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

vorgelegt von
Ingenieur-Informatiker Hoang Duc Minh
aus Hue, Vietnam

Tag der mündlichen Prüfung: 22. Dezember 2005

Numerical Methods for Simulation and Optimization of Chemically Reacting Flows in Catalytic Monoliths

Gutachter: Prof. Dr. Dr. h. c. Hans Georg Bock
Gutachter: Prof. Dr. Olaf Deutschmann

Abstract

The aim of this work is to develop numerical methods and software for simulation and optimization of complex processes in catalytic monoliths to achieve better understanding of the physic-chemical processes in catalytic reactors.

The fluid dynamics are modelled by the boundary layer equations (BLEs), which are a large system of parabolic partial differential equations (PDEs) with highly nonlinear boundary conditions arising from the coupling of surface processes with the flow field inside the channel. The BLEs are obtained by simplifying the comprehensive model described by the Navier-Stokes equations and applying the boundary approximation theory. The surface and gas-phase chemical reactions are described by detailed models.

The PDEs are semi-discretized using the method of lines leading to a structured system of differential-algebraic equations (DAEs). The DAEs are integrated by an implicit method, based on backward differentiation formulas (BDF). We develop a new BDF code with tailored efficient and robust numerical methods by exploiting the structure, and by an appropriate scaling for ill-conditioned iteration matrices, and by computing consistent initial values. Efficient methods for computation of partial derivatives in the framework of automatic differentiation and of finite differences are introduced and compared. Our newly developed simulation tool is more stable than the existing simulation tool, and faster than by a factor of ten to more than 60, depending on the applications.

To improve the performance of catalytic reactors (e.g., maximizing gas conversion or selectivity) we can control certain process conditions, such as temperature at the catalyst wall or the ratio of catalytic active surface area to the geometric surface area or the gas composition, the temperature, or the velocity at the inlet of the catalyst. It is the first time that this problem is generally formulated as an optimal control problem constrained by a system of PDEs describing the chemical fluid dynamics process and additional constraints. The direct shooting approach in combination with sequential quadratic programming (SQP) method is used for solving the resulting optimal control problem. An efficient numerical method for computation of the derivatives required by the SQP method is introduced.

In addition, error analysis for the numerical Newton method is investigated in detail. We introduce a new error model. Based on our error model and analysis, the limiting accuracy of the solution of nonlinear equations by the numerical Newton method can be obtained.

Our newly developed software package for simulation and optimization can be applied to different reaction mechanisms and channel settings with dif-

ferent initial/boundary conditions. This software is applied to two practical applications: catalytic combustion of methane and conversion of ethane to ethylene. The numerical results are presented. The simulation software provides a useful tool for the validation of reactions mechanisms. The software package allows, e.g., for a better design and operation of the conversion of natural gas to higher hydrocarbons or the improvement of exhaust treatment in cars.

Kurzfassung

Ziel dieser Arbeit ist die Entwicklung numerischer Methoden und Programme zur Simulation und Optimierung komplexer Prozesse in katalytischen Monolithen, um die physikalisch-chemischen Prozesse in katalytischen Reaktoren besser verstehen zu können.

Die Strömungen werden mittels der Grenzschichtgleichungen modelliert. Sie bilden ein großes System von partiellen Differentialgleichungen (PDEs) mit hochgradig nichtlinearen Randbedingungen, die sich aus der Kopplung der Oberflächenprozesse mit dem Strömungsfeld innerhalb des Kanals ergeben. Die Grenzschichtgleichungen werden abgeleitet, indem das Navier-Stokes-Modell vereinfacht und die Grenzschichtnäherung angewendet wird. Die Beschreibung der Gasphasen- und Oberflächenreaktionen erfolgt durch detaillierte Modelle.

Die PDEs werden mit der Hilfe der Linienmethode semi-diskretisiert. Daraus ergibt sich ein differential-algebraisches Gleichungssystem. Das differential-algebraische Gleichungssystem wird durch eine implizite Methode integriert, die auf den "Backward-Differentiation-Formulae" (BDF) beruht. Es wird ein neuer BDF-Code mit speziell zugeschnittenen, effizienten und robusten numerischen Methoden entwickelt, der insbesondere alle Strukturen ausnutzt, schlecht-konditionierte Iterationsmatrizen geeignet skaliert und konsistente Anfangswerte berechnet. Effiziente Methoden zur Berechnung der partiellen Ableitungen im Rahmen der automatischen Differenzierung und der finiten Differenzen werden eingeführt und miteinander gekoppelt. Das neu entwickelte Simulationswerkzeug ist stabiler als das existierende und in Abhängigkeit der Anwendung 10 bis 60-mal schneller.

Durch Variation bestimmter Prozessparameter lässt sich das Verhalten katalytischer Reaktoren verbessern (z.B. durch Maximierung von Umsatz oder Selektivität). Dazu zählen die Temperatur an der Katalysatorwand, das Verhältnis von katalytisch aktiver und Gesamtoberfläche sowie die Gaszusammensetzung, die Temperatur und die Geschwindigkeit am Eingang des Katalysators. Dieses Problem wird zum ersten Mal als Optimierungsproblem allgemein formuliert, das durch ein System von PDEs dargestellt wird. Dabei beschreiben die PDEs die reaktive Strömung sowie zusätzliche Bedingungen. Zur Lösung des sich ergebenden Optimierungsproblems wird ein "direktes Schießverfahren" in Verbindung mit der Methode der sequentiellen quadratischen Programmierung (SQP) benutzt. Eine effiziente Vorgehensweise zur Berechnung der für die SQP-Methode erforderlichen Ableitungen wird dargestellt.

Ein neues Modell zur Fehleranalyse der Newton-Methode wird eingeführt. Dadurch lässt sich die maximal erzielbare Genauigkeit der Lösung von nicht-

linearen Gleichungen besser abschätzen.

Das neu entwickelte Softwarepaket zur Simulation und Optimierung eignet sich für verschiedene Reaktionsmechanismen und Kanäle mit verschiedenen Anfangs- und/oder Randbedingungen. Die Software wird exemplarisch für zwei Anwendungen eingesetzt: katalytische Verbrennung von Methan und Umsetzung von Ethan zu Ethylen. Die numerischen Ergebnisse werden dargestellt. Diese Simulationssoftware ist geeignet zur Validierung von Reaktionsmechanismen. Sie ermöglicht die Optimierung chemischer Prozesse, wie zum Beispiel die Umsetzung von Erdgas in wertvolle Kohlenwasserstoffe oder die Abgasnachbehandlung in Kraftfahrzeugen.

Acknowledgments

I would be extremely grateful to my advisors Prof. Dr. Dr. h.c. Hans Georg Bock, Prof. Dr. Dr. h.c. Jürgen Warnatz, Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, and Prof. Dr. Hoang Xuan Phu, Institute of Mathematics, Vietnamese Academy of Science and Technology, for supervising this interdisciplinary project, for continuous support and encouragement, and for many inspiring discussions. I am greatly indebted to Dr. Johannes Schlöder for many fruitful discussions and numerous valueable advices. His patience and availability for any help whenever needed with his heavy workload is appreciated.

I wish to thank Prof. Dr. Olaf Deutschman and Dr. Steffen Tischer, Institute for Chemical Technology and Polymer Chemistry, University of Karlsruhe, for many stimulating conversations and suggestions, for giving me the DETCHEM source code, which is partly used in this work, and for introducing me to many interesting practical applications.

I would like to thank my colleague and former roommate Dr. Stefan Körkel, for always having an open ear for my questions and for his friendly help with many problems in my social and study life in Germany. I would like to thank my colleagues in the group of Prof. Dr. Dr. h.c. Hans Georg Bock and Dr. Johannes Schlöder for their friendship and their assistance. In particular, I would like to mention Dr. Tran Van Hoai and Dr. Tran Hong Thai for many useful helps and for sharing the study life with me in Germany, and Dr. Moritz Diehl, Dr. Andreas Schäfer for their friendly helps on the software MUSCOD-II. Special thanks to Margret Rothfuß for helping me dealing with many documents.

I gratefully acknowledge the many helpful suggestions and comments of Prof. Dr. Robert J. Kee, Division of Engineering Colorado School of Mines, USA, especially during the time he visited the IWR. I also would like to thank Prof. Dr. Claudia Bruno, Department of Mechanics and Aeronautics, University of Rome, Italy and Prof. Dr. William E. Schiesser, Lehigh University, USA for useful conversations. I wish to thank my former advisor Dr. Nguyen Thanh Son, Department of Information Technology, HCMC Univer-

sity of Technology, who always encourages and supports me to continue my higher education.

I would like to thank Dr. Johannes Schlöder, Dr. Steffen Tischer, Dr. Moritz Diehl, Dr. Ekaterina Kostina, Kaspar Sakmann, and Nikolay Mladenov for proof-reading the thesis.

I am very much appreciate the finanical support from the German Science Foundation (DFG—Deutsche Forschungsgemeinschaft) within the Graduiertenkolleg program: "Complex Processes: Modeling, Simulation and Optimization", and SFB (Sonderforschungsbereich) 359 "Reactive Flows, Diffusion and Transport".

Most of all, I wish to thank my parents and my brothers and sisters for their love and support, and for always being there. They also always encourages me to continue my higher education, in which this research work could be carried out.

Contents

List of Tables

List of Figures

Nomenclature

Uppercase Latin Characters

Symbol	Meaning	SI units
D_k^{m}	mixture-averaged difusion coefficient of thee kth species in the mixture	$\mathrm{m}^2 \cdot \mathrm{s}^{-1}$
D_k^T	thermal diffusion coefficients	$\mathrm{kg} \cdot \mathrm{m}^{-1} \cdot \mathrm{s}^{-1}$
J_k	diffusion mass flux of the kth species	$\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
$J_{k,r}$	the radial component of the mass flux vector J	$\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
$J_{k,z}$	the axial component of the mass flux vector J	$\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \mathrm{s}^{-1}$
K_{g}	total number of gas-phase reactions	
K_{s}	total number of surface reactions	
N_{g}	total number of gas-phase species	
N_{s}	total number of surface species	
Pr	Prandtl number	
R	universal gas constant, $R = 8.313$	$\mathrm{J} \cdot \mathrm{mol}^{-1} \cdot \mathrm{K}^{-1}$
Sc_k	Schmidt number	
T	temperature	K
T_0	ambient temperature	K
T_{gas}	gas temperature	K
T_{wall}	wall temperature	K
Re_r	Reynolds number	
X_k	mole fraction of the kth species	
Y_k	mass fraction of the kth species	
W_k	molecular weight of the kth species	$\mathrm{kg} \cdot \mathrm{mol}^{-1}$
$\overline{W}$	mixture mean molecular weight	$\mathrm{kg} \cdot \mathrm{mol}^{-1}$

Lowercase Latin Characters

Symbol	Meaning	SI units
c_p	mixture specific heat	$\mathrm{J \cdot kg^{-1} \cdot K^{-1}}$
c_{pk}	specific heat at constant pressure of the kth species	$\mathrm{J \cdot kg^{-1} \cdot K^{-1}}$
h_k	specific heat enthapy of the kth species by surface reactions	$\mathrm{J \cdot kg^{-1}}$
p	pressure	Pa
$\dot{s}_k$	rate of production of the kth species by surface reactions	$\mathrm{mol \cdot m^{-2} \cdot s}$
u	axial velocity	$\mathrm{m \cdot s^{-1}}$
v	radial velocity	$\mathrm{m \cdot s^{-1}}$
v_{stef}	Stefan flow velocity	$\mathrm{m \cdot s^{-1}}$
r	radial spatial coordinate, independent variable	m
z	axial spatial coordinate, independent variable	m

Uppercase Greek Characters

Symbol	Meaning	SI units
Γ	site density	$\mathrm{mol/m^2}$
Θ_k	surface coverage of the kth surface species	

Lowercase Greek Characters

Symbol	Meaning	SI units
θ	circumferential coordinate	
λ	thermal/heat conductivity	$\mathrm{J \cdot m^{-1} \cdot K^{-1} \cdot s^{-1}}$
λ_k	thermal conductivity of the kth species	$\mathrm{J \cdot m^{-1} \cdot K^{-1} \cdot s^{-1}}$
μ	viscosity	$\mathrm{kg \cdot m^{-1} \cdot s^{-1}}$
ρ	mass density	$\mathrm{kg \cdot m^{-3}}$
χ_k	chemical symbol of the kth species	
$\dot{\omega}_k$	rate of production of the kth species by gas-phase reactions	$\mathrm{mol \cdot m^{-3} \cdot s^{-1}}$

Introduction

Catalysis is a viable technology to achieve ultra-low emission of NO_x, CO in applications like gas turbines for electric-power generation and catalytic burners for heating and drying (e.g., [12] and [106]). Today, catalysts are increasingly applied in industry in particular due to the concerns for environmental protection. A typical application of catalysis is the reduction of exhaust gas pollution in automotive catalytic converters.

The major concern is the need for better understanding of the physical-chemical processes in catalytic reactors, which is critical for improving the performance of catalytic reactors. This leads to the need for the development of robust and reliable numerical software which takes into account the modeling of fluid mechanics and the detailed models of chemical reactions. However, the use of detailed models for chemical reactions is still very challenging due to a large number of species involved, due to the nonlinearity, and due to the multiple time scales arising from the complex reaction systems, that leads to very large and stiff systems (one mass conservation equation for each species) of partial differential equations (PDEs) with highly nonlinear boundary conditions. Figure 1 shows a typical catalytic monolith and the physical-chemical processes in a single channel of this monolith.

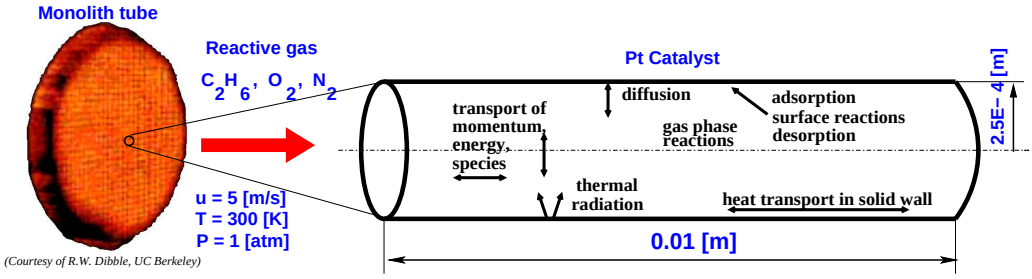

Figure 1: Catalytic monolith and dynamics processes in a single channel

The most comprehensive model for the coupled fluid mechanics and chemical kinetics in a channel of catalytic monoliths is provided by the steady-state Navier-Stokes equations coupled with detailed models for the chemical reac-

tions, which are a large system of *elliptic* partial differential-algebraic equations. Even for a medium size problem, the computational time for simulation is within the range of a few hours [101]. On the other hand, the chemically reacting flows can be modeled by using boundary layer approximation yielding a system of *parabolic* partial differential equations, a simplified version of the Navier-Stokes equations by using the boundary layer approximation theory. By numerical investigation, it is shown in [101] that the solutions of boundary layer and Navier-Stokes equations are in excellent agreement over a wide range of flow conditions. Therefore, in this work we use the boundary layer equations as our mathematical model along with detailed models for chemical reactions. For coupling the surface chemistry with the surrounding flow field we treat the nonlinear boundary conditions directly as algebraic constraints. Along with the semi-discretization of the PDEs describing the boundary layer model this results in a system of differential-algebraic equations (DAEs). The DAEs are solved by an implicit method, based on backward differentiation formula (BDF). The numerical methods for the solution of the DAEs are investigated, comprehending in particular scaling for ill-conditioned iteration matrices, exploiting the structure of the DAEs, computation of derivatives necessary for the solution of nonlinear equations arising in the BDF methods.

In addition, the error analysis for the numerical Newton method, which is used for solving the corrector equations arising in the BDF methods, is investigated in detail. We introduce a new error model and point out that some previous error models are inappropriate.

To improve the performance of catalytic reactors (e.g., maximizing gas conversion or selectivity) we can control certain process conditions, such as temperature at the catalyst wall T_{wall} or the ratio of catalytic active surface area to the geometric surface area $F_{\text{cat/geo}}$ or the gas composition, the temperature, the velocity at inlet of the catalyst. It is the first time that this problem is generally formulated as an optimal control problem constrained by a system of PDEs describing the chemical fluid dynamics process and additional constraints. The direct shooting approach combined with a sequential quadratic programming (SQP) method is used for solving the optimal control problem. An efficient numerical method for computation of derivatives required by the SQP method is introduced.

A few software packages for simulation of chemically reacting flows with detailed models have been developed. A popular one is the software package CHEMKIN developed at the Sandia National Laboratories in the 1980s and 1990s. It is now commercialized by Reaction Design, Inc [75]. The program CRESLAF [35] in the CHEMKIN library is designed for simulation of chemically reacting flows with detailed models. In CRESLAF, the boundary layer

equations are used to describe the fluid dynamics process. Another one is the software package DETCHEM [46], which was developed at the Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg and now at the Institute for Chemical Technology and Polymer Chemistry, University of Karlsruhe, for the simulation of reacting gaseous flows including complex models for heterogeneous reactions on solid surfaces. In particular, the DETCHEM$^{\text{CHANNEL}}$ software uses the boundary layer equations as a mathematical model for simulation of chemically reacting flows in a channel.

For the solution of DAEs, there are some BDF codes available, e.g, DAESOL [10], DASSL and DASPK [24], and DDASAC [28]. DAESOL was developled at the Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg. It has been sucessfully used for solving many practical applications in chemical engineering. Based on the DAESOL code, we develop a new DAESOLE code, which is tailored for treatment of the structured DAEs in our problem.

In this work, we focus on developing efficient numerical methods and software for simulation and optimization of the complex processes in a channel of catalytic monoliths. We have developed a numerical software package BLAYER, which consists of two programs BLAYER$^{\text{sim}}$ for simulation and BLAYER$^{\text{opt}}$ for optimization. The chemical reaction rates, heat capacities, entropies, enthalpies, heat conductivity, and diffusion coefficients are evaluated by appropriate calls to the code DETCHEM, which has been developed over years. To maintain compatibility with DETCHEM$^{\text{CHANNEL}}$, we use the same formats of reaction mechanisms and thermodynamics data as in DETCHEM$^{\text{CHANNEL}}$.

BLAYER can be applied to different reaction mechanisms and channel settings with different initial/boundary conditions. Given conditions at inlet (velocity, temperature, pressure, mass/mole fraction), the temperature and $F_{\text{cat/geo}}$ at the wall, geometry of the channel (length and radius), and gas- and surface-phase reaction mechanisms with thermodynamic data, BLAYER$^{\text{sim}}$ computes the flow field in the channel. In numerical investigations BLAYER$^{\text{sim}}$ proofed to be more stable and faster than the software DETCHEM$^{\text{CHANNEL}}$. Based on numerical tests with a medium sized problem, the speedup is about a factor of 10, for a large problem, the speedup is about a factor of 66 (see Chapter 5 for more details). Moreover, the solutions obtained by the software DETCHEM$^{\text{CHANNEL}}$, in particular surface species, display some abnormal phenomena (see Chapter 5 for more details), which would not allow the optimization to be realized.

BLAYER$^{\text{opt}}$ can be used for optimization with different controls: initial values (gas temperature, mass/mole fractions at inlet), and/or the temperature profile at the wall $T_{\text{wall}}(z)$, and $F_{\text{cat/geo}}(z)$. The objective to be minimized

can be the mass fraction of certain species or total amount of catalyst. Based on BLAYER$^{\text{opt}}$, other objectives and controls (e.g., inlet velocity, radius and length of the channel) can be easily realized.

The software package BLAYER allows, e.g., for a better design and operation of the conversion of natural gas to higher hydrocarbon or the improvement of exhaust treatment in cars.

Thesis outline

This is an interdisciplinary work, and thus we have written it in a style that it can address to different people from different disciplines. This thesis comprises 6 essentially independent chapters. To facilitate access to the individual topics, the chapters are rendered as self-contained as possible. The outline is as follows.

Chapter 1 is devoted to the modeling of fluid dynamics and chemical kinetics in a channel of catalytic monoliths. Various models from the time-dependent Navier-Stokes equations to the steady state boundary layer equations for fluid dynamics are discussed. Detailed models for gas-phase and surface chemical reactions are also described.

In Chapter 2, we investigate the numerical methods for DAEs arising from the semi-discretization of the PDE model. In particular, we discuss the BDF methods used for discretizing the DAEs, the solution of corrector equations, which is a system of nonlinear equations arising the BDF methods, automatic scaling of the iteration matrix arising the Newton iteration, error analysis for the numerical Newton method, and computation of derivatives and specially tailored methods for DAEs.

Chapter 3 deals with the solution method for the simulation problem. In particular, we go through the semi-discretization of the PDEs and treatment of nonlinear boundary conditions and numerical methods for solving the nonlinear equations imposed by the boundary conditions to obtain consistent initial values of the DAEs.

Chapter 4 concentrates on the treatment of the optimal control problem. First, practical optimization problems are discussed, then an optimal control problem is formulated. Numerical methods for the optimal control problem, in particular, the direct shooting approach combined with a SQP method is examined in detail. A method for efficient computation of derivatives necessary for the solution by the SQP method is presented.

In Chapter 5, we apply the our software BLAYER$^{\text{sim}}$ and BLAYER$^{\text{opt}}$ to two practical applications: catalytic combustion of methane and conversion of ethane to ethylene. A numerical comparison of the new simulation software

BLAYER$^{\text{sim}}$ and the existing DETCHEM$^{\text{CHANNEL}}$ is presented.

The thesis concludes in Chapter 6 with a summary of the obtained results and discussion of the contributions made, as well as suggestions for further research.

Chapter 1

Modeling of Chemically Reactive Flows

1.1 Introduction

In this thesis, we focus on modeling fluid flow in a single channel of a catalytic monolith, which composes of thousands of such channel, as a first step to study the complex physical-chemical processes in a complete monolith. In general, the flow field can be modeled by Navier-Stokes equations which are derived based on the fundamental physical principles from the laws of physics, e.g., mass is conserved, Newton's 2nd law, and energy is conserved. The governing equations have been derived and presented in many textbooks (e.g., [58], [124], [120] and [73]). Therefore, in the following sections, we will not re-derive them again but only present a set of governing equations with a brief description for readers easy to follow. Solving the Navier-Stokes equations requires tremendous computing time. Moreover, under our flow conditions the flow field can be well described by the boundary layer equations, which are a simplified version of the Navier-Stokes equations by applying the boundary layer theory. The closeness between the solutions of boundary layer equations and of Navier-Stokes equations are theoretically studied in [90] and [51] as cited in [92]. By numerical investigation, it is shown in [101] that the solutions of the boundary layer equations have an excellent agreement with the solution of Navier-Stokes equations for our flow conditions while the solution of the plug flow equations does not. Therefore, in this work we use the boundary layer equations for modeling the flow field, and in Section 1.6 we present in detail a derivation of them from the Navier-Stokes equations. We assume that the channel is axisymmetric, and in order to take the symmetry into account easily, we write the governing equations in cylindrical coordinates.

It is noticed that the Navier-Stokes equations and their variants can only describe fluid flow approximately. Certain assumptions have been made when deriving the equations. Therefore, under extreme conditions such as very small scales, the equations may not correctly describe the flows. The Navier-Stokes equations are essentially derived based on the assumption of the *continuum hypothesis*, which assumes that the fluid under consideration is a continuum. The hypothesis is perfectly reasonable as long as the macroscopic length and time scales are considerably larger than the largest length and time scales. However, the continuum hypothesis will eventually break down as the length and time scales of a particular problem approaches molecular scales. For example, the order of mean-free-path length is typically about 10^{-7} meters, and molecular diameters are typically of the order of a few 10^{-10} meters. Therefore, if the feature sizes is about 10^{-6} meters and the pressure is about 10^{-3} Pa, the continuum hypothesis can be questionable. For more details, see e.g., [73].

In the following, we use these notations.

- t is the time.
- z, r, and θ are the three components of the cylindrical coordinates.
- u, v, and w are the axial, radial, circumferential components of the velocity vector.
- p is the pressure.
- T is the temperature.
- Y_k is the mass fraction of the kth species.
- μ is the viscosity.
- ρ is the mass density.
- c_p is the heat capacity of mixture.
- λ is the thermal conductivity of mixture.
- c_{pk} is the specific heat capacity of the kth species.
- $J_{k,z}$, $J_{k,r}$, and $J_{k,\theta}$ are the axial, radial, circumferential components of the mass flux vector.
- $\dot{\omega}_k$ is the rate of creation of the kth species by the gas phase reactions.

– h_k is the specific heat enthalpy of the kth species.

– W_k is the molecular weight of the kth species.

1.2 Transient three-dimensional Navier–Stokes equations

The transient three-dimensional Navier–Stokes equations in their general form, which are written in cylindrical coordinates, are presented. These governing equations are for time dependent, transient problems. These equations are derived by applying conservation laws to a certain region, called *control volume* or a *fluid element*. The principle of *mass conservation* is described by the *overall mass continuity equation* which is written in the differential form as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial z} + \frac{1}{r}\frac{\partial (r\rho v)}{\partial r} + \frac{1}{r}\frac{\partial (\rho w)}{\partial \theta} = 0. \tag{1.1}$$

In the chemically reacting flow of a gas mixture, the mass conservation law applied to the mass m_k of the kth species leads to the following equation:

$$\begin{aligned} &\frac{\partial \rho_k}{\partial t} + \frac{\partial \rho_k u}{\partial z} + \frac{1}{r}\frac{\partial (r\rho_k v)}{\partial r} + \frac{1}{r}\frac{\partial (\rho_k w)}{\partial \theta} \\ &+ \left(\frac{\partial J_{k,z}}{\partial z} + \frac{1}{r}\frac{\partial r J_{k,r}}{\partial r} + \frac{1}{r}\frac{\partial J_{k,\theta}}{\partial \theta} \right) = \dot{\omega}_k W_k, \end{aligned} \tag{1.2}$$

where ρ_k is the density of the kth species (the mass of kth species per unit volume). The term $\dot{\omega}_k W_k$ on the left-hand side of the above equation is the net mass rate of production of the kth species due to the homogeneous chemical reactions.

By definition $\rho_k = \rho Y_k$, where Y_k is the mass fraction of the kth species in the mixture ($Y_k = m_k/m$ wheren m is the total mass of fluid), and subtracting the overall mass continuity equation (1.1) from (1.2), we obtain the following *species mass continuity* equation

$$\begin{aligned} &\rho \frac{\partial Y_k}{\partial t} + \rho \left(u\frac{\partial Y_k}{\partial z} + v\frac{\partial Y_k}{\partial r} + \frac{w}{r}\frac{\partial Y_k}{\partial \theta} \right) \\ &+ \left(\frac{\partial J_{k,z}}{\partial z} + \frac{1}{r}\frac{\partial r J_{k,r}}{\partial r} + \frac{1}{r}\frac{\partial J_{k,\theta}}{\partial \theta} \right) = \dot{\omega}_k W_k. \end{aligned} \tag{1.3}$$

Here $J_{k,z}$, $J_{k,r}$ and $J_{k,\theta}$ are three corresponding components of the diffusive mass flux vector $\overrightarrow{J_k}$ of the kth species.

Now we sum Equation (1.3) over all $N_{\rm g}$ species and noting that $\sum_{k=1}^{N_{\rm g}} Y_k = 1$ by definition, and $\sum_{k=1}^{N_{\rm g}} \dot{\omega}_k W_k = 0$ because chemical reactions neither create nor destroy mass, we obtain the condition,

$$\sum_{k=1}^{N_{\rm g}} \overrightarrow{J_k} = 0. \tag{1.4}$$

The diffusive mass flux $\overrightarrow{J_k}$ are calculated based on the gradient of concentration (*Fick's law*) and the gradient of temperature (*thermal diffusion*), its three components are calculated as

$$\begin{aligned} J_{k,z} &= -\rho \frac{W_k}{\overline{W}} D_k^{\rm m} \frac{\partial X_k}{\partial z} - D_k^T \frac{1}{T} \frac{\partial T}{\partial z} \\ J_{k,r} &= -\rho \frac{W_k}{\overline{W}} D_k^{\rm m} \frac{\partial X_k}{\partial r} - D_k^T \frac{1}{T} \frac{\partial T}{\partial r} \\ J_{k,\theta} &= -\rho \frac{W_k}{\overline{W}} D_k^{\rm m} \frac{\partial X_k}{\partial \theta} - D_k^T \frac{1}{T} \frac{\partial T}{\partial \theta}, \end{aligned}$$

where $D_k^{\rm m}$ is the mixture-averaged diffusion coefficient of the kth species into the mixture, W_k and $\overline{W}$ are the molar masses of the kth species and of the mixture, respectively, and D_k^T is the thermal diffusion coefficient of the kth species, X_k is the mole fraction of kth species defined by $X_k = c_k/c$ where c_k and c are the concentration of the kth species and the total concentration of the mixture, respectively. One can convert between the mole fraction and the mass fraction using the following easily derived relations:

$$X_k = \frac{1}{\sum_{j=1}^{N_{\rm g}} Y_j/W_j} \frac{Y_k}{W_k} = \frac{\overline{W}}{W_k} Y_k$$

and

$$Y_k = \frac{W_k}{\overline{W}} X_k = \frac{W_k}{\sum_{j=1}^{N_{\rm g}} X_j W_j} X_k.$$

As mass fraction, Y_k must satisfy

$$0 \le Y_k \le 1 \quad (k = 1, \ldots, N_{\rm g}), \quad \sum_{k=1}^{N_{\rm g}} Y_k = 1. \tag{1.5}$$

Newton's second law says that the net force on the fluid element equals its mass times the acceleration of the element, i.e., $\overrightarrow{F} = m\overrightarrow{a}$, which leads to the three following equations.

Axial momentum:

$$\begin{aligned}
&\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial z}+v\frac{\partial u}{\partial r}+\frac{w}{r}\frac{\partial u}{\partial \theta}\right)\\
=\;& -f_z-\frac{\partial p}{\partial z}+\frac{\partial}{\partial z}\left[2\mu\frac{\partial u}{\partial z}+\kappa\nabla\cdot V\right]+\frac{1}{r}\frac{\partial}{\partial r}\left[\mu r\left(\frac{\partial v}{\partial z}+\frac{\partial u}{\partial r}\right)\right]\\
&+\frac{1}{r}\frac{\partial}{\partial \theta}\left[\mu\left(\frac{1}{r}\frac{\partial u}{\partial \theta}+\frac{\partial w}{\partial z}\right)\right], \qquad (1.6)
\end{aligned}$$

where $\nabla\cdot V$ is the divergence of the velocity field

$$\nabla\cdot V=\frac{\partial u}{\partial z}+\frac{1}{r}\frac{\partial rv}{\partial r}+\frac{1}{r}\frac{\partial w}{\partial \theta}$$

and f is the body force, and f_z, f_r, and f_θ are the axial, radial, and circumferential components of f, respectively.
Radial momentum:

$$\begin{aligned}
&\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{z}+v\frac{\partial v}{r}+\frac{w}{r}\frac{\partial v}{\partial \theta}-\frac{w^2}{r}\right)\\
=\;& f_r-\frac{\partial p}{\partial r}+\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial v}{\partial z}+\frac{\partial u}{\partial r}\right)\right]+\frac{\partial}{\partial r}\left[2\mu\frac{\partial v}{\partial r}+\kappa\nabla\cdot V\right]\\
&+\frac{1}{r}\frac{\partial}{\partial \theta}\left[\mu\left(\frac{1}{r}\frac{\partial v}{\partial \theta}+\frac{\partial w}{\partial r}-\frac{w}{r}\right)\right]+\frac{2\mu}{r}\left[\frac{\partial v}{r}-\frac{1}{r}\frac{\partial w}{\partial \theta}-\frac{v}{r}\right] \qquad (1.7)
\end{aligned}$$

Circumferential momentum:

$$\begin{aligned}
&\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial z}+v\frac{\partial w}{\partial r}+\frac{w}{r}\frac{\partial w}{\partial \theta}+\frac{vw}{r}\right)\\
=\;& f_\theta-\frac{1}{r}\frac{\partial p}{\partial \theta}+\frac{\partial}{\partial z}\left[\mu\left(\frac{1}{r}\frac{\partial u}{\partial \theta}+\frac{\partial w}{\partial z}\right)\right]\\
&+\frac{\partial}{\partial r}\left[\mu\left(\frac{1}{r}\frac{\partial v}{\partial \theta}+\frac{\partial w}{\partial r}-\frac{w}{r}\right)\right]\\
&+\frac{1}{r}\frac{\partial}{\partial \theta}\left[\frac{2\mu}{r}\frac{\partial w}{\partial \theta}+\kappa\nabla\cdot V\right]+\frac{2\mu}{r}\left[\frac{1}{r}\frac{\partial v}{\partial \theta}+\frac{\partial w}{\partial r}-\frac{w}{r}\right] \qquad (1.8)
\end{aligned}$$

The first law of thermodynamics states that energy is conserved, which

leads to the following equation

$$\begin{aligned}
&\rho c_p \frac{\partial T}{\partial t} + \rho c_p \left(u \frac{\partial T}{\partial z} + v \frac{\partial T}{\partial r} + \frac{w}{r} \frac{\partial T}{\partial \theta} \right) \\
= \;& \frac{\partial p}{\partial t} + \left(u \frac{\partial p}{\partial z} + v \frac{\partial p}{\partial r} + \frac{w}{r} \frac{\partial p}{\partial \theta} \right) \\
&+ \frac{\partial}{\partial z} \left(\lambda \frac{\partial T}{\partial z} \right) + \frac{1}{r} \frac{\partial}{\partial r} \left(r \lambda \frac{\partial T}{\partial r} \right) + \frac{1}{r^2} \frac{\partial}{\partial \theta} \left(\lambda \frac{\partial T}{\partial \theta} \right) \\
&- \sum_{k=1}^{N_\mathrm{g}} c_{pk} \left(J_{k,z} \frac{\partial T}{\partial z} + J_{k,r} \frac{\partial T}{\partial r} + \frac{J_{k,\theta}}{r} \frac{\partial T}{\partial \theta} \right) - \sum_{k=1}^{N_\mathrm{g}} h_k \dot{\omega}_k W_k. \qquad (1.9)
\end{aligned}$$

The relation between the density ρ, the pressure p, the molar mass of the mixture, and the temperature is called the *state equation*. For gaseous flows, we can use the ideal gas equation

$$p = \frac{\rho R T}{\overline{W}}$$

where R is the universal gas constant, $R = 8.314\ [\mathrm{J} \cdot \mathrm{mol}^{-1} \cdot \mathrm{K}^{-1}]$. Although the above state equation already provides an accurate representation for gases at low pressure, as in our case. There are also certain circumstances, one needs to use other relations for real gases, see e.g., [73].

In addition, heat transfer and mass transfer (diffusion of gas species) in the solid wall (catalyst wall) can also be modeled (see e.g., [52] and [29]). However, for the adiabatic cases we also solve the heat balance equations at the wall.

Boundary conditions at a gas-surface interface

The chemical processes at the catalytic surface are coupled with the flow field inside channel by the following boundary conditions at the gas-surface interface, which describes the fundamental physical principle that mass is conserved applied to the control volume V_gas adjacent to the surface.

$$\begin{aligned}
\int \rho \frac{\partial Y_k}{\partial t} \mathrm{d}V_\mathrm{gas} = \;& - \int (\overrightarrow{J_k} + \rho v \overrightarrow{v}_\mathrm{stef} Y_k) \overrightarrow{n} \mathrm{d}A \\
& + \int \dot{s}_k W_k F_\mathrm{cat/geo} \mathrm{d}A + \int \dot{\omega}_k W_k \mathrm{d}V_\mathrm{gas} \quad (k = 1, \ldots, N_\mathrm{g}),
\end{aligned} \qquad (1.10)$$

where $\overrightarrow{n}$ is the outward-pointing unit vector normal to the surface and $\overrightarrow{v}_\mathrm{stef}$ is the so-called *Stefan-velocity*, and $F_\mathrm{cat/geo}$ is the ratio of catalytic active surface area to geometric surface area. The first term on the left-hand side

of (1.10) is the total time rate of change of the mass of the kth species inside the control volume V_{gas}. The first term on the right-hand side of (1.10) is the net mass of the kth species flow out/in of the control volume V_{gas} through surface A due to convection and diffusion. The second term on the right-hand side of (1.10) is the net mass of the kth species due to the creation or depletion at the surface A by the surface reactions. The third term on the right-hand side of (1.10) is the net mass of the kth species due to gas-phase reactions inside the control volume V_{gas}.

The Stefan-velocity occurs at the surface if there is a net mass flux between the surface and the gas phase. Taking the sum of (1.10) over k species (all gas-phase species) and using the identities $\sum_{k=1}^{N_{\text{g}}} Y_k = 1$, $\sum_{k=1}^{N_{\text{g}}} \overrightarrow{J_k} = 0$ and $\sum_{k=1}^{N_{\text{g}}} \dot{\omega}_k W_k = 0$ (due to conservation of mass), we obtain

$$\overrightarrow{v}_{\text{stef}} \overrightarrow{n} = \frac{1}{\rho} \sum_{k=1}^{N_{\text{g}}} \dot{s}_k W_k. \tag{1.11}$$

For modeling a single channel of catalytic monoliths, we assume that the channel geometry is symmetric around the axial axis z, and also assume that with certain initial conditions and boundary conditions the flows in the channel is symmetric around axis z. Therefore, the third components (on the θ-axis) and partial derivatives with respect to θ of the all quantities, e.g., the velocity vector and mass flux vector, etc., vanish. Taking into account the symmetry around the axis z, we obtained a simplifying governing equations by eliminating terms containing w and θ from the above equations. For example, the fourth term of equation (1.1) vanishes, the overall mass continuity equation becomes

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial z} + \frac{1}{r} \frac{\partial (r \rho v)}{\partial r} = 0. \tag{1.12}$$

Similarly, we obtain the following equations.

Axial momentum:

$$\begin{aligned} \rho \left(\frac{Du}{Dt} \right) &= \rho \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial z} + v \frac{\partial u}{\partial r} \right) \\ &= -f_z - \frac{\partial p}{\partial z} + \frac{\partial}{\partial z} \left[2\mu \frac{\partial u}{\partial z} + \kappa \nabla \cdot V \right] + \frac{1}{r} \frac{\partial}{\partial r} \left[\mu r \left(\frac{\partial v}{\partial z} + \frac{\partial u}{\partial r} \right) \right] \end{aligned}$$

Radial momentum:

$$\begin{aligned}\rho\left(\frac{Dv}{Dt}\right) &= \rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{z}+v\frac{\partial v}{r}\right)\\ &= f_r-\frac{\partial p}{\partial r}+\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial v}{\partial z}+\frac{\partial u}{\partial r}\right)\right]+\frac{\partial}{\partial r}\left[2\mu\frac{\partial v}{\partial r}+\kappa\nabla\cdot V\right]\\ &\quad+\frac{2\mu}{r}\left[\frac{\partial v}{r}-\frac{v}{r}\right]\end{aligned}$$

Species mass continuity:

$$\begin{aligned}\rho\frac{DY_k}{Dt} &= \rho\left(\frac{\partial Y_k}{\partial t}+u\frac{\partial Y_k}{\partial z}+v\frac{\partial Y_k}{\partial r}\right)\\ &= -\left(\frac{\partial J_{k,z}}{\partial z}+\frac{1}{r}\frac{\partial r J_{k,r}}{\partial r}\right)+\dot{\omega}_k W_k\end{aligned}$$

Thermal energy:

$$\begin{aligned}\rho c_p\left(\frac{\partial T}{\partial t}+u\frac{\partial T}{\partial z}+v\frac{\partial T}{\partial r}\right) = &\quad\left(\frac{\partial p}{\partial t}+u\frac{\partial p}{\partial z}+v\frac{\partial p}{\partial r}\right)\\ &+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda\frac{\partial T}{\partial r}\right)\\ &-\sum_{k=1}^{N_g} c_{pk}\left(J_{k,z}\frac{\partial T}{\partial z}+J_{k,r}\frac{\partial T}{\partial r}\right)\\ &-\sum_{k=1}^{N_g} h_k\dot{\omega}_k W_k.\end{aligned}$$

1.3 Modeling of chemical reactions

Remember that in the species mass continuity equation (1.2) there is a chemical source term $\dot{\omega}_k W_k$ which is the net mass of production of the kth species due to the homogeneous chemical reactions. We use detailed models for describing the gas and surface chemical reactions (see e.g., [118] and [45]).

1.3.1 Gas-phase reactions

A chemical reaction involving N_g species can be represented in the following general form

$$\sum_{k=1}^{N_g}\nu'_k\chi_k \rightarrow \sum_{k=1}^{N_g}\nu''_k\chi_k, \tag{1.13}$$

where ν'_k and ν''_k are the stoichiometric coefficients of the kth species and χ_k is the chemical symbol for the kth species.

A reaction is called an *elementary reaction* if it occurs on a molecular level exactly the same as described by the reaction equation, otherwise the reaction is called *global reaction, overall reaction, complex reaction*, or *net reaction* [118]. An elementary reaction involves only a small number of molecules or ions. Another definition of an elementary reaction, which is close to this one, is given in [88] as follows. *"A reaction for which no* ***reaction intermediates*** *have been detected or need to be postulated in order to describe the chemical reaction on a molecular scale. An elementary reaction is assumed to occur in a single step and to pass through a single* ***transition state****"*. Hence, a global reaction usually takes place via a series of elementary reactions.

The rate of creation or consumption of a species in a chemical reaction is called *reaction rate* and is described by the *rate law* (empirical differential rate equation) which is an expression representing the rate of reaction in terms of concentrations of chemical species and constant parameters (normally *rate coefficients*) and partial *orders of reactions* only. It is written as

$$\frac{\mathrm{d}c_k}{\mathrm{d}t} = \nu_k k_\mathrm{f} \prod_{i=1}^{N_\mathrm{g}} c_i^{a'_i}, \tag{1.14}$$

where

$$\nu_k = \nu''_k - \nu'_k,$$

and c_k denotes the concentration of the kth species, k_f is the forward rate coefficient or the *rate constant*, and a'_i is the *reaction order* with respect to the ith species. Global reactions have complex rate laws where reaction orders may be non-integers and depend on time and reaction conditions. On the other hand, elementary reactions always have integer reaction orders that are valid for all experimental conditions. For an elementary reaction as (1.13) we have $a'_i = \nu'_i$, then the general rate expression (1.14) now becomes

$$\frac{\mathrm{d}c_k}{\mathrm{d}t} = \nu_k k_\mathrm{f} \prod_{i=1}^{N_\mathrm{g}} c_i^{\nu'_i}. \tag{1.15}$$

The forward rate constant of the ith reaction $k_{\mathrm{f}i}$ is determined using the following *modified Arrhenius equation*

$$k_{\mathrm{f}i} = A_i T^{\beta_i} \exp\left(-\frac{E_i}{RT}\right), \tag{1.16}$$

where

A_i is the pre-exponential factor of the ith reaction,

β_i is the temperature exponent of the ith reaction,

E_i is the activation energy of the ith reaction,

R is the universal gas constant, $R = 8.314\ [\mathrm{J \cdot mol^{-1} \cdot K^{-1}}]$,

T is the temperature.

For the reverse reaction of the reaction (1.13)

$$\sum_{k=1}^{N_\mathrm{g}} \nu_k'' \chi_k \rightarrow \sum_{k=1}^{N_\mathrm{g}} \nu_k' \chi_k, \tag{1.17}$$

the rate law can be obtained similarly to (1.15)

$$\frac{\mathrm{d}c_k}{\mathrm{d}t} = -\nu_k k_\mathrm{r} \prod_{i=1}^{N_\mathrm{g}} c_i^{a_i''}, \tag{1.18}$$

for elementary reactions $a_i'' = \nu_i''$, where k_r is the backward rate coefficient.

The relation between the forward and backward rate coefficients k_f and k_r is derived based on the chemical equilibrium. That is at the chemical equilibrium the forward and backward reactions have the same rate on a microscopic level,

$$\nu_k k_\mathrm{f} \prod_{i=1}^{N_\mathrm{g}} c_i^{\nu_i'} = \nu_k k_\mathrm{r} \prod_{i=1}^{N_\mathrm{g}} c_i^{\nu_i''}, \tag{1.19}$$

which means, no net reaction rate can be observed on a macroscopic level. The ratio

$$\frac{k_\mathrm{f}}{k_\mathrm{r}} = \prod_{i=1}^{N_\mathrm{g}} c_i^{\nu_i}$$

is called the *equilibrium constant* K_c of the reaction. The equilibrium constant K_c is determined from the thermodynamic properties

$$\begin{aligned} K_\mathrm{c} &= \left(\frac{p^0}{RT}\right)^{\sum_{i=1}^{N_\mathrm{g}} \nu_i} K_p, \\ K_p &= \exp\left(\frac{\Delta S^0}{R} - \frac{\Delta H^0}{RT}\right) \end{aligned} \tag{1.20}$$

with the molar entropy of the reaction

$$\Delta S^0 = \sum_{i=1}^{N_g} \nu_i S_i^0$$

and the molar enthalpy of the reaction

$$\Delta H^0 = \sum_{i=1}^{N_g} \nu_i H_i^0.$$

Here, $p^0 = 1$ bar is the standard pressure, S_i^0 and H_i^0 are the standard molar entropy and standard molar enthalpy, respectively, of the ith species involved in the reaction.

For a system of K_g (irreversible) elementary reactions involving N_g chemical species, where both the forward and reverse reactions are considered as individual elementary reactions, the net production rate of the kth species, denoted by $\dot{\omega}_k$, equals the sum of the rate of production of the kth species for all reactions involving the kth species, that is,

$$\frac{\mathrm{d}c_k}{\mathrm{d}t} = \dot{\omega}_k = \sum_{i=1}^{K_g} \nu_{ki} k_{fi} \prod_{j=1}^{N_g} c_j^{\nu'_{ji}}. \tag{1.21}$$

Here, $\nu_{ki} = \nu''_{ki} - \nu'_{ki}$ is the stoichiometric coefficient of the ith reaction involving the kth species; the first subscript refers to the reaction number and the second one refers to the species, this rule is also applied for ν'_{ji}, and k_{fi} is the rate coefficient of the ith species. Note that the expression (1.21) is a general one. In fact an elementary reaction involves only three or four species. Therefore, the term $c_j^{\nu'_{ji}}$ in (1.21) should be included if the jth species is involved in the ith reaction otherwise takes the value of one. Similarly, the term $(\nu_{ki} k_{fi} \prod_{j=1}^{N_g} c_j^{\nu'_{ji}})$ should only be covered if the kth species is involved in the ith reaction.

1.3.2 Surface reactions

At the catalytic surface, a gas-phase species may (e.g., O_2) be adsorbed on the surface becoming a *surface species* (e.g., O_2(s)). This process is called *adsorption*. The adsorbed species may decompose to yield either gas-phase products or other surface species, or they may react with the substrate to yield a specific surface compound. The adsorbed species which are present on a surface may desorb from the surface and return into the gas phase, this is called the *desorption* process. The reactions on the solid surface can

be between surface species or between surface species and gas-phase species. The state of the catalytic surface is described by the temperature T and a set of *surface coverages* Θ_k that is the fraction of the surface covered by the kth surface species. The surface structure is characterized by the *surface site density* Γ (in mol/m^2) that describes the maximum number of molar species that can be adsorbed on a unit surface area. Then each surface species, say the ith species, is characterized by a number σ_i that is the number of sites that the ith species occupies.

Similarly to the gas-phase reactions, a chemical reaction which includes adsorption, surface reaction, desorption, on the surface involving N_g gas-phase species and N_s surface species, can be represented in the general form

$$\sum_{k=1}^{N_\mathrm{g}+N_\mathrm{s}} \nu'_k \chi_k \rightarrow \sum_{k=1}^{N_\mathrm{g}+N_\mathrm{s}} \nu''_k \chi_k, \tag{1.22}$$

Also similarly to the gas-phase reactions, the molar net production rate of the kth species (a gas-phase species or a surface species), denoted by $\dot{s}_k$, due to the heterogeneous reactions on the solid surface is the sum of the rate of production for all reactions involving the kth species:

$$\frac{\mathrm{d}c_k}{\mathrm{d}t} = \dot{s}_k = \sum_{i=1}^{K_\mathrm{s}} \nu_{ki} k_{\mathrm{f}i} \prod_{j=1}^{N_\mathrm{g}+N_\mathrm{s}} c_j^{\nu'_{ji}}. \tag{1.23}$$

Here, K_s is the number of surface reactions, c_k is the molar concentration of the kth species. For gas-phase species the molar concentration c_k (in mol/m^3) can be calculated by using

$$c_k = \frac{\rho Y_k}{W_k}, \tag{1.24}$$

where ρ is the gas-phase mass density, Y_k and W_k are the mass fraction and the molecular weight of the kth species, respectively. For surface species, the surface molar concentration of the kth species c_k (in mol/m^2), $k = N_\mathrm{g} + i$, is computed by using

$$c_k = \frac{\Gamma \Theta_i}{\sigma_i}, \quad i = k - N_\mathrm{g}. \tag{1.25}$$

From (1.23) and (1.25), we have

$$\frac{\mathrm{d}\Theta_i}{\mathrm{d}t} = \frac{\dot{s}_k \sigma_i}{\Gamma}, \quad k = N_\mathrm{g} + i, \quad (i = 1, \ldots, N_\mathrm{s}). \tag{1.26}$$

The sum of the surface fractions of all species including the solid surface species (the solid surface itself that is the uncovered site of the solid surface is also counted as a surface species) equals to one, that is,

$$\sum_{i=1}^{N_\mathrm{s}} \Theta_i = 1, \quad 0 \le \Theta_i \le 1 \quad (i = 1, \ldots, N_\mathrm{s}). \tag{1.27}$$

For the forward rate coefficient $k_{\mathrm{f}i}$ of the ith reaction, some are calculated by the modified Arrhenius formula (1.16) as in the gas-phase reactions, some others are computed by using the following formula (see e.g., [34])

$$k_{\mathrm{f}i} = A_i T^{\beta_i} \exp\left(-\frac{E_i}{RT}\right) \prod_{k=1}^{N_\mathrm{s}} \Theta_k^{\mu_{ki}} \exp\left(\frac{\epsilon_{ki}\Theta_k}{RT}\right). \tag{1.28}$$

Here, μ_{ki} and ϵ_{ki} are surface parameters for the kth species in the ith reaction, which describe the dependence of the rate coefficients on the surface coverage of the kth species.

Similarly to the gas-phase reactions, the rate coefficient k_r of the reverse reaction is determined by

$$k_\mathrm{r} = k_\mathrm{f}/K_\mathrm{c}.$$

Here, K_c is the equilibrium constant which is calculated by using the following relation:

$$K_\mathrm{c} = K_p \left(\frac{p^0}{RT}\right)^{\sum_{i=1}^{N_\mathrm{g}} \nu_i} \Gamma^{\sum_{i=N_\mathrm{g}+1}^{N_\mathrm{g}+N_\mathrm{s}} \nu_i} \prod_{i=N_\mathrm{g}+1}^{N_\mathrm{g}+N_\mathrm{s}} \sigma_i^{-\nu_i}$$

with K_p is computed as (1.20) in the gas-phase reactions.

Remark 1.3.1
The rates of gas phase and surface reactions (1.21) and (1.23) depend polynomially on the molar concentrations and exponentially on the temperature and the surface coverages (see expressions (1.16) and (1.28)). This introduces nonlinearity into the system.

The detailed chemistry models usually involve many species, many of them are free radicals which have very small characteristic time scales and are usually governed by fast reactions, and others have larger time scales and are governed by relatively slow reactions. This makes the system stiff.

1.4 Thermodynamic and transport properties

The viscosity μ, the mixture heat conductivity λ, the enthalpy and heat capacity of the kth species h_k and c_{pk}, and the diffusion coefficients D_k^{m}, appearing in Equations (1.6)–(1.9) are calculated as follows.

The enthalpy and heat capacity of the kth species are determined by a fitted polynomial of the temperature T as follows

$$\begin{aligned} c_{pk} &= R(h_{1,k} + h_{2,k}T + h_{3,k}T^2 + h_{4,k}T^3 + h_{5,k}T^4) \\ h_k &= \int_0^T c_{pk}\mathrm{d}\overline{T} = RT(h_{1,k} + h_{2,k}\frac{T}{2} + h_{3,k}\frac{T^2}{3} + h_{4,k}\frac{T^3}{4} + h_{5,k}\frac{T^4}{5} + \frac{h_{6,k}}{T}) \end{aligned}$$

and the enthalpy and heat capacity of the mixture are evaluated by

$$h = \sum_{k=1}^{N_{\mathrm{g}}} h_k Y_k, \quad c_p = \sum_{k=1}^{N_{\mathrm{g}}} c_{pk} Y_k,$$

where R is the universal gas constant, $h_{i,k}$ are the polynomial coefficients, Y_k is the mass fraction of kth species. We use two polynomial coefficient sets $h_{i,k}$, one for temperatures below 1000 [K], and another for temperatures greater than or equal 1000 [K].

The pure species viscosity of the kth species is determined based on the logarithm of its value, which is evaluated by the fourth order fitted polynomial of the logarithm of the temperature

$$\ln(\mu_k) = \sum_{i=1}^{5} a_{i,k} \ln(T)^{n-1}.$$

Then, the mixture viscosity μ is determined based on the pure species viscosities

$$\mu = \frac{1}{2}\left(\sum X_k \mu_k + \frac{1}{\sum_{k=1}^{N_{\mathrm{g}}} X_k/\mu_k}\right),$$

where $a_{i,k}$ are the coefficients of the fitted polynomial, which must be determined for each problem, X_k is the mole fraction of the kth species, μ_k is the pure species viscosity of the kth species, and μ is the mixture viscosity.

Similarly, the thermal conductivity λ can be determined by

$$\ln(\lambda_k) = \sum_{i=1}^{5} b_{i,k} \ln(T)^{n-1}$$

and the mixture heat conductivity is determined based on the species heat conductivities as follows

$$\lambda = \frac{1}{2}\left(\sum X_k\lambda_k + \frac{1}{\sum_{k=1}^{N_g} X_k/\lambda_k}\right),$$

where $b_{i,k}$ are the coefficients of the fitted polynomial, which must be determined for each problem, X_k is the mole fraction of the kth species, λ_k is the heat conductivity of the kth species, λ is the mixture heat conductivity.

Alternatively, the mixture heat conductivity can also be determined by Wilke's formula as follows

$$\lambda = \sum_{k=1}^{N_g} \frac{X_k\lambda_k}{X_k + 1.065\sum_{j=1,j\neq k}^{N_g} X_j\Phi_{kj}},$$

where

$$\Phi_{kj} = \frac{1}{\sqrt{8}}\left(1+\frac{W_k}{W_j}\right)^{-1/2}\left(1+\left(\frac{\lambda_k}{\lambda_j}\right)^{1/2}\left(\frac{W_j}{W_k}\right)^{1/4}\right)^2.$$

Here, W_k is the molecular weight of kth species, and λ_k is the monoatomic part of heat conductivity of the kth species determined by

$$\ln(\lambda_k) = \sum_{i=1}^{5} c_{i,k}\ln(T)^{n-1}.$$

The binary diffusion coefficients D_{ik} are also estimated based on the logarithm of its value, which is evaluated by the fourth-order polynomial of the logarithm of the temperature

$$\ln D_{ik} = d_{1,ik} + d_{2,ik}\ln T + d_{3,ik}(\ln T)^2 + d_{4,ik}(\ln T)^3 + d_{5,ik}(\ln T)^4 - \ln p,$$

where p is the pressure and $d_{l,ik}$ are the coefficients of the fitted polynomial of the binary diffusion coefficients. The effective diffusion coefficient D_k^{m} of the kth species into a mixture is approximated by the following mixture-averaged diffusion coefficient formula (see e.g., [74], [117] and [76])

$$D_k^{\mathrm{m}} = \frac{1-Y_k}{\sum_{j=1,j\neq k}^{N_g} X_k/D_{jk}},$$

where Y_k and X_k are the mass fraction and the mole fraction of the kth species. The diffusion coefficients computed using the above approximation formula, in general, do not satisfy the condition (1.4). Therefore, the diffusion fluxes must be corrected. There are two approaches for correcting the

deficiencies (see e.g., [32] and [73]). The first one is that we introduce a correction velocity, which in turn are applied to the mass fluxes defined by

$$\begin{aligned} \overrightarrow{J_{\text{cor}}} &= -\sum_{k=1}^{N_{\text{g}}} \overrightarrow{J_k} \\ \overrightarrow{J_{k\ \text{cor}}} &= \overrightarrow{J_k} + Y_k \overrightarrow{J_{\text{cor}}}. \end{aligned}$$

The second one, which can be applied in case there is one species present in large excess (such as a carrier gas, say species named N_{g}), is that we replace the species mass conservation equation of the species (carrier gas) by

$$Y_{N_{\text{g}}} = 1 - \sum_{k=1}^{N_{\text{g}}-1} Y_k.$$

1.5 Steady-state three-dimensional Navier-Stokes equations

The flow can be laminar or turbulent depending on the flow conditions being characterized by the Reynolds number Re_r. For flow with low Reynolds number, the flow are laminar and a steady state can be reached. For flow with high Reynolds number, the flow is turbulent.For example, it is reported in [58] that for water flows in a pipe, the critical Reynolds number is 1.3×10^4 with smooth conditions of entry, flows with higher the critical Reynolds number turn to turbulent. After a certain time interval, the laminar flow comes to a stable state, each of the physical quantities, e.g., velocity, temperature, pressure, etc., at each position in the channel does not change in time.It follows that the partial derivatives of the quantities with respect to time vanish. Thus, the steady-state equations are obtained by deleting the time-derivatives in the above transient three-dimensional Navier-Stokes governing equations. Moreover, we also replace κ by $-2\mu/3$ due to the Stokes hypothesis.

For example, the first term of equation (1.12) vanishes, then the steady state of overall mass continuity equation becomes

$$\frac{\partial \rho u}{\partial z} + \frac{1}{r}\frac{\partial (r\rho v)}{\partial r} = 0 \tag{1.29}$$

Similarly, we obtain the following equations.

Axial momentum:

$$\begin{aligned}\rho u\frac{\partial u}{\partial z}+\rho v\frac{\partial u}{\partial r} = & -\frac{\partial p}{\partial z}+\frac{\partial}{\partial z}\left(2\mu\frac{\partial u}{\partial z}-\frac{2}{3}\mu\nabla\cdot V\right)\\ & +\frac{1}{r}\frac{\partial}{\partial r}\left[\mu r\left(\frac{\partial v}{\partial z}+\frac{\partial u}{\partial r}\right)\right]\end{aligned} \tag{1.30}$$

Radial momentum:

$$\begin{aligned}\rho u\frac{\partial v}{\partial z}+\rho v\frac{\partial u}{\partial r} = & -\frac{\partial p}{\partial r}+\frac{\partial}{\partial z}\left[\rho\left(\frac{\partial v}{\partial z}+\frac{\partial u}{\partial r}\right)\right]\\ & +\frac{\partial}{\partial r}\left(2\mu\frac{\partial v}{\partial r}-\frac{2}{3}\mu\nabla\cdot V\right)+\frac{2\mu}{r}\left(\frac{\partial v}{\partial r}-\frac{v}{r}\right)\end{aligned} \tag{1.31}$$

Species mass continuity:

$$\rho u\frac{\partial Y_k}{\partial z}+\rho v\frac{\partial Y_k}{\partial r}=-\left(\frac{\partial J_{k,z}}{\partial z}+\frac{1}{r}\frac{\partial(rJ_{k,r})}{\partial r}\right)+\dot{\omega}_k W_k \quad (k=1,\ldots,N_{\mathrm{g}}) \tag{1.32}$$

Thermal energy:

$$\begin{aligned}\rho c_p(u\frac{\partial T}{\partial z}+v\frac{\partial T}{\partial r}) = & \left(u\frac{\partial p}{\partial z}+v\frac{\partial p}{\partial r}\right)+\frac{\partial}{\partial z}\left(\lambda\frac{\partial T}{\partial z}\right)+\frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda\frac{\partial T}{\partial r}\right)\\ & -\sum_{k=1}^{K}c_{pk}\left(J_{k,z}\frac{\partial T}{\partial z}+J_{k,r}\frac{\partial T}{\partial r}\right)-\sum_{k=1}^{K}h_k\dot{\omega}_k W_k\end{aligned} \tag{1.33}$$

For a compressible fluid, we need an equation of state, which represents the relationship among density, temperature, pressure, and species composition. The equation of state for ideal gas can be used to describe the relations with accurate enough for gas flow at low pressure [73]. The equation of state is as follows
State:

$$p=\frac{\rho RT}{\overline{W}}.$$

In these equations, the independent variables are the axial and radial spatial coordinates z and r. The dependent variables are: axial velocity u, radial velocity v, $V=(u,v)$, species mass fractions Y_k, temperature T, and pressure p. Other variables are: mass density ρ, viscosity μ, thermal conductivity λ, species enthalpies h_k, and specific heat c_p, species molecular weights W_k, mean molecular weight $\overline{W}$, diffusive mass flux J, more details can be found

in, e.g., [73], [118] and [124]. The two components of mass flux vector J are as follows.

$$J_{k,r} = -\rho\frac{W_k}{\overline{W}}D_k^{\mathrm{m}}\frac{\partial X_k}{\partial r} - \frac{D_k^T}{T}\frac{\partial T}{\partial r},$$
$$J_{k,z} = -\rho\frac{W_k}{\overline{W}}D_k^{\mathrm{m}}\frac{\partial X_k}{\partial z} - D_k^T\frac{1}{T}\frac{\partial T}{\partial z}.$$

The quantities, such as mass density ρ, viscosity μ, thermal conductivity λ, enthalpies of species h_k, enthalpy of mixture h, specific heat of mixture c_p, specific heat of species c_{pk}, depend on species composition and temperature, and the chemical source terms $\dot{\omega}_k$ are functions of species composition, temperature. These relations can be represented abstractly as

$$\mu = \mu(Y,T), \quad \lambda = \lambda(Y,T), \quad c_p = c_p(Y,T), \quad c_{pk} = c_{pk}(Y,T),$$
$$h = h(Y,T), \quad \dot{\omega}_k = \dot{\omega}_k(Y,T,p), \quad V = (u,v), \quad Y = (Y_1, Y_2, \cdots, Y_{N_{\mathrm{g}}}).$$

For the more detailed relations see Section 1.4.

Boundary conditions

For steady state, the boundary conditions (1.10) become

$$(\overrightarrow{J_k} + \rho v\overrightarrow{v}_{\mathrm{stef}}Y_k)\overrightarrow{n} = \dot{s}_k W_k F_{\mathrm{cat/geo}} \quad (k = 1, \ldots, N_{\mathrm{g}}). \tag{1.34}$$

At a steady state, the Stefan velocity $\overrightarrow{v}_{\mathrm{stef}}$ vanishes unless there is mass deposited on the surface, as in case of chemical vapor deposition (CVD). In this thesis, we assume that $\overrightarrow{v}_{\mathrm{stef}} = 0$ which is appropriate for our problems where the deposition of mass on the surface does not occur or can be neglected. Therefore, the conditions (1.34) become

$$\overrightarrow{J_k}\overrightarrow{n} = \dot{s}_k W_k F_{\mathrm{cat/geo}} \quad (k = 1, \ldots, N_{\mathrm{g}}). \tag{1.35}$$

The diffusive and convective fluxes in the gas phase are balanced by thermal radiative and chemically released heat at the surface, which is stated as

$$\left(\lambda\nabla T - \sum_{k=1}^{N_{\mathrm{g}}}(\overrightarrow{J_k} + \rho Y_k\overrightarrow{v}_{\mathrm{stef}})h_k\right)\overrightarrow{n} = \sigma_{\mathrm{sb}}\epsilon_{\mathrm{emis}}(T^4 - T_0^4) + \lambda_s\nabla T\overrightarrow{n} + \sum_{k=N_{\mathrm{g}}+1}^{N_{\mathrm{g}}+N_{\mathrm{s}}}\dot{s}_k W_k h_k, \tag{1.36}$$

where λ and λ_s are the thermal conductivity of the gas mixture and of the solid wall, respectively. σ_{sb} is the Stefan-Boltzmann constant, ϵ_{emis} is the emissivity of the surface, and T_0 is the ambient temperature.

1.6 Boundary layer equations

The Navier-Stokes equations in Section 1.5, which are a large system of *elliptic partial differential equations*, require a huge amount of computing time to a numerical solution. In this section, we derive a simplified version of the Navier-Stokes equations, called *boundary layer equations*, which is a system of *parabolic partial differential equations*. While it still delivers an excellent approximation of the flow field of problems under study, it needs significantly shorter time to solve.[101]. Therefore, we use the boundary layer equations as our main mathematical model for the subsequent chapters.

The boundary layer theory was first introduced by L. Prandtl [99] at the International Congress of Mathematicians, Heidelberg, 1904. Basing on results of experiments of fluid flows along a fixed solid surface, he saw that there is a small region near the surface, in which the effect of viscosity is important, even for a fluid of small viscosity. Because in this region, the velocity is rising rapidly from zero at the wall to its value in the main stream. The flow can be divided in two regions: the first region is near the solid surface and the other region, so-called outer flow is next to the first region where the effect of viscosity is neglected. In the region near the wall, the component normal to the wall of the velocity is small compared to the component in the direction of the flow along the wall, and the influence of the viscosity normal to the wall is dominant, this thin region is called *boundary layer* and the *boundary layer approximation* is used. The outer flow can be considered as an inviscid flow. At the outer edge of the boundary layer the two flows are properly matched. The result is that the complicated Navier-Stokes equations are reduced to a simpler system of equations. Generally speaking, the boundary layer equations can be applicable for case, where there is a principal flow direction, and in such direction the convective transport often dominates over diffusive transport. Under such conditions and some others, some terms in the Navier-Stokes equations are small compared to others, thus they are neglected. For more details and references on boundary layer theory, see [58], [104], [92], [90], and [120].

In the following, we derive the boundary layer equations in three major steps. At first, the Navier-Stokes equations are re-written in a dimensionless form. Secondly, the dimensionless form of the Navier-Stokes equations is simplified by using certain assumptions and conditions of the flow. Finally, the simplified version of the Navier-Stokes equations is brought back to its dimensional form.

We introduce the following reference scales for the independent and dependent variables and write the Navier-Stokes equations in its nondimensional form. The reference scales for the axial coordinate and radial coordi-

nate are z_s and r_s, respectively. The inlet velocity u_0 is used as the reference scale for the axial velocity, and v_s is used as the reference scale for the radial velocity. We use ρ_0 and μ_0 at inlet as the reference scale for density and viscosity, respectively. Using these reference scales, we bring all independent and dependent variables to order-one variables. The new nondimensional variables can be written as

$$\widehat{z} = \frac{z}{z_s},\ \widehat{r} = \frac{r}{r_s},\ \widehat{u} = \frac{u}{u_0},\ \widehat{v} = \frac{v}{v_s},\ \widehat{\rho} = \frac{\rho}{\rho_0},\ \widehat{\mu} = \frac{\mu}{\mu_0},\ \widehat{p} = \frac{p}{\rho_0 u_0^2}\ .$$

Using these new nondimensional variables, the mass continuity equation (1.29) becomes

$$\left(\frac{\rho_0 u_0}{z_s}\right)\frac{\partial \widehat{\rho}\widehat{u}}{\partial \widehat{z}} + \left(\frac{\rho_0 v_s}{r_s}\right)\frac{1}{\widehat{r}}\frac{\partial \widehat{r}\widehat{\rho}\widehat{v}}{\partial \widehat{r}} = 0.$$

If we choose the reference scale value for the radial velocity satisfying

$$v_s = \frac{r_s u_0}{z_s},$$

then the mass continuity equation in the nondimensional form is as follows.

$$\frac{\partial \widehat{\rho}\widehat{u}}{\partial \widehat{z}} + \frac{1}{\widehat{r}}\frac{\partial \widehat{r}\widehat{\rho}\widehat{v}}{\partial \widehat{r}} = 0. \tag{1.37}$$

Similarly, the axial-momentum equation can be written in nondimensional form as

$$\begin{aligned}
&\left(\frac{\rho_0 u_0^2}{z_s}\right)\widehat{\rho}\widehat{u}\frac{\partial \widehat{u}}{\partial \widehat{z}} + \left(\frac{\rho_0 u_0 (r_s u_0/z_s)}{r_s}\right)\widehat{\rho}\widehat{v}\frac{\partial \widehat{u}}{\partial \widehat{r}} \\
= &-\left(\frac{\rho_0 u_0^2}{z_s}\right)\frac{\partial \widehat{p}}{\partial \widehat{z}} + \left(\frac{\mu_0 u_0}{z_s^2}\right)\frac{\partial}{\partial \widehat{z}}\left[\frac{4}{3}\widehat{\mu}\frac{\partial \widehat{u}}{\partial \widehat{z}} - \frac{2}{3}\widehat{\mu}\frac{1}{\widehat{r}}\frac{\partial \widehat{r}\widehat{v}}{\partial \widehat{r}}\right] \\
&+ \left(\frac{\mu_0 (r_s u_0/z_s)}{r_s z_s}\right)\frac{1}{\widehat{r}}\frac{\partial}{\partial \widehat{r}}\left(\widehat{\mu}\widehat{r}\frac{\partial \widehat{v}}{\partial \widehat{z}}\right) \qquad (1.38) \\
&+ \left(\frac{\partial \mu_0 u_0}{r_s^2}\right)\frac{1}{\widehat{r}}\frac{\partial}{\partial \widehat{r}}\left(\widehat{\mu}\widehat{r}\frac{\partial \widehat{u}}{\partial \widehat{r}}\right). \qquad (1.39)
\end{aligned}$$

Let us define the Reynolds number by its conventional form:

$$\mathrm{Re}_r = \frac{\rho_0 u_0 r_s}{\mu_0}.$$

When both side of Equation (1.38) are multiplied by $z_s/\rho_0 u_0^2$, the axial-momentum equation becomes

$$\begin{aligned}
\widehat{\rho}\widehat{u}\frac{\partial \widehat{u}}{\partial \widehat{z}} + \widehat{\rho}\widehat{v}\frac{\partial \widehat{u}}{\partial \widehat{r}} &= -\frac{\partial \widehat{p}}{\partial \widehat{z}} + \left(\frac{z_s}{r_s}\frac{1}{\mathrm{Re}_r}\right)\frac{1}{\widehat{r}}\frac{\partial}{\partial \widehat{r}}\left(\widehat{\mu}\widehat{r}\frac{\partial \widehat{u}}{\partial \widehat{r}}\right) \\
&+ \left(\frac{r_s}{z_s}\frac{1}{\mathrm{Re}_r}\right)\left\{\frac{\partial}{\partial \widehat{z}}\left[\frac{4}{3}\widehat{\mu}\frac{\partial \widehat{u}}{\partial \widehat{z}} - \frac{2}{3}\mu\frac{1}{\widehat{r}}\frac{\partial \widehat{r}\widehat{v}}{\partial \widehat{v}}\right] + \frac{1}{\widehat{r}}\frac{\partial}{\partial \widehat{r}}\left(\widehat{\mu}\widehat{r}\frac{\partial \widehat{v}}{\partial \widehat{z}}\right)\right\}. \qquad (1.40)
\end{aligned}$$

For channels which are narrow compared to their length $r_\mathrm{s} \ll z_\mathrm{s}$ and $\mathrm{Re}_r > 1$, we would have the following relation

$$\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r} \gg \frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r}.$$

Consider three limiting cases for the leading coefficient of the radial-diffusion term

$$\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r} \sim 0, \quad \frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r} \sim 1, \quad \frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r} \sim \infty.$$

For the first case, from Equation (1.40), we have a inviscid flow, in which viscous effects are neglected. Thus, the no-slip condition at the wall is impossible to be satisfied. For the third case, we do not have convective effects. For the second case, if we choose the channel radius r_0 as the characteristic radial length scale, $r_\mathrm{s} = r_0$, then $z_\mathrm{s} \sim r_0\mathrm{Re}_r$. We can neglect the last term in Equation (1.40), and obtain

$$\widehat{\rho}\widehat{u}\frac{\partial \widehat{u}}{\partial \widehat{z}} + \widehat{\rho}\widehat{v}\frac{\partial \widehat{u}}{\partial \widehat{r}} = -\frac{\partial \widehat{p}}{\partial \widehat{z}} + \left(\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r}\right)\frac{1}{\widehat{r}}\frac{\partial}{\partial \widehat{r}}\left(\widehat{\mu}\widehat{r}\frac{\partial \widehat{u}}{\partial \widehat{r}}\right). \tag{1.41}$$

Similarly, the radial momentum equation (1.31) can be made dimensionless as

$$\begin{aligned}
&\left(\frac{\rho_0 u_0 (r_\mathrm{s}u_0/z_\mathrm{s})}{z_\mathrm{s}}\right)\widehat{r}\widehat{u}\frac{\partial \widehat{v}}{\partial \widehat{z}} + \left(\frac{\rho_0 u_0 (r_\mathrm{s}u_0/z_\mathrm{s})^2 r_\mathrm{s}}{r_\mathrm{s}}\right)\widehat{\rho}\widehat{v}\frac{\partial \widehat{v}}{\partial \widehat{r}} \\
= \;& -\left(\frac{\rho_0 u_0^2}{r_\mathrm{s}}\right)\frac{\partial \widehat{p}}{\partial \widehat{r}} + \left(\frac{\mu_0 (r_\mathrm{s}u_0/z_\mathrm{s})}{z_\mathrm{s}^2}\right)\frac{\partial}{\partial \widehat{z}}\left(\widehat{\mu}\frac{\partial \widehat{v}}{\partial \widehat{z}}\right) \\
&+ \left(\frac{\mu_0 u_0}{z_\mathrm{s} r_\mathrm{s}}\right)\frac{\partial}{\partial \widehat{z}}\left(\widehat{\mu}\frac{\partial \widehat{u}}{\partial \widehat{r}}\right) + \left(\frac{\mu_0 (r_\mathrm{s}u_0/z_\mathrm{s})}{r_\mathrm{s}^2}\right)\frac{\partial}{\partial \widehat{r}}\left(\frac{4}{3}\widehat{\mu}\frac{\partial \widehat{v}}{\partial \widehat{r}} - \frac{2}{3}\widehat{\mu}\frac{\widehat{v}}{\widehat{r}}\right) \\
&- \left(\frac{\mu_0 u_0}{r_\mathrm{s} z_\mathrm{s}}\right)\frac{\partial}{\partial \widehat{r}}\left(\frac{2}{3}\widehat{\mu}\frac{\partial \widehat{u}}{\partial \widehat{z}}\right) + \left(\frac{\mu_0 (r_\mathrm{s}u_0/z_\mathrm{s})}{r_\mathrm{s}^2}\right)\frac{2\widehat{\mu}}{\widehat{r}}\left[\frac{\partial \widehat{v}}{\partial \widehat{v}} - \frac{\widehat{v}}{\widehat{r}}\right].
\end{aligned}$$

After simplification, we obtain

$$\begin{aligned}
&\left(\frac{r_\mathrm{s}^2}{z_\mathrm{s}^2}\right)\widehat{\rho}\widehat{\mu}\frac{\partial \widehat{v}}{\partial \widehat{z}} + \left(\frac{r_\mathrm{s}^2}{z_\mathrm{s}^2}\right)\widehat{\rho}\widehat{v}\frac{\partial \widehat{v}}{\partial \widehat{v}} = -\frac{\partial \widehat{p}}{\partial \widehat{r}} + \left(\frac{1}{\mathrm{Re}_r}\frac{r_\mathrm{s}^3}{z_\mathrm{s}^3}\right)\frac{\partial}{\partial \widehat{z}}\left(\widehat{\mu}\frac{\partial \widehat{v}}{\partial \widehat{z}}\right) \\
&+ \left(\frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r}\right)\left\{\frac{\partial}{\partial \widehat{z}}\left(\widehat{\mu}\frac{\partial \widehat{u}}{\partial \widehat{r}}\right) + \frac{\partial}{\partial \widehat{r}}\left[\frac{4}{3}\widehat{\mu}\frac{\partial \widehat{v}}{\partial \widehat{r}} - \frac{2}{3}\widehat{\mu}\left(\frac{\partial \widehat{u}}{\partial \widehat{z}} + \frac{\widehat{v}}{\widehat{r}}\right)\right] + \frac{2\widehat{\mu}}{\widehat{r}}\left[\frac{\partial \widehat{v}}{\partial \widehat{r}} - \frac{\widehat{v}}{\widehat{r}}\right]\right\}.
\end{aligned} \tag{1.42}$$

For channels that have $r_\mathrm{s} \ll z_\mathrm{s}$ and $\mathrm{Re}_r > 1$, in equation (1.42), only order-one term is the pressure gradient. Therefore, the equation (1.42) is reduced to

$$\frac{\partial p}{\partial r} = 0. \tag{1.43}$$

By introducing reference scale for diffusion coefficient $D_{k,0}$ as

$$J_{k,z} = -\frac{\rho_0 D_{k,0}}{z_\mathrm{s}} \widehat{\rho} \frac{W_k}{\overline{\overline{W}}} \widehat{D}_k^m \frac{\partial X_k}{\partial \widehat{z}}, \quad J_{k,r} = -\frac{\rho_0 D_{k,0}}{z_\mathrm{s}} \widehat{\rho} \frac{W_k}{\overline{\overline{W}}} \widehat{D}_k^m \frac{\partial X_k}{\partial \widehat{r}},$$

i.e.,

$$J_{k,z} = \frac{\rho D_{k,0}}{z_\mathrm{s}} \widehat{J}_{k,z}, \quad J_{k,r} = \frac{\rho D_{k,0}}{r_\mathrm{s}} \widehat{J}_{k,r},$$

with

$$\widehat{J}_{k,z} = \widehat{\rho} \frac{W_k}{\overline{\overline{W}}} \widehat{D}_k^m \frac{\partial X_k}{\partial \widehat{z}}, \quad \widehat{J}_{k,r} = \widehat{\rho} \frac{W_k}{\overline{\overline{W}}} \widehat{D}_k^m \frac{\partial X_k}{\partial \widehat{r}},$$

and changing the independent variables and the unknown functions as above, we obtain the energy and species equations in a nondimensional form as follows

$$\begin{aligned} \widehat{\rho}\widehat{u}\frac{\partial Y_k}{\partial \widehat{z}} + \widehat{\rho}\widehat{v}\frac{\partial Y_k}{\partial \widehat{r}} = &-\left(\frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k}\right)\frac{\partial \widehat{J}_{k,z}}{\partial \widehat{z}} \\ &- \left(\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k}\right)\frac{1}{\widehat{r}}\frac{\partial \widehat{r}\widehat{J}_{k,r}}{\partial \widehat{r}} \\ &+ \frac{z_\mathrm{s}}{\rho_0 u_0}\dot{\omega}_k W_k, \end{aligned} \tag{1.44}$$

$$\begin{aligned} \widehat{\rho}\widehat{c}_p\widehat{u}\frac{\partial \widehat{T}}{\partial \widehat{z}} + \widehat{\rho}\widehat{c}_p\widehat{v}\frac{\partial \widehat{T}}{\partial \widehat{r}} = &\quad \frac{u_0^2}{c_{p,0}\Delta T}\widehat{u}\frac{\partial \widehat{p}}{\partial \widehat{z}} + \left(\frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Pr}}\right)\frac{\partial}{\partial \widehat{z}}\left(\widehat{\lambda}\frac{\partial \widehat{T}}{\partial \widehat{z}}\right) \\ &+ \left(\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Pr}}\right)\frac{1}{\widehat{r}}\frac{\partial}{\partial \widehat{r}}\left(\widehat{r}\widehat{\lambda}\frac{\partial \widehat{T}}{\partial \widehat{r}}\right) \\ &- \sum_{k=1}^{N_\mathrm{g}}\left[\left(\frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k}\right)\widehat{c}_{pk}\widehat{J}_{k,z}\frac{\partial \widehat{T}}{\partial z} + \left(\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k}\right)\widehat{c}_{pk}\widehat{J}_{k,r}\frac{\partial T}{\partial \widehat{r}}\right] \\ &- \frac{\Delta T z_\mathrm{s}}{\rho_0 c_{p,0} u_0}\sum_{k=1}^{N_\mathrm{g}} h_k \dot{\omega}_k W_k, \end{aligned} \tag{1.45}$$

where the Prandtl and Schmidt numbers are defined as $\mathrm{Pr} = \mu c_p/\lambda$, $\mathrm{Sc}_k = \mu/(\rho D_k)$.

Considering the case

$$\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Pr}} \approx 1, \quad \frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k} \approx 1,$$

then by multiplying both sides with $r_\mathrm{s}^2/z_\mathrm{s}^2$ we have

$$\frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Pr}} \approx \frac{r_\mathrm{s}^2}{z_\mathrm{s}^2}, \quad \frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k} \approx \frac{r_\mathrm{s}^2}{z_\mathrm{s}^2}. \tag{1.46}$$

From equations (1.44), (1.45) and (1.46), we see that the axial diffusion terms are too small compared with other terms as $r_\mathrm{s}/z_\mathrm{s} \to 0$. Therefore, the energy and species equations are reduced to

$$\widehat{\rho}\widehat{u}\frac{\partial Y_k}{\partial \widehat{z}} + \widehat{\rho}\widehat{v}\frac{\partial Y_k}{\partial \widehat{r}} = \left(\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k}\right)\frac{1}{\widehat{r}}\frac{\partial \widehat{r}\widehat{J}_{k,r}}{\partial \widehat{r}} + \frac{z_\mathrm{s}}{\rho_0 u_0}\dot{\omega}_k W_k, \tag{1.47}$$

$$\begin{aligned}\widehat{\rho}\widehat{c_p}\widehat{u}\frac{\partial \widehat{T}}{\partial \widehat{z}} + \widehat{\rho}\widehat{c_p}\widehat{v}\frac{\partial \widehat{T}}{\partial \widehat{r}} = \quad & \frac{u_0^2}{c_{p0}\Delta T}\widehat{u}\frac{\partial \widehat{p}}{\partial \widehat{z}} + \left(\frac{r_\mathrm{s}}{z_\mathrm{s}}\frac{1}{\mathrm{Re}_r \,\mathrm{Pr}}\right)\frac{\partial}{\partial \widehat{z}}\left(\widehat{\lambda}\frac{\partial \widehat{T}}{\partial \widehat{z}}\right) \\ & - \sum_{k=1}^{N_\mathrm{g}}\left(\frac{z_\mathrm{s}}{r_\mathrm{s}}\frac{1}{\mathrm{Re}_r \mathrm{Sc}_k}\right)\widehat{c}_{pk}\widehat{J}_{k,r}\frac{\partial T}{\partial \widehat{r}} \\ & - \frac{\Delta T z_\mathrm{s}}{\rho_0 c_{p0} u_0}\sum_{k=1}^{N_\mathrm{g}} h_k \dot{\omega}_k W_k. \end{aligned} \tag{1.48}$$

Applying the inverse transformation back to the original coordinates to the reduced set of equations in the nondimensional form (1.37), (1.41), (1.43), (1.47), and (1.48), we obtain the following set of equations.

Boundary layer equations

Mass continuity:

$$\frac{\partial \rho u}{\partial z} + \frac{1}{r}\frac{\partial (r\rho v)}{\partial r} = 0. \tag{1.49}$$

Axial momentum:

$$\rho u\frac{\partial u}{\partial z} + \rho v\frac{\partial u}{\partial r} = -\frac{\partial p}{\partial z} + \frac{1}{r}\frac{\partial}{\partial r}\left(\mu r\frac{\partial u}{\partial r}\right). \tag{1.50}$$

Radial momentum:

$$0 = \frac{\partial p}{\partial r}. \tag{1.51}$$

Species continuity:

$$\rho u\frac{\partial Y_k}{\partial z} + \rho v\frac{\partial Y_k}{\partial r} = -\frac{1}{r}\frac{\partial (r J_{k,r})}{\partial r} + \dot{\omega}_k W_k \quad (k = 1, \ldots, N_\mathrm{g}). \tag{1.52}$$

Thermal energy:

$$\rho c_p\left(u\frac{\partial T}{\partial z} + v\frac{\partial T}{\partial r}\right) = \frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda\frac{\partial T}{\partial r}\right) - \sum_{k=1}^{N_\mathrm{g}} c_{pk} J_{k,r}\frac{\partial T}{\partial r} - \sum_{k=1}^{N_\mathrm{g}} h_k \dot{\omega}_k W_k, \tag{1.53}$$

State equation:

$$p = \frac{\rho RT}{\overline{W}}, \tag{1.54}$$

where

$$\begin{aligned}
&J_{k,r} = -D_k^m \frac{W_k}{\overline{W}} \rho \frac{\partial X_k}{\partial r} - \frac{D_k^T}{T} \frac{\partial T}{\partial r}, \\
&\mu = \mu(Y,T), \quad \lambda = \lambda(Y,T), \quad c_p = c_p(Y,T), \\
&c_{pk} = c_{pk}(Y,T), \quad h = h(Y,T), \quad \dot{\omega}_k = \dot{\omega}_k(Y,T,p), \quad Y = (Y_1, Y_2, \cdots, Y_{N_g}).
\end{aligned}$$

These relations are discussed in Sections 1.3 and 1.4, more details can be seen in, e.g., [73] and [101].

Remark 1.6.1
Although the steady-state boundary layer equations do have a full two-dimensional representation of all the field variables as well as nonlinear behavior of Navier-Stokes equations, it is a system of ***parabolic*** *partial differential equations instead of* ***elliptic*** *ones as the Navier-Stokes equations. This is a huge simplification for numerical treatment.*

1.7 Boundary conditions

In this section, we only discuss the boundary conditions needed for the boundary layer equations.

1.7.1 Conditions at the inlet

At the inlet, the entrance of the channel, the initial profiles of u, T_{gas}, Y_k, p, surface site fraction Θ_k, T_{wall}, which are usually referred to as *initial conditions*, must be specified.

1.7.2 Conditions at the catalytic wall and at the centerline

At the centerline of the cylinder, the cylinder symmetry is used to determine the boundary conditions

$$\left.\frac{\partial u}{\partial r}\right|_{r=0} = 0, \quad \left.\frac{\partial T}{\partial r}\right|_{r=0} = 0, \quad \left.\frac{\partial p}{\partial r}\right|_{r=0} = 0, \quad \left.\frac{\partial Y_k}{\partial r}\right|_{r=0} = 0.$$

At the wall, the *no-slip boundary condition* is assumed, i.e., the axial and radial velocity vanish

$$u = 0, \quad v = 0.$$

The condition for the pressure is

$$\frac{\partial p}{\partial r} = 0,$$

which is the same as the simplified radial momentum equation (1.51).

The boundary condition for mass species Y_k is more complicated. If the wall is not a catalytic surface, the condition is

$$\frac{\partial Y_k}{\partial r} = 0 \ .$$

If the wall is a catalytic surface, the boundary conditions at the catalytic wall require that the gas-phase species mass flux produced by heterogeneous chemical-reaction must be balanced by the diffusive and convective flux of that species in the gas, see, e.g., [73] and [101]

$$\dot{s}_k W_k = -(J_{k,r} + \rho Y_k v_{\text{stef}}) \quad (k = 1, \ldots, N_{\text{g}}), \tag{1.55}$$

where $\dot{s}_k$ is the rate of creation/depletion of the kth gas phase species by surface reactions. The dependent variables in this expression are temperature, pressure, mass fractions and surface coverages at the wall, which do not appear explicitly in this expression, but they apprear implicitly in the term $\dot{s}_k$ and $J_{k,r}$, see Section 1.3.2 for more details.

At steady state, the time variation of the surface coverage Θ_k (see Equation 1.26) vanishes:

$$\dot{s}_k = 0 \quad (k = N_{\text{g}} + 1, \ldots, N_{\text{g}} + N_{\text{s}}). \tag{1.56}$$

The boundary conditions (1.55) and (1.56) are highly nonlinear and used for determining the mass fractions Y_k and the surface coverages Θ_i at the wall. This is unusual case where the values of the variables at the boundary are not given explicitly. These boundary conditions are sometimes called *implicit boundary conditions*.

The condition for temperature depends on adiabatic or isothermal reactor conditions. For isothermal reactor, we require that the temperature profile at the wall is specified:

$$T(z) = T_{\text{wall}}(z).$$

For adiabatic case, the temperature boundary condition (1.36) becomes

$$-\lambda \frac{\partial T}{\partial r} + \sum_{k=1}^{N_{\text{g}}} (J_{k,r} + \rho Y_k v_{\text{stef}}) h_k = -\lambda_s \frac{\partial T}{\partial r} + \sum_{k=N_{\text{g}}+1}^{N_{\text{g}}+N_{\text{s}}} \dot{s}_k W_k h_k. \tag{1.57}$$

Here we neglect the thermal radiation term. It follows from (1.55), (1.56) and (1.57) that

$$-\lambda\frac{\partial T}{\partial r} = -\lambda_s\frac{\partial T}{\partial r} + \sum_{k=1}^{N_\mathrm{g}} \dot{s}_k W_k h_k. \tag{1.58}$$

This is also a highly nonlinear equation, because T appears in the exponent of the rate coefficients (see Section 1.3.2).

1.8 Summary

In this chapter, we examine different models for chemically reacting flows in a channel of catalytic monoliths. These include time-dependent Navier-Stokes equations, steady-state Navier-Stokes equations, and steady-state boundary layer equations. In general, the boundary layer equations can be applicable for the case, where there is a principal flow direction, and in such a direction the convective transport often dominates over diffusive transport. Under such conditions and some others, some terms in the Navier-Stokes equations are small compared to others, thus they are neglected.

In Section 1.6 the boundary layer equations are obtained by simplifying the steady-state Navier-Stokes equations under the assumptions

$$\mathrm{Re}_r \approx \frac{z_\mathrm{s}}{r_\mathrm{s}}, \quad \text{and} \quad r_\mathrm{s} \ll z_\mathrm{s},$$

and

$$\mathrm{Sc}_k \approx 1, \quad \mathrm{Pr} \approx 1,$$

where the Reynolds number Re_r, the Schmidt numbers Sc_k and the Prandtl number Pr are defined as

$$\mathrm{Re}_r = \frac{\rho_0 u_0 r_\mathrm{s}}{\mu_0}, \quad \mathrm{Sc}_k = \frac{\mu}{\rho D_k}, \quad \mathrm{Pr} = \frac{\mu c_p}{\lambda}.$$

In words, when the length of the channel is large compared to the channel radius, and the Reynolds number is at the same order of magnitude of the ratio between the length and the radius, and the Schmidt numbers and the Prandtl number are of order one, then the steady Navier-Stokes equations, which is a system of *elliptic* partial differential equations, can be simplified to obtain the boundary layer equations, which is a system of *parabolic* partial differential equations. The equation systems are stiff and nonlinear. The surface and gas-phase chemical reactions are described by detailed chemistry models.

Chapter 2

Numerical Methods for Differential-Algebraic Equations

The governing model equations mentioned in Chapter 1, which is a systen of parabolic partial differential equations, is semi-discretized in the spatial direction ψ, where the semi-discretization is described in Chapter 3, leading to a large-scale stiff structured system of *differential-algebraic equations* (DAEs). The sources of stiffness are arising from the discretization of PDEs and due to the modeling of chemical processes, in particular using detailed models.

To solve the stiff DAEs we use an implicit method, based on *backward differentiation formulas* (BDF), which has proved to be the best method for stiff DAEs. For details on theory and numerical methods for DAEs, see e.g., [67], [24], [5], and [100].

In this chapter, the solution techniques for BDF methods are discussed. In Section 2.1 we introduce some basic terms and properties used for DAEs. Section 2.2, is a brief summary and some definitions for the multistep methods, of which the BDF methods are members. Section 2.3 is devoted to the BDF methods. Applying the BDF methods to discretize a DAE system leads to a system of nonlinear equations. The numerical methods for the nonlinear equations are discussed in Section 2.4. Section 2.6 concentrates on error analysis for the solution process in Section 2.4. An automatic scaling of the linear algebraic equations arising from the solution process of the nonlinear equations is introduced in Section 2.7. Section 2.8 describes automatic differentiation, a method for computation of derivatives needed for the solution of nonlinear equations. In Section 2.10, specially tailored methods, in particular exploiting the structure of the system with efficient methods for

computation of derivatives which are needed for the solution of the nonlinear equations in Section 2.4, are presented.

2.1 Basic definitions and properties

Generally, a DAE system can be written as

$$f(t, x, \dot{x}) = 0. \tag{2.1}$$

To describe the property of the DAEs, we use the following definition of index defined in [24].

Definition 2.1.1 (Differential index)
*The minimum number of times that all or part of the DAE system (2.1) must be differentiated with respect to t in order to determine $\dot{x}$ as a continuous function of (t, x), is the **index** of the DAE. The index defined here is also referred to as the **global index**.*

The following definitions and results are partially based on [24], [86], [112], and [95].

Definition 2.1.2 (Consistent initial values)
*A set of initial values of x and $\dot{x}$ is said to be **consistent** if it satisfies the original system (2.1) and all systems obtained by differentiating (2.1) with respect to t.*

Definition 2.1.3 (Structural property of matrix)
*A square matrix $A \in \mathbb{R}^{n\times n}$ is called **structurally singular** if every matrix $B \in \mathbb{R}^{n\times n}$ with $B_{i,j} = 0$ if $A_{i,j} = 0$ is singular, or **structurally nonsingular** otherwise.*

Definition 2.1.4 (Structural matrix)
The structural matrix of a matrix $A \in \mathbb{R}^{m\times n}$, denoted by S^A, $S^A \in \mathbb{R}^{m\times n}$, is defined as

$$S^A_{i,j} = \begin{cases} 1 & \text{if } A_{i,j} \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 2.1.1
A square matrix A is structurally nonsingular if and only if there exists at least one permutation matrix P such that all diagonal elements of PA are nonzero.

Definition 2.1.5 (Structural matrix of DAE)
The structural matrix of a DAE system is the structural matrix of the Jacobian matrix of the DAE with respect to the highest-order time derivatives in the DAE.

In particular for the semi-implicit DAE system

$$\begin{aligned} f(t, x, \dot{x}, y) &= 0 \\ g(t, x, y) &= 0 \end{aligned} \tag{2.2}$$

*where x and y are called **differential variable** and **algebraic variable**, respectively, the structural matrix of the DAE (2.2) is the structural matrix of the matrix*

$$\begin{bmatrix} \dfrac{\partial f}{\partial \dot{x}} & \dfrac{\partial f}{\partial y} \\ 0 & \dfrac{\partial g}{\partial y} \end{bmatrix}.$$

Definition 2.1.6 (Structurally singular DAE)
*A DAE system is said to be **structurally singular** if its structural matrix is structurally singular, or **structurally nonsingular** otherwise.*

Note that the structural properties of a square matrix A are equivalent to the same structural properties of its structural matrix S^A. Thus, to study structural properties of a matrix A, one can use its structural matrix S^A instead.

Example 2.1.1 (Structurally singular DAE)
The following DAE is structurally singular

$$\begin{array}{rcrcrcl} \dot{x}_1 & + & \dot{x}_2 & + & y & = & a(t) \\ x_1 & + & 2x_2 & + & 3y & = & b(t) \\ 3x_1 & + & 4x_2 & + & 5y & = & c(t) \end{array}$$

because the structural matrix

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

is structurally singular.

Lemma 2.1.2 (Sufficient condition for at most index 1)
Sufficient condition for a DAE to be at most index 1 is non-singularity of the Jacobian matrix of the DAE with respect to the highest-order time derivatives in the DAE.

In particular for the semi-implicit DAE system (2.2) if the matrix

$$\begin{bmatrix} \dfrac{\partial f}{\partial \dot{x}} & \dfrac{\partial f}{\partial y} \\ 0 & \dfrac{\partial f}{\partial y} \end{bmatrix}$$

is nonsingular, then the index of the DAE is at most 1.

Note that a structurally singular DAE may have index 1 but it does not satisfy the sufficient condition, such as the DAE in Example 2.1.1.

A semi-explicit DAE

$$\begin{aligned} \dot{x} &= f(t, x, y) \\ 0 &= g(t, x, y) \end{aligned} \tag{2.3}$$

is index one if and only if $\partial g/\partial y$ is nonsingular.

In the following, the functions f in the fully-implicit DAE (2.1) or f and g in the semi-implicit DAE (2.2) or (2.3) or B, f and g in the semi-implicit quasilinear DAE (2.11) are called *model functions*.

2.2 Linear multistep methods

A general k-step multistep method with constant step h is given by

$$\sum_{j=0}^{k} \alpha_i x_{n-j} = h \sum_{j=0}^{k} \beta_j \dot{x}_{n-j}, \tag{2.4}$$

where α_i and β_j are the method's coefficients, $\alpha_0 \neq 0$ and $|\alpha_k| + |\beta_k| \neq 0$. The linear multistep method is *explicit* if $\beta_0 = 0$ and *implicit* otherwise.

2.2.1 Error, order and convergence

The *local truncation error* of the linear multistep method (2.4) at t_n is defined as the defect obtained when plugging the exact solution $x(t)$ into the formula (2.4), which is written as

$$\tau_n = \frac{1}{h} \sum_{j=0}^{k} (\alpha_i x(t_n - jh) - h\beta_j \dot{x}(t_n - jh)). \tag{2.5}$$

It measures how closely the difference operator approximates the differential operator. This definition of the local truncation error is the same as one

defined in [5], p. 132. This should not be confused with the *truncation error* as defined in [72], which is actually defined in [27] as the *local discretization error* or *local error* for short.

The *local discretization errror* (see [27], [5], p. 43) (or *local error*) at t_n is defined as the difference between the exact solution $x^l(t_n)$ of the differential equation on the interval $[t_{n-1}, t_n]$ at t_n with the initial value $x^l(t_{n-1}) = x_{n-1}$, and the solution x_n of the difference equation also at t_n,

$$\begin{aligned} l_n &= x^l(t_n) - x_n \\ &= x^l(t_n) - \frac{1}{\alpha_0}\left(-\sum_{j=1}^{k}\alpha_i x^l_{n-j} + h\sum_{j=0}^{k}\beta_j \dot{x}^l_{n-j}\right), \end{aligned} \tag{2.6}$$

where x^l_{n-j} and $\dot{x}^l_{n-j}$ are supposed to be the exact values of $x^l(t_{n-j})$ and $\dot{x}^l(t_{n-j})$ for the true solution x^l of the differential equation with $x^l(t_{n-1}) = x_{n-1}$. $x^l(t)$ should not be confused with $x(t)$, which is the true solution of the differential equation with $x(t_0) = x_0$. This definition of local discretization error is somewhat similar to the definition of *local truncation error* defined in [24]. However, in the this thesis we use the definitions defined above, i.e., the local discretization errror refers to l_n and the local truncation error refers to τ_n.

It is shown that (see e.g., [5] and [24]) under certain assumptions we have the following relation:

$$h_n\|\tau_n\| = \|l_n\|(1 + O(h_n)).$$

Definition 2.2.1 (Order of method)
The linear multistep method (2.4) is said to be ***consistent*** *(or* ***accurate****)* ***of order*** *p if for any C^∞ function $x(t) : \mathcal{T} \to \mathbb{R}^n$ and any $t \in \mathbb{R}$ the condition*

$$\tau(t,h) = \frac{1}{h}\sum_{j=0}^{k}(\alpha_i x(t-jh) - h\beta_j \dot{x}(t-jh)) = O(h^p),\ \text{as } h \to 0, \tag{2.7}$$

holds. A linear multistep method is ***consistent*** *if it is consistent of order $p \geq 1$.*

The necessary and sufficient conditions for consistency of a linear multistep method are stated in the following theorem [5].

Theorem 2.2.1
The linear multistep method (2.4) is consistent iff

$$\sum_{j=0}^{k}\alpha_j = 0, \quad \sum_{j=1}^{k} j\alpha_j + \sum_{j=0}^{k}\beta_j = 0.$$

To measure the total error (true error) of the approximate solution, the *global error* is defined as the difference between the exact solution $x(t)$ and the approximate solution x_n

$$e_n = x(t_n) - x_n, \text{ with } x(t_0) = x_0.$$

Definition 2.2.2 (Order of convergence of method)
The linear multistep method (2.4) is said to be ***convergent of order*** *p if*

$$e_n = O(h^p).$$

The convergence of a method ensures that the approximate solution approaches the true solution (i.e., the global error approaches zero) when the stepsize approaches zero.

2.2.2 Stability and stiffness

Definition 2.2.3 (Stability of problem)
An initial value problem DAE (2.1) with the initial value $x(0)$, or its exact solution $x(t)$ for $(t \geq 0)$, is said to be

- ***stable*** *if given any $\epsilon > 0$ there is a $\delta > 0$ such that any other solution $\widehat{x}(t)$ satisfying the DAE and*

  $$\|x(0) - \widehat{x}(0)\| \leq \delta$$

  also satisfies

  $$\|x(t) - \widehat{x}(t)\| \leq \epsilon \text{ for all } t \geq 0;$$

- ***asymptotically stable*** *if, in addition to be stable,*

  $$\|x(t) - \widehat{x}(t)\| \to 0 \text{ as } t \to \infty.$$

- ***unstable*** *otherwise.*

In particular for linear constant coefficient ODE

$$\dot{x} = Ax, \tag{2.8}$$

the solution of (2.8) is: (a) *stable* iff all eigenvalues λ of A satisfy either $\Re(\lambda) < 0$ or $\Re(\lambda) = 0$ and λ is *simple*, (b) *asymptotically stable* iff all eigenvalues λ of A satisfy $\Re(\lambda) < 0$.

Similarly, we define the stability of linear difference equations with constant coefficients

$$a_k x_{n-k} + a_{k-1} x_{n-k+1} + \cdots + a_0 x_0 = q_n. \tag{2.9}$$

The *characteristic polynomial* of the difference equation (2.9) is defined as

$$\phi(\xi) = \sum_{j=0}^{k} a_j \xi^{k-j},$$

and the equation $\phi(\xi) = 0$ is called *characteristic equation*. For a small perturbation of the initial condition of the difference equation not to grow unboundedly, the bound on the root of $\phi(\xi)$ is needed.

Definition 2.2.4
The difference equation (2.9) is ***stable*** *if all k roots of $\phi(\xi)$ satisfy $|\xi_i| < 1$ or if $|\xi_i| = 1$, then ξ_i is a simple root. The difference equation is* ***asymptotically stable*** *if all k roots of $\phi(\xi)$ satisfy $|\xi_i| < 1$.*

Absolute stability

Consider the *Dahlquist test equation*

$$\dot{x} = \lambda x, \; x_0 = 1. \tag{2.10}$$

If $\Re\lambda < 0$ then $|x(t)|$ decays exponentially. It means that this problem is stable if $\Re\lambda < 0$. Therefore, it is required that the solution of a numerical method, that is used for discretizing (2.10), satisfies the *absolute stability* condition

$$|x_n| \leq |x_{n-1}|, \quad n = 1, 2, \ldots.$$

Definition 2.2.5 (Stability domain of a numerical method)
The ***region of absolute stability****, also called* ***stability domain****, of a numerical method is a region in the complex z-plane such that the numerical solution, obtained by applying the method to the test equation (2.10) with $z = \lambda h$, where h is the step size, from within this region, satisfies the absolute stability condition.*

Definition 2.2.6 (A-stable method)
A numerical method is ***A-stable*** *if its region of absolute stability contains the entire left half-plane of $z = h\lambda$.*

Now applying the multistep method (2.4) to the test equation $\dot{x} = \lambda x$, we obtain the following difference equation:

$$\sum_{j=0}^{k} (\alpha_i - h\lambda\beta_j) x_{n-j} = 0.$$

Its characteristic equation is given by

$$\phi(\xi) = \sum_{j=1}^{k} (\alpha_j - h\lambda\beta_j)\xi^{k-j} = \rho(\xi) - h\lambda\sigma(\xi) = 0,$$

where $\rho(\xi)$ and $\sigma(\xi)$ are defined as

$$\rho(\xi) = \sum_{j=1}^{k} \alpha_j \xi^{k-j}, \ \sigma(\xi) = \sum_{j=1}^{k} \beta_j \xi^{k-j},$$

and are called *generating polynomials*. The boundary of the stability domain is determined by

$$\mu = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})}, \quad 0 \leq \theta \leq 2\pi,$$

is called the *root locus curve*.

Theorem 2.2.2 (Dahlquist 1963)
An A-stable multistep method must be of order $p \leq 2$.

Stiffness

The concept of stiffness is usually described using qualitative properties rather than quantitative terms. It usually refer to problems with multiple time scales, such as in chemical reaction systems stiffness is due to the fact that some reactions occur much more rapidly than others. The first definition of stiff equations is given in [38] as: *"stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit one"*. Alternatively, the stiffness of problem is also defined in [5] as: *"the problem is **stiff** if the step size needed to maintain absolute stability of the forward Euler method is much smaller than the step size needed to represent the solution accurately"*.

2.2.3 Stability of BDF methods

The BDF methods are a family of the multistep methods (2.4), where $\beta_j = 0$ for $j = 2, \ldots, k$. The k-step BDF method with a constant step size h can be written as

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j x_n = h\dot{x}_n.$$

This method has order $p = k$. The root locus curves is given by

$$\mu = \sum_{j=1}^{k} \frac{1}{j}(1 - \frac{1}{\xi})^j = \sum_{j=1}^{k} \frac{1}{j}(1 - e^{-i\theta})^j, \quad 0 \leq \theta \leq 2\pi.$$

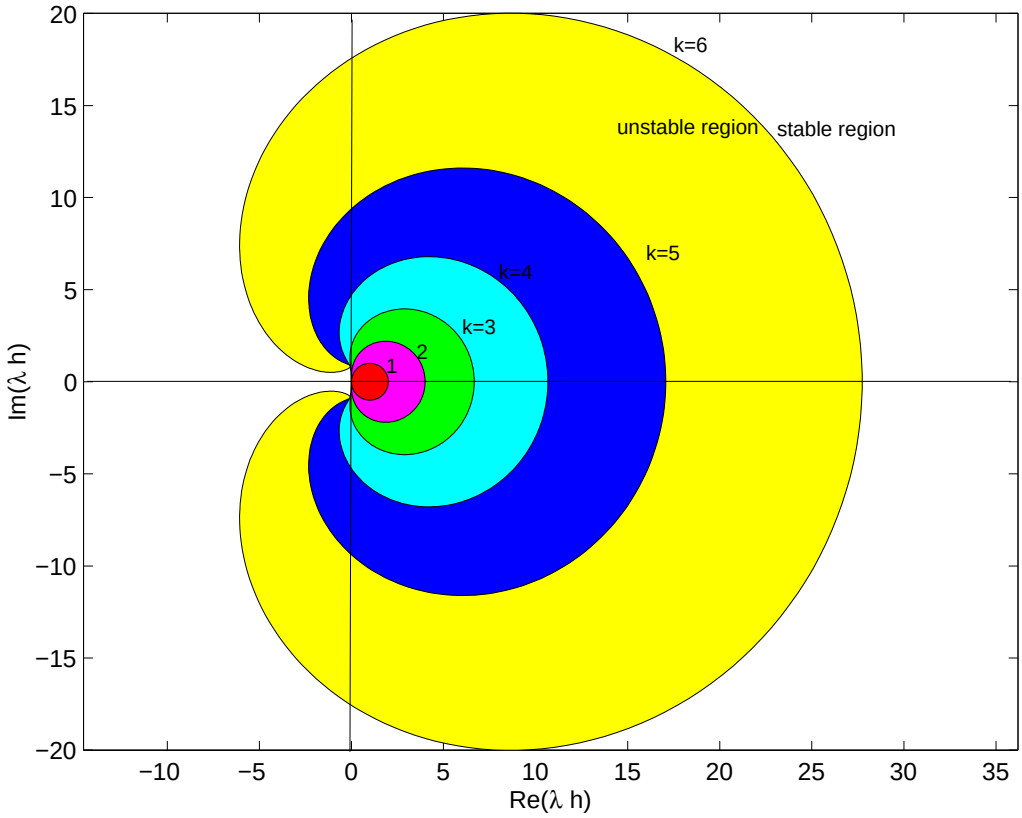

Figure 2.1: Stability region of BDF methods up to order 6.

Figure 2.1 shows the stability region of the BDF methods up to order 6. The stability region of each k-step BDF method is the region outside the corresponding colored area.

2.3 BDF methods for index-1 DAE

Consider the semi-implicit quasilinear DAE

$$\begin{aligned} B(t,x,y)\dot{x} &= f(t,x,y) \\ 0 &= g(t,x,y), \end{aligned} \tag{2.11}$$

where

$$\begin{aligned} B: &\quad [t_0,t_e]\times\mathbb{R}^{n_\mathrm{d}}\times\mathbb{R}^{n_\mathrm{a}}\to\mathbb{R}^{n_f\times n_\mathrm{d}}, \\ f: &\quad [t_0,t_e]\times\mathbb{R}^{n_\mathrm{d}}\times\mathbb{R}^{n_\mathrm{a}}\to\mathbb{R}^{n_f}, \\ g: &\quad [t_0,t_e]\times\mathbb{R}^{n_\mathrm{d}}\times\mathbb{R}^{n_\mathrm{a}}\to\mathbb{R}^{n_g}. \end{aligned}$$

In addition, $n_\mathrm{d}\geq n_f$ and B has full range. Here, x is the differential variables and y is the algebraic variables.

The basic idea of BDF discretization is to approximate the derivative of differential variables $\dot{x}$ by the derivative of an interpolating polynomial. Given k values of $x(t)$ at t_{m-k+1}, t_{m-k+2}, ..., t_m. We construct a k-order *corrector polynomial* $p^{\mathrm{c}}_{m+1}(t)$ which interpolates $x(t)$ using support points (t_{m-k+1}, x_{m-k+1}), (t_{m-k+2}, x_{m-k+2}), ..., (t_m, x_m), which are already known, and (t_{m+1}, x_{m+1}). That is,

$$p^{\mathrm{c}}_{m+1}(t_{m+1-i}) = x_{m+1-i}, \quad i = 0, \ldots, k.$$

Note that $p^{\mathrm{c}}_{m+1}(t)$ is unique and its coefficients also depend on the unknown x_{m+1}.

Now the unknown derivative $\dot{x}_{m+1}$ is approximated by the derivative of the corrector polynomial $p^{\mathrm{c}}_{m+1}(t)$ at t_{m+1}

$$\begin{aligned} \dot{x}_{m+1} &= \dot{p}^{\mathrm{c}}_{m+1}(t_{m+1}) \\ &= -\frac{1}{h_{m+1}} \left(\alpha_0^{(m+1)} x_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i} \right), \end{aligned} \tag{2.12}$$

where $h_{m+1} = t_{m+1} - t_m$ and $\alpha_i^{(m+1)}$, $i = 1, \ldots, k$, are the coefficients of the BDF method. For example, using Lagrange interpolation formula, these coefficients are determined as follows.

The corrector polynomial is determined by

$$p^{\mathrm{c}}_{m+1}(t; x_{m+1}, x_m, \ldots, x_{m-k+1}) = \sum_{i=m-k+1}^{m+1} x_i l_i(t), \tag{2.13}$$

where l_i are the Lagrange basis polynomial which are defined as

$$l_i(t) = \prod_{\substack{j=m-k+1 \\ j \neq i}}^{m+1} \frac{t - t_j}{t_i - t_j}, \quad i = m-k+1, \ldots, m+1. \tag{2.14}$$

Then, the coefficients α_0^{m+1}, ..., α_{k+1}^{m+1} are determined by

$$\alpha_i^{m+1} = -h_{m+1} \dot{l}_{m+1-i}(t_{m+1}), \quad i = 0, \ldots, k. \tag{2.15}$$

Let us denote

$$\begin{aligned} B^{m+1} &= B(t_{m+1}, x_{m+1}, y_{m+1}) \\ f^{m+1} &= f(t_{m+1}, x_{m+1}, y_{m+1}) \\ g^{m+1} &= g(t_{m+1}, x_{m+1}, y_{m+1}) \\ \beta_{m+1} &= \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i}. \end{aligned}$$

Replacing $\dot{x}_{m+1}$ in (2.11) evaluated at the step $(n+1)$ by the right-hand side of (2.12) we obtain the algebraic equation system

$$\begin{aligned} B^{m+1}\left(\alpha_0^{(m+1)} x_{m+1} + \beta_{m+1}\right) + h_{m+1} f^{m+1} &= 0 \\ g^{m+1} &= 0 \end{aligned} \tag{2.16}$$

This nonlinear equation system is solved by a modified Newton method, which will be discussed in Section 2.4.

A suitable initial guess w_{m+1}^0, where $w_{m+1} = (x_{m+1}, y_{m+1})$, being close enough to the solution of equation (2.16), is determined by extrapolating a k-order *predictor polynomial* $p_{m+1}^{\mathrm{p}}(t)$, which interpolates $w(t)$ using $k+1$ already known support points (t_{m-k}, w_{m-k}), (t_{m-k+1}, w_{m-k+1}), $\ldots$, (t_m, w_m)

$$p_{m+1}^{\mathrm{p}}(t_{m+1-i}) = w_{m+1-i}, \quad i = 1, \ldots, k+1,$$

by setting

$$w_{m+1}^0 = p_{m+1}^{\mathrm{p}}(t_{m+1}).$$

The predictor polynomial is given by

$$p_{m+1}^{\mathrm{p}}(t) = \sum_{i=0}^{k} p_i(t) \nabla^i w_m,$$

where

$$p_i(t) = \begin{cases} 1, & \text{if } i = 0 \\ \prod_{j=1}^{i} (t - t_{m+1-j}), & \text{if } i = 1, \ldots, k+1 \end{cases}$$

and

$$\begin{aligned} \nabla^0 w_m &= w_m\,, \\ \nabla^i w_m &= \frac{\nabla^{i-1} w_m - \nabla^{i-1} w_{m-1}}{t_m - t_{m-i}}\,. \end{aligned} \tag{2.17}$$

Remark 2.3.1
The coefficients α_0^{m+1}, $\ldots$, α_{k+1}^{m+1} depend on the order k and step sizes h_{m+1-i} $(i = 0, \ldots, k)$.

If the step sizes do not change, i.e., constant step-size, then the coefficients only depend on the order k. For a given order k, the coefficients are constant. Table 2.3 gives the coefficients of BDF methods up to order 6.

In DAESOL the step size and order are changed adaptively, thus the coefficients are evaluated at each step.

k	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	-1	1					
2	-3/2	2	-1/2				
3	-11/6	3	-3/2	1/3			
4	-25/12	4	-3	4/3	-1/4		
5	-137/60	5	-5	10/3	-15/12	1/5	
6	-147/60	6	-15/2	20/3	-15/4	6/5	-1/6

Table 2.1: Coefficients of BDF methods up to order 6

2.4 Solution of corrector equation

For solving large scale DAEs, the most time consuming parts are computation and decomposition of the Jacobian, in addition, the Jacobian usually changes very little during the Newton iteration and even during several integration steps. Therefore, to save computing time, the corrector equation (2.16) are solved by a modified Newton method instead of the standard Newton's method. The corrector equation (2.16) can be written as

$$h(s) = 0, \tag{2.18}$$

where $s = (x_{m+1}, y_{m+1})$ and

$$h = \begin{bmatrix} B^{m+1}\left(\alpha_0^{(m+1)} x_{m+1} + \beta_{m+1}\right) + h_{m+1} f^{m+1} \\ g^{m+1} \end{bmatrix}.$$

Notation used in the remaining of this section is not related to previous sections.

The iteration scheme of the modified Newton method is defined as

$$\begin{aligned} \widetilde{J}(s_k)\Delta s_k &= -h(s_k), \\ s_{k+1} &= s_k + \Delta s_k. \end{aligned} \tag{2.19}$$

where $\widetilde{J}(s_k)$ is an approximation to the Jacobian $J(s_k)$. The convergence properties of the modified Newton method can be formulated as follows [19].

Theorem 2.4.1
Let $\mathcal{D} \subseteq \mathbb{R}^n$, $h \in C^1(\mathcal{D})$ and $J(s) = \partial h(s)/\partial s$ be the Jacobian of $h(s)$ and $\widetilde{J}^{-1}(s)$ be the approximate inverse of $J(s)$. For all $\tau \in [0,1]$ and all k there are bounds ω and κ such that

(i) $\|\widetilde{J}^{-1}(s_{k+1})(J(s_k) - J(s_k - \tau\Delta s_k))\Delta s_k\| \leq \omega\tau\|\Delta s_k\|^2, \quad \omega < \infty,$

(ii) $\|\widetilde{J}^{-1}(s_k)(h(s_k) - J(s_k)\widetilde{J}^{-1}(s_k)h(s_k))\| \leq \kappa\|\Delta s_k\|, \quad \kappa < 1$

with $\Delta s_i = -\widetilde{J}^{-1}(s_i)h(s_i)$ *and the starting point of the iteration has to fulfill*

(iii) $\delta_0 := \dfrac{\omega}{2}\|\Delta s_0\| + \kappa < 1,$

(iv) The ball $\mathcal{D}_0 := S\left(s_0, \dfrac{\|\Delta s_0\|}{1-\delta_0}\right) \subset \mathcal{D}.$

Then the following holds:

- *The iteration* $s_{k+1} = s_k + \Delta s_k$ *is well-defined and remains in* $\mathcal{D}_0$.
- *There exists* $s^* \in \mathcal{D}^0$ *with* $\widetilde{J}^{-1}(s^*)h(s^*) = 0$ *and* $s_k \to s^*$ $(k \to \infty)$.
- *The convergence is linear with*

$$\|\Delta s_{k+1}\| \leq \left(\frac{\omega}{2}\|\Delta s_k\| + \kappa\right)\|\Delta s_k\| = \delta_k\|\Delta s_k\|. \tag{2.20}$$

- *For the k-th iteration the following a priori estimate holds*

$$\|s_k - s^*\| \leq \|\Delta s_0\|\frac{\delta_0^k}{1-\delta_0}. \tag{2.21}$$

Remark 2.4.1

- *The conditions (i) and (ii) only need to be satisfied for* $\Delta s_k = -\widetilde{J}^{-1}(s_k)h(s_k)$ *and not nessarily for arbitrary* Δs_k.
- *The Lipschitz constant* ω *in the condition (i) measures the relative nonlinearity of h. This condition is usually replaced by two conditions:* $\|\widetilde{J}^{-1}\| \leq \beta < \infty$ *and* $\|J(y) - J(x)\| \leq \gamma\|y - x\|$, $\gamma < \infty$. *And* ω *can be thought as* $\beta\gamma$. *However,* $\beta\gamma$ *grossly over-estimates the weaker bound* ω.
- κ *is a measure for the quality of the approximate inverse* $\widetilde{J}^{-1}$. *The condition (ii) can be replaced by* $\|I - \widetilde{J}^{-1}J\| \leq \kappa < 1$.

The main use of Newton's method in the corrector stage is to compute a "good" solution by using a quite good initial value obtaining from the predictor. Thus, the use of of the Newton's method in context of predictor-corrector method is not the same as in the standard procedure for solving nonlinear equations.

In DAESOL, after the first iteration, if the weighted norm of $\|\Delta s_0\|$ is less than or equal to the requested iteration tolerance NTOL

$$\|\Delta s_0\| \leq \text{NTOL}, \tag{2.22}$$

then the iteration is considered to be successful. Here $\|\cdot\|$ is the weighted root mean square norm defined as

$$\|e\|_{\text{WRMS}} = \left(\frac{1}{n} \sum_{i=1}^{n} \left(\frac{e_i}{\text{yscal}(i)} \right)^2 \right)^{1/2}, \tag{2.23}$$

where $e = (e_1, e_2, \ldots, e_n)$, $\text{yscal}(i) = \text{RTOL} \times |y(i)| + \text{ATOL}(i)$, RTOL is the relative error tolerance, $\text{ATOL}(i)$ is the ith component of absolute error tolerance, and $y(i)$ is the ith component of the solution vector of the DAEs. This is one of the scaling schemes implemented in DAESOL, which is used for the numerical results in this thesis, see [10] for more details. The requested iteration tolerance is chosen as follows

$$\text{NTOL} = \text{RTOL} \times \text{RFAC},$$

where RFAC = 0.08, RTOL is the user requested relative integration tolerance. The aim here is to control the error of the solution of the corrector equation such that it does not effect the local discretization error estimates. If the condition (2.22) is not satisfied after the first iteration, then another iteration is taken. The convergence ratio δ is estimated by

$$\delta_0 = \frac{\|\Delta s_1\|}{\|\Delta s_0\|}. \tag{2.24}$$

If the convergence ratio is less than 0.25, $\delta < 0.25$, or $\|\Delta s_1\| < \text{TOL}$ then we stop the iteration and it is considered as convergent. If the estimated convergence ratio δ_0 is greater than 0.3, $\delta_0 > 0.3$, then the iteration is considered to be unsuccessful and if the Jacobian is old, we restart the iteration process from beginning with a new updated Jacobian, otherwise we stop the iteration process and request the step size to be reduced. Otherwise the third iteration is made, and a new estimate of convergence ratio δ_1 is computed

$$\delta_1 = \frac{\|\Delta s_2\|}{\|\Delta s_1\|}. \tag{2.25}$$

A new convergence test is performed: if $\|\Delta s_2\|$ is less than TOL, then the iteration is stopped with a successful return. Otherwise, if the estimated

convergence ratio δ_1 is greater than 0.3, $\delta_1 > 0.3$, then the iteration is considered to be unsuccessful and if the Jacobian is old, we restart the iteration process from beginning with a new updated Jacobian, otherwise we stop the iteration process with a unsuccessful return. For estimation of the step size after a rejected step, see [10] and [11].

The estimate of convergence ratio $\delta_0 = \|\Delta s_1\|/\|\Delta s_0\|$ as above is a lower bound for the δ_0 in Theorem 2.4.1, and this should be taken into account.

As proved in [94], p. 301, the root-convergence rate (r-factor) $\rho^{(\mathrm{r})}$ of the iteration mentioned in Theorem 2.4.1 is the spectral radius of $(I - \widetilde{J}^{-1}J_*)$, $\rho^{(\mathrm{r})} = \rho(I - \widetilde{J}^{-1}J_*)$, with $J_* = \partial h(s^*)/\partial s$. Actually, the iteration mentioned in Theorem 2.4.1 can be written as

$$s_{k+1} = \overline{h}(s_k), \quad \overline{h}(s_k) = s_k - \widetilde{J}^{-1}h(s_k), \quad \overline{h}'(s^*) = I - \widetilde{J}^{-1}J_*.$$

Many authors (see [107], [97], [25], [24], [5], [77]) estimate the quotient-convergence rate (q-factor) $\rho^{(\mathrm{q})}$ as

$$\widetilde{\rho}_k^{(\mathrm{q})} = \frac{\|\Delta s_k\|}{\|\Delta s_{k-1}\|} \quad \text{or } \widetilde{\rho}_k^{(\mathrm{q})} = \left(\frac{\|\Delta s_k\|}{\|\Delta s_0\|}\right)^{1/k}, \tag{2.26}$$

which use the same or slightly different formula from the convergence ratio δ as we mention above (δ_0 and δ_1), and then these authors use the well known result that if a sequence $\{s_k\}$ converges to s^* with a quotient-convergence rate (q-factor) $\rho^{(\mathrm{q})}$, then

$$\|s_{k+1} - s^*\| \leq \frac{\rho^{(\mathrm{q})}}{1 - \rho^{(\mathrm{q})}}\|s_{k+1} - s_k\|,$$

to estimate how far s_{k+1} is from the solution s^*.

It is easy to prove that for a contraction mapping $c : \mathbb{R}^n \to \mathbb{R}^n$ defined by $\|c(x) - c(y)\| \leq \delta\|x - y\|$, $\forall x, y \in \mathbb{R}^n$, $\delta < 1$, then the iteration $s_{k+1} = c(s_k)$ converges to the unique fixed point of c and its asymptotic quotient-convergence $\rho^{(\mathrm{q})}$ and root-convergence $\rho^{(\mathrm{r})}$ factors have the upper bound δ

$$\rho^{(\mathrm{q})} \leq \delta \quad \text{and} \quad \rho^{(\mathrm{r})} \leq \delta.$$

Note that δ here is also an upper bound of δ_k in Theorem 2.4.1: $\delta_k \leq \delta$.

Moreover, it is proved in [40] (Lemma 8.2.3, p. 180) that if a sequence $\{s_k\}$ ($s_k \in \mathbb{R}^n$) converges q-superlinearly to $s^* \in \mathbb{R}^n$ in some norm $\|\cdot\|$, then

$$\lim_{k\to\infty} \frac{\|s_{k+1} - s_k\|}{\|s_k - s^*\|} = 1.$$

It suggests that one can replace $s_k - s^*$ by $\Delta s_k = s_{k+1} - s_k$ if the convergence is q-superlinear. However, in the corrector iteration we can obtain linear convergence, thus Δs_k could not be a good approximation for $s_k - s^*$. Therefore, the estimate (2.26) may be not a good approximation of the q-factor.

Now suppose that $\rho^{(\mathrm{q})} < 0.5$ and denote $\rho_k^{(\mathrm{q})}$ and δ_k as

$$\rho_k^{(\mathrm{q})} = \frac{\|s_{k+1} - s^*\|}{\|s_k - s^*\|}, \quad \delta_k = \frac{\|s_{k+2} - s_{k+1}\|}{\|s_{k+1} - s_k\|} = \frac{\|\Delta s_{k+1}\|}{\|\Delta s_k\|},$$

then

$$\begin{aligned}
\rho_k^{(\mathrm{q})} &= \frac{\|s_{k+1} - s^*\|}{\|s_k - s^*\|} \\
&\le \frac{\|s_{k+1} - s_{k+2}\| + \|s_{k+2} - s^*\|}{\|s_k - s_{k+1}\| - \|s_{k+1} - s^*\|} \\
&\le \frac{\|s_{k+1} - s_{k+2}\| + \dfrac{\rho^{(\mathrm{q})}}{1-\rho^{(\mathrm{q})}}\|s_{k+1} - s_{k+2}\|}{\|s_k - s_{k+1}\| - \dfrac{\rho^{(\mathrm{q})}}{1-\rho^{(\mathrm{q})}}\|s_k - s_{k+1}\|} \\
&= \frac{1}{1-2\rho^{(\mathrm{q})}}\frac{\|s_{k+2} - s_{k+1}\|}{\|s_{k+1} - s_k\|} = \frac{\delta_k}{1-2\rho^{(\mathrm{q})}},
\end{aligned}$$

and

$$\begin{aligned}
\rho_k^{(\mathrm{q})} &= \frac{\|s_{k+1} - s^*\|}{\|s_k - s^*\|} \\
&\ge \frac{\|s_{k+1} - s_{k+2}\| - \|s_{k+2} - s^*\|}{\|s_k - s_{k+1}\| + \|s_{k+1} - s^*\|} \\
&\ge \frac{\|s_{k+1} - s_{k+2}\| - \dfrac{\rho^{(\mathrm{q})}}{1-\rho^{(\mathrm{q})}}\|s_{k+1} - s_{k+2}\|}{\|s_k - s_{k+1}\| + \dfrac{\rho^{(\mathrm{q})}}{1-\rho^{(\mathrm{q})}}\|s_k - s_{k+1}\|} \\
&= (1-2\rho^{(\mathrm{q})})\frac{\|s_{k+2} - s_{k+1}\|}{\|s_{k+1} - s_k\|} = (1-2\rho^{(\mathrm{q})})\delta_k.
\end{aligned}$$

Hence,

$$(1-2\rho^{(\mathrm{q})})\delta_k \le \rho_k^{(\mathrm{q})} \le \frac{\delta_k}{1-2\rho^{(\mathrm{q})}}.$$

It follows that if $\rho^{(\mathrm{q})}$ is small enough, say $\rho^{(\mathrm{q})} = 0.1$, then $0.8 \times \delta_k \le \rho_k^{(\mathrm{q})} \le 1.2 \times \delta_k$, i.e., δ_k can be a good approximation of $\rho^{(\mathrm{q})}$.

Fortunately, as mentioned above, the q-factor estimated using (2.26) could be an upper bound if one expects $\delta_k \approx \delta$. In other words, $\rho^{(q)}$ computed using (2.26) over-estimates the q-factor, and its use to estimate how far the current point is from the solution is safer because the bigger the q-factor is, the slower convergence rate is, is used to estimate $\|s_{k+1} - s^*\|$. It is emphasize that we estimate the convergence ratio δ, not the quotient-convergence rate $\rho^{(q)}$ as others do, and use (2.21) to estimate the distance between the current point s_k and the true solution s^*. This estimate does not depend on how good Δs_k approximates $s_k - s^*$.

2.5 Error control, order and step size selection

Because the global error is not easy to obtain and even inefficient, thus, most available DAE solvers (e.g., DAESOL [10] and DASSL[24]) do not try to control the global error but control the local error instead. The error is also referred to as the *integration error*, which is different from the *iteration error* of the solution of the corrector equation. At each integration step, after each successful corrector iteration we need to check for local error to decide if the step is accepted or rejected.

Let us denote (x, y) in (2.11) by w. As mentioned in Section 2.2, the local error l_{m+1} at t_{m+1} is defined as

$$l_{m+1} = w^{[1]}(t_{m+1}) - w_{m+1}.$$

As shown in, e.g., [27] and [24], the local error relates to the difference between the corrector and predictor values as

$$l_{m+1} = \zeta \times (w_{m+1} - w^0_{m+1}) + O(h^{k+2}), \tag{2.27}$$

where ζ only depends on the step sizes and orders, and w^0_{m+1} is the predictor value and w_{m+1} is the (exact) solution of the corrector equation. In DASSL, the local error is estimated by (2.27).

Alternatively, as we know that the k-step BDF has order k (see e.g., [5], [24]), i.e, the first k terms of the local error in the Taylor expansion at t_{m+1} vanish, thus, as in DAESOL, the local error is estimated by taking two major terms of order $k+1$ and $k+2$ in local Taylor expansion at t_{m+1} as

$$\begin{aligned} E_k(m+1) = \quad & h_{m+1}\psi_1(m+1)\ldots\psi_k(m+1) \\ & \left(\|\nabla^{k+1} w_{m+1}\| + \psi_{k+1}(m+1)\cdot\|\nabla^{k+2} w_{m+1}\|\right). \end{aligned} \tag{2.28}$$

After each step the local error estimates using the above formula (2.28) is evaluated and compared with the user requested tolerance TOL. If the estimated error $E_k(m+1)$ is greater than TOL, then the step is rejected and the step size is reduced.

Now assume that the computations are performed using floating-point arithmetics. We will analyze how this effects the above error estimates.

Denote

- w_{m+1} be the exact solution of the corrector equation with the computation using exact arithmetic and solving the corrector equation exactly (w_{m+1} denotes the true solution of the corrector equation), which is not the same as the true solution $w(t_{m+1})$ because we use the BDF formula to approximate the differential equation by the difference equation. The difference here is due to the BDF approximation.

- $\overline{w}_{n+1}$ be the numerical computed solution of the corrector equation by solving the corrector equation inexactly (due to terminating the iteration earlier) with exact arithmetic,

- $\widehat{w}_{m+1}$ be the approximation of $\overline{w}_{m+1}$ with the computation using floating point arithmetic.

We have

$$\begin{aligned} l_{m+1} &= (w^{\mathrm{l}}(t_{m+1}) - \widehat{w}_{m+1}) + (\widehat{w}_{m+1} - w_{m+1}) \\ &= (w^{\mathrm{l}}(t_{m+1}) - \widehat{w}_{m+1}) + (\widehat{w}_{m+1} - \overline{w}_{m+1}) + (\overline{w}_{m+1} - w_{m+1}). \end{aligned}$$

The term $\widehat{w}_{m+1} - w_{m+1}$ represents the error due to solving the corrector equation numerically. It consists of two sources of error, the error from terminating the corrector iteration after a finite number of iterations (*iteration error* $= \overline{w}_{m+1} - w_{m+1}$), and the error due to propagation of rounding error during solving linear system (2.19) at each corrector iteration (*roundoff error* $= \widehat{w}_{m+1} - \overline{w}_{m+1}$). It is usually assumed that the roundoff error is insignificant, and the iteration error is controlled to be much smaller than the local error as in Section 2.4 such that it does not affect the local error estimate. Then, the local error can be approximated as

$$l_{m+1} \approx w^{l}(t_{m+1}) - \widehat{w}_{m+1}.$$

The bound in (2.21) allows us to control the iteration error based on the Newton update Δw^0_{m+1}. We can determine how far from the current point to the true solution of the corrector equation is. Since what we have is the numerical computed solution $\widehat{\Delta} w^0_{m+1} = \Delta w^0_{m+1} + \delta w^0_{m+1}$, where δw^0_{m+1} is the

error of the computed solution of the linear system (2.19) due to computation using floating point arithmetic instead of the exact solution Δw^0_{m+1}, thus, if $\delta w^0_{m+1} \ll \Delta w^0_{m+1}$ then one can use $\widehat{\Delta} w^0_{m+1}$ as a good approximation of Δw^0_{m+1}, $\widehat{\Delta} w^0_{m+1} \approx \Delta w^0_{m+1}$. Therefore, we can bound the iteration error based on the numerical computed Newton update $\widehat{\Delta} w^0_{m+1}$. In the next section we will discuss how to estimate the error of the solution of the linear system and how it affects the iteration errror.

The error estimates (2.27) or (2.28) depend explicitly or implicitly on the approximate solution of the corrector equation, which depends on the initial Newton update Δw^0_{m+1} and the estimated convergene ratio (or convergence rate). These quantities depend on the solution of underlying linear system. These error estimates are reliable if the error induced by the numerical computations is small compared to the quatities of interest.

For estimation of a new step size and order for the next integration step, see [10] and [11].

2.6 Error analysis

It is well known that when one uses a digital computer to find a numerical solution of a problem, there will be have a certain error in the obtained solution due to only finite approximate representatives of numbers and inexact arithmetic operations in the computer, i.e., floating-point numbers and floating-point operations. One would expect that numerical computational solutions cannot be more accurate than machines allow. This fact was pointed out in [116] as:
"... when a problem in pure or applied mathematics is "solved" by numerical computation, errors, that is, deviations of the numerical "solution" obtained from the true, rigorous one, are unavoidable. Such a "solution" is therefore meaningless, unless there is an estimate of the total error in the above sense ..." .
In the rest of this section, at first, major classical results [123] and [59] on error analysis of linear equation systems are briefed, then we move on to error analysis for the solution of nonlinear systems by the numerical Newton method in the BDF methods.

2.6.1 Error analysis of direct Gaussian elimination for the solution linear equation systems

Consider the linear system (2.19) arising in solving the corrector equation

$$\widetilde{J}(x_k)\Delta x_k = -h(x_k).$$

To avoid using complicated notations, we recast the notations as $x = \Delta s^m$, $A = \widetilde{J}$, $b = -h(s^m)$, and $n = n_{\rm d} + n_{\rm a}$, then this linear equation becomes

$$Ax = b. \tag{2.29}$$

Notation used in the remaining of this section is not related to previous sections.

We study the problem (2.29) with uncertainty in data A and b. Consider the equation (2.29) with the right-hand side is changed from b to $b + \Delta b$, then the exact solution will be changed from x to $x + \Delta x$, and we have

$$A(x + \Delta x) = b + \Delta b.$$

The following bound for the relative error is obtained

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\|\|\Delta b\|}{\|A\|^{-1}\|b\|} = \|A\|\|A^{-1}\|\frac{\|\Delta b\|}{\|b\|},$$

where $\|\cdot\|$ denotes any *vector norm* and the corresponding *operator norm* (also referred to as *induced norm* or *subordinate matrix norm*, see e.g., [39] p. 22, [110] p. 186, which is *consistent* with the vector norm).

Define $\kappa(A) = \|A\|\|A^{-1}\|$, called *condition number* of A with respect to the given norm, then

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A)\frac{\|\Delta b\|}{\|b\|}. \tag{2.30}$$

The condition number $\kappa(A)$ measures the relative change $\|\Delta x\|/\|x\|$ in the solution as a multiple of the relative change $\|\Delta b\|/\|b\|$ in the data. Indeed, it reflects the maximum relative change in the solution in response to the relative change in the data. If the condition number is very large, then a small change in the data *could* cause a big change in the solution. As proved in [39] (Theorem 2.1, pp. 33–34), the reciprocal of the condition number equals the distance to the nearest singular matrix:

$$\min\left\{\frac{\|\Delta A\|_2}{\|A\|_2} : A + \Delta A \text{ } \textit{singular}\right\} = \frac{1}{\|A^{-1}\|_2\|A\|_2} = \frac{1}{\kappa_2(A)},$$

and here we assume that A is *non-singular*. For each norm, we have a corresponding condition number associated with that norm. For three most often used norms: $1-$, $2-$, and $\infty-$norms; we have the following relations.

$$\begin{aligned}
\|A\|_1 &= \max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|,\\
\|A\|_\infty &= \max_{1\le i\le n}\sum_{j=1}^{n}|a_{ij}|,\\
\|A\|_2 &= (\text{maximum eigenvalue of } A^TA)^{\frac{1}{2}},\\
&= \text{maximum singular value of A},
\end{aligned}$$

and

$$\begin{aligned}
\frac{1}{n}\kappa_2(A) \le \kappa_1(A) \le n\kappa_2(A),\\
\frac{1}{n}\kappa_\infty(A) \le \kappa_2(A) \le n\kappa_\infty(A),\\
\frac{1}{n^2}\kappa_1(A) \le \kappa_\infty(A) \le n^2\kappa_1(A).
\end{aligned}$$

The condition number $\kappa_2(A)$, associated with the 2-norm, is called the *spectral condition number*.

Now we consider the equation (2.29) but the left-hand side matrix A is changed from A to $A+\Delta A$, then the exact solution will change from x to $x+\Delta x$ and we have

$$(A+\Delta A)(x+\Delta x) = b.$$

The following bound for the relative error is obtained

$$\frac{\|\Delta x\|}{\|x\|} \le \frac{\kappa(A)}{1-\kappa(A)\dfrac{\|\Delta A\|}{\|A\|}}\frac{\|\Delta A\|}{\|A\|}, \tag{2.31}$$

with the condition that $\|A^{-1}\Delta A\| < 1$.

Remark 2.6.1
A similar bound for the relative change of the first order approximation δx of Δx is obtained in [115]:

$$\delta x = -A^{-1}(\Delta A)x \tag{2.32}$$

$$\frac{\|\delta x\|}{\|x\|} \le \kappa(A)\frac{\|\Delta A\|}{\|A\|}. \tag{2.33}$$

It is also shown in [115] that for any positive γ there exists a ΔA with $\|\Delta A\| = \gamma$ such that the equality sign in (2.33) occurs.

Now we consider the general case where both A and b are changed. Assume that we have

$$(A + \Delta A)(x + \Delta x) = b + \Delta b. \tag{2.34}$$

The following error bound for the relative error is obtained

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\dfrac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta b\|}{\|b\|} + \frac{\|\Delta A\|}{\|A\|} \right) \tag{2.35}$$

with the condition that $\|A^{-1}\Delta A\| < 1$.

Note that the quantity $\kappa(A)/(1 - \kappa(A)\|\Delta A\|/\|A\|)$ in the bounds (2.31) and (2.35) is close to the condition $\kappa(A)$ if $(\kappa(A)\|\Delta A\|/\|A\|) \ll 1$.

If the condition number is very large, this bound is too pessimistic. Using the *componentwise relative perturbation* $|\Delta A| \leq \alpha |A|$ where the absolute $|\cdot|$ of a vector or matrix means that the components are replaced by their absolute values, one can obtain (see, e.g., [39])

$$\frac{\|\Delta x\|}{\|x\|} \leq \alpha \kappa_{\mathrm{CR}}(A), \tag{2.36}$$

where $\kappa_{\mathrm{CR}} = \| \, |A^{-1}| \, |A| \, \|$ is the *componentwise relative condition number* or *relative condition number* for short.

Remark 2.6.2
The equality sign in (2.30) could happen for special A and/or b. That is Δb to be a maximizing vector for A^{-1}, i.e. $\|A\| = \|Ax\|/\|x\|, x \neq 0$, and x to be a maximizing vector for A, and $b = Ax$. For example, with the 2-norm, this occurs only when b is in the direction of the eigenvector of $A^T A$ corresponding to the largest singular value of A, and Δb is in the direction of the eigenvector of $A^T A$ corresponding to the smallest singular value of A. Furthermore, if b is a maximizing vector of A^{-1}, we even have $\|\Delta x\|/\|x\| \leq \|\Delta b\|/\|b\|$.

However, the equality sign in (2.30) happens only for special A and/or b (as with 2-norm, b and Δb are in a 1-dimensional subspace of $\mathbb{R}^n$ with assumption that all singular values are separated). Thus, the probability that the equality takes place for A, of which all singular values are not the same, is zero. Moreover, we know no place in the literature where the equality signs in (2.31) and (2.35) are given explicitly. Following the above derivation, we see that the equality sign in (2.31) and in (2.35) could be hardly possible, if not impossible. Therefore, although no sharper bound could be obtained, but in practice when $\kappa(A)$ is very large, these bounds are too pessimistic.

The above defined condition number $\kappa(A)$ reflects the maximum relative change in the solution in responding to a relative change in the data, and it only depends on the matrix of coefficients A, not depends on the right-hand side b. The above obtained bounds are valid for all values of b, as we discussed above, they are too pessimistic. On the other hand, one is usually interested in how is the solution x of $Ax = b$ sensitive to relative small changes of data (A and/or b). This means that we are interested in how the solution behaves by small perturbations of A and of b in the *locally* regions of A and b. This leads to the definitions of condition numbers which involve A and b or A and x. Somewhat similar concepts are introduced by Sluis in [115], and by Skeel in [109] which defines the condition number for a problem with data ξ and the solution $\phi(\xi)$ to be

$$\lim_{\widetilde{\xi}\to\xi} \frac{\text{relative distance from } \phi(\widetilde{\xi}) \text{ to } \phi(\xi)}{\text{relative distance from } \widetilde{\xi} \text{ to } \xi}, \tag{2.37}$$

and in [8] Bauer defines a similar quantity $\xi\phi'(\xi)/\phi(\xi)$, called the *relative derivative* or the *local differential condition number* of the scalar function ϕ of the scalar ξ. For a linear system, $\xi = (A, b)$ and $\phi(\xi) = A^{-1}b$. Based on this derivative-based definition of condition number, one can generalize it for multi-variable vector functions by using appropriate norms.

As shown in [109], the condition number as defined in (2.37) with the ∞-norm applied for a linear system $Ax = b$ equals [109]

$$\kappa^{(\mathrm{S})}(A, b; x) = \frac{\| \, |A^{-1}| \, |A| \, |x| + |A^{-1}| \, |b| \, ||_\infty}{\|x\|_\infty},$$

where $x = A^{-1}b$. When only A is subjected to uncertainty (data $\xi = A$), the condition number, as defined in (2.37), is

$$\kappa^{(\mathrm{S})}(A; x) = \frac{\| \, |A^{-1}| \, |A| \, |x| \, \|_\infty}{\|x\|_\infty}.$$

Note that we have

$$\kappa^{(\mathrm{S})}(A; x) \le \| \, |A^{-1}| \, |A| \, \|_\infty \le \|A\|\|A^{-1}\|_\infty,$$

and $\| \, |A^{-1}| \, |A| \, \|$ is the componentwise relative condition number as defined in (2.36). By using componentwise relative perturbation for A and b ($|\Delta A| \le \epsilon|A|, |\Delta b| \le \epsilon|b|$), the following bound is obtained [109]:

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \le \epsilon \frac{\| \, |A^{-1}| \, |A| \, |x| + |A^{-1}| \, |b| \, \|_\infty}{(1 - \epsilon\| \, |A^{-1}| \, |A| \, \|_\infty)\|x\|_\infty},$$

where $Ax = b$ and $(A + \Delta A)(x + \Delta x) = b + \Delta b$ and assume that $(1 - \epsilon \| \, |A^{-1}| \, |A| \, \|) > 0$. However, we will not discuss the componentwise relative perturbation and its error bounds any more because we think that assumption of the componentwise relative error ($|\Delta A| \leq \epsilon |A|, |\Delta b| \leq \epsilon |b|$) would be inappropriate or impractical for the error analysis in the following. For example, if one of entries of A equals zero, $A_{i\,j} = 0$, then due to roundoff errors in the numerical computation during solving the linear system, it is possible that the error of A at this location (i, j) does not vanish, i.e., $\Delta A_{i,j} \neq 0$, thus the condition $|\Delta A| \leq \epsilon |A|$ cannot be satisfied.

As we known from the backward error analysis that the computed solution x of the equation (2.29) is the exact solution of a perturbed one of the equation (2.29), i.e.,

$$(A + \Delta A)x = b + \Delta b, \tag{2.38}$$

where ΔA and Δb are some perturbation values, which depend on the numerical method used for solving (2.29).

Typically, in a BDF code, one needs to solve many linear systems with different b but with the same A. This is due to the fact that for efficiency of the code, a usually used strategy is to keep the iteration matrix (here is A) as long as possible, which in turn reduces the number of costly derivatives and iteration matrix evaluations and factorization. Thus, the linear system is not solved directly in its original form, such as using direct Gaussian elimination, but instead the matrix A is factored into the product of triangular matrices and then the backward substitution is used to solve the triangular linear systems. Note that the cost for factorizing a dense matrix is $O(n^3)$, where n is the dimension of the matrix, while the cost for solving a triangular linear system is $O(n^2)$. The advantage is that we only need to factor A once, and use it to solve the linear system with different b. In the following we determine the bounds for ΔA and Δb when the matrix A is factored into the LU form, where L is a lower triangular matrix and U is a upper triangular matrix. This is the approach used in DAESOL. The solution process of the linear system (2.29) by LU factorization and backward substitution can be summarized as follows.

$$\begin{aligned} LU &= A + E \\ (L + \delta L)y &= b \\ (U + \delta U)x &= y, \end{aligned} \tag{2.39}$$

where E, δL, and δU are error matrices due to floating point arithmetic.

From (2.39), it follows that

$$\begin{aligned}
&(L+\delta L)(U+\delta U)x = (L+\delta L)y = b \\
&(LU + U\delta L + L\delta U + \delta L\delta U)x = b \\
&(A + E + U\delta L + L\delta U + \delta L\delta U)x = b. \qquad (2.40)
\end{aligned}$$

Hence, the computed solution x is the exact solution of a perturbed equation

$$(A+\delta A)x = b, \qquad (2.41)$$

where

$$\delta A = E + U\delta L + L\delta U + \delta L\delta U.$$

If pivoting has been used, the $|l_{ij}| \leq 1$ for all i, j and denote

$$a = \max_k a_k = \max_k \max_{i,j} |\overline{a}_{i,j}^k|,$$

where $\overline{a}_{i,j}^k$ is the (i,j)-th element of $A^{(k)}$ at the k-th step of Gaussian elimination, then as shown in [122] the following relations hold

$$\begin{aligned}
&\|E\|_\infty \leq 2.01(0.5n+1)(n-1)a\epsilon_{\text{mach}} \\
&\|\delta L\|_\infty \leq 0.5(n^2+n+2)\epsilon_{\text{mach}} \\
&\|\delta U\|_\infty \leq 0.5(n^2+n+2)a\epsilon_{\text{mach}} \\
&\|L\|_\infty \leq 1 \\
&\|U\|_\infty \leq an.
\end{aligned}$$

Thus,

$$\begin{aligned}
\|\delta A\|_\infty &\leq (2.005n^2 + n^3 + 0.25n^4\epsilon_{\text{mach}})a\epsilon_{\text{mach}} \qquad (2.42) \\
a &= g(n)a_0
\end{aligned}$$

with

$$a_0 = \max_{i,j} a_{ij}$$

and $g(n)$ is the growth factor. A theoretical bound for the growth factor [121] is

$$g(n) \leq 2^{n-1} \text{ for partial pivot selection}$$

and

$$g(n) < (n-1)^{1/2}[2^1 3^{1/2} 4^{1/3} \dots (n-1)^{1/(n-2)}]^{1/2}$$

for complete pivot selection. A counter example (see, e.g., [121] and [123]) has been found for partial pivoting with the growth factor as large as 2^{n-1}. The theoretical bounds for the growth factor may be too pessimistic. For partial pivoting, as stated in [122] such growth is very rare and it is uncommon for $g(n)$ greater than 8. If A is not ill-conditioned, it is likely that the elements of successive $A^{(k)}$ will decrease. For complete pivoting, $g(n)$ could be not as large as n, but a counter example has been found (see [61] and [49]). However, we can monitor $g(n)$ during a triangular decomposition.

The bound in (2.42) can be written as

$$\|\delta A\|_\infty \quad \leq \quad g(n)(2.005n^2 + n^3 + 0.25n^4\epsilon_{\text{mach}})\epsilon_{\text{mach}}\|A\|_\infty \tag{2.43}$$

This bound is too optimistic and hardly attained in practice and as stated in [122] and [123] that $\|\delta A\|_\infty$ is rarely larger than $n\|A\|_\infty\epsilon_{\text{mach}}$. The bound in (2.43) is rewritten as

$$\|\delta A\|_\infty \quad \leq \quad f(n)\epsilon_{\text{mach}}\|A\|_\infty, \tag{2.44}$$

where

$$f(n) = g(n)(2.005n^2 + n^3 + 0.25n^4\epsilon_{\text{mach}}).$$

From (2.35) and (2.44) we obtain the following relative error bound (measured in the infinity norm) of the computed solution of the linear system by triangular decomposition LU and substitutions.

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \leq \frac{\kappa(A)_\infty}{1 - \kappa(A)_\infty f(n)\epsilon_{\text{mach}}} f(n)\epsilon_{\text{mach}}. \tag{2.45}$$

Here, we assume that $\|\Delta b\|_\infty/\|b\|_\infty \ll f(n)\epsilon_{\text{mach}}$. Recently, Amodio and Mazzia [2] are able to obtain a new bound for the relative error of the solution of linear system by triangular decomposition (LU) and substitutions:

$$\frac{\|\Delta x\|_\infty}{\|\widehat{x}\|_\infty} \leq \left(\frac{n(n+1)}{2} + 4(n-1)\right) g^{(\text{AM})}(n)\kappa(A)_\infty\epsilon_{\text{mach}}, \tag{2.46}$$

where $\widehat{x}$ is the computed solution and $g^{(\text{AM})}(n)$ is a newly-defined growth factor

$$g^{(\text{AM})}(n) = \frac{\max_k \|\widehat{A}^{(k)}\|_\infty}{\|A\|_\infty},$$

where $\widehat{A}^{(k)}$ is the computed value of $A^{(k)}$.

From the bound (2.45), we see that for a linear system having very large condition number such that

$$\kappa(A)_\infty f(n)\epsilon_{\text{mach}} \geq 1,$$

then the computed solution is unreliable and may not have any correct significant digits at all. However, the bound (2.45) maybe overestimate the error for particular cases. Alternatively, to estimate the error bound for a computed solution $\widehat{x}_0$ of $Ax = b$, we can solve a new system $Ax = \widehat{b}$, where $\widehat{b} = A\widehat{x}_0$, using the already factorized LU of A to obtain $\widehat{x}_1$, then the relative error can be estimated as $\|\widehat{x}_1 - \widehat{x}_0\|/\|\widehat{x}_1\|$. But with this approach, we need to solve an extra linear system with the same A but a different b.

2.6.2 Error analysis for Newton-like methods

Now, we investigate the behavior of Newton's method with floating point computation. The numerical Newton-like methods have been studied in, e.g., [79], [127], [126], [26], [41] and [111]. The results presented here , with the exception of [111] involve several assumptions and constants which are difficult to realized in practice and even using some assumptions we think that are not appropriate. These assumptions will be discussed later. Based on the results in [111] but with different interpretations and assumptions, we explicitly point out the limiting accuracy for the solution of nonlinear equations. From our analysis, limiting accuracy of certain classes of problems can also be obtained.

Notation used in the remaining of this section is not related to previous sections.

Consider Newton's method applied to the nonlinear equations

$$f(x) = 0,$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable on $\mathbb{R}^n$. Let J be the Jacobian matrix $\partial f/\partial x$ of f and assume that J is Lipschitz continuous with constant β in $\mathbb{R}^n$, i.e.,

$$\|J(x) - J(y)\| \leq \beta\|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

The iteration scheme defined by Newton's method with exact arithmetic is as follows

$$\begin{aligned} J(x_i)\Delta x_i &= -f(x_i), \\ x_{i+1} &= x_i + \Delta x_i. \end{aligned}$$

Using the floating point computation, the above iteration scheme becomes

$$\begin{aligned} (J(\widehat{x}_i) + E_i)\Delta\widehat{x}_i &= -(f(\widehat{x}_i) + e_i) \\ \widehat{x}_{i+1} &= \widehat{x}_i + \Delta\widehat{x}_i + \epsilon_i, \end{aligned} \tag{2.47}$$

where

- E_i is the error incurred in forming and solving the linear system for $\Delta\widehat{x}_i$,
- e_i is the error appearing in the evaluation of $f(\widehat{x}_i)$,
- ϵ_i is the error generated when adding $\Delta\widehat{x}_i$ to $\widehat{x}_i$.

We assume that the error E_i satisfies

$$\|E_i\| \leq \epsilon_{\text{mach}}\phi(f, \widehat{x}_i, n, \epsilon_{\text{mach}}). \tag{2.48}$$

We will describe how to determine the function ϕ later. According to the above linear error analysis (see (2.41) and (2.44)), we can safely assume that

$$\|E_i\| \leq \xi(n)\|J(\widehat{x}_i)\|,$$

for some small positive quantity $\xi(n)$.

For the error ϵ_i we have

$$\|\epsilon_i\| \leq \epsilon_{\text{mach}}(\|\widehat{x}_i\| + \|\Delta\widehat{x}_i)\|).$$

For stronger bound, the componentwise norm can be used. On a computer with a sufficiently accurate accumulator [122], we have

$$|\epsilon_i| \leq \epsilon_{\text{mach}}|(\widehat{x}_i + \Delta\widehat{x}_i)|,$$

here the componentwise norm is used.

The error e_i is usually assumed to obey the following error model [26], [79], [41]:

$$\|e_i\| \leq \epsilon_{\text{mach}}\|f(\widehat{x}_i)\|.$$

By using this assumption and others (see [26] for more detailes), Theorem 3.5 in [26] shows that the convergence point x^* of the numerical iterative process satisfies

$$\|x^* - x\| \leq \epsilon_{\text{mach}}\|x^*\|, \tag{2.49}$$

where x is the exact solution. We see that this assumption is inappropriate in particular at points near the solution. Therefore, the result (2.49) could not be obtained in practice. The following simple example shows that this assumption would be not appropriate.

Example 2.6.1
Consider the function $f(x)$ defined by

$$f(x) = -[(x+1) - 1 - \epsilon_{\text{mach}}]\epsilon_{\text{mach}}^{-1}.$$

Then the exact value of $f(\epsilon_{\text{mach}})$ would be zero $f(\epsilon_{\text{mach}}) = 0$, but the numerical computed value of $f(\epsilon_{\text{mach}})$ using the floating point arithmetic would be one, $\text{fl}(f(\epsilon_{\text{mach}})) = 1$, where $\text{fl}(f(x))$ denotes the value of $f(x)$ obtained by using floating point computation. Because with floating point computation the computed value of $(x+1)$ would be 1 when $x \leq \epsilon_{\text{mach}}$. Thus, the error in this case is $e = 1$.

We see that error models for $f(x)$ using the "relative" relation (the error of $\text{fl}(f(x))$ to be a factor of $f(x)$) would be inappropriate. By this relative model, one would expect the error of $\text{fl}(f(x))$ is always relative smaller than $f(x)$ even $f(x)$ is very small. As the above example shows this is not always true.

Therefore, we use the following error model for e_i:

$$\|e_i\| \leq \epsilon_{\text{mach}}\|f(\widehat{x}_i)\| + \psi(f, \widehat{x}_i, \epsilon_{\text{mach}}, \overline{\epsilon}).$$

Here, it is our intention to use the same formula for the error e_i as in [111] because we want to use some results in [111]. In [111], it is assumed that $f(\widehat{x}_i)$ is computed in the extended precision $\overline{\epsilon} \leq \epsilon_{\text{mach}}$ before rounding back to the working precision ϵ_{mach}, and $\delta\widehat{x}_i$, $\widehat{x}_i$ are computed using the precision ϵ_{mach}. We want to emphasize that the introducing of $\psi(f, \widehat{x}_i, \epsilon_{\text{mach}}, \overline{\epsilon})$ in the error model for e_i is based on our above observation, and it is not only because of evaluation of $f(\widehat{x}_i)$ in the extended precision as in [111]. The behavior of Newton's method with floating point arithmetic can be summarized as the following theorem [111].

Theorem 2.6.1
Assume that there is an x_* such that $f(x_*) = 0$, $J_* = J(x_*)$ is non-singular and that

$$\|J^{-1}E\| \leq \nu < 1.$$

Then, for all x such that

$$\beta\|J_*^{-1}\|\|x - x_*\| \leq \mu < 1,$$

$\widehat{x}_1$ in (2.47) is well defined and

$$\|\widehat{x}_1 - x_*\| \leq G\|\widehat{x}_0 - x_*\| + g,$$

where

$$\begin{aligned} G &= \frac{1}{1-\nu}\|J^{-1}E\| + \frac{(1+\epsilon_{\text{mach}})^2}{2(1-\mu)(1-\nu)}\beta\|J_*^{-1}\|\|\widehat{x}_0 - x_*\| \\ &\quad + \frac{\epsilon_{\text{mach}}(2+\epsilon_{\text{mach}})}{(1-\mu)(1-\nu)}\kappa(J_*) + \epsilon_{\text{mach}} \end{aligned}$$

and

$$g = \frac{1+\epsilon_{\text{mach}}}{(1-\mu)(1-\nu)}\|J_*^{-1}\|\psi(f, \widehat{x}_0, \epsilon_{\text{mach}}, \overline{\epsilon}) + \epsilon_{\text{mach}}\|x_*\|.$$

This theorem allows some interpretations. In the following, we assume that x fulfills all conditions in the theorem.

- If x_0 is very far away from the solution x_* such that $\|x_0 - x_*\| > g$, then the iteration (2.47) can improve x_0 to a new point nearer to x_* than x.
- If x_0 is far away from the solution x_* such that $\|x_0 - x_*\|$ is still large enough for the second term in G to be large compared to the other terms, then one can expect the numerical Newton method with quadratic improvement.
- If x_0 is near x_* such that $\|x_0 - x_*\|$ is small enough for the second term in G to be small compared to other terms, then the third term could be approximated by $\epsilon_{\text{mach}}\kappa(J_*)$. Thus, one can only expect linear improvement.

The limiting accuracy of the computed solution can be estimated by using the following corollary based on the above theorem [111].

Corollary 2.6.2
Assume that there is an x_* such that $f(x_*) = 0$ and $J_* = J(x_*)$ is non-singular and satisfies

$$\epsilon_{\text{mach}}\kappa(J_*) \leq \frac{1}{8}.$$

Assume also that for ϕ in (2.48),

$$\epsilon_{\text{mach}}\|J(\widehat{x}_i)^{-1}\|\phi(f, \widehat{x}_i, n, \epsilon_{\text{mach}}) \leq \frac{1}{8} \quad \forall i.$$

Then, for all x_0 such that

$$\beta\|J_*^{-1}\|\|x_0 - x_*\| \leq \frac{1}{8},$$

Newton's method in floating point arithmetic generates a sequence $\widehat{x}_i$ whose normwise relative error decreases until the first i for which

$$\frac{\|\widehat{x}_i - x_*\|}{\|x_*\|} \approx \frac{\|J_*^{-1}\|}{\|x_*\|}\psi(f, x_*, \epsilon_{\text{mach}}, \overline{\epsilon})$$

Now, we determine $\psi(f, \widehat{x}_i, \epsilon_{\text{mach}}, \overline{\epsilon})$ for some classes of $f(x)$.

Consider linear systems, $Ax = b$ where $A \in \mathbb{R}^{n\times n}$ is non-singular and b. To improve a computed solution $\widehat{x}$, the iterative refinement is used, which computes $r = b - A\widehat{x}$, then solves $A\Delta\widehat{x} = r$ for $\Delta\widehat{x}$, and compute an improved solution $y = \widehat{x} + \Delta\widehat{x}$. This process could be repeated with $\widehat{x}$ replaced by y. This is equivalent to Newton's method with $f(x) = b - Ax$, with $J(x) = -A$. Here, we are interested in determining the function ψ.

Recall that we assumed that $f(\widehat{x})$ is computed in the extended precision $\overline{\epsilon} \leq \epsilon_{\text{mach}}$ before rounding back to working precision ϵ_{mach}. Following the standard model of floating point arithmetic in [122], we have

$$\text{fl}(f(x))_i = [b_i(1 + \overline{\xi}_{ib}) - \sum_{i=1}^{n} a_{ij}x_j(1 + \overline{\xi}_{ij})](1 + \xi_i)$$

and the ith component $e(i)$ of the error vector e

$$e(i) = (\text{fl}(f(x)) - f(x))_i = \xi_i f_i(x) + \left(b_i\xi_{ib} - \sum_{i=1}^{n} a_{ij}x_j\overline{\xi}_{ij}\right)(1 + \xi_i),$$

where $|\xi_i| \leq \epsilon_{\text{mach}}$, $|\xi_{ib}| < (3/2)n\overline{\epsilon}$ and $|\xi_{ij}| < (3/2)(n + 1 - j)\overline{\epsilon}$. Denote $\gamma = (3/2)n\overline{\epsilon}$. Hence, it follows that

$$|e(i)| \leq \epsilon_{\text{mach}}|f_i(x)| + \gamma(|b_i| + \sum_{i=1}^{n} |a_{ij}x_j|).$$

Here we take $(1 + \xi_i) \approx 1$, and then for any *monotonic* norm (see [9], and [94] p.52, [113]) such as l_p-norms ($1 \leq p \leq \infty$) we have

$$\begin{aligned} \|e\| &\leq \epsilon_{\text{mach}}\|f(x)\| + \gamma(\|b\| + \|Ax\|) \\ &\leq \epsilon_{\text{mach}}\|f(x)\| + \gamma(\|b\| + \|A\|\|x\|). \end{aligned}$$

Thus, we take

$$\begin{aligned} \psi(f, \widehat{x}, \epsilon_{\text{mach}}, \overline{\epsilon}) &= \gamma(\|b\| + \|A\widehat{x}\|), \quad \text{or} \\ \psi(f, \widehat{x}, \epsilon_{\text{mach}}, \overline{\epsilon}) &= \gamma(\|b\| + \|A\|\|\widehat{x}\|). \end{aligned}$$

This result is similar with the one given in [111]. It follows from Corollary 2.6.2 that iterative refinement could reduce the relative forward error to $2\gamma\|A^{-1}\|\|b\|/\|x\| \leq 2\gamma\kappa(A)$.

Consider the case where $f(x)$ is a polynomial

$$f(x) = \sum_{i=0}^{n} a_i x^i,$$

and is evaluated using Horner's rule as

$$f(x) = (\ldots(((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x + \cdots + a_1)x + a_0.$$

Then, the value of $f(x)$ computed using floating point arithmetic is

$$\mathrm{fl}(f(x)) = \left(\sum_{i=0}^{n}(1+\overline{\epsilon}_i)a_i x_i\right)(1+\epsilon),$$

where $|\overline{\epsilon}_i| \leq 2n\overline{\epsilon}$ (see [39], p. 16) and $|\epsilon| \leq \epsilon_{\mathrm{mach}}$. It follows that

$$\begin{aligned} |\mathrm{fl}(f(x)) - f(x)| &= \left|\left(\sum_{i=0}^{n}(1+\overline{\epsilon}_i)a_i x_i\right)(1+\epsilon) - \sum_{i=0}^{n} a_i x^i\right| \\ &= |\epsilon f(x) + (1+\epsilon)\sum_{i=0}^{n}\overline{\epsilon}_i a_i x^i| \\ &\leq \epsilon_{\mathrm{mach}}|f(x)| + 2n\overline{\epsilon}(1+|\epsilon|)\sum_{i=0}^{n}|a_i x^i|. \end{aligned}$$

Based on this result, we take

$$\begin{aligned} \psi(f, \widehat{x}, \epsilon_{\mathrm{mach}}, \overline{\epsilon}) &= 2n\overline{\epsilon}(1+|\epsilon|)\sum_{i=0}^{n}|a_i x^i| \\ &\approx 2n\overline{\epsilon}\sum_{i=0}^{n}|a_i x^i|. \end{aligned}$$

Note that the above obtained error bounds depend on the mechanism used to evaluate the polynomial (here it is Horner's rule)

The error bound for the numerical computed value of $f(x)$ can be obtained if the reverse mode of automatic differentitation (see Section 2.8) is available. The following error bound can be expected to fulfil

$$|\widehat{x}_m - f(x)| \leq \sum_{i=1}^{m}|\overline{x}_i|\delta x_i, \tag{2.50}$$

where $\widehat{x}_m$ is the computed value of $f(x)$, δx_i is assumed to satisfy

$$|\widehat{x}_i - f_i(\widehat{x}_j)_{j\in\mathcal{J}_i}| \leq \delta x_i, \quad i > n.$$

Here $\widehat{x}_i$ $(i = n+1, \ldots, m)$ are the computed value of x_i, $\overline{x}_i$ is the derivative of $f(x)$ with respect to x_i, and δx_i $(i = 1, \ldots, n)$ is the bound for the error of the input x_i, which is usually the machine precision if the input data error is smaller compared to the machine precision. It is shown in [8] that the bound (2.50) holds if f_i are linear and $\overline{x}_i$ are exact. A similar bound is obtained in [71].

The conclusion to draw from this analysis is that the limiting accuracy of the solution of nonlinear equations by the numerical Newton method depends on the accuracy with which the residual $f(x)$ is computed, and does not depend on the condition number of the associated linear system as long as it is not too ill-conditioned ($\epsilon_{\text{mach}}\kappa(J*) \leq 1/8$). That is, roughly speaking, the condition of a nonlinear system is somewhat not related to the condition of the underlying linear system. The accuracy of the residual $f(x)$ does not only depend on the precision used for evaluating $f(x)$ but also depends on the way of which the code for evaluating $f(x)$ is programmed, such as the order of line codes (the evaluation sequence of the code).

2.7 Scaling ill-conditioned iteration matrices

The nonlinear equation system (2.16) is solved by a modified Newton method. At each Newton iteration, we need to solve the linear equation system (2.19) which has the form

$$Ax = b, \tag{2.51}$$

where $x = \Delta s^m \in \mathbb{R}^n$, $b = -h(s_k) \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n\times n}$ is regular, $A = \widetilde{J}$.

Notation used in the remaining of this section is not related to previous sections.

For problems under study the corresponding linear systems is likely to be ill-conditioning. Figures 2.2 and 2.3 show the condition numbers ($\kappa(A)_\infty = \|A\|_\infty \|A^{-1}\|_\infty$) of the NO2 and METHANE problems (see Chapter 5 for details) estimated by using LAPACK [3]. As one can see, the estimated condition numbers for unscaled case are very large, from 10^{14} to 10^{18}, while the machine precision is about 10^{-16}. It is well known that if the condition number of a linear system is greater than the reciprocal of machine precision, then the computed solution is untrustworthy. Hence, it is theoretically possible that we do not have any reliable digit for the unscaled case. This fact may cause severe error in the solution of linear system. Consequently, the error estimates in the DAE solver based on this solution is unreliable. A linear system is said to be ill-conditioned if a relatively small changes in data (A and/or b) can produce a relatively large changes in the solution. It is worth

noting that a linear system which has a high condition number in the sense of $\kappa(A) = \|A\|\|A^{-1}\|$ is not necessary ill-conditioned but an ill-conditioned system must have the high condition number. This can be easily seen from the bounds, such as (2.30), (2.31) and (2.35). On the other hand, with the condition number in the sense of (2.37) a high condition number means that the linear system is ill-conditioning and ill-conditioning also implies having the high condition number. However, to reduce the error of the solution of a linear system one usually applies an appropriate scaling to the linear system.

As shown in [109], an "optimal" diagonal scaling matrix (row scaling) for a linear system using Gaussian elimination with column pivoting would depend on the solution of the linear system itself. This would be impractical because the solution of the linear system is not available before the scaling. Here the word "optimal" means that the backward error η bound is minimized, i.e, $\eta \leq \chi(n)\epsilon_{\text{mach}}$, where $\chi(n)$ is a function of n, and η is defined as the smallest real number such that $(A + \Delta A)\widehat{x} = b$ for some ΔA with $|\Delta A| \leq \eta |A|$, and $\widehat{x}$ is a computed solution. Of course, one can estimates the solution in advance and uses it for determining the scaling matrix. Alternatively, one can also apply scaling to to reduce its condition number, this has been studied by many authors (see, e.g., [121], [6], [113], [7], [114], [87], [119], [60] [108], [96], [93], [1] and [89]), which in turn could reduce the error of the solution of the linear system.

The pivot selection in the Gaussian elimination with partial pivoting (colum pivoting—row interchanges) does *only* guarantee that for any non-singular matrix the process does not break down due to the fact that some pivot element vanishes. This pivot selection strategy does not always deliver a more accurate solution than other pivot selection strategies do, see example in [110], pp. 191–193. It is worth noting that scaling to reduce the condition number does not necessarily reduce the error of the numerical computed solution (as shown by examples in [110], pp. 191–193) and also because the scaled matrix $\widetilde{A}$ will have a different norm and perturbation matrix $\Delta\widetilde{A}$ (see, e.g., [115]), i.e., $\|\widetilde{A}\| \neq \|A\|$ and $\|\Delta\widetilde{A}\| \neq \|\Delta A\|$, although one usually expects that it does so, but it tends to allow us to obtain better error bounds for the solution of the scaled system, and from this bound we can obtain the error bound for the solution of the unscaled system. Up to now, no explicit realistic solution for scaling any matrix, such that partial pivot selection is numerically stable, is given in the literature.

The scaling scheme applied to the solution of the corrector equation should satisfy two conditions: (i) it should be cheap, not computational expensive, (ii) it should not depend on the right-hand side b because the iteration matrix A is factorized once and is reused for many iterations. Therefore, in the following we will focus only on scaling of the matrix A to reduce

its condition number. Instead of solving the linear system (2.51) directly, we solve

$$D_1 A D_2 x' = D_1 b, \quad x = D_2 x',$$

where D_1 and D_2 are the scaling matrices. The aim here is to determine D_1 and D_2 such that $\kappa(D_1 A D_2)$ is smaller than $\kappa(A)$. To save computing time, one often chooses diagonal matrices D_1 and D_2 to scale rows and columns of the matrix A, respectively. To avoid scaling roundoff error, integer powers of machine base are chosen for elements of D_1 and D_2. In fact, if a scaling number has such a form, then the mantissa of its floating-point representation is exactly 1., i.e., there arises no roundoff error when converting the original scaling number into its floating-point form. Moreover, the multiplication is faster because, to multiply a scaling number with a matrix entry, one has only to add two integers, namely the exponents of the scaling number and of the matrix entry. Alternatively, one can implement scaling *implicitly* (see, e.g., [109] and [110], p. 193), that means we do not really multiply the scale matrix but only choose pivoting elements based on the scaling matrices, so this will not introduce additional roundoff errors due to scaling. The disadvantage of this approach is that we need to modify the pivoting strategy in the existing factorization subroutine. Therefore, we choose to multiply the scaling matrices in advance before factorize it. With this approach we do not need to modify existing factorization code. We use functions recommended by the IEEE-754 standard for floating-point arithmetic [31], such as `scalb` for multiplying 2^n or `logb` for computing logarithm of base 2. These functions are efficiently implemented in most programming libraries. How to choose the two diagonal matrices D_1 and D_2? Here, D_1 scales the equations and D_2 scales the unknowns. With this scaling, the relative error bound of x' of the scaled system is $\|\widehat{x}' - x'\|/\|x'\| = \|D_2^{-1}(\widehat{x} - x)\|/\|D_2^{-1} x\|$. In other words, the relative error is being measured in a different norm.

In the context of BDF-methods, the solution of the linear system is used to check convergence of the corrector iteration to the solution of the corrector equations, and also is used to estimate the convergence ratio, etc. Because the components of the solution vector often have different magnitudes, a weighted norm (2.23) is used instead of the l_2-norm for error control or other purposes. As discussed in previous sections, we could be able to obtain the error bounds measured in the ∞-norm of the computed solution of a linear system, which is solved by direct Gaussian elimination (triangular decomposition) and forward and backward substitutions. However, as mentioned above we want to obtain the error bounds of the linear system measured in the scaled norm (2.23). Therefore, the column scaling diagonal matrix D_2 is chosen as

$$D_2 = \mathrm{diag}(2^{\alpha_1}, 2^{\alpha_2}, \ldots, 2^{\alpha_n}),\ \alpha_i = [\log_2 \mathrm{yscal}(i)],\ (i = 1, \ldots, n),$$

where $[a]$ is the integer closest to a. Then, we have

$$\begin{aligned}\frac{\sqrt{n}\|\widehat{x'} - x'\|_\infty}{\|x'\|_\infty} &\geq \frac{\|\widehat{x'} - x'\|_2}{\|x'\|_2} \\ &= \frac{\|D_2^{-1}(\widehat{x} - x)\|_2}{\|D_2^{-1}x\|_2} \\ &\approx \frac{\sqrt{n}\|\widehat{x} - x\|_{\text{WRMS}}}{\sqrt{n}\|x\|_{\text{WRMS}}} \\ &= \frac{\|\widehat{x} - x\|_{\text{WRMS}}}{\|x\|_{\text{WRMS}}}\end{aligned}$$

where $\widehat{x}$ and $\widehat{x'}$ are the computed solution of x and $x' = D_2^{-1}x$, respectively. It means that, after column scaling by D_2, the error bound of computed solution $\widehat{x'}$ of scaled system measured in the 2-norm is approximately the same as the one of computed solution $\widehat{x}$ of the original system measured in the scaled norm, and less than by a factor of $\sqrt{n}$ of the error bound of the scaled system measured in ∞-norm.

After column scaling, we need to find a diagonal matrix D_1 such that the condition number $\kappa(D_1\underline{A})$ is minimal, where $\underline{A} = AD_2$. As proved in [113], for the condition number defined as

$$\kappa^{(S)}(A) = \|A\|_\infty \|A^{-1}\|^*,$$

where $\|.\|^*$ is any Hölder norm or the Frobenius norm, or

$$\kappa^{(S)}(A) = \|A\|/\text{glb}_{\text{pq}}(A),$$

where $\text{glb}_{\text{pq}}(A) = \min_{x \neq 0} \|Ax\|_p/\|x\|_q$, the condition number $\kappa^{(S)}(D_1\underline{A})$ is minimal if all rows of the matrix $D_1\underline{A}$ have the same 1-norm. Thus, the diagonal matrix D_1 is chosen as follows

$$D_1 = \text{diag}(2^{\beta_1}, 2^{\beta_2}, \ldots, 2^{\beta_n}),\ \beta_i = -[\log_2(\sum_{j=1}^{n} |\underline{A}_{i,j}|)]^+ \ (i = 1, \ldots, n),$$

where $[a]^+$ denotes the integer part of a if $a \leq 0$ and the integer part of $a + 1$ if $a > 0$, e.g., $[3.1]^+ = 4$, $[-3.7]^+ = -3$. This is defined to follow the *round up* rule, alternatively the *round down* can also be used. The matrix obtained after scaling using D_1 and D_2 is nearly of row equilibrated.

Let $\mathcal{D}_n$ be the class of non-singular $n \times n$ diagonal matrices. Let D_1^* be defined by

$$D_1^* = \text{diag}(d_1^*, d_2^*, \ldots, d_n^*),\ d_i^* = 1/\sum_{j=1}^{n} |\underline{A}_{i,j}|,$$

then $\kappa(D_1^*\underline{A}) = \min_{\widetilde{D}\in\mathcal{D}_n} \kappa_\infty(\widetilde{D}\underline{A})$ because all rows of $D_1^*\underline{A}$ have equal 1-norm and equal 1, $\|(D_1^*\underline{A})_{i,.}\|_1 = 1$.

The row scaling matrix D_1 defined as above satisfies

$$\min_{\widetilde{D}\in\mathcal{D}_n} \kappa_\infty(\widetilde{D}\underline{A}) \le \kappa_\infty(D_1\underline{A}) < 2 \min_{\widetilde{D}\in\mathcal{D}_n} \kappa_\infty(\widetilde{D}\underline{A}). \tag{2.52}$$

The first inequality is obvious because $D_1 \in \mathcal{D}_n$. For the second inequality, we have

$$\begin{aligned}
\kappa_\infty(D_1\underline{A}) &= \|D_1\underline{A}\|_\infty \|(D_1\underline{A})^{-1}\|_\infty \\
&= d_i^{(1)} \sum_{k=1}^{n} |\underline{A}_{i,k}| {d_j^{(1)}}^{-1} \sum_{l=1}^{n} |\underline{A}_{j,l}^{-1}| \\
&< d_i^* \sum_{k=1}^{n} |\underline{A}_{i,k}| 2 {d_j^*}^{-1} \sum_{l=1}^{n} |\underline{A}_{j,l}^{-1}| \\
&\le 2\|D^*\underline{A}\|_\infty \|(\underline{A}D^*)^{-1}\|_\infty \\
&= 2\kappa_\infty(D^*\underline{A}).
\end{aligned}$$

As shown in Figures 2.2 and 2.3, the estimated condition numbers are reduced a lot after scaling, from in range 10^{14}–10^{18} to in range 10^6–10^8. Theoretically, one can expect about 5 reliable digits in the solution of the linear system.

Remark 2.7.1
Since we use integer powers of machine-base for scaling matrix to avoid round-off errors in multiplication of floating point numbers, the pivot elements with or without column scaling are the same. Hence, the column scaling with partial pivot does not affect the solution of the linear system. Nevertheless, column scaling helps to reduce the condition number in most cases, so that we can obtain a better error bound of the solution. On the other hand, the row scaling could affect the computed solution because it possibly changes the pivot elements, even for mildly ill-conditioned systems.

2.8 Automatic differentiation

The solution of the corrector equation in the BDF methods requires the partial derivatives of the model functions with respect to the state variables. Furthermore, later in Chapter 4, for generating the sensitivity equations for computing derivatives of the solution of the DAEs with respect to parameters we also need the partial derivatives of the model functions with respect to the state variables and parameters.

In general, there are four appoaches to computing derivatives:

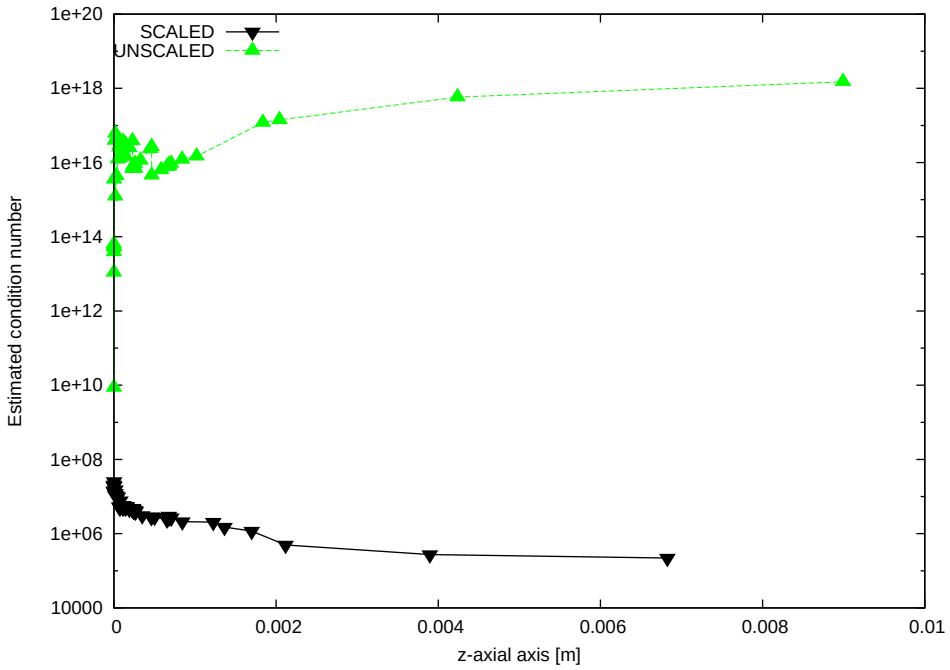

Figure 2.2: Estimated condition number of the iteration matrix of NO2 problem (30 grid points, RTOL=10^{-5}, ATOL=10^{-14})

- **By hand:** The derivatives are manually coded, in some cases this could be a very efficient method. However it is very difficult and tedious for coding derivatives of very complex functions, and errors in derivative code are difficult to avoid.

- **Divided differences:** The derivatives can be approximated by using finite differences. However, the main drawback is that it may lead to numerical cancellation and loss of many digits of accuracy. Usually the best number of precision one can expect is about a half of number of precision digits of the function of which the derivatives to be computed.

- **Symbolic Differentiation:** There are a number of symbolic manipulation packages such as Maples, Mathematica, Reduce, and Macsyma, which can generate exact derivatives for given the definition of a function. However, the generated derivatives are not very efficient to compute due to a lot of common subexpressions in the different derivative expressions (see, e.g., [13] and [63]) unless using very a good compiler which can optimize the derivative codes. It may run into resource limitations (i.e, out of memory) when the function description is complicated.

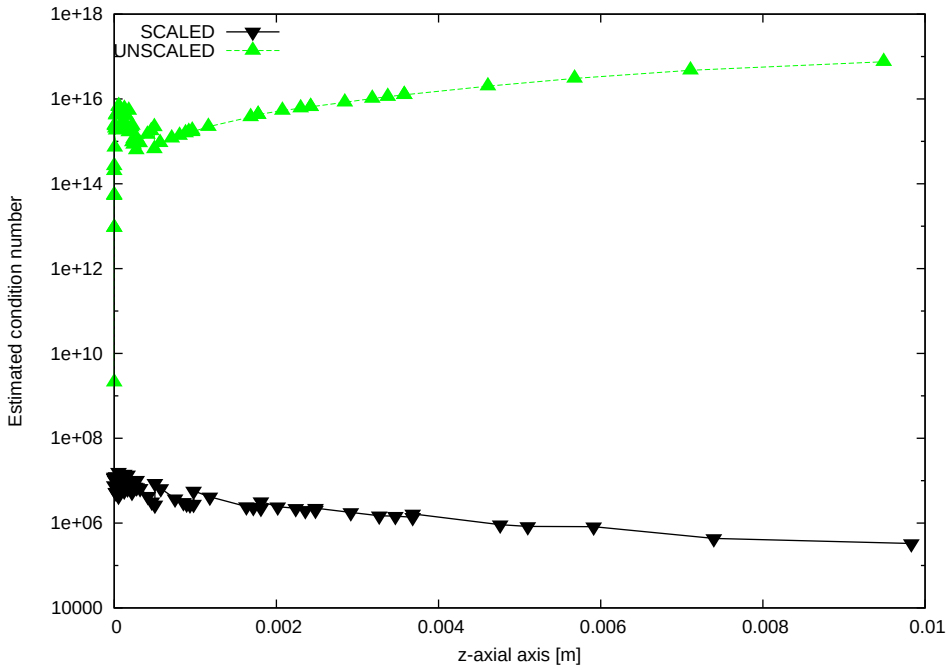

Figure 2.3: Estimated condition number of the iteration matrix of METHAN problem (12 grid points, RTOL=10^{-5}, ATOL=10^{-14})

- **Automatic differentiation:** The derivatives are computed based on the chain rule applied to the code of the function at elementary level. The computed derivatives are correct up to the machine precision [66]. We discuss about it further in the following.

Suppose that a scalar function $y = f(x)$, $x \in \mathbb{R}^n$ is defined by a sequential code which can be written as in the following form.

Function definition
For $\mathtt{i} = \mathtt{n}+1, \mathtt{n}+2, \ldots, \mathtt{m}$
$\quad \mathtt{x_i} = \mathtt{f_i}(\mathtt{x_j}), \quad \mathtt{j} \in \mathcal{J}_\mathtt{i}$
$\mathtt{y} = \mathtt{x_m}$

Here $\mathtt{f_i}$ are the elementary functions depending on the already computed quantities $\mathtt{x_j}$ and the index set

$$\mathcal{J}_\mathtt{i} = \{1, 2, \ldots, \mathtt{i}-1\}, \quad \mathtt{i} = \mathtt{n}+1, \mathtt{n}+2, \ldots, \mathtt{m}.$$

The gradient of $\mathtt{f_i}$

$$\nabla \mathtt{f_i} = \partial \mathtt{f_i} / \partial \mathtt{x_j} \quad \mathtt{j} \in \mathcal{J}_\mathtt{i}$$

are assumed to be computable, e.g, by the chain rule and the derivative of elemenary functions.

There are two approaches in the framework of automatic differentiation for computing derivatives: *forward mode* and *reverse mode*. The forward mode applied to computing derivative of the above function is as follows.

Derivative by forward mode

For $i = 1, 2, \ldots, n$

$\nabla x_i = e_i$

For $i = n + 1, n + 2, \ldots, m$

$x_i = f_i(x_j), \quad j \in \mathcal{J}_i$

$\nabla x_i = \sum_{j \in \mathcal{J}_i} \frac{\partial f_i}{\partial x_j} \nabla x_j$

$y = x_m$

$g = \nabla x_m$

The reverse mode derivative code for the above function is as follows

Derivative by reverse mode

For $i = n + 1, n + 2, \ldots, m$

$x_i = f_i(x_j), \quad j \in \mathcal{J}_i$

$\overline{x}_i = 0$

$y = x_m$

$\overline{x}_m = 1$

$(\overline{x}_i)_{i=1}^n = 0$

For $i = m, m - 1, \ldots, n + 1$

$\overline{x}_j = \overline{x}_j + \frac{\partial f_i}{\partial x_j} \overline{x}_i, \quad j \in \mathcal{J}_i$

$g = (\overline{x}_i)_{i=1}^n$

In the forward mode the derivatives of intermediate values with respect to the input variables are computed along with their values. Roughly speaking, the cost of computation of derivative by using forward mode is linear in the number of input variables. Thus, it is suitable for computation of derivatives of vector function where the number of components of the function vector is large compared to the number of input variables. On the other hand, in the reverse mode the derivative of the final outputs with respect to the intermediate values are computed. The cost of computation of derivative by using reverse mode is roughly linear in the number of components of the output function. Thus, it is efficient for computation of derivatives of a vector function when the number of components of the output function vector is much smaller than the number of input variables, e.g., gradient. However, in the reverse mode usually much more storage is required than

in the forward mode because in the reverse mode the intermediate values and their derivatives have to be stored. It is shown in [63] that *"under quite realistic assumptions the evaluation of a gradient requires never more than five times the effort of evaluating the underlying function by itself"*. For the references, see e.g., [63] and [64] and the references given there.

There are a number of software packages for computation of derivatives based on automatic differentiation, just to mention a few of them, e.g., ADIFOR [16] and [15], JAKEF [68], GRESS [69] and ADOL-C [65]. There are two main approaches for computations of derivatives by automatic differentiation. The first one is *source code translation* (also known as *precompiler*). The user is required to supply the source code of the function, e.g., in Fortran, then these tools parse the code into elementary operations and generate the derivative code. The derivative and source codes can be compiled and linked together into a single program. The software packages following this approach include ADIFOR, JAKEF, and GRESS. The second one is based on the *operator overloading*, which is available in some programming languages such as C++, where (for automatic differentiation) the floating-point operations are redefined (overloading) to include the associate derivatives. The source codes remain essentially unchanged except new data types for floating-point variables. The rest of the work is the job of the compiler. ADOL-C follows this approach.

Since our model would rather be complicated involving many functions in the DETCHEM [46] library for computations of chemical and physical quantities of a composition, which have been developed over years of research and experiences and coded in Fortran and it is still continuously extended, and thus we think that we should not try to re-implement all these codes into a new language such as C++, therefore for coding our model the Fortran language is used so that we can use ADIFOR to generate derivative codes. Moreover we use ADIFOR because it is easy to use in particular for complex models involving many functions from different modules and quite efficient.

2.9 Computation of the time derivatives at the initial point

As mentioned in Sections 3.3 and 3.4, our DAE system (DAE1) can be written as

$$\begin{aligned} B(t,x,y)\dot{x} &= f(t,x,y), \\ 0 &= g(t,x,y), \end{aligned} \tag{2.53}$$

where

$$\begin{aligned} B: &\quad [t_0, t_e] \times \mathbb{R}^{n_d} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_f \times n_d}, \\ f: &\quad [t_0, t_e] \times \mathbb{R}^{n_d} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_f}, \\ g: &\quad [t_0, t_e] \times \mathbb{R}^{n_d} \times \mathbb{R}^{n_a} \to \mathbb{R}^{n_g}. \end{aligned}$$

In addition, $n_d \geq n_f$ and B has full range. To solve (2.53), consistent initial values for x, y and $\dot{x}$ must be determined. The way to calculate $y(t_0)$ was discussed in the previous section. We now present how to compute $\dot{x}(t_0)$ and $\dot{y}(t_0)$, which is needed for predicting $x(t_0 + h)$ and $y(t_0 + h)$, where h is the step size.

An essential trouble is that our equation system is *structurally singular*. To calculate $\dot{x}(t_0)$ and $\dot{y}(t_0)$ when (2.53) is structurally singular and of *index 1*, we differentiate the algebraic constraint $g(t, x, y) = 0$ and obtain the equation system (applied at $t = t_0$)

$$\left[\begin{array}{c} B(t,x,y)\dot{x} - f(t,x,y) \\ \dfrac{\partial g}{\partial t} + \dfrac{\partial g}{\partial x}\dot{x} + \dfrac{\partial g}{\partial y}\dot{y} \end{array} \right]_{t=t_0} = 0 \tag{2.54}$$

with the unknown $\dot{x}(t_0)$ and $\dot{y}(t_0)$. Hence, we have to solve the linear system

$$\left[\begin{array}{cc} B(t,x,y) & 0 \\ \dfrac{\partial g}{\partial x} & \dfrac{\partial g}{\partial y} \end{array} \right]_{t=t_0} \left[\begin{array}{c} \dot{x}(t_0) \\ \dot{y}(t_0) \end{array} \right] = \left[\begin{array}{c} -f(t,x,y) \\ -\dfrac{\partial g}{\partial t} \end{array} \right]_{t=t_0}.$$

For DAE (2.53) with index 1, the matrix on the left is regular. Therefore, this linear equation system can be solved by standard methods.

2.10 Specially tailored methods for DAEs

Since the old DAESOL code [10] is designed for treating problem in the form

$$\begin{aligned} B(t,x,y,p)\dot{x} &= f(t,x,y,p), \\ 0 &= g(t,x,y,p), \end{aligned} \tag{2.55}$$

where $t \in [t_0, t_{\text{end}}]$ is "time" variable, $x \in \mathbb{R}^{n_d}$ is differential variable, $y \in \mathbb{R}^{n_a}$ is algebraic variables, $p \in \mathbb{R}^{n_p}$ is parameter, and $B : \mathbb{R} \times R^{n_d} \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_f \times n_d}$, $f : \mathbb{R} \times R^{n_d} \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_f}$, $g : \mathbb{R} \times R^{n_d} \times \mathbb{R}^{n_a} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_g}$, with non-negative integers, n_d, n_a, n_p, n_f, and n_g and assume that $n_f = n_d$, $n_a = n_g$, B and $\partial g/\partial y$ are non-singular. In words, the code is designed and implemented for solving problems with B to be non-singular square matrix and its number of rows equals the number of differential variables x, and the

number of algebraic variables y equals the number of components of g. To treat our problems, which the resulting DAEs also have the form as (2.55) but do not satisfy the assumptions, that is, the number of differential equations does not equal the number of differential variables, $n_f \neq n_{\rm d}$, and the number of algebraic variables does not equals the number of algebraic constraints, $n_{\rm a} \neq n_g$ and B is singular, we develop a new code DAESOLE (extended features of DAESOL), based on the old DAESOL code, which allows us to treat problems in a more general form, that is, the new code DAESOLE can treat problems which does not require $n_f = n_{\rm d}$, $n_{\rm a} = n_g$, and B non-singular; only need $n_f + n_g = n_{\rm d} + n_{\rm a}$. The new code is tailored to take the advantage of structured DAEs obtained from discretized PDE systems and treats some difficulties in solving the DAEs:

(i) realizing the band structure of the iteration matrix (included banded linear solver and derivative evaluations for banded or block tridiagonal iteration matrix), and

(ii) treating structurally singular DAEs,

(iii) automatic scaling of the linear system in the corrector iteration as discussed in Section 2.7.

Solving DAEs systems by the BDF methods, the computing time is mainly due to evaluation and factorization of the Jacobian matrix, and the solution of linear equation at each corrector iteration.

Our DAE system obtained from semi-discretization of PDEs has a band iteration matrix. In general, the bandwidth depends on the discretization scheme used, such as, the number of points for the finite differences and the order of the approximation. In our case, we only use at most three points for the approximation of the derivatives with respect to ψ, then the total bandwidth of the iteration matrix is $3 \times n_{\rm PDE}$, where $n_{\rm PDE} = N_{\rm g} + 4$ is the number of PDEs, $N_{\rm g}$ is the number of gas phase species. However, the upper and lower bandwidths are $2 \times n_{\rm PDE}$. Moreover, the iteration matrix is block diagonal one. (Note that in this thesis, the derivatives with respect to ψ and ψ-direction are also referred to as the *spatial derivatives* and the *spatial direction* respectively, and the derivatives with respect to the timelike coordinate z and z-direction are referred to as the *time derivatives* and the *time direction* respectively. **Nodes** denotes the number of disretization points (grid points) in the spatial direction, see Chapter 3 for more details).

It is well known that the number of operations to factor and to solve a band linear system is $\mathrm{O}(\mathrm{n_b} \times n^2)$ ($n_{\rm b}$ is the bandwidth of the matrix, which is usually less than n, $n_{\rm b} \ll n$, n is the dimension of the variables), while

for the dense linear system the number of operations is $O(n^3)$. To take into account the band structure of the iteration matrix, in DAESOLE a band linear solver is added in addition to the available ones, together with new options for the user to specify the upper and lower bandwidth.

To illustrate efficiency of the new approaches and the standard approach, we apply them to the following practical problems.

Example 2.10.1 (Simulation of NO_2 oxidation process)
*which is referred to as **NO2** problem. A gas mixture flows in a channel with the following setting.*

- *Channel geometry: the radius* $r_{\max} = 2.5 \times 10^{-4}$ *[m], the channel length* $z_{\max} = 0.01$ *[m].*
- *Initial conditions: at inlet, two species are present* $X_{NO_2} = 0.10$, $X_{N_2} = 0.90$, *the other species are absent at inlet. The initial gas temperature is* $T_{\text{gas}} = 300$ *[K], the initial pressure is* $p = 10^5$ *[Pa], and the initial velocity is* $u = 1$ *[m/s].*
- *Boundary conditions: the temperature at the wall is* $T_{\text{wall}} = 1200$ *[K].*
- *Reaction mechanisms: 5 gas-phase species, 4 surface species, 9 surface reactions, and 8 gas-phase reactions. The gas-phase reactions and surface reactions are given in the Appendix.*

Example 2.10.2 (Catalytic combustion of methane)
*which is referred to as **METHANE1** problem. A gas mixture flows in the channel with the following setting.*

- *Channel geometry: the radius* $r_{\max} = 2.5 \times 10^{-4}$ *[m], the channel length* $z_{\max} = 0.01$ *[m].*
- *Initial conditions: at inlet,three species are present* $X_{CH_4} = 0.3$, $X_{O_2} = 0.5$, $X_{N_2} = 0.2$, *the other species are absent at inlet. The initial gas temperature is* $T_{\text{gas}} = 298$ *[K], the initial pressure is* $p = 1.2 \times 10^5$ *[Pa], and the initial velocity is* $u = 1$ *[m/s].*
- *Boundary conditions: the temperature at the wall is* $T_{\text{wall}} = 1200$ *[K].*
- *Reaction mechanisms: 21 gas-phase species, 11 surface species, 23 surface reactions, and 128 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.*

Example 2.10.3 (Conversion of ethane to ethylene)
which is referred to as ***ETHANE*** *problem. A gas mixture flows in the channel with the following setting.*

- *Channel geometry: the radius* $r_{\max} = 2.5 \times 10^{-4}$ *[m], the channel length* $z_{\max} = 0.01$ *[m].*
- *Initial conditions: at inlet, three species are present* $X_{C_2H_6} = 0.16$, $X_{O_2} = 0.16$, *and* $X_{N_2} = 0.68$, *other species are absent at inlet. The initial gas temperature is* $T_{\text{gas}} = 300$ *[K], the initial pressure is* $p = 1.2 \times 10^5$ *[Pa], and the initial velocity is* $u = 1$ *[m/s].*
- *Boundary conditions: the temperature at the wall is* $T_{\text{wall}} = 973$ *[K].*
- *Reaction mechanisms: 25 gas-phase species, 20 surface species, 82 surface reactions, and 261 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.*

Note that in the above examples, the initial and boundary conditions are the ones given by the user, and these are used to specify the complete initial and boundary conditions for the numerical problem as discused in the previous sections. The three above examples in that order are increasing in the complexity ranging from a simple one with a few species and reactions to a complex one with many species and reactions. Increasing the number of reactions and species results in increasing the cost of evaluating the model functions.

In the following, all computations are performed on a Pentium 4, 2.6Ghz, Linux with Intel Fortran compiler, and computation using double precision. The integration error is controlled with the relative error tolerance RTOL $= 10^{-4}$ and the absolute error tolerance ATOL $= 10^{-12}$. **FD** and **AD** are the abbreviations for Finite Differences and Automatic Differentiation, respectively. LA is the abbreviation for linear solver.

Tables 2.2 and 2.3 show that the computing time is reduced if the band linear solver (in the BDF code) is used. One would expect that when the number of nodes (number of grid points) increases, the difference between computing time with using the dense solver and the banded solver is increased. Here, we only see litle improvement in performance, about 5 percents, due to the fact that the number of nodes is not high enough such that $n_{\text{b}} \ll n$ for which the band solver is more efficient than the dense solver.

Solving the nonlinear equation system (2.16) using Newton-like methods requires the partial derivatives

$$\frac{\partial f}{\partial y}, \ \frac{\partial g}{\partial y}, \text{ and } \frac{\partial B}{\partial y} v.$$

Nodes	30	50	70	90
Standard linear solver (secs)	6.4	28.7	79.4	178.3
Banded linear solver (secs)	5.9	27.9	73.2	152.9

Table 2.2: Timings of NO2 problem with the standard linear solver and banded linear solver.

Nodes	12	20	28	36
Standard linear solver (secs)	13.6	73.4	216.6	440.5
Banded linear solver (secs)	12.9	70.4	210.5	418.4

Table 2.3: Timings of METHANE1 problem with the standard linear solver and banded linear solver.

In general, the derivative of a function $f(y) : \mathbb{R}^n \to \mathbb{R}^n$ with respect to y, $J = \partial f/\partial y \in \mathbb{R}^{n\times n}$, can be computed by the finite differences. The ith column of J can be estimated by the forward finite difference as

$$J_{.i} = Je_i = \frac{\partial f(y)}{\partial y_i} = \frac{f(y + \eta e_i) - f(y)}{\eta}, \tag{2.56}$$

where e_i is the ith canonical unit vector, i.e., an n-vector of all zeros except for an entry of 1 in the ith position, and η is a positive increment. In this case we need $n + 1$ function evaluations (evaluation of f) to determine the full Jacobian J. However, it is well known that the number of function evaluations can be reduced if the Jacobian is sparse. Curtis, Powell and Reid [37] propose a method using finite differences to compute sparse Jacobian efficiently. The key idea is to identify *structurally orthogonal* columns of J, i.e., columns whose inner product is zero, independent of the values of y, in other words, these columns do not have non-zeros in the same row position. Instead of computing the full Jacobian directly, we compute a compressed Jacobian, whose the number of columns is usually less than n, by taking into account of the sparsity, then the full Jacobian is extracted from the compressed Jacobian. The structurally orthogonal property of columns allows that these columns can be computed by only using one directional derivative. For example, if the columns 1-, 3- and 6-th are structurally orthogonal, then

$$J(e_1 + e_3 + e_6) = \frac{f(y + \eta(e_1 + e_3 + e_6)) - f(y)}{\eta}$$

and the columns 1, 3 and 6 can be extracted from product $J(e_1 + e_3 + e_6)$. Generally, to compute the full sparse Jacobian J, the columns of J are

Nodes	30	50	70	90
Standard FD(secs)	6.4	28.7	79.4	178.3
Band FD (secs)	2.4	5.6	11.8	20.5
Blk.Trid. FD (secs)	1.8	5.3	11.3	19.8

Table 2.4: Timings of NO2 problem with the standard FD, band FD and block tridiagonal FD (FD denotes Finite Differences)

partitioned into groups, such that columns in the same group are structurally orthogonal, this is called a *consistent partition* of J. A consistent partition is said to be optimal if the number of its group is minimal. The number of groups is the number of function evaluations needed to determine the Jacobian, in addition to one function evaluation at y, $f(y)$. To determine an optimal consistent partition of J, the algorithm in [37] scans J column by column. Each new column is checked if it can be put in one of the available groups (if it passes the structurally orthogonal check), otherwise it is put in a new created group. This is a greedy algorithm, and it does not ensure to generate an optimal partition of J. The problem of determining an optimal consistent partition of J was proved to be equivalent to the coloring problem of a graph using minimum number of colors [33], where each vertex of the graph corresponds to each column of J and there is one edge between two vertexes if the corresponding columns of the vertexes have nonzeros in the same row position. The graph coloring problem is NP-hard. Therefore, to determine an optimal consistent partition of J, a heuristic algorithm is usually used. Fortunately, for a band matrix J with the total bandwidth n_{b}, the column partition generated by the algorithm in [37] is optimal, and the number of groups generated is n_{b}.

Based on the idea in [37], a general finite difference approximation for the model functions with band iteration matrices is developed and coupled in DAESOLE. This allows to treat problems with the iteration matrix having band structure, given the upper and lower bandwidth. For our problem, the number of function evaluations is $4 \times n_{\mathrm{PDE}}$ with $n_{\mathrm{PDE}} = N_{\mathrm{g}} + 4$.

Tables 2.4 and 2.5 show the computing times of the simulation code with three different options (dense FD, band FD, and block tridiagonal FD). The block tridiagonal and band FD options are few times faster than the dense FD option. This is due to a large amount of time for model function calls, for computing the Jacobian by FD, in the dense case. On the other hand, only a fixed number of model function calls for the block tridiagonal and band FD (independent of the number of grid points) is needed for approximation the Jacobian. The results shown in Tables 2.4 and 2.5 also reflect the prediction

Nodes	12	20	28	36
Standard FD (secs)	13.62	73.43	216.66	440.55
Band FD (secs)	5.66	16.48	33.19	56.11
Blk.Tri. FD (secs)	4.724	13.892	27.845	47.960

Table 2.5: Timings of METHANE1 problem with the standard FD, band FD and block tridiagonal FD.

that when the number of grid points is increased, the computing time for the dense case is much greater then for the block tridiagonal and band FD cases.

Our iteration matrices are not only band matrices, actually they are block tridiagonal matrices. Therefore, we further develop another general finite difference approximation for derivatives of model functions for block tridiagonal matrices, that only need $3 \times n_{\mathrm{PDE}}$ number of function evaluations for a full Jacobian instead of $4 \times n_{\mathrm{PDE}}$ as in band cases.

Although the above approaches have improved the performance of the simulation software, we also investigate a new technique for computation of derivatives, namely *automatic differentiation*, which allows evaluation of derivatives with accuracy up to the machine precision. In particular, we employ the automatic differentiation tool ADIFOR [14], which can be used for generating the derivative Fortran code from the Fortran source code of a function. To compute the derivative J of a function $f(y) \in \mathbb{R}^n$ using ADIFOR, a so-called seed matrix $S \in \mathbb{R}^{n \times n}$ needs to be supplied. For computing a full Jacobian matrix, the seed matrix S is the identity matrix. Indeed, ADIFOR computes directional derivative of f, where the directional vectors are specified in the seed matrix S. The number of columns of S is the number of directional derivatives to be computed. The result returned from ADIFOR is the transpose of the Jacobian times the seed matrix, $(J \cdot S)^T$. The number of operations in the derivative code for one directional derivative is about from 2 to 5 times the number of operations for the function evaluation. Therefore, for computation of a full Jacobian, we need about $O(n_y \times n_{of})$ operations, where n_{of} is the number of operations for evaluation of f. To use ADIFOR with DAESOLE, a subroutine for computing required derivatives, which calls ADIFOR generated subroutines, is provided.

In Tables 2.6, 2.7 and 2.10 computational statistics of the simulation code applying to NO2, METHANE1 and ETHANE problems (for increasingly the number of grid points) with standard FD and AD mode are summarized. Here, the second column, named **Nodes & (#DAE)**, is the number of discretization points in the spatial direction and the number of resulting DAEs, the third column, named **Time**, is the total CPU time for one simulation

run, and the fourth column, named **#model funcs. calls**, is the number of the model function calls (included B, f, and g), and the fifth column, named **Time for derivs. calls**, is the total time for compution of derivatives (of model functions w.r.t. the state variables).

It also shows that the time for computing the derivatives generated by ADIFOR is much smaller than the time for computing the derivatives by the finite differences. It should not be surprised as one would expect that computing derivative by the derivative codes is theoretically more expensive than by the finite differences. But for computing a full Jacobian, in the forward finite differences case we need $n+1$ function evaluation calls, in the automatic differentiation case, we only need one derivative code call although the derivative code contains a loop to compute derivative of all directions but many terms are being shared between the directional derivatives.

	Nodes & (#DAE)	Time (secs)	#model funcs. calls	Time for derivs. calls (secs)
Dense FD	30 (265)	6.4	3226	4.6
	50 (445)	28.7	5506	23.4
	70 (625)	79.4	7549	65.5
	90 (805)	178.3	10346	153.3
Dense AD	30 (265)	2.2	590	0.3
	50 (445)	6.3	622	0.6
	70 (625)	14.4	671	1.5
	90 (805)	23.7	709	2.7

Table 2.6: Computational statistics of simulation of NO2 problem using dense LA with dense FD and AD.

Similarly to the finite difference approach, the sparsity of the Jacobian can also be exploited for automatic differentiation (see e.g., [14] and [48]). If the sparsity structure of the Jacobian is known in advance, one can apply the CPR algorithm [37] for determining the seed matrix based on a consistent partition of J. The number of directional derivatives is n_{b}, where n_{b} is the total bandwidth of the Jacobian, which is independent of the number of discretization points in ψ. In our case $n_{\mathrm{b}} = 3 \times n_{\mathrm{PDE}}$ with $n_{\mathrm{PDE}} = N_{\mathrm{g}} + 4$. Using this seed matrix, one can compute a compressed Jacobian from the derivative code generated by ADIFOR. The number of operations for computing the compressed Jacobian is about $O(n_{\mathrm{b}} \times n_{of})$ where n_{of} is the number of operations for computing f, instead of $O(n \times n_{of})$ as in the standard approach. Usually, $n = (n_{\mathrm{dis}} - 1) \times n_{\mathrm{PDE}}$ (or $n = n_{\mathrm{dis}} \times n_{\mathrm{PDE}}$ depending on a particular implementation) where n_{dis} is the number of discretization points

	Nodes & (#DAE)	Time (secs)	#model funcs. calls	Time for derivs. calls (secs)
Dense FD	12 (286)	13.6	4476	11.2
	20 (486)	73.4	8805	66.0
	28 (686)	216.6	13604	204.1
	36 (886)	440.5	16567	441.2
Dense AD	12 (286)	3.5	453	1.2
	20 (486)	12.3	550	4.6
	28 (686)	27.4	580	9.8
	36 (886)	46.6	616	15.1

Table 2.7: Computational statistics of simulation of METHANE1 problem using dense LA with dense FD and AD.

	Nodes & (#DAE)	Time (secs)	#model funcs. calls	Time for derivs. calls (secs)
Blk. Trid. FD	30 (265)	1.8	839	0.46
	50 (445)	5.3	908	1.35
	70 (625)	11.3	971	2.57
	90 (805)	19.8	1010	4.72
Blk. Trid. AD	30 (265)	1.4	453	0.05
	50 (445)	4.2	550	0.11
	70 (625)	8.5	580	0.18
	90 (805)	15.6	616	0.33

Table 2.8: Computational statistics of simulation of NO2 problem using band LA with block tridiagonal FD and AD.

in the spatial direction, but for our problem we have additional N_s algebraic constraints at the wall, thus $n = (n_{dis} - 1) \times n_{PDE} + N_s$. Tables 2.8, 2.9, and 2.11 present the total computing times, the numbers of model functions calls and the total times for computing derivatives. These are the best results we can obtain. For NO2 problem with 90 grid points, 805 DAEs, it take only 19.8 seconds for the block diagonal FD and 15.6 seconds for the block diagonal AD, about 10 times faster than the standard FD mode 178.3 seconds; and for METHANE1 problem with 36 grid points, 886 DAEs, it takes only 24.3 seconds, about 18 times faster than the standard approach, which takes about 440.5 seconds. Figures 2.4 and 2.5 summarized the performance measure of the simulation code with different options for NO2 and METHANE1

	Nodes & (#DAE)	Time (secs)	#model funcs. calls	Time for derivs. calls (secs)
Blk.	12 (286)	4.9	1516	2.7
Trid.	20 (486)	14.4	1818	8.1
FD	28 (686)	29.1	1997	16.7
	36 (886)	50.1	2039	26.3
Blk.	12 (286)	2.4	453	0.3
Trid.	20 (486)	7.5	550	0.8
AD	28 (686)	14.2	580	1.4
	36 (886)	24.3	616	1.7

Table 2.9: Computational statistics of simulation of METHANE1 problem using band LA with block tridiagonal FD and AD.

	Nodes & (#DAE)	Time (secs)	#model funcs. calls	Time for derivs. calls (secs)
	12 (339)	92.15	18354	80.28
Dense	18 (513)	318.20	27748	291.13
FD	24 (687)	678.46	38242	627.58
	30 (861)	1231.31	45916	1147.48
	12 (339)	20.78	1079	8.96
Dense	18 (513)	47.76	1116	20.20
AD	24 (687)	81.09	1119	34.50
	30 (861)	138.10	1154	55.05

Table 2.10: Computational statistics of simulation of ETHANE problem using dense LA with dense FD and AD.

problems. In these figures, the three upper case letters are represented for a chosen combined option. The first letter represents the type of linear solver: **B** for band, **D** for dense; the second letter represents the method for computing derivatives: **F** for finite differences, **A** for automatic differentiation; the last letter represents the computing mode of the derivatives: **D** for dense mode, **B** for band mode, and **T** for block tridiagonal mode. For example, **BFT** represents the case using the band linear solver, with finite differences, and the block tridiagonal property are exploited.

Let us define the **Speedup** to be the ratio between the CPU time for solving a problem using the standard method in DAESOLE (finite differences for computation of derivatives) which is named as DFD, and the CPU time

	Nodes & (#DAE)	Time (secs)	#model funcs. calls	Time for derivs. calls (secs)
Blk.	12 (339)	28.77	5599	18.66
Trid.	18 (513)	59.50	5447	37.92
FD	24 (687)	88.89	5478	53.39
	30 (861)	143.78	5779	82.05
Blk.	12 (339)	12.71	1086	2.67
Trid.	18 (513)	24.72	1070	4.32
AD	24 (687)	40.36	1135	5.99
	30 (861)	63.88	1126	7.12

Table 2.11: Computational statistics of simulation of ETHANE problem using band LA with block tridiagonal FD and AD.

for solving the problem by an other method. Table 2.12 shows the speed up gained by different methods applied to three applications: the NO2, which is a small size problem (5 gas species, 4 surface species, 9 surface reactions, and 8 gas phase reactions); the METHANE1, which is a medium size problem (21 gas species, 11 surface species, 2128 gas reactions, 23 surface reactions); the ETHANE, which is a large size problem (25 gas species, 20 surface species, 261 gas phase reactions, 82 surface reactions).

2.11 Summary

In this chapter we have discussed numerical methods for differential algebraic equations (DAEs). The BDF methods are used for discretizing the DAEs leading to a system of nonlinear equations at each integration step. The nonlinear equations are solved by a modified Newton method. An automatic scaling method is proposed to scale the linear equations arising in the Newton iteration. The scaling reduces the condition numbers of the linear equations from a range $[10^{14}$–$10^{18}]$, that nearly equal the reciprocal of the machine precision and this do not allow us to obtain an solution of the linear systems with a few significant digits, to a range $[10^{6}$–$10^{8}]$, that allow us to obtain a solution with a few significant digits. This makes the estimation of errors of the solutions of DAEs more reliable. Tailored methods in particular for structured DAEs are presented. Efficient methods for computation of derivatives required for solving the nonlinear equations are described. We exploit the structure of the derivative matrices, which are of block diagonal ones, and derive methods for computation of derivatives in

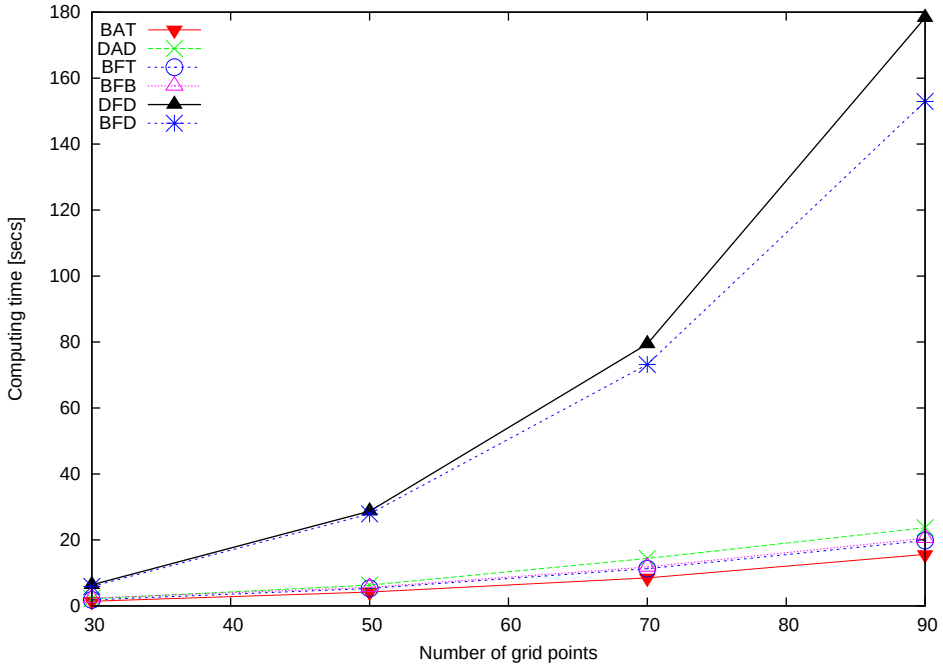

Figure 2.4: Total CPU times for solving NO2 problem using FD and AD (see page 83 for the notations)

Prolem	Nodes & (#DAE)	BAT	DAD	BFT
NO2 (small)	30 (265)	4.57	2.90	3.55
	50 (445)	6.83	4.55	5.41
	70 (625)	9.34	5.51	7.02
	90 (805)	11.42	7.52	9.00
METHANE1 (medium)	12 (286)	5.66	3.88	2.77
	20 (486)	9.78	5.96	5.09
	28 (686)	15.25	7.90	7.44
	36 (886)	28.23	9.45	8.79
ETHANE (large)	12 (339)	7.21	4.43	3.20
	18 (513)	12.87	6.66	5.34
	24 (687)	16.81	8.28	7.63
	30 (861)	19.27	8.91	8.56

Table 2.12: Speedup gained by different methods

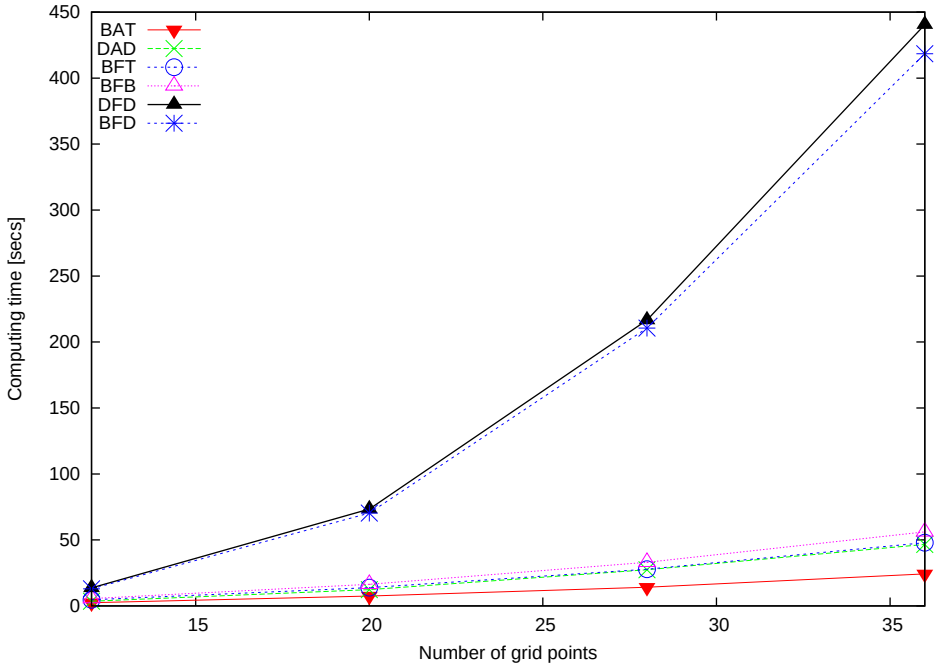

Figure 2.5: Total CPU times for solving METHANE1 problem using FD and AD (see page 83 for the notations)

frameworks of finite differences and automatic differentiation. As a result speedups by a factor of five to more then ten, depending on the applications, are obtained. For example, for simulation of catalytic combustion of methane problem, named METHANE1, using 36 spatial discretization points, the resulting speed up is 28.23. The obtained results also show that for solution of DAE systems computation of the derivatives by automatic differentiation, here is ADIFOR, always outperforms computation of the derivatives by the finite differences.

Chapter 3

Numerical Methods for Simulation

As mentioned in Chapter 1 we use the boundary layer equations as our mathematical model. In this chapter, we will discuss methods to approximate the partial differential equations (PDEs) by a system of a differential algebraic equations (DAEs). In particular, we apply the *von Mises* transformation (see e.g., [104], [35], [92], and [73]), which eliminates the overall mass continuity equation and replaces it with an integral. The elimination of the overall mass continuity equation is particularly important because it allows us to avoid some difficulties in the numerical computation arising when directly discretizing the overall mass continuity equation, which is a first-order one and is of different form than other equations (except the simple radial momentum equation), which are second-order ones. In addition, all the radial convective terms along with the radial velocity v are also eliminated. The obtained system of PDEs are then semi-discretized in the stream direction ψ by the *method of lines* [103] with non-uniform grid, which leads to large stiff structured DAEs. The numerical treatment for the DAEs is discussed, in particular, the computation of consistent initial values, which partially arise from the nonlinear boundary conditions, are considered. Some important properties of the DAEs, such as the index and structural properties, are also investigated. In the following, these topics are discussed in detail.

3.1 Von Mises transformation

We define the stream function ψ from the following relations

$$\rho u r = \frac{\partial \psi}{\partial r}, \quad \rho v r = -\frac{\partial \psi}{\partial z}, \tag{3.1}$$

then the overall mass continuity equation (1.49) is always fulfilled. It means that this equation is eliminated.

Now we want to transform the boundary layer equations written in coordinate system (z, r) to the coordinate system (z, ψ). Let consider a transformation from an (x, y) coordinate system to a (η, ξ) system, some dependent variables $f(x, y)$ can be written as

$$f(x, y) = f(\eta(x, y), \xi(x, y)).$$

Applying the chain-rule of differentiation, we obtain

$$\begin{aligned}
\frac{\partial f(x,y)}{\partial x} &= \frac{\partial f(x,y)}{\partial \eta}\frac{\partial \eta(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial \xi}\frac{\partial \xi(x,y)}{\partial x} \\
\frac{\partial f(x,y)}{\partial y} &= \frac{\partial f(x,y)}{\partial \eta}\frac{\partial \eta(x,y)}{\partial y} + \frac{\partial f(x,y)}{\partial \xi}\frac{\partial \xi(x,y)}{\partial y}.
\end{aligned}$$

Thus the differential operators are transformed as

$$\begin{aligned}
\frac{\partial}{\partial x} &= \frac{\partial \eta(x,y)}{\partial x}\frac{\partial}{\partial \eta} + \frac{\partial \xi(x,y)}{\partial x}\frac{\partial}{\partial \xi} \\
\frac{\partial}{\partial y} &= \frac{\partial \eta(x,y)}{\partial y}\frac{\partial}{\partial \eta} + \frac{\partial \xi(x,y)}{\partial y}\frac{\partial}{\partial \xi},
\end{aligned}$$

and the matrix

$$\begin{bmatrix}
\dfrac{\partial \eta(x,y)}{\partial x} & \dfrac{\partial \xi(x,y)}{\partial x} \\
\dfrac{\partial \eta(x,y)}{\partial y} & \dfrac{\partial \xi(x,y)}{\partial y}
\end{bmatrix}$$

is called the *coordinate transformation matrix*. For the von Mises transformation, where z coordinate is unchanged, the coefficients of the coordinate transformation matrix are

$$\begin{aligned}
\frac{\partial z}{\partial z} = 1, \quad \frac{\partial \psi}{\partial z} &= -\rho v r \\
\frac{\partial z}{\partial r} = 0, \quad \frac{\partial \psi}{\partial r} &= \rho u r.
\end{aligned}$$

Hence, the differential operators become

$$\left(\frac{\partial}{\partial z}\right)_r = \left(\frac{\partial}{\partial z}\right)_\psi - \rho v r \left(\frac{\partial}{\partial \psi}\right)_z \tag{3.2}$$

$$\left(\frac{\partial}{\partial r}\right)_z = \rho u r \left(\frac{\partial}{\partial \psi}\right)_x. \tag{3.3}$$

Note that $(\partial r/\partial z)_\psi = v/u$, it means that the velocity vector is parallel to lines of constant ψ.

By applying (3.2) and (3.3), the boundary layer equations, which are now written in the new coordinate (z, ψ) instead of (z, r), become as follows.
Momentum:

$$\rho u \frac{\partial u}{\partial z} + \frac{\partial p}{dz} = \rho u \frac{\partial}{\partial \psi}\left(\rho u \mu r^2 \frac{\partial u}{\partial \psi}\right), \tag{3.4}$$

$$\frac{\partial p}{\partial \psi} = 0. \tag{3.5}$$

Species:

$$\rho u \frac{\partial Y_k}{\partial z} = \dot{\omega}_k W_k - \rho u \frac{\partial}{\partial \psi}(r J_{k,r}), \quad (k = 1, \ldots, N_\mathrm{g}). \tag{3.6}$$

Energy:

$$\rho u c_p \frac{\partial T}{\partial z} = \rho u \frac{\partial}{\partial \psi}\left(\rho u \lambda r^2 \frac{\partial T}{\partial \psi}\right) - \sum_{k=1}^{N_\mathrm{g}} \dot{\omega}_k W_k h_k - \rho u r \sum_{k=1}^{N_\mathrm{g}} J_{k,r} c_{pk} \frac{\partial T}{\partial \psi}. \tag{3.7}$$

State:

$$p = \frac{\rho R T}{\overline{W}}. \tag{3.8}$$

The radial diffusion mass flux $J_{k,r}$ is given in the new coordinates by

$$J_{k,r} = -D_k \frac{W_k}{\overline{W}} \rho^2 u r \frac{\partial X_k}{\partial \psi} - D_k^T \frac{\rho u r}{T} \frac{\partial T}{\partial \psi}. \tag{3.9}$$

In these equations, the independent variables z and ψ represent the axial coordinate and the stream function, respectively. The radial coordinate r is a dependent variable and is given in terms of the stream function by integrating the first of equations (3.1) as follows

$$\frac{r^2}{2} = \int_0^\psi \frac{d\psi'}{\rho u}$$

or, in the differential equation form by

$$\frac{\partial r^2}{\partial \psi} = \frac{2}{\rho u}. \tag{3.10}$$

The dependent variables in the above equation system are

- the axial velocity $u(z, \psi)$,
- the temperature $T(z, \psi)$,
- the pressure $p(z, \psi)$,
- the mass fraction $Y_k(z, \psi)$, $k = 1, \ldots, N_{\mathrm{g}}$,
- the radial coordinate $r(z, \psi)$, which in the cross-stream coordinate (z, r) is an independent variable,
- the surface coverage $\Theta_i(z)$, $i = 1, \ldots, N_{\mathrm{s}}$, which apear in the boundary conditions and are mentioned in the following sections.

Other quantities and terms, such as the mass density ρ, the mixture specific heat c_p, etc., are treated as functions of the above dependent variables. For example, when we want to compute the mass density ρ, we use the relation (3.8) to compute ρ from T, p, and $\overline{W}$, which is also a function of Y_k.

Note that the equations (3.4), (3.6) and (3.7) have parabolic characteristic with the axial coordinate z being the timelike direction. After semi-discretization in the ψ direction, these equations will become differential equations. The other equations (3.5) and (3.10) are considered as algebraic constraints.

3.2 Initial and boundary conditions

The boundary conditions, which was stated in Section 1.7, have to be transfered to new coordinate accordingly.

3.2.1 Initial conditions

At the inlet, the entrance of the channel, the initial profiles of u, T_{gas}, Y_k, p, T_{wall} must be specified.

3.2.2 Boundary conditions

At the centerline of cylinder, the symmetric property of cylinder is used to determine the boundary conditions

$$\begin{aligned} r(z, 0) &= 0 \\ \frac{\partial u}{\partial \psi} &= \frac{\partial T}{\partial \psi} = \frac{\partial Y_k}{\partial \psi} = 0. \end{aligned}$$

If the wall is not a catalytic surface, the condition for mass fraction the wall is

$$\frac{\partial Y_k}{\partial \psi} = 0.$$

The boundary conditions (1.55) for the catalytic wall is repeated in the following for convenience

$$\dot{s}_k W_k = -J_{k,r} \quad (k = 1, \ldots, N_{\mathrm{g}}), \tag{3.11}$$

where $\dot{s}_k$ is the rate of creation/depletion of the kth gas phase species by the surface reactions.

The boundary condition for axial velocity u at the wall is $u = 0$ (no-slip condition).

At the steady state, the surface coverage fractions Θ_i do not depend on time (as in Section 1.7).

$$\frac{\partial \Theta_i}{\partial t} = \frac{\dot{s}_i \sigma_i}{\Gamma} = 0 \quad (i = N_{\mathrm{g}} + 1, \ldots, N_{\mathrm{g}} + N_{\mathrm{s}}), \tag{3.12}$$

where Γ is total available site density, N_{s} is the number of surface species, Θ_i is surface coverage fraction.

The boundary condition for the temperature T at the wall depends on a specific problem, as mentioned in Section 1.7. For isotherm reactors,

$$T(z) = T_{\mathrm{wall}}(z)$$

or adiabatic cases

$$-\lambda \rho u r \frac{\partial T}{\partial \psi} = -\lambda_s \rho u r \frac{\partial T}{\partial \psi} + \sum_{k=1}^{N_{\mathrm{g}}} \dot{s}_k W_k h_k. \tag{3.13}$$

At the wall we have the following boundary condition for r:

$$r(z, \psi_{\max}) = r_{\max},$$

where $\psi_{\max}$ is defined as

$$\psi_{\max} = \int_0^{r_{\max}} \rho_0 u_0 r dr.$$

Here u_0, ρ_0 are at the inlet conditions, and $r_{\max}$ is the channel radius.

3.3 Semi-discretization

Using subscript for denoting partial derivatives and the abbreviations

$$\mathcal{E} = \begin{bmatrix} \rho u u_z + p_z \\ 0 \\ \rho u c_p T_z \\ 0 \\ \rho u Y_{1z} \\ \vdots \\ \rho u Y_{N_g z} \end{bmatrix}, \; \mathcal{F} = \begin{bmatrix} \rho u(\rho u \mu r^2 u_\psi)_\psi \\ p_\psi \\ \rho u(\rho u \lambda r^2 T_\psi)_\psi - \sum_{k=1}^{N_g} \dot{\omega}_k W_k h_k - \rho u r \sum_{k=1}^{N_g} J_{k,r} c_{pk} T_\psi \\ \dfrac{\partial r^2}{\partial \psi} - \dfrac{2}{\rho u} \\ \dot{\omega}_1 W_1 - \rho u (r J_{1,r})_\psi \\ \vdots \\ \dot{\omega}_{N_g} W_{N_g} - \rho u (r J_{N_g,r})_\psi \end{bmatrix},$$

equations (3.4)- (3.7) and (3.10) can be summarized to the following system

$$\mathcal{E} = \mathcal{F}, \tag{3.14}$$

which forms, along with (3.8), (3.11) and (3.12), our entire mathematical model.

The solution of the partial differential equations is functions of the axial coordinate z and the stream coordinate ψ, i.e., $u = u(z, \psi)$, $T = T(z, \psi)$, $p = p(z, \psi)$, $r = r(z, \psi)$, and $Y_k = Y_k(z, \psi)$.

A standard numerical procedure for solving the PDEs is to determine the values of these functional quantities at certain discrete points (z_j, ψ_i). The whole domain $\{(z, \psi) : 0 \leq z \leq z_{max},\ 0 \leq \psi \leq \psi_{max}\}$ is partitioned by a mesh, consisting of grid points. The partial derivatives in these equations are replaced by algebraic approximations evaluated at these grid points. This process leads to a system of linear algebraic equations that approximate the system of partial differential equations. The system of algebraic equations can be solved by using any standard linear equation solver to obtain an approximate numerical solution of the PDEs. This procedure is the basis for the well-known classical finite difference, finite element and finite volume methods for PDEs.

In order to take the advantage of available strong DAE solvers, such as DAESOL [10], that have a variable order and variable step size control, we use the *method of lines* that has some differences from the above standard procedure. Instead of discretizing both in the space and time directions (for time-dependent problems), we only discretize in the spatial directions. Here, we have two spatial independent variables z and ψ but do not have the independent variable "time" as usual. The axial direction z is now treated

as the time-like direction. As mentioned in Chapter 2 and repeated here for convenience: in this thesis the derivatives with respect to ψ and ψ-direction are referred to as the *spatial derivatives* and the *spatial direction* respectively, and the derivatives with respect to the timelike coordinate z and z-direction are referred to as the *time derivatives* and the *time direction* respectively.

The spatial domain $\{\psi : 0 \leq \psi \leq \psi_{\max}\}$ is discretized by an appropriate grid

$$\psi_1 = 0 < \psi_2 < \psi_3 < \cdots < \psi_N = \psi_{\max}.$$

The distance between two adjacent points $\Delta\psi_i = \psi_{i+1} - \psi_i$ may be the same for all $i = 1, 2, \ldots, N-1$: $\Delta\psi_i = \Delta\psi$, then we have a uniform grid discretization; or may be different, then we have a non-uniform grid discretization. We replace the spatial derivatives (partial derivatives with respect to ψ) in the PDEs by appropriate finite-difference approximations. Each dependent variable in the PDE is replaced by N dependent variables at each grid point. For example, $u(z, \psi)$ is replaced by $u_i(z)$, $i = 1, \ldots, N$. After this step, the PDEs are semi-discretized in the spatial direction ψ. This leads to a system of differential-algebraic equation (DAEs), of which the number of dependent variables equals the number of grid point times the number of dependent variables of the PDE, and the number of equations equals the number of grid point times the number of PDEs.

Let us denote the function section corresponding to $\psi = \psi_i$ by the subscript i. For instance,

$$u_i = u_i(z) = u(z, \psi_i).$$

This rule is also applied to partial derivatives, e.g.,

$$u_{\psi_i} = u_{\psi_i}(z) = \left.\frac{\partial u(z, \psi)}{\partial \psi}\right|_{\psi=\psi_i}$$

and other quantities, such as temperature T, pressure p, radial coordinate r, and mass fraction Y_k.

Let $\mathcal{A} = (A_{j,k})$ be the matrix defined by

$$A_{j,k} = \begin{cases} \rho u, & \text{if } j = k = 1 \text{ and } 5 \leq j = k \leq N_{\mathrm{g}} \\ 1, & \text{if } j = 1,\ k = 2 \\ \rho u c_p, & \text{if } j = k = 3 \\ 0, & \text{otherwise,} \end{cases}$$

and let

$$\mathcal{Q} = \left[u, p, T, r, Y_1, Y_2, \ldots, Y_{N_{\mathrm{g}}}\right].$$

Then we have

$$\mathcal{E} = \mathcal{A}\,\mathcal{Q}_z^T. \tag{3.15}$$

By our convention,

$$\mathcal{E}_i = \mathcal{E}|_{\psi=\psi_i}\,,\ \mathcal{A}_i = \mathcal{A}|_{\psi=\psi_i}\,,\ \mathcal{Q}_i = \mathcal{Q}|_{\psi=\psi_i}\,,\ \mathcal{Q}_{zi} = \mathcal{Q}_z|_{\psi=\psi_i}\,.$$

With

$$\begin{aligned} E &= \left[\mathcal{E}_1^T, \mathcal{E}_2^T, \ldots, \mathcal{E}_{N-1}^T\right]^T, \\ A &= \mathrm{diag}(\mathcal{A}_i), \\ Q &= \left[\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_{N-1}\right], \\ Q_z &= \left[\mathcal{Q}_{z1}, \mathcal{Q}_{z2}, \ldots, \mathcal{Q}_{zN-1}\right], \end{aligned}$$

(3.15) implies

$$E = AQ_z^T,$$

which is the discretization result of the left-hand side of equation (3.14).

Note that $\mathcal{A}_i$, $i = 1, \ldots, N-1$, are band matrices with upper bandwidth equal to 1 and lower bandwidth equal to 0. Therefore, A inherits this property, too.

We use the forward finite difference to approximate p_ψ:

$$p_{\psi i} \approx \frac{p_{i+1} - p_i}{\psi_{i+1} - \psi_i}. \tag{3.16}$$

The terms including second derivatives are approximated by the central differences

$$\begin{aligned} \frac{\partial}{\partial\psi}\left(a\frac{\partial f}{\partial\psi}\right)_i \approx \quad & \left(\frac{2}{\psi_{i+1} - \psi_{i-1}}\right) \\ & \left[\left(\frac{a_{i+1} + a_i}{2}\right)\left(\frac{f_{i+1} - f_i}{\psi_{i+1} - \psi_i}\right) - \left(\frac{a_i + a_{i-1}}{2}\right)\left(\frac{f_i - f_{i-1}}{\psi_i - \psi_{i-1}}\right)\right]. \end{aligned}$$

In particular, this scheme is applied to the following terms

$$(\rho u \mu r^2 u_\psi)_\psi,\ (\rho u \lambda r^2 T_\psi)_\psi,\ \text{and}\ (r J_{k,r})_\psi. \tag{3.17}$$

The first derivatives; T_ψ in (3.7) and (3.9), and $X_{k\psi}$ in (3.9); are approximated by the central differences as

$$\begin{aligned} T_{\psi i} &= \left[\frac{\partial T}{\partial\psi}\right]_i \approx \frac{T_{i+1} - T_{i-1}}{\psi_{i+1} - \psi_{i-1}}, \\ X_{k\psi i} &= \left[\frac{\partial X_k}{\partial\psi}\right]_i \approx \frac{X_{ki+1} - X_{ki-1}}{\psi_{i+1} - \psi_{i-1}}. \end{aligned}$$

The fourth component of $\mathcal{F}$ is discretized by trapezoidal rule:

$$\left[\frac{\partial r^2}{\partial \psi} - \frac{2}{\rho u}\right]_i \approx \frac{r_i^2 - r_{i-1}^2}{\psi_i - \psi_{i-1}} - \frac{4}{\rho_i u_i + \rho_{i-1} u_{i-1}} \quad (i = 2, \ldots, N). \tag{3.18}$$

For our problems, there is a contact with catalyst wall. The areas near the wall have high spatial derivatives which require finer grid to resolve than other areas. Therefore, a non-uniform grid should be used. We use the approach suggested in [35]. The radial domain $\{r : 0 \leq r \leq r_{\max}\}$ is divided by grid points from the centerline to the wall of the channel, the location of the jth grid point is

$$r_j = r_{\max}\left(1 - \frac{(N-j)^\gamma}{(N-1)^\gamma}\right) \quad (j = 1, \ldots, N)$$

where $r_{\max}$ is the radius of the channel, N is the number of grid points, γ is a real factor used to control the distribution of grid points. With $\gamma = 1$, we have a uniform grid. The higher the value of γ is, the finer the grid near the wall is. For approximation of spatial derivatives on a non-uniform grid, see [54].

Let F_i denote the semi-discretized form of $\mathcal{F}_i = \mathcal{F}|_{\psi=\psi_i}$ by using the above described approximation schemes. Then

$$F = [F_1^T, F_2^T, \ldots, F_{N-1}^T]^T$$

is the discretization result of the right-hand side of equation (3.14). Hence, the PDE system (3.14) corresponds to

$$A(Q)Q_z^T = F(Q). \tag{3.19}$$

With

$$P = \begin{cases} \dot{s}_k W_k + J_{k,r}|_{\psi=\psi_N}, & \text{if } 1 \leq k \leq N_{\mathrm{g}} \\ \dot{s}_k, & \text{if } N_{\mathrm{g}} + 1 \leq k \leq N_{\mathrm{g}} + N_{\mathrm{s}} \end{cases}$$

the boundary conditions (3.11)–(3.12) can be written as

$$P = 0. \tag{3.20}$$

At channel wall $\psi = \psi_N$, u, T, p, and r must fulfill

$$\begin{bmatrix} u_N \\ p_N \\ T_N \\ r_N \end{bmatrix} - \begin{bmatrix} 0 \\ p_{N-1} \\ T_{\mathrm{wall}} \\ r_{\max} \end{bmatrix} = 0. \tag{3.21}$$

Finally, equations (3.19), (3.20), and (3.21), and $r_1 = 0$ form a DAE system with the unknowns

$$[Q_1, Q_2, \ldots, Q_N, \Theta_1, \ldots, \Theta_{N_s}],$$

which satisfy, in addition, conditions (1.5) and (1.27). In the following, the DAE system is referred to as (DAE1). Note that Θ_1,..., Θ_{N_s} belong to the variables of $\dot{s}_k$, i.e., of function P in (3.20).

The (DAE1) is a stiff system. The sources of stiffness are from the inherent stiffness in the chemical system with detailed models and also from the semi-discretization of the PDE. Therefore, for solving it we use an implicit method, based on backward differentiation formulas (BDF) with efficient adaptive step size, order control and efficient monitoring strategies for integration along the z-direction, which is described in Chapter 2. Roughly speaking, implicit methods are not restricted by the the well-known CFL (R. Courant, K. Friedrichs and H. Lewy) condition for stability of an explicit finite difference discretization as the classical direct method for PDEs, which limits the time stepsize by the spatial stepsize. For a parabolic PDE, the CFL number is proportional to $\Delta t/(\Delta x)^2$, where Δt is the time stepsize and Δx is the spatial stepsize (for our problem here, the time stepsize is in z (timelike) direction and spatial stepsize is in ψ direction). Thus, the time step must be very small for an explicit scheme to be stable.

3.4 Structure and index of the DAEs

Before presenting a numerical method for solving the DAEs obtained in Section 3.3, we analyze some important properties of the DAEs. Our DAEs in Section 3.3 (DAE1) can be written in general form as

$$\begin{array}{rcl} B(t,x,y)\dot{x} & = & f(t,x,y), \\ 0 & = & g(t,x,y), \end{array} \tag{3.22}$$

where

$$\begin{array}{ll} B: & [t_0,t_e]\times\mathbb{R}^{n_d}\times\mathbb{R}^{n_a}\to\mathbb{R}^{n_f\times n_d}, \\ f: & [t_0,t_e]\times\mathbb{R}^{n_d}\times\mathbb{R}^{n_a}\to\mathbb{R}^{n_f}, \\ g: & [t_0,t_e]\times\mathbb{R}^{n_d}\times\mathbb{R}^{n_a}\to\mathbb{R}^{n_g}. \end{array}$$

In addition, $n_d \geq n_f$ and B has full range. Here, for our formulation, z is denoted by t, Q and Θ are denoted by x and y.

In the (DAE1); u_i, T_i, p_i, and Y_{ki} $(k = 1, \ldots, N_g)$, $(i = 1, \ldots, N-1)$ are the differential variables; r_i $(i = 1, \ldots, N)$ and u_N, T_N, p_N, and Y_{kN} $(k = 1, \ldots, N_g)$, and Θ_j $(j = 1, \ldots, N_s)$ are the algebraic variables.

The structural matrix of the DAE system (DAE1) in Section 3.3 is

$$S = \begin{bmatrix} S_{11} & S_{12} & 0 & & & & & & \\ S_{21} & S_{22} & S_{23} & \ddots & & & & \mathrm{O} & \\ 0 & S_{32} & S_{33} & \ddots & & & & & \\ \vdots & \ddots & \ddots & \ddots & & & & & \\ & & \dots & S_{i\,i-1} & S_{i\,i} & S_{i\,i+1} & \ddots & & \\ \vdots & \vdots & & & & & & & \\ & \mathrm{O} & \dots & & & S_{N-2\,N-1} & S_{N-1\,N-1} & S_{N-1\,N} & \\ & & \dots & & & & S_{N-1\,N} & S_{N\,N} & W_1 \\ & & \dots & & & & & W_2 & W_3 \end{bmatrix}$$

where $S_{i\,i}$ is a $(N_{\mathrm{g}}+4)\times(N_{\mathrm{g}}+4)$ matrix

$$S_{i\,i} = \begin{array}{c} \begin{matrix} \dot{u}_i & \dot{p}_i & \dot{T}_i & r_i & \dot{Y}_{1i} & \dot{Y}_{2i} & \dot{Y}_{3i} & \dots & \dot{Y}_{N_g\,i} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \end{array} \qquad (i = 1, \dots, N-2).$$

In the following, we derive the relation between each equation of (DAE1) and the algebraic variables and the derivative of the differential variables.

To derive the structural matrix we need to "look at" the equations to see if the ith equation involves the derivative of the jth differential variables or the jth algebraic variables, then $S_{ij} = 1$ otherwise $S_{ij} = 0$. Alternatively, one can compute the structural matrix of the DAE as in Chapter 2, i.e., first computing the Jacobian matrix of the DAE with respect to highest-order "time"(z) derivatives and then computing the structure matrix of the Jacobian.

The first row of $S_{i\,i}$ reflects the dependency of the momentum equation (3.4) discretized at the ith grid point on the algebraic variables and derivative of differential variables. One can see that the momentum equation depends on u_z, p_z and r. Here we denote $\dot{u}$ by u_z, $\dot{p}$ by p_z, etc. The second row of $S_{i\,i}$ reflects the dependency of the momentum equation (3.5) discretized at the

ith grid point. The discretization of this equation is (3.16), which depends on p_{i+1} and p_i, and not on the algebraic variables or derivative of the differential variables. Therefore, all entries in this row equal zero. Similarly, the third row represents the dependency of the energy equation (3.7) discretized at the ith grid point. The 4-th row corresponds to (3.18), and the 5-th row, etc., corresponds to the (3.6) discretized at the ith grid point.

$$S_{N-1\,N-1} = \begin{array}{c} \begin{matrix} \dot{u}_{N-1} & \dot{p}_{N-1} & \dot{T}_{N-1} & r_{N-1} & \dot{Y}_{1N-1} & \dot{Y}_{2N-1} & \dot{Y}_{3N-1} & \dots & \dot{Y}_{N_g\,N-1} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \end{array}$$

$$S_{i\,i+1} = \begin{array}{c} \begin{matrix} \dot{u}_{i+1} & \dot{p}_{i+1} & \dot{T}_{i+1} & r_{i+1} & \dot{Y}_{1i+1} & \dot{Y}_{2i+1} & \dot{Y}_{3i+1} & \dots & \dot{Y}_{N_g\,i+1} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \end{array} \qquad (i = 1, \dots, N-2),$$

$$S_{i\,i-1} = \begin{array}{c} \begin{matrix} \dot{u}_{i-1} & \dot{p}_{i-1} & \dot{T}_{i-1} & r_{i-1} & \dot{Y}_{1i-1} & \dot{Y}_{2i-1} & \dot{Y}_{3i-1} & \dots & \dot{Y}_{N_g\,i-1} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \end{bmatrix} \end{array} \qquad (i = 2, \dots, N-1).$$

W_1 is an $(N_\mathrm{g}+4)\times N_\mathrm{s}$ matrix, which corresponds to the equations (3.21) and the first part P in (3.20),

$$W_1 = \begin{array}{c} \begin{matrix} \Theta_1 & \Theta_2 & \Theta_3 & \dots & \Theta_{N_\mathrm{s}} \end{matrix} \\ \left[\begin{array}{ccccc} 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{array}\right] \end{array}.$$

The first four rows of W_1 represent (3.21) and the rest represents the first part of P in (3.20).

W_3 is an $N_\mathrm{s}\times N_\mathrm{s}$ matrix, which represents the dependency of the second part of P in (3.20) on the algebraic variables Θ_i, $i=1,\dots,N_\mathrm{s}$,

$$W_3 = \begin{array}{c} \begin{matrix} \Theta_1 & \Theta_2 & \Theta_3 & \dots & \Theta_{N_\mathrm{s}} \end{matrix} \\ \left[\begin{array}{ccccc} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{array}\right] \end{array},$$

$$S_{N\,N} = \begin{array}{c} \begin{matrix} u_N & p_N & T_N & r_N & Y_{1N} & Y_{2N} & Y_{3N} & \dots & Y_{N_g\,N} \end{matrix} \\ \left[\begin{array}{ccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & \dots & 1 \end{array}\right] \end{array},$$

and W_2 is an $N_\mathrm{s} \times (N_\mathrm{g} + 4)$ matrix,

$$W_2 = \begin{array}{c} \begin{array}{ccccccccccc} u_N & p_N & T_N & r_N & Y_{1N} & Y_{2N} & Y_{3N} & & \dots & & Y_{N_g N} \end{array} \\ \left[\begin{array}{ccccccccccc} 0 & 1 & 1 & 0 & 1 & 1 & 1 & & \dots & & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & & \dots & & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & & \dots & & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & & \dots & & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \ddots & & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & & \dots & & 1 \end{array}\right] \end{array}.$$

Because all entries of the second row of $S_{i\,i}$, $S_{i\,i+1}$ and $S_{i-1\,i}$ equal zero, the consequence is that S is structurally singular. Therefore, (DAE1) is structurally singular.

If one differentiates all algebraic constraints $r_1 = 0$, (3.20), (3.21) and the second and the 4-th components of the discretized version of (3.14), then the resulting equations along with other equations of (DAE1) form an implicit ODE which can be solved to obtain the first derivatives of all variables of (DAE1). Thus, (DAE1) is of index 1.

3.5 Solving nonlinear boundary conditions

To integrate (DAE1), a set of consistent initial values (of differential variables and algebraic variables) is needed. The differential variables are specified as stated in Section 3.2. The algebraic variables r_i $(i = 2, \dots, N-1)$ are supplied by taking into account of (3.18) and $r_1 = 0$ and $r_N = r_\mathrm{max}$. The variables u_N, p_N and T_N are given according to (3.21). The remaining algebraic variables are the mass fractions Y_k $(k = 1, \dots, N_\mathrm{g})$ at the catalytic wall $\psi = \psi_N$ and the surface coverage fractions Θ_i $(i = 1, \dots, N_\mathrm{s})$ which are implicitly defined by the highly nonlinear equations (3.20) and the constraints (1.5) and (1.27). Due to restrictions (1.5) and (1.27), standard methods are no more suitable, and methods following feasible paths should be applied instead. Newton's method or globalized Newton methods by linesearch fails due to the fact that with a starting guess the Newton direction at the first step is a "wrong direction", i.e., because with such direction we cannot advance to the next iteration, which fulfills the constraints (1.5) and (1.27). In the following, we present a method for solving the nonlinear equations based on a time-stepping method. This is a standard practice for determining the steady state of a dynamic system (see e.g., [55]). This method is quite stable and it is able to obtain a steady state solution if the dynamic system starting with the initial values leads to the stable steady state. Roughly speaking, in general

the method guarantees obtaining a steady state if that state is attractive from any initial condition of the system. However, it is much slower than the Newton-like methods when they converge. Therefore, in the following a combination of time-stepping and Newton's method is employed to speedup the convergence.

Consider a nonlinear equation

$$f(x) = 0, \tag{3.23}$$

where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$. Equation (3.20) is a special case of this one with $x = (Y_{1N}, \ldots, Y_{N_{\mathrm{g}}N}, \Theta_1, \ldots, \Theta_{N_{\mathrm{s}}})$ and $n = N_{\mathrm{g}} + N_{\mathrm{s}}$.

Suppose that the system (3.23) describes the steady state of a system whose transient state model is given by

$$\dot{\mathcal{X}} = f(\mathcal{X}), \quad \mathcal{X} = \mathcal{X}(t). \tag{3.24}$$

3.5.1 Properties of Newton's method and quasi-Newton methods

Newton's method is a standard one for solving nonlinear equations. Starting with $x_0 \in \mathbb{R}^n$, the initial approximation to the solution of (3.23), Newton's method tries to improve x_0 using the iteration scheme defined as

$$J(x_k)\Delta x_k = -f(x_k), \qquad x_{k+1} = x_k + \Delta x_k, \tag{3.25}$$

where the Jacobian $J(x_k) = \partial f(x)/\partial x|_{x=x_k}$. In some modified variants, called quasi-Newton methods, the Jacobian $J(x_k)$ is approximated by finite differences or update methods. This defines a sequence of approximations $\{x_0, x_1, x_2, \ldots, x_{k-1}, x_k, \ldots\}$ to an exact solution. To describe properties of a sequence, we need to define the following terminologies.

Definition 3.5.1 (Convergence and rates of convergence)
A sequence $\{x_k\}$, $k = 0, 1, 2 \ldots$ *is said to* converge *to* x^* *if*

$$\lim_{k\to\infty} \|x_k - x^*\| = 0.$$

A sequence $\{x_k\}$ *is said to converge* ***linearly*** *to* x^* *if there exists a constant* $c \in [0, 1)$ *and an integer* $\widehat{k} \geq 0$ *such that for all* $k \geq \widehat{k}$,

$$\|x_{k+1} - x^*\| \leq c\|x_k - x^*\|.$$

The ***quotient-convergence rate*** *or* ***q-factor*** $\rho^{(q)}$ *is defined as*

$$\rho^{(q)} = \limsup_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|},$$

and ***root-convergence rate*** *or* ***r-factor*** $\rho^{(r)}$ *is defined as*

$$\rho^{(r)} = \limsup_{k\to\infty} \|x_k - x^*\|^{1/k}.$$

If there is a sequence $\{c_k\}$ *that converges to zero, and*

$$\|x_{k+1} - x^*\| \le c_k \|x_k - x^*\|, \quad \forall k \ge \widehat{k},$$

then $\{x_k\}$ *is said to be* ***superlinearly convergent*** *to* x^**. A sequence* $\{x_k\}$ *is said to be* ***quadratically convergent*** *to* x^* *if*

$$\|x_{k+1} - x^*\| \le c\|x_k - x^*\|^2 \quad \forall k \ge \widehat{k},$$

where c *is a positive constant.*

When the initial guess x_0 is chosen close to a solution x^*, and the function is continuously differentiable, and the Jacobian is nonsingular, the Newton iteration will converge to x^*. The local convergence properties of Newton's method are summarized as follows [21].

Theorem 3.5.1 (Local convergence properties)
Let $f\colon \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}^n$ *be twice continuously differentiable,* $J(x)$ *be nonsingular for all* $x \in \mathcal{D}$*, and* $\mathcal{D}$ *be a domain. Assume further that*

$$\|J(y)^{-1}(J(x+\tau\Delta x) - J(x))\Delta x)\| \le \omega\tau\|\Delta x\|^2, \qquad \omega < \infty, \tag{3.26}$$

for all $\tau \in (0,1]$*,* $x, y = x + \Delta x \in \mathcal{D}$ *with* $\Delta x = -J(x)^{-1}f(x) \ne 0$*, i.e. a global bound* ω *for the "curvature" exists, and that the initial guess* x_0 *is sufficiently near to a solution:*

$$\delta_0 = \frac{\omega}{2}\|\Delta x_0\| < 1. \tag{3.27}$$

Then the following holds:

- *if* $\mathcal{D}_0 = B(x_0, \|\Delta x_0\|/(1-\delta_0)) \subset \mathcal{D}$*, then the sequence of iterates defined by (3.25) remains in* $\mathcal{D}_0$*,*
- *there exists* $x^* \in \mathcal{D}_0$ *with* $f(x^*) = 0$ *and* $x_k \to x^* \quad (k \to \infty)$*,*
- *an a priori error estimate holds*

$$\|x_k - x^*\| \le \delta_0^k \frac{\|\Delta x_0\|}{1-\delta_0}, \tag{3.28}$$

- *and convergence is quadratic with*

$$\|\Delta x_{k+1}\| \leq \frac{\omega}{2}\|\Delta x_k\|^2. \tag{3.29}$$

When the starting guess x_0 does not lie close enough to a solution, it is not guaranteed that the sequence $\{x_k\}$ converges to the solution. To have globally convergent behavior, these methods can be modified by damping or under-relaxation. The iteration is defined then by

$$x_{k+1} = x_k + \tau_k \Delta x_k,$$

where the step size $\tau_k \in (0, 1]$ determined by a line search or trust region method, which uses an appropriate *level function* (also sometimes called *merit function*) $h(x)$, such as $h(x) = \|f(x)\|_2^2$, and requires that the sequence $\{h(x_k)\}$ is strictly monotone decreasing. This ensures global convergence if the Jacobians are bounded away from singularity. For problems whose the Jacobian is (mildly) ill-conditioned one should use the *natural level function* (see [42] and [43]) instead.

However, Newton's method or quasi-Newton methods may fail to converge to some solution in some cases, such as, when the Newton direction Δx_0 points in a "wrong" direction or the nonsingularity of Jacobians are not guaranteed, therefore, the globalization techniques using the merit function do not make any sense at all.

It is well known that Newton's method could fail when the initial point is far away from a solution. Even the globalized Newton-like methods may fail if the condition of nonsingularity of Jacobian is violated.

3.5.2 Combining pseudo-time integration and Newton's method

Instead of using Newton's method or a globalized Newton method which will fail to find a solution of equation (3.23), we integrates the ordinary differential equation (3.24) with the initial values

$$\mathcal{X}(t_0) = \mathcal{X}_0$$

to be known, for a long enough time interval. In other words, to find the solution of the steady state equation, we solve the related transient equation until it reaches steady state conditions.

In fact, we have tried to solve (3.20) by methods in [40] and [21] but without success. At the first step, the search direction points to an infeasible region (the region where the conditions (1.5) and (1.27) are not being fulfilled). This does not allow us to advance to the next step.

Since in our problems the ODEs (3.24) describing the chemical process modelled using detailed chemistry, are very stiff, we use the BDF method, which is a implicit one, to solve it. The solution of (3.24) using the BDF method requires the partial derivative $\partial P/\partial \mathcal{X}$, which is generated by automatic differentiation tool ADIFOR 2.0.

If one integrates the ODE over a quite long interval approximating the time for the physical system reaches the steady state conditions, then the solution of equation (3.23) is the value of $\mathcal{X}(t)$ at the end of the interval. The implicit-time stepping process is time consuming, because it requires to solve a nonlinear equation system at each integration step. Instead of taking a long integration interval, we use a reasonable duration. Then we apply Newton's method with the final value of $\mathcal{X}(t_{\rm f}) = \mathcal{X}_{\rm f}$ as the starting guess. This idea is proposed in [55] for the solution of steady, laminar, one-dimensional, premixed flames. If $\mathcal{X}_{\rm f}$ lies in the local domain of convergence, we only need a few Newton steps to converge to the solution within an acceptable tolerance. If it fails to converge, we do another time integration with a new interval. Now, the solution of the ODE is only used as the initial value for the subsequent Newton's method. Therefore, taking high tolerance value for integrating the ODE is not necessary because to achieve a high accurate solution the ODE solver usually takes many steps. Moreover, choosing a high integration tolerance may even leads to failure of the solver because of stiffness of the ODE.

It is worth noting that care should be taken if one tries to use methods in [40]: applying linesearch procedure and perturbation of the local model when the Jacobian $J(x_k)$ is ill-conditioned. Actually, we have implemented the methods in [40] and it fails to converge to the solution after 40 iterations (for our practical problem) even using a good initial guess obtained from the time-stepping procedure for our practical problems. This phenomenon is also discussed in [21].

Now we return back to the above assumption that the nonlinear equation $f(x) = 0$ is the steady state equation whose transient state model is as in (3.24). The original form of the equation $f(x) = 0$ are

$$\dot{s}_k W_k + J_{k,r}|_{\psi=\psi_N} = 0 \quad (k = 1, \ldots, N_{\rm g}) \tag{3.30}$$

$$\dot{s}_k = 0 \quad (k = N_{\rm g} + 1, \ldots, N_{\rm g} + N_{\rm s}). \tag{3.31}$$

The left-hand side of equation (3.31) $\dot{s}_k$ $(k = N_{\rm g} + 1, \ldots, N_{\rm g} + N_{\rm s})$ are the rate of creation/depletion of the surface coverage of the kth surface species

multiplied by the site density Γ as described by (3.12), and we repeat it here for convenience

$$\frac{\partial \Theta_i}{\partial t} = \frac{\dot{s}_k \sigma_i}{\Gamma} \quad (k = N_{\mathrm{g}} + 1, \ldots, N_{\mathrm{g}} + N_{\mathrm{s}}). \tag{3.32}$$

Similarly, the left-hand side of equation (3.30) can be considered as the mass rate of creation/depletion of the kth gas species by surface reactions and diffusion process multiplied by a some length dr, i.e.,

$$\rho dr \frac{\partial Y_k}{\partial t} = \dot{s}_k W_k + J_{k,r}|_{\psi=\psi_N} \quad (k = 1, \ldots, N_{\mathrm{g}}). \tag{3.33}$$

Remark 3.5.1
The constant positive factors $1/\Gamma$ and ρdr in Equations (3.32) and (3.33) can be eliminated without changing the qualitative properties of the system, i.e. the sign of eigenvalues are not changed. However, we introduce them here to emphasize that the ODE should be derived based on the physical behavior of the system, i.e., the ODE should describe the related transient system.

Remark 3.5.2
The iteration matrix of Newton's method applied to $f(x) = 0$ is

$$J^{\mathrm{Newton}} = \frac{\partial f(x)}{\partial x},$$

and the iteration matrix for the time-stepping method, e.g., using the backward Euler method is

$$J^{\mathrm{time-step}} = I - hJ^{\mathrm{Newton}},$$

where I is the identity matrix. For small integration step h the iteration matrix $J^{\mathrm{time-step}}$ approaches the identity matrix. Thus, the time-stepping method is very stable even for ill-conditioned (or nearly singular) Jacobian of the steady-state problem J^{Newton}.

To globalize the convergence of Newton's method, a linesearch procedure or trust region method can be applied. On the other hand, globalization can slow down the convergence speed because it needs extra computation in the globalized stage and may take small stepsizes in the region of the local convergence domain. In particular, when the Jacobian matrix of nonlinear equations is (mildly) ill-conditioned, this is in the case of our problems (3.30) and (3.31), the standard linesearch with the level function $h(x) = \|f(x)\|_2^2$) gives very small stepsize. In [21] an illustrative example for this phenomenon is given.

The reason, that the stepsize is very small, is that the search direction Δx_0 and the direction of the steepest decent of the level function h at x_0 is nearly orthogonal. To avoid this effect one can: (a) modify the search direction such as using the Levenberg–Marquardt or trust region variant, in which the search direction is replaced by

$$\Delta x_k(\gamma) = -(J(x_k)^T J(x_k) + \gamma I)^{-1} J(x_k)^T f(x_k),$$

or (b) modify the level function by using the *natural level function* (see [42] and [43])

$$h(x) = \|J(x_k)^{-1} f(x)\|_2^2,$$

which seems to work well in practical applications (see e.g., [42], [43], [4], [17], [18], and [19]) but no global convergence proof is given because the level function is changed at each iteration. Recently, in [21] a stepsize strategy based on the *restricted monotonicity test* (RMT) is proposed, the stepsize is controlled by

$$\begin{aligned} & \tau_k = \max \tau \\ \text{subject to} \quad & \tau \le 1,\ \tau\omega_1(\tau)\|\Delta x_k\| \le \eta,\ 0 < \eta < 2, \\ & \omega_1(\tau) = \sup_{0\le\beta\le\tau} \frac{\|J(x_k)^{-1}(J(x_k+\beta\Delta x_k) - J(x_k)\|)}{\beta\|\Delta x_k\|}. \end{aligned} \tag{3.34}$$

In practice, the curvature ω_1 is replaced by the weaker estimate ω_3

$$\omega_3(\tau) = \frac{2\|J(x_k)^{-1}(f(x_k+\tau\Delta x_k) - (1-\tau)f(x_k))\|}{\tau^2\|\Delta x_k\|^2}. \tag{3.35}$$

It is interesting to note that the conditions (1.5) and (1.27) are being satisfied during the integration of ODE if they are being fulfilled at the initial and the governing ODE model has certain physical meaning. Specifically for our problem, the equality constraints $\sum_{k=1}^{N_g} Y_k = 1$ and $\sum_{i=1}^{N_s} \Theta_i = 1$, which become $\sum_{k=1}^{N_g} \mathcal{X}_k(t) = 1$ and $\sum_{k=1}^{N_s} \mathcal{X}_{N_g+k}(t) = 1$ accordingly, are followed immediately if $\sum_{k=1}^{N_g} \mathcal{X}_k(t_0) = 1$ and $\sum_{k=1}^{N_s} \mathcal{X}_{(N_g+k)}(t_0) = 1$, and also

$$\sum_{k=1}^{N_s} \dot{s}_{(N_g+k)}/\Gamma = 0, \quad \sum_{k=1}^{N_g} (\dot{s}_k W_k + J_{k,r}|_{\psi=\psi_N}) = 0. \tag{3.36}$$

The two latter restrictions are being implicitly met due to the conservation law of mass applying to Θ_i and Y_k. The positive constraints $Y_k \ge 0$ and $\Theta_i \ge 0$ are automatically fulfilled if the ODE interprets physical nature, meaning that if $Y_k = 0$ then $\dot{s}_k W_k + J_{k,r}|_{\psi=\psi_N} \ge 0$ and also if $\Theta_i = 0$ then $\dot{s}_{(N_g+i)} \ge 0$. It means that if a species does not exist at all then its

depletion rate cannot be positive. Finally, the conditions $Y_k \leq 1$ and $\Theta_i \leq 1$ are automatically satisfied if the all other ones are being met. However, in the second stage when we solve $f(x) = 0$ by Newton's method, all these conditions may not be fulfilled. In addition, since (3.36) is implicitly met, therefore if one uses the original form of $f(x) = 0$, then the Jacobian of $f(x)$ is singular. In our implementation, we replace one of equations in (3.30) by equation $\sum_{k=1}^{N_g} Y_k = 1$, and one of equations in (3.31) by equation $\sum_{i=1}^{N_s} \Theta_i = 1$ and during the iteration we check for the conditions $0 \leq Y_k \leq 1$ and $0 \leq \Theta_i \leq 1$. If these are not satisfied, then we try to reduce the step size α_k. If α_k is too small, we suspect that the initial guess does not lie in the domain of convergence, so another time integration is being made to get a better guess.

To conclude this section, we would like to summarize it as the follows.

- We have nonlinear equations including certain constraints on the unknown. These constraints make the problem more difficult to solve and restricts the methods to be used. One should take care of the constraints and use a tailored method.

- We show some situations where Newton's method or quasi-Newton methods fail to converge to a solution. These are the cases when the initial Newton direction points in a "wrong" direction.

- A combination of pseudo-time integration and Newton's method is applied. The pseudo-time integration is employed for finding a "good" initial guess, then Newton's method is used to obtain a faster convergence to the solution.

- The global convergence of the combining time-integration and Newton's method mainly depends on the existence of steady state of the underlying physical system. The solution of the nonlinear equation is the steady state of a some physical system. The steady state system is the asymptotic limit of corresponding transient system. The convergence to the solution depends on the existence of the steady state. If the system does not reach a steady state, then obviously the method would fail. The failure of the method could reflect the non-existence of a steady state of the system. This is a expected behavior of the approach, then one should re-consider the mathematical model of the system.

- For the pseudo-time integration interval, we choose value of 1 for each time integration. The chosen value of time interval is quite successful

from numerical experiments with our test problems, this also indicates that the system is going to reach the steady state after 1 second, but the chosen value is not a general one and only based on our experiences.

It is worth noting that the above time-integration may fail to converge to a solution, and care should be taken when one formulates the problem, especially in the construction of the ODE, otherwise the ODE may be unstable. The ODE should be derived based on the physical nature of the system, for example, the process of forming steady state from transient process and using appropriate initial conditions. By taking care of this, we know that the system reaches a steady state after a certain interval. In the following, we construct an example that the above time-integration procedure fails to obtain the solution.

Consider the following system of equation

$$f(x) = \begin{bmatrix} -x_2 \\ x_1 \\ x_1 + x_2 + x_3 - 4 \end{bmatrix} = 0.$$

This equation system has a unique solution $x_1 = 0$, $x_2 = 0$, $x_3 = 4$. If one tries to do time-integration the corresponding ODE, difference solutions of the ODE are obtained depending on choosing initial values, such as, $\mathcal{X}_1(t) = \cos t$, $\mathcal{X}_2(t) = \sin t$, and $\mathcal{X}_3(t) = 4 - \mathcal{X}_1(t) - \mathcal{X}_2(t) = 4 - \cos t - \sin t$ for $\mathcal{X}_1(0) = 1, \mathcal{X}_2(0) = 0, \mathcal{X}_3(0) = 3$. This solution never reaches a steady state.

3.5.3 Multiple solutions of boundary conditions

It is well know that nonlinear equations in general could have multiple solutions. The aim of this section is to report that the nonlinear equations (3.30) and (3.31) could have multiple solutions by giving certain problem settings with which the equations (3.30) and (3.31) have at least two solutions.

Let consider the following example.

Example 3.5.1 (Catalytic combustion of methane)
A gas mixture flows in the channel with the following setting.

- *Channel geometry: the radius* $r_{\max} = 2.5 \times 10^{-4}$ *[m], the channel length* $z_{\max} = 0.01$ *[m].*
- *Initial conditions: the initial mole fraction of each species* $X_{\mathrm{CH_4}} = 0.5$, $X_{\mathrm{O_2}} = 0.4$, $X_{\mathrm{N_2}} = 0.1$, *other species are absent at inlet. The initial gas temperature is* $T_{\mathrm{gas}} = 298$ *[K], the initial pressure is* $p = 1.2 \times 10^5$ *[Pa], and the initial velocity is* $u = 0.5$ *[m/s].*

– *Boundary conditions: the temperature at the wall is* $T_{\text{wall}} = 1200$ *[K].*

– *Reaction mechanisms: 21 gas-phase species, 11 surface species, 23 surface reactions, and 128 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.*

– *Number of grid points: 12.*

(a) Applying Newton's method with the initial guess (Initial Value column) as in Table 3.1 and Table 3.2, we obtain a solution (Computed Solution column) as in Table 3.1 and Table 3.2.

Table 3.1: Initial value and computed solution of the surface coverage Θ_i (a).

	Surface Coverage Θ_i			Surface Coverage Θ_i	
Species	Initial Value	Computed Solution	Species	Initial Value	Computed Solution
PT(s)	4.061E-01	4.123E-01	C(s)	7.768E-06	8.119E-06
H2O(s)	8.219E-06	9.116E-06	CH3(s)	1.548E-07	1.576E-07
H(s)	5.899E-06	6.290E-06	CH2(s)	1.548E-07	1.576E-07
O(s)	5.905E-01	5.841E-01	CH(s)	1.548E-07	1.576E-07
OH(s)	2.789E-03	2.899E-03	CO2(s)	6.362E-08	6.585E-08
CO(s)	5.101E-04	5.338E-04			

(b) Applying Newton's method with the initial guess (Initial Value column) as in Table 3.4 and Table 3.5, we obtain a solution (Computed Solution column) as in Table 3.4 and Table 3.5.

We apply the linearized analysis at the solutions of the corresponding ODEs. The solutions of $f(x) = 0$ are the *critical points* of the ODE $\dot{\mathcal{X}} = f(\mathcal{X})$. As Table 3.7 shows the solution (a) is unstable because the ODE have one eigenvalue with positive real part, and the solution (b) is stable.

The conclusion to draw from this is that the nonlinear equations (3.30) and (3.31) may have multiple steady state solutions in some cases, which depends on the initial values. Thus, the choice of the initial values has an important influence on the convergence to a particular solution, which may be a physically realizable or a nonphysical steady state. Concerning the initial values, it is also confirmed by experiments [105] that the behavior of the system, i.e., oscillations, depends on the fact, where a brand new tube (catalyst) or a tube with aged surfaces is used for the experiment.

Table 3.2: Initial value and computed solution of the mass fraction Y_k at the wall (a).

	Mass Fraction Y_k			Mass Fraction Y_k	
Species	Initial Value	Computed Solution	Species	Initial Value	Computed Solution
H2	1.636E-09	1.887E-09	O	0	0
O2	4.846E-01	4.733E-01	HO2	0	0
H2O	2.434E-02	2.679E-02	H2O2	0	0
CO	1.122E-03	1.149E-03	CHO	0	0
CO2	4.029E-02	5.004E-02	CH2O	0	0
CH4	3.305E-01	3.295E-01	CH3	0	0
C2H6	0	0	CH3O	0	0
C2H4	0	0	C2H3	0	0
C2H2	0	0	C2H5	0	0
OH	0	0	N2	1.190E-01	1.191E-01
H	0	0			

Table 3.3: Newton iterations with initial conditions as in Tables 3.4 and 3.2.

# Iteration	$\|f(x_k)\|$	$\|\Delta x_k\|$	(Est. cond)$^{-1}$
1	3.231E+02	5.033E-02	4.344E-08
2	2.531E+04	2.183E-03	3.908E-08
3	5.531E+01	5.374E-06	3.888E-08
4	3.180E-04	2.822E-11	3.888E-08
5	3.612E-09		

3.5.4 Special problems with abnormal solutions and their numerical treatment

In this section we discuss problems with having special properties, such as singularity or discontinuity, and propose practical numerical treatments. General theory and numerical treatments for those problems are beyond the scope of this thesis, we refer to [100] and the references therein for those interested in. Here, we focus on our practical applications, which is index-1 DAE, and how they should be treated from a practical point of view. One common characteristic, which can be observed from numerical computation, of these problems is the failure of standard software for index-1 DAE when applying to these problems.

We identify two cases:

Table 3.4: Initial value and computed solution of the surface coverage Θ_i (b).

	Surface Coverage Θ_i			Surface Coverage Θ_i	
Species	Initial Value	Computed Solution	Species	Initial Value	Computed Solution
PT(s)	8.368E-01	8.396E-01	C(s)	1.042E-02	9.553E-03
H2O(s)	7.462E-05	7.488E-05	CH3(s)	3.463E-07	3.448E-07
H(s)	9.602E-04	9.186E-04	CH2(s)	3.463E-07	3.448E-07
O(s)	2.028E-03	2.211E-03	CH(s)	3.463E-07	3.448E-07
OH(s)	6.788E-04	7.098E-04	CO2(s)	6.380E-08	6.859E-08
CO(s)	1.489E-01	1.467E-01			

Table 3.5: Initial value and computed solution of the mass fraction Y_k at the wall (b).

	Mass Fraction Y_k			Mass Fraction Y_k	
Species	Initial Value	Computed Solution	Species	Initial Value	Computed Solution
H2	3.317E-05	3.021E-05	O	0	0
O2	3.184E-01	3.167E-01	HO2	0	0
H2O	1.190E-01	1.186E-01	H2O2	0	0
CO	9.600E-02	9.378E-02	CHO	0	0
CO2	4.848E-02	5.209E-02	CH2O	0	0
CH4	3.005E-01	2.970E-01	CH3	0	0
C2H6	0	0	CH3O	0	0
C2H4	0	0	C2H3	0	0
C2H2	0	0	C2H5	0	0
OH	0	0	N2	1.174E-01	1.216E-01
H	0	0			

(a) The (true) solution cannot be continued beyond some point t. Here, the solution is a continuous one, we do not consider discontinuous solutions.

(b) The solution can be continued at a certain point but a numerical solution may be stopped at that point due to numerical difficulties.

Loosely speaking, these points are usually referred to as *impasse points* (see [100], [30] and [102]).

Unlike ODEs $\dot{x} = f(t, x)$ where the smoothness of the model function f will ensure the smoothness of the solution, DAEs in general do not have that property.

To illustrate the case (a), let consider the following example.

Table 3.6: Newton iterations with initial conditions as in Tables 3.4 and 3.5.

# Iteration	$\|f(x_k)\|$	$\|\Delta x_k\|$	(Est. cond)$^{-1}$
1	8.189E-01	3.641E-02	3.921E-08
2	4.585E+04	1.510E-03	3.834E-08
3	4.984E+01	2.403E-06	3.837E-08
4	8.583E-06	1.700E-12	3.837E-08
5	3.318E-09		

Example 3.5.2

$$\begin{aligned} \dot{x}_1 &= -1 \\ x_1 - x_2^2 &= 0 \\ x(0) &= (4, 2). \end{aligned}$$

The solution is $x(t) = (-t+4, (-t+4)^{1/2})$*, the solution cannot be continued beyond* $t = 4$*.*

To illustrate the case (b), let consider the following example.

Example 3.5.3

$$\begin{aligned} \dot{x}_1 &= -2\sin(t+\alpha), \quad 0 < \alpha < \pi/2 \\ x_2^2 + x_1^2 + \beta - 4 &= 0, \quad 0 \le \beta < 4 \\ x(0) &= (2\cos\alpha, \sqrt{4 - \beta - 4\cos^2\alpha}). \end{aligned}$$

For $\beta = 0$*, the solution is* $x(t) = (2\cos(t+\alpha), \sqrt{4 - 4\cos^2(t+\alpha)})$*, which is defined for any value of* t*. Here, the model functions are smooth, but the solution is not smooth, i.e, the solution is not differentiable at points* t *such that* $x_2(t) = 0$ *(*$\cos(t+\alpha) = +1$ *or* $\cos(t+\alpha) = -1$*). The index-1 assumption (*$\partial g/\partial y$ *at (3.22) with* $B = I$ *is nonsingular) is violated at these points (*$\partial g/\partial y = 2x_2 = 0$*). Standard software for index-1 DAEs are usually failed and stopped at those points. To overcome the numerical difficulties, a well know method is transformation of the independent variable* t *(see e.g., [100] and [125]). For* $\beta > 0$*, the solution is* $x(t) = (2\cos(t+\alpha), \sqrt{4 - \beta - 4\cos^2(t+\alpha)})$*, and it cannot be continued beyond the point* t *(*$t > 0$*) where* $\cos(t+\alpha) = -(4-\beta)/4$*.*

Now we consider the following practical problem of interest.

Example 3.5.4
Let us consider the problem of conversion of ethane to ethylene with the following setting.

- *Channel geometry: the radius* $r_{\max} = 2.5 \times 10^{-4}$ *[m], the channel length* $z_{\max} = 0.01$ *[m].*
- *Initial conditions: the initial gas temperature is* $T_{\text{gas}} = 300$ *[K], the initial pressure is* $p = 1.2 \times 10^5$ *[Pa], and the initial velocity is* $u = 0.5$ *[m/s]. The initial mole fraction of nitrogen is* $X_{\text{N}_2} = 0.3$*, and the mole fraction of ethylene and oxygen are varied, which we will discuss later, other species are absent at inlet.*
- *Boundary conditions: the temperature at the wall is* $T_{\text{wall}} = 1000$ *[K].*
- *Reaction mechanisms: 25 gas-phase species, 20 surface species, 82 surface reactions, and 261 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.*
- *Number of grid points in the radial axis: 12.*
- $F_{\text{cat/geo}} = 1$.

We examine the numerical solutions for 5 cases, where the initial values of mole fractions of ethylene and oxygen, which are decreased in the mole fraction of ethylene and increased in the mole fraction of oxygen, and we keep the sum of mole fraction of ethane and of oxygen at 0.7, as in Table 3.8. Here, due to space restriction we only show the trajectories of surface species. Figures 3.1–3.3 show the surface coverages of the solutions corresponding to the different cases in Table 3.8. For cases (3) and (4), integration stopped at $z = 0.00799$ [m] and $z = 0.00995$ [m], respectively. The code BLAYER$^{\text{sim}}$ runs smoothly with the values of mole fractions of ethane and oxygen at the inlet as in case (1). The code also works well when we decrease the mole fraction of ethane (increase the mole fraction of oxygen) until reaching case (2). Further more decreasing the mole fraction of ethane below the value in case (2), the code fails as in cases (3) and (4). Furthermore decreasing the mole fraction of ethane until reaching the same value as in case (5), the code turns to work fine again. What we observe from Figures 3.1–3.3 (particularly see surface coverages of OH(s), CO2(s) and C2H3(2s)) and our numerical computations by using the method of transformation of the independent variable [100] is that the solution (in cases (3) and (4)) cannot be continued beyond the point where the code stops, and the boundary conditions (1.55) and (1.56) cease to have a solution (e.g., $\dot{s}_k > 0$ or $\dot{s}_k < 0$ for some k,

$N_g + 1 \leq k \leq N_g + N_s$) as illustrated by Examples 3.5.2 and 3.5.3. From this, we conclude that the system maybe do not have a stable steady state, or our approximation model is defective, because there is no reason that the real flow is not defined beyond that point.

3.6 Summary

In this chapter we have discussed techniques for solving the simulation problem. At first, we describe the von Mises transformation and apply it to the boundary layer equations. This allows us to eliminate the overall mass coninuity equation and replace it with an integral. Then, the resulting PDEs are semi-discretized by the method of lines, leading to a large stiff structured DAEs. We show that the DAEs is of index-1 and structurally singular. For solving the DAEs, consistent initial values are required and are obtained by solving the nonlinear equations, which are part of the algebraic constraints arising essentially from the nonlinear boundary conditions. The nonlinear equations are solved by a time-stepping and Newton's method. The time-stepping is used for obtaining a better initial guess before applying Newton's method, which is used for speeding up the convergence. By giving some examples, we also discuss some problems, such as the existence of multiple solutions of the nonlinear equations of the boundary conditions and the question, whether a stable steady state exists at all.

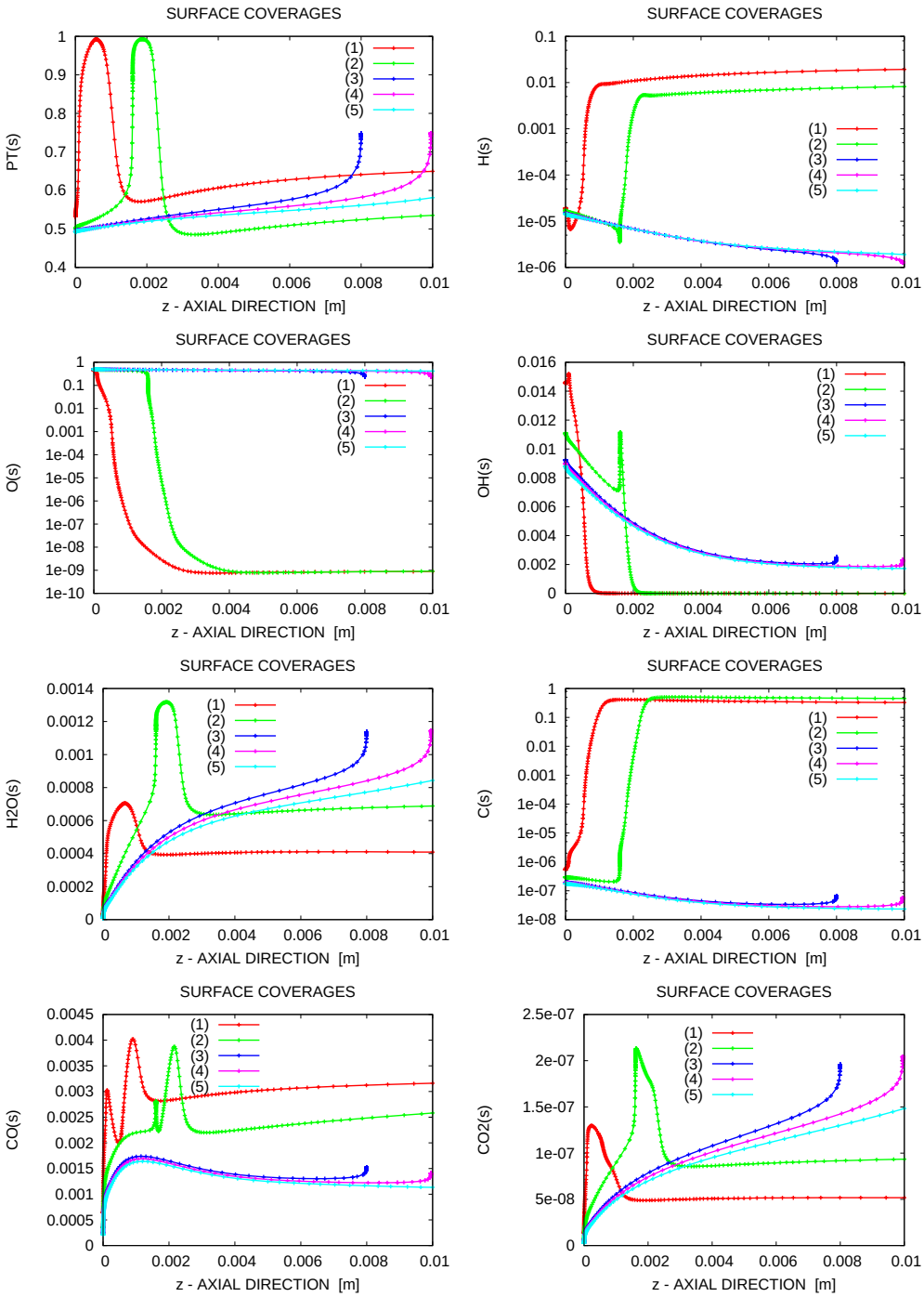

Figure 3.1: Surface coverages of the solution of Example 3.5.4 (I)

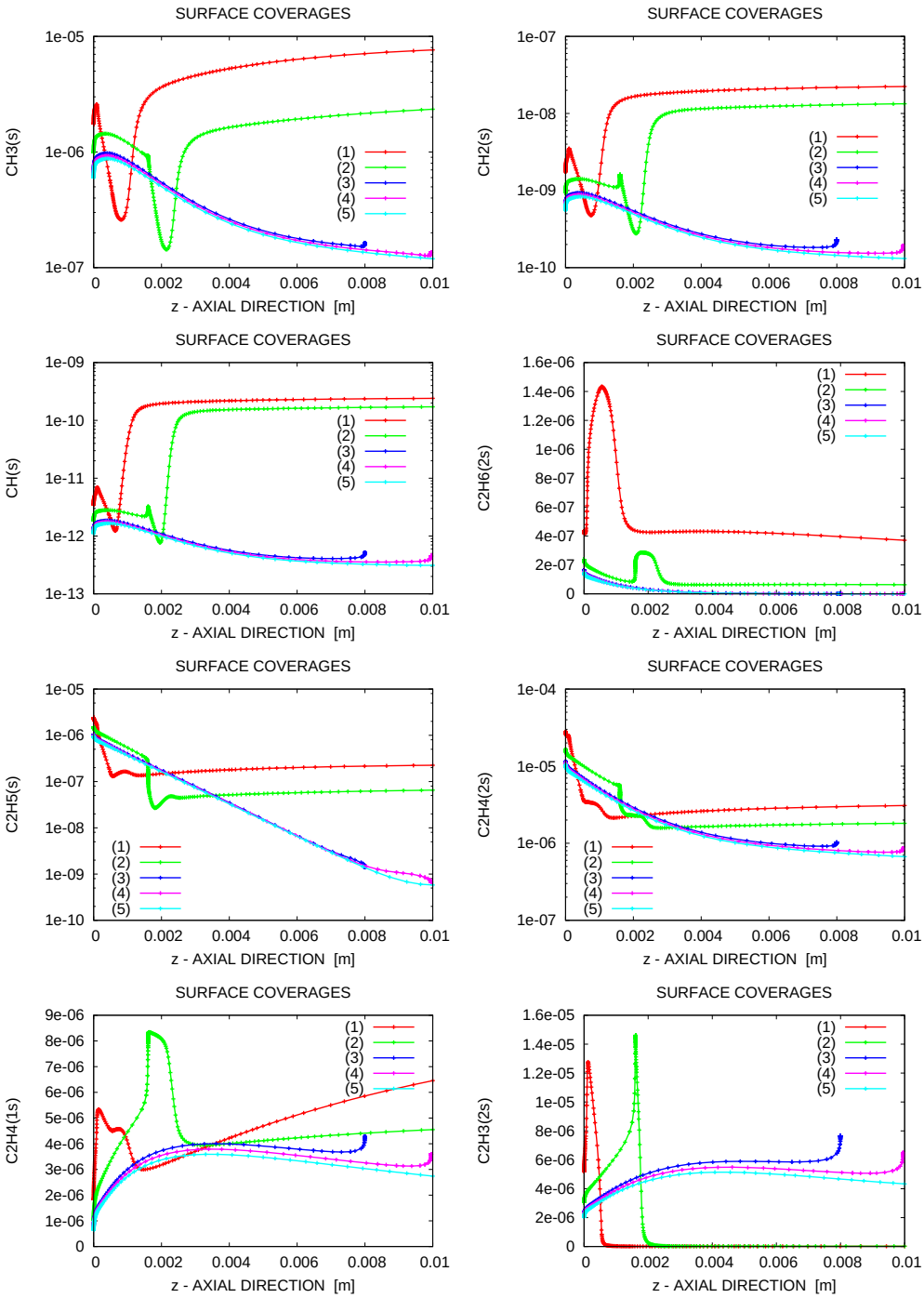

Figure 3.2: Surface coverages of the solution of Example 3.5.4 (II)

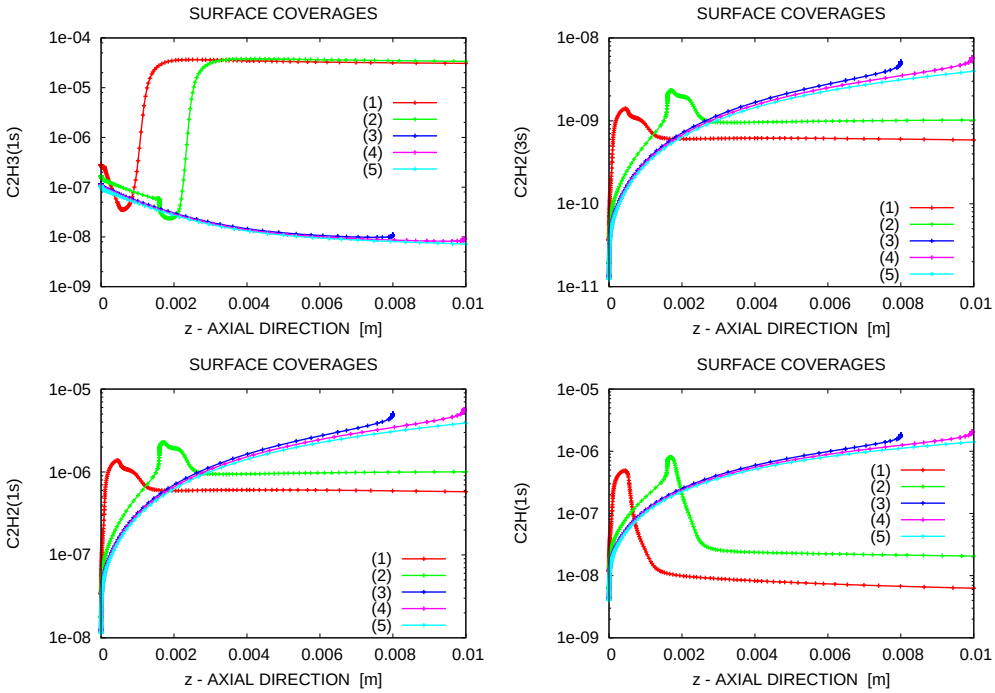

Figure 3.3: Surface coverages of the solution of Example 3.5.4 (III)

Table 3.7: Eigenvalues at the solutions. (*) positive eigenvalue.

Eigenvalues (a)	Eigenvalues (b)
$-5.574 \times 10^{11} + 2.837 \times 10^{9} \times i$	$-1.143 \times 10^{12} + 7.974 \times 10^{9} \times i$
$-5.574 \times 10^{11} - 2.837 \times 10^{9} \times i$	$-1.143 \times 10^{12} - 7.974 \times 10^{9} \times i$
-1.865×10^{12}	-1.129×10^{12}
-5.623×10^{12}	-1.281×10^{12}
-1.281×10^{12}	-1.816×10^{11}
-2.345×10^{11}	-1.812×10^{10}
-1.085×10^{10}	-2.886×10^{9}
-7.328×10^{8}	-2.198×10^{8}
-1.936×10^{8}	-4.743×10^{7}
-1.537×10^{6}	-7.182×10^{6}
-1.347×10^{1}	-1.690×10^{1}
-1.019×10^{1}	$-4.380 + 1.411 \times i$
-3.600	$-4.380 - 1.411 \times i$
-2.652	-2.049
-2.234	-2.560
-1.703	-3.293
-2.037×10^{-2}	-8.802×10^{-3}
$+1.641 \times 10^{-2}$ (*)	-2.732×10^{-2}
-1.902	-1.905
-2.075	-2.075
-2.113	-2.113
-3.785	-3.773
-1.434×10^{1}	-1.415×10^{1}
-3.852	-3.838
-2.535	-2.538
-2.519	-2.523
-2.170	-2.164
-2.153	-2.148
-2.788	-2.775
-2.113	-2.101
-2.094	-2.094
-1.917	-1.919

Case	$X_{C_2H_6}$	X_{O_2}
(1)	0.50	0.20
(2)	0.31	0.39
(3)	0.22	0.48
(4)	0.21	0.49
(5)	0.20	0.50

Table 3.8: Initial mole fractions of ethylene and oxygen with different test cases.

Chapter 4

Numerical Methods for Optimization

4.1 Introduction

There are demands for improving performance of catalytic reactors by determining the process conditions that lead to maximizing the performance of reactors. To achieve this goal, previous approaches in published works were manually trying with different process conditions (e.g., [47] and [105]). No systematic approach had been done for this problem before. In this chapter, for the first time a systematic approach for optimizing the process conditions is discussed.

In Section 4.2, we discuss about practical applications and the different types of optimization variables and objective functions which could be optimized in general. Mathematical formulation of the optimal control problem is presented in Section 4.3. Section 4.4 is devoted to the solution approach to the optimal control problem. Sequential Quadratic Programming (SQP) methods are discussed in Section 4.5. Computation of derivatives, which are necessary for the solution of SQP methods, are discussed in Section 4.6.

4.2 Practical optimization problems

In catalytic combustions, one usually can control the temperature profile at the wall $T_{\text{wall}}(z)$, or the initial conditions at inlet such as T_{gas}, u_0, and Yk_0, or the ratio of catalytic active surface area to geometric surface area $F_{\text{cat/geo}}$, to maximize the gas conversion or maximize the selectivity. The conditions to be fulfilled and objective function to be optimized are dependent on the particular application. In the following, we distinguish two types of control

quantities:

- **Control parameters** are control variables which do not depend on "time". In our case the axial coordinate z is treated as the time-like independent variable. These include the initial conditions: the inlet gas temperature T_{gas}, the inlet velocity u_0, and/or the inlet mass fractions Yk_0; and/or the channel length $z_{\max}$.

- **Control functions** are control variables which are functions of the axial coordinate z, such as the temperature profile at the wall $T_{\text{wall}}(z)$, and/or the ratio of catalytic active surface area to geometric surface area $F_{\text{cat/geo}}(z)$.

In the following the control parameters and control functions sometimes are referred to as *control variables*.

For practical reasons, there are often equality and inequality constraints such as the upper and lower bounds for the wall/gas temperature, or sum of all mass fractions must be one, or the mass fractions must be between zero and one, or the bounds for the inlet velocity.

4.3 Formulation of the optimal control problem

All above optimization problems can be formulated mathematically as a general optimization problem which minimizes a certain scalar function subject to the model equations and maybe additional constraints. In general, this optimization problem can be stated as

$$\begin{aligned} \min_{\mathbf{w},\,\mathbf{q}} \quad & \phi(\mathbf{w},\mathbf{q}) \\ \text{subject to} \quad & \text{PDE Model}(\mathbf{w},\mathbf{q}) \\ & \text{Initial and Boundary Conditions}(\mathbf{w},\mathbf{q}) \\ & \text{State and Control Constraints}(\mathbf{w},\mathbf{q}) \end{aligned} \tag{4.1}$$

where the PDE model is the system of partial differential equations describing the fluid dynamical process (1.49)-(1.53), which include gas-phase chemistry (1.21). The initial and boundary conditions are described in Section 1.7 which includes in particular the surface chemistry (1.23)-(1.28). Here, $\mathbf{w}$ denotes the state vector

$$\mathbf{w} = \left(u, p, T, r, Y_1, Y_2, \ldots, Y_{N_{\text{g}}}, \theta_1, \ldots, \theta_{N_{\text{s}}}\right)$$

and $\mathbf{q}$ are the control variables.

One approach used for solving a PDE-constrained optimization problem is to discretize simultaneously the PDEs and to parameterize the controls using finite element, or finite volume, or finite differences, ... The infinite dimensional optimization problem is replaced by a very large, finite dimensional, constrained, usually nonlinear, programming problem (NLP). Then, available methods can be used to solve the NLP. The disadvantage of this approach is that the NLP is very large, especially in our problems having large-scale PDEs as the result of modeling using detailed chemistry with many species. Moreover, numerical methods for large-scale NLPs are currently active research topics and it is still very difficult to solve large NLPs from poor initial guesses.

We take another approach, which allows us to take the advantage of available efficient DAE solvers with adaptive error control strategy. As in Chapter 3, we semi-discretize the PDE using the method of lines on the grid ψ_i, $i = 1, \ldots, N$. This transforms the optimal control problem in a PDE (4.1) to an optimal control problem in a DAE which can be stated as

$$\begin{array}{rl} \min\limits_{w,\mathbf{q}} & \Phi(w, \mathbf{q}) \\ \text{subject to} & \text{DAE Model}(w, \mathbf{q}) \\ & \text{Initial Conditions}(w, \mathbf{q}) \\ & \text{State and Control Constraints}(w, \mathbf{q}), \end{array} \tag{4.2}$$

where the DAE model is described in Chapter 3, Section 3.3, and is repeated here for convenience (using the same notation of Section 3.3)

$$\text{DAE Model}(w, \mathbf{q}) : \begin{cases} A(Q)Q_z^T = F(Q) \\ 0 = \dot{s}_k W_k + J_{k,r}|_{\psi=\psi_N}, \text{ if } 1 \le k \le N_\text{g} \\ 0 = \dot{s}_k, \text{ if } N_\text{g} + 1 \le k \le N_\text{g} + N_\text{s} \\ 0 = u_N \\ 0 = p_N - p_{N-1} \\ 0 = T_N - T_\text{wall} \\ 0 = r_N - r_\text{max} \end{cases} \tag{4.3}$$

Here, the vector of state variables is

$$w = [Q_1, Q_2, \ldots, Q_N, \theta_1, \ldots, \theta_{N_\text{s}}] .$$

The initial conditions are described in Section 3.3 and also repeated here with suitable modification for the optimization problem.

$$\text{Initial Conditions}(w, \mathbf{q}) : \begin{cases} u = u_0(\mathbf{q}) \\ p = p_0(\mathbf{q}) \\ T = T_0(\mathbf{q}) \\ Y_k = Y_{k0}(\mathbf{q}), \ (k = 1, \ldots, N_\text{g}) \end{cases} \quad \text{at } z = 0. \tag{4.4}$$

In our problems, the state and control constraints are mentioned in Section 4.2.

The control vector $\mathbf{q}$ are

$$\text{Control } \mathbf{q}: \begin{cases} T_{\text{wall}}(z) \\ F_{\text{cat/geo}}(z) \\ u_0 \\ T_0 \\ Y_{k0} \ (k = 1, \ldots, N_{\text{g}}). \end{cases}$$

Remark 4.3.1
When the control parameters such as the initial values of the state variables, are included in the control variables, then this is modeled in (4.4). The boundary conditions in the PDE model is coupled in the DAE model. The controls $T_{\text{wall}}(z)$ and $F_{\text{cat/geo}}(z)$ also appear in the boundary conditions.

4.4 Direct approach

To transform the infinite-dimensional optimal control problem (4.2) to a finite dimensional optimization problem, we apply the direct shooting approach. This is accomplished by restricting the control functions to lie in a subspace of function space that is characterized by a finite number of parameters. Figure 4.1 describes the general framework for solving the optimal control problem.

4.4.1 Parameterization of the control functions

The control functions, such as the temperature profile at the wall $T_{\text{wall}}(z)$ or the ratio of catalytic active surface area to geometric surface area $F_{\text{cat/geo}}(z)$, are treated as control functions in the optimal control problem.

Control functions are discretized on an appropriate user-defined grid

$$z_1 = 0 < z_2 < \ldots z_{(n_{\overline{\text{q}}}-1)} < z_{n_{\overline{\text{q}}}} = z_{\max}$$

using any suitable functional basis, and generally approximation $\widehat{\mathbf{q}}^i$ of $\mathbf{q}^i$ can be written as

$$\widehat{\mathbf{q}}^i(z) = \varphi^i(z, \overline{q}^i_j), \quad \overline{q}^i_j \in \mathbb{R}^{n_{\text{cdis}}} \quad (j = 1, 2, \ldots, n_{\overline{\text{q}}}). \tag{4.5}$$

Usually, the controls are approximated by piecewise continuous functions, e.g., piecewise constant or piecewise linear but also other schemes are applicable. The control functions are described by the coefficients in

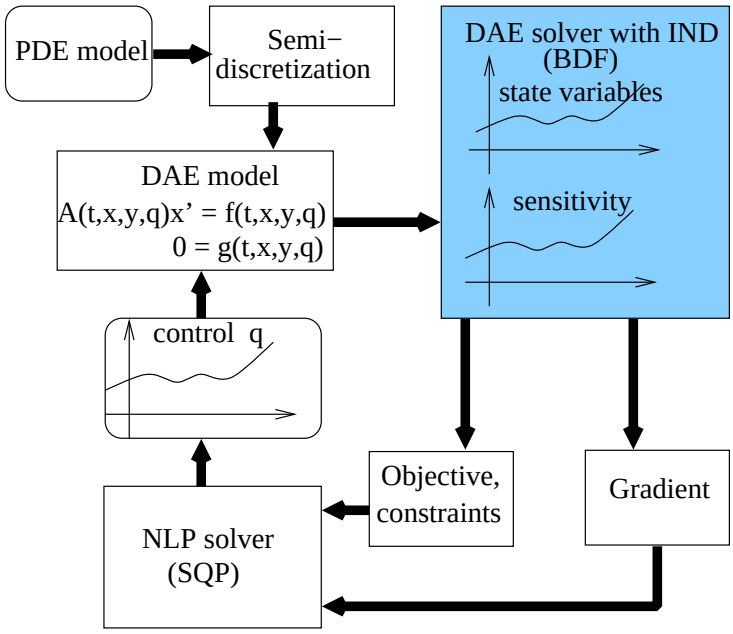

Figure 4.1: General framework for solving the PDE-constrained optimal control problem

these approximation schemes. By this way, the control functions in infinite-dimensional spaces are approximated by their piecewise representation in a finite-dimensional spaces. If the piecewise linear approximation is applied, then, e.g.,

$$\widehat{\mathbf{q}}^i(z) = \overline{q}^i_j + (\overline{q}^i_{j+1} - \overline{q}^i_j)\frac{z - z_j}{z_{j+1} - z_j},$$

in particular,

$$T_{\text{wall}}(z) = T_{\text{wall},\,j} + (T_{\text{wall},\,j+1} - T_{\text{wall},\,j})\frac{z - z_j}{z_{j+1} - z_j}$$
$$F_{\text{wall/geo}}(z) = F_{\text{cat/geo},\,j} + (F_{\text{cat/geo},\,j+1} - F_{\text{cat/geo},\,j})\frac{z - z_j}{z_{j+1} - z_j}.$$

For every control function $\mathbf{q}^i$ $(i = 1, 2\ )$ $n_{\overline{\mathrm{q}}}$ coefficients $\overline{q}^i_j$ $(j = 1, \ldots, n_{\overline{\mathrm{q}}})$ are introduced. Together with the control parameters will be called *optimization variables* q and defined as

$$q = (\overline{q}^1_1, \ldots, \overline{q}^1_{n_{\overline{\mathrm{q}}}}, \overline{q}^2_1, \ldots, \overline{q}^2_{n_{\overline{\mathrm{q}}}}, \mathbf{q}^3, \ldots, \mathbf{q}^m)^T. \tag{4.6}$$

Here, we include all possible controls: $\mathbf{q}^1$ and $\mathbf{q}^2$ are supposed to be $T_{\text{wall}}(z)$ and $F_{\text{cat/geo}}(z)$, $\mathbf{q}^3, \ldots, \mathbf{q}^m$ are supposed to be u_0, T_0, and Yk_0 $(k = 1, \ldots, N_{\mathrm{g}})$.

However, in a particular application only one or some controls are to be optimized, in our implementation the optimization variables are also modified accordingly to include only those controls.

Note that by the approximation (4.5) the bounds on the controls are transformed to bounds on the parameterization coefficients. The optimization variables q will be the variable in the nonlinear constrained optimization problem, which will be discussed in the following sections.

4.4.2 The nonlinear optimization problem

By replacing all the control functions by their approximations, the optimal control problem (4.2) becomes

$$\begin{aligned} \min_{q} \quad & h(q) \\ \text{subject to} \quad & e(q) = 0 \\ & c(q) \leq 0, \end{aligned} \tag{4.7}$$

and the controls q in the DAE model (4.3) and the initial conditions (4.4) are also replaced by their approximations.

For the evaluations of the objective function and the constraints in (4.7) with given initial values and control parameters, we solve the DAE initial value problem (IVP) (4.3).

4.4.3 Optimization methods

To solve constrained nonlinear optimization problems, the method of Sequential Quadratic Programming (SQP) is the most efficient available method. It consists of the solution of a sequence of quadratic optimization problems and can be regarded as a Newton-like method for the optimality conditions of the problem (4.7). We use the implementation SNOPT [56] which employs BFGS updates for the approximation of the Hessian and an Active-Set strategy for the treatment of the inequalities.

As discussed above, a solution of the semi-discretized PDE only makes sense if the algebraic equations (the boundary condition of the PDE) are consistent. As a consequence, our optimization follows the so-called sequential approach solving the algebraic constraints in every iteration. Fortunately, in our case this is not time consuming and the computing time for consistency calculations is negligible compared to the solution time for the whole discretized PDE.

4.5 SQP methods

A basic idea of solving a general NLP problem is to replace it by solving a sequence of appropriate easier subproblems. Such as, to solve a nonlinear equation, one usually solve it by a sequence of linear problems as in Newton-like methods, or for unconstrained optimization, one usually replace it by a sequence of quadratic problems, or homotopy methods. Applying this principle to the nonlinear constrained optimization, sequential quadratic programming (SQP methods) have been developed, which are the most powerful methods we know today for solving nonlinear constrained smooth optimization problems. For detailed surveys on the SQP methods, see e.g., [22], [36], [62], and [80].

4.5.1 SQP algorithm framework

The basic idea of SQP methods is to formulate and solve a quadratic programming (QP) subproblem at each iteration. The QP subproblem is obtained by using the quadratic approximation of the scalar-valued Lagrangian function

$$\ell(q, \lambda, \mu) = h(q) + \lambda^T e(q) + \mu^T c(q),$$

and linearizing the constraints. It is well known that the optimality conditions based on the Lagrangian function ℓ rather than the objective functions, thus the local quadratic model here is of the Lagrangian function. For optimization, quadratic models are chosen instead of linear ones because in general linear models do not reflect the nonlinearity of the problem and do not give a good local approximation of the problems, and linear functions are unbounded. At the k-th iteration, given q_k, an approximation of the solution, and λ_k and μ_k, an approximation of the Lagrangian multipliers, and B_k, an approximation of the Hessian H_k of the Lagrangian function, then the QP subproblem is as follows.

$$\begin{array}{rl} \min\limits_{\Delta q_k} & \nabla_q \ell(q_k)^T \Delta q_k + \frac{1}{2} \Delta {q_k}^T B_k \Delta q_k \\ \text{subject to} & c(q_k) + \nabla c(q_k)^T \Delta q_k = 0 \\ & e(q_k) + \nabla e(q_k)^T \Delta q_k \leq 0. \end{array}$$

Another form of the quadratic subproblem, which is most often used in practice, is

$$\begin{array}{rl} \min\limits_{\Delta q_k} & \nabla_q h(q_k)^T \Delta q_k + \frac{1}{2} \Delta {q_k}^T B_k \Delta q_k \\ \text{subject to} & c(q_k) + \nabla c(q_k)^T \Delta q_k = 0 \\ & e(q_k) + \nabla e(q_k)^T \Delta q_k \leq 0. \end{array}$$

These two forms of the quadratic subproblem are equivalent for problems with only equality constraints, and in general are not equivalent in the inequality constrained cases. However, if the current estimate μ_k is zero for all *inactive* constraints, then it is easily to verify that

$$\nabla_q \ell(q_k)^T \Delta q_k = \nabla_q h(q_k)^T \Delta q_k$$

for all Δq_k satisfying the linearized *active* constraints, thus the two forms are equivalent in this case. A general framework of a SQP algorithm is presented in the following.

Algorithm 4.5.1 (SQP algorithm framework)

Input: Initial guesses for q_0, λ_0, μ_0, B_0, $k = 0$

Output: Approximate solution q^*, λ^*, μ^*

1. *Form and solve (QP) subproblem to obtain $(\Delta q_k, \Delta\lambda_k, \Delta\mu_k)$:*

$$\begin{array}{rl} \min\limits_{\Delta q_k} & \nabla h(q_k)^T \Delta q_k + \frac{1}{2} \Delta q_k{}^T B_k \Delta q_k \\ \textit{subject to} & \nabla e(q_k)^T \Delta q_k + e(q_k) = 0 \\ & \nabla c(q_k)^T \Delta q_k + c(q_k) \leq 0 \end{array}$$

2. *Choose step-length α by a line-search method or a trust region method.*

3. *Compute new estimate*

$$\begin{aligned} q_{k+1} &= q_k + \alpha \Delta q_k \\ \lambda_{k+1} &= \lambda_k + \alpha \Delta \lambda_k \\ \mu_{k+1} &= \mu_k + \alpha \Delta \mu_k \end{aligned}$$

4. *Convergence test: Stop if a convergence criterion is met.*

5. *Compute B_{k+1}*

6. *Set $k = k + 1$, goto 1.*

Note that in the step 1 in Algorithm 4.5.1, in addition to the optimal solution Δq_k, we also obtain the optimal multipliers of the (QP), which are denoted by $\lambda_k^{(\mathrm{qp})}$ and $\mu_k^{(\mathrm{qp})}$. The updates $\Delta\lambda_k$ and $\Delta\mu_k$ are computed as

$$\begin{aligned} \Delta\lambda_k &= \lambda_k^{(\mathrm{qp})} - \lambda_k \\ \Delta\mu_k &= \mu_k^{(\mathrm{qp})} - \mu_k. \end{aligned}$$

Indeed, with this setting of the Lagrangian multipliers we take the optimal multipliers of the (QP) as the estimate multipliers for the original NLP.

The step 2 in Algorithm 4.5.1 is to ensure the *global convergence* of the algorithm. This is usually done with a *merit function* ϕ, whose reduction implies progress towards a solution. A typical merit function is

$$\phi(q,\eta) = h(q) + \eta\left(\sum_{i=1}^{m}|e_i(q)| + \sum_{j=1}^{l}|\max(0,c_j(q))|\right),$$

which is usually known as the l_1 *exact penalty function*.

4.5.2 Hessian approximations

There are two approaches for approximation of the Hessian matrix H_k: *full Hessian* approximation and *reduced Hessian* approximation. Natural methods for approximation of Hessian are computed analytically such as by automatic differentiation, or by finite differences. Alternatively, *scant* approximations can be used as in the unconstrained optimization. Two typical updating schemes of this class are the PSB formula and BFGS formula

The rank-two PSB update formula for the constrained optimization is as

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)s_k^T + s_k(y - B_k s)^T}{s_k^T s_k} - \frac{(y - B_k s_k)^T s_k}{(s_k^T s_k)^2} s_k s_k^T,$$

where

$$s_k = q_{k+1} - q_k$$

and

$$y_k = \nabla_q \ell(q_{k+1}, \lambda_{k+1}, \mu_{k+1}) - \nabla_q \ell(q_k, \lambda_k, \mu_k).$$

Similarly to the unconstrained case, the rank-two BFGS update formula for the constrained case is

$$B_{k+1} = B_k + \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}.$$

Note that with PSB update the matrices B_k are not necessarily positive definite but with the BFGS update, the matrix B_{k+1} is positive definite if $y_k^T s_k > 0$ and B_k is positive definite, this condition is satisfied if the Hessian of Lagrangian is positive definite. However, if the condition $y_k^T s_k > 0$ is not satisfied, this could be the case for constrained optimization, then the positive definite property of B_{k+1} cannot ensure. In order to maintain the

positive definite property of B_{k+1}, the following modified BFGS update (see [98]) can be used

$$B_{k+1} = B_k + \frac{r_k r_k^T}{s_k^T r_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k},$$

where

$$r_k = \theta_k y_k + (1 - \theta_k) B_k s_k, \quad 0 < \theta_k \leq 1,$$

and

$$\theta_k = \begin{cases} 1 & \text{if } s_k^T y_k \geq \epsilon_\theta s_k^T B_k s_k \\ \dfrac{(1-\epsilon_\theta) s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k} & \text{otherwise} \end{cases}$$

and $\epsilon_\theta \in [0.1, 0.2]$.

4.5.3 Convergence of the methods

In the local convergence domain, where the initial guess is sufficiently close to a solution, the active set of the QP subproblem will have the same active set as the NLP. Thus, in the following, only equality-constrained problem is studied, in particular, we consider

$$\begin{array}{rl} \min & h(q) \\ \text{subject to} & e(q) = 0. \end{array} \tag{4.8}$$

The Karush-Kuhn-Tucker optimality conditions for the equality-constrained problem (4.8) are given by

$$\begin{array}{rcl} \nabla_q \ell(q, \mu) = \nabla_q h(q) + \nabla e(q) \mu & = & 0 \\ e(q) & = & 0. \end{array} \tag{4.9}$$

Applying Newton's method to Equation (4.9), we obtain the following iteration scheme

$$\begin{pmatrix} \nabla_q^2 \ell(q_k, \mu_k) & \nabla e(q_k) \\ \nabla e(q_k)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta q_k \\ \Delta \mu_k \end{pmatrix} = - \begin{pmatrix} \nabla_q \ell(q_k, \mu_k) \\ e(q_k) \end{pmatrix} \tag{4.10}$$

Substituting

$$\Delta \mu_k = \mu_{k+1} - \mu_k, \ \nabla \ell(q_k, \mu_k) = \nabla h_k + \nabla e_k \mu_k$$

into (4.10), we obtain

$$\begin{pmatrix} \nabla_q^2 \ell(q_k, \mu_k) & \nabla e(q_k) \\ \nabla e(q_k)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta q_k \\ \mu_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla_q h(q_k, \mu_k) \\ e(q_k) \end{pmatrix} \tag{4.11}$$

Now consider the equality-constrained (QP) subproblem

$$\begin{array}{rl} \min\limits_{\Delta q_k} & \nabla_q h(q_k)^T \Delta q_k + \frac{1}{2} \Delta q_k^T \nabla_q^2 \ell_k \Delta q_k^T \\ \text{subject to} & e(q_k) + \nabla e(q_k)^T \Delta q_k = 0, \end{array} \tag{4.12}$$

the first order conditions for this problem is

$$\begin{array}{rcl} \nabla_q^2 \ell_k \Delta q_k + \nabla_q h(q_k) + \nabla e(q_k) \mu^{\text{qp}} & = & 0 \\ \nabla e(q_k)^T \Delta q_k & = & -e(q_k), \end{array} \tag{4.13}$$

which can be rewritten as

$$\begin{pmatrix} \nabla_q^2 \ell(q_k, \mu_k) & \nabla e(q_k) \\ \nabla e(q_k)^T & 0 \end{pmatrix} \begin{pmatrix} \Delta q_k \\ \mu^{\text{qp}} \end{pmatrix} = - \begin{pmatrix} \nabla_q h(q_k, \mu_k) \\ e(q_k) \end{pmatrix}. \tag{4.14}$$

This is the same as (4.11) with μ^{qp} replaced by μ_{k+1}. This means that the solution of the QP subproblem is exactly the solution of one step Newton iteration. This result can be formulated formally as the following theorem [91].

Theorem 4.5.1 (Local Convergence of SQP with exact Hessian) *Suppose that*

(a) $h(q)$ and $e(q)$ are twice differentiable, with Lipschitz continuous second derivatives in a neighborhood of (q^*, μ^*),

(b) At the solution point q^* with optimal Lagrange multipliers m^*, the constraint Jacobian $\nabla e(q^*)^T$ has full row rank, and the Hessian of the Lagrangian $\nabla_q^2 \ell(q^*, \mu^*)$ is positive definite on the tangent space of the constraints.

Then if q_0 and μ_0 are sufficiently close to q^* and μ^*, the pair (q_k, μ_k) generated by the SQP Algorithm 4.5.1 with H_k defined as the Hessian of the Lagrangian and the step-length $\alpha = 1$ converge quadratically to (q^*, μ^*).

For the SQP methods using a quasi-Newton approximation, the following result is obtained [23].

Theorem 4.5.2 (Convergence of SQP with Hessian Approximation) *Suppose that*

(a) *At the solution point q^* with optimal Lagrange multipliers μ^*, the constraint Jacobian $\nabla e(q^*)^T$ has full row rank, and the Hessian of the Lagrangian $\nabla_q^2 \ell(q^*, \mu^*)$ is positive definite on the tangent space of the constraints.*

(b) *The sequence $\{q_k\}$ generated by the Algorithm 4.5.1 with quasi-Newton approximate Hessian B_k converges to q^*.*

Then the sequence $\{q_k\}$ converges superlinearly if and only if the Hessian approximation B_k satisfies

$$\lim_{k\to\infty} \frac{\|P_k(B_k - H_*)(q_{k+1} - q_k)\|}{\|(q_{k+1} - q_k)\|} = 0,$$

where $P_k = I - \nabla e_k(\nabla e_k^T \nabla e_k)^{-1} \nabla e_k^T$ and $H_* = \nabla^2 \ell_q(q^*, \mu^*)$.

4.6 Computation of derivatives

As we discuss in previous sections, the solution of the optimization problem (4.7) by the SQP method requires the solution of the IVP (4.3) and the derivatives of the objective function and the constraints with respect to the optimization variables. In our case, this is somewhat intricate because these functions are implicitly defined from the solution of the DAE system (4.3) derived from the semi-discretization of the PDE.

Efficient methods for the solution of the IVP are discussed in Chapters 2 and 3, which include fast methods for computing the derivatives and scaling techniques to improve accuracy for the solution of the IVP.

The derivatives of the objective function and the constraints with respect to the optimization variables q are obtained by applying the chain rule. For example,

$$\frac{\mathrm{d}h(q)}{\mathrm{d}q} = \frac{\partial \Phi}{\partial w}\frac{\partial w}{\partial q} + \frac{\partial \Phi}{\partial q}.$$

This in turn requires the derivatives of the objective and the constraint functions with respect to the state variables w and the optimization variables q, $\partial\Phi/\partial w$ and $\partial\Phi/\partial q$, and the derivatives of the state variables with respect to the optimization variables $\partial w/\partial q$.

The derivatives of the state variables with respect to the optimization variables $\partial w/\partial q$ are the derivatives of the solution of the DAE (4.3) with respect to the optimization variables q, which are considered as *parameters*

of the DAE (4.3). The derivatives are sometimes also called *sensitivities*. The DAE (4.3) with parameters can be written as

$$\begin{aligned} B(t,x,y,q)\dot{x} &= f(t,x,y,q) \\ 0 &= g(t,x,y,q) \\ x(t_0) &= x_0. \end{aligned} \tag{4.15}$$

Here, (x, y) denotes w and t denotes z in (4.3).

The derivatives of the solution of the DAE with respect to the optimization variables q can be computed using finite differences

$$\frac{\mathrm{d}w(z,q)}{\mathrm{d}q_i} = \frac{w(z, q+\eta e_i) - w(z,q)}{\eta} + \epsilon, \quad \epsilon = O(\eta)$$

where e_i is a specified direction and η is an appropriate value. This mean that we need to solve the DAE $(n_v + 1)$ times for computing the required derivatives $\mathrm{d}w/\mathrm{d}q$, using different directional vectors e_i. Moreover, the computed numerical solution is the output of an integrator. If the integrator uses automatic step size and order control strategy, which is usually done in modern integrators for efficiency, then the output is generally a discontinuous or undifferentiable function of the input, i.e., the optimization variables, and has the staircase-like shape, i.e., piecewise constant, and the derivatives are piecewise zero. For a given integration tolerance TOL, the best accuracy ϵ one can expect is

$$\epsilon = O(\sqrt{\mathrm{TOL}}) \quad \text{if } \eta = \sqrt{\mathrm{TOL}}.$$

Here we assume that the order of q is one. If components of v are at different orders, then one can choose $\eta_i = \sqrt{\mathrm{TOL}} \times \max(|q(i)|, \mathrm{ATOL}(i))$. It means that even we employ high accuracy integration only low accuracy derivatives are obtained. In particular for our problems as mentioned in Chapters 2 and 3, it is very difficult, if not impossible, to integrate the DAE with a small integration tolerance TOL. This approach is also referred to as *External Numerical Differentiation (END)*.

Another reliable and efficient approach introduced by Bock [17] is based on the concept of *Internal Numerical Differentiation (IND)*. The basic idea of IND is to calculate the "exact" derivative of the approximate solution of the IVP. Here, we approximate the solution of the IVP by the BDF-discretization scheme (2.16). We consider the discretization as a mapping of the parameters to the discretized solution trajectory and differentiate this mapping by applying the chain rule. According to the implicit function theorem, the mapping is continuous and differentiable with respect to the parameters, if we freeze the adaptive grid and all other adaptive decisions made by the integrator.

BDF-discretization for DAE

As in Chapter 2, applying the BDF formula (2.12), which is derived in Section 2.3 and are repeated here for convenience

$$\dot{x}_{m+1} = -\frac{1}{h_{m+1}} \left(\alpha_0^{(m+1)} x_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i} \right)$$

to discretize (4.15) at the $(m+1)$-th step, similar to Section 2.3, we obtain the following nonlinear equations

$$\begin{array}{rcl} B^{m+1} \left(\alpha_0^{(m+1)} x_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i} \right) + h_{m+1} f_{m+1} & = & 0 \\ g^{m+1} & = & 0, \end{array} \tag{4.16}$$

where B_{m+1}, f_{m+1} and g_{m+1} are now defined as

$$\begin{array}{rcl} B^{m+1} & = & B(t_{m+1}, x_{m+1}, y_{m+1}, q) \\ f^{m+1} & = & f(t_{m+1}, x_{m+1}, y_{m+1}, q) \\ g^{m+1} & = & g(t_{m+1}, x_{m+1}, y_{m+1}, q). \end{array}$$

The Jacobian matrix, also called *iteration matrix* of the DAE, for the nonlinear equations (4.16) is

$$J = \begin{pmatrix} \alpha_0^{(m+1)} B^{m+1} + B_x^{m+1} c_{m+1} + h_{m+1} f_y & B_y^{m+1} c_{m+1} + h_{m+1} f_y \\ g_x^{m+1} & g_y^{m+1} \end{pmatrix} \tag{4.17}$$

where

$$c_{m+1} = \alpha_0^{(m+1)} x_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i}.$$

Variational DAE

Differentiating the IVP (4.15) with respect to parameter q, we obtain the following *variational DAE* (VDAE), sometimes also called *sensitivity equations* of (4.15),

$$\begin{array}{rcl} \left(\dfrac{\partial B}{\partial w} w^q + \dfrac{\partial B}{\partial q} \right) \dot{x} + B \dot{x}^q & = & \dfrac{\partial f}{\partial w} w^q + \dfrac{\partial f}{\partial q} \\ 0 & = & \dfrac{\partial g}{\partial w} w^q + \dfrac{\partial g}{\partial q} \end{array}$$

where $w = (x, y)$, $w^q = (x^q, y^q)$, $x^q = \partial x / \partial q$ and $y^q = \partial y / \partial q$, and B, f, and g are evaluated at (t, x, y).

By denoting

$$\begin{aligned} B_w &= \frac{\partial B}{\partial w}, \quad B_q &= \frac{\partial B}{\partial q} \\ f_w &= \frac{\partial f}{\partial w}, \quad f_q &= \frac{\partial f}{\partial q} \\ g_w &= \frac{\partial g}{\partial w}, \quad g_q &= \frac{\partial g}{\partial q}, \end{aligned}$$

the variational DAE can be rewritten as

$$\begin{aligned} B\dot{x}^q &= f_w w^q + f_q - (B_w w^q + B_q)\dot{x} \\ 0 &= g_w w^q + g_q. \end{aligned} \tag{4.18}$$

Note that the variational DAE (4.18) is linear, and the derivative of the solution of (4.15) satisfies the variational DAE (4.18) with $w_0^q = \partial w_0/\partial q = (\partial x_0/\partial q, \partial y_0/\partial q)$.

BDF-discretization for the variational DAE

Applying the BDF-formula (2.12) for the variable $\dot{x}^q$ to (4.18), i.e.,

$$\dot{x}^q_{m+1} = -\frac{1}{h_{m+1}} \left(\alpha_0^{(m+1)} x^q_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x^q_{m+1-i} \right)$$

we obtain

$$\begin{aligned} B^{m+1}(\alpha_0^{(m+1)} x^q_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x^q_{m+1-i}) & \\ +(B_w^{m+1} w^q_{m+1} + B_q^{m+1})(\alpha_0^{(m+1)} x_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i}) & \\ +h_{m+1}(f_w^{m+1} w^q_{m+1} + f_q^{m+1}) &= 0 \\ g_w^{m+1} w^q_{m+1} + g_q^{m+1} &= 0 \end{aligned} \tag{4.19}$$

Here we discretize the variational DAE (4.18) using the same the step-size and order as in the nominal trajectory, i.e., h and α_i are the same as in the nominal trajectory.

Note that the equation (4.19) is linear in the unknowns w^q_{m+1}, and its iteration matrix is

$$J = \begin{pmatrix} \alpha_0^{(m+1)} B^{m+1} + B_x^{m+1} c_{m+1} + h_{m+1} f_y & B_y^{m+1} c_{m+1} + h_{m+1} f_y \\ g_x^{m+1} & g_y^{m+1} \end{pmatrix} \tag{4.20}$$

where

$$c_{m+1} = \alpha_0^{(m+1)} x_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i}.$$

Differentiation of the BDF-discretization for the DAE

Differentiating the discretized DAE system (4.16) with respect to the parameter q we obtain

$$\begin{aligned} B^{m+1}(\alpha_0^{(m+1)} x_{m+1}^q + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i}^q) & \\ +(B_w^{m+1} w_{m+1}^q + B_q^{m+1})(\alpha_0^{(m+1)} x_{m+1} + \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i}) & \\ +h_{m+1}(f_w^{m+1} w_{m+1}^q + f_q^{m+1}) &= 0 \\ g_w^{m+1} w_{m+1}^q + g_q^{m+1} &= 0. \end{aligned} \tag{4.21}$$

The equations (4.19) and (4.21) are the same. It means that when the VDAE (4.18) is solved using the same BDF-discretization scheme (the same step-size and order) as in the nominal trajectory (4.15), then the computed solution for the VDAE is the exact derivative of the nonimal trajectory approximation. Moreover, the variational DAE (4.18) has the same iteration matrix as the original system (4.15). There are three major methods solving the sensitivity equations in the framework of IND: *staggered direct method*, *simultaneous corrector method*, and *staggered corrector method*.

4.6.1 The staggered direct method

There are two major stages at each integration step in this method (see e.g., [28], [78] and [10]). First the nominal trajectory of the DAE system are computed by solving the nonlinear corrector equations. Once an acceptable solution is obtained, then the solution of the linear sensitivity equation (4.19) is computed through the direct solution of the linear system:

$$J \begin{pmatrix} x_{m+1}^q \\ y_{m+1}^q \end{pmatrix} = - \begin{pmatrix} B^{m+1} \beta_{m+1}^q + B_q^{m+1} c_{m+1} + h_{m+1} f_q^{m+1} \\ g_q^{m+1} \end{pmatrix}, \tag{4.22}$$

where

$$\beta_{m+1}^q = \sum_{i=1}^{k} \alpha_i^{(m+1)} x_{m+1-i}^q,$$

and J is defined as in (4.20).

Although the iteration matrix J is the same as in the nominal trajectory, we need to re-evaluate the partial derivatives B_w, f_w, g_w, B_q, f_q, and g_q after solving the nominal trajectory to correctly define the VDAE (4.18) at the newly computed value of the state variables. Thus, J also needs to be recomputed and factorized. There is a alternative approach, which do not recompute and factorize J, is discussed later.

4.6.2 The simultaneous corrector method

This method solves the DAE and the VDAE simultaneously. For easy of presentation, we rewrite the DAE (4.15) in an implicit form as

$$F(t, w, \dot{w}, q) = 0,$$

where $w = (x, y)$ and

$$F(t, w, \dot{w}, q) = \begin{pmatrix} B(t, x, y, q)\dot{x} - f(t, x, y, q) \\ g(t, x, y, q) \end{pmatrix},$$

thus the VDAE can be written as

$$\frac{\partial F}{\partial w}\frac{\partial w}{\partial q} + \frac{\partial F}{\partial \dot{w}}\frac{\partial \dot{w}}{\partial q} + \frac{\partial F}{\partial q} = 0.$$

Define $W = (w, w_1^q, w_2^q, \ldots, w_{n_q}^q)^T$ with $w_i^q = \partial w/\partial q_i$ and

$$\mathbf{F} = \left(F(t, w, \dot{w}, q), \frac{\partial F}{\partial w} w_1^q + \frac{\partial F}{\partial \dot{w}} \dot{w}_1^q + \frac{\partial F}{\partial q_1}, \ldots, \frac{\partial F}{\partial w} w_{n_q}^q + \frac{\partial F}{\partial \dot{w}} \dot{w}_{n_q}^q + \frac{\partial F}{\partial q_{n_q}} \right)^T,$$

the combined system of the DAE and its VDAE can be rewritten as

$$\begin{aligned} &\mathbf{F}(t, W, \dot{W}, q) = 0 \\ &W(0) = \left(w_0, \frac{\partial w_0}{\partial q_1}, \ldots, \frac{\partial w_0}{\partial q_{n_q}} \right)^T. \end{aligned} \tag{4.23}$$

Discretization of the system (4.23) using the kth order BDF formula (2.12) for $\dot{W}$ yields the following nonlinear equation system

$$G(W_{m+1}) = \mathbf{F}\left(t_{m+1}, W_{m+1}, \frac{-1}{h_{m+1}} \sum_{i=0}^{k} \alpha_i^{m+1} W_{m+1-i}, q \right) = 0, \tag{4.24}$$

which can be solved by Newton's method with the iteration

$$\begin{aligned} \mathbf{J}\Delta W_{m+1}^k &= -G(W_{m+1}^k) \\ W_{m+1}^{k+1} &= W_{m+1}^k + \Delta W_{m+1}^k, \end{aligned}$$

where

$$\mathbf{J} = \begin{pmatrix} J & & & & \\ J_1 & J & & & \\ J_2 & 0 & J & & \\ \vdots & \vdots & \vdots & \ddots & \\ J_{n_q} & 0 & \ldots & 0 & J \end{pmatrix}, \tag{4.25}$$

$$J = \frac{-\alpha_0^{m+1}}{h_{m+1}}\frac{\partial F}{\partial \dot{w}} + \frac{\partial F}{\partial w},\ J_i = \frac{\partial J}{\partial w}w_i^q + \frac{\partial J}{\partial q_i},\ i = (1, \ldots, n_q).$$

The Jacobian **J** can be approximated by its block diagonal in Newton iteration, and the resulting iteration, as shown in [85], is two-step quadratically convergent for the full Newton iteration and convergent for modified Newton iteration. Thus, the Jacobian matrix J can be reused and evaluated and factored when needed. However, to evaluate $G(W_{m+1}^k)$ at each Newton iteration, we need the derivatives $\partial F/\partial w$, $\partial F/\partial \dot{w}$, and $\partial F/\partial q$. These derivatives need to be evaluated at every corrector iteration.

4.6.3 The staggered corrector method

The staggered corrector method [50] is similar to the the staggered direct method. On each integration step, the state variables are solved first, then the Newton iteration is used to solve the linear sensitivity equations instead of solving the linear system directly as in the staggered direct method. Here, a modified Newton method is used for solving the linear equation. The modified Newton iteration for the sensitivity variables is

$$\widetilde{J}(w_{m+1}^{q^{k+1}} - w_{m+1}^{q^k}) = -\left(Jw_{m+1}^{q^k} + \frac{\partial F}{\partial \dot{w}}\beta_{m+1}^q + \frac{\partial F}{\partial q}\right),$$

where

$$\beta_{m+1}^q = \frac{-1}{h_{m+1}}\sum_{i=1}^{k}\alpha_i^{m+1}w_{m+1-i}^q.$$

Here J is the current unfactored Jacobian, and $\widetilde{J}$ is a factored Jacobian from the previous steps. Note that the partial derivatives $\partial F/\partial w$, $\partial F/\partial \dot{w}$, and $\partial F/\partial q$ only need to be computed once per integration step, after the corrector iteration of the nominal trajectory and before the corrector iteration for sensitivity variables.

4.6.4 Comparison of the methods

It is not so easy to give a precise comparison of these methods. Which method is more efficient than others depending on particular problems. Here we give qualitative comparison and analysis, for comparison with numerical experiments see [81].

In the following, the *partial derivatives* are referred to the partial derivatives of the model functions with respect to the state variables and parameters, in particular, $\partial F/\partial w$, $\partial F/\partial \dot{w}$, and $\partial F/\partial q$ for the fully implicit DAE

form, or $\partial B/\partial w$, $\partial f/\partial \dot{w}$, $\partial g/\partial \dot{w}$, $\partial B/\partial p$, $\partial f/\partial p$, and $\partial g/\partial \dot{p}$ for the quasi-linear form. The *corrector iteration* is referred to the iteration process for solving the corrector equations by a modified Newton method.

As mentioned in previous sections, for the staggered direct method the partial derivatives are evaluated at each integration step, and the Jacobian is factored at each integration step. For the simultaneous corrector method, the partial derivatives are evaluated at each corrector iteration and the Jacobian is reused. For the staggered corrector method, the partial derivatives are evaluated at each integration step. Table 4.1 shows qualitative comparison of the methods.

Since the staggered direct method evaluates the partial derivatives and factorizes the Jacobian matrix at each step, the corrector iteration needs only one or two iterations to converge because of the good approximation of the Jacobian, or even it needs least integration steps. In particular, for highly nonlinear problems, where the Jacobian changes significantly in the course of the integration, and the cost for factoring the Jacobian is not too large compared to the cost for evaluating the partial derivatives, this method seems to be more efficient than others. However, our numerical experiments show that the number of integration steps does not change much—only few integration steps—when a very good approximation or an acceptable approximation of Jacobian is used. On the other hand, as the staggered corrector method use the old factorized Jacobian, the corrector iteration could need more iterations to convergence especially for nonlinear problems, and it may take more integration steps thus a higher number of step for solving the linear equations than the staggered direct method. But in total it may need less the number of Jacobian factorizations than the staggered direct method. For the simultaneous corrector method, because the Jacobian $\mathbf{J}$ is only approximated by its block diagonal using the old J, the slow convergence of corrector iteration may occur. Therefore, the number of Newton iterations per integration step may increase. In addition, this method evaluates the partial derivatives at each Newton iteration, thus the total number of partial derivatives could be higher than others. Finally, if the cost of factor Jacobian dominates, this is in particular true for very large scale problems, the staggered corrector method would be more efficient than others. If the linear system of the Newton iteration is solved by iterative methods, then the factorization of the Jacobian is eliminated, thus the staggered direct method may be favored.

	# Eval. derivatives	# Fac. J	# Integ. step	# Iter. per step
Stag. direct	medium$^-$	high	low	low
Stag. corrector	medium$^+$	medium$^-$	medium$^+$	medium
Simul. corrector	high	medium$^+$	high$^-$	high

Table 4.1: Qualitative comparison of the computing sensitivity methods

4.7 Performance comparison of different methods for computation of derivatives

Again as in Section 2.10, to generate the sensitivity equations we use automatic differentiation in addition to the finite difference methods. We apply the same techniques as in Section 2.10 for computation of derivatives. Here, we investigate the performance of these methods applied to the solution of the optimal control prolem. In particular, we compare the performance of the following methods, which are presented in Section 2.10 and are briefly included here for convenience.

- **DFD:** The dense linear solver is used, the derivatives are computed using the forward finite difference and treated as a dense matrix. This mode is an adaptively modification for the DAESOLE code from the DAESOL code.
- **DAD:** The dense linear solver is used, the derivatives are computed using automatic differentiation and treated as a dense matrix.
- **BFT:** The band linear solver is used, the derivatives are computed using the forward finite difference particular for the block tridiagonal matrix.
- **BAT:** The band linear solver is used, the derivatives are computed using automatic differentiation taking the block tridiagonal structure into account.

Three modes DAD, BFT and BAT are newly implemented and coupled with DAESOLE. The optimal control problem is to maximize the efficiency of the conversion of ethane to ethylene, see Section 5.1.4 for detailed description of the problem.

Let us define the **Speedup** to be the ratio between the CPU time for solving the optimal control problem using the standard model in DAESOLE (dense finite differences for computation of derivatives), which is named as

	DFD	DAD	BFT	BAT
Total # model func. (mf) calls	2510526	35893	610984	41698
Total # mf-calls for derivs.	2470856		586763	
Total CPU times for derivs. (secs)	51289	1705	2843	814
Total CPU times (secs)	52309	2461	3352	1553

Table 4.2: Computational statistics of the optimal control problem (conversion of ethane to ethylene) with 12 spatial grid points

	DFD	DAD	BFT	BAT
Total # model func. (mf) calls	3196289	43913	910551	33171
Total # mf-calls for derivs.	3158544		874904	
Total CPU times for derivs. (secs)	109699	3598	5973	893
Total CPU times (secs)	111581	5334	7324	2021

Table 4.3: Computational statistics of the optimal control problem (conversion of ethane to ethylene) with 16 spatial grid points

DFD, and the CPU time for solving the same problem by an other method, e.g., DAD, BFT, or BAT.

Tables 4.2, 4.3 and 4.4 summarize the computational statistics of the optimal control problem with different methods and the number of spatial grid points.

Table 4.5 shows the speedup gained by different methods. It shows that computation of derivatives by automatic differentiation (for solving optimal control problem) always outperforms computation by finite differences. In our problem, the Speedup is quite large because the model functions are complicated, and the cost for evaluating it is expensive. Thus, the computation of derivatives by the finite differences with dense mode takes many model function calls, and the result is that the computation of derivatives takes a lot of times. As the number of the spatial disretization points (Nodes) is large

	DFD	DAD	BFT	BAT
Total # model func. (mf) calls	3995935	46514	677008	40115
Total # mf-calls for derivs.	3957916		650696	
Total CPU times for derivs. (secs)	214686	5961	5936	1380
Total CPU times (secs)	217236	8627	7428	3246

Table 4.4: Computational statistics of the optimal control problem (conversion of ethane to ethylene) with 20 spatial grid points

Nodes	BAT	DAD	BFT
12	33.68	21.25	15.69
16	55.21	20.91	15.23
20	66.92	25.18	29.24

Table 4.5: Speedup gained by different methods applied to the optimal control problem (conversion of ethane to ethylene)

enough (i.e., 20), the speedup by DAD is smaller than by BFT because the cost for evaluation of derivatives also increases in dense mode even compute by DFD or DAD.

Chapter 5

Numerical Results

In this chapter our simulation and optimization software BLAYER$^{\text{sim}}$ and BLAYER$^{\text{opt}}$ are applied to several practical applications. In addition, we make a comparison between the existing DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ with respect to performance and numerical results. In addition, for performance comparison of different methods for simulation, see Chapter 2, Section 2.10 and for optimzation, see Chapter 4, Section 4.7, for numerical results of the scaling method, see Chapter 2, Section 2.7.

In Section 5.1, we present the simulation results obtained by BLAYER$^{\text{sim}}$ of four applications: (1) NO_2 oxidation (2) catalytic partial oxidation of methane (only consider surface reactions) (3) catalytic combustion of methane, and (4) conversion of ethane to ethylene. Section 5.2 is devoted to comparison of the our simulation software BLAYER$^{\text{sim}}$ with the simulation software DETCHEM$^{\text{CHANNEL}}$ with respect to performance and numerical results. In Section 5.3, we present some optimization results obtained by using BLAYER$^{\text{opt}}$ software applying to applications: (a) catalytic combustion of methane and (b) conversion of ethane to ethylene.

In the following, the initial and boundary conditions are the ones given by the user, and these are used to formulate the complete initial and boundary conditions for the numerical problem as discussed in the previous chapters.

5.1 Simulation results

5.1.1 NO_2 oxidation process

A gas mixture flows in a channel with the following setting.

– Channel geometry: the radius $r_{\max} = 2.5 \times 10^{-4}$ [m], the channel length $z_{\max} = 0.01$ [m].

– Initial conditions: the initial mole fraction of each species $X_{NO_2} = 0.10$, $X_{N_2} = 0.90$, other species are absent at inlet. The initial gas temperature is $T_{gas} = 300$ [K], the initial pressure is $p = 10^5$ [Pa], and the initial velocity $u = 1$ [m/s].

– Boundary conditions: the temperature at the wall is $T_{wall} = 1200$ [K].

– Reaction mechanisms: 5 gas-phase species, 4 surface species, 9 surface reactions, and 8 gas-phase reactions. The gas-phase reactions and surface reactions are given in the Appendix.

– Number of grid points in the radial axis: 30.

Figure 5.1 shows the results of simulation. In this figure, the axial velocity, the temperature, and the mass fractions of gas phase species as functions of radial coordinate r and axial coordinate z are shown. In addition, the pressure and surface coverages of surface species as functions of the axial coordinate z are also presented.

5.1.2 Catalytic partial oxidation of methane

A mixture of methane CH_4, hydrogen H_2, oxygen O_2, and nitrogen N_2 enter the channel with the following setting.

– Channel geometry: the radius $r_{max} = 9.0 \times 10^{-4}$ [m], the channel length $z_{max} = 0.5$ [m].

– Initial conditions: the initial mole fraction of each species $X_{CH_4} = 0.03$, $X_{H_2} = 0.05$, $X_{O_2} = 0.19$, and $X_{N_2} = 0.73$, other species are absent at inlet. The initial gas temperature is $T_{gas} = 298$ [K], the initial pressure is $p = 10^5$ [Pa], and the initial velocity is $u = 0.8$ [m/s].

– Boundary conditions: the temperature at the wall is $T_{wall} = 1200$ [K].

– Reaction mechanisms: 7 gas-phase species, 11 surface species, 23 surface reactions. The surface reactions are given in the Appendix, the gas-phase reactions are not considered.

– Number of grid points in the radial axis: 20.

Figures 5.2 and 5.3 show the results of simulation.

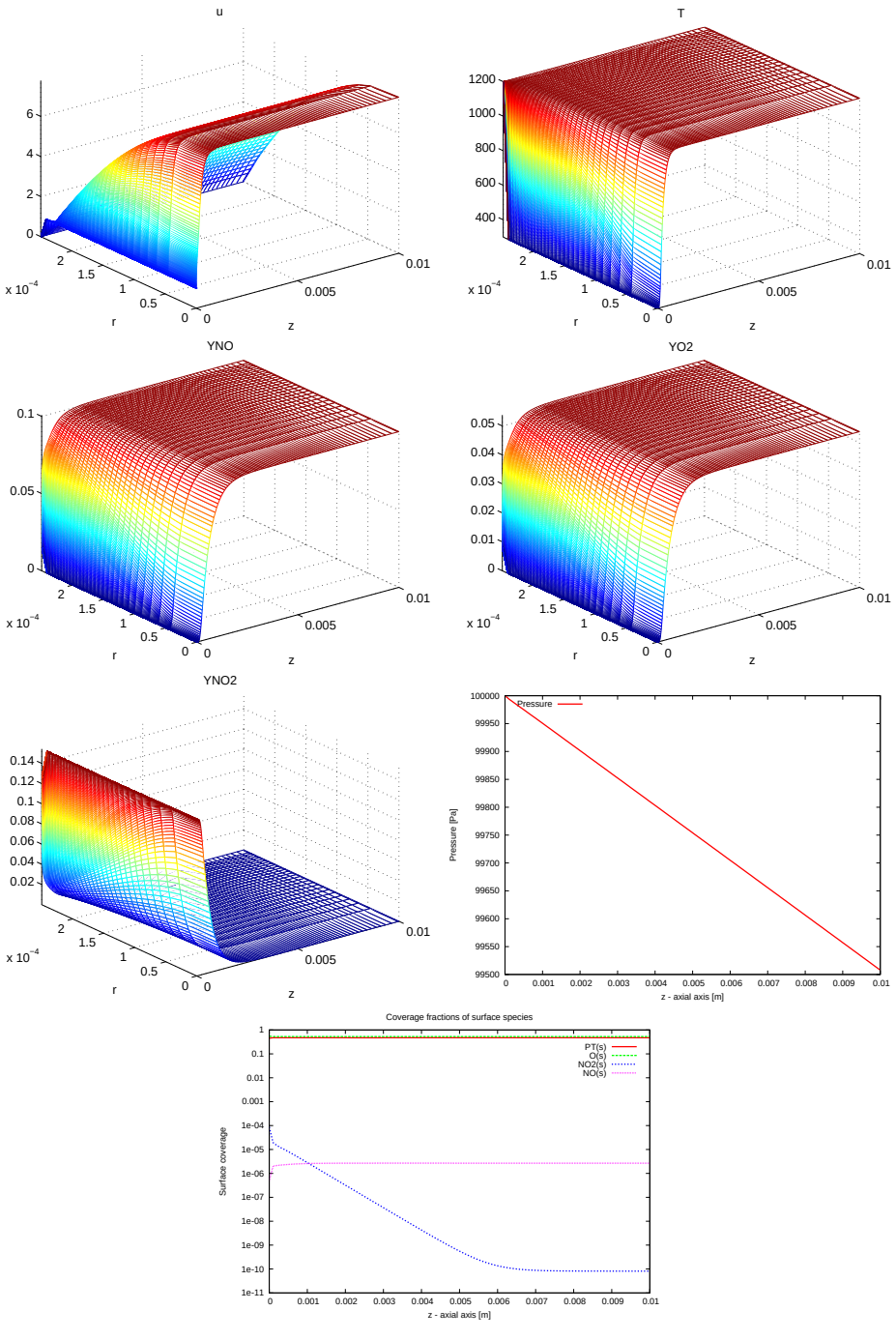

Figure 5.1: Simulation results of NO_2 oxidation

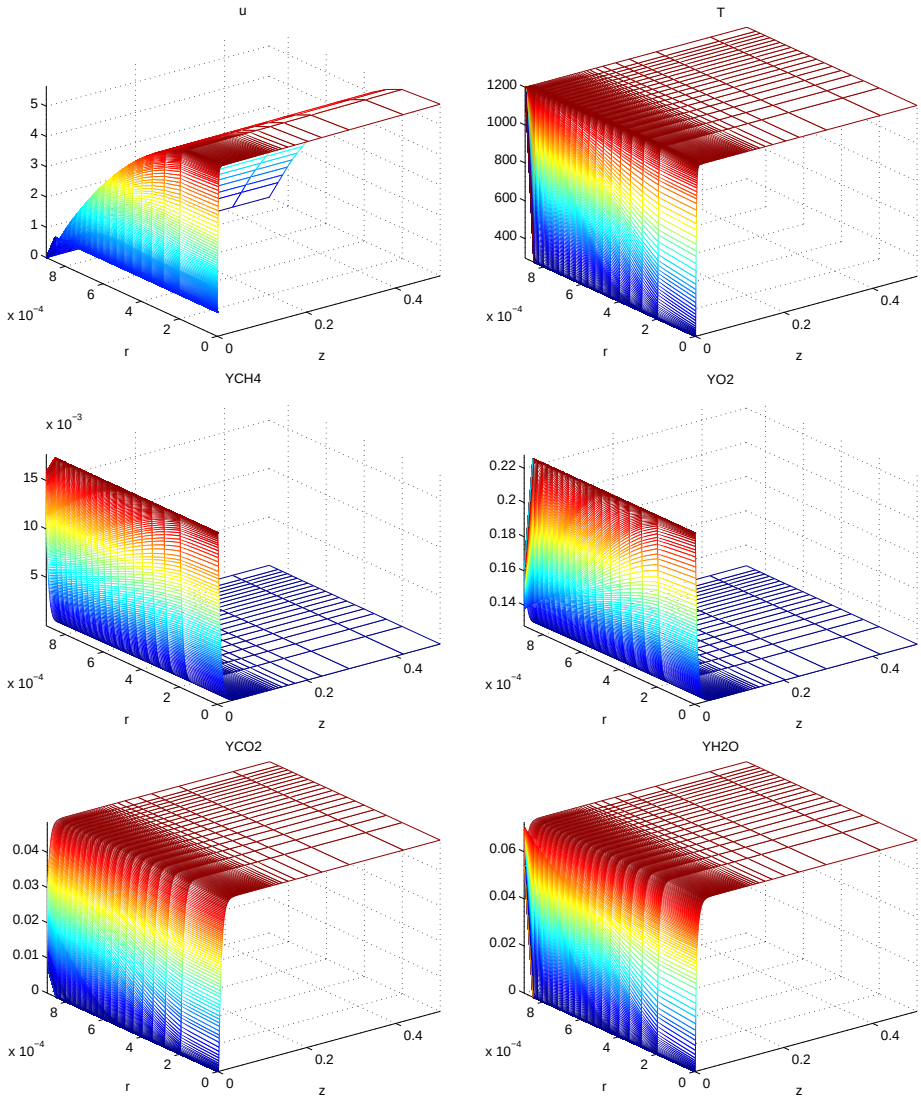

Figure 5.2: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of catalytic partial oxidation of methane (I)

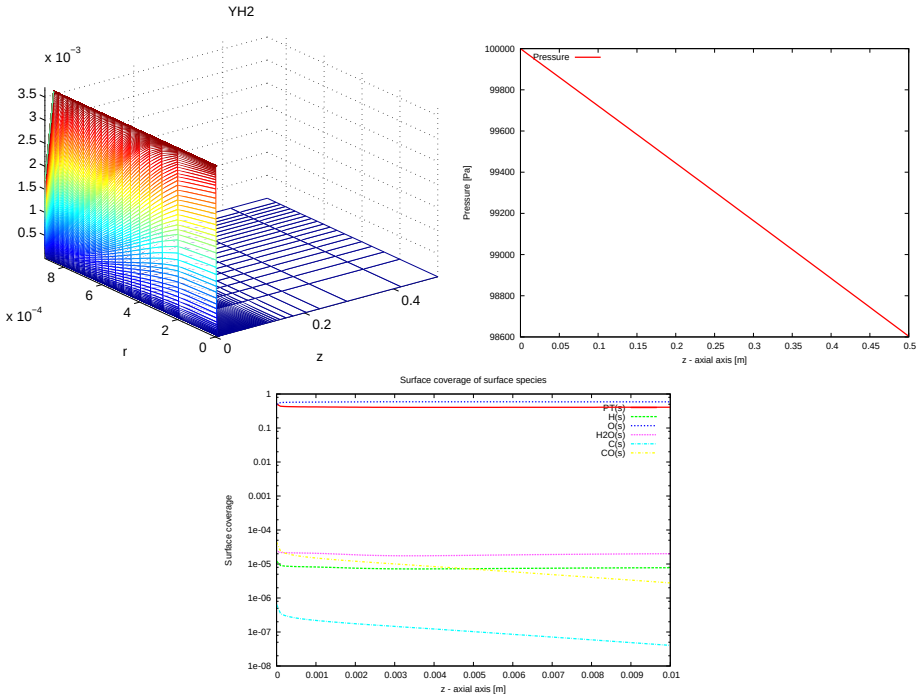

Figure 5.3: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of catalytic partial oxidation of methane (II)

5.1.3 Catalytic combustion of methane

A gas mixture flows in the channel with the following setting.

- Channel geometry: the radius $r_{\max} = 2.5 \times 10^{-4}$ [m], the channel length $z_{\max} = 0.01$ [m].
- Initial conditions: the initial mole fraction of each species $X_{\mathrm{CH_4}} = 0.5$, $X_{\mathrm{O_2}} = 0.3$, and $X_{\mathrm{N_2}} = 0.2$, other species are absent at inlet. The initial gas temperature is $T_{\mathrm{gas}} = 298$ [K], the initial pressure is $p = 1.2 \times 10^5$ [Pa], and the initial velocity is $u = 1$ [m/s].
- Boundary conditions: the temperature at the wall is $T_{\mathrm{wall}} = 1373$ [K].
- Reaction mechanisms: 21 gas-phase species, 11 surface species, 23 surface reactions, and 128 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.
- Number of grid points in the radial axis: 20.

Figures 5.4, 5.5 and 5.6 show the results of simulation.

5.1.4 Conversion of ethane to ethylene

A gas mixture flows in the channel with the following setting.

- Channel geometry: the radius $r_{\max} = 2.5 \times 10^{-4}$ [m], the channel length $z_{\max} = 0.01$ [m].
- Initial conditions: the initial mole fraction of each species $X_{\mathrm{C_2H_6}} = 0.44$, $X_{\mathrm{O_2}} = 0.26$, and $X_{\mathrm{N_2}} = 0.30$, other species are absent at inlet. The initial gas temperature is $T_{\mathrm{gas}} = 650$ [K], the initial pressure is $p = 1.2 \times 10^5$ [Pa], and the initial velocity is $u = 0.5$ [m/s].
- Boundary conditions: the temperature at the wall $T_{\mathrm{wall}} = 1300$ [K].
- Reaction mechanisms: 25 gas-phase species, 20 surface species, 82 surface reactions, and 261 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.
- Number of grid points in the radial axis: 20.

Figures 5.7, 5.8 and 5.9 show the results of simulation.

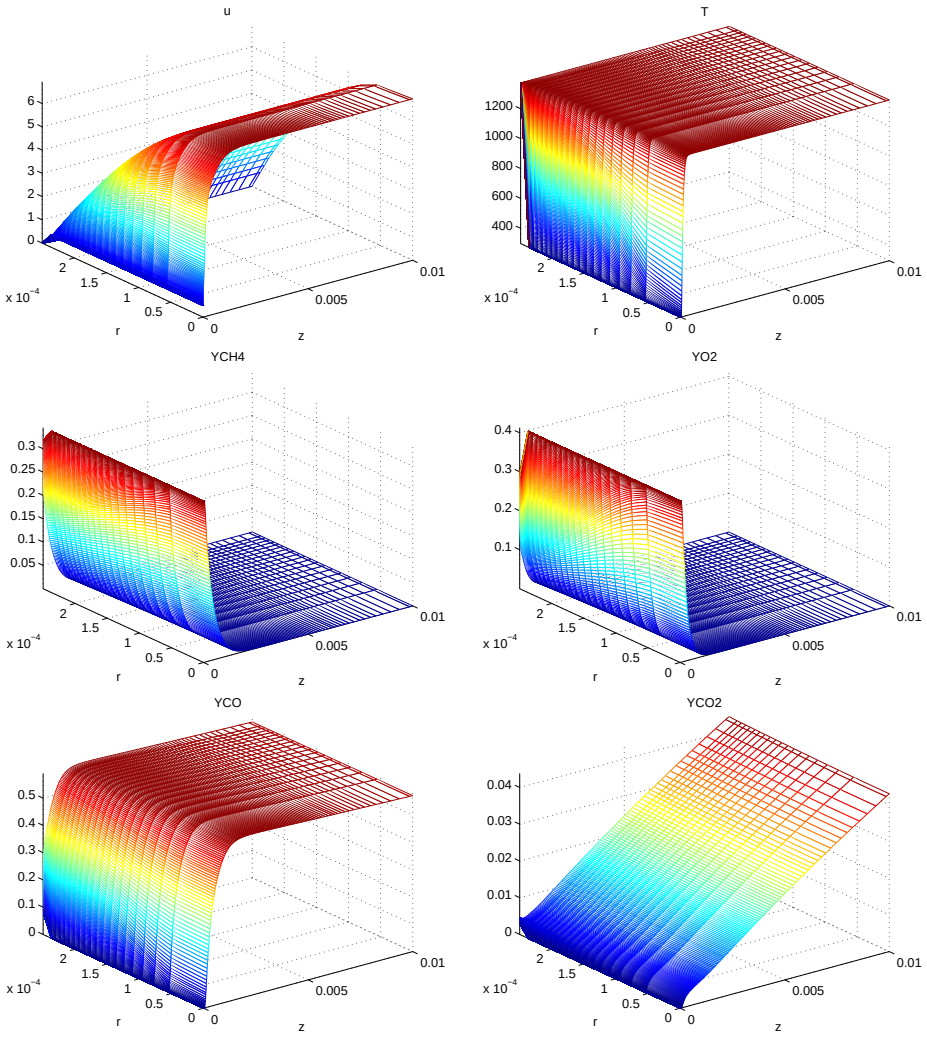

Figure 5.4: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of catalytic combustion of methane (I)

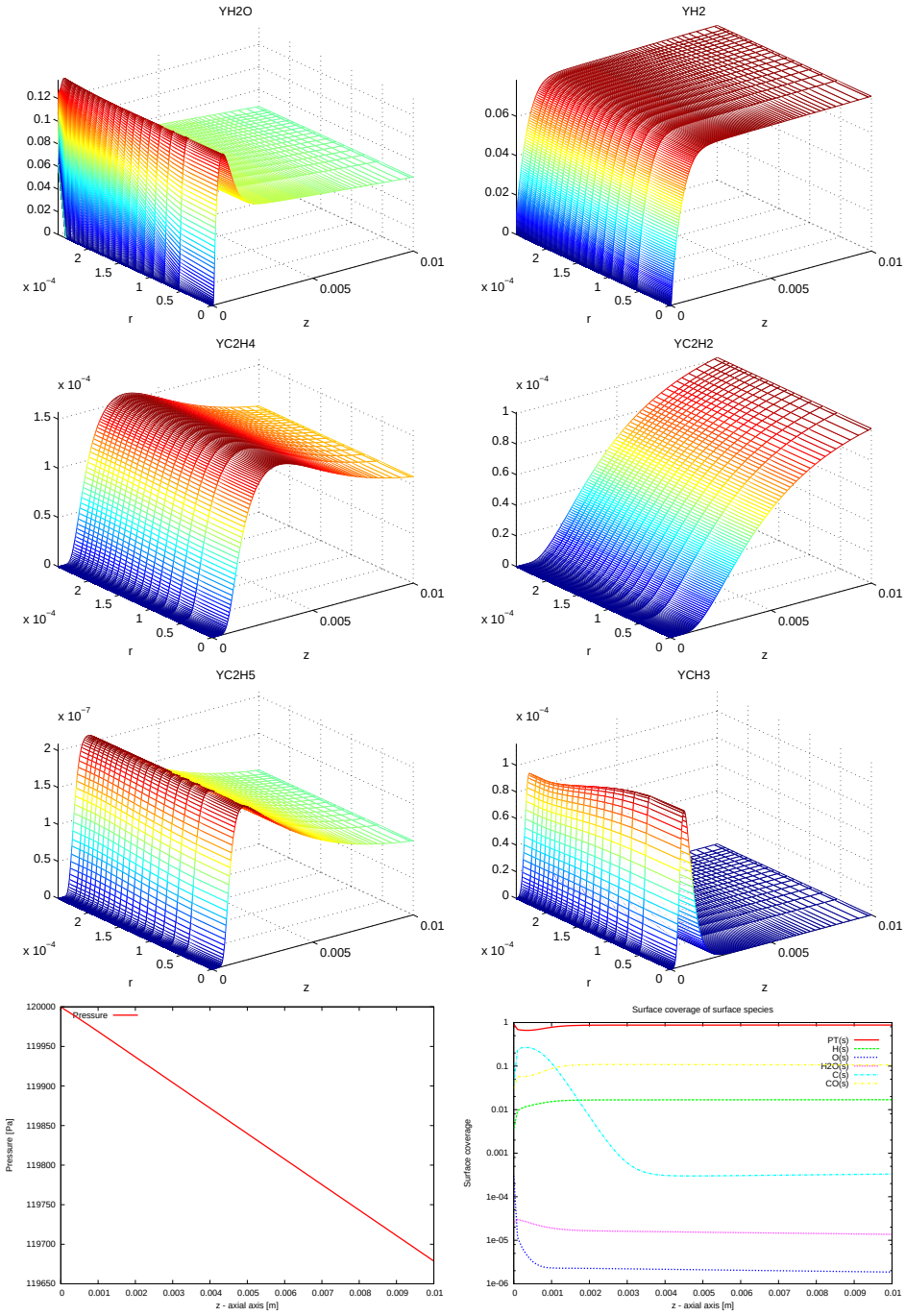

Figure 5.5: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of catalytic combustion of methane (II)

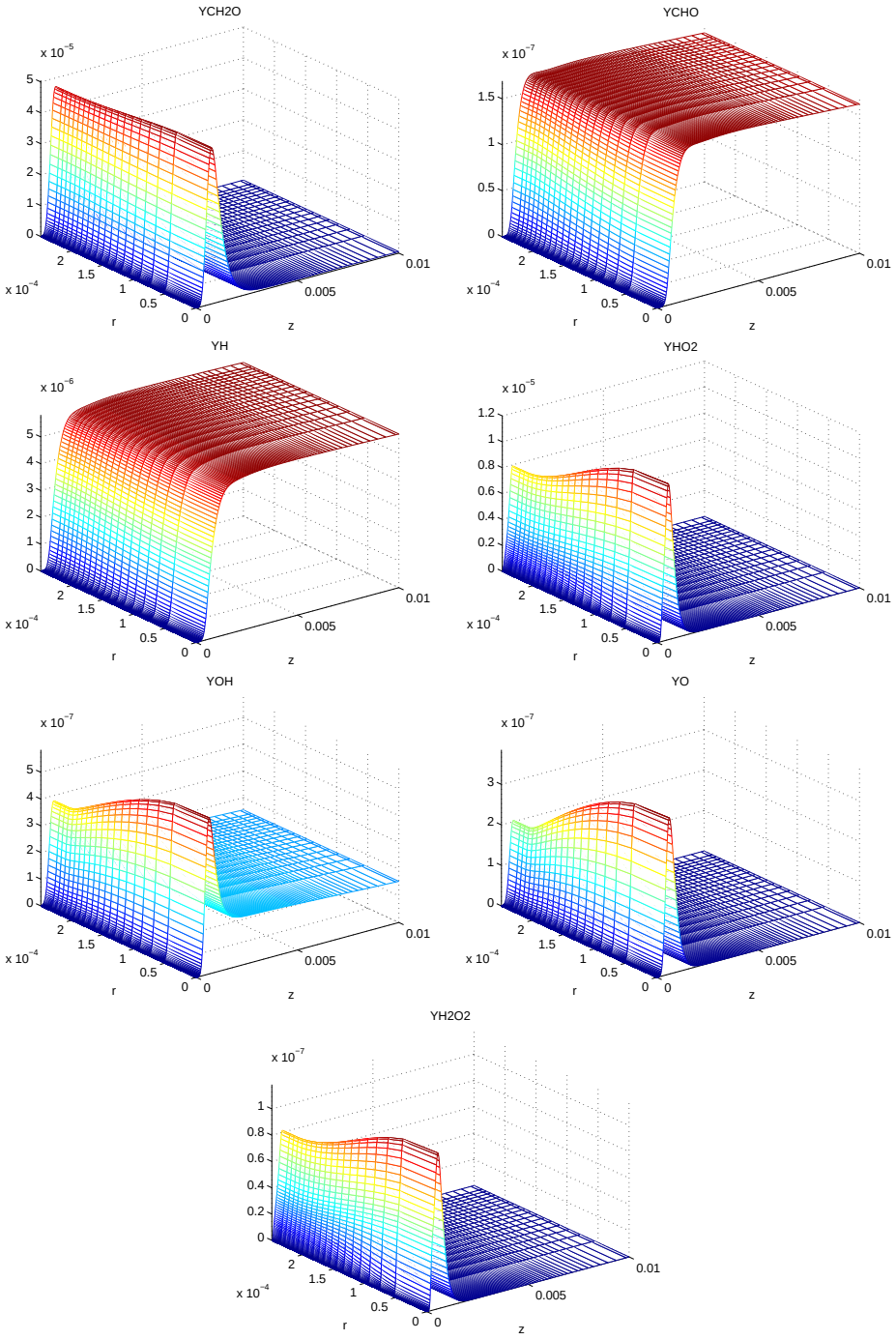

Figure 5.6: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of catalytic combustion of methane (III).

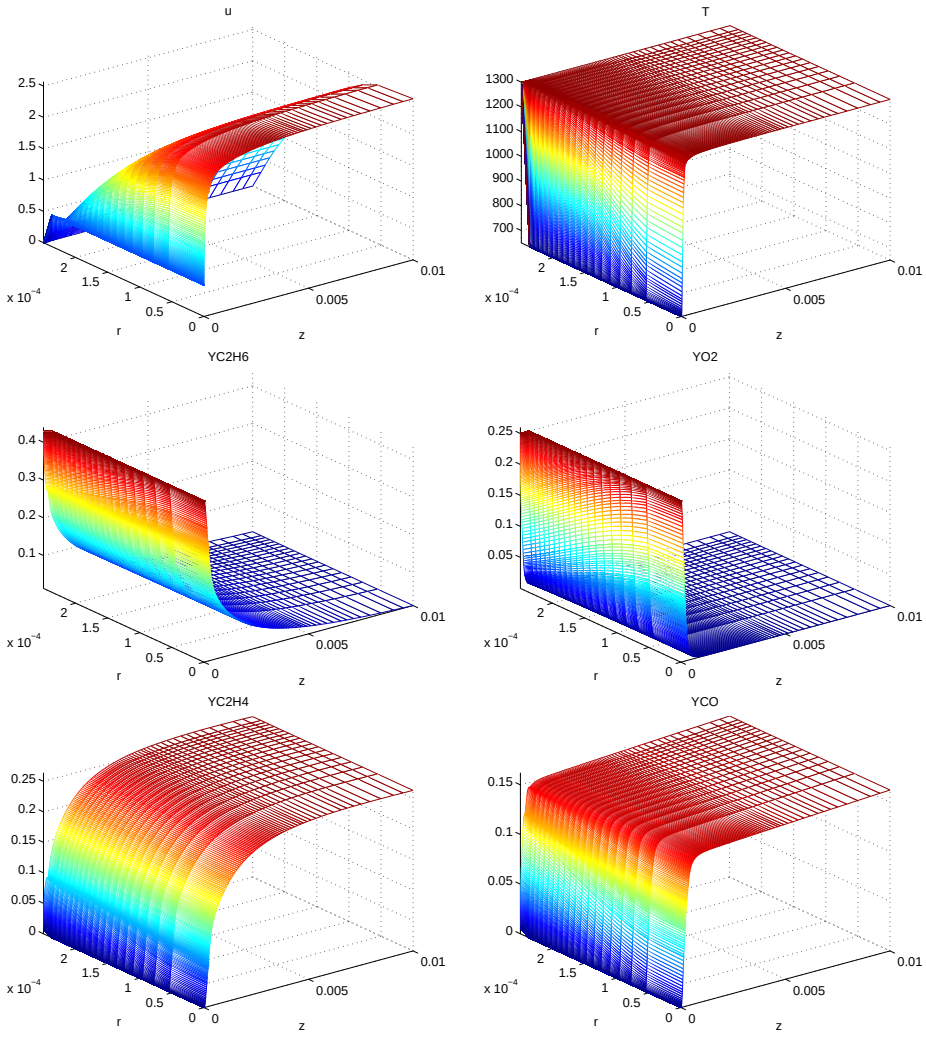

Figure 5.7: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of conversion of ethane to ethylene (I).

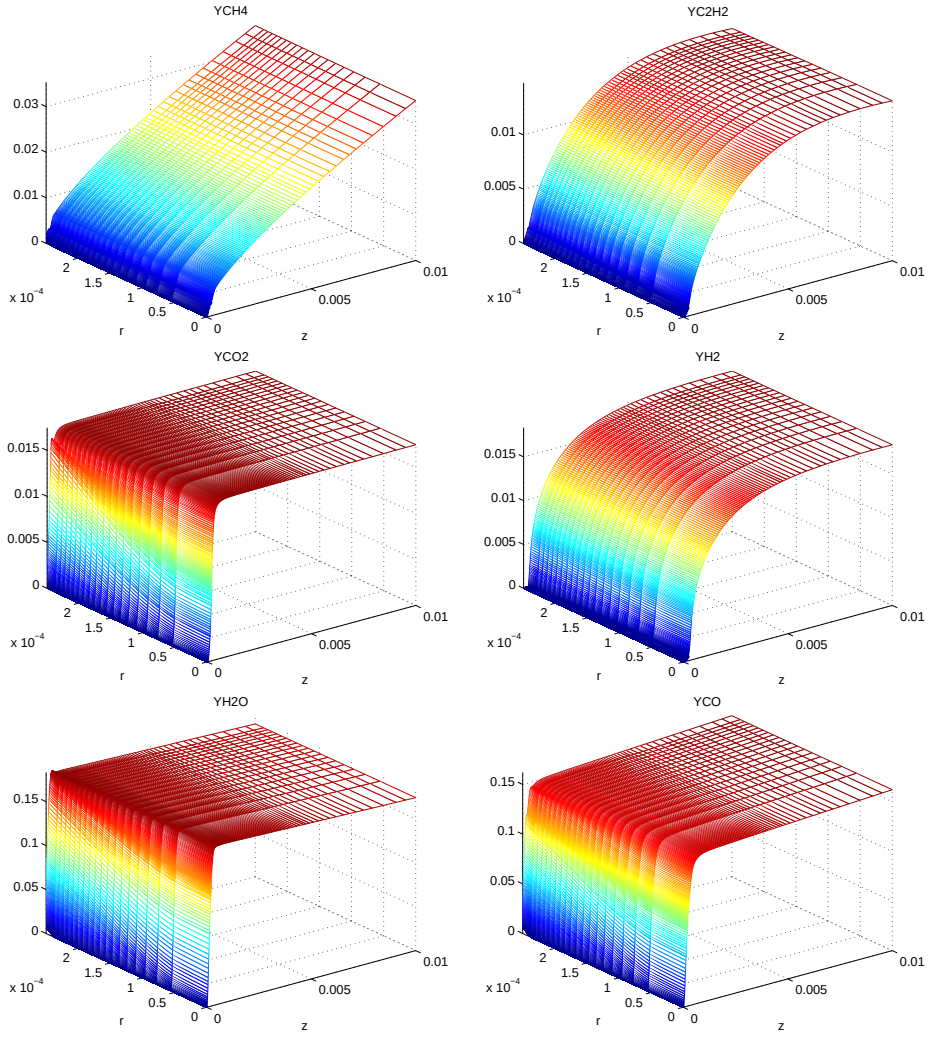

Figure 5.8: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of conversion of ethane to ethylene (II).

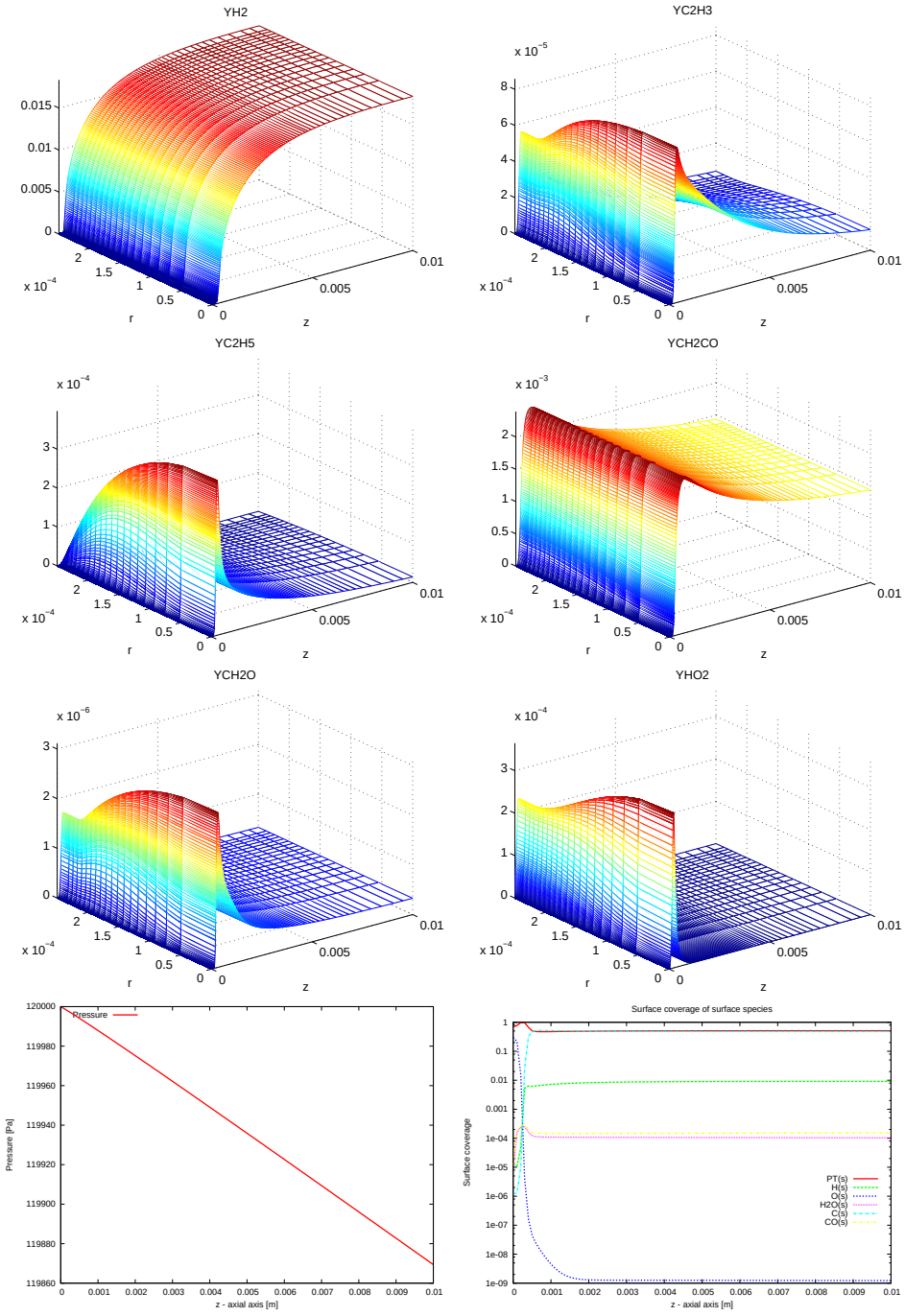

Figure 5.9: Profiles of axial velocity, pressure, temperature and some selected species from simulation results of conversion of ethane to ethylene (III).

5.2 Comparison with the software DETCHEM$^{\text{CHANNEL}}$ V.1.1

In this section, we give a comparison between the new developed program BLAYER$^{\text{sim}}$ with the program DETCHEM$^{\text{CHANNEL}}$ version 1.1.

The code DETCHEM$^{\text{CHANNEL}}$ also uses the same boundary layer model as we do, but with a different approach and numerical methods: In DETCHEM$^{\text{CHANNEL}}$ code, the PDE model is directly discretized by a finite volume method as the approach of the method of lines, and use implicit extrapolation code LIMEX [44] to solve the resulting DAEs. In addition, due to certain difficulties, a relaxation mechanism is used, and the boundary conditions (1.55) and (1.56) not solved as algebraic constraints and are relaxed them as differential equations for the surface species. The boundary conditions (1.55) are not solved but instead the radial diffusion fluxes $J_{k,r}$ at the wall are calculated as $J_{k,r} = -\dot{r}_k$, where r_k is the surface reaction rate.

In the following, all floating-point computations are performed on a Pentium 4, 2.6 GHz, Suse Linux 9.0. Both codes DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ are compiled with GNU Fortran/C compilers version 3.3.1 using compilers' optimization flag -O2 on the same computer. The integration errors are controlled with the relative error tolerance RTOL $= 10^{-3}$ and the absolute error tolerance ATOL $= 10^{-9}$. In the two following sections, comparison between DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ is presented with respect to numerical results and performance.

5.2.1 Comparison of numerical results

Catalytic combustion of methane

A gas mixture flows into the channel with the following setting.

- Channel geometry: the radius $r_{\max} = 2.5 \times 10^{-4}$ [m], the channel length $z_{\max} = 0.01$ [m].
- Initial conditions: the initial mole fraction of each species $X_{\text{CH}_4} = 0.5$, $X_{\text{O}_2} = 0.3$, and $X_{\text{N}_2} = 0.2$, other species are absent at inlet. The initial gas temperature $T_{\text{gas}} = 298$ [K], and the initial pressure $p = 1.2 \times 10^5$ [Pa], and the initial velocity $u = 1$ [m/s].
- Boundary conditions: the temperature at the wall $T_{\text{wall}} = 1373$ [K].
- Reaction mechanisms: 21 gas-phase species, 11 surface species, 23 surface reactions, and 128 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.

– Number of grid points in the radial direction: 12.

This problem is referred to as *methane12* .

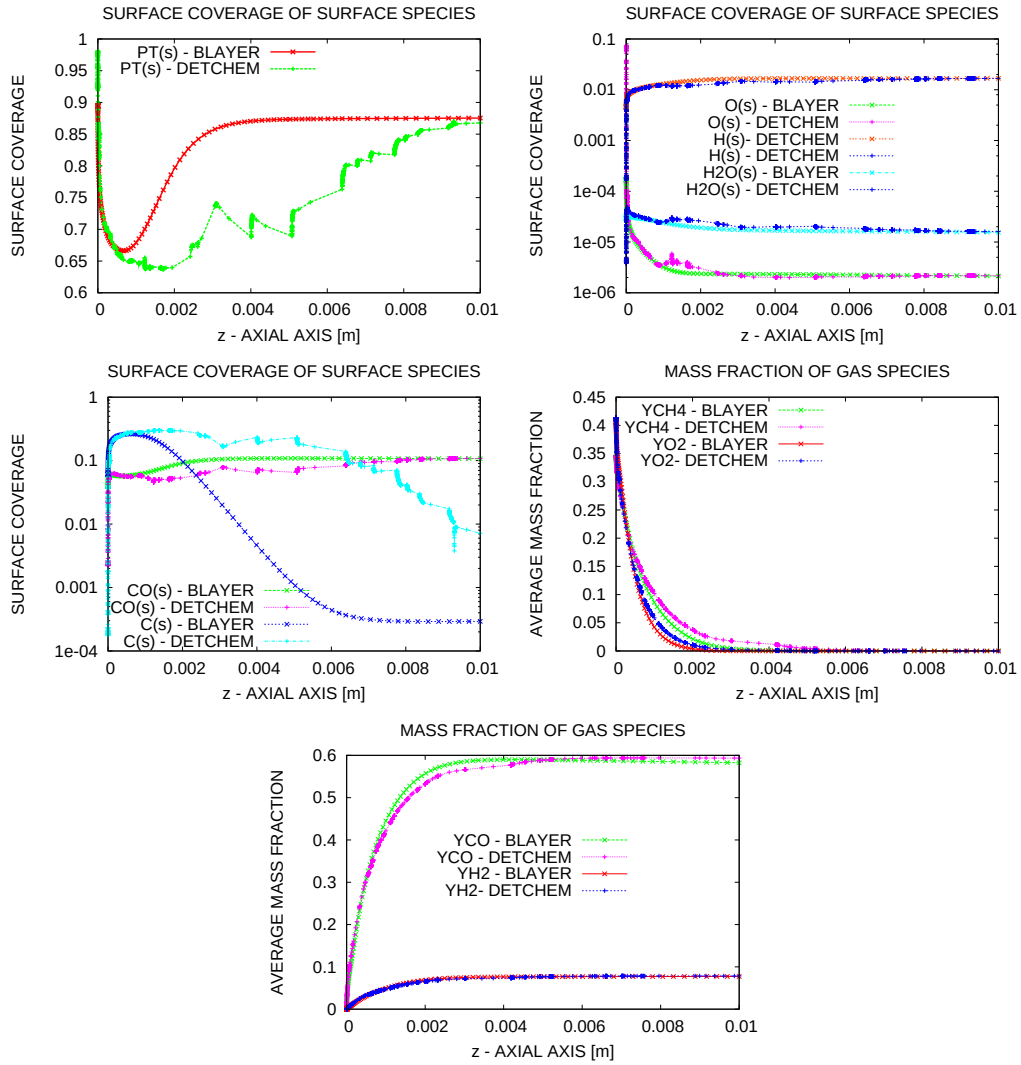

Figure 5.10: Surface species and average mass fractions of some selected gases by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ (*methane12*).

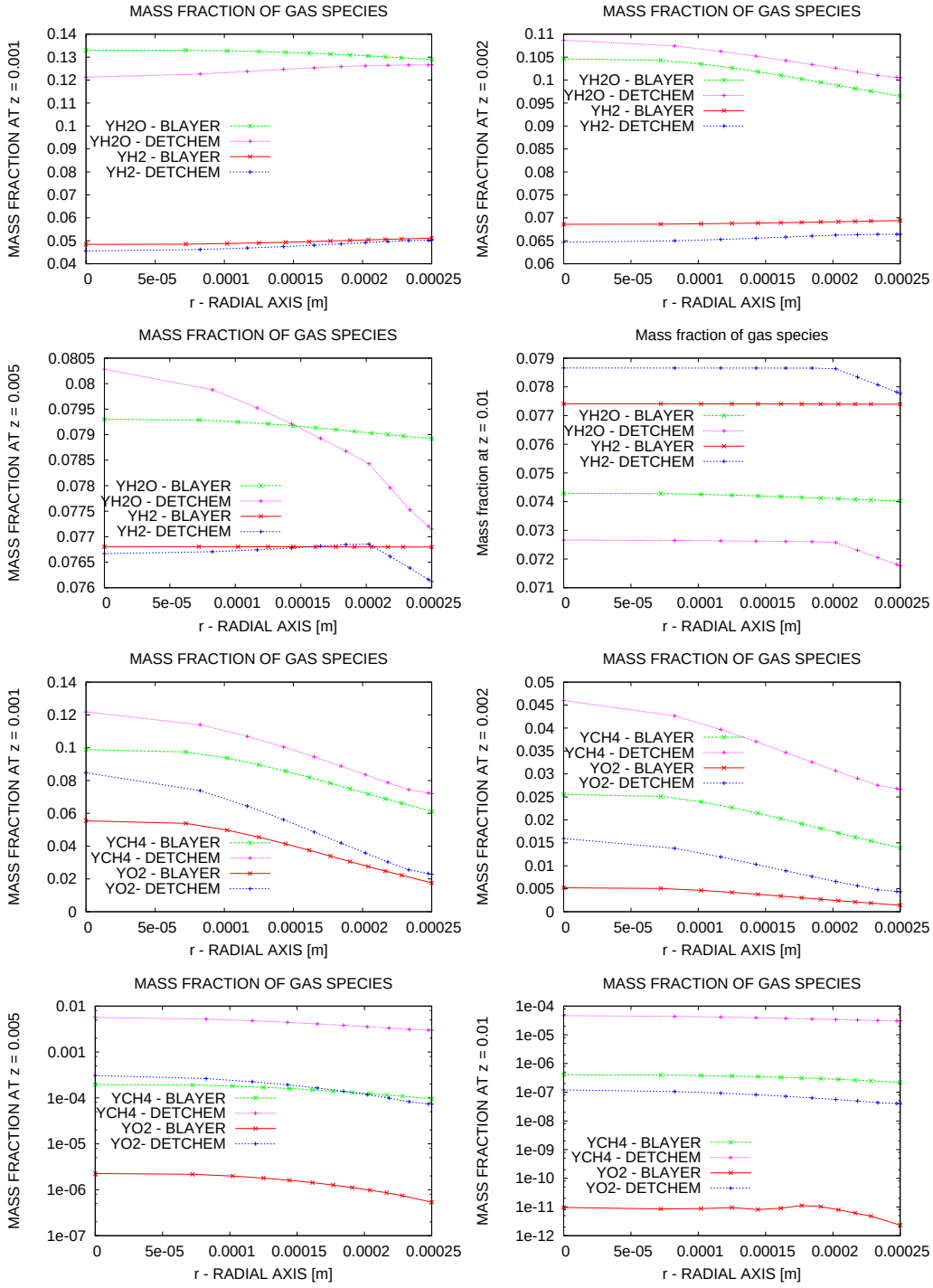

Figure 5.11: Gas-phase species by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ (*methane12*).

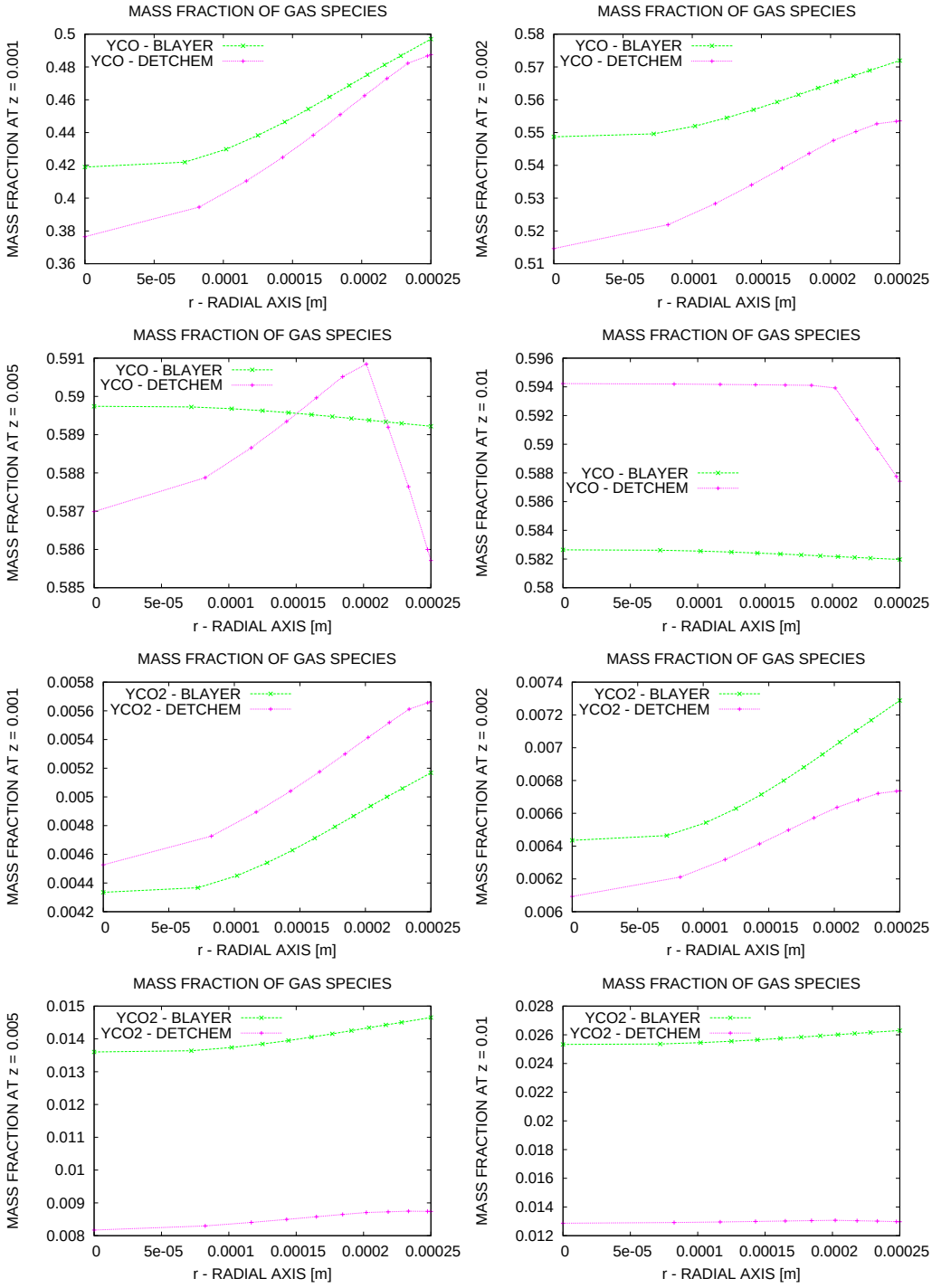

Figure 5.12: Gas-phase species by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ (*methane12*).

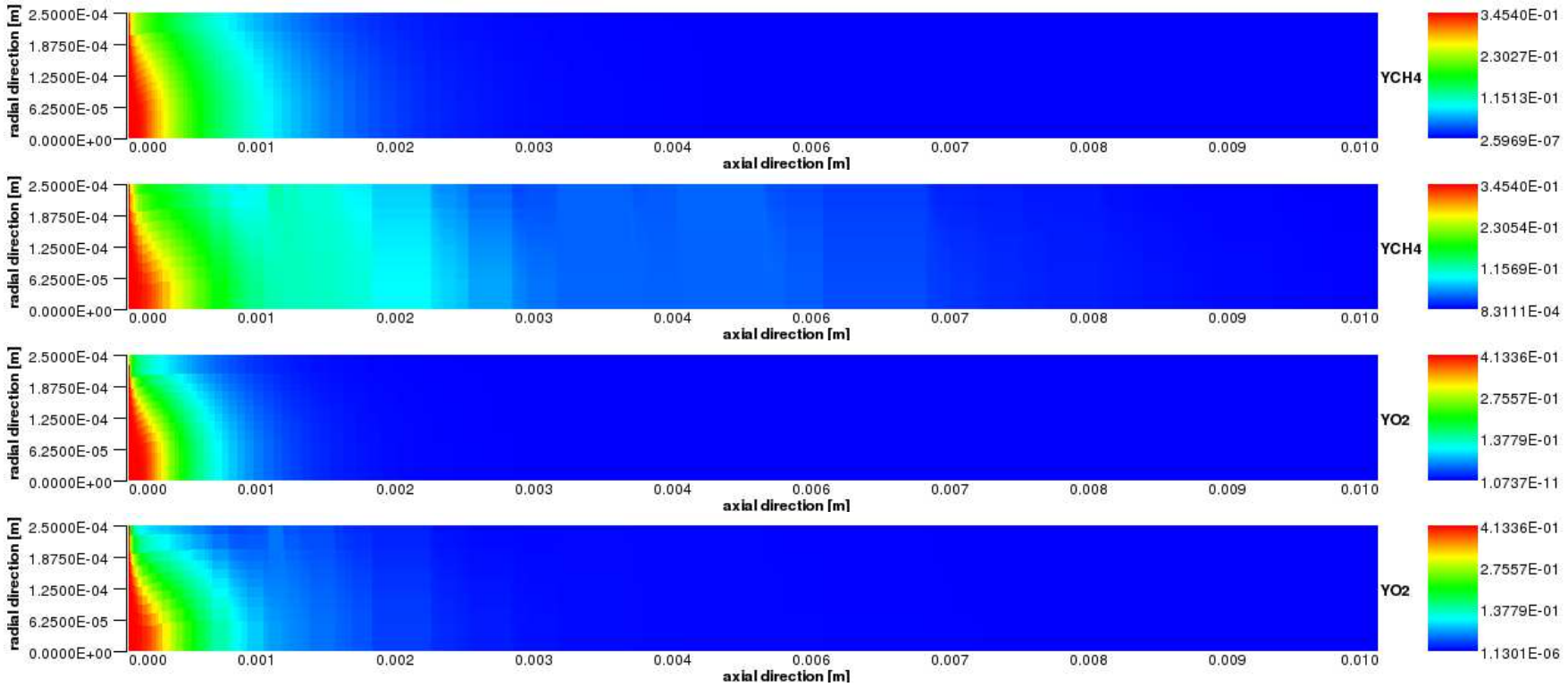

Figure 5.13: Mass fraction of source species profiles: the upper is obtained by BLAYER$^{\text{sim}}$ and the lower is obtained by DETCHEM$^{\text{CHANNEL}}$ (*methane12*).

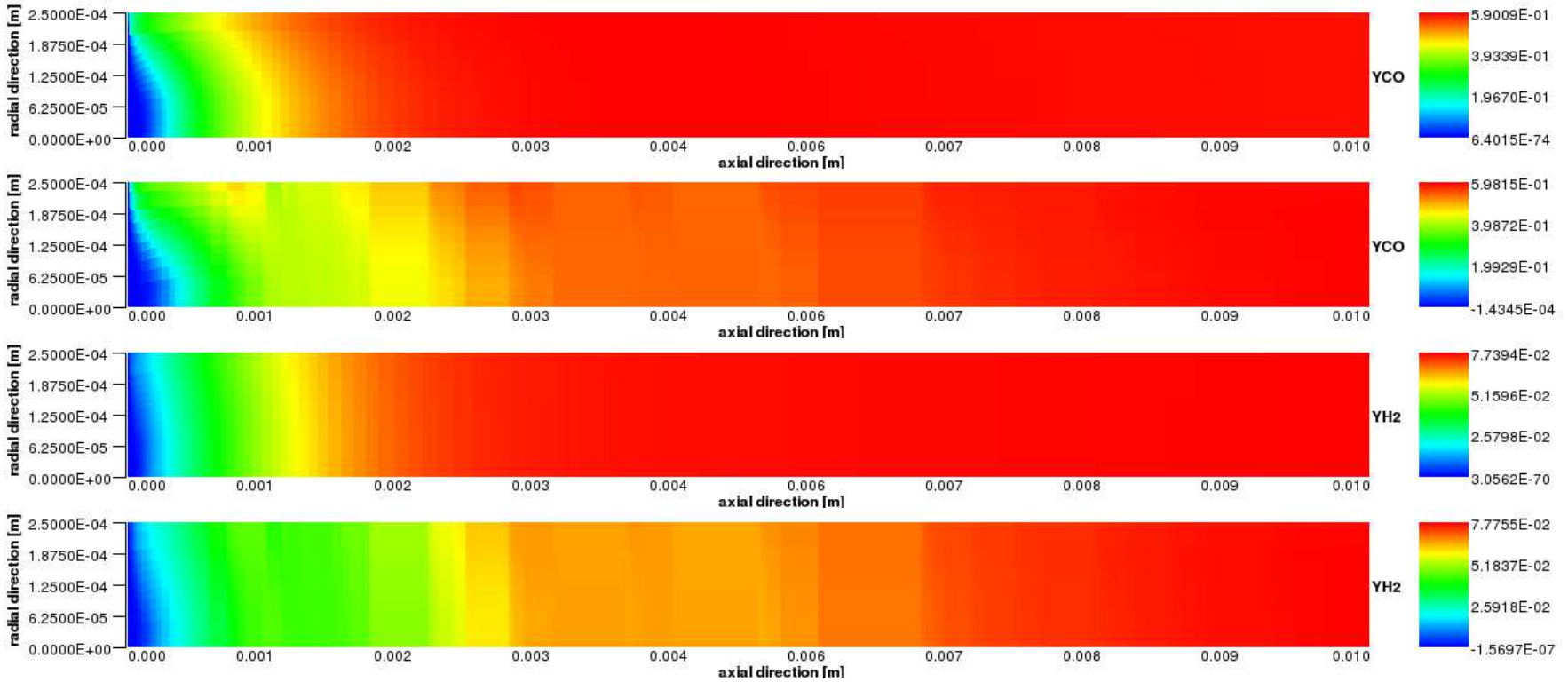

Figure 5.14: Mass fraction of product species profiles: the upper is obtained by BLAYER$^{\text{sim}}$ and the lower is obtained by DETCHEM$^{\text{CHANNEL}}$ (*methane12*).

Figures 5.10–5.14 show the numerical results obtained by DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$. Because of space restriction here we only show some major product species and source species although as mentioned above, in total this problem involves 23 gas species and 11 surface species.

The two figures at the top of Figure 5.10 show the profile of surface coverage of surface species along the channel. The solid surface (uncovered site of the solid surface) PT(s) fractions are much different. The surface coverage PT(s) obtained by BLAYER$^{\text{sim}}$ is smooth along the channel, while PT(s) obtained by DETCHEM$^{\text{CHANNEL}}$ seems to be not smooth and is changed abnormally along the channel. Other surface species seem to be close between the two results, although certain difference still show up, for instance, a typical one is the surface coverage of C(s), and the profiles of surface species obtained by DETCHEM$^{\text{CHANNEL}}$ are not smooth compared to the corresponding ones obtained by BLAYER$^{\text{sim}}$.

The two figures at the bottom of Figure 5.10 show the average mass fractions of some selected gas species. The average mass fractions are defined as

$$Y_k^{\text{avg}} = \frac{\int_0^{r_{\max}} Y_k \mathrm{d}r}{r_{\max}} \quad (k = 1, \ldots, N_g).$$

There are some differences between the two results, in particular, at the first half of the channel. The chemical source species profiles (methane and oxygen) by BLAYER$^{\text{sim}}$ are consumed faster than by DETCHEM$^{\text{CHANNEL}}$, this is because of the fact that BLAYER$^{\text{sim}}$ solves the boundary conditions in its steady state form rather than by a relaxation one as in DETCHEM$^{\text{CHANNEL}}$. At the second half of the channel, in particular near the outlet, the two results are nearly the same, only slightly different, and the differences are below 3 percents for major species products (CO—carbon monoxide and H_2—hygroden).

Figures 5.11 and 5.12 show the mass fractions of gas species at different locations along the channel. Figures 5.13 and 5.14 show the flow field of the source species (CH_4—methane and O_2—oxygen) and the major products (CO—carbon monoxide and H_2—hydrogen). There are differences between the two results of the source species. At the outlet, the mass fractions of methane and oxygen obtained by BLAYER$^{\text{sim}}$ are 2.59E-7 and 1.07E-11, while the ones obtained by DETCHEM$^{\text{CHANNEL}}$ are 8.31E-4 and 1.13E-6, respectively. However, the major products—CO and H_2—are nearly the same at the end of outlet, and the differences are below one percent, while there are a bit differences at the first 2 [mm] of the channel.

Conversion of ethane to ethylene

The simulation setting is as follows [20]:

- Channel geometry: the radius $r_{\max} = 2.5 \times 10^{-4}$ [m], the channel length $z_{\max} = 0.01$ [m].
- Initial conditions: the initial mole fraction of each species $X_{C_2H_6} = 0.44$, $X_{O_2} = 0.26$, and $X_{N_2} = 0.3$, other species are absence at inlet. The initial gas temperature $T_{\text{gas}} = 650$ [K], and the initial pressure $p = 1.2 \times 10^5$ [Pa], and the initial velocity $u = 0.5$ [m/s].
- Boundary conditions: the temperature at the wall $T_{\text{wall}} = 1300$ [K].
- Reaction mechanisms: 25 gas-phase species, 20 surface species, 82 surface reactions, and 261 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.
- Number of grid points in the radial axis: 12.

(a) $F_{\text{cat/geo}} = 1$.

The problem of conversion of ethane to ethylene with the above setting is referred to as *ethane1* .

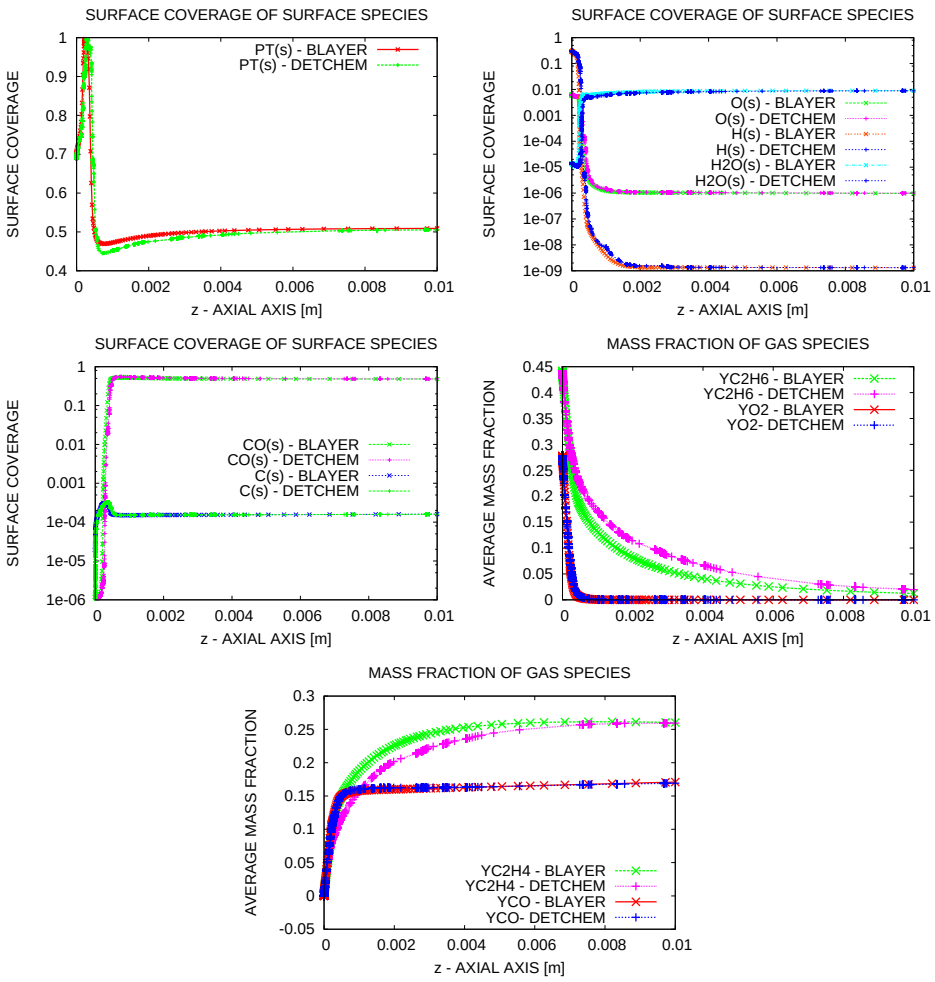

Figure 5.15: Surface species and average mass fractions of some selected gases by BLAYER$^{\text{sim}}$and DETCHEM$^{\text{CHANNEL}}$(*ethane1*).

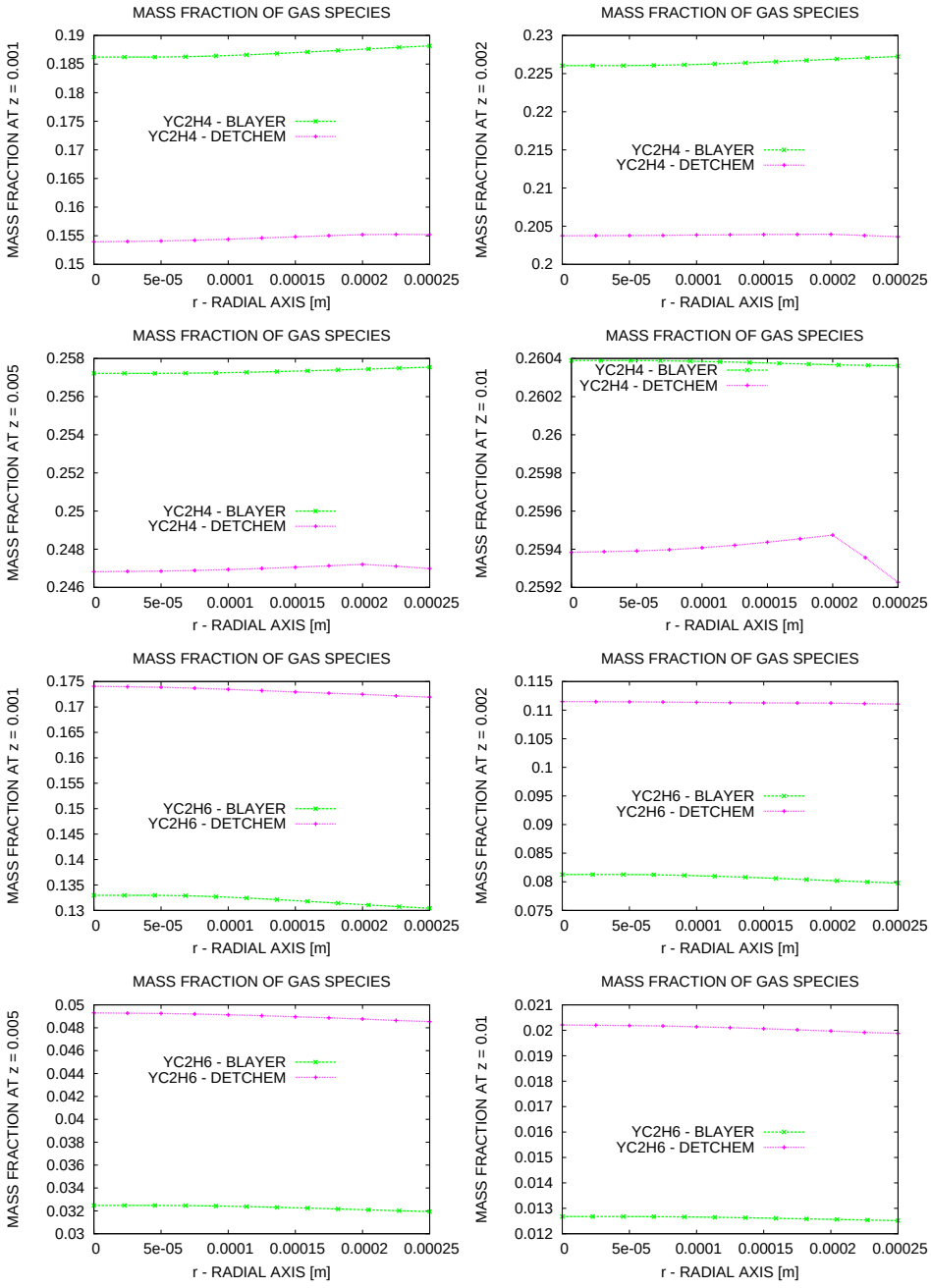

Figure 5.16: Gas-phase species by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ (*ethane1*).

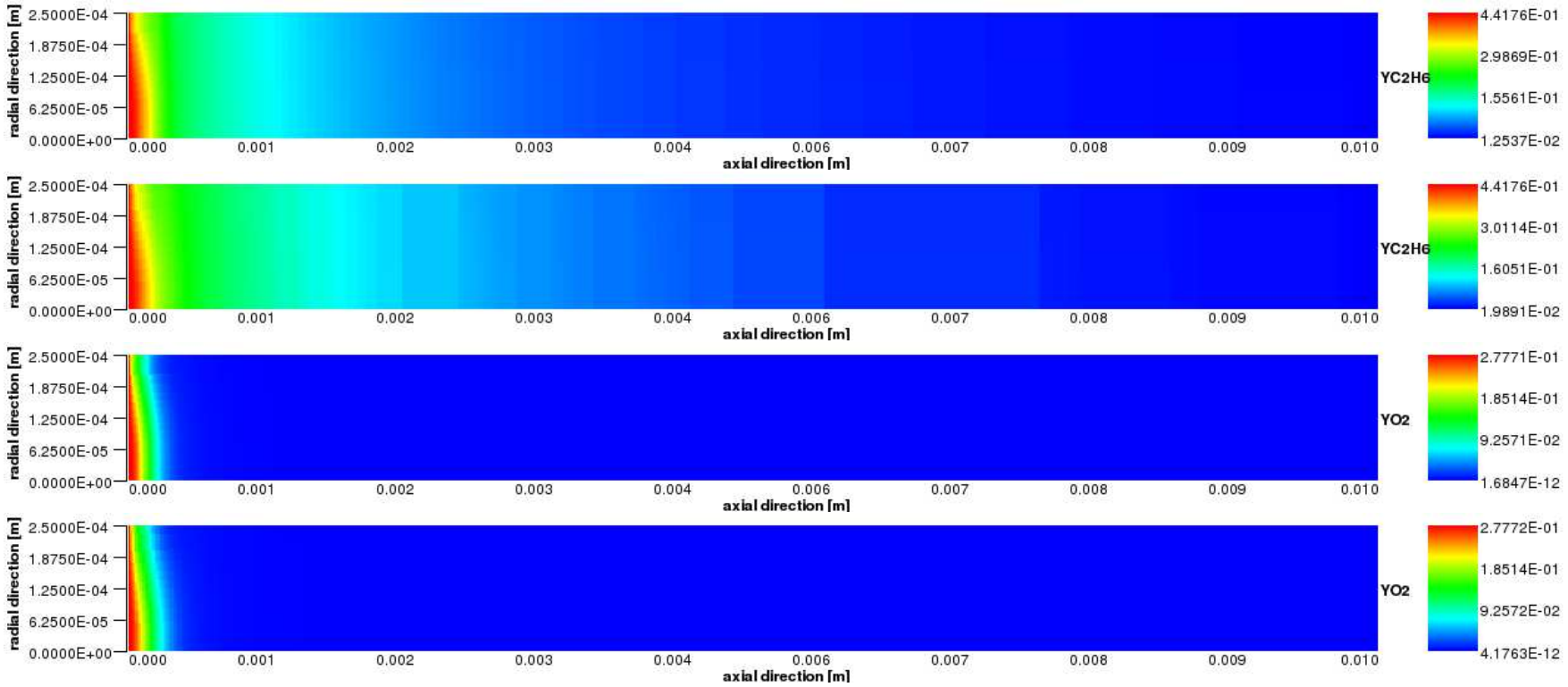

Figure 5.17: Mass fraction of source species profiles: the upper is obtained by BLAYER$^{\text{sim}}$ and the lower is obtained by DETCHEM$^{\text{CHANNEL}}$ (*ethane1*).

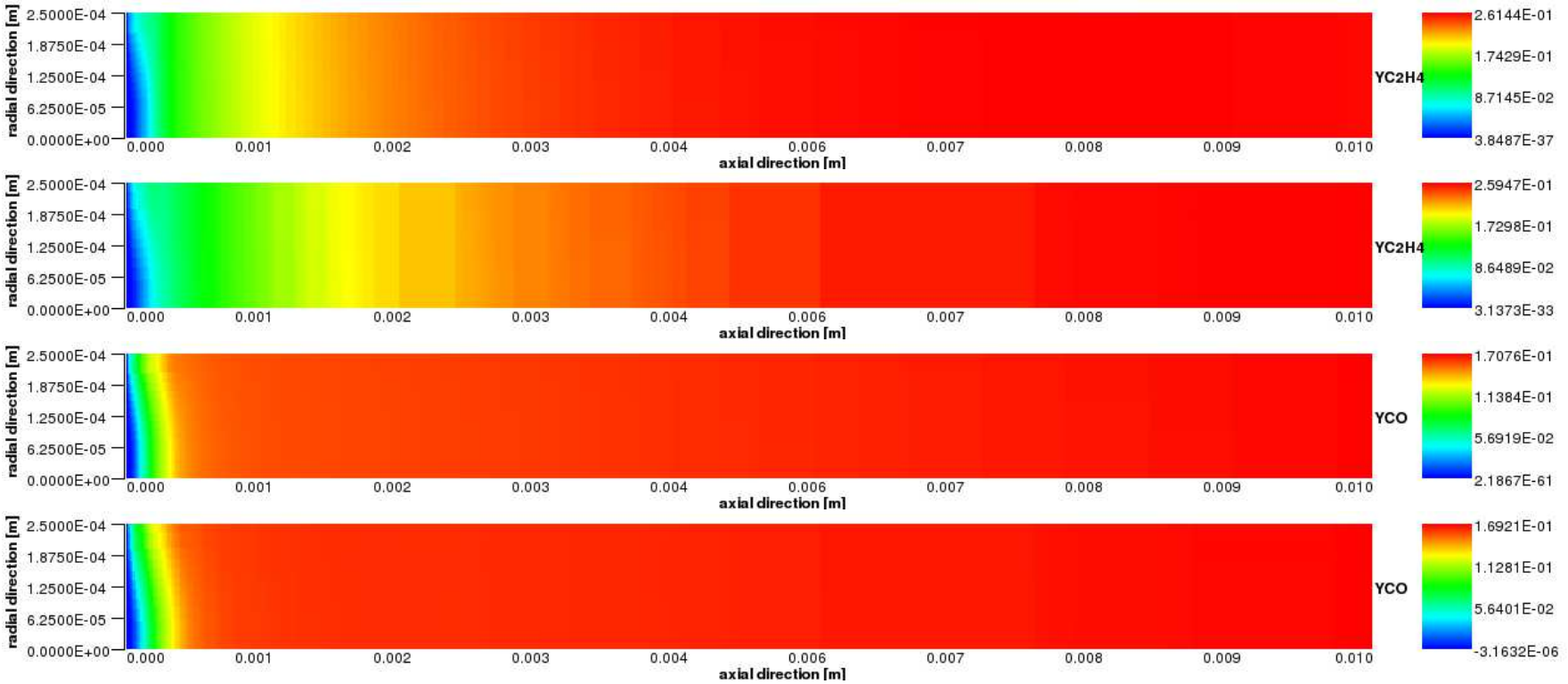

Figure 5.18: Mass fraction of product species profiles: the upper is obtained by BLAYER$^{\text{sim}}$ and the lower is obtained by DETCHEM$^{\text{CHANNEL}}$ (*ethane1*).

Figures 5.15–5.18 show some parts of the results of the simulation obtained by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$. Again because of space restriction, here we only show some major product species and source species although as mentioned above, in total this problem involves 25 gas species and 20 surface species. The two results are nearly the same, the differences between the two solutions are below 3 percents, although there are differences shown up as in Figure 5.15. The surface species profiles obtained by BLAYER$^{\text{sim}}$ are smoother than the corresponding ones obtained by DETCHEM$^{\text{CHANNEL}}$. The chemical source species profiles (ethane and oxygen) by BLAYER$^{\text{sim}}$ are consumed faster than by DETCHEM$^{\text{CHANNEL}}$. Figures 5.17 and 5.18 show the flow field of the source species (C_2H_6—ethane and O_2—oxygen) and the major products (C_2H_4—ethylene and CO—carbon monoxide). There are differences between the two results of the source species, in particluar ethane. At the outlet, the mass fractions of ethane and oxygen obtained by BLAYER$^{\text{sim}}$ are 1.25E-2 and 1.68E-12, while the ones obtained by DETCHEM$^{\text{CHANNEL}}$ are 1.98E-2 and 4.17E-12, respectively. However, the major products—C_2H_4 and CO—are nearly the same at the end of outlet, and the differences are below 1 percents, while there are little differences at the first 40 percents of the channel length. At the outlet, the mass fractions of ethylene and carbon monoxide obtained by BLAYER$^{\text{sim}}$ are 2.61E-1 and 1.70E-1 and by DETCHEM$^{\text{CHANNEL}}$ are 2.59E-1and 1.69E-1, respectively.

(b) $F_{\text{cat/geo}} = 0.01$.

The problem of conversion of ethane to ethylene with the above setting is referred to as *ethane2* .

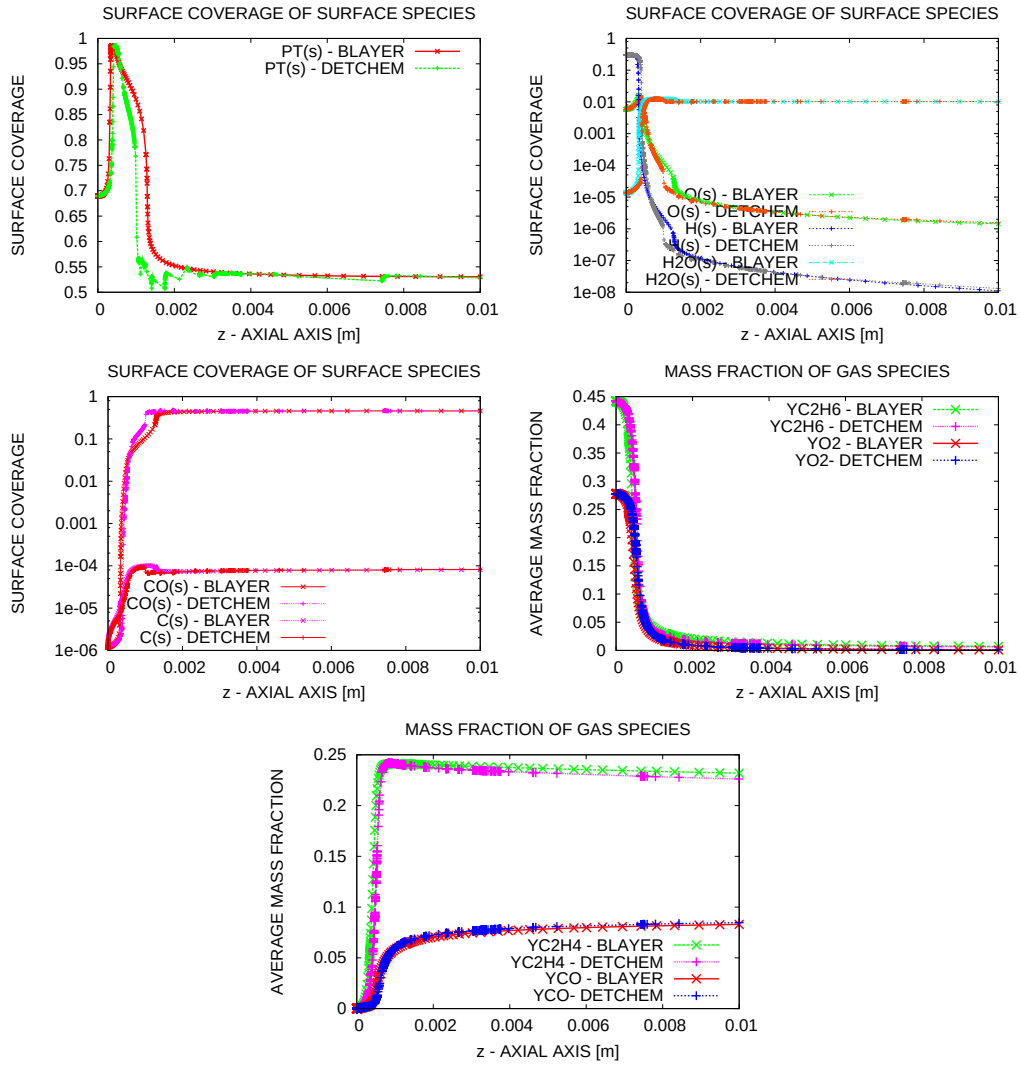

Figure 5.19: Surface species and average mass fractions of some selected gases by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ (*ethane2*).

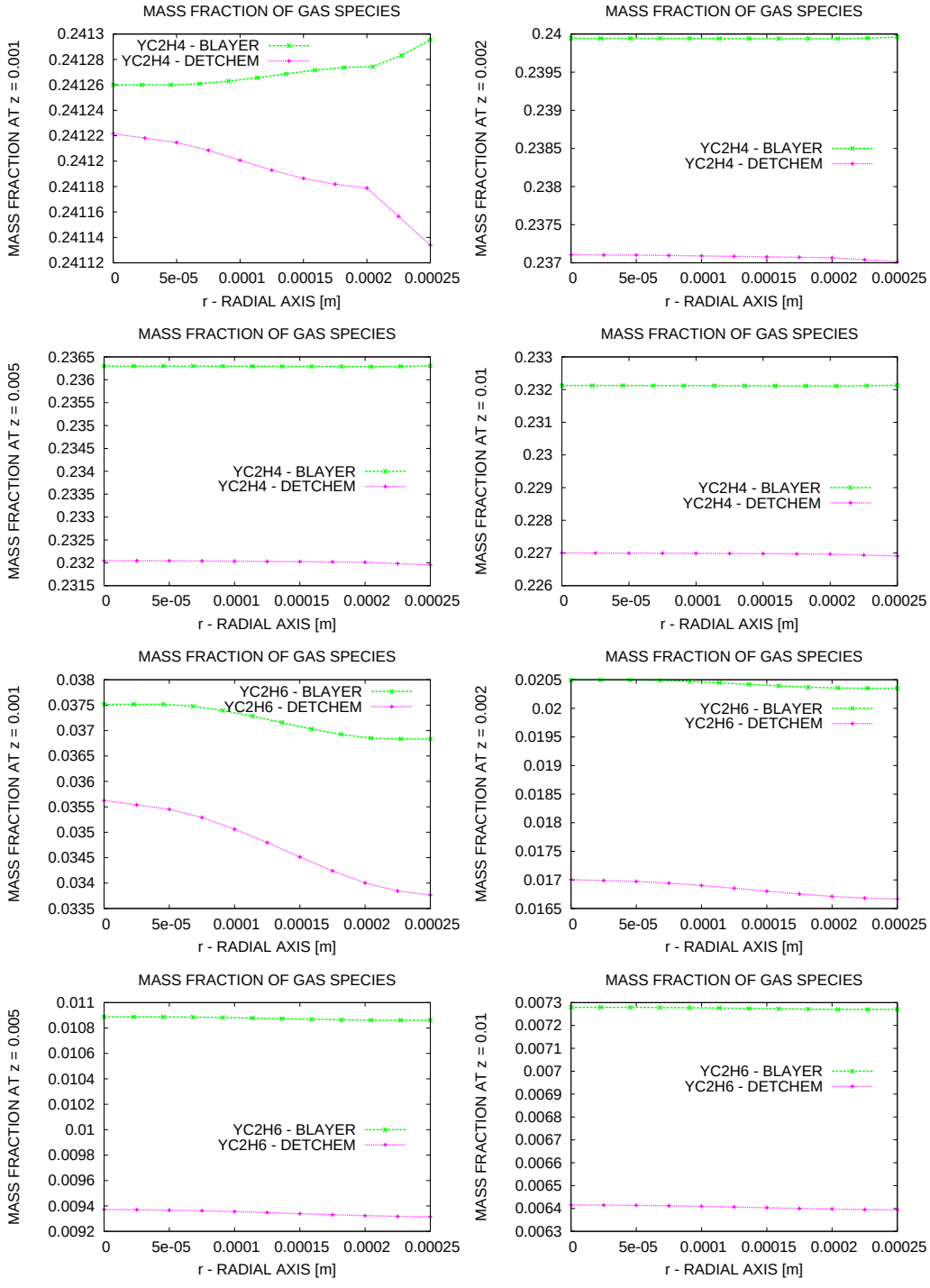

Figure 5.20: Gas-phase species by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ (*ethane2*).

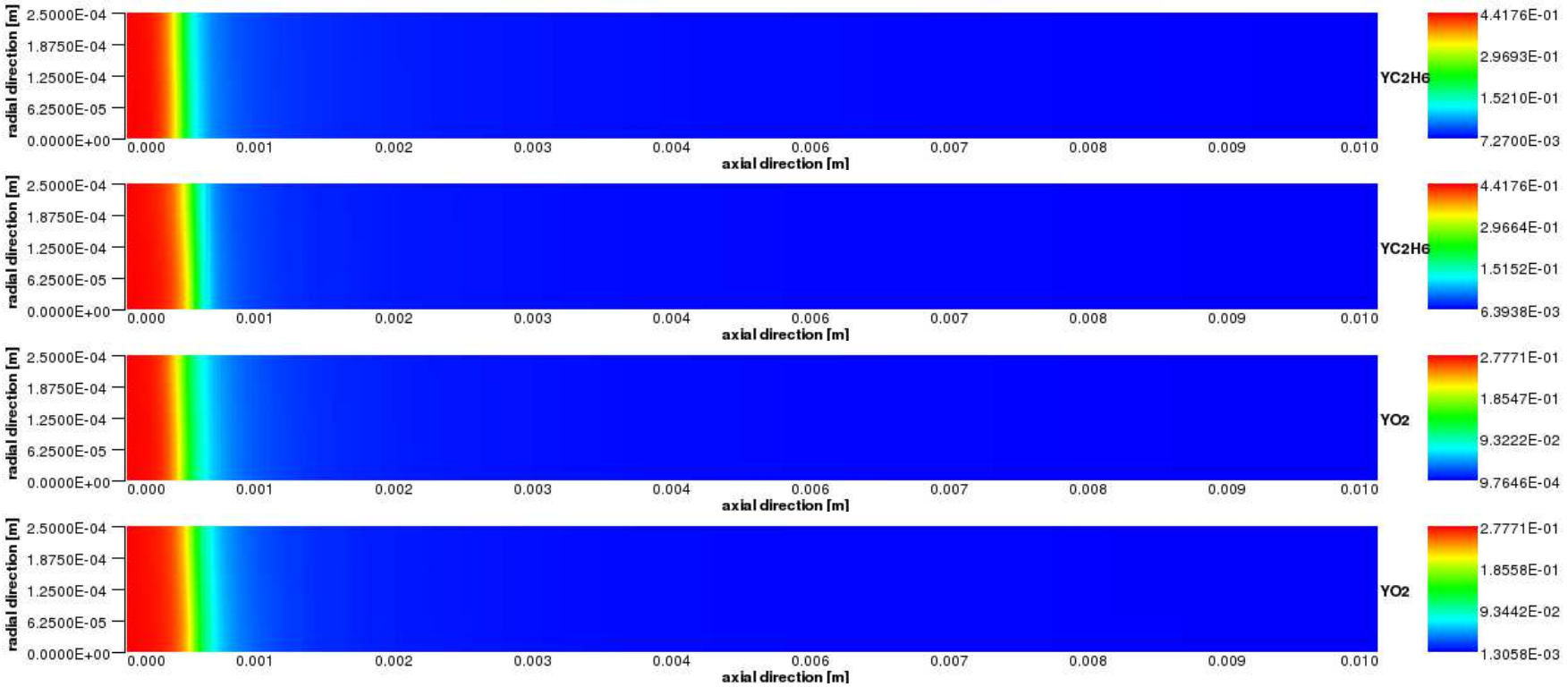

Figure 5.21: Mass fraction of source species profiles: the first is obtained by BLAYER$^{\text{sim}}$ and the next is obtained by DETCHEM$^{\text{CHANNEL}}$ (*ethane2*).

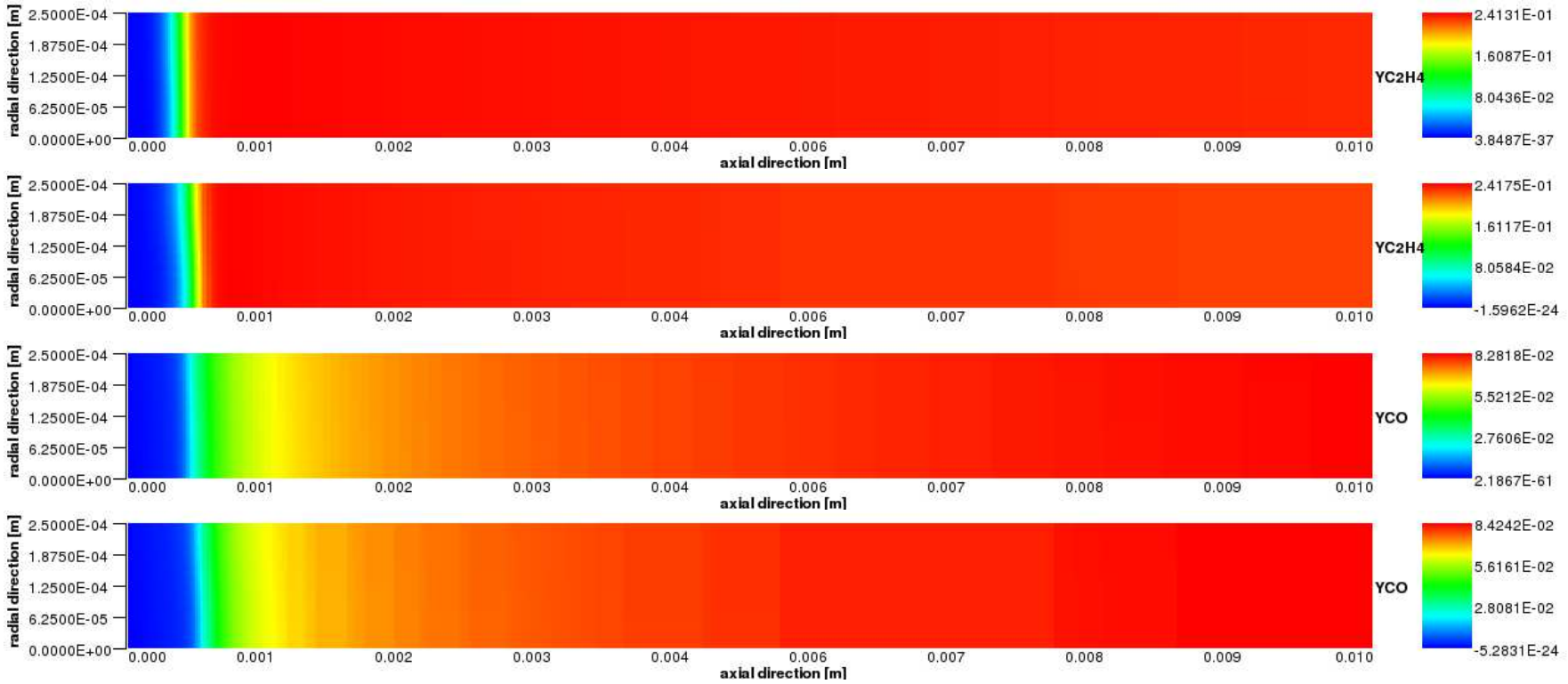

Figure 5.22: Mass fraction of product species profiles: the first is obtained by BLAYER$^{\text{sim}}$ and the next is obtained by DETCHEM$^{\text{CHANNEL}}$ (*ethane2*).

The results obtained BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ are similar in qualitative behavior as in the case *ethane1*. The two results are nearly the same.

Stability of the solution with variation in the number of grid points

In this part we study the behavior of the solution when the spatial grid is refined (grid in the radial direction). In particular, we run the two simulation tools BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ with different numbers of grid points. In accordance to the theory, by refinement of the spatial grid, i.e., inreasing the number of spatial disretization grid points, the spatial disretization error is reduced. The expected solutions with different grid points in the radial direction should approach each other, e.g., well behaved.

Here, we investigate two problems, named *methane-grids* and *ethane1-grids*.

(a) The setting for the problem *methane-grids* is the same as for the *methane12* (see page 156) except now the number of grid points in the spatial direction is changed . Figures 5.23 and 5.24 show the results of the simulation runs.

(b) The setting for the problem *ethane1-grids* is the same as for the *ethane1* (see page 162) problem except now the number of grid points in the spatial direction is changed . Figures 5.25 and 5.26 show the results of the simulation runs.

The solutions obtained by BLAYER$^{\text{sim}}$ and by DETCHEM$^{\text{CHANNEL}}$ are nearly the same, although the surface species obtained by DETCHEM$^{\text{CHANNEL}}$ are not smooth.

Note that the average mass fraction profile of oxygen in Figure 5.26 obtained by BLAYER$^{\text{sim}}$ at the second half of the channel seems to be not stable but in fact this is a correct behaviour because the simulation run using the absolute integration error tolerance ATOL = 10^{-9}. Thus, the value of variables (here the mass fraction of oxygen—YO_2), which is below ATOL, is not controlled by the error control of the integrator and it is treated as having value of ATOL, and can be considered as numerical noise.

Remark 5.2.1
It is well known that the solution of parabolic partial differential equations is dominated by the boundary conditions when the "time" (here in our problem z) is large. The boundary conditions in our problem are the coupling of surface chemistry with the surrounding flow field, in addition, the no-slip condition and boundary condition of the temperature. The boundary conditions

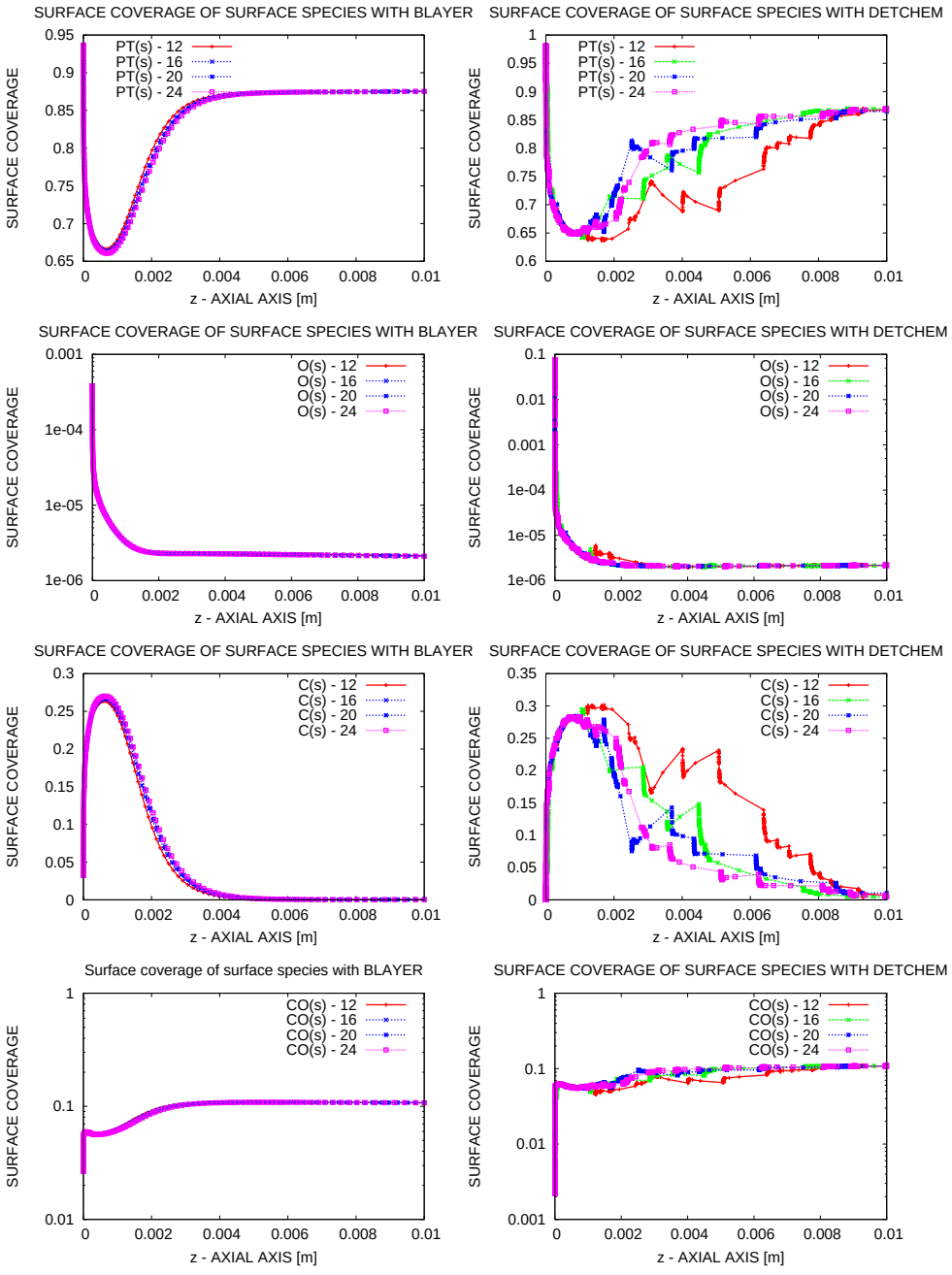

Figure 5.23: Surface species with a different number of grid points in the radial axis by DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ (*methane-grids*).

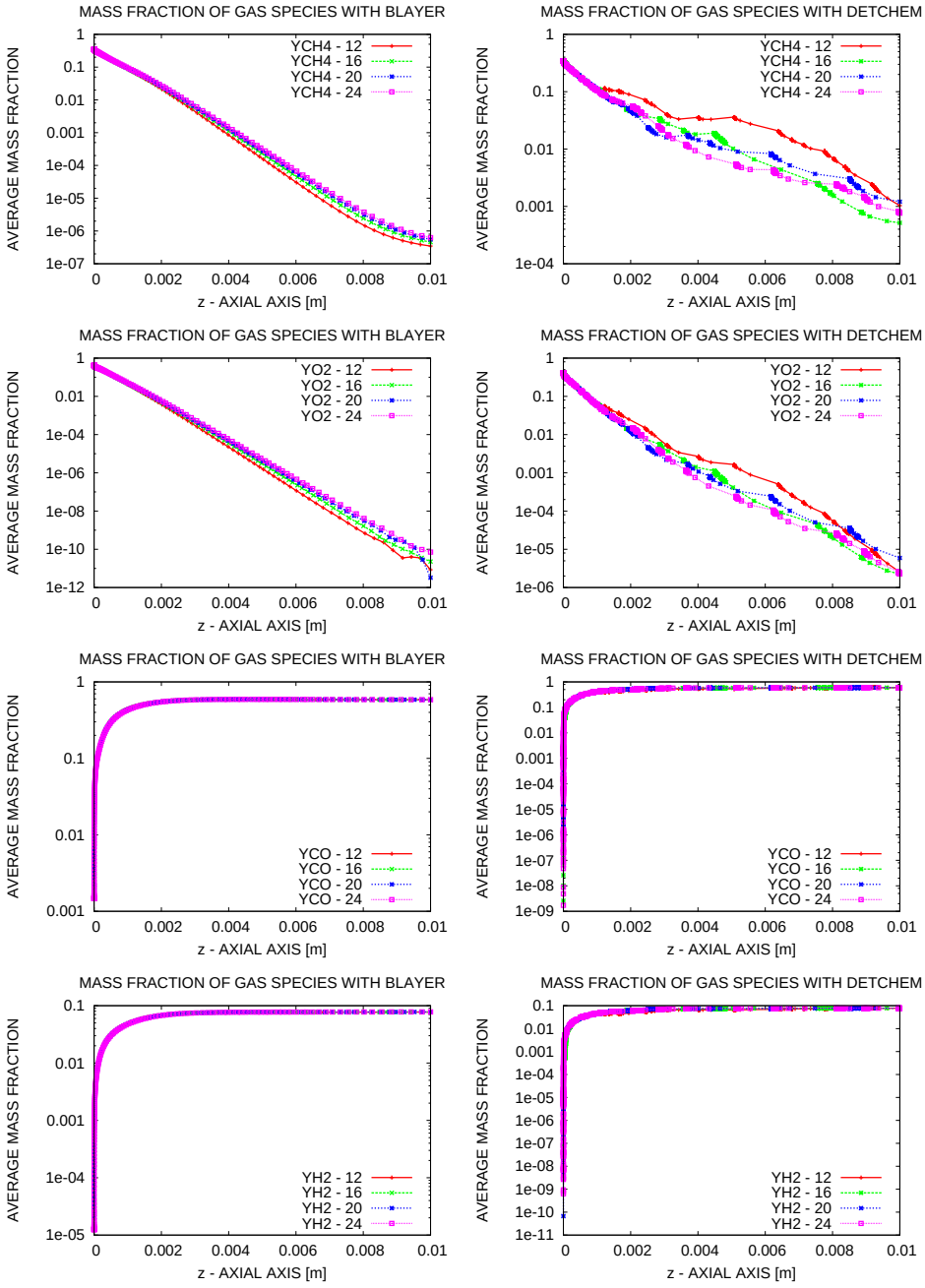

Figure 5.24: Average of mass fraction of gas species with different numbers of grid points in the radial axis by DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ (*methane-grids*).

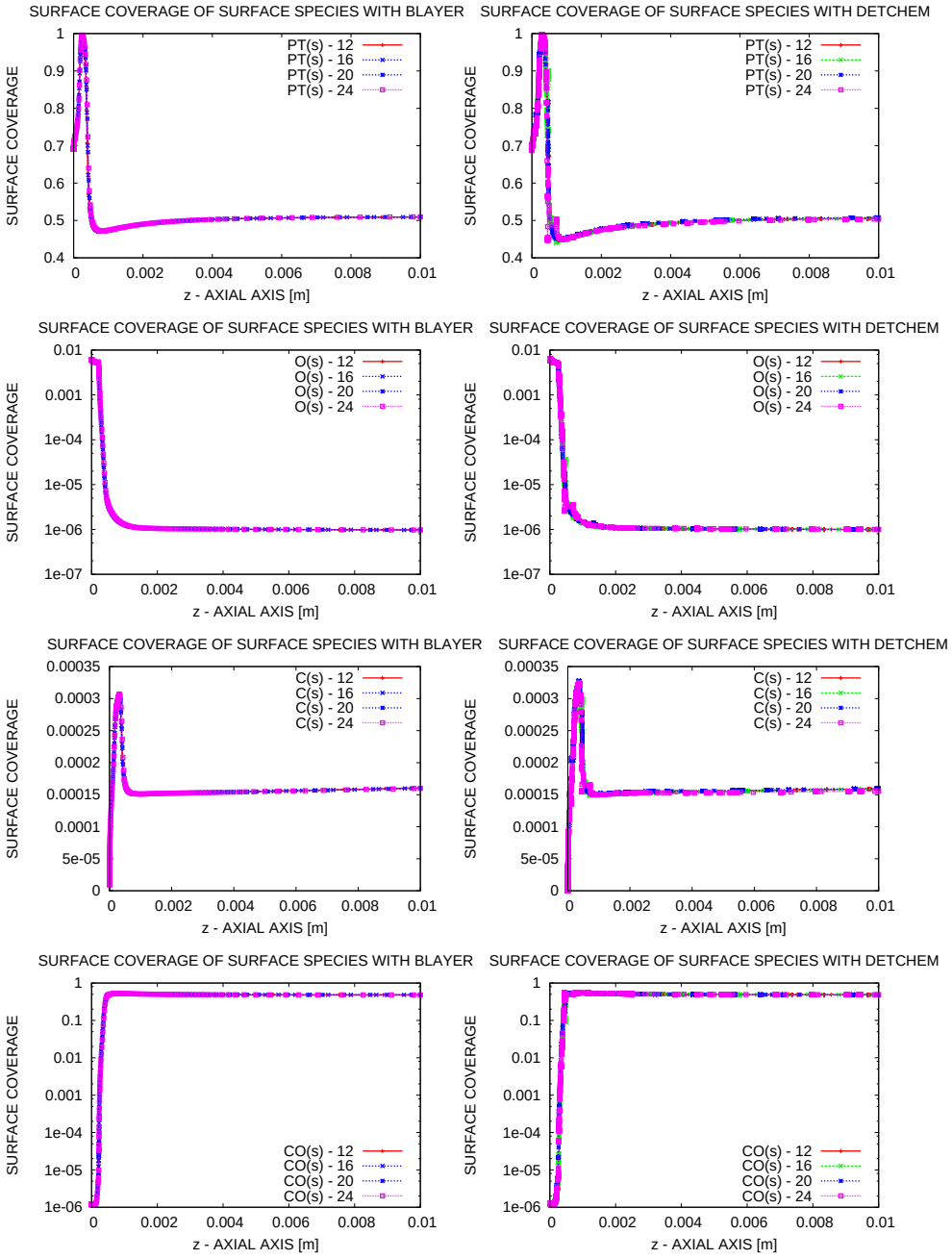

Figure 5.25: Surface species with different numbers of grid points in the radial axis for the ethane problem by DETCHEM$^{\mathrm{CHANNEL}}$ and BLAYER$^{\mathrm{sim}}$ (*ethane1-grids*).

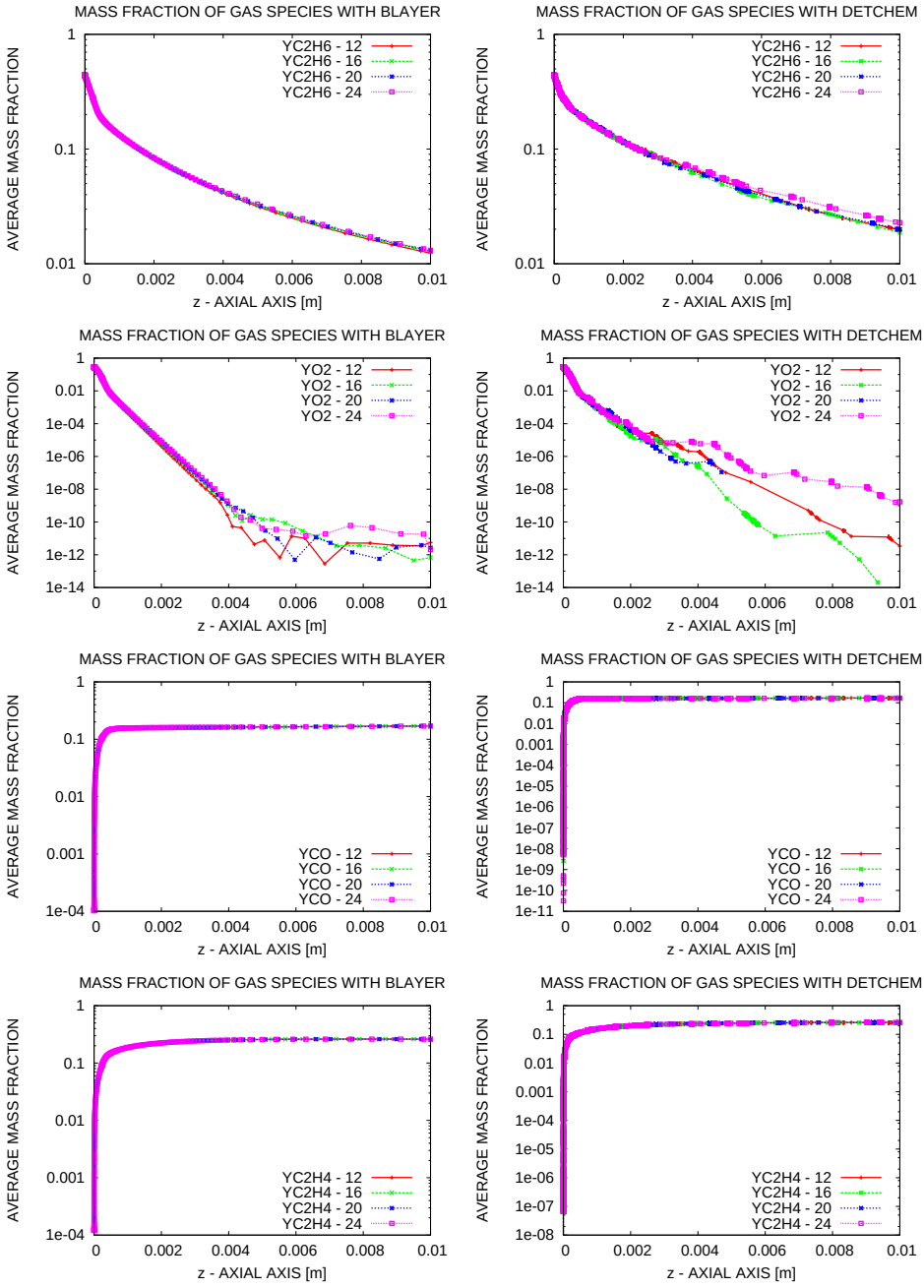

Figure 5.26: Average of mass fraction of gas species with different numbers of grid points in the radial axis for the ethane problem by DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ (*ethane1-grids*).

propagate into the interior region (inside channel) by the finite difference operator in the spatial direction ψ. Therefore, the speed of propagation of the boundary conditions into the interior region by diffusion (of the numerical solution) depends on how one treats the boundary conditions: loosely coupling as in DETCHEM$^{\text{CHANNEL}}$ or strictly coupling as in BLAYER$^{\text{sim}}$. Loosely speaking, if a loosely coupling method is applied the influence of the boundary condtions is slower than the influence of the boundary conditions if a strictly coupling method is applied. This behavior is confirmed by the above numerical results where the conversions obtained by DETCHEM$^{\text{CHANNEL}}$ are slower than the ones obtained by BLAYER$^{\text{sim}}$. As we see from the numerical results obtained by DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$, in spite of slow propagation or fast propagation of the boundary conditions, the final major products (ethylene and carbon monoxide) obtained by DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ are nearly the same because sooner or later if the source species (oxygen and ethane) do not react at the first few millimeters of the channel, then they react (i.e., taking into account by the simulation code) along the remaining part of the channel. The oxygen and ethane (source species) are exhausted.

5.2.2 Performance comparison

We investigate three problems, namely *methane-grids* (see page 172) and *ethane1-grids* (see page 172) as the above, and *ethane2-grids* which is the same as the *ethane2* (see page 167) except the number of grid points in the radial direction is varied. Tables 5.1–5.3 show CPU time and number of steps and speedup for these problems by BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$, where the speedup is defined as

$$\text{Speedup} = \frac{\text{CPU time by DETCHEM}^{\text{CHANNEL}}}{\text{CPU time by BLAYER}^{\text{sim}}}.$$

5.3 Optimization results

This is the first time a systematic approach is used to determine the optimal process conditions for these applications. In the following, the BLAYER$^{\text{opt}}$ software are applied to different practical applications.

5.3.1 Catalytic combustion of methane

The flow conditions are as the following.

	Nodes	DETCHEM$^{\text{CHANNEL}}$	BLAYER$^{\text{sim}}$	Speedup
CPU	12	218.60	3.5	62.45
Time	16	221.85	5.95	37.28
(secs)	20	218.8	9.23	23.70
	24	312.11	13.5	23.11
Number	12	1286	145	
of	16	1057	154	
Steps	20	908	159	
	24	1067	162	

Table 5.1: CPU time and number of integration steps using DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ (*methane-grids*).

	Nodes	DETCHEM$^{\text{CHANNEL}}$	BLAYER$^{\text{sim}}$	Speedup
CPU	12	1059.56	18.1	58.53
Time	16	1413	26.95	52.43
(secs)	20	1880	38.82	48.42
	24	2357	53.3	44.22
Number	12	1818	194	
of	16	2058	194	
Steps	20	2262	201	
	24	2643	196	

Table 5.2: CPU time and number of integration steps using DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ (*ethane1-grids*).

- Channel geometry: the radius $r_{\max} = 2.5 \times 10^{-4}$ [m], the channel length $z_{\max} = 0.01$ [m].

- Reaction mechanisms: 7 gas-phase species, 11 surface species, 23 surface reactions. The gas-phase and surface reactions are given in the Appendix.

The initial values at the inlet are kept fixed: $\mathrm{X}_{\mathrm{CH_4}} = 0.2$, $\mathrm{X}_{\mathrm{O_2}} = 0.1$, $\mathrm{X}_{\mathrm{N_2}} = 0.7$, the inlet gas temperature $T_{\mathrm{gas}} = 300$ [K], the inlet velocity $u_0 = 0.5$ [m/s]. The ratio of catalytic active surface area to geometric surface area are kept fixed at one, $F_{\mathrm{cat/geo}} = 1$.

The wall temperature profile is optimized. We use a piecewise linear parameterization with 8 intervals. The objective is to maximize the mass fraction of carbon monoxide H_2 at the outlet. As constraint the temperature

	Nodes	DETCHEM$^{\text{CHANNEL}}$	BLAYER$^{\text{sim}}$	Speed up
CPU	12	654.6	19.4	33.74
Time	16	703.2	31.7	22.18
(secs)	20	812.97	45.38	17.91
	24	943.98	63.4	14.88
Number	12	1175	199	
of	16	1306	203	
Steps	20	1119	205	
	24	1175	204	

Table 5.3: CPU time and number of integration steps using DETCHEM$^{\text{CHANNEL}}$ and BLAYER$^{\text{sim}}$ (*ethane2-grids*).

is required to be between 600 [K] and 1800 [K].

The optimization was started with a constant temperature profile of 1200 [K] and the corresponding objective value of 0.011. The optimization run took 25 minutes computational time on a 2.5 GHz Pentium 4 Linux PC. In the optimal solution the objective value is 0.030. Figure 5.27 shows the temperature profile and Figures 5.28 and 5.29 the mass fractions of methane and carbon monoxide before and after optimization.

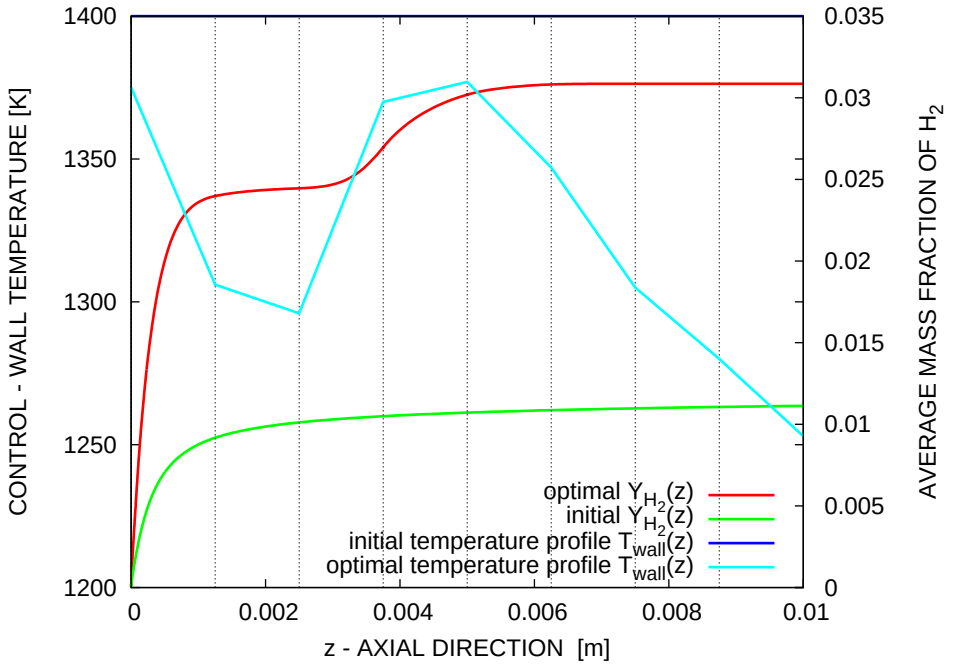

Figure 5.27: Temperature profile at the wall and the average mass fraction of H_2 at the initial and optimal solution.

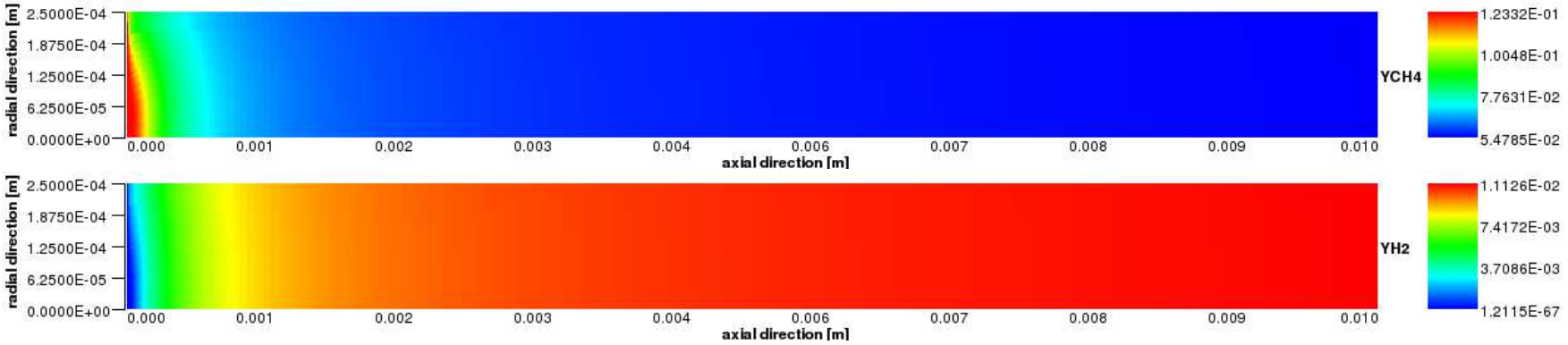

Figure 5.28: Mass fraction profiles of CH_4 and H_2 the initial setting.

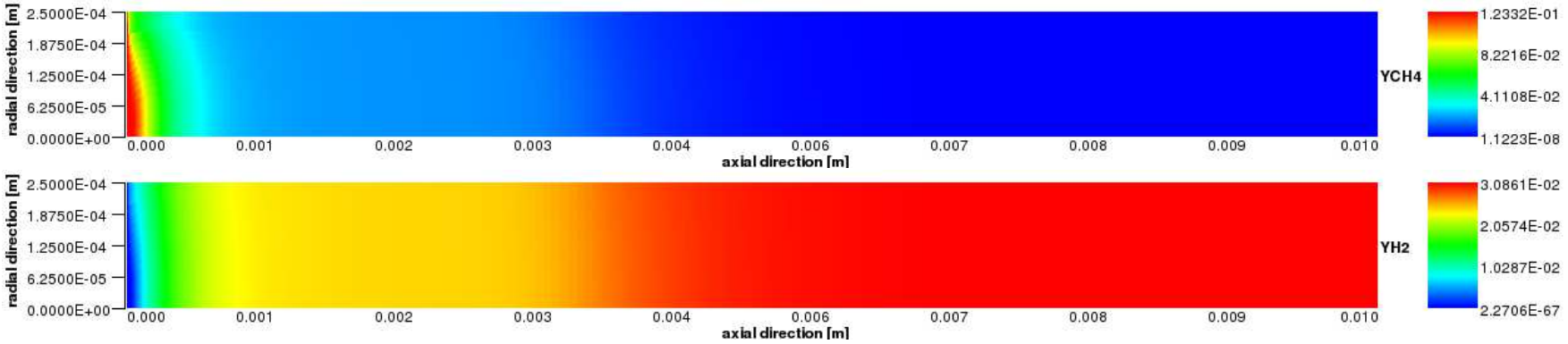

Figure 5.29: Mass fraction profiles of CH_4 and H_2 with the optimial setting.

5.3.2 Conversion of ethane to ethylene

The flow conditions are as the following [20].

- Channel geometry: the radius $r_{\max} = 2.5 \times 10^{-4}$ [m], the channel length $z_{\max} = 0.01$ [m].
- Reaction mechanisms: 25 gas-phase species, 20 surface species, 82 surface reactions, and 261 gas-phase reactions. The gas-phase and surface reactions are given in the Appendix.

For our optimization case study we keep the initial values at the inlet fixed: $\mathrm{Y}_{\mathrm{C_2H_6}} = 0.44$, $\mathrm{Y}_{\mathrm{O_2}} = 0.26$, $\mathrm{Y}_{\mathrm{N_2}} = 0.30$, and the inlet velocity $u_0 = 0.5$ [m/s].

(a) Control the wall temperature $T_{\mathrm{wall}}(z)$

The ratio of catalytic active surface area to geometric surface area is kept fixed at one, $F_{\mathrm{cat/geo}} = 1$, and the inlet gas temperature $T_{\mathrm{gas}} = 650$ [K]. The wall temperature profile is optimized. We use a piecewise linear parameterization with 8 intervals. The objective is to maximize the mass fraction of ethylene $\mathrm{Y}_{\mathrm{C_2H_4}}$ at the outlet. As constraint the temperature is required to be between 800 [K] and 1500 [K].

The optimization was started with a constant temperature profile of 930 [K] leading to an objective value of 0.132. The optimization run took 30 min computational time on a 2.5 GHz Pentium 4 Linux PC. In the optimal solution the objective value is 0.280, which is more than doubled the objective value with the standard setting. Figure 5.30 shows the temperature profile and Figures 5.31 and 5.32 show the mass fractions of ethane and ethylene before and after optimization. The results show that temperatures around 1300 K give maximum yield in the ethylene production. At inlet the temperatures only need to be sufficiently high enough for ignition of the combustion to occur. An autothermal reactor–where the temperature is only controlled by the exothermic reaction–should therefore maintain a temperature around 1300 K. This is nearly the same temperature as observed in experiments [70]. The optimal oxygen content can be determined by the amount of heat necessary to maintain this temperature.

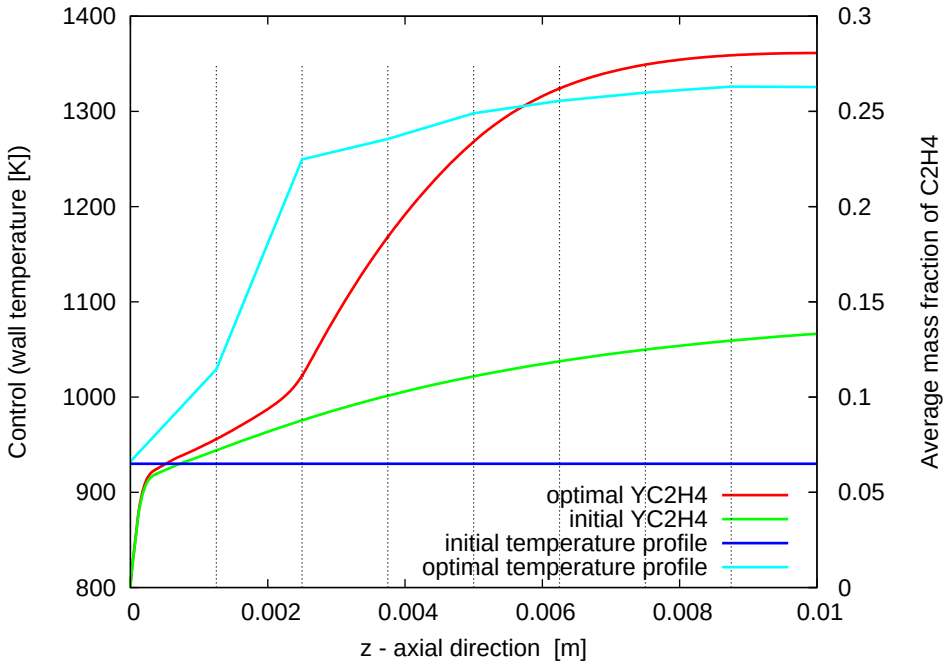

Figure 5.30: Temperature profile at the wall and the average mass fraction of C_2H_4 at the initial and optimal solution.

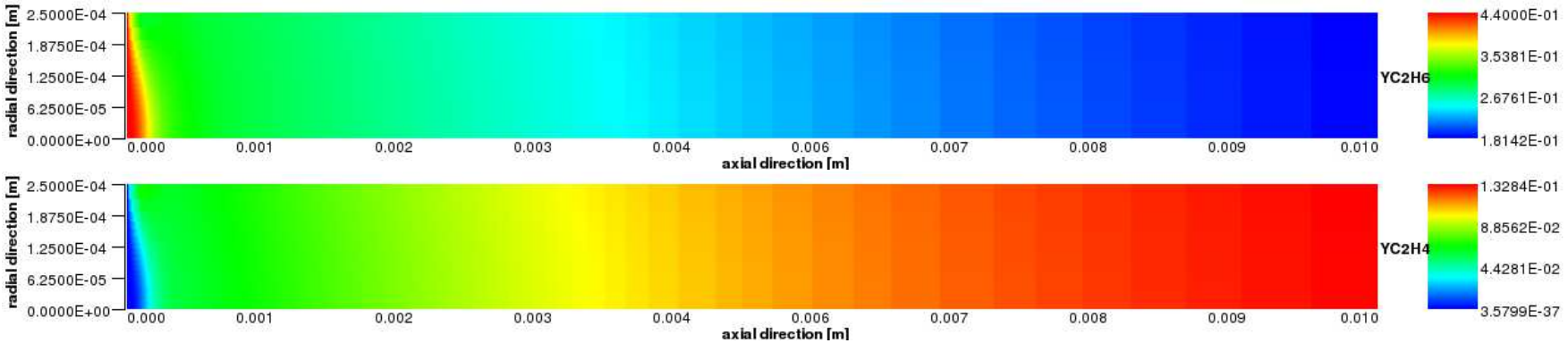

Figure 5.31: Mass fraction profiles of C_2H_6 and C_2H_4 with the initial setting.

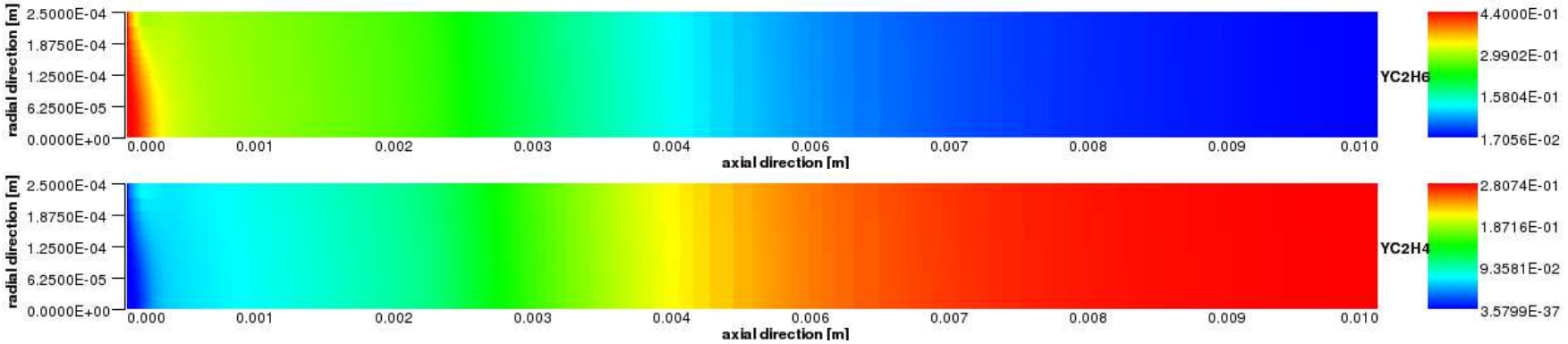

Figure 5.32: Mass fraction profiles of C_2H_6 and C_2H_4 with the optimial setting.

(b) Control $F_{\mathrm{cat/geo}}(z)$

The ratio of catalytic active surface area to geometric surface area $F_{\mathrm{cat/geo}}(z)$ is to be optimized. The inlet gas temperature is $T_{\mathrm{gas}} = 300$ [K], and the wall temperature $T_{\mathrm{wall}}(z)$ is kept fixed at 1000 [K]. The objective is to maximize the mass fraction of ethylene $\mathrm{Y}_{\mathrm{C_2H_4}}$ at the outlet. As constraint the $F_{\mathrm{cat/geo}}$ is required to be between 0 and 100.

The optimization was started with a constant $F_{\mathrm{cat/geo}}(z)$ profile of 20.0 leading to an objective value of 0.065. In the optimal solution the objective value is 0.191. Figure 5.33 shows the standard and optimal profiles of $F_{\mathrm{cat/geo}}$ and average mass fraction profiles of ethylene. Figures 5.34 and 5.35 show the mass fraction profiles of ethane and ethylene with the standard and optimal profiles of $F_{\mathrm{cat/geo}}(z)$.

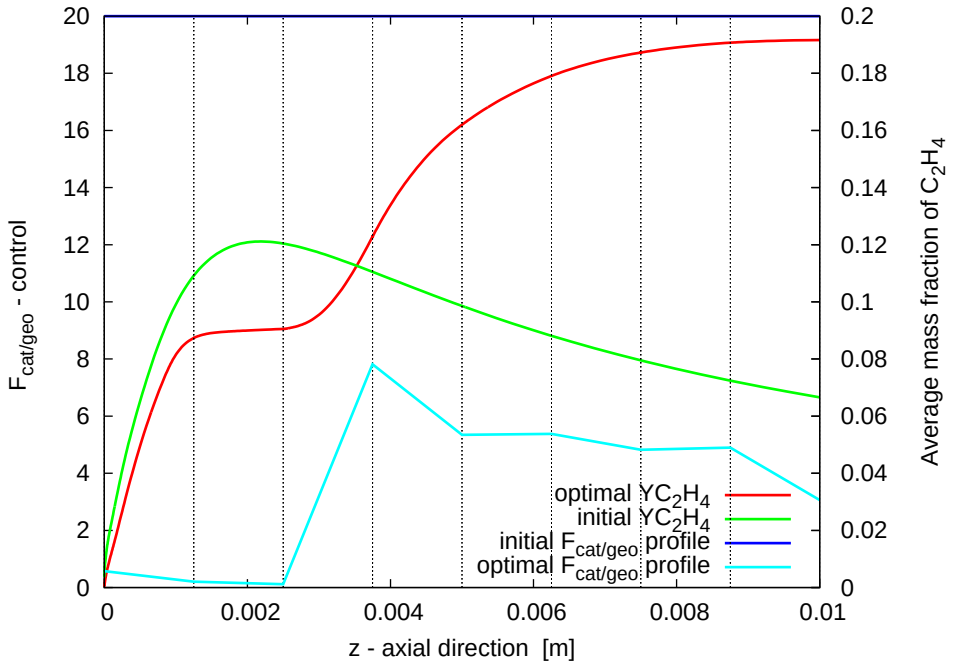

Figure 5.33: $F_{\mathrm{cat/geo}}(z)$ profile and the average mass fraction of C_2H_4 at the initial and at optimal solutions.

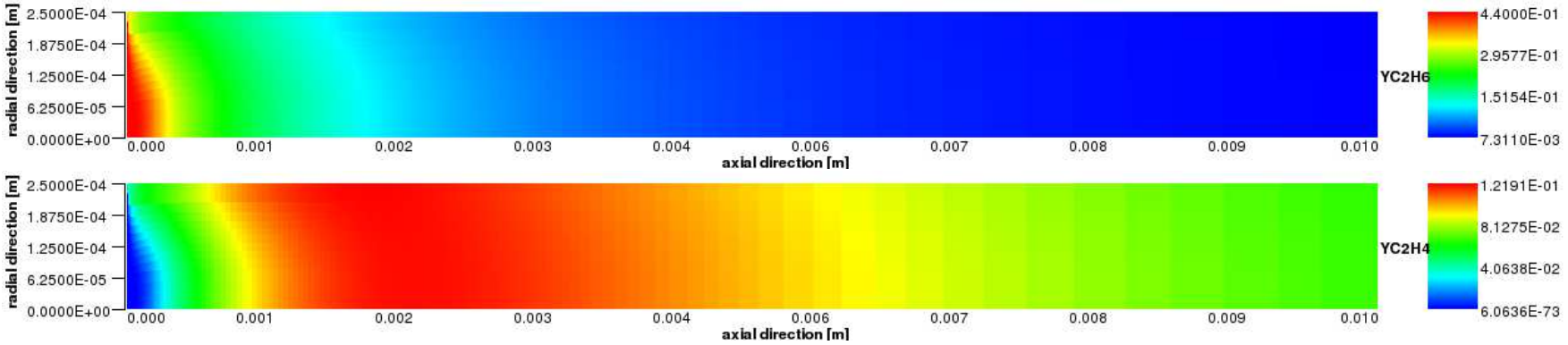

Figure 5.34: Mass fraction profiles of C_2H_6 and C_2H_4 with the initial setting.

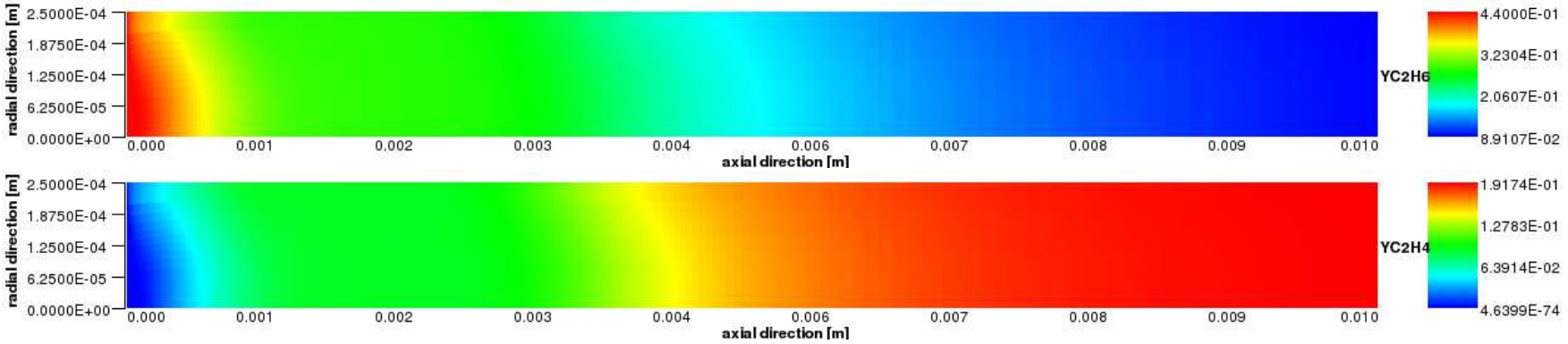

Figure 5.35: Mass fraction profiles of C_2H_6 and C_2H_4 with the optimial setting.

5.4 Summary

In this chapter, we have presented the simulation and optimization results. The comparison results between our simulation software BLAYER$^{\text{sim}}$ and DETCHEM$^{\text{CHANNEL}}$ shows that the numerical solutions are nearly the same except little differences in surface coverage profiles. BLAYER$^{\text{sim}}$ is more stable and faster than DETCHEM$^{\text{CHANNEL}}$, the speedup is about a factor of ten, to more than 60, depending on the applications. The optimization results obtained by BLAYER$^{\text{opt}}$ for two practical problems (a) catalytic combustion of methane and (b) conversion of ethane to ethylene show that the objective functions of the solutions with the optimal setting are more than doubled, to a factor of 3, compared with the solutions with the standard setting.

Chapter 6

Conclusions and Outlook

Summary and conclusions

The contributions of this thesis are summarized as follows.

- **Modeling for simulation and optimization of catalytic monoliths**

  The coupled fluid mechanics and chemical kinetics in a channel of catalytic monoliths are modeled by using the boundary layer approximation theory, the chemical kinetics are described by detailed chemistry models. Physically, this is a 3-D stationary problem, by taking radial symmetry into account, we obtain a 2-D stationary problem. The governing equations are a large system of parabolic partial differential equations (PDEs) coupled with highly nonlinear boundary conditions.

  To improve the performance of catalytic reactors (e.g., maximizing gas conversion or selectivity) we can control certain process conditions, such as temperature at the catalyst wall T_{wall} or the ratio of the catalytic active surface area to the geometric surface area $F_{\text{cat/geo}}$ or the gas ratio, temperature, velocity at inlet of the catalyst. This is for the first time generally formulated as a PDE-constrained optimal control problem.

- **Numerical methods for simulation of the chemically reacting flows in catalytic monoliths**

  The PDEs are semi-disretized in one direction using a non-uniform discretization scheme. The use of non-uniform grid is required for the resolution of high spatial gradients near the catalytic wall, and the boundary conditions are treated directly as algebraic constraints. This leads to a large system of stiff structured differential-algebraic equations

(DAEs). The resulting DAEs are structurally singular, stiff and of index 1.

- **Numerical methods for differential-algebraic equations**

  The DAEs are solved by an implicit method, based on backward differentiation formulas (BDF). Based on the BDF code DAESOL [10], we develop a new code DAESOLE, which allows us to solve DAEs, appearing in the problem under investigation. In particular, we exploit the block tridiagonal structure of the iteration matrix, which is the result of the semi-discretization of PDEs along with an appropriate ordering of the semi-discretized equations. From this observation, we apply a band solver for the linear sytem arising in the corrector iteration, and develop efficient methods for computing the derivatives of the model functions (functions defining the DAEs) with respect to the state variables. By identifying the structural orthogonal columns, we develop efficient methods for computation of the derivatives by finite differences or by automatic differentiation. A significant improvement in computing time is obtained.

  Solution of the DAEs also needs consistent initial values, which in turn requires the solution of highly nonlinear equations. That mainly arise from the nonlinear boundary conditions. These equations are solved by a time-stepping method combined with Newton's method.

  The linear equation systems in the corrector iterations are ill-conditioned. To treat this, we introduce an automatic scaling method, which allows us to obtain a better error bound. In particular, the condition numbers of the iteration matrices are reduced from 10^{16}–10^{22} to 10^{6}–10^{8}.

  We introduce a new error model for error analysis of numerical Newton's method, and analyze the limiting accuracy of the solution of nonlinear equations by numerical Newton's method. We also point out that some previous error models are inappropriate.

- **Numerical methods for solving the optimal control problem for catalytic monolits**

  To the best of our knowledge, this is the first time that a systematic approach for optimization of reactor conditions to catalytic monoliths is introduced. We apply the semi-discretization of one spatial direction to transform the PDE-constrained optimal control problem into a DAE-constrained one. We employ the direct shooting approach to approximate the infinite-dimensional optimal control problem by a finite

dimensional optimization problem, which is then solved by a sequential quadractic programming (SQP) method. The solution by the SQP method requires the derivatives of the objective and constraints with respect to the optimization variables. This in turn needs the derivatives of the solution of DAEs with respect to the optimization variables. These derivatives are computed by solving the sensitivity equations based on internal numerical differentiation (IND)[17], which is more robust and efficient than using external finite differences.

To obtain the sensitivity equations, we need the derivatives of model equations with respect to the state variables and the optimization variables. We apply the approach in [37] to compute these derivatives, which allows us to reduce dramatically the time for solving the optimization problem. For example, for solving the optimal control problem of conversion ethane to ethylene with complex reaction mechanisms [1] the computation time by the standard approach is 743 minutes and by our approach is only 9.2 minutes.

- **Software BLAYER**

  We have developed a software package BLAYER, which consists of two programs BLAYER$^{\text{sim}}$ for simulation and BLAYER$^{\text{opt}}$ for optimization of catalytic monoliths. The software package can be applied to different reaction mechanisms and channel settings with different initial/boundary conditions. Given conditions at inlet (velocity, temperature, pressure, mass/mole fraction), the temperature and $F_{\text{cat/geo}}$ at the wall, geometry of the channel (length and radius), and gas- and surface-phase reaction mechanisms with thermaldynamic data, BLAYER$^{\text{sim}}$ computes the flow field in the channel. BLAYER$^{\text{sim}}$ is more stable and faster than the existing software DETCHEM$^{\text{CHANNEL}}$, with a medium size problem, the speedup is a factor of 10, for large problem, the speedup is a factor of 62. We also avoid some abnormal solutions obtained by DETCHEM$^{\text{CHANNEL}}$.

  BLAYER$^{\text{opt}}$ can be used for optimization with different controls: initial values (gas temperature, mass/mole fractions at inlet), and/or temperature profile at the wall $T_{\text{wall}}(z)$, and $F_{\text{cat/geo}}(z)$. The objective to be minimized can be the mass fraction of certain species or the distribution of $F_{\text{cat/geo}}(z)$ at the wall, other objectives and controls (inlet velocity, radius and length of the channel) can be easily realized.

[1] 25 gas-phase species, 20 surface species, 82 surface reactions, and 261 gas-phase reactions, leading to 29 PDEs and 49 algebraic constraints, semi-discretizing with 12 grid points in the radial direction leads to 342 DAEs

We have developed robust and efficient numerical software for simulation and optimization of catalytic reaction processes in monoliths. They allow, e.g., for a better design and operation of the conversion of natural gas to higher hydrocarbon or the improvement of exhaust treatment in cars.

Directions for future research

Based on the results of this work, we suggest the following further research topics:

- Based on the BLAYER$^{\text{opt}}$, where the derivatives of the solution with respect to parameters are calculated efficiently, a sensitivity analysis tool can be developed for studying reaction mechanisms [57]. A sensitivity analysis tool would allow us to determine, e.g., which reations or species play an important role in the processes and how the solution is sensitive with such reactions or species. It can also identify which reactions or species are decisive ones. These functions are necessary for developing new reaction mechanisms.

- Model reductions by using *partial-equilibrium assumptions* and *intrinsic low-dimensional manifold* methods, see [83] and [84].

- Adaptive mesh methods for spatial discretization. Note that in this case multi-step methods, e.g., BDF methods, may not be more efficient than one-step methods, e.g., implicit Runge-Kutta methods, because of often changing of model equations and discontinuity due to remeshing. Although it is mentioned in [82] that the singly implicit Runge-Kutta (SIRK) methods for integration in time of an adaptive grid method for parabolic systems, which is described in [53], are more costly than a multistep method. The adaptive mesh methods for parabolic PDEs based on method of lines and BDF methods are realized in e.g., [82].

- Application of the software to other practical applications.

- Treatment of heat and mass transfer in the solid wall. In the current work, heat and mass transfer in the solid wall are not treated. However, this can be done by taking into account the governing equations for tranfering of heat and mass in porous media , see e.g., [45].

Alternatively, a shape optimization problem, which optimizes the shape of monoliths, e.g., distribution of holes (channel), could be an interesting prob-

lem [2].

In our opinion, the deterministic approach for modeling using detailed chemistry can be used for studying problems with hundreds of species. For problems with thousands of species (e.g., biochemical systems), stochastic models or a combination of stochastic models and deterministic models should be used instead because it is difficult, if not impossible, to obtain an accurate solution, and it is very inefficient. Moreover, the diffusion laws (Fick's law or similar ones) for species with very small amounts are suspected to be good approximations.

[2]Thanks to Prof. Dr. Dr. h. c. mult. Willi Jäger for suggesting this problem when I gave a talk at the annual meeting of the Graduiertenkolleg (post-graduate college)

Bibliography

[1] R. J. Allgor. *Modeling and computational issues in the development of batch processes*. PhD thesis, Department of Chemical Engineering, MIT, June 1997.

[2] P. Amodio and F. Mazzia. A new approach to backward error analysis of LU factorization. *BIT Numerical Mathematics*, 39(3):385–402, 1999.

[3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorenson. *LAPACK Users' Guide*. SIAM, Philadelphia, 3rd edition, 1999.

[4] U. Ascher, R. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problem for Ordinary Differential Equations*. Prentice Hall, Englewood Cliffs, New Jersey, 1988.

[5] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations*. SIAM, Philadephia, 1998.

[6] F. L. Bauer. Optimally scaled matrices. *Numerische Mathematik*, 5:73–87, 1963.

[7] F. L. Bauer. Remarks on optimally scaled matrices. *Numerische Mathematik*, 13:1–3, 1969.

[8] F. L. Bauer. Computational graphs and rounding error. *SIAM Journal on Numerical Analysis*, 11(1):87–96, Mar. 1974.

[9] F. L. Bauer, J. Stoer, and C. Witzgall. Absolute and monotonic norms. *Numerische Mathematik*, 3:257–264, 1961.

[10] I. Bauer, H. G. Bock, and J. P. Schlöder. DAESOL – a BDF-code for the numerical solution of differential algebraic equations. Technical report, SFB 359, IWR, University of Heidelberg, 1999.

[11] I. Bauer, F. Finocchi, W. Duschl, H. Gail, and J. Schlöder. Simulation of chemical reactions and dust destruction in protoplanetary accretion disks. *Astronomy & Astrophys.*, 317:273–289, 1997.

[12] R. A. D. Betta. Catalytic combustion gas turbine systems: the preferred technology for low emissions electric power production and cogeneration. *Catalysis Today*, 35(1–2):129–135, 1997.

[13] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland. ADIFOR - generating derivative codes from fortran programs. *Scientific Computing*, 1(1):1–29, 1992.

[14] C. Bischof, A. Carle, P. Hovland, P. Khademi, and A. Mauer. *ADIFOR 2.0 User's Guide*, 1995.

[15] C. Bischof, A. Carle, P. Hovland, P. Khademi, and A. Mauer. ADIFOR 2.0 user's guide (revision D). Technical Report CRPC-TR95516-S, Center for Research on Parallel Computation, Rice University, Houston, TX, 1995, revised 1998.

[16] C. Bischof, A. Carle, P. Khademi, and A. Mauer. The ADIFOR 2.0 system for the automatic differentiation of fortran 77 programs. Technical Report CRPC-TR94491, Center for Research on Parallel Computation, Rice University, Houston, TX, 1994.

[17] H. G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K. H. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, Springer Series in Chemical Physics 18, pages 102–125. Springer, Heidelberg, 1981.

[18] H. G. Bock. Recent advances in parameter identification techniques for O.D.E. In P. Deuflhard and E. Hairer, editors, *Numerical treatment of inverse problems in differential and integral equations*, Progress in Scientific Computing 2, pages 95–121. Birkhäuser, Boston, 1983.

[19] H. G. Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen.* Bonner Mathematische Schriften 183, 1987.

[20] H. G. Bock, O. Deutschmann, S. Körkel, L. Maier, H. D. Minh, J. P. Schlöder, S. Tischer, and J. Warnatz. Optimization of reactive flows in a single channel of a catalytic monolith: conversion of ethane to ethylene. In R. Rannacher et al, editor, *Reaktive Flows, Diffusion and Transport.* Springer Verlag, 2005.

[21] H. G. Bock, E. Kostina, and J. P. Schlöder. On the role of natural level functions to achieve global convergence for damped Newton methods. In M. J. D. Powell and S. Scholtes, editors, *System Modelling and Optimization*, pages 51–74. Kluwer Academic Publishers, Dordrecht, The Netherlands / Boston, MA, 2000.

[22] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4:1–52, 1996.

[23] P. T. Boggs, J. W. Tolle, and P. Wang. On the local convergence of quasi-Newton methods for constrained optimization. *SIAM Journal on Control and Optimization*, 20(2):161–171, 1982.

[24] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial–Value Problems in Differential–Algebraic Equations*. SIAM, Philadelphia, 1996.

[25] P. N. Brown and A. C. Hindmarsh. Reduced storage matrix method in stiff ODE systems. *Journal of Applied Mathematics & Computation*, 31:40–91, 1989.

[26] J. C. P. Bus. Convergence of Newton-like methods for solving systems of nonlinear equations. *Numerische Mathematik*, 27:271–281, 1977.

[27] G. D. Byrne and A. C. Hindmarsh. A polyalgorithm for the numerical solution of ordinary differential equations. *ACM Transactions on Mathematical Software*, 1(1):71–96, 1975.

[28] M. Caracotsios and W. E. Stewart. Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. *Computers & Chemical Engineering*, 9(4):359–365, 1985.

[29] C. P. Chou, J. Y. Chen, G. H. Evans, and W. S. Winters. Numerical studies of methane catalytic combustion inside a monolith honeycomb reactor using multi-step surface reactions. *Combustion Science and Technology*, 114:27–58, 2000.

[30] L. O. Chua and A. Deng. Impasse points. part i: numerical aspects, part ii: Analytical aspects. *International Journal of Circuit Theory and applications*, 17:213–235, 271–282, 1989.

[31] W. Cody and J. T. Coonen. Algorithm 722: Functions to support the IEEE standard for binary floating-point arithmetic. *ACM Transactions on Mathematical Software*, 19(4):443–451, 1993.

[32] T. Coffee and J. Heimerl. Transport algorithms for premixed laminar, steady-state flames. *Combustion Flame*, 43:273–289, 1981.

[33] T. Coleman and J. Moré. Estimation of sparse jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20:187–209, 1983.

[34] M. E. Coltrin, R. J. Kee, and F. M. Rupley. SURFACE CHEMKIN (Version 4.0): A Fortran package for analyzing heterogeneous chemical kinetics at a solid-surface - gas-phase interface. Technical Report SAND90–8003B, Sandia National Laboratories, 1990.

[35] M. E. Coltrin, H. K. Moffat, R. J. Kee, and F. M. Rupley. CRESLAF (verison 4.0): A Fortran program for modelling laminar, chemically reacting, boundary-layer flow in the cylindrical or planar channels. Technical Report SAND93–0478, Sandia National Laboratories, Apr 1993.

[36] A. R. Conn, N. I. M. Gould, and P. L. Toint. Methods for nonlinear constraints in optimization calculations. In I. Duff and A. Watson, editors, *The State of the Art in Numerical Analysis*, pages 363–390. Oxford University Press, New York, 1997.

[37] A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse Jacobian matrices. *Journal of the Institute of Mathematical Applications*, 13:117–119, 1974.

[38] C. Curtiss and J. Hirschfelder. Integration of stiff equations. *Proceedings of the National Academy of Sciences*, 38:235–243, 1952.

[39] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[40] J. Dennis, Jr and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice Hall Series in Computational Mathematics. Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632, 1983.

[41] J. E. Dennis, Jr. and H. F. Walker. Inaccuracy in quasi-Newton methods: Local improvement theorems. *Mathematical Programming Study*, 22:70–85, 1984.

[42] P. Deuflhard. A modified Newton method for the solution of ill-conditioned system of nonlinear equations with applications to multiple shooting. *Numerische Mathematik*, 22:289–315, 1974.

[43] P. Deuflhard. A relaxation strategy for the modified Newton method. In R. Bulirsch, W. Oettli, and J. Stoer, editors, *Optimization and Optimal Control*, Springer Lecture Notes in Mathematics 447, pages 59–73. Springer-Verlag, 1975.

[44] P. Deuflhard, E. Hairer, and J. Zugk. One-step and extrapolation methods for differential-algebraic equations. *Numerische Mathematik*, 51:501–516, 1987.

[45] O. Deutschmann. *Interactions between transport and chemistry in catalytic reactors*. Habilitationsschrift, Fakultät für Chemie, University of Heidelberg, 2001.

[46] O. Deutschmann, C. Correa, S. Tischer, D. Chatterjee, and J. Warnatz. *DETCHEM - PACKAGE, User manual, version 1.4.1*. IWR, University of Heidelberg, 2001.

[47] O. Deutschmann and L. D. Schmidt. Modeling the partial oxidation of methane in a short contact time reactor. *AIChEJ*, 44:2465–2476, 1998.

[48] A. Dieses. *Numerical methods for optimization problems in water flow and reactive solute transport processes of xenobiotics in soils*. PhD thesis, University of Heidelberg, 2000.

[49] A. Edelman. The complete pivoting conjecture for Gaussian elimination is false. *Mathematica Journal*, 2:58–61, 1992.

[50] W. F. Feehery, J. E. Tolsma, and P. I. Barton. Efficient sensitivity analysis of large-scale differential-algebraic systems. *Applied Numerical Mathematics*, 25:41–54, 1997.

[51] P. Fife. Toward the validity of Prandtl's approximation in a boundary layer. *Archive for Rational Mechanics and Analysis*, 18(1):1–13, 1965.

[52] B. A. Finlayson and L. C. Young. Mathematical models of the monolith catalytic converter: Part I. development of model and application of orthogonal collocation. *AIChE Journal*, 22(2):331–343, 1976.

[53] J. E. Flaherty, P. K. Moore, and C. Ozturan. Adaptive overlapping grid methods for parabolic systems. In J. E. Flaherty, P. J. Paslow, M. S. Shephard, and J. D. Vasilakis, editors, *Adaptive Methods for Partial Differential Equations*. SIAM, Philadelphia, 1989.

[54] B. Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–706, 1988.

[55] J. F. Gear, R. J. Kee, M. D. Smooke, and J. A. Miller. A hybrid Newton/time-integration procedure for the solution of steady, laminar, one-dimensional, premixed flames. *Proceedings of The Combustion Institute*, 21:1773–1782, 1986.

[56] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[57] P. Glarborg, R. J. Kee, and J. A. Miller. Kinetic modeling and sensitivity analysis on Nitrogen oxide formation in well stirred reactors. *Combustion & Flame*, 65:177–202, 1986.

[58] S. Goldstein, editor. *Modern Developments in Fluid Dynamics: An account of theory and experiment Relating to Boundary Layers, Turbulent Motion and Wakes*, volume I and II. Dover, New York, 1965.

[59] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Mathematical Sciences. The Johns Hopkins University Press, Baltimore and London, 2nd edition, 1989.

[60] G. H. Golub and J. M. Varah. On a characterization of the best l_2-scaling of a matrix. *SIAM Journal on Numerical Analysis*, 11(3):472–479, 1974.

[61] N. Gould. On growth in Gaussian elimination with complete pivoting. *SIAM Journal on Matrix Analysis and Applications*, 12:354–361, 1991.

[62] N. I. M. Gould and P. L. Toint. SQP methods for large-scale nonlinear programming. In M. J. D. Powell and S. Scholtes, editors, *System Modelling and Optimization*, pages 149–178. Kluwer Academic Publishers, Dordrecht, The Netherlands / Boston, MA, 2000.

[63] A. Griewank. On automatic differentiation. In *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, Amsterdam, 1989.

[64] A. Griewank. *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation*. Frontiers in Applied Mathematics 19. SIAM, Philadelphia, 2000.

[65] A. Griewank, D. Juedes, H. Mitev, J. Utke, O. Vogel, and A. Walther. ADOL-C: A package for the automatic differentiation of algorithms written in C/C++. *ACM TOMS*, 2(22):131–167, 1996.

[66] A. Griewank and S. Reese. On the calculation of Jacobian matrices by the Markowitz rule. In A. Griewank and G. F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 126–135. SIAM, Philadelphia, 1991.

[67] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.* Springer Series in Computational Mathematics 14. Springer-Verlag, 1996.

[68] K. E. Hillstrom. *Users Guide for JAKEF,* Technical Memorandum ANL/MCS-TM-16. Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL 60439, 1985.

[69] J. E. Horwedel, B. A. Worley, E. M. Oblow, and F. G. Pin. *GRESS Version 0.0 Users Manual,* ORNL/TM 10835. Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830, USA, 1988.

[70] M. Huff and L. D. Schmidt. Ethylene formation by oxidative dehydrogenation of ethane over monoliths at very short contact times. *Journal of Physical Chemistry*, 97(45):11815–11822, 1993.

[71] M. Iri and K. Kubota. *Methods of Fast Automatic Differentitation and Applications,* Research memorandum RMI 87-06. Department of Mathematical Engineering and Instrumentation Physics, Faculty of Engineering, University of Tokyo, 1987.

[72] K. R. Jackson and R. Sacks-Davis. An alternative implementation of variable step-size multistep formulas for stiff ODEs. *ACM Transactions on Mathematical Software*, 6(3):295–318, 1980.

[73] R. J. Kee, M. E. Coltrin, and P. Glarborg. *Chemically Reacting Flow: Theory and Practice.* Willey, Hoboken, New Jersey, 2003.

[74] R. J. Kee and J. A. Miller. A computational model for chemically reacting flow in boundary layers, shear layers, and ducts. Technical Report SAND81-8241, Sandia National Laboratories, Albuquerque, NM, 1981.

[75] R. J. Kee, F. M. Rupley, J. A. Miller, M. E. Coltrin, J. F. Grcar, E. Meeks, K. H. Moffat, A. E. Lutz, G. Dixon-Levis, M. D. Smooke, J. Warnatz, G. H. Evans, R. S. Larson, R. E. Mitchell, L. R. Petzold, W. C. Reynolds, M. Caracotsios, W. E. Stewart, P. Glarborg, C. Wang, and O. Adigun. *CHEMKIN Collection, Version 3.6.* Reaction Design, Inc, San Diego, 2000.

[76] R. J. Kee, J. Warnatz, and J. A. Miller. A Fortran computer code package for the evaluation of gas phase viscosities, heat conductivities, and diffusion coefficients. Technical Report SAND83-8209, Sandia National Laboratories, 1983.

[77] C. T. Kelley, C. T. Miller, and M. D. Tocci. Termination of Newton/chord iterations and the method of lines. *SIAM Journal on Scientific and Statistical Computing*, 19(1):280–290, 1998.

[78] M. A. Kramer and J. R. Leis. The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations. *ACM Transactions on Mathematical Software*, 14:44–60, 1988.

[79] P. Lancaster. Error analysis for the Newton-Raphson method. *Numerische Mathematik*, 9:55–68, 1966.

[80] D. B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*. PhD thesis, IWR, University of Heidelberg, 1999.

[81] S. Li, L. Petzold, and W. Zhu. Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem. *Applied Numerical Mathematics*, 32:161–174, 2000.

[82] S. Li, L. R. Petzold, and J. M. Hyman. Solution adapted mesh refinement and sensitivity analysis for parabolic partial differential equation systems. In L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, Lecture Notes in Computational Science and Engineering 30, pages 117–132. Springer, New York, 2003.

[83] U. Maas and S. B. Pope. Implementation of simplified chemical kinetics based on intrinsic low-dimensional manifolds. *Proceedings of The Combustion Institute*, 24:103–112, 1992.

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

[85] T. Maly and L. R. Petzold. Numerical methods and software for sensitivity analysis of differential-algebraic equations. *Applied Numerical Mathematics*, 20:57–79, 1997.

[86] S. E. Mattsson and G. Söderlind. Index reduction in differential-algebraic equations using dummy derivatives. *SIAM Journal on Scientific Computing*, 14(3):677–692, 1993.

[87] C. McCarthy and G. Strang. Optimal conditioning of matrices. *SIAM Journal on Numerical Analysis*, 10(2):370–388, 1973.

[88] A. D. McNaught and A. Wilkinson. *Compendium of Chemical Terminology*. Blackwell Science, the gold book, 2nd edition, 1997.

[89] A. Neumaier. Scaling and structural condition numbers. *Linear Algebra and its Applications*, 263:157–165, 1997.

[90] K. Nickel. Die Prandtlschen Grenzschichtdifferentialgleichungen als asymptotischer Grenzfall der Navier-Stokesschen und der Eulerschen Differentialgleichungen. *Archive for Rational Mechanics and Analysis*, 13(1):1–14, 1963.

[91] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operation Research. Springer, New York, 1999.

[92] O. Oleinik and V. Samokhin. *Mathematical Models in Boundary Layer Theory*. Applied Mathematics and Mathematical Computation 15. Chapman & Hall/CRC, Boca Raton, Florida, 1999.

[93] M. Olschowka and A. Neumaier. A new pivoting strategy for Gaussian elimination. *Linear Algebra and its Applications*, 240:131–151, 1996.

[94] J. Ortega and W. Rheinboldt. *Iterative solution of nonlinear equations in serveral variables*. Computer Science and Applied Mathematics. Academic Press, New York, 1970.

[95] C. C. Pantelides. The consistent intialization of differential-algebraic systems. *SIAM Journal on Scientific and Statistical Computing*, 9(2):213–231, 1988.

[96] B. N. Parlett and T. L. Landis. Methods for scaling to doubly stochastic form. *Linear Algebra and its Applications*, 48:53–79, 1982.

[97] L. R. Petzold. A description of DASSL: a differential/algebraic system solver. In R. S. Stepleman et al., editor, *Scientific Computing*, pages 65–68. North–Holland, Amsterdam, 1983.

[98] M. J. D. Powell. The performance of two subroutines for constrained optimization on some difficult test problems. In P. T. Boggs, R. H. Byrd, and R. B. Schnabel, editors, *Numerical Optimization 1984*, pages 160–177. SIAM, Philadelphia, 1985.

[99] L. Prandtl. Über Flüssigkeitsbewegungen bei sehr kleiner Reibung. In A. Krazer, editor, *Verhandlungen des dritten internationalen Mathematiker-Kongresses Heidelberg 1904*, pages 484–491. Teubner, Leipzig, 1905.

[100] P. J. Rabier and W. C. Rheinboldt. Theoretical and numerical analysis of differential-algebraic equations. In P. G. Ciarlet and J.-L. Lions, editors, *Handbook of Numerical Analysis*, volume VIII, pages 183–540. ELSEVIER, Amsterdam, 2002.

[101] L. L. Raja, R. J. Kee, O. Deutschmann, J. Warnatz, and L. D. Schmidt. A critical evaluation of Navier-Stokes, boundary-layer, and plug-flow models of the flow and chemistry in a catalytic-combustion monolith. *Catalysis Today*, 59:47–60, 2000.

[102] G. Reißig. Differential-algebraic equations and impasse points. *IEEE Transactions on Circuits and Systems. I. Fundamental Theory and Applications*, 43(2):122–133, 1996.

[103] W. E. Schiesser. *The Numerical Method of Lines, Integration of Partial Differential Equations*. Academmic Press, San Diego, CA, 1991.

[104] H. Schlichting. *Boundary Layer Theory*. McGraw–Hill, New York, 7th edition, 1979.

[105] R. Schwiedernoch. *Partial and Total Oxidation of Methane in Monolithic Catalysts at Short Contact Times*. PhD thesis, University of Heidelberg, July 2005.

[106] Y. S. Seo, S. K. Kang, and H. D. Shin. A catalytic burner using propane and toluene alternately for the drying of textile coatings. *International Journal of Energy Research*, 23(6):543–556, 1999.

[107] L. F. Shampine. Implementation of implicit formulas for the solution of ODEs. *SIAM Journal on Scientific and Statistical Computing*, 1(1):103–118, 1980.

[108] A. Shapiro. Optimally scalled matrices, necessary and sufficient conditions. *Numerische Mathematik*, 39:239–245, 1982.

[109] R. D. Skeel. Scaling for numerical stability in Gaussian elimination. *Journal of the Association for Computing Machinery*, 26(3):494–526, July 1979.

[110] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis*. Number 12. Springer, New York, 1992.

[111] F. Tisseur. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22(4):1038–1057, 2001.

[112] J. Unger, A. Kröner, and W. Marquardt. Structural analysis of differential-algebraic equation systems–theory and applications. *Computers & Chemical Engineering*, 19(8):867–882, 1995.

[113] A. van der Sluis. Condition numbers and equilibration of matrices. *Numerische Mathematik*, 14:14–23, 1969.

[114] A. van der Sluis. Condition, equilibration and pivoting in linear algebraic systems. *Numerische Mathematik*, 15:74–86, 1970.

[115] A. van der Sluis. Stability of solutions of linear algebraic systems. *Numerische Mathematik*, 14:246–251, 1970.

[116] J. von Neumann and H. H. Goldstine. Numerical inverting of matrices of high order. *Bulletin of the American Mathematical Society*, 53:1021–1099, 1947.

[117] J. Warnatz. Influence of transport models and boundary conditions on flame structure. In N. Peters and J. Warnatz, editors, *Numerical Methods in Flame Propagation*. Fridr. Vieweg and Sohn, Wiesbaden, 1982.

[118] J. Warnatz, R. Dibble, and U. Mass. *Combustion, Physical and Chemical Fundamentals, Modeling and Simulation, Experiments, Pullutant Formation*. Springer-Verlag, New York, 1996.

[119] G. A. Watson. An algorithm for optimal l_2 scaling of matrices. *IMA Journal of Numerical Analysis*, 11:481–492, 1991.

[120] J. F. Wendt, J. Anderson, G. Degrez, E. Dick, and R. Grundmann. *Computational Fluid Dynamics: An Introduction*. Springer, Berlin, New York, 2nd edition, 1996.

[121] J. Wilkinson. Error analysis of direct methods of matrix inversion. *Journal of the Association for Computing Machinery*, 8:281–330, 1961.

[122] J. Wilkinson. *Rounding errors in algebraic processes.* Prentice–Hall, Englewood Cliffs, NJ, 1963.

[123] J. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, London, 1965.

[124] F. A. Williams. *Combustion Theory.* Combustion Science and Engineering. Addison–Wesley, New York, 2nd edition, 1988.

[125] R. Winkler. On simple impasse points and their numerical computation. Technical Report PR-94-15, Institut für Mathematik, Humboldt-Universität, Berlin, 1994.

[126] H. Wozniakowski. Numerical stability for solving nonlinear equations. *Numerische Mathematik*, 27:373–390, 1977.

[127] T. J. Ypma. The effect of rounding errors on Newton-like methods. *IMA Journal of Numerical Analysis*, 3:109–118, 1983.

Appendix
Reaction Mechanisms

Reaction mechanisms in this appendix are taken from Prof. Dr. Olaf Deutschmann and Dr. Steffen Tischer, Institut für Technische Chemie und Polymerchemie, Universitt Karlsruhe.

A-1 Gas-Phase Reaction Mechanisms

Table 1: Gas-phase reaction mechanism of the NO-O2 reaction(P. Klaus 1997). $M(1)$ is third body. A, β, E are Arrhenius parameters for the rate constants written in the form: $k = AT^{\beta}\exp(-E/RT)$. The units of A are given in terms of moles, cubic meters, and seconds. E is in J/mol

Reaction	A	β	E
20 + M(1) → 02 + M(1)	2.90E+05	-1.00E+00	0.00E+00
O2 + M(1) → 2O + M(1)	6.77E+12	-1.00E+00	4.96E+05
NO2 + M(1) → NO + O + M(1)	1.10E+10	0.00E+00	2.75E+05
NO + O + M(1) → NO2 + M(1)	1.39E+02	0.00E+00	-2.58E+04
NO2 + O → NO + O2	1.00E+07	0.00E+00	2.51E+03
NO + O2 → NO2 + O	2.96E+06	0.00E+00	1.97E+05
2NO2 → 2NO + O2	1.60E+06	0.00E+00	1.09E+05
2NO + O2 → 2NO2	6.01E-03	0.00E+00	2.12E+03

Table 2: Gas-phase reaction mechanism of the methane oxidation. $M(i)$ is third body. A, β, E are Arrhenius parameters for the rate constants written in the form: $k = AT^{\beta}\exp(-E/RT)$. The units of A are given in terms of moles, cubic meters, and seconds. E is in J/mol.
$^{(*)}$ is non-Arrhenius reactions, Troe reactions .

Reaction mechanism	A	β	E
1. O2 + H → OH + O	0.00E+00	0.00E+00	0.00E+00
2. OH + O → O2 + H	0.00E+00	0.00E+00	0.00E+00
3. H2 + O → OH + H	0.00E+00	0.00E+00	0.00E+00
4. OH + H → H2 + O	0.00E+00	0.00E+00	0.00E+00
5. H2 + OH → H2O + H	0.00E+00	0.00E+00	0.00E+00
6. H2O + H → H2 + OH	0.00E+00	0.00E+00	0.00E+00

Table 2: continued

Reaction mechanism	A	β	E
7. OH + OH → H2O + O	0.00E+00	0.00E+00	0.00E+00
8. H2O + O → OH + OH	0.00E+00	0.00E+00	0.00E+00
9. O2 + H + M(2) → HO2 + M(2)	0.00E+00	0.00E+00	0.00E+00
10. HO2 + M(2) → O2 + H + M(2)	0.00E+00	0.00E+00	0.00E+00
11. H + HO2 → OH + OH	0.00E+00	0.00E+00	0.00E+00
12. OH + OH → H + HO2	0.00E+00	0.00E+00	0.00E+00
13. H + HO2 → H2 + O2	0.00E+00	0.00E+00	0.00E+00
14. H2 + O2 → H + HO2	0.00E+00	0.00E+00	0.00E+00
15. H + HO2 → H2O + O	0.00E+00	0.00E+00	0.00E+00
16. H2O + O → H + HO2	0.00E+00	0.00E+00	0.00E+00
17. O + HO2 → O2 + OH	0.00E+00	0.00E+00	0.00E+00
18. O2 + OH → O + HO2	0.00E+00	0.00E+00	0.00E+00
19. OH + HO2 → O2 + H2O	0.00E+00	0.00E+00	0.00E+00
20. O2 + H2O → OH + HO2	0.00E+00	0.00E+00	0.00E+00
21. HO2 + HO2 → O2 + H2O2	0.00E+00	0.00E+00	0.00E+00
22. O2 + H2O2 → HO2 + HO2	0.00E+00	0.00E+00	0.00E+00
23. OH + OH + M(2) → H2O2 + M(2)	0.00E+00	0.00E+00	0.00E+00
24. H2O2 + M(2) → OH + OH + M(2)	0.00E+00	0.00E+00	0.00E+00
25. H + H2O2 → H2 + HO2	0.00E+00	0.00E+00	0.00E+00
26. H2 + HO2 → H + H2O2	0.00E+00	0.00E+00	0.00E+00
27. OH + H2O2 → H2O + HO2	0.00E+00	0.00E+00	0.00E+00
28. H2O + HO2 → OH + H2O2	0.00E+00	0.00E+00	0.00E+00
29. CO + OH → CO2 + H	0.00E+00	0.00E+00	0.00E+00
30. CO2 + H → CO + OH	0.00E+00	0.00E+00	0.00E+00
31. CO + HO2 → CO2 + OH	0.00E+00	0.00E+00	0.00E+00
32. CO2 + OH → CO + HO2	0.00E+00	0.00E+00	0.00E+00
33. CO + O + M(2) → CO2 + M(2)	0.00E+00	0.00E+00	0.00E+00
34. CO2 + M(2) → CO + O + M(2)	0.00E+00	0.00E+00	0.00E+00
35. O2 + CO → CO2 + O	0.00E+00	0.00E+00	0.00E+00
36. CO2 + O → O2 + CO	0.00E+00	0.00E+00	0.00E+00
37. CHO + M(2) → CO + H + M(2)	0.00E+00	0.00E+00	0.00E+00
38. CO + H + M(2) → CHO + M(2)	0.00E+00	0.00E+00	0.00E+00
39. O2 + CHO → CO + HO2	0.00E+00	0.00E+00	0.00E+00
40. CO + HO2 → O2 + CHO	0.00E+00	0.00E+00	0.00E+00
41. CH2O + M(2) → H + CHO + M(2)	0.00E+00	0.00E+00	0.00E+00
42. H + CHO + M(2) → CH2O + M(2)	0.00E+00	0.00E+00	0.00E+00
43. H + CH2O → H2 + CHO	0.00E+00	0.00E+00	0.00E+00
44. H2 + CHO → H + CH2O	0.00E+00	0.00E+00	0.00E+00
45. O + CH2O → OH + CHO	0.00E+00	0.00E+00	0.00E+00
46. OH + CHO → O + CH2O	0.00E+00	0.00E+00	0.00E+00
47. OH + CH2O → H2O + CHO	0.00E+00	0.00E+00	0.00E+00
48. H2O + CHO → OH + CH2O	0.00E+00	0.00E+00	0.00E+00
49. HO2 + CH2O → H2O2 + CHO	0.00E+00	0.00E+00	0.00E+00
50. H2O2 + CHO → HO2 + CH2O	0.00E+00	0.00E+00	0.00E+00
51. CH2O + CH3 → CH4 + CHO	0.00E+00	0.00E+00	0.00E+00
52. CH4 + CHO → CH2O + CH3	0.00E+00	0.00E+00	0.00E+00
53. O2 + CH2O → HO2 + CHO	0.00E+00	0.00E+00	0.00E+00
54. HO2 + CHO → O2 + CH2O	0.00E+00	0.00E+00	0.00E+00
55. O + CH3 → H + CH2O	0.00E+00	0.00E+00	0.00E+00
56. H + CH2O → O + CH3	0.00E+00	0.00E+00	0.00E+00
57. CH4 + M(3) → H + CH3 + M(3)	1.14E+06	0.00E+00	0.00E+00$^{(*)}$
58. H + CH3 + M(3) → CH4 + M(3)	1.99E+08	0.00E+00	2.86E+05$^{(*)}$

Table 2: continued

Reaction mechanism	A	β	E
59. OH + CH3 → H + CH3O	0.00E+00	0.00E+00	0.00E+00
60. H + CH3O → OH + CH3	0.00E+00	0.00E+00	0.00E+00
61. O2 + CH3 → OH + CH2O	0.00E+00	0.00E+00	0.00E+00
62. HO2 + CH3 → OH + CH3O	0.00E+00	0.00E+00	0.00E+00
63. OH + CH3O → HO2 + CH3	0.00E+00	0.00E+00	0.00E+00
64. HO2 + CH3 → O2 + CH4	0.00E+00	0.00E+00	0.00E+00
65. O2 + CH4 → HO2 + CH3	0.00E+00	0.00E+00	0.00E+00
66. CH3 + CH3 → H2 + C2H4	0.00E+00	0.00E+00	0.00E+00
67. CH3 + CH3 + M(2) → C2H6 + M(2)	1.40E+06	0.00E+00	0.00E+00(*)
68. C2H6 + M(2) → CH3 + CH3 + M(2)	4.12E+08	0.00E+00	2.62E+05(*)
69. CH3O + M(2) → H + CH2O + M(2)	0.00E+00	0.00E+00	0.00E+00
70. H + CH2O + M(2) → CH3O + M(2)	0.00E+00	0.00E+00	0.00E+00
71. H + CH3O → H2 + CH2O	0.00E+00	0.00E+00	0.00E+00
72. H2 + CH2O → H + CH3O	0.00E+00	0.00E+00	0.00E+00
73. O2 + CH3O → HO2 + CH2O	0.00E+00	0.00E+00	0.00E+00
74. HO2 + CH2O → O2 + CH3O	0.00E+00	0.00E+00	0.00E+00
75. O + CH3O → O2 + CH3	0.00E+00	0.00E+00	0.00E+00
76. O2 + CH3 → O + CH3O	0.00E+00	0.00E+00	0.00E+00
77. CH4 + H → H2 + CH3	0.00E+00	0.00E+00	0.00E+00
78. H2 + CH3 → CH4 + H	0.00E+00	0.00E+00	0.00E+00
79. CH4 + O → OH + CH3	0.00E+00	0.00E+00	0.00E+00
80. OH + CH3 → CH4 + O	0.00E+00	0.00E+00	0.00E+00
81. CH4 + OH → H2O + CH3	0.00E+00	0.00E+00	0.00E+00
82. H2O + CH3 → CH4 + OH	0.00E+00	0.00E+00	0.00E+00
83. CH4 + HO2 → H2O2 + CH3	0.00E+00	0.00E+00	0.00E+00
84. H2O2 + CH3 → CH4 + HO2	0.00E+00	0.00E+00	0.00E+00
85. C2H3 + M(2) → C2H2 + H + M(2)	1.40E+03	1.50E+00	3.11E+04(*)
86. C2H2 + H + M(2) → C2H3 + M(2)	2.91E+01	1.50E+00	5.19E+04(*)
87. OH + C2H3 → H2O + C2H2	0.00E+00	0.00E+00	0.00E+00
88. H2O + C2H2 → OH + C2H3	0.00E+00	0.00E+00	0.00E+00
89. H + C2H3 → H2 + C2H2	0.00E+00	0.00E+00	0.00E+00
90. H2 + C2H2 → H + C2H3	0.00E+00	0.00E+00	0.00E+00
91. O2 + C2H3 → CHO + CH2O	0.00E+00	0.00E+00	0.00E+00
92. CHO + CH2O → O2 + C2H3	0.00E+00	0.00E+00	0.00E+00
93. C2H4 + M(2) → H2 + C2H2 + M(2)	0.00E+00	0.00E+00	0.00E+00
94. H2 + C2H2 + M(2) → C2H4 + M(2)	0.00E+00	0.00E+00	0.00E+00
95. C2H4 + M(2) → H + C2H3 + M(2)	0.00E+00	0.00E+00	0.00E+00
96. H + C2H3 + M(2) → C2H4 + M(2)	0.00E+00	0.00E+00	0.00E+00
97. C2H4 + H → H2 + C2H3	0.00E+00	0.00E+00	0.00E+00
98. H2 + C2H3 → C2H4 + H	0.00E+00	0.00E+00	0.00E+00
99. C2H4 + O → CHO + CH3	0.00E+00	0.00E+00	0.00E+00
100. CHO + CH3 → C2H4 + O	0.00E+00	0.00E+00	0.00E+00
101. C2H4 + OH → H2O + C2H3	0.00E+00	0.00E+00	0.00E+00
102. H2O + C2H3 → C2H4 + OH	0.00E+00	0.00E+00	0.00E+00
103. C2H4 + H + M(2) → C2H5 + M(2)	1.00E+03	1.50E+00	2.44E+04(*)
104. C2H5 + M(2) → C2H4 + H + M(2)	9.18E+00	1.50E+00	3.73E+04(*)
105. H + C2H5 → CH3 + CH3	0.00E+00	0.00E+00	0.00E+00
106. CH3 + CH3 → H + C2H5	0.00E+00	0.00E+00	0.00E+00
107. O2 + C2H5 → C2H4 + HO2	0.00E+00	0.00E+00	0.00E+00
108. C2H4 + HO2 → O2 + C2H5	0.00E+00	0.00E+00	0.00E+00
109. CH3 + C2H5 → CH4 + C2H4	0.00E+00	0.00E+00	0.00E+00

Table 2: continued

Reaction mechanism	A	β	E
110. CH4 + C2H4 → CH3 + C2H5	0.00E+00	0.00E+00	0.00E+00
111. C2H5 + C2H5 → C2H6 + C2H4	0.00E+00	0.00E+00	0.00E+00
112. C2H6 + C2H4 → C2H5 + C2H5	0.00E+00	0.00E+00	0.00E+00
113. C2H6 + H → H2 + C2H5	0.00E+00	0.00E+00	0.00E+00
114. H2 + C2H5 → C2H6 + H	0.00E+00	0.00E+00	0.00E+00
115. C2H6 + O → OH + C2H5	0.00E+00	0.00E+00	0.00E+00
116. OH + C2H5 → C2H6 + O	0.00E+00	0.00E+00	0.00E+00
117. C2H6 + OH → H2O + C2H5	0.00E+00	0.00E+00	0.00E+00
118. H2O + C2H5 → C2H6 + OH	0.00E+00	0.00E+00	0.00E+00
119. C2H6 + HO2 → H2O2 + C2H5	0.00E+00	0.00E+00	0.00E+00
120. H2O2 + C2H5 → C2H6 + HO2	0.00E+00	0.00E+00	0.00E+00
121. O2 + C2H6 → HO2 + C2H5	0.00E+00	0.00E+00	0.00E+00
122. HO2 + C2H5 → O2 + C2H6	0.00E+00	0.00E+00	0.00E+00
123. C2H6 + CH3 → CH4 + C2H5	0.00E+00	0.00E+00	0.00E+00
124. CH4 + C2H5 → C2H6 + CH3	0.00E+00	0.00E+00	0.00E+00

Table 3: Catalytic conversion of ethane to ethylene. $M(i)$ is third body. A, β, E are Arrhenius parameters for the rate constants written in the form: $k = AT^{\beta}\exp(-E/RT)$. The units of A are given in terms of moles, cubic meters, and seconds. E is in J/mol.
$^{(*)}$ is non-Arrhenius reactions, Troe reactions .

Reaction mechanism	A	β	E
1. H2 + OH → H + H2O	2.14E+02	1.52E+00	1.44E+04
2. H + H2O → H2 + OH	9.53E+02	1.52E+00	7.78E+04
3. O + OH → H + O2	2.02E+08	-4.00E-01	0.00E+00
4. H + O2 → O + OH	2.76E+09	-4.00E-01	6.82E+04
5. H2 + O → H + OH	5.06E-02	2.67E+00	2.63E+04
6. H + OH → H2 + O	2.24E-02	2.67E+00	1.85E+04
7. H + O2 + M(2) → HO2 + M(2)	3.00E+07	0.00E+00	0.00E+00$^{(*)}$
8. HO2 + M(2) → H + O2 + M(2)	9.70E+08	0.00E+00	7.47E+05$^{(*)}$
9. H2 + H + O2 → H2 + HO2	3.29E+15	-3.30E+00	1.20E+04$^{(*)}$
10. H2 + HO2 → H2 + H + O2	4.15E+16	-3.30E+00	2.65E+05$^{(*)}$
11. H + O2 + H2O → HO2 + H2O	3.29E+15	-3.30E+00	1.20E+04$^{(*)}$
12. HO2 + H2O → H + O2 + H2O	2.78E+11	-3.30E+00	3.52E+05$^{(*)}$
13. OH + HO2 → O2 + H2O	2.13E+22	-4.83E+00	1.46E+04
14. O2 + H2O → OH + HO2	2.59E+23	-4.83E+00	3.18E+05
15. OH + HO2 → O2 + H2O	9.10E+08	0.00E+00	4.59E+04
16. O2 + H2O → OH + HO2	1.11E+10	0.00E+00	3.49E+05
17. H + HO2 → OH + OH	1.50E+08	0.00E+00	4.18E+03
18. OH + OH → H + HO2	1.33E+07	0.00E+00	1.68E+05
19. H + HO2 → H2 + O2	8.45E+05	6.50E-01	5.19E+03
20. H2 + O2 → H + HO2	2.31E+06	6.50E-01	2.45E+05
21. H + HO2 → O + H2O	3.01E+07	0.00E+00	7.20E+03
22. O + H2O → H + HO2	2.68E+07	0.00E+00	2.42E+05
23. O + HO2 → O2 + OH	3.25E+07	0.00E+00	0.00E+00
24. O2 + OH → O + HO2	3.93E+07	0.00E+00	2.32E+05
25. OH + OH → O + H2O	3.57E-02	2.40E+00	-8.84E+03
26. O + H2O → OH + OH	3.59E-01	2.40E+00	6.24E+04
27. H + H + M(3) → H2 + M(3)	1.00E+06	-1.00E+00	0.00E+00
28. H2 + M(3) → H + H + M(3)	3.88E+12	-1.00E+00	4.36E+05

Table 3: continued

Reaction mechanism	A	β	E
29. H2 + H + H → H2 + H2	9.20E+04	-6.00E-01	0.00E+00
30. H2 + H2 → H2 + H + H	3.57E+11	-6.00E-01	4.36E+05
31. H + H + H2O → H2 + H2O	6.00E+07	-1.25E+00	0.00E+00
32. H2 + H2O → H + H + H2O	2.33E+14	-1.25E+00	4.36E+05
33. H + OH + M(4) → H2O + M(4)	2.21E+10	-2.00E+00	0.00E+00
34. H2O + M(4) → H + OH + M(4)	3.82E+17	-2.00E+00	4.99E+05
35. H + O + M(4) → OH + M(4)	4.71E+06	-1.00E+00	0.00E+00
36. OH + M(4) → H + O + M(4)	8.08E+12	-1.00E+00	4.28E+05
37. O + O + M(1) → O2 + M(1)	1.89E+01	0.00E+00	-7.48E+03
38. O2 + M(1) → O + O + M(1)	4.44E+08	0.00E+00	4.89E+05
39. HO2 + HO2 → O2 + H2O2	4.20E+08	0.00E+00	5.01E+04
40. O2 + H2O2 → HO2 + HO2	1.69E+09	0.00E+00	2.25E+05
41. HO2 + HO2 → O2 + H2O2	1.30E+05	0.00E+00	-6.82E+03
42. O2 + H2O2 → HO2 + HO2	5.24E+05	0.00E+00	1.68E+05
43. OH + OH + M(1) → H2O2 + M(1)	1.01E+15	-3.30E+00	6.31E+03$^{(*)}$
44. H2O2 + M(1) → OH + OH + M(1)	3.31E+17	-3.30E+00	7.82E+05$^{(*)}$
45. H + H2O2 → H2 + HO2	1.98E+00	2.00E+00	1.02E+04
46. H2 + HO2 → H + H2O2	1.34E+00	2.00E+00	7.53E+04
47. H + H2O2 → OH + H2O	3.07E+07	0.00E+00	1.76E+04
48. OH + H2O → H + H2O2	8.19E+06	0.00E+00	3.10E+05
49. O + H2O2 → OH + HO2	9.55E+00	2.00E+00	1.66E+04
50. OH + HO2 → O + H2O2	2.86E+00	2.00E+00	7.39E+04
51. OH + H2O2 → HO2 + H2O	2.40E-06	4.04E+00	-9.04E+03
52. HO2 + H2O → OH + H2O2	7.25E-06	4.04E+00	1.19E+05
53. CH3 + CH3 + M(5) → C2H6 + M(5)	7.28E+13	-2.54E+00	7.57E+03$^{(*)}$
54. C2H6 + M(5) → CH3 + CH3 + M(5)	7.66E+14	-2.54E+00	7.50E+05$^{(*)}$
55. H + CH3 + M(5) → CH4 + M(5)	1.29E+14	-3.30E+00	1.19E+03$^{(*)}$
56. CH4 + M(5) → H + CH3 + M(5)	5.23E+13	-3.30E+00	3.06E+05$^{(*)}$
57. H + CH4 → H2 + CH3	2.20E-02	3.00E+00	3.66E+04
58. H2 + CH3 → H + CH4	7.70E-04	3.00E+00	3.28E+04
59. CH4 + OH → CH3 + H2O	4.19E+00	2.00E+00	1.07E+04
60. CH3 + H2O → CH4 + OH	6.53E-01	2.00E+00	7.02E+04
61. CH4 + O → CH3 + OH	6.92E+02	1.56E+00	3.55E+04
62. CH3 + OH → CH4 + O	1.07E+01	1.56E+00	2.38E+04
63. CH4 + HO2 → CH3 + H2O2	1.12E+07	0.00E+00	1.03E+05
64. CH3 + H2O2 → CH4 + HO2	5.78E+05	0.00E+00	3.41E+04
65. CH3 + HO2 → OH + CH3O	7.00E+06	0.00E+00	0.00E+00
66. OH + CH3O → CH3 + HO2	2.67E+07	0.00E+00	1.12E+05
67. CH3 + HO2 → CH4 + O2	3.00E+06	0.00E+00	0.00E+00
68. CH4 + O2 → CH3 + HO2	2.34E+08	0.00E+00	2.44E+05
69. CH3 + O → H + CH2O	8.00E+07	0.00E+00	0.00E+00
70. H + CH2O → CH3 + O	1.11E+09	0.00E+00	2.92E+05
71. CH3 + O2 → O + CH3O	1.45E+07	0.00E+00	1.22E+05
72. O + CH3O → CH3 + O2	4.57E+07	0.00E+00	1.66E+03
73. CH3 + O2 → CH2O + OH	2.51E+05	0.00E+00	6.13E+04
74. CH2O + OH → CH3 + O2	2.55E+05	0.00E+00	2.85E+05
75. H + CH3O → CH3 + OH	1.00E+08	0.00E+00	0.00E+00
76. CH3 + OH → H + CH3O	2.32E+06	0.00E+00	5.23E+04
77. CH3 + OH → H2O + 3CH2	3.00E+00	2.00E+00	1.05E+04
78. H2O + 3CH2 → CH3 + OH	2.43E+00	2.00E+00	5.27E+04
79. CH3 + OH → H2 + CH2O	5.48E+07	0.00E+00	1.25E+04

Table 3: continued

Reaction mechanism	A	β	E
80. H2 + CH2O → CH3 + OH	1.72E+09	0.00E+00	3.13E+05
81. CH3 + OH → H2 + CH2O	2.25E+07	0.00E+00	1.80E+04
82. H2 + CH2O → CH3 + OH	7.07E+08	0.00E+00	3.18E+05
83. H + CH3 → H2 + 3CH2	9.00E+07	0.00E+00	6.32E+04
84. H2 + 3CH2 → H + CH3	1.64E+07	0.00E+00	4.21E+04
85. CH3 + M(1) → H + 3CH2 + M(1)	1.96E+10	0.00E+00	3.82E+05
86. H + 3CH2 + M(1) → CH3 + M(1)	9.19E+02	0.00E+00	-7.47E+04
87. H + CH2O + M(6) → CH3O + M(6)	4.00E+07	0.00E+00	0.00E+00$^{(*)}$
88. CH3O + M(6) → H + CH2O + M(6)	1.34E+10	0.00E+00	2.57E+05$^{(*)}$
89. H + CH3O → H2 + CH2O	2.00E+07	0.00E+00	0.00E+00
90. H2 + CH2O → H + CH3O	1.46E+07	0.00E+00	3.53E+05
91. OH + CH3O → CH2O + H2O	1.00E+07	0.00E+00	0.00E+00
92. CH2O + H2O → OH + CH3O	3.24E+07	0.00E+00	4.16E+05
93. O + CH3O → CH2O + OH	1.00E+07	0.00E+00	0.00E+00
94. CH2O + OH → O + CH3O	3.22E+06	0.00E+00	3.45E+05
95. O2 + CH3O → CH2O + HO2	6.30E+04	0.00E+00	1.09E+04
96. CH2O + HO2 → O2 + CH3O	1.68E+04	0.00E+00	1.23E+05
97. CH2O + OH → CHO + H2O	2.00E+07	0.00E+00	0.00E+00
98. CHO + H2O → CH2O + OH	6.47E+06	0.00E+00	1.23E+05
99. H + CH2O → H + CH2O	2.00E+08	0.00E+00	0.00E+00
100. H + CH2O → H + CH2O	2.00E+08	0.00E+00	0.00E+00
101. CH2O + O → H + H + CO2	5.00E+07	0.00E+00	0.00E+00
102. H + H + CO2 → CH2O + O	3.35E+02	0.00E+00	8.69E+04
103. CH2O + O → H + CO + OH	3.00E+07	0.00E+00	0.00E+00
104. H + CO + OH → CH2O + O	1.45E+00	0.00E+00	-1.03E+04
105. CH2O + O2 → H + CO2 + OH	5.00E+06	0.00E+00	0.00E+00
106. H + CO2 + OH → CH2O + O2	2.45E+00	0.00E+00	1.87E+04
107. CH2O + O2 → CO2 + H2O	3.00E+07	0.00E+00	0.00E+00
108. CO2 + H2O → CH2O + O2	2.54E+08	0.00E+00	5.18E+05
109. OH + 3CH2 → H + CH2O	2.50E+07	0.00E+00	0.00E+00
110. H + CH2O → OH + 3CH2	4.32E+09	0.00E+00	3.21E+05
111. CO2 + 3CH2 → CH2O + CO	1.10E+05	0.00E+00	4.18E+03
112. CH2O + CO → CO2 + 3CH2	1.37E+05	0.00E+00	2.28E+05
113. O + 3CH2 → H + H + CO	5.00E+07	0.00E+00	0.00E+00
114. H + H + CO → O + 3CH2	4.17E+02	0.00E+00	3.11E+05
115. O + 3CH2 → H2 + CO	3.00E+07	0.00E+00	0.00E+00
116. H2 + CO → O + 3CH2	9.70E+08	0.00E+00	7.47E+05
117. O2 + 3CH2 → CH2O + O	3.29E+15	-3.30E+00	1.20E+04
118. CH2O + O → O2 + 3CH2	4.15E+16	-3.30E+00	2.65E+05
119. O2 + 3CH2 → H + H + CO2	3.29E+15	-3.30E+00	1.20E+04
120. H + H + CO2 → O2 + 3CH2	2.78E+11	-3.30E+00	3.52E+05
121. O2 + 3CH2 → H2 + CO2	1.01E+15	-3.30E+00	6.31E+03
122. H2 + CO2 → O2 + 3CH2	3.31E+17	-3.30E+00	7.82E+05
123. O2 + 3CH2 → CO + H2O	7.28E+13	-2.54E+00	7.57E+03
124. CO + H2O → O2 + 3CH2	7.66E+14	-2.54E+00	7.50E+05
125. O2 + 3CH2 → CHO + OH	1.29E+14	-3.30E+00	1.19E+03
126. CHO + OH → O2 + 3CH2	5.23E+13	-3.30E+00	3.06E+05
127. CH3 + 3CH2 → H + C2H4	4.00E+07	0.00E+00	0.00E+00
128. H + C2H4 → CH3 + 3CH2	1.34E+10	0.00E+00	2.57E+05
129. 3CH2 + 3CH2 → H + H + C2H2	4.00E+07	0.00E+00	0.00E+00
130. H + H + C2H2 → 3CH2 + 3CH2	2.23E+03	0.00E+00	1.03E+05
131. 3CH2 + HCCO → CO + C2H3	3.00E+07	0.00E+00	0.00E+00
132. CO + C2H3 → 3CH2 + HCCO	1.90E+09	0.00E+00	3.95E+05

Table 3: continued

Reaction mechanism	A	β	E
133. CH2O + OH → CHO + H2O	3.43E+03	1.18E+00	-1.87E+03
134. CHO + H2O → CH2O + OH	1.11E+03	1.18E+00	1.21E+05
135. H + CH2O → H2 + CHO	2.19E+02	1.77E+00	1.26E+04
136. H2 + CHO → H + CH2O	1.59E+01	1.77E+00	7.18E+04
137. CH2O + M(1) → H + CHO + M(1)	3.31E+10	0.00E+00	3.39E+05
138. H + CHO + M(1) → CH2O + M(1)	6.20E+02	0.00E+00	-3.79E+04
139. CH2O + O → CHO + OH	1.80E+07	0.00E+00	1.29E+04
140. CHO + OH → CH2O + O	5.79E+05	0.00E+00	6.43E+04
141. CHO + O2 → CO + HO2	7.58E+06	0.00E+00	1.72E+03
142. CO + HO2 → CHO + O2	1.61E+07	0.00E+00	1.36E+05
143. CHO + M(7) → H + CO + M(7)	1.86E+11	-1.00E+00	7.11E+04
144. H + CO + M(7) → CHO + M(7)	2.79E+05	-1.00E+00	9.47E+03
145. CHO + OH → CO + H2O	1.00E+08	0.00E+00	0.00E+00
146. CO + H2O → CHO + OH	2.59E+09	0.00E+00	4.38E+05
147. H + CHO → H2 + CO	1.19E+07	2.50E-01	0.00E+00
148. H2 + CO → H + CHO	6.93E+07	2.50E-01	3.74E+05
149. CHO + O → CO + OH	3.00E+07	0.00E+00	0.00E+00
150. CO + OH → CHO + O	7.73E+07	0.00E+00	3.67E+05
151. CHO + O → H + CO2	3.00E+07	0.00E+00	0.00E+00
152. H + CO2 → CHO + O	1.07E+10	0.00E+00	4.64E+05
153. CO + OH → H + CO2	9.42E-03	2.25E+00	-9.84E+03
154. H + CO2 → CO + OH	1.31E+00	2.25E+00	8.74E+04
155. CO + O + M(1) → CO2 + M(1)	6.17E+02	0.00E+00	1.26E+04
156. CO2 + M(1) → CO + O + M(1)	1.47E+11	0.00E+00	5.38E+05
157. CO + O2 → CO2 + O	2.53E+06	0.00E+00	2.00E+05
158. CO2 + O → CO + O2	2.57E+07	0.00E+00	2.28E+05
159. CO + HO2 → CO2 + OH	5.80E+07	0.00E+00	9.60E+04
160. CO2 + OH → CO + HO2	7.12E+08	0.00E+00	3.57E+05
161. CH3 + C2H6 → CH4 + C2H5	5.50E-07	4.00E+00	3.47E+04
162. CH4 + C2H5 → CH3 + C2H6	3.27E-07	4.00E+00	5.93E+04
163. H + C2H6 → H2 + C2H5	5.40E-04	3.50E+00	2.18E+04
164. H2 + C2H5 → H + C2H6	1.12E-05	3.50E+00	4.26E+04
165. O + C2H6 → OH + C2H5	3.00E+01	2.00E+00	2.14E+04
166. OH + C2H5 → O + C2H6	2.76E-01	2.00E+00	3.43E+04
167. OH + C2H6 → H2O + C2H5	7.23E+00	2.00E+00	3.61E+03
168. H2O + C2H5 → OH + C2H6	6.69E-01	2.00E+00	8.77E+04
169. H + C2H5 → H2 + C2H4	1.25E+08	0.00E+00	3.35E+04
170. H2 + C2H4 → H + C2H5	7.65E+08	0.00E+00	3.16E+05
171. H + C2H5 → CH3 + CH3	3.00E+07	0.00E+00	0.00E+00
172. CH3 + CH3 → H + C2H5	3.01E+06	0.00E+00	4.63E+04
173. H + C2H5 → C2H6	7.00E+07	0.00E+00	0.00E+00
174. C2H6 → H + C2H5	1.31E+16	0.00E+00	4.15E+05
175. OH + C2H5 → H2O + C2H4	4.00E+07	0.00E+00	0.00E+00
176. H2O + C2H4 → OH + C2H5	1.09E+09	0.00E+00	3.46E+05
177. O + C2H5 → CH3 + CH2O	1.00E+08	0.00E+00	0.00E+00
178. CH3 + CH2O → O + C2H5	1.39E+08	0.00E+00	3.39E+05
179. HO2 + C2H5 → CH3 + CH2O + OH	3.00E+07	0.00E+00	0.00E+00
180. CH3 + CH2O + OH → HO2 + C2H5	2.16E+00	0.00E+00	7.46E+04
181. O2 + C2H5 → HO2 + C2H4	3.00E+14	-2.86E+00	2.83E+04
182. HO2 + C2H4 → O2 + C2H5	6.71E+14	-2.86E+00	7.05E+04
183. O2 + C2H5 → HO2 + C2H4	2.12E-12	6.00E+00	3.97E+04
184. HO2 + C2H4 → O2 + C2H5	4.74E-12	6.00E+00	8.19E+04
185. H + C2H4 → H2 + C2H3	3.36E-13	6.00E+00	7.08E+03

Table 3: continued

Reaction mechanism	A	β	E
186. H2 + C2H3 → H + C2H4	1.87E-14	6.00E+00	-5.86E+03
187. OH + C2H4 → H2O + C2H3	2.02E+07	0.00E+00	2.48E+04
188. H2O + C2H3 → OH + C2H4	5.01E+06	0.00E+00	7.52E+04
189. O + C2H4 → CH3 + CHO	1.02E+01	1.88E+00	7.48E+02
190. CH3 + CHO → O + C2H4	1.69E-01	1.88E+00	1.16E+05
191. O + C2H4 → H + CH2CHO	3.39E+00	1.88E+00	7.48E+02
192. H + CH2CHO → O + C2H4	8.51E-01	1.88E+00	5.49E+04
193. CH3 + C2H4 → CH4 + C2H3	6.62E-06	3.70E+00	3.97E+04
194. CH4 + C2H3 → CH3 + C2H4	1.05E-05	3.70E+00	3.06E+04
195. H + C2H4 + M(5) → C2H5 + M(5)	4.00E+07	0.00E+00	0.00E+00(*)
196. C2H5 + M(5) → H + C2H4 + M(5)	2.23E+03	0.00E+00	1.03E+05(*)
197. C2H4 + M(1) → H2 + C2H2 + M(1)	3.00E+07	0.00E+00	0.00E+00(*)
198. H2 + C2H2 + M(1) → C2H4 + M(1)	1.90E+09	0.00E+00	3.95E+05(*)
199. H + C2H3 + M(6) → C2H4 + M(6)	3.43E+03	1.18E+00	-1.87E+03(*)
200. C2H4 + M(6) → H + C2H3 + M(6)	1.11E+03	1.18E+00	1.21E+05(*)
201. H + C2H3 → H2 + C2H2	4.00E+07	0.00E+00	0.00E+00
202. H2 + C2H2 → H + C2H3	8.44E+07	0.00E+00	2.74E+05
203. O + C2H3 → H + CH2CO	3.00E+07	0.00E+00	0.00E+00
204. H + CH2CO → O + C2H3	4.36E+08	0.00E+00	3.62E+05
205. O2 + C2H3 → CHO + CH2O	1.70E+23	-5.31E+00	2.72E+04
206. CHO + CH2O → O2 + C2H3	1.16E+23	-5.31E+00	3.88E+05
207. O2 + C2H3 → O + CH2CHO	3.50E+08	-6.11E-01	2.20E+04
208. O + CH2CHO → O2 + C2H3	2.60E+08	-6.11E-01	2.87E+04
209. O2 + C2H3 → HO2 + C2H2	2.12E-12	6.00E+00	3.97E+04
210. HO2 + C2H2 → O2 + C2H3	1.63E-12	6.00E+00	7.33E+04
211. OH + C2H3 → H2O + C2H2	2.00E+07	0.00E+00	0.00E+00
212. H2O + C2H2 → OH + C2H3	1.88E+08	0.00E+00	3.37E+05
213. CH3 + C2H3 → CH4 + C2H2	2.00E+07	0.00E+00	0.00E+00
214. CH4 + C2H2 → CH3 + C2H3	1.21E+09	0.00E+00	2.78E+05
215. C2H3 + C2H3 → C2H2 + C2H4	1.45E+07	0.00E+00	0.00E+00
216. C2H2 + C2H4 → C2H3 + C2H3	5.48E+08	0.00E+00	2.87E+05
217. OH + C2H2 → H + CH2CO	2.18E-10	4.50E+00	-4.18E+03
218. H + CH2CO → OH + C2H2	3.40E-09	4.50E+00	9.23E+04
219. OH + C2H2 → H + CH2CO	2.00E+05	0.00E+00	0.00E+00
220. H + CH2CO → OH + C2H2	3.12E+06	0.00E+00	9.65E+04
221. OH + C2H2 → CH3 + CO	4.83E-10	4.00E+00	-8.37E+03
222. CH3 + CO → OH + C2H2	8.97E-10	4.00E+00	2.29E+05
223. O + C2H2 → CO + 3CH2	6.12E+00	2.00E+00	7.95E+03
224. CO + 3CH2 → O + C2H2	9.16E-01	2.00E+00	2.16E+05
225. O + C2H2 → H + HCCO	1.43E+01	2.00E+00	7.95E+03
226. H + HCCO → O + C2H2	3.46E+00	2.00E+00	8.61E+04
227. O2 + C2H2 → OH + HCCO	4.00E+01	1.50E+00	1.26E+05
228. OH + HCCO → O2 + C2H2	7.06E-01	1.50E+00	1.36E+05
229. H + C2H2 + M(5) → C2H3 + M(5)	2.19E+02	1.77E+00	1.26E+04(*)
230. C2H3 + M(5) → H + C2H2 + M(5)	1.59E+01	1.77E+00	7.18E+04(*)
231. H + CH2CHO → H2 + CH2CO	4.00E+07	0.00E+00	0.00E+00
232. H2 + CH2CO → H + CH2CHO	1.29E+08	0.00E+00	2.95E+05
233. O + CH2CHO → CHO + CH2O	1.00E+08	0.00E+00	0.00E+00
234. CHO + CH2O → O + CH2CHO	9.18E+07	0.00E+00	3.54E+05
235. OH + CH2CHO → H2O + CH2CO	3.00E+07	0.00E+00	0.00E+00
236. H2O + CH2CO → OH + CH2CHO	4.31E+08	0.00E+00	3.59E+05

Table 3: continued

Reaction mechanism	A	β	E
237. O2 + CH2CHO → CH2O + CO + OH	3.00E+04	0.00E+00	0.00E+00
238. CH2O + CO + OH → O2 + CH2CHO	3.02E-03	0.00E+00	2.24E+05
239. CH3 + CH2CHO → H + CO + C2H5	4.90E+08	-5.00E-01	0.00E+00
240. CH2CHO → H + CH2CO	3.95E+38	-7.65E+00	1.89E+05
241. H + CH2CO → CH2CHO	3.29E+32	-7.65E+00	4.80E+04
242. O + CH2CO → CO2 + 3CH2	1.75E+06	0.00E+00	5.65E+03
243. CO2 + 3CH2 → O + CH2CO	2.33E+06	0.00E+00	2.14E+05
244. H + CH2CO → CH3 + CO	7.00E+06	0.00E+00	1.26E+04
245. CH3 + CO → H + CH2CO	8.34E+05	0.00E+00	1.53E+05
246. H + CH2CO → H2 + HCCO	2.00E+08	0.00E+00	3.35E+04
247. H2 + HCCO → H + CH2CO	7.01E+06	0.00E+00	2.30E+04
248. O + CH2CO → OH + HCCO	1.00E+07	0.00E+00	3.35E+04
249. OH + HCCO → O + CH2CO	1.55E+05	0.00E+00	1.51E+04
250. OH + CH2CO → H2O + HCCO	1.00E+07	0.00E+00	8.37E+03
251. H2O + HCCO → OH + CH2CO	1.56E+06	0.00E+00	6.12E+04
252. CO + 3CH2 + M(6) → CH2CO + M(6)	3.31E+10	0.00E+00	3.39E+05$^{(*)}$
253. CH2CO + M(6) → CO + 3CH2 + M(6)	6.20E+02	0.00E+00	-3.79E+04$^{(*)}$
254. O + HCCO → H + CO + CO	8.00E+07	0.00E+00	0.00E+00
255. H + CO + CO → O + HCCO	4.13E+02	0.00E+00	4.41E+05
256. O2 + HCCO → CHO + CO + O	2.50E+02	1.00E+00	0.00E+00
257. CHO + CO + O → O2 + HCCO	3.66E-05	1.00E+00	6.37E+03
258. O2 + HCCO → CHO + CO2	2.40E+05	0.00E+00	-3.57E+03
259. CHO + CO2 → O2 + HCCO	8.36E+06	0.00E+00	5.28E+05
260. HCCO + HCCO → CO + CO + C2H2	1.00E+07	0.00E+00	0.00E+00
261. CO + CO + C2H2 → HCCO + HCCO	2.13E+02	0.00E+00	3.63E+05

A-2 Surface Reaction Mechanisms

Table 5: Surface-reaction mechanism of the catalytic combustion of methane over platinum $M(i)$ is third body. A, β, E are Arrhenius parameters for the rate constants written in the form: $k = AT^{\beta}\exp(-E/RT)$ or in a modified Arrhenius expression $k_{f_k} = A_k T^{\beta_k}\exp\left(-\frac{E_{ak}}{RT}\right)\prod_{i=1}^{N_s}\Theta_i^{\mu_{ik}}\exp\left(\frac{\epsilon_{ik}\Theta_i}{RT}\right)$. The units of A are given in terms of moles, cubic meters, and seconds. E is in J/mol.
$^{(*)}$ stick, non-Arrhenius reaction, modified Arrhenius rate expression is used.

Reaction mechanism	A	β	E
1. H2 + 2PT(s) → 2H(s)	1.59E+09	5.00E-01	0.00E+00 $^{(*)}$
2. O2 + 2PT(s) → 2O(s)	1.80E+11	-5.00E-01	0.00E+00
3. O2 + 2PT(s) → 2O(s)	1.99E+08	5.00E-01	0.00E+00
4. CH4 + 2PT(s) → CH3(s) + H(s)	1.22E+08	5.00E-01	0.00E+00 $^{(*)}$
5. H2O + PT(s) → H2O(s)	2.36E+05	5.00E-01	0.00E+00
6. CO + PT(s) → CO(s)	2.12E+05	5.00E-01	0.00E+00 $^{(*)}$
7. 2H(s) → H2 + 2PT(s)	3.70E+17	0.00E+00	6.74E+04 $^{(*)}$
8. 2O(s) → O2 + 2PT(s)	3.70E+17	0.00E+00	2.13E+05 $^{(*)}$
9. H2O(s) → H2O + PT(s)	1.00E+13	0.00E+00	4.03E+04
10. CO(s) → CO + PT(s)	1.00E+13	0.00E+00	1.25E+05

Table 5: continued

Reaction mechanism	A	β	E
11. CO2(s) → CO2 + PT(s)	1.00E+13	0.00E+00	2.05E+04
12. O(s) + H(s) → OH(s) + PT(s)	3.70E+17	0.00E+00	1.15E+04
13. OH(s) + PT(s) → O(s) + H(s)	1.02E+18	0.00E+00	7.96E+04
14. OH(s) + H(s) → H2O(s) + PT(s)	3.70E+17	0.00E+00	1.74E+04
15. H2O(s) + PT(s) → OH(s) + H(s)	3.66E+17	0.00E+00	7.36E+04
16. 2OH(s) → O(s) + H2O(s)	3.70E+17	0.00E+00	4.82E+04
17. O(s) + H2O(s) → 2OH(s)	1.32E+17	0.00E+00	3.62E+04
18. CO(s) + O(s) → CO2(s) + PT(s)	3.70E+17	0.00E+00	1.05E+05
19. C(s) + O(s) → CO(s) + PT(s)	3.70E+17	0.00E+00	6.28E+04
20. CO(s) + PT(s) → C(s) + O(s)	1.00E+14	0.00E+00	1.84E+05
21. CH3(s) + PT(s) → CH2(s) + H(s)	3.70E+17	0.00E+00	2.00E+04
22. CH2(s) + PT(s) → CH(s) + H(s)	3.70E+17	0.00E+00	2.00E+04
23. CH(s) + PT(s) → C(s) + H(s)	3.70E+17	0.00E+00	2.00E+04

Table 6: Surface-reaction mechanism of conversion of ethane to ethylene $M(i)$ is third body. A, β, E are Arrhenius parameters for the rate constants written in the form: $k = AT^{\beta}\exp(-E/RT)$ or in a modified Arrhenius expression $k_{f_k} = A_k T^{\beta_k}\exp\left(-\frac{E_{ak}}{RT}\right)\prod_{i=1}^{N_s}\Theta_i^{\mu_{ik}}\exp\left(\frac{\epsilon_{ik}\Theta_i}{RT}\right)$. The units of A are given in terms of moles, cubic meters, and seconds. E is in J/mol.
$^{(*)}$ stick, non-Arrhenius reaction, modified Arrhenius rate expression is used.

Reaction mechanism	A	β	E
1. H + PT(s) → H(s)	1.33E+06	5.00E-01	0.00E+00 $^{(*)}$
2. H2 + 2PT(s) → 2H(s)	1.59E+09	5.00E-01	0.00E+00 $^{(*)}$
3. H2 + C(s) → CH2(s)	3.77E+04	5.00E-01	2.97E+04 $^{(*)}$
4. O + PT(s) → O(s)	3.34E+05	5.00E-01	0.00E+00 $^{(*)}$
5. O2 + 2PT(s) → 2O(s)	1.89E+11	-5.00E-01	0.00E+00
6. OH + PT(s) → OH(s)	3.24E+05	5.00E-01	0.00E+00 $^{(*)}$
7. H2O + PT(s) → H2O(s)	2.36E+05	5.00E-01	0.00E+00 $^{(*)}$
8. CO + PT(s) → CO(s)	2.12E+05	5.00E-01	0.00E+00 $^{(*)}$
9. CO2 + PT(s) → CO2(s)	1.01E+03	5.00E-01	0.00E+00 $^{(*)}$
10. CH3 + PT(s) → CH3(s)	3.45E+05	5.00E-01	0.00E+00 $^{(*)}$
11. CH4 + C(s) → C2H4(2s)	2.34E-03	5.00E-01	2.30E+04 $^{(*)}$
12. CH4 + 2PT(s) → CH3(s) + H(s)	1.10E+07	5.00E-01	7.22E+04 $^{(*)}$
13. CH4 + O(s) + PT(s) → CH3(s) + OH(s)	5.00E+08	7.00E-01	4.20E+04
14. CH4 + OH(s) + PT(s) → CH3(s) + H2O(s)	1.23E+10	5.00E-01	1.00E+04 $^{(*)}$
15. C2H2 + PT(s) → C2H2(1s)	1.31E+04	5.00E-01	0.00E+00 $^{(*)}$
16. C2H4 + PT(s) → C2H4(1s)	3.79E+03	5.00E-01	0.00E+00 $^{(*)}$
17. C2H5 + PT(s) → C2H5(s)	2.48E+05	5.00E-01	0.00E+00 $^{(*)}$
18. C2H6 + 2PT(s) → C2H6(2s)	1.34E+08	5.00E-01	0.00E+00 $^{(*)}$
19. H(s) → H + PT(s)	6.00E+13	0.00E+00	2.54E+05
20. 2H(s) → H2 + 2PT(s)	3.70E+17	0.00E+00	6.74E+04
21. CH2(s) → H2 + C(s)	7.69E+13	0.00E+00	2.51E+04
22. O(s) → O + PT(s)	1.00E+13	0.00E+00	3.59E+05
23. 2O(s) → O2 + 2PT(s)	3.70E+17	0.00E+00	2.27E+05

Table 6: continued

Reaction mechanism	A	β	E
24. OH(s) → OH + PT(s)	5.00E+13	0.00E+00	2.51E+05
25. H2O(s) → H2O + PT(s)	4.50E+12	0.00E+00	4.18E+04
26. CO(s) → CO + PT(s)	2.50E+16	0.00E+00	1.46E+05
27. CO2(s) → CO2 + PT(s)	1.00E+13	0.00E+00	2.71E+04
28. CH3(s) → CH3 + PT(s)	1.00E+13	0.00E+00	1.63E+05
29. CH3(s) + H(s) → CH4 + 2PT(s)	1.50E+16	0.00E+00	5.00E+04
30. CH3(s) + H2O(s) → CH4 + OH(s) + PT(s)	2.50E+16	0.00E+00	2.30E+04
31. CH3(s) + OH(s) → CH4 + O(s) + PT(s)	3.70E+17	0.00E+00	8.59E+04
32. C2H2(1s) → C2H2 + PT(s)	1.00E+12	0.00E+00	5.86E+04
33. C2H4(1s) → C2H4 + PT(s)	1.00E+13	0.00E+00	5.02E+04
34. C2H4(2s) → CH4 + C(s)	1.00E+10	0.00E+00	2.55E+04
35. C2H5(s) → C2H5 + PT(s)	1.00E+13	0.00E+00	1.73E+05
36. C2H6(2s) → C2H6 + 2PT(s)	1.00E+13	0.00E+00	2.09E+04
37. O(s) + H(s) → OH(s) + PT(s)	1.28E+17	0.00E+00	1.12E+04
38. OH(s) + PT(s) → O(s) + H(s)	7.39E+15	0.00E+00	7.73E+04
39. OH(s) + H(s) → H2O(s) + PT(s)	2.04E+17	0.00E+00	6.62E+04
40. H2O(s) + PT(s) → OH(s) + H(s)	1.15E+15	0.00E+00	1.01E+05
41. 2OH(s) → H2O(s) + O(s)	7.40E+16	0.00E+00	7.40E+04
42. H2O(s) + O(s) → 2OH(s)	1.00E+16	0.00E+00	4.31E+04
43. C(s) + O(s) → CO(s) + PT(s)	3.70E+15	0.00E+00	0.00E+00
44. CO(s) + PT(s) → C(s) + O(s)	3.70E+15	0.00E+00	2.36E+05
45. CO(s) + O(s) → CO2(s) + PT(s)	3.70E+15	0.00E+00	1.18E+05
46. CO2(s) + PT(s) → CO(s) + O(s)	3.70E+15	0.00E+00	1.73E+05
47. CO(s) + OH(s) → CO2(s) + H(s)	2.00E+15	0.00E+00	3.87E+04
48. CO2(s) + H(s) → CO(s) + OH(s)	2.00E+15	0.00E+00	2.83E+04
49. CH3(s) + PT(s) → CH2(s) + H(s)	1.26E+18	0.00E+00	7.03E+04
50. CH2(s) + H(s) → CH3(s) + PT(s)	3.09E+18	0.00E+00	0.00E+00
51. CH2(s) + PT(s) → CH(s) + H(s)	7.31E+18	0.00E+00	5.89E+04
52. CH(s) + H(s) → CH2(s) + PT(s)	3.09E+18	0.00E+00	0.00E+00
53. CH(s) + PT(s) → C(s) + H(s)	3.09E+18	0.00E+00	0.00E+00
54. C(s) + H(s) → CH(s) + PT(s)	1.25E+18	0.00E+00	1.38E+05
55. C2H6(2s)+ O(s) → C2H5(s) + OH(s) + PT(s)	3.70E+17	0.00E+00	2.51E+04
56. C2H5(s) + OH(s) + PT(s) → C2H6(2s)+ O(s)	1.35E+22	0.00E+00	7.74E+04
57. C2H4(1s) → C2H4(2s)	1.00E+13	0.00E+00	8.33E+04
58. C2H4(2s) → C2H4(1s)	1.00E+13	0.00E+00	7.53E+04
59. C2H5(s) + H(s) → C2H6(2s)	3.70E+17	0.00E+00	4.18E+04
60. C2H6(2s) → C2H5(s) + H(s)	7.00E+12	0.00E+00	5.77E+04
61. 2CH3(s) → C2H6(2s)	1.00E+17	0.00E+00	1.45E+04
62. C2H6(2s) → 2CH3(s)	1.00E+13	0.00E+00	8.90E+04
63. C2H5(s) + PT(s) → C2H4(2s)+ H(s)	1.00E+18	0.00E+00	5.44E+04
64. C2H4(2s)+ H(s) → C2H5(s) + PT(s)	1.00E+17	0.00E+00	2.93E+04
65. C2H4(2s)+ PT(s) → C2H3(1s)+ H(s)	2.00E+18	0.00E+00	9.91E+04
66. C2H3(1s)+ H(s) → C2H4(2s)+ PT(s)	3.70E+17	0.00E+00	7.53E+04
67. C2H4(2s)+ PT(s) → C2H3(2s)+ H(s)	3.70E+17	0.00E+00	1.28E+05
68. C2H3(2s)+ H(s) → C2H4(2s)+ PT(s)	3.70E+17	0.00E+00	5.73E+04
69. C2H4(1s)+ PT(s) → C2H3(2s)+ H(s)	3.70E+17	0.00E+00	1.13E+05
70. C2H3(2s)+ H(s) → C2H4(1s)+ PT(s)	3.70E+17	0.00E+00	3.35E+04
71. C2H3(2s)+ PT(s) → C2H2(3s)+ H(s)	3.70E+17	0.00E+00	1.21E+05
72. C2H2(3s)+ H(s) → C2H3(2s)+ PT(s)	3.70E+17	0.00E+00	5.17E+04
73. C2H3(1s)+ PT(s) → CH3(s) + C(s)	3.70E+17	0.00E+00	4.69E+04

Table 6: continued

Reaction mechanism	A	β	E
74. CH3(s) + C(s) → C2H3(1s)+ PT(s)	3.70E+17	0.00E+00	4.60E+04
75. C2H2(1s) → C2H2(3s)	1.00E+13	0.00E+00	6.15E+04
76. C2H2(3s) → C2H2(1s)	1.00E+13	0.00E+00	4.20E+03
77. C2H3(1s) → C2H3(2s)	1.00E+13	0.00E+00	1.76E+05
78. C2H3(2s) → C2H3(1s)	1.00E+13	0.00E+00	1.29E+05
79. C2H2(1s)+ PT(s) → C2H(1s) + H(s)	3.70E+17	0.00E+00	1.34E+05
80. C2H(1s) + H(s) → C2H2(1s)+ PT(s)	3.70E+17	0.00E+00	6.69E+04
81. C2H(1s) + PT(s) → CH(s) + C(s)	3.70E+17	0.00E+00	1.25E+05
82. CH(s) + C(s) → C2H(1s) + PT(s)	3.70E+17	0.00E+00	1.21E+05

Table 4: Surface-reaction mechanism of the NO-NO2. A, β, E are Arrhenius parameters for the rate constants written in the form: $k = AT^{\beta}\exp(-E/RT)$ or in a modified Arrhenius expression $k_{f_k} = A_k T^{\beta_k}\exp\left(-\frac{E_{ak}}{RT}\right)\prod_{i=1}^{N_s}\Theta_i^{\mu_{ik}}\exp\left(\frac{\epsilon_{ik}\Theta_i}{RT}\right)$. The units of A are given in terms of moles, cubic meters, and seconds. E is in J/mol.
(*) is non-Arrhenius reaction, modified Arrhenius reaction

Reaction mechanism	A	β	E
1. O2 + 2PT(s) → 2O(s)	6.08E+08	5.00E-01	0.00E+00
2. NO + PT(s) → NO(s)	2.07E+05	5.00E-01	0.00E+00
3. NO2 + PT(s) → NO2(s)	1.77E+05	5.00E-01	0.00E+00
4. O + PT(s) → O(s)	3.34E+05	5.00E-01	0.00E+00
5. 2O(s) → O2 + 2PT(s)	3.70E+17	0.00E+00	2.13E+05 (*)
6. NO(s) → NO + PT(s)	1.00E+16	0.00E+00	9.00E+04
7. NO2(s) → NO2 + PT(s)	1.00E+13	0.00E+00	6.00E+04
8. NO(s) + O(s) → NO2(s) + PT(s)	3.70E+17	0.00E+00	9.63E+04 (*)
9. NO2(s) + PT(s) → NO(s) + O(s)	3.70E+17	0.00E+00	7.95E+04