NUMERICAL ASPECTS OF AIR QUALITY MODELING

by

Adrian Sandu

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Applied Mathematical and Computational Sciences
in the Graduate College of
The University of Iowa

August 1997

Thesis supervisors: Professor Florian A. Potra
Professor Gregory R. Carmichael

ABSTRACT

Numerical simulation of atmospheric transport-chemistry processes is a computationally intensive problem. A major task is the integration of the stiff systems of ordinary differential equations describing the chemical transformations. It is therefore of interest to develop stiff solvers which can be identified as close to optimal for atmospheric applications.

In this thesis we analyze and propose a number of methods, both implicit and explicit, suitable for this task. A theoretical analysis of the proposed methods is given. Optimal methods are identified via a large number of numerical experiments. A set of test problems representative for the application is put together and coded for this purpose.

The symbolic preprocessor KPP (kinetic preprocessor) is developed and presented; KPP is able to translate chemical equations into ready to run simulation code.

Several additional issues related to atmospheric numerical modeling are discussed: boundary conditions and the possibility of eliminating the splitting errors.

This work conclusively shows that implicit methods can be useful for air quality modeling. For example, one of the newly proposed methods (Rodas3) is, in the STEM-II model, as fast as Qssa with 30 seconds time step. However, Rodas3 is much more accurate and can be applied to any chemical mechanism, as opposed to Qssa (and any explicit method) which are unstable on multiphase systems.

Abstract approved: _____
Thesis supervisor

_____
Title and department

_____
Date

Abstract approved: _____
Thesis supervisor

_____
Title and department

_____
Date

NUMERICAL ASPECTS OF AIR QUALITY MODELING

by

Adrian Sandu

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy degree in
Applied Mathematical and Computational Sciences
in the Graduate College of
The University of Iowa

August 1997

Thesis supervisors: Professor Florian A. Potra

Professor Gregory R. Carmichael

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Adrian Sandu

has been approved by the Examining Committee
for the thesis requirement for the
Doctor of Philosophy degree in
Applied Mathematical and Computational Sciences
at the August 1997 graduation.

Thesis committee: _____
Thesis supervisor

_____
Thesis supervisor

_____
Member

_____
Member

_____
Member

To my family

ACKNOWLEDGMENTS

This work was supported in part by grants from DOE (DE–FG02–94ER61855), NASA (NAGW–2428) and the Center for Global and Regional Environmental Research.

I want to thank to several people who have made this possible. To my advisors, professor Florian Potra and professor Gregory Carmichael. To my colleagues and co-workers Valeriu Damian, Mirela Damian, Laurent Jay, Jan Verwer, Maarten van Loon, Edwin Spee, Joke Blom, Richard Arndt, Mahesh Phadnis, Chung Song, Li Ling Chen, Jane Frank.

Special thoughts go to my wife Corina for her understanding, encouragement, support and affection, and to my daughter Andreea, for giving a purpose to all this.

ABSTRACT

Numerical simulation of atmospheric transport-chemistry processes is a computationally intensive problem. A major task is the integration of the stiff systems of ordinary differential equations describing the chemical transformations. It is therefore of interest to develop stiff solvers which can be identified as close to optimal for atmospheric applications.

In this thesis we analyze and propose a number of methods, both implicit and explicit, suitable for this task. A theoretical analysis of the proposed methods is given. Optimal methods are identified via a large number of numerical experiments. A set of test problems representative for the application is put together and coded for this purpose.

The symbolic preprocessor KPP (kinetic preprocessor) is developed and presented; KPP is able to translate chemical equations into ready to run simulation code.

Several additional issues related to atmospheric numerical modeling are discussed: boundary conditions and the possibility of eliminating the splitting errors.

This work conclusively shows that implicit methods can be useful for air quality modeling. For example, one of the newly proposed methods (Rodas3) is, in the STEM-II model, as fast as Qssa with 30 seconds time step. However, Rodas3 is much more accurate and can be applied to any chemical mechanism, as opposed to Qssa (and any explicit method) which are unstable on multiphase systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Overview of air quality modeling

A detailed understanding of the relationships between the emissions and the resulting distribution of primary and secondary species in the atmosphere is a requisite to designing actions for the maintenance of a healthy environment. Scientific efforts to understand the atmospheric processes governing these relationships involve a combination of laboratory experiments, field studies, and modelling analysis. Laboratory experiments provide basic data on individual physical and chemical processes. Field studies are designed to investigate a limited number of processes under conditions in which a few processes dominate. Unlike controlled laboratory experiments, field studies cannot be parametrically controlled. Since laboratory experiments and field studies by themselves cannot fully elucidate complex atmospheric phenomena, comprehensive models that allow multiple processes to occur simultaneously are required for data analysis and scientific inquiry.

The models by Carmichael et al. [14], Jacob et al. [53], Dentener and Crutzen [29] are examples of regional and global scale atmospheric chemistry models in use today. These models treat transport, chemical transformations, emissions and deposition processes in an integrated framework, and serve as representations of our current understanding of the complex atmospheric processes. They provide a means to perform parametric studies for quantitative analysis of the relationships between emissions and the resulting distribution, and can also be used to study

the response of the pollutant distributions to system perturbations, and to link pollutant distributions to environmental effects.

As our scientific understanding of atmospheric chemistry and dynamics has expanded in recent years, so has our ability to construct comprehensive models which describe the relevant processes. However, these comprehensive atmospheric chemistry models are computationally intensive because the governing equations are nonlinear, highly coupled, and extremely stiff. As with other computationally intensive problems, the ability to fully utilize these models remain severely limited by today's computer technology.

Grid resolutions of $0.5^o * 0.5^o$ or better in the horizontal, 20 vertical grids, and as many as 40 to 100 species are necessary for adequate analysis of perturbations to atmospheric chemistry on the global and regional scales. With such detail, the finest grid resolution that can be achieved for global-scale analysis on todays computers is about $2^o * 2^o * 10$ vertical levels, even if one is content with a $1:1$ simulation time to CPU time ratio. This latter ratio should be at least $10:1$, and preferably $100:1$ or better, to effectively address many of the important scientific questions.

The scientific issues associated with analysis of our chemically perturbed atmospheres are dominated by a number of underlying considerations. Several of the more important ones are: (a) the anthropogenic sources of trace species are quite localized and occur only over a fraction of the Earth's land area; (b) natural sources of trace species are, for the most part, very disperse and are not in the same areas as the anthropogenic sources (although this trend may be changing in regions such as tropical rain forests and the savannahs); (c) in virtually no case can an individual species be studied in isolation from other species; (d) many of the mechanisms that effect transformation of the species are non-linear (e.g.,

chemical reactions and nucleation processes); and (e) species of importance have atmospheric lifetimes that range from milliseconds and shorter to years (e.g., OH radical to CH_4). These considerations require: finer grid resolutions than currently existing ones in present-day models; simultaneous treatment of many species; and long simulation times (i.e. months to years) to assess the impacts. These demands present considerable challenges to the air quality modeling community.

1.2 Objectives of this thesis

Air quality models are conceptually complex and computationally very expensive. A full understanding of the underlying physical and chemical processes is necessary before building good numerical approximations. Models are in need of improved algorithms that are able to understand the sources and control the magnitudes of the numerical errors, to perform robustly in a wide variety of physical and chemical conditions and, last but not least, be fast enough to provide good results in an acceptable amount of CPU time.

To see why computational speed is so important, let us mention that, on a HP-935A workstation, one day of chemistry simulation with DAE Qssa (at a single grid-point, with our comprehensive model) takes roughly one second. A three-dimensional model may have $50 \times 50 \times 20$ grid-points (a realistic value for a regional model) and the chemistry must be evaluated at each grid-point. A simple calculation shows that, with a serial code, one day of simulation will need at least 15 CPU-hours (without counting the transport part and the overhead associated with reading and writing Megabytes of data). The net result is that the simulation time is of the same order as the wall-clock time. One possible solution is to move the

codes on more powerful machines (e.g., a version of STEM-II, see [14], is currently running on a CRAY-C90). Another direction would be to take advantage of the inner parallelism of the problem, and to write parallel versions of the simulation codes. The increase in machine power, however, does not anihilate the need for faster (and more reliable) numerical methods.

The thesis looks at some of the key numerical aspects of these models:

1. Since up to 90% of the CPU time is consumed in the integration of chemical equations, this study pays special attention to developing different methods for this task.

   - In chapter 2 a summary of the existing explicit methods is given; an analysis of Qssa and also some new methods in this class are presented.
   - In chapter 3 implicit methods are investigated; a systematic strategy for reducing the linear algebra costs, plus new Rosenbrock type methods are discussed.

2. Another important aspect is how to evaluate and compare different algorithms. This requires a standard computational environment in which different routines are uniformly tested. In this thesis a series of benchmark problems are established and and used. All the numerical methods - both new and from the literature - are tested on this set of problems; chapter 4 presents the test problems together with the numerical results.

3. To facilitate the benchmark problems and to make it easy to modify complex and comprehensive codes a symbolic preprocessor has been developed. This preprocessor "reads in" chemical kinetic equations and is able to produce C

or FORTRAN code associated with the corresponding differential equations. Chapter 5 is devoted to the presentation of KPP - the kinetic preprocessor.

4. Improving the integration of chemical rate equations is just one aspect of the big picture. The integration of the transport equations needs improvements also. Most models use operator splitting, and numerical algorithms for each of the subproblems are tested in very simple settings. Very little attention has been paid to the proper treatment of boundary conditions and to the control/elimination of splitting errors. Chapter 6 presents some ideas related to the solution of these problems.

5. Finally, chapter 7 draws conclusions and pinpoints further research directions.

1.3 Problem statement

Air quality models are computer-based models which calculate the distribution of trace gases in the troposphere from specified emissions distributions and meteorological scenarios. The major features consist of:

1. a transport component (or module) to describe the wind speed and direction, the eddy diffusivity and mixing layer height, the temperature, the water vapor, cloud water content, and the radiation intensity of each location as a function of time;

2. a chemical kinetic mechanism to describe the rates of atmospheric reactions, including homogeneous gas-phase, heterogeneous, and liquid phase reactions;

3. removal modules to describe the dry deposition of material, and the in-cloud and below-cloud removal processes.

Each process incorporated into a model is itself a very complex and incompletely understood phenomenon. Therefore, in formulating such models it is necessary to incorporate the processes into the model framework by utilizing chemical, dynamic, and thermodynamic parameterizations. Furthermore, even processes that are quite well understood may require parameterization to maintain some balance of the details among the different processes that are treated in the model.

The theoretical basis is the atmospheric advection-diffusion equations (i.e., the mass balance equations):

$$\begin{array}{cccccccc} \frac{\partial C_i}{\partial t} + \frac{\partial (U_j C_i)}{\partial x_j} & = & \frac{\partial}{\partial x}[K_{jj}\frac{\partial C_i}{\partial x_j}] + R_i + E_i + G_i & & & & i = 1, \cdots, \#\text{ of species} \\ (A) \quad (B) & & (C) & & (D) & (E) & (F) & \end{array} \tag{1.1}$$

where C_i denotes the gas phase concentrations, U_j are velocity components, x_j represents the spatially coordinates, most generally three dimensional, K_{jj} are the eddy diffusivities, R_i, G_i and E_i are the rates of chemical reactions, mass transfer and emissions, respectively.

In the above equations term (A) represents the unsteady accumulation of mass, (B) changes in mass due to advective fluxes, (C) changes in mass due to turbulent diffusive fluxes, (D) the rate of production/destruction due to chemical reaction, (E) the source term due to emissions, and (F) the rate of mass transfer between phases. These equations are nonlinear due to the nonlinear nature of the chemical processes, and are also highly coupled within a given phase, again due to the chemical processes, and coupled between phases through the inter-phase mass transfer processes (e.g., gas absorption, nucleation, and accretion processes).

Comprehensive air quality models presently explicitly treat a wide variety of sulfur compounds (e.g., SO_2, sulfate, some reduced sulfur species), nitrogen compounds (e.g., NO,NO_2, HNO_3, NO_3, N_2O_5, PAN), and hydrocarbon species (e.g., CO, CO_2, CH_4, a variety of alkenes, alkanes, aromatics, aldehydes and organic acids), along with the important oxidizing species such a O_3, OH, and H_2O_2.

Typically, a regional model consists of $50 \times 50 \times 10 \times 80$ sp. $= 2 \cdot 10^6$ variables.

In order to manage the computational complexity, operator splitting is applied. One example, that uses both a splitting between chemistry and transport, and also a directional splitting for solving the transport is:

$$T_x^k \cdot T_y^{\Delta t} \cdot T_z^{\Delta t} \cdot C^{2\Delta t} \cdot T_z^{\Delta t} \cdot T_y^{\Delta t} \cdot T_x^{\Delta t}$$

This means that one solves the subsystem associated with the transport along x direction, together with the initial conditions for the problem and suitable boundary conditions; after the solution was "advanced along x" with Δt, it is "advanced along y", then along z with Δt. Chemistry is then integrated (as an ordinary differential equation) for a time interval $2\Delta t$; finally, the subsystems associated with z, y, x transport are solved in this order. We will call Δt *splitting interval.*

The numerical values obtained from integrating a subsystem are used as initial values for the next subsystem. This amounts to solving

$$\left\{ \begin{array}{lllll} \frac{\partial C_i^*}{\partial t} & = & \frac{\partial}{\partial x}[K \frac{\partial C_i^*}{\partial x}], & C_i^*(t_0) = C_i(t_0) & \text{i=1,NS} \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ \frac{\partial \hat{C}_i}{\partial t} & = & \frac{\partial [U_x \hat{C}_i]}{\partial x}, & \hat{C}_i(t_0) = C_i^*(t_F) & \text{i=1,NS} \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\ \frac{\partial \bar{C}}{\partial t} & = & \{R + E + G\}(t, \bar{C}), & \bar{C}_i(t_0) = \hat{C}_i(t_F) & \\ \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \end{array} \right.$$

The operator splitting approach comes with some inherent questions:

- Does the splitting solution converge to the original solution as splitting interval decreases ?
- How can we set consistent intermediate boundary conditions ?
- Is it possible estimate and control the splitting error ?

For an answer to the convergence question we reffer to Temam and Wheeler. A discussion on the boundary value problem can be found in chapter 7; error estimation in this context is a subject of future research.

The large computational requirements in the study of chemically perturbed environments arise from the complexity of the chemistry of the atmosphere. Integration of the chemistry rate equations typically consumes as much as 90 percent of the total CPU time! Obviously, more efficient integration schemes for the chemistry solvers would result in immediate benefits through the reduction of CPU time necessary for each simulation. As more and more chemical species and reactions are added to the chemical scheme for valid scientific reasons the need for faster yet more accurate chemical integrators becomes even more critical. It is well known that the chemistry rate equations comprise a system of stiff ordinary differential equations (ODE). At each operator split step, this system is restarted. A typical behaviour for a stiff system is to go through a rapid, transient evolution in the beginning; during this transient fast components "die out"and the system approaches a slow manifold; afterwards the system evoluates along the slow manifold. The existence of the transients at the beginning of each operator split time step is not a feature of the physical system, but is related to the splitting method. During the transients ODE solvers are forced to choose very small step sizes (from accuracy restrictions). This is where a large part of the computing time is spent.

1.4 Mass action kinetics

The chemical subsystem can be described as

$$C' = f(t, C) \tag{1.2}$$

The rate of change f is given by the empirical law of mass action kinetics.

Let $\mathcal{P}_i$ be the set of chemical reactions in which species C_i appears as a product, and $\mathcal{D}_i$ the set of chemical reactions in which C_i is a reactant. Oviously, $\mathcal{P}_i$ produce and $\mathcal{D}_i$ consume C_i.

The rate of change of species C_i is given by

$$C_i' = f_i(t, C) = \sum_{j \in \mathcal{P}_i} s_j - \sum_{k \in \mathcal{D}_i} s_k$$

where s_j is the "speed of chemical transformation" associated with chemical reaction j (the quantity of reactants which is transformed into products each time unit).

The law of mass action kinetics states that the reactions speeds s_j are proportional to the concentrations of the reactants, with proportionality constants K_j ("reaction rates").

For example, the reaction

$$A + B \xrightarrow{K_j} C$$

has a "speed" of $s_j = K_j \cdot [A] \cdot [B]$. This reaction has a positive contribution to the rate of change of C (production) and negative contributions to the derivatives of A and B (consumption):

$$\begin{aligned} \frac{d[A]}{dt} &= \cdots - s_j \cdots \\ \frac{d[B]}{dt} &= \cdots - s_j \cdots \\ \frac{d[C]}{dt} &= \cdots + s_j \cdots \end{aligned}$$

In the general case arbitrary stoichiometric coefficients $\alpha \geq 0,\ \beta \geq 0$ are

present, and a typical reaction involving $C_1 \dots C_N$ is

$$\sum_k \alpha_j^k C_k \xrightarrow{K_j} \sum_k \beta_j^k C_k$$

The speed of this reaction is given by

$$s_j = K_j \prod_k [C_k]^{\alpha_j^k} \geq 0$$

The α coefficients are taken to be natural numbers, with the interpretation that an integer number of molecules (or moles) of the reactants enter the reaction; β coefficients are also natural numbers for elementary reactions (an integer number of molecules is formed through reaction); however, we will allow β's to take fractional values also; this will give the possibility of modeling complex, chained reactions with noninteresting intermediate compounds.

Thus, for every species C_i we have that

$$\frac{dC_i}{dt} = \sum_{j \in \mathcal{P}_i} \alpha_j^i s_j - \sum_{j \in \mathcal{D}_i} \beta_j^i s_j$$

Define the "production term" of species C_i as

$$P_i = \sum_{j \in \mathcal{P}_i} \alpha_j^i s_j$$

and the "destruction term"

$$\begin{aligned} D_i &= \left(\sum_{j \in \mathcal{D}_i} \beta_j^i s_j \right) / C_i \\ &= \left(\sum_{j \in \mathcal{D}_i} \beta_j^i K_j \prod_k [C_k]^{\alpha_j^k} \right) / C_i \\ &= \sum_{j \in \mathcal{D}_i} \beta_j^i K_j \prod_{k \neq i} [C_k]^{\alpha_j^k} [C_i]^{\alpha_j^i - 1} \end{aligned}$$

Note that if species C_i is a reactant in reaction j ($j \in \mathcal{D}_i$) then $\alpha_j^i \geq 1$ and all the powers in the last equation are positive. Also note that for $C_i \geq 0, \ \forall i$ we have $P_i, D_i \geq 0, \ \forall i$.

With these definitions, the system (1.2) can be rewritten in production destruction form as:

$$C_i' = f_i(t, C) = P_i(t, C) - D_i(t, C) \cdot C_i \tag{1.3}$$

1.4.1 Positivity

Physically, the concentrations $[C_i]$ are nonnegative; the mathematical model build upon mass action kinetics preserves this property; for if $C_i = 0$ and $C_j \geq 0,\ j \neq i$ then, by (1.3) we have that

$$f_i(t, C) = P_i(t, C) \geq 0 \ .$$

1.4.2 Conservation of mass

Physically the mass of a closed chemical system is conserved. Also, the number of atoms of a certain type that enter a chemical transformation is preserved, and can be found in the products (although in a possible different molecular arrangement); this implies that the mass of certain families of species is preserved (e.g. sulphur or nitrate compounds).

The mathematical model build upon mass action kinetics is mass conservative if

$$\sum_i \nu_i f_i(t, C) = 0$$

for all nonnegative concentration vectors. Here ν_i are some scaling factors equal to one if we work with masses, and depending upon molar masses if we work with concentrations.

In order to have a qualitatively correct solution, good numerical methods should preserve the positivity and the mass of the system. While preserving positivity is a demanding task, the preservation of the linear invariants of the system is

easier and we will pursue it further.

1.5 Review of the literature

Efficient chemistry integration algorithms for atmospheric chemistry have been obtained by carefully exploiting the particular properties of the model. One of the commonly used methods is the Qssa method of Hesstvedt et al. [50]. The performance of the Qssa scheme can be further improved by using the lumping technique which leads to mass conservation of groups of species. Practical Qssa performance is discussed in the instructive paper [79], by Shieh, Carmichael et al. where different integrators are compared on specific atmospheric chemistry problems. An evaluation of the local truncation error of the Qssa scheme can be found in [92].

There are many specially-tailored methods in use in atmospheric chemistry models. One of the first proposed methods, and which has been extensively used, is the hybrid predictor-corrector algorithm of Young and Boris [96]. Species are divided into stiff and nonstiff; the explicit Euler method (predictor) and an explicit trapezoidal method (corrector) are used for the nonstiff part, while the stiff part is integrated with a modified midpoint scheme.

S. Sillman in [80] developed an integration scheme based on the implicit Euler formula. Following a careful analysis of sources and sinks of odd-hydrogen radicals in the troposphere, the author reorders the vector of species such that the resulting Jacobian is nearly lower block triangular; this enables an elegant "decoupling" between short-lived species (integrated implicitly) and long-lived species (integrated semi-implicitly). The scheme is efficient, but difficult to generalize.

Hertel, Berkowicz, Christensen, and Hov [48] proposed an algorithm based on the implicit Euler method. Using only linear operators it preserves the total mass. The nonlinear system is solved using functional iterations. The main idea is to speed up these iterations using explicit solutions for several groups of species. The method seems to work fine for very large step-sizes.

A particularly clear approach was taken by Gong and Cho [42]. They divide the species into slow and fast, according to their lifetimes; the slow species are estimated using an explicit Euler scheme; the implicit ones are integrated with the implicit Euler scheme (and Newton-Raphson iterations for solving the nonlinear system); as a last step, the slow species are "corrected", reiterating the explicit Euler step.

A fancy projection/forward differencing method was proposed by Elliot, Turco, and Jacobson [38]. The species are grouped together in families; the distribution of the constituents inside a family is recalculated before each integration step using an implicit relation and solving the corresponding nonlinear system (this "projection" can be viewed as a "predictor"); then the integration is carried out for families using a significantly improved time step.

Dabdub and Seinfeld investigated in [23] an extrapolation algorithm whose underlying numerical scheme is based on a Qssa predictor and on a hybrid corrector (with a trapezoidal method for nonstiff components and a modified Qssa formula for the stiff components). The authors report good results, however, a theoretical analysis of the method is not presented.

Verwer [91] proposed an extension of Qssa to a second order consistent scheme, and also a "two-step method" which is the second order BDF formula plus Gauss-Seidel iterations for solving the nonlinear system (according to the author, these

iterations perform similarly to the modified Newton method, but with less overhead). The two-step method enables very large step-sizes.

A different approach was taken in [54] by Jacobson. The 3-D calculations are vectorized around the grid-cell dimension (very interesting idea) and advantage is taken of the sparse structure of Jacobians and a specific reordering of species (that makes Jacobians close to lower triangular form).

1.6 Thesis organization

The rest of the thesis is organized as follows.

In chapter 2 a summary of the existing explicit methods is given; an analysis of Qssa and also some new methods in this class are presented.

In chapter 3 implicit methods are investigated; a systematic strategy for reducing the linear algebra costs, plus new Rosenbrock type methods are discussed.

Chapter 4 presents numerical results.

Chapter 5 is devoted to KPP - the kinetic preprocessor.

Chapter 6 treats some related problems, and

Chapter 7 draws conclusions and pinpoints further research directions.

CHAPTER 2
EXPLICIT METHODS

2.1 Introduction

In this paper we look in detail at the widely used Qssa method and demonstrate that significant improvements in the efficiency of this type of method can be achieved. We consider two extrapolation algorithms based on Qssa. In particular we obtain an order two method that uses two function evaluations per step which we call the **Extrapolated Qssa** method. We also construct a nontrivial modification of the well known GBS extrapolation algorithm based on an appropriate Qssa modification. In particular we obtain an order two method that uses three function evaluations per step which we call (for good reason) the **Symmetric Qssa** method. In the stiff case the extrapolated methods no longer have a higher order than the Plain Qssa method. Nevertheless, we prove that the Extrapolated Qssa method and the Symmetric Qssa method have a smaller error constant which explains their superior performance. We also prove that under certain conditions the Plain Qssa method is convergent when applied to a particular singular perturbation problem. Numerical experiments on a test problem used in a regional-scale model are also presented.

2.2 Idea of explicitness

The dedicated explicit algorithms are scalarly implicit and exploit the production-loss form of the ODE system. They are based on the assumption that all short lived

species, causing the problem to be stiff, are only weakly coupled to all other species. In mathematical terms this implies that for these short lived species the loss term $-L_i(t,y)$ is close to a stiff eigenvalue of the Jacobian matrix, and that no stiff eigenvalues exist which are not close to a loss term. Following [61], the following reasoning explains this. Consider atomic oxygen O which is a very fast reacting species. In many models a typical predominant removal step for O (by some orders of magnitude) is reaction with O_2 and a third body M, i.e.,

$$O + O_2 + M \xrightarrow{k} O_3 + M.$$

The kinetics for O is then well described by the scalar ODE

$$\frac{d[O]}{dt} = P_{[O]} - L_{[O]}[O], \quad L_{[O]} = k[O_2][M],$$

where $L_{[O]}$ is a constant since the concentrations of O_2 and M are fixed. Hence there exists no coupling with other species through the loss rate. Because for atomic oxygen coupling through the production rate $P_{[0]}$ is very weak too, the predominance of the above reaction now trivially implies that $-L_{[O]}$ is close to a stiff eigenvalue of the Jacobian matrix of the whole system and, in addition, that in first approximation the exact solution for O is given by

$$[O](t+h) = \frac{P_{[O]}}{L_{[O]}} + ([O](t) - \frac{P_{[O]}}{L_{[O]}})\, e^{-L_{[O]}h}.$$

This exact solution for truly constant coefficients forms the starting point for the popular Qssa methods.

Although this explanation is not mathematically rigorous for the nonlinear problems we deal with, it predicts to a great extent whether an explicit solver of the type used in this paper can be justified in advance (for one of these, namely Twostep using Gauss-Seidel iteration, the coupling between short and long lived

species may be stronger, since Gauss-Seidel iteration introduces a form of "triangular implicitness"). For each problem we therefore illustrate the eigenvalue relationship in a table showing the species and eigenvalues for which the relationship is found to exist and the distribution of the remaining part of the spectrum (all the eigenvalues were computed with the routine `dgees` from Lapack) of the Jacobian (see also Fig. 8 in [61]). In this remaining part, eigenvalues thus can be of two sorts, either they are small and hence do not introduce stiffness, or they are large but cannot be associated with a single short lived species. If these latter eigenvalues exist, then the special purpose explicit methods can fail completely. Inspection of all the tables presented here will reveal that these latter eigenvalues exist only for the tropospheric wet problem `G`. This observation is in line with our test results.

2.3 Plain, DAE, and iterated Qssa

If $y \in \mathbf{R}^n$ denotes the vector of concentrations, the differential equations arising from the chemical mass balance relation can be written in the form

$$\frac{dy_j}{dt} = P_j(y_1, \ldots, y_n) - D_j(y_1, \ldots, y_n) y_j \quad \text{for } j = 1, \ldots, n \tag{2.1}$$

where $P_j(y)$ and $D_j(y)y_j$ are *production* and *destruction* terms respectively. These equations have an exponential analytical solution provided that $P_j(y)$ and $D_j(y)$ are constant. For an initial value $y(t_0) = y_0$ and a step-size h the approximation

$$y_j(t_0 + h) \approx \frac{P_j(y_0)}{D_j(y_0)} - \left(\frac{P_j(y_0)}{D_j(y_0)} - y_{0,j} \right) \cdot e^{-hD_j(y_0)} =: \bar{y}_j(t_0 + h) \tag{2.2}$$

forms the basis of the **Qssa** method. For species with a very long *lifetime* $\tau_j = 1/D_j$, i.e., with very small D_j, this equation can be simplified by replacing the exponential term with $1 - hD_j(y_0)$, thus obtaining the explicit Euler formula

$$y_j(t_0 + h) \approx y_{0,j} + h \left(P_j(y_0) - D_j(y_0) y_{0,j} \right) \ . \tag{2.3}$$

For species with a very short lifetime, i.e., with very large positive D_j, the exponential term can be approximated by 0, and the following steady-state relation is obtained

$$y_j(t_0+h) \approx \frac{P_j(y(t_0+h))}{D_j(y(t_0+h))} \ . \tag{2.4}$$

For short-lived species these equalities form a system of nonlinear equations which is usually solved by a fixed point iteration scheme. This is in fact equivalent to solve the system of differential-algebraic equations (DAE) obtained by replacing in (2.1) the differential equations corresponding to short-lived species by their corresponding steady-state equations

$$\begin{aligned} \frac{dy_j}{dt} &= P_j(y_1,\ldots,y_n) - D_j(y_1,\ldots,y_n)y_j, \ j \in \mathcal{J} \ , \\ 0 &= P_i(y_1,\ldots,y_n) - D_i(y_1,\ldots,y_n)y_i, \ i \in \mathcal{I} \ , \end{aligned} \tag{2.5}$$

where $\mathcal{I}$ is the set of indices corresponding to the short-lived species and the set $\mathcal{J}$ consists of the remaining indices. We call the scheme based on (2.2)-(2.3)-(2.4) the **DAE Qssa** method. This is clearly distinct from the method consisting of applying (2.2) to all species which will be called the **Plain Qssa** method. We note that the DAE Qssa method described in this paper is usually known in the literature as the Qssa method and has been extensively used in solving atmospheric chemistry equations.

Consider now the Plain Qssa scheme. By construction we have

$$\bar{y}(t_0) = y_0 = y(t_0) \ , \qquad \bar{y}'(t_0) = P(y_0) - D(y_0)y_0 = y'(t_0) \ .$$

A simple analysis for the second derivatives at t_0 gives

$$\begin{aligned} \bar{y}''(t_0) &= -D(y_0)\bar{y}'(t_0) = -D(y_0)(P(y_0) - D(y_0)y_0) \ , \\ y''(t_0) &= P_y(y_0)y'(t_0) - D_y(y_0)(y'(t_0), y(t_0)) - D(y_0)y'(t_0) \\ &= (P_y(y_0) - D(y_0))\,(P(y_0) - D(y_0)y_0) - D_y(y_0)(P(y_0) - D(y_0)y_0, y_0) \ , \end{aligned}$$

showing that $\bar{y}''(t_0) \neq y''(t_0)$ in general. Thus the order of Plain Qssa is equal to

one.

In an attempt to improve Plain Qssa, the chemists working on atmospheric models have developed the **Iterated Qssa** method. The formula (2.2) is re-applied with P_j and D_j recomputed at the point $y_1 := \bar{y}(t_0 + h)$ giving

$$\hat{y}_{1,j} := \frac{P_j(y_1)}{D_j(y_1)} - \left(\frac{P_j(y_1)}{D_j(y_1)} - y_{0,j} \right) \cdot e^{-hD_j(y_1)} \ . \tag{2.6}$$

The work per step is approximately doubled, as compared to Plain Qssa. Numerical experiments have shown that Iterated Qssa performs better than Plain Qssa (in terms of precision/work ratio) only for large tolerances.

2.4 Qssa is an exponentially fitted method

Apply exponentially fitted Euler method [52] to (2.1):

$$y_1 = y_0 + h\phi(hJ)f(y_0)$$

where

$$\phi(z) = \frac{e^z - 1}{z}$$

If the Jacobian is approximated by the destruction matrix in the exponentially fitted Euler formula one obtains the plain Qssa formula (2.2)

$$\begin{aligned} y_1 &= y_0 + h\phi(-hD)f(y_0) \\ &= y_0 - h(hD)^{-1}(e^{-hD} - I)(P - Dy_0) \\ &= y_0 + (e^{-hD} - I)(y_0 - D^{-1}P) \\ &= D^{-1}P - (D^{-1}P - y_0)e^{-hD} \end{aligned}$$

In [52] W-methods are considered, which are of exponentially fitted type and have arbitrarily high orders using any approximation A of the Jacobian J. Such methods are defined by

$$
\begin{aligned}
K_i &= \phi(h\gamma A)\left(f(u_i) + hA\sum_{j=1}^{i-1}\gamma_{ij}K_j\right)\\
u_i &= y_0 + h\sum_{j=1}^{i-1}\alpha_{ij}K_j\\
y_1 &= y_0 + h\sum_{j=1}^{s}b_jK_j
\end{aligned}
$$

Following this approach and the Qssa approximation $J \approx -D$ we propose the following new exponentially-fitted-Qssa methods:

QS2O2A (2-stage, order 2)

$$(\alpha_{ij}) = \begin{pmatrix} 0 & \\ 1 & 0 \end{pmatrix}, \quad (\gamma_{ij}) = \begin{pmatrix} 1 & \\ -1 & 1 \end{pmatrix},$$

$$(b_i) = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

QS2O2B (2-stage, order 2)

$$(\alpha_{ij}) = \begin{pmatrix} 0 & \\ 2/3 & 0 \end{pmatrix}, \quad (\gamma_{ij}) = \begin{pmatrix} 1 & \\ -2/3 & 1 \end{pmatrix},$$

$$(b_i) = \begin{pmatrix} \frac{1}{4} & \frac{3}{4} \end{pmatrix}.$$

QS4O2 (4-stage order 2)

$$(\alpha_{ij}) = \begin{pmatrix} 0 & & & \\ 0 & 0 & & \\ 0 & 0 & 0 & \\ 1 & 0 & -\frac{1}{3} & 0 \end{pmatrix}, \quad (\gamma_{ij}) = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ 0 & 1 & 1 & \\ 0 & 0 & 1 & 1 \end{pmatrix},$$

$$(b_i) = \begin{pmatrix} \frac{3}{2} & -\frac{5}{6} & -\frac{5}{12} & \frac{3}{4} \end{pmatrix}.$$

QS4O3 (4-stage order 3)

$$(\alpha_{ij}) = \begin{pmatrix} 0 & & & \\ 0 & 0 & & \\ \frac{4}{3} & -\frac{2}{3} & 0 & \\ \frac{2}{3} & 0 & -\frac{2}{3} & 0 \end{pmatrix}, \quad (\gamma_{ij}) = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ 0 & 1 & 1 & \\ 0 & \frac{1}{9} & 1 & 1 \end{pmatrix},$$

with the weights

$$(b_i) = \begin{pmatrix} \frac{35}{24} & -\frac{5}{6} & \frac{3}{4} & -\frac{3}{8} \end{pmatrix}, \quad (\tilde{b}_i) = \begin{pmatrix} \frac{11}{18} & -\frac{31}{36} & \frac{3}{4} & 1 \end{pmatrix}.$$

All exponentially fitted Qssa are able to integrate chemical systems arising from atmospheric reactions. Their practical performance, however, was not found to be better than the versions of Qssa proposed further. Thus, we will not pursue this line anymore.

2.5 Extrapolation algorithms based on Qssa

A natural way to build new methods based on Qssa in the hope of better efficiency is to consider extrapolation algorithms. Some extrapolation methods have proved to be successful for very stiff problems arising in chemistry, e.g., extrapolation based on the linearly implicit Euler method or on the linearly implicit mid-point rule, see [6, 30] and [47, Section IV.9]. Therefore, extrapolation cannot be a priori discarded as a viable technique for solving the stiff systems arising in atmospheric chemistry. In general for high accuracy requirements extrapolation to high order is used, but here we are mainly interested in low order extrapolation since the accuracy requirements in atmospheric chemistry are low. One goal of this paper to analyze two extrapolation algorithms based on Qssa. Extrapolation is based on the

existence of an asymptotic expansion in h-powers for the global error. In the presence of stiffness such an expansion does not hold in general however. Nevertheless, extrapolation may already lead to a certain improvement just by reducing the error constants.

From the nonstiff situation the extrapolation algorithm based on Qssa is defined as follows. Considering a step-size H and a sequence of positive integers $n_1 < n_2 < n_3 < \ldots$, we perform n_j times the Qssa formula (2.2) with step-size $h_j = H/n_j$, and denote the result by T_{j1}. We then extrapolate these values via the recursion

$$T_{j,k+1} = T_{jk} + \frac{T_{jk} - T_{j-1,k}}{(n_j/n_{j-k}) - 1} \ . \tag{2.7}$$

The extrapolated values T_{jk} are approximations of order k to the exact solution $y(t+H)$ in the nonstiff situation.

Another type of extrapolation algorithm makes use of asymptotic expansions in even powers of h. The following algorithm is similar to the well-known GBS algorithm [46, Formula II.9.13] but it is based on Qssa. We compute

$$y_1 = e^{-D(y_0)h}\left(y_0 - D(y_0)^{-1}P(y_0)\right) + D(y_0)^{-1}P(y_0) \ , \tag{2.8a}$$

$$y_{i+1} = e^{-D(y_i)2h}\left(y_{i-1} - D(y_i)^{-1}P(y_i)\right) + D(y_i)^{-1}P(y_i) \tag{2.8b}$$

$$\text{for } i = 1, \ldots, 2n-1 \ ,$$

and then perform the following step

$$S_h(t_n) = e^{-D(y_{2n})h}\left(y_{2n-1} - D(y_{2n})^{-1}P(y_{2n})\right) + D(y_{2n})^{-1}P(y_{2n}) \ , \tag{2.8c}$$

where $t_n = t_0 + 2nh$. The extrapolation algorithm is slightly different. Here, considering a step-size H and a sequence of positive integers $n_1 < n_2 < n_3 < \ldots$, we perform the algorithm (2.8a)-(2.8c) with step-size $h_j = H/(2n_j)$ and denote the

result by $T_{j1} := S_{h_j}(t_n)$. We then extrapolate these values with the recursion

$$T_{j,k+1} = T_{jk} + \frac{T_{jk} - T_{j-1,k}}{(n_j/n_{j-k})^2 - 1} \ . \tag{2.9}$$

The extrapolated values T_{jk} are approximations of order $2k$ to the exact solution in the nonstiff situation.

In the next two sections we analyze what may happen with stiffness by considering a singular perturbation problem and its related reduced system.

2.6 The reduced system of a singular perturbation problem

Since the differential equations (2.1) modelling chemical reactions are generally stiff, the well-known phenomenon of order reduction may occur [71]. As a simplified model problem for the forthcoming analysis we consider the following *singular perturbation problem*

$$y' = -D_1(y,z)y + P_1(y,z) \ , \tag{2.10a}$$

$$z' = -\left(\frac{1}{\varepsilon}D_2(y,z) + D_3(y,z)\right) z + \left(\frac{1}{\varepsilon}P_2(y,z) + P_3(y,z)\right) \tag{2.10b}$$

with $0 < \varepsilon \ll 1$ and $D_2(y,z)$ supposed to be a diagonal matrix strictly positive definite in a neighbourhood of the solution. The above division into two classes of species is rather restrictive, but it will already give certain insights into the behaviour of the Qssa method and of extrapolation algorithms based on Qssa in the presence of stiffness.

The equations (2.10a)-(2.10b) can be rewritten as

$$\begin{aligned} y' &= D_1(y,z)(-y + C_1(y,z)) \ , \\ z' &= D_4(y,z)(-z + C_4(y,z)) \ , \end{aligned} \tag{2.11}$$

where

$$C_1(y,z) = D_1(y,z)^{-1}P_1(y,z)\ , \qquad D_4(y,z) = \frac{1}{\varepsilon}D_2(y,z) + D_3(y,z)\ ,$$
$$P_4(y,z) = \frac{1}{\varepsilon}P_2(y,z) + P_3(y,z)\ , \qquad C_4(y,z) = D_4(y,z)^{-1}P_4(y,z)\ .$$

Multiplying the equation (2.10b) by ε and letting $\varepsilon \to 0$ we obtain the *reduced system*

$$y' = -D_1(y,z)y + P_1(y,z) = D_1(y,z)(-y + C_1(y,z)) =: f(y,z)\ , \quad (2.12a)$$
$$0 = -D_2(y,z)z + P_2(y,z) = D_2(y,z)(-z + C_2(y,z)) =: g(y,z)\ . \quad (2.12b)$$

We assume that

$$g_z(y,z) \text{ is invertible} \quad (2.13)$$

in a neighbourhood of the solution which implies that the differential-algebraic system (2.12a)-(2.12b) has *index one* (cf. [47]). This assumption is actually quite natural for species with very short life-times (see (2.5)). In order to prove the convergence of the Qssa method we will need the stability assumption

$$C_2(y,z) = D_2(y,z)^{-1}P_2(y,z) \text{ is a contraction in } z \text{ for the norm } \|\cdot\| \quad (2.14)$$

be satisfied in a neighbourhood of the solution. We denote the related contractivity constant by ρ. We will see in Theorem 2.7.1 that (2.14) implies (2.13).

Let us apply the Qssa method to the stiff equations (2.10a)-(2.10b). Since $D_2(y,z)$ is a diagonal matrix with strictly positive coefficients we can take the limit $\varepsilon \to 0$ and we obtain

$$y_1 = e^{-D_1(y_0,z_0)h}\,(y_0 - C_1(y_0,z_0)) + C_1(y_0,z_0)\ , \quad (2.15a)$$
$$z_1 = C_2(y_0,z_0)\ . \quad (2.15b)$$

This is the definition of the *direct approach* of the Qssa method applied to the reduced problem (2.12a)-(2.12b). It will help us later on in Section 2.7 for the convergence analysis of the Qssa method applied to the singular perturbation problem (2.10a)-(2.10b).

Now we restrict our analysis to the differential-algebraic system (2.12a)-(2.12b) of index one and the method (2.15a)-(2.15b). Differentiating the algebraic equation (2.12b) with respect to t and omitting the obvious function arguments we obtain

$$z' = (I - C_{2z})^{-1} C_{2y} D_1(-y + C_1).$$

By expanding into Taylor-series the exact and the numerical solutions, it can be seen that the local error $\delta y_h(t_0) := y_1 - y(t_0 + h)$ and $\delta z_h(t_0) := z_1 - z(t_0 + h)$ of the direct approach Qssa method (2.15a)-(2.15b) is given by

$$\begin{aligned} \delta y_h(t_0) &= -\frac{h^2}{2}\Big(D_{1y_0}\left(D_{10}(-y_0 + C_{10}), -y_0 + C_{10}\right) \\ &\quad + D_{10} C_{1y_0} D_{10}(-y_0 + C_{10}) \\ &\quad + D_{1z0}(-y_0 + C_{10}, z_0') + D_{10} C_{1z0} z_0'\Big) + O(h^3)\ , \end{aligned} \tag{2.16a}$$

$$\delta z_h(t_0) = -h z_0' + O(h^2)\ , \tag{2.16b}$$

where the subscript 0 indicates that the function arguments are the initial values (y_0, z_0). We have given the complete expression of the first term of the error because we will make a comparison with some other methods later on. It must be noticed that even if $C_{2z}(y, z) = 0$ the local error remains $\delta y_h(t_0) = O(h^2)$ and $\delta z_h(t_0) = O(h)$. In the following theorem we give a perturbed asymptotic expansion of the global error for a constant step-size application of the method (2.15a)-(2.15b).

THEOREM 2.6.1. *Consider the index one system (2.12a)-(2.12b) with consistent initial values (y_0, z_0) and suppose that (2.14) is satisfied in a neighbourhood of the solution. Then the global error of the direct approach Qssa method (2.15a)-(2.15b) at $t_i = t_0 + ih$ satisfies for $ih \le H$*

$$\begin{aligned} y_i - y(t_i) &= h a_1(t_i) + h^2(a_2(t_i) + \alpha_i^2) + O(h^3)\ , \\ z_i - z(t_i) &= h(b_1(t_i) + \beta_i^1) + O(h^2)\ . \end{aligned}$$

The error terms are uniformly bounded for H sufficiently small. The functions

$a_1(t)$, $a_2(t)$, and $b_1(t)$ are smooth. The perturbations α_i^2, β_i^1 are independent of h and they do not vanish in general. At t_0 we have $a_1(t_0) = 0$, $a_2(t_0) + \alpha_0^2 = 0$, and $b_1(t_0) + \beta_0^1 = 0$.

Proof. To start the proof, we first show convergence of order one for the Qssa method (direct approach). It is worth noting that this part of the proof remains valid for variable step-sizes with $h = max_i|h_i|$. We use standard techniques (see, e.g., [45, Theorem 4.4] and [47, Theorem VI.7.5]). We denote two neighbouring Qssa solutions by $\{\widetilde{y}_n, \widetilde{z}_n\}$, $\{\widehat{y}_n, \widehat{z}_n\}$ and their difference by $\Delta y_n = \widetilde{y}_n - \widehat{y}_n$, $\Delta z_n = \widetilde{z}_n - \widehat{z}_n$. We suppose for the moment that

$$\|\widehat{y}_n - y(t_n)\| \le C_0 h \ , \quad \|\widehat{z}_n - z(t_n)\| \le C_1 h \ , \quad \|\Delta y_n\| \le C_2 h^2 \ , \quad \|\Delta z_n\| \le C_3 h \ . \tag{2.17}$$

This will be justified by induction below. For the Qssa method (2.15a)-(2.15b) we have the inequalities

$$\|\Delta y_{n+1}\| \ \le \ \|\Delta y_n\| + O(h\|\Delta y_n\| + h\|\Delta z_n\|) \ , \tag{2.18a}$$

$$\|\Delta z_{n+1}\| \ \le \ \rho \cdot \|\Delta z_n\| + O(\|\Delta y_n\| + h\|\Delta z_n\|) \tag{2.18b}$$

with $0 \le \rho < 1$. Applying [47, Lemma VI.2.9] we get

$$\|\Delta y_n\| \ \le \ C_4 \left(\|\Delta y_0\| + h\|\Delta z_0\|\right) \ ,$$

$$\|\Delta z_n\| \ \le \ C_5 \left(\|\Delta y_0\| + (h + \rho^n) \cdot \|\Delta z_0\|\right) \ .$$

If (y_n^k, z_n^k) with $k \le n$ denotes the Qssa solution starting on the exact solution at t_k, then the previous formula and (2.16a)-(2.16b) imply

$$\|y_n^k - y_n^{k+1}\| \ \le \ C_4 \left(\|\delta y_h(t_k)\| + h\|\delta z_h(t_k)\|\right) \le C_6 h^2 \ ,$$

$$\|z_n^k - z_n^{k+1}\| \ \le \ C_5 \left(\|\delta y_h(t_k)\| + (h + \rho^{n-k-1}) \cdot \|\delta z_h(t_k)\|\right) \le C_7 h^2 + C_8 \rho^{n-k-1} h \ .$$

Summing up we obtain

$$\sum_{k=0}^{n-1} \|y_n^k - y_n^{k+1}\| \le C_9 h \ , \qquad \sum_{k=0}^{n-1} \|z_n^k - z_n^{k+1}\| \le C_{10} h + \frac{C_{11}}{1-\rho} h \le C_{12} h \ .$$

Since the constants C_6, C_7, C_8, C_9, and C_{12} do not depend on the constants C_0, C_1, C_2,

and C_3, the assumption (2.17) is justified by induction on n provided the constants C_0, C_1, C_2, and C_3 are chosen sufficiently large and h sufficiently small.

In the second part of our proof we assume that the step-size h is constant. As in [47, Theorem VI.4.3] we are looking for a perturbed asymptotic expansion of the global error of the form

$$\begin{aligned} y_i - y(t_i) &= \sum_{j=1}^{N} h^j (a_j(t_i) + \alpha_i^j) + O(h^{N+1}) \ , \\ z_i - z(t_i) &= \sum_{j=1}^{N} h^j (b_j(t_i) + \beta_i^j) + O(h^{N+1}) \end{aligned}$$

with smooth functions $a_j(t)$, $b_j(t)$, and perturbations α_i^j, β_i^j satisfying $a_j(t_0) + \alpha_0^j = 0$, $b_j(t_0) + \beta_0^j = 0$, and

$$\alpha_i^1 \to 0 \quad \text{for} \quad i \to \infty \ . \tag{2.19}$$

For this purpose we construct recursively truncated expansions

$$\begin{aligned} \widehat{y}_i &= y(t_i) + \sum_{j=1}^{M} h^j (a_j(t_i) + \alpha_i^j) + h^{M+1} \alpha_i^{M+1} \ , \\ \widehat{z}_i &= z(t_i) + \sum_{j=1}^{M} h^j (b_j(t_i) + \beta_i^j) \ , \end{aligned}$$

such that when inserted into (2.15a)- (2.15b) we have

$$\begin{aligned} \widehat{y}_{i+1} &= e^{-D_1(\widehat{y}_i, \widehat{z}_i)h} \left(\widehat{y}_i - C_1(\widehat{y}_i, \widehat{z}_i) \right) + C_1(\widehat{y}_i, \widehat{z}_i) + O\left(h^{M+2}\right) \ , \\ \widehat{z}_{i+1} &= C_2(\widehat{y}_i, \widehat{z}_i) + O\left(h^{M+1}\right) \ . \end{aligned}$$

We first develop the above expressions into Taylor-series at t_i to obtain conditions for the smooth functions. We then develop the terms involved with the perturbations at t_0 to obtain conditions for the perturbations independently of h. Each power of h leads to two types of conditions, one for the smooth functions $a_j(t)$, $b_j(t)$ and the other for the perturbations α_i^j, β_i^j. After some tedious computations we have the following results. For $M = 0$ we simply obtain the condition $\alpha_{i+1}^1 = \alpha_i^1$ for the perturbations. Therefore by the hypothesis (2.19) we must necessarily have $\alpha_i^1 = 0$ for all $i \geq 0$. For $i = 0$ it implies that $a_1(t_0) = 0$. For $M = 1$ the smooth functions

$a_1(t)$ and $b_1(t)$ must satisfy

$$\begin{aligned} 0 &= D_{2z}(t)\left(z(t), b_1(t)\right) + D_2(t)\left(z'(t) + b_1(t)\right) \qquad (2.20a) \\ &\quad -P_{2z}(t)b_1(t) + D_{2y}(t)\left(z(t), a_1(t)\right) - P_{2y}(t)a_1(t)\ , \\ a_1'(t) &= -\frac{1}{2}y''(t) - D_1(t)a_1(t) - D_{1y}(t)\left(y(t), a_1(t)\right) \qquad (2.20b) \\ &\quad -D_{1z}(t)\left(y(t), b_1(t)\right) + P_{1y}(t)a_1(t) + P_{1z}(t)b_1(t) \\ &\quad +\frac{1}{2}D_1(t)\left(-D_1(t)y(t) + P_1(t)\right)\ . \end{aligned}$$

We have used the notation $D_1(t) = D_1\left(y(t), z(t)\right)$, etc. We can compute $b_1(t)$ from (2.20a) because of the invertibility of the matrix $g_z(t) = D_{2z}(t)(z(t), \cdot) + D_2(t) - P_{2z}(t)$. We then insert its expression into (2.20b) leading to a linear differential equation for $a_1(t)$ with initial condition $a_1(t_0) = 0$. Therefore $a_1(t)$ and $b_1(t)$ are determined uniquely from the two above equations. Putting $t = t_0$ in (2.20a), we have $b_1(t_0) \neq 0$ in general, implying that $\beta_0^1 \neq 0$. For the perturbations β_i^1 and α_i^2 we get the recurrences

$$\begin{aligned} \beta_{i+1}^1 &= D_2(t_0)^{-1}\left(P_{2z}(t_0)\beta_i^1 - D_{2z}(t_0)\left(z(t_0), \beta_i^1\right)\right)\ , \\ \alpha_{i+1}^2 &= \alpha_i^2 + P_{1z}(t_0)\beta_i^1 - D_{1z}(t_0)\left(y(t_0), \beta_i^1\right)\ . \end{aligned}$$

Therefore in general $\beta_i^1 \neq 0$ and $\alpha_i^2 \neq 0$ for all i. The remainder can be estimated as in part d) of the proof of [47, Theorem VI.4.3]. We obtain recurrence relations similar to (2.18a)-(2.18b). □

The process of determining the perturbed asymptotic expansion may be continued if the perturbations are exponentially decaying to zero. For $j \geq 2$, $a_j(t)$ and $b_j(t)$ are computed similarly to $a_1(t)$ and $b_1(t)$, and we obtain other very intricate recurrence relations for α_i^{j+1} and β_i^j. In fact it is not worth to continue this process, because here the aim of computing a perturbed asymptotic expansion is to see if the extrapolated values could be of higher order than one. Unfortunately, this cannot

happen since only the smooth function terms $a_1(t)$, $a_2(t)$, and $b_1(t)$ are eliminated by extrapolation, not the perturbation terms $\beta_i^1 \neq 0$ and $\alpha_i^2 \neq 0$. Thus Theorem 2.6.1 shows that the order of the standard extrapolation (2.7) of Qssa remains equal to one for all extrapolated values and this is a negative result. We can therefore expect that in the stiff situation the standard extrapolation of the Qssa values will generally not improve the order of the Plain Qssa. This result was confirmed numerically. Although the order remains equal to one when doing extrapolation the error constants are actually smaller and this can lead to certain improvements in efficiency for the first values of the extrapolation tableau.

We call the element T_{22} of the extrapolation tableau (2.7) with $n_1 = 1$ and $n_2 = 2$ the **Extrapolated Qssa** method. Applied with a step-size $H = 2h$ this method can be expressed as a multistage method as follows

$$\begin{aligned} Y_1 &= y_0 + \Big(e^{-D(y_0)2h} - 1\Big)\Big(y_0 - C(y_0)\Big) \ , \\ Y_2 &= y_0 + \Big(e^{-D(y_0)h} - 1\Big)\Big(y_0 - C(y_0)\Big) \ , \\ Y_3 &= Y_2 + \Big(e^{-D(Y_2)h} - 1\Big)\Big(Y_2 - C(Y_2)\Big) \ , \\ y_1 &= 2Y_3 - Y_1 \ . \end{aligned} \tag{2.21}$$

and it necessitates only two function evaluations. It is an order two method in the nonstiff case. We analyze what happens to this method when applied to the reduced system (2.12a)-(2.12b). We get for the direct approach

$$\begin{aligned} Y_1 &= y_0 + \Big(e^{-D_1(y_0,z_0)2h} - 1\Big)\Big(y_0 - C_1(y_0,z_0)\Big) \ , \qquad Z_1 = C_2(y_0,z_0) \ , \\ Y_2 &= y_0 + \Big(e^{-D_1(y_0,z_0)h} - 1\Big)\Big(y_0 - C_1(y_0,z_0)\Big) \ , \qquad Z_2 = C_2(y_0,z_0) \ , \\ Y_3 &= Y_1 + \Big(e^{-D_1(Y_2,Z_2)h} - 1\Big)\Big(Y_2 - C_1(Y_2,Z_2)\Big) \ , \qquad Z_3 = C_2(Y_2,Z_2) \ , \\ y_1 &= 2Y_3 - Y_1 \ , \qquad z_1 = 2Z_3 - Z_1 \ . \end{aligned}$$

Using Taylor-series to compute the local error of this method, we arrive at

$$\begin{aligned}
\delta y_H(t_0) &= -\frac{H^2}{2}\Big(D_{1z0}\left(-y_0+C_{10},(I-C_{2z0})^{-1}C_{2y_0}D_{10}(-y_0+C_{10})\right) \\
&\qquad + D_{10}C_{1z0}(I-C_{2z0})^{-1}C_{2y_0}D_{10}(-y_0+C_{10})\Big) \\
&\quad +O(H^3)\ , \\
\delta z_H(t_0) &= H\Big(I-(I-C_{2z0})^{-1}\Big)C_{2y_0}D_{10}(-y_0+C_{10})+O(H^2)\ .
\end{aligned}$$

We clearly see that the local error of this method contains less terms than the error (2.16a)-(2.16b) of Plain Qssa. We observe that if $C_{2z}(y,z)=0$ the local error of the Extrapolated Qssa method is $\delta y_H(t_0)=O(H^2)$ and $\delta z_H(t_0)=O(H^2)$. For this method, using a convergence proof similar to that given in Theorem 2.6.1 for Plain Qssa, we obtain convergence of order one for the y- and z-components but with a smaller error constant.

A similar analysis for the GBS-type algorithm (2.8a)-(2.8c) would be very intricate, but it has been observed numerically that there is no significant improvement when the extrapolation algorithm is used. Nevertheless, the first element of the extrapolation tableau with $n_1=2$ has given good results. Applied with a step-size $H=2h$ this multistage method reads

$$\begin{aligned}
Y_1 &= y_0+\Big(e^{-D(y_0)h}-1\Big)\Big(y_0-C(y_0)\Big)\ , \\
Y_2 &= y_0+\Big(e^{-D(Y_1)2h}-1\Big)\Big(y_0-C(Y_1)\Big)\ , \\
Y_3 &= Y_1+\Big(e^{-D(Y_2)h}-1\Big)\Big(Y_1-C(Y_2)\Big)\ , \\
y_1 &= Y_3\ , \qquad (2.22)
\end{aligned}$$

and it necessitates three function evaluations. We call this method the **Symmetric Qssa** method. It is an order two method in the nonstiff case. We analyze what happens to this method when applied to the reduced system (2.12a)-(2.12b). We

get for the direct approach

$$
\begin{aligned}
Y_1 &= y_0 + \Big(e^{-D_1(y_0,z_0)h} - 1\Big)\Big(y_0 - C_1(y_0,z_0)\Big) , \qquad Z_1 = C_2(y_0,z_0) , \\
Y_2 &= y_0 + \Big(e^{-D_1(Y_1,Z_1)2h} - 1\Big)\Big(y_0 - C_1(Y_1,Z_1)\Big) , \qquad Z_2 = C_2(Y_1,Z_1) , \\
Y_3 &= Y_1 + \Big(e^{-D_1(Y_2,Z_2)h} - 1\Big)\Big(Y_1 - C_1(Y_2,Z_2)\Big) , \qquad Z_3 = C_2(Y_2,Z_2) , \\
y_1 &= Y_3 , \qquad z_1 = Z_3 .
\end{aligned}
$$

Using Taylor-series to compute the local error of this method, we arrive at

$$
\begin{aligned}
\delta y_H(t_0) &= \frac{H^2}{2}\Big(D_{1z0}\Big(-y_0 + C_{10}, \Big(\frac{1}{2}I - (I - C_{2z0})^{-1}\Big)C_{2y_0}D_{10}(-y_0 + C_{10})\Big) \\
&\qquad + D_{10}C_{1z0}\Big(\frac{1}{2}I - (I - C_{2z0})^{-1}\Big)C_{2y_0}D_{10}(-y_0 + C_{10})\Big) \\
&\quad + O(H^3) , \\
\delta z_H(t_0) &= H\Big(I + \frac{1}{2}C_{2z0} - (I - C_{2z0})^{-1}\Big)C_{2y_0}D_{10}(-y_0 + C_{10}) + O(H^2) .
\end{aligned}
$$

We clearly see that the local error of this method contains less terms than the error (2.16a)-(2.16b) of Plain Qssa. It is also clear here that the extrapolation algorithm (2.9) cannot increase the order because the error of the first element of the extrapolation tableau does not have an asymptotic expansion in even powers of H. In fact any explicit method of Qssa-type cannot be of order greater than one for the reduced system (2.12a)-(2.12b), because of the presence of the expression $(I - C_{2z})^{-1}$ in the first derivative of the exact solution for the z-component. We observe that if $C_{2z}(y,z) = 0$ the local error of the Symmetric Qssa method is $\delta y_H(t_0) = O(H^2)$ and $\delta z_H(t_0) = O(H^2)$. For this method, using a convergence proof similar to that given in Theorem 2.6.1 for Plain Qssa, we obtain convergence of order one for the y- and z-components but with a smaller error constant.

2.7 Convergence of Qssa for the

singular perturbation problem

In this section we give a proof of convergence under certain conditions of the Plain Qssa method when applied to the singularly perturbation problem (2.10a)-(2.10b).

Because we are mainly interested in smooth solutions to (2.10a)-(2.10b) (see [47, Section VI.2]) we require as a stability assumption that the logarithmic norm of $g_z(y,z)$ satisfies [47, Formula VI.2.11]

$$\mu(g_z(y,z)) < 0 \tag{2.23}$$

in an ε-independent neighbourhood of the solution. By definition, the logarithmic norm of a matrix A is given by

$$\mu(A) = \lim_{h\to 0, h>0} \frac{\|I + hA\| - 1}{h} \tag{2.24}$$

where I is the identity matrix.

In the following theorem we show that the stability assumptions (2.14) and (2.23) may be related for the matrix norm induced by the max-norm $\|z\|_\infty = \max_{i=1}^n |z_i|$. For other norms some counterexamples below demonstrate that the two assumptions are unrelated.

THEOREM 2.7.1. *In a neighbourhood of the solution:*

1. *If $C_2(y,z)$ is a contraction in z for the norm $\|\cdot\|$ then $g_z(y,z)$ is invertible;*

2. *If in addition $D_2(y,z)$ is diagonal and strictly positive definite, and the induced matrix norm of any diagonal matrix $D = diag(d_{11},\ldots,d_{nn})$ satisfies $\|D\| = \max_{i=1}^n |d_{ii}|$, then the real parts of the eigenvalues of $g_z(y,z)$ are strictly negative;*

3. *Moreover, if $C_2(y,z)$ is a contraction in z for the max-norm then for the*

induced logarithmic norm we have

$$\mu_\infty(g_z(y,z)) < 0 \ .$$

Conversely:

4. *If $\mu(g_z(y,z)) < 0$ then the real parts of the eigenvalues of $g_z(y,z)$ are strictly negative and $g_z(y,z)$ is therefore invertible;*

5. *If $\mu_\infty(g_z(y,z)) < 0$, $D_2(y,z)$ is diagonal and strictly positive definite, and the diagonal elements of $C_{2z}(y,z)$ are non-negative, then $C_2(y,z)$ is a contraction in z for the max-norm.*

Proof. For part 1 we rewrite

$$g(y,z) = D_2(y,z)\,(-z + C_2(y,z)) \ .$$

Differentiating this expression with respect to z leads to

$$g_z(y,z) = D_{2z}(y,z)\,(-z + C_2(y,z)) + D_2(y,z)\,(-I + C_{2z}(y,z)) \ .$$

Since $g(y_0,z_0) = 0$ and $D_2(y_0,z_0)$ is invertible we have $-z_0 + C_2(y_0,z_0) = 0$. Hence we get

$$g_z(y_0,z_0) = D_2(y_0,z_0)\,(-I + C_{2z}(y_0,z_0)) \ .$$

Because $C_2(y,z)$ is a contraction in z with constant ρ for the given norm $\|\cdot\|$ we have equivalently for the induced matrix norm

$$\|C_{2z}(y,z)\| \leq \rho < 1 \ . \tag{2.25}$$

Since $D_2(y,z)$ is invertible, this completes the proof of the first part of our theorem.

For part 2 we suppose by contradiction that there exists an eigenvalue λ of $g_z(y,z)$ with non-negative real part. We denote by $v \neq 0$ a corresponding eigenvector. We will show that $v = 0$, giving the desired contradiction. We use the notation $D := D_2(y_0,z_0)$ and $C := C_{2z}(y_0,z_0)$. We have $D(C-I)v = \lambda v$ which implies that

$(I + \lambda D^{-1} - C)v = 0$. We thus obtain

$$\Big(I + \lambda D^{-1}\Big)\Big(I - \Big(I + \lambda D^{-1}\Big)^{-1} C\Big)v = 0 \ .$$

The matrix $I + \lambda D^{-1}$ is clearly invertible. The matrix $I - (I + \lambda D^{-1})^{-1} C$ is invertible too, because of the estimate

$$\| \Big(I + \lambda D^{-1}\Big)^{-1} C\| \leq \| \Big(I + \lambda D^{-1}\Big)^{-1} \| \cdot \|C\| \leq \frac{1}{|1 + \lambda / \max_{i=1}^{n} d_{ii}|} \cdot \rho \leq \rho < 1 \ .$$

We thus arrive at the contradiction $v = 0$.

For part 3, the logarithmic norm associated with the max-norm of a matrix A is given by [46, Formula I.10.20']

$$\mu_\infty(A) = \max_{i=1}^{n} \Big(a_{ii} + \sum_{j \neq i} |a_{ij}|\Big) \ .$$

For the matrix $C - I$ we get

$$\mu_\infty(C - I) = \max_{i=1}^{n} \Big(c_{ii} - 1 + \sum_{j \neq i} |c_{ij}|\Big) \leq \max_{i=1}^{n} \Big(\sum_{j=1}^{n} |c_{ij}|\Big) - 1 = \|C\|_\infty - 1 < 0 \ .$$

For the matrix $D(C - I)$ we thus have the estimate

$$\mu_\infty \left(D(C - I)\right) = \max_{i=1}^{n} \Big(d_{ii}\Big(c_{ii} - 1 + \sum_{j \neq i} |c_{ij}|\Big)\Big) \leq \min_{i=1}^{n} d_{ii} \cdot \mu_\infty(C - I) < 0 \ .$$

Conversely, for part 4, if a matrix A satisfies $\mu(A) < \alpha$ then the real part of the eigenvalues of A are strictly smaller than α. This result is a simple consequence of the definition of the logarithmic norm (2.24). We suppose by contradiction that there exists an eigenvalue λ of A satisfying $Re(\lambda) \geq \alpha$ with a corresponding eigenvector v of unit norm. We have for $h > 0$ sufficiently small

$$\frac{\|(I + hA)v\| - 1}{h} = \frac{|1 + h\lambda| - 1}{h} \geq \frac{1 + hRe(\lambda) - 1}{h} = Re(\lambda) \geq \alpha \ ,$$

implying that $\mu(A) \geq \alpha$ and giving the desired contradiction.

Finally for the last part, we have by hypothesis that

$$\mu_\infty \left(D(C - I)\right) = \max_{i=1}^{n} \Big(d_{ii}\Big(c_{ii} - 1 + \sum_{j \neq i} |c_{ij}|\Big)\Big) < 0 \ .$$

Since d_{ii} and c_{ii} are supposed to be positive we obtain

$$\sum_{j=1}^{n} |c_{ij}| - 1 < 0 \ \text{ for all } i \ .$$

Thus we get

$$\|C\|_\infty = \max_{i=1}^{n}\Big(\sum_{j=1}^{n}|c_{ij}|\Big) < 1 \ .$$

□

Here is a counterexample which shows that (2.14) does not imply (2.23) in general. We take

$$C_2(y,z) = \begin{pmatrix} 0.9z_2 \\ \\ 0.9z_1 \end{pmatrix} \ , \qquad D_2(y,z) = \begin{pmatrix} 0.1 & 0 \\ \\ 0 & 10 \end{pmatrix} \ .$$

Although $C_2(y,z)$ is a contraction for the 1-norm $\|z\|_1 = \sum_{i=1}^{n}|z_i|$ and the Euclidean norm $\|z\|_2 = (\sum_{i=1}^{n}|z_i|^2)^{1/2}$, for the corresponding induced logarithmic norms (see [46, Theorem I.10.5]) we have $\mu_1(g_z(y,z)) = 8.9$ and $\mu_2(g_z(y,z)) \approx 1.67$. It is quite amazing to notice that $C_2(y,z)$ is a contraction for the max-norm and $\mu_\infty(g_z(y,z)) = -0.01$. Most common matrix norms satisfy the condition enounced in the part 2 of the above Theorem 2.7.1, e.g., for all norms induced by the p-norms $\|z\|_p = (\sum_{i=1}^{n}|z_i|^p)^{1/p}$ with $p \geq 1$. Here is a counterexample for a norm which cannot satisfy this condition. We take

$$C_2(y,z) = \begin{pmatrix} 5.9z_1 - 5z_2 \\ \\ 5z_1 - 4.1z_2 \end{pmatrix} \ , \qquad D_2(y,z) = \begin{pmatrix} 200 & 0 \\ \\ 0 & 2 \end{pmatrix} \ .$$

The spectral radius of $C_{2z}(y,z)$ is equal to 0.9, hence there exists a norm $\|\cdot\|$ for which $\|C_{2z}(y,z)\| < 0.95$ say, but an eigenvalue of $g_z(y,z)$ is approximately equal to 978.99. A concrete example in $\mathbf{R}^2$ of a norm whose induced matrix norm does not satisfy the hypothesis in part 2 of Theorem 2.7.1 is given by $\|z\| = |z_2| + |z_2 - z_1|$. There are also counterexamples for the converse part of the Theorem 2.7.1. We choose

$$C_2(y,z) = \begin{pmatrix} 2z_2 \\ \\ 0 \end{pmatrix} \ , \qquad D_2(y,z) = \begin{pmatrix} 1 & 0 \\ \\ 0 & 4 \end{pmatrix} \ .$$

For the 1-norm and the Euclidean norm we have $\mu_1(g_z(y,z)) = -1$ and $\mu_2(g_z(y,z)) \approx -0.69$, but $\|C_{2z}(y,z)\|_1 = 2$ and $\|C_{2z}(y,z)\|_2 = 2$, i.e., $C_2(y,z)$ is not a contraction

for these norms.

We now analyze the behaviour of the Qssa method when applied to the singular perturbation problem (2.10a)-(2.10b). We will do an analysis similar to that in [47, Section VI.2]. We are mainly interested in smooth solutions of the form

$$y(t) = y^0(t) + \varepsilon y^1(t) + \varepsilon^2 y^2(t) + \dots \ , \tag{2.26a}$$

$$z(t) = z^0(t) + \varepsilon z^1(t) + \varepsilon^2 z^2(t) + \dots \ . \tag{2.26b}$$

Inserting these expansions into (2.10a)-(2.10b), multiplying (2.10b) by ε, and comparing equal powers of ε we get for ε^0,

$$\begin{aligned} y^{0\prime} &= -D_1(y^0,z^0)y^0 + P_1(y^0,z^0) \ , \\ 0 &= -D_2(y^0,z^0)z^0 + P_2(y^0,z^0) \ , \end{aligned}$$

for ε^1,

$$\begin{aligned} y^{1\prime} &= -D_{1y}(y^0,z^0)(y^0,y^1) - D_1(y^0,z^0)y^1 + P_{1y}(y^0,z^0)y^1 - D_{1z}(y^0,z^0)(y^0,z^1) \\ &\quad + P_{1z}(y^0,z^0)z^1 \ , \\ z^{0\prime} &= -D_{2y}(y^0,z^0)(z^0,y^1) + P_{2y}(y^0,z^0)y^1 - D_{2z}(y^0,z^0)(z^0,z^1) - D_2(y^0,z^0)z^1 \\ &\quad + P_{2z}(y^0,z^0)z^1 - D_3(y^0,z^0)z^0 + P_3(y^0,z^0) \ , \end{aligned}$$

etc. For the Qssa method

$$y_{n+1} = y_n + k_n \ , \qquad k_n = (e^{-D_{1,n}h} - 1)(y_n - C_{1,n}) \ , \tag{2.27a}$$

$$z_{n+1} = z_n + \ell_n \ , \qquad \ell_n = (e^{-D_{4,n}h} - 1)(z_n - C_{4,n}) \tag{2.27b}$$

we consider similar expansions

$$y_n = y_n^0 + \varepsilon y_n^1 + \varepsilon^2 y_n^2 + \dots \ , \qquad k_n = k_n^0 + \varepsilon k_n^1 + \varepsilon^2 k_n^2 + \dots \ , \tag{2.28a}$$

$$z_n = z_n^0 + \varepsilon z_n^1 + \varepsilon^2 z_n^2 + \dots \ , \qquad \ell_n = \ell_n^0 + \varepsilon \ell_n^1 + \varepsilon^2 \ell_n^2 + \dots \ . \tag{2.28b}$$

We use the notation $D_{1,n}$ for $D_1(y_n, z_n)$, $D_{1,n}^0$ for $D_1(y_n^0, z_n^0)$, etc. $C_4(y,z)$ can be developed in powers of ε as follows

$$C_4 = D_2^{-1}(1 + \varepsilon D_2^{-1} D_3)^{-1}(P_2 + \varepsilon P_3) = C_2 + \varepsilon D_2^{-1}(-D_2^{-1} D_3 P_2 + P_3) + O(\varepsilon^2) \ .$$

THEOREM 2.7.2. *Consider the singular perturbation problem (2.10a)-(2.10b) with $D_2(y,z)$ diagonal and strictly positive definite, satisfying the assumptions (2.14) and (2.23) for the max-norm, and admitting a smooth solution of the form (2.26a)-(2.26b) with initial values (y_0, z_0). Then for any fixed constant $c > 0$ the global error of the Qssa method (2.27a)-(2.27b) satisfies for $\varepsilon \leq ch$*

$$y_n - y(t_n) = O(h) \ , \qquad z_n - z(t_n) = O(h)$$

uniformly for $h \leq h_0$ and $nh \leq Const.$

Before giving the proof of this theorem we first need a perturbation lemma:

LEMMA 2.7.3. *Consider the singular perturbation problem (2.10a)-(2.10b) with $D_2(y,z)$ diagonal and strictly positive definite, satisfying the assumptions (2.14) and (2.23) for the max-norm and the Qssa method (2.27a)- (2.27b). Assume that $\|\widehat{z}_n - C_2(\widehat{y}_n, \widehat{z}_n)\|_\infty \leq Ah$, $\|\widehat{y}_n - y_n\|_\infty \leq Bh$, $\|\widehat{z}_n - z_n\|_\infty \leq Ch$, $\|\delta_n\|_\infty \leq Dh$, and $\|\theta_n\|_\infty \leq Eh$. Then for any fixed constant $c > 0$, the perturbed values*

$$\widehat{y}_{n+1} = e^{-D_1(\widehat{y}_n, \widehat{z}_n)h}(\widehat{y}_n - C_1(\widehat{y}_n, \widehat{z}_n)) + C_1(\widehat{y}_n, \widehat{z}_n) + \delta_n \ , \tag{2.29a}$$

$$\widehat{z}_{n+1} = e^{-D_4(\widehat{y}_n, \widehat{z}_n)h}(\widehat{z}_n - C_4(\widehat{y}_n, \widehat{z}_n)) + C_4(\widehat{y}_n, \widehat{z}_n) + \theta_n \tag{2.29b}$$

satisfy

$$\|\widehat{y}_{n+1} - y_{n+1}\|_\infty \leq (1 + Fh)\|\widehat{y}_n - y_n\|_\infty + Gh\|\widehat{z}_n - z_n\|_\infty + \|\delta_n\|_\infty \ , \tag{2.30a}$$

$$\|\widehat{z}_{n+1} - z_{n+1}\|_\infty \leq K\|\widehat{y}_n - y_n\|_\infty + (\alpha + Lh)\|\widehat{z}_n - z_n\|_\infty + \|\theta_n\|_\infty \tag{2.30b}$$

for $\varepsilon \leq ch$, $h \leq h_0$, where $\alpha < 1$. The constants F, G, and K do not depend on the constants A, B, C, D, and E. The constant L depends on the constants A, B, and C.

Proof. For the y-component the result is obtained by direct estimation. For the z-component this result is proved by applying the mean value theorem. We consider the vector-valued function

$$F(y,z) = e^{-D_4(y,z)h}(z - C_4(y,z)) + C_4(y,z)$$

for (y,z) in a $O(h)$-neighbourhood of (y_n, z_n). Clearly

$$\begin{aligned} F_y(y,z) &= e^{-D_4(y,z)h}\left(-hD_{4y}(y,z)(z-C_4(y,z))+I-C_{4y}(y,z)\right)+C_{4y}(y,z)\ , \\ F_z(y,z) &= e^{-D_4(y,z)h}\left(-hD_{4z}(y,z)(z-C_4(y,z))+I-C_{4z}(y,z)\right)+C_{4z}(y,z)\ . \end{aligned}$$

We have the estimate $\|F_y(y,z)\|_\infty \le k$ independently of the constants A, B, C, D, and E. In this lemma we consider by hypothesis values satisfying $\|z - C_2(y,z)\|_\infty \le \ell h$ where the constant ℓ depends on the constants A, B, and C. We get (omitting the obvious function arguments)

$$\|F_z\|_\infty \le m\ell h + \|e^{-D_4 h}(I - C_{2z}) + C_{2z}\|_\infty\ ,$$

where m is independent of the constants A, B, C, D, and E. For $h > 0$ we have

$$\begin{aligned} \|e^{-D_4 h}(I-C_{2z})+C_{2z}\|_\infty &= \max_{i=1}^{n}\Big(e^{-D_{4ii}h}+(1-e^{-D_{4ii}h})\sum_{j=1}^{n}|C_{2zij}|\Big) \\ &\le \max_{i=1}^{n}\left(e^{-D_{4ii}h}+(1-e^{-D_{4ii}h})\|C_{2z}\|_\infty\right) \\ &\le \alpha < 1 \end{aligned}$$

as a consequence of $e^{-D_{4ii}h} + (1 - e^{-D_{4ii}h})\rho \le \alpha < 1$ for all i. □

We are now in position to give a proof of Theorem 2.7.2.

Proof of Theorem 2.7.2. We insert y_n^0 and z_n^0 into the Qssa method (2.27a)-(2.27b). According to Theorem 2.6.1 the reduced system is convergent of order one so that $\|z_n^0 - C_2(y_n^0, z_n^0)\|_\infty \le Ah$. The defects satisfy

$$\begin{aligned} \delta_n &= y_{n+1}^0 - y_n^0 - \left(e^{D_1(y_n^0,z_n^0)h}-1\right)\left(y_n^0 - C_1(y_n^0,z_n^0)\right) = 0 \\ \theta_n &= z_{n+1}^0 - C_4(y_n^0,z_n^0) - e^{D_4(y_n^0,z_n^0)h}\left(z_n^0 - C_4(y_n^0,z_n^0)\right) \\ &= z_{n+1}^0 - C_2(y_n^0,z_n^0) - e^{D_4(y_n^0,z_n^0)h}\left(z_n^0 - C_2(y_n^0,z_n^0)\right) + O(\varepsilon) = O(h)\ , \end{aligned}$$

i.e., $\|\theta_n\|_\infty \le Eh$. Denoting $\Delta y_n = y_n^0 - y_n$ and $\Delta z_n = z_n^0 - z_n$, we assume that (y_n, z_n) and (y_n^0, z_n^0) satisfy

$$\|\Delta y_n\|_\infty \le Bh\ , \qquad \|\Delta z_n\|_\infty \le Ch\ , \tag{2.31}$$

this will be justified by induction below. We apply Lemma 2.7.3 to obtain

$$\begin{aligned} \|\Delta y_{n+1}\|_\infty &\leq (1+Fh)\|\Delta y_n\|_\infty + Gh\|\Delta z_n\|_\infty \,, \\ \|\Delta z_{n+1}\|_\infty &\leq K\|\Delta y_n\|_\infty + (\alpha + Lh)\|\Delta z_n\|_\infty + Eh \end{aligned}$$

where the constants F, G, and K do not depend on the constants A, B, C, and E. The constant L depends on the constants A, B, and C, but does not vary with n. We can apply [47, Lemma VI.2.9] to get the desired result. The hypotheses (2.31) are satisfied by induction on n provided the constants A, B, and C are chosen sufficiently large and h is sufficiently small, but independently of ε.

2.8 Description of the test problem

To test the properties of different numerical methods we have chosen the Carbon Bond Mechanism IV (CBM-IV) (Gery et al., [40]), consisting of 32 chemical species involved in 70 thermal and 11 photolytic reactions. The concentration of H_2O is held fixed throughout the simulation. This mechanism is designed for the numerical simulation of chemical processes in urban and in regional scale models. For the numerical experiments the chemical mechanism is run for a simulation time of 5 days. The rate constants and initial conditions follow the IPCC (Intergovernmental Panel on Climate Change) Chemistry Intercomparison study (see [70]) scenario 3 ("Bio"). An operator-splitting environment (the atmospheric convection-diffusion-reaction equation is solved with the method of fractional steps [95]; chemistry and transport are considered separately and integrated with different step-sizes) is simulated with a time step of 20 [*minutes*] for the transport scheme. Emission levels of 0.01 [*ppb/hour*] of NO, 0.01 [*ppb/hour*] of NO_2 and 0.1 [*ppb/hour*] of isoprene are considered. These emissions are injected in the system in equal quantities at

the beginning of each 20 $[minutes]$ interval.

To describe the stiffness of the problem we have computed both the eigenvalues λ_i of the Jacobian and the destruction rates D_i. The relation $-D_i \approx \lambda_i$ allows us to associate the eigenvalues with the largest negative real parts to certain short-lived species. For the real part of the spectrum we have found the following values: $-8.11 \cdot 10^8$ $[O(^1D)]$, $-8.26 \cdot 10^4$ $[O(^3P)]$ $-2.47 \cdot 10^3$ $[ROR]$, -3.5 $[OH]$, -4.2 $[TO_2]$, all others being in the interval $[-0.14,\ -10^{-8}]$. The problem is very stiff since time steps of 1 $[nanosecond]$ are prohibitively small considering an integration interval of 20 $[minutes]$ and the low accuracy required. For this problem the fact that the eigenvalues with the largest negative real parts are isolated and can be associated with certain species indicates that the singular perturbation model (2.10a)-(2.10b) makes sense.

2.9 Numerical results

In this section the results for the test problem are compared to the solution computed by the code RADAU5 of E. Hairer and G. Wanner [47] with very tight tolerances $rtol = 10^{-12}$ and $atol = 10^{-10}$ $[molecules/cm^3]$.

As a measure of the accuracy we have employed the *number of accurate digits (NAD)* computed as follows

$$NAD = \frac{1}{N}\sum_{i=1}^{N} NAD_i\ , \qquad NAD_i = -\log_{10}\left(ERR_i\right)\ ,$$

where N is the number of species, ERR_i a measure of the relative error in the numerical solution of species i and NAD_i the corresponding number of accurate digits. With the "exact" solution $y(t)$ (computed by RADAU5) and the numerical solution $\hat{y}(t)$ at hand at discrete times $\{t_j = t_0 + j \cdot \Delta t,\ 0 \le j \le M\}$ the measure

of the relative error is computed as follows

$$ERR_i = \sqrt{\frac{1}{|\mathcal{J}_i|} \cdot \sum_{j \in \mathcal{J}_i} \left| \frac{y_i(t_j) - \hat{y}_i(t_j)}{y_i(t_j)} \right|^2}, \qquad \mathcal{J}_i = \{0 \leq j \leq M : \ |y_i(t_j)| \geq a\} .$$

The threshold factor used here is $a = 1$ $[molecules/cm^3]$. If the set $\mathcal{J}_i$ is empty, the value of ERR_i is neglected. The purpose of considering the above defined error measure instead of the root mean square norm ($a = 0$ $[molecules/cm^3]$) is to suppress from the error calculation the times where the absolute value of the concentration falls below $a = 1$ $[molecules/cm^3]$; these values are very likely corrupted and the corresponding large relative errors say nothing about the general computational accuracy. From a physical standpoint, for atmospheric chemistry applications, values of $a = 1$ $[molecules/cm^3]$ or less can be assimilated to a complete extinction of the species.

In what follows we denote the current step-size by h. The integrators used are the following:

1. **Plain Qssa** integrates all the species with formula (2.2).

2. **DAE Qssa** is used with a dynamic partition of the species into slow, fast, and normal. At each time step we have:

   - If $\tau_i > 100 \cdot h$ the species is slow and is integrated with (2.3);
   - If $\tau_i < 0.1 \cdot h$ the species is fast and is integrated with (2.4);
   - Otherwise, formula (2.2) is applied.

3. **Iterated Qssa** is similar to DAE Qssa, but has one extra iteration (2.6).

4. **Chemeq** (see [96], implemented in CALGRID) is used as specified in [77]:

   - If $\tau_i < 0.2 \cdot h$ the species is fast and is integrated with (2.4);

- If $\tau_i > 5 \cdot h$ the species is slow and is integrated with the nonstiff Chemeq formula;
- For all other species the Chemeq stiff formula is used.

5. **Extrapolated Qssa** (2.21) which uses the difference $y_1 - Y_3 = Y_3 - Y_1$ as an error estimator for the step-size control.

6. **Symmetric Qssa** (2.22) which uses for the step-size control the difference $y_1 - Y_4$ where Y_4 is just one cheap extra Qssa step using the function evaluation at y_0 needed for Y_1

$$Y_4 = y_0 + \Big(e^{-D(y_0)2h} - 1\Big)\Big(y_0 - C(y_0)\Big) \ .$$

7. **Twostep** is based on the variable step size, two-step backward differentiation formula BDF2 [87, 91, 89]. Instead of a modified Newton process, Gauss-Seidel iterations are used for solving the nonlinear system of equations. This technique carefully exploits the production-loss form of the differential equation (see [87] for details). We have used the original implementation obtained directly from the authors. To accelerate the convergence of the Gauss-Seidel iterations, the species have been sorted in decreasing order relatively to the size of their destruction rates.

8. **Vode** (a BDF code, see [11, 12])) is similar to LSODE (Livermore Solver for ODE, see [51]), widely used by atmospheric modellers. Vode is considered to have several advantages over LSODE when used to integrate systems of ODE arising from chemical kinetics (see [12]). In order to take full advantage of the sparsity pattern of the Jacobian, Vode has been modified as described in [75] by replacing the general factorization and substitution routines `dgefa`

and `dgesl` with specialized sparse routines. Results for both the standard and the modified Vode are presented.

All integrators have been used with a lower bound of 0.01 [*seconds*] imposed on the chosen step-size.

The emissions of NO, NO_2 and $ISOP$ introduce transient regimes at the beginning of each hourly interval. At these moments, a complete restart is carried out for all integrators. More exactly, an exit from the integration subroutine is performed, the step-size is reset to its default value of 1 [*second*], and the subroutine is called again (each call to Vode has been done with `istate` $= 1$). In a 3-D operator-splitting model each two consecutive calls to the chemical kinetics integrator are separated by a step of the transport scheme, which may change significantly the concentration values. As a consequence, for comprehensive atmospheric models, a periodic restart of the chemical integrator is a necessity.

Figure 2.1 reports the CPU time versus the number of accurate digits (NDA) for the different integrators.

The efficiency of Plain Qssa is improved by treating with DAE Qssa separately the steady-state species on one hand and the slow species on the other hand. This conclusion is in agreement with the practical experience of Qssa users.

The extra function evaluation used in Iterated Qssa pays back for large values of *rtol*; if more accuracy is needed then this strategy is not better than the classic DAE Qssa approach. Several numerical tests have shown that employing more than one iteration decreases the efficiency of Iterated Qssa.

Vode uses the highest order formulas among the tested algorithms. This fact can be observed from the smaller slope in the work-precision diagram of Figure 2.1. A high order method pays back when an accurate solution is needed; Sparse Vode

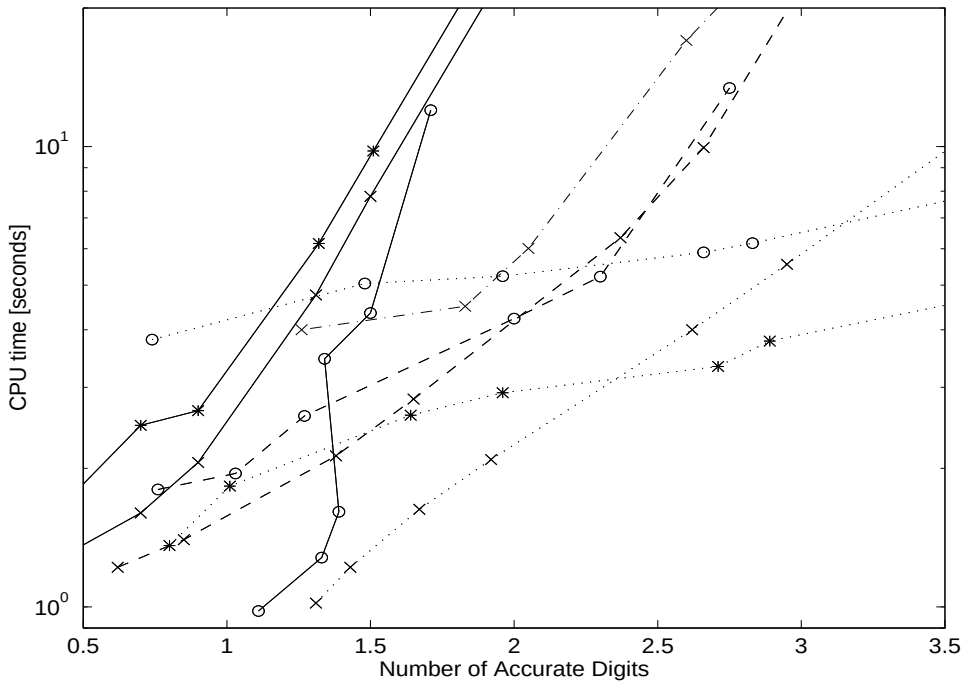

Figure 2.1. Work-precision diagram for CBM-IV. Plain Qssa (solid with "∗"), DAE Qssa (solid with "x"), Iterated Qssa (solid with "o"), Chemeq (dash-dots with "x"), Extrapolated Qssa (dashed with "x"), Symmetric Qssa (dashed with "o"), Twostep (dotted with "x"), Vode (dotted with "o"), and Sparse Vode (dotted with "∗").

is the most efficient for 2.5 or more NAD. In the low accuracy range required by atmospheric chemistry simulation the off-the-shelf code Vode is not competitive, since its performance is affected by frequent restarts; this is one of the reasons why atmospheric scientists have chosen to develop their own integrators rather than using general solvers. However, if the linear algebra is done such that full advantage of the structure of the problem is taken (see [75]) the computational time of Vode is greatly reduced (the use of sparse linear algebra routines with Vode reduces the total computational time for our test problem by a factor between two to three) and the code becomes competitive.

Extrapolated Qssa and Symmetric Qssa perform well compared to DAE Qssa, Iterated Qssa or Chemeq (especially when a NAD higher than 1 is required) but not better than Sparse Vode or Twostep. In three dimensional atmospheric models, two accurate digits in the solution of chemical kinetics equations is an acceptable value. More precision is thought to be redundant due to inaccuracies in the transport scheme; less precision can have an unpredictable effect on the overall accuracy through the transport scheme with an operator splitting algorithm. For this level of accuracy (two significant digits) Twostep performs as the best among the tested numerical integrators.

A componentwise analysis of the numerical error shows that smooth components like O_3 are integrated correctly by all methods. However, the species involved in fast photo-chemistry are integrated less precisely. Peaks of error appear exactly during sunset and sunrise periods (although in the measure reported here this is not apparent). The two new methods are more accurate and efficient than the classic Qssa ones or Chemeq, but they are not as fast as the BDF codes Twostep and Sparse Vode.

The experimental conclusions presented here are restricted to the model used and to the set of algorithms employed. More numerical tests are necessary before drawing a general conclusion. The authors are currently involved in a comprehensive benchmark work that will test most of the old and new algorithms (see [76]).

2.10 Lumping

Modifications, such as lumping, can improve their performance notably (see e.g. [89] where this is illustrated for the EUSMOG problem), but have the disadvantage of being problem dependent. With lumping we mean here the technique of grouping species into chemical families to reduce, for example, the stiffness, or to enforce conservation for a chemical family (see [49, 50] where this form of lumping was proposed first for the Qssa method).

2.11 Other methods

We briefly present other methods from the literature.

2.11.1 Twostep

Twostep [87, 89, 91] is based on the variable step size, two-step backward differentiation formula (BDF)

$$y^{n+1} = Y^n + \gamma h f(t_{n+1}, y^{n+1}), \qquad h = t_{n+1} - t_n, \tag{2.32}$$

where $\gamma = (c+1)/(c+2)$, $c = (t_n - t_{n-1})/(t_{n+1} - t_n)$ and

$$Y^n = \left((c+1)^2 y^n - y^{n-1}\right)/(c^2 + 2c).$$

The 2nd-order BDF formula (also denoted by BDF2 has been chosen in view of the modest accuracy requirement. Rather than using the common modified Newton iteration, the classical Gauss-Seidel or Jacobi iteration technique is used for computing an approximation to the implicitly defined y^{n+1}. In the application of these techniques we exploit, to some extent, the production-loss form, by which (2.32) can be written as

$$y^{n+1} = F(y^{n+1}) := \left(I + \gamma h L(t_{n+1}, y^{n+1})\right)^{-1} \left(Y^n + \gamma h P(t_{n+1}, y^{n+1})\right). \qquad (2.33)$$

The Gauss-Seidel technique is now applied to the nonlinear system of equations $y = F(y)$. That is, given the iterate $y^{(i)}$ as the i-th approximation for the sought solution y^{n+1}, Twostep computes the next iterate $y^{(i+1)}$ by the componentwise formula

$$y_k^{(i+1)} = F_k\left(y_1^{(i+1)}, \ \ldots\ , \ y_{k-1}^{(i+1)}, y_k^{(i)}, \ \ldots\ , \ y_m^{(i)}\right), \qquad k = 1, ..., m. \qquad (2.34)$$

This results in an explicit computation owing to the diagonal form of L. More precisely, for the computation of $y_k^{(i+1)}$ only division by the scalar variable $1 + \gamma h L_k(t_{n+1}, v)$ is required, where v denotes the intermediate vector

$$v = [y_1^{(i+1)}, \ \ldots\ , \ y_{k-1}^{(i+1)}, y_k^{(i)}, \ \ldots\ , \ y_m^{(i)}]^T.$$

The fact that in v the first $(k-1)$ components are taken from the new iterate, makes (2.34) a Gauss-Seidel type iteration process. When we take all m components in v from the old iterate $y^{(i)}$, then an iteration method of Jacobi or Picard type results. Computationally there is not much difference between the two, although Gauss-Seidel has to be programmed in line. Hence Jacobi iteration is somewhat easier to use. Here we will use both types of iteration techniques. Note that for the Gauss-Seidel technique the order of the species plays a role when only a small number of iterations are used. In all experiments we in fact restrict the number of iterations to only two. In [89, 91] this has been shown to work well.

Twostep is based on a two-step formula which cannot be applied at an operator-split restart. At restart the one-step backward Euler formula is therefore used, which simply means that $Y^n = y^n$. The explicit iteration process is the same, so that there is no additional penalty at restart associated with extra linear algebra computations as is the case for implicit solvers using a Newton type iteration. Twostep allows a maximal growth in step size at restart by a factor of two. This is less than one-step solvers usually allow, but still quite acceptable in view of the explicit iteration process. For more details on the code we refer to [89, 91] and [39].

2.11.2 Chemeq

One of the first dedicated, explicit methods for solving chemical equations in comprehensive advection-reaction models is the hybrid algorithm of Young and Boris [96]. It is currently implemented in the Calgrid mesoscale model [94]. In the original algorithm, the species are dynamically separated into two categories (fast and slow) according to the relative magnitude of their life-times $\tau_k = 1/L_k$ with respect to the current step size h. Each category is integrated with a special predictor-corrector formula. Our implementation of Chemeq follows the one described in [77] and is based on the following predictor-corrector pairs (the abbreviation P_k^n stands for $P_k(t_n, y^n)$, etc.):

- If $\tau_k > 5h$ (slow species):

$$\text{predictor} \quad : \quad y_k^{n+1} = y_k^n + h(P_k^n - L_k^n\, y_k^n) \tag{2.35a}$$

$$\text{corrector} \quad : \quad y_k^{n+1} = y_k^n + \frac{h}{2}(P_k^n - L_k^n\, y_k^n + P_k^{n+1} - L_k^{n+1} y_k^{n+1}) \tag{2.35b}$$

- If $0.2h \le \tau_k \le 5h$ (intermediate species):

$$\text{predictor} \quad : \quad y_k^{n+1} = \frac{y_k^n\,(2\tau_k^n - h) + 2hP_k^n\,\tau_k^n}{2\tau_k^n + h} \tag{2.36a}$$

$$\text{corrector} \quad : \quad y_k^{n+1} = \frac{y_k^n \left(\tau_k^n + \tau_k^{n+1} - h\right) + \frac{h}{2}\left(P_k^n + P_k^{n+1}\right)\left(\tau_k^n + \tau_k^{n+1}\right)}{\tau_k^n + \tau_k^{n+1} + h} \tag{2.36b}$$

- If $\tau_k < 0.2h$ (fast species):

$$\text{steady state assumption} : \; y_k^{n+1} = \frac{P_k^n}{L_k^n} \tag{2.37}$$

A quick inspection will reveal that the corrector formulas are all derived from the implicit trapezoidal rule. They are applied, however, in an explicit manner. Hence, denoting $y_k^{(i)}$ as the k-th component of the i-th corrected approximation $y^{(i)}$ for y^{n+1}, in all occurrences $y_k^{(i)}$ is simply substituted in the right-hand sides of the corrector formulas, so as to compute the new approximations $y_k^{(i+1)}$. The correctors are applied until

$$\max_k \left| \frac{y_k^{(i+1)} - y_k^{(i)}}{y_k^{(i+1)}} \right| \leq 0.3 \; tol,$$

where tol is an imposed tolerance value for the step size control which is based on $atol$ and $rtol$. In case of non-convergence, the step is rejected and the computations restarted with $h_{new} = 0.6h$. In case of convergence the integration proceeds with $h_{new} = h \min(1.1, fac)$, where $fac = \sqrt{tol/err}$ with err the estimation for the local truncation error. It is emphasized that the step selection in the original Chemeq is based uniquely on the convergence of corrector iterates, whereas we use an estimation for the local error to govern the step size. However, following Chemeq philosophy, a local error greater than tol does not force a step rejection. It only restricts h_{new}. See [39] for more details.

2.11.3 ET

The solver ET uses an extrapolation algorithm proposed in [23]. The approximations used for the extrapolation are computed with a predictor-corrector pair of which the corrector is a Qssa type formula. To describe the formulas used, we adopt

the implicit notation used for Chemeq. Hence the evaluations of the right-hand side of the implicit formulas have to be thought of as carried out in the same way as for Chemeq. The predictor formula implemented in the solver tested in this paper is the simple Qssa formula

$$y^{n+1} = e^{-hL^n} y^n + (I - e^{-hL^n})(L^n)^{-1} P^n. \tag{2.38}$$

Like for Chemeq, the corrector dynamically separates the species into three categories:

- If $\tau_k > 100h$, the trapezoidal rule is used,
$$y_k^{n+1} = y_k^n + \frac{h}{2}(P_k^n - L_k^n y_k^n + P_k^{n+1} - L_k^{n+1} y_k^{n+1}). \tag{2.39}$$
- If $0.1h \leq \tau_k \leq 100h$, the equations are corrected using the Qssa type formula
$$y_k^{n+1} = \psi_k^{n+1} + (y_k^n - \psi_k^{n+1}) \exp\left[-\left(\frac{1}{L_k^n} + \frac{1}{L_k^{n+1}}\right)\frac{h}{2}\right], \tag{2.40}$$
where ψ_k^{n+1} is defined by
$$\psi_k^{n+1} = \frac{1}{4}\left(P_k^{n+1} + P_k^n\right)\left(\frac{1}{L_k^{n+1}} + \frac{1}{L_k^n}\right). \tag{2.41}$$
- If $\tau_k < 0.1h$, the steady state assumption is made, i,e.,
$$y_k^{n+1} = \psi_k^{n+1}. \tag{2.42}$$

The actual implementation uses a variable step size. For details about how the extrapolation is organized and the corrector is used we refer to [23]. See also [39] for more details.

2.11.4 Ebi

The Euler Backward Iterative (Ebi) method was proposed by (Hertel et al., 1993). Being based on the Euler backward implicit formula (3.2), its main feature is that, instead of using Newton's method, the implicit solution is approximated through a semi-analytical, problem dependent iteration process. This process groups species together which allow an exact solution of the implicit equations after putting part of them at the old time level. Species equations which do not fit in an appropriate grouping are treated with a form of Jacobi iteration. Satisfactory results are reported (Hertel et al., 1993) for different scenario's based on the CBM-IV mechanism. The approach can also be applied when using higher BDF methods since use of these implicit methods leads to a similar system of equations, but a considerable drawback is that the iterative solution method is adapted to the particular chemistry scheme. We therefore have tested the method only for the TMk model, using an implementation obtained from (Dentener, 1996). This implementation contains no local error control mechanism so that constant step sizes are taken.

2.12 Concluding remarks

Qssa-based algorithms are explicit methods and yet they enjoy a remarkable stability. They behave like implicit methods although their evaluation formulae is explicit. Although their relative error can be large, we must mention that their absolute error is small and that the Qssa solutions are close to the exact solution even for rapidly-varying components like NO; Qssa-based methods preserve quite well the overall behaviour of the solution. This explains why these methods have

been successfully employed for many years for problems where relatively large errors are accepted and small computing times are desired.

In [92] the local truncation error for Plain Qssa scheme is shown to be only $O(h)$ for the components with small lifetimes $\tau_i \ll h$. However, numerical experiments have shown that the Qssa solutions still converge to the exact solution. The fact that the local order reduction is not felt globally is in line with the theoretical convergence analysis presented here.

The analysis and experiments show that the most attractive features of Qssa-type methods are their small computational time and their easy coding, while their main weaknesses are their low order and their relatively low accuracy. In an attempt to overcome these weaknesses, the analysis of Qssa has led us to two new methods, the Extrapolated and the Symmetric Qssa. They clearly perform better than the classic Qssa versions and the hybrid algorithm Chemeq. Then considering a complete 3-D model involving transport of chemical species, Qssa-type methods allow for lumping of species that results in increased efficiency. However, they are not as fast as the BDF codes Sparse Vode and Twostep for the test problem presented here. As will we see later, Qssa type methods are not competitive at all when compared to sparse Rosenbrock methods.

CHAPTER 3
IMPLICIT METHODS

3.1 Introduction

The equations arising from chemical kinetics comprise a system of stiff ordinary differential equations (ODE). Characteristic times in the range of *nanoseconds* are present in the system (due to e.g. atomic oxygen O^{1D}), while numerical time steps of several *minutes* are needed for an efficient integration; the smallest time scales of interest are minutes to hours. For solving these equations numerically, implicit integrators with infinite stability regions are likely to work with large step-sizes when the accuracy requirements are not too stringent. In consequence, for comprehensive air pollution models it is of interest to replace traditional explicit integrators (Qssa, Chemeq) by more robust implicit integrators. Besides stability several other arguments further motivate this interest:

- The wide range of chemical conditions that are to be simulated can cause numerical problems when explicit integrators are used; on the other hand, implicit methods offer uniformity in performance for equations of variable stiffness and difficulty. This uniformity is important as comprehensive air quality models usually contain chemical conditions ranging from ground level to upper troposphere and from marine environments to heavily polutted urban centers.

- For problems involving inter-phase mass transfer explicit codes may become unstable; an example of gas-liquid chemistry for which all explicit integrators

completely fail can be found in [76].

- The higher orders of consistency of standard implicit methods lead to substantial improvements in accuracy. Using higher order methods based on explicit formulas may not pay off, because of the order reduction phenomenon (see [56, 92]), and since higher accuracy and stability are in general contradictory requirements (see [47]).

- Multistep, Runge-Kutta and standard Rosenbrock methods all enjoy the property of conserving the linear invariants of the system (for example, they are *structurally* mass-conservative). Neither Qssa nor Chemeq have this desirable property.

What prevented so far implicit methods from being widely used in three dimensional, comprehensive atmospheric models is the fact that they are considered too slow for this type of application (except for the case when special hardware is available - see the Smvgear code [54], running on CRAY-YMP). At each integration step, a nonlinear system of equations has to be solved. This involves the repeated evaluation of Jacobians and the solution of linear algebraic systems of dimension n, the number of species considered in the model. However, we show that *this is not the case* when the linear algebra is carefully implemented. One can enjoy all the above benefits of implicit methods while remaining computationally very competitive.

General stiff ODE solvers do not take advantage of the sparsity pattern of the Jacobian, and the number of arithmetic operations required for the numerical solution of the corresponding linear system is proportional to n^3. This is one of the reasons why general stiff ODE solvers are not very efficient for integration of chemical rate equations with a moderate to large number of species. A comparison

of the exactness and time efficiency of different integrators can be found in the paper of Shieh, Carmichael et al. [79].

On the other hand, exploitation of sparsity may significantly reduce the linear algebra overhead. Recently, several authors showed promising results with sparse BDF codes in atmospheric chemistry models [54, 89]. In this chapter we develop a systematic way of exploiting sparsity when integrating atmospheric chemistry equations. Unlike [54], our target is not a specialized architecture; we concentrate on developing machine-independent algorithms. In section 3.3 we discuss and evaluate reordering techniques that lead to minimal fill-in during LU decomposition. We then (section 3.4.3) test various linear system solvers, in particular showing that the chosen routine is twice as fast as the one used in [89]; in section 2.9 we demonstrate how the chosen solver can improve the efficiency of some state-of-the-art stiff ODE solvers. These ideas are tested on two comprehensive chemical mechanisms used to study stratospheric and tropospheric chemistry; both models are described in section 3.4.6.

Among implicit methods special attention is paid to the Runge-Kutta-Rosenbrock family; being linearly implicit, the methods from this family do not need an iteration process to obtain the numerical solution; yet, they have ideal linear stability properties, which make them well suited for the stiff atmospheric chemistry problems; they are structurally mass preserving; thus, they are excellent candidates for the integration of chemical systems. Two new methods - ROS3 and RODAS3 - are then proposed for being used in air quality models. They benefit from the special treatment of sparsity and also from their inherent properties.

3.2 About pivoting

Implicit algorithms advance the numerical solution of differential equations one step in time by solving a nonlinear system of equations. This is done by using a (modified) Newton method, which results in solving a sequence of linear systems. The numerical solution of the linear system is usually done by a direct method, i.e., by employing a (LU) factorization. The matrix that is to be factorized in implicit solvers (sometimes called "prediction matrix") is of the form

$$P = I - h \cdot \gamma \cdot J$$

where I is the identity matrix, J an approximation to the Jacobian, h the attempted step-size and γ a coefficient dependent on the method. This form of the prediction matrix holds for multistep schemes and, after an equivalence transformation, for Runge-Kutta methods as well.

Several authors ([26], [54], [89]) report good results with non-pivoting sparse linear algebra solvers, when integrating atmospheric chemistry equations. This saves computational time. Several arguments sustain this practice:

- The presence of I ensures that diagonal elements are not structurally zero;
- If a pivotal element is zero or very small, then the step-size is rejected and a new P is constructed, with a smaller h. P is diagonally dominant for all h sufficiently small, say $0 \leq h \leq h_0$, so that, at least in the limit case, no pivoting is required. We should point out here that the restriction on step-size needed for diagonal dominance may be as severe as the stability restriction imposed by an explicit method.
- Reordering the species (see below) is equivalent to performing a diagonal pivoting; of course, this does not take into account numerical values, but it

helps, because a pivot with absolute value greater than one will lead to an error amplification, and this error will corrupt each row processed at that stage. Since the columns with less elements come first, fewer row operations are needed in the initial stages; hence the error introduced by a very small pivot is not greatly amplified.

- A solution of the linear system corrupted by numerical errors (due to non-pivoting) may be thought of as the exact solution of a system with inexact Jacobian. The convergence of Newton iterations may not be affected by this approximation. This argument is, of course, not valid with Rosenbrock methods.

Theoretically, because of non-pivoting, the matrix P may be falsely "detected" as singular, and unnecessary step-size rejections may occur. Moreover, even if the step-size is accepted, more iterations per Newton step may be needed because of the increased errors in the solution of linear systems. Numerical experience (ours, and that of the authors cited earlier) shows that the above phenomena are not that important in practice.

3.3 Species ordering

The sparsity structure of the permutation matrix P is given by the sparsity structure of the Jacobian J, which in its turn is determined by the chemical interactions. Thus, for a given chemical system, the sparsity structure is constant and can be predetermined. More exactly, a maximal sparsity structure can be predetermined; some entries in J can become zero during computation, as is the case with the photolysis terms during night.

To take full advantage of the sparse structure of P, we need to permute its rows and columns such that the sparsity of the L and U factors is maximized. Among different possible strategies, we look at symmetric permutations:

$$P \leftarrow \Pi \cdot P \cdot \Pi^T = I - h \cdot \gamma \cdot \Pi \cdot J \cdot \Pi^T$$

which preserve the presence of ones in the diagonal elements and hence give the possibility of factorization without pivoting. Such symmetric permutations can be viewed as a rearrangement (renumbering) of the species involved in the chemical mechanism (if the initial ordering was $[1, ..., n]^T$, the new ordering will be $\Pi \cdot [1, ..., n]^T$). We further restrict the class of possible permutations to those given by a global strategy; more exactly, we want to compute the permutation off-line, in the preprocessing stage (hence considering only the structure of the matrix and not particular numerical entries). The same permutation is then used throughout the computation, thus reducing the workload associated with sparse data structure manipulation.

The following strategies were considered:

1. **Intuitive.** A reordering based on the reactivity of the species. This requires specific knowledge regarding the chemical mechanism and has been applied successfully in [80]. Since our aim here is to minimize the fill-in, we ordered the species decreasingly after their characteristic life-time (the inverse of average destruction term, the average being taken over the two day integration interval);

2. **Row.** The rows of P are sorted in increasing order, according to the number of their nonzero elements. If two or more rows have the same number of nonzero elements, their initial relative ordering is preserved.

3. **Column.** Same strategy as above, but applied on columns.

4. **Row∗Col.** For each diagonal element k, the value $\alpha(k) = r(k) \cdot c(k)$ is computed, where $r(k)$ is the number of nonzero elements in row k and $c(k)$ is the number of nonzero elements in column k. The permutation Π is such that the diagonal elements of $\Pi \cdot P \cdot \Pi^T$ are sorted in increasing order with $\alpha(k)$.

5. **Diagonal Markowitz.** The algorithm emulates the LU decomposition (say, the column-oriented version) of the initial matrix P. Consider that at the current step, the first k-1 columns have been processed.

   - *Step k:* To minimize the fill-in, look at the diagonal elements of the active sub-matrix $P(k : n, k : n)$ and choose the one with the lowest $\beta(k) = (r(k) - 1) \cdot (c(k) - 1)$, where $r(k)$, $c(k)$ are relative to the sub-matrix P(k:n,k:n). If the diagonal element found is (i,i), permute rows k and i and columns k and i. The procedure is equivalent to a diagonal pivoting; unlike the standard Markowitz rule, this diagonal strategy does not use numerical values; hence, it is a global strategy.

   - *Step $k+\frac{1}{2}$:* Do a symbolic decomposition step on column k, i.e. emulate the k^{th} step of a real decomposition, counting the fill-in (zero elements becoming nonzero) in P(k:n,k:n).

   - *Step k+1:* Repeat step k with the new active sub-matrix P(k+1:n, k+1:n).

   A nice feature with this algorithm is that, as a by-product, one obtains the sparsity pattern of the L and U factors.

6. **Local minimum fill-in.** Resembles the Markowitz strategy, except that, at each step, the diagonal element (i,i) is chosen such that the fill-in of P(k:n,k:n)

after performing step k of elimination is minimized. This strategy is much more expensive than Markowitz, since at stage k, there are k (symbolic) factorizations of dimension $(n-k)^2$ to be performed. The gain, compared to the diagonal Markowitz strategy, is minimal.

The results for the test problems described in section 3.4.6 are presented in Table 3.1. We report the number of non-zeros resulting after reordering and an in-place LU factorization. The real target of reordering is to minimize this fill-in. The results show that all the considered strategies (except the "Intuitive" mode) perform similarly. Interestingly, the species considered most reactive appear last in the "pure numerical" re-orderings as well. For example, for model 2 the last species (in reversed column order) are: OH, NO, HO2, NO2, NO3, O3, HCHO, ALD2 ...

Conclusion: the diagonal Markowitz criterion has a slight advantage over the others, so we recommend its use.

3.4 An evaluation of different sparse subroutines

3.4.1 Test systems

In order to evaluate the performance of different sparse solvers, we employed two test linear systems based on the chemical problems described in section (3.4.6). One numerical value of the Jacobian (corresponding to noon-time, first day) was computed. The test systems correspond to $\gamma \cdot h = 1$ and the exact solution is a vector of ones:

$$\begin{aligned} P &= I - J \,, \\ b &= P \cdot (1, ..., 1)^T \end{aligned}$$

Table 3.1. Resulting fill-ins (number of non-zeros after an in-place factorization) for the different reorderings analyzed. The test problems are discussed in section 4.6.

STRATEGY	# of non-zeros in LU	
	Model 1	Model 2
INITIAL	243	673
Intuitive	385	2900
Row	287	810
Column	278	806
Row∗Col	275	789
Diag Markowitz	275	768
Local min fill-in	274	761

The sparsity pattern of P was evaluated off-line, and the data structures needed by each tested solver were generated using a small MATLAB routine. Each of the routines described in section 3.4.3 received the linear system in its own input format. For all the tests in the next subsection, this exact solution was recovered within an error of 10^{-9}.

3.4.2 Test methodology

Usually, the solution of a sparse linear system is done in three distinct steps: `a)` analysis of sparsity pattern and preparation of data structures; `b)` sparse LU decomposition, using the information gathered at the first step; and `c)` solution of resulting pair of triangular systems.

The following particularities appear when solving ODE's arising from atmospheric chemistry:

- The sparsity structure of the Jacobian (and hence, of the prediction matrix) is given by the interactions among chemical species and hence is constant on large time intervals. This structure may be sparser during night-time when photolysis reactions shut down; since it is cumbersome to either work with different sparsity structures on different intervals or update this structure periodically, we take the simpler approach ([89]) of moving the analysis step off-line, computing a (maximal) sparsity structure, and working with it throughout the integration time interval.

- Because chord iterations are used when solving the nonlinear system, the same prediction matrix is used for more than one iteration. Hence, to each LU decomposition (step `b)`) correspond several calls to the substitution routine (step `c)`). For example, with Vode on atmospheric chemistry problems, the number of calls is about six or seven.

To emulate these characteristics, the codes were benchmarked as follows: the ANALYSE routine (corresponding to step `a)`) was called once, in the beginning; then, the DEC and SOL (corresponding to phases `b)` and `c)`, respectively) were called 10^5 times. Each call to DEC was followed by 0 and by 7 calls to SOL (for timing DEC only, and for simulating the calls made by an implicit integrator).

3.4.3 Short description of

linear system solvers tested.

3.4.3.1 Off-the-shelf solvers

1. **LINPACK** The code is available on netlib, see [31]. LINPACK uses column-oriented algorithms to increase efficiency by preserving locality of reference. Here we test the routines `dgefa` and `dgesl`. **LINPACK_U** is LINPACK, acting on the unordered matrix. More exactly, the order of the species is the reverse of that resulting from the Markowitz diagonal criterion. As a consequence, the LU factors are almost full. **LINPACK_O** is LINPACK acting on the ordered matrix. Without any further intervention, other than species reordering, the decomposition time may be cut down significantly, as seen in Tables 3.2, 3.3. This is explained by the sparsity of L.

2. **LAPACK** The code available on netlib, see [3]. LAPACK is a collection of Fortran subroutines that supersedes both LINPACK and EISPACK. Tested routines: `dgetrf` and `dgetrs`. Results for both the ordered and the unordered system are given.

3. **HARWELL MA28** The code, available on netlib, is written by I. S. Duff, Computing and Information Systems Department Rutherford Appleton Laboratory and J. K. Reid. MA28 is a Markowitz general purpose linear algebra package. The tests with Harwell package were made by calling `ma28ad` once, then calling the sequence `ma28bd`, `ma28cd` 10^5 times.

4. **Y12M** of Z. Zlatev, University of Copenhagen, is a general package for sparse systems of linear equations and is also available on netlib. The routines used: `y12mb` (analyse), `y12mc` (decompose), `y12md` (solve).

5. **SuperLU** written by J.W. Demmel, J. R. Gilbert S. Eisenstat, X. S. Li, J. Liu, J. Teo uses a supernodal approach to sparse partial pivoting. Routines used in tests: `dgstrf` (*refact = 'Y' first call, then with refactorization option) for factorization, and `dgstrs` for solution of triangular systems.

6. **UMFPACK2** of T. A. Davis, Computer and Information Science and Engineering Department , University of Florida, and I. S. Duff. Version 2.0 from September, 13, 1995 is available on netlib. UMFPACK2 uses an unsymmetric-pattern multifrontal approach (wherefrom it derives its name). `udm2fa` was called once, then `udm2rf` and `udm2so` were called 10^5 times. Different combinations of input parameters were tested (with/without permutation to block triangular form, with/without preferring diagonal pivots, with different number of columns to be examined during the pivot search, with/without iterative refinement). The timings correspond to the default values of parameters.

7. **SLAP.** was written by A. Greenbaum, Courant Institute, and M. K. Seager, Lawrence Livermore National Laboratory and is available on netlib.

   The following routines were used: `dsilus` (for performing an in-place LDU decomposition), and `dslui2` (for performing the back and forward substitutions). `dsilus` and `dslui2` were designed to perform an incomplete LDU decomposition of a sparse matrix, thus providing a preconditioner for iterative methods. No permutations are performed, and no fill-in is considered (i.e., only the elements in L and U that correspond to a nonzero position in A are computed). This features make the code fast. However, its performance is affected by the large number of indirect addressings in the inner loops. In order to use the pair `dsilus` and `dslui2` as an exact (complete) solver, we

predicted off-line the sparsity structure of the L and U factors; then we "extended" the matrix A to this structure, explicitly inserting zeros on the fill-in positions. Two versions (the original and a modified one) were considered: **SLAP_1** represents the routines `dsilus` and `dslui2` without any modification. **SLAP_2** represents the routine `dsilus` with the following modification: the ANALYSE phase (corresponding to first part of `dsilus`) is done once (off-line); the routine is then passed directly A (copied into L, D and U), as well as the sparsity pointers (IL, JL, etc.). As a consequence, a substantial improvement (as compared to SLAP_1) was obtained.

The vendor BLAS version was used with all the routines that required it, except for SuperLU (where some incompatibilities appeared and we used the provided BLAS library).

3.4.3.2 Tailored solvers

Since the main idea is to do the analysis step off-line, and then use the resulting data structures throughout the computation, we consider several implementations of Gaussian elimination without pivoting.

1. **Mod_Gauss_1.** Is a modified (column-oriented) Gauss algorithm. Its distinctive feature is that in order to reduce the number of indirect addressings, we work directly on the square matrix, rather than on a compressed row or column or on a linked list format. The sparsity patterns of L and U are computed in the preprocessing stage. All the algebraic manipulations are done in a sparse mode. The factorization (decomposition) is done in-place, and the resulting "triangular" L and U factors are used. This lead to a loss in

performance in the SOLVE phase, so that the following hybrid data structure was considered:

2. **Mod_Gauss_2** uses the full representation in the DECOMPOSITION phase, but then copies the nonzero elements into (the vectors) L and U; then SOLVE phase uses a sparse data structure. This results in a small overhead when DECOMPOSE-ing, and a large speed-up when solving. This version is more efficient when a large number of SOLVEs follow each decomposition.

3. **DOOLITTLE** This routine uses a row-oriented Doolittle factorization (see [33] for more details on this algorithm). Line k in L is computed, followed by the computation of line k in U. The initial matrix is stored in compressed row format; in addition to the vector of pointers IROW (IROW(k) indicates the beggining of line k) there is a vector of pointers IDIAG (IDIAG(k) is the position of k-th diagonal element). The presence of both IROW, IDIAG enables the code to perform an in-place LU decomposition, IROW being associated with the beginning of lines in L and IDIAG with the beginning of lines in U. No pivoting is performed, and no fill-in is considered. Thus, the initial matrix A has to be "extended", in the sense that the positions of fill-ins are computed off-line, then explicit zero entries are considered in these positions. Two versions were tested:

   - **DOOLITTLE_1.** The SOLVE phase works with the sparse data structure resulting from LU decomposition.
   - **DOOLITTLE_2.** In order to completely avoid indirect addressings, a loop-free code is generated (via the KPP symbolic preprocessor) for the SOLVE phase only.

3.4.4 Results

Timings for solving the systems resulting from our test problems with the above routines are presented in Tables 3.2 and 3.3. All the routines solved the linear system

$$P \cdot x = (I - h \cdot \gamma \cdot Jac) \cdot x = b$$

with $h \cdot \gamma = 1$ and $b = P \cdot (1, ..., 1)^T$. In all the cases the exact solution $(1, ..., 1)^T$ was recovered within an error of 10^{-10}. Several remarks:

- The new solvers SuperLU and UMFPACK are designed for very large systems of equations (several thousands - by - several thousands); their use is not justified for small to moderate size problems, as those arising from present-day atmospheric chemistry models.

- General purpose solvers like Harwell's MA package and Y12M have a significant overhead associated with pivoting and handling of more general data structures. Their results are reliable; again, this does not seem to pay for small systems (at most several hundreds by several hundreds) arising from atmospheric chemistry kinetics. However, the reasonable performance of MA28 gives us the hint that a Markowitz code (working with simplified data structures) may be well-suited for the application.

- Modified Gauss. According to the results from Tables 3.2, 3.3 the strategy of performing sparse operations on the full structure seems to work well.

  One obvious disadvantage of working with the whole matrix is the increase in the required amount of memory. More tests on different machines show

a dramatic degradation in performance if the amount of RAM is restricted. For example, Mod_Gauss on the 160 M RAM machine (see Tables 3.2 and 3.3), performs comparable with MA28. However, on a 64 M RAM machine, it is two times slower than Harwell's MA. This may be explained as follows: the elements of the matrix are addressed directly, but, because of sparsity, successive references require big jumps in the $n * n$ vector (the internal representation of the whole matrix). Thus, the references are no longer local, and the amount of cache memory influences very much the performance.

- **Doolittle.** Has the advantage of working on a uniform representation of the matrix (one vector of nonzero elements, unlike SLAP, which requires one vector for L, one for U and one for D). Each row is decompressed before, and re-compressed again after processing (see [33]). This moves most of the indirect addressings from the $O(n^2)$ loop to two $O(n)$ loops. The technique of generating loop free code for the SOLVE phase (as with Doolittle_2) speeds up considerably the code (in our test problem, a factor of three is gained in SOLVE phase). It has the disadvantage of requiring a specialised preprocessing software (in this study we generated the code with the symbolic preprocessor KPP, but the user may write a very simple C routine for this purpose).

  A loop-free code for the DECOMPOSE phase may, in principle, substantially improve the timing; on the other hand, this solution would significantly increase the resulting code, and memory problems associated with that may counterbalance the speed gain.

Table 3.2. Model A. Times per call (10^{-6} seconds) for different solvers on a HP-UX A 9000/735 with 160 M RAM machine. "DEC" is the time for one decomposition, "SOL" the time for one backward-forward substitution and "1D+7S" the time for one decomposition, followed by seven backward-forward substitutions.

Model A			
ROUTINE	DEC	SOL	1D+7S
LINPACK_U	714	73	1225
LINPACK_O	411	73	922
LAPACK_U	694	102	1408
LAPACK_O	341	102	1055
HARWELL	393	39	666
Y12	832	28	1028
UMFPACK2	900	73	1411
SuperLU	948	95	1613
SLAP_1	432	25	607
SLAP_2	263	25	438
Mod_Gauss_1	205	55	590
Mod_Gauss_2	228	26	410
DOOLITTLE_1	135	29	338
DOOLITTLE_2	135	10	205

Table 3.3. Model B. Times per call (10^{-6} seconds) for different solvers on a HP-UX A 9000/735 with 160 M RAM machine. "DEC" is the time for one decomposition, "SOL" the time for one backward-forward substitution and "1D+7S" the time for one decomposition, followed by seven backward-forward substitutions.

Model B			
ROUTINE	DEC	SOL	1D+7S
LINPACK_U	6870	355	9355
LINPACK_O	2240	355	4725
LAPACK_U	8000	493	11451
LAPACK_O	1900	493	5351
HARWELL	1150	103	1871
Y12	2880	82	3454
UMFPACK2	2720	176	3952
SuperLU	2660	246	4382
SLAP_1	2030	67	2499
SLAP_2	1100	67	1569
Mod_Gauss_1	730	138	1696
Mod_Gauss_2	840	67	1309
DOOLITTLE_1	440	93	1091
DOOLITTLE_2	440	30	650

Conclusion: Doolittle with loop-free code generated for phase `c)` seems to be the fastest routine available for atmospheric chemistry applications (this conclusion by no means applies to general linear systems).

3.4.5 Integrators used

Since the off-line estimation has shown Doolittle_2 to be the most promising sparse solver, we used it in all the numerical experiments which will be reported here. Three off-the-shelf integrators were used in the numerical experiments in both original and modified (sparse) versions. Each code is based on a different numerical scheme.

- **Vode**, the Variable coefficient ODE solver of Hindmarsch, Brown and Byrne, a BDF code. For details see [11];
- **Sdirk4** written by Hairer and Wanner, part of [47]. Is based on a stiffly accurate, five stages, order four, singly diagonally implicit Runge-Kutta method;
- **Rodas** written by Hairer and Wanner, part of [47]. Based on stiffly accurate Rosenbrock method of order four with six stages.

3.4.6 Test problems

Test problem A corresponds to a stratospheric (altitude 40 Km) box model. It is available at NASA ftp site, contact Douglas E. Kinnison, kinnison1@llnl.gov. There are 38 species involved in 84 thermal and 25 photolytic reactions at the following physical and geographical conditions: latitude 65N, temperature 241.43 K,pressure 2.7 hPa, air density $8.12 \cdot 10^{16}$ $molecules/cm^3$. This mechanism and rate constants

Table 3.4. Initial concentrations for stratospheric model A.

Species name	Initial (ppb)	Species name	Initial (ppb)
O	8.15	O_3	656
NO	10.7	NO_2	2.75
HNO_3	0.35	H_2O	6100
OH	0.2	HO_2	0.14
H_2	370	CH_4	490
CO	20	$C\ell O$	1
$HC\ell$	2.15	$HOC\ell$	0.22

have been used in the NASA HSRP/AESA stratospheric chemistry models intercomparisons. The values of initial concentrations for the most important species are given in Table 3.4.

Test problem B employs the chemical mechanism that is presently used in the STEM-II regional-scale transport/ chemistry/ removal model (Carmichael et al., [14]), consisting of 86 chemical species involved in 142 thermal and 36 photolytic reactions . The mechanism, based on the work of Lurmann et al. [59] and Atkinson et. al. [5] is representative of those presently being used in the study of chemically perturbed environments; it represents the major features of the photochemical oxidant cycle in the troposphere and can be used to study the chemistry of both highly polluted (e.g., near urban centers) and remote (e.g., marine) environments. The photochemical oxidant cycle is driven by solar energy and involves nitrogen

oxides, reactive hydrocarbons, sulfur oxides and water vapour. The chemistry also involves naturally occurring species as well as those produced by anthropogenic activities. The model was used to simulate urban conditions at ground level, temperature of 288 K and an air density of $2.55 \cdot 10^{19}$ $molecules/cm^3$. The values of initial concentrations, and the values of hourly emissions are given in Table 3.5. These emissions were performed in equal quantities at the beginning of each time interval. The chemical simulations were run for 5 days.

3.5 Numerical results

Each of the modified codes was tested for different tolerances and different restart intervals.

In this section the results for the test problem are compared to the solution computed by the code RADAU5 of E. Hairer and G. Wanner [47] with very tight tolerances $rtol = 10^{-12}$ and $atol = 10^{-10}$ $[mlc/cm^3]$.

As a measure of the accuracy we have employed the *number of accurate digits (NAD)* computed as follows

$$NAD = \frac{1}{N} \sum_{i=1}^{N} NAD_i \ , \qquad NAD_i = -\log_{10} (ERR_i) \ ,$$

where N is the number of species, ERR_i a measure of the relative error in the numerical solution of species i and NAD_i the corresponding number of accurate digits. With the "exact" solution $y(t)$ (computed by RADAU5) and the numerical solution $\hat{y}(t)$ at hand at discrete times $\{t_j = t_0 + j \cdot \Delta t, \ 0 \leq j \leq M\}$ the measure of the relative error is computed as follows

$$\mathcal{J}_i \ = \ \{0 \leq j \leq M : \ |y_i(t_j)| \geq a\} \ ,$$

Table 3.5. Initial concentrations and hourly injections for tropospheric model B.

Species name	Initial (ppb)	Injection (ppb/hour)
NO	50	1
NO_2	20	0.2
$HONO$	1	0
O_3	100	0
H_2O_2	1	0
CO	300	0
$HCHO$	10	0.2
$Aldehyde$	10	0.2
PAN	1	0
$Alkans$	50	2
$Alkens$	10	1
$Ethene$	10	0.2
$Aromatics$	20	4
$Isoprene$	10	1.0

$$ERR_i \quad = \quad \sqrt{\frac{1}{|\mathcal{J}_i|} \cdot \sum_{j \in \mathcal{J}_i} \left| \frac{y_i(t_j) - \hat{y}_i(t_j)}{y_i(t_j)} \right|^2} \; .$$

The threshold factor used here is $a = 100\ mlc/cm^3$. If the set $\mathcal{J}_i$ is empty, the value of ERR_i is neglected. The purpose of considering the above defined error measure instead of the root mean square norm ($a = 0\ mlc/cm^3$) is to suppress from the error calculation the times where the absolute value of the concentration falls below $a = 100\ mlc/cm^3$; these values are very likely corrupted and the corresponding large relative errors say nothing about the general computational accuracy. From a physical standpoint, for atmospheric chemistry applications, values of $a = 100\ mlc/cm^3$ or less can be assumed to correspond to the complete dissapearance of the species.

Table 3.6. Average speed-ups obtained.

	Model A		Model B	
$\Delta t\ [sec]$	900	3600	900	3600
Vode	2.7	2.6	4.33	4.12
Sdirk4	1.7	2.2	3.25	3.00
Rodas	1.4	1.5	2.50	2.90

In addition to presenting the results for sparse implicit integrators, we perform a comparison with the widely used algorithms Qssa (see [50]), Chemeq (see [96]) and the sparse BDF code Lsodes (see [51]).

- **Qssa** is used with a dynamic partitioning of the species into slow, fast and normal, function of step-size h and the species life-time $\tau_i = 1/D_i$.

  - If $\tau_i > 100 \cdot h$ the species is slow and is integrated with forward Euler formula;
  - If $\tau_i < 0.1 \cdot h$ the species is fast and is considered at steady state;
  - Otherwise, exponential Qssa formula is applied.

- **Chemeq** is used as specified in [77]:
  - If $\tau_i < 0.2 \cdot h$ the species is fast and is considered at steady-state;
  - If $\tau_i > 5 \cdot h$ the species is slow and is integrated with the nonstiff Chemeq formula;
  - For all other species the Chemeq stiff formula is used.

- **Lsodes** is the sparse version of the popular BDF code LSODE. LSODE and Lsodes are often used to solve the atmospheric chemical kinetics equations (see [77]). The code was used with $MF = 121$, i.e. analytical Jacobian with an inner estimation of the sparsity structure.

For implicit integrators, the same parameter setting for both sparse and off-the-shelf versions was used. The accuracy of sparse codes is (despite non-pivoting) very similar to that of the original ones.

Timings and accuracies for the original codes, the sparse versions and the explicit algorithms are presented in Figures 3.1 (test problem A) and 3.2 (test problem B). The numerical accuracy (expressed as *the number of accurate digits*) is plotted versus the CPU time (in *seconds*, as given by the UNIX routine `timex`). The important points are:

- In Figures 3.1 and 3.2 the BDF codes are represented by solid lines, Runge-Kutta by dashed lines, Rosenbrock by dash-dots and explicit methods by

dotted curves.

- The slopes of these work-precision diagrams measure the orders of convergence of different methods; implicit integrators used here have higher convergence orders, hence they display higher slopes compared to Qssa and Chemeq. For less accuracy, the latter are faster, while for higher accuracy the former become preferable. Thus, when more accuracy is desired, the higher convergence order of the implicit methods used here pays off.

- To compare the performance of different methods on the work-precision diagram, "draw" an imaginary horizontal line through the desired level of accuracy (say, two digits) and read from its intersection with the code plots the necessary CPU time for achieving that level of accuracy; the code that gives the leftmost intersection point will be the fastest for that problem.

- Note the shift in the time scales for the lower diagrams versus the upper ones. This shows the increase in CPU time associated with lowering the restart time. This increase affects mainly the performance of implicit methods; although they are capable of working with very large step-sizes, frequent restarts and the transient regimes force lower step-sizes and a large number of LU decompositions right after each restart.

- Figure 3.2 shows nicely why Chemeq may be preferred to Vode for the tropospheric test problem `B` when the restart time is 15 minutes or less. The deterioration of performance with frequent restarts precluded BDF methods to be used in 3-D air pollution models. Moreover, if the implicit codes are not supplied analytical Jacobians their performance further decreases.

- Decreasing the transport step and increasing the dimension of the chemical mechanism results in a relative advantage for special explicit methods. However, even with a restart time of 15 minutes, and fairly large chemical mechanisms, the standard integrators endowed with the above presented sparse linear algebra techniques are competitive with Chemeq and Qssa, when the requested computational accuracy is one significant digit or more.

- Each implicit code traces a pair of almost parallel lines in the diagrams, one for the off-the-shelf version and one for the sparse version; roughly speaking, these lines differ by a translation along the time axis (which corresponds to the speed-up). The line parallelism shows that both code versions perform similarly in terms of accuracy; in particular, the accuracy does not seem to be affected by non-pivoting. This experimental conclusion is in agreement with the findings of other authors (see [26, 54, 89]).

To summarize, figures 3.1 and 3.2 show that careful exploitation of sparsity leads to significant improvements in the efficiency of implicit numerical integrators. These improvements depend on the size of the problem and on the percent of the total CPU time a particular code spends in solving linear algebra. In table 3.6 we report the average speed-ups obtained with the considered codes and problems.

The results presented here are for transport time steps (i.e., restart times) of 15 minutes (a typical value for regional scale models) and 1 hour (a typical value for global scale models). Many models may use a transport time step between 15 minutes and 1 hour. The results in figures 3.1 and 3.2 point to the conclusion that, if the transport time is sufficiently large, sparse implicit methods outperform dedicated integrators in terms of accuracy/time ratio.

3.6 Conclusions on sparsity treatment

The specialized techniques for treating the sparsity described here lead to significant improvements over a general sparse code like Lsodes when integrating chemical systems. For the model problems considered here, the standard (full linear algebra) versions of Lsode and Vode perform almost similar in terms of work/ accuracy ratio; in contrast, sparse Vode is about two times faster than Lsodes.

The treatment of sparsity described here is rather conservative, since the off-line analysis of the chemical system counts every possible non-zero entry in the Jacobian. Further improvements seem possible by dynamically approximating the Jacobian with a matrix of higher sparsity.

So far, the implicit methods widely used in atmospheric modelling are Gear (BDF) methods. This work shows that methods from other families, like Runge-Kutta and Rosenbrock, can be equally competitive. We have considered three off-the-shelf codes endowed with the above described sparse techniques. Their good performance motivates further search for integrators for atmospheric modelling within the class of implicit methods.

Finally, let us mention that this treatment of sparsity is not restricted to atmospheric modelling, being applicable to the numerical solution of general chemical systems.

3.7 Rosenbrock methods

This section is devoted to a brief introduction to Rosenbrock methods; part of the notation has been adopted from (Hairer et al., 1991), where Rosenbrock methods are described in much greater detail (Sections IV.7, IV.10 and VI.3).

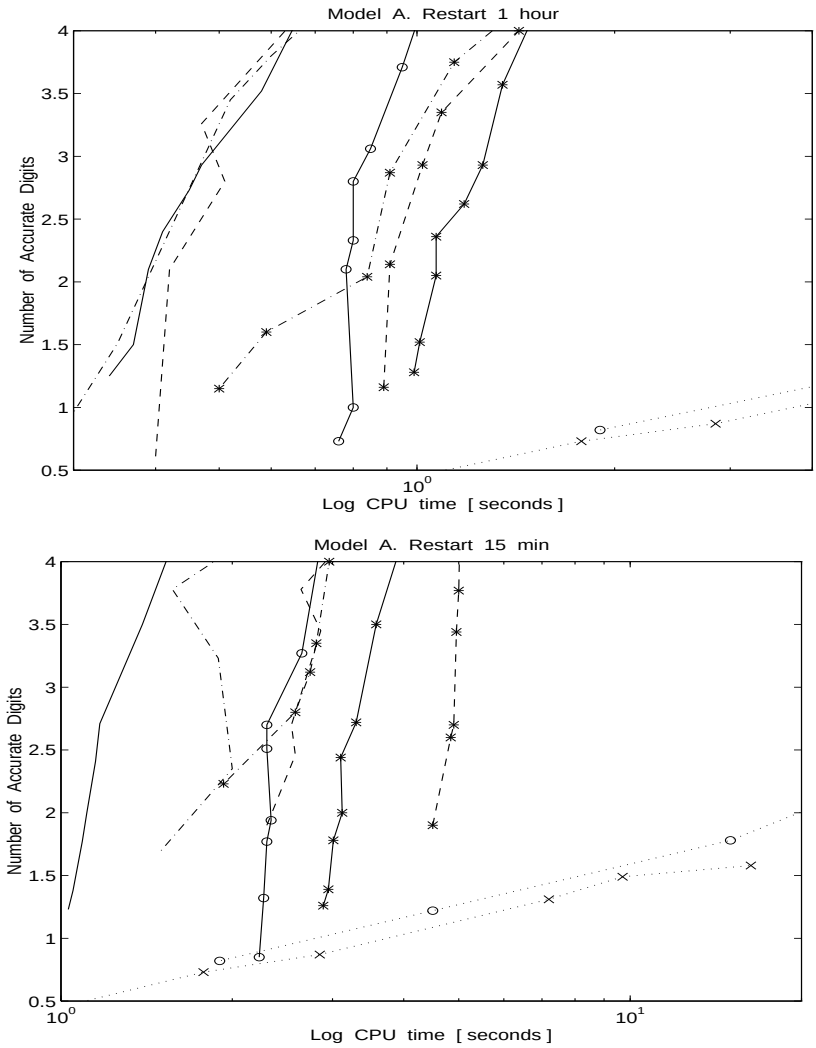

Figure 3.1. Model A. Work-precision diagram. A restart was carried each 1 hour (upper diagram) and each 15 minutes (lower diagram). Sparse Vode (solid), Vode (solid with "∗"), Lsodes (solid with "*o*"), Sparse Rodas (dash - dots), Rodas (dash - dots with "∗"), Sparse Sdirk4 (dashed), Sdirk4 (dashed with "∗"), Qssa (dots with "x") and Chemeq (dots with "o").

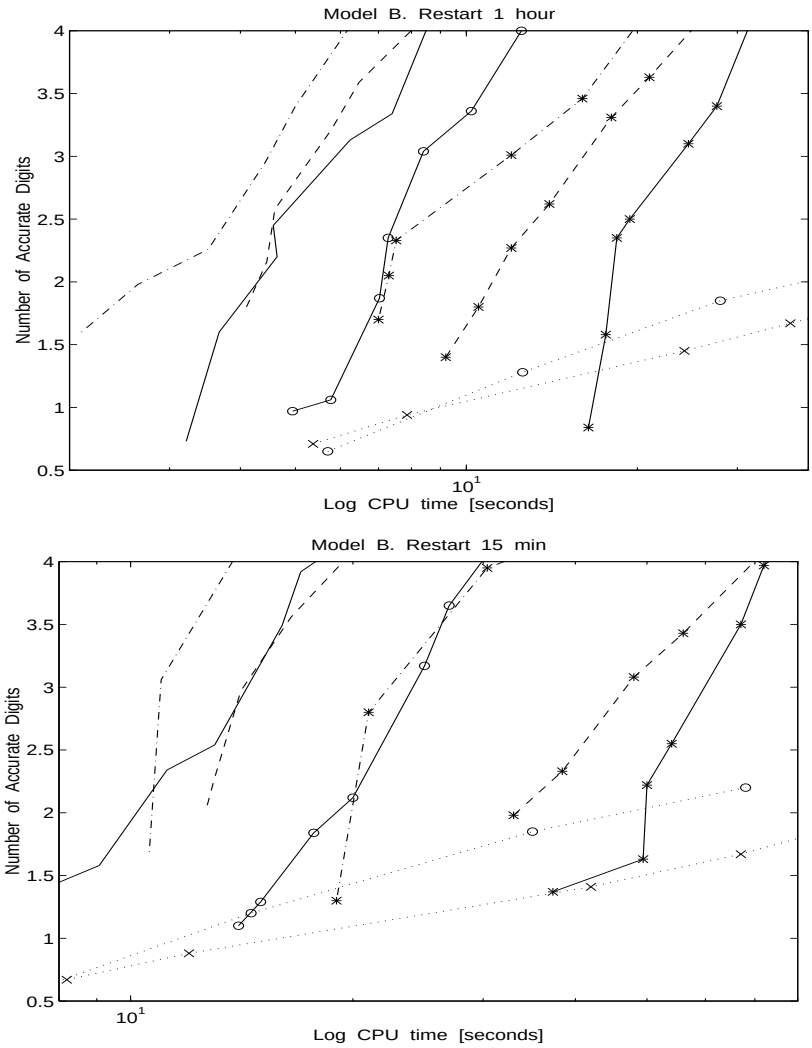

Figure 3.2. Model B. Work-precision diagram. A restart was carried each 1 hour (upper diagram) and each 15 minutes (lower diagram). Sparse Vode (solid), Vode (solid with "∗"), Lsodes (solid with "*o*"), Sparse Rodas (dash - dots), Rodas (dash - dots with "∗"), Sparse Sdirk4 (dashed), Sdirk4 (dashed with "∗"), Qssa (dots with "x") and Chemeq (dots with "o").

3.7.1 The integration formula

Rosenbrock methods are usually considered in conjunction with stiff ODE systems in the autonomous form

$$\dot{y} = f(y), \quad t > t_0, \quad y(t_0) = y_0. \tag{3.1}$$

This places no restriction since every non-autonomous system $\dot{y} = f(t, y)$ can be put in the form (3.1) by treating time t also as a dependent variable, i.e. by augmenting the system with the equation $\dot{t} = 1$. In atmospheric applications it is often the case that the reaction coefficients are held constant on each split step interval; the chemical rate equations obtained this way are in autonomous form.

Usually stiff ODE solvers use some form of implicitness in the discretization formula for reasons of numerical stability. The simplest implicit scheme is the backward Euler method

$$y_{n+1} = y_n + hf(y_{n+1}), \tag{3.2}$$

where $h = t_{n+1} - t_n$ is the step size and y_n the approximation to $y(t)$ at time $t = t_n$. Since y_{n+1} is defined implicitly, this numerical solution itself must also be approximated. Usually some modification of the iterative Newton method is used, again for reasons of numerical stability. Suppose that just one iteration per time step is applied. If we then assume that y_n is used as the initial iterate, the following numerical result is found

$$y_{n+1} = y_n + k, \tag{3.3a}$$

$$k = hf(y_n) + hJk, \tag{3.3b}$$

where J denotes the Jacobian matrix $f'(y_n)$ of the vector function f.

The numerical solution is now effectively computed by solving the system of linear algebraic equations that defines the increment vector k, rather than a system of nonlinear equations. Rosenbrock (1963) proposed to generalize this linearly

implicit approach to methods using more stages, so as to achieve a higher order of consistency. The crucial consideration put forth was to no longer use the iterative Newton method, but instead to derive stable formulas by working the Jacobian matrix directly into the integration formula. His idea has found widespread use and a generally accepted formula (Hairer et al., 1991) for a so-called s-stage Rosenbrock method, is

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i, \tag{3.4a}$$

$$k_i = hf(y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j) + hJ \sum_{j=1}^{i} \gamma_{ij} k_j, \tag{3.4b}$$

where s and the formula coefficients b_i, α_{ij} and γ_{ij} are chosen to obtain a desired order of consistency and stability for stiff problems. An introduction on the properties of consistency, stability and stiff accuracy for Rosenbrock methods is presented in an appendix.

For a reason explained later, the coefficients γ_{ii} are taken equal for all stages, i.e. $\gamma_{ii} = \gamma$ for all $i = 1, \ldots, s$. For $s = 1$, $\gamma = 1$ the above linearized implicit Euler formula is recovered. For the non-autonomous system $\dot{y} = f(t, y)$, the definition of k_i is changed to

$$k_i = hf(t_n + \alpha_i h, y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j) + \gamma_i h^2 \frac{\partial f}{\partial t}(t_n, y_n) + hJ \sum_{j=1}^{i} \gamma_{ij} k_j,$$

where

$$\alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}, \quad \gamma_i = \sum_{j=1}^{i} \gamma_{ij}.$$

Like Runge-Kutta methods, Rosenbrock methods successively form intermediate results

$$Y_i = y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j, \quad 1 \leq i \leq s, \tag{3.5}$$

which approximate the solution at the intermediate time points $t_n + \alpha_i h$. Rosenbrock methods are therefore also called Runge-Kutta-Rosenbrock methods. Observe that

if we identify J with the zero matrix and omit the $\partial f/\partial t$ term, a classical explicit Runge-Kutta method results.

Rosenbrock methods are attractive for a number of reasons. Like fully implicit methods, they preserve exact conservation properties due to the use of the analytic Jacobian matrix. However, they do not require an iteration procedure as for truly implicit methods and are therefore more easy to implement. They can be developed to possess optimal linear stability properties for stiff problems. They are of one-step type, and thus can rapidly change step size. We recall that this is of particular importance for our application in view of the many operator-split restarts.

3.7.2 Reducing computational costs

Each time step requires an evaluation of the Jacobian J, s matrix-vector multiplications with J and, assuming that $\gamma_{ii} = \gamma$, s solutions of a linear system with (the same) matrix $I - \gamma hJ$, accompanied with s derivative evaluations. The multiplications with J are easily avoided in the actual implementation by a simple transformation (see Section IV.7 of (Hairer et al., 1991)). Because of the multistage nature, the computational costs for a Rosenbrock method, spent within one time step, are often considered to be high compared to the costs of say a linear multistep method of the BDF type. In particular, the Jacobian update and the solution of the s linear systems, requiring one matrix factorization (LU-decomposition) and s backsolves (forward-backward substitutions) typically account for most of the CPU time used by a Rosenbrock method. On the other hand, if a Rosenbrock code solves the problem efficiently in fewer steps than a BDF code needs, then the CPU time for a whole integration using a Rosenbrock method can become significantly less

then for a BDF method.

Sparsity

For large atmospheric chemistry models the number of zeroes in J readily amounts to $\approx 90\%$. This high level of sparsity can be exploited to significantly reduce the costs of the linear algebra calculations. For this task we use the symbolic preprocessor KPP [25, 76]. KPP prepares a sparse matrix factorization with only a minimal fill-in (see Table 1 in (Sandu et al. 1996a)) and delivers a Fortran routine for the backsolve without indirect addressing. Altogether this means that the numerical algebra is handled very efficiently. The sparse matrix technique implemented in KPP is based on a diagonal Markowitz criterion.

Approximate Jacobians

It is conceivable to attempt to further reduce the numerical algebra costs through an approximate Jacobian.

- One possibility is to use a time-lagged Jacobian $J = f'(y_{n+\eta})$ where $\eta = 0, -1, \ldots$ such that $n+\eta$ is constant. If we define J this way, and in addition keep h fixed, then $I - \gamma h J$ is a constant matrix during the number of times that the parameter η is decreased; hence one can advance several time steps using the same LU-decomposition. The derivation of order conditions (which circumvents the order reduction associated with the time-lagging of the Jacobian) can be found in (Verwer et al, 1983a, 1983b). Since the exact Jacobians are used, conservation properties will still be maintained.

- Replacing J by a matrix with a simpler structure, say a matrix of higher sparsity, may result in further savings in linear algebra costs, but will destroy the

conservation properties. Also, the number of order conditions will significantly increase (see the W-methods of (Steihaug et al., 1979)).

- One can devise methods based on a partitioning of the species into slow and fast ones where part of the entries of J is put to zero. This approach does not maintain conservation properties either and adds the problem of devising a good partitioning strategy.

In the current paper the above ideas are not explored: only exact Jacobians are considered.

3.7.3 Step size control

General purpose stiff ODE solvers normally adapt the step size in an automatic manner to enable small step sizes at times when the solution gradients are large and large step sizes when solution gradients are small. For Runge-Kutta solvers an effective and simple step size control can be based on a so-called embedded formula

$$\tilde{y}_{n+1} = y_n + \sum_{i=1}^{s} \tilde{b}_i k_i,$$

which uses the already computed increment vectors k_i. The approximation $\tilde{y}_{n+1}$ thus differs only in the choice of the weights $\tilde{b}_i$ and hence is available at no extra costs. Usually, the weights are chosen such that the order of consistency of $\tilde{y}_{n+1}$ is $\tilde{p} = p - 1$, if p is the order of y_{n+1}. This suggests to use the difference vector $Est = \tilde{y}_{n+1} - y_{n+1}$ as a local error estimator. In what follows we will denote the order of such a pair of formulas by $p(\tilde{p})$. All the Rosenbrock solvers (Rodas, Rodas3, Ros4 and Ros3) use embedded formulas to estimate the local error.

The specific step size strategy goes as follows. Let m denote the dimension of the ODE system. Let $Tol_k = atol + rtol\,|y_{n+1,k}|$, where $atol$ and $rtol$ represent a

user-specified absolute and relative error tolerance and $y_{n+1,k}$ the k-th component of y_{n+1}. Tolerances may differ componentwise, but are here taken equal for all components for simplicity of testing. Denote

$$Err = \sqrt{\frac{1}{m}\sum_{k=1}^{m}\left(\frac{Est_k}{Tol_k}\right)^2}.$$

The integration step is accepted if $Err < 1$ and rejected otherwise and redone. The step size for the new step, both in the rejected and accepted case, is estimated by the usual step size prediction formula

$$h_{\text{new}} = h \,.\, \min\left(10,\, \max\left(0.1,\, 0.9/(Err)^{1/(\tilde{p}+1)}\right)\right).$$

At the first step after a rejection, the maximal growth factor of 10 is set to 1.0. Further, h is constrained by a minimum $h_{\min}$ and a maximum $h_{\max}$ and at any start of the integration for each operator-split interval we begin with a starting step size $h = h_{\text{start}}$. A rejection of the first step is followed by a ten times reduction of h. These step size constraints will be specified later. Because the maximal growth factor is equal to 10, the step size adjusts very rapidly and quickly attains large values if the solution is sufficiently smooth and $h = h_{\text{start}}$ is chosen small.

3.7.4 Consistency and stability

The performance of an integration method largely depends on its order of consistency and its stability properties. Again for the convenience of readers from the atmospheric research community, in this section we will briefly discuss the consistency property for the Rosenbrock method, as well as some useful results from the linear stability theory. Also some attention will be paid to the notion of stiff-accuracy.

Consistency conditions

The consistency conditions are found from a formal Taylor expansion of the local error. Let $y_{n+1} = E(y_n)$ be a compact notation for the Rosenbrock method. The difference

$$\delta_h(t) = E(y(t)) - y(t+h), \tag{3.6}$$

where y is the exact (local) solution of the ODE system $\dot{y} = f(y)$ passing through $y(t)$, is called the local error and the largest integer p for which

$$\delta_h(t) = O(h^{p+1}), \quad h \to 0,$$

is called the order of consistency. Hence $\delta_h(t)$ is the error after a single step from an exact solution, while the order reveals how rapidly $\delta_h(t)$ approaches zero for a decreasing step size. Assuming sufficient differentiability of y and f, the order p is determined by Taylor expanding the local error and equating to zero the resulting terms up to the p-th one. This leads to the so-called consistency conditions which are expressions in the formula coefficients. Satisfying these conditions gives the desired order p. While the expansion is technically complicated and the resulting conditions can become quite lengthy for a large p, the derivations are conceptually simple. For a maximum of four stages, the conditions for order $p \leq 3$ are:

$$p = 1 \quad : \quad b_1 + b_2 + b_3 + b_4 = 1, \tag{3.7a}$$

$$p = 2 \quad : \quad b_2\beta_2' + b_3\beta_3' + b_4\beta_4' = \frac{1}{2} - \gamma, \tag{3.7b}$$

$$p = 3 \quad : \quad b_2\alpha_2^2 + b_3\alpha_3^2 + b_4\alpha_4^2 = \frac{1}{3}, \tag{3.7c}$$

$$\quad : \quad b_3\beta_{32}\beta_2' + b_4(\beta_{42}\beta_2' + \beta_{43}\beta_3') = \frac{1}{6} - \gamma + \gamma^2, \tag{3.7d}$$

where

$$\beta_{ij} = \alpha_{ij} + \gamma_{ij}, \quad \alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}, \quad \beta_i' = \sum_{j=1}^{i-1} \beta_{ij}.$$

The conditions for $p \leq 5$ and general s can be found in Section IV.7 of (Hairer et al., 1991).

Linear stability

Let $\epsilon_n = y_n - y(t_n)$ denote the global error: the difference between the sought exact solution of the ODE system $\dot{y} = f(y)$ and the computed approximation. The global error at the forward time level $t = t_{n+1}$ can be seen to satisfy

$$\epsilon_{n+1} = E(\epsilon_n + y(t_n)) - E(y(t_n)) + \delta_h(t_n), \tag{3.8}$$

showing that this error consists of two parts: the local error (3.6), which is a functional of the exact solution, and the difference

$$E(\epsilon_n + y(t_n)) - E(y(t_n)),$$

where $E(\epsilon_n + y(t_n))$ represents the actual Rosenbrock step taken from the approximation $y_n = \epsilon_n + y(t_n)$ and $E(y(t_n))$ represents the hypothetical Rosenbrock step taken from the exact solution $y(t_n)$. This difference term reveals a dependence of ϵ_{n+1} on ϵ_n. For a proper functioning of the Rosenbrock method it is desirable that, in an appropriate norm $\| \ \|$,

$$\| E(\epsilon_n + y(t_n)) - E(y(t_n)) \| \leq \| \epsilon_n \|, \tag{3.9}$$

because then the integration is stable in the sense that

$$\| \epsilon_{n+1} \| \leq \| \epsilon_n \| + \| \delta_h(t_n) \|.$$

This error inequality is elementary, but also fundamental for one-step integration methods. It simply shows that all local errors add up to the global error,

$$\| \epsilon_n \| \leq \sum_{j=0}^{n-1} \| \delta_h(t_j) \|,$$

if we assume that at the initial time t_0 the error $\epsilon_0 = 0$. From inserting $\delta_h(t_j) = O(h^{p+1})$, while assuming $h \to 0$ and $n \to \infty$ such that $t_n = nh$ is fixed, it follows that $\epsilon_n = O(h^p)$. By adding up all local errors one power of h is lost, resulting in a

convergence order p.

If (3.9) does not hold, the global error can accumulate unboundedly. The integration is then unstable and of no practical use. Whereas for general nonlinear stiff ODEs from chemistry no stability analysis exists for Rosenbrock methods, their stability is well understood for stable, linear systems

$$\dot{y} = Jy, \tag{3.10}$$

with eigenvalues λ satisfying $\mathrm{Re}(\lambda) \leq 0$. From practical experience we know that linear stability often provides a satisfactory prediction of stability for nonlinear problems if J is interpreted as the Jacobian matrix $f'(y)$. This interpretation is based on a linearization argument, see (Dekker et al., 1984) and (Hairer et al., 1991). Applied to (3.10), the Rosenbrock method $y_{n+1} = E(y_n)$ reduces to the linear recursion

$$y_{n+1} = R(hJ)y_n, \tag{3.11}$$

where $R(hJ)$ is a matrix-valued rational function that approximates the matrix-valued exponential function e^{hJ}, being the solution operator of (3.10). By inserting (3.11) into the error equation (3.8), we obtain

$$\epsilon_{n+1} = R(hJ)\epsilon_n + \delta_h(t_n),$$

or, equivalently,

$$\epsilon_n = R^n(hJ)\epsilon_0 + \sum_{j=0}^{n-1} R^{n-1-j}(hJ)\delta_h(t_j),$$

where, as before, $n = 1, 2, \ldots$. We see that the demand of stability can now be expressed as boundedness of powers of $R(hJ)$, i.e.,

$$\| R^n(hJ)\| \leq C, \tag{3.12}$$

where C is a constant which is independent of n and hJ. This independence guarantees unconditional stability in the sense that no restrictions exist on the step size. Condition (3.12) holds if we require that the scalar rational function $R(z)$, which

is called the stability function, satisfies $|R(z)| \leq 1$ for arbitrary $z = h\lambda$, $\mathrm{Re}(z) \leq 0$. This is the famous property of A-stability originally proposed by Dahlquist (see (Hairer et al., 1991)). We note in passing that for our application we do not really need A-stability, since for atmospheric chemistry the eigenvalues of the Jacobian are always located in the neighbourhood of the real axis. So we actually need the boundedness property only near the negative half line.

We will impose the condition of L-stability, which in addition to A-stability, requires $R(\infty) = 0$. L-stability is known to lead to a somewhat more robust approach and better mimics the damping property of e^z for $\mathrm{Re}(z) \leq 0$. The property of L-stability is easily verified. The stability function R is found by applying the method to the scalar problem $\dot{y} = \lambda y$. This yields a rational function of the form

$$R(z) = \frac{P(z)}{(1 - \gamma z)^{s'}}, \tag{3.13}$$

where P is a polynomial of degree $s', s' \leq s$, and the degree of P is less than or equal to $s' - 1$ if the stability function is to be L-stable. Mostly, s' is equal to the number of stages s, but s' can be smaller. In this paper we only consider methods for which $s' = s$.

Stability properties of rational functions of the type (3.13) have been studied extensively. For our purpose the following results are very useful. Suppose that the order of consistency p of the Rosenbrock method is also the order of consistency of R, i.e., p is the largest integer for which $R(z) = e^z + O(z^{p+1}), z \to 0$. For L-stable functions we then usually have $p = s$ or $p = s - 1$. In both cases R is uniquely determined by γ. For the case $p = s - 1$, L-stability holds for certain intervals for γ and if $p = s$ for one particular value of γ (see Section IV.6 and Table 6.4 in (Hairer et al., 1991)). By way of illustration we list the values of γ for $1 \leq s \leq 4$ in Table 3.7.

Table 3.7. Values of γ for L-stability.

s	L-stability, $p \geq s-1$	L-stability, $p = s$
1		$\gamma = 1$
2	$(2-\sqrt{2})/2 \leq \gamma \leq (2+\sqrt{2})/2$	$\gamma = (2 \pm \sqrt{2})/2$
3	$0.18042531 \leq \gamma \leq 2.18560010$	$\gamma = 0.43586652$
4	$0.22364780 \leq \gamma \leq 0.57281606$	$\gamma = 0.57281606$

Stiff accuracy

Stiff accuracy is a property related to the Prothero-Robinson model problem

$$\dot{y} = \lambda(y - \phi(t)) + \dot{\phi}(t),$$

where ϕ is some known function. Its solution reads

$$y(t+h) = e^{\lambda h}(y(t) - \phi(t)) + \phi(t+h)$$

and if $\mathrm{Re}(\lambda h) \to -\infty$, the solution $y(t+h) \to \phi(t+h)$, irrespective the size of h. Prothero and Robinson have investigated under which conditions on the formula coefficients, implicit Runge-Kutta solutions mimic this property. Because, then a method can handle this particular transition to infinite stiffness in an accurate manner, which has been the main motivation for this test model (see (Dekker et al., 1984) and (Hairer et al., 1991)) They proposed the term stiff accuracy for this phenomenon.

For the current test model, the global error recursion (3.8) reads

$$\epsilon_{n+1} = R(z)\epsilon_n + \delta_h(t_n),$$

where $\delta_h(t_n)$ depends in a certain way on $z = h\lambda$, h and ϕ. Hairer and Wanner

(1991) show, in Section IV.15, that for any consistent Rosenbrock method,

$$\delta_h(t_n) = O(h^2/z), \quad \text{for} \quad h \to 0,\ z \to \infty,$$

if

$$\alpha_{si} + \gamma_{si} = b_i \quad (i = 1, \ldots, s) \quad \text{and} \quad \alpha_s = 1. \tag{3.14}$$

Hence, the desired transition property holds for the local error and because (3.14) also implies $R(\infty) = 0$, this property holds for the global error as well. They therefore call a Rosenbrock method stiffly accurate if (3.14) holds.

For general nonlinear stiff problems the virtue of stiff accuracy is not so clear. In (Hairer et al., 1991) it is argued that stiff accuracy is advantageous when solving stiff differential-algebraic systems with a Rosenbrock method (cf. Proposition 3.12, Sect. VI.3). For ODEs a similar argument exists which goes as follows. Suppose (3.14) holds. A straightforward computation then reveals the following relation between y_{n+1} and the final stage quantities k_s and Y_s,

$$k_s = hJy_{n+1} + hf(Y_s) - hJY_s. \tag{3.15}$$

Assuming that J is invertible, we may write

$$y_{n+1} = Y_s - (hJ)^{-1}(hf(Y_s) - k_s), \tag{3.16}$$

which is the result of one modified Newton iteration for the equation

$$hf(y) - k_s = 0, \tag{3.17}$$

using Y_s as starting value. For given k_s this equation can be interpreted as a collocation equation for a numerical solution. Hence, if the property of stiff accuracy holds, if J is invertible and Y_s a sufficiently good starting guess, then the Rosenbrock solution y_{n+1} is close to a collocation solution. Observe that for linear systems $\dot{y} = Jy$ we always have $hJy_{n+1} = k_s$ according to (3.15). If the final increment vector k_s is close to a true derivative, this collocation property seems recommendable. Other arguments supporting the notion of stiff accuracy for nonlinear problems do

not exist as far as we know.

3.7.5 New methods - Rodas3 and Ros3

RODAS3

This solver was designed along the same principles as Rodas. It is based on a stiffly accurate, embedded pair of order 3(2). The number of stages is $s = 4$, requiring four backsolves but only three derivative evaluations are used. Hence per step it needs less work than Rodas, but it is one order lower. We have selected this pair since we aim at optimal efficiency for low accuracies. To the best of our knowledge, this pair of formulas has not yet been proposed elsewhere. The coefficients α_{ij} and γ_{ij} are

$$(\alpha_{ij}) = \begin{pmatrix} 0 & & & \\ 0 & 0 & & \\ 1 & 0 & 0 & \\ 3/4 & -1/4 & 1/2 & 0 \end{pmatrix}, \quad (\gamma_{ij}) = \begin{pmatrix} 1/2 & & & \\ 1 & 1/2 & & \\ -1/4 & -1/4 & 1/2 & \\ 1/12 & 1/12 & -2/3 & 1/2 \end{pmatrix},$$

and the weights are

$$(b_i) = \begin{pmatrix} 5/6 & -1/6 & -1/6 & 1/2 \end{pmatrix}, \quad (\tilde{b}_i) = \begin{pmatrix} 3/4 & -1/4 & 1/2 & 0 \end{pmatrix}.$$

Both formulas are L-stable. Observe that the embedded one is defined by the final intermediate approximation Y_4.

ROS3

This solver is based on an embedded pair of order 3(2) and is also new. The number of stages is $s = 3$ involving three backsolves and two derivative evaluations. The third order method is L-stable and the embedded second order method is strongly A-stable ($R(\infty) = 0.5$). The stiff accuracy property is not valid for Ros3.

The method was constructed under the design criteria: order three, L-stability for both the stability function and the internal stability functions, and a strongly A-stable second order embedding. The internal stability functions are associated with the intermediate approximations (3.5). Imposing stability for these internal functions was advocated in (Verwer, 1977) as a means to improve a Rosenbrock method for strongly nonlinear stiff problems. We note in passing that if the order of consistency equals 3 and $s = 3$, then the requirement of L-stability prevents the existence of an L-stable second order embedding. The coefficients are:

$$\begin{aligned}
\gamma &= 0.435866521508458999416019451193 56 \\
\gamma_{21} &= -0.192946556960290955750096954360 41 \\
\gamma_{32} &= 1.749271481257946851735297497389 60 \\
b_1 &= -0.754574123854043158298189986465 89 \\
b_2 &= 1.941004070619644202928401233794 19 \\
b_3 &= -0.186429946765601044630211247328 29 \\
\tilde{b}_1 &= -1.533587457841495853707665239130 02 \\
\tilde{b}_2 &= 2.817451311486257722139317454576 22 \\
\tilde{b}_3 &= -0.283863853644761868431652215446 19
\end{aligned}$$

The remaining coefficients are $\alpha_{21} = \alpha_{31} = \gamma$ and $\alpha_{32} = \gamma_{31} = 0$.

3.8 Other methods

3.8.1 VODE

Vode is a "Variable coefficient Ordinary Differential Equation" solver based on the implicit BDF formulas [11, 47] and a successor of the "Livermore Solver"

Lsode from [51]. The latter is popular in the field of atmospheric chemistry. For a discussion of the mathematical techniques implemented we refer to [11, 47]. We used Vode as a black box with its user parameter $istart = 1$, except that we modified the code to carefully exploit the sparsity of the Jacobian matrix. This reduces the costs of solving the linear algebraic systems arising in the modified Newton iteration. In [54] and [89] it has been shown that this is very profitable for atmospheric chemistry problems. We used the sparse linear algebra implementation described in [75]. The necessary routines are automatically generated by the symbolic chemical preprocessor KPP [25], which transparently:

- determines the sparse analytical Jacobian,
- reorders the species using a diagonal Markowitz criterion, in order to minimize the fill-in resulting from the LU decomposition of the matrix used in the modified Newton process,
- analyses the pattern of zeros in the Jacobian and builds the data structures needed for the sparse Doolittle LU decomposition,
- generates loop-free code for the forward-backward substitution routines.

The performance of Vode appeared to be sensitive to the choice of the absolute tolerances. Using the natural value $atol = 1.0$ was not always optimal. We therefore set them componentwise as

$$atol_i = \max\left(10^{-2}, 10^{-2} \cdot rtol \cdot \eta_i\right) \quad (\mathrm{mlc/cm^3})$$

where η_i estimates the magnitude of the concentration of species i. See [39] for other specific parameter settings.

3.8.2 LSODES

Lsodes is a version of the popular BDF code Lsode which exploits the sparsity in the Jacobian matrix by calling linear algebra routines from the Yale Sparse Matrix Package [36, 35]. It is obvious that Vode and Lsodes are closely related. For our application an important difference is that Vode uses a dedicated sparsity technique, whereas Lsodes uses the general Yale package, which is less efficient, in general. Lsode and Lsodes are often used to solve atmospheric chemical kinetics equations (see e.g. [77]). The code was applied with its user parameter setting MF = 121, i.e. analytical Jacobian with an inner estimation of the sparsity structure. See [39] for other specific parameter settings.

3.8.3 SDIRK4

This solver has been borrowed from Hairer and Wanner [47] where a full description along with numerical results can be found. It is based on a 4-th order, diagonally implicit Runge-Kutta method using five stages. Because this solver is of one-step type, it allows a fast increase in step size after a restart. For atmospheric chemistry applications this is an obvious advantage. We have only modified it for the treatment of sparsity as described in Section 3.8.1. Hence all strategies were unaltered. See [39] for specific parameter settings.

3.8.4 RODAS

This solver has also been borrowed from Hairer and Wanner [47]. It is based on a 4-th order, Runge-Kutta-Rosenbrock method using six stages. This solver is also of one-step type and hence shares the advantage of a fast increase in step

size after a restart with Sdirk4. The code has been modified for the treatment of sparsity as described in Section 3.8.1. All strategies were unaltered. See [39] for specific parameter settings.

3.8.5 ROS4

This Rosenbrock solver is also taken from (Hairer et al., 1991). It implements a number of 4-stage 4(3) pairs which all require four derivative evaluations and four backsolves. Hence, per step Ros4 is somewhat cheaper than Rodas. However, in (Hairer et al., 1991) a comparison is presented favouring Rodas, which is attributed to the stiff accuracy property (the methods of Ros4 are not stiffly accurate). We have tested its L-stable version (see Table 7.2, (Hairer et al., 1991)) and found that generally its performance was very close to that of Ros3 and Rodas3. We therefore decided to omit presenting results for Ros4.

3.8.6 SEULEX

The solver Seulex is also taken from (Hairer et al., 1991). It bears a relationship with the Rosenbrock solvers, as it builds up a solution from the (non-autonomous) linearly implicit Euler method, i.e., $y_{n+1} = y_n + (I - hJ)^{-1} hf(t_n, y_n)$, by Richardson extrapolation. The use of this Euler method in an extrapolation code for stiff ODEs was first suggested in Deuflhard (1985) A rule of thumb is that the virtue of extrapolation manifests itself most clearly when high accuracy is required (see also (Hairer et al., 1991)). We have included Seulex in our benchmarking as the extrapolation approach is mentioned by Zlatev [97] (see Section 3.4.3) as a viable one for atmospheric ODE problems, although no results seem to have been reported

yet. The same sparse linear algebra as used for the other solvers was implemented. The extrapolation sequence defined by iwork(4) = 4 was used. This sequence was found to work well for our application. Other settings are given default values.

CHAPTER 4
BENCHMARK PROBLEMS AND NUMERICAL RESULTS

4.1 Introduction

This chapter presents a comprehensive set of seven benchmark problems based on chemical mechanisms used in actual applications. This set consists of seven problems based on three tropospheric gas-phase chemistry schemes, namely a small, a medium and a large scheme (used to simulate both a rural and an urban scenario). one stratospheric scheme coming from NASA, and one hybrid gas-liquid phase scheme from cloud modeling which we obtained from Matthijsen [60]. Because of the diverse applications, these chemical schemes constitute a representative test set for evaluating and comparing numerical solvers.

Numerical comparisons are performed between several dedicated explicit and several implicit solvers. For the clarity of presentation the exposition of results is divided into two parts: the first one focuses on explicit solvers, while the second one focuses on linearly-implicit methods. Several solvers are present in both parts, making the link between them.

The implicit solvers use sparse matrix techniques to economize on the numerical linear algebra overhead. As a result they are often more efficient than the dedicated explicit ones, particularly when approximately two or more figures of accuracy are required. The results presented may constitute a guide for atmospheric modelers to select a suitable integrator based on the type and dimension of their chemical mechanism and on the desired level of accuracy.

All experiments discussed in this paper were carried out on a single processor workstation and concern box-model tests. We emphasize that promising solvers should also be compared in actual 3D transport applications where the issues of memory use, vectorization [54, 92] and parallelization are of great practical importance. Code changes connected with vectorization and/or parallelization for a particular architecture can result in CPU time decreases of orders of magnitude.

To enable interested readers to further extend this benchmark comparison using their own solvers, as well as to extend the problem set with other challenging example problems from atmospheric chemistry, all the software we have used for the problems and the solvers has been put on a ftp-site (see [39] and the instructions therein).

4.2 The benchmark problems

The seven test problems are based on a set of five chemical schemes which are presently being used in various studies. Four of them describe gas-phase, and one describes gas-liquid phase chemistry. All are fully documented elsewhere. Before briefly describing each problem, several general remarks are in order:

- All the test problems were uniformly coded in FORTRAN using the KPP symbolic preprocessor [25]. This uniformity is important for a meaningful intercomparison, since part of the algorithms need the derivative function in production-destruction form, part need it in the standard form, and some of them need an analytical Jacobian. None of the solvers was favoured/inhibited by a cheaper/ more expensive implementation of these functions. FORTRAN code defining the test problems can be obtained from [39].

- All problems were run for five days. This time interval is sufficiently large for taking into account several diurnal cycles originating from the photochemical reactions. For all models the unit of components of dy/dt used in the numerical tests is number of molecules/cm^3/second.

- The tropospheric gas-phase problems are based on three different chemical schemes. These are the 15-species EUSMOG scheme, the 32-species CBM-IV scheme, and the 84-species mechanism implemented in the STEM-II model. This mechanism will be referred to as AL. All three are used in present applications and are representative of those being used in the atmospheric chemistry models. Varying the size of the mechanism is important since both implicit and explicit solvers are considered for this benchmark. The stratospheric gas-phase problem contains 34 species and the tropospheric gas-liquid phase one 65.

- The same urban and rural scenarios are simulated with CBM-IV and with AL. Although the chemical conditions are identical and the calculated results very close, the performances of the numerical solvers depend on the chemical mechanism used. We will make this point later in the paper.

- An important issue in our numerical comparison is the use of a sparse matrix technique [75] to economize on the linear algebra costs which the stiff solvers spend in the modified Newton iteration. As a measure of these costs, we give in Table 4.1 the number of nonzero elements in the Jacobian matrix, as well as the number of nonzero entries in the Newton matrix after the LU factorization. The ratio between the number of nonzeroes in the Jacobian matrix and the square of the dimension gives an indication to which extent a

Table 4.1. The dimension of the test problems, the number of nonzeroes in the Jacobian matrix, and the number of nonzeroes in the Newton matrix after the LU factorization. The difference between the numbers in the third and second row is the fill-in.

Problem	A	B, C	D, E	F	G
Dimension	15	32	84	34	65
Jacobian	57	276	674	246	506
Factorized	57	300	768	280	629

sparse matrix solution may improve the timing compared to a standard dense matrix solution. If this ratio is small, say less than about 1/4, and a reordering of the species exists which gives rise to a small fill-in after the factorization, then a good sparse solution technique will be significantly more efficient than the standard dense solution. The table shows that for our test problems both the ratio for the Jacobian and the resulting fill-in are quite small. The sparse matrix technique we have used is based on a diagonal Markowitz criterion (see Section 3.5 for some more details). Lest we miss the obvious, for problems of a small dimension for which the dense matrix numerical algebra costs are not dominating, the gain in using a more efficient matrix solution will be hardly noticeable in the overall costs. This is the case for Problem A (see also [89]).

The dedicated explicit algorithms are scalarly implicit and exploit the production-loss form of the ODE system.

4.2.1 Problem A1: TMk model

The problem was borrowed from (Dentener, 1993, 1996). It describes the reduced $CH_4/CO/HO_x/NO_x$ chemistry and is used in the global dispersion model TMk (Heimann, 1983). It consists of 36 reactions between 18 species of which 2 were held fixed, namely H_2O and O_2. Since new values of the photolysis rates are available every 40 minutes, we split accordingly the five day period (see Section 5 for more details). The simulated conditions correspond to a polluted air parcel in summer time, at 45 degrees north latitude and at ground level (pressure = 1000 mbar). We have included emissions of NO at a constant level of 10^6 mlc/cm^3/s. More information about this model can be found in (Dentener, 1993). We note that for this small problem (17 components) the exploitation of sparsity results in limited benefits. The Jacobian matrix has 90 nonzero entries and 93 after the factorization.

4.2.2 Problem A2: EUSMOG model

This problem is borrowed from a model which is currently implemented and tested at the CWI in a Dutch smog prediction code in the framework of the project EUSMOG [85, 86]. The problem is a highly parameterized version of the EMEP MSC-W ozone chemistry scheme [82]. It consists of 15 reactions between 15 species and is extensively described in [85]. It has been used before in the comparisons reported in [89], where it has also been documented. Information about the eigenvalues can be found in Table 4.2.

Table 4.2. Distribution of real part of the spectrum of the Jacobian for EU-SMOG problem `A`.

Problem	`A`
Species i	$\mathrm{Re}\,(\lambda_i)$
OH	-9
All other $\in$	$[-4 \cdot 10^{-4}, \approx 0]$

4.2.3 Problems `B` and `C`: CBM-IV model

These problems are based on the Carbon Bond Mechanism IV (CBM-IV) [40], consisting of 32 chemical species involved in 70 thermal and 11 photolytic reactions. The concentration of H_2O was held fixed throughout simulation. The CBM-IV mechanism was designed for the numerical simulation of chemical processes in urban and in regional scale models. Test problem `B` describes an urban scenario and simulates a heavily polluted atmosphere. The initial concentrations and the levels of hourly emissions follow those described in [77]. This helps to relate our results to those presented in the above mentioned paper. Test problem `C` describes a rural scenario and simulates a clean atmosphere. It follows the IPCC Chemistry Intercomparison Study, third Bio scenario (see [70]). The values of initial concentrations and the values of hourly emissions are given in Table 4.5. The emission was released in equal quantities at the beginning of each time interval. Information about the stiffness of the models in terms of the eigenvalues of the Jacobian is presented in Table 4.3.

Table 4.3. Distribution of real part of the spectrum of the Jacobian for CBM-IV problems B and C.

Problem	B (urban)	C (rural)
Species i	$\mathrm{Re}(\lambda_i)$	$\mathrm{Re}(\lambda_i)$
$O(^1D)$	$-8.11 \cdot 10^8$	$-8.11 \cdot 10^8$
$O(^3P)$	$-8.26 \cdot 10^4$	$-8.26 \cdot 10^4$
ROR	$-2.47 \cdot 10^3$	$-2.46 \cdot 10^3$
OH	-46	-3.5
TO_2	-4.27	-4.2
All other $\in$	$[-1.5, \approx 0]$	$[-0.14, \approx 0]$

4.2.4 Problems D and E: AL model

The test problems D and E are based on the largest chemical system tested here. They employ the kinetic mechanism that is presently used in the STEM-II regional-scale/transport/chemistry/removal model [14], consisting of 84 chemical species (plus 4 species whose concentrations were held fixed throughout simulation: H_2O, CO_2, O_2, H_2) involved in 142 thermal and 36 photolytic reactions. The mechanism, based on the work of Atkinson et. al. [5] and Lurmann et al. [59] can be used to study the chemistry of both highly polluted (e.g., near urban centers) and remote (e.g., marine) environments. Problem D is an urban scenario, while problem E a rural one, based on IPCC scenario 3. The simulated conditions and initial concentrations are identical to those employed in problems B and C, respectively. The values of initial concentrations, and the values of hourly emissions are given in Table 4.5. The emission was performed in equal quantities at the beginning of

Table 4.4. Distribution of real part of the spectrum of the Jacobian for AL problems `D` and `E`.

Problem	`D` (urban)	`E` (rural)
Species i	$\mathrm{Re}\,(\lambda_i)$	$\mathrm{Re}\,(\lambda_i)$
OH	-28	-1.81
CRO_2	-1.45	-1.38
CHO_2	-1.45	-1.30
$MAOO$	-1.23	-1.17
$MVKO$	-1.23	-1.17
$MCRG$	-1.23	-1.17
All other $\in$	$[-0.3, \approx 0]$	$[-0.03, \approx 0]$

each time interval. Information about the stiffness of the models in terms of the eigenvalues of the Jacobian is given in Table 4.4. Since AL does not treat explicitly $O(^1D)$ and $O(^3P)$, the large negative eigenvalues associated with these species are not present.

4.2.5 Problem `F`: stratospheric model

This test problem is based on the chemical mechanism that has been used in the NASA HSRP/AESA stratospheric models intercomparison. The initial concentrations and the values of the rate constants follow the NASA region A scenario (model available at NASA ftp site, contact Douglas E. Kinnison, kinnison1@llnl.gov) with the difference that the photolysis rates were piecewise linearly interpolated.

Table 4.5. Initial concentrations and hourly emissions for the tropospheric problems `B`, `C`, `D` and `E`. *Toluene* and *Xylene*, which are treated independently in CBM-IV, are lumped as *Aromatics* in the AL model.

Problem	`B, D` (urban)		`C, E` (rural)	
Species	Initial (ppb)	Emission (ppb/hour)	Initial (ppb)	Emission (ppb/hour)
NO	50	1	0.1	0.01
NO_2	20	0.2	0.1	0.01
$HONO$	1	—	30	—
O_3	100	—	0	—
H_2O_2	1	—	2	—
CO	300	2	100	—
$HCHO$	10	0.2	0	—
ALD_2	10	0.2	0	—
PAN	1	—	0	—
Alkans	50	2	0	—
Alkens	10	1	0	—
Ethene	10	0.2	0	—
Aromatics (AL)	20	0.4	0	—
Toluene (CBM-IV)	10	0.2	0	—
Xylene (CBM-IV)	10	0.2	0	—
Isoprene	10	1	1	0.1

Table 4.6. Physical conditions for the tropospheric problems `B`, `C`, `D` and `E`.

Relative humidity	80%
Temperature	288.15 K
Altitude	0 km
Pressure	1013.25 mbar
Air density	$2.55 \cdot 10^{19}$ mlc/cm^3

There are 34 species (plus 6 species whose concentrations were held fixed throughout simulation: H_2O, CO, O_2, H_2, N_2, CH_4) involved in 84 thermal and 25 photolytic reactions. The values of initial concentrations for the most important species are given in Table 4.7. No emissions have been prescribed. For a complete description of the problem we refer to the NASA ftp site. Information about the stiffness of the problem in terms of the eigenvalues of the Jacobian is given in Table 4.8.

4.2.6 Problem `G`: aqueous model

From the numerical point of view, this test problem is the most difficult one. It contains 65 species (plus 5 species whose concentrations were held fixed: H_2O (vapour), H_2O (drops), CH_4, O_2, CO_2(aq)) involved in 77 thermal and 11 photolytic gas-phase chemical reactions, 39 liquid-phase chemical reactions and 39 gas-liquid mass transfer reactions. The gas-phase mechanism is based on CBM-IV, while the liquid-phase mechanism is based on a chemical scheme the authors obtained from [60]. Initial concentrations are given in Table 4.10. Information about the stiffness

Table 4.7. Initial concentrations and physical conditions for the stratospheric problem `F`.

Species	Initial (ppb)	Species	Initial (ppb)
O	8.15	O_3	656
NO	10.7	NO_2	2.75
HNO_3	0.35	H_2O	6100
OH	0.2	HO_2	0.14
H_2	370	CH_4	490
CO	20	$C\ell O$	1
$HC\ell$	2.15	$HOC\ell$	0.22
Temperature	241.43 K		
Altitude	40 km		
Latitude	65^o N		
Pressure	2.7 mbar		
Air density	$8.12 \cdot 10^{16}$ mlc/cm^3		

Table 4.8. Distribution of real part of the spectrum of the Jacobian for the stratospheric problem `F`.

Problem	`F`
Species i	$\mathrm{Re}\,(\lambda_i)$
$O(^1D)$	$-2.53 \cdot 10^6$
HCO	$-1.06 \cdot 10^5$
$C\ell OO$	$-1.70 \cdot 10^4$
CH_3	$-9.98 \cdot 10^2$
H	$-1.17 \cdot 10^2$
CH_3O	-16
$C\ell$	-4.5
$O(^3P)$	-1.37
All other $\in$	$[-0.5, \approx 0]$

Table 4.9. Distribution of real part of the spectrum of the Jacobian for the aqueous problem G.

Problem	G
Species i	$\mathrm{Re}\,(\lambda_i)$
$HNO_{3\ (aq)}$	$-2.2 \cdot 10^9$
$O(^1D)$	$-8.1 \cdot 10^8$
$OH^-_{(aq)}$	$-1.8 \cdot 10^7$
$SO^{2-}_{3\ (aq)}$	$-1.3 \cdot 10^7$
$HCOOH_{(aq)}$, $SO_{2\ (aq)}$,	-1.25e+7, -3.65e+6, -1e+6,
$HCOO^-_{(aq)}$, $NH_{3\ (aq)}$, $O^{2-}_{(aq)}$,	-4.2e+5, -1.3e+5, -8.2e+4,
$HO_{2\ (aq)}$, OH, O, $H^+_{(aq)}$,	-2e+4, -9e+3, -2.46e+3,
$HSO^-_{3\ (aq)}$,$OH_{(aq)}$, CH_3O,	-2.2e+3, -2e+2, -1.5e+2,
$NO^-_{3\ (aq)}$,ROR,$NO_{3\ (aq)}$	-90, -30,-15
All other $\in$	$[-10, \approx 0]$

of the problem in terms of the eigenvalues of the Jacobian is given in Table 4.9. Of numerical interest is the fact that only for the four most negative eigenvalues does the relation $\lambda_i \approx -L_i$ hold, while (different from the gas-phase-only test problems) the number of stiff eigenvalues is much larger. This is due to the rapid gas-liquid phase kinetics. In Table 4.9 we have listed these large negative eigenvalues and the species with large L_i, but without making a one-to-one correspondence between them.

Table 4.10. Initial concentrations and emissions for the aqueous problem `G`.

Species	Initial (ppb)	Emission (ppb/hour)	Species	Initial (ppb)
NO	0.2	0.01	O_3	60
NO_2	0.5	0.01	CO	200
H_2O_2	1.5	—	$HCHO$	1.0
PAR	1.2	—	$Ethene$	$2.4 \cdot 10^{-2}$
$ISOP$	1.0	0.05	$Xylene$	$2 \cdot 10^{-2}$
SO_2	$3.3 \cdot 10^{-2}$	—	$Toluene$	$3 \cdot 10^{-2}$
Temperature			288.15 K	
Altitude			0 km	
Relative humidity			80 %	
Liquid water			0.0436 %	
Air density			$2.50 \cdot 10^{19}$ mlc/cm^3	

4.3 Setup of experiments

- The variable step size control requires the choice of a relative error tolerance *rtol* and an absolute error tolerance *atol*. The choices made for *atol* and *rtol* differ per solver and are not specified here. Noteworthy is that at certain times the concentrations of some species (e.g. radicals) can become smaller than 1.0 mlc/cm^3. Because these values are insignificant, they are ignored in the step size control.

- Often we also prescribe a minimal step size. The use of a minimal step size improves efficiency since extremely small steps can be selected by a variable step size selection scheme. Atmospheric chemistry problems containing photochemical reactions can possess time constants as small as 10^{-8} to 10^{-9} seconds and step size selection mechanisms do signal these. However, these extremely small step sizes are redundant because the minimal time constants of importance for photochemical chemistry models are of the order of seconds and species with a time constant truly smaller almost instantaneously get in their (solution dependent) steady state when they are perturbed. On the other hand, too large lower bounds for the step size can cause convergence and loss of accuracy problems to the numerical solvers.

- All sparse implicit solvers work with the analytical Jacobian matrix and can be shown to mimic conservation rules which exist for the ODE system. This does not hold for all of the explicit solvers.

- A numerical comparison should focus on modest accuracies, say relative accuracies near 1%. Higher accuracy levels are redundant for the actual practice of air pollution modelling.

The solvers are tested as if they were used in an operator splitting environment. This means that we split the total integration interval into N subintervals of length Δt. For each subinterval we then restart the integration of the stiff solvers. For all test problems the length of the subintervals equals $\Delta t = 3600$ sec. leading to $N = 120$ new starts over the 5 day period. The 3600 sec. subinterval means that we reckon with a 1800 sec. transport time step, assuming a Strang splitting symmetrized around the chemistry step. It should be stressed that the choice of the subinterval length is important for a box-model comparison, since this length determines the number of restarts. Restarts are expensive for implicit solvers using a Newton type iteration due to the linear algebra overhead. This holds in particular for the multistep solvers Vode and Lsodes. For the one-step implicit solvers Rodas and Sdirk4 the penalty is less, since they are able to enlarge the step size after restart considerably faster. Generally, for any explicit solver the penalty is also less because of the absence of linear algebra overhead.

All test problems contain photochemical reactions. This means that part of the reaction constants are determined by time of the day dependent photolysis rates which undergo a rapid change at sunrise and sunset. This change gives rise to large variations in concentration values and normally force a solver to adjust the step size. In Problem `A` the photolysis rates are given by a C^0 function, while in Problems `B, C, D, E, F` and `G` by a C^1 function which are zero at nightly periods.

All the runs were made in double precision on a HP-UX 935 A workstation with a CPU clock frequency of 125 megaherz and 160 Mbytes RAM. The numerical results for all test problems are compared to a very accurate reference solution computed by the code Radau5 from [47] with the very tight tolerances $rtol = 10^{-12}$, $atol_i = 10^{-15}\eta_i$, where η_i estimates the magnitude of the concentration of

species i in unit mlc/cm^3. Our measure of accuracy is based on a modified root mean square norm of the relative error. With the reference solution y and the numerical solution $\hat{y}$ available at $\{t_n = t_0 + n\Delta t,\ 0 \le n \le N\}$, we first compute for each species k

$$ER_k = \sqrt{\frac{1}{|\mathcal{J}_k|} \cdot \sum_{n \in \mathcal{J}_k} \left| \frac{y_k(t_n) - \hat{y}_k(t_n)}{y_k(t_n)} \right|^2}\ , \quad \mathcal{J}_k = \{n : |y_k(t_n)| \ge a\}\ . \tag{4.1}$$

Hence we compute specieswise a temporal error measure, which we then represent in the plots in two ways. Through the number of significant digits for the average of ER_k, defined by

$$SDA_1 = -\log_{10}\left(\frac{1}{m}\sum_{k=1}^{m} ER_k\right), \tag{4.2}$$

and the number of significant digits for the maximum of ER_k, defined by

$$SDA_\infty = -\log_{10}(\max_k\ ER_k). \tag{4.3}$$

The threshold factor a used here is given the value $a = 1.0$ mlc/cm^3. If the set $\mathcal{J}_k$ is empty, the value of ER_k is neglected. The purpose of considering the above defined error measure instead of the root mean square norm ($a = 0$) is to avoid chemically meaningless large relative errors for concentration values smaller than 1.0 mlc/cm^3. It is instructive to present the accuracy of the computed results using averaged and maximal errors taken over the number of species. Finally, in the work-precision diagrams of the following section, efficiency is measured by CPU time. This is of course dependent on the computer architecture used. However, the relative magnitudes shown here should be applicable to scalar computers in general. We thus plot the SDA values against the measured CPU times on a logarithmic scale in unit seconds. Observe that $SDA = 2$ means 1% accuracy in the error measure used. In discussing the results presented in the next section we focus on this accuracy level. The different data points in the plots for the same solvers are

associated to runs carried out for different relative tolerances.

4.3.1 Splitting interval

The tests are intended to simulate an operator splitting environment. In air quality models, most often a symmetric splitting is used, for example:

$$T_x^{\Delta t} \circ T_y^{\Delta t} \circ T_z^{\Delta t} \circ C^{2\Delta t} \circ T_z^{\Delta t} \circ T_y^{\Delta t} \circ T_x^{\Delta t}$$

where $T_j^{\Delta t}$ stands for transport in direction j for a time interval Δt and C is the chemistry solution operator. Thus the restart time or splitting interval equals $2\Delta t$. For Problem `A` we have chosen a restart time of 40 min. and for all other problems 60 min. A restart time of 60 min. corresponds to a transport step size of 30 minutes due to the symmetry of splitting. For the two urban scenarios (test problems `B` and `D`) additional simulations were carried out with a restart each 15 minutes; this corresponds to a splitting interval of 7.5 minutes for the transport scheme.

4.3.2 Emissions

For all problems except the stratospheric problem `F`, emissions are prescribed. In the experiments we have computed emissions at the beginning of each split interval, simulating a form of operator splitting. This means that species solutions for which emissions occur, are made discontinuous so that at any restart initial transients occur. We thus simulate, to some extent, what happens in a true transport computation where one also encounters initial transients at any restart. As a rule, strong initial transients make the nonlinear stiff problems harder to solve. If we would compute the emissions along with the integration over the split intervals,

then all species solutions remain continuous at restart.

4.3.3 Steering parameters

For variable step size solvers the important steering parameters are h_{start}, h_{min} and the local error tolerances $atol, rtol$. A user-specified h_{min} is important. Without a prescribed minimum, step sizes can result as small as the shortest time constants, sometimes even $\approx 10^{-8}$ to 10^{-9} sec. Step size values close to these extremely short time constants are redundant, since the minimal time constants of importance for photochemical models lie between 1 sec. and 1 min., approximately. On the temporal scale of interest, species with a smaller time constant quickly reach their (solution dependent) steady state when they are perturbed. On the other hand, most solvers require a relatively small step size at the start to resolve the initial transients.

Through trial and error we have prescribed the following values for h_{min} and h_{start} which are imposed for all solvers (except EBI): for the tropospheric Problems `A-E`, $h_{\text{min}} = 0.1$ sec and $h_{\text{start}} = 60$ sec; for the stratospheric Problem `F`, $h_{\text{min}} = h_{\text{start}} = 0.001$ sec; and for the aqueous Problem `G`, $h_{\text{min}} = h_{\text{start}} = 0.0001$ sec. These values concern the 1 hour restart time. For the tropospheric problems `B,D` with the 15 min. restart time, we have taken $h_{\text{start}} = h_{\text{min}} = 0.1$ sec.

For all problems and all solvers except EBI, we have prescribed the absolute tolerance value $atol = 0.01$ mlc/cm^3 along with a sequence of relative tolerance values $rtol$ such that effectively relative local error control is imposed. For a given method , the different data points in the accuracy-efficiency plots correspond to this

sequence. The values used are

$rtol = 1.0, 3.0\,10^{-1}, 1.0\,10^{-1}, 3.0\,10^{-2}, 1.0\,10^{-2}, 3.0\,10^{-3}, 1.0\,10^{-3}, 3.0\,10^{-4}, 1.0\,10^{-4}$.

Needless to mention that the actual resulting accuracies are always different from the given local tolerances. The tolerances merely govern the local error and step size control. Also note that for very large values of $rtol$, say $rtol > 0.1$, the control is very loose so that a negative number of significant digits (4.4) or even a breakdown may be the result. Note that a negative number of significant digits (4.4) means relative errors greater than 100%. The Rosenbrock solvers Rodas, Ros3 and the BDF solver Vode showed breakdowns more often, Rodas3 only for $rtol = 1$, while Seulex never failed and always returned a positive SDA. Also Twostep never encountered a breakdown, only minor negative SDA values. Data points corresponding to a breakdown or a negative result, as well as points with $SDA > 4$ or with an exceptionally large CPU time have been skipped from the plots. We have used a wide range of tolerances merely for illustrative purposes.

4.3.4 Accuracy

The numerical results were compared to a very accurate reference solution (given by Radau5, $rtol = 10^{-12}$, componentwise set $atol$) using a temporal modified root mean square norm of the relative error. With the reference solution y and the numerical solution $\hat{y}$ available at $\{t_n = t_0 + n\Delta t,\ 0 \le n \le N\}$, we first compute for each species k

$$ER_k = \sqrt{\frac{1}{|\mathcal{J}_k|} \cdot \sum_{n \in \mathcal{J}_k} \left| \frac{y_k(t_n) - \hat{y}_k(t_n)}{y_k(t_n)} \right|^2},$$

where $\mathcal{J}_k = \{0 \leq n \leq N : \ y_k(t_n) \geq a\}$. This value is then represented in the plots through the number of significant digits for the maximum of ER_k, defined by

$$SDA = -\log_{10}(\max_k ER_k). \tag{4.4}$$

Note that if the set $\mathcal{J}_k$ is empty for a chosen threshold a, the value of ER_k is neglected. This threshold factor serves to eliminate chemically meaningless large relative errors for concentration values smaller than a mlc/cm^3 in the error measure. We used $a = 10^6$ mlc/cm^3 for all tropospheric problems and $a = 10^4$ mlc/cm^3 for the stratospheric one. Additional experiments performed with $a = 1$ mlc/cm^3 led to nearly the same conclusions. In all plots presented in the remainder for problems `B - F`, including those for the 15 min. restart times for `B, D`, we have used N = 120. So we always sample at the endpoint of each hour over the whole 5 days. For Problem `A` we sample at the end of every 40 min. Observe that $SDA = 2$ means 1% accuracy in the error measure used. In discussing the results in the next section we focus on this accuracy level.

4.3.5 Timing

The answer to the question of which method is "the fastest" may depend also on the machine. In order to measure the influence of the hardware on the relative performance of integrators we have performed all the numerical experiments on two completely different architectures, namely a HP-UX 935 A workstation (double precision, $\approx$ 14 digits) and a Cray C98 (scalar mode, single precision, $\approx$ 14 digits); in addition, some of the experiments were also repeated on a SGI workstation (double precision, $\approx$ 14 digits). Very similar results were found. As a consequence, in what follows only the HP work-precision diagrams are presented. We plot the SDA

values against efficiency, i.e., the measured CPU times on a logarithmic scale in unit seconds.

4.3.6 Reaction coefficients

In practice the rate coefficients can be implemented in two ways, either as time-continuous functions or as functions piecewise constant per split interval. The time-continuous function implementation of the thermal rate coefficients may lead to a large number of exponential function evaluations per time step, which are very costly. For example, with Rosenbrock methods we observed that these calculations can be as expensive as the sparse matrix factorization. Since for the actual practice true time dependency seems redundant, we have used piecewise constant rate coefficients per operator-split subinterval (temperature and solar angle frozen using values halfway). Observe that in (Sandu et al., 1996a) time-continuous values were used. We did again a number of tests with time-continuous values in the current investigation but observed no notable differences in the relative performances of the solvers.

4.4 Results - part 1

4.4.1 Problem A: EUSMOG model

The work-precision diagram is given in Figure 4.1. The implicit integrators and the two versions of Twostep perform very well. For an accuracy of two digits, sparse Vode appears to be the fastest. The good performance of implicit integrators is due in part to the small dimension of the system which means less work with

the linear algebra. Note that the Jacobi and Gauss-Seidel versions of Twostep have similar performance for this test problem (see also [89]). It is clear that the simplest Qssa solver, ET and Chemeq are the slowest among the tested routines. It is also worth noting that the Qssa performance greatly improves by extrapolation.

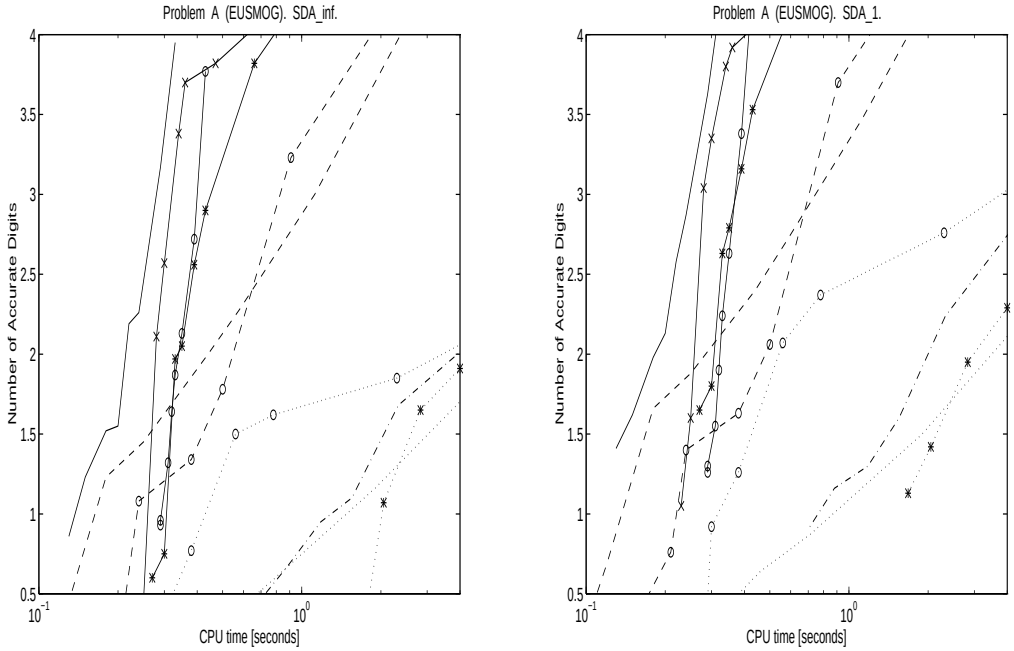

Figure 4.1. Work-precision diagram for test problem `A` (EUSMOG): Twostep Seidel (dashed), Twostep Jacobi (dashed with "*o*"), Qssa (dots), Extrapolated Qssa (dots with "*o*"), ET (dots with "∗"), Chemeq (dash-dots), Sparse Vode (solid), Sparse Sdirk4 (solid with "∗"), Sparse Rodas (solid with "x") and Lsodes (solid with "*o*").

4.4.2 Problems `B` and `C`: CBM-IV model

In Figure 4.2 the numerical results for test problems `B` and `C` are presented. For obtaining two accurate digits, Rodas appears to be the fastest, followed by

Sdirk4, Vode, Lsodes and Twostep Seidel. The latter is the best when less accuracy is demanded, while the implicit codes are preferable for higher precisions. This is due to the fact that they use higher order formulas (nicely represented by the higher slopes of their diagrams). An interesting remark is that the slope of the Twostep Jacobi diagram decreases when higher accuracies are required. This is due to an imposed minimal step size of 0.01 sec., which makes the convergence of the Jacobi iteration very slow on part of the time interval. Since Twostep was used with a fixed number of only two iterations, this is directly reflected in the accuracy of the solution. A minimal step size larger than 0.1 sec. creates similar problems in Twostep Seidel. The gap between the two Twostep diagrams is due primarily to the different values of the minimal step size used: 0.01 sec. for Jacobi and 0.1 sec. for Seidel. The explicit solver ET is not competitive at all. Its work-precision diagram is situated to the far right of Figure 4.2. Among the other integrators, Qssa is clearly the slowest, but by extrapolation it gains about one accurate digit for the same CPU time. In both scenarios Chemeq performs better than the plain Qssa scheme but worse than Extrapolated Qssa. As expected, the Lsodes and sparse Vode diagrams are similar, except that the Lsodes one is shifted to the right. Working with predefined sparsity data structures, Vode is consistently faster than the general purpose Lsodes. However, despite its generality, for test problems `B` and `C` Lsodes performs very well.

4.4.3 Problems `D` and `E`: AL model

The results are reported in Figure 4.3. It is interesting to compare code performances to those obtained for test problems `B` and `C`, since the same urban

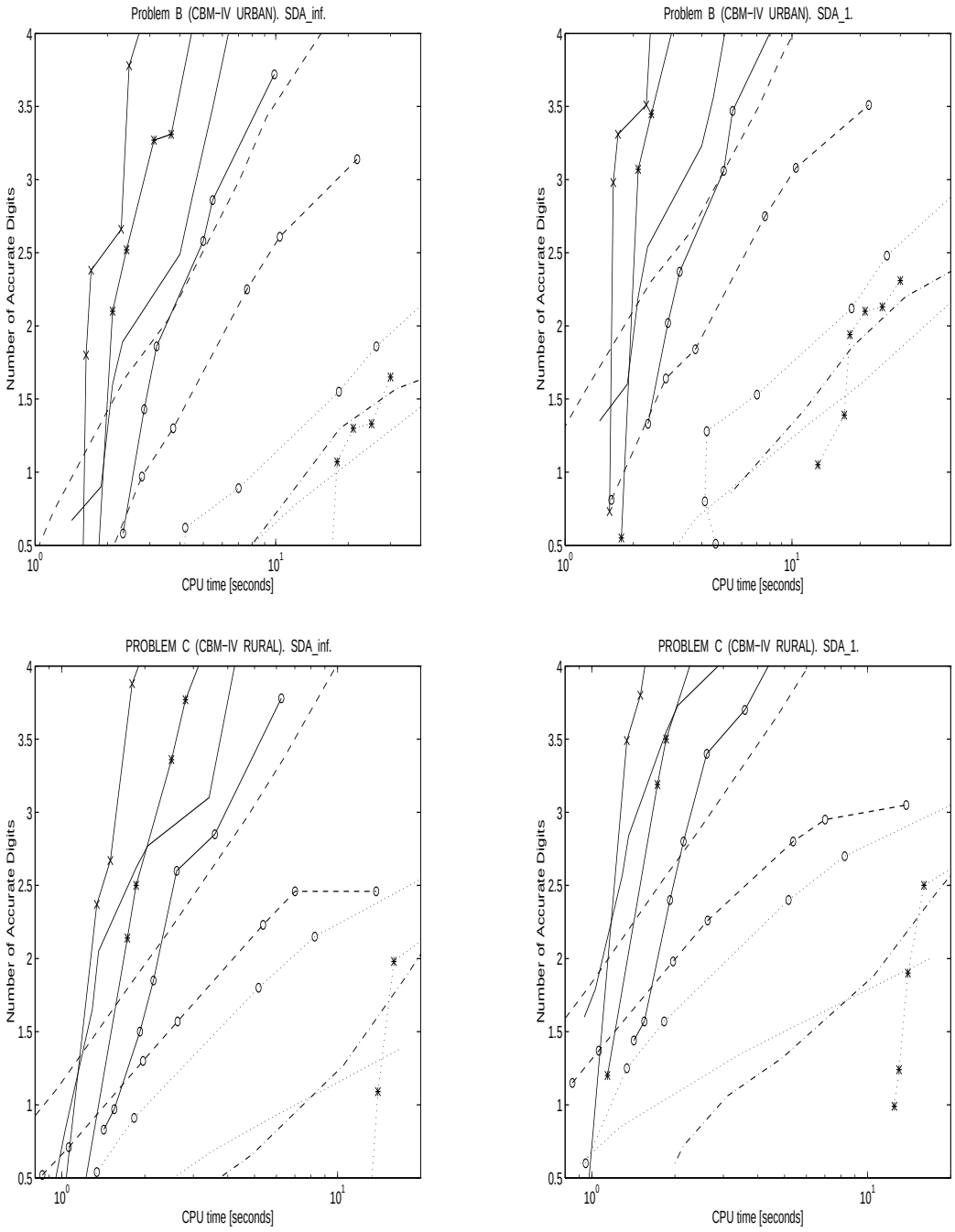

Figure 4.2. Work-precision diagram for test problems `B` and `C` (CBM-IV): The upper pair of diagrams correspond to problem `B` and the lower pair to problem `C`. Twostep Seidel (dashed), Twostep Jacobi (dashed with "*o*"), Qssa (dots), Extrapolated Qssa (dots with "*o*"), ET (dots with "∗") Chemeq (dash-dots), Sparse Vode (solid), Sparse Sdirk4 (solid with "∗"), Sparse Rodas (solid with "x") and Lsodes (solid with "*o*").

and rural scenarios are simulated with both CBM-IV and AL. They differ however in the number of reactions and species, the current problems `D` and `E` being much larger. If a standard implementation of the implicit solvers was used, their linear algebra workload would have increased as m^3, with m the number of species, while for dedicated explicit integrators the workload increases linearly with m. Thus, at first sight, for sufficiently large problems use of explicit integrators seems to be preferable. Because we use a sparse linear algebra implementation, the situation becomes truly different. For the sparse implementation a rough estimation of the linear algebra workload is given by the number of nonzero elements in the Newton matrix. As seen in Table 4.1, this number increases almost linearly with m for the test mechanisms considered here. This means that even for fairly large chemical systems, sparse implicit solvers may very well remain competitive. Our test results shown in Figure 4.3 clearly illustrate this. For both problems all sparse implicit solvers outperform the dedicated explicit ones, with the exception of Twostep Seidel. The gap between this code and its Jacobi version is again due to the fact that Seidel iterations allow the use of larger values for the minimal step size. Still, this imposed minimal step size causes convergence problems at part of the time interval, which explains the curious slopes of Twostep in the range of high accuracies for the more difficult urban scenario. Noteworthy is that the one-step solver Rodas is the fastest in the 1% error region. For the more difficult urban problem, the two one-step solvers Rodas and Sdirk4 are always faster than their BDF counterparts Vode and Lsodes. The explicit solver ET fails to integrate problem `D` and is among the slowest for problem `E`. The Chemeq diagram is in between that for Qssa and Extrapolated Qssa. The latter clearly performs better in the rural cases than in the urban ones for both CBM-IV and AL mechanisms.

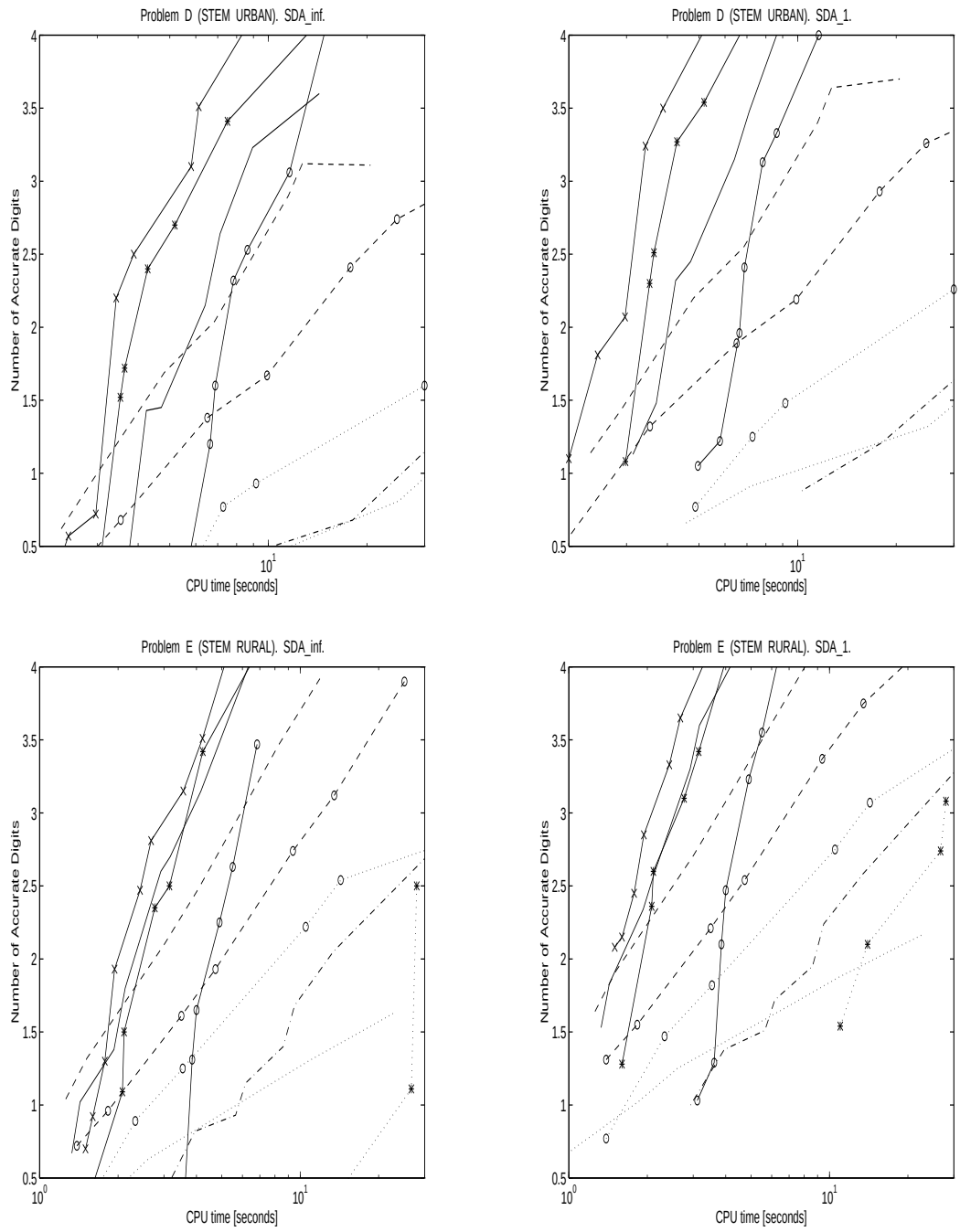

Figure 4.3. Work-precision diagram for test problems D and E (AL): The upper pair of diagrams correspond to test problem D, while lower pair to test problem E. Twostep Seidel (dashed), Twostep Jacobi (dashed with "*o*"), Qssa (dots), Extrapolated Qssa (dots with "*o*"), ET (dots with "∗"), Chemeq (dash-dots), Sparse Vode (solid), Sparse Sdirk4 (solid with "∗"), Sparse Rodas (solid with "x") and Lsodes (solid with "*o*").

4.4.4 Problem F: stratospheric model

This problem has about the same dimension as the two CBM-IV problems, but the integrators perform quite differently relative to each other as shown in Figure 4.4. The implicit integrators work best. There is not much difference between their performances, but there is a large gap between them and the explicit codes, with the largest for ET, Chemeq and the two Qssa solvers. In particular, for this problem the implicit codes require less CPU time than for CBM-IV, whereas for Twostep the amount of CPU time almost remains equal. From an additional investigation we learned that the absence of emissions in the stratospheric problem must play a role here. Without emissions, concentration values vary less in between the one hour subintervals. The implicit solvers enjoy this, since as a rule they allow larger step sizes than Twostep. We have checked this observation by removing the emissions in the CBM-IV model. In this case the implicit solvers also integrate much faster, the CPU timings being then similar to those obtained for the stratospheric problem.

4.4.5 Problem G: aqueous model

As we have previously seen, this test problem has a large number of stiff eigenvalues, most of which cannot be associated to certain short lived species. This makes the test problem G numerically challenging in the sense that the explicit methods fail. Numerical results confirm that the explicit formulas described in Section 3 cannot solve this problem with a reasonable efficiency. We tested each routine with the loosest restrictions on step size and tolerance for which the numerical results

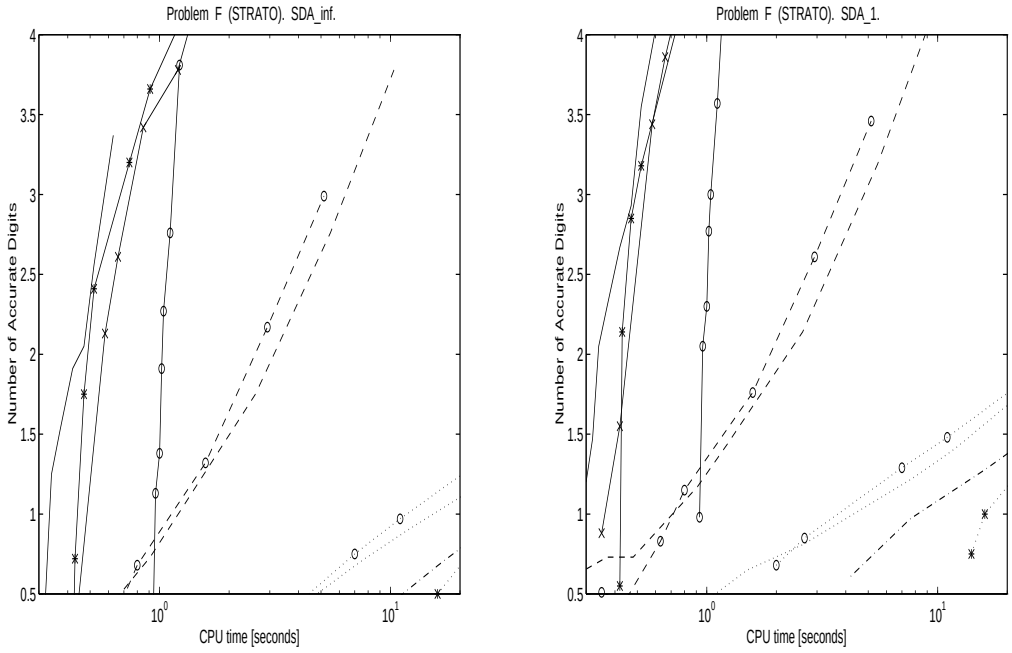

Figure 4.4. Work-precision diagram for test problem F (STRATO): Twostep Seidel (dashed), Twostep Jacobi (dashed with "*o*"), Qssa (dots), Extrapolated Qssa (dots with "*o*"), ET (dots with "∗"), Chemeq (dash-dots), Sparse Vode (solid), Sparse Sdirk4 (solid with "∗"), Sparse Rodas (solid with "x") and Lsodes (solid with "*o*").

were still meaningfull. See the codes [39] for the exact setting of the parameters. The timings obtained for integrating the first 10 seconds (or 1 hour) of evolution are given in Table 4.11. When looking at the results, keep in mind the fact that all the implicit solvers integrate the test problem along the five days interval in less than 10 seconds CPU time. Among the dedicated integrators, Twostep Seidel performs best. However, even this code selects very small step sizes, which is not the case when a fully implicit implementation of the BDF2 formula is used (Vode with restricted maximal order). The behaviour of the dedicated integrators is typical for standard explicit formulas applied to general stiff problems. To make the terms of comparison more clear, we estimated the CPU time that would be needed to complete the five days simulation. This estimated time is given in the last column of Table 4.11. The explicit solver ET gives a floating point exception on this test problem and is not included in the table.

Some effort has been made to optimize the performance of the implicit solvers for this test problem. Rodas and Sdirk4 were used with a minimal stepsize of 10^{-3} sec., while for Vode a minimum of 10^{-8} sec. appeared to be best. No minimal step size was prescribed for Lsodes. For sparse Vode the maximal order was restricted to 3. Although the original Vode had no problems integrating this model, the sparse version selected tiny step sizes when a maximal order of 4 or 5 was used. The work-precision diagram for the implicit schemes is given in Figure 4.5, which shows that Rodas, Vode and Sdirk4 perform equally well.

Table 4.11. The timing of dedicated integrators on test problem G. The last column represents the estimated CPU time needed to complete the five days simulation.

Integrator	Simulated interval	CPU time	Estimated total CPU
Qssa	10 sec	19 min	350 days
Extrap Qssa	10 sec	33 min	650 days
Chemeq	10 sec	0.5 min	15 days
Twostep J	10 sec	9 min	270 days
Twostep S	3600 sec	5.3 min	10.3 hours

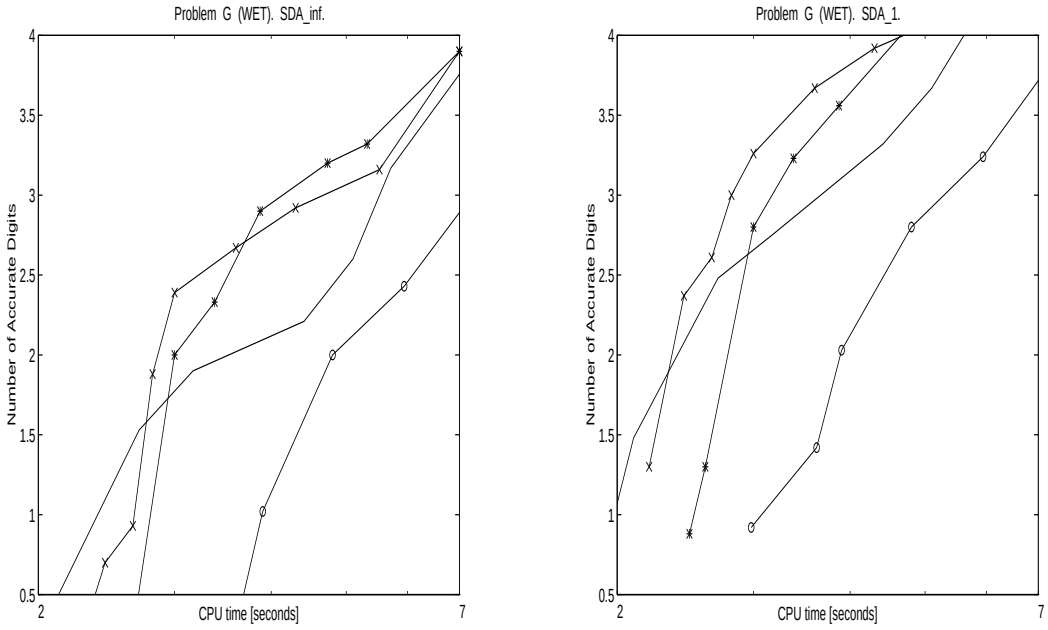

Figure 4.5. Work-precision diagram for test problem G (Aqueous): Sparse (solid), Sparse Sdirk4 (solid with "∗"), Sparse Rodas (solid with "x") and Lsodes (solid with "*o*").

4.5 Results - part 2

4.5.1 Problem A: TMk model

The work precision diagram is given in Figure 4.6. Results are presented for all the solvers discussed above, including EBI. The EBI results are obtained with constant step sizes of length

$$h = 40/80, 40/40, 40/20, 40/10, 40/6, 40/5, 40/4, 40/3 \,\text{min}.$$

The number of iterations within EBI was in all runs equal to 8 (cf. (Dentener, 1996)). The results show that the variable step size Rosenbrock solvers are clearly superior to all others for 1% accuracy. Seulex appears to be faster than Vode, but slower than the Rosenbrock codes. However, the gap between these solvers decreases for higher accuracies. Among the Rosenbrock codes, Rodas3 and Ros3 have similar performance in the low accuracy domain; they are followed closely by Rodas. EBI and Twostep perform reliably but cannot compete with Rodas3 over the whole accuracy range.

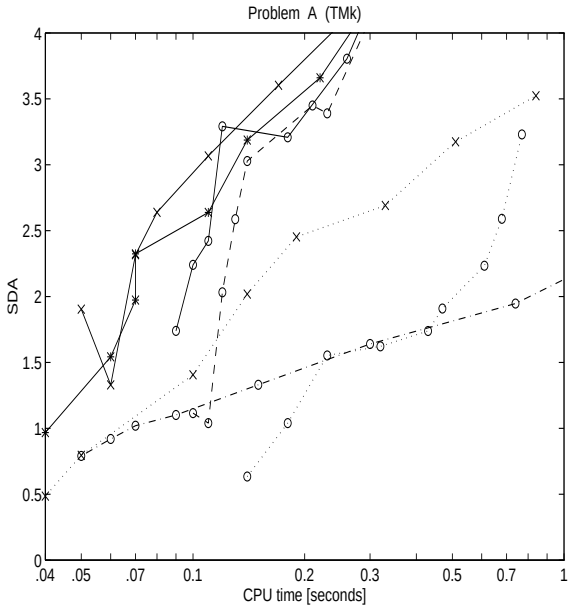

Figure 4.6. Work-precision diagram for test problem `A` (TMk): Sparse Rodas3 (solid with "∗"), Sparse Ros3 (solid with "x"), Sparse Rodas (solid with "o"), Twostep Seidel (dots with "x"), Sparse Vode (dots with "o"), Sparse Seulex (dashed with "o") and EBI (dash-dots with "o").

4.5.2 Problems `B` and `C`: CBM-IV model

In Figure 4.7 the results for test problems `B` (one hour restart time) and `C` are presented. For the rural problem all Rosenbrock solvers perform equally well, followed by Seulex, while Vode and Twostep fall behind. This also holds for the urban problem, but now a distinction exists between Rodas3,Ros3 and Seulex, Rodas. Up to about 3 digits Rodas3 and Ros3 perform best. For accuracies higher than 3 digits Rodas takes over.

The results for the urban problem with a 15 min. restart time are presented in Figure 4.9. The relative performances between the solvers remains almost the same. The main difference with the 1 hour restart time is seen in the CPU times. All integrations become roughly 3 to 4 times more expensive, showing that all solvers spend most of their time in the start-up phase. Recall that the start-up phase has become longer as we have lowered h_{start} from 60 sec. to $h_{\text{min}} = 0.1$ sec. The 60 sec. starting step size was found too large for the Rosenbrock solvers for a good performance. This indicates that they must spent quite an effort in resolving the initial transients. However, they remain competitive, in particular Ros3 and Rodas3. The figure also contains results for the most simple Qssa solver we previously applied in (Sandu et al., 1996a). However, this solver again lags behind to all others.

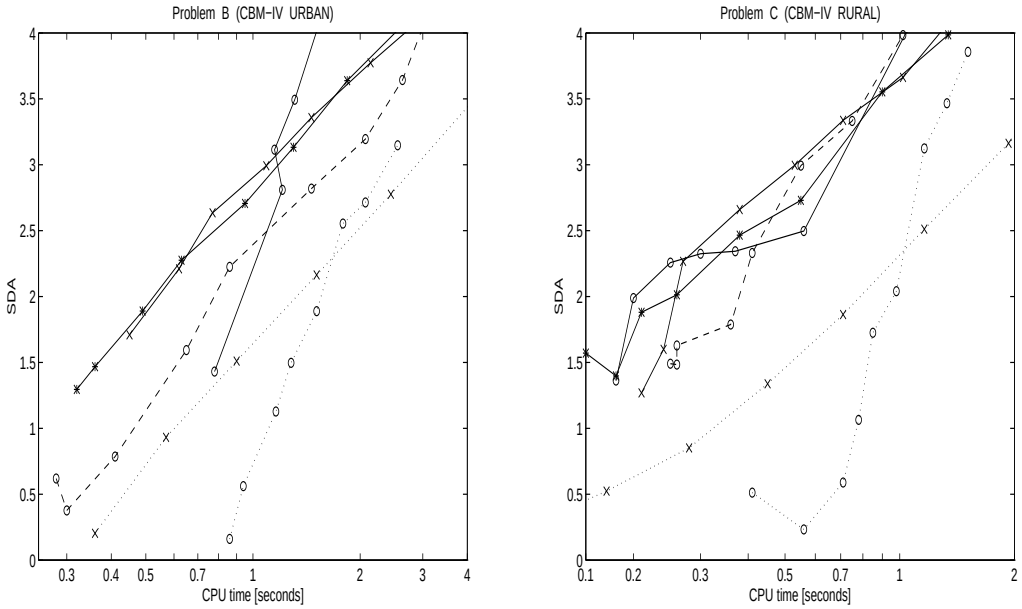

Figure 4.7. Work-precision diagram for test problems `B` and `C` (CBM-IV): Sparse Rodas3 (solid with "∗"), Sparse Ros3 (solid with "x"), Sparse Rodas (solid with "o"), Twostep Seidel (dots with "x"), Sparse Vode (dots with "o") and Sparse Seulex (dashed with "o").

4.5.3 Problems `D` and `E`: AL model

For problems `D` and `E` with 1 hour restart time the results are given in Figure 4.8. It is interesting to compare code performances to those obtained for the CBM-IV model since the same urban and rural scenario's are simulated. They differ, however, in the number of species and reactions, the AL model being considerably larger. For the urban problem Rodas3 and Ros3 are again the fastest, up to 3 digits, while for higher accuracies Rodas becomes better. Seulex now performs somewhat

less than for the CBM-IV model, whereas Twostep is notably better positioned. In the rural case all solvers perform close, except Vode; both in the rural and urban case Vode falls behind. Notable is the close performance of Ros3 and Rodas3. As a general conclusion, Rosenbrock codes are again superior to the BDF ones. The better relative positioning of Twostep (as compared to the CBM-IV cases) is most likely due to the increased number of species in AL.

The results for the urban problem with a 15 min. restart time are presented in Figure 4.9. We see more or less the same behaviour relative to the 1 hour restart time as for the CBM-IV problem. Now Twostep has become competitive to Rodas3 and Ros3 in the 10% error range, while the curve for Rodas reveals a rather strong non-monotonic accuracy-efficiency behaviour. Again the Qssa solver we previously applied in (Sandu et al., 1996a) severely lags behind to all others.

4.5.4 Problem F: stratospheric model

The work-precision diagram given in Figure 4.10 again reveals a very good performance of the Rosenbrock solvers compared to the other three. The higher order of accuracy of Rodas is again borne out and again notable is the close performance of Ros3 and Rodas3. Vode and Seulex have similar performance, but are more than 2 times slower than the Rosenbrock codes in the 1% accuracy range. Twostep follows at a large distance. We should recall, however, that for this problem no emissions were prescribed.

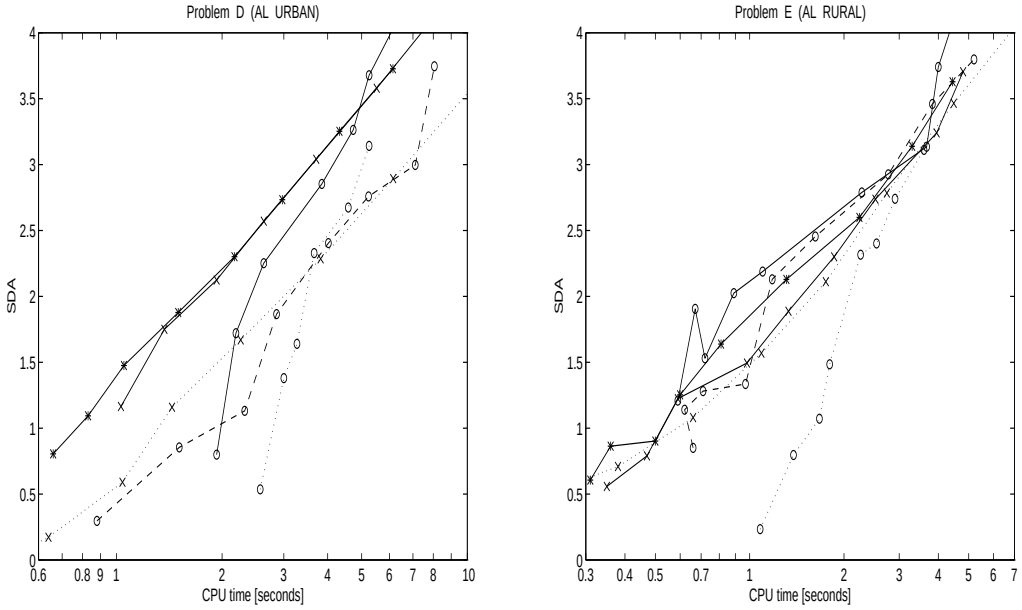

Figure 4.8. Work-precision diagram for test problems D and E (AL): Sparse Rodas3 (solid with "∗"), Sparse Ros3 (solid with "x"), Sparse Rodas (solid with "o"), Twostep Seidel (dots with "x"), Sparse Vode (dots with "o") and Sparse Seulex (dashed with "o").

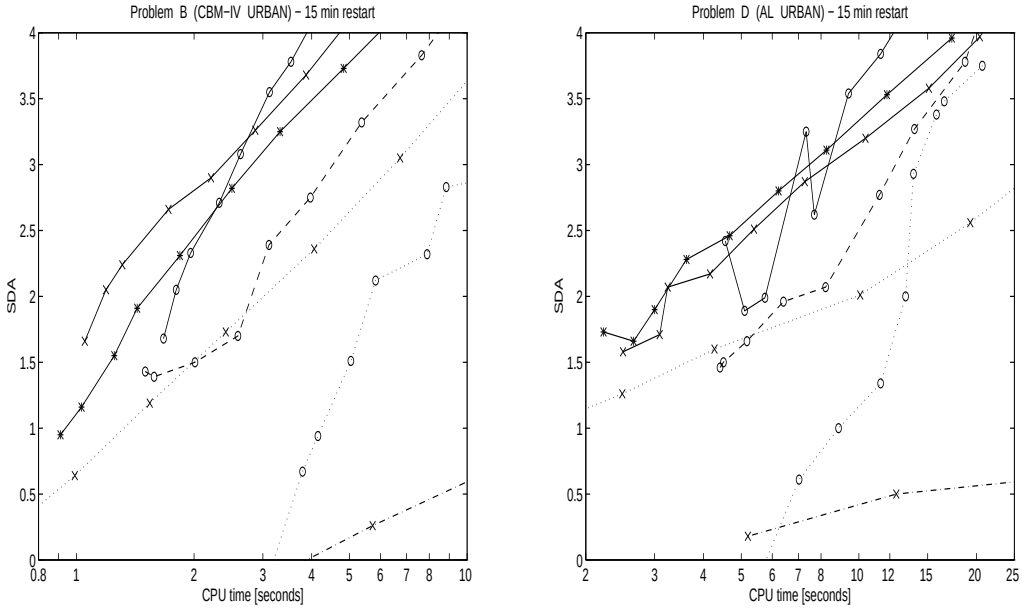

Figure 4.9. Work-precision diagram for test problems `B` (CBM-IV urban) and `D` (AL urban), with a restart each 15 minutes: Sparse Rodas3 (solid with "*"), Sparse Ros3 (solid with "x"), Sparse Rodas (solid with "o"), Twostep Seidel (dots with "x"), Sparse Vode (dots with "o"), Sparse Seulex (dashed with "o") and Qssa (dash-dots with "x").

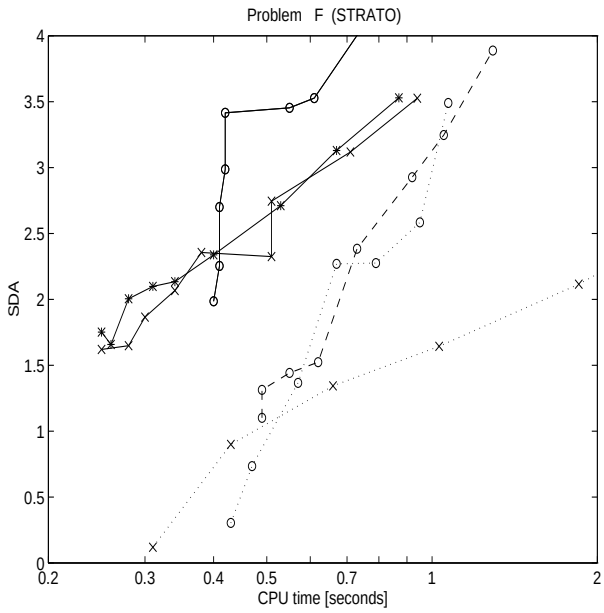

Figure 4.10. Work-precision diagram for test problem `F` (Strato): Sparse Rodas3 (solid with "∗"), Sparse Ros3 (solid with "x"), Sparse Rodas (solid with "o"), Twostep Seidel (dots with "x"), Sparse Vode (dots with "o") and Sparse Seulex (dashed with "o").

4.5.5 Problem G: aqueous model

As pointed out in [76], this test problem is the most difficult one from the numerical point of view. The Jacobian $f'(y)$ of the derivative function (3.1) contains stiff eigenvalues for which the relation $\lambda_i \approx -L_i$ (with L_i the destruction term associated with species i) does not hold. Such eigenvalues are due to the rapid gas-liquid phase interactions and cannot be associated with certain species; for this reason, all the explicit solvers tested in [76] failed to efficiently integrate the Aqueous model. As a consequence, in the present work Twostep was not applied to this problem. The results plotted in Figure 4.11 for the other solvers are very much in line with those for the stratospheric problem. In the low accuracy range the Rosenbrock family has the lead again, the performances of Rodas, Rodas3 and Ros3 being very close to each other. Seulex is about three times slower for 2 accurate digits, but seems to become the best for more than 4 digits; for higher accuracies, Vode changes slope and is not competitive.

4.6 Overall conclusions and remarks

The answer to the question of which stiff integrator is "the best" for being used in air quality models depends on a multitude of factors, some of the most important being the specific chemical mechanism employed, the desired accuracy level and the hardware on which the code runs. In the present work we considered a variety of chemical models, we covered the whole range of accuracy levels of practical interest and tested everything on two machines with completely different architectures. The set of tested codes includes Twostep and sparse versions of the extrapolation code Seulex, of the state-of-the-art BDF codes Lsode, Vode, the

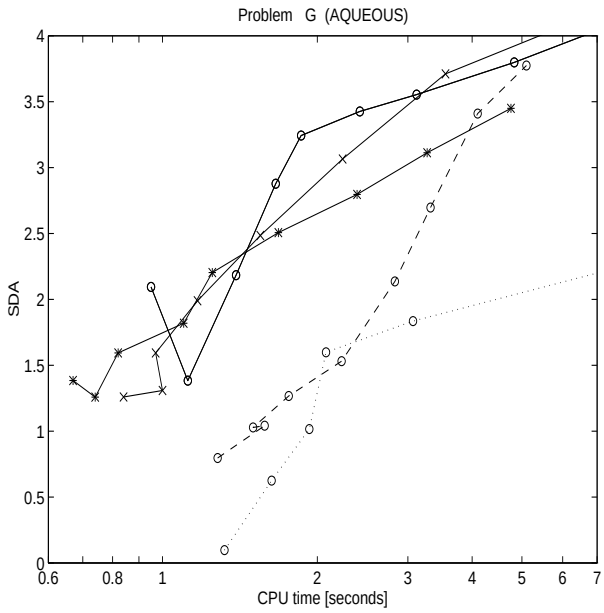

Figure 4.11. Work-precision diagram for test problem `G` (Aqueous): Sparse Rodas3 (solid with "∗"), Sparse Ros3 (solid with "x"), Sparse Rodas (solid with "o"), Sparse Vode (dots with "o") and Sparse Seulex (dashed with "o").

Runge-Kutta-type code Sdirk4 and the Runge-Kutta-Rosenbrock-type codes Rodas, Rodas3, Ros3 and Ros4. We have not considered in this benchmark the widely used BDF code Smvgear (Jacobson, 1994). This code is organized to specifically target a vector machine; running it in scalar mode on box models would lead to less than optimal results. We expect that for box models the performance of Smvgear will not differ much from that of sparse Lsode and Vode, as it is based on the first BDF code from Gear (1971). In the numerical ODE literature, this first Gear code has been replaced by the related solvers Lsode and Vode.

Although we have used utmost precaution in implementing the models and in testing the codes, still undiscovered errors and/or less optimal settings of user parameters may have affected part of the numerical results. The interested reader therefore is invited to repeat the experiments using our codes from (CGRER ftp site, 1996). The present results give rise to the following main conclusions:

- The sparse implicit solvers work efficiently for all problems tested here, including the gas-liquid phase one. In all the cases they give the fastest solution, when two or more accurate digits are required. In general Rodas, Vode and Sdirk4 have comparable performances, although their ranking relative to each other may differ per problem. It should be noted that Vode appeared to be somewhat sensitive to the choice of the absolute tolerances and the choice of a minimal step size.

- For all the test problems considered here and within 4 digits of accuracy the Rosenbrock solvers clearly provide the most cost-effective solutions among the codes tested in this paper and in (Sandu et al., 1996a).

- The relative ranking between the four sparse Rosenbrock solvers differs per

problem, but only to a limited amount. For lower accuracies of practical interest Rodas3 and Ros3 are usually the best. As expected, for higher accuracies Rodas is mostly competitive; the performance of the solver Ros4 is close to that of Rodas3 and Ros3. In passing we note that our test results do not consistently show that the property of stiff accuracy is truly advantageous for nonlinear problems.

- The above conclusion about the computational speed and accuracy of Rosenbrock methods is also supported by the comparison with the EBI method for Problem `A` and with the Qssa method for Problems `B,D` with a 15 min. restart time (the latter has been tested more extensively in (Sandu et al., 1996a)). Noteworthy is that the Qssa solver lags very far behind in all our experiments.

- Also robustness and ease of use are very important since in actual 3D transport a subtle tuning of the ODE code is cumbersome due to the large variety of conditions that will occur at different grid points. In this respect the Rosenbrock solvers are advocated as well. With the preprocessor KPP at hand, they are easy to use.

- Concerning robustness we have to point out that large values of $rtol$ (≥ 0.1 say) combined with too large values for $h_{\min}$ and h_{start} can cause the Rosenbrock solvers to drift away from the real solution, see Table 4.12. In these cases the initial transients are not resolved sufficiently accurately. Implicit solvers can also get into trouble here through convergence failures in the iterative modified Newton process. These problems can easily be avoided by choosing $h_{\min}$ and h_{start} sufficiently small and $rtol \leq 0.01$. Since Rosenbrock solvers may increase the step size rapidly, they can remain cost effective even with

smaller starting values.

- The extrapolation code Seulex never ran into a breakdown or returned a negative result. Apparently this code works very robust. However, in the low accuracy range Seulex is always significantly more expensive than Rodas3.

- With regard to robustness, also EBI performs outstanding. The method does not break down when used with a very large step size. We have only applied it to the TMk model we got from (Dentener, 1996), but our experience is in accordance with that reported in (Hertel et al., 1993) and (Krol, 1996) for different variants of the CBM-IV mechanism. Of course, the main drawback of the EBI approach is that it is intertwined with the chemistry and needs to be adjusted and retested any time the chemistry model is changed. The low accuracy of the solver is mainly due to the use of the first order Euler backward method. Implementation of the EBI approach with a higher order solver (e.g. Twostep) may lead to a notable improvement of accuracy.

- Twostep also performs extremely well with regard to robustness. The entries in Table 4.12 are due to negative SDA values, rather than breakdowns. This solver can handle both very large step sizes and crude tolerances. It seems to have only one serious limitation, which concerns gas-aqueous phase models. These models do require a linearly or fully implicit solver (Sandu et al., 1996a). Even though in our test problems it lags behind Rosenbrock solvers, Twostep remains, due to its explicit nature, an excellent candidate for very large tropospheric gas-phase problems with very small operator split steps. An additional advantage is that the Gauss-Seidel approach on which Twostep

is based, can be effectively extended towards a tridiagonal Gauss-Seidel approach for the coupled solution of chemistry and vertical turbulent diffusion (Verwer et al., 1996a, Spee et al., 1996).

- Of the dedicated explicit solvers, Twostep is clearly the best. This solver outperforms the other explicit solvers on all problems, often with a wide gap. Twostep is advocated for gas-phase problems only. This code should not be applied to gas-liquid phase problems. In general Twostep Seidel is more efficient than Twostep Jacobi. However, Gauss-Seidel iteration must be programmed in line which makes Twostep Jacobi somewhat easier to use. It is important to note that dedicated explicit solvers can sometimes be significantly improved by problem dependent modifications like lumping and/or group iteration. Of course, this requires a considerable knowledge of the reaction mechanism.

- In most cases sparse Vode is more efficient than the related sparse BDF solver Lsodes. We owe this to the fact that Vode uses a dedicated sparsity technique, whereas Lsodes uses the general Yale sparse matrix package. We should also mention that Lsodes is used without a precribed minimal stepsize. Some additional runs with sparse Vode without a minimal step size as well, demonstrated that this plays a minor role, the difference in sparse matrix treatment being more important. Both have been applied with the extra storage option for the Jacobian matrix so as to avoid Jacobian updates when possible.

- The best sparse implicit solvers and the best explicit solver (Twostep) should also be compared in 3D model applications. While box model tests are needed to select and develop promising ODE solvers, in real 3D transport-chemistry

models other factors should be taken into account as well. Quite important is the length of the time step in the operator splitting, since this determines the number of restarts. Restarts are expensive for implicit codes and one-step methods of Runge-Kutta type have an advantage here over multistep methods. Also robustness and ease of use are important in 3D models, since a subtle tuning of the ODE code is cumbersome due to the large variety of conditions that will occur at different gridpoints. Finally, the issues of memory use, vectorization [54, 92] and parallelization are of great practical importance too. Optimal ODE solvers should be tested in a 3D software environment where vectorization and parallelization take place.

- Often the work-precision curves are non-monotonic, revealing the situation that more CPU time is spent, yet a less accurate solution is obtained. This non-monotonicity is seen mostly for very low tolerances and is caused by the step size selection process (and dynamic iteration strategies in implicit solvers). These work out non-smoothly, as is for example shown very clearly in the diagrams in (Hairer et al., 1991). Inspection of our diagrams shows that the only variable step size solver yielding monotonic curves in all tests presented is Twostep.

- Finally, one word to the interested modellers. In this paper we present several options not considered before for choosing a chemical solver. As mentioned above, the performance depends on a multitude of factors; thus selecting an integrator should involve testing the most promising codes on the particular application considered. In this context our benchmark results should be thought of as guidelines, but they are no substitute for a careful, problem

specific testing.

Table 4.12. The values of *rtol* for which the codes either break down or give a solution with more than 100 % relative error (negative SDA).

Test / Code	B	D	E	F	G
Rodas	1, .3, .1	1, .3	–	–	–
RodasR3	1	1	–	–	–
Ros3	1, .3	1, .3	–	–	–
Twostep	1 , .3	1	–	1, .3, .1	all
Vode	1	1, .3, .1	1	1, .3	1

CHAPTER 5
KPP - AUTOMATIC GENERATION OF KINETIC EQUATIONS

5.1 Introduction

A significant part of the research in the atmospheric chemistry community is to develop chemical mechanisms able to accurately describe the chemical processes that determine the composition of trace species in the atmosphere. These chemical mechanisms are then used for simulating on computer the time evolution of the components of the atmosphere.

The kinetic preprocessor (KPP) described here solves the important problem of automatically translating the chemical equations into FORTRAN or C code that computes the evolution in time of the species starting with the specification of the chemical mechanism. KPP translates the chemical mechanism into a set of ordinary differential equations (more exactly, into a subroutine that computes the derivatives) and links them to a numerical integration routine.

The set of kinetic equations describing the chemical mechanism form a very stiff ODE system. General purpose integrators do not always offer the best solution, therefore many specialized numerical integration schemes have been proposed. KPP allows the selection of a numerical integrator from a rich set of integration schemes and provides a benchmarking platform for evaluating new integrators.

The complexity of the transport problem requires parallel processing power to be solved. KPP is able to generate parallel MPI (Message Passing Interface) code. In this way by simple providing KPP with a model description, it is capable

of generating correct parallel simulation code for the model.

In the following I will briefly outline the mathematical formulation of the kinetic problem, the user view of KPP and a short description of preprocessor's capabilities and language. A complete KPP documentation can be found in [25].

5.2 The kinetic equations

We will first describe the problem with the help of an example. Consider the following generalized reaction mechanism for photochemical smog:

We know the initial concentrations, for example:

and for each case we would like to trace the evolution of the concentrations in time for the next, say, 10 hours.

We know that the chemical kinetics is described by a set of ordinary differential equations of the form:

$$\frac{dc_i}{dt} = \sum_{j \in production[c_i]} KR(j) \cdot \prod_l c_l - \sum_{m \in destruction[c_i]} KR(m) \cdot \prod_n c_n$$

where $KR(i)$ is the rate constant of reaction i.

To be more exact, let us consider the ozone O_3. It is produced in reaction 2 and destroyed in reaction 3. The rate of variation of ozone concentration is therefore given by:

$$\frac{d[O_3]}{dt} = \underbrace{KR(1) \cdot [O] \cdot [O_2] \cdot [M]}_{O_3 \ production} - \underbrace{KR(2) \cdot [NO] \cdot [O_3]}_{O_3 \ destruction} \tag{5.1}$$

If we denote the "ozone production term" by:

$$P(O_3) = KR(1) \cdot [O] \cdot [O_2] \cdot [M]$$

and the "ozone destruction term" by:

$$D(O_3) = KR(2) \cdot [NO]$$

Table 5.1. Photochemical smog mechanism.

REACTION	RATE CONSTANT $(cm^{\alpha} molec^{\beta} sec^{-1})$
$NO_2 + h\nu \rightarrow NO + O$	$0.533\ min^{-1}$
$O + O_2 + M \rightarrow O_3 + M$	$6 * 10^{-34} * \left(\frac{T}{300}\right)^{2.3}$
$NO + O_3 \rightarrow NO_2 + O_2$	$2.2 * 10^{-12} exp\left(\frac{-1430}{T}\right)$
$RH + OH\cdot \rightarrow RO_2\cdot + H_2O$	$1.68 * 10^{-11} exp\left(\frac{-559}{T}\right)$
$RCHO + OH\cdot \rightarrow RC(O)O_2\cdot + H2O$	$6.9 * 10^{-12} exp\left(\frac{250}{T}\right)$
$RCHO + h\nu \rightarrow RO_2\cdot + HO_2\cdot + CO$	$1.91E-4\ min^{-1}$
$HO_2\cdot + NO \rightarrow NO_2 + OH\cdot$	$3.7 * 10^{-12} exp\left(\frac{240}{T}\right)$
$RO_2\cdot + NO \rightarrow NO_2 + RCHO + HO_2\cdot$	$4.2 * 10^{-12} exp\left(\frac{180}{T}\right)$
$RC(O)O_2\cdot + NO \rightarrow NO_2 + RO_2\cdot + CO_2$	$4.2 * 10^{-12} exp\left(\frac{180}{T}\right)$
$OH\cdot + NO_2 \rightarrow HNO_3$	$1.11 * 10^{-11}$
$RC(O)O_2\cdot + NO_2 \rightarrow RC(O)O_2NO_2$	$4.7 * 10^{-12}$
$RC(O)O_2NO_2 \rightarrow RC(O)O_2\cdot + NO_2$	$1.95 * 10^{16} exp\left(\frac{-13{,}543}{T}\right)$

Table 5.2. Different smog scenarios.

Initial concentration [ppm]	Case 1	Case 2	Case 3
RH	0.1	0.5	2.0
$RCHO$	0.1	0.5	2.0
NO	0.5	0.5	0.5
NO_2	0.1	0.1	0.1
$others$	0	0	0

we obtain:

$$\frac{d[O_3]}{dt} = P(O_3) - D(O_3) \cdot [O3] \tag{5.2}$$

This is a general pattern; the evolution of any species can be described in terms of P and D. In order to trace the evolution of the chemical system in time, we have to write the equations (5.1) or (5.2) for all the species in the reaction mechanism and then to solve numerically this system starting with the given initial values (cases 1, 2 and 3).

5.3 Possible implementations to solve the kinetic problem

There are several different ways to solve the kinetical equations above. We will discuss three of them which have been used in practice:

- **The hardcoded methode**. This approach means that the production and the destruction functions related to the chemical equations are written by hand and then translated into a programming language. This has been used in STEM-I and STEM-II developed by Carmichael et. al. [14]. The language used was FORTRAN-77. The advantage of this method conststs in the fact that the subroutines that implement the production and the destruction terms can be coded very efficiently in order to run very fast on a given computer. The major disadvantage is that in order to make even minor changes in the chemical mechanism one has to rewrite all the equations and therefore to rewrite all the code from scratch. This make the methode very unreliable.

- At the other end is the **totally integrated method**. Here we have the chemistry equation given in a special file, written in a specific description

language. The program parses the equations and stores them in memory as arrays of coefficients. Then the production and destruction functions are implemented by scanning these arrays at run time. This solve the major disadvantage of the first method. Now the code can adapt easily to any chemical mechanism. This approach was taken by LARKIN in [64]. The drawback of this approach is the fact that all the computation is done at runtime which results in reduced speed.

- The last method is the **preprocessing method**. Here, as with the totally integrated method, the chemical mechanism is described in a specific language. Then a preprocessing step parses the chemical equations and generates the apropriate code in a high level language (FORTRAN-77 or C). In this way the speed is about the same as for the hardcoded method, and changes of the chemical mechanism can be easily made. We also have the guarantee that the generated code is correct. This methode was partially adopted by CHEMKIN in [57].

Our implementation further extends the preprocessing method. A specific language has been designed for specifying the chemical mechanism, the initial values, integration options, including the capability to select the integration method and the integration driver. The language is called KPP-language and a reference of its syntax is given in section 5.6.

5.4 User view

There are several reasons for giving a different view of the KPP to the users. First, a non-experienced user may be concerned about having to manage a big file

with a lot of information that he does not care about that much as long as it works. The second reason, and may be the most important one, is that anyone working in atmospheric chemistry field usually thinks in terms of these logical modules: **chemistry model**, **numerical integrator**, **driver**.

5.4.1 The chemical model

The chemical model contains the description of the atoms, species, and equations. It also contains default initial values for the species and default options including the best integrator for the model. In terms of files, the structure is detailed in figures 5.2 and 5.3.

In the case described in figure 5.2, the main description file, i.e. the one passed as parameter to KPP, can contain just a single line selecting the model. KPP trys to find a file with the name of the model and the extension **.def** in the **models** subdirectory. This file is then parsed. The content of the model definition file is written in the KPP language. The model definition file has to point to a species file and an equation file. Both are included in the description. The species file includes further the atom definition file.

The species file list all the species in the model in a **#DEFVAR**, or **#DEFRAD**, or **#DEFFIX** section. The atom file lists the Periodic Table of Elements in an **#ATOM** section. The equation file contains the description of the equations in an **#EQUATIONS** section.

All default values regarding the model are automatically selected. For convenience, the best integrator and driver for the given model are also automatically selected.

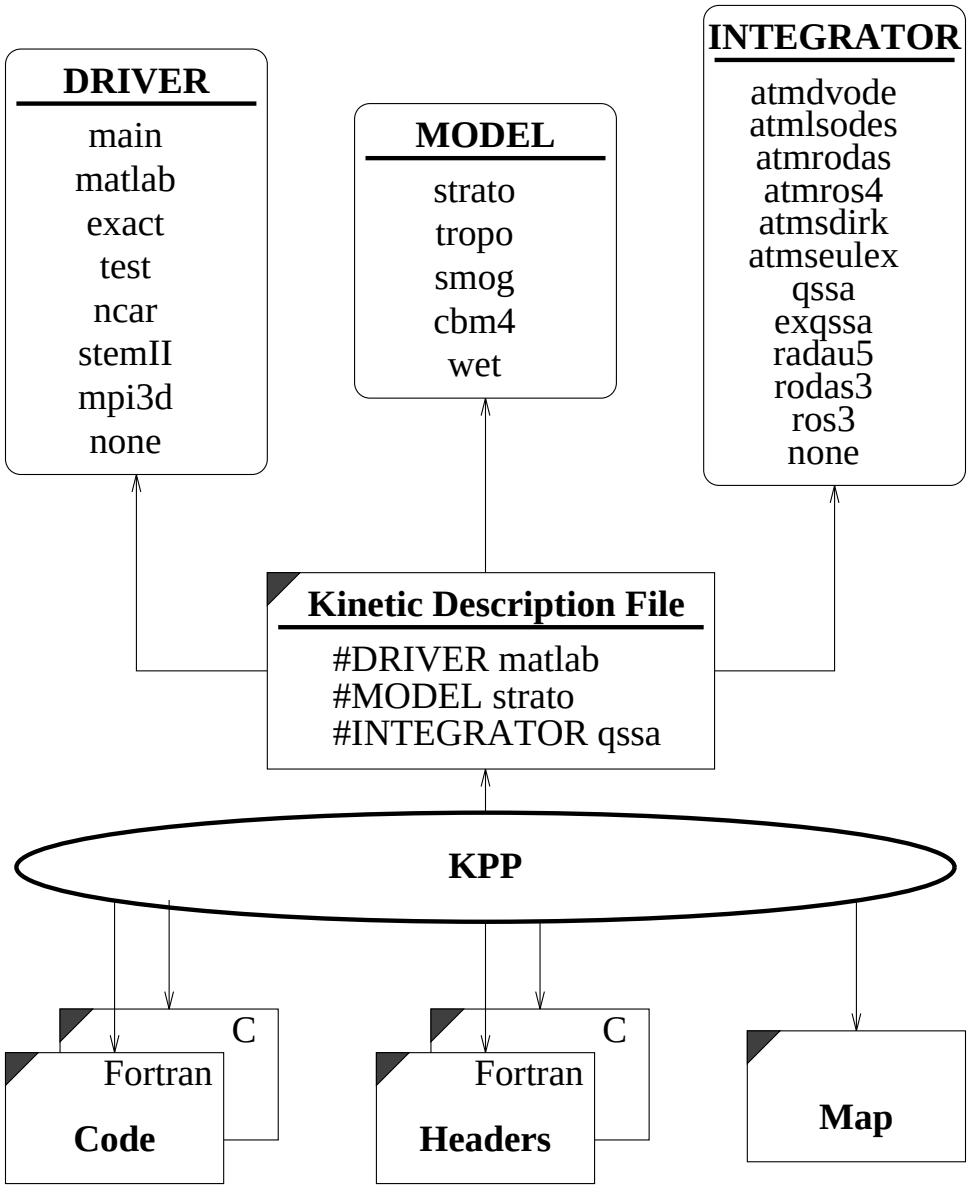

Figure 5.1. KPP user perspective

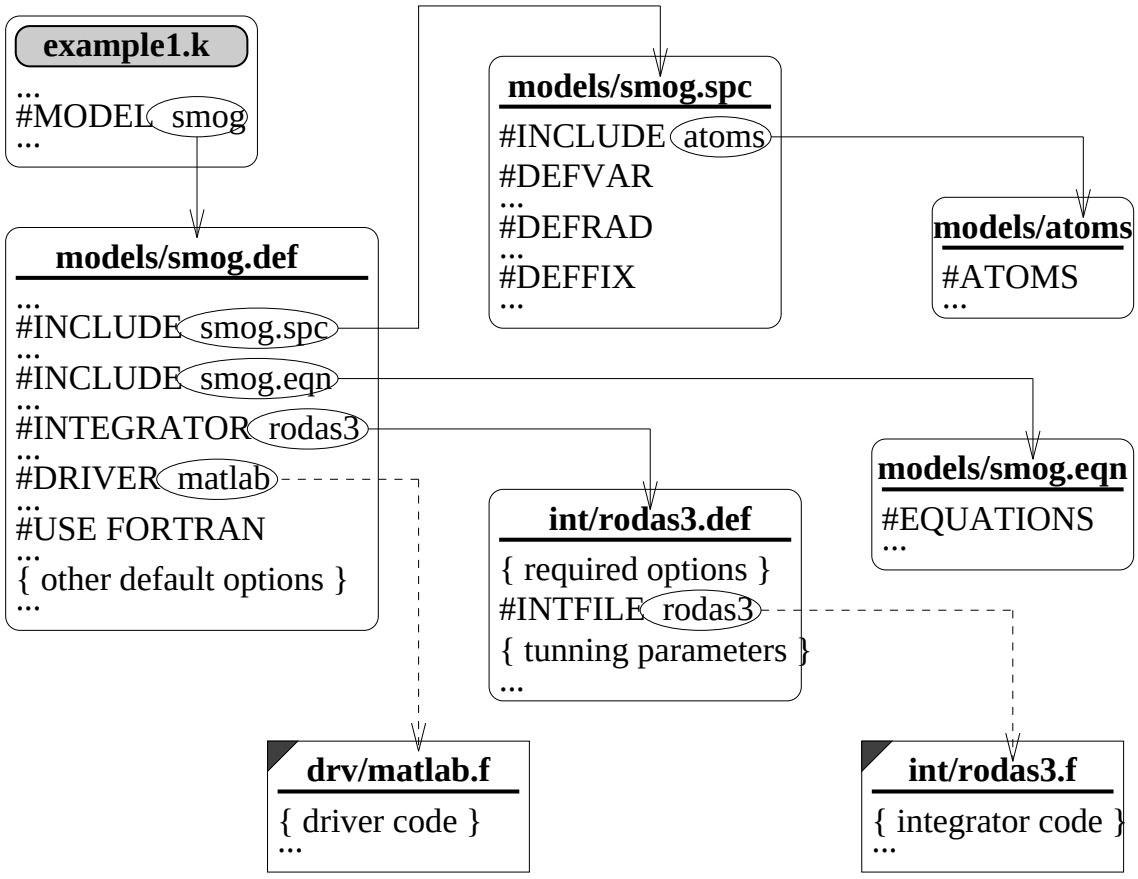

Figure 5.2. Input files details (default integrator)

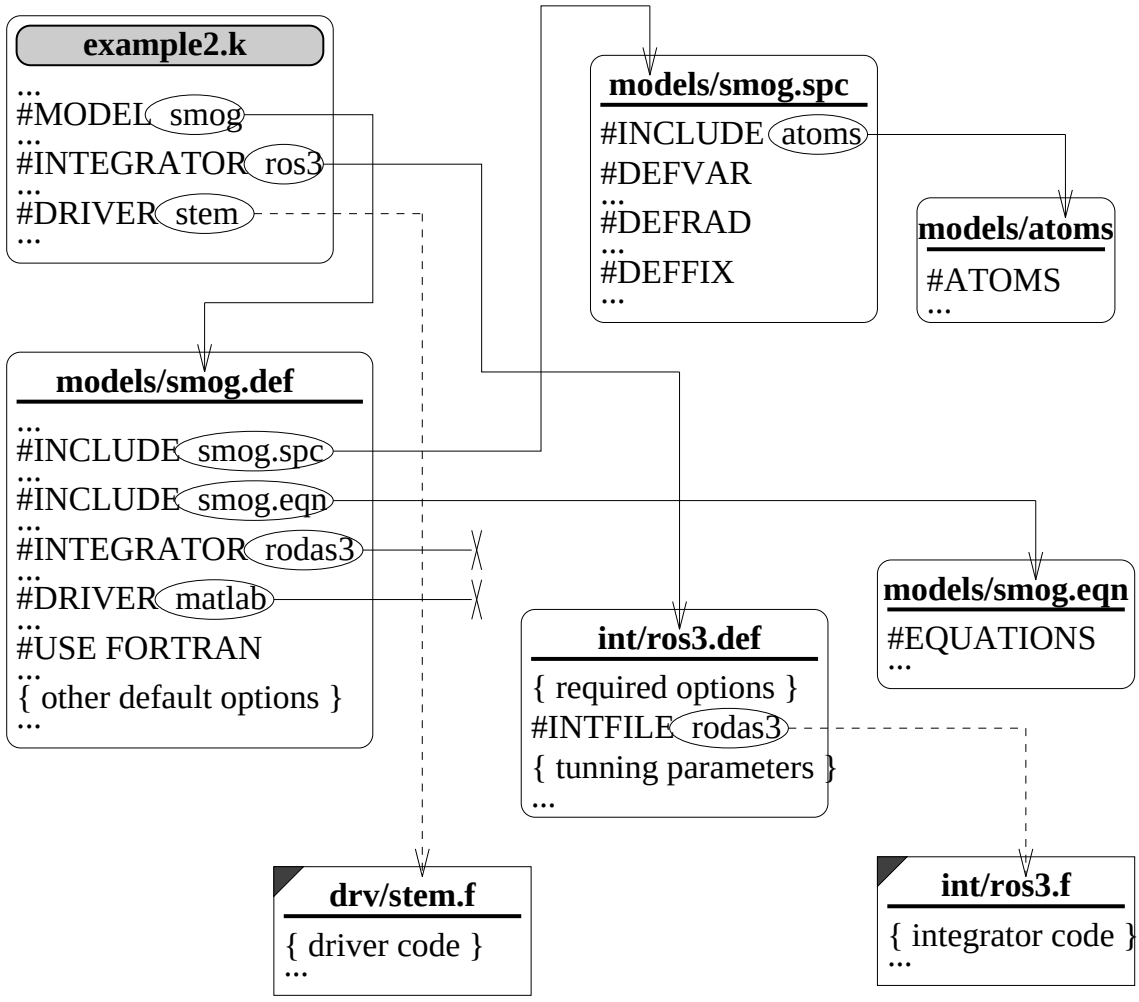

Figure 5.3. Input files details (selected integrator)

If a different integrator or a different driver is desired, then by selecting them in the main description file, they override the default settings as shown in Figure 5.3

5.4.2 The integrator

The integrator selection (either the default one from the model definition file or the one selected by the user) points to an **integrator definition file**. This file is also written in the KPP language and contains a reference to the auxiliary file containing the integrator code, options required by the integrator and tuning parameters.

Each solver may require KPP to generate different functions, for example one integrator may need the production/destruction function in the aggregate form, others may need it in the split form. Similarly, some integrators may need the full Jacobian, and others may want the Jacobian in sparse format. These options are selected through appropriate parameters given in the integrator definition file.

Some integrators have additional parameters that can be fine tuned for better performance. The values for these parameters are also included in the integrator definition file.

5.4.3 The driver

The driver is basically the main program. It is responsible for calling the integrator routine, reading the data from files and writing the results. Different existing drivers differ from one another by their input and output data file format, and by the auxiliary files created for interfacing with visualization tools. A program that performs 3D atmospheric chemistry simulation is also called a driver, if it calls the integrator routine generated by KPP for the chemistry integration.

5.5 KPP capabilities

The following is a list of the KPP-preprocessor capabilities:

- The preprocessor generates code for the following **functions**:
  - `F_VAR ( V, R, A_VAR )`

    function for the derivatives of variables - Agregate form
  - `F_RAD ( V, R, A_RAD )`

    function for the derivatives of radicals - Agregate form
  - `FSPLIT_VAR ( V, R, P_VAR, D_VAR )`

    function for the derivatives of variables - Split form
  - `FSPLIT_RAD ( V, R, P_RAD, D_RAD)`

    function for the derivatives of radicals - Split form
  - `JACVAR ( V, R, JV )`

    function for the Jacobian of Variables
  - `JACRAD ( V, R, JR )`

    function for the Jacobian of Radicals
  - `JACVAR_SP ( V, R, JVS )`

    function for the Jacobian of Variables using sparse matrix representation
  - `JACRAD_SP ( V, R, JRS )`

    function for the Jacobian of Radicals using sparse matrix representation
  - `JACVAR_VEC ( JV, UV, JUV )`

    function for sparse multiplication

- `JACRAD_VEC ( JR, UR, JUR )`

  function for sparse multiplication

- `INITVAL()`

  function to update initial concentrations

- `UPDATE_SUN()`

  function to update SUN light using TIME

- `UPDATE_RCONST()`

  function to update rate constants

- `UPDATE_C()`

  function to update concentrations for transport

- `GETMASS( MASS )`

  function that compute total mass of selected atoms

- `INTEGRATE( DT )`

  Integrator routine

- `Driver - MAIN()`

  Main program - driver routine

- `UTIL`

  Utility functions

- `BLOCK DATA SPARSE_DATA`

  Sparse data

- `SPARSE_UTIL`

  SPARSE Utility functions

- `SOLVE ( JVS, X )`

  function for sparse back substitution

- **Balance checking** All the species must be declared (this declaration resembles the declaration of variables in programming languages like C and Fortran) before their first appearance in any equation. When defining a species its exact composition (i.e. the atomic structure of the molecule) can be specified such that each equation can then be checked for atom balance.

- **Generates FORTRAN-77 or C code**

- **Optimizes the code.** The generated code is optimized by plugging into the code the numerical values of all coefficients and rate constants and by performing all possible calculation at the preprocessing time. Repeated computations are detected and avoided by the use of intermediate variables.

- **Automatically inlines appropriate comments.** Comments are automatically inlined in the generated code to make it readable. Information like date of creation, directory, input files are also provided. This is specially usefull to distinguish between programs generated from different versions of the same chemical model.

- **Allows expressions in defining the rate coefficients.** Part of the equation rates depend on the time of day, on temperature, on light intensity and so on. The preprocessor treats them seperatelly and therefore allows expressions in defining such equation rates.

- **Generates code for the Jacobian of the Production and Destruction functions.** Implicit integration methods require this Jacobian. Generating

efficient code for the Jacobian improves the quality of such integrators.

- **Support for sparse techniques.** The Jacobians have a very good sparsity. The preprocessor can generate code that computes them in the usual sparse formats (row format, compressed row and compressed column format).

- **Reordering species** The KPP-preprocessor reorders species such that the sparsity is preserved to a certain extent by an LU-decomposition. This is done in an optimal way by using a diagonal Markowitz algorithm.

- **Transparent mapping of species** One of the cumbersome problems with the STEM-II aproach is that in order to interpret the results one has to know the numbering of species used when the program was coded. KPP generates MAP files, header files and initialisation files (for MATLAB) that describes this mapping.

- **Off-line interface with MATLAB.** The values of the concentration are saved in specific data files during the integration. An initialization MATLAB file is also generated, that reads all data in MATLAB and performs the species translation. Therefore one can plot species calling them with their names instead of their coresponding number.

- **On-line interface with NCAR-Graphics** Allows selected species to be plotted, and monitorized during the integration.

5.6 KPP comand language

The KPP language allows us to specify the chemical equations, the initial values, integration parameters, program code and many other options. All these

information are gathered in a filed called KPP **model file**. The KPP-preprocessor parses this **model file** and generates the appropriate output files. It basically generates a **code file** , **header files** and a **map file**.

- The **code file** is a C or FORTRAN source file that integrates the kinetic equations described in the **model file**.
- The **header files** are C or FORTRAN header files used by the **code file** and by any other user written file.
- The **map file** contains a summary of all the functions, subroutines and data structures defined in the **code file** plus a summary of the numbering and category of the species envolved.

The following general rules define the structure of a **model file**, sometimes called equation file.

- A KPP program is made up of **KPP sections**, **KPP comands** and **program fragments**.
- Anything enclosed between `{` and `}` is a comment as is ignored by the preprocessor.
- Any name given by the user to denote an atom or a species for example has to have less than 32 character in length and cannot contain blanks, tabs, new lines, `#, +, -, ;,` :. All names are **case insensitive**.
- A section begins on a new line with a `#` sign followed by a section name. Then a list of items separated by semicolon follow. The syntax of an item definition is different for each particular section.

- A command begins on a new line with a # sign followed by a command name and one or more parameters.

- A program fragment begins on a new line with #INLINE and a fragment type. It ends with #ENDINLINE. Inbetween is a program fragment written in FORTRAN or C as specified by the fragment type. This piece of code is inserted in the code file in places determined also by the fragment type.

Following is the BNF-like specification of the language:

```
program :: module | module program
module :: section | command | code_fragment

section :: #ATOMS atom_definition_list |
           #DEFVAR species_definition_list |
           #DEFRAD species_definition_list |
           #DEFFIX species_definition_list |
           #EQUATIONS equation_list |
           #INITVALUES initvalues_list |
           #CHECK atom_list |
           #LOOKAT species_list atom_list |
           #MONITOR species_list atom_list |
           #SETVAR species_list_plus |
           #SETRAD species_list_plus |
           #SETFIX species_list_plus |
           #TRANSPORT species_list |
           #LUMP lump_list |

command :: #INCLUDE file_name |
           #JACOBIAN < OFF | ON | ALL | SPARSE > |
           #DOUBLE < OFF | ON > |
           #FUNCTION < AGGREGATE | SPLIT > |
           #CHECKALL |
           #LOOKATALL |
           #TRANSPORTALL |
           #XGRID integer |
           #YGRID integer |
           #ZGRID integer |
           #USE < FORTRAN | C > |
           #INTEGRATOR file_name |
           #DRIVER file_name
```

```
code_fragment ::
          #INLINE fragment_type fragment_code #ENDINLINE
fragment_type :: F_DECL | F_INIT | F_DATA | F_3DINIT | F_3DDATA |
                 C_DECL | C_INIT | C_DATA | C_3DINIT | C_3DDATA

atom_definition_list :: atom_definition |
                        atom_definition atom_definition_list
atom_definition :: name ;

species_definition_list :: species_definition |
                           species_definition  species_definition_list
species_definition :: name = atom_composition;
atom_composition :: atom_count |
                    IGNORE |
                    atom_count + species_composition
atom_count :: count atom_name | atom_name
count :: integer

equation_list :: equation |
                 equation equation_list
equation :: expresion = expresion : rate ;
expresion :: term | term + expresion | term - expresion
term :: number species_name | species_name
rate :: program_expresion

initvalues_list :: initvalues_assignment |
                   initvalues_assignment initvalues_list
initvalues_assignment :: species_name = number ; |
                         VAR_SPEC = number ; |
                         RAD_SPEC = number ; |
                         FIX_SPEC = number ; |
                         ALL_SPEC = number ; |
                         CFACTOR = number ;

atom_list :: atom_name |
             atom_name atom_list

species_list :: species_name |
                species_name species_list

species_list_plus :: species_list |
                     VAR_SPEC; species_list |
                     RAD_SPEC; species_list |
```

```
                    FIX_SPEC; species_list |
                    ALL_SPEC; species_list

lump_list :: lump |
             lump lump_list
lump :: lump_sum : species_name ;
lump_sum :: species_name |
            species_name + lump_sum
```

In the following we give a precise definition of each section.

- ATOMS

  This section defines all the atoms that will be further used. Usualy the names of the atoms are the ones specified in the Periodic Table of Elemants. For this table there is a predefined file containing all definitions that can be used by the command:

  ```
  #INCLUDE atoms
  ```

  If this is the first line in the model file then we can use any atom in the Periodic Table of Elements.

- DEFVAR, DEFRAD, DEFFIX

  These sections define all the species that will be used in the chemical mecanism. Species can be **variable**, **radical** or **fixed**. The type is specified by inclusion in the appropriate section. Moreover for each species we have to declare the atom composition. If the species is a generic species and we do not have an exact composition we can alwais ignore it. This can be done stating that the species is composed by a special, predefined atom IGNORE.

- EQUATIONS

This is the section were the chemical mechanism is specified. Each equation is written in the natural way in which a chemist would write it, using only the names of already defined species. At the end of each equation, separated by a colon, has to be placed the rate constant. This does not necesarily need to be a numerical value. Instead it can be a valid expresion in the target language.

- `INITVALUES`

  Here we define the initial concentration values for all species. If no value is specified for a particular species, a default value is used. One can set the default values using the generic species names: VAR_SPEC, RAD_SPEC, FIX_SPEC, ALL_SPEC. In order to use coherent units for concentration and rate constants, it is sometimes necessary to multiply each value by a constant factor. This factor can be set by using the generic name CFACTOR. Then each of the initial values will be multiplied by this factor before being used.

- `CHECK`

  This is a facility that enables atom balance checking in each equation. By default, if this section is missing, no checking is performed. Else each equation is checked to have proper balancing for each of the listed atoms. To enable checking for all atoms one can use the command CHECKALL. By convention an IGNORE atom in a species S matches any number of atoms of any type that are not already explicitly defined in S.

- `LOOKAT`

  This section instructs the preprocessor what are the species for which the evolution of the concentration, should be saved in a data file. By default

the LOOKATALL command is assumed, which activates all the species. If an atom is specified then the total mass of the particular atom is reported. This allows to check how the mass of a specific atom was conserved by the integration method.

- `MONITOR`

  This list of species and atoms is similar to the one used in LOOKAT section. The difference is that this list is used by the graphics driver to offer a feedback of the evolution of the selected species during the integration.

- `SETVAR, SETRAD, SETFIX`

  Here we can change the type of an already defined species. The use of the generic species VAR_SPEC, RAD_SPEC, FIX_SPEC, ALL_SPEC is also permitted. The purpose of this section is to allow species to be defined in one category or the other according to their usual behavior, and then depending on the integration methode used, one can use or not the initial classification, or can easily move one species to another category.

- `TRANSPORT`

  This section is only used for transport and chemistry models. It specifies which species for which the chemistry part is computed should be transported. TRANSPORTALL will transport all species.

- `LUMP`

  Here one can define various lumping of the species to reduce the stiffness of some models. This facility is currently under development.

  The commands have the following meaning:

◊ `INCLUDE`

The file name specified as a parameter will be parsed by the preprocessor before proceding to the next line. This allows the atoms definition, the species definition and even the equation definition to be shared between several models.

◊ `JACOBIAN`

- OFF - means the integrator does not need the jacobian and therefore the preprocessor will not generate it.
- ON - should be used if the integrator needs separate jacobians for variables and radicals.
- ALL - should be used if the integrator needs the whole jacobian for all species.
- SPARSE - should be used if the integrator needs the whole jacobian, but in a sparse form. The format used is compressed on rows.

◊ `DOUBLE`

ON means use double precision, OFF means use single precision.

◊ `FUNCTION`

The functions that compute the right hand side of the differential equations for variable and radical species are generated in one of the following formats:

- AGGREGATE - computes the normal derivatives.
- SPLIT - gives the derivatives in production-destruction form.

◇ `CHECKALL, LOOKATALL, TRANSPORTALL`

These commands, make all species to belong to the coresponding list as described above.

◇ `XGRID, YGRID, ZGRID`

These commands are used only for the 3D-transport part, and allow setting of the grid dimension on X, Y and Z directions.

◇ `USE`

- FORTRAN - means generate FORTRAN code. Note that the selected driver and integrator should be available in FORTRAN.
- C - means generate C code. Note also that the selected driver and integrator should be available in C.

◇ `INTEGRATOR`

The parameter here is a file name, without extension. The apropriate extension (.f or .c) is appended and the result selects the file that contain the integrator to be used. This command allows use of different integration techniques on the same model.This file will be copied into the code file in the apropriate place. All integrators have to conform to the same specific calling sequence. See section 5.7 for details.

◇ `DRIVER`

The parameter here is also a file name, without extension. The apropriate extension (.f or .c) is appended, and the result selects the file that contain the driver to be used. This file will be copied into the code file in the apropriate place. See section 5.7 for details on how to write new drivers.

5.7 KPP data structures

In adition to the function that the preprocessor generates it also defines some constants and a global data structure. All the constants are defined using `#define` in C and `PARAMETER` in FORTRAN.

Defined constants	
Name	Description
NSPEC	The number of species involved
NVAR	The number of Variable species
NVARACT	The number of Active species
NRAD	The number of Radical species
NFIX	The number of Fixed species
NREACT	The number of reactions
NVARST	Starting of variables in conc. vect.
NRADST	Starting of radicals in conc. vect.
NFIXST	Starting of fixed in conc. vect.
NONZERO_V	Number of nonzero variable elements
LU_NONZERO_V	Number of nonzero variable LU elements
NONZERO_R	Number of nonzero radical elements
CNVAR	Nr of elem in compressed row format for variables
CNRAD	Nr of elem in compressed row format for radicals
PI	Value of pi
NLOOKAT	number of species to look at
NMONITOR	number of species to monitor
NX	X grid dimension
NY	Y grid dimension
NZ	Z grid dimension
MAX_XYZ	maximum grid dimension
NS	number of species to transport
I_*name*	Index of species *name* in concentration vector

Defined global variables		
Name	Dimension	Description
C_DEFAULT	NSPEC	Default concentration for all species
C	NSPEC	Concentration for all species
VAR	NVAR	Conc variables = C[NVARST..NVARST + NVAR]
RAD	NRAD	Conc radicals = C[NRADST..NRADST + NRAD]
FIX	NFIX	Conc fixed = C[NFIXST..NFIXST + NFIX]
RCONST	NREACT	Rate constants
TIME	1	current integration time
SUN	1	light intensity
TEMP	1	temperature
RTOLS	1	(scalar) relative tolerance
TSTART	1	integration start time
TEND	1	integration end time
DT	1	integration step
ATOL	NSPEC	Absolute tolerance
RTOL	NSPEC	Relative tolerance
STEPMIN	1	minimum allowed intergation step
STEPMAX	1	maximum allowed integration step
CFACTOR	1	Conversion factor
LOOKAT	NLOOKAT	indexes of species to look at
SLOOKAT	NLOOKAT	names of species to look at
MONITOR	NMONITOR	indexes of species to monitor
SMONITOR	NMONITOR	names of species to monitor
TRANS	NS	indexes of species to transport
STRANS	NS	names of species to transport
NMASS	1	number of atoms to check mass balance
SMASS	NMASS	names of atoms for mass balance

The names of parameters and global variables are important for writing new integrators and new drivers.

In order to write a new integrator one has to write a function:

```
INTEGRATE( DT )
```

with DT the the time interval for the integration. All other parameters are taken from the global data structure.

In order to write a new driver one has to write the `MAIN` function/subroutine. From here it can call the `INTEGRATE` function as desired, `INIT_VAL` to initialize the initial concentrations, and any other function defined by the preprocessor.

5.8 Other points

At the moment there are two kind of drivers: general purpose drivers, that are integrator and model independent and special purpose drivers build to work only with certain models or integrators. In the future we will try to have a set of general purpose drivers and discourage the use of special purpose drivers.

The following is the list of available general purpose drivers, and the language in which they are available.

Driver	Description	Language
main	offers MATLAB output and onscreen monitoring	FORTRAN,C
maingr	adds on-line graphics using NCAR	C
3d	solves the 3D model	C

The KPP on line documentation, models and examples are available from the CGRER home page at `http://www.cgrer.uiowa.edu/`

CHAPTER 6
RELATED TOPICS

6.1 Introduction

In air quality models flux or Dirichlet boundary conditions are specified for the whole problem. Usually operator splitting is employed to decouple the non-linear chemistry from the (linear) advection-diffusion part; and further, directional splitting is used to reduce the advection-diffusion equation to a sequence of one-dimensional subproblems. Sometimes advection is also decoupled from the diffusion. In all the cases consistent boundary conditions are to be set for each subproblem. In the first part of this chapter we discuss the correct setting of the intermediate boundary conditions; in the second part we explore the possibility of avoiding this complication alltogether while still maintaining the computational efficiency. The new family of methods (ELADI - extended linearized alternating direction implicit) performs the splitting "inside" the numerical method, rather than separating the operators; thus splitting error becomes numerical error and can be estimated. Several ideas are presented; more work needs to be done to carefully analyse the properties of these methods.

6.2 Consistent boundary values

When operator splitting is employed to numerically simulate the time evolution of the system, consistent boundary conditions are to be specified for each of the subproblems (except for the chemical interactions). By consistent boundary

conditions we mean that:

1. each subproblem is well posed;

2. the theoretical order of accuracy of the time split approximation is not lowered by the actual specification of intermediate boundary conditions.

Several papers were devoted to an analysis of the impact of intermediate boundary conditions on the accuracy of the solution (several ciations here). Our approach goes along the lines of LeVeque (citation). Since the boundary conditions are specified in terms of the exact solution of the entire problem, the idea is to express the intermediate solutions in terms of this global solution. These relations are then used to derive accurate boundary conditions.

6.3 How accurate should the boundary conditions be

For initial-value problems with smooth enough solutions, operator splitting offers an $O(k^2)$ approximation to the initial problem ($O(k^3)$ for symmetric splitting) where k is the split time-step (Strang, citation). For boundary value problems care has to be taken in specifying intermediate boundary conditions.

In what follows we will consider that each advection/diffusion subproblem (i.e., along a direction)

$$\begin{cases} u_t = L(t)u + f(t,x) \text{ in } \Omega \\ u(t_0,x) = u_0(x), \qquad B(t)u = \beta(t,x) \text{ on } \partial\Omega \end{cases}$$

is well-posed in the (rather strong) sense that

$$\max_{t_0 \le t \le t_0+k} (\|u(t)\|_\Omega^2 + |u(t)|_{\partial\Omega}^2 \le C(k) \left[\max_{t_0 \le t \le t_0+k} \left(|\beta(t,x)|_{\partial\Omega}^2 + \|f(t,x)\|_\Omega^2 \right) + \|u_0\|_\Omega^2 \right] \tag{6.1}$$

The imprecision in specifying the boundary conditions, the initial values and the forcing term should be of order $O(k^p)$ to guarantee that an $O(k^p)$ error is made in the solution u (argument by linearity, bound $\|u - \tilde{u}\|$).

The same holds true for the chemical subsystem

$$u' = f(t,u)$$

if, for example, f satisfies a one-sided Lipschitz condition. We then have

$$\max_{t_0 \le t \le t_0+k} |\tilde{u}(t) - u(t)| \le C(k) |\tilde{u}(t_0) - u(t_0)|$$

6.4 An example

Consider the advection of a rectangular profile over a square domain, as in figure 6.1 a). We consider $v_1 = v_2 = v = 1$. The transported profile has concentration 1, on a concentration 0 background. We perform directional splitting, that is we alternatively advect the profile along x and along y. Note that, if the profile was to be advected in $\Re^2$ this splitting gave the exact solution. Boundary conditions are of Dirichlet type, the value being 1 on the intersection of the domain's inflow boundary with the profile, and 0 elsewhere. Since the profile to be transported is rectangular with sides parallel to the diagonal of the domain, and the advection takes place along the diagonal, the boundary conditions are time-independent.

After a split step of length k, the difference between the exact and the approximated solution is

$$\|u(k) - u_{split}(k)\|_{L2}^2 = k^2 v^2$$

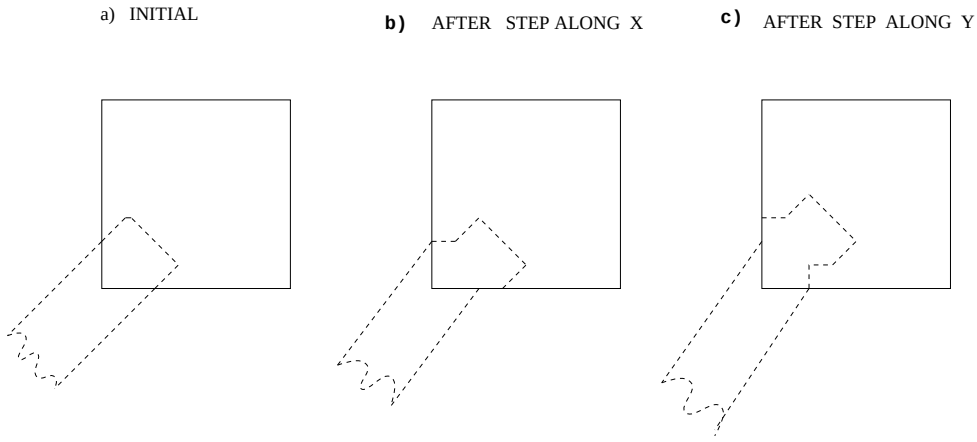

Figure 6.1. Splitting error for a simple test problem. A profile of height 1 is advected over a square domain; boundary conditions are of Dirichlet type and are time invariant. A simple directional splitting is considered; this splitting is exact for an infinite domain. The errors in the final solution are due to the incorrect prescription of boundary values.

When approximating the equation on $[0, T]$ with a step-size k the error will be

$$\|u(T) - u_{split}(T)\|_{L2}^2 = Tkv^2$$

which shows a convergence order $O(\sqrt{k})$ in L_2 norm. This is due to the boundary conditions, and has nothing to do with the smoothness of the solution. Note that the time-independence of boundary values is not sufficient to preserve the theoretical order of 1; a necessary and sufficient condition is that boundary values are constant along each side of the domain, as will be shown later.

6.5 Directional splitting

The following examples (two-dimensional transport equation and heat equation) will help us understand the key points. Let $\Omega = [0, 1] \times [0, 1]$. For the heat equation let $L_1 = K\partial/\partial x^2$, $L_2 = K\partial/\partial y^2$. Denote $\{0, 1\} \times [0, 1] = \partial^x\Omega$,

$[0,1] \times \{0,1\} = \partial^y\Omega$ and $\partial\Omega = \partial^x\Omega \cup \partial^y\Omega$. For the transport equation let $L_1 = -v_1\partial/\partial x$, $L_2 = -v_2\partial/\partial y$ with $v_1, v_2 > 0$. Denote the upwind boundaries by $\{0\}\times[0,1] = \partial^x\Omega$, $[0,1]\times\{0\} = \partial^y\Omega$ and $\partial\Omega = \partial^x\Omega \cup \partial^y\Omega$. Since (on smooth enough arguments) L_1 and L_2 commute we will consider only a simple (non-symmetric) splitting.

Consider the initial-value problem (with Dirichlet boundary conditions) :

$$\begin{cases} u_t = L_1 u + L_2 u \text{ in } \Omega, \\ u(0,x) = u_0(x), \qquad u = \beta \text{ on } \partial\Omega \end{cases}$$

On the time interval $[0,k]$ we approximate the above system by the sequence

$$\begin{cases} u_t^* = L_1 u^*, \quad u^*(0,x) = u(0,x), \quad u^* = b^* \quad \text{on } \partial\Omega \ ; \\ u_t^{**} = L_2 u^{**}, \quad u^{**}(0,x) = u^*(k,x), \quad u^{**} = b^{**} \quad \text{on } \partial\Omega \ . \end{cases}$$

Our goal is to approximate the exact solution $u(t)$ at $t = k$ by $u^{**}(k)$ with an $O(k^3)$ error. Question is, how should one set b^*, b^{**} to achieve this goal; are the "intuitive" boundary conditions $b^* = b^{**} = b$ good enough ?

To perform the analysis, we consider the first subproblem

$$u_t^* = L_1 u^*, \quad u^*(0,x) = u(0,x), \quad u^* = b^* \text{ on } \partial\Omega \ .$$

The solution at (t,x)

$$\begin{aligned} u^*(t,x) &= u^*(0,x) + t u_t^*(0,x) + \frac{t^2}{2} u_{tt}^*(0,x) + \ldots \\ &= u^*(0,x) + t L_1 u(0,x) + \frac{t^2}{2} L_1^2 u^*(0,x) + \ldots \\ &= (I + t L_1 + \frac{t^2}{2} L_1^2 + \ldots) u^*(0,x) \\ &= e^{tL_1} u^*(0,x) \ . \end{aligned}$$

The idea is now to express u^* in terms of u using the fact that $u^*(0,x) =$

$u(0,x), \quad \forall x$:

$$\begin{aligned} u^*(t,x) &= e^{tL_1}u^*(0,x) &&= e^{tL_1}u(0,x) \\ e^{tL_2}u^*(t,x) &= e^{tL_2}e^{tL_1}u(0,x) &&= e^{t(L_1+L_2)}u(0,x) \\ &= u(t,x)\ . \end{aligned} \tag{6.2}$$

Sending now $x \to \partial^x\Omega$ and using the continuity of u on $\Omega \cup \partial\Omega$ and of u^* on $\Omega \cup \partial^x\Omega$ we have that

$$b^*(t,\xi) = e^{-tL_2}b(t,\xi) \quad \forall \xi \in \partial^x\Omega\ . \tag{6.3}$$

This relation makes sense, as $b^*(t,x)$ is to be defined on $\partial^x\Omega$ and L_2 involves only y-derivatives. The meaning of (6.3) is that b^* *is obtained from* b *by "undoing"* L_2 *for a time period of* t. For the transport equation, the above relation translates into $b^*(t,x,y) = b(t,x,y-tv_2)$. For the heat equation, (6.3) tells us to *antidiffuse* b in the y direction for a period t.

For the second subproblem

$$u_t^{**} = L_2u^{**}, \quad u^{**}(0,\xi) = u^*(k,\xi), \quad u^{**} = b^{**} \text{ on } \partial\Omega\ .$$

similar arguments lead to

$$u^{**}(t,\xi) = e^{(k-t)L_1}u(t,\xi) \quad \forall \xi \in \Omega\ . \tag{6.4}$$

and hence

$$b^{**}(t,\xi) = e^{(k-t)L_1}b(t,\xi) \quad \forall \xi \in \partial^y\Omega\ . \tag{6.5}$$

For the transport equation, the above relation translates into $b^{**}(t,x,y) = b(t,x+(k-t)v_1,y)$. For the heat equation, (6.3) tells us to *diffuse* b in the x direction for a period t.

Remarks:

1. As can be seen from (6.4), for both examples the exact solution is recovered (i.e. $u^{**}(k) = u(k)$) provided that the correct boundary values are used.

2. For the transport problem splitting and setting the boundary conditions (6.3,

6.5) is equivalent to the method of characteristics. In fact,

3. For the heat equation, setting the "correct" boundary condition for the first subproblem is problematic, since (6.3) is, of course, an ill-posed system.

4. The "intuitive" boundary conditions $b^* = b^{**} = b$ lead to an $O(k)$ error in (6.1).

6.6 Numerical example

To illustrate the effect of boundary conditions on the accuracy of directional split solution, consider the following 2-D convection-diffusion test problem:

$$c_t = xc_y - yc_x + \frac{1}{1+x^2+y^2}(c_{xx} + c_{yy}), \quad -1 \le x, y \le 1 \ , \quad 0 \le t \le 1 \ . \qquad (6.6)$$

The initial and Dirichlet-type boundary values value are given by the exact solution

$$c_{exact}(t) = e^{4t+x^2+y^2} \ .$$

We discretized the space derivatives using standard central differences on a 61×61 uniform grid with meshsize h:

$$\begin{aligned} \frac{d}{dt}c_{i,j} &= -y_j \frac{c_{i+1,j} - c_{i-1,j}}{2h} + x_i \frac{c_{i,j+1} - c_{i-1,j}}{2h} \\ &\quad + \frac{1}{h^2(1+x_i^2+y_j^2)} \left(c_{i+1,j} + c_{i-1,j} + c_{i,j+1} + c_{i,j-1} - 4c_{i,j}\right) \end{aligned}$$

We integrated this system of ODE with Lsodes (in a classical method of lines approach) and found a relative error in norm 2 of the order 10^{-5}. Then we used a locally one dimensional splitting with a split time step of 1/10 and integrated the subsystems with Lsodes on each subinterval. The Dirichlet boundary conditions for each subsystem were set equal to the exact solution in one experiment and were properly corrected in another; the errors are reported in Figure 6.2 (left and right, respectively). The error plots speak for themselves on the importance of correct

prescription of subsystems boundary conditions.

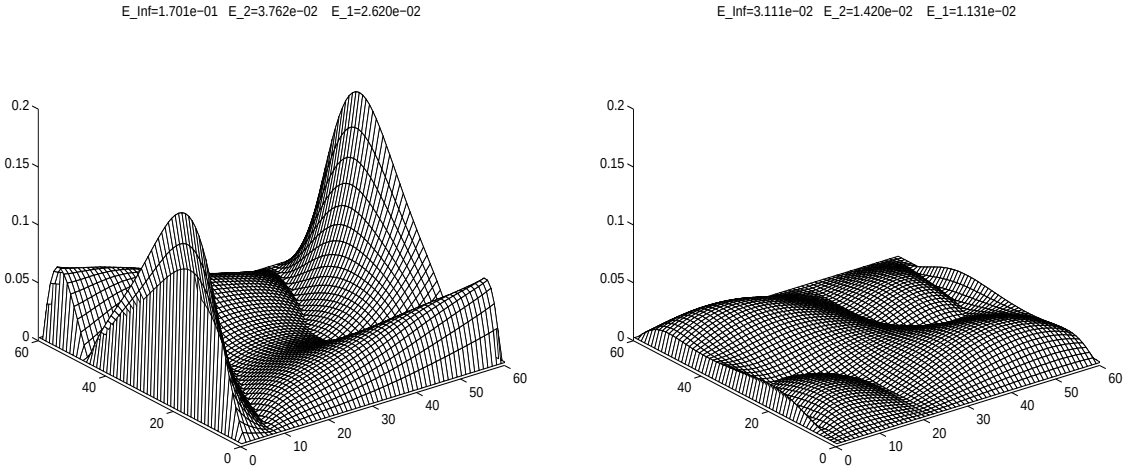

Figure 6.2. Relative errors for the local one dimensional splitting applied to the test problem. The Dirichlet conditions were given by the exact solution, applied directly (left) and corrected (right).

The equation(6.6) can be rewritten in the conservative form as:

$$\begin{aligned} c_t &= \operatorname{div}\left(\begin{bmatrix} -y \\ x \end{bmatrix} c\right) + \operatorname{div}\left(\begin{bmatrix} \frac{1}{1+x^2+y^2} & 0 \\ 0 & \frac{1}{1+x^2+y^2} \end{bmatrix} \cdot \nabla c\right) - \frac{4(x^2+y^2)^2}{(1+x^2+y^2)^2} c \qquad (6.7) \\ &= F(x,y,c,c_x)_x + G(x,y,c,c_y)_y - \frac{4(x^2+y^2)^2}{(1+x^2+y^2)^2} c \ , \end{aligned}$$

where the last term can be thought of as a sink chemical reaction, and the fluxes are given by

$$F(x,y,c,c_x) = \frac{1}{1+x^2+y^2} c_x - yc \qquad (6.8a)$$

$$G(x,y,c,c_y) = \frac{1}{1+x^2+y^2} c_y + xc \ . \qquad (6.8b)$$

For the second numerical experiment we consider flux boundary conditions; more exactly, we prescribe the value of F on $\{(x,y) \in \partial\Omega | x = -1, 1\}$ and the value

of G on $\{(x,y) \in \partial\Omega | y = -1, 1\}$. Replacing the spatial derivatives in the flux expressions by standard one-sided differences one obtains equations from which boundary values can be retrieved. In our numerical experiment we used third order one sided differences. Results are given in figure 6.3.

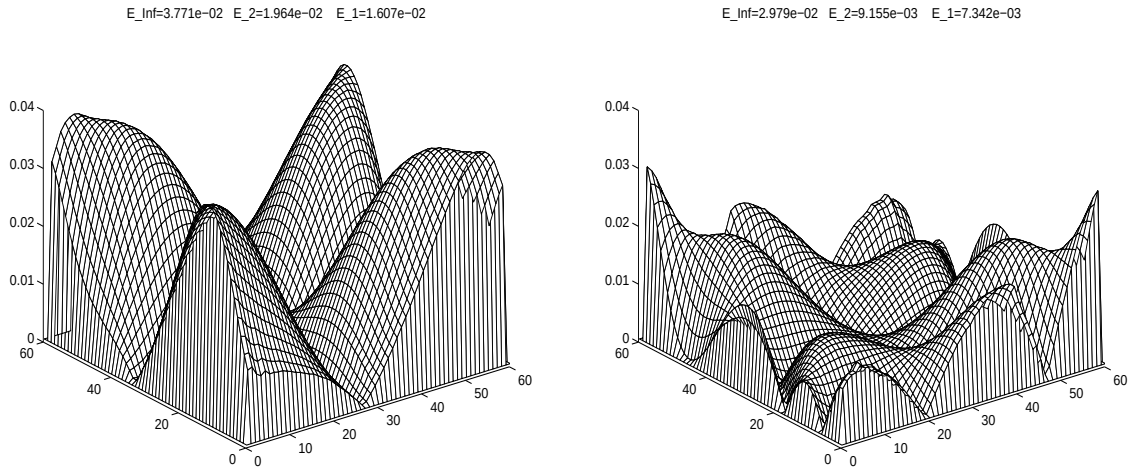

Figure 6.3. Relative errors for the local one dimensional splitting applied to the test problem. The flux boundary conditions conditions were calculated analytically and applied directly (left) and corrected (right).

6.7 Impact of the nonlinear terms

on boundary conditions

Consider the atmospheric advection-diffusion equation

$$\begin{cases} u'(t,\xi) = L(t,\xi)u(t,\xi) + f(t,\xi,u), & \xi \in \Omega \\ B(t,\xi)u(t,\xi) = b(t,\xi), & \xi \in \partial\Omega \\ u(t_0,\xi) = u_0(\xi), & \xi \in \Omega \end{cases} \tag{6.9}$$

where

$$\begin{aligned} L(t,\xi)u &= -\nabla(v(t,\xi)u) + \nabla(K(t,\xi)\nabla u) \\ B(t,\xi)u &= \alpha(t,\xi)\cdot\nabla u + \beta(t,\xi)u\,, \qquad \text{and} \\ L'(t,\xi)u &= -\nabla(v_t(t,\xi)u) + \nabla(K_t(t,\xi)\nabla u) \end{aligned}$$

After a tedious, but straightforward calculation we arrive at

THEOREM 6.7.1.

(i) For the "transport-first" *splitting*

$$\begin{cases} u_t^*(t,\xi) = L(t,\xi)u^*(t,\xi)\,, \quad u^*(0,\xi) = u(0,\xi)\,, \quad B(t,\xi)u^* = b^* \text{ on } \partial\Omega\,; \\ u_t^{**}(t,\xi) = f(t,\xi,u^{**}(t,\xi))\,, \quad u^{**}(0,\xi) = u^*(k,\xi) \end{cases}$$

the solution of the transport subproblem can be expressed in terms of the global solution as

$$\begin{aligned} u^*(t,\xi) &= u(t,\xi) - tf(0,\xi,u(0,\xi)) \\ &\quad -\frac{t^2}{2}\{L(0,\xi)f(0,\xi,u(0,\xi)) + f_t(0,\xi,u(0,\xi)) \\ &\quad +f_u(0,\xi,u(0,\xi))L(0,\xi)u(0,\xi) + f_u(0,\xi,u(0,\xi))f(0,\xi,u(0,\xi))\} + O(t^3) \end{aligned} \tag{6.10}$$

(ii) For the "chemistry-first" *splitting*

$$\begin{cases} u_t^*(t,\xi) = f(t,\xi,u^*(t,\xi))\,, \quad u^*(0,\xi) = u(0,\xi) \\ u_t^{**}(t,\xi) = L(t,\xi)u^{**}(t,\xi)\,, \quad u^{**}(0,\xi) = u^*(k,\xi)\,, \quad B(t,\xi)u^{**} = b^{**} \text{ on } \partial\Omega\,; \end{cases}$$

the solution of the transport subproblem can be expressed in terms of the global solution as

$$\begin{aligned} u^{**}(t,\xi) &= u(t,\xi) + (k-t)f(0,\xi,u(0,\xi)) \\ &\quad +\frac{k^2-t^2}{2}\{f_t(0,\xi,u(0,\xi)) + f_u(0,\xi,u(0,\xi))u(0,\xi)\} \end{aligned} \tag{6.11}$$

$$-\frac{t^2}{2}\{f_u(0,\xi,u(0,\xi))L(0,\xi)u(0,\xi)+L(0,\xi)f(0,\xi,u(0,\xi))\}+O(t^3)$$

(iii) For the symmetric *splitting*

$$\begin{cases} u_t^* = L(t,\xi)u^*, & u^*(0,x) = u(0,x), & B(t,\xi)u^* = b^* & \text{on } \partial\Omega\ ; \\ u_t^{**} = f(t,\xi,u^{**}), & u^{**}(0,x) = u^*(k/2,x), & & \\ u_t^{***} = L(t,\xi)u^{***}, & u^{***}(0,x) = u^{**}(k,x), & B(t,\xi)u^{***} = b^{***} & \text{on } \partial\Omega\ ; \end{cases}$$

the solution of the transport subproblem can be expressed in terms of the global solution as

$$\begin{aligned} u^*(t,\xi) &= u(t,\xi)-tf(0,\xi,u(0,\xi)) \\ &\quad -\frac{t^2}{2}\{L(0,\xi)f(0,\xi,u(0,\xi))+f_t(0,\xi,u(0,\xi)) \\ &\quad +f_u(0,\xi,u(0,\xi))L(0,\xi)u(0,\xi) \\ &\quad +f_u(0,\xi,u(0,\xi))f(0,\xi,u(0,\xi))\}+O(t^3) \\ u^{***}(t+\frac{k}{2},\xi) &= u(t+\frac{k}{2},\xi)+(\frac{k}{2}-t)(f(0,\xi,u(0,\xi)) \\ &\quad +kf_t(0,\xi,u(0,\xi))+kf_u(0,\xi,u(0,\xi))u(0,\xi) \\ &\quad -\frac{(\frac{k}{2}-t)^2}{2}(L'u+L^2u+Lf+f_t+f_uLu+f_uf) \\ &\quad +\frac{t^2}{2}(L'(0,\xi)u(0,\xi)+L^2(0,\xi)u(0,\xi))+O(k^3) \end{aligned} \tag{6.12}$$

Remarks:

1. For Dirichlet boundary conditions, the "transport-first" splitting, relation (6.11) shows that one needs to "undo" the chemistry in order to obtain consistent boundary conditions.

2. For Dirichlet boundary conditions, the "chemistry-first" splitting, relation (6.12) shows that we need to correct the boundary conditions by an extra "chemistry prediction".

3. For flux boundary conditions, the prescribed boundary flux need to be corrected in all cases in order to obtain consistent boundary conditions.

6.8 Extended linearized alternating direction implicit methods

Consider the following advection - diffusion - reaction system in $\Omega \in \Re^3$:

$$\frac{\partial c_i}{\partial t} = Lc_i(x,t) + g_i(c_1, ..., c_p, x, t) \tag{6.13}$$

plus some initial and boundary conditions where

$$Lc = \nabla(u \cdot c) + \nabla(K \cdot \nabla c)$$

is the elliptic operator describing the advection and the diffusion and g represent the chemical transformations, depositions, emissions. c_1 to c_p are the concentrations of the p reacting species. After the semidiscretization in space on a N-point grid Ω_d we get:

$$\frac{dc}{dt} = f(c,t) + g(c,t) \ , \ c(t_0) = c_0 \tag{6.14}$$

where $c \in \Re^{N \times p}$ and f describes both L and the boundary conditions. This system is usually solved for $t \in [t_0, t_1]$ using operator splitting:

$$\frac{dc^*}{dt} = f(c^*,t) \ , \ c^*(t_0) = c_0 \tag{6.15a}$$

$$\frac{dc}{dt} = g(c,t) \ , \ c(t_0) = c^*(t_1). \tag{6.15b}$$

Solving (6.15b) is equivalent to solving transport only. Since the species are individually transported (there is no coupling among species due to L) (6.15b) is equivalent to solving p systems of dimension N (there is a coupling, given by space discretization formula, between the values of c_i at the N points of Ω_d). Similarly, since the chemistry is local, g is not coupled in Ω_d and hence solving (6.15b) is equivalent to solving N systems of dimension p. With an implicit time stepping

method, the computational effort for (6.14) is $O(N^3p^3)$, while for (6.15b - 6.15b) is $O(Np^3 + pN^3)$, whence the computational advantage.

6.9 The methods

We consider systems of the form (6.14)

$$\frac{dy}{dt} = f(y) + g(y) \ , \ y(t_0) = y_0 \tag{6.16}$$

with the property that f and g can each be decoupled, but along different coordinates.

The idea is now to employ W-methods for time stepping and to choose an appropriate approximation for the Jacobian. This choice will enable us to make a "decoupling" of the system into the subsystems f and g inside the method (unlike the operator splitting, where the decoupling is made independent of the numerical method).

An s-stage W-method is defined as:

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i, \tag{6.17a}$$

$$k_i = h(f+g)(y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j) + hA \sum_{j=1}^{i} \gamma_{ij} k_j, \tag{6.17b}$$

where s and the formula coefficients b_i, α_{ij} and γ_{ij} are chosen to obtain a desired order of consistency and stability for stiff problems. A is any approximation of the Jacobian; the key point is, of course, that the order of the method is independent of the choice of A. The only restriction is the one imposed by numerical stability. It is easy to see that computing k_i requires the numerical solution of a system of the form

$$(I - h\gamma A) \cdot k_i = b$$

If the exact Jacobian is used ($A = J$) we are in the framework of classical Rosenbrock methods. For (6.14) the Jacobian reads

$$J = F + G$$

where $F = f_y$, $G = g_y$, so one needs to "invert" $(I - h\gamma F - h\gamma G)$.

By the decoupling property, it is relatively easy to "invert" $(I - h\gamma F)$ and $(I - h\gamma G)$ (operator splitting also takes advantage of this property). Hence, **the idea is to define A as follows**:

$$I - h\gamma A = (I - h\gamma F)(I - h\gamma G) \tag{6.18a}$$

$$A = J - h\gamma FG \tag{6.18b}$$

The hope is that, since both subprocesses are stable (see 6.15b, 6.15b) the resulting scheme will also be stable.

The computational advantage is clear:

$$(I - h\gamma A)^{-1} = (I - h\gamma G)^{-1}(I - h\gamma F)^{-1}$$

In other words, at each step we treat implicitly subprocess f, then subprocess g, with a computational load comparable to the load of operator splitting. The coupling is done by the fact that in the right hand side of (6.17b) f and g are treated together, while the main computational burden, viz. the decomposition of $(I - h\gamma A)$ is managed by splitting into subprocesses.

6.10 Conservation properties

Many systems evolve along trajectories which conserve linear invariants, $w^T y(t) = const, \forall t$. An example is total mass which, for balanced boundary fluxes and in absence of emissions and of depositions is preserved. We have that $w^T(f + g) = 0$ and also $w^T(F + G) = 0$.

A desirable property for a numerical method is to structurally preserve the linear invariants of the system. Multistep, Runge-Kutta and Rosenbrock are example of such conservative methods.

Now suppose that each of the subprocesses f, g are conservative, i.e. $w^T f = 0$ and $w^T g = 0$. For example, transport is mass conservative, as well as the chemical reactions. Then we have that $w^T F = w^T G = 0$ and

$$w^T A = w^T F + w^T G - h\gamma w^T FG = 0$$

From (6.17b) we have

$$w^T k_i = h w^T f(y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j) + h w^T A \sum_{j=1}^{i} \gamma_{ij} k_j = 0$$

hence from (6.17b) we obtain the structural conservativity:

$$w^T y_{n+1} = w^T y_n + \sum_{i=1}^{s} b_i w^T k_i = w^T y_n$$

6.11 Implicit - explicit ELADI

Implicit-explicit multistep methods (Crouzeix) are well-known. A similar approach can be obtained in the ELADI framework.

Suppose subprocess g is "fast" and requires an implicit treatment, while subprocess f is "slow" and can be treated explicitly.

Then we can choose $A = G$.

In the absence of the "fast" component, the "slow" part f is integrated with the explicit Runge-Kutta method:

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i, \tag{6.19a}$$

$$k_i = h f(y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j) \tag{6.19b}$$

6.11.1 Example: TVD advection - reaction

Suppose g describes chemistry plus diffusion, while f is the discretization of the advection operator. One desirable property of the advection schemes is to be TVD. Once a TVD discretization was chosen, we require that the restriction placed by the time integration on the CFL condition is minimal.

For 2 stages, order 2 (6.19b - 6.19b) reads:

$$\begin{aligned} u &= y_0 + h\alpha_{21} f(y_0) \\ y_1 &= y_0 + hb_1 f(y_0) + hb_2 f(u) \end{aligned}$$

We will continue with an analysis "a la Shu and Osher". Let $\beta_1 + \beta_2 = 1$. Then

$$y_1 = \beta_1(y_0 + \frac{b_1 - \alpha_{21}\beta_2}{\beta_1} hf(y_0)) + \beta_2(u + \frac{b_2}{\beta_2} hf(u))$$

CFL restriction is:

$$\lambda \leq \lambda_0 \min\left(\frac{\beta_1}{b_1 - \alpha_{21}\beta_2}\ ,\ \frac{\beta_2}{b_2}\right)$$

One possible solution, for a family of W-methods whose explicit correspondents maintain CFL = 1 is: $b_1 = b_2 = 1/2$, $\alpha_{21} = 1$, $\gamma_{21} = -2\gamma$. Of course, we may choose $\gamma = (2 - \sqrt{2})/2$ for L-stability in the eventuality that an exact Jacobian is used.

6.12 Benefits for atmospheric modeling

In the numerical simulation of atmospheric transport-chemistry processes, a major task is the integration of the stiff systems of ordinary differential equations describing the chemical transformations. Usually this is done using operator splitting. Two major drawbacks of this technique are known, but usually accepted:

- Separating the chemical transformations from vertical diffusion, for example, may lead to large numerical errors.

- Each transport step modifies the chemical concentrations; since the chemical rate equations are stiff, and the transport step modifies the pseudo-equilibrium of the fast species, every restart of the chemical integrator is followed by a transient phase. This transient is non-physical, on one hand, and consumes about 50% of the chemistry computational time, on the other (since accuracy restriction imposes very small step-sizes).

Both these drawbacks can be avoided with ELADI. However, memory restrictions may impose to still decouple horizontal transport from vertical transport (but compute the chemistry and the vertical transport together).

One thought advantage of classical operator splitting is that each subprocess can be treated with an appropriate step-size (in particular chemistry can be integrated with much smaller step-sizes than required for transport). In our experience, outside the transient, stiff ODE solvers applied to chemical rate equations can take steps of 1 hour or more. The very small steps usually assumed by chemical integrators are due to the frequent restarts (and transients). Thus, we think that eliminating the restarts will make feasible to integrate chemistry and transport with the same step (see Hunsdorfer-Verwer, implicit- explicit approach).

6.13 About stability

In this section we will discuss the stability of the 1-stage, order 1 ELADI formula. Consider the test problem

$$y' = \lambda_1 y + \lambda_2 y$$

and apply the method corresponding to linearized Euler. Denoting $z_i = h\lambda_i$ we obtain:

$$y_1 = \left(\frac{1 + z_1 z_2}{(1 - z_1)(1 - z_2)} \right) y_0$$

The following lemma assures us that the method is stable.

LEMMA 6.13.1. *The function*

$$R(z_1, z_2) = 1 + \frac{z_1 + z_2}{(1 - z_1)(1 - z_2)}$$

has $|R| \leq 1 \ \forall z_i \in C^-$.

Proof.

$$|R|^2 = \left| \frac{1 + z_1 z_2}{(1 - z_1)(1 - z_2)} \right|^2 = \frac{N}{D}$$

Let $z_i = a_i + b_i, i = 1, 2$.

$$D - N = (a_1 + a_2)^2 + (b_1 + b_2)^2 - 2a_1(1 + a_2^2 + b_2^2) - 2a_2(1 + a_1^2 + b_1^2)$$

from which conclusion follows, as $a_i \leq 0$. Moreover, $|R| = 1 \Leftrightarrow z_1 = z_2 = 0$. □

For a strictly parabolic problem the one dimensional linearized model is

$$y' = \sum_i \lambda_i y$$

with real negative $\lambda_i < 0$. For such a problem the simple scheme is also stable, as shown by the following

LEMMA 6.13.2. *Let* $z_k, \ k = 1, \ldots, N$ *be nonpositive real numbers. Then*

$$0 \leq 1 + \frac{\sum_{k=1}^N z_k}{\prod_{k=1}^N (1 - z_k)} \leq 1$$

Proof. The second inequality is obvious. For the first one we note that

$$\prod_{k=1}^N (1 - z_k) = 1 - \sum_{k=1}^N z_k + P \geq - \sum_{k=1}^N z_k$$

since P is a nonnegative quantity because $z_k \leq 0 \ \forall k$. □

An extension of this analysis to higher order eladi formulas and to test problems with multiple splittings will be the subject of further investigations.

CHAPTER 7
CONCLUSIONS

7.1 Summary and concluding remarks

In air quality models the integration of stiff ODE's modelling chemical interactions is a major computational task: it counts for up to 90% of the total CPU time. The chemical integration has to be carried out for every grid cell and for every operator split interval; frequent restarts mean frequent transients, which substantially reduce the average step size taken by numerical integrators.

Due to critical need for speed, many specially tailored methods have been proposed. They are mainly of explicit type. The QSSA idea is extended and several new methods in this class are proposed.

QSSA-based algorithms are explicit methods and yet they enjoy a remarkable stability. They behave like implicit methods although their evaluation formulae is explicit. In this thesis QSSA type methods are investigated theoretically, and their excellent stability properties rigorously explained. In [92] the local truncation error for Plain QSSA scheme is shown to be only $O(h)$ for the components with small lifetimes $\tau_i \ll h$. However, numerical experiments have shown that the QSSA solutions still converge to the exact solution. The fact that the local order reduction is not felt globally is in line with the theoretical convergence analysis presented here.

Although the QSSA methods are in general quite inaccurate, QSSA-based methods preserve quite well the overall behaviour of the solution. This explains why these methods have been successfully employed for many years for problems

where relatively large errors are accepted and small computing times are desired.

One of the main contributions of this work is to show that implicit methods are very competitive, if they are carefully designed and rely on special sparse linear algebra routines; in fact, they outperform different QSSA solvers, often by a wide gap.

New fast implicit methods were designed for being used in air quality models. They are of Runge-Kutta-Rosenbrock type, and thus only linearly implicit. An iteration procedure is therefore not needed for computing the numerical solution - this fact has a positive effect for the application, since:

- the computational time is lowered;
- enables cellwise vectorization;
- does not harm the linear stability of the methods (fact confirmed by experiments).

In addition, methods in this family preserve the linear invariants of the system - like total mass and mass of families of species. Thus no additional effort (e.g. lumping) is necessary to impose mass conservation. Being of one-step type, these solvers rapidly adjust the step size after an operator split restart; thus they vcan deal with transients very efficiently. Ros3 is an L-stable, 3-stage formula of order 3, with an embedded A-stable method of order 2. Rodas3 is a stiffly accurate pair of order 3(2), using 4 stages but only 3 function evaluations.

All the new implicit methods rely on very efficient sparse linear algebra routines developed in Chapter 3. The routine "Doolittle-2" uses a row-wise LU decomposition algorithm without pivoting, and is paired by loop-free substitution routines. Indirect addressing and data moving (at run time) are thus minimized;

overall this sparse scheme is about 3 times faster than Harwell's MA28 on the test problems of interest.

The answer to the question of which stiff integrator is "the best" for being used in air quality models depends on a multitude of factors, some of the most important being the specific chemical mechanism employed, the desired accuracy level and the hardware on which the code runs. In chapter 4 a collection of benchmark problems is put together, spanning a wide range of chemical mechanisms and chemical conditions. The new QSSA solvers, the new Rosenbrock solvers as well as a number of methods from the literature are carefully tested. The main conclusions of this benchmark are

- For all the test problems considered here and within 4 digits of accuracy the Rosenbrock solvers provide the most cost-effective solutions. For lower accuracies of practical interest Rodas3 and Ros3 are usually the best. As expected, for higher accuracies Rodas is mostly competitive;

- Robustness and ease of use are very important since in actual 3D transport a subtle tuning of the ODE code is cumbersome due to the large variety of conditions that will occur at different grid points. In this respect the Rosenbrock solvers are advocated as well. With the preprocessor KPP at hand, they are easy to use.

- Among explicit codes, Twostep performs the best. It is however outperformed by the Rosenbrock methods. QSSA type solvers show good stability, but their lack of accuracy cause them to fall behind.

The cumbersome problem of translating a chemical scheme into code is addressed with the Kinetic Preprocessor (KPP). KPP not only generates the function

and the associated Jacobian, but also reorders the species and builds sparse data structures, overall preparing a very efficient and sparse matrix routine, adjusted to the desired chemical mechanism. KPP links this info with the numerical integrator of choice, the result being a ready to run C or Fortran code.

The issue of boundary conditions in regional air quality models is discussed in Chapter 6. It is shown that the splitting between chemistry and transport requires a correction of the imposed boundary conditions - essentially by "undoing" (integrating backwards) the chemical subsystem; dimensional splitting also requires boundary corrections, which can be analysed along the lines of LeVeque. Several ideas of time integration without splitting, but with the efficiency of splitting, are exposed. The "extended linearized" ADI methods require no correction for the boundary fluxes; however, work still needs to be done to rigorously prove the stability of these methods.

7.2 Future research

The development of chemical schemes for heterogeneous chemistry components of air quality modeling puts new demands on the quality of numerical schemes. For these models the traditional integrators (e.g. QSSA) may prove useless. One example is the gas-and-liquid system from Chapter 4 on which all explicit integrators fail (because of stability problems). The new multiple phase chemical models raise new issues. For example, modeling the gas-aerosol equilibria by algebraic relations transforms the ODE (ordinary differential equation) system into a DAE (differential-algebraic equation) system and special techniques to numerically simulate the time evolution of such a constrained system has to be developed for this

application. In particular we need to design special, highly efficient DAE solvers for use in heterogeneous chemistry modeling.

We think that further improvements of the chemical integrators are needed and possible. Future investigation is to be carried out along the following directions:

1. The use of approximate Jacobians in implicit integrators. By replacing the Jacobian with a matrix of higher sparsity, the cost of linear algebra is further decreased. Stability, accuracy and conservativity aspects of such methods are to be analyzed.

2. At every operator-splitting restart, the numerical integrator has to resolve the transient part, which implies the use of small step-sizes and hence reduced efficiency. We plan to investigate ways of reducing this overhead by properly pre-adjusting the initial values and by using fully implicit methods in the first several steps.

3. Partitioned methods (either partitioning the reactions into fast and slow ones, or partitioning the species into stiff and nonstiff ones) offer a possibility to further reduce computational time (see for example [80] and [97, 83]). This may be done by efficiently combining an implicit method for the fast (stiff) part with an explicit method for the slow (non-stiff) part. Possible candidates are Partitioned Runge-Kutta methods (PRK) from [47], Implicit-Explicit Multistep methods (IEM) from [4] and SPARK methods from [55].

4. Positivity preserving. This aspect is important for preserving the stability of the chemical system itself, but has not been fully explored so far.

5. Use of differential algebraic solvers for problems involving multiple phase equilibria.

For the solution of the transport problem, new research directions are sought for

- estimation and control the splitting error; one possible way is to use extrapolation;
- numerical methods that use an "internal splitting", thus requiring no boundary corrections. ELADI or implicit-explicit methods of Crouzeix are such possible examples.
- monotone implicit methods that will allow the discretization of convection and diffusion simultaneously.

APPENDIX A
CHEMICAL MECHANISMS

A.1 The CBM-IV mechanism

```
{ 1.} NO2 + hv =  NO + O    :  8.89E-3*SUN;

{ 2.} O {+ O2 + M} = O3     :  ARR(1.4E+3, 1175);

{ 3.} O3 + NO = NO2  :  ARR(1.8E-12, -1370);

{ 4.} O + NO2 = NO   :  9.3E-12;

{ 5.} O + NO2   = NO3:  ARR(1.6E-13, 687);

{ 6.} O + NO   =  NO2:  ARR(2.2E-13, 602);

{ 7.} O3 + NO2 =  NO3:  ARR(1.2E-13, -2450);

{ 8.} O3 + hv = O    :  4.0E-2*RCONST(1);

{ 9.} O3 + hv = O1D  :  2.8E-3*RCONST(1);

{10.} O1D   = O      :  ARR(1.9E+8, 390) ;


{11.} O1D + H2O = 2OH:  2.2E-10;

{12.} O3 + OH = HO2  :  ARR(1.6E-12, -940);

{13.} O3 + HO2 = OH  :  ARR(1.4E-14, -580);

{14.} NO3 + hv = 0.89 NO2 + 0.89 O
                 + 0.11 NO  :  15.5*RCONST(1) ;

{15.} NO3 + NO = 2 NO2      :  ARR(1.3E-11, 250);

{16.} NO3 + NO2 = NO + NO2  :  ARR(2.5E-14, -1230);

{17.} NO3 + NO2   =  N2O5   :  ARR(5.3E-13, 256);
```

```
{18.} N2O5 + H2O = 2 HNO3   :  1.3E-21;
{19.} N2O5   =  NO3 + NO2   :  ARR(3.5E+14, -10897);
{20.} 2 NO  =  2 NO2 :  ARR(1.8E-20, 530);


{21.} NO + NO2 + H2O = 2 HONO         :  4.4E-40;
{22.} OH + NO   =  HONO     :  ARR(4.5E-13, 806);
{23.} HONO + hv =  OH + NO  :  0.17*RCONST(1);
{24.} OH + HONO =  NO2      :  6.6E-12;
{25.} 2 HONO = NO + NO2     :  1.0E-20;
{26.} OH + NO2   =  HNO3    :  ARR(1.0E-12, 713);
{27.} OH + HNO3   =  NO3    :  ARR(5.1E-15, 1000);
{28.} HO2 + NO = OH + NO2   :  ARR(3.7E-12, 240);
{29.} HO2 + NO2   =  PNA    :  ARR(1.2E-13, 749);
{30.} PNA   = HO2 + NO2     :  ARR(4.8E+13, -10121);

{31.} OH + PNA = NO2 :  ARR(1.3E-12, 380);
{32.} 2 HO2 = H2O2   :  ARR(5.9E-14, 1150) ;
{33.} 2 HO2 + H2O = H2O2    :  ARR(2.2E-38, 5800);
{34.} H2O2 + hv = 2 OH      :  7.1E-4*RCONST(1);
{35.} OH + H2O2 = HO2:  ARR(3.1E-12, -187);
{36.} OH + CO  = HO2 :  2.2E-13;
{37.} HCHO + OH  =  HO2 + CO    :  1.0E-11;
{38.} HCHO + hv {+ 2 O2} = 2 HO2 + CO :  3.2E-3*RCONST(1);
{39.} HCHO +  hv = CO:  4.2E-3*RCONST(1);
```

```
{40.} HCHO + O = OH + HO2 + CO        :  ARR(3.0E-11, -1550);


{41.} HCHO + NO3  = HNO3
                        + HO2 + CO    :  6.3E-16;
{42.} ALD2 + O  =  C2O3 + OH:  ARR(1.2E-11, -986);
{43.} ALD2 + OH = C2O3      :  ARR(7.0E-12, 250);
{44.} ALD2 + NO3  = C2O3 + HNO3 :  2.5E-15 ;
{45.} ALD2 + hv {+ 2 O2} = HCHO + XO2
                         + CO + 2 HO2 :  4.5E-4*RCONST(1);
{46.} C2O3 + NO  = HCHO + XO2
                       + HO2 + NO2    :  ARR(5.4E-12, 250);
{47.} C2O3 + NO2 = PAN      :  ARR(8.0E-20, 5500);
{48.} PAN = C2O3 + NO2      :  ARR(9.4E+16, -14000);
{49.} 2 C2O3 = 2 HCHO + 2 XO2 + 2 HO2 : 2.0E-12 ;
{50.} C2O3 + HO2 = 0.79 HCHO
   + 0.79 XO2 + 0.79 HO2 + 0.79 OH    :  6.5E-12;


{51.} OH = HCHO + XO2 + HO2 :  ARR(1.1E+2, -1710);
{52.} PAR + OH =  0.87 XO2 + 0.13 XO2N
                  + 0.11 HO2 + 0.11 ALD2
                  + 0.76 ROR - 0.11 PAR        :  8.1E-13;
{53.} ROR = 1.1 ALD2 + 0.96 XO2 + 0.94 HO2 + 0.04 XO2N
            + 0.02 ROR - 2.10 PAR     :  ARR(1.0E+15, -8000);
{54.} ROR = HO2      :  1.6E+3;
{55.} ROR + NO2 =  PROD     : 1.5E-11 ;
```

```
{56.} O + OLE = 0.63 ALD2 + 0.38 HO2
                + 0.28 XO2 + 0.3 CO
                + 0.2 HCHO + 0.02 XO2N
                + 0.22 PAR + 0.2 OH    :  ARR(1.2E-11, -324);
{57.} OH + OLE = HCHO + ALD2 + XO2
                 + HO2 - PAR:  ARR(5.2E-12, 504);
{58.} O3 + OLE = 0.5 ALD2 + 0.74 HCHO
                 + 0.33 CO + 0.44 HO2
                 + 0.22 XO2
                 + 0.1 OH - PAR       :   ARR(1.4E-14, -2105) ;
{59.} NO3 + OLE = 0.91 XO2 + HCHO
                  + ALD2 + 0.09 XO2N
                  + NO2 - PAR         :  7.7E-15;
{60.} O + ETH = HCHO + 0.7 XO2 + CO + 1.7 HO2 + 0.3 OH
        :  ARR(1.0E-11, -792);
{61.} OH + ETH = XO2 + 1.56 HCHO + HO2 + 0.22 ALD2
        :  ARR(2.0E-12, 411) ;
{62.} O3 + ETH = HCHO + 0.42 CO + 0.12 HO2
       :  ARR(1.3E-14, -2633);
{63.} OH + TOL = 0.08 XO2 + 0.36 CRES
                 + 0.44 HO2 + 0.56 TO2   :  ARR(2.1E-12, 322);
{64.} TO2 + NO =  0.9 NO2 + 0.9 OPEN + 0.9 HO2   :  8.1E-12;
{65.} TO2 = HO2 + CRES         :  4.2;
{66.} OH + CRES = 0.4 CRO + 0.6 XO2 + 0.6 HO2 + 0.3 OPEN:4.1E-11;
{67.} NO3 + CRES = CRO + HNO3  :  2.2E-11;
```

```
{68.} CRO + NO2 = PROD          :  1.4E-11;
{69.} OH + XYL = 0.7 HO2 + 0.5 XO2 + 0.2 CRES + 0.8 MGLY
                + 1.10 PAR + 0.3 TO2    :  ARR(1.7E-11, 116);
{70.} OH + OPEN = XO2 + C2O3 + 2 HO2 + 2 CO + HCHO:3.0E-11;

{71.} OPEN + hv = C2O3 + CO + HO2        :  6.0E-3*RCONST(1);
{72.} O3 + OPEN = 0.03 ALD2 + 0.62 C2O3
                  + 0.7 HCHO + 0.03 XO2 + 0.69 CO
                  + 0.08 OH + 0.76 HO2 + 0.2 MGLY
                                  : ARR(5.4E-17,-500);
{73.} OH + MGLY =  XO2 + C2O3  : 1.7E-11;
{74.} MGLY + hv = C2O3 + CO + HO2        : 1.86E-2*RCONST(1) ;
{75.} O + ISOP =  0.6 HO2 + 0.8 ALD2 + 0.55 OLE + 0.5 XO2
                  + 0.5 CO + 0.45 ETH + 0.9 PAR   :  1.8E-11 ;
{76.} OH + ISOP = HCHO + XO2 + 0.67 HO2
                  + 0.4 MGLY + 0.2 C2O3
                  + ETH + 0.2 ALD2 + 0.13 XO2N :   9.6E-11 ;
{77.} O3 + ISOP = HCHO + 0.4 ALD2 + 0.55 ETH + 0.2 MGLY
                  + 0.06 CO + 0.1 PAR + 0.44 HO2 + 0.1 OH :
                  1.2E-17;
{78.} NO3 + ISOP =  XO2N       :  3.2E-13;
{79.} XO2 + NO = NO2           :  8.1E-12;
{80.} 2 XO2 =  PROD            :  ARR(1.7E-14, 1300);
{81.} XO2N + NO =  PROD        :  6.8E-13;
```

A.2 The Lloyd- Atkinson- Lurmann mechanism

```
{1} NO2 + hv = NO + O3 :              9.236E-3*SUN;

{2} NO + O3 = NO2 + O2 :              ARR(2.2E-12,-1430);

{3} NO2 + O3 = NO3 + O2 :             ARR(1.2E-13,-2450);

{4} NO + NO3 = 2NO2 :                 ARR(1.7E-11,150);

{5} NO2 + NO3 = N2O5 :                1.327E-12;

{6} N2O5 = NO2 + NO3 :                2.013E-2;

{7} NO2 + NO3 = NO + NO2 + O2 :       ARR(2.5E-14,-1230);

{8} NO3 + hv = 0.15NO + 0.85NO2 + 0.85O3 + O2 :
                                      3.039E-2*SUN;

{9} NO3 + HO2 = HNO3 + O2 :           0;

{10} O3 + H2O + hv = 2OH :            9.34E-22*RCONST(1);

{11} NO + OH = HONO :                 7.35E-12;

{12} HONO + hv = NO + OH :            1.893E-3*SUN;

{13} NO2 + OH = HNO3 :                1.2794E-11;

{14} HNO3 + hv = NO2 + OH :           6.684E-7*SUN;

{15} HNO3 + OH = NO3 + H2O :          1.629E-13;

{16} N2O5 + H2O = 2HNO3 :             0;

{17} CO + OH = HO2 + CO2 :            2.22E-13;

{18} O3 + OH = HO2 + O2 :             ARR(1.6E-12,-1000);

{19} NO + HO2 = NO2 + OH :            ARR(3.7E-12,240);

{20} NO2 + HO2 = HNO4 :          1.333E-12; {1.785E-12;}
```

```
{21} HNO4 = NO2 + HO2 :          0.02313;
{22} O3 + HO2 = OH + 2O2 :       ARR(1.4E-14,-600);
{23} HO2 + HO2 = H2O2 + O2 :     5.816E-12;
{24} H2O2 + hv = 2OH :           7.829E-6*SUN;
{25} H2O2 + OH = HO2 + H2O :     1.57E-12;
{26} NO2 + H2O = HONO + HNO3 - NO2 :   4.00E-24;
{27} HNO4 + hv = NO2 + HO2 :     1.11E-5*SUN;
{28} HNO4 + OH = NO2 + H2O + O2 :ARR(1.3E-12,380);
{29} SO2 + OH = SO4 + HO2 :      1.E-20;
{30} HCHO + hv = 2HO2 + CO :     1.569E-5*SUN;
{31} HCHO + hv = CO + H2 :       6.063E-5*SUN;
{32} HCHO + OH = HO2 + CO + H2O :1.00E-11;
{33} HCHO + HO2 = AHO2 :         1.00E-14;
{34} AHO2 + NO = ACO2 + HO2 + NO2 :     ARR(4.2E-12,180);
{35} AHO2 + HO2 = ACO2 + H2O + O2 :     2.00E-12;
{36} 2AHO2 = ACO2 + 2HO2 + 2O2 : 1.00E-13;
{37} ACO2 + OH = HO2 + H2O + CO2 :      3.20E-13;
{38} NO3 + HCHO = HNO3 + HO2 +CO :      6.00E-16;
{39} ALD2 + OH = MCO3 + H2O :    ARR(6.9E-12,250);
{40} ALD2 + NO3 = HNO3 + MCO3 :  2.70E-15;
{41} ALD2 + hv = MO2 + HO2 + CO :2.383E-6*SUN;
{42} ALD2 + hv = CH4 + CO :      6.063E-5*SUN;
{43} MCO3 + NO2 = PAN :          4.70E-12;
{44} PAN = MCO3 + NO2 :          ARR(2.2E+16,-13435);
{45} MCO3 + NO = MO2 + NO2 + CO2 : ARR(4.2E-12,180);
```

```
{46} MO2 + NO = HCHO + NO2 + HO2 : ARR(4.2E-12,180);

{47} CH4 + OH = MO2 + H2O :      6.35E-15;

{48} C2H6 + OH = ETO2 + H2O :    ARR(1.7E-11,-1232);

{49} ETO2 + NO = ALD2 + HO2 + NO2 :   ARR(4.2E-12,180);

{50} C3H8 + OH = R3O2 :          ARR(1.18E-11,-679);

{51} R3O2 + NO = 0.03R3N2 + 0.46ALD2 + 0.97NO2 +

             0.97HO2 + 0.49KET :  ARR(4.2E-12,180);

{52} ALKA + OH = RAO2 :          ARR(2E-11,-500);

{---- IN THE NEXT REACTION

  BETA 1...9 WHERE REPLACED BY NUMBERS,

  AS READ FROM LILING STEM CODE ----}

{53} RAO2 + NO = 0.9261NO2 - 0.1892NO + 0.263RAN2 +

             1.0482ALD2 + 0.3KET +

             0.1879ETO2 + 0.1116MO2 +

             0.28HO2 + 0.1057R3O2 + 0.06RAO2 :ARR(4.2E-12,180);

{54} ALKA + NO3 = HNO3 + RAO2 :  4.00E-17;

{55} RAN2 + OH = RAN1 + H2O :    2.00E-12;

{56} RAN1 + NO = 2.5NO2  - 0.5NO + 0.8HCHO

                    + 2.1ALD2 : ARR(4.2E-12,180);

{57} MO2 + MO2 = 1.4HCHO + 0.8HO2 + O2 :  ARR(1.5E-13,220);

{58} 2ETO2 = 1.6ALD2 + 1.2HO2 :  5.00E-14;

{59} R3O2 + R3O2 = 1.9ALD2 + 0.28KET + 0.37HO2 :5.00E-14;

{60} HO2 + MO2 = ROOH + O2 :     3.00E-12;

{61} HO2 + ETO2 = ROOH + O2 :    3.00E-12;

{62} HO2 + R3O2 = ROOH + O2 :    3.00E-12;
```

```
{63} HO2 + RAO2 = ROOH + O2 :    3.00E-12;

{64} HO2 + MCO3 = ROOH + O2 :    3.00E-12;

{65} KET + OH = KO2 :            ARR(1.2E-11,-890);

{66} KO2 + NO = 0.05RAN2 + 0.95NO2 + 0.94ALD2 +
           0.94MCO3 :            ARR(4.2E-12,180);

{67} KET + hv = MCO3 + ETO2 + H2O :      2.401E-6*SUN;

{68} KET + NO3 = HNO3 + KO2 :    7.00E-16;

{69} KO2 + HO2 = MGLY + MO2 + H2O :  3.00E-12;

{70} ETHE + OH = EO2 :           ARR(1.66E-12,474);

{71} EO2 + NO = NO2 + 2.0HCHO + HO2 :  ARR(4.2E-12,180);

{72} ALKE + OH = PO2 :           ARR(4.1E-12,537);

{73} PO2 + NO = NO2 + ALD2 + HCHO + HO2 : ARR(4.2E-12,180);

{74} ETHE + O3 = HCHO + 0.4CHO2 + 0.12HO2 + 0.42CO +
            0.06CH4 :            ARR(1.2E-14,-2633);

{75} ALKE + O3 = 0.525HCHO + 0.5ALD2 + 0.2CHO2 +
            0.2CRO2 + 0.23HO2 + 0.215MO2 : ARR(7.8E-14,-2105);

{76} CHO2 + NO = HCHO + NO2 :    7.00E-12;

{77} CHO2 + NO2 = HCHO+ NO3 :    7.00E-13;

{78} CHO2 + H2O = ACO2 :         4.00E-18;

{79} CRO2 + NO = ALD2 + NO2 :    7.00E-12;

{80} CRO2 + NO2 = ALD2 + NO3 :   7.00E-13;

{81} CRO2 + H2O = ACTA :         4.00E-18;

{82} EO2 + EO2 = 2.4HCHO + 1.2HO2 + 0.4ALD2 : 5.00E-14;

{83} PO2 + PO2 = 2.2ALD2 + 1.2HO2 :          5.00E-14;

{84} HO2 + EO2 = ROOH + O2 :     3.00E-12;
```

```
{85} HO2 + PO2 = ROOH + O2 :     3.00E-12;

{86} SO2 + CHO2 = SO4 + HCHO :   {0;} 7.00E-14;

{87} SO2 + CRO2 = SO4 + ALD2 :   {0;} 7.00E-14;

{88} ALKE + NO3 = PRN1 :         1.26E-13;

{89} PRN1 + NO2 = PRN2 :         6.80E-12;

{90} PRN1 + HO2 = PRPN + O2 :    3.00E-12;

{91} PRN1 + NO = 2NO2 + HCHO + ALD2 : ARR(4.2E-12,180);

{92} CHO2 + HCHO = OZID :        1.36E-14;

{93} CHO2 + ALD2 = OZID :        1.36E-14;

{94} CRO2 + HCHO = OZID :        1.36E-14;

{95} CRO2 + ALD2 = OZID :        1.36E-14;

{96} AROM + OH = 0.84TO2 + 0.16CRES + 0.16HO2 : 1.52E-11;

{97} TO2 + NO = NO2 + HO2 + 0.72MGLY + 0.18GLYX
                    + DIAL : ARR(4.2E-12,180);

{98} GLYX + hv = PROD :          7.389E-5*SUN;

{99} GLYX + OH = HO2 + 2CO + H2O :    1.15E-11;

{100} MGLY + hv = MCO3 + HO2 + CO :   1.755E-4*SUN;

{101} MGLY + OH = MCO3 + CO + H2O :   1.73E-11;

{---- IN THE NEXT REACTION
  BETA 12, 13 WHERE REPLACED BY NUMBERS
  AS GIVEN IN LILING'S STEM CODE ----}

{102} CRES + OH = 0.83HO2 + 0.9ZO2 + 0.9TCO3
            -0.9OH - 0.0315NO2 : 4.25E-11;

{103} NO3 + CRES = HNO3 :        1.00E-11;

{104} OH + DIAL = TCO3 :         2.80E-11;
```

```
{105} TCO3 + NO2 = TPAN :         4.70E-12;

{106} TPAN = TCO3 + NO2:          ARR(2.2E+16,-13435);

{107} TCO3 + NO = NO2 + 0.92HO2 + 0.89GLYX + 0.11MGLY +

            0.05MCO3 :            ARR(4.2E-12,180);

{108} ZO2 + NO = NO2 :            ARR(4.2E-12,180);

{109} DIAL + hv = 0.98HO2 + 0.02MCO3 + TCO3 :  9.236E-5*SUN;

{110} HO2 + TO2 = ROOH + O2 :     4.00E-12;

{111} HO2 + TCO3 = ROOH + O2 :    4.00E-12;

{112} HO2 + ZO2 = ROOH + O2 :     1.00E-12;

{---- HERE COMES AEROSOL PART ----}

{113} AHO2 = HCHO + HO2 :         0;

{114} ISOP + OH = RIO2 :          8.505E-11;

{115} RIO2 + NO = 0.9NO2 + 0.45MVK + 0.45MACR +

            0.9HO2 + 0.9HCHO :    ARR(4.2E-12,180);

{116} RIO2 + HO2 = ROOH :         3.00E-12;

{117} MVK + OH = VRO2 :           ARR(3.0E-12,500);

{118} VRO2 + NO = 0.9NO2 + 0.6MCO3 + 0.6HAC +

            0.3HO2 + 0.3HCHO + 0.3MGGY : ARR(4.2E-12,180);

{119} VRO2 + HO2 = ROOH :         3.0E-12;

{120} OH + MACR = MAO3 :          1.02E-11;

{121} MAO3 + NO2 = MPAN :         4.7E-12;

{122} MPAN = MAO3 + NO2 :         ARR(2.2E+16,-13435);

{123} MAO3 + NO = 3NO3 - 2NO + HO2 + MGGY :ARR(4.2E-12,180);

{124} MAO3 + HO2 = ROOH :         3.00E-12;

{125} MACR + OH = MRO2 :          ARR(3.86E-12,500);
```

```
{126} MRO2 + NO = 0.9NO2 + 0.9HO2 +
            0.9HCHO + 0.9MGGY :  ARR(4.2E-12,180);
{127} MRO2 + HO2 = ROOH :        3.00E-12;
{128} HAC + OH = HACO :          1.5E-11;
{129} HACO + NO2 = IPAN :        4.7E-12;
{130} IPAN = HACO + NO2 :        ARR(2.2E+16,-13435);
{131} HACO + NO = NO2 + HO2 + HCHO : ARR(4.2E-12,180);
{132} HACO + HO2 = ROOH :        3.00E-12;
{133} ISOP + O3 = 0.5HCHO + 0.2MVK + 0.3MACR + 0.2CHO2 +
            0.06HO2 + 0.2MVKO + 0.3MAOO :ARR(7.0E-15,-1900);
{134} MVK + O3  = 0.5HCHO + 0.2CHO2 + 0.21HO2 + 0.2MCRG +
            0.15ALD2 + 0.5MGGY + 0.15MCO3 :ARR(4.0E-15,-2000);
{135} MACR + O3 = 0.65HCHO + 0.2CHO2 + 0.36HO2 +
            0.15NO2 - 0.15NO + 0.5MGGY + 0.2MCRG :
                     ARR(4.4E-15,-2500);
{136} MVKO + NO = MVK + NO2 :    ARR(4.2E-12,180);
{137} MVKO + NO2 = MVK + NO3 :   ARR(4.2E-13,180);
{138} MVKO + H2O = PROD :        3.4E-18;
{139} MAOO + NO = MACR + NO2 :   ARR(4.2E-12,180);
{140} MAOO + NO2 = MACR + NO3 :  ARR(4.2E-13,180);
{141} MAOO + H2O = MACA :        3.4E-18;
{142} MCRG + NO = MGGY + NO2 :   ARR(4.2E-12,180);
{143} MCRG + NO2 = MGGY + NO3 :  ARR(4.2E-13,180);
{144} MCRG + H2O = PYVA :        3.4E-18;
{145} HAC + hv = HCHO + 2HO2 :   4.618E-6*SUN;
```

```
{146} MGGY + hv = MCO3 + HO2 :    1.385E-3*SUN;
{147} MGGY + OH = MCO3 :          1.7E-11;
{148} ISOP + NO3 = INO2 :         ARR(3.00E-12,-450);
{149} INO2 + NO = 2NO2 + HCHO + 0.5MVK + 0.5MACR :
                                  ARR(4.2E-12,180);
{150} INO2 + NO2 = PROD :         ARR(4.2E-13,180);
{151} INO2 + HO2 = PROD :         3.00E-12;
{152} MVK + NO3 = MVN2 :          6.00E-14;
{153} MVN2 + NO = 2NO2 + HCHO + 0.5MCO3 +
            0.5MGGY + 0.5HO2 :    ARR(4.2E-12,180);
{154} MVN2 + HO2 = PROD :         3.0E-12;
{155} MACR + NO3 = MAO3 + HNO3 : 3.3E-15;
{156} MACR + NO3 = MAN2 :         6.7E-15;
{157} MAN2 + NO = 2NO2 + HCHO + MGGY :  ARR(4.2E-12,180);
{157} MAN2 + HO2 = PROD :         3.00E-12;
{159} HAC + NO3 = HNO3 + HACO :   5.2E-16;
{160} MAOO + HO2 = ROOH :         3.00E-12;
{161} MVKO + HO2 = ROOH :         3.00E-12;
{162} MVKO + SO2 = SO4 + MVK :    {0;} 7.00E-14;
{163} MAOO + SO2 = SO4 + MACR :   {0;} 7.00E-14;
{164} MCRG + SO2 = SO4 + MVK :    {0;} 7.00E-14;
{165} MACA + OH = PROD :          ARR(1.2E-11,500);
{166} PYVA + OH = PROD :          5.00E-14;
{167} DOL6 + O3 = 0.11SUCA :      5.44E-17;
{168} DOL7 + O3 = 0.19GLUA :      3.46E-17;
```

```
{169} DOL8 + O3 = 0.15ADIA :     2.21E-17;
{170} CPET + O3 = 0.39GLUA :     1.03E-15;
{171} CHEX + O3 = 0.15ADIA :     2.16E-16;
{172} OH + HO2 = H2O + O2 :      ARR(4.6E-11,230);
{173} ROOH + hv = HCHO + OH + HO2 :       4.618E-6*SUN;
{174} ROOH + OH = 0.5MO2 + 0.5OH + 0.5HCHO : 1.00E-11;
{175} DMS + OH = SO2 + MSA :     8.3E-12;
{176} NO3 + NO3 = 2NO2 + O2 :    ARR(8.5E-13,-2450);
{177} OH + PAN = PROD :          ARR(1.23E-12,-651);
{178} NO3 + HO2 = 0.6OH + 0.6NO2 + 0.4HNO3 : ARR(2.3E-12,170);
```

A.3 The TMk model

```
#EQUATIONS {of reduced methane/CO/HOx/NOx chemistry:
           Frank Dentener}
{ 1.} NO2 + hv {+ O2} =  NO + O3
   : RJF(ISTEP,1);{j1..j11 now from file}
{ 2.} HNO3 + hv = NO2 + OH   : RJF(ISTEP,2);
{ 3.} O3 + hv {+H2O} = 2 OH  : j3;
{ 4.} H2O2 + hv = 2 OH       : RJF(ISTEP,4);
{ 5.} MEP  + hv {+ O2}= HCHO + HO2 + OH: RJF(ISTEP,5);
{ 6.} HCHO + hv {+ 2 O2} = 2 HO2 + CO  : RJF(ISTEP,6);
{ 7.} HCHO + hv = CO         : RJF(ISTEP,7);
{ 8.} N2O5 + hv = NO3 + NO2  : RJF(ISTEP,8);
{ 9.} NO3  + hv {+O2} = NO2 + O3       : RJF(ISTEP,9);
```

```
{10.} NO3  + hv = NO + O2    : RJF(ISTEP,10);

{11.} PNA  + hv = NO2 + HO2  : RJF(ISTEP,11);

{12.} O3 + NO = NO2          : ARR(2.E-12, -1400.0);

{13.} HO2 + NO = OH + NO2    : ARR(3.7E-12, 250.0);

{14.} CH3O2 + NO {+O2} = HCHO +HO2 + NO2 : ARR(4.2E-12, 180.0);

{15.} OH + NO2   =  HNO3     : ZF3BOD(RX1_15,RX2_15);

{16.} OH + HNO3  = NO3 + H2O : RX1_16;

{17.} O3 + NO2 =  NO3        : ARR(1.2E-13, -2450.0);

{18.} NO3 + NO = 2 NO2       : ARR(1.5E-11,170.0);

{19.} NO3 + NO2 = N2O5       : ZF3BOD(RX1_19,RX2_19);

{20.} N2O5 {+M}  =  NO3 + NO2 +M       : RX1_20;

{21.} OH + PNA = NO2         : ARR(1.3E-12, 380.0);

{22.} HO2 + NO2   =  PNA     : ZF3BOD(RX1_22,RX2_22);

{23.} PNA   = HO2 + NO2      : RX1_23;

{24.} O3 + HO2 = OH +O2      : ARR(1.1E-14, -500.0);

{25.} O3 + OH = HO2          : ARR(1.6E-12, -940.0);

{26.} OH + CO  = HO2 + CO2   : RX1_26;

{27.} HCHO + OH  =  HO2 + CO : 1.0E-11;

{28.} N2O5 {+ H2O} = 2 HNO3  : 5E-5;

{29.} CH4 + OH = CH3O2 + H2O : ARR(2.9E-12,-1820.0);

{30.} 2 HO2 {+H2O} = H2O2    : RX1_30;

{31.} OH + H2O2 = HO2        : ARR(2.9E-12, -160.0);

{32.} MEP + OH =  CH3O2 + H2O: 0.7*ARR(3.8E-12,200.0);

{33.} MEP + OH =  HCHO + OH + H2O      : 0.3*ARR(3.8E-12,200.0);

{34.} CH3O2 + HO2 = MEP + OH : ARR(3.8E-13,800.0);
```

```
{35.} HO2 + OH    = H2O + O2 : ARR(4.8E-11,250.0);

{emi} SOURCE = NO : fnoxemi;
```

A.4 The stratospheric model

```
{ 1.} O + O2 = O3 :  8.018216E-17;

{ 2.} O + O3 = 2O2 : 1.575705E-15;

{ 3.} O1D + N2 = O + N2 :                 2.838868E-11;

{ 4.} O1D + O2 = O + O2 :                 4.276308E-11;

{ 5.} O1D + O3 = 2O2 :                    1.200000E-10;

{ 6.} H2O + O1D = 2OH :                   2.200000E-10;

{ 7.} H2 + O1D = OH + H :                 1.000000E-10;

{ 8.} CH4 + O1D = CH2O + H2 :             7.500000E-12;

{ 9.} CH4 + O1D = CH3 + OH :              1.425000E-10;

{10.} H + O2 = HO2 : 6.353884E-15;

{11.} H + O3 = OH + O2 :                  1.998353E-11;

{12.} H2 + OH = H2O + H :                 1.388923E-15;

{13.} OH + O3 = HO2 + O2 :                3.259930E-14;

{14.} OH + O = O2 + H :                   3.616461E-11;

{15.} OH + OH = H2O + O :                 1.554272E-12;

{16.} HO2 + O = OH + O2 :                 6.868940E-11;

{17.} HO2 + O3 = OH + 2O2 :               1.386665E-15;

{18.} H + HO2 = 2OH :7.000000E-11;

{19.} HO2 + OH = H2O + O2 :               1.351923E-10;

{20.} HO2 + HO2 = H2O2 + O2 :             2.769464E-12;
```

```
{21.} HO2 + HO2 + H2O = H2O2 + O2 + H2O : 3.515415E-29;
{22.} H2O2 + OH = H2O + HO2 :             1.494801E-12;
{23.} NO + O3 = NO2 + O2 :                6.062488E-15;
{24.} NO + HO2 = NO2 + OH :               1.042107E-11;
{25.} NO2 + O = NO + O2 :                 1.068500E-11;
{26.} NO2 + O3 = NO3 + O2 :               4.699133E-18;
{27.} NO2 + OH = HNO3 :                   3.724129E-13;
{28.} NO2 + HO2 = HNO4 :                  2.690292E-14;
{29.} NO3 + O = O2 + NO2 :                1.000000E-11;
{30.} NO3 + NO = 2NO2 :                   3.033154E-11;
{31.} NO3 + NO2 = N2O5 :                  2.325973E-13;
{32.} N2O5 = NO2 + NO3 :                  1.405991E-06;
{33.} HNO3 + OH = H2O + NO3 :             1.890063E-13;
{34.} HNO4 + OH = H2O + NO2 + O2 :        6.273406E-12;
{35.} HNO4 = HO2 + NO2 :                  3.163299E-07;
{36.} N2O5 = 2HNO3 : 0.000000E+00;
{37.} Cl + O2 = ClOO :                    3.032678E-16;
{38.} Cl + O3 = ClO + O2 :                9.878682E-12;
{39.} Cl + H2 = HCl + H :                 2.696898E-15;
{40.} Cl + HO2 = HCl + O2 :               3.639785E-11;
{41.} Cl + HO2 = OH + ClO :               6.357771E-12;
{42.} Cl + H2O2 = HCl + HO2 :             1.899010E-13;
{43.} ClO + O = Cl + O2 :                 4.009038E-11;
{44.} ClO + OH = HO2 + Cl :               1.808230E-11;
{45.} ClO + OH = HCl + O2 :               0.000000E+00;
```

{46.} ClO + HO2 = O2 + HOCl : 8.718280E-12;

{47.} ClO + NO = NO2 + Cl : 2.127376E-11;

{48.} ClO + NO2 = ClONO2 : 2.886454E-14;

{49.} ClO + ClO = Cl + OClO : 1.201307E-15;

{50.} ClO + ClO = Cl + ClOO : 1.174783E-15;

{51.} ClO + ClO = Cl2 + O2 : 1.379878E-15;

{52.} ClO + ClO = Cl2O2 : 3.330338E-15;

{53.} ClOO = Cl + O2 : 1.694865E+04;

{54.} Cl2O2 = 2ClO : 4.773959E-04;

{55.} HCl + OH = H2O + Cl : 6.100684E-13;

{56.} HOCl + OH = H2O + ClO : 3.781814E-13;

{57.} ClONO2 + O = ClO + NO3 : 1.055174E-13;

{58.} ClONO2 + OH = HOCl + NO3 : 3.021112E-13;

{59.} ClONO2 + Cl = Cl2 + NO3 : 1.319240E-11;

{60.} ClONO2 = HOCl + HNO3 : 0.000000E+00;

{61.} Br + O3 = BrO + O2 : 6.185505E-13;

{62.} Br + HO2 = HBr + O2 : 1.249643E-12;

{63.} Br + CH2O = HBr + HCO : 6.185505E-13;

{64.} BrO + O = Br + O2 : 4.990544E-11;

{65.} BrO + HO2 = HOBr + O2 : 4.918275E-11;

{66.} BrO + NO = Br + NO2 : 2.583341E-11;

{67.} BrO + NO2 = BrONO2 : 7.684217E-14;

{68.} BrO + ClO = Br + OClO : 9.497781E-12;

{69.} BrO + ClO = Br + ClOO : 7.213454E-12;

{70.} BrO + ClO = BrCl + O2 : 1.172819E-12;

```
{71.} BrO + BrO = 2Br + O2 :              2.467250E-12;

{72.} HBr + OH = Br + H2O :               1.100000E-11;

{73.} CO + OH = H :  1.502401E-13;

{74.} CH4 + OH = CH3 + H2O :              1.532251E-15;

{75.} CH2O + OH = HCO + H2O :             1.000000E-11;

{76.} CH2O + O = HCO + OH  :              4.501231E-14;

{77.} Cl + CH4 = CH3 + HCl :              3.334368E-14;

{78.} Cl + CH2O = HCl + HCO :             7.153519E-11;

{79.} HCO + O2 = CO + HO2 :               6.250373E-12;

{80.} CH3 + O2 = CH3O2 :                  5.883651E-14;

{81.} CH3O + O2 = CH2O + HO2 :            9.377921E-16;

{82.} CH3O2 + NO = CH3O + NO2 :           8.851990E-12;

{83.} CH3O2 + HO2 = CH3OOH + O2 :         1.044377E-11;

{84.} CH3OOH + OH = CH3O2 + H2O :         6.182046E-12;


{ 1.} O2 + hv = 2O : 2.643E-10 * SUN**3;

{ 2.} O3 + hv = O + O2 :                  6.120E-04 * SUN;

{ 3.} O3 + hv = O1D + O2 :                1.070E-03 * SUN**2;

{ 4.} HO2 + hv = OH + O :                 1.817E-04 * SUN**2;

{ 5.} H2O2 + hv = 2OH :                   3.537E-05 * SUN;

{ 6.} NO2 + hv = NO + O :                 1.289E-02 * SUN;

{ 7.} NO3 + hv = NO2 + O :                2.359E-01 * SUN;

{ 8.} NO3 + hv = NO + O2 :                2.883E-02 * SUN;

{ 9.} N2O5 + hv = NO2 + NO3 :             2.594E-04 * SUN**2;

{10.} HNO3 + hv = OH + NO2 :              5.269E-05 * SUN**2;
```

```
{11.} HNO4 + hv = OH + NO3 :              4.557E-05 * SUN**2;
{12.} HNO4 + hv = HO2 + NO2 :             9.113E-05 * SUN**2;
{13.} Cl2 + hv = 2Cl :                    3.692E-03 * SUN;
{14.} OClO + hv = O + ClO :               1.243E-01 * SUN;
{15.} Cl2O2 + hv = Cl + ClOO :            3.421E-03 * SUN;
{16.} HOCl + hv = OH + Cl :               4.606E-04 * SUN;
{17.} ClONO2 + hv = Cl + NO3 :            2.254E-04 * SUN**2;
{18.} ClONO2 + hv = Cl + NO2 + O :        2.505E-05 * SUN**2;
{19.} BrCl + hv = Br + Cl :               1.867E-02 * SUN;
{20.} BrO + hv = Br + O :                 5.950E-02 * SUN;
{21.} HOBr + hv = Br + OH :               1.284E-03 * SUN;
{22.} BrONO2 + hv = Br + NO3 :            2.074E-03 * SUN;
{23.} CH2O + hv = HCO + H :               7.553E-05 * SUN;
{24.} CH2O + hv = CO + H2 :               8.303E-05 * SUN;
{25.} CH3OOH + hv = CH3O + OH :           2.285E-05 * SUN;
```

A.5 The aqueous model

```
{ 1.} NO2 + hv =  NO + O    :  8.89E-3*SUN;
{ 2.} O {+ O2 + M} = O3     :  ARR(1.4E+3, 1175);
{ 3.} O3 + NO = NO2         :  ARR(1.8E-12, -1370);
{ 4.} O + NO2 = NO          :  9.3E-12;
{ 5.} O + NO2   = NO3       :  ARR(1.6E-13, 687);
{ 6.} O + NO   =  NO2       :  ARR(2.2E-13, 602);
{ 7.} O3 + NO2 =  NO3       :  ARR(1.2E-13, -2450);
```

```
{ 8.} O3 + hv = O           :  4.0E-2*RCONST(1);
{ 9.} O3 + hv = O1D         :  2.8E-3*RCONST(1);
{10.} O1D   = O             :  ARR(1.9E+8, 390) ;

{11.} O1D + H2O = 2OH       :  2.2E-10;
{12.} O3 + OH = HO2         :  ARR(1.6E-12, -940);
{13.} O3 + HO2 = OH         :  ARR(1.4E-14, -580);
{14.} NO3 + hv = 0.89 NO2 + 0.89 O
                + 0.11 NO   :  15.5*RCONST(1) ;
{15.} NO3 + NO = 2 NO2      :  ARR(1.3E-11, 250);
{16.} NO3 + NO2 = NO + NO2  :  ARR(2.5E-14, -1230);
{17.} NO3 + NO2   =  N2O5   :  ARR(5.3E-13, 256);
{18.} N2O5 + H2O = 2 HNO3   :  1.3E-21;
{19.} N2O5   =  NO3 + NO2   :  ARR(3.5E+14, -10897);
{20.} 2 NO  =  2 NO2        :  ARR(1.8E-20, 530);


{21.} NO + NO2 + H2O = 2 HONO         :  4.4E-40;
{22.} OH + NO   =  HONO     :  ARR(4.5E-13, 806);
{23.} HONO + hv =  OH + NO  :  0.17*RCONST(1);
{24.} OH + HONO =  NO2      :  6.6E-12;
{25.} 2 HONO = NO + NO2     :  1.0E-20;
{26.} OH + NO2   =  HNO3    :  ARR(1.0E-12, 713);
{27.} OH + HNO3   =  NO3    :  ARR(5.1E-15, 1000);
{28.} HO2 + NO = OH + NO2   :  ARR(3.7E-12, 240);
```

```
{29.} HO2 + NO2   =  PNA    :  ARR(1.2E-13, 749);
{30.} PNA   = HO2 + NO2     :  ARR(4.8E+13, -10121);

{31.} OH + PNA = NO2        :  ARR(1.3E-12, 380);
{32.} 2 HO2 = H2O2          :  ARR(5.9E-14, 1150) ;
{33.} 2 HO2 + H2O = H2O2    :  ARR(2.2E-38, 5800);
{34.} H2O2 + hv = 2 OH      :  7.1E-4*RCONST(1);
{35.} OH + H2O2 = HO2       :  ARR(3.1E-12, -187);
{36.} OH + CO  = HO2        :  2.2E-13;
{37.} HCHO + OH  =  HO2 + CO    :  1.0E-11;
{38.} HCHO + hv {+ 2 O2} = 2 HO2 + CO :  3.2E-3*RCONST(1);
{39.} HCHO +  hv = CO       :  4.2E-3*RCONST(1);
{40.} HCHO + O = OH + HO2 + CO        :  ARR(3.0E-11, -1550);

{41.} HCHO + NO3  = HNO3
                       + HO2 + CO    :  6.3E-16;
{42.} ALD2 + O  =  C2O3 + OH:  ARR(1.2E-11, -986);
{43.} ALD2 + OH = C2O3      :  ARR(7.0E-12, 250);
{44.} ALD2 + NO3  = C2O3 + HNO3 :  2.5E-15 ;
{45.} ALD2 + hv {+ 2 O2} = HCHO + XO2
                         + CO + 2 HO2 :  4.5E-4*RCONST(1);
{46.} C2O3 + NO  = HCHO + XO2
                      + HO2 + NO2    :  ARR(5.4E-12, 250);
{47.} C2O3 + NO2 = PAN      :  ARR(8.0E-20, 5500);
{48.} PAN = C2O3 + NO2      :  ARR(9.4E+16, -14000);
```

```
{49.} 2 C2O3 = 2 HCHO + 2 XO2 + 2 HO2 : 2.0E-12 ;

{50.} C2O3 + HO2 = 0.79 HCHO
   + 0.79 XO2 + 0.79 HO2 + 0.79 OH    :  6.5E-12;


{51.} OH = HCHO + XO2 + HO2 :  ARR(1.1E+2, -1710);
{52.} PAR + OH =  0.87 XO2 + 0.13 XO2N
                  + 0.11 HO2 + 0.11 ALD2
                  + 0.76 ROR - 0.11 PAR        :  8.1E-13;
{53.} ROR = 1.1 ALD2 + 0.96 XO2 + 0.94 HO2 + 0.04 XO2N
            + 0.02 ROR - 2.10 PAR     :  ARR(1.0E+15, -8000);
{54.} ROR = HO2              :  1.6E+3;
{55.} ROR + NO2 =  PROD     : 1.5E-11 ;
{56.} O + OLE = 0.63 ALD2 + 0.38 HO2
                + 0.28 XO2 + 0.3 CO
                + 0.2 HCHO + 0.02 XO2N
                + 0.22 PAR + 0.2 OH   :  ARR(1.2E-11, -324);
{57.} OH + OLE = HCHO + ALD2 + XO2
                 + HO2 - PAR:  ARR(5.2E-12, 504);
{58.} O3 + OLE = 0.5 ALD2 + 0.74 HCHO
                 + 0.33 CO + 0.44 HO2
                 + 0.22 XO2
                 + 0.1 OH - PAR       :   ARR(1.4E-14, -2105) ;
{59.} NO3 + OLE = 0.91 XO2 + HCHO
                  + ALD2 + 0.09 XO2N
                  + NO2 - PAR         :  7.7E-15;
```

```
{60.} O + ETH = HCHO + 0.7 XO2 + CO + 1.7 HO2 + 0.3 OH :
   ARR(1.0E-11, -792);
{61.} OH + ETH = XO2 + 1.56 HCHO + HO2 + 0.22 ALD2  :
    ARR(2.0E-12, 411) ;
{62.} O3 + ETH = HCHO + 0.42 CO + 0.12 HO2: ARR(1.3E-14, -2633);

{63.} OH + TOL = 0.08 XO2 + 0.36 CRES
                + 0.44 HO2 + 0.56 TO2:  ARR(2.1E-12, 322);
{64.} TO2 + NO =  0.9 NO2 + 0.9 OPEN + 0.9 HO2  :  8.1E-12;
{65.} TO2 = HO2 + CRES               :  4.2;
{66.} OH + CRES = 0.4 CRO + 0.6 XO2 + 0.6 HO2 + 0.3 OPEN:4.1E-11;
{67.} NO3 + CRES = CRO + HNO3               :  2.2E-11;
{68.} CRO + NO2 = PROD                      :  1.4E-11;
{69.} OH + XYL = 0.7 HO2 + 0.5 XO2 + 0.2 CRES + 0.8 MGLY
                + 1.10 PAR + 0.3 TO2    :  ARR(1.7E-11, 116);
{70.} OH + OPEN = XO2 + C2O3 + 2 HO2 + 2 CO
                             + HCHO :  3.0E-11;
{71.} OPEN + hv = C2O3 + CO + HO2        :  6.0E-3*RCONST(1);
{72.} O3 + OPEN = 0.03 ALD2 + 0.62 C2O3
                 + 0.7 HCHO + 0.03 XO2 + 0.69 CO
                 + 0.08 OH + 0.76 HO2
                 + 0.2 MGLY  : ARR(5.4E-17, -500);
{73.} OH + MGLY =  XO2 + C2O3            : 1.7E-11;
{74.} MGLY + hv = C2O3 + CO + HO2        : 1.86E-2*RCONST(1) ;
{75.} O + ISOP =  0.6 HO2 + 0.8 ALD2 + 0.55 OLE + 0.5 XO2
```

```
                + 0.5 CO + 0.45 ETH + 0.9 PAR :   1.8E-11 ;
{76.} OH + ISOP = HCHO + XO2 + 0.67 HO2
                + 0.4 MGLY + 0.2 C2O3
                + ETH + 0.2 ALD2 + 0.13 XO2N  :   9.6E-11 ;
{77.} O3 + ISOP = HCHO + 0.4 ALD2 + 0.55 ETH + 0.2 MGLY
                + 0.06 CO + 0.1 PAR + 0.44 HO2 + 0.1 OH:
                          1.2E-17;
{78.} NO3 + ISOP =  XO2N                    :  3.2E-13;
{79.} XO2 + NO = NO2                        :  8.1E-12;
{80.} 2 XO2 =  PROD               :  ARR(1.7E-14, 1300);
{81.} XO2N + NO =  PROD           :  6.8E-13;

{62/RG71}      SO2  + OH  = HO2  + PROD  :  1.0870e-12;
{63/RG72}      SO2  = PROD  :   1.3833e-06;

{Methane and Methylperoxyl Radical Reactions}
{}  CH4 + OH = CH3O2      :   ARR(2.4E-12,-1710);
{}  CH3O2 + NO = CH3O + NO2 : ARR(4.15E-12,180) ;
{}  CH3O2 + HO2 = PROD :      ARR(1.7E-13,1000);
{}  2 CH3O2 = 0.76 CH3O + 0.62 HCHO + PROD : ARR(1.62E-13, 220);
{}  CH3O + O2 = HCHO + HO2 :  ARR(7.12E-14,-1080);

{82/RE21}      HO2  = HO2aq :   3.3000e-01;
{83/RE22}      HO2aq = HO2 :    2.2492e+01;
{84/RE31}      H2O2  = H2O2aq  :3.3000e-01;
```

```
{85/RE32}      H2O2aq = H2O2  : 6.0790e-01;
{86/RE41}      O3  = O3aq :     1.0500e-02;
{87/RE42}      O3aq = O3  :     1.3012e+05;
{88/RE51}      HCHO  = HCHOaq : 2.4000e-01;
{89/RE52}      HCHOaq = HCHO :  5.1931e+00;
{90/RE61}      HCOOH  = HCOOHaq  :                    2.4000e-01;
{91/RE62}      HCOOHaq = HCOOH :8.8422e+00;
{92/RE71}      ROOH  = CH3O2Haq :                     2.4000e-01;
{93/RE72}      CH3O2Haq = ROOH  :                     1.4871e+02;
{94/RE81}      XO2  = CH3O2aq : 2.4000e-01;
{95/RE82}      CH3O2aq = XO2  : 1.6358e+01;
{96/RE91}      HNO3  = HNO3aq  :2.4000e-01;
{97/RE92}      HNO3aq = HNO3  : 1.5579e-01;
{98/RE101}     NO3  = NO3aq  :  2.4000e-01;
{99/RE102}     NO3aq = NO3 :    2.1811e+03;
{100/RE111}    NH3  = NH3aq :   2.4000e-01;
{101/RE112}    NH3aq = NH3  :   4.3622e+02;
{102/RE121}    SO2  = SO2aq :   2.4000e-01;
{103/RE122}    SO2aq = SO2 :    2.6599e+04;
{104/RE131}    OH  = OHaq :     2.4000e-01;
{105/RE132}    OHaq = OH  :     3.6351e+00;
{106/RE141}    N2O5  = HNO3aq + HNO3aq :              7.5000e-02;
{107/RE151}    H2Oaq = HPLUSaq + OHMINaq   :          2.3636e-05;
{108/RE152}    HPLUSaq + OHMINaq = H2Oaq  :           7.1958e-04;
{109/RE171}    HCOOHaq = HPLUSaq + HCOOMINaq  :       8.6000e+06;
```

```
{110/RE172}    HPLUSaq + HCOOMINaq = HCOOHaq  :      2.6016e-04;
{111/RE181}    HO2aq = HPLUSaq + O2MINaq :           1.6000e+05;
{112/RE182}    HPLUSaq + O2MINaq = HO2aq  :          5.5353e-05;
{113/RE191}    SO2aq = HSO3MINaq + HPLUSaq :         3.6000e+06;
{114/RE192}    HSO3MINaq + HPLUSaq = SO2aq  :        1.6606e-06;
{115/RE201}    HSO3MINaq = SO32MINaq + HPLUSaq  :    1.0000e+04;
{116/RE202}    SO32MINaq + HPLUSaq = HSO3MINaq  :    8.3029e-04;
{117/RE211}    NH3aq = NH4aq + OHMINaq  :            6.0000e+05;
{118/RE212}    NH4aq + OHMINaq = NH3aq :             1.8820e-04;
{119/RE221}    HNO3aq = NO3MINaq + HPLUSaq :         2.2000e+09;
{120/RE222}    NO3MINaq + HPLUSaq = HNO3aq  :        7.7494e-07;

{121/RW01}      FE3PLUSaq = FE2PLUSaq + OHaq :       0.4583E-03 ;
{122/RW02}      H2O2aq = OHaq + OHaq :               0.7219E-05;
{123/RW03 }     HCHOaq + OHaq = HCOOHaq
                      + HO2aq + H2Oaq :              0.1107E-04 ;
{124/RW05}      HCOOMINaq + OHaq = O2MINaq + CO2aq : 0.1771E-04 ;
{125/RW06}      HCOOHaq + OHaq = O2MINaq
                    + HPLUSaq + CO2aq :              0.7196E-06 ;
{126/RW07}      HSO3MINaq + OHaq = SO42MINaq + HPLUSaq:0.2491E-04;
{127/RW08}      HMSAaq + OHaq = SO42MINaq
                      + HCHOaq + HPLUSaq :            0.5535E-05 ;
{128/RW09}      FE2PLUSaq + OHaq = FE3PLUSaq
                      + OHMINaq: 0.2380E-05 ;
{129/RW10}      H2O2aq + OHaq = H2Oaq + HO2aq  :      0.1495E-06;
```

```
{130/RW11}      O3aq + OHaq = HO2aq  :                0.1661E-04;

{131/RW13}      O2MINaq + FE2PLUSaq = FE3PLUSaq
                     + H2O2aq + OHMINaq + OHMINaq :   0.5535E-07;
{132/RW14}      O2MINaq + FE3PLUSaq = FE2PLUSaq :     0.8303E-06;

{133/RW15}      O2MINaq + CUPLUSaq = CU2PLUSaq
                  + H2O2aq + OHMINaq + OHMINaq :      0.4982E-04;
{134/RW16}      O2MINaq + CU2PLUSaq = CUPLUSaq  :     0.4428E-04;

{135/RW17}      O2MINaq + O3aq = OHaq + OHMINaq  :    0.8303E-05;

{136/RW18}      HO2aq   + FE2PLUSaq = FE3PLUSaq
                    + H2O2aq + OHMINaq   :            0.6642E-08;
{137/RW19}      HO2aq + FE3PLUSaq = FE2PLUSaq + HPLUSaq :
                                          0.5535E-10;
{138/RW20}      HO2aq + CUPLUSaq = CU2PLUSaq
                    + H2O2aq + OHMINaq  :             0.5535E-05;
{139/RW21}      HO2aq + CU2PLUSaq = CUPLUSaq + HPLUSaq:0.2768E-06;
{140/RW22}      H2O2aq  + HSO3MINaq = SO42MINaq
                    + H2Oaq + HPLUSaq  :              0.5535E-14;
{141/RW23}      H2O2aq  + FE2PLUSaq = FE3PLUSaq
                   + OHaq + OHMINaq :                 0.5535E-14 ;
{142/RW24}      H2O2aq  + CUPLUSaq = CU2PLUSaq
                   + OHaq + OHMINaq  :                0.2214E-08 ;
```

```
{143/RW25}      CUPLUSaq = CU2PLUSaq + O2MINaq :      0.2500E-03 ;

{144/RW26}      CUPLUSaq + FE3PLUSaq = CU2PLUSaq

                   + FE2PLUSaq : 0.1661E-06;

{145/RW27}      CUPLUSaq + O3aq = CU2PLUSaq

                   + OHaq + OHMINaq :                 0.5535E-07 ;

{146/RW28}      FE2PLUSaq + O3aq = FE3PLUSaq

                   + OHaq + OHMINaq :                 0.4539E-08 ;

{147/RW29}      FE3PLUSaq + HSO3MINaq = FE2PLUSaq

                      + SO42MINaq + HPLUSaq  :        0.5535E-14 ;

{148/RW30}      O3aq + HSO3MINaq = SO42MINaq + HPLUSaq:

                                              0.1716E-08;

{149/RW31}      O3aq + SO32MINaq = SO42MINaq  :       0.5535E-05 ;


{150/RW32}      O3aq = H2O2aq  : 0.2355E-04 ;

{151/RW34}      HO2aq + O2MINaq = H2O2aq + OHMINaq :  0.5535E-06 ;

{152/RW35}      HO2aq + HO2aq = H2O2aq  :             0.4594E-08 ;


{153/RW39}      SO32MINaq + HCHOaq = HMSAaq

                     + OHMINaq  :0.1384E-06 ;

{154/RW40}      HMSAaq + OHMINaq = SO32MINaq

                     + HCHOaq :  0.1993E-10 ;

{155/RW42}      CH3O2aq + O2MINaq = CH3O2Haq

                     + OHMINaq  :0.2768E-06 ;

{156/RW43}      CH3O2Haq + OHaq = CH3O2aq + H2Oaq  :  0.1495E-06 ;

{157/RW44}      CH3O2Haq + OHaq = HCHOaq + OHaq  :    0.1052E-06 ;
```

```
{CO2 reactions}
CO2aq = HPLUSaq + HCO3MINaq :  1.0E-4;
HPLUSaq + HCO3MINaq = CO2aq :  1.2178E-12;

CO2aq + H2Oaq = H2CO3aq :                    0*3.0E-2;
H2CO3aq = CO2aq + H2Oaq : 0*1.19E+1;
H2CO3aq = HPLUSaq + HCO3MINaq :             0*8.0E+6 ;
HPLUSaq + HCO3MINaq = H2CO3aq :             0*4.7E+10;
HPLUSaq + HCO3MINaq  = CO2aq :              0*1.5E+4;
CO2aq  + H2Oaq = HPLUSaq + HCO3MINaq :      0*1.0E-2;
```

REFERENCES

[1] T. Alishenas and Ö. Ólafsson. Modeling and velocity stabilization of constrained mechanical systems with comparative study of two test problems. Preprint, NADA, Royal Institute of Technology, Stockholm, 1993.

[2] H. Amann. *Ordinary Differential Equations: An Introduction to Nonlinear Analysis.* Walter de Gruyter, 1990.

[3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. LAPACK User's Guide, second edition. Technical report, SIAM, Philadelphia, PA, 1995.

[4] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton. Implicit-Explicit methods for time dependent PDE's. *Technical Report 93-15*, 1993.

[5] R.D. Atkinson, D.L. Baulch, R.A. Cox, R.F.JR. Hampson, J.A. Kerr, and J. Troe. Evaluated kinetic and photochemical data for atmospheric chemistry. *International Journal of Chemical Kinetics*, 21:115–190, 1989.

[6] G. Bader and P. Deuflhard. A semi-implicit mid-point rule for stiff systems of ordinary differential equations. *Numer. Math.*, 41:373–398, 1983.

[7] Ch. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR generating derivative codes from FORTRAN programs. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1992.

[8] Ch. Bischof, A. Carle, P. Khademi, and A. Mauer. The ADIFOR2.0 system for the automatic differentiation of FORTRAN77 programs. Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1994.

[9] H.G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. *Modelling of Chemical Reaction Systems, K.H. Ebert, P. Deuflhard and W. Jaeger editors, Springer Series in Chem. Phys.*, 18:102–125, 1981.

[10] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations.* Elsevier Science Publishers, 1989.

[11] P.N. Brown, G.D. Byrne, and A.C. Hindmarsh. VODE: A Variable Step ODE Solver. *SIAM J. Sci. Stat. Comput.*, 10:1038–1051, 1989.

[12] G.D. Byrne and A.M. Dean. The numerical solution of some chemical kinetics models with VODE and CHEMKIN II. *Computers Chem.*, 17:297–302, 1993.

[13] D. G. Cacuci. Sensitivity theory for nonlinear systems. I. Nonlinear functional analysis approach. II. Extensions to additional classes of responses. *J. Math. Phys.*, 22:2794–2812, 1981.

[14] G.R. Carmichael, L.K. Peters, and T. Kitada. A second generation model for regional-scale transport/ chemistry/ deposition. *Atmospheric environment*, 20:173–188, 1986.

[15] G.R. Carmichael, A. Sandu, and F.A. Potra. Sensitivity Analysis for Atmospheric Chemistry models via Automatic Differentiation. *Atmospheric Environment*, 31:475 – 489, 1997.

[16] B.W. Char, K.O. Geddes, G.H. Gonnet, M.B. Monagan, and S.M. Watt. *Maple V Language Reference Manual.* Springer-Verlag, New York, 1991.

[17] M. Chin, D. Jacob, J. Munger, D. Parrish, and B. Doddridge. Relationship of ozone and carbon monoxide over north america. *J. Geophys. Res.*, 99:14,565–14,573, 1994.

[18] Y. S. Cho. Ph.d. thesis. *The University of Iowa*, 1986.

[19] Y. S. Cho and G.R. Carmichael. Evaluation of liquid phase chemical production of sulfate using sensitivity analysis. *Atmospheric Environment*, 20:l959–l988, 1986.

[20] Y. S. Cho, G.R. Carmichael, and H. Rabitz. Sensitivity analysis of the advection-diffusion equation. *Atmospheric Environment*, 21:2589–2598, 1987.

[21] Y. S. Cho, G.R. Carmichael, and H. Rabitz. The relationship between primary emissions and acid deposition in eulerian models determined by sensitivity analysis. *Water, Air and Soil Pollution*, 40:9–31, 1988.

[22] D. P. Chock and S. L. Winkler. A comparison of advection algorithms coupled with chemistry. *Atmospheric Environment*, 28(16):2659–2675, 1994.

[23] D. Dabdub and J.H. Seinfeld. Extrapolation techniques used in the solution of stiff odes associated with chemical kinetics of air quality models. *Atmospheric Environment*, 29:403–410, 1995.

[24] V. Damian-Iordache. KPP - a chemical development environment. Technical report, The University of Iowa, Iowa City, IA 52246, 1996.

[25] V. Damian-Iordache, A. Sandu, M. Damian-Iordache, G. R. carmichael, and F. A. Potra. KPP - A symbolic preprocessor for chemistry kinetics - User's guide. Technical report, The University of Iowa, Iowa City, IA 52246, 1995.

[26] J. J. B. de Swart and J. G. Blom. Experiences with sparse matrix solvers in parallel ODE software. Technical report, Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, 1995.

[27] J. E. Dennis. On the kanrovitch hypothesis for Newton's method. *SIAM Journal on Numerical Analysis*, 6:493–507, 1969.

[28] J. E. Dennis and R. B. Schnabel. *Numerical Methods for unconstrained optimization and nonlinear equations.* Prentice Hall Inc, Englewood Cliffs, New Jersey 07632, 1985.

[29] R. Dentener and P. Crutzen. Reaction of N_2O_5 on tropospheric aerosols : impact of the global distributions of NO_x, O_3 and OH. *Journal of Geophysical Research*, 98:7149–7163, 1993.

[30] P. Deuflhard. Recent progress in extrapolation methods for ordinary differential equations. *SIAM Review*, 27:505–535, 1985.

[31] J. J. Dongarra, J. R. Bunch, C. B. Moller, and G. W. Stewart. LINPACK User's Guide. Technical report, SIAM, Philadelphia, PA, 1979.

[32] J.J. Dongarra and E. Grosse. Distribution of software via electronic mail. *Communications ACM*, pages 403–407, 1987.

[33] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford Science Publications, Clarendon Press Oxford, 1986.

[34] A. M. Dunker. The decoupled direct method for calculating sensitivity coefficients in chemical kinetics. *J. Chemical Physics*, 81:2385, 1984.

[35] S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman. Yale Sparse Matrix Package. ii. The nonsymmetric codes. Research Report 114, Department of Computer Science, Yale University, 1977.

[36] S.C. Eisenstat, M.C. Gursky, M.H. Schultz, and A.H. Sherman. Yale Sparse Matrix Package. i. The symmetric codes. *Int. J. Num. Meth. Eng.*, 18:1145–1151, 1982.

[37] A.S. El-Bakry, R.A. Tapia, T. Tsuchia, and Y. Zhang. On the Formulation of the Primal-Dual Newton Interior-Point Method for Nonlinear Programming. *To appear in Journal of Optimization Theory and Applications*, 1996.

[38] S. Elliot, R.P. Turco, and M.Z. Jacobson. Tests on combined projection/forward differencing integration for stiff photochemical family systems at long time step. *Computers Chem*, 17:91–102, 1993.

[39] ftp.cgrer.uiowa.edu. Ftp site. *Ftp site at The Center for Global and Regional Environmental Research, University of Iowa, (pub/Ode_benchmark, pub/KPP)*, 1996.

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

[41] G. Golub and C. F. van Loan. *Matrix computaions.* Johns Hopkins University Press, Baltimore and London, 1983.

[42] W. Gong and H.R. Cho. A numerical scheme for the integration of the gas phase chemical rate equations in 3D atmospheric models. *Atmospheric Environment*, 27A:2147–2160, 1993.

[43] A. Griewank, C. Bischof, G. Corliss, A. Carle, and K. Williamson. Derivative convergence for iterative equation solvers. *Optimization methods and software*, 2:321–355, 1993.

[44] A. Griewank and G. Corliss. Automatic differentiation of algorithms: Theory, implementation, and application. *SIAM, Philadelphia, Pennsylvania*, 1991.

[45] E. Hairer, Ch. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods.* Springer-Verlag, Berlin, New-York, 1989.

[46] E. Hairer, S.P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems.* Springer-Verlag, Berlin, 1993.

[47] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer-Verlag, Berlin, 1991.

[48] O. Hertel, R. Berkowicz, J. Christensen, and O. Hov. Test of two numerical schemes for use in atmospheric transport-chemistry models. *Atmospheric Environment*, 27A:2591–2611, 1993.

[49] E. Hesstvedt, O. Hov, and I. Isaacsen. A numerical method to predict secondary air pollutants with an application on oxidant generation in an urban atmosphere. *WMO publication*, 510:219–226, 1978.

[50] E. Hesstvedt, O. Hov, and I. Isaacsen. Quasi-steady-state-approximation in air pollution modelling: comparison of two numerical schemes for oxidant prediction. *Int. J. Chem. Kinet.*, 10:971–994, 1978.

[51] A. Hindmarsch. *ODEPACK: A systematized collection of ODE solvers.* Ed. North Holland, Amsterdam, 1983.

[52] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM Journal of Scientific Computing*, to appear, 1997.

[53] D.J. Jacob, J.A. Logan, G.M. Gardner, C.M. Spivakovsky R.M. Yevich, S.C. Wofsy, S. Sillman, and M.J. Prather. Factors regulating ozone over the United

States and its export to the global atmosphere. *J. Geophys. Res.*, 98:14,817–14,826, 1993.

[54] M.Z. Jacobson and R.P. Turco. SMVGEAR: a sparse-matrix, vectorized Gear code for atmospheric models. *Atmospheric Environment*, 17:273–284, 1994.

[55] L. O. Jay. Structure-Preserving Integrators. *University of Minnesota AHPCRC, Preprint 95-038*, 1995.

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

[57] R. J. Kee, F. M. Rupley, and J. A. Miller. CHEMKIN II: A FORTRAN package for the analysis of gas phase chemical kinetics. Technical report, Sandia National Laboratory, Livermore, CA, 1989.

[58] X. Lin, M. Trainer, and S. Liu. On the nonlinearity of tropospheric ozone production. *Journal of Geophysical Research*, 93:15,879–15,888, 1988.

[59] F.W. Lurmann, A.C. Loyd, and R. Atkinson. A chemical mechanism for use in long-range transport/acid deposition computer modeling. *Journal of Geophysical Research*, 91:10,905–10,936, 1986.

[60] J. Matthijsen. Private Communication. 1995.

[61] 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*, 45:1–42, 1982.

[62] Hoa D. Nguyen and Seungho Paik. Solution Domain decomposition with Finite Difference Methods for PDE. *Numerical methods for PDE*, 11:453–466, 1995.

[63] Jorge Nocedal. Theory of Algorithms for Unconstrained Optimization. *Acta Numerica*, pages 1–37, 1991.

[64] U. Nowak. A short user's guide to LARKIN. Technical report, Konrad-Zuse-Zentrum fuer, Informationstechnik Berlin, 1982.

[65] J. Olson, M. Prather, T. Berntsen, G. R. Carmichael, R. Chatfield, P. Connell, R. Derwent, L. Horowitz, S. Jin, M. Kanakidou, P. Kasibhatla, R. Kotomarthi, M. Kuhn, K. Law, S. Sillman, J. Penner, L. Perliski, F. Stordal, A. Thompson, and O. Wild. Results from the IPCC Photochemical Model Intercomparison (PhotoComp): Some Insights into Tropospheric Chemistry. *submitted to Journal of Geophysical Research*, March 1996.

[66] K. Olszyna, E. Bailey, R. Simonaites, and J. Meagher. O_3 and NO_y relationships at a rural site. *Journal of Geophysical Research*, 99:14,557–14,563, 1994.

[67] D. Parrish, J. Holloway, M. Trainer, P. Murphy, G. Forbes, and F. Fehsenfeld. Export of north american ozone pollution to the north atlantic ocean. *Science*, 259:1436–1439, 1993.

[68] F.A. Potra, K. Kortanek, and Y. Ye. On some efficient interior point methods for nonlinear convex programming. *Linear Algebra and its Applications*, 152:191–222, 1991.

[69] M.J.D. Powell. Convergence properties of algorithms for nonlinear optimization. Report DAMTP 1985/NA1, University of Cambridge, Department of Applied Mathematics and Theoretical Physics, Cambridge, October 1985.

[70] M. Prather. Intercomparison of tropospheric chemistry/ transport models. *Scientific assesment of ozone depletion, World meteorological organization*, 1995.

[71] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Math. of Comput.*, 28:145–162, 1974.

[72] H. Rabitz, M. Hramer, and D. Dacol. Sensitivity analysis in chemical kinetics. *Anual review of physical chemistry*, 34, 1983.

[73] D. Ralph and S. Wright. Superlinear convergence of an interior point method for monotone variational inequalities. *Preprint MCS-P556-0196*, Argonne National Laboratory, 1996.

[74] A. Sandu, J. G. Blom, E. Spee, J. G. Verwer, F.A. Potra, and G.R. Carmichael. Benchmarking stiff ODE solvers for atmospheric chemistry equations II - Rosenbrock Solvers. Report on Computational Mathematics 90, The University of Iowa, Department of Mathematics, Iowa City, July 1996.

[75] A. Sandu, F.A. Potra, V. Damian, and G.R. Carmichael. Efficient implementation of fully implicit methods for atmospheric chemistry. *Journal of Computational Physics*, 129:101 – 110, 1996.

[76] A. Sandu, M. van Loon, F.A. Potra, G.R. Carmichael, and J. G. Verwer. Benchmarking stiff ODE solvers for atmospheric chemistry equations I - Implicit vs. Explicit. Report on Computational Mathematics 85, The University of Iowa, Department of Mathematics, Iowa City, January 1996.

[77] R. D. Saylor and G. D. Ford. On the comparison of numerical methods for the integration of kinetic equations in atmospheric chemistry and transport models. *Atmospheric Environment*, 29:2585–2593, 1995.

[78] A.H. Sherman and A.C. Hindmarsh. GEARS: a package for the solution of sparse, stiff ordinary differential equations. *Lawrence Livermore Laboratory Report*, UCRL-84102.

[79] D. Shyan-Shu Shieh, Y. Chang, and G.R. Carmichael. The evaluation of numerical techniques for solution of stiff ODE arising from chemical kinetic problems. *Environmental Software*, 3, 1988.

[80] S. Sillman. A numerical solution for the equations of tropospheric chemistry based on an analysis of sources and sinks of odd hydrogen. *Journal of Geophysical Research*, 96:20735–20744, 1991.

[81] D. Simpson. Biogenic VOC in Europe. Part II: implications for ozone control strategies. *EMEP MSC-W*, 1994.

[82] D. Simpson, Y. Andersson-Skold, and M.E. Jenkin. Updating the chemical scheme for the EMEP MSC-W oxidant model: current status. *EMEP MSC-W*, Technical Report 2/93, 1993.

[83] S. Skelboe and Z. Zlatev. Exploiting the natural partitioning in the numerical solution of ODE systems arising from atmospheric chemistry. Report, University of Copenhagen, Department of Computer Science, Copenhagen, Denmark, 1996.

[84] D. Stoffer. Variable steps for reversible integration methods. *Computing*, 55:1–22, 1995.

[85] M. van Loon. Numerical smog prediction I: the physical and chemical model. *CWI Report NM-R9411*, 1995.

[86] M. van Loon. Numerical smog prediction II: Grid refinement and its application to the Dutch smog prediction model. *CWI Report NM-R95xx*, 1995.

[87] J. Verwer. Gauss-Seidel iterations for stiff ODEs from chemical kinetics. *SIAM Journal of Scientific Computing*, 15:1243–1250, 1994.

[88] J. Verwer, J. G. Blom, and W. Hunsdorfer. An Implicit-Explicit Approach for Atmospheric Transport-Chemistry Problems. *Applied Numerical Mathematics*, 20:191–209, 1996.

[89] J. 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.

[90] J. Verwer and W. Hunsdorfer. A note on Splitting Errors for Advection-Reaction Equations. *CWI Report NM-R9424*.

[91] J. Verwer and D. Simpson. Explicit Methods for Stiff Odes from Atmospheric Chemistry. *Applied Numerical Mathematics*, 18:413–430, 1995.

[92] J. Verwer and M. van Loon. An evaluation of explicit Pseudo-Steady-State Approximation achemes for stiff ODE systems from chemical kinetics. *Journal of Computational Physics*, 113:347–352, 1994.

[93] P. Werbos. Applications of advances in nonlinear sensitivity analysis. *System modelling and optimization*, Springer-Verlag:762–777, 1982.

[94] R. Yamartino, J. Scire, G.R. Carmichael, and Y.S. Chang. The CALGRID mesoscale photochemical grid model. *Atmospheric Environment*, 26 A:1493–1512, 1992.

[95] N. N. Yanenko. *The method of fractional steps*. Springer-Verlag, New-York, Heidelberg, Berlin, 1971.

[96] T. R. Young and J. P. Boris. A numerical technique for solving stiff ODE associated with the chemical kinetics of reactive flow problems. *Journal of Physical Chemistry*, 81:2424–2427, 1977.

[97] Z. Zlatev. *Computer Treatment of Large Air Pollution Models*. Kluwer Academic Publishers, 1995.