [29].

In the following, the mathematical approach to lumping is outlined briefly, along with the principle of reactivity weighting. For a more detailed description, see Li and Rabitz [65, 66], Wang et al. [132] or Whitehouse et al. [137].

5.2.1 Mathematical approach to linear lumping

Let the chemical kinetics of a reaction system with s species be described by

$$\frac{\partial c}{\partial t} = f(c), \quad c(0) = c_0, \tag{5.2.1}$$

with the concentration vector $c = c(t) \in \mathbb{R}^s$ and the vector function $f(c) \in \mathbb{R}^s$. Suppose, the $s-$dimensional system can be reduced to an $\hat{s}-$dimensional lumped system with $0 < \hat{s} \leq s$. Let the temporal evolvement of the lumped species be defined by a vector function $\hat{f} : \mathbb{R}^{\hat{s}} \to \mathbb{R}^{\hat{s}}$, with

$$\frac{\partial \hat{c}}{\partial t} \quad = \quad \hat{f}(\hat{c}), \tag{5.2.2}$$

whereat $\hat{c}$ is the vector of lumped species, defined as a function of c, i.e.

$$\hat{c} = m(c), \tag{5.2.3}$$

with $m : \mathbb{R}^s \to \mathbb{R}^{\hat{s}}$. We restrict here to a linear function $m(c)$, i.e. the lumped concentration vector is a linear combination of the components of the original concentration vector, i.e.

$\hat{c}_i = a_{i,1}c_1 + a_{i,2}c_2 + ... + a_{i,s}c_s$ for $i = 1, \dots, \hat{s}$. Then, a constant lumping matrix $M \in \mathbb{R}^{\hat{s}\times s}$ exists and Eq. (5.2.3) can be simplified to $\hat{c} = Mc$. We insert this relation into Eq. (5.2.1). Consequently, the temporal change for the lumped species $\hat{c}$ is given by

$$\frac{\partial \hat{c}}{\partial t} = Mf(c). \tag{5.2.4}$$

According to Li and Rabitz [65], the lumped system is *exact*, if $Mf(c)$ can be expressed as a function of $\hat{c}$, i.e. $c = \overline{M}\hat{c}$ exists, with the generalized inverse $\overline{M}$ satisfying $M\overline{M} = \mathbb{I}_{\hat{s}}$ and $\mathbb{I}_{\hat{s}}$ being the $\hat{s}-$dimensional real identity matrix. Then, the residual between the lumped species vector $\hat{c}$, mapped onto the full variable space and the full vector c vanishes, i.e.$\left|c - \overline{M}\hat{c}\right| = 0$ and it is

$$\hat{f}(\hat{c}) = Mf(\overline{M}\hat{c}) \Leftrightarrow f(c) = f(\overline{M}Mc).$$

Exact lumping hence equates a representation of c in the basis $V = \{v_1, v_2, ..., v_s\}$

$$c = \sum_{i=1}^{\hat{s}} \hat{c}_i v_i,$$

with V spanned by the columns of $\overline{M}$

$$v_i = \left(\overline{M}\right)_{1:n,i} \quad \forall 1 \le i \le \hat{s}.$$

In the linear case (which is commonly practiced), the challenge in lumping means finding a suitable transformation matrix M along with a unique inverse $\overline{M}$. For decades, the invertible (in a generalized sense) mapping M and $\overline{M}$ has been based on expert knowledge about the chemical species' character, structure or reactivity characteristics. Recently, more formal methods have come up, that all adopt a kinetic approach and exploit the system's timescales. For a profound theory, we refer to Li and Rabitz [65, 66]. Such conditions have been described and employed for example by Li and Rabitz [67, 68], Wang et al. [132] and Djouad and Sportisse [29].

If a chemical species is represented by a lumped model species, the original species is forced to react at the reaction rate of the model species - which in general does not equate its individual reaction rate. Lumping adds an error to the reduced model, which can be diminished by the principle of reactivity weighting, as introduced by Middleton et al. [89]. Emissions of individual compounds are weighted with the ratio F, that describes the ratio of the reacted fraction of the emitted vs. the reacted fraction of the model species,

$$F = \frac{1 - \exp(-k_{\text{HO, emitt.}}\, A)}{1 - \exp(-k_{\text{HO, Model}}\, A)}$$

with the rate constant of the emitted species $k_{\text{HO, emitt.}}$ for the reaction with HO , the rate constant of the model species $k_{\text{HO, Model}}$ for the reaction with HO and A representing a daily average integrated HO concentration, which depends on the conditions of the model simulation. This method can be considered as a parametrization of the solution to the lumping problem by means of a constant value for the concentration of HO. A more dynamic alternative is presented by the concept of hybrid reactivity weighting (cf. Makar et al. [85]): The integrated HO concentration is not considered as a parameter, but as a an additional pseudo-species. This leads to a direct altering in the differential equations at each time-step. The lumping error has proven to be very small, typically within the range of the accuracy of the solver for most species. A major drawback of this method is its strong dependency on relative emissions for the different VOC compounds.

5.2.2 Qualification as a Coarse Propagator

In principle, using a lumped model as a coarse propagator within the parareal algorithm is possible. One minor technical challenge is given by the fact, that the dimensions of the models used for the fine and the coarse propagator do not coincide anymore. This challenge can be managed with the construction of mapping operators between the coarse and the fine model as proposed in Samaey et al. [62].

The scheme, that is being considered in this work (RADM2 [119]), is already a highly condensed model: Hundreds of VOCs have been aggregated into only 63 chemical species. Certainly it is possible, to aggregate the system into an even smaller number of lumped variables. This however requires a high degree of expertise in the chemical system for choosing an adequate invertible mapping M and $\overline{M}$. Lumping does not provide a generic model reduction framework.

The result of a lumping procedure will be a lower-dimensional ODE system, which is not necessarily less stiff. However, the overall computational effort for the solution of the chemical system is mainly dominated by the stiffness of a system, since it determines the size of the time-steps and hence the total number of time-steps. The computational effort for the solution of one time-step decreases with the number of chemical species involved. The ratio is strongly influenced by the chemical character of the remaining system: Solving an $s-$dimensional, stiff system is not necessarily faster than solving a completely uncoupled $m-$dimensional system with $m > s$. It is therefore hard to determine the influence of a reduction in the number of unknowns on the computing time a priori. A reduction in computing time can in any case be expected, when tackling the system's stiffness. This can be accomplished by a separation of timescales. Such methods will be presented in the following section.

5.3 Quasi-Steady State Approximation (QSSA)

Similarly as for lumping methods, the key idea of the QSSA [47] is to reduce the number of unknowns within a chemical kinetics system. The QSSA bases on a separation of timescales. Highly reactive species are treated as being in a quasi-steady state (sometimes also denoted as quasi stationary state), i.e. their temporal development is based on the timescales of the slowly reacting species. Therefore, they are removed from the original ODE system and expressed though algebraic relations in terms of the slower species. The result is a coupled system of differential algebraic equations, composed of a lower dimensional ODE system and a set of algebraic constraints. Typically, the fast variables are also the source of the problem's stiffness. Then, removing the fast species from the ODE system also leads to a reduction in stiffness for the ODE system. QSSA reduced models can also be interpreted as sets of lumped species with sets of algebraic constraints. The key point of the approach is the partitioning into slow and fast dynamics, which equates the definition of the lumping matrix M under additional constraints. A theoretical base of the QSSA is provided by the singular perturbation theory (cf. Tikhonov et al. [120]), which will be outlined briefly in advance.

5.3.1 Singular Perturbation Theory

Let the chemical kinetics of a reaction system with n species be described by Eq. (5.2.1). We assume, the system can be partitioned into a slow part $c_{\text{slow}} \in \mathbb{R}^{s_{\text{slow}}}$ and a fast part $c_{\text{fast}} \in \mathbb{R}^{s_{\text{fast}}}$ with $s_{\text{slow}} + s_{\text{fast}} = s$. We rewrite Eq. (5.2.1) in a singular perturbation form

$$\frac{\partial c_{\text{slow}}}{\partial t} = g(c_{\text{slow}},\, c_{\text{fast}}), \quad \mu \frac{\partial c_{\text{fast}}}{\partial t} = h(c_{\text{slow}},\, c_{\text{fast}}), \tag{5.3.1}$$

with the right-hand side function of Eq. (5.2.1) split into the functions $g(\cdot,\cdot) \in G \subseteq \mathbb{R}^{s_{\text{slow}}}$ and $h(\cdot,\cdot) \in H \subset \mathbb{R}^{s_{\text{fast}}}$ with $f = (g,\, h) \in \mathbb{R}^s$ and a small asymptotic parameter $\mu \in \mathbb{R}$ and $\mu > 0$. We assume, that equation system (5.3.1) has uniquely defined solutions c_{slow} and c_{fast} and $g(\cdot,\cdot)$ and $h(\cdot,\cdot)$ are continuously differentiable in c_{slow} and c_{fast} in G and H. In the singular limit $\mu = 0$ we obtain a *degenerate system*

$$\frac{\partial c_{\text{slow}}}{\partial t} = g(c_{\text{slow}},\, c_{\text{fast}}), \quad c_{\text{fast}} = \psi(c_{\text{slow}}), \tag{5.3.2}$$

where $c_{\text{fast}} = \psi(c_{\text{slow}})$ is a solution of the system of algebraic equations $h(c_{\text{slow}},\, c_{\text{fast}}) = 0$ and denoted a *root*. In the general case, several roots can exist. The question now is, which root has to be chosen, such that the solution to the equation system (5.3.2) is close to the solution of system (5.3.1).

Theorem 6 (Tikhonov's theorem)**.** *We define a root $c_{\text{fast}} = \psi(c_{\text{slow}})$ to be stable in the domain G, if for all points $c_{slow} \in G$ we have*

$$\frac{\partial h(c_{slow,}\psi(c_{slow}))}{\partial c_{fast}} < 0.$$

For $\mu \to 0$, the solution of the original system (5.3.1) approaches the solution of the degenerate system (5.3.2), if $c_{fast} = \psi(c_{slow})$ is a stable root of the degenerate system (5.3.2).

For a proof, see Tikhonov [120].

As a consequence of Tikhonov's theorem and for sufficiently small μ, system (5.3.1) can be approximated by the degenerate system (5.3.2).

5.3.2 Model Reduction Procedure

In a first step, species are selected, that are considered to be in a quasi-steady state. The identification of QSS species can be accomplished in many ways, as will be outlined later.

Once the QSS species are selected, their right-hand-side terms in Eq. (5.2.1) are set to zero. This leads to a differential algebraic system of equations (DAE),

$$\frac{\partial c_{\text{slow}}}{\partial t} = g(c_{\text{slow}},\ c_{\text{fast}}), \tag{5.3.3}$$

$$0 = \frac{\partial c_{\text{fast}}}{\partial t} = h(c_{\text{slow}},\ c_{\text{fast}}). \tag{5.3.4}$$

The second equation implicitly defines an algebraic relation ψ_{QSSA} for the concentrations of the fast species c_{fast} in terms of the slow ones $c_{\text{fast}} = \psi_{\text{QSSA}}(c_{\text{slow}})$,

$$h(c_{\text{slow}},\ \psi_{\text{QSSA}}(c_{\text{slow}})) = 0. \tag{5.3.5}$$

5.3.3 Selection of QSS Species

The key to the success of the QSSA is the proper selection of the QSS species. This can for example be based on a priori expert knowledge about chemical species. An automatized and more common approach is based on comparing their production and destruction terms, which is comparable for very fast species. A more sophisticated method for the selection of QSS species is based on the error induced by the approximation (c.f. Turányi et al. [124], Whitehouse et al. [137]). An expression for the instantaneous, local error of the QSS approximation in the $i-$th species Δc_i, i.e. the concentration difference between the solutions of the full ODE Eq. (5.2.1) and the reduced DAE system (5.3.3-5.3.4) $\Delta c_i = |c_{i,\text{QSSA}} - c_i|$, has been derived by Turányi et al. [124]:

$$\Delta c_i = \left| \frac{\partial c_i}{\partial t} \frac{1}{J_{ii}} \right|,$$

with J_{ii} being the $i-$th diagonal element of the Jacobian matrix. If the error remains small throughout the simulation, this species can be considered as a valid QSS species [124].

Alternatively, the choice of the QSS species can also be guided by Computational Singular Perturbation (CSP) theory, as introduced by Lam and Goussis [59]. The basic idea of the CSP method is a decoupling of the right-hand-side term in Eq. (5.2.1) according to fast and slow components. Characteristic timescales are identified through an eigenvalue analysis of the Jacobian of the right-hand-side term. Individual reaction steps are then associated with characteristic timescales and grouped into reaction groups. A CSP reduced model equates a mechanism consisting of virtual reaction steps, with rates as linear combinations of the original reaction steps of the mechanism. This technique has been utilized successfully for the identification of steady state species in atmospheric chemistry by Neophytou et al. [95], Løvås et al. [76] and Mora-Ramriez and Velasco [94]. Løvås et al. [76] employed the CSP method for the reduction of the Regional Atmospheric Chemistry Mechanism (RACM) [118], which originally contains 77 species and 237 reactions. Reduced mechanisms with 16 steps and 56 of the 77 species in steady state produced excellent results in terms of accuracy for most species and the scenarios considered.

5.3.4 Solving the Reduced Model

The DAE system (5.3.3-5.3.4) can be solved with a numerical scheme such as DASSL (Differential-Algebraic System Solver [98]), but can be very time consuming, since it represents implicit nonlinear equations. No great gain in computing time in comparison to the solution of the original system is expectable. Computationally more promising concepts base on the decoupling of the two equations, i.e. by solving Eq. (5.3.4) explicitly and independent from Eq. (5.3.3) with fixed values for c_{slow}. The concentrations of the QSS species are then substituted into the ODEs for the slow species. Such predictor-corrector type numerical schemes have for example been proposed by Young and Boris [140] and Jay et al. [50]. Sandu et al. [109, 110] provided a detailed comparison of different solvers applied to atmospheric chemical kinetics. They showed, that the performance of QSS solvers is not competitive with specialized stiff solvers like the Rosenbrock schemes for their investigated atmospheric chemistry scenarios.

5.3.5 Qualification as a Coarse Propagator

Due to the poor performance results in comparison to Rosenbrock solvers [109, 110], QSSA approaches do not qualify as suitable coarse propagators within the parareal algorithm. Another major drawback of the QSSA method is, that the sharp differentiation between fast and slow species in practice often is not feasible, since many species typically are involved in both fast and slow processes. The QSSA then indeed leads to a reduction in the size of the ODE system, but not necessarily also to a reduction in stiffness. The following method of intrinsic low dimensional manifolds overcomes this idea, by distinguishing between fast and slow processes instead of fast and slow species.

5.4 Intrinsic Low Dimensional Manifold (ILDM) Method

The key idea of the ILDM (c.f. Maas and Pope [81]) method is to describe the system's dynamics on the lowest-dimensional attracting *slow manifold*. Lower dimensional manifolds in chemical systems have been studied by several authors, e.g. Roussel and Fraser [106]. The latter investigated trajectories in the concentration phase space for different initial conditions for enzyme kinetic systems. They found the trajectories to be attracted from smooth hyper-surfaces for all initial conditions and depicted the relaxation of chemical kinetics as a "cascade through a nested hierarchy of smooth hyper-surfaces (inertial manifolds)" [106]. With the reaction proceeding, more and more fast processes equilibrate, and hence the dimension of the attracting surfaces (i.e. the *slower* manifolds), that contain the reaction trajectory, decreases with elapsing time. The lowest-dimensional attracting slow manifold is denoted as the *intrinsic manifold*. Figure 5.1 shows sample trajectories for the six-dimensional nonlinear reaction system as introduced in Sec. 2.2.2, plotted in the two-dimensional phase space of HO_2 and HO and for a simulation time of 10^3 sec. Independent of the initial values, the concentration of HO follows a slower manifold after some time, which is represented by a one-dimensional line in the two-dimensional projection of the six-dimensional reaction system depicted in Fig. 5.1. Having collapsed to the lower dimensional manifold, the concentration of HO_2 can be

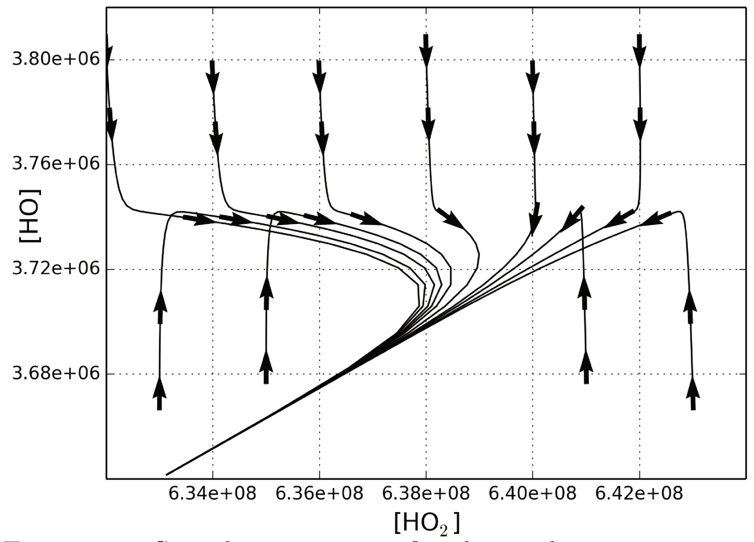


Figure 5.1: Sample trajectories for the nonlinear reaction system as introduced in Sec. 2.2.2, plotted in the phase space of HO_2 and HO for a simulation time of 10^3 sec. Entities on both axes both are in molecules/cm^3. Independent of the initial values, all trajectories are attracted by the same one-dimensional manifold.

expressed as a function of HO and vice versa, hence the number of global degrees of freedom is reduced by one.

Same as for the QSSA method, the basic idea of the ILDM method is a decoupling of fast from slow timescales. Fast timescales are similarly expressed in terms of algebraic relations of the slow ones. Different than in the QSSA method, timescales are not associated with individual species, but with fast and slow processes. In order to distinguish these processes, the system is transformed to the space of *modes* by a projection into the basis of eigenvectors of the Jacobian of the source term $f(c)$ given at some time t for $c = c(t)$. Each mode is associated with one single timescale. Slow modes are assumed to constitute a lower-dimensional manifold, onto which the system collapses. A reduced model is constructed, taking only those modes into account. Since fast modes are filtered out, the reduced model will be less stiff and lower in dimension than the original system, given by Eq. (5.2.1). During the solution process, an ODE solver computes the solution for the reduced model. The total state of the full chemical system, including also fast timescales, can be reconstructed at any time. The calculation of the fast modes' contribution, parametrized as functions of the slow ones, however is computationally intensive. Significant computational savings during run-time can be achieved from their a priori calculation: Fast modes are calculated for a lattice of possible input values for the slow modes and stored in look-up tables. During run-time, the full system can easily be recalled for any slow mode by accessing and interpolating from the respective look-up table.

5.4.1 Theory

The theory behind the ILDM shall be outlined in the following. For a detailed overview, we refer to the comprehensive work by Maas et al. [79–82]. Again, we consider the system of

ODEs defined by Eq. (5.2.1) with initial conditions $c(0) = c^0$. We disturb the system at a particular point c^{f_x} with a small perturbation $\Delta c := c - c^{f_x}$. After a Taylor series expansion and substitution, it can be seen, that a locally linear representation of the motion of the perturbation is given by

$$\Delta c(t) = e^{\int_0^t J(\tau)\, d\tau}\, \Delta c^0, \tag{5.4.1}$$

with the Jacobian $J(t) = \partial f(t)/\partial c \in \mathbb{R}^{s\times s}$ and an initial perturbation $\Delta c^0 = c^0 - c^{f_x}$. Each eigenvalue of $J(t)$ is associated with a timescale or mode of the locally linear solution and effectively gives the speed of relaxation of a small perturbation of the system. Large negative eigenvalues (or in the complex case eigenvalues with large negative real part) correspond to rapidly equilibrating processes, that are not necessary to represent long-time dynamics. For simplicity, we assume the Jacobian is constant in time in the following. Then, Eq. (5.4.1) reduces to

$$\Delta c(t) = e^{J\cdot t}\Delta c^0.$$

We denote the eigenvalues of J by $\lambda_1, \lambda_2, \ldots, \lambda_s \in \mathbb{C}$ and corresponding eigenvectors by $v_1, v_2, \ldots, v_s \in \mathbb{C}^s$. We introduce the matrix of eigenvectors V and its inverse $V^{-1} \in \mathbb{C}^{s\times s}$ by

$$V := \begin{pmatrix} | & | & & | \\ v_1 & v_2 & \ldots & v_s \\ | & | & & | \end{pmatrix} \in \mathbb{C}^{s\times s} \text{ and } V^{-1} := \begin{pmatrix} — & \tilde{v}_1 & — \\ — & \tilde{v}_2 & — \\ & \vdots & \\ — & \tilde{v}_s & — \end{pmatrix} \in \mathbb{C}^{s\times s}.$$

Then it holds $J = V\Lambda V^{-1}$, where the diagonal matrix of eigenvalues is denoted by $\Lambda := \operatorname{diag}(\lambda_1, \lambda_2, ..., \lambda_s) \in \mathbb{C}^{s\times s}$. A locally linear representation of the development of perturbations can then be written as

$$\Delta c(t) = V e^{\Lambda t} V^{-1} \Delta c^0.$$

Since the matrices V and V^{-1} in general don't have diagonal structure, the off diagonal terms represent the couplings of species and their contributions to different timescales. Hence, each species grows/decays to multiple timescales. From this, a new set of variables can be constructed from linear combinations of the original ones: We define a vector $z \in \mathbb{C}^s$, that is the representation of c in the basis of eigenvectors of the Jacobian J. A perturbation in z,

$$\Delta z := z - z^{f_x}$$

will then grow or decay exponentially with

$$\Delta z(t) = e^{\Lambda t}\Delta z^0, \tag{5.4.2}$$

depending on the size of the eigenvectors. Since Λ is a diagonal matrix, a perturbation in the $i-$th component of Δz grows/decays according to one single timescale τ_i with $\lambda_i = \tau_i^{-1}$,

$$\Delta z_i(t) = e^{\lambda_i t}\Delta z_i^0. \tag{5.4.3}$$

Ordering the eigenvalues upon the size of their real part from large and negative to large and positive allows to identify fast and slow modes. From Eq. (5.4.3) it can directly be seen, that an initial perturbation Δz_i^0 decays, if $\mathrm{Re}(\lambda_i) < 0$. Very fast modes, i.e. modes with very small τ_i, correspond to rapidly equilibrating processes and can be identified by eigenvalues with large, negative real part, i.e. $\mathrm{Re}(\lambda_i) \ll 0$. Eigenvalues with $\mathrm{Re}(\lambda_i) \approx 0$ represent constant modes. Eigenvalues with positive real part correspond to non-equilibrating modes, since a perturbation will grow in time. Complex eigenvalues in general represent an oscillatory behavior. Depending on the absolute value of their imaginary part, they are typically treated as real eigenvalues, in practice.

Since the system responds according to single timescales, we consider the components of z as the modes of the system. The transformation matrix V^{-1} shows, how each species contributes to the modes associated with each eigenvalues. We rewrite Eq. (5.2.1) in terms of the new variable $z := V^{-1}c$,

$$\begin{aligned} V^{-1}\frac{\partial c}{\partial t} &= V^{-1}f(c) \\ \frac{\partial z}{\partial t} &= V^{-1}f(c) \\ &=: F(c). \end{aligned}$$

If an individual mode of the s-dimensional phase space has been attracted by an $s-1$-dimensional, slower manifold, we say that it has collapsed to the intrinsic manifold. A collapsed mode will not change in time anymore, i.e.

$$0 = \frac{\partial z_i}{\partial t} = \tilde{v}_i f(c),$$

with $\tilde{v}_i$ denoting the $i-$th row of the inverse of V^{-1}. Ordering the system according to the real parts of the eigenvalues (as described above), allows to divide the vector of modes into a fast and a slow part, $z_{\mathrm{slow}} \in \mathbb{C}^{s_{\mathrm{slow}}}$ and $z_{\mathrm{fast}} \in \mathbb{C}^{s_{\mathrm{fast}}}$ with $s_{\mathrm{slow}} + s_{\mathrm{fast}} = s$,

$$z = \begin{pmatrix} z_{\mathrm{slow}} \\ z_{\mathrm{fast}} \end{pmatrix} = \begin{pmatrix} V_{\mathrm{slow}}^{-1}c \\ V_{\mathrm{fast}}^{-1}c \end{pmatrix},$$

where $V_{\mathrm{slow}}^{-1} \in \mathbb{C}^{s_{\mathrm{slow}}\times s}$ and $V_{\mathrm{fast}}^{-1} \in \mathbb{C}^{s_{\mathrm{fast}}\times s}$ denote the first s_{slow} and the last s_{fast} rows in V^{-1}, respectively. Assuming the fast modes to be in local equilibrium, it is

$$0 = \frac{\partial z_{\mathrm{fast}}}{\partial t} = V_{\mathrm{fast}}^{-1}f(c). \tag{5.4.4}$$

Equation (5.4.4) describes the characteristic property of the fast modes, that serves as definition of the ILDM.

We can further split the vector of concentrations $c = Vz$ in parts arising from slow and parts arising from fast dynamics, $c_{\mathrm{slow}} \in \mathbb{R}^s$ and $c_{\mathrm{fast}} \in \mathbb{R}^s$,

$$\begin{aligned} c &= Vz \\ &= \begin{pmatrix} V_{\mathrm{slow}} & V_{\mathrm{fast}} \end{pmatrix} \begin{pmatrix} z_{\mathrm{slow}} \\ z_{\mathrm{fast}} \end{pmatrix} \\ &= \underbrace{V_{\mathrm{slow}} z_{\mathrm{slow}}}_{=c_{\mathrm{slow}}} + \underbrace{V_{\mathrm{fast}} z_{\mathrm{fast}}}_{=c_{\mathrm{fast}}}, \end{aligned} \tag{5.4.5}$$

with $V_{\text{slow}} \in \mathbb{R}^{s \times s_{\text{slow}}}$ composed of the eigenvectors in V for the s_{slow} eigenvalues with smallest absolute values, and $V_{\text{fast}} \in \mathbb{R}^{s \times s_{\text{fast}}}$ composed of the s_{fast} eigenvalues with biggest absolute values,

$$\begin{aligned} V &= \left(\begin{array}{cc} \underbrace{\begin{array}{ccc} | & & | \\ v_1 & \dots & v_{s_{\text{slow}}} \\ | & & | \end{array}}_{V_{\text{slow}}} & \underbrace{\begin{array}{ccc} | & & | \\ v_{s_{\text{slow}}+1} & \dots & v_s \\ | & & | \end{array}}_{V_{\text{fast}}} \end{array} \right), \\ &= \left(\begin{array}{cc} V_{\text{slow}} & V_{\text{fast}} \end{array} \right). \end{aligned}$$

Note, that in general $(V_{\text{slow}})^{-1} \neq V_{\text{slow}}^{-1}$. Inserting Eq. (5.4.5) in the definition of the ILDM, Eq. (5.4.4) yields an implicit definition of $z_{\text{fast}} = \phi(z_{\text{slow}})$,

$$0 = V_{\text{fast}}^{-1} f(V_{\text{slow}} z_{\text{slow}} + V_{\text{fast}} z_{\text{fast}}) = V_{\text{fast}}^{-1} f(V_{\text{slow}} z_{\text{slow}} + V_{\text{fast}} \phi(z_{\text{slow}})). \tag{5.4.6}$$

Since the original system of ODEs may be nonlinear, V is not necessarily constant in time and with varying $c(t)$. As pointed out earlier, we stick to linear problems here for the sake of simplicity. Then it $J =$ const. and following also $V =$ const.. Then, Eq. (5.4.6) can also be written in terms of $c_{\text{slow}} = V_{\text{slow}} z_{\text{slow}}$ and

$$\begin{aligned} c_{\text{fast}} &= V_{\text{fast}} z_{\text{fast}} = V_{\text{fast}} \phi(z_{\text{slow}}) \\ &= V_{\text{fast}} \phi(V_{\text{slow}}^{-1} c_{\text{slow}}) \\ &=: \psi_{\text{ILDM}}(c_{\text{slow}}) \end{aligned}$$

and therefore it is also

$$0 = V_{\text{fast}}^{-1} f(c_{\text{slow}} + \psi_{\text{ILDM}}(c_{\text{slow}})). \tag{5.4.7}$$

Same as for the previously introduced QSSA method, were we have ended up with an algebraic relation ψ_{QSSA} (see Eq. (5.3.5)) for the contributions of the fast species c_{fast} in terms of the slow ones $c_{\text{fast}} = \psi_{\text{QSSA}}(c_{\text{slow}})$, we have now found an implicit definition for $c_{\text{fast}} = \psi_{\text{ILDM}}(c_{\text{slow}})$.

5.4.2 Model Reduction Procedure

To reduce a complex chemical kinetics model with the ILDM only two input information are essential: The detailed chemical mechanism and the number of degrees of freedom in the simplified scheme s_{ILDM} that define the size of the intrinsic manifold.

The procedure starts with an eigenvalue analysis, whereat the eigenvalues are ordered with descending absolute values of their real parts. We associate an eigenvalue with one timescale each and assume that the system is equilibrated with respect to the fastest timescales. During the integration, only slow processes are explicitly considered. Instead of approximating a solution $c \in \mathbb{R}^s$ at time $t = T_{\text{end}}$ to Eq. (5.2.1), we only approximate a solution $c_{\text{slow}} \in \mathbb{R}^s$ at time $t = T_0$ with respective *slow* initial conditions, i.e. a projection of some initial conditions into the space of slow modes. The resulting initial value problem does not inhere fast processes, hence it presents a less stiff system, which allows to use significantly bigger time-steps during the integration, or respectively less small time-steps if an adaptive time-stepping scheme is used. At final time T_{end} we have to incorporate the contributions of the fast modes, parametrized by means of the slow ones, $c_{\text{fast}} = \psi_{\text{ILDM}}(c_{\text{slow}})$. This algebraic relation is given implicitly by Eq. (5.4.7). Solving the latter however is a tricky issue, both from an implementational and a computational point of view. Equation (5.4.7) characterizes the solution implicitly and based on a (in general) nonlinear system. Since $f(\cdot) \in \mathbb{C}^s$, but $V_{\text{fast}}^{-1} \in \mathbb{C}^{s_{\text{fast}} \times s}$, it further represents a system of s_{fast} equations with $s > s_{\text{fast}}$ unknowns, hence it is under-determined. One common approach in literature and which we will make use of is to amend the problem with additional parametrization equations and thereby regularize the original problem. Once a suitable parametrization has been found, the closed system can be solved numerically. As f depends nonlinearily on c, we use Newton's method to approximate a solution $x \in \mathbb{R}^s$ of $V_{\text{fast}}^{-1} f(x) = 0$, which implicitly gives the solution for $c_{\text{fast}} = \psi_{\text{ILDM}}(c_{\text{slow}})$, as $x := c_{\text{slow}} + \psi_{\text{ILDM}}(c_{\text{slow}}) \in \mathbb{R}^s$. Newton's Method is an iterative approach to find the roots of a real-valued function $\Phi(x)$: Find $x \in \mathbb{R}^s$ so that $\Phi(x) = 0$. With an initial guess x_0, we iterate over i and repeat

$$x_{i+1} := x_i - [\nabla\Phi(x_i)]^{-1}\,\Phi(x_i)$$

until convergence. Here, we seek the root of $\Phi(x) := V_{\text{fast}}^{-1}\, f(x)$. Assuming again, that V_{fast}^{-1} is constant, the Jacobian is $\nabla\Phi(x) = V_{\text{fast}}^{-1}\nabla(x)$. As an initial guess, we take $x_0 = c_{\text{slow}}$.

5.4.3 Parametrization of the ILDM

By now, we have not taken into account, that both $V_{\text{fast}}^{-1} \in \mathbb{R}^{s_{\text{fast}} \times s}$ and $\nabla\Phi(x) = V_{\text{fast}}^{-1}\nabla(x) \in \mathbb{R}^{s_{\text{fast}} \times s}$ are not of full rank and therefore not invertible. Eq. (5.4.7) represents a set of s_{fast} equations for s unknowns with $s \geq s_{\text{fast}}$. We therefore add s_{slow} additional parametrization equations, to restrict the system's solution space:

$$\begin{pmatrix} V_{\text{fast}}^{-1}\, f(x) \\ P(x) \end{pmatrix} \quad = \quad 0, \tag{5.4.8}$$

and solve

$$x_{i+1} = x_i - \begin{bmatrix} V_{\text{fast}}^{-1}\, \nabla f(x_i) \\ \nabla P(x_i) \end{bmatrix}^{-1} \begin{pmatrix} V_{\text{fast}}^{-1}\, f(x_i) \\ P(x_i) \end{pmatrix}.$$

The choice of the parametrization equations influences existence and uniqueness of a solution to Eq. (5.4.8), but does not affect the manifold itself. Choosing adequate parametrization equations is one of the crucial parts of the ILDM method.

Reaction Progress Variables One typical approach for adding additional parametrization equations found in literature, is the usage of specific element mole numbers, concentrations or even linear combinations of them, see e.g. [81, 82]:

$$P(x) = c_{\text{SPC}} - \gamma_{\text{SPC}},$$

with c_{SPC} being the actual species concentration of species SPC and γ_{SPC} being a reference value on the manifold. This reference value could for example be the value of species SPC at steady state. The species, that are chosen to parametrize the manifold, are called reaction progress variables. No matter, how these reaction progress variables are chosen, the manifold will be the same, as the reaction progress variables only allow for *some* parametrization. The manifold can now be constructed by varying the values of the reaction progress variables. By solving Eq. (5.4.8) for each new input value, we construct the ILDM. As a result, we get an $s_{\mathrm{slow}}-$dimensional table for each of the s_{fast} species that holds its values as a function of the reaction progress variables. Once the manifold has been constructed, we can store it in tables and use it in subsequent simulations. During the simulation, the respective values can be looked up in the tables. This way, the time-intensive calculation of the ILDM is omitted during the simulation. Instead of solving a stiff problem, one solves a less stiff problem, and updates fast mode contributions by looking up values in the update table. The price of this relief is the a priori construction of the manifold, which - depending on the requested detailedness - can be very time-consuming.

Several approaches exist, towards an efficient construction of the manifold. This includes the choice of the mesh, whose nodal points define the discrete values for the reaction progress variables, for which we solve Eq. (5.4.8). As the state vector c is bounded by physical constraints, e.g. temperature and pressure, the manifold will be bounded, too. This characteristics can be exploited during the construction manifold and the mesh can be limited in preface. Typically, the parametrization is constant during the construction. The approach inheres difficulties: In regions, where the reaction progress variables are constant, but the parametrized species are not, the parametrization will be ill-conditioned. This can be circumvented by a local adaptation of the parametrization. Aforementioned and more approaches for an efficient construction of the manifold can be found in [79].

Figure 5.2 visualizes the parametrization using reaction progress variables by means of the six-dimensional nonlinear example, introduced in Sec. 2.2.2. As reaction progress variables HO and HO_2 were chosen. The ILDM was calculated on an equidistant mesh with 441 nodal points, whereat HO ranges from $3.0 \cdot 10^5 - 3.9 \cdot 10^6$ molecules/cm^3 and HO_2 from $5.6 \cdot 10^8 - 6.8 \cdot 10^8$ molecules/cm^3. The colored lines show the development of the specific mole number of O_3 during a simulation time of 10^3 sec, plotted as a function of the specific mole numbers of HO and HO_2. Each of the five runs was started with a slightly modified initial state. Independent of their initial values, all trajectories are attracted by a slower manifold, represented as a lattice. Before a trajectory collapses onto this manifold, it is depicted in yellow, after the collapse in pink. For convenience, the shadow of a trajectory before it has collapsed (yellow) onto the slower manifold is depicted as a gray dashed line. Having collapsed, the trajectory will remain close to the manifold for all times. With elapsing simulation time, all trajectories are further attracted by a lower-dimensional manifold, that is represented as a blue line here. All trajectories approach one another far before equilibrium is reached. In this 3D-projection of the 6-dimensional chemical system, the manifolds depicted as lattice and line appear to be two- and one-dimensional. In fact, they are 5- and 4-dimensional.

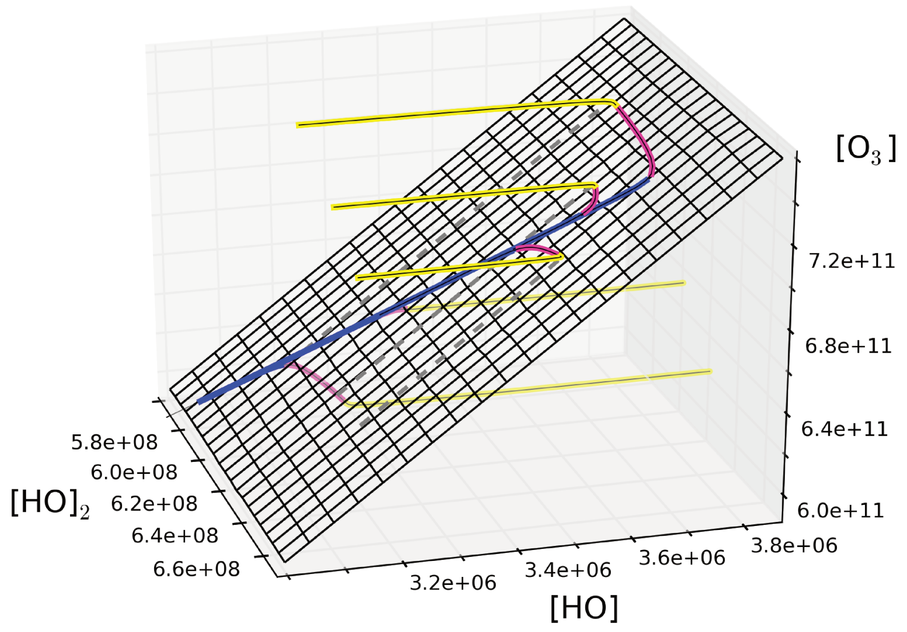


Figure 5.2: Sample trajectories for the nonlinear reaction system as introduced in Sec. 2.2.2, plotted in the phase space of HO_2, HO and O_3 for a simulation time of 10^3 sec. Entities in all axes are molecules/cm^3.

Discussion Parametrization The great success of the ILDM method in combustion simulation stems from its potential to use look-up tables. However, the usage of look-up tables is limited to applications, where the size of the ILDM is very small, typically below 4. For the nonlinear ozone formation scenario from Sec. 2.2.2 a tabulation strategy may be promising. Depending on the choice of parameters, ILDMs with sizes below 3 can be identified. As soon as we switch to more realistic scenarios, as will be the case in Sec. 6, we will find higher dimensional intrinsic manifolds. For those scenarios, tabulation methods are not promising anymore, since the storage demands become paramount. In general, also functional fitting methods can be used, to replace the look-up tables for the parametrized ILDM. The model reduction procedure presented so far is then amended with an additional functional fitting step, that describes the parametrized ILDM as a function of the reaction progress variables. Instead of accessing high-dimensional look-up tables that hold values for the individual species, one has to access a look-up table holding the coefficients of the fitted function and evaluate the latter.

5.4.4 Qualification as a Coarse Propagator

An ILDM-reduced model consists of a less stiff and lower dimensional system of ODEs than the original system Eq. (5.2.1). Since all fast modes are filtered out, the reduced ODE system represents only the change in concentration due to slow modes c_{slow}. It is therefore less stiff than the original system and will be easier and faster to solve numerically. The contributions of the fast modes c_{fast} can be accounted for through an additional update step at any time of the simulation, whereat c_{fast} is implicitly defined by Eq. (5.4.7). Solving the latter however is computationally demanding and should be avoided during run-time of the reduced model for the purpose of a good performance. For small ILDM sizes, the fast mode updates can be calculated a priori over a lattice of possible slow mode contributions c_{slow} and stored in look-up tables, that are being accessed during runtime. Especially in the field of combustion simulation, this technique has become increasingly popular within the last decades. For those applications, very low dimensional manifolds can be found - typically it is $s_{\text{slow}} = 2$ or 3.

While the ILDM method is common use in combustion simulation for more than 20 years, it has not gained popularity in atmospheric chemistry. Lowe et al. [77] and Tomlin et al. [121] investigated the method's potential for several problems arising in atmospheric chemistry, such as the Carbon bond mechanism, which models the formation of tropospheric ozone or the oxidation of butane. For simulated periods of multiple hours up to multiple days, the authors investigated existence and size of intrinsic low dimensional manifolds. All results indicate, that the intrinsic manifolds vary diurnally and depending on the photolysis rates. This poses difficulties for the practical application of the ILDM, since large ILDM sizes lead to high-dimensional look-up tables. In some cases, the memory requirements associated with the look-up tables even exceed the storage potential of respective state-of-the art hardware. For the Carbon bond mechanism, investigated in [77], the lowest dimensional manifold was shown to vary between 2 at night and 9 at daytime. In practice with the RADM2 scheme, this would lead to huge look-up tables: All fast modes have to be tabulated parameterized by the 9 intrinsic, slow modes. In total, $63 - 9 = 54$ tables will be necessary. We assume, the parametrization is carried out at p discrete values over a realistic range for each of the 9 parameterizing species. Each table then contains p^9 scalar values. If storing the parametrization in single precision, the resulting storage demands are $54 \cdot p^9 \cdot 4$ Bytes. Even for a very rough parametrization using a dimension of $p = 12$, the total storage demand already exceeds a TeraByte, for $p = 56$ an ExaByte. Since the dimension of the ILDM varies with daytime, multiple such tables have to be constructed and stored. In its classical form, the ILDM method therefore does not present a promising framework for a reduced model coarse propagator.

The existence of low dimensional intrinsic manifolds can however be exploited to construct low-dimensional representations of the chemical reaction kinetics by means of so called repro-models. Repro-modeling will be outlined in the following section.

5.5 Repro-Modeling

Different than most model reduction approaches, repro-modeling methods do not pursue a reduction of the dimension or the stiffness of the ODEs. Instead, one seeks for functional representations of the time-dependent kinetic change within chemical systems. To this end, for example polynomial functions can be fitted, to map sets of input to sets of output concentrations. The result is an explicit expression for the chemical species after certain integration time. The major drawback of repro-modeling is, that they are only accurate for the conditions, the fitting has been obtained from. However, they have been successfully applied to a wide range of atmospheric chemistry problems:

An early application of repro-modeling in atmospheric chemistry dates back to 1985, when Dunker [30] parameterized a smog mechanism using second-order Taylor expansions to describe the changes in the concentrations. The coefficients of the expansion were calculated by solving sensitivity equations and stored for subsequent use. Using the parametrized functional representation instead of an implicit integration of the full mechanism led to a reduction of computational efforts of over two orders of magnitude, while maintaining an accuracy of approx. 10%. In 1987, Marsden et al. [86] used second-order log-linear functions to parametrize a smog mechanism. The resulting functional representation was approximately $10-20\%$ less accurate than the original model. In 1990, Spivakovsky et al. [115] developed a sophisticated procedure to parametrize a global model for HO−concentrations. Higher-order polynomials were generated with a least-squares fitting procedure. Probability density functions (PDFs) for the lattice of all possible concentrations were used as input data. Ineffective coefficients were finally sorted out, to make the polynomials easier to evaluate. Crucial modifications to this method have been proposed by Turányi [122] in 1994, who parametrized a skeleton model of the oscillating Belousov-Zhabotinsky reaction (c.f. [11] and [141]). Turányi introduced two crucial modifications to Spivakovsky's method: First, he replaced PDFs as input data by box-model simulations with input values in realistic ranges. Second, he used orthonormal polynomials instead of usual polynomials, leading to a simpler and more effective fitting procedure. Section 5.5.1 outlines the basic principles of a general repro-modeling approach as proposed Turányi [122].

An application of polynomial repro-modeling to a realistic 90−species tropospheric chemical mechanism has been presented by Lowe and Tomlin [77]. Since the number of polynomial terms, and therefore also the fitting and simulation time increase with the number of variables, it is desirable to choose the lowest dimensional polynomial grade possible. To this end, the authors first identified the dimension of the intrinsic lowest dimensional manifolds. Following, they showed, that the full model with 90 species can accurately be represented over a wide range of initial conditions using a 9−th grade polynomial repro-model. The choice of a nine-dimensional representation was guided by the existence of low dimensional intrinsic manifolds with sizes varying between 2 (nighttime) and 9 (daytime). Once a system has collapsed onto a lower dimensional manifold, less functional expansions will be required. For each of the 90 species 24 repro-models were fitted by means of a 9−th grade polynomial. Each of the 24 repro-models per species equates a functional representation of the change within one hour of a day with different photolysis conditions. The CPU time required to evaluate the repro-model was approximately 25% below the computing time needed for the box-model simulations. In total, 360 learning model runs were necessary.

An essential challenge within polynomial fitting approaches is, that the number of data sets required for an accurate fitting grows exponentially with the number of chemical species involved. The family of High Dimensional Model Representation (HDMR) [101, 102] methods present a special class of repro-modeling without requiring large numbers of model runs. Since an HDMR expansion is represented by very few component functions, it is further faster to evaluate than a polynomial repro-model. The basic idea is, to express the output of the full chemical model as an expansion of correlated functions. For this approach, the number of input data sets grows polynomially with the number of species. Since its appearance in 1999, the HDMR method has rapidly gained popularity in atmospheric chemical kinetics. Also in 1999, first applications of the HDMR to stratospheric box-models have been made by Rabitz and Aliş [101] and Shorter et al. [113]. In the same year, Wang et al. [134] applied the HDMR method to derive a Fully Equivalent Operational Model (FEOM) for usage in a global chemistry-transport models. Later, Wang et al. [131, 133] proved it in practice: Using HDMR expansions with first-order expansion led to a speed-up of a factor of up to 1 000 compared to an implicit integration of the chemical system. The theoretical base behind the HDMR method will be explained in Section 5.5.2.

5.5.1 Principles of Repro-Modeling

A repro-model can be denoted as a functional representation $F : \mathbb{R}^s \to \mathbb{R}^s$ of the time-dependent kinetic change within a chemical system (given by Eq. (5.2.1)), that maps vectors of given concentrations at time t, $c(t) \in \mathcal{I} \subset \mathbb{R}^s$, to respective output vectors at time $t + \Delta t$ for a given $\Delta t > 0$, $c(t + \Delta t) \in \mathbb{R}^s$:

$$c(t + \Delta t) \quad = \quad F(c(t)).$$

A General Repro-Modeling Algorithm can be stated as follows (c.f. [123]):

1. Select a characteristic time-step size Δt, for which the repro-model shall be valid.
2. Carry out several thousand simulations $\frac{\partial c}{\partial t} = f(c)$ for an integration time of Δt and with differing input concentrations $c(t)$ (that serve as initial values), which are typical for the circumstances, under which the repro-model will be used. Additionally, also the environmental conditions can be varied and serve as input data. They are neglected in the notation for simplicity.
3. Store pairs of input values $c(t)$ and output values $c(t + \Delta t)$ in a database.
4. Fit a function F to map $c(t)$ to $c(t + \Delta t)$.

Once a function F is found, the repro-model can be used instead of an implicit integration of Eq. (5.2.1) subject to the initial conditions $c(t)$, for example for the solution of the chemical reaction kinetics during a splitting interval within an AQM.

5.5.2 High Dimensional Model Representation (HDMR)

High Dimensional Model Representation (HDMR) is a family of methods, that provide hierarchical, functional representations of the input-output relationship of chemical kinetics systems with many input variables. The input-output relationship is typically composed of independent and/or cooperative effects between multiple chemical species. This suggest to express the output $c(t + \Delta t) = F(c)$ as a hierarchical expansion in terms of the input variables,

$$F(c) \quad = \quad F_0 + \sum_{i=1}^{s} F_i(c_i) + \sum_{1 \le i,j \le s} F_{ij}(c_i, c_j) + \sum_{1 \le i,j,k \le s} F_{ijk}(c_i, c_j, c_k) + ... + F_{12...s}(c_1, c_2, ..., c_s) \qquad (5.5.1)$$

The constant zeroth-order term F_0 denotes the chemical system's mean effect. The second term, $F_i(c_i)$, denotes the independent effect of variable c_i, the term $F_{ij}(c_i, c_j)$ denotes the interactive effect of a variation of c_i and c_j, etc.. Finally, the term $F_{12...s}$ denotes the $s-$th order interactive effect of all s input variables on the outputs.

To construct an HDMR expansion, suitable expressions of the component functions $F_{i_1 i_2 ... i_l}(c_{i_1}, c_{i_2}, ..., c_{i_l})$ have to be found with $l = 0, 1, ...s$. They are constructed as optimal choices for the output $F(c)$ for all $c \in \mathcal{I}$, i.e. through a minimization of the functional

$$\min_{f_{i_1 i_2 ... i_l}} \int_{\mathcal{I}} w_{i_1 i_2 ... i_l}(\hat{c}, c) \quad \Big[F(c) - F_0 - \textstyle\sum_{1 \leq i \leq s} F_i(c_i) - \sum_{1 \leq i,j \leq s} F_{ij}(c_i, c_j) \\ - \ldots - \textstyle\sum_{1 \leq i_1, i_2, ..., i_l \leq s} F_{i_1 i_2 ... i_l}(c_{i_1}, c_{i_2}, ..., c_{i_l}) \Big]^2 \; dc,$$

with the weight functions $w_{i_1 i_2 i_3 ... i_l}(\hat{c}, c)$.

A popular method to compute the component functions is presented by the Cut-HDMR method (c.f. [101, 102]). The component functions are determined by evaluating input-output responses relative to a reference point $\hat{c} \in \mathcal{I}$ for a wide range of input values $c \in \mathcal{I}$,

$$\begin{aligned} F_0 &= f(\hat{c}) \\ F_i(c_i) &= f(c_i, \hat{c}^i) - h_0 \\ F_{ij}(c_i, c_j) &= f(c_i, c_j, \hat{c}^{ij}) - F_i(c_i) - F_j(c_j) - F_0 \\ &\cdots \end{aligned}$$

The notation $f(c_i, \hat{c}^i)$ signifies, that all input values are at the reference values of $\hat{c}$, except for the $i-$th concentration of c:

$$f(c_i, \hat{c}^i) \equiv f(\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{i-1}, c_i, \hat{c}_{i+1}, \ldots \hat{c}_s).$$

The input values c are thereby chosen along associated lines, surfaces, sub-volumes,..., i.e. along discrete *cut*s in the input space $\mathcal{I}$ through the reference point $\hat{c}$. The computational effort to learn the component functions scales polynomially with the number of key variables s_{HDMR}, i.e. the number of variables, for which the input values are varied. When taken to convergence, the Cut-HDMR is invariant to the choice of $\hat{c} \in \mathcal{I}$, it is however advisable to choose $\hat{c}$ in the neighborhood of interest in $\mathcal{I}$.

Once the component functions $F_0,\ F_i,\ F_{ij}, \ldots$ have been obtained, they can be used to predict the output behavior of the system for any input value $c \in \mathcal{I}$. To this end, they are stored in low-dimensional look-up tables over the input variables. During run-time, the output value $F(c)$ for any arbitrary point $c \in \mathcal{I}$ can be determined by performing low dimensional interpolation over $F_i(c_i),\ F_{ij}(c_i, c_j), \ldots$. A considerable reduction of the complexity of the HDMR expansion is given by the assumption, that higher-order correlated effects of the inputs are negligible with respect to the outputs. In practice, an expansion up to 1st or 2nd order is typically sufficient. The total number of model runs to construct F_0, F_i and F_{ij} for a 1st/2nd order HDMR is given by

$$\begin{aligned}\#\text{runs}_{\text{1st}} &= 1 + s_{\text{HDMR}} \cdot p, && (5.5.2)\\ \#\text{runs}_{\text{2nd}} &= \#\text{runs}_{\text{1st}} + \frac{s_{\text{HDMR}} \cdot (s_{\text{HDMR}} - 1) \cdot p^2}{2}, && (5.5.3)\end{aligned}$$

for a given number s_{HDMR} of key variables, that each is varied over p discrete value. This equates also the dimension of the look-up table for each of the s species, leading to a total storage demand of $s \cdot \#\text{runs} \cdot 4$ Bytes when using single-precision. For an application to the 63-variable RADM2 scheme with $s = s_{\text{HDMR}} = 63$ and $p = 10$, 631 model runs for a first-order, 281 989 model runs for a second-order HDMR are necessary. Respectively approx. 0.16 MB storage is necessary to hold the look-up table for a first-order expansion, 50 MB for a second-order expansion.

The storage demands can be reduced dramatically, when defining the HDMR as a function of few s_{HDMR} process variables only with $s_{\text{HDMR}} < s$. As the number of component functions decreases with the number of key variables, also the fitting and simulation time decreases. Therefore, it is desirable, to choose a low number of key variables. A species can for example be omitted during the parametrization, if it is constant over the simulation interval $[t,\ t + \Delta t]$ or if it is always initialized with the same initial concentration, e.g. with zero or a prescribed emission.

5.5.3 Construction of the HDMR model

In order to construct an HDMR model, we first need to clarify the purpose of the HDMR expansion, i.e. for which input space $\mathcal{I}$ and which step-size Δt shall it be valid and around which reference point $\hat{c} \in \mathcal{I}$ do we want to construct it. The remaining variable parameters are:

– the degree of the HDMR expansion

– the number of parametrization points p

– the number of key variables s_{HDMR}, to parametrize the repro-model

We showcase some of the variable parameters by means of the Lotka-Volterra problem, that describes the dynamics of biological systems with two interacting species, one acting as prey x and the other one as predator y. The temporal evolution of the populations of prey and predator can be described by a first-order, nonlinear ordinary differential equation system,

$$\frac{\partial x}{\partial t} = \alpha x - \beta xy, \quad \frac{\partial y}{\partial t} = \delta xy - \gamma y, \qquad (5.5.4)$$

where the parameters α, β, γ and δ describe the interaction of the two species. For the following examples, we choose $\alpha = 2.0$, $\beta = 1$, $\gamma = 0.25$, $\delta = 1.0$, $x(0) = 1.0$, $y(0) = 1.0$. Figure 5.2 shows the temporal evolution of the populations of the predator (black) and the prey (blue) within an simulation interval of 30 seconds. The oscillatory interaction between prey and predator populations can also be seen in Fig. 5.1, where sample trajectories are plotted in the phase space of prey and predator. Sample trajectories were created from evenly varying the initial concentrations within an input space of $\mathcal{I} = [0.5,\ 1.5] \times [0.5,\ 1.5]$, depicted as a dark gray box. The respective solution space after a simulation time of $t = 1$ sec is highlighted as a lightly shaded box.

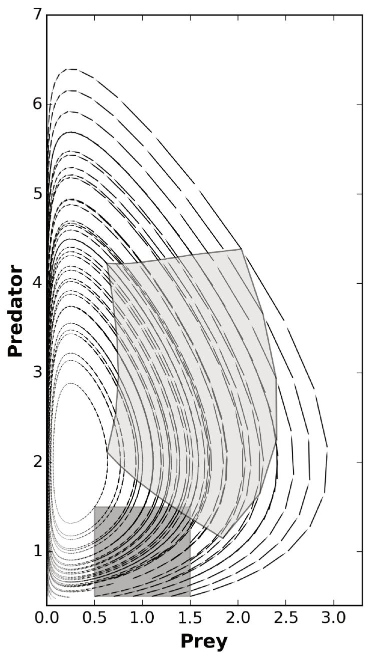


Figure 5.1: Prey and predator populations of the Lotka-Volterra problem.

In the following, we want to construct HDMR repro models with varying parameter choices. In all cases, we choose the input space as $\mathcal{I} = [0.5,\ 1.5] \times [0.5,\ 1.5]$, the reference point as $\hat{c} = (1.0,\ 1.0)$ and $\Delta t = 1.0$ sec. Figure 5.3 shows the input-output responses used for the construction of the zeroth-, first- and second-order component functions F_0, F_i and F_{ij} during the construction of an HDMR repro model for the two dimensional Lotka-Volterra problem and with $p = 4$ parametrization points. In all plots, the input space is depicted as a gray square, the respective output space at $t = \Delta t$ as the light gray area. Black arrows showcase the trajectories, that need to be calculated for the construction of the respective component function.

The left plot showcases the construction of the zeroth-order term, that is defined by the input/output response of the reference point, here marked with a dark blue dot. For the construction of the first-order term, each one species is held at the reference state, while the other species is varied along the so called cutting lines over p values. The associated cutting lines are depicted as blue lines, respective initial values are marked as circles. In total $s \cdot p = 8$ trajectories need to be calculated for the construction of the first-order component functions F_i. For the construction of the second-order term, both species are varied at the same time over the cutting surface, here indicated with the black grid. For the construction of the second-order component functions F_{ij} another $\frac{s(s-1)\cdot p^2}{2} = 16$ trajectories need to be calculated. To construct a

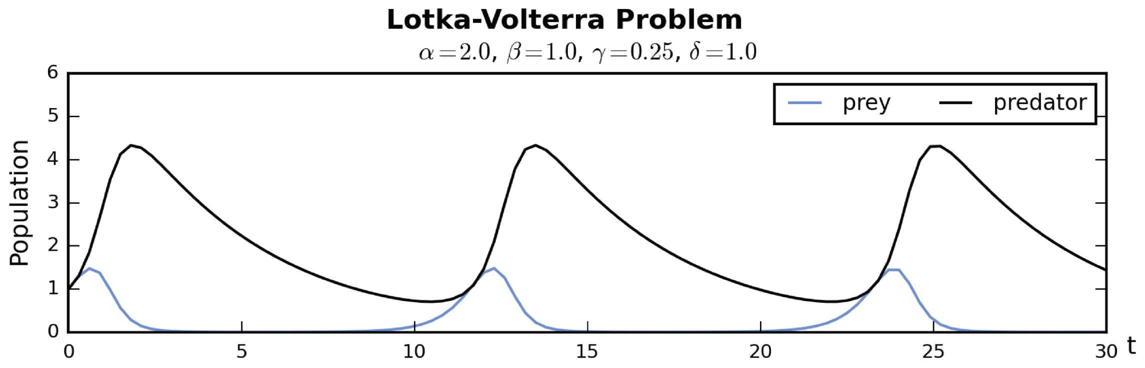


Figure 5.2: Temporal evolution of predator and prey populations for the Lotka-Volterra problem over 30 sec.

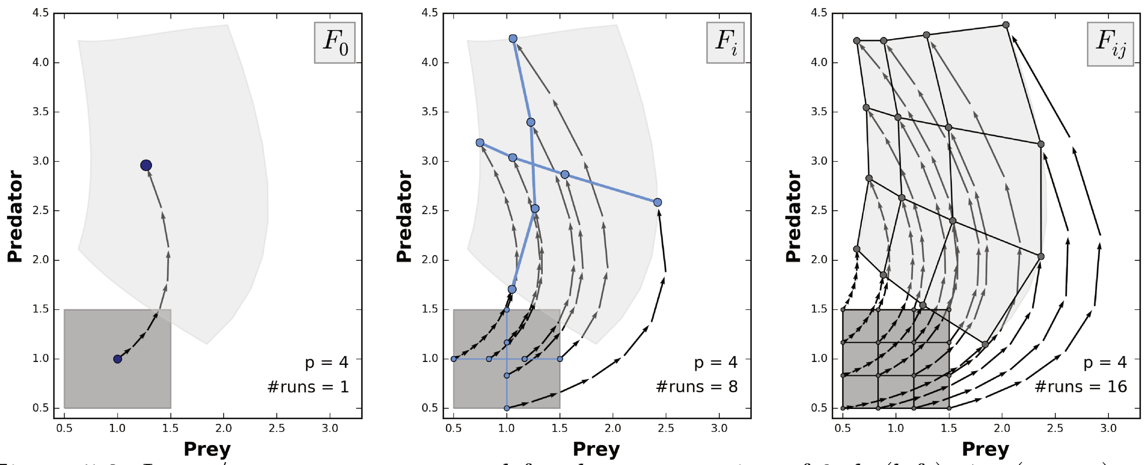


Figure 5.3: Input/output responses used for the construction of 0-th (left), 1st (center) and 2nd (right) order component functions $f_0,\ f_i,\ f_{ij}$ during the construction of a HDMR repromodel for the two-dimensional Lotka-Volterra problem, with $s_{\mathrm{HDMR}} = 2$ and $p = 4$. The input space $\mathcal{I} = [0.5,\ 1.5] \times [0.5,\ 1.5]$ is depicted as a dark gray square, the respective output space at $t = \Delta t = 1$ sec as the light gray area. The reference point is chosen as $\hat{c} = (1.0,\ 1.0)$ (dark blue dot). Respective solution trajectories needed for the construction of $f_0,\ f_i,\ f_{ij}$ are depicted as black arrows.

second-order HDMR, $1 + 8 + 16$ learning runs are therefore necessary in total.

For an uneven number of parametrization points, the HDMR is exact at the reference point. When adding a random perturbation to the initial conditions, the quality of the predicted output of the HDMR decreases. Figure 5.4 shows the absolute error of first-order HDMR expansions with $p = 3,\ 5,\ 7$ of the Lotka-Volterra problem, while adding random perturbations within $(0 - 10\%),\ (10 - 20\%),\ (20 - 30\%),\ (30 - 40\%)$ of a species' range in the input space. To that end, each 100 test runs were carried out, with perturbations chosen randomly within the respective perturbation interval. The x-axis denotes the percentage of the perturbation, that has been added to the initial concentration. The y-axis denotes the absolute error of the HDMR expansion at $t = \Delta t$, in comparison to the reference solution, calculated from the full system and integrated using RODAS(3)4 [46]. The absolute error at time Δt is being calculated as $\|u(\Delta t) - F(\Delta t)\|_{L^2} = \sqrt{\sum_{i=1}^{s} \left(u_i(\Delta t) - F(\Delta t)|_i\right)^2}$.

As can be seen, the HDMR expansion is exact when applying it to the reference point, i.e. at 0%. With increasing perturbation, the quality of the HDMR expansion deteriorates. For $p = 3$, the HDMR prediction is significantly worse than for $p = 5$ and $p = 7$ for all perturbations. For $p = 5$ and $p = 7$, relative errors are comparable for the first-order HDMR.

Figure 5.5 shows respective absolute errors of a second-order HDMR expansion. All absolute errors are significantly smaller than for the first-order expansion. At the same time, the number of learning runs required for the construction of the HDMR and the number of floating-point operations and operations during the evaluation of the HDMR increases, as can be seen in Figure 5.6. The black dots again shows the absolute errors, now only for random perturbations within $30 - 40\%$. The white bars depict the number of learning runs to construct the HDMR expansion. Grey bars depict the number of (theoretical) floating-point

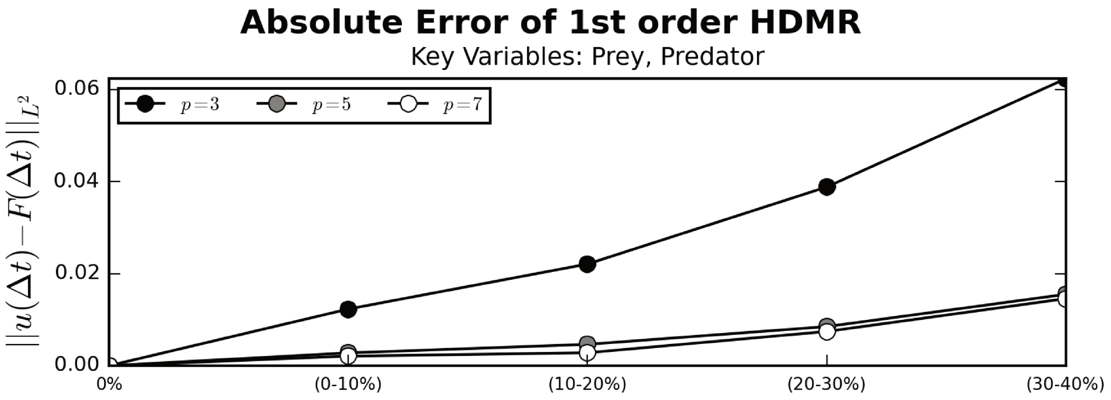


Figure 5.4: Relative error of a two-dimensional, first-order HDMR expansion constructed for the Lotka-Volterra problem over $\Delta t = 1$ sec and varying choices of p.

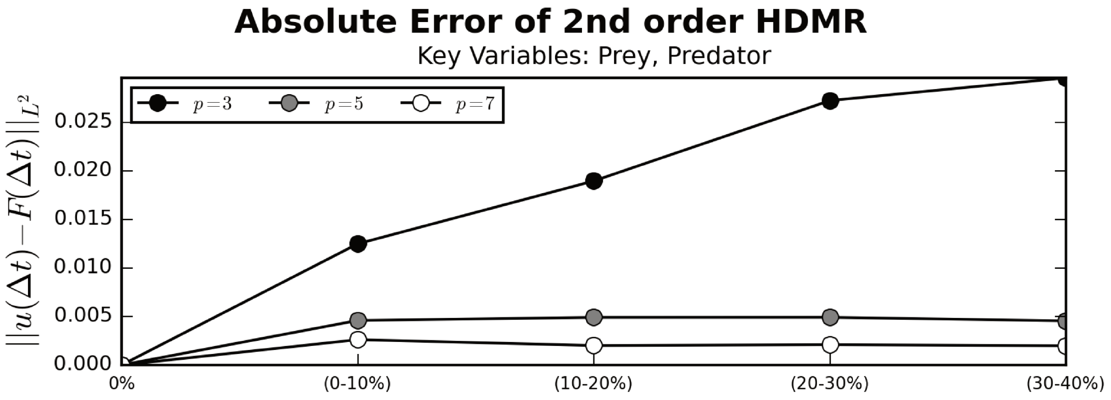


Figure 5.5: Relative error of a two-dimensional, second-order HDMR expansion constructed for the Lotka-Volterra problem over $\Delta t = 1$ sec and varying choices of p.

operations, that are required for a single evaluation of the HDMR expansion once. On the x-axis, the different HDMRs with varying degree and number of parametrization points are shown.

5.5.4 Qualification as a Coarse Propagator

An application to tropospheric chemistry of an HDMR expansion has for example been presented by Wang et al. [131, 133]: In a first study [133], the authors proposed first-order HDMRs for an application to a box-model study of complex alkane/NO_x/O_3 photochemistry, including 52 chemical species. For each hour of the day, one first-order HDMR expansion with 32 key variables was constructed, each with different photolysis conditions. In total, only 289 box-model runs were necessary. The HDMR operations were about 400 times faster than the box-model simulations. In a following study [131], Wang et al. applied the HDMR to a condensed, lumped form of the same chemical mechanism. This time, 6-dimensional first-order HDMR expansions were constructed for each species and each hour, which required a total of 55 box-model runs. The CPU time required to evaluate the HDMR expansions was more than 1 000 times faster than the computing time needed for the box-model simulations. The

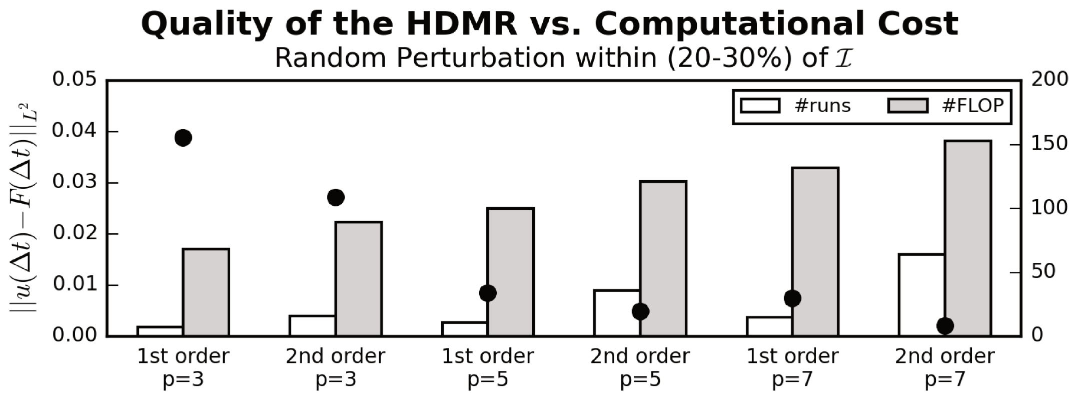


Figure 5.6: Quality of the HDMR versus computational cost for the Lotka-Volterra problem. Black dots depict absolute errors of the respective HDMRs (as described on the x-axis). White bars signify the number of learning runs necessary for their construction, gray bars the number of floating-point operations for their evaluation during runtime.

first-order HDMR was tested with 1 000 random input samples within the full input space $\mathcal{I}$. The relative error to the solution of Eq. (5.2.1) was below 5% for only 40% of the random samples. When choosing samples within a smaller subset of $\mathcal{I}$ the relative error was below 5% for more than 85% of the samples. A second-order HDMR expansion was further presented, that accurately represents the full dynamic range. In terms, another 1 215 model runs were required for the construction of the second-order component functions.

Due to the promising performance speed-ups shown in [131, 133], the HDMR method appears to be a viable choice for the coarse propagator within the parareal application. Since the number of key variables defines the number of model runs and the computational effort for the evaluation of the HDMR expansion, it is advisable to choose a low number of key variables.

The major drawback of repro-modeling approaches however is, that a repro-model is only valid for the fixed time-step size, it has been constructed for. In a combined application with the parareal algorithm, this requires either an equidistant decomposition into time-slabs or a decomposition into time-slabs with sizes being multiples of each other, such that the repro-model is being evaluated multiple times on bigger time-slabs. For the six-variable mechanism considered in Chapter 4.2, equidistant decompositions disqualified, since they have shown to result in load-imbalances. When using a decomposition with time-slab sizes being multiples of each other, the respective repro-model must be valid even for the smallest step-sizes. In Chapter 4.2, we had further seen time-step sizes ranging from approx. 10^{-9} sec to 10^2 sec. Respectively, the repro-model would have to be called sequentially $(T_{\text{end}} - T_0)/10^{-9} = 3.6 \cdot 10^{12}$ times during the sequential prediction for this scenario. A performance gain can not be expected here, unless a single evaluation of the repro-model is more than $3.6 \cdot 10^{12}$ times faster than the total computational effort of the sequential scheme. Performance gains found in literature typically do not exceed a factor of 1 000. On these grounds, we refrain from a repro-model based parareal approach for the usage in the framework of an internal parallelization of the six-variable mechanism.

However, when using an equidistant decomposition in time, a combined application of repro-modeling and the parareal algorithm still is promising. In the following Chapter we will therefore apply the approaches to enable an external parallelization of the more sophisticated chemical mechanism RADM2. Different than in the examples presented so far, we now also taking into account external disturbances and varying photolysis rates.

5.6 Conclusions

We have seen different model reduction techniques. For the application as a coarse propagator within the parareal algorithm, a suitable reduced model must be significantly faster than the full model, while maintaining a relatively good accuracy. Lumping methods require a high degree of expertise in the chemical system and do not present a generic model reduction framework. A lumped model will be smaller in dimension, but not necessarily less stiff. Since stiffness is the dominant factor on the computing time, a significant reduction in computing time can not per se be expected from a lumped model. QSSA methods present a way to tackle the system's stiffness by a separation of timescales. Numerical investigations [109, 110] however have shown, that such methods are not promising in terms of computing time compared to Rosenbrock schemes. The ILDM method presents a similar approach, that bases on a description of the kinetics on lower dimensional manifolds. Computational gains compared to a solution of the full system can only be expected, if the ILDM can be calculated a priori and accessed from look-up tables during run-time. Due to the enormous storage demands for the look-up tables, this is technically not feasible for an application to atmospheric chemical kinetics. Repro-models are an alternative to aforegoing model reduction approaches, by providing functional representations of the time-dependent kinetic change within chemical systems. The HDMR method thereunder presents a viable choice as a coarse propagator within the parareal algorithm when using equidistant decompositions in time. For the application to atmospheric chemical kinetics, HDMRs have been presented in [131] with speed-ups up to a factor of 1 000×.

Since repro-models are only valid for a fixed time-stepping size, their usage is promising only, when considering equidistant decompositions in time. For the six-variable example considered in Ch. 4.2, load-imbalances imposed the usage of adaptive decompositions. The search for a faster coarse propagator for this scenario ends here - fruitless. In the context of an external parallelization, where the time-slab size is fix and preset by the size of the splitting intervals, the approach is useful in contrast. In the next Chapter, we will see a successful application of using repro-models as coarse propagators, to enable an external parallelization of the sophisticated chemical mechanism RADM2.

Chapter 6

Numerical Experiments - RADM2

In Chapter 4, we had investigated an internal in-time parallelization of an example six-variable tropospheric mechanism. We had identified some fundamental requirements, an efficient parareal scheme for stiff ODEs must meet. We had seen, that an efficient coarse propagator should be considerably faster than the full model without a significant loss of accuracy. We had identified needs for adaptive coarse and fine propagators. By means of the experimental results for approach AP-I, we had seen, that a uniform decomposition into equidistant time-slab leads to load-imbalances during the parallel correction step. With AP-II we consequently introduced an adaptive initial decomposition, which is suggested by the coarse propagator. Both for AP-I and AP-II, we had further seen, that the overall computational cost is dominated by the coarse propagator during the sequential prediction within each iteration. Following, the results for AP-III had showcased, that a one-step coarse propagator diminishes this dominance, but at the same time it is not very accurate and hence increases the number of iterations until convergence. All adaptive parareal approaches had shown to converge, but none of them outperformed the adaptive Rosenbrock solver RODAS(3)4 in a purely sequential application. Instead of applying an adaptive solver as the coarse propagator, we then intended to use a faster and more accurate coarse propagator. For that purpose, we have been considering different reduced models in Chapter 5. From all the approaches discussed, the HDMR repro-modeling approach turned out to be the most promising choice - but only, under the constraint of an equidistant decompositions in time. Since for the example from Sec. 2.2.2 load-imbalances forced the usage of adaptive decompositions, repro-models disqualified in that context. For an overview on the specifications of the approaches discussed, see Tab. 6.1.

A combined application of repro-modeling and the parareal algorithm is however promising in the context of an external parallelization. An external parallelization equates a parallelization of N subsequent splitting intervals. Different than for the examples presented in Sec. 2.2.2, the decomposition into time-slabs is now equidistant and preset by the size of the splitting intervals Δt_{split}. This enables a straightforward construction and usage of respective repro-models as coarse propagators in the parareal scheme. For the experiments presented for an internal parallelization, we had seen equidistant decompositions to result in global load-imbalances. To lower the load-imbalances of an equidistant decomposition, we had pro-

	Decomposition	**Initialization $\mathcal{G}_{\text{init}}$**	**Prediction $\mathcal{G}$**	**Correction $\mathcal{F}$**	
CP	equidistant	fixed δT	fixed δT	fixed δt	with $\delta t < \delta T$
AP-I	equidistant	adaptive	adaptive	adaptive	with rtol$_{\mathcal{F}}$ < rtol$_{\mathcal{G}}$
AP-II	adaptive - suggested by $\mathcal{G}_{\text{init}}$		adaptive	adaptive	with rtol$_{\mathcal{F}}$ < rtol$_{\mathcal{G}}$
AP-III	adaptive - suggested by $\mathcal{G}_{\text{init}}$		one-step	adaptive	with rtol$_{\mathcal{F}}$ < rtol$_{\mathcal{G}}$
RM-P	equidistant	HDMR	HDMR	adaptive	

Table 6.1: Overview of the parareal approaches discussed in this work: Classical Parareal (CP), Adaptive Parareal (AP-I, AP-II and AP-III) and repro-model Parareal (RM-P).

posed two adaptive schemes, AP-II and AP-III. Since we now are restricted to equidistant decompositions, both approaches disqualify in the context of the external parallelization. A repro-model parareal approach will therefore lead to load-imbalances caused by the fine correction within the parareal iteration. The main source of these load-imbalances is now given by the different levels of photolytic activity and hence different degrees of stiffness at different daytimes. Consequently, the computational effort for the fine correction within the parareal scheme will vary for different daytimes, as will later be seen in Fig. 6.8.

An external parallelization equates a parallelization of subsequent splitting intervals. For the application within a three-dimensional AQM, further challenges arise: Due to the operator splitting within the CTM, disturbances between two intervals, caused by advection, diffusion, changing photolysis rates and other processes, must be considered in between two intervals. For the sake of simplicity, we do not take into account all such processes, but emission and varying photolytic activities only.

6.1 Test Scenarios

Up to now, we have been utilizing very simple models to explain and visualize the parareal algorithm and the basic principles of ILDM and HDMR. From now on, more realistic test models shall be considered, similar to the chemical kinetics within in a three-dimensional complex AQM. We choose three zero-dimensional box-models, as proposed by Kuhn et al. [1, 56]: LAND, PLUME and URBAN. Those test problems have been used for numerous validation tests of chemical models, e.g. by Poppe et al. [100] or Gross and Stockwell [42].

The most simple scenario, LAND, covers a continental planetary boundary layer with a low burden of pollutants and without emission. It covers only very little photochemical activity. Scenario two, PLUME, represents a moderately polluted planetary boundary layer with continuous emissions of a complex mixture of organic compounds. The last scenario, URBAN, represents a polluted planetary boundary layer with emissions as for PLUME. All simulations start at noon on July 1st with a solar zenith angle of +22°. The integration period covers 5 days, with a clear sky and a constant base temperature throughout the simulation. Photolysis rates are calculated every hour with an albedo of 0.1 at LAT=45° and LON=0° and a constant solar declination of +23°. The meteorological parameters and the initial concentrations for all chemical species can be found in Tab. 6.2. The chemical mechanism used in all simulations is RADM2, as introduced in Sec. 2.2. Reaction rate constants were

	LAND	**PLUME**	**URBAN**
Start	July 1st, 12 pm	July 1st, 12 pm	July 1st, 12 pm
End	July 6th, 12 pm	July 6th, 12 pm	July 6th, 12 pm
Ground Albedo	0.1	0.1	0.1
Solar Declination	+23°	+23°	+23°
Longitude	0°	0°	0°
Latitude	45°	45°	45°
Altitude [km]	0.0	0.0	0.0
Temperature [K]	288.15	288.15	298.15
Pressure [mbar]	1013.25	1013.25	1013.25
M [#molecules/cm^3]	2.55E19	2.55E19	2.46E19
H_2O [%]	1.0	1.0	1.0
O_3 [ppbV]	30.0	50.0	30.0
NO [ppbV]	0.1	0.2	0.1
NO_2 [ppbV]	0.1	0.5	0.1
HNO_3 [ppbV]	0.1	0.1	1.5
CO [ppbV]	100.0	200.0	100.0
CH_4 [ppbV]	1700.0	1700.0	1700.0
H_2 [ppbV]	500.0	500.0	500.0
H_2O_2 [ppbV]	2.0	2.0	2.0
HCHO [ppbV]	1.0	1.0	1.0
O_2[%]	20.9	20.9	20.9
N_2[%]	78.1	78.1	78.1
ISO [ppbV]	0.0	0.0	0.0
DMS [ppbV]	0.0	0.0	0.0
else [ppbV]	10^{-20}	10^{-20}	10^{-20}

Table 6.2: Initial values for LAND, PLUME and URBAN.

chosen as presented in the original publication from 1990 by Stockwell et al. [119], except for the reactions of $HO + ETH$ and $HO + CH_4$. Since the best agreement with the box-model results presented in [42, 56, 100] was achieved with this choice, those two reaction rates were calculated as proposed in Stockwell et al. [118]. Lists of the chemical species, reactions and the reaction rates are also presented in the Appendix. For the implementation in RADM2 of the tests proposed in [56], emission rates for VOCs had to be splitted as proposed by Gross and Stockwell [42]. The diurnal variation of the photolysis intensity is calculated based on the radiation transfer model by Roeth and adapted by Kuhn [1]. The functions defining the calculation of emissions, photolysis and reaction rates are also presented in the Appendix.

An appropriate ODE solver (we use RODAS(3)4, again) is re-initiated every hour, so that a splitting interval covers a period of $[t_n,\ t_n + 3600 \text{ sec}]$. At the beginning of each splitting interval, photolysis rates are updated and calculated from prescribed diurnal variations of radiation. Within the splitting interval, the system is in an autonomous mode. In total, 120 subsequent splitting intervals are necessary to cover the full integration period of 5 days. Since these scenarios represent idealized three-dimensional box-models, meteorological feedbacks, advection or diffusion are not incorporated between the intervals.

6.2 Repro-Model Parareal (RM-P)

As a coarse propagator, we now take HDMR repro-models into account. To this end, one repro-model is constructed for each hour of the day, leading to a total of 24 different, 63-dimensional HDMR models for every scenario, each set-up around 11 parametrization points for every species. The input range $\mathcal{I}_i$ for species SPC_i at time t was defined as

$$\begin{aligned} \mathcal{I}_i(t) &:= [\max(0, \min(\mathrm{SPC}_i) - 0.05 \cdot \delta_i)\,,\ \max(\mathrm{SPC}_i) + 0.05 \cdot \delta_i] \\ \delta_i(t) &:= \max(\mathrm{SPC}_i) - \min(\mathrm{SPC}_i), \end{aligned}$$

whereat max and min were taken as the maximum/minimum over all 5 days at the respective hour of the day. Only first-order terms were constructed. At this point, the flexibility is high: The input range is tailored to the actual trajectory of the solution. It is though possible, to generate more universal HDMR models using wider input ranges and more parametrization points. It is as well possible, to parametrize also the photolysis, i.e. to construct $(63 + 1)$-dimensional HDMR models. We will compare the repro-model parareal approach to the adaptive parareal approach AP-I, as presented in Sec. 4.2. Since the decomposition into time-slabs here is preset by the size of the splitting intervals used for the operator splitting within an AQM, both approaches using an adaptive decomposition (AP-II and AP-III) would lead to additional interpolations to meet the time interval's end. Such interpolation could cause side-effects and should not be discussed subsequently. As a coarse propagator to be used for AP-I, we again utilize a RODAS(3)4 integrator, now with coarser tolerances. We choose the coarsest tolerances, that allowed for a stable integration: As a starting time-step size for $\mathcal{G}$, we choose $h_{\mathrm{init}} = 10^{-3}$ sec and a relative tolerance of $\mathrm{rtol}_{\mathcal{G}} = 1.0$.

6.2.1 Convergence

Figures 6.1-6.3 show the convergence behavior of RM-P and AP-I for simulations over 24 hours (left) and over 5 days (right), each parallelized over 24 or 120 processors, respectively. The x-axis shows the number of iterations, starting with the initialization (iteration 0). On the y-axis, we see the relative error at final integration time (24 or 120 hours), calculated as defined by Eq. (4.3.1).

For scenario RADM2 LAND (Fig. 6.1), RM-P leads to relative errors below 10^{-1} from the first iteration on for the 24 hour simulation (left) and from the second iteration on for the 5 day simulation (right). For small iteration numbers, the RM-P approximation is better than the AP-I approximation. RM-P errors are smaller than for AP-I up to the 3rd for the 24 hour and up to 7th iteration for the 5 day simulation.

Figure 6.2 exemplarily shows the solution trajectories of O3 for an AP-I (upper) and RM-P (lower) parareal integration. The blue lines denote the actual trajectory, dotted lines the parareal solutions. The color of the dots signify the iteration number and ranges from white (iteration 0, i.e. initialization) to black (7th iteration). For AP-I the initial guess under-, the

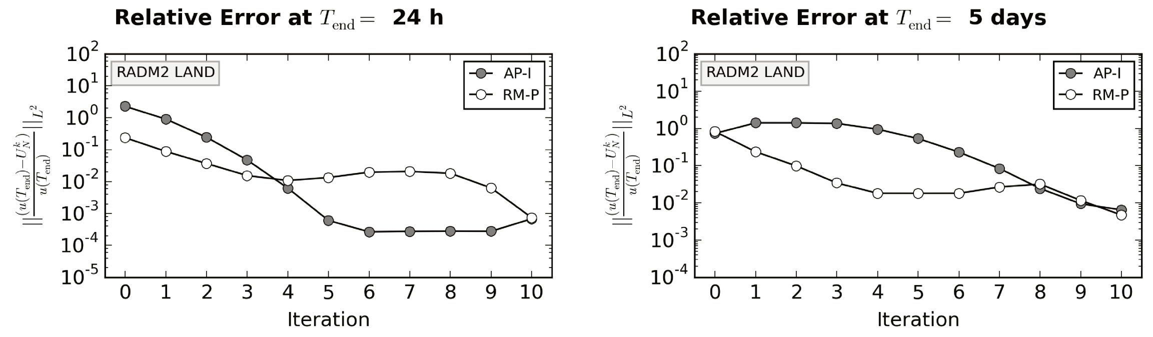


Figure 6.1: Relative Errors at final integration time for RADM2 LAND.

second overestimates the real trajectory. In the following iterations, the solution is approached from below. After 4 iterations, the parareal solution appears to have converged (from a visual point of view). For RM-P already the initial guess is very close to the real trajectory for RM-P. As a convergence criterion, we shall again consider the relative change in the solution at final integration time, as defined by Eq. (4.0.1). Convergence is reached, as soon as the relative change falls below a threshold of $\text{tol}_{\text{stop}} = 10^{-2}$. For the simulation over 24 hours this is given within 4 parareal iterations for RM-P, and 5 for AP-I. For the simulation over 5 days, RM-P needs 5 iterations, AP-I 10.

For scenarios PLUME (Fig. 6.3) and URBAN (Fig. 6.4) again both RM-P and AP-I converge for the simulation time of 24 hours. As for LAND, the relative error is smaller for RM-P for both scenarios (with one exception in iteration 8 for scenario URBAN). For the respective 5 day simulations, RM-P again converges - slowly, while AP-I shows an oscillatory behavior. For nitric oxide NO, this behavior is showcased in Fig. 6.5, where the parareal NO−trajectories with varying iteration number are plotted over 120 hours. The presence of a polluted atmosphere leads to the necessity of very small time-steps at the beginning of each time-slab. Obviously, the tolerance of the adaptive coarse propagator is not sufficient, to capture the real trajectory. From iteration to iteration, varying solution branches are being predicted. In comparison to RM-P, AP-I leads to bigger variations from the actual trajectory (blue). Those variations are high especially during daytime. Since Fig. 6.4 shows the error at final integration time, i.e. noon on July 6th, the variation equates the maximal amplitude during 24 hours and the error is correspondingly high.

For the iteration numbers shown here, the AP-I parareal solution does not converge to the actual solution. Both for PLUME and URBAN, AP-I finally does converge - at the latest after 120 iterations. From a practical point of view, high iteration numbers are not interesting, since they will lead to high WCTs.

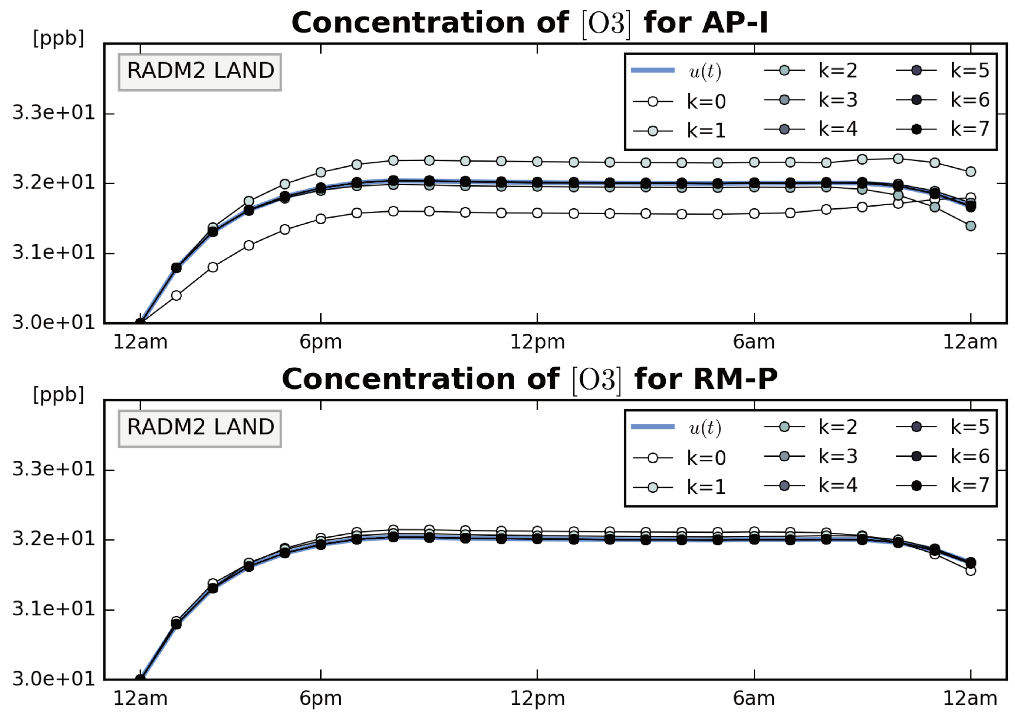


Figure 6.2: Concentration of O3 for RADM2 LAND over 24 hours.

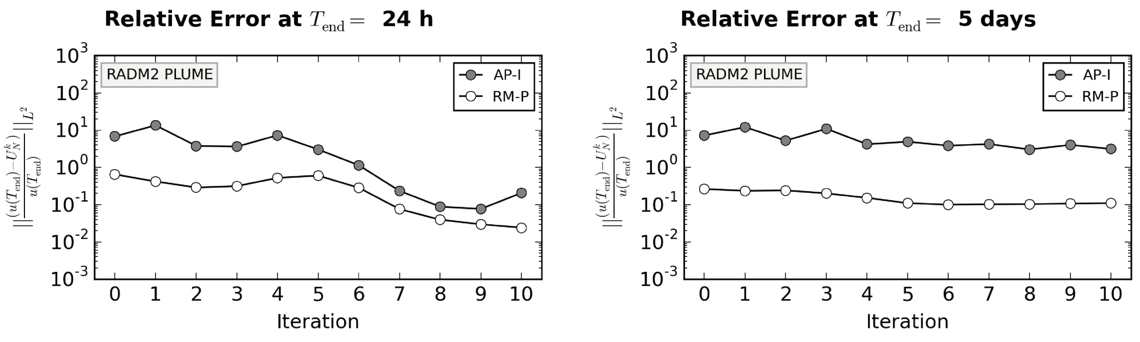


Figure 6.3: Relative Errors at final integration time for RADM2 PLUME.

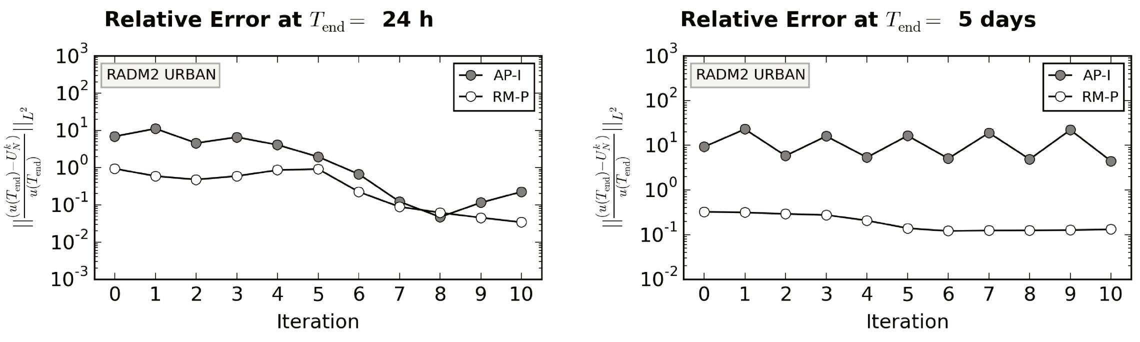


Figure 6.4: Relative Errors at final integration time for RADM2 URBAN.

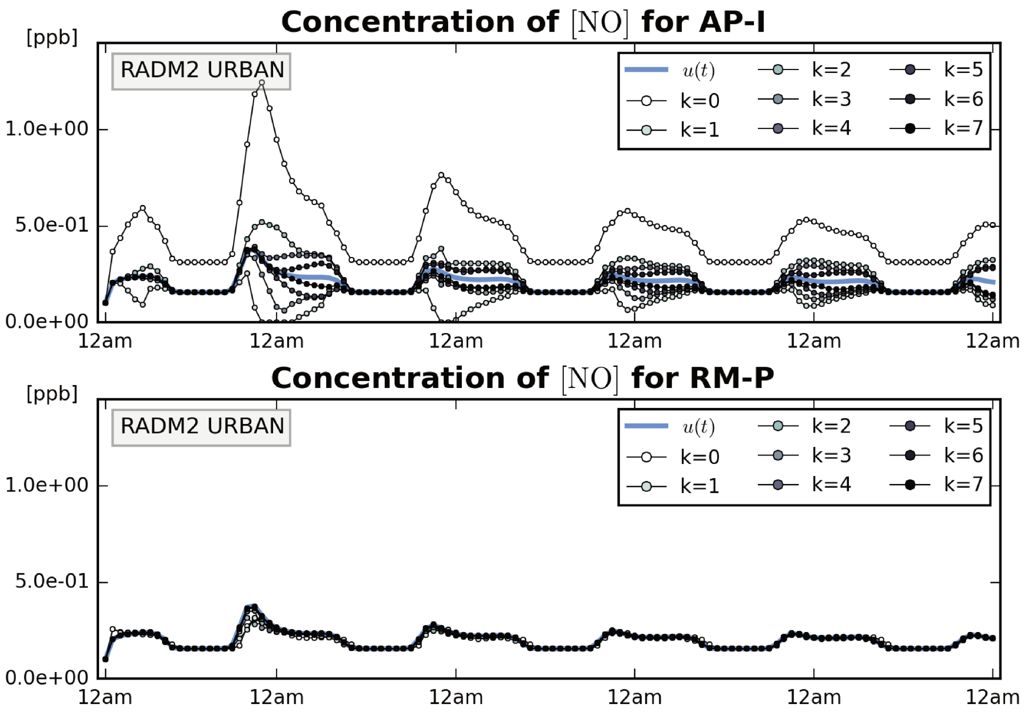


Figure 6.5: Concentration of NO for RADM2 URBAN over 5 days.

6.2.2 Parallel Wall Clock Time

We have seen, that both RM-P and AP-I do converge, i.e. the relative change falls below a threshold of $\mathrm{tol}_{\mathrm{stop}} = 10^{-2}$. In the following, we want to take into account the respective parallel WCTs until convergence. The parallel WCT is composed of the WCT of the initialization phase ($\mathrm{WCT}_{\mathrm{init}}$) plus the sum of the WCT per iteration (WCT_k) over all k_{conv} iterations.

For AP-I, the WCT for the initialization phase is defined by the sum of the number of internal time-steps used on each time-slab by the coarse solver $\mathcal{G}$, multiplied with the computing time per step (which is approx. constant),

$$\mathrm{WCT}_{\mathrm{init}} = \sum_{i=0}^{N-1} N_{i,\mathcal{G}} \cdot \mathrm{CT}_{\mathrm{step}}.$$

For RM-P the initialization WCT is defined by

$$\mathrm{WCT}_{\mathrm{init}} = N \cdot \mathrm{WCT}_{\mathrm{HDMR}},$$

whereat $\mathrm{WCT}_{\mathrm{HDMR}}$ denotes the WCT required for one single evaluation of an HDMR repro-model. Since all HDMR repro-models are constructed with the same degree, dimension and number of parametrization points, their evaluation effort is constant for all iterations and all time-slabs. The WCT_k per iteration is defined by

$$\mathrm{WCT}_k \quad = \quad \mathrm{WCT}_{\mathrm{corr}}^k + \mathrm{WCT}_{\mathrm{pred}}^k,$$

whereat the $\mathrm{WCT}_{\mathrm{corr}}$ of the parallel correction in both cases is defined by the maximum WCT of the fine propagator over all time-slabs,

$$\mathrm{WCT}^k_{\mathrm{corr}} = \max_{i=0,N-1}(\mathrm{WCT}^k_{i,\mathcal{F}}).$$

The $\mathrm{WCT}_{\mathrm{pred}}$ of the sequential prediction equates $\mathrm{WCT}_{\mathrm{init}}$ - approximately for AP-I and exactly in case of RM-P. This leads to a parallel $\mathrm{WCT}_{\mathrm{par}}$ of

$$\mathrm{WCT}_{\mathrm{par}} = \mathrm{WCT}_{\mathrm{init}} + \sum_{k=1}^{k_{\mathrm{conv}}} k \cdot \mathrm{WCT}_k.$$

Figure 6.6 contrasts the averaged WCT_k per iteration (left) and parallel $\mathrm{WCT}_{\mathrm{par}}$ (right) for AP-I and RM-P for scenario LAND and a parallel-in-time simulation over 24 hours in comparison to the sequential solution procedure. In both plots, white bars stand for RM-P, gray bars for AP-I. The shaded blue bars in the left picture denote the average contribution of the parallel correction $\mathrm{WCT}_{\mathrm{corr}}$. The blue bar in the right plot shows the WCT of a sequential integration using the fine propagator. Respective values are also denoted in Tab. 6.3.

The $\mathrm{WCT}_{\mathrm{k}}$ per iteration (left plot in Fig. 6.6) is composed of the parallel correction time (shaded blue) plus the $\mathrm{WCT}_{\mathrm{pred}}$ of the sequential prediction. For RM-P, the cost per iteration $\mathrm{WCT}_{\mathrm{k}}$ is clearly dominated by the correction time, i.e. $\mathrm{WCT}^k_{\mathrm{corr}} = \max_{i=0,N-1}(\mathrm{WCT}^k_{i,\mathcal{F}})$. This part is the parallelizable part of the code. The cost for the sequential prediction is significantly smaller. In contrast, WCT_k for AP-I is dominated by the sequential prediction by means of an adaptive propagator. In total, WCT_k is about 80 % smaller for RM-P than it is for AP-I.

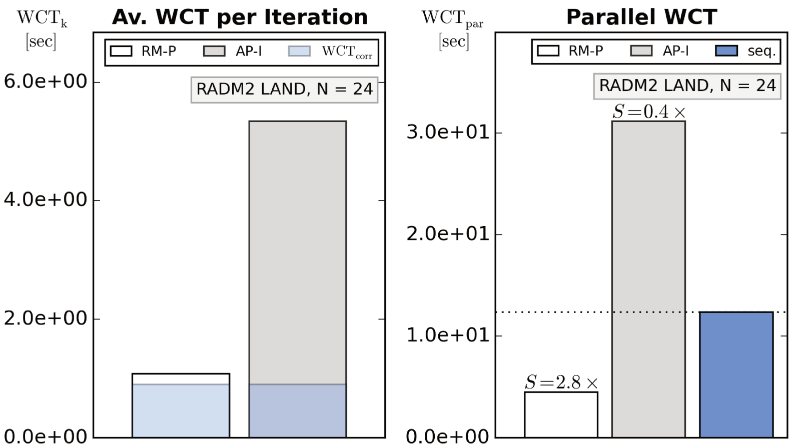


Figure 6.6: Averaged WCT per iteration (left) and parallel WCT (right) for a RADM2 Land simulation over 24 hours. Shaded blue bars in the left picture denote the percentage of the fine propagator during the parallel prediction. The blue bar in the right picture denotes the sequential reference run. Respective speed-ups in comparison to the sequential reference are printed above the bars .

	N	k_{conv}	$\mathrm{WCT}_{\mathrm{init}}$	in average over k_{conv} $\mathrm{WCT}_{\mathrm{corr}}$	$\mathrm{WCT}_{\mathrm{pred}}$	$\mathrm{WCT}_{\mathrm{k}}$	$\mathrm{WCT}_{\mathrm{par}}$
AP-I	24	5	4.44e+00	9.03e-01	4.44e+00	5.34e+00	3.12e+01
RM-P	24	4	1.70e-01	9.03e-01	1.70e-01	1.07e+00	**4.49e+00**
RODAS(3)4	24	-	-	-	-	-	1.23e+01
AP-I	120	10	2.16e+01	8.94e-01	2.16e+01	2.25e+01	2.47e+02
RM-P	120	5	8.76e-01	8.94e-01	8.76e-01	1.77e+00	**9.74e+00**
RODAS(3)4	120	-	-	-	-	-	6.04e+01

Table 6.3: Parallel WCT until convergence for a simulation of RADM2 LAND.

In total, the parallel $\mathrm{WCT}_{\mathrm{par}}$ for RM-P is approx. 2.8 times smaller than the $\mathrm{WCT}_{\mathrm{seq}}$ of the sequential integrator. A parallelization over 24 processors here leads to a speed-up of a factor of $S = 2.8$ compared to the sequential scheme with $S = \mathrm{WCT}_{\mathrm{seq}}/\mathrm{WCT}_{\mathrm{par}}$. AP-I in contrast is slower than the sequential scheme with $S = 0.4$.

Figure 6.7 contrasts $\mathrm{WCT}_{\mathrm{par}}$ for the same scenario LAND but a parallel-in-time simulation over $5 \cdot 24$ hours. The effort per iteration for RM-P now equates approx. 8 % of WCT_k for AP-I. Since 5 iterations were necessary until convergence for RM-P, but 120 time-slabs are computed in parallel, the performance improvement is better than for the 24 hour simulation. In total, RM-P is now 6.2 times faster than the sequential scheme. In contrast, 10 iterations are necessary for AP-I, leading to a speed-up of $S = 0.2$.

For scenarios PLUME and URBAN, more than 10 iterations were necessary for the simulation over 24 hours, both for AP-I and RM-P. No speed-ups were observed. For the simulation

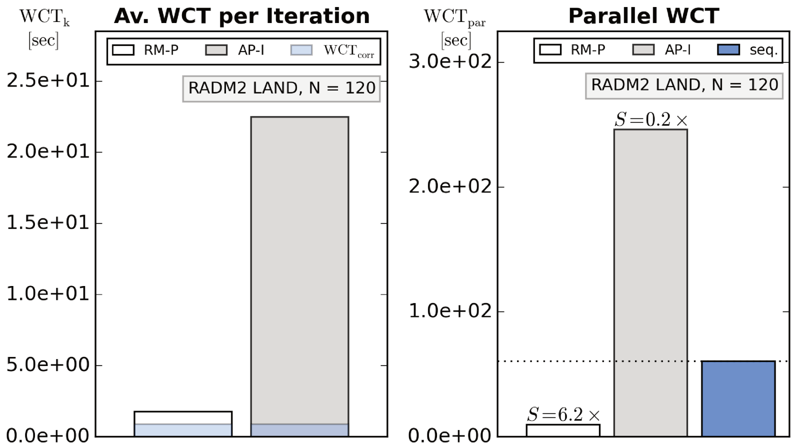


Figure 6.7: Averaged WCT per iteration (left) and parallel WCT (right) for a RADM2 Land simulation over 120 hours. Shaded blue bars in the left picture denote the percentage of the fine propagator during the parallel prediction. The blue bar in the right picture denotes the sequential reference run. Respective speed-ups in comparison to the sequential reference are printed above the bars .

over 5 days, RM-P converged in 8 iterations for PLUME and 9 iterations for URBAN, while AP-I again required more than 10 iterations in both cases. A repro-model parareal approach here enables speed-ups of $S = 5.6$ for PLUME and $S = 4.7$ for URBAN compared to a sequential scheme.

From the results presented in this section, we see, that HDMR models indeed present coarse propagators, that are significantly faster than adaptive coarse propagators - in average approx. 25 times for the scenarios presented here. Potential ways to further improve the performance of the coarse prediction using HDMR models, will be presented in the concluding remarks.

6.2.3 Load Imbalances

Now, we are interested in the distribution of parallel work. The parallelizable part within the parareal algorithm is the correction phase, in which the coarsely and sequentially predicted trajectories are corrected, see also Eq. 3.2.2. To that end, both the fine and the coarse propagator are started in parallel on each of the time-slabs. Since the coarse propagator is faster than the fine propagator (per design), $\mathrm{WCT}^k_{\mathrm{corr}}$ is defined by the maximum WCT over all time-slabs,

$$\mathrm{WCT}^k_{\mathrm{corr}} \quad = \quad \max_{i=0,N-1}(\mathrm{WCT}^k_{i,\mathcal{F}}).$$

Figure 6.8 shows the $\mathrm{WCT}_{i,\mathcal{F}}$ for the parallel application of the fine propagator on all time-slabs. The y-axis denotes the WCT in seconds for the integration of one time-slab of 3 600 sec using the fine-propagator $\mathcal{F}$. On the x-axis we see the physical time, whereat one bar stands for one processor. Each bar indicates the computing time required by one processor to integrate the solution on one time-slab. Additionally, the average $\mathrm{WCT}_{i,\mathcal{F}}$ is plotted as a blue horizontal line. In this plot, the fine propagator was initialized with the exact solution. This allows to derive the load-imbalances for the parareal integration, which is formulated over the full interval of 5 days.

The highest computing time for one such time-slab over 5 days is required on the very first time-slab. This is caused by the fact, that the initial conditions here are disturbed and out of balance, leading to a stiff problem. Throughout the simulation, the computing time shows a clear dependency on the level of photolytic activity, present on the time-slab: Highest efforts are observed during the transition phases between day- and nighttime, when fast photolytic processes take place. With the inset of the sunlight in the morning, photolytic radicals such as HO are being created, which are later destroyed with sunset. Lowest efforts can be observed at nighttime, were no photolytic activity appears. The lowest WCT over all time-slabs, $\min_{i=0,N}(\mathrm{WCT}_{i,\mathcal{F}})$, equates approx. 27% of the highest effort of $\mathrm{WCT}^k_{\mathrm{corr}}$. The consequence is, that these processors will idle for about 73 % of $\mathrm{WCT}^k_{\mathrm{corr}}$.

From Tab. 6.3 and the blue shaded bars in the left plots of 6.6 and 6.7 we can see the domination of WCT_k by means of the parallel correction using the fine propagator, whereat the effect diminishes with increasing N. An unbalanced computational effort therefore has a major impact on the efficiency of the whole algorithm. Since the size of the time-slabs is preset by the size of the splitting intervals, it is not possible to use a fully adaptive decomposition to decrease these imbalances. It is however possible, to adjust the decomposition a-priori, such that the size of the biggest intervals are multiples of the size of the smallest. In that framework, HDMRs would have to be constructed for the smallest Δt.

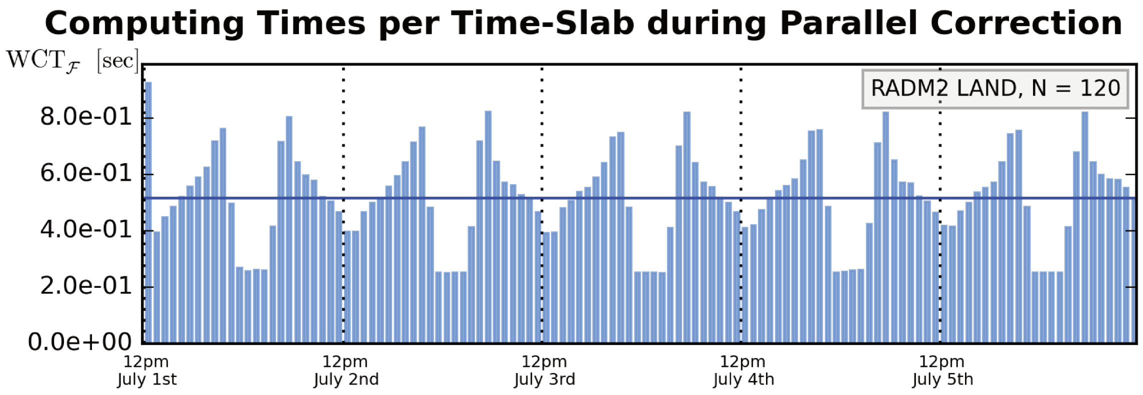


Figure 6.8: Computing times per time-slab to propagate the solution over $[t_n,\ t_{n+1}]$ using the fine propagator $\mathcal{F}$. Each bar signifies one time-slab, which each is calculated on one processors. The horizontal blue lines denotes the average computing time over all time-slabs.

6.3 Summary and Concluding Remarks

In this Chapter, we investigated the application of the parareal algorithm in combination with repro-modeling approaches. Since repro-models represent functional mappings of the solution from time t to time $t+\Delta t$ for a fixed time-step Δt size, we restricted to the case of an external parallelization with uniform decompositions in time and time-slab sizes equal to Δt. Three zero-dimensional box-models with varying photolysis rates and different levels of pollution were chosen as test beds. These scenarios represent typical set-ups for the calculation of the atmospheric chemistry as it is incorporated in typical AQMs. Photolysis rates were updated every hour, respectively we defined splitting intervals of $\Delta t_{\text{split}} = 3\ 600$ sec. Different than in a real AQM, advection and diffusion were not considered. A parallelization in time was realized via a parallelization of subsequent splitting intervals. This directly prescribed the size of the parareal time-slabs, i.e. an adaptive decomposition was not possible anymore. A new repro-model parareal approach (RM-P) using an HDMR-based coarse propagator was contrasted to AP-I, as introduced in Sec. 4.2.

For the external parallelization considered in this section, the HDMR-based coarse propagator in average was 25 times faster than an adaptive coarse propagator $\mathcal{G}$ with $\text{rtol}_{\mathcal{G}} = 1.0$ - the coarsest tolerance, that allowed for a stable computation. The computational cost per iteration has again shown to be governed by the cost of the parallel fine correction. For the fine propagator during the parallel correction, we have noticed diurnally varying computational efforts. Those differences are directly caused by the presence of different levels of photolytic activity throughout a day. Similar than for the internal parallelization by means of the scenarios examined in Sec. 4.2, this led to load-imbalances during the parallel correction. Those can potentially be decreased by using a-priori adjusted decompositions, with biggest interval sizes being multiples of the smallest.

In all cases and for all three scenarios, AP-I was slower than the sequential reference. For the scenario RADM2 LAND and a simulation over 24 hours, the repro-model parareal approach, distributed over 24 processors, was 2.8 times faster than the sequential integration scheme. Convergence was reached in 4 iterations. The respective adaptive parareal approach AP-I was slower than the sequential integrator. For a simulation over 120 hours and distributed over 120 processors, the repro-model parareal approach needed 5 iterations until convergence, leading to a speed-up of a factor of 6.2 compared to the sequential integrator. For the two polluted scenarios, RADM2 URBAN and PLUME, and for a simulation over 5 days, speed-ups of $S = 5.6$ for PLUME and $S = 4.7$ for URBAN were identified. However, no speed-ups were observed for a parallel-in-time integration over 24 hours. Presumably, the respective repro-models were not accurate enough. More profound HDMR repro-models could be constructed by means of higher numbers of parametrization points and/or adding second-order terms.

Further performance improvements can potentially be achieved, when parameterizing the HDMR not in the phase space of chemical species, but in terms of modes (c.f. Sec. 5.4). Then, the number of key variables would be defined by the dimension of the lowest dimensional intrinsic manifold. Typically, the size of the ILDM is a lot smaller than the dimension size, as can be seen in Fig. 6.1 for the scenarios presented here. For scenario RADM2 LAND, 21 parameters could be chosen at daytime, 3 at nighttime instead of 63 parameters all-over. In turns, additional matrix vector operations would be necessary for the mapping between the space of species and the space of modes.

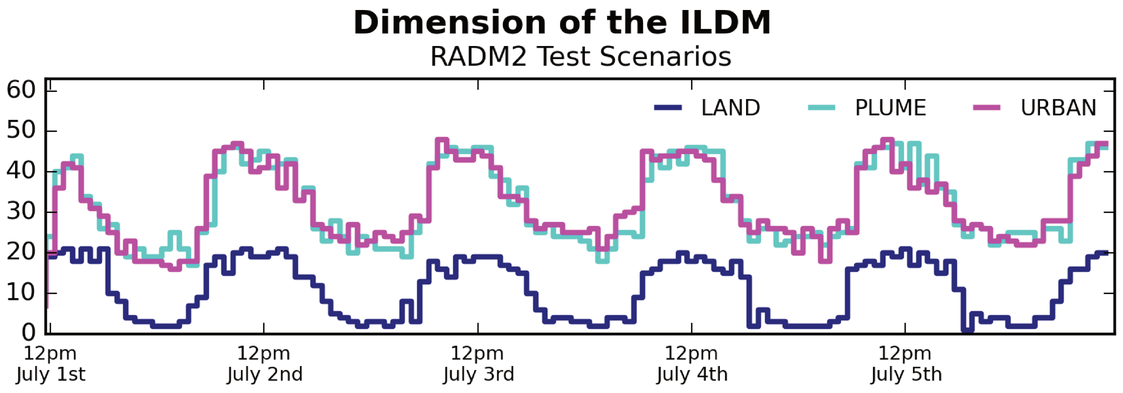


Figure 6.1: Dimension of the intrinsic low dimensional manifold for all three scenarios, calculated as proposed by Tomlin et al. in [121] with $\kappa = 10^{-12}$, $\mathrm{tol}_{\mathrm{rel}} = 10^{-8}$, $\mathrm{tol}_{\mathrm{fast}} = -10^{3}$ and $\mathrm{tol}_{\mathrm{cons}} = 10^{-6}$.

Chapter 7

Conclusions and Outlook

In this thesis, we investigated the potential of an in-time parallelization of atmospheric chemical kinetics within an atmospheric chemical transport model. Within such a framework, chemistry is being solved isolatedly over splitting intervals, decoupled from other processes. We identified and investigated two fields of application of an in-time parallelization of atmospheric chemistry: An internal parallelization within a single and an external parallelization across multiple splitting intervals. A key challenge in both cases is the presence of a wide range of time-scales in the chemistry of the atmosphere. Among different time-parallel integration techniques, we focused on the parareal algorithm in this work. The multi-scale nature of the chemistry forces the usage of adaptive fine and coarse propagators within the parareal algorithm. Further, it leads to load-imbalances in case of equidistant decompositions in time.

For the internal parallelization, adaptive decompositions allowed to balance the load of work. Using adaptive coarse propagators on non-equidistant decompositions led to overheads in the sequential part of the parareal algorithm. Those could be diminished by using coarse propagators, forced to use a single time-step on each interval. Still, all adaptive parareal approaches presented were slower than the sequential reference. This motivated the search for faster, but still accurate, reduced-models. In that context, only repro-modeling approaches qualified as coarse propagators. Adaptive decompositions in time however turned out to be incompatible with such models.

For the external parallelization, the decomposition was prescribed by the size of the splitting intervals. This enabled the usage of repro-models as coarse propagators. These turned out to be in average 25 faster than adaptive coarse propagators, which allowed for an overall speed-up of the parareal algorithm up to $6.2\times$ in comparison to the sequential reference. Three different scenarios with increasing complexity were considered for the external parallelization. For those scenarios, a further challenge has been the incorporation of external disturbances across two subsequent splitting intervals. Exemplarily, we incorporated changing photolysis rates and emission. For a parallel-in-time simulation over 5 days, distributed over 120 processors, speed-ups were observed for all three scenarios. For a simulation over 24 hours and distributed over 24 processors, speed-up was observed only for one scenario representing a non-polluted atmosphere without emission. Due to higher numbers of iterations until convergence, no speed-ups could be achieved for the other scenarios. More sophisticated HDMR models, determined by means of further parameter studies, are likely to improve the quality of a repro-model parareal approach for these scenarios.

In case of the external parallelization, the load-imbalances were caused by diurnally varying photolysis frequencies. These can roughly be estimated a priori, such that the expected computational effort for each time of the day can potentially be derived from empirical values in preface. This would enable an a priori adjustment of the decomposition, such that interval sizes are multiples of each other. Respectively, the repro-models would have to be constructed for the smallest interval size and applied multiple times on bigger intervals. Non-uniform decompositions, adjusted on the base of empirical values, are likely to decrease the load-imbalances on equidistant decompositions and to further improve speed-ups. A truly adaptive decomposition however seems not to be possible.

This work was driven by the search for scalable and load-balancing parallel algorithms for the integration of atmospheric chemical kinetics. Load-balancing, but no speed-ups could be achieved for an internal in-time-parallelization of atmospheric chemical kinetics. Speed-ups could only be achieved for the external parallelization and when considering repro-models as coarse propagators. For the external parallelization, the number of parallel processes equates the size of the interval to be parallelized (i.e. the problem size), divided by the size of the splitting intervals. A scalability study for a fixed problem-size was hindered by the fact, that increasing the size of the splitting intervals changes the physical nature of the problem.

The usage of repro-models as coarse propagators seems to be an attractive alternative to using time-stepping schemes as coarse propagators. For the scenarios considered here, solution trajectories were known a priori. This allowed to construct highly tailored repro-models. For a real application within an AQM, the solution trajectories of the chemical species will not be known in advance. A high degree of expertise and understanding in the chemical systems will be necessary, to define respective input spaces for each species, for which the model shall be valid. Possibly, higher numbers of parametrization points will be required, leading to higher computational costs. On the other side, the full potential to decrease the cost for the evaluation of the repro-model has not yet been exploited. Instead of parameterizing in the phase space of chemical species, one could have alternatively used a parametrization in terms of modes. Then, the number of parameters would be defined by the dimension of the lowest dimensional intrinsic manifold, which typically is a lot smaller than the dimension size. Hence, both the cost for the construction and the evaluation of the repro-model would decrease. In turn, additional matrix vector multiplication would be necessary to map between the space of chemical species and modes.

The promising results presented for the external parallelization suggest, that parallel-in-time techniques could potentially be applied to parallelize chemical transport models. For the numerical approximation of the chemical transport model, one typically applies an operator splitting approach. Instead of solving the full chemical transport equation, one solves simplified problems on short intervals in time, sequentially one after another. In fact, the operator splitting approach is an artificial decoupling and sequentialization of processes, that in reality are parallel. Why not compute those parallel processes in a parareal fashion? A possible approach would be to solve the decoupled problems in parallel on bigger intervals in time, while adding an outer, parareal iteration.

Chemical transport models can typically be found as parts of compound air quality models, which further comprise an interaction with a meteorological model. Chemistry and meteorology are only loosely coupled and solved in parallel. Data is exchanged at so called coupling intervals. A parallelization across the coupling intervals is challenged by the mutual interaction of chemistry and meteorology. Respectively, also the meteorological model would have to be recast in a parareal fashion to account for these interactions.

The observation, that all methods presented for the internal parallelization fail, can be blamed on the overall discretization of the chemistry within an air quality model. In fact, both the coupling between meteorology and chemistry and the operator splitting used for the solution of the chemical transport model intensify the emergence of stiff problems out of balance at the beginning of every splitting interval. This challenge can be damped, when solving the chemical transport models in a fully coupled fashion and when increasing the level of coupling between chemistry and meteorology. Respectively, all parallelization strategies would have to be recast and expanded on the full chemical transport model, or even the full air quality model. Further investigations will be necessary, to estimate, if this effort eventually pays off or not.

Appendix

Acronyms

Acronym	Term
AP	Adaptive Parareal
AQM	Air Quality Model
CBEA	Cell Broadband Engine Architecture
CP	Classical Parareal
CPU	Central Processing Unit
CT	Computing Time
CTM	Chemical Transport Model
DAE	Differential Algebraic Equation
DLP	Data Level Parallelism
FEOM	Fully Equivalent Operational Model
GPU	Graphic Processing Unit
HDMR	High Dimensional Model Representation
HPC	High Performance Computing
ILDM	Intrinsic Low Dimensional Manifold
MCM	Master Chemical Mechanism
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
QSSA	Quasi-Steady State Approximation
RADM2	Regional Acid Decomposition Model Version 2
RM-P	Repro-Model Parareal
SDC	Spectral Deferred Corrections
SIMD	Single-Instruction-Multiple-Data
VOC	Volatile Organic Compounds
WCT	Wall Clock Time

RADM2 Chemical Mechanism

Table 1: RADM2 chemical reactions and rate constants as presented by Stockwell et al. [119].

	Reaction	Reaction rate
1	NO2 = O3P + NO	PHUX(1.03D-02,9.61800D-01,8.46710D-01,Chi)
2	O3 = O1D + O2	PHUX(5.00D-05,3.29332D+00,8.07820D-01,Chi)
3	O3 = O3P + O2	PHUX(5.11D-04,3.71950D-01,9.22890D-01,Chi)
4	HONO = HO + NO	PHUX(2.36D-03,1.06560D+00,8.36440D-01,Chi)
5	HNO3 = HO + NO2	PHUX(8.07D-07,2.30845D+00,8.13640D-01,Chi)
6	HNO4 = HO2 + NO2	PHUX(4.88D-06,2.08052D+00,8.13200D-01,Chi)+EQT2(M,TEMP)
7	NO3 = NO + O2	PHUX(2.59D-02,2.96180D-01,9.37480D-01,Chi)
8	NO3 = NO2 + O3P	PHUX(2.30D-01,3.35180D-01,9.30590D-01,Chi)
9	H2O2 = 2*HO	PHUX(1.18D-05,1.65050D+00,8.16060D-01,Chi)
10	HCHO = H2 + CO	PHUX(5.12D-05,1.44263D+00,8.18510D-01,Chi)
11	HCHO = 2*HO2 + CO	PHUX(4.51D-05,1.81238D+00,8.19300D-01,Chi)
12	ALD = MO2 + HO2 + CO	PHUX(7.49D-06,2.20021D+00,8.15430D-01,Chi)
13	OP1 = HCHO + HO2 + HO	PHUX(6.81D-06,1.60212D+00,8.16880D-01,Chi)
14	OP2 = ALD + HO2 + HO	PHUX(6.81D-06,1.60212D+00,8.16880D-01,Chi)
15	PAA = MO2 + HO	PHUX(1.28D-08,7.94062D+00,7.44350D-01,Chi)
16	KET = ACO3 + ETHP	PHUX(6.46D-07,2.99467D+00,8.09690D-01,Chi)
17	GLY = 0.130*HCHO + 1.870*CO	PHUX(2.89D-03,5.76430D-01,8.90430D-01,Chi)
18	GLY = 0.450*HCHO + 1.550*CO + 0.800*HO2	PHUX(2.89D-03,5.76430D-01,8.90430D-01,Chi)
19	MGLY = ACO3 + HO2 + CO	PHUX(3.15D-03,6.15570D-01,8.85050D-01,Chi)
20	DCB = 0.980*HO2 + 0.020*ACO3 + TCO3	PHUX(6.30D-04,1.27788D+00,8.25020D-01,Chi)
21	ONIT = 0.200*ALD + 0.800*KET + HO2+ NO2	PHUX(1.50D-07,7.85847D+00,7.44730D-01,Chi)
22	O3P + O2 = O3	M * 6.0000E-34 * (TEMP/300)**(-2.30)
23	O3P + NO2 = NO	6.5000E-12 * exp(120.0/TEMP)
24	O1D + N2 = O3P	1.8000E-11 * exp(110.0/TEMP)
25	O1D + O2 = O3P	3.2000E-11 * exp(70.0/TEMP)
26	O1D + H2O = 2*HO	2.20E-010
27	O3 + NO = NO2	2.0000E-12 * exp(-1400.0/TEMP)
28	O3 + HO = HO2	1.6000E-12 * exp(-940.0/TEMP)
29	O3 + HO2 = HO	1.1000E-14 * exp(-500.0/TEMP)
30	HO2 + NO = NO2 + HO	3.7000E-12 * exp(240.0/TEMP)
31	HO2 + NO2 = HNO4	TROE(1.8D-31, 3.2d0, 4.7D-12, 1.4d0, M, TEMP)
32	HO2 + HO2 = H2O2	2.2E-13 * EXP(620./TEMP) + 1.9E-33 * M * EXP(980./TEMP)
33	HO2 + HO2 + H2O = H2O2	3.0800E-34 * exp(2820.0/TEMP) + M * (2.660E-54) * EXP(3180.0/TEMP)
34	H2O2 + HO = HO2	3.3000E-12 * exp(-200.0/TEMP)
35	NO + HO = HONO	TROE(7.0d-31, 2.6d0, 1.5d-11, 0.5d0, M, TEMP)
36	NO + NO + O2 = 2*NO2	3.3000E-39 * exp(530.0/TEMP)
37	O3 + NO2 = NO3	1.4000E-13 * exp(-2500.0/TEMP)
38	NO3 + NO = 2*NO2	1.7000E-11 * exp(150.0/TEMP)
39	NO3 + NO2 = NO + NO2	2.5000E-14 * exp(-1230.0/TEMP)
40	NO3 + HO2 = HNO3	2.50E-012
41	NO3 + NO2 = N2O5	TROE(2.2d-30, 4.3d0, 1.5d-12, 0.5d0, M, TEMP)
42	N2O5 = NO2 + NO3	EQT(2.2d-30, 4.3d0,1.5d-12,0.5d0,M,TEMP,9.09d+26,11200.d0)
43	N2O5 + H2O = 2.000*HNO3	2.00E-021
44	HO + NO2 = HNO3	TROE(2.6d-30, 3.2d0, 2.4d-11, 1.3d0, M, TEMP)
45	HO + HNO3 = NO3	SPEZ(7.2d-15,785.d0,4.1d-16,1440.d0,1.9d-33,725.d0,M,TEMP)
46	HO + HNO4 = NO2	1.3000E-12 * exp(380.0/TEMP)
47	HO + HO2 = DUMMY	4.6000E-11 * exp(230.0/TEMP)
48	HO + SO2 = SULF + HO2	TROE(3.0d-31, 3.3d0, 1.5d-12, 0.0d0, M, TEMP)
49	CO + HO = HO2	1.5E-13 * (1.0 + 2.439E-20 * M)

Table 1: RADM2 chemical reactions and rate constants as presented by Stockwell et al. [119].

	Reaction	Reaction rate
50[1]	CH4 + HO = MO2 + H2O	TEMP * TEMP * 6.95E-18 * EXP(-1280. / TEMP)
51[1]	ETH + HO = ETHP	TEMP * TEMP * 1.37E-17 * EXP(-444. /TEMP)
52	HC3 + HO = 0.830*HC3P + 0.170*HO2 + 0.009*HCHO + 0.075*ALD + 0.025*KET	1.5900E-11 * exp(-540.0/TEMP)
53	HC5 + HO = HC5P + 0.250*XO2	1.7300E-11 * exp(-380.0/TEMP)
54	HC8 + HO = HC8P + 0.750*XO2	3.6400E-11 * exp(-380.0/TEMP)
55	OL2 + HO = OL2P	2.1500E-12 * exp(411.0/TEMP)
56	OLT + HO = OLTP	5.3200E-12 * exp(504.0/TEMP)
57	OLI + HO = OLIP	1.0700E-11 * exp(549.0/TEMP)
58	TOL + HO = 0.750*TOLP + 0.250*CSL + 0.250*HO2	2.1000E-12 * exp(322.0/TEMP)
59	XYL + HO = 0.830*XYLP + 0.170*CSL + 0.170*HO2	1.8900E-11 * exp(116.0/TEMP)
60	CSL + HO = 0.100*HO2 + 0.900*XO2 + 0.900*TCO3	4.00E-011
61	CSL + HO = CSL	9.0000E-01 * 4.0000E-11
62	HCHO + HO = HO2 + CO	9.00E-012
63	ALD + HO = ACO3	6.8700E-12 * exp(256.0/TEMP)
64	KET + HO = KETP	1.2000E-11 * exp(-745.0/TEMP)
65	GLY + HO = HO2 + 2.000*CO	1.15E-011
66	MGLY + HO = ACO3 + CO	1.70E-011
67	DCB + HO = TCO3	2.80E-011
68	OP1 + HO = 0.500*MO2 + 0.500*HCHO + 0.500*HO	1.00E-011
69	OP2 + HO = 0.500*HC3P + 0.500*ALD	1.00E-011
70	PAA + HO = ACO3	1.00E-011
71	PAN + HO = HCHO + NO3 + XO2	6.1650E-13 * (TEMP/300)**2 * exp(-444.0/TEMP)
72	ONIT + HO = HC3P + NO2	1.5500E-11 * exp(-540.0/TEMP)
73	ISO + HO = OLTP	2.5500E-11 * exp(409.0/TEMP)
74	ACO3 + NO2 = PAN	2.8000E-12 * exp(181.0/TEMP)
75	PAN = ACO3 + NO2	1.9500E+16 * exp(-13543.0/TEMP)
76	TCO3 + NO2 = TPAN	4.70E-012
77	TPAN = TCO3 + NO2	1.9500E+16 * exp(-13543.0/TEMP)
78	MO2 + NO = HCHO + HO2 + NO2	4.2000E-12 * exp(180.0/TEMP)
79	HC3P + NO = 0.750*ALD + 0.250*KET + 0.090*HCHO + 0.036*ONIT + 0.964*NO2 + 0.964*HO2	4.2000E-12 * exp(180.0/TEMP)
80	HC5P + NO = 0.380*ALD + 0.690*KET + 0.080*ONIT + 0.920*NO2 + 0.920*HO2	4.2000E-12 * exp(180.0/TEMP)
81	HC8P + NO = 0.350*ALD + 1.060*KET + 0.040*HCHO + 0.240*ONIT + 0.760*NO2 + 0.760*HO2	4.2000E-12 * exp(180.0/TEMP)
82	OL2P + NO = 1.600*HCHO + HO2 + NO2 + 0.200*ALD	4.2000E-12 * exp(180.0/TEMP)
83	OLTP + NO = ALD + HCHO + HO2 + NO2	4.2000E-12 * exp(180.0/TEMP)
84	OLIP + NO = HO2 + 1.450*ALD + 0.280*HCHO + 0.100*KET + NO2	4.2000E-12 * exp(180.0/TEMP)
85	ACO3 + NO = MO2 + NO2	4.2000E-12 * exp(180.0/TEMP)
86	TCO3 + NO = NO2 + 0.920*HO2 + 0.890*GLY + 0.110*MGLY + 0.050*ACO3 + 0.950*CO + 2.000*XO2	4.2000E-12 * exp(180.0/TEMP)
87	TOLP + NO = NO2 + HO2 + 0.170*MGLY + 0.160*GLY + 0.700*DCB	4.2000E-12 * exp(180.0/TEMP)
88	XYLP + NO = NO2 + HO2 + 0.450*MGLY + 0.806*DCB	4.2000E-12 * exp(180.0/TEMP)
89	ETHP + NO = ALD + HO2 + NO2	4.2000E-12 * exp(180.0/TEMP)

[1]Reaction rate as proposed in [118], see also Sec. 6.1.

Table 1: RADM2 chemical reactions and rate constants as presented by Stockwell et al. [119].

	Reaction	Reaction rate
90	KETP + NO = MGLY + NO2 + HO2	4.2000E-12 * exp(180.0/TEMP)
91	OLN + NO = HCHO + ALD + 2.000*NO2	4.2000E-12 * exp(180.0/TEMP)
92	HCHO + NO3 = HO2 + HNO3 + CO	6.0000E-13 * exp(-2058.0/TEMP)
93	ALD + NO3 = ACO3 + HNO3	1.4000E-12 * exp(-1900.0/TEMP)
94	GLY + NO3 = HNO3 + HO2 + 2.000*CO	6.0000E-13 * exp(-2058.0/TEMP)
95	MGLY + NO3 = HNO3 + ACO3 + CO	1.4000E-12 * exp(-1900.0/TEMP)
97	DCB + NO3 = HNO3 + TCO3	1.4000E-12 * exp(-1900.0/TEMP)
98	CSL + NO3 = HNO3 + XNO2 + 0.500*CSL	2.20E-011
99	OL2 + NO3 = OLN	2.0000E-12 * exp(-2923.0/TEMP)
100	OLT + NO3 = OLN	3.2300E-11 * exp(-975.0/TEMP)
101	ISO + NO3 = OLN	5.81E-013
102	OL2 + O3 = HCHO + 0.400*ORA1 + 0.420*CO + 0.120*HO2	1.2E-14 * exp(-2633.0/TEMP)
103	OLT + O3 = 0.530*HCHO + 0.5*ALD + 0.330*CO + 0.2*ORA1 + 0.200*ORA2 + 0.230*HO2 + 0.220*MO2 + 0.100*HO	1.32E-14 * exp(-2105.0/TEMP)
104	OLI + O3 = 0.180*HCHO + 0.720*ALD + 0.1*KET + 0.23*CO + 0.06*ORA1 + 0.29ORA2 + 0.26*HO2 + 0.14*HO + 0.31*MO2	7.29E-15 * exp(-1136.0/TEMP)
105	ISO + O3 = 0.53*HCHO + 0.5*ALD + 0.330*CO + 0.20*ORA1 + 0.2*ORA2 + 0.230*HO2 + 0.22*MO2 + 0.1*HO	1.23E-14 * exp(-2013.0/TEMP)
106	HO2 + MO2 = OP1	7.7E-14 * exp(1300.0/TEMP)
107	HO2 + ETHP = OP2	7.7E-14 * exp(1300.0/TEMP)
108	HO2 + HC3P = OP2	7.7E-14 * exp(1300.0/TEMP)
109	HO2 + HC5P = OP2	7.7E-14 * exp(1300.0/TEMP)
110	HO2 + HC8P = OP2	7.7E-14 * exp(1300.0/TEMP)
111	HO2 + OL2P = OP2	7.7E-14 * exp(1300.0/TEMP)
112	HO2 + OLTP = OP2	7.7E-14 * exp(1300.0/TEMP)
113	HO2 + OLIP = OP2	7.7E-14 * exp(1300.0/TEMP)
114	HO2 + KETP = OP2	7.7E-14 * exp(1300.0/TEMP)
115	HO2 + ACO3 = PAA	7.7E-14 * exp(1300.0/TEMP)
116	HO2 + TOLP = OP2	7.7E-14 * exp(1300.0/TEMP)
117	HO2 + XYLP = OP2	7.7E-14 * exp(1300.0/TEMP)
118	HO2 + TCO3 = OP2	7.7E-14 * exp(1300.0/TEMP)
119	HO2 + OLN = ONIT	7.7E-14 * exp(1300.0/TEMP)
120	MO2 + MO2 = 1.500*HCHO + HO2	1.9E-13 * exp(220.0/TEMP)
121	MO2 + ETHP = 0.750*HCHO + HO2 + 0.750*ALD	1.4E-13 * exp(220.0/TEMP)
122	MO2 + HC3P = 0.840*HCHO + 0.770*ALD + 0.260*KET + HO2	4.2E-14 * exp(220.0/TEMP)
123	MO2 + HC5P = 0.770*HCHO + 0.410*ALD + 0.750*KET + HO2	3.4E-14 * exp(220.0/TEMP)
124	MO2 + HC8P = 0.800*HCHO + 0.460*ALD + 1.390*KET + HO2	2.9E-14 * exp(220.0/TEMP)
125	MO2 + OL2P = 1.550*HCHO + 0.350*ALD + HO2	1.4E-13 * exp(220.0/TEMP)
126	MO2 + OLTP = 1.250*HCHO + 0.750*ALD + HO2	1.4E-13 * exp(220.0/TEMP)
127	MO2 + OLIP = 0.890*HCHO + 0.725*ALD + HO2 + 0.550*KET	1.7E-14 * exp(220.0/TEMP)
128	MO2 + KETP = 0.750*HCHO + 0.750*MGLY + HO2	1.7E-14 * exp(220.0/TEMP)
129	MO2 + ACO3 = HCHO + 0.5*HO2 + 0.5*MO2 + 0.5*ORA2	9.6E-13 * exp(220.0/TEMP)
130	MO2 + TOLP = HCHO + 0.170*MGLY + 0.16*GLY + 0.7*DCB + 2*HO2	1.7E-14 * exp(220.0/TEMP)

Table 1: RADM2 chemical reactions and rate constants as presented by Stockwell et al. [119].

	Reaction	Reaction rate
131	MO2 + XYLP = HCHO + 0.450*MGLY + 0.806*DCB + 2*HO2	1.7E-14 * exp(220.0/TEMP)
132	MO2 + TCO3 = 0.500*HCHO + 0.445*GLY + 0.055*MGLY + 0.5*ORA2 + 0.025*ACO3 + 0.460*HO2 + 0.475*CO + XO2	9.6E-13 * exp(220.0/TEMP)
133	MO2 + OLN = 1.750*HCHO + 0.500*HO2 + ALD + NO2	1.7E-14 * exp(220.0/TEMP)
134	ETHP + ACO3 = ALD + 0.500*HO2 + 0.500*MO2 + 0.500*ORA2	3.4E-13 * exp(220.0/TEMP)
135	HC3P + ACO3 = 0.770*ALD + 0.260*KET + 0.5*HO2 + 0.5*MO2 + 0.5*ORA2	1.E-13 * exp(220.0/TEMP)
136	HC5P + ACO3 = 0.410*ALD + 0.750*KET + 0.5*HO2 + 0.5*MO2 + 0.5*ORA2	8.4E-14 * exp(220.0/TEMP)
137	HC8P + ACO3 = 0.460*ALD + 1.390*KET + 0.5*HO2 + 0.5*MO2 + 0.5*ORA2	7.2E-14 * exp(220.0/TEMP)
138	OL2P + ACO3 = 0.8*HCHO + 0.6*ALD + 0.5*HO2 + 0.5*MO2 + 0.5*ORA2	3.4E-13 * exp(220.0/TEMP)
139	OLTP + ACO3 = ALD + 0.5*HCHO + 0.5*HO2 + 0.5*MO2 + 0.5*ORA2	3.4E-13 * exp(220.0/TEMP)
140	OLIP + ACO3 = 0.725*ALD + 0.550*KET + 0.140*HCHO + 0.500*HO2 + 0.500*MO2 + 0.500*ORA2	4.2000E-14 * exp(220.0/TEMP)
141	KETP + ACO3 = MGLY + 0.5*HO2 + 0.5*MO2 + 0.5*ORA2	4.2000E-14 * exp(220.0/TEMP)
142	ACO3 + ACO3 = 2.000*MO2	1.1900E-12 * exp(220.0/TEMP)
143	ACO3 + TOLP = MO2 + 0.170*MGLY + 0.160*GLY + 0.700*DCB + HO2	4.2000E-14 * exp(220.0/TEMP)
144	ACO3 + XYLP = MO2 + 0.450*MGLY + 0.806*DCB + HO2	4.2000E-14 * exp(220.0/TEMP)
145	ACO3 + TCO3 = MO2 + 0.920*HO2 + 0.890*GLY + 0.110*MGLY + 0.05*ACO3 + 0.950*CO + 2*XO2	1.1900E-12 * exp(220.0/TEMP)
146	ACO3 + OLN = HCHO + ALD + 0.5*ORA2 + NO2 + 0.5*MO2	4.2000E-14 * exp(220.0/TEMP)
147	OLN + OLN = 2*HCHO + 2*ALD + 2*NO2	3.6000E-16 * exp(220.0/TEMP)
148	XO2 + HO2 = OP2	7.7000E-14 * exp(1300.0/TEMP)
150	XO2 + MO2 = HCHO + HO2	4.2000E-14 * exp(220.0/TEMP)
151	XO2 + XO2 = DUMMY	3.6000E-16 * exp(220.0/TEMP)
152	XO2 + NO = NO2	4.2000E-12 * exp(180.0/TEMP)
153	XNO2 + NO2 = ONIT	4.2000E-12 * exp(180.0/TEMP)
154	XNO2 + HO2 = OP2	7.7000E-14 * exp(1300.0/TEMP)
155	XNO2 + MO2 = HCHO + HO2	1.7000E-14 * exp(220.0/TEMP)
156	XNO2 + ACO3 = MO2	4.2000E-14 * exp(220.0/TEMP)
157	XNO2 + XNO2 = DUMMY	3.6000E-16 * exp(220.0/TEMP)

Table 2: RADM2 chemical species, c.f. Stockwell et al. [119].

	Species	Description
Stable Inorganic Compounds		
Nitrogen		
1	NO2	Nitrogen dioxid
2	NO	Nitric oxide
3	HONO	Nitrous acid
4	NO3	Nitrogen trioxide
5	N2O5	Nitrogen pentoxide
6	HNO4	Pemitric acid
7	HNO3	Nitric acid
Oxidants		
8	O3	Ozone
9	H2O2	Hydrogen peroxide
Sulfur		
10	SO2	Sulfur dioxide
11	SULF	Sulfuric acid
Carbon Oxides		
12	CO	Carbon monoxide
13	CO2	Carbon dioxide
14	H2	Hydrogen
Inorganic Short-Lived Intermediates		
Atomic species		
15	O3P	Ground state oxygen atom
16	O1D	Excited state oxygen atom
Odd hydrogen		
17	HO	Hydroxyl radical
18	HO2	Hydroperoxyl radical
Abundant Stable Species		
19	O2	Oxygen
20	N2	Nitrogen
21	H2O	Water
Stable Organic Compounds		
Alkanes		
22	CH4	Methane
23	ETH	Ethane
24	HC3	Alkanes with HO rate constant (298K, 1atm) between 2.7x10-13 and 3.4x10-12
25	HC5	Alkanes with HO rate constant (298K, 1atm) between 3.4x10-12 and 6.8x10-12
26	HC8	Alkanes with HO rate constant (298K, 1atm) greater than 6.8x10-12
Alkenes		
27	OL2	Ethene
28	OLT	Terminal alkenes
29	OLI	Intemal alkenes
30	ISO	Isoprene
Aromatics		
31	TOL	Toluene and less reactive aromatics
32	CSL	Cresol and other hydroxy substituted aromatics
33	XYL	Xylene and more reactive aromatics
Carbonyls		
34	HCHO	Formaldehyde
35	ALD	Acetaldehyde and higher aldehydes
36	KET	Ketones
37	GLY	Glyoxal
38	MGLY	Methylglyoxal
39	DCB	Unsaturated dicarbonyl
Organic nitrogen		
40	PAN	Peroxyacetyl nitrate and higher PANs
41	TPAN	H(CO)CH=CHCO3NO2
42	ONIT	Organic nitrate
Organic peroxides		
43	OP1	Methyl hydrogen peroxide
44	OP2	Higher organic peroxides
45	PAA	Peroxyacetic acid
Organic acids		
46	ORA1	Formic acid
47	ORA2	Acetic acid and higher acids
Organic Short-Lived Intermediates		
Peroxy radicals from alkanes		
48	MO2	Methyl peroxy radical
49	ETHP	Peroxy radical formed from ETH
50	HC3P	Peroxy radical formed from HC3
51	HC5P	Peroxy radical formed from HC5
52	HC8P	Peroxy radical formed from HC8
Peroxy radicals from alkenes		
53	OL2P	Peroxy radical formed from OL2
54	OLTP	Peroxy radical formed from OLT
55	OLIP	Peroxy radical formed from OLIP
Peroxy radicals from aromatics		
56	TOLP	Peroxy radical formed from TOL
57	XYLP	Peroxy radical formed from XYL
Peroxy radicals with carbonyl groups		
58	ACO3	Acetylperoxy radical
59	KETP	Peroxy radical formed from KET
60	TCO3	H(CO)CH=CHCO3
Peroxy radicals involving nitrogen		
61	OLN	NO3-alkene adduct
62	XNO2	Accounts for additional organic nitrate formation affected by the lumped organic species
63	XO2	Accounts for additional NO to NO2 conversions affected by the lumped organic species

Listing 1: Calculation of photolysis rates in 0-d boxmodel. Extracted from Kuhn et al. [56].

```
C
C******************************************************************
C CALCULATION OF PHOTOLYSIS FREQUENCY WITH ALGORITHM FROM ROETHS FLUX-PROGRAM
C CHI IN RADIAN
C X,Y,Z FROM ROETH
C X IS PHOTOLYSIS FREQUENCY IN 1/S FOR SOLAR ZENITH ANGLE CHI=0 DEG
C AUTHOR: KUHN 07.09.93
C******************************************************************
C
FUNCTION PHUX(X,Y,Z,CHI)
      REAL*8 X,Y,Z,CHI,CHIZ,YCHIZ,MINYZ,EYCHIZ,EMINYZ,PHUX
      PARAMETER (MINYZ = -30, EMINYZ = 9.357623D-14 ) !EMINYZ=EXP(MINYZ)
      IF (CHIZ.LT.1.57079632679489D0) THEN
         YCHIZ = Y * (1.0 - (1.0/ COS(CHIZ) ))
         IF (YCHIZ.GT.MINYZ) THEN
            EYCHIZ = DEXP (YCHIZ)
         ELSE
            EYCHIZ = EMINYZ
         ENDIF
      ELSE
         EYCHIZ = EMINYZ
      ENDIF
      PHUX = X * EYCHIZ
END FUNCTION PHUX

C
C******************************************************************
C     CALCULATION OF SOLAR ZENITH ANGLE CHI
C******************************************************************
C
FUNCTION CHIBE(ZEIT,GB,GL,TAG)
      REAL(dp) :: CHIBE,ZEIT,GB,GL,TAG,SD,CD,FP,OM,SH,SH1,CHI,DEKL
      FP = 0.01745329E0
      DEKL = 23.45E0*SIN((280.1E0+0.981E0*TAG)*0.01745329E0)
      SD = SIN(DEKL*FP)
      CD = COS(DEKL*FP)
      OM = (12.E0-(ZEIT/60.+GL/15.E0))*0.261799388E0
      SH = SIN(GB*FP)*SD+COS(GB*FP)*CD*COS(OM)
      SH1 = ASIN(SH)*180./3.141592
      CHI=90.-SH1
      CHI=3.141592/180.*CHI
      CHIBE=CHI
END FUNCTION CHIBE
```

```
C
C*****************************************************************
C CALCULATION OF REACTION RATES for pressure depending rate constants
C according to Stockwell et al.
C*****************************************************************
C
FUNCTION TROE( K0300, Q, KU300, R, M, T )
       REAL(dp) :: K0300, Q, KU300, R, M, T, TROE
       REAL(dp) :: tt, k0, ku, k0m, kk, lgkk, e, f
       TT= T / 3.D2
       K0= K0300 / TT**Q
       KU= KU300 / TT**R
       K0M= K0 * M
       KK= K0M / KU
       LGKK=0.434294481D0 * LOG(KK)
       E=1.D0 / ( 1.D0 + LGKK*LGKK )
       F=0.6D0 ** E
       TROE= F * K0M / ( 1.D0 + KK )
END FUNCTION TROE

C
C*****************************************************************
C CALCULATION OF REACTION RATE for N2O5 --> NO2 + NO3
C*****************************************************************
C
FUNCTION EQT( K0300, Q, KU300, R, M, T, A, B )
       REAL(dp) :: K0300, Q, KU300, R, M, T, A, B, EQT
       REAL(dp) :: kh
       KH= TROE( K0300, Q, KU300, R, M, T )
       EQT= KH * A *DEXP( -B / T )
END FUNCTION EQT

C
C*****************************************************************
C CALCULATION OF REACTION RATE
C for HNO4   --> HO2 + NO2
C*****************************************************************
C
FUNCTION EQT2(M, T)
       REAL(dp) :: K0300, Q, KU300, R, M, T, A, B, EQT2
       REAL(dp) :: kh
       K0300=1.8d-31
       Q=3.2d0
       KU300=4.7d-12
       R=1.4d0
       A=4.76d+26
       B=10900.d0
       KH= TROE( K0300, Q, KU300, R, M, T )
       EQT2= KH * A *DEXP( -B / T )
END FUNCTION EQT2
```

```
C
C*****************************************************************
C CALCULATION OF REACTION RATE for HNO3 + HO --> H2O + NO3
C*****************************************************************
C
FUNCTION SPEZ(A0,B0,A2,B2,A3,B3,M,T)
      REAL(dp) :: A0,B0,A2,B2,A3,B3,M,T, SPEZ
      REAL(dp) :: k0, k2, k3
      K0=A0*DEXP(B0/T)
      K2=A2*DEXP(B2/T)
      K3=A3*M*DEXP(B3/T)
      SPEZ= K0 + K3 / ( 1 + K3/K2 )
END FUNCTION SPEZ

C
C*****************************************************************
C CALCULATION OF EMISSIONS FOR SCENARIOS URBAN AND PLUME
C*****************************************************************
C
SUBROUTINE EMISSION
      C(ind_NO)      = C(ind_NO)      + 1.1e6*DT              ! NO
      C(ind_SO2)     = C(ind_SO2)     + 2.2e5*DT              ! SO2
      C(ind_CO)      = C(ind_CO)      + 2.4e5*DT              ! CO
      C(ind_ALD)     = C(ind_ALD)     + 0.0051221*3.0e6*DT ! ALD
      C(ind_HCHO)    = C(ind_HCHO)    + 0.0196705*3.0e6*DT ! HCHO
      C(ind_HC3)     = C(ind_HC3)     + 0.3633745*3.0e6*DT ! HC3
      C(ind_HC5)     = C(ind_HC5)     + 0.1076187*3.0e6*DT ! HC5
      C(ind_HC8)     = C(ind_HC8)     + 0.0643181*3.0e6*DT ! HC8
      C(ind_ETH)     = C(ind_ETH)     + 0.0340956*3.0e6*DT ! ETH
      C(ind_OL2)     = C(ind_OL2)     + 0.0646318*3.0e6*DT ! OL2
      C(ind_OLT)     = C(ind_OLT)     + 0.0309905*3.0e6*DT ! OLT
      C(ind_OLI)     = C(ind_OLI)     + 0.0265693*3.0e6*DT ! OLI
      C(ind_TOL)     = C(ind_TOL)     + 0.0811276*3.0e6*DT ! TOL
      C(ind_XYL)     = C(ind_XYL)     + 0.0734949*3.0e6*DT ! XYL
      C(ind_KET)     = C(ind_KET)     + 0.0453156*3.0e6*DT ! KET
END SUBROUTINE EMISSION
```

Convergence of the Parareal Algorithm

Theorem. *At any parareal iteration $k \geq 1$ the parareal solution at all times $n \leq k$ equate the solution of the fine propagator $\mathcal{F}_{\Delta t}$, i.e.*

$$U_n^k = \mathcal{F}_{\Delta t}^n(u_0), \quad \forall n \leq k \quad \textit{with } \mathcal{F}_{\Delta t}^n(u_0) = \underbrace{\mathcal{F}_{\Delta t} \circ \ldots \circ \mathcal{F}_{\Delta t}}_{n \textit{ times}}(u_0). \tag{1}$$

Proof. By induction.

1. Base Case:

   Consider $k = 1$: The parareal scheme (3.2.2) at iteration $k = 1$ reads

   $$U_n^1 = \mathcal{G}_{\Delta t}(U_{n-1}^1) + (\mathcal{F}_{\Delta t} - \mathcal{G}_{\Delta t})\,(U_{n-1}^0). \tag{2}$$

   We distinguish two cases:

   – $n = 0$: Since $U_0^k = u_0$ for all k, it must also be $U_0^1 = u_0 = \mathcal{F}_{\Delta t}^0(u_0)$.

   – $n = 1$: We insert $U_0^1 = U_0^0 = u_0$ into Eq. (2) and

   $$\begin{aligned} U_1^1 &= \mathcal{G}_{\Delta t}(u_0) + \mathcal{F}_{\Delta t}(u_0) - \mathcal{G}_{\Delta t}(u_0) \\ &= \mathcal{F}_{\Delta t}^1(u_0). \end{aligned}$$

   Eq. (1) is true for $k = 1$ and $n = 0$ and $n = 1$.

2. Induction Step:

   Now assume, that Eq. (1) holds for all $k \geq 1$. Then it must also hold for all $k \to k + 1$. The parareal scheme at iteration $k + 1$ reads

   $$U_n^{k+1} = \mathcal{G}_{\Delta t}(U_{n-1}^{k+1}) + (\mathcal{F}_{\Delta t} - \mathcal{G}_{\Delta t})\,(U_{n-1}^k) \quad \forall n. \tag{3}$$

   As we have already proven Eq. (1) for $k = 0$ and $k = 1$, we only have to consider all $1 < n \leq k+1$. Since we assume Eq. (1) to hold for all $n \leq k$, it is $\mathcal{F}_{\Delta t}(U_{n-1}^k) = \mathcal{F}_{\Delta t}^n(u_0)$ and therefore also

   $$U_n^{k+1} = \mathcal{F}_{\Delta t}^n(u_0) + \mathcal{G}_{\Delta t}\left(U_{n-1}^{k+1} - U_{n-1}^k\right).$$

   We are left to show, that

   $$U_{n-1}^{\hat{k}+1} = U_{n-1}^{\hat{k}} \quad \forall 1 + k \geq 1 + \hat{k} \geq n \geq 1, \tag{4}$$

   which we will proof in a double nested inner induction.

   – Consider $\hat{k} = 1$:

   · $n = 1$: $U_0^2 = U_0^1 = u_0$

   · $n = 2$: $U_1^2 = \mathcal{G}_{\Delta t}(U_0^2) + (\mathcal{F}_{\Delta t} - \mathcal{G}_{\Delta t})\,(U_0^1) = \mathcal{F}_{\Delta t}U_0^1 = U_1^1$

   – We assume that Eq. (4) holds for $\hat{k}$. Then it must also hold for $\hat{k} = \hat{k} + 1$.

- $n = 1$:
$$U_0^{\hat{k}+1} = U_0^{\hat{k}} = u_0$$

- $n = n + 1$: We assume Eq. (4) holds for n. Then it must also hold for $n = n + 1 \leq \hat{k} + 1$:
$$\begin{aligned} U_{n-1}^{\hat{k}+1} &= \mathcal{G}_{\Delta t}(U_{n-2}^{\hat{k}+1}) + (\mathcal{F}_{\Delta t} - \mathcal{G}_{\Delta t})\,(U_{n-2}^{\hat{k}}) \\ &= \mathcal{F}_{\Delta t}^{n-1} u_0 + \mathcal{G}_{\Delta t}(U_{n-2}^{\hat{k}+1} - U_{n-2}^{\hat{k}}). \end{aligned}$$

  Since we assume Eq. (4) to hold for $\hat{k}$, it must be $U_{n-2}^{\hat{k}+1} - U_{n-2}^{\hat{k}}$ and therefore it is
$$U_{n-1}^{\hat{k}+1} = \mathcal{F}_{\Delta t}^{n-1} u_0.$$

  Further, we use that $\mathcal{F}_{\Delta t}^{n-1} u_0 = U_{n-1}^{k}$ for all $n \leq k$. Then it is
$$U_{n-1}^{\hat{k}+1} = U_{n-1}^{\hat{k}} \quad \forall 1 + k \geq 1 + \hat{k} \geq n \geq 1$$

  and Eq. (4) holds for all $\hat{k} + 1$.

3. Conclusion:

   With the equality $U_{n-1}^{\hat{k}+1} = U_{n-1}^{\hat{k}}$ shown for all $1 + k \geq 1 + \hat{k} \geq n \geq 1$, it is shown, that Eq. (1) holds for all $k \to k + 1$ and the proof of Eq. (1) is finished.

$\square$

Bibliography

[1] Protocol scenarios for modelling of multi-phase tropospheric chemistry, version 2. `http://www.fz-juelich.de/iek/iek-8/EN/Research/Modelling/RegionalAndInverseModelling/CMD/Downloads/MULT_pdf.pdf`.

[2] R.C. Aiken. *Stiff Computation*. Oxford University Press, New York, 1985.

[3] S.P. Arya. *Air Pollution Meteorology and Dispersion*. Oxford University Press, New York, 1999.

[4] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25:151–167, 1997.

[5] R. Atkinson. Kinetics and mechanisms of the gas-phase reactions of the hydroxyl radical with organic compounds under atmospheric conditions. *Chemical Reviews*, 86:69–201, 1986.

[6] E. Aubanel. Scheduling of tasks in the parareal algorithm. *Parallel Computing*, 37:172–182, 2011.

[7] L. Baffico, S. Bernard, Y. Maday, G. Turinici, and G. Zérah. Parallel-in-time molecular-dynamics simulations. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 66:3–6, 2002.

[8] A. Baklanov, K. Schlünzen, P. Suppan, J. Baldasano, D. Brunner, S. Aksoyoglu, G. Carmichael, J. Douros, J. Flemming, R. Forkel, S. Galmarini, M. Gauss, G. Grell, M. Hirtl, S. Joffre, O. Jorba, E. Kaas, M. Kaasik, G. Kallos, X. Kong, U. Korsholm, A. Kurganskiy, J. Kushta, U. Lohmann, A. Mahura, A. Manders-Groot, A. Maurizi, N. Moussiopoulos, S. T. Rao, N. Savage, C. Seigneur, R. S. Sokhi, E. Solazzo, S. Solomos, B. Sørensen, G. Tsegas, E. Vignati, B. Vogel, and Y. Zhang. Online coupled regional meteorology chemistry models in europe: current status and prospects. *Atmospheric Chemistry and Physics*, 14(1):317–398, 2014.

[9] G. Bal. Parallelization in time of (stochastic) ordinary differential equations. *(unpublished)*, 2003.

[10] G. Bal and Y. Maday. A parareal time discretization for nonlinear pde's with applications to the pricing of and American put. In *Recent Developments in Domain Decomposition Methods*, volume 23, pages 189–202. 2002.

[11] B.P. Belousov. Periodically acting reaction and its mechanism. *Collection of Abstracts on Radiation Medicine*, 147, 1958.

[12] C. Bendtsen. A parallel stiff ODE solver based on MIRKs. *Advances in Computational Mathematics*, 7:27–36, 1997.

[13] L.A. Berry, W. Elwasif, J.M. Reynolds-Barredo, D. Samaddar, R. Sanchez, and D.E. Newman. Event-based parareal: A data-flow based implementation of parareal. *Journal of Computational Physics*, 231:5945–5954, 2012.

[14] A. Blouza, L. Boudin, and S.M. Kaber. Parallel in time algorithms with reduction methods for solving chemical kinetics. *Communications in Applied Mathematics and Computational Science*, 5:93–115, 2010.

[15] C. Bolley and M. Crouzeix. Consérvation de la positivité lors de la discrétisation des problèmes d'évolution paraboliques. *RAIRO Analyse Numérique*, 12:81–88, 1978.

[16] J. Boutahar and B. Sportisse. Reduction methods and uncertainty propagation: application to a Chemistry-Transport-Model. In *Proceedings of the TAM-TAM conference*, volume 1, pages 1–6, 2005.

[17] S. Bu and J.Y. Lee. An enhanced parareal algorithm based on the deferred correction methods for a stiff system. *Journal of Computational and Applied Mathematics*, 255:297–305, 2014.

[18] A. Büki, T. Perger, T. Turányi, and U. Maas. Repro-modelling based generation of intrinsic low-dimensional manifolds. *Journal of Mathematical Chemistry*, 31(4):345–362, 2002.

[19] K. Burrage. *Parallel and sequential methods for ordinary differential equations.* The Clarendon Press Oxford University Press, Burlington, MA, USA, 1995.

[20] X.N. Cao, S.F. Li, and D.G. Liu. Modified parallel Rosenbrock methods for stiff differential equations. *Journal of Computational Mathematics*, 20:23–34, 2002.

[21] W.P.L. Carter. Documentation of the SAPRC-99 Chemical Mechanism for VOC Reactivity Assessment. 2000.

[22] W.P.L. Carter. Implementation of the SAPRC-99 Chemical Mechanism into the Models-3 Framework. 2000.

[23] W.P.L. Carter. Development of the SAPRC-07 chemical mechanism. *Atmospheric Environment*, 44(40):5324–5335, 2010.

[24] B. Cumming, G. Fourestey, O. Fuhrer, T. Gysi, M. Fatica, and T.C. Schulthess. Application centric energy-efficiency study of distributed multi-core and hybrid cpu-gpu systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 819–829, Piscataway, NJ, USA, 2014. IEEE Press.

[25] C.F. Curtiss and J.O. Hirschfelder. Integration of stiff equations. In *Proceedings of the National Academy of Sciences US*, volume 38, pages 235–243, 1952.

[26] G. Dahlquist. Convergence and stability in the numerical integration of ordinary differential equations. *Mathematica Scandinavica*, 4:33–53, 1956.

[27] G.G. Dahlquist. A special stability problem for linear multistep methods. *BIT Numerical Mathematics*, 3(1):27–43, 1963.

[28] K. Dekker and J.G. Verwer. *Stability of Runge-Kutta methods for stiff nonlinear differential equations.* North-Holland, Amsterdam, 1984.

[29] R. Djouad and B. Sportisse. Partitioning techniques and lumping computation for reducing chemical kinetics. APLA: An automatic partitioning and lumping algorithm. *Applied Numerical Mathematics*, 43(4):383–398, 2002.

[30] A.M. Dunker. The Reduction and Parametrization of Chemical Mechanisms for Inclusion in Atmospheric Reaction-Transport-Models. *Atmospheric Environment*, 20(3):479–486, 1985.

[31] A. Dutt, L. Greengard, and V. Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT Numerical Mathematics*, 40(2):241–266, 2000.

[32] B.L. Ehle. On padé approximations to the exponential function and a-stable methods for the numerical solution of initial value problems. *Research report CSRR 2010*, 1969.

[33] N. Fenichel. Geometric singular perturbation theory for ordinary differential equations. *Journal of Differential Equations*, 31(1):53–98, 1979.

[34] P.F. Fischer, F. Hecht, and Y. Maday. A parareal in time semi-implicit approximation of the Navier-Stokes equations. In *Domain Decomposition Methods in Science and Engineering XI*, volume 40. 2005.

[35] M.J. Gander. 50 years of Time Parallel Time Integration. In *Multiple Shooting and Time Domain Decomposition.* Springer, 2015.

[36] M.J. Gander and E. Hairer. Nonlinear convergence analysis for the parareal algorithm. *Lecture Notes in Computational Science and Engineering*, 60:45–56, 2008.

[37] M.J. Gander and S. Vandewalle. Analysis of the Parareal Time-Parallel Time-Integration Method. *SIAM Journal on Scientific Computing*, 29:556–578, 2007.

[38] M.J. Gander and S. Vandewalle. On the superlinear and linear convergence of the parareal algorithm. In *Domain Decomposition Methods in Science and Engineering XVI*, volume 55 of *Lecture Notes in Computational Science and Engineering*, pages 291–298. Springer, 2007.

[39] M.J. Gander and S.G. Vandewalle. On the superlinear and linear convergence of the parareal algorithm. *Domain Decomposition Methods in Science and Engineering XVI*, 55:291–298, 2007.

[40] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1971.

[41] M.W. Gery, G.Z. Whitten, J.P. Killus, and M.C. Dodge. A photochemical kinetics mechanism for urban and regional scale computer modeling. *Journal of Geophysical Research: Atmospheres*, 94(D10):12925–12956, 1989.

[42] A. Gross and W.R. Stockwell. Comparison of the EMEP, RADM2 and RACM mechanisms. *Journal of Atmospheric Chemistry*, 44:151–170, 2003.

[43] D. Guibert and D. Tromeur-Dervout. Parallel adaptive time domain decomposition for stiff systems of ODEs/DAEs. *Computers and Structures*, 85:553–562, 2007.

[44] W. Hackbusch. Parabolic multi-grid methods. In *Proceedings of the Sixth International Symposium on Computing Methods in Applied Sciences*, pages 198–197, 1985.

[45] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems.* Springer series in computational mathematics. Springer, second edition, 2009.

[46] E. Hairer and G. Wanner. *Solving ordinary differential equations II: Stiff and differential-algebraic problems.* Springer series in computational mathematics. Springer, second edition, 1996.

[47] E. Hesstvedt, Ø. Hov, and I.S.A. Isaksen. Quasi-steady-state approximations in air pollution modeling: Comparison of two numerical schemes for oxidant prediction. *International Journal of Chemical Kinetics*, 10:971–994, 1978.

[48] J.R. Holton. *An introduction to dynamic meteorology.* Elsevier Academic Press, Amsterdam, Boston, Heidelberg, 2004.

[49] J. Huang, J. Jia, and M. Minion. Accelerating the convergence of spectral deferred correction methods. *Journal of Computational Physics*, 214:633–656, 2006.

[50] L.O. Jay, A. Sandu, F.A. Potra, and G.R. Carmichael. Improved QSSA methods for atmospheric chemistry integration. *SIAM Journal on Scientific Computing*, 18(1):182–202, 1997.

[51] M.E. Jenkin, S.M. Saunders, and M.J. Pilling. The tropospheric degredation of volatile organic compounds: a protocol for mechanism developmen. *Atmospheric Environment*, 31:81–104, 1997.

[52] M.E. Jenkin, S.M. Saunders, V. Wagner, and M.J. Pilling. Protocol for the development of the Master Chemical Mechanism, MCM v3 (Part B): tropospheric degradation of aromatic volatile organic compounds. *Atmospheric Chemistry and Physics*, 3(1):181–193, 2003.

[53] J.A. Kerr, J.G. Calvert, and Atmospheric Sciences Research Laboratory. *Chemical transformation modules for Eulerian acid deposition models*. U.S. Environmental Protection Agency, Atmospheric Sciences Research Laboratory Research Triangle Park, NC, 1985.

[54] B.M.S. Khalaf and D. Hutchinson. Parallel algorithms for initial value problems: Parallel shooting. *Parallel Computing*, 18(6):661–673, 1992.

[55] M. Kiehl. Parallel multiple shooting for the solution of initial value problems. *Parallel Computing*, 20(3):275–295, 1994.

[56] M. Kuhn, P.J.H. Builtjes, D. Poppe, D. Simpson, W.R. Stockwell, Y. Andersson-Sköld, A. Baart, M. Das, F. Fiedler, Hov, F. Kirchner, P.A. Makar, J.B. Milford, M.G.M. Roemer, R. Ruhnke, A. Strand, B. Vogel, and H. Vogel. Intercomparison of the gas-phase chemistry in several chemistry and transport models. *Atmospheric Environment*, 32:693–709, 1998.

[57] K. Kunisch and S. Volkwein. Control of the Burgers Equation by a Reduced-Order Approach Using Proper Orthogonal Decomposition 1. *Journal of Optimization and Application*, 102(2):345–371, 1999.

[58] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numerische Mathematik*, 148:117–148, 2001.

[59] S.H. Lam and D.A. Goussis. Understanding complex chemical kinetics with computational singular perturbation. *Symposium (International) on Combustion*, 22:931–941, 1989.

[60] J.D. Lambert. *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley & Sons, Inc., New York, 1991.

[61] D. Lanser and J.G. Verwer. Analysis of operator splitting for advection-diffusion-reaction problems from air pollution modelling. *Journal of Computational and Applied Mathematics*, 111(1–2):201–216, 1999.

[62] F. Legoll, T. Lelievre, and G. Samaey. A micro-macro parareal algorithm: application to singularly perturbed ordinary differential equations. *SIAM Journal on Scientific Computing*, 35:A1951–A1986, 2013.

[63] E. Lelarasmee, A.E. Ruehli, and A.L. Sangiovanni-Vincentelli. The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1, 1982.

[64] J.A. Leone and J.H. Seinfeld. Comparative analysis of chemical reaction mechanisms for photochemical smog. *Atmospheric Environment*, 19:437–464, 1985.

[65] G. Li and H. Rabitz. A general analysis of exact lumping in chemical kinetics. *Chemical Engineering Science*, 44(6):1413–1430, 1989.

[66] G. Li and H. Rabitz. A general analysis of approximate lumping in chemical kinetics. *Chemical Engineering Science*, 45(4):977–1002, 1990.

[67] G. Li and H. Rabitz. Determination of constrained lumping schemes for nonisothermal first-order reaction systems. *Chemical Engineering Science*, 46(2):583–596, 1991.

[68] G. Li and H. Rabitz. New approaches to determination of constrained lumping schemes for a reaction system in the whole composition space. *Chemical Engineering Science*, 46(1):95–111, 1991.

[69] J. Linford and A. Sandu. Scalable parallelism for atmospheric modeling and simulation. *Journal of Supercomputing*, 56:300–327, 2010.

[70] J.C. Linford. *Accelerating Atmospheric Modeling Through Emerging Multi-core Technologies Emerging Multi-core Technologies.* PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA, 2010.

[71] J.C. Linford, J. Michalakes, M. Vachharajani, and A. Sandu. Multi-core acceleration of chemical kinetics for simulation and prediction. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis - SC '09*, page 1. ACM Press, 2009.

[72] J.C. Linford, J. Michalakes, M. Vachharajani, and A. Sandu. Automatic Generation of Multicore Chemical Kernels. *IEEE Transactions on Parallel and Distributed Systems*, 22(1):119–131, 2011.

[73] J.C. Linford and A. Sandu. Optimizing large scale chemical transport models for multicore platforms. In *SpringSim '08: Proceedings of the 2008 Spring simulation multiconference*, pages 369–376, 2008.

[74] J.-L. Lions, Y. Maday, and G. Turinici. A "parareal" in time discretization of PDE's. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332(2):661–668, 2001.

[75] C. Lirong and L. Degui. A class of parallel rosenbrock formulas. In *Proceeding of 3rd BICSC*, volume 95, 1995.

[76] T. Løvås, E. Mastorakos, and D.A. Goussis. Reduction of the RACM scheme using Computational Singular Perturbation Analysis. *Journal of Geophysical Research*, 111(D13):D13302, 2006.

[77] R. Lowe and A. Tomlin. Low-dimensional manifolds and reduced chemical models for tropospheric chemistry simulations. *Atmospheric Environment*, 34(15):2425–2436, 2000.

[78] F.W. Lurmann, A.C. Lloyd, and R. Atkinson. A chemical mechanism for use in long-range transport/acid deposition computer modeling. *Journal of Geophysical Research: Atmospheres*, 91(D10):10905–10936, 1986.

[79] U. Maas. Efficient Calculation of Intrinsic Low-Dimensional Manifolds for the Simplification of Chemical Kinetics. *Computing and Visualization in Science*, 1:69–81, 1997.

[80] U. Maas and S.B. Pope. Implementation of simplified chemical kinetics based on intrinsic low dimensional manifolds. *Symposium (International) on Combustion*, 24:103–112, 1992.

[81] U. Maas and S.B. Pope. Simplifying chemical kinetics: Intrinsic low-dimensional manifolds in composition space. *Combustion and Flame*, 88:239–264, 1992.

[82] U. Maas and S.B. Pope. Laminar flame calculations using simplified chemical kinetics based on intrinsic low-dimensional manifolds. *Symposium (International) on Combustion*, 25:1349–1356, 1994.

[83] Y. Maday. Parareal in time algorithm for kinetic systems based on model reduction. *High-Dimensional Partial Differential Equations in Science and Engineering*, 41:183–194, 2007.

[84] Y. Maday, E. Ronquist, and G. Staff. The parareal-in-time algorithm: Basics, stability analysis and more. *(Preprint)*, pages 1–20, 2007.

[85] P.A. Makar and S.M. Polavarapu. Analytic solutions for gas-phase chemical mechanism compression. *Atmospheric Environment*, 31(7):1025–1039, 1997.

[86] A.R. Marsden, M. Frenklach, and D.D. Reible. Increasing the Computational Feasibility of Urban Air Quality Models that Employ Complex Chemical Mechanisms. *Journal of the Air Pollution Control Association*, 37(4):370–376, 1987.

[87] G.J. McRae, W.R. Goodin, and J.H. Seinfeld. Numerical solution of the atmospheric diffusion equation for chemically reacting flows. *Journal of Computational Physics*, 45(1):1–42, 1982.

[88] J. Michalakes and M. Vachharajani. GPU Acceleration of Numerical Weather Prediction. In *International Parallel and Distributed Processing Symposium*, pages 1–18, 2008.

[89] P. Middleton, W.R. Stockwell, and W.P.L. Carter. Aggregation and Analysis of Volatile Organic Compound Emissions for Regional Modeling. *Atmospheric Environment*, 24(5):1107–1133, 1990.

[90] M. Minion. A hybrid parareal spectral deferred corrections method. *Communications in Applied Mathematics and Computational Science*, 5(2):265–301, 2010.

[91] M. Minion and S.A. Williams. Parareal and spectral deferred corrections. In *AIP Conference Proceedings*, volume 1048, pages 388–391, 2008.

[92] W. Miranker and W. Liniger. Parallel methods for the numerical integration of ordinary differential equations. *Mathematics of Computation*, 21:303–320, 1967.

[93] G. E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114–117, 1965.

[94] M.A. Mora-Ramirez and R.M. Velasco. Reduction of CB05 mechanism according to the CSP method. *Atmospheric Environment*, 45(1):235–243, 2011.

[95] M.K. Neophytou, D.A. Goussis, M. Van Loon, and E. Mastorakos. Reduced chemical mechanisms for atmospheric pollution using Computational Singular Perturbation analysis. *Atmospheric Environment*, 38:3661–3673, 2004.

[96] A.S. Nielsen. Feasibility study of the parareal algorithm. Master's thesis, Technical University of Denmark, Asmussens Alle, DK-2800 Kgs. Lyngby, Denmark, 2012.

[97] J. Nievergelt. Parallel methods for integrating ordinary differential equations. 7:731–733, 1964.

[98] L.R. Petzold. A description of DASSL: a differential/algebraic system solver. In *Scientific computing*, pages 65–68. IMACS, New Brunswick, New Jersey, 1983.

[99] R. Ponalagusamy and K. Ponnammal. A Parallel Fourth Order Rosenbrock Method: Construction, Analysis and Numerical Comparison. *International Journal of Applied and Computational Mathematics*, 1:45–68, 2014.

[100] D. Poppe, B. Aumont, B. Ervens, H. Geiger, H. Herrmann, E.P. Röth, W. Seidl, W.R. Stockwell, B. Vogel, S. Wagner, and D. Weise. Scenarios for modeling multiphase tropospheric chemistry. *Journal of Atmospheric Chemistry*, 40:77–86, 2001.

[101] H. Rabitz and Ö. Aliş. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25:197–233, 1999.

[102] H. Rabitz, Ö. Aliş, J. Shorter, and K. Shim. Efficient input-output model representations. *Computer Physics Communications*, 117:11–20, 1999.

[103] L.F. Richardson. *Weather Prediction by Numerical Process.* Cambridge University Press, Cambridge, New York, 2007.

[104] J.S. Rosenbaum. Conservation properties of numerical integration methods for systems of ordinary differential equations. *Journal of Computational Physics*, 267:259–267, 1976.

[105] H.H. Rosenbrock. Some general implicit processes for the numerical solution of differential equations. *The Computer Journal*, 5(4):329–330, 1963.

[106] M.R. Roussel and S.J. Fraser. On the geometry of transient relaxation. *Journal of Chemical Physics*, 94(1991):7106–7113, 1991.

[107] D. Ruprecht and R. Krause. Explicit parallel-in-time integration of a linear acoustic-advection system. *Computers and Fluids*, 59:72–83, 2012.

[108] A. Sandu. Time-stepping methods that favor positivity for atmospheric chemistry modeling. In *IMA Volume on Atmospheric Modeling*, pages 1–21. Springer, 2001.

[109] A. Sandu, J.G. Verwer, J.G. Blom, E.J. Spee, G.R. Carmichael, and F.A. Potra. Benchmarking stiff ODE solvers for atmospheric chemistry problems. II: Rosenbrock methods. *Atmospheric Environment*, 31:3459–3472, 1997.

[110] A. Sandu, J.G. Verwer, M. van Loon, G.R. Carmichael, F.A. Potra, D. Dabdub, and J.H. Seinfeld. Benchmarking stiff ODE solvers for atmospheric chemistry problems. I: implicit versus explicit. *Atmospheric Environment*, 31:3151–3166, 1997.

[111] J.H. Seinfeld and S.N. Pandis. *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change.* John Wiley & Sons, second edition, 2012.

[112] L.F. Shampine. Error estimation and control for ODEs. *Journal of Scientific Computing*, 25:3–16, 2005.

[113] J.A. Shorter, P.C. Ip, and H. Rabitz. An Efficient Chemical Kinetics Solver Using High Dimensional Model Representation. *The Journal of Physical Chemistry*, 103(36):7192–7198, 1999.

[114] M.N. Spijker. Stiffness in numerical initial-value problems. *Journal of Computational and Applied Mathematics*, 72(2):393–406, 1996.

[115] C.M. Spivakovsky, Wofsy S.C., and Prather M.J. A numerical method for parameterization of atmospheric chemistry: Computation of tropospheric OH. *Journal of Geophysical Research*, 95(18), 1990.

[116] B. Sportisse and R. Djouad. Use of proper orthogonal decomposition for the reduction of atmospheric chemical kinetics. *Journal of Geophysical Research*, 112(D6):D06303, 2007.

[117] G.A. Staff. The Parareal Algorithm. *Science And Technology*, 60(2):173–184, 2003.

[118] W.R. Stockwell, F. Kirchner, M. Kuhn, and S. Seefeld. A new mechanism for regional atmospheric chemistry modeling. *Journal of Geophysical Research*, 102:25847, 1997.

[119] W.R. Stockwell, P. Middleton, J.S. Chang, and X. Tang. The second generation regional acid deposition model chemical mechanism for regional air quality modeling. *Journal of Geophysical Research*, 95(D10):16343, 1990.

[120] A.N. Tikhonov, A.B. Vasileva, and Sveshnikov A.G. *Differential Equations*. Springer, 1985.

[121] A.S. Tomlin, L. Whitehouse, R. Lowe, and M.J. Pilling. Low-dimensional manifolds in tropospheric chemical systems. *Faraday discussions*, (120):125–146; discussion 197–213, 2001.

[122] T. Turányi. Parameterization of reaction mechanisms using orthonormal polynomials. *Computers & chemistry*, 18(I):45–54, 1994.

[123] T. Turányi and A.S. Tomlin. *Analysis of Kinetic Reaction Mechanisms*. Springer, 2014.

[124] T. Turányi, A.S. Tomlin, and M.J. Pilling. On the error of the quasi-steady-state approximation. *The Journal of Physical Chemistry*, 97(1):163–172, 1993.

[125] P.J. Van Der Houwen and B.P. Sommeijer. Parallel iteration of high-order Runge-Kutta methods with stepsize control. *Journal of Computational and Applied Mathematics*, 29:111–127, 1990.

[126] S.G. Vandewalle and E.F. van de Velde. Space-time Concurrent Multigrid Waveform Relaxation. *Annals of Numerical Mathematics*, 1(1):347–360, 1994.

[127] J.G. Verwer, J.G. Blom, M. Van Loon, and E.J. Spee. A comparison of stiff ode solvers for atmospheric chemistry problems. *Atmospheric Environment*, 30:49–58, 1996.

[128] J.G. Verwer, W.H. Hundsdorfer, and J.G. Blom. Numerical time integration for air pollution models. *Surveys on Mathematics for Industry*, 10(2):107–174, 2002.

[129] D.A. Voss. Fourth-order parallel Rosenbrock formulae for stiff systems. *Mathematical and Computer Modelling*, 40:1193–1198, 2004.

[130] D.A. Voss and A.Q.M. Khaliq. Parallel Rosenbrock methods for chemical systems. *Computers and Chemistry*, 25:101–107, 2001.

[131] S.W. Wang, S. Balakrishnan, and P. Georgopoulos. Fast equivalent operational model of tropospheric alkane photochemistry. *AIChE Journal*, 51(4):1297–1303, 2005.

[132] S.W. Wang, P.G. Georgopoulos, G. Li, and H. Rabitz. Condensing Complex Atmospheric Chemistry Mechanisms. 1. The Direct Constrained Approximate Lumping Method Applied to Alkane Photochemistry. *Environmental Science & Technology*, 32(13):2018–2024, 1998.

[133] S.W. Wang, P.G. Georgopoulos, G. Li, and H. Rabitz. Computationally efficient atmospheric chemical kinetic modeling by means of high dimensional model representation. In *Large-Scale Scientific Computing*, volume 2179 of *Lecture Notes in Computer Science*, pages 326–333. Springer, 2001.

[134] S.W. Wang, H. Levy, G. Li, and H. Rabitz. Fully equivalent operational models for atmospheric chemical kinetics within global chemistry-transport models. *Journal of Geophysical Research*, 104(D23):30417, 1999.

[135] Z. Wang and S. Wu. Parareal Algorithms Implemented with IMEX Runge-Kutta Methods. 2014.

[136] J. Warnatz, U. Maas, and R.W. Dibble. *Combustion: Physical and chemical fundamentals, modeling and simulation, experiments, pollutant formation.* Springer, 2006.

[137] L.E. Whitehouse, A.S. Tomlin, and M.J. Pilling. Systematic reduction of complex tropospheric chemical mechanisms using sensitivity and time-scale analyses. *Atmospheric Chemistry and Physics*, 4(4):3721–3783, 2004.

[138] G.Z. Whitten, J.P. Killus, R.G. Johnson, and Atmospheric Sciences Research Laboratory. *Modeling of auto exhaust smog chamber data for EKMA development.* U.S. Environmental Protection Agency, Atmospheric Sciences Research Laboratory Research Triangle Park, NC, 1985.

[139] S. Wu, B. Shi, and C. Huang. Parareal-richardson algorithm for solving nonlinear ODEs and PDEs. *Communications in Computational Physics*, 6(4):883–902, 2009.

[140] T.R. Young and J.P. Boris. Numerical Technique for solving stiff ordinary differential equations associated with the chemical kinetics. *The Journal of Physical Chemistry*, 81:2424–2427, 1977.

[141] A.M. Zhabotinsky. Periodical process of oxidation of malonic acid solution. *Biophysics*, 9:306 – 311, 1964.

[142] H. Zhang, J.C. Linford, A. Sandu, and R. Sander. Chemical mechanism solvers in air quality models. *Atmosphere*, 2(3):510–532, 2011.