DELFT UNIVERSITY OF TECHNOLOGY

REPORT 05-05

Efficient Methods for Solving Advection Diffusion Reaction Equations

S. van Veldhuizen

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2005

Efficient Methods for Solving Advection Diffusion Reaction Equations -Literature Study-

S. van Veldhuizen

August 23, 2005

Contents

Contents **2**

Preface **6**

I Mathematical Model of CVD **9**

1 Introduction **11**

2 The Mathematical Model **13**
2.1 The Transport Model 13
2.1.1 Transport Equations for Gas Species 15
2.1.2 Complete Mathematical Model 20
2.2 Boundary Conditions 20

3 Stiffness **25**
3.1 Example of Stiff Equation 25
3.2 Stability Analysis for Euler Forward Method 28

4 Test Problem **31**
4.1 The Computational Grid 32
4.2 Finite Volume Discretization of the General Transport Equation 32
4.3 Boundary Conditions 36
4.4 Chemistry Properties 38
4.4.1 Gas-Phase Reaction Model 38

II Time Integration Methods **43**

1 Introduction **45**

2 Runge-Kutta Methods **47**
2.1 The Stability Function 49
2.2 Rosenbrock Methods 50
2.3 Runge Kutta Chebyshev Methods 53
2.3.1 First Order (Damped) Stability Polynomials and Schemes 55
2.3.2 Second Order Schemes 57
2.4 Some Remarks on Runge-Kutta Methods 59
2.4.1 Properties of Implicit Runge-Kutta methods 60
2.4.2 Diagonally Implicit Runge-Kutta methods 62
2.4.3 The Order Reduction Phenomenon for RK-methods . 64
2.4.4 Order Reduction for Rosenbrock Methods 67

3 Time Splitting **71**
3.1 Operator Splitting 71
3.1.1 First Order Splitting of Linear ODE Problems 72
3.1.2 Nonlinear ODE problems 73
3.1.3 Second and Higher Order Splitting. 73
3.2 Boundary Values, Stiff Terms and Steady State 75
3.2.1 Boundary Values 75
3.2.2 Stiff Terms 75
3.2.3 Steady State 76

4 IMEX Methods **77**
4.1 IMEX- θ Method 77
4.2 Concluding Remarks 80

5 IMEX Runge-Kutta-Chebyshev Methods **83**
5.1 Construction of the IMEX scheme 84
5.2 Stability 85
5.3 Consistency 86
5.4 Final Remarks 86

6 Linear Multi-Step Methods **89**
6.1 The Order Conditions 90
6.2 Stability Properties 92
6.2.1 Zero-Stability 93
6.2.2 The Stability Region 93

7 Multi Rate Runge Kutta Methods **97**
7.1 Multi-Rate Runge-Kutta Methods 98
7.2 Order Conditions 100
7.3 Stability 101

III Nonlinear Solvers 105

1 Introduction 107

2 Newton's Method 109
2.1 Newton's Method in One Variable . . . 109
2.2 General Remarks on Newton's Method . . . 109
2.3 Convergence Properties . . . 112
2.4 Criteria for Termination of the Iteration . . . 113
2.5 Inexact Newton Methods . . . 114
2.6 Global Convergence . . . 115
2.7 Extension of Secant Method to n Dimensions . . . 117
2.8 Failures . . . 119
2.8.1 Non-smooth Functions . . . 119
2.8.2 Slow Convergence . . . 119
2.8.3 No Convergence . . . 120
2.8.4 Failure of the Line Search . . . 120

3 Picard Iteration 121
3.1 One Dimension . . . 121
3.2 Higher Dimensions . . . 123
3.3 Some Last Remarks . . . 124

IV Linear Solvers 125

1 Introduction 127

2 Krylov Subspace Methods 131
2.1 Krylov Subspace Methods for Symmetric Matrices . . . 132
2.1.1 Arnoldi's Method and the Lanczos Algorithm . . . 132
2.1.2 The Conjugate Gradient Algorithm . . . 134
2.2 Krylov Subspace Methods for General Matrices . . . 136
2.2.1 BiCG Type Methods . . . 137
2.2.2 GMRES Methods . . . 141
2.2.3 The GMRES Algorithm . . . 142
2.3 Stopcriterium . . . 143

3 Precondition Techniques 145
3.1 Preconditioned Iterations . . . 145
3.1.1 Preconditioned Conjugate Gradient . . . 145
3.1.2 Preconditioned GMRES . . . 147
3.1.3 Preconditioned Bi-CGSTAB . . . 149
3.2 Preconditioned Techniques . . . 150
3.2.1 Diagonal Scaling . . . 151

3.2.2 Incomplete LU Factorization 151
3.3 Incomplete Choleski Factorization 153
3.4 Multigrid 153

V Concluding Remarks **155**

1 Summary and Conclusions **157**
1.1 Time Integration Methods 159
1.2 Nonlinear and Linear Solvers 159

2 Future Research **161**
2.1 Test-problem and Time Integration Methods 161
2.2 Stability 161
2.3 (Non)-Linear Solvers 162
2.4 Extension to Three Dimensions 162

VI Appendices **163**

A Collocation Methods **165**

B Padé Approximations **167**

Bibliography **169**

Preface

Aim of the Research Project

Chemical Vapor Deposition creates thin films of material on a substrate via the use of chemical reactions. Reactive gases are fed into a reaction chamber and these gases react in the gas phase or on a substrate and form a thin film or a powder. In order to model this phenomenon the flow of gases is described by the incompressible Navier-Stokes equations, whereby the density variations are taken into account by the perfect gas law. The temperature is governed by the heat equation, whereas the chemical species satisfy the advection-diffusion-reaction equations. Since the reaction of chemical species generates or costs energy, there is a coupling between the heat equation and the advection-diffusion-reaction equations. In classical CVD the generated heat is rather small, so it is possible to decouple the heat equation. Our aim is to develop numerical methods , which are also applicable to laminar combustion simulations, where there is a strong coupling between the heat equation and the advection-diffusion-reaction equations.

Typically, in the advection-diffusion-reaction equations the time scales of the slow and fast reaction terms differ order of magnitudes from each other and from the time scales of diffusion and advection, leading to extremely stiff systems. The purpose of this project is to develop robust and efficient solvers for the heat/reaction system, where we firstly assume that the velocity is given. Thereafter this solver should be integrated in a CFD package. For a typical 3D problem a $50 \times 50 \times 50$ grid with 50 chemical species are used.

Structure of the Report

This report, a summary of recent, relevant literature, is divided into five parts. The first part consists of a general introduction to the C(hemical) V(apor) D(eposition) process and the corresponding mathematical model which describes that process. It appears that the system of PDEs describing

CVD is a so-called stiff system. In this first part the notion of stiffness is explained. We conclude with the finite volume discretization of the general transport equation.

The second part of this report presents time integration methods that can 'handle' stiff problems. Some 'old' methods, which work fine for stiff problems. as well as recently developed integration methods will be treated.

Since the PDEs describing CVD are of the nonlinear type, time integration methods will in general result in solving nonlinear equations. Therefore, the topic of the third part is 'nonlinear solvers'. We give a treatment of all well known nonlinear solvers.

Newton-type nonlinear solvers need solutions of linear systems. The latter is (evidently) the subject of Part IV. Since our aim is to solve three dimensional problems, this will result in huge nonlinear systems. Thus also huge linear systems have to be solved. The most attractive class of linear solvers for three dimensional problems are iterative methods. In Part IV we treat generally known iterative linear solvers for symmetric and general matrices.

In Part V some conclusions are made and the scheme for future research is presented.

Part I

Mathematical Model of CVD

Chapter 1

Introduction

Thin solid films are used as insulating and (semi)conducting layers in many technological areas such as micro-electronics, optical devices and so on. Therefore, deposition processes are required which produce solid films on a wide variety of materials. Of course, these processes need to fulfill specific requirements with regard to safety, economics, etc.

Chemical Vapor Deposition (CVD) is a process that synthesizes a thin solid film from the gaseous phase by a chemical reaction on a solid material. The chemical reactions involved distinguish CVD from other physical deposition processes, such as sputtering and evaporation.

A CVD system is a chemical reactor, wherein the material to be deposited is injected as a gas and which contains the substrates on which deposition takes place. The energy to drive the chemical reaction is (usually) thermal energy. On the substrates surface reactions will take place resulting in deposition of a thin film. Basically, the following nine steps, taken from [16], occur in every CVD reaction process:

1. Convective and diffusive transport of reactants from the reactor inlet to the reaction zone within the reactor chamber,

2. Chemical reactions in the gas phase, leading to a multitude of new reactive species and byproducts,

3. Convective and diffusive transport of the initial reactants and the reaction products from the homogeneous reactions to the susceptor surface,

4. Adsorption or chemisorption of these species on the susceptor surface,

5. Surface diffusion of adsorbed species over the surface,

6. Heterogeneous surface reactions catalyzed by the surface, leading to the formation of a solid film,

7. Desorption of gaseous reaction products,
8. Diffusive transport of reaction products away from the surface,
9. Convective and/or diffusive transport of reaction products away from the reaction zone to the outlet of the reactor.

In the case that a CVD process is considered to be fully heterogeneous, step (2) in the above enumeration is does not take place. The above enumeration is illustrated in Figure 1.1, taken from [14].

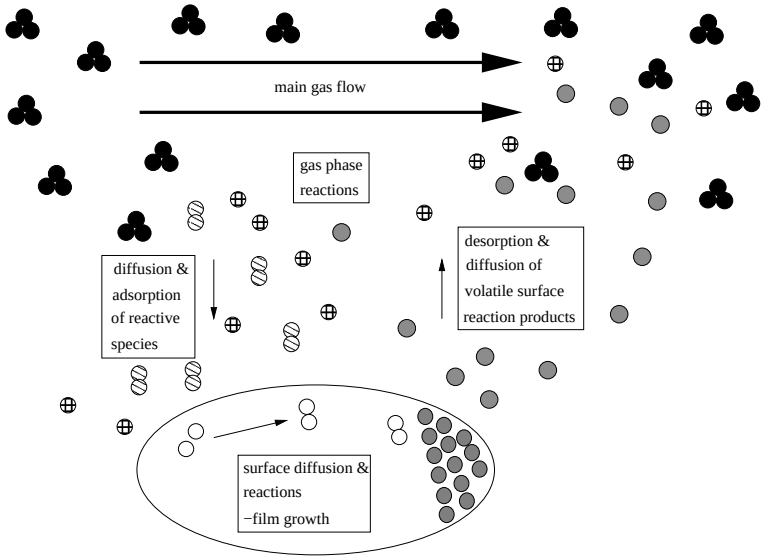

Figure 1.1: Schematic representation of basic steps in CVD (after Jensen, see [14])

Chapter 2

The Mathematical Model

The mathematical model describing the CVD process consists of a set of partial differential equations, with appropriate boundary conditions, describing the gas flow, transport of energy, transport of species and the chemical reactions in the reactor.

The gas mixture is assumed to behave as a continuum. This assumption is valid when the mean free path length of the molecules is much smaller than a characteristic dimension of the reactor. The *Knudsen Number* Kn is defined as

$$\mathrm{Kn} = \frac{\xi}{L}, \tag{2.1}$$

where ξ is the mean free path length of the molecules and L a typical characteristic dimension of the reactor. Thus, the gas-mixture behaves as a continuum when $Kn < 0.01$. For pressures larger than 100 Pa and typical reactor dimensions larger than 0.01 m the continuum approach can be used safely. See also [16] and [21, Chapter 4].

2.1 The Transport Model

The gas mixture in the reactor is assumed to behave as a continuum, ideal and transparent gas[1] behaving in accordance with Newton's law of viscosity. Furthermore, the gas flow is assumed to be laminar (low Reynolds number flow). Since no large velocity gradients appear in CVD gas flows, viscous heating due to dissipation will be neglected. Furthermore, we neglect the effects of pressure variations in the energy equation.

The composition of the N component gas mixture is described in terms

[1] By transparent we mean that the adsorption of heat radiation by the gas(es) will be small.

of the dimensionless mass fractions $\omega_i = \frac{\rho_i}{\rho}$, $i = 1, \ldots, N$, with the property

$$\sum_{i=1}^{N} \omega_i = 1.$$

The density $\rho = \sum_{i=1}^{N} \omega_i \rho_i$ of the gas mixture depends on the spatial variable $\mathbf{x}$, temperature T, the pressure P, time t, etc. Usually, in chemistry reaction equilibria are expressed in terms of (dimensionless) molar fractions. The molar fraction of species i is denoted by f_i and is related to the mass fraction ω_i as

$$\omega_i = \frac{f_i m_i}{m},$$

where m_i is the molar mass of specie i and

$$m = \sum_{i=1}^{N} f_i m_i.$$

The mass averaged velocity $\mathbf{v}$ in an N component gas mixture is defined as

$$\mathbf{v} = \sum_{i=1}^{N} \omega_i \mathbf{v}_i.$$

As a consequence, the diffusive mass fluxes $\mathbf{j}_i$ (see Section 2.1.1) sum up to zero, e.g.,

$$\sum_{i=1}^{N} \mathbf{j}_i = \sum_{i=1}^{N} \rho \omega_i (\mathbf{v}_i - \mathbf{v}) = \rho \sum_{i=1}^{N} \omega_i \mathbf{v}_i - \rho \mathbf{v} \sum_{i=1}^{N} \omega_i = \rho \mathbf{v} - \rho \mathbf{v} = 0.$$

The transport of mass, momentum and heat are described respectively by the continuity equation, the Navier-Stokes equations and the transport equation for thermal energy expressed in terms of temperature T. The *continuity equation* is given as

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v}), \tag{2.2}$$

where ρ is the gas mixture density ($\frac{kg}{m^3}$) and $\mathbf{v}$ the mass averaged velocity vector ($\frac{m}{s}$). The *Navier-Stokes equations* are

$$\frac{\partial (\rho \mathbf{v})}{\partial t} = -(\nabla \rho \mathbf{v}) \cdot \mathbf{v} + \nabla \cdot \left[\mu \left(\nabla \mathbf{v} + (\nabla \mathbf{v})^{\mathbf{T}} \right) - \frac{\mathbf{2}}{\mathbf{3}} \mu (\nabla \cdot \mathbf{v}) \mathbf{I} \right] - \nabla \mathbf{P} + \rho \mathbf{g}, \tag{2.3}$$

where μ is the viscosity ($\frac{kg}{m \cdot s}$), $\mathbf{I}$ the unit tensor and $\mathbf{g}$ the gravitational acceleration. The *transport equation for thermal energy*[2] is

$$\begin{aligned} c_p \frac{\partial(\rho T)}{\partial t} &= -c_p \nabla \cdot (\rho \mathbf{v} T) + \nabla \cdot (\lambda \nabla T) + \\ &\quad + \nabla \cdot \left(RT \sum_{i=1}^{N} \frac{\mathbb{D}_i^T}{M_i} \frac{\nabla f_i}{f_i} \right) + \sum_{i=1}^{N} \frac{H_i}{m_i} \nabla \cdot \mathbf{j}_i \\ &\quad - \sum_{i=1}^{N} \sum_{k=1}^{K} H_i \nu_{ik} R_k^g, \end{aligned} \tag{2.4}$$

with c_p specific heat ($\frac{J}{\text{mol} \cdot K}$), λ the thermal conductivity ($\frac{W}{m \cdot K}$) and R the gas constant. Gas species i has a mole fraction f_i, a molar mass m_i, a thermal diffusion coefficient $\mathbb{D}_i^T$, a molar enthalpy H_i and a diffusive mass flux $\mathbf{j}_i$. For a definition of diffusive flux we refer to Section 2.1.1. The stoichiometric coefficient of the i^{th} species in the k^{th} gas-phase reaction with net molar reaction rate R_k^g is ν_{ik}.

The third term on the right-hand side of (2.4) is due to the *Dufour effect* (diffusion-thermo effect). The Dufour effect is the reversed process of the *Soret effect*, which is the process of thermal diffusion. The fourth term on the right represents the transport of heat associated with the inter-diffusion of the chemical species. Both terms, the third and the fourth, are probably not too important in CVD. The last term at the right-hand side of (2.4) represents the heat production or consumption due to chemical reactions in the gas mixture. In processes where the gas-phase reactions are neglected this term will not be important. Also in processes where the reactants are highly diluted in an inert carrier gas this term is negligible.

2.1.1 Transport Equations for Gas Species

We formulate the transport equations for the gas species in terms of mass fractions and diffusive mass fluxes. The convective mass flux for the i^{th} gas species is $\rho \omega_i \mathbf{v}$. The diffusive mass flux $\mathbf{j}_i$ of the i^{th} species is defined as

$$\mathbf{j}_i = \rho \omega_i (\mathbf{v}_i - \mathbf{v}),$$

with $\mathbf{v}_i$ the velocity vector of species i and $\mathbf{v}$ the mass averaged velocity. See for instance [2]. Mass diffusion can be decomposed in concentration diffusion $\mathbf{j}_i^C$ and thermal diffusion $\mathbf{j}_i^T$:

$$\mathbf{j}_i = \mathbf{j}_i^C + \mathbf{j}_i^T.$$

[2] Professor Kleijn remarked that the well-posedness of this equation is doubtful. Due to the fact that the zero of the enthalpy can be chosen in a free way, the 'meaning' of this equation comes into play. Choosing a not-suitable zero of the enthalpy causes undesired large or small values of the enthalpy.

See [16]. The first type of diffusion, $\mathbf{j}_i^C$, occurs as a result of a concentration gradient in the system. We will refer to it as ordinary diffusion. Thermal diffusion is the kind of diffusion resulting from a temperature gradient.

Ordinary Diffusion

Depending on the properties of the gas mixture, different approaches are possible to describe ordinary diffusion. First, in a binary gas mixture, e.g., a gas mixture consisting of *two species* ($N = 2$), the ordinary diffusion mass flux is given by Fick's Law :

$$\begin{aligned} \mathbf{j}_1 &= -\rho D_{12} \nabla \omega_1, \\ \mathbf{j}_2 &= -\rho D_{21} \nabla \omega_2, \end{aligned}$$

with D_{ij} the diffusion coefficient for ordinary diffusion in the pair of gases 1 and 2. In for instance [2] is derived that for a binary mixture there is just one ordinary diffusity,

$$D_{12} = D_{21}.$$

For a multi-component gas mixture there are two approaches, the Stefan-Maxwell equations and an alternative approximation derived by Wilke. The Stefan-Maxwell equations are a general expression for the ordinary diffusion fluxes $\mathbf{j}_i^C$ in a multi-component gas mixture. In terms of mass fraction and fluxes they are given as

$$\nabla \omega_i + \omega_i \nabla(\ln m) = \frac{m}{\rho} \sum_{j=1}^{N} \frac{1}{m_j D_{ij}} \left(\omega_i \mathbf{j}_j^C - \omega_j \mathbf{j}_i^C \right), \quad i = 1, \ldots, N-1, \tag{2.5}$$

with m the average mole mass of the mixture

$$m = \sum_{i=1}^{N} f_i m_i.$$

The coefficients D_{ij} are the binary diffusion coefficients for gas species i and j. In an N-component gas mixture there are $N - 1$ Stefan-Maxwell equations (2.5) and the additional equation

$$\sum_{i=1}^{N} \mathbf{j}_i^C = 0.$$

The following explicit expression for $\mathbf{j}_i^C$ is taken from [16],

$$\mathbf{j}_i^C = \rho \mathbb{D}_i - \rho \omega_i \mathbb{D}_i \nabla(\ln m) + m \omega_i \mathbb{D}_i \sum_{j=1, j \neq i}^{N} \frac{\mathbf{j}_i^C}{m_j D_{ij}}, \tag{2.6}$$

with $\mathbb{D}_i$ an effective diffusion coefficient for species i

$$\mathbb{D}_i = \frac{1}{\sum_{j=1,j\neq i}^{N} \frac{f_j}{D_{ij}}}.$$

An other approximate expression for the ordinary diffusion fluxes has been derived by Wilke in the 1950's. In Wilkes' approach the diffusion of species i is

$$\mathbf{j}_i^C = \rho \mathbb{D}_i' \nabla \omega_i, \tag{2.7}$$

with effective diffusion coefficient

$$\mathbb{D}_i' = (1 - f_i) \frac{1}{\sum_{j=1,j\neq i}^{N} \frac{f_j}{D_{ij}}}.$$

In Wilke's approach the diffusion is written in the form of Fick's Law of diffusion with an effective diffusion coefficient instead of a binary diffusion coefficient.

Finally we remark that in the case of a binary gas mixture, the Stefan-Maxwell equations and the Wilke approach lead to Fick's Law of binary diffusion. In the case of a multi-component gas mixture both approaches are identical if the gas species are highly diluted ($f_1, \omega_1 \ll 1$). The following remark is taken from [16]. When Wilke's approach to the Stefan-Maxwell equations is used to compute the diffusive fluxes, the set of transport equations (2.9) form an independent set, which is not consistent with

$$\sum_{i=1}^{N} \omega_i = 1.$$

To fulfill this constraint one of the transport equations has to be replaced by the constraint itself.

Thermal Diffusion

A homogeneous gas mixture will be separated due to the effect of thermal diffusion (Soret effect) under the influence of a temperature gradient. This so-called Soret effect is usually small in comparison with ordinary diffusion. Only in systems, actually we mean CVD reactors, with large temperature gradients thermal diffusion may be an important effect. For instance in cold-wall CVD reactors we have large temperature gradients. Thermal diffusion causes in general a movement of relatively heavy molecules to 'colder' regions of the reactor chamber and a movement of light molecules to hotter parts. The thermal diffusive mass flux is given by

$$\mathbf{j}_i^T = -\mathbb{D}_i^T \nabla (\ln T). \tag{2.8}$$

In (2.8) $\mathbb{D}_i^T$ is the multi-component thermal diffusion coefficient for species i. In general it will be a function of the temperature T and the composition of the gas mixture. As remarked in [16] the coefficient $\mathbb{D}_i^T$ is *not* a function of the pressure. An exact computation of $\mathbb{D}_i^T$ can be found in [16, Appendix F], where can be observed that indeed no pressure is used.

Finally, we remark that adiabatic systems do not have thermal diffusion.

The Species Concentration Equations

We assume that in the gas-phase there are K reversible reactions. For the k^{th} reaction a *net* molar reaction rate R_k^g $\left(\frac{\text{mole}}{m^3 \cdot s}\right)$ is given. The balance equation for the i^{th} gas species, $i = 1, \ldots, N$, in terms of mass fractions and diffusive mass fluxes is then given as

$$\frac{\partial(\rho\omega_i)}{\partial t} = -\nabla \cdot (\rho \mathbf{v} \omega_i) - \nabla \cdot \mathbf{j}_i + m_i \sum_{k=1}^{K} \nu_{ik} R_k^g. \tag{2.9}$$

From the above general PDE for species transport and chemical gas phase reactions, the approximate Wilke and exact Stefan-Maxwell expressions for ordinary diffusion and expression for thermal diffusion we have the following PDEs for solving the species concentrations :

Using the Wilke approximation :

$$\begin{aligned} \frac{\partial(\rho\omega_i)}{\partial t} &= -\nabla \cdot (\rho \mathbf{v} \omega_i) + \nabla \cdot (\rho \mathbb{D}_i' \nabla \omega_i) + \nabla \cdot (\mathbb{D}_i^T \nabla(\ln T)) + \\ &\quad + m_i \sum_{k=1}^{K} \nu_{ik} R_k^g, \end{aligned} \tag{2.10}$$

Using the exact Stefan-Maxwell equations :

$$\begin{aligned} \frac{\partial(\rho\omega_i)}{\partial t} &= -\nabla \cdot (\rho \mathbf{v} \omega_i) + \nabla \cdot (\rho \mathbb{D}_i \nabla \omega_i) + \\ &\quad + \nabla \cdot (\rho \omega_i \mathbb{D}_i \nabla(\ln m)) - \nabla \cdot \left(m \omega_i \mathbb{D}_i \sum_{j=1, j \neq i}^{N} \frac{\mathbf{j}_i^C}{m_j D_{ij}} \right) + \\ &\quad + \nabla \cdot (\mathbb{D}_i^T \nabla(\ln T)) + m_i \sum_{k=1}^{K} \nu_{ik} R_k^g. \end{aligned} \tag{2.11}$$

Net Molar Reaction Rates

The last term on the right hand side of (2.9), (2.10) and (2.11) describes the consumption and production of the i^{th} species due to homogeneous reactions

in the gas-phase. In this section we give expressions for the net molar reaction rates R_k^g. As before, we assume that there are K reversible gas-phase reactions. Since different species can act as reactant and as product, we use the general notation

$$\sum_{i=1}^{N} \| - \nu_{ik}\| \mathcal{A}_i \underset{k^g_{k,\text{backward}}}{\overset{k^g_{k,\text{forward}}}{\rightleftarrows}} \sum_{i=1}^{N} \|\nu_{ik}\| \mathcal{A}_i \quad k = 1, \ldots, K. \tag{2.12}$$

In (2.12) $\mathcal{A}_i$, $i = 1, \ldots, N$, represent the different gas species in the reactor chamber, $k^g_{k,\text{forward}}$ the forward reaction rate constant and $k^g_{k,\text{backward}}$ the backward reaction rate constant. By taking

$$\begin{aligned} &\nu_{ik} > 0 \text{ for the products of the forward reaction} \\ &\nu_{ik} < 0 \text{ for the reactants of the forward reaction} \end{aligned}$$

and

$$\begin{aligned} &\|\nu_{ik}\| = \nu_{ik} \quad \text{and} \quad \| - \nu_{ik}\| = 0 \qquad \text{for} \quad \nu_{ik} \geq 0 \\ &\|\nu_{ik}\| = 0 \quad \text{and} \quad \| - \nu_{ik}\| = |\nu_{ik}| \quad \text{for} \quad \nu_{ik} \leq 0 \end{aligned}$$

equation (2.12) represents a general equilibrium reaction, with reactants appearing at the left-hand side and products on the right-hand side. The net reaction rate R_k^g for the k^{th} reaction is given as

$$\begin{aligned} R_k^g &= R_k^{g,\text{forward}} - R_k^{g,\text{backward}} = \\ &= k^g_{k,\text{forward}} \prod_{i=1}^{N} c_i^{\|-\nu_{ik}\|} - k^g_{k,\text{backward}} \prod_{i=1}^{N} c_i^{\|\nu_{ik}\|}, \end{aligned} \tag{2.13}$$

with $c_i = \frac{P f_i}{RT}$.[3] See for instance [16] and [21, Chapter 4].

The forward reaction rate constants $k^g_{k,\text{forward}}$ are fitted as a function of the temperature T as follows:

$$k^g_{k,\text{forward}}(T) = A_k T^{\beta_k} e^{\frac{-E_k}{RT}}, \tag{2.14}$$

where A_k is the pre-exponential factor homogeneous reaction rate for the k^{th} reaction, β_k the temperature coefficient and E_k the activation energy for the k^{th} reaction. Expressing the rate constant $k^g_{k,\text{forward}}$ as done in (2.14) is the (modified) *Law of Arrhenius*. See also [17]. The backward reaction rate constants are self-consistent with the forward reaction rate constants. Using thermo-chemistry and doing some calculations the following relation appears

$$k^g_{k,\text{backward}}(T) = \frac{k^g_{k,\text{forward}}(T)}{K^g_k} \left(\frac{RT}{P^0}\right)^{\sum_{i=1}^{N} \nu_{ik}},$$

[3] Recall that the mole fraction of the i^{th} gas species f_i is related with the mass fraction ω_i in the following way

$$f_i = \frac{m_i}{m} \omega_i,$$

with m_i the molar mass of species i and m the average molar mass of the gas-mixture.

with K_k^g the reaction equilibrium constant given by

$$\begin{aligned} K_k^g(T) &= e^{-\frac{\Delta H_k^0(T) - T\Delta S_k^0(T)}{RT}}, \text{ with} \\ \Delta H_k^0(T) &= \sum_{i=1}^{N} \nu_{ik} H_i^0(T) \text{ and} \\ \Delta S_k^0(T) &= \sum_{i=1}^{N} \nu_{ik} S_i^0(T). \end{aligned}$$

Remark 2.1. *Typically, the pre-exponential factor homogeneous reaction rate for the k^{th} reaction A_k are of order $10^5 - 10^{29}$. Due to these large factors A_k the equations (2.10) and (2.11) become very stiff.*

2.1.2 Complete Mathematical Model

To complete the set of equations we add the ideal gas law, i.e., we assumed the gas mixture to behave as an ideal gas. In terms of of the mass density ρ and molar mass m of the gas it is given as

$$Pm = \rho RT, \tag{2.15}$$

with P the pressure, R the gas constant and T the temperature.

As final result, in the CVD reactor the following coupled set of an algebraic equation and partial differential equations has to be solved. This set consists of the following $N+3+d$, with d the dimension of space, equations :

1. Continuity equation (2.2),
2. Navier-Stokes equations (d equations, $d = 1, 2, 3$) (2.3),
3. Transport equation for thermal energy (2.4),
4. Transport equations for the i^{th} gas species, $i = 1, \ldots, N$ (2.9) and
5. Ideal gas law (2.15).

The variables to be solved from this set of equations are the mass averaged velocity vector $\mathbf{v}$, the pressure $\mathbf{P}$, the temperature T, the density ρ and the mass fractions ω_i for $i = 1, \ldots, N$.

2.2 Boundary Conditions

In order to solve the system of PDE's describing the CVD process appropriate boundary conditions have to be chosen. If we start with a simple reactor chamber, consisting of an inflow and an outflow boundary, a number of solid walls and a reacting surface, then on each of these boundary we have to prescribe a condition.

To avoid misunderstanding, we will first give a definition of the normal **n**. By the normal **n** we mean the unit normal vector pointing outward the plane or boundary.

Solid Wall

On the solid, *non reacting* walls we apply the no slip and impermeability conditions for the velocity vector,

$$\mathbf{v} = \mathbf{0}.$$

On this wall we can have either a prescribed temperature T,

$$T = T_{\text{wall}},$$

or a zero temperature gradient for adiabatic walls,

$$\mathbf{n} \cdot \nabla T = 0.$$

Furthermore, on the solid wall there is no mass diffusion,

$$\mathbf{n} \cdot \mathbf{j}_i = \mathbf{n} \cdot (\mathbf{j}_i^C + \mathbf{j}_i^T) = 0.$$

For adiabatic walls and adiabatic systems the above boundary condition reduces to

$$\mathbf{n} \cdot \nabla \omega_i = 0.$$

Reacting Surface

On the reacting surface we have to prescribe the surface reactions. We assume that on the surface S different surface reactions can take place. There will be a net mass production $\mathcal{P}_i$ at these surfaces. For ω_i this net mass production is given by

$$\mathcal{P}_i = m_i \sum_{s=1}^{S} \sigma_{is} R_s^S,$$

where m_i is the mole mass, σ_{is} the stoichiometric coefficient for the gaseous and solid species for surface reaction s ($s = 1, \ldots, S$) and R_s^S the reaction rate of reaction s on the surface.

We assume the no-slip condition on the wafer surface. Since there is mass production in normal direction the normal component of the velocity will not be equal to zero. Now, we find for the velocity the conditions

$$\begin{aligned} \mathbf{n} \cdot \mathbf{v} &= \frac{1}{\rho} \sum_{i=1}^{N} m_i \sum_{s=1}^{S} \sigma_{is} R_s^S, \\ \mathbf{n} \times \mathbf{v} &= 0. \end{aligned} \tag{2.16}$$

on the reacting boundary.[4] For the wafer temperature we have

$$T = T_{\text{wafer}}.$$

On the wafer surface the net mass production of species i must be equal to the normal direction of the total mass flux of species i. The total mass flux of species i is given by

$$\rho\omega_i\mathbf{v} + \mathbf{j}_i.$$

Therefore we have on the wafer surface the following boundary condition

$$\mathbf{n} \cdot (\rho\omega_i\mathbf{v} + \mathbf{j}_i) = m_i \sum_{s=1}^{S} \sigma_{is} R_s^S.$$

Condition (2.16), $\mathbf{n} \times \mathbf{v} = 0$, gives always two conditions if we consider the three dimensional case. The normal on the reacting surface is known, $\mathbf{n} = [n_1, n_2, n_3]^T$ and the velocity vector is the vector $\mathbf{v} = [v_1, v_2, v_3]^T$, with v_1, v_2 and v_3 unknowns. We compute $\mathbf{n} \times \mathbf{v}$ to derive the number of conditions for $\mathbf{v}$. Hence,

$$\mathbf{n} \times \mathbf{v} = \begin{bmatrix} n_2v_3 - n_3v_2 \\ n_3v_1 - n_1v_3 \\ n_1v_2 - n_2v_1 \end{bmatrix}.$$

Since n_i is known and v_i not, we are able to compute the three velocity components from the equation $\mathbf{n} \times \mathbf{v} = 0$. The first equation gives $v_3 = \frac{n_3v_2}{n_2}$ and the second $v_1 = \frac{n_1n_3v_2}{n_2n_3}$. Hence, the third equation is also satisfied. Therefore, from an outer product follows a condition for only two components of the velocity vector.

Inflow

We take as boundary conditions in the inflow

$$\begin{aligned} \mathbf{n} \cdot \mathbf{v} &= v_{\text{in}} \\ \mathbf{n} \times \mathbf{v} &= 0, \end{aligned}$$

where the inflow velocity v_{in} with prescribed inflow of each species, denoted by Q_i for species i, is given as

$$v_{\text{in}} = \frac{T_{\text{in}}}{T^0} \frac{P^0}{P_{\text{in}}} \frac{10^{-3}}{60} \frac{1}{A_{\text{in}}} \sum_{i=1}^{N} Q_i = 5.66 \cdot 10^{-3} \frac{T_{\text{in}}}{A_{\text{in}} P_{\text{in}}} \sum_{i=1}^{N} Q_i.^5 \qquad (2.17)$$

[4] The outer product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$\mathbf{u} \times \mathbf{v} = (u_2v_3 - u_3v_2)\mathbf{e}_{(1)} + (u_3v_1 - u_1v_3)\mathbf{e}_{(2)} + (u_1v_2 - u_2v_1)\mathbf{e}_{(3)},$$

with $\mathbf{e}_{(\alpha)}$ the unit vector in the x_α direction. The outer product is not symmetric, hence, $\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}$.

[5] We used in (2.17) that the standard pressure $P^0 = 1.013 \cdot 10^5$Pa and the standard temperature $T^0 = 298.15K$.

The temperature in the inflow is prescribed as

$$T = T_{\text{in}}.$$

In [16] the author also prescribes

$$\mathbf{n} \cdot (\lambda \nabla T) = 0,$$

to prohibit a conductive heat flow through the inflow opening. After further analysis it was clear that $\lambda = 0$ is taken on the boundary[6].

Total mass flow of species i flowing into the reactor chamber must correspond to Q_i in the following way, see [16]:

$$\mathbf{n} \cdot \left(\rho \omega_i \mathbf{v} + \mathbf{j}_i^C + \mathbf{j}_i^T\right) = 5.66 \cdot 10^{-3} \frac{Q_i m_i}{R A_{\text{in}}}, \tag{2.18}$$

with R the universal gas constant. To satisfy (2.18) we put

$$\mathbf{n} \cdot \left(\mathbf{j}_i^C + \mathbf{j}_i^T\right) = 0, \tag{2.19}$$

and

$$\mathbf{n} \cdot (\rho \omega_i \mathbf{v}) = 5.66 \cdot 10^{-3} \frac{Q_i m_i}{R A_{\text{in}}}. \tag{2.20}$$

By taking the concentration- and thermal- diffusion coefficients on the boundary equal to zero, we satisfy (2.19). Elementary analysis of (2.20) gives that the inlet concentrations should be taken fixed as

$$\omega_i = \frac{m_i Q_i}{\sum_{j=1}^{N} M_j Q_j}. \tag{2.21}$$

Outflow

In the outflow opening we assume zero gradients in the normal direction of the outflow for the total mass flux vector, the heat flux and the diffusion fluxes. Furthermore, we assume that the velocity at the outflow boundary is in the normal direction. For the velocity we have the boundary conditions

$$\begin{aligned} \mathbf{n} \cdot (\nabla \rho \mathbf{v}) &= 0, \\ \mathbf{n} \times \mathbf{v} &= 0. \end{aligned}$$

For the heat and diffusion fluxes we get

$$\begin{aligned} \mathbf{n} \cdot (\lambda \nabla T) &= 0, \text{ and} \\ \mathbf{n} \cdot (\mathbf{j}_i^C + \mathbf{j}_i^T) &= 0. \end{aligned}$$

[6] Is it possible to have a discontinuous thermal conductivity coefficient λ?

Chapter 3

Stiffness

The note of *stiffness* has already been mentioned in Chapter 2. In this (short) chapter we pay some attention to this notion.

While the intuitive meaning of stiff is clear to all specialists, much controversy is going on about it's correct 'mathematical definition'. The most pragmatical opinion is also historically the first one, Curtiss & Hirschfelder [4] :

> "Stiff equations are equations where certain implicit methods, in particular BDF, perform better, usually tremendously better, than explicit ones."

A more recent opinion of Hundsdorfer & Verwer [13] :

> "Stiffness has no mathematical definition. Instead it is an operational concept, indicating the class of problems for which implicit methods perform (much) better than explicit methods."

The eigenvalues of the Jacobian $\frac{\delta f}{\delta y}$ play certainly a role in this decision, but quantities such as the dimension of the system, the smoothness of the solution or the integration interval are also important.

In this chapter we give first an appropriate example. Subsequently we give a stability analysis for the Euler forward method, where the role of the eigenvalues of the Jacobian become clear.

3.1 Example of Stiff Equation

As mentioned before, stiff equations belong to the class of ODEs for which explicit methods do not work. There are many 'standard' examples of stiff equations. Next, we treat an example coming from chemical reacting systems, taken from [26]. We selected this 'standard' example because of the connectivity with the topic of this research.

Example 3.1. *Consider the chemical species A, B and C. The Robertson reaction system is then given as:*

$$\begin{array}{lcll} A & \xrightarrow{0.04} & B & (slow) \\ B+B & \xrightarrow{3\cdot 10^7} & C+B & (very\ fast) \\ B+C & \xrightarrow{10^4} & A+C & (fast) \end{array}$$

which leads to the equations

$$\begin{array}{lllrrrl} A: & y_1' & = & -0.04y_1 & +10^4 y_2 y_3, & & y_1(0)=1 \\ B: & y_2' & = & 0.04y_1 & -10^4 y_2 y_3 & -3\cdot 10^7 y_2^2, & y_2(0)=0 \\ C: & y_3' & = & & & 3\cdot 10^7 y_2^2, & y_3(0)=0. \end{array} \tag{3.1}$$

Introduce

$$y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}, \tag{3.2}$$

so that (3.1) can be written as

$$y' = f(y), \quad y(0) = [1,0,0]^T, \tag{3.3}$$

where the definition of $f(y)$ is self-evident.

The ODE system will be solved using two methods:

1. *Euler Forward, which is an explicit method,*

$$\frac{y_{n+1}-y_n}{\tau} = f(y_n), \tag{3.4}$$

2. *Euler Backward, which is implicit,*

$$\frac{y_{n+1}-y_n}{\tau} = f(y_{n+1}). \tag{3.5}$$

The numerical solutions obtained for $y_2(t)$ with Euler Forward and Backward are displayed in Figure 3.1. We observe that the solution y_2 rapidly reaches a quasi-stationary position in the vicinity of $y_2' = 0$, which in the beginning is at $0.04 \approx 3\cdot 10^7 y_2^2$, hence $y_2 \approx 3.65\cdot^{-5}$, and then very slowly goes back to zero again.

It can be seen in Figure 3.1 that the implicit method, i.e. Euler Backward, integrates (3.1) without problem. The explicit Euler Forward has violent oscillations when the step-size τ is too large. We remark that the solution of Euler Backward is of good quality.

Apparently, in the sense of Curtiss and Hirschfelder (and Hundsdorfer and Verwer) the system of ODEs (3.1) is stiff.

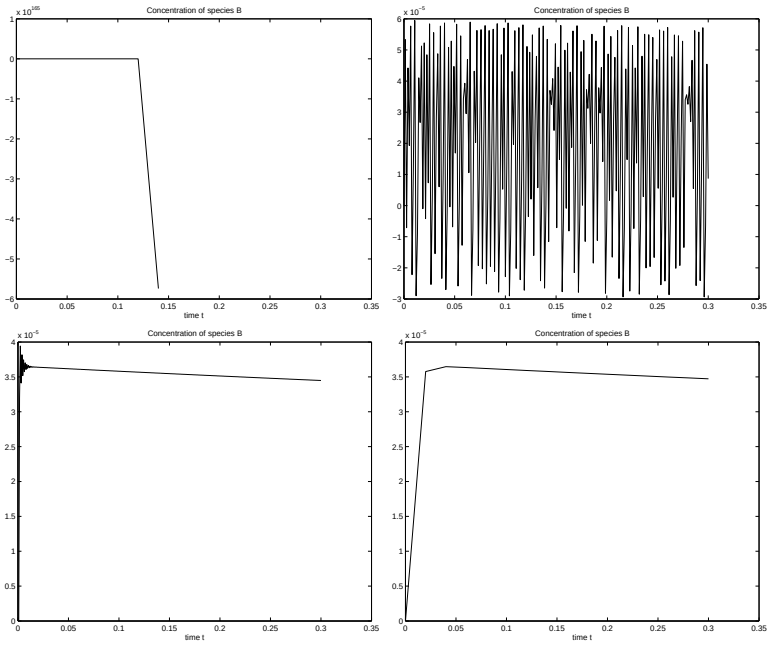

Figure 3.1: Numerical solution for $y_2(t)$ with Euler Forward with step-size $\tau = \frac{1}{50}$ (upper-left), with $\tau = \frac{1}{1000}$ (upper-right), with $\tau = \frac{1}{1250}$ (lower-left) and Euler Backward with steps-size $\tau = \frac{1}{50}$ (lower-right). Remark that the axis in the upper-left figure has another scale, i.e., it is scaled by a factor 10^{165}.

3.2 Stability Analysis for Euler Forward Method

Let $\varphi(t)$ be a smooth solution of $y' = f(t, y)$. We linearize f in its neighborhood as follows

$$y'(t) = f(t, \varphi(t)) + \frac{\delta f}{\delta y}(t, \varphi(t))(y(t) - \varphi(t)) + \cdots \tag{3.6}$$

and introduce $\bar{y}(t) = y(t) - \varphi(t)$ to obtain

$$\bar{y}'(t) = f(t, \varphi(t)) + \frac{\delta f}{\delta y}(t, \varphi(t))\bar{y}(t) + \cdots = J(t)\bar{y}(t) + \cdots \tag{3.7}$$

As a first approximation, we consider the Jacobian $J(t)$ as constant and neglect the error terms. Omitting the bars we arrive at

$$y' = Jy. \tag{3.8}$$

If we now apply the Euler forward method to (3.8), we obtain

$$y_{n+1} = R(hJ)y_n, \tag{3.9}$$

with

$$R(z) = 1 + z. \tag{3.10}$$

To study the behavior of (3.9), we assume that J is diagonalizable with eigenvectors $v_1, \ldots, v_k$. As a consequence, y_0 can be written in this basis as

$$y_0 = \sum_{i=1}^{k} \alpha_i v_i. \tag{3.11}$$

If the λ_i are the eigenvalues of J, then using (3.11), (3.9) can be written as

$$y_m = \sum_{i=1}^{k} (R(h\lambda_i))^m \alpha_i v_i. \tag{3.12}$$

It is clear that y_m remains bounded for $m \to \infty$, if for all eigenvalues the complex number $z = h\lambda_i$ lies in the set

$$\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}. \tag{3.13}$$

This leads to the explanation of the results found in the example of the previous section.

Example 3.1. (Continued) *The Jacobian for the Robertson reaction system (3.1) is given by*

$$\begin{pmatrix} -0.04 & 10^4 y_3 & 10^4 y_2 \\ 0.04 & -10^4 y_3 - 6 \cdot 10^7 y_2 & -10^4 y_2 \\ 0 & 6 \cdot 10^7 y_2 & 0 \end{pmatrix}. \tag{3.14}$$

In the neighborhood of the equilibrium $y_1 = 1$, $y_2 = 0.0000365$, $y_3 = 0$ *the Jacobian is equal to*

$$\begin{pmatrix} -0.04 & 0 & 0.365 \\ 0.04 & -2190 & 0.365 \\ 0 & 2190 & 0 \end{pmatrix}, \tag{3.15}$$

with eigenvalues

$$\lambda_1 = 0, \quad \lambda_2 = -0.405 \quad \text{and} \quad \lambda_3 = -2189.6. \tag{3.16}$$

The third eigenvalue, λ_3*, produces stiffness. For stability for Euler Forward we need* $\lambda_3 \tau \geq -2$*. Hence,* $\tau \leq \frac{2}{2189.6} = \frac{5}{5474} \approx 0.0009$.

The implicit Euler Backward method is unconditionally stable, thus there are no restrictions on the step-size τ*. This confirms the numerical observations.*

Chapter 4

Test Problem

The mathematical model of the CVD process described in Chapter 2 consists of a number of coupled non-linear partial differential equations with boundary conditions. In general, it is impossible to solve this set of PDEs analytically. As mentioned in [16], for simple reactor configurations and a drastically simplified transport model useful analytic solutions can be obtained. See for example [7, 18]. In this study we have a general model, which can be applied to various reactor configurations and processes. Therefore, we use numerical methods to find approximate solutions of the full set of PDEs. The aim is to find the most efficient solution method.

The finite volume approach is the most widely used class of numerical methods for computing heat and mass transfer in variable property fluid flows with chemical reactions. Therefore, the finite volume approach will be used in this study. A detailed description of the finite volume approach can be found in [24] and numerous other publications.

The main goal of this project is to find an efficient solution method for solving the advection-diffusion-reaction equations, which are also known in this report as species equations. In the test problem we restrict ourselves to solve a coupled system of species equations on a 2D computational grid. As a first approximation we also neglect the surface reactions. This results in having inflow, outflow and solid wall boundary conditions. Stiffness in this system comes from the gas phase reactions.

In this chapter a 2D axisymmetric test problem will be described. The chapter is organized as follows. We start with defining the computational grid. Subsequently, we give the finite volume discretization of the species equations. The last sections of this chapter are reserved for more details on the chemical properties of the test model.

4.1 The Computational Grid

We restrict ourselves to the two-dimensional case. We use a cell-centered non-uniform grid. There are two ways to arrange the unknowns on the grid :

1. colocated arrangement,
2. staggered arrangement.

Visualization of both arrangements is done on a cell-centered uniform grid in Figure 4.1. When all discrete unknowns are located in the center of a cell, the arrangement is called colocated. In the case that the pressure, temperature and species mass fraction are located in the cell-centers and the velocity components are located at the cell face centers, the arrangement is called staggered.

In this test problem we use a colocated grid, because the CFD package X-stream also uses a colocated grid. Recall that the solvers developed in this project will be implemented in X-stream.

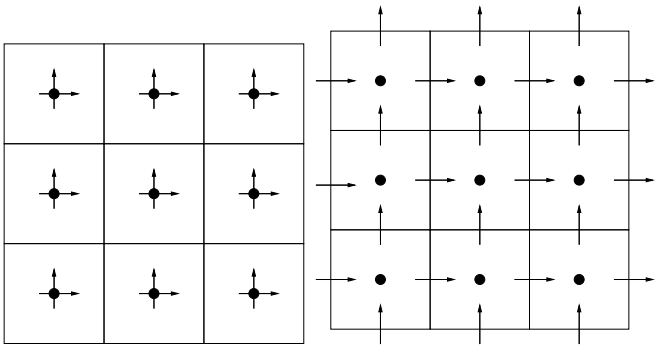

Figure 4.1: Two-dimensional cell-centered uniform grid with a colocated arrangement of the unknowns (left) and a staggered arrangement of the unknowns (right).

4.2 Finite Volume Discretization of the General Transport Equation

The test problem we have in this chapter consists of N coupled species concentration equations. Recall that N represents the number of gas species in the reactor. With respect to the diffusion flux we assume the following :

- we neglect thermo-diffusion, thus we consider diffusion to exist of ordinary diffusion only,
- ordinary diffusion is described by the Wilke approach (Fick's law).

The species concentration equations then take the form

$$\frac{\partial(\rho\omega_i)}{\partial t} = -\nabla\cdot(\rho v\omega_i) + \nabla\cdot(\rho\mathbb{D}_i'\nabla\omega_i) + m_i\sum_{k=1}^{K}\nu_{ik}R_k^g, \tag{4.1}$$

where K is the number of reactions in the gas phase and $\mathbb{D}_i'$ the effective diffusion coefficient. Equation (4.1) is valid for all species in the gas mixture. Since the relation

$$\sum_{i=1}^{N}\omega_i = 1, \tag{4.2}$$

holds, only $(N-1)$ coupled species concentration equations have to be solved.

The general form of equation (4.1) is

$$\frac{\partial(\rho\phi)}{\partial t} = -\nabla\cdot(\rho v\phi) + \nabla\cdot(\Gamma_\phi\nabla\phi) + S_\phi, \tag{4.3}$$

whereby

$$\phi = \omega_i, \quad \Gamma_\phi = \rho\mathbb{D}_i' \quad \text{and} \quad S_\phi = m_i\sum_{k=1}^{K}\nu_{ik}R_k^g. \tag{4.4}$$

The discretization of advection-diffusion-reaction equation (4.3) will be done for its 2D axisymmetric (r,z) form, that is

$$\frac{\partial(r\rho\phi)}{\partial t} = -\nabla_{r,z}\cdot(r\rho v\phi) + \nabla_{r,z}\cdot(r\Gamma_\phi\nabla_{r,z}\phi) + S_\phi, \tag{4.5}$$

where $\nabla_{r,z}$ is the operator $\nabla_{r,z}(\cdot) = (\frac{\partial\cdot}{\partial r}, \frac{\partial\cdot}{\partial z})^T$. Remark that the 2D Cartesian form (4.3) of (4.5) can be obtained by taking $r = 1$ in (4.5).

Next, we will compute the volume integral over the control volume surrounding the grid point P, see Figure 4.2.

Grid point P has four neighbors, indicated by N(orth), S(outh), E(ast) and W(est), and the corresponding walls are indicated by n, s, e and w. For the sake of clarity, we write in the remainder of this chapter

$$\mathbf{v} = \begin{bmatrix} u \\ v \end{bmatrix}, \tag{4.6}$$

where u is the velocity component in r-direction and v the velocity component in z-direction.

Integrating Equation (4.5) over the control volume $\Delta r\Delta z$ surrounding P gives

$$\iint_{\Delta r\Delta z}\frac{\partial(r\rho\phi)}{\partial t}\,dr\,dz = \tag{4.7}$$

$$\iint_{\Delta r\Delta z} -\nabla_{r,z}\cdot(r\rho v\phi) + \nabla_{r,z}\cdot(r\Gamma_\phi\nabla_{r,z}\phi) + S_\phi\,dr\,dz. \tag{4.8}$$

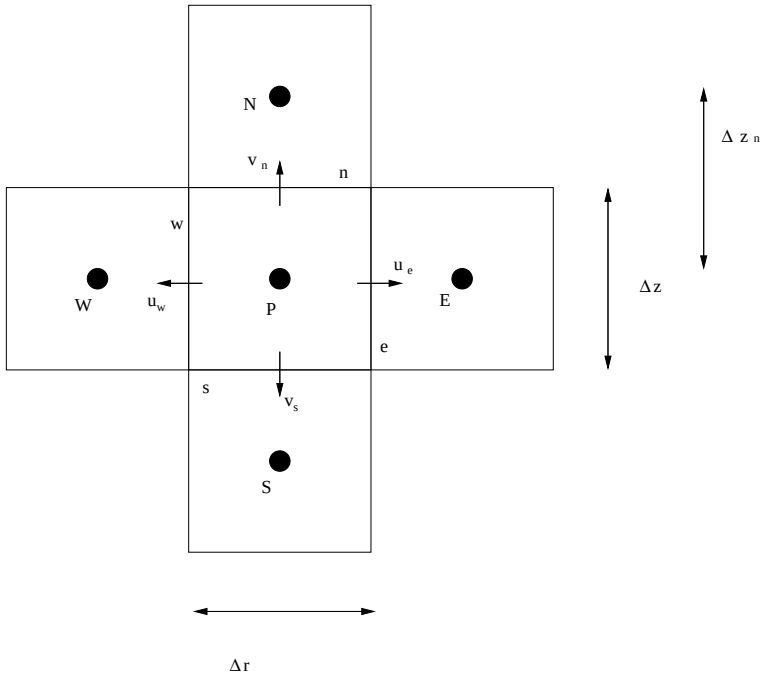

Figure 4.2: Grid cells

By using the Gauß theorem[1] we may write

$$\int\int_{\Delta r \Delta z} \frac{\partial(r\rho\phi)}{\partial t}\,dr\,dz = -\int_n r\rho v\phi\,dr + \int_s r\rho v\phi\,dr - \int_e r\rho u\phi\,dz + \int_w r\rho u\phi\,dz + \int_n r\Gamma_\phi \frac{\partial\phi}{\partial z}\,dr - \int_s r\Gamma_\phi \frac{\partial\phi}{\partial z}\,dr + \int_e r\Gamma_\phi \frac{\partial\phi}{\partial r}\,dz - \int_w r\Gamma_\phi \frac{\partial\phi}{\partial r}\,dz + \int\int_{\Delta r \Delta z} rS_\phi\,dr\,dz \quad (4.9)$$

[1]

Theorem (Divergence Theorem of Gauß). *For any volume $V \subset \mathbb{R}^d$ with piecewise smooth closed surface S and any differentiable vector field $\mathbf{F}$ we have*

$$\int_V \nabla \cdot \mathbf{F}\,dV = \int_S \mathbf{F} \cdot \mathbf{n}\,dS,$$

where $\mathbf{n}$ is the outward unit normal on S.

The remaining integrals are approximated as

$$\frac{d(r_P\rho_P\phi_P)}{dt}\Delta r\Delta z = -r_n\rho_n v_n\phi_n\Delta r + r_s\rho_s v_s\phi_s\Delta r - r_e\rho_e v_e\phi_e\Delta z + r_w\rho_w v_w\phi_w\Delta z + r_n\Gamma_{\phi,n}\left.\frac{\partial\phi}{\partial z}\right|_n\Delta r - r_s\Gamma_{\phi,s}\left.\frac{\partial\phi}{\partial z}\right|_s\Delta r + r_e\Gamma_{\phi,e}\left.\frac{\partial\phi}{\partial r}\right|_e\Delta z - r_w\Gamma_{\phi,w}\left.\frac{\partial\phi}{\partial r}\right|_w\Delta z + r_pS_{\phi,P}\Delta r\Delta z \quad (4.10)$$

In the above formulation the time derivative will not be approximated yet. The next part of this report is dedicated to the subject of time integration methods.

The densities ρ and the diffusion coefficients Γ_ϕ at the cell walls are approximated by the harmonic mean as

$$\rho_n = \frac{2\rho_P\rho_N}{\rho_P+\rho_N},$$
$$\Gamma_n = \frac{2\Gamma_P\Gamma_N}{\Gamma_P+\Gamma_N},$$

and similar for the other cell walls. Because the density function is smooth in the computational domain, its (harmonic mean) approximation can be replaced by

$$\rho_n = \frac{1}{2}(\rho_N+\rho_P). \quad (4.11)$$

We remark that the latter is done in [16].

For the approximation of the value of ϕ and its first derivative on the cell walls, we consider the following three methods:

central scheme :
$$\phi_n = \tfrac{1}{2}(\phi_N+\phi_P)$$
$$\left.\frac{\partial\phi}{\partial z}\right|_n = \frac{\phi_N-\phi_P}{\Delta z_n}$$

1st order upwind scheme :
$$\phi_n = \begin{cases} \phi_P & \text{for} \quad v_n \geq 0 \\ \phi_N & \text{for} \quad v_n < 0 \end{cases}$$
$$\left.\frac{\partial\phi}{\partial z}\right|_n = \frac{\phi_N-\phi_P}{\Delta z_n}$$

hybrid scheme :
$$\phi_n = \begin{cases} \phi_N & \text{for} \quad Pe_{\Delta n} < -2 \\ \tfrac{1}{2}(\phi_N+\phi_P) & \text{for} \quad |Pe_{\Delta n}| \leq 2 \\ \phi_P & \text{for} \quad Pe_{\Delta n} > 2 \end{cases}$$
$$\left.\frac{\partial\phi}{\partial z}\right|_n = \begin{cases} \frac{\phi_N-\phi_P}{\Delta z_n} & \text{for} \quad |Pe_{\Delta n}| \leq 2 \\ 0 & \text{for} \quad |Pe_{\Delta n}| > 2 \end{cases}$$

where the cell Peclet number Pe_Δ is on the n-wall is defined as

$$Pe_{\Delta n} = \frac{\rho_n v_n\Delta z_n}{\Gamma_{\phi,n}}, \quad (4.12)$$

and similar for the other walls.

The central scheme has second order accuracy, but becomes unstable for large $|Pe_{\Delta n}|$. It is generally known that for large $|Pe_{\Delta n}|$ the central scheme leads to wiggles in the solution.

The upwind scheme damps these wiggles. The disadvantage is that in turn for damping these wiggles one gets loss of accuracy, i.e. the upwind scheme has only first order accuracy. Furthermore, the upwind scheme produces 'large' numerical diffusion.

The hybrid scheme combines the advantages of the central and the upwind scheme. For $|Pe_{\Delta}| > 2$, it locally switches to the upwind scheme and sets the diffusion contribution to zero.

Finally we remark that it is common in CVD processes that the Reynolds number is low. Based on previous research, low Reynolds number imply low Peclet numbers. This means that the using the hybrid scheme results usually in a central differencing scheme.

4.3 Boundary Conditions

For every ADR-equation of the form (4.5) a set of discretized boundary conditions is needed. Subsequently we will discretize the inflow, solid wall and outflow boundary conditions. Recall that in the test-problem we do not consider reacting surfaces.

Inflow

In Figure 4.3 an inflow opening normal to the z-direction has been illustrated. For the species concentration equations (4.5) we wish to impose

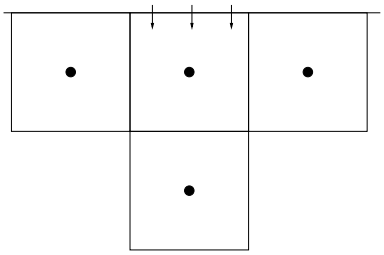

Figure 4.3: Inflow boundary condition

boundary conditions (2.19) and (2.21). The first is simply done by setting the diffusion constant $\Gamma_{\phi,n}$ equal to zero. The second is satisfied by replacing ϕ_n in (4.10) by ϕ_{in}, where ϕ_{in} is the prescribed inlet concentration.

Solid Wall

In Figure 4.4 a solid wall is illustrated. Since there is assumed that there is no thermal diffusion, we only have

$$\mathbf{n} \cdot \nabla \phi = 0, \tag{4.13}$$

on wall w. Substituting this condition into (4.9) gives a modification of the integrals

- $\int_w r\rho u\phi \, dz$ changes into

$$\int_w r\rho u\phi \, dz = r\rho_w u_w \phi_P \Delta z \tag{4.14}$$

  where we used that $\phi_W = \phi_P$ (comes from the fact that $\partial\phi/\partial r = 0$) and $\phi_w = \frac{1}{2}(\phi_P + \phi_W)$,

- $\int_w r\Gamma_\phi \frac{\partial \phi}{\partial r} \, dz$ becomes zero, because

$$\frac{\partial \phi}{\partial r} = 0. \tag{4.15}$$

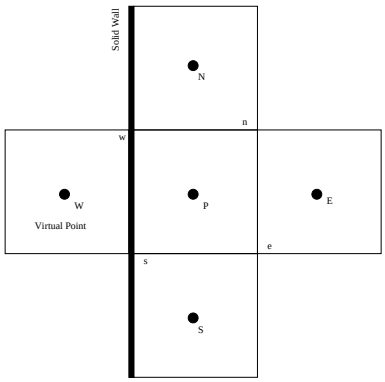

Figure 4.4: Solid wall boundary condition

Outflow

The outflow boundary condition with respect to the concentrations, without thermo diffusion, is $\mathbf{n} \cdot \nabla \phi = 0$. Remark that this b.c. is equal to the solid wall boundary condition and therefore we refer to the previous subsection for the discretization.

4.4 Chemistry Properties

In this section we present the details of the chemical model used for this test problem. The gas mixture in the CVD reactor of this test problem consists of 7 species, see Section 4.4.1. This model is a simplified model of a CVD process that deposits silicon Si from silane SiH_4. The process of depositing silicon from silane is one of the most applied and studied CVD processes in the IC industry.

In this problem we use the reactor configuration as given in Figure 4.5. This reactor has one inflow boundary, a number of solid walls, one reacting surface and two outflow boundaries. The reactor is axisymmetric, and hence, the computational domain results into the domain given in Figure 4.6. Furthermore, we assume that the gas flow in the reactor is also axisymmetric.

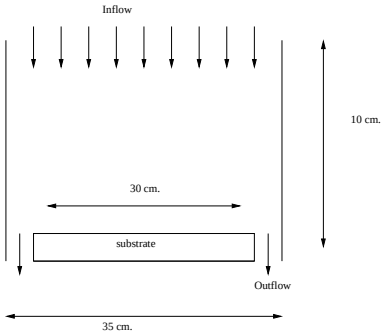

Figure 4.5: Reactor geometry

We conclude with some last remarks about this CVD process. From the top of the reactor, the inflow boundary, a gas mixture enters the reactor. This gas mixture consists of silane SiH_4 and the carrier gas helium He. When the mixture enters the reactor it has a uniform temperature $T_{\text{in}} = 300$ K and a uniform velocity U_{in}. The inlet silane mole fraction is $f_{\text{in,SiH}_4} = 0.001$, the rest is helium. At a distance of 10 cm. below the inlet, a susceptor with temperature $T_{\text{s}} = 1000$ K is placed. The susceptor has a diameter of 30 cm. On the susceptor surface reactions take place leading to the deposition of silicon. In the first numerical experiments, these surface reactions will be omitted.

4.4.1 Gas-Phase Reaction Model

Due to gas phase reactions in the reactor the gas mixture consists, besides helium and silane, of the following species :

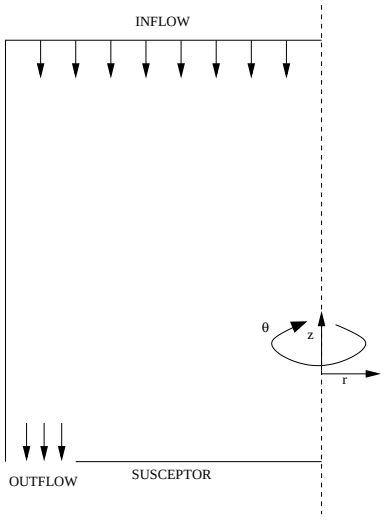

Figure 4.6: Computational domain

SiH_4	He
SiH_2	H_2SiSiH_2
Si_2H_6	Si_3H_8
H_2	

We use the following reaction mechanism :

G1 :	SiH_4	$\rightleftarrows$	$SiH_2 + H_4$
G2 :	Si_2H_6	$\rightleftarrows$	$SiH_4 + SiH_2$
G3 :	Si_2H_6	$\rightleftarrows$	$H_2SiSiH_2 + H_2$
G4 :	SiH_2+Si_2H_6	$\rightleftarrows$	Si_3H_8
G5 :	$2Si_2$	$\rightleftarrows$	H_2SiSiH_2

Recall from Chapter 2 that the forward reaction rate k_{forward} is fitted according to the modified Law of Arrhenius as

$$k^g_{\text{forward}}(T) = A_k T^{\beta_k} e^{\frac{-E_k}{RT}}. \tag{4.16}$$

The backward reaction rates are self consistent with

$$k^g_{\text{backward}}(T) = \frac{k^g_{\text{forward}}(T)}{K^g} \left(\frac{RT}{P^0}\right)^{\sum_{i=1}^{N} \nu_{ik}},$$

where the reaction equilibrium K^g is approximated by

$$K^g(T) = A_{\text{eq}} T^{\beta_{\text{eq}}} e^{\frac{-E_{\text{eq}}}{RT}}. \tag{4.17}$$

In Table 4.1 the forward rate constants A_k, β_k and E_k are given. The fit parameters for the gas phase equilibrium are given in Table 4.2.

Reaction	A_k	β_k	E_k
G1	$1.09 \cdot 10^{25}$	-3.37	256000
G2	$3.24 \cdot 10^{29}$	-4.24	243000
G3	$7.94 \cdot 10^{15}$	0.00	236000
G4	$1.81 \cdot 10^{8}$	0.00	0
G5	$1.81 \cdot 10^{8}$	0.00	0

Table 4.1: Fit parameters for the forward reaction rates (4.16)

Reaction	A_{eq}	β_{eq}	E_{eq}
G1	$6.85 \cdot 10^{5}$	0.48	235000
G2	$1.96 \cdot 10^{12}$	-1.68	229000
G3	$3.70 \cdot 10^{7}$	0.00	187000
G4	$1.36 \cdot 10^{-12}$	1.64	233000
G5	$2.00 \cdot 10^{-7}$	0.00	272000

Table 4.2: Fit parameters for the equilibrium constants (4.17)

Other Data

Besides the chemical model of the reacting gases, also some other properties of the gas mixture are needed. As mentioned before, the inlet temperature of the mixture is 300 K, and the temperature at the wafer surface is 1000 K. The pressure in the reactor is 1 atm., which is equal to $1.013 \cdot 10^5$ Pa. Other properties are given in Table 4.3. The diffusion coefficients, according to Fick's Law, are given in Table 4.4.

Density $\rho(T)$	$1.637 \cdot 10^{-1} \cdot \frac{300}{T}$	[kg/m^3]
Specific heat c_p	$5.163 \cdot 10^3$	[J/kg/K]
Viscosity μ	$1.990 \cdot 10^{-5} \left(\frac{T}{300}\right)^{0.7}$	[kg/m/s]
Thermal conductivity λ	$1.547 \cdot 10^{-1} \left(\frac{T}{300}\right)^{0.8}$	[W/m/K]

Table 4.3: Gas mixture properties

SiH_4	$4.77 \cdot 10^{-6} \left(\frac{T}{300}\right)^{1.7}$
SiH_2	$5.38 \cdot 10^{-6} \left(\frac{T}{300}\right)^{1.7}$
H_2SiSiH_2	$3.94 \cdot 10^{-6} \left(\frac{T}{300}\right)^{1.7}$
Si_2H_6	$3.72 \cdot 10^{-6} \left(\frac{T}{300}\right)^{1.7}$
Si_3H_8	$3.05 \cdot 10^{-6} \left(\frac{T}{300}\right)^{1.7}$
H_2	$8.02 \cdot 10^{-6} \left(\frac{T}{300}\right)^{1.7}$

Table 4.4: Diffusion coefficients $\mathbb{D}'_i$, according to Fick's Law

Part II

Time Integration Methods

Chapter 1

Introduction

In this part of the thesis we present time integration methods which are of practical relevance for solving advection diffusion reaction problems. These methods have been selected from recent literature concerning this problem, e.g. [13, 20, 36, 37]

As seen in Part I the advection diffusion reaction equation is a PDE with appropriate boundary conditions and initial values. To approximate the solution $u(x,t)$ of an ADR equation, or system of ADR equations, we use the Method of Lines approach. The PDE is then first discretized in space on a certain grid Ω_h with mesh width $h > 0$ to yield a semi discrete system

$$w'(t) = F(t, w(t)), \quad 0 < t \leq T, \quad w(0) \text{ given}, \tag{1.1}$$

where $w(t) = (w_j(t))_{j=1}^m \in \mathbb{R}^m$, with m proportional to the number of grid points in spatial direction. The next step is to integrate the ODE system (1.1) with an appropriate time integration method. A number of methods is treated in the following sections. Among others we will discuss

1. The class of Runge Kutta Methods including the Rosenbrock Methods,
2. Time Splitting Methods,
3. Runge Kutta Chebyshev Methods,
4. Multi-rate (Runge-Kutta) Schemes.

We consider the non autonomous initial value problem of the system of ODEs

$$w'(t) = F(t, w(t)), \quad t > 0, \quad w(0) = w_0, \tag{1.2}$$

with given $F : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ and $w_0 \in \mathbb{R}^m$. The exact solution will be approximated in the grid points $t_n = n\tau$, $n = 0, 1, 2, \ldots, N$, where $\tau > 0$ is the time step. We denote the numerical approximations by $w_n \approx w(t_n)$.

To keep the presentation of the methods short, we assume the time step τ to be constant. However, we emphasize that in many applications variable step sizes should be used to obtain efficient codes and solvers.

Chapter 2

Runge-Kutta Methods

Runge-Kutta Methods belong to the class of one-step methods, e.g. this is the class of methods that step forward from computed approximations w_n at time t_n to new approximations w_{n+1} at the forward time t_{n+1} using as input only w_n. To compute the new approximation w_{n+1} the Runge-Kutta methods use a number of auxiliary intermediate approximations $w_{ni} \approx w(t_n + c_i\tau)$, $i = 1, 2, \ldots, s$, where s is the number of stages. These intermediate approximations serve to obtain a sufficiently level of accuracy for the approximations w_n at the grid points t_n. The general form of a Runge-Kutta method is

$$\begin{aligned} w_{ni} &= w_n + \tau \sum_{j=1}^{s} \alpha_{ij} F(t_n + c_j\tau, w_{nj}), \quad i = 1, 2, \ldots, s, \\ w_{n+1} &= w_n + \tau \sum_{i=1}^{s} b_i F(t_n + c_i\tau, w_{ni}). \end{aligned} \tag{2.1}$$

The coefficients α_{ij} and b_i define the particular method and $c_i = \sum_{j=1}^{s} \alpha_{ij}$.

Observe that a particular Runge-Kutta method is *explicit* if $\alpha_{ij} = 0$ for all $j \geq i$. All other particular methods are *implicit*. If $\alpha_{ij} = 0$ for all $j > i$, then we will call this method *diagonally implicit*. Particular Runge-Kutta methods can be represented in a so-called *Butcher-array*, e.g.,

$$\begin{array}{c|ccc} c_1 & \alpha_{11} & \cdots & \alpha_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & \alpha_{s1} & \cdots & \alpha_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$

The order p of a Runge-Kutta method is determined by its coefficients α_{ij}, b_i and c_i. By making use of Taylor series, one can derive conditions in

order to make the method consistent of order p. These conditions are given in Table 2.1 for $p = 1, 2, 3$ and 4. Table 2.1 is taken from [13].

order p	order conditions	
1	$b^T e = 1$	
2	$b^T ce = \frac{1}{2}$	
3	$b^T c^2 = \frac{1}{3}$	$b^T \mathcal{A} c = \frac{1}{6}$
4	$b^T c^3 = \frac{1}{4}$	$b^T \mathcal{C}\mathcal{A} c = \frac{1}{8}$
	$b^T \mathcal{A} c^2 = \frac{1}{12}$	$b^T \mathcal{A}^2 c = \frac{1}{24}$

Table 2.1: Order conditions of Runge-Kutta methods for $p = 1, 2, 3, 4$. The vector $b = (b_1, \ldots, b_s)^T$, $c = (c_1, \ldots, c_s)^T$, $e = (1, \ldots, 1)^T$, the matrix $\mathcal{A} = (\alpha_{ij})_{ij}$, $\mathcal{C} = \mathrm{diag}(c_i)$ and $c^k = \mathcal{C}^k e$.

Example 2.1. *We give some Butcher arrays of simple explicit Runge-Kutta methods.*

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

At the left we have the Butcher array of the familiar Euler Forward method of order one. At the right we have a method known as the second-order explicit trapezoidal rule or modified Euler method, e.g.,

$$w_{n+1} = w_n + \frac{1}{2}\tau F(t_n, w_n) + \frac{1}{2}\tau F(t_n + \tau, w - n + \tau F(t_n, w_n)).$$

An example of explicit method of order four is given by the Butcher array

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

This method is better known as the method of Runge-Kutta. Completely

written out it is

$$\begin{aligned}
w_{n1} &= w_n, \\
w_{n2} &= w_n + \frac{1}{2}\tau F(t_n, w_{n1}), \\
w_{n3} &= w_n + \frac{1}{2}\tau F\left(t_n + \frac{1}{2}\tau, w_{n2}\right), \\
w_{n4} &= w_n + \tau F\left(t_n + \frac{1}{2}\tau, w_{n3}\right), \\
w_{n+1} &= w_n + \tau\left(\frac{1}{6}F(t_n, w_{n1}) + \frac{1}{3}F\left(t_n + 1/2\tau, w_{n2}\right)\right. \\
&\quad \left. + \frac{1}{3}F\left(t_n + 1/2\tau, w_{n3}\right) + \frac{1}{6}F(t_n + \tau, w_{n4})\right).
\end{aligned}$$

Taking $k_i = \tau F(t_n + c_i\tau, w_{ni})$, the above method is in the more familiar form

$$\begin{aligned}
k_1 &= \tau F(t_n, w_n), \\
k_2 &= \tau F\left(t_n + \frac{1}{2}\tau, w_n + \frac{1}{2}k_1\right), \\
k_3 &= \tau F\left(t_n + \frac{1}{2}\tau, w_n + \frac{1}{2}k_2\right), \\
k_4 &= \tau F(t_n + \tau, w_n + k_3), \\
w_{n+1} &= w_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).
\end{aligned}$$

We will refer to it as the classical fourth order method.

2.1 The Stability Function

Consider the Dahlquist test equation $w'(t) = \lambda w(t), \lambda \in \mathbb{C}$. Applying a Runge-Kutta method (2.1) to the test equation gives the recursion

$$w_{n+1} = R(z)w_n,$$

with $R(z)$ the stability function and $z = \tau\lambda$. This function can be found to be

$$R(z) = 1 + zb^T(\mathbf{I} - z\mathcal{A})^{-1}e, \tag{2.2}$$

where $e = (1, 1, \ldots, 1)^T$. It follows from (2.2) that for explicit R-K methods the stability function $R(z)$ becomes a polynomial of degree $\leq s$. For implicit R-K methods it becomes a rational function with both denominator and numerator polynomials of degree $\leq s$.

Example 2.2. *For every explicit R-K method of order $p = s$ and $s \leq 4$ the stability function is given by the polynomial*

$$R(z) = 1 + z + \frac{1}{2}z^2 + \cdots + \frac{1}{s!}z^s.$$

For $s = 3$ and 4 the stability regions $\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1\}$ have been plotted in Figure 2.1.

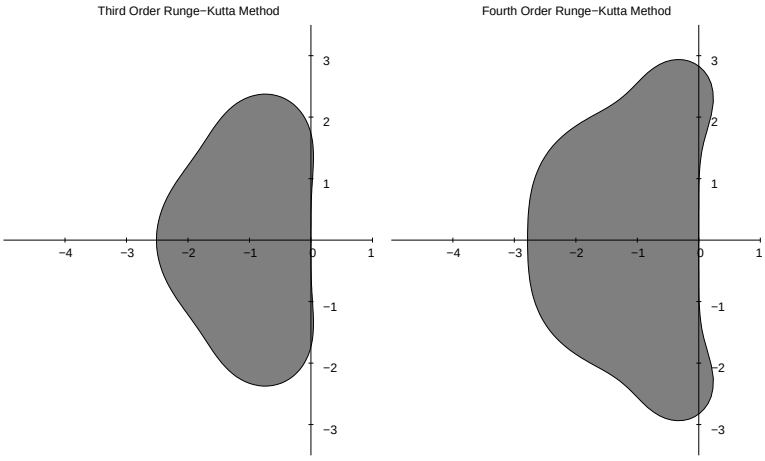

Figure 2.1: Stability regions $\mathcal{S} = \{z \in \mathbb{C} : |R(z)| \leq 1$ for $s = 3$ (left) and $s = 4$ (right).

2.2 Rosenbrock Methods

Rosenbrock methods belong to the family of Runge-Kutta methods. They are named after Rosenbrock, see [27], who was the first one that proposed methods of this kind. In literature different forms have been used. Nowadays Rosenbrock methods are understood to solve an autonomous, stiff ODE system $w'(t) = F(w(t))$. We derive the class of Rosenbrock methods from the diagonally implicit Runge-Kutta methods. Recall that an s-stage diagonally implicit RK-method, with $F = F(w(t))$, is given as

$$\begin{aligned} w_{ni} &= w_n + \tau \sum_{j=1}^{s} \alpha_{ij} F(w_{nj}), \quad i = 1, 2, \ldots, s, \\ w_{n+1} &= w_n + \tau \sum_{i=1}^{s} b_i F(w_{ni}) \end{aligned} \tag{2.3}$$

with Butcher array as in Figure 2.2.
Rewriting (2.3) gives

$$\begin{aligned} k_i &= \tau F\left(w_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j + \alpha_{ii} k_i \right), \quad i = 1, \ldots, s, \\ w_{n+1} &= w_n + \sum_{i=1}^{s} b_i k_i. \end{aligned} \tag{2.4}$$

$$\begin{array}{c|cccc} c_1 & \alpha_{11} & & & \\ & \alpha_{21} & \alpha_{22} & & \\ \vdots & \vdots & & \ddots & \\ c_s & \alpha_{s1} & \cdots & & \alpha_{ss} \\ \hline & b_1 & \cdots & & b_s \end{array}$$

Figure 2.2: Butcher array of an s-stage diagonally implicit RK-method

Now, the idea is not to solve k_i from (2.4), but to linearize $F\left(w_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j + \alpha_{ii} k_i\right)$ around $x = w_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j$:

$$k_i = \tau F(x) + \tau F'(x) \alpha_{ii} k_i. \tag{2.5}$$

This can be interpreted as applying *one* Newton iteration to (2.4) with starting value $k_i = 0$. Instead of continuing the Newton iteration we consider (2.5) as a new class of methods.

Before defining the actual class of Rosenbrock methods, we give some additional remarks to obtain more efficient methods. A lot of computational advantage can be obtained by replacing the Jacobians $F'(x)$ by the Jacobian $\mathbf{A} = F'(w_n)$, in each time step only one Jacobian has to be computed. Furthermore, we gain more freedom by introducing linear combinations of the terms $\mathbf{A}k_j$ into (2.5). The following class of methods can be defined :

Definition 2.3. *An s-stage Rosenbrock method is given as*

$$\begin{aligned} k_i &= \tau F(w_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j) + \tau \mathbf{A} \sum_{j=1}^{i} \gamma_{ij} k_j, \\ w_{n+1} &= w_n + \sum_{i=1}^{s} b_i k_i \end{aligned} \tag{2.6}$$

where $\mathbf{A} = \mathbf{A}_n$ is the Jacobian $F'(w(t))$.

Definition 2.3 is taken from [13]. Again, the coefficients b_{ij}, α_{ij} and γ_{ij} define a particular method and are selected to obtain a desired level of consistency and stability.

Remark that computing an approximation w_{n+1} from w_n, in each stage i a linear system of algebraic equations with the matrix $\mathbf{I} - \gamma_{ii}\tau\mathbf{A}$ has to be solved. To save computing time for large dimension systems $w'(t) = F(w(t))$ the coefficients γ_{ii} are taken constant, e.g., $\gamma_{ii} = \gamma$. Then in every time-step the matrix $(\mathbf{I} - \gamma_{\mathbf{ii}}\mathbf{A})$ is the same. To solve these large systems LU decomposition or (preconditioned) iterative methods could be used.

Finally we remark that taking $\mathbf{A}$ to the zero matrix in (2.6), this leads to a standard explicit Runge-Kutta method. The final remark, taken from

[13], Rosenbrock methods have proven to be successful in many stiff ODE and PDE problems, especially in the low to moderate accuracy range.

As can be found in [13] the definition of order of consistency is the same as for Runge-Kutta. Define

$$\beta_{ij} = \alpha_{ij} + \gamma_{ij}, \quad c_i = \sum_{j=1}^{i-1} \alpha_{ij} \quad \text{and} \quad d_i = \sum_{j=1}^{i-1} \beta_{ij}.$$

In Table 2.2, taken from [13], the order conditions for $s \leq 4$ and $p \leq 3$ and $\gamma_{ii} = \gamma =$ constant can be found. For Rosenbrock methods with constant

order p	order conditions
1	$b_1 + b_2 + b_3 + b_4 = 1$
2	$b_1 d_2 + b_3 d_3 + b_4 d_4 = \frac{1}{2} - \gamma$
3	$b_2 c_2^2 + b_3 c_3^2 + b_4 d_4^2 = \frac{1}{3}$
	$b_3 \beta_{32} d_2 + b_4(\beta_{42} d_2 + \beta_{43} d_3) = \frac{1}{6} - \gamma + \gamma^2$

Table 2.2: Order conditions of Rosenbrock methods with $\gamma_{ii} = \gamma$ for $s \leq 4$ and $p \leq 3$.

coefficients $\gamma_{ii} = \gamma$ the stability function has the following form,

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

where $P(z)$ is a polynomial of degree $\leq s$.

Example 2.4. *By taking $s = 1$, $b_1 = 1$, $\alpha_{11} = 1$ and $\gamma_{11} = \gamma$, γ a free parameter, in (2.6) we defined the one-stage method*

$$\begin{aligned} w_{n+1} &= w_n + k_1 \\ k_1 &= \tau F(w_n) + \tau \mathbf{A} k_1. \end{aligned}$$

Since $\beta_{11} = 1 + \gamma$ and $\beta_{ij} = c_i = d_i = 0$ for all $i = 1, 2, 3, 4$ and $j = 2, 3, 4$, we only have a second order method if $\gamma = 2$.

The stability function is

$$R(z) = \frac{1 + (1 - \gamma)z}{1 - \gamma z}. \tag{2.7}$$

Hence, this method is A-stable [1] *for all $\gamma \geq \frac{1}{2}$ and L-stable* [2] *for $\gamma = 1$.*

[1] A method is called A-stable if the stability region $\mathcal{S} = \{z \in \mathbb{C} : |R()z| \leq 1\}$ contains the left half plane $\mathbb{C}^-$.

[2] A method is called L-stable if the method is A-stable and $R|\infty| = 0$.

Example 2.5. *We consider the 2-stage method*

$$\begin{aligned} w_{n+1} &= w_n + b_1 k_1 + b_2 k_2 \\ k_1 &= \tau F(w_n) + \gamma\tau \mathbf{A} k_1 \\ k_2 &= \tau F(w_n + \alpha_{21} k_1) + \gamma_{21}\tau \mathbf{A} k_1 + \gamma\tau \mathbf{A} k_2, \end{aligned} \tag{2.8}$$

with coefficients

$$b_1 = 1 - b_2, \quad \alpha_{21} = \frac{1}{2b_2} \quad \text{and} \quad \gamma_{21} = -\frac{\gamma}{b_2}.$$

Hence, this method is of order 2 for all γ and as long as $b_2 \neq 0$. The stability function is

$$R(z) = \frac{1 + (1-2\gamma)z + (\gamma^2 - 2\gamma + \frac{1}{2})z^2}{(1-\gamma z)^2}. \tag{2.9}$$

The method is A-stable for $\gamma \geq \frac{1}{4}$ and L-stable if $\gamma = 1 \pm \frac{1}{2}\sqrt{2}$.

2.3 Runge Kutta Chebyshev Methods

The family of Runge Kutta methods discussed in this section are explicit. They will avoid solving algebraic systems, but posses extended real stability interval with a length proportional to s^2, with s the number of stages. Therefore the Runge Kutta Chebyshev (RKC) methods could be attractive to solve moderate stiff systems. The principle goal of constructing Runge Kutta methods is to achieve the highest order consistency with a given number of stages s. The methods discussed in this section use a few stages to achieve a low order consistency and the additional stages are used to increase the stability boundary $\beta(s)$.

Definition 2.6. *The stability boundary $\beta(s)$ is the number $\beta(s)$ such that $[-\beta(s), 0]$ is the largest segment of the negative real axis contained in the stability region*

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

To construct the family of RKC methods we start with the explicit Runge-Kutta methods which have the stability polynomial

$$R(z) = \gamma_0 + \gamma_1 z + \cdots + \gamma_s z^s. \tag{2.10}$$

In order to have first order consistency we take $\gamma_0 = \gamma_1 = 1$.[3] The following theorem states that every explicit Runge-Kutta method has as optimal

[3]This can be verified by considering the test equation $y' = \lambda y$. The local error of the test equation satisfies

$$\frac{e^z - R(z)}{\tau} = \mathcal{O}(\tau^p). \tag{2.11}$$

To achieve p^{th} order consistency the coefficients γ_i has to be chosen in such a way that (2.11) satisfies for p.

stability boundary $\beta(s) = 2s^2$, thus the maximum value of $\beta(s)$ is $2s^2$. The upper boundary of $\beta(s)$ is achieved if we take the shifted Chebyshev polynomials of the first kind as stability polynomial. The following theorem is taken from [13].

Theorem 2.7. *For any explicit, consistent Runge-Kutta method we have $\beta(s) \leq 2s^2$. The optimal stability polynomial is the shifted Chebyshev polynomial of the first kind*

$$P_s(z) = T_s\left(1 + \frac{z}{s^2}\right), \tag{2.12}$$

where the polynomials $T_s(z)$[4] for $z \in \mathbb{C}$ are recursively defined as

$$\begin{aligned} T_0(z) &= 1, \\ T_1(z) &= z, \\ T_j(z) &= 2zT_{j-1}(z) - T_{j-2}(z) \quad 2 \leq j \leq s. \end{aligned} \tag{2.13}$$

If we take (2.12) as stability polynomial, then we achieve the optimum value for $\beta(s)$ equal to $2s^2$.

Proof. Since $P_s(z) = 1 + z + \mathcal{O}(z^2)$ these polynomials give first order consistency and belong to the class of stability polynomials (2.10) that can be generated by explicit Runge Kutta methods. By definition, it follows that $|T_s(x)| \leq 1$ for $-1 \leq x \leq 1$. Therefore, $|P_s(x)| \leq 1$ for $-2s^2 \leq x \leq 0$. As known, on the interval $[-1, 1]$, $T_s(x)$ has $s - 1$ points of tangency with the lines $y = \pm 1$. As a consequence, $P_s(x)$ has also $s - 1$ tangential points with these lines.

This property of the shifted Chebyshev polynomials of the first kind determines it as the unique polynomial with largest stability boundary. Suppose there exists a second stability polynomial of the class (2.10) with $\beta(s) \geq 2s^s$. Since $P_s(x)$ has $s - 1$ points of tangency with the lines $y = \pm 1$, this second stability polynomial then intersects $P_s(x)$ at least $s-1$ times for $x < 0$, where intersection points with common tangent are counted double. The difference polynomial has at least $s - 1$ negative roots, where roots of multiplicity 2 are counted double. Since this second polynomial also belongs to the class (2.10), the difference polynomial is of the form

$$x^2(\tilde{\gamma}_2 + \cdots + \tilde{\gamma}_s x^{s-2}).$$

This difference polynomial can have at most $s-2$ roots on the negative real axis, thus we have a contradiction. □

[4] On the real interval $[-1, 1]$ the Chebyshev polynomials can also be defined as $T_s(x) = \cos(s \arccos x)$.

2.3.1 First Order (Damped) Stability Polynomials and Schemes

In the previous section we proved that a stability polynomial that is generated by a first order explicit R-K method has as optimal stability boundary $2s^2$. The class of shifted Chebyshev polynomials of the first kind seemed to be the class of stability polynomials that achieve the stability boundary $2s^2$. If one considers the stability regions of $P_s(z)$, then one observes that there are interior points $z \in (-\beta(s), 0)$ where $|P_s(z)| = 1$. This means that a small perturbation in imaginary direction near these points might cause instability. To avoid that kind of situations, the polynomials are slightly modified, so called damping.

Following [13], adopting the choice made by Guillo and Lago [9] (already in 1961), the damped form of (2.12) reads

$$P_s(z) = \frac{T_s(\omega_0 + \omega_1 z)}{T_s(\omega_0)}, \tag{2.14}$$

where $\omega_0 = 1 + \frac{\varepsilon}{s^2}$ and

$$\omega_1 = \frac{T_s(\omega_0)}{T_s'(\omega_0)}. \tag{2.15}$$

The polynomials (2.14) satisfy $R_s(z) = 1 + z + \mathcal{O}(z^2)$ and thus generate a first order method.[5] The stability interval is then determined by the relation

$$|P_s(z)| = \left| \frac{T_s(\omega_0 + \omega_1 z)}{T_s(\omega_0)} \right| \leq 1.$$

Using Taylor series of $T_s(z)$ near $z = 0$ we obtain that the stability interval is determined by the relation

$$-\omega_0 \leq \omega_0 + \omega_1 z \leq \omega_0,$$

and thus it follows that

$$\beta(s) = \frac{2\omega_0}{\omega_1} \approx (2 - \frac{4}{3}\varepsilon)s^2. \tag{2.16}$$

In Figure 2.3 the stability region of the first order shifted Chebyshev polynomial (with damping), for $s = 5$, is given. The next problem is to find R-K methods that have stability polynomials like (2.14).

The Approach of Van Der Houwen & Sommeijer

To find explicit R-K methods with stability polynomials (2.14) we use the idea of van der Houwen and Sommeijer, as we will now explain. Their elegant idea was to use the scaled and damped Chebyshev polynomials of

[5] The conditions for first order are $R_s(0) = 1$ and $R_s'(0) = 1$.

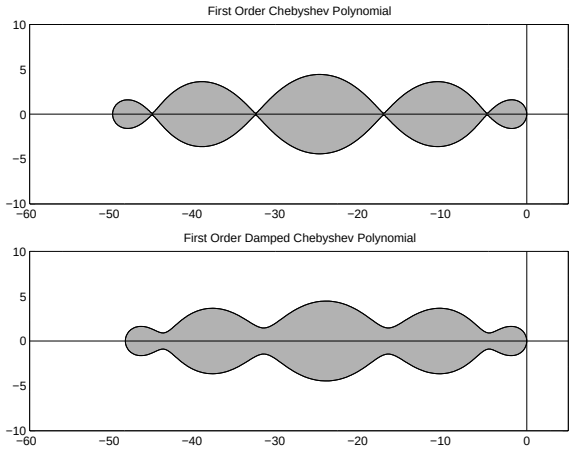

Figure 2.3: Stability region of $P_5(z)$ without damping (upper) and with damping (lower).

order p as stability polynomials to generate an R-K method of order p. To derive the internal stages they use the three term recursion (2.13) and the damped Chebyshev polynomials to define the stability functions of the stability functions of the internal stages. Furthermore they made the ansatz that

$$R_s(z) = a_s + b_s T_s(\omega_0 + \omega_1 z).$$

We construct the first order explicit R-K method with stability polynomial (2.14). Then, ω_0 and ω_1 are already defined, see (2.15). To have a first order method $R_s(z)$ has to satisfy $R_s(0) = 1$ and $R_s'(0) = 1$. It follows that

$$b_s = \frac{1}{T_s(\omega_0)} \quad \text{and} \quad a_s = 1 - b_s T_s(\omega_0).$$

For the internal stages to be of order one, we also put

$$R_j(z) = a_j + b_j T_j(\omega_0 + \omega_1 z) \quad a_j = 1 - b_j T_j(\omega_0), \tag{2.17}$$

with

$$b_j = \frac{1}{T_j(\omega_0)}.$$

Define $R_0(z) = a_0 + b_0 = 1$ and imposing the recursion (2.13) where we use that $R_j(0) = 1$ for all $0 \leq j \leq s$ then shows after some calculations

$$R_0(z) = 1, \quad R_1(z) = 1 + \tilde{\mu}_1 z,$$

$$R_j(z) = (1 - \mu_j - \nu_j) + \mu_j R_{j-1}(z) + \nu_j R_{j-2}(z) + \tilde{\mu}_j R_{j-1}(z) z + \tilde{\gamma}_j z,$$

for $j = 2, \ldots, s$ and where

$$\tilde{\mu}_1 = b_1\omega_1, \quad \mu_j = \frac{2b_j\omega_0}{b_{j-1}}, \quad \nu_j = -\frac{b_j}{b_{j-2}},$$
$$\tilde{\mu}_j = \frac{2b_j\omega_1}{b_{j-1}}, \quad \tilde{\gamma}_j = -a_{j-1}\tilde{\mu}_j. \tag{2.18}$$

From the above relation the RKC integration formula for the nonlinear problem $w'(t) = F(t, w(t))$ can de derived by associating $R_j(z)$ with the intermediate approximation w_{nj} and the occurrence of z with a function evaluation :

$$\begin{aligned} w_{n0} &= w_n, \\ w_{n1} &= w_n + \tilde{\mu}_1\tau F(t_n + c_0\tau, w_{n0}), \\ w_{nj} &= (1 - \mu_j - \nu_j)w_n + \mu_j w_{n,j-1} + \nu_j w_{n,j-2} + \\ &\quad +\tilde{\mu}_1\tau F(t_n + c_{j-1}\tau, w_{n,j-1}) + \tilde{\gamma}_j\tau F(t_n + c_0\tau, w_{n0}), \\ w_{n+1} &= w_{ns}. \end{aligned} \tag{2.19}$$

The above scheme obviously belongs to the class of explicit R-K methods. The only coefficient that remains to be defined are c_j for $1 \leq j < s$. Observe that $R_j(z) = e^{c_j z} + \mathcal{O}(z^2)$ with

$$c_j = \frac{T_s(\omega_0)}{T_s'(\omega_0)}\frac{T_j'(\omega_0)}{T_s(\omega_0)} \approx \frac{j^2}{s^2} \quad c_s = 1.$$

Remark 2.8. *Comparing the s-stage RKC method with a straightforward explicit method like Euler Forward we can conclude the following. The stability boundary of the s-stage RKC method equals $\beta_{\mathrm{RKC}} = 2s^2$, while for the Euler forward method the stability boundary is $\beta_{\mathrm{EF}} = 2$. See Figures 2.3 and 2.4. To have stability for both methods, the time-step τ_{RKC} for the RKC method can be s^2 as large as the time step of the Euler forward method, i.e., $\tau_{\mathrm{RKC}} = s^2\tau_{\mathrm{EF}}$.*

The number of function evaluations for Euler forward is one per time step. For the s-stage RKC method is the number of function evaluations per time step equal to s.

We conclude that to integrate one time step with the s-stage RKC method we need s function evaluations. To integrate the same time step with Euler forward we need to integrate s^2 smaller time steps τ_{EF}, because of the relation $\tau_{\mathrm{RKC}} = s^2\tau_{\mathrm{EF}}$. Per time step Euler forward needs one function evaluation per time step τ_{EF}. Integrating with RKC pays of with a factor s less function evaluations.

2.3.2 Second Order Schemes

If we are looking for second order explicit R-K methods with s internal stages, then we first have to find analytical expressions for second order

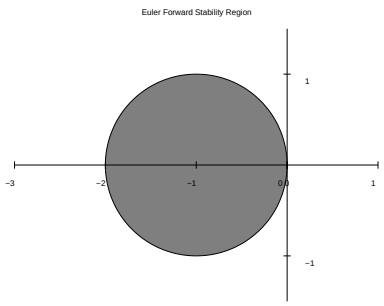

Figure 2.4: Stability region of Euler Forward

stability polynomials

$$R(z) = 1 + z + \frac{1}{2}z^2 + \gamma_3 z^3 + \cdots + \gamma_s z^s.$$

The free coefficients γ_i, $i = 3, \ldots, s$ have to be chosen such that $\beta(s)$ is as large as possible. A suitable approximate polynomial in analytical form was given by Bakker [1]:

$$B_s(z) = \frac{2}{3} + \frac{1}{3s^2} + \left(\frac{1}{3} - \frac{1}{3s^2}\right) T_s\left(1 + \frac{3z}{s^2 - 1}\right), \tag{2.20}$$

with stability boundary $\beta(s) \approx \frac{2}{3}(s^2 - 1)$. This polynomial generates about 80% of the optimal interval. The damped version of (2.20) reads ([12])

$$B_s(z) = 1 + \frac{T_s''(\omega_0)}{(T_s'(\omega_0))^2}\left(T_s(\omega_0 + \omega_1 z) - T_s(\omega_0)\right) \tag{2.21}$$

with

$$\omega_0 = 1 + \frac{\varepsilon}{s^2} \quad \omega_1 = \frac{T_s'(\omega_0)}{T_s''(\omega_0)}.$$

The stability boundary is in the damped case equal to

$$\beta(s) = \frac{2}{3}(s^2 - 1)(1 - \frac{2}{15}\varepsilon).$$

In Figure 2.5 the stability regions $\mathcal{S} = \{|B_5(z)| \leq 1 : z \in \mathcal{C}\}$ are given in the undamped as well as the damped case.

Using the method of van der Houwen and Sommeijer to find a second order explicit R-K method which has stability polynomial (2.21) gives the following method. Again we choose all the internal stages to have second order consistency, thus

$$R_j(z) = 1 + b_j\omega_1 T_j'(\omega_0)z + \frac{1}{2}b_j\omega_1^2 T_j'(\omega_0)z^2 + \mathcal{O}(z^3),$$

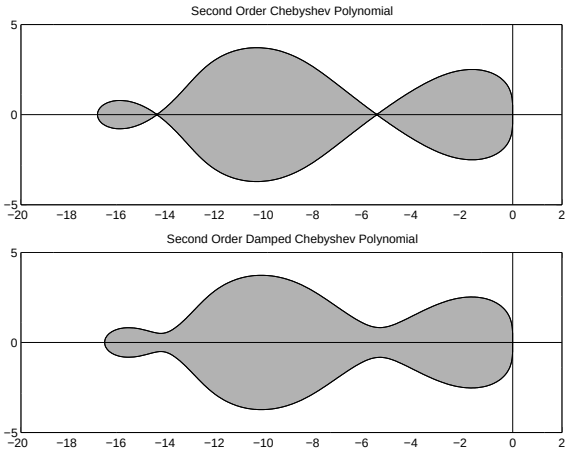

Figure 2.5: Stability region of $B_5(z)$ without damping (upper) and with damping (lower).

has to match with

$$R_j(z) = 1 + c_j z + \frac{1}{2}(c_j z)^2 + \mathcal{O}(z^3).$$

Elementary calculations yields the following method :

$$\begin{aligned} w_{n0} &= w_n, \\ w_{n1} &= w_n + \tilde{\mu}_1 \tau F(t_n + c_0\tau, w_{n0}), \\ w_{nj} &= (1-\mu_j-\nu_j)w_n + \mu_j w_{n,j-1} + \nu_j w_{n,j-2} + \\ &\quad + \tilde{\mu}_1 \tau F(t_n + c_{j-1}\tau, w_{n,j-1}) + \tilde{\gamma}_j \tau F(t_n + c_0\tau, w_{n0}), \\ w_{n+1} &= w_{ns}, \end{aligned} \tag{2.22}$$

with coefficients

$$\omega_0 = 1 + \frac{\varepsilon}{s^2}, \quad \omega_1 = \frac{T_s'(\omega_0)}{T_s''(\omega_0)}, \tag{2.23}$$

$$b_j = \frac{T_j''(\omega_0)}{(T_j'(\omega_0))^2}, \quad c_j = \frac{T_s'(\omega_0)}{T_s''(\omega_0)} \frac{T_j''(\omega_0)}{T_j'(\omega_0)} \approx \frac{j^2-1}{s^2-1}, \tag{2.24}$$

$$\tilde{\mu}_1 = b_1\omega_1, \quad \mu_j = \frac{2b_j\omega_0}{b_{j-1}}, \quad \nu_j = -\frac{b_j}{b_{j-2}}, \tag{2.25}$$

$$\tilde{\mu}_j = \frac{2b_j\omega_1}{b_{j-1}}, \quad \tilde{\gamma}_j = -a_{j-1}\tilde{\mu}_j. \tag{2.26}$$

2.4 Some Remarks on Runge-Kutta Methods

This part of Chapter 2 is reserved to mention some extra properties of Runge-Kutta methods. The properties mainly apply to Implicit RK-methods.

2.4.1 Properties of Implicit Runge-Kutta methods

A-stability

As observed, some Runge-Kutta methods have a stability region equal to the left half plane $\mathbb{C}^-$. For stiff problems such stability regions are favorable.

Definition 2.9 (Dahlquist 1963). *A method with stability region $\mathcal{S}$ such that,*

$$\mathcal{S} \supset \mathbb{C}^- = \{z \mid \text{Re } z \leq 0\},$$

is called A-stable.

Implicit Runge-Kutta methods have stability polynomials of the form

$$R(z) = \frac{P(z)}{Q(z)},$$

with $\deg(P(z)), \deg(Q(z)) \leq s$ Recall that s is the number of stages. The following observation is easily observed by using the *Minimum Modulus Theorem*. An implicit Runge-Kutta method is A-stable if and only if

$$|R(iy)| \leq 1 \quad \text{for all real } y,$$

and

$$R(z) \text{ analitic} \quad \text{for all Re } z \leq 0.$$

Construction of Implicit Runge-Kutta methods

In this part we give basic conditions to derive classes of fully implicit Runge-Kutta methods having good stability properties. The construction of these methods relies on the following assumptions, taken from [11] :

$$\begin{array}{lll}
B(p): & \sum_{i=1}^{s} b_i c_i^{q-1} = \frac{1}{q} & q = 1, \ldots, p, \\
C(\eta): & \sum_{j=1}^{s} \alpha_{ij} c_j^{q-1} = \frac{c_i^q}{q} & i = 1, \ldots, s, \quad q = 1, \ldots, \eta, \\
D(\zeta): & \sum_{i=1}^{s} b_i c_i^{q-1} \alpha_{ij} = \frac{b_j}{q}(1 - c_j^q) & j = 1, \ldots, s, \quad q = 1, \ldots, \zeta.
\end{array}$$

Condition $B(p)$ means that the quadrature formula (b_i, c_i) is of order p. The following fundamental theorem derived by Butcher gives the relation between the conditions stated above and the order of the Runge-Kutta method.

Theorem 2.10 (Butcher 1964). *If the coefficients b_i, c_i, α_{ij} of a Runge-Kutta method satisfy $B(p), C(\eta)$ and $D(\zeta)$ with $p \leq \eta + \zeta + 1$ and $p \leq 2\eta + 2$, then the method is of order p.*

Gauss Methods

Gauss methods are collocation methods[6] based on the Gauss-Legendre quadrature formulas, i.e., $c_1, \ldots, c_s$ are the zeros of the shifted Legendre polynomial of degree s

$$\frac{d^s}{dx^s}\left(x^s(x-1)^s\right).$$

The weights $b_1, \ldots, b_s$ are chosen such that $B(s)$ is satisfied. The following theorem is taken from [11].

Theorem 2.11. *The s-stage Gauss method is of order $2s$. Its stability function is the (s,s)- Padé approximation[7] and the method is A-stable.*

Examples of Gauss methods are given in Figure 2.6.

$$\begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array} \qquad \begin{array}{cc|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & & \frac{1}{2} & \frac{1}{2} \end{array}$$

Figure 2.6: Butcher tableaus of Gauss methods of order 2 (left) and 4 (right)

Radau Methods

Based on the Radau and Lobatto quadrature formulas other Runge-Kutta methods can be constructed. Taking $c_1, \ldots, c_s$ as zeros of respectively

$$\frac{d^{s-1}}{dx^{s-1}}\left(x^s(x-1)^{s-1}\right), \tag{2.27}$$

$$\frac{d^{s-1}}{dx^{s-1}}\left(x^{s-1}(x-1)^s\right), \quad \text{and} \tag{2.28}$$

$$\frac{d^{s-2}}{dx^{s-2}}\left(x^{s-1}(x-1)^{s-1}\right), \tag{2.29}$$

we call the method Radau left (2.27), Radau right (2.28) and Lobatto (2.29). For all three methods the weights $b_1, \ldots, b_s$ are chosen such that $B(s)$ is satisfied.

The s-stage *Radau IA method* is defined by the Radau left method. The coefficients α_{ij}, $i,j = 1, \ldots, s$ are defined by condition $D(s)$. Since all c_j are distinct and the $b_i \neq 0$ this is uniquely possible. Figure 2.7 represents two examples of this method.

The s-stage *Radau IIA method* is defined by the Radau right method and the coefficients α_{ij}, $i,j = 1, \ldots, s$ are obtained by condition $C(s)$. Theorem II.7.7 of [10] implies that this results in the collocation method based on the zeros of (2.28). Examples are given in Figure 2.8. The following theorem is

[6] See Appendix A
[7] See Appendix B

$$\begin{array}{c|c} 0 & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} 0 & \frac{1}{4} & -\frac{1}{4} \\ \frac{2}{3} & \frac{1}{4} & \frac{5}{12} \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$$

Figure 2.7: Butcher tableaus of Radau IA methods of order 1 (left) and 3 (right)

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad \begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array}$$

Figure 2.8: Butcher tableaus of Radau IIA methods of order 1 (left) and 3 (right)

taken from [11] and stated without proof.

Theorem 2.12. *The s-stage Radau IA method and the s stage Radau IIA method are of order* $2s-1$*. Their stability function is the* $(s-1,s)$ *(sub-diagonal) Padé approximation. Also, both methods are A-stable.*

For the Lobatto IIIA methods the coefficients α_{ij} are defined by $C(s)$ and therefore it is a collocation method. Lobatto IIIB $D(s)$ is used to define the α_{ij}. Finally, for the Lobatto IIIC methods we put

$$\alpha_{i1} = b_1 \quad \text{for} \quad i = 1, \ldots, s,$$

and determine the remaining α_{ij} by $C(s-1)$. The following theorem, taken without proof from [11], gives the order and stability properties of Lobatto methods.

Theorem 2.13. *The s-stage Lobatto IIIA, IIIB and IIIC methods are of order* $2s-2$*. The stability function for the Lobatto IIIA and IIIB methods is the diagonal* $(s-1,s-1)$*-Padé approximation. The Lobatto IIIC method has as stability function the* $(s-2,s)$*-Padé approximation. Finally, all the Lobatto methods are A-stable.*

In Table 2.3 we give a summary of these statements. The implicit methods Radau IA and IIA are widely used to solve stiff problems due to their favorable stability properties and simplicity.

2.4.2 Diagonally Implicit Runge-Kutta methods

Fully implicit Runge-Kutta methods like Radau IA and IIA are often used in stiff chemistry problems for their superior stability properties. A clearly disadvantage of these methods is that the Runge-Kutta matrix $\mathcal{A}$ is full. This means that the system of algebraic equations for w_{ni} must be solved simultaneously. In the situation that the number of stages s increases, this

method	assumptions			order	stability function
Gauss	$B(2s)$	$C(s)$	$D(s)$	2s	(s,s)-Padé
Radau IA	$B(2s-1)$	$C(s-1)$	$D(s)$	2s-1	$(s-1,s)$-Padé
Radau IIA	$B(2s-1)$	$C(s)$	$D(s-1)$	2s-1	$(s-1,s)$-Padé
Lobatto IIIA	$B(2s-2)$	$C(s)$	$D(s-2)$	2s-2	$(s-1,s-1)$-Padé
Lobatto IIIB	$B(2s-2)$	$C(s-2)$	$D(s)$	2s-2	$(s-1,s-1)$-Padé
Lobatto IIIA	$B(2s-2)$	$C(s-1)$	$D(s-1)$	2s-2	$(s-2,s)$-Padé

Table 2.3: Implicit Runge-Kutta methods

can become very costly. To avoid these situations, one can take for $\mathcal{A}$ a lower triangular matrix, see Figure 2.9. Then, the s approximations w_{ni} can be solved sub-sequently for $i = 1, \ldots, s$. Such methods we call *Diagonally implicit Runge-Kutta methods*, or shorter DIRK.

$$\begin{array}{c|cccc} c_1 & \alpha_{11} & & & \\ c_2 & \alpha_{21} & \alpha_{22} & & \\ \vdots & \vdots & & \ddots & \\ c_s & \alpha_{s1} & \alpha_{s2} & \cdots & \alpha_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Figure 2.9: Lower Triangular matrix $\mathcal{A}$

To solve each approximation w_{ni}, in general a Newton-type iteration with a coefficient matrix $\mathbf{I} - \tau a_{ii} F'$ will be needed. By taking the diagonal entries of $\mathcal{A}$ equal, one may hope to use repeatedly the stored LU factorization of $\mathbf{I} - \tau a_{ii} F'$. DIRK methods with this additional property are called *Singly Diagonally Implicit RK* methods (SDIRK). The general structure of the Runge-Kutta matrix belonging to the SDIRK is given in Figure 2.10. The

$$\begin{array}{c|cccc} c_1 & \gamma & & & \\ c_2 & \alpha_{21} & \gamma & & \\ \vdots & \vdots & & \ddots & \\ c_s & \alpha_{s1} & \alpha_{s2} & \cdots & \gamma \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

Figure 2.10: Lower Triangular matrix $\mathcal{A}$

special structure of the SDIRK schemes makes it possible to simplify the

order conditions, see Table 2.4, taken from [11].

order	previous conditions	simplified conditions
1	$\sum b_j = 1$	$\sum b_j = 1$
2	$\sum b_j \alpha_{jk} = \frac{1}{2}$	$\sum b_j \alpha_{jk} = \frac{1}{2} - \gamma$
3	$\sum b_j \alpha_{jk} \alpha_{jl} = \frac{1}{3}$	$\sum b_j \alpha_{jk} \alpha_{jl} = \frac{1}{3} - \gamma + \gamma^2$
	$\sum b_j \alpha_{jk} \alpha_{jl} = \frac{1}{6}$	$\sum b_j \alpha_{jk} \alpha_{jl} = \frac{1}{6} - \gamma + \gamma^2$
4	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{4}$	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{4} - \gamma + \frac{3}{2}\gamma^2 - \gamma^3$
	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{8}$	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{8} - \frac{5}{6}\gamma + \frac{3}{2}\gamma^2 - \gamma^3$
	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{12}$	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{12} - \frac{2}{3}\gamma + \frac{3}{2}\gamma^2 - \gamma^3$
	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{24}$	$\sum b_j \alpha_{jk} \alpha_{jl} \alpha_{jm} = \frac{1}{24} - \frac{1}{2}\gamma + \frac{3}{2}\gamma^2 - \gamma^3$

Table 2.4: Order conditions for SDIRK methods (simplified conditions) and the 'original' conditions of RK methods (previous conditions)

The stability function of DIRK methods have the form

$$R(z) = \frac{P(z)}{(1 - \alpha_{11} z)(1 - \alpha_{22} z) \cdots (1 - \alpha_{ss} z)}, \tag{2.30}$$

where the numerator $P(z)$ is of degree $\leq s$. In the case of an SDIRK scheme (2.30) changes into

$$R(z) = \frac{P(z)}{(1 - \gamma z)^s}.$$

For more information with respect to stability properties and order conditions of specific (S)DIRK methods we refer to [11].

The following theorem, also taken from [11], gives *order barriers* for (S)DIRK methods with respect to the vector **b**.

Theorem 2.14. *1. A DIRK method with all b_i positive has order at most* 6,

2. A SDIRK method with all b_i positive has order at most 4.

2.4.3 The Order Reduction Phenomenon for RK-methods

To study the accuracy of RK-methods applied to stiff equations Prothero and Robinson [25] proposed to consider the following problem

$$\begin{cases} y' &= \lambda(y - \varphi(x)) + \varphi'(x) \\ y(x_0) &= \varphi(x_0) \end{cases}, \tag{2.31}$$

with $\lambda \in \mathbb{C}$ and $\mathrm{Re}\,\lambda \leq 0$. Note that the exact solution of (2.31) is given by $y(t) = \varphi(t)$. Applying a general RK-method to (2.31) yields

$$\begin{aligned} w_{ni} &= w_n + h\sum_{j=1}^{s} \alpha_{ij}(\lambda((w_{ni} - \varphi(x_0 + c_j h))) + \varphi'(x_0 + c_j h)), \\ w_{n+1} &= w_n + h\sum_{j=1}^{s} b_j(\lambda((w_{ni} - \varphi(x_0 + c_j h))) + \varphi'(x_0 + c_j h)). \end{aligned} \tag{2.32}$$

We assume that for this RK-method the conditions $B(p)$ and $C(q)$ hold. Then, by replacing w_{ni}, w_n and w_{n+1} by the exact solutions

$$\begin{aligned} w_{ni} &\rightsquigarrow \varphi(x_0 + c_i h), \\ w_n &\rightsquigarrow \varphi(x_0), \\ w_{n+1} &\rightsquigarrow \varphi(x_0 + h), \end{aligned}$$

and using Taylor expansions of $\varphi(x_0 + c_j h)$, $\varphi(x_0 + h)$ and $\varphi'(x_0 + c_j h)$ near x_0 gives

$$\begin{aligned} \varphi(x_0 + c_i h) &= \varphi(x_0) + h\sum_{j=1}^{s} \alpha_{ij}\varphi'(x_0 + c_j h) + \Delta_{i,h}(x_0), \\ \varphi(x_0 + h) &= \varphi(x_0) + h\sum_{j=1}^{s} b_j\varphi'(x_0 + c_j h) + \Delta_{0,h}(x_0), \end{aligned} \tag{2.33}$$

where

$$\Delta_{0,h}(x_0) = \mathcal{O}(h^{p+1}) \quad \text{and} \quad \Delta_{i,h}(x_0) = \mathcal{O}(h^{q+1}). \tag{2.34}$$

Eliminating internal stages and subtracting (2.33) from (2.32) yields for $n = 0$

$$w_1 - \varphi(x_0 + h) = R(z)(w_0 - \varphi(x_0)) + \delta_h(x_0), \tag{2.35}$$

with $R(z)$ the stability function ($R(z) = 1 + z\mathbf{b}^T(\mathbf{I} - z\mathcal{A})^{-1}\mathbf{e}$). Remark that $\delta_h(x_0)$ is the local error when $w_0 = \varphi(x_0)$ in (2.35). It is given by

$$\delta_h(x) = -z\mathbf{b}^T(\mathbf{I} - z\mathcal{A})^{-1}\Delta_h(x) - \Delta_{0,h}(x), \tag{2.36}$$

where $\Delta_h(x) = (\Delta_{1,h}(x), \ldots, \Delta_{s,h}(x))^T$. Replacing x_0 by x_n and so on, one obtains instead of (2.35) the general recursion

$$w_{n+1} - \varphi(x_{n+1}) = R(z)(w_n - \varphi(x_n)) + \delta_h(x_n). \tag{2.37}$$

Applying the above recursion n times gives the following general expression for the *global error*

$$w_{n+1} - \varphi(x_{n+1}) = R(z)^{n+1}(w_0 - \varphi(x_0)) + \sum_{j=0}^{n} R(z)^{n-j}\delta_h(x_j). \tag{2.38}$$

Remark 2.15. *In the non-stiff theory we have $z = \mathcal{O}(h)$. Observe that in the non-stiff case the global error (2.38) behaves like $\mathcal{O}(h^p)$.*

In the stiff case we would like to have a time-step that is larger than $\frac{1}{|\lambda|}$, with $|\lambda| \gg 1$. Therefore, the global error (2.38) is studied under the assumption that simultaneously $h \to 0$ and $z = h\lambda \to \infty$. In Table 2.5 the results for this analysis are given for the s-stage Gauss methods, s-stage Radau methods and s-stage Labotto methods of Section 2.4.1. Comparing Table 2.5 with Table 2.3 we observe that for several methods we have order reduction.

Method		local error	global error
Gauss	s odd	$\mathcal{O}(h^{s+1})$	$\mathcal{O}(h^{s+1})$
	s even		$\mathcal{O}(h^{s})$
Radau IA		$\mathcal{O}(h^{s})$	$\mathcal{O}(h^{s})$
Radau IIA		$z^{-1}\mathcal{O}(h^{s+1})$	$z^{-1}\mathcal{O}(h^{s+1})$
Labotto IIIA	s odd	$z^{-1}\mathcal{O}(h^{s+1})$	$z^{-1}\mathcal{O}(h^{s})$
	s even		$z^{-1}\mathcal{O}(h^{s+1})$
Labotto IIIB	s odd	$z\mathcal{O}(h^{s+1})$	$z\mathcal{O}(h^{s})$
	s even		$z\mathcal{O}(h^{s+1})$
Labotto IIIC		$z^{-1}\mathcal{O}(h^{s})$	$z^{-1}\mathcal{O}(h^{s})$

Table 2.5: Error for (2.31) in the stiff case under the assumption that simultaneously $h \to 0$ and $z = h\lambda \to \infty$.

For the Gauss methods we verify the results obtained in Table 2.5. Since the Runge-Kutta matrix $\mathcal{A}$ is invertible, the vector-matrix product $-z\mathbf{b}^T(\mathbf{I}-z\mathcal{A})^{-1}$ is equal to

$$-z\mathbf{b}^T(\mathbf{I} - z\mathcal{A})^{-1} = \mathbf{b}^T\mathcal{A}^{-1} + \mathcal{O}\left(\frac{1}{z}\right).$$

Observing that for Gauss methods the condition $C(\eta)$ holds for $\eta = s$ yields that $q = s$. Also recall that the construction of Gauss methods implies that $B(p)$ is satisfied for $p = s$. It follows that the local error $\delta_h(x)$ (2.36) is of order $\mathcal{O}(h^{s+1})$.

Denote $e_n = y_n - \varphi(x_n)$. We obtain from the recursion (2.37) and the

fact that $|R(z)| \leq 1$ for all z with Re $z \leq 0$ that

$$\begin{aligned} |e_{n+1}| &\leq |R(z)||e_n| + |\delta_h(x_n)| \\ &= |R(z)||e_n| + Ch^{s+1} \\ &\leq |e_n| + Ch^{s+1} \\ &\vdots \\ &\leq |e_0| + nCh^{s+1} \\ &= |e_0| + Ch^s = \mathcal{O}(h^s). \end{aligned}$$

In the last step we used that $nh = \mathcal{O}(1)$.

Since the stability function of a Gauss method is the (s, s) Padé approximation, we have for odd s $R(\infty) = -1$. For odd s the global error estimate can be improved using the partial summation

$$\sum_{j=0}^{n} \kappa^{n-j}\delta(x_j) = \frac{1-\kappa^{n+1}}{1-\kappa}\delta(x_0) + \sum_{j=1}^{n} \frac{1-\kappa^{n+1-j}}{1-\kappa}(\delta(x_j) - \delta(x_{j-1})).$$

The fact that $(\delta(x_j) - \delta(x_{j-1})) = \mathcal{O}(h^{q+2})$ and the above partial summation gives, when substituted in (2.38), the desired result. For the verification of the results of the other methods we refer to [11].

2.4.4 Order Reduction for Rosenbrock Methods

Applying Rosenbrock methods (2.6) to the Prothero & Robinson test equation (2.31) gives by straightforward calculation that the global error $\varepsilon_n = y_n - y(x_n)$ satisfies

$$\varepsilon_{n+1} = R(z)\varepsilon_n + \delta_h(x_n), \tag{2.39}$$

where $R(z)$ is the stability function. The local error $\delta_h(x_n)$ in (2.39) is given by

$$\delta_h(x) = \varphi(x) - \varphi(x+h) + \mathbf{b}^T(\mathbf{I} - z\mathbf{B})^{-1}\Delta, \tag{2.40}$$

with $\mathbf{B}$ an $s \times s$ matrix with entries $\mathbf{B}_{i,j} = (\alpha_{ij} + \gamma_{ij})_{ij}$. In (2.40) vector $\mathbf{b}$ has as i^{th} entry b_i, $i = 1, \ldots, s$. Finally, the vector Δ is the $s \times 1$ vector with entries

$$\Delta_i = z\left(\varphi(x) - \varphi(x + \alpha_i h) - \gamma_i h\varphi'(x)\right) + h\varphi'(x + \alpha_i h) + \gamma_i h^2\varphi''(x),$$

for $i = 1, \ldots, s$. The following result on order reduction for Rosenbrock methods can be proven.

Theorem 2.16. *The local error $\delta_h(x)$ of a Rosenbrock method applied to the test equation of Prothero & Robinson (2.31) satisfies for $h \to 0$ and $z = h\lambda \to \infty$*

$$\delta_h(x) = \left(\sum_{i,j} b_i\omega_{ij}\alpha_j^2 - 1\right)\frac{h^2}{2}\varphi''(x) + \mathcal{O}(h^3) + \mathcal{O}\left(\frac{h^2}{z}\right), \tag{2.41}$$

where ω_{ij} are the entries of $\mathbf{B}^{-1}$.

Proof. Using the Neumann series expansion

$$(\mathbf{I}-\mathbf{E})^{-1} = \mathbf{I}+\mathbf{E}+\mathbf{E}^2+\cdots$$

we obtain for

$$\begin{aligned}(\mathbf{I}-z\mathbf{B})^{-1} &= \left(\left(\frac{\mathbf{B}^{-1}}{z}-\mathbf{I}\right)(z\mathbf{B})\right)^{-1} = \\ &= -\frac{\mathbf{B}^{-1}}{z}(\mathbf{I}-\frac{\mathbf{B}^{-1}}{z})^{-1} = \\ &= -\frac{\mathbf{B}^{-1}}{z}\left(\mathbf{I}+\frac{\mathbf{B}^{-1}}{z}+\left(\frac{\mathbf{B}^{-1}}{z}\right)^2+\cdots\right) = \\ &= -\frac{\mathbf{B}^{-1}}{z}-\left(\frac{\mathbf{B}^{-1}}{z}\right)^2+\mathcal{O}\left(\frac{1}{z^3}\right).\end{aligned}$$

The product $\mathbf{b}^T(\mathbf{I}-z\mathbf{B})^{-1}\Delta$ can then be written as

$$\mathbf{b}^T(\mathbf{I}-z\mathbf{B})^{-1} = -\frac{1}{z}\mathbf{b}^T\mathbf{B}^{-1}-\frac{1}{z^2}\mathbf{b}^T\mathbf{B}^{-2}+\mathcal{O}\left(\frac{1}{z^3}\right). \tag{2.42}$$

The first term on the right-hand side of (2.42) is with $(\mathbf{B}^{-1})_{ij}=\omega_{ij}$ and using Taylor expansions for $\varphi(x+h)$ equal to

$$\begin{aligned}-\frac{1}{z}\sum_{i,j}b_i\omega_{ij}\Bigg(z\left(-\alpha_j h\varphi'(x)-\frac{(\alpha_j h)^2}{2}\varphi''(x)-\gamma_j h\varphi'(x)+\mathcal{O}(h^3)\right)+ \\ +h\varphi'(x)+\alpha_j h^2\varphi''(x)+\gamma_j h^2\varphi''(x)+\mathcal{O}(h^3)\Bigg).\end{aligned}$$

Rearranging gives

$$\begin{aligned}&\sum_{i,j}b_i\omega_{ij}(\alpha_j+\gamma_j)h\varphi'(x)+\sum_{i,j}b_i\omega_{ij}\frac{\alpha_j^2h^2}{2}\varphi''(x)+\mathcal{O}(h^3)+ \\ &-\frac{1}{z}\sum_{i,j}b_i\omega_{ij}\left(h\varphi'(x)+\alpha_j h^2\varphi''(x)+\gamma_j h^2\varphi''(x)+\mathcal{O}(h^3)\right).\end{aligned}$$

Recall that $(B)_{ij}=\alpha_{ij}+\gamma_{ij}$, $\alpha_j=\sum_{k=1}^{j-1}\alpha_{jk}$ and $\gamma_j=\sum_{k=1}^{j}\gamma_{jk}$. Using these properties one can derive

$$\sum_i b_i\sum_j\omega_{ij}(\alpha_j+\gamma_j)=1.$$[8]

[8] Since $\mathbf{B}_{ij}=\alpha_{ij}+\gamma_{ij}$ it follows with the definition of α_j and γ_j that $\mathbf{B}\mathbb{I}=[\alpha_1+\gamma_1,\ldots,\alpha_n+\gamma_n]^T$, where $\mathbb{I}$ is the vector consisting of ones only. Thus, the summation $\sum_i b_i\sum_j\omega_{ij}(\alpha_j+\gamma_j)$ is equal to $\mathbf{b}^T\mathbf{B}^{-1}\mathbf{B}\mathbb{I}=\mathbf{b}^T\mathbb{I}=1$.

It follows that

$$\begin{aligned}\sum_i b_i \sum_j \omega_{ij}(\alpha_j+\gamma_j)h\varphi'(x) &= h\varphi'(x), \quad \text{and,}\\ -\frac{1}{z}\sum_i b_i \sum_j \omega_{ij}(\alpha_j+\gamma_j)h^2\varphi''(x) &= -\frac{h^2\varphi''(x)}{z}.\end{aligned}$$

Thus, $\mathbf{b}^T(\mathbf{I}-z\mathbf{B})^{-1}\Delta$ is equal to

$$h\varphi'(x)+\sum_{i,j} b_i\omega_{ij}\frac{\alpha_j^2h^2}{2}\varphi''(x)+\mathcal{O}(h^3)-\frac{1}{z}\sum_{i,j} b_i\omega_{ij}h\varphi'(x)+\mathcal{O}\left(\frac{h^2}{z}\right). \quad (2.43)$$

The second term on the right-hand side of (2.42), equal to

$$-\frac{1}{z^2}\mathbf{b}^T\frac{\mathbf{B}^{-2}}{z^2},$$

can be rewritten, almost in the same way as for the first term on the right-hand side of (2.42), as

$$\frac{1}{z}\sum_{i,j} b_i\omega_{ij}h\varphi'(x)+\mathcal{O}\left(\frac{h^2}{z^2}\right). \quad (2.44)$$

The third term on the right-hand side of (2.42) equals $\mathcal{O}\left(\frac{1}{z^2}\right)$. Adding up (2.43), (2.44) and the last term on the right-hand side of (2.42) equal to $\mathcal{O}\left(\frac{1}{z^2}\right)$, and letting $z\to\infty$ gives

$$\begin{aligned}&\mathbf{b}^T(\mathbf{I}-z\mathbf{B})^{-1}\Delta =\\ &\quad h\varphi'(x)+\sum_{i,j} b_i\omega_{ij}\frac{\alpha_j^2h^2}{2}\varphi''(x)+\mathcal{O}(h^3)+\mathcal{O}\left(\frac{h^2}{z}\right).\end{aligned}$$

Applying Taylor expansion of $\varphi(x+h)$ near x in (2.40) gives the desired result. □

Remark 2.17. *In the stiff case a Rosenbrock is only third order when*

$$\sum_{i,j} b_i\omega_{ij}\alpha_j^2 = 1,$$

is satisfied. Since none of the Rosenbrock methods defined in Section 2.2 satisfies this extra constraint, their order of convergence is only two for the Prothero & Robinson test equation.

To satisfy this extra constraint, a convenient way is to require

$$\alpha_{si}+\gamma_{si} = b_i, \quad i=1,\dots,s, \quad \text{and} \quad \alpha_s = 1. \quad (2.45)$$

This extra requirement implies even

$$\delta_h(x) = \mathcal{O}\left(\frac{h^2}{z}\right).$$

In that case the method yields asymptotically exact results for $z\to\infty$.

Chapter 3

Time Splitting

In general it is inefficient (or even infeasible) to apply the same integration formula to different parts of the advection diffusion reaction system in higher dimensions,

$$u_t + \nabla \cdot (\mathbf{a}u) = \nabla \cdot (\mathbf{D}\nabla u) + f(u).$$

If, for example, the discretization of the advection (and diffusion) results in a stiff system, then it calls for an implicit method to solve that part of the equation. If on the other hand the reaction terms are not stiff, then explicit methods are often more suitable for that part of the equation. Or, it could also be the other way around. If we have a stiff reaction term, then we would like to use implicit methods. If we used for spatial discretization limiters, then explicit methods are often more suitable.

Solving the system using a simple implicit integration rule results into a large system of nonlinear algebraic equations. Due to simultaneous coupling over species and space the nonlinear system becomes too large to handle. In such cases we want to have an appropriate form of splitting. The general idea behind splitting is to split a complicated system into smaller parts, which can be solved efficiently with suitable integration formulas, for the sake of time stepping.

This chapter is organized as follows. We start with the explanation of the concept of (first and second order) time splitting. The chapter is concluded with remarks on the treatment of boundary conditions and stiffness.

3.1 Operator Splitting

The technique in this section is called *Operator Splitting* or *Time Splitting*. In this section we threat on first order splitting. We focus more on concepts rather than on actual methods.

3.1.1 First Order Splitting of Linear ODE Problems

Consider the *linear* homogeneous ODE system

$$w'(t) = \mathbf{A}w(t), \quad t > 0 \quad \text{and} \quad w(0) = w_0. \tag{3.1}$$

This system can be seen as a semi-discretization of a linear PDE. Assume that we have for $\mathbf{A}$ a two term splitting, e.g.,

$$\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2.$$

The exact solution of (3.1) is given by

$$w(t_{n+1}) = e^{\tau \mathbf{A}} w(t_n),$$

where $\tau = t_{n+1} - t_n$. Since $\mathbf{A}$ has a splitting one can use it to approximate a solution of (3.1). The exact solution of (3.1) is then approximated by

$$w_{n+1} = e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1} w_n, \tag{3.2}$$

where w_n is an approximation of $w(t_n)$ and $\tau = t_{n+1} - t_n$. We have found the simplest splitting, in which the two subproblems

$$\begin{array}{rclcl} \frac{d}{dt}w^*(t) & = & \mathbf{A}_1 w^*(t) & \text{for } t_n < t \le t_{n+1} & \text{with} \quad w^*(t_n) = w_n, \\ \frac{d}{dt}w^{**}(t) & = & \mathbf{A}_2 w^{**}(t) & \text{for } t_n < t \le t_{n+1} & \text{with} \quad w^{**}(t_n) = w^*(t_{n+1}), \end{array}$$

have to be solved one after each other starting from w_n. To complete the integration after one step take $w_{n+1} = w^{**}(t_{n+1})$.

The error introduced by splitting, called the *splitting error*, can be found by inserting (3.2) into the exact solution. This gives

$$w(t_{n+1}) = e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1} w(t_n) + \tau \rho_n,$$

with ρ_n the local truncation error. The fact that $\tau\rho_n$ is the error introduced per step starting from the exact solution it is the local splitting error. Since

$$\begin{aligned} e^{\tau \mathbf{A}} &= \mathbf{I} + \tau(\mathbf{A}_1 + \mathbf{A}_2) + \frac{1}{2}\tau^2(\mathbf{A}_1 + \mathbf{A}_2)^2 + \mathcal{O}(\tau^3), \\ e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1} &= \mathbf{I} + \tau(\mathbf{A}_1 + \mathbf{A}_2) + \frac{1}{2}\tau^2(\mathbf{A}_1^2 + 2\mathbf{A}_1\mathbf{A}_2 + \mathbf{A}_2^2) + \mathcal{O}(\tau^3), \end{aligned}$$

ρ_n satisfies

$$\rho_n = \frac{1}{\tau}\left(e^{\tau \mathbf{A}} - e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1}\right) w(t_n) = \frac{1}{2}\tau[\mathbf{A}_1, \mathbf{A}_2]w(t_n) + \mathcal{O}(\tau^2).$$

The commutator of $\mathbf{A}_1$ and $\mathbf{A}_2$ is defined as

$$[\mathbf{A}_1, \mathbf{A}_2] = \mathbf{A}_1\mathbf{A}_2 - \mathbf{A}_2\mathbf{A}_1.$$

It is obvious that the splitting defined in (3.2) has order one, unless $\mathbf{A}_1$ and $\mathbf{A}_2$ commute.

As can be found in [13], splitting appears to be stable provided that the sub steps themselves are stable.

3.1.2 Nonlinear ODE problems

For general nonlinear ODE problems

$$w'(t) = F(t, w(t)), \quad t > 0 \quad \text{and} \quad w(0) = w_0. \tag{3.3}$$

with a two term splitting

$$F(t, w) = F_1(t, w) + F_2(t, w),$$

the first order linear splitting (3.2) is

$$\begin{array}{rclcrcl} \frac{d}{dt}w^*(t) & = & F_1(t, w^*(t)) & \text{for } t_n < t \le t_{n+1} & \text{with} & w^*(t_n) & = & w_n, \\ \frac{d}{dt}w^{**}(t) & = & F_2(t, w^{**}(t)) & \text{for } t_n < t \le t_{n+1} & \text{with} & w^{**}(t_n) & = & w^*(t_{n+1}), \end{array}$$

giving $w_{n+1} = w^{**}(t_{n+1})$ as the next approximation.

The nonlinear counterpart of the splitting error is again of order one. As seen in [13], this error can be derived by Taylor expansions of $w^*(t_{n+1})$ and $w^{**}(t_{n+1})$ around $t = t_n$. One then obtains that the local splitting error ρ_n is equal to

$$\rho_n = \frac{1}{2}\tau \left[\frac{\partial F_1}{\partial w} F_2 - \frac{\partial F_2}{\partial w} F_1 \right] (t_n, w(t_n)) + \mathcal{O}(\tau^2).$$

If the bracketed term equals zero the splitting error is of order two.

3.1.3 Second and Higher Order Splitting.

In this section we treat second and higher order splitting methods for linear and nonlinear problems. To get a second order splitting error the idea of symmetry in splitting is proposed by Strang.

Linear ODE Problems

Consider the linear ODE system (3.1) and assume for $\mathbf{A}$ the two-term splitting $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$. The exact solution of (3.1) can be approximated by (3.2). Interchanging the order of $\mathbf{A}_1$ and $\mathbf{A}_2$ after each half time step will lead to symmetry and to better accuracy. The solution of (3.1) is then approximated by

$$w_{n+1} = \left(e^{\frac{1}{2}\tau \mathbf{A}_1} e^{\frac{1}{2}\tau \mathbf{A}_2} \right) \left(e^{\frac{1}{2}\tau \mathbf{A}_2} e^{\frac{1}{2}\tau \mathbf{A}_1} \right) w_n = \left(e^{\frac{1}{2}\tau \mathbf{A}_1} e^{\tau \mathbf{A}_2} e^{\frac{1}{2}\tau \mathbf{A}_1} \right) w_n. \tag{3.4}$$

The above splitting is proposed by Strang [33] and is therefore called *Strang Splitting*.

The splitting error of Strang splitting has a formal consistency of order two. By series expansion the local truncation error, or splitting error, is found as

$$\rho_n = \frac{1}{24}\tau^2 \left([\mathbf{A}_1, [\mathbf{A}_1, \mathbf{A}_2]] + 2[\mathbf{A}_2, [\mathbf{A}_1, \mathbf{A}_2]] \right) w(t_{n+\frac{1}{2}}) + \mathcal{O}(\tau^4). \tag{3.5}$$

Another second order symmetrical splitting of Strang is

$$w_{n+1} = \frac{1}{2}\left(e^{\tau \mathbf{A}_1} e^{\tau \mathbf{A}_2} + e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1}\right) w_n, \tag{3.6}$$

with splitting error

$$\rho_n = -\frac{1}{12}\tau^2 \left([\mathbf{A}_1, [\mathbf{A}_1, \mathbf{A}_2]] + [\mathbf{A}_2, [\mathbf{A}_2, \mathbf{A}_1]]\right) w_(t_n) + \mathcal{O}(\tau^3).$$

The splitting (3.6) is more expensive than (3.4)[1], but has as advantage that the factors $e^{\tau \mathbf{A}_1} e^{\tau \mathbf{A}_2}$ and $e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1}$ can be computed parallel.

In the case of a multi-component splitting of $\mathbf{A}$, e.g.,

$$\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2 + \mathbf{A}_3,$$

the symmetrical Strang splitting method (3.4) is just a repeated application of itself. Hence, for the three-term splitting we obtain

$$w_{n+1} = e^{\frac{1}{2}\tau \mathbf{A}_1} e^{\frac{1}{2}\tau \mathbf{A}_2} e^{\tau \mathbf{A}_3} e^{\frac{1}{2}\tau \mathbf{A}_2} e^{\frac{1}{2}\tau \mathbf{A}_1} w_n, \tag{3.7}$$

which has also a second order splitting error. In the same way the alternative Strang splitting (3.6) can be generalized.

Nonlinear ODE Problems

The extension of the Strang splitting (3.4) is straightforward,

$$\frac{d}{dt} w^*(t) = F_1(t, w^*(t)) \quad \text{for } t_n < t \le t_{n+\frac{1}{2}}$$
$$\text{with} \qquad w^*(t_n) = w_n,$$

$$\frac{d}{dt} w^{**}(t) = F_2(t, w^{**}(t)) \quad \text{for } t_n < t \le t_{n+1}$$
$$\text{with} \qquad w^{**}(t_n) = w^*(t_{n+\frac{1}{2}}),$$

$$\frac{d}{dt} w^{***}(t) = F_1(t, w^{***}(t)) \quad \text{for } t_{n+\frac{1}{2}} < t \le t_{n+1}$$
$$\text{with} \qquad w^{***}(t_{n+\frac{1}{2}}) = w^{**}_{n+1},$$

giving $w_{n+1} = w^{***}(t_{n+1})$ as the next approximation. Again we will have formal consistency of order two. This result can be obtained using Taylor expansion under the condition that the ODE is autonomous. Referring to [13] we state that for autonomous problems we also have a second order splitting error.

[1] This follows by writing (3.6) out as

$$w_n = \left(\frac{1}{2}\right)^n \left(e^{\tau \mathbf{A}_1} e^{\tau \mathbf{A}_2} + e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1}\right) \cdots \left(e^{\tau \mathbf{A}_1} e^{\tau \mathbf{A}_2} + e^{\tau \mathbf{A}_2} e^{\tau \mathbf{A}_1}\right) w_0.$$

Higher-Order Splittings

Examples of higher-order splittings are the fourth order splitting

$$w_{n+1} = \left(\frac{4}{3}(S_{\frac{1}{2}\tau})^2 - \frac{1}{3}S_\tau\right) w_n, \tag{3.8}$$

where $S_\tau = e^{\frac{1}{2}\tau\mathbf{A}_1}e^{\tau\mathbf{A}_2}e^{\frac{1}{2}\tau\mathbf{A}_1}$ is the second order Strang splitting operator. Since the above splitting (3.8) has negative a negative weight, it is not known for what kind of problems this scheme will be stable or not.

Another fourth order splitting scheme derived by Yoshida [39] and Suzuki [34] reads

$$w_{n+1} = S_{\theta\tau}S_{(1-2\theta)\tau}S_{\theta\tau}w_n, \tag{3.9}$$

with $\theta = (2-\sqrt[3]{2})^{-1} \approx 1.35$. Observe that $2-\theta < 0$, which means that a time step with negative time has to be taken. As mentioned in [13], reversing the time for diffusion or (stiff) reaction terms this will lead to ill-posedness. Higher order splittings can be used for conservative problems where boundary conditions are not relevant, such as the Schrödinger equation, see [13].

3.2 Boundary Values, Stiff Terms and Steady State

3.2.1 Boundary Values

For PDE problems, where the boundary conditions are important, difficulties with splitting may occur. The boundary conditions for the PDE problem have a physical interpretation, while boundary conditions for the sub steps are missing. These sub steps have sometimes a physical interpretation. Therefore one may have to reconstruct boundary conditions for the specific splitting under consideration.

General analysis of boundary conditions for splitting methods is, at present, still lacking. The rule of thumb we will use is that the treatment of the boundaries should coincide as much as possible with the scheme in the interior of the domain.

3.2.2 Stiff Terms

Assume we have the ODE problem $w'(t) = \mathbf{A}w(t)$, where $\mathbf{A}$ has a two-term splitting $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$. Assume also that $\|\mathbf{A}_1\|$ is bounded and $\mathbf{A}_2$ has an eigenvalue equal to $-\frac{1}{\epsilon}$, $1 \gg \epsilon > 0$. Verwer et. al. [35, 32] showed that the first order splitting with $\mathbf{A}$ as above, see Section 3.1, will retain its first order accuracy. In [35, 32] also is shown that the second order Strang splitting will in general give only an order one accuracy.

3.2.3 Steady State

In the case we have to solve an advection-diffusion-reaction problem with stiff reaction terms, the following behavior usually occurs. First there will be a short time interval where concentrations rapidly change, the so called transient phase. After the transient phase, there will be a steady state, e.g. the reactions between the species are in balance.

If one solves an advection-diffusion-reaction system with time or operator splitting, then the steady states are not returned exactly. This can easily seen in the following linear ODE case. Suppose the advection-diffusion-reaction system has been discretized in space. We then have the semi discrete system $w'(t) = \mathbf{A}w(t)$, which we solve with time splitting, thus $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$. In the steady state the concentrations are in balance, thus $w'(t) = 0$. This means that in the steady state yields $\mathbf{A}_1 + \mathbf{A}_2 = 0$. Since the two term splitting introduces an error of at least order one, this error will also be returned in the steady state. Hence, this is in general the case for all splitting methods.

Other time integration methods like Runge-Kutta methods, Runge-Kutta-Chebyshev methods, etc. return steady states exactly.

Chapter 4

IMEX Methods

Many advection-diffusion-reaction equations have a natural splitting into a non-stiff or moderate stiff part and a stiff part. To solve the non-stiff or moderate part explicit solvers are suitable. To solve the stiff part one would use an implicit method. To split the non-stiff and stiff part time splitting is an option, but has disadvantages like splitting errors, boundary conditions, etc. Another disadvantage could be that the subproblem cannot be solved by a multi-step method, i.e. it needs information from previous time steps.

In this part we consider IMEX methods, which are methods that are a suitable mix of implicit and explicit methods. For instance, there exist IMEX multi-step and IMEX Runge-Kutta methods. The main idea of an IMEX method will be sketched in the next section, where we treat the IMEX- θ method.

4.1 IMEX- θ Method

Suppose we have the nonlinear system or semi-discretization

$$w'(t) = F(t, w(t)),$$

where $F(t, w(t))$ has the natural splitting

$$F(t, w(t)) = F_0(t, w(t)) + F_1(t, w(t)),$$

with F_0 is a non-stiff and F_1 stiff. In advection-diffusion-reaction systems the non-stiff term is for instance advection and the stiff terms the discretized diffusion and reactions. The non-stiff term is suitable for explicit time integration while the stiff terms are more suitable for implicit integration methods.

An example of a method that combines explicit as well as implicit treatment of respectively the non-stiff term $F_0(t, w(t))$ and stiff term $F_1(t, w(t))$

is the following one:

$$w_{n+1} = w_n + \tau \left(F_0(t_n, w_n) + (1-\theta) F_1(t_n, w_n) + \theta F_1(t_{n+1}, w_{n+1}) \right), \quad (4.1)$$

where the parameter $\theta \geq \frac{1}{2}$. This method is a combination of the Euler Forward method, which is explicit, and the implicit θ -method. Methods that are mixtures of *IM*plicit and *EX*plicit methods are called *IMEX* methods. The method given in (4.1) is called the IMEX- θ Method.

Truncation error

The truncation error of the IMEX- θ method can be derived by inserting the exact solution of $w'(t) = F(t, w(t))$ into (4.1). We then obtain

$$\begin{aligned} w(t_{n+1}) &= w(t_n) + \tau \left(F_0(t_n, w(t_n)) + (1-\theta) F_1(t_n, w_n) + \right. \\ &\quad \left. \theta F_1(t_{n+1}, w(t_{n+1})) \right) + \tau p_n, \end{aligned} \quad (4.2)$$

where p_n is the truncation error. Using Taylor series of $w(t_{n+1})$, $F_1(t_{n+1}, w(t_{n+1}))$ and $F_0(t_{n+1}, w(t_{n+1}))$ near t_n, we obtain

$$p_n = \left(\frac{1}{2} - \theta \right) \tau w''(t_n) + \theta \tau F_0'(t_n, w(t_n)) + \mathcal{O}(\tau^2).$$

Stability

Consider the test equation

$$w'(t) = \lambda_0 w(t) + \lambda_1 w(t),$$

and let $z_j = \tau \lambda_j$ for $j = 0, 1$. Applying the IMEX- θ method to this test equation gives

$$R(z_0, z_1) = \frac{1 + z_0 + (1-\theta) z_1}{1 - \theta z_1}. \quad (4.3)$$

Stability of the IMEX- θ method thus requires

$$|R(z_0, z_1)| \leq 1. \quad (4.4)$$

To analyze the stability region (4.4) we have two starting points:

1. Assume the implicit part of the method to be stable, in fact A-stable, and investigate the stability region of the explicit part,

2. Assume the explicit part of the method to be stable and investigate the stability region of the implicit part.

Starting with the first point, we assume the implicit part of the IMEX-θ method to be A-stable. Define the set

$$\mathcal{D}_0 = \{z_0 \in \mathbb{C} : \text{the IMEX scheme is stable } \forall z_1 \in \mathbb{C}^-\},$$

where $\mathbb{C}^-$ is the set

$$\mathbb{C}^- = \{z \in \mathbb{C} : \text{Re } z \leq 0\}.$$

The question is whether the set $\mathcal{D}_0$ is smaller, larger or equally shaped in comparison with the stability region of Euler Forward. Using the maximum modulus theorem and the assumption that the implicit part of the IMEX- θ method gives that z_1 can be replaced by $z_1 = i \cdot r, r \in \mathbb{R}$. Substituting (4.3) into (4.4) gives after some elementary calculations that $z_0 = x_0 + iy_0 \in \mathcal{D}_0$ if and only if for all $r \in \mathbb{R}$ yields

$$(2\theta - 1)r^2 + 2(\theta - 1)y_0 r - (2x_0 + x_0^2 + y_0^2) \geq 0.$$

In the above inequality the left-hand side is a cubic function of r. For $\theta > \frac{1}{2}$ this function is larger or equal than zero when the discriminant is less or equal than zero. We then obtain that $z_0 = x_0 + iy_0 \in \mathcal{D}_0$ iff

$$\theta^2 y_0^2 + (2\theta - 1)(1 + x_0)^2 \leq 2\theta - 1.$$

For $\theta = \frac{1}{2}$ the stability region of the IMEX method reduces to the line segment $[-2, 0]$, while for $\theta = 1$ the stability region of Euler Forward is obtained. See Figure 4.1.

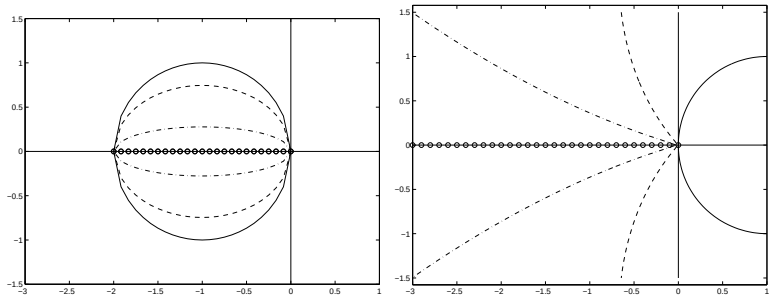

Figure 4.1: Boundary of regions $\mathcal{D}_0$ (left) and $\mathcal{D}_1$ for $\theta = 0.5$ (circles), 0.51 (-·-), 0.6 (- -) and 1 (solid)

The alternative is to assume that the explicit part of the IMEX method is stable, which means that z_0 is an element of the set $\mathcal{S}_0 = \{z_0 : |1 + z_0| \leq 1\}$. The question now is, what is the set

$$\mathcal{D}_1 = \{z_1 \in \mathbb{C} : \text{the IMEX scheme is stable } \forall z_0 \in \mathcal{S}_0\}.$$

Some elementary calculations yield that $z_1 \in \mathcal{D}_1$ iff

$$1 + |(1 - \theta)z_1| \leq |1 - \theta z_1|,$$

is smaller, larger or equally shaped in comparison with the stability region of Euler Backward. The boundary of the stability regions are for several values of θ plotted in Figure 4.1. Hence, for $\theta = \frac{1}{2}$ we obtain as stability region for the IMEX method the negative real axis. For $\theta = 1$ we obtain as stability region the stability region of Euler Backward.

We remark that for $\theta = 1$ the IMEX method has favorable stability properties. It could be seen as a form of time splitting where we first solve the explicit part with Euler Forward and the implicit part with Euler Backward. However, using this IMEX method we do not have errors as consequence of

- intermediate results that are inconsistent with the full equation,
- intermediate boundary conditions to solve these intermediate results.

If one uses this IMEX- θ method with $\theta = \frac{1}{2}$, then one has to pay a little more attention to stability. If, for instance, one has a system with complex eigenvalues, then the IMEX method will not be stable for $\theta = \frac{1}{2}$.

Steady State

If we are in steady state, which means that $F(w) = 0$, F is independent of t, then the splitting of $F(t, w(t))$ becomes in steady state $F(w) = F_0(w) + F_1(w)$, where F, F_0 and F_1 are independent of t.

If this steady state w is a stationary point of the IMEX-θ scheme, then this means that the IMEX-θ scheme returns steady states exactly. Inserting the steady state w into (4.1) gives

$$w = w + \tau \left(F_0(w) + (1-\theta)F_1(w) + \theta F_1(w)\right) = w.$$

The conclusion is that steady states are returned exactly by the IMEX-θ scheme.

4.2 Concluding Remarks

The general idea of IMEX methods is explained in Section 4.1 using the IMEX- θ method. This idea can also be used in Multi-step methods, Runge-Kutta methods and so on. The IMEX- θ method is also the simplest example of an IMEX R-K method. In the next example we give a more sophisticated example of an IMEX R-K method.

Example 4.1. *We combine the explicit and implicit trapezoidal rule in the following way:*

$$\begin{aligned} w^*_{n+1} &= w_n + \tau F_0(t_n, w_n) + \frac{1}{2}\tau F_1(t_n, w_n) + \frac{1}{2}\tau F_1(t_{n+1}, w^*_{n+1}), \\ w_{n+1} &= w_n + \frac{1}{2}\tau F(t_n, w_n) + \frac{1}{2}\tau F(t_{n+1}, w^*_{n+1}). \end{aligned}$$

This method has second order consistency, which can be verified by simple calculations. Applying this IMEX trapezoidal rule to the test equation $y' = (\lambda_0 + \lambda_1)y$ *we get*

$$R(z_0, z_1) = 1 + \frac{1 + \frac{1}{2}z_0}{1 - \frac{1}{2}z_1}(z_0 + z_1),$$

as stability function. If we take the limit for $z_1 \to -\infty$[1]*, then we have as stability criterion*

$$|R(z)| = |1 + z_0| \leq 1.$$

Hence, this IMEX trapezoidal rule is not suitable for advection-diffusion-reaction problems if advection is treated explicitly and discretized with higher order differences. For more detailed information we refer to [13].

Another example of an IMEX R-K method is the subject of the next chapter, titled IMEX Runge-Kutta Chebyshev methods. Since this class of IMEX R-K methods recently is developed for solving advection-diffusion-reaction equations, see [36], we devote the next chapter to it.

[1] The implicit part is unconditionally stable.

Chapter 5

IMEX Runge-Kutta-Chebyshev Methods

In this chapter the IMEX extension of Runge-Kutta-Chebyshev methods are introduced. The IMEX extension proposed in [36, 37] is an RKC method that also deals with highly stiff (reaction) terms. The highly stiff terms are treated implicitly and the moderate stiff terms, for instance coming from diffusion, are treated in an explicit way.

As starting point we have the Runge Kutta Chebyshev methods defined in Section 2.3. Recall that the general form of a RKC method is

$$\begin{aligned} W_0 &= w_n, \\ W_1 &= w_n + \tilde{\mu}_1 \tau F(t_n + c_0\tau, W_0), \\ W_j &= (1-\mu_j-\nu_j)w_n + \mu_j W_{j-1} + \nu_j W_{j-2} + \\ &\quad + \tilde{\mu}_1 \tau F(t_n + c_{j-1}\tau, W_{j-1}) + \tilde{\gamma}_j \tau F(t_n + c_0\tau, W_0), \\ w_{n+1} &= W_{ns}. \end{aligned} \tag{5.1}$$

Suppose we have a linear system

$$w'(t) = F(t, w(t)), \tag{5.2}$$

where $F(t, w(t))$ can be split as

$$F(t, w(t)) = F_E(t, w(t)) + F_I(t, w(t)).$$

The term $F_I(t, w(t))$ is supposed to be too stiff to be treated efficiently by a Runge-Kutta-Chebyshev method (5.1). The term $F_E(t, w(t))$ is the moderate stiff term which can still be treated explicitly by (5.1).

5.1 Construction of the IMEX scheme

The first stage of (5.1) becomes in the IMEX-RKC scheme

$$W_1 = w_n + \tilde{\mu}_1 \tau F_E(t_n + c_0\tau, W_0) + \tilde{\mu}_1 \tau F_I(t_n + c_1\tau, W_1), \tag{5.3}$$

with $\tilde{\mu}_1 = b_1\omega_1$. Note that the highly stiff term is treated implicitly. The other $s-1$ subsequent stages of (5.1) will be modified in a similar way.

The stability function of the first stage of (5.1) is equal to

$$R_1(z_0, z_1) = \frac{1 + b_1\omega_1 z_0}{1 - b_1\omega_1 z_1}, \tag{5.4}$$

when (5.3) is applied to the test equation $w'(t) = \lambda_0 w(t) + \lambda_1 w(t)$ with $z_j = \lambda_j \tau$. Following [36], we impose

$$b_1 = \frac{1}{\omega_0},$$

so that

$$R_1(z_0, z_1) = \frac{1 + b_1 \frac{\omega_1}{\omega_0} z_0}{1 - \frac{\omega_1}{\omega_0} z_1}. \tag{5.5}$$

Observe that the choice for b_1 is different than in Section 2.3. This choice for b_1 enables the following ansatz

Ansatz 5.1. *All stability functions $R_j(z_0, z_1)$ from the internal stages j, $j = 1, \ldots, s$, of the IMEX-RKC scheme are taken of the form*

$$R_j(z_0, z_1) = a_j + b_j T_j\left(\frac{\omega_0 + \omega_1 z_0}{1 - \frac{\omega_1}{\omega_0} z_1}\right), \quad a_j = 1 - b_j T_j(\omega_0). \tag{5.6}$$

The coefficients $b_1 = \frac{1}{\omega_0}$, $b_0 = b_2$ and b_j for $j \geq 2$ is taken as

$$b_j = \frac{T_j''(\omega_0)}{(T_j'(\omega_0))^2}.$$

These stability functions of the IMEX-RKC scheme are an extension of the stability functions of the RKC scheme. Taking $z_1 = 0$ in (5.6) we obtain the internal stability functions (2.17).

The construction of the IMEX scheme is based on the following. Rewrite (5.6) into

$$T_j(x) = \frac{-a_j}{b_j} + \frac{R_j(z_0, z_1)}{b_j}, \quad x = \frac{\omega_0 + \omega_1 z_0}{1 - \frac{\omega_1}{\omega_0} z_1},$$

and apply the recursion (2.13). Inserting x gives

$$\begin{aligned}(1-\frac{\omega_1}{\omega_0})R_j &= a_j(1-\frac{\omega_1}{\omega_0})+2\frac{b_j}{b_{j-1}}R_{j-1}\cdot(\omega_0+\omega_1 z_0)\\ &\quad -2\frac{b_j}{b_{j-1}}a_{j-1}(\omega_0+\omega_1 z_0)+\frac{b_j}{b_{j-1}}a_{j-2}(1-\frac{\omega_1}{\omega_0}z_1)\\ &\quad -\frac{b_j}{b_{j-2}}R_{j-2}\cdot(1-\frac{\omega_1}{\omega_0}z_1).\end{aligned}$$

Identifying R_j with W_j and $R_j z_1$ with $\tau F_I(t_n+c_j\tau, W_j)$ and so on we obtain

$$\begin{aligned}W_j-\tilde{\mu}_1\tau F_{I,j} &= (a_j-\mu_j a_{j-1}-\nu_j a_{j-2})W_0+\mu_j W_{j-1}+\nu_j W_{j-2}+\\ &\quad \tilde{\mu}_j\tau F_{E,j-1}+\tilde{\gamma}_j\tau F_{E,0}-\nu_j\tilde{\mu}_1\tau F_{I,j-2}-\tilde{\mu}_1(a_j-\nu_j a_{j-2})\tau F_{I,0},\end{aligned}$$

where $F_{I,j}=F_I(t_n+c_j\tau,W_j)$ and so on. The coefficients $\tilde{\mu}_j,\tilde{\gamma}_j,\ldots$ are defined as in Section 2.3. Using $a_j-\mu_j a_{j-1}-\nu_j a_{j-2}=1-\nu_j-\mu_j$ we get

$$\begin{aligned}W_0 &= w_n,\\ W_1 &= w_n+\tilde{\mu}_1\tau F_{E,0}+\tilde{\mu}_1\tau F_{I,1},\\ W_j &= (1-\mu_j-\nu_j)w_n+\mu_j W_{j-1}+\nu_j W_{j-2}+\\ &\quad +\tilde{\mu}_j\tau F_{E,j-1}+\tilde{\gamma}_j\tau F_{E,0},\\ &\quad [\tilde{\gamma}_j-(1-\mu_j-\nu_j)\tilde{\mu}_1]\tau F_{I,0}-\nu_j\tilde{\mu}_1\tau F_{I,j-2}+\tilde{\mu}_1 F_{I,j}\\ w_{n+1} &= W_s.\end{aligned}\tag{5.7}$$

Remark 5.2. *As already remarked before, the scheme (5.7) is an IMEX extension of (5.1) in a natural way.*

Using easy calculations there can be obtained that also this IMEX extension of RKC returns the steady state exactly.

Remark 5.3. *In each stage the following system of nonlinear algebraic equations has to be solved*

$$W_j-\tilde{\mu}_1\tau F_I(t_n+c_j\tau,W_j)=V_j,$$

with V_j known and W_j unknown. In the case that the implicit part of $F(t,w(t))$ consists of the 'stiff' reaction only, then F_I has no underlying spatial grid connectivity.

5.2 Stability

Consider the test-equation

$$w'(t)=\lambda_E w(t)+\lambda_I w(t),$$

and assume that both eigenvalues are real and non-positive. For many practical cases this imposes no restriction.

Applying the IMEX-RKC method on the above test-equation gives the stability function

$$R_s(z_0, z_1) = a_s + b_s T_s \left(\frac{\omega_0 + \omega_1 z_0}{1 - \frac{\omega_1}{\omega_0} z_1} \right),$$

with $z_0 = \lambda_E \tau$ and $z_1 = \lambda_I \tau$. For stability we require

$$|R_s(z_0, z_1)| \leq 1.$$

Since we assumed that $\lambda_I \in [-\infty, 0]$ it follows that $z_1 \in [-\infty, 0]$. Therefore

$$\left| \frac{\omega_0 + \omega_1 z_0}{1 - \frac{\omega_1}{\omega_0} z_1} \right| \leq |\omega_0 + \omega_1 z_0| \,. \tag{5.8}$$

From (5.8) it follows easily that

$$|R_s(z_0, z_1)| \leq 1,$$

as long as $z_0 \in [-\beta(s), 0]$. The IMEX extension of the RKC scheme is unconditionally stable for the implicit part and the stability condition for the explicit part remains unchanged.

5.3 Consistency

The local truncation error of the RKC scheme is of order two. The IMEX extension of the second order RKC scheme has the following local error,

$$\rho_n = \frac{\tau}{s^2 - 1} S_n + \mathcal{O}(\tau^2),$$

with s the number of stages and $S_n = F_I'(w(t_n))F(w(t_n))$ (F_I' denotes the Jacobian of F_I).

If the number of stages s is large, then the influence of the extra term above does not harm. In the case the number of stages is relatively small, there will be order reduction. This means that in that case we have only order one consistency.

5.4 Final Remarks

In actual computations this IMEX extension of the RKC scheme is used in the same manner as the RKC scheme. The only difference is that in each stage a nonlinear algebraic system has to be solved. When solving this nonlinear system is cheap, as e.g. with stiff chemical reactions giving rise to small sized systems decoupled over the grid, the IMEX-RKC scheme (5.7) is more efficient than the fully explicit form (5.1).

Advection-diffusion-reaction problems that are advection dominated or problems where advection is discretized with (higher order) upwind schemes give rise to (more) complex eigenvalues of the Jacobians of the right-hand-side of (5.2). In that particular case one would like to increase the stability region in complex direction. By increasing the damping parameter ε the strip along the real line becomes wider. See Figure 5.1. A disadvantage is

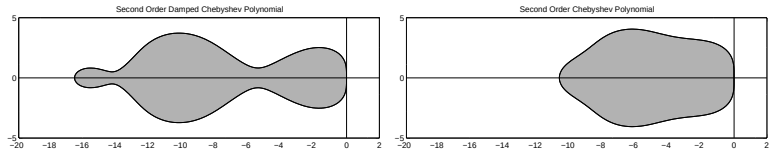

Figure 5.1: Stability regions for the second order polynomial P_5 with damping parameter ε small ($\varepsilon = \frac{2}{13}$) (left) and large ($\varepsilon = 10$)

that the stability boundary $\beta(s)$ will become smaller. The result is that for the moderate stiff terms like discretized diffusion, a smaller time-step must be taken to maintain stability.

Chapter 6

Linear Multi-Step Methods

One-step methods like Runge-Kutta methods compute w_{n+1} using only the previous approximation w_n. Linear multi-step methods on the other hand, use also additional previous approximations w_{n-i}, $i = 0, \ldots, k$, with a fixed integer k. This integer k is the total number of previous approximations w_{n-i} used to compute the next approximation w_{n+1}. Next, we give the definition of an k stage linear multi-step method.

Definition 6.1. *The linear k-step method is defined by the formula*

$$\sum_{j=0}^{k} \alpha_j w_{n+j} = \tau \sum_{j=0}^{k} \beta_j F(t_{n+j}, w_{n+j}), \quad n = 0, 1, \ldots, \tag{6.1}$$

which uses the k past values $w_n, \ldots, w_{n+k-1}$ to compute w_{n+k}. Remark that the most advanced level is t_{n+k} instead of t_{n+1}.

The method is explicit when $\beta_k = 0$ and implicit otherwise. Furthermore, we will assume that $\alpha_k > 0$.

The relation with the class of Runge-Kutta methods is described in the following remark.

Remark 6.2. *Taking $k = 1$ in Definition 6.1 gives a one-step method. In this case of linear one-step methods there is some overlap with Runge-Kutta methods. For instance, the θ method,*

$$w_{n+1} = w_n + \tau \left((1 - \theta) F(t_n, w_n) + \theta F(t_{n+1}, w_{n+1}) \right),$$

belongs to the class of Runge-Kutta methods. Its Butcher array is given in Table 6.1. Obviously, the θ method also belongs to the class of linear multi-step methods.

An advantage of linear multi-step methods with respect to Runge-Kutta methods is the following. In every new approximation w_{n+k} of *explicit* linear multi-step methods only one function evaluation is needed against s

0	0	0
1	0	1
	$1-\theta$	θ

Table 6.1: Butcher array of the θ method

for explicit s stage Runge-Kutta methods. We remark that explicit linear multi-step methods use more memory to store the previous $(k-2)$ function evaluations.

A disadvantage of linear multi-step methods is that the first $k-1$ approximations can *not* be computed with the linear k-step scheme. To compute the first $(k-1)$ approximations, one could use a

1. Runge-Kutta scheme,
2. use for the first step a linear 1-step method, for the second approximation a linear 2-step method, ... and for the $(k-1)^{\text{st}}$ approximation a linear $(k-1)$-step scheme.

6.1 The Order Conditions

A linear multi-step method is called consistent of order p if the local error satisfies

$$y(t_{n+k}) - \overline{w}_{n+k} = \mathcal{O}(\tau^{p+1}),$$

where $y(t_{n+k})$ is the exact solution and $\overline{w}_{n+k}$ the approximation obtained by inserting the past k exact values in (6.1).

To find order conditions for order p consistency we will do the following. First we introduce the linear difference operator L, associated with (6.1), as

$$L(y,t,\tau) = \sum_{i=0}^{k} \left[\alpha_i y(t+i\tau) - \tau\beta_i y'(t+i\tau)\right].$$

Here $y(t)$ is some differentiable function defined on an interval that contains $t_n, \ldots, t_{n+k}$.

Lemma 6.3. *Consider $y'(t) = F(t,y)$ with $F(t,y)$ continuously differentiable. Let $y(t)$ be its exact solution. For the local error one has*

$$y(t_{n+k}) - \overline{w}_{n+k} = \left(\alpha_k \mathbf{I} - \tau\beta_k \frac{\partial F}{\partial y}(t_{n+k},\eta)\right)^{-1} L(y,t_n,\tau). \tag{6.2}$$

In the case that $F(t,y)$ is a scalar function η is some value between $y(t_{n+1})$ and w_{n+1}. If $F(t,y)$ is a vector valued function, $\frac{\partial F}{\partial y}(t_{n+k},\eta)$ is the Jacobian of $F(t,y)$, whose rows are evaluated at possibly different values η_j lying on the segment between $(y(t_{n+k}))_j$ and $(w_{n+k})_j$.

Proof. By definition, the 'numerical' solution w_{n+k} is obtained by the equation

$$\sum_{j=0}^{k-1} \alpha_j y(t_{n+j}) - \tau\beta_j F(t_{n+j}, y(t_{n+j})) + \alpha_k w_{n+k} - \tau\beta_k F(t_{n+k}, w_{n+k}) = 0.$$

Inserting the operator $L(y, t_n, \tau)$ gives

$$\begin{aligned} L(y, t_n, \tau) = \\ \alpha_k(y(t_{n+k}) - w_{n+k}) - \tau\beta_k(F(t_{n+k}, y(t_{n+k})) - F(t_{n+k}, w_{n+k})). \end{aligned} \tag{6.3}$$

Statement (6.2) in Lemma 6.3 follows from the mean value theorem applied to the right-hand side of (6.3). □

Following [10], a multi-step method (6.1) is said to be of order p, if one of the following conditions is satisfied

1. for all sufficiently regular functions $y(t)$ we have $L(y, t, \tau) = \mathcal{O}(\tau^{p+1})$,
2. the local error of (6.1) is $\mathcal{O}(\tau^{p+1})$ for all sufficiently regular differential equations $y'(t) = F(t, y(t))$.

Next, observe that Lemma 6.3 implies that the above conditions 1. and 2. are equivalent. The following theorem gives the conditions for coefficients α_i and β_i, $i = 0, \ldots, k$, to get a multi-step scheme of order p.

Theorem 6.4. *The multi-step method (6.1) is of order p if and only if*

$$\sum_{i=0}^{k} \alpha_i = 0 \quad \text{and} \quad \sum_{i=0}^{k} \alpha_i i^j = j \sum_{i=0}^{k} \beta_i i^{j-1} \quad \text{for} \quad j = 1, 2, \ldots, p. \tag{6.4}$$

Proof. The method is of order p when $L(y, t, \tau) = \mathcal{O}(\tau^{p+1})$. Since

$$L(y, t, \tau) = \sum_{i=0}^{k} \left[\alpha_i y(t + i\tau) - \tau\beta_i y'(t + i\tau)\right],$$

Taylor expansions of $y(t+i\tau)$ and $y'(t+i\tau)$ near t gives the conditions (6.4) by setting $L(y, t, \tau) = \mathcal{O}(\tau^{p+1})$. □

Example 6.5. *Backward Differentiation Formulas, or shorter BDF methods, are implicit and defined by*

$$\beta_k = 1 \quad \text{and} \quad \beta_j = 0 \quad \text{for} \quad j = 0, \ldots, k-1,$$

with α_j chosen such that the order is optimal, namely k. The 1-step BDF method is Backward Euler. The 2-step method is

$$\frac{3}{2} w_{n+2} - 2w_{n+1} + \frac{1}{2} w_n = \tau F(t_{n+2}, w_{n+2}), \tag{6.5}$$

and the three step BDF is given by

$$\frac{11}{6}w_{n+3} - 3w_{n+2} + \frac{3}{2}w_{n+1} - \frac{1}{3}w_n = \tau F(t_{n+3}, w_{n+3}). \tag{6.6}$$

In chemistry applications the BDF methods belong to the most widely used methods to solve stiff chemical reaction equations, due to their favorable stability properties. However, for $k > 6$ the BDF-methods are unstable, see [10, Theorem 3.4, page 329].

See [13]. For more background on stability properties for linear multi-step methods, we refer to the next section.

6.2 Stability Properties

We study in this part the stability properties of multi-step methods. As usual we will use the familiar test equation $y'(t) = \lambda y(t)$, where $\lambda \in \mathbb{C}$.

First, some properties on linear recursions are needed. Consider the linear recursion formula

$$\alpha_k y_{n+k} + \alpha_{k-1} y_{n+k-1} + \cdots + \alpha_0 y_n = 0, \tag{6.7}$$

with $\alpha_i \in \mathbb{C}$. We define its characteristic polynomial as the polynomial $\pi(X)$ of degree k

$$\pi(X) = \sum_{i=0}^{k} \alpha_i X^i.$$

The following lemma, formulated without proof, gives the general solution of (6.7) in terms of roots of it characteristic polynomial.

Lemma 6.6. *Let $\chi_1, \ldots, \chi_l$ be the roots of $\pi(X)$ of respective multiplicity $m_1, \ldots, m_l$. If each root of $\pi(X)$ has multiplicity 1, then $l = k$. The general solution of (6.7) is given by*

$$y_n = p_1(n)\chi_1^n + \cdots + p_l(n)\chi_l^n, \tag{6.8}$$

where $p_i(n)$ are polynomials of degree $(m_i - 1)$.

Then, the following can be deduced. From (6.8) it follows that boundedness of the sequence $(y_n)_{n\geq 0}$, is guaranteed when the roots $\chi_1, \ldots, \chi_l$ lie in the unit disc and that roots *on* the unit disc are simple. The latter two conditions are called the *root condition*.

To derive stability properties we will use the above results. The coefficients of the linear multi-step method will be contained in the polynomials

$$p(X) = \sum_{i=0}^{k} \alpha_i X^i \quad \text{and} \quad \sigma(X) = \sum_{i=0}^{k} \beta_i X^i. \tag{6.9}$$

6.2.1 Zero-Stability

A typical multi-step requirement is the notion of zero-stability. Consider the trivial equation $w'(t) = 0$. Applying a linear k-step method to this equation gives, by substituting $F(t, w(t)) = 0$ in (6.1), the linear recursion

$$\sum_{i=0}^{k} \alpha_i w_{n+i} = 0.$$

The above linear recursion has characteristic polynomial $p(X)$.

Definition 6.7. *A linear multi-step method (6.1) is said to be zero-stable, when its characteristic polynomial $p(X)$ satisfies the root condition, i.e., the roots χ_i, $i = 0, \ldots, k$, with multiple roots repeated, satisfy*

$$|\chi_i| \leq 1 \quad \textit{and} \quad |\chi_i| < 1 \quad \textit{if } \chi_i \textit{ is not simple.}$$

In other words, Definition 6.7 puts forward that if a linear multi-step method is not zero stable, then this method even fails to solve the trivial equation $w'(t) = 0$ in a stable manner. We remark that this requirement holds trivially for consistent one-step methods.

The First Dahlquist Barrier

Next, some remarks on maximal attainable orders of linear multi-step methods will follow. Firstly, we remark that due to the order conditions (6.4) it follows by counting the number of coefficients α_i and β_i that the maximal order of an implicit linear k-step method (i.e. $\beta_k \neq 0$) is $2k$. For an explicit linear k-step method (i.e. $\beta_k = 0$) this is $2k - 1$. As mentioned above, a linear multi-step method makes only sense if it is also zero-stable. Zero-stability reduces for explicit methods the maximal attainable order to $p = k$. For implicit methods it reduces to $p = 2\lfloor \frac{k+2}{2} \rfloor$. This is better known as the first Dahlquist barrier. For a derivation of this result, we refer to [5, 10].

6.2.2 The Stability Region

Applying the linear k-step method (6.1) to the test equation $w'(t) = \lambda w(t)$, with $\lambda \in \mathbb{C}$ gives

$$\sum_{j=0}^{k} (\alpha_j - z\beta_j) w_{n+j} = 0, \tag{6.10}$$

for $n = 0, 1, \ldots$ and where $z = h\lambda$. The linear recursion (6.10) has the characteristic polynomial

$$\rho(X) = p(X) - z\sigma(X) = \sum_{j=0}^{k} (\alpha_j - z\beta_j) X^j.$$

By definition, the multi-step method is stable if the sequence $(w_n)_{n\geq 0}$ is bounded. Boundedness of $(w_n)_{n\geq 0}$ is guaranteed when $\rho(X)$ satisfies the root condition[1]. Therefore, the *stability region* is the set $\mathcal{S} \subset \mathbb{C}$ such that

$$z \in \mathcal{S} \Leftrightarrow \rho_z(X) \text{ satisfies the root condition.}$$

Furthermore, zero-stability is guaranteed if and only if $0 \in \mathcal{S}$.

The characteristic polynomial $\rho(X)$ has k roots. As generally known [2], only one of these roots approximates the exact solution $w(t) = e^{\lambda t} = e^z$, up to $\mathcal{O}(z^{p+1})$. This particular root is called the superior root. The other $k-1$ roots are called the spurious roots. Spurious roots do not play a role in the accuracy of the method. Sometimes they can cause instability.

Determination of the Stability Region

To determine the stability region of a linear multi-step method we observe that its boundary is described by $|\rho_z(\tilde{\chi})| = 1$, where $\tilde{\chi}$ is one of the roots of the characteristic polynomial $\rho_z(X)$. Recall that

$$z \in \mathcal{S} \Leftrightarrow \rho_z(\chi) \text{ satisfies the root condition.}$$

In the case that $\rho_z(\chi)$ satisfies the root condition, the boundary of this region is described in the χ-plane is the unit disk, i.e. $\chi = e^{i\theta}$ with $0 \leq \theta \leq 2\pi$. In the z-plane this boundary $\partial\mathcal{S}$ can be found as follows. The parameter z and χ are related to each other by $\rho_z(\chi) \equiv p(\chi) - z\sigma(\chi) = 0$. From this relation follows that the map $\chi \mapsto z$ is given by

$$z = \frac{p(\chi)}{\sigma(\chi)}. \tag{6.11}$$

The boundary $\partial\mathcal{S}$ is then described by inserting $\chi = e^{i\theta}$, with $0 \leq \theta \leq 2\pi$. Thus, the boundary $\partial\mathcal{S}$ of the stability region $\mathcal{S}$ is described by

$$z = \frac{p(e^{i\theta})}{\sigma(e^{i\theta})}, \quad 0 \leq \theta \leq 2\pi. \tag{6.12}$$

The curve (6.12) must be considered as a left oriented curve. The stability region $\mathcal{S}$ is the part left from the curve, as long as it is not empty.

[1] The root condition is among others given in Definition 6.7

[2] This can be shown by inserting $w(t) = e^{\lambda t}$ and $F(t, w) = \lambda w$ into

$$\sum_{j=0}^{k} \alpha_j w(t_{n+j}) = \tau \sum_{j=0}^{k} \beta_j F(t_{n+j}, w(t_{n+j})) + \tau\rho_{n+k-1},$$

with ρ_{n+k-1} the defect of order p.

A-Stability

To use linear multi-step methods in for instance chemical applications, *A*-stability of the particular method is desirable. Define $\bar{\mathbb{C}} = \mathbb{C} \cup \infty$. We say that $z = \infty$ belongs to the stability region if the polynomial $\sigma(\chi)$ satisfies the root condition. This makes indeed sense, because the roots of $\rho_z(\chi)$ tend to the roots of $\sigma(\chi)$ for $z \to \infty$. Therefore, a linear multi-step method is said to be *A*-stable if

$$\mathcal{S} \supset \left\{ z \in \bar{\mathbb{C}} : \operatorname{Re} z \leq 0 \text{ or } z = \infty \right\}.$$

Unlike Runge-Kutta methods, there are not many A-stable linear multi-step methods. It can be derived that a linear multi-step method can be at most second order consistent. This fundamental result is called the *second Dahlquist barrier*. At first sight, this second Dahlquist barrier seems to be a disappointing result. However, by introducing the notion of $A(\alpha)$ stability the problem is (partially) solved. A method is said to be $A(\alpha)$ stable if

$$\mathcal{S} \supset \left\{ z \in \bar{\mathbb{C}} : z = 0, \infty \text{ or } |\arg(-z)| \leq \alpha \right\}.$$

For many stiff ODE problems the *A*-stability property is not needed since for these problems the eigenvalues with large modulus stay away from the imaginary axis.

Example 6.8. *The BDF methods are for $k = 1, 2$ A-stable. For $3 \leq k \leq 6$ the BDF methods is $A(\alpha)$-stable, with α depending on k. See Table 6.2.*

k	1	2	3	4	5	6
α	90°	90°	86°	73°	51°	17°

Table 6.2: Values of α for given k.

In Figure 6.1 the stability regions for the BDF-2 method (6.5) and the BDF-3 method (6.6) are given. Notice that the BDF-3 method is indeed not A-stable, as can be seen in Figure 6.1. For the BDF-3 method we show how the boundary $\partial\mathcal{S}$ and the stability region $\mathcal{S}$ can be found.

Recall that the three step BDF method is

$$\frac{11}{6} w_{n+3} - 3w_{n+2} + \frac{3}{2} w_{n+1} - \frac{1}{3} w_n = \tau F(t_{n+3}, w_{n+3}).$$

Applying this method to the test equation gives the recursion

$$\frac{11}{6} w_{n+3} - 3w_{n+2} + \frac{3}{2} w_{n+1} - \frac{1}{3} w_n - z w_{n+3} = 0. \tag{6.13}$$

The characteristic polynomial corresponding to the above recursion (6.13) is then

$$\rho(X) = p(X) - z\sigma(X),$$

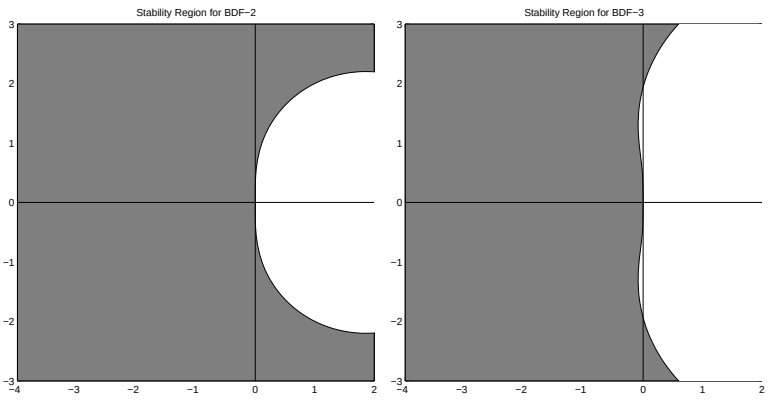

Figure 6.1: Stability regions of the BDF-2 method (left) and the BDF-3 method (right).

with

$$\begin{aligned} p(X) &= \frac{11}{6}X^3 - 3X^2 + \frac{3}{2}X - \frac{1}{3}, \quad \textit{and} \\ \sigma(X) &= X^3. \end{aligned}$$

The boundary $\partial\mathcal{S}$ of the stability region $\mathcal{S}$ of BDF-3 is described by

$$\begin{aligned} z &= \frac{p(e^{i\theta})}{\sigma(e^{i\theta})} \\ &= \frac{\frac{11}{6}(e^{i\theta})^3 - 3(e^{i\theta})^2 + \frac{3}{2}e^{i\theta} - \frac{1}{3}}{(e^{i\theta})^3} \quad , \quad 0 \le \theta \le 2\pi. \end{aligned}$$

One have to consider this as an oriented curve. The stability region is then the part left of it.

Chapter 7

Multi Rate Runge Kutta Methods

In this chapter we consider ODEs

$$y' = f(t, y), \tag{7.1}$$

where $y \in \mathbb{R}^n$ and $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$. We assume that the solution y and the right-hand side of (7.1) can be split up into rapidly (Active) and slowly (Latent) varying subsystems. By splitting up

$$y(t) = \begin{pmatrix} y_A(t) \\ y_L(t) \end{pmatrix}, \quad y_A(t) \in \mathbb{R}^{n_A}, \quad y_L(t) \in \mathbb{R}^{n_L} \quad \text{and} \quad n_A + n_L = n, \tag{7.2}$$

the ODE system (7.1) can then be written as a split system

$$\begin{array}{l} y'_A(t) = f_A(y_A(t), y_L(t)), \quad y_A(t_0) = y_{A,0} \\ y'_L(t) = f_L(y_A(t), y_L(t)), \quad y_L(t_0) = y_{L,0} \end{array}, \tag{7.3}$$

which for simplicity, but without loss of generality, is assumed to be autonomous.

Problems like (7.1) with a distinction between active and latent terms / subsystems arise frequently form discretization of PDEs of the advection-diffusion-reaction type, see e.g. [13]. Similar problems arise from applications in electronic circuits, multi-body dynamics and pneumatics.

With respect to splitting up the solution $y(t)$ we have some remarks. The first remark is that the splitting is time dependent, that is, for time $t+\Delta t$ the splitting can be different from the splitting at time t. Secondly, it can also vary in space. For example, in a chemical reactor with a heated susceptor, only in the area above the susceptor will be some active components in a gas mixture. In all other parts of the reactor all components will be slow, in general.

For more information with respect to the first remark we refer to [6]. With respect to the second remark, no literature could be found. However,

recently (February 2004), a research project started at the CWI, Amsterdam, which has as topic the contents of the above remarks. For sake of complicity, this research project is within the cluster *MAS3*, theme *Nonlinear Dynamics and Complex Systems*.

In this chapter we introduce some basics on multi-rate strategies and define a multi-rate Runge-Kutta approach.

7.1 Multi-Rate Runge-Kutta Methods

In the following we assume that the solution $y(t)$ of the initial value problem (7.1) is split into active components $y_A(t)$, hopefully a small subset of $y(t)$, and latent components $y_L(t)$. This results in a split system (7.3), which is assumed to be autonomous.

The active components y_A are integrated with a small step-size h, whereas the latent components are integrated with the large stpsize H. We remark that the methods to integrate the subsystems can, but do not have to, be the same. At the end of each macro-step H a synchronization of the micro-steps h and macro-steps H is performed.

The following definition of a multi-rate Runge-Kutta method (MRK) for the numerical solution of (7.3) is taken from [19].

Definition 7.1. *Assume we have split system (7.3) and that the active respectively latent part is integrated with a micro-step h respectively macro-step H. For each macro-step H and micro-step h we have the relation*

$$h = \frac{1}{m} H. \tag{7.4}$$

The active components y_A are integrated by

for each $\lambda = 0, 1, 2, \ldots, m-1$:

$$k_{A,i}^{\lambda} = f_A\left(y_{A,\lambda} + h \sum_{j=1}^{s} a_{ij} k_{A,j}^{\lambda}, \tilde{Y}_{L,i}^{\lambda} \right), \quad i = 1, 2, \ldots, s, \tag{7.5}$$

$$y_{A,\lambda+1} = y_{A,\lambda} + h \sum_{i=1}^{s} b_i k_{A,i}^{\lambda} \approx y(t_0 + (\lambda+1)h), \tag{7.6}$$

Here, $\tilde{Y}_{L,i}^{\lambda} \approx y_L(t_0 + (\lambda + c_i)h)$ and $c_i = \sum_{j=1}^{s} a_{ij}$. More details on $\tilde{Y}_{L,i}^{\lambda}$ will follow in the next section.

The latent components y_L are integrated by

$$k_{L,i}^{\lambda} = f_L\left(\tilde{Y}_{A,i}, y_{L,0} + H\sum_{j=1}^{s}\bar{a}_{ij}k_{L,j}\right), \quad j = 1,2,\ldots,\bar{s}, \tag{7.7}$$

$$y_{L,1} = y_{L,0} + H\sum_{i=1}^{\bar{s}}\bar{b}_i k_{L,i} \approx y(t_0+H), \tag{7.8}$$

where $\tilde{Y}_{A,i} \approx y_A(t_0 + \bar{c}_i H)$ and $\bar{c}_i = \sum_{j=1}^{\bar{s}}\bar{a}_{ij}$. In the next section we give more detailed information on $\tilde{Y}_{A,i}$.

As can be seen in the above definition, the coupling between the active and latent subsystem, and vice versa, is performed by the intermediate stage values $\tilde{Y}_{A,i}$ and $\tilde{Y}_{L,i}^{\lambda}$. These intermediate values can be computed by interpolation or extrapolation according to the following strategies:

- *Fastest first strategy*:
  - Integrate the active components y_A with m steps of step-size h; y_L-values are obtained by extrapolation from the previous step,
  - Integrate the latent components y_L with one macro-step H; y_A-values are obtained by interpolation,
- *Slowest first strategy*:
  - Integrate the latent components y_L with one macro-step H; y_A-values are obtained by extrapolation from the previous step,
  - Integrate the active components y_A with with m steps of step-size h; y_L-values are obtained by interpolation,

The extrapolation / interpolation strategies described above are natural choices, but these approaches inevitably turn the Runge-Kutta method into a two-step method. In [19] an approach for the stage values $\tilde{Y}_{A,i}$ and $\tilde{Y}_{L,i}^{\lambda}$ is given such that the multi-rate Runge-Kutta method is a one (macro) step method. This can be achieved, with no extra cost,by using the following formulas for the stage values:

$$\tilde{Y}_{L,i}^{\lambda} = y_{L,0} + h\sum_{j=1}^{\bar{s}}(\gamma_{ij} + \eta_j(\lambda))k_{L,j} \quad i = 1,2,\ldots s, \qquad \lambda = 0,1,\ldots,m-1, \tag{7.9}$$

$$\tilde{Y}_{A,i}^{\lambda} = y_{A,0} + H\sum_{j=1}^{s}(\bar{\gamma}_{ij})k_{A,j}^{0} \quad i = 1,2,\ldots \bar{s}. \tag{7.10}$$

This strategy resembles the slowest first strategy in the sense that y_A-values are obtained by extrapolation from the first micro-step.

Another idea to compute the $\tilde{Y}_{A,i}$-values is to use a lower order method. We refer to [19] for more information behind this idea. We remark that the method using this idea is called MRKII. The method given by (7.9) is called MRKI.

7.2 Order Conditions

We remark that we have not mentioned any conditions for γ_{ij}, $\bar{\gamma}_{ij}$ and $\eta_j(\lambda)$ to maintain the order of the given MRK methods. We will follow reference [19] to give some conditions to obtain a method of respectively order two and three.

Given two *third* order RK-methods for the integration of the active and latent components of (7.3). These methods are joined in an MRK by the coupling coefficients γ_{ij}, $\bar{\gamma}_{ij}$ and $\eta_j(\lambda)$, which must be chosen carefully to retain the order of the method.

To ensure the MRKI to be of order two the following assumptions have to be satisfied:

$$\sum_{j=1}^{\bar{s}} \eta_j(\lambda) = \lambda, \quad \sum_{j=1}^{\bar{s}} \gamma_{ij} = c_i \quad i = 1, 2, \ldots, s, \tag{7.11}$$

and

$$\sum_{j=1}^{s} \bar{\gamma}_{ij} = \bar{c}_i \quad i = 1, 2, \ldots, s. \tag{7.12}$$

Remark that no extra conditions are needed.

To be of order three the following conditions, in addition to the classical ones, have to be fulfilled:

$$\sum_{i=1}^{s}\sum_{j=1}^{\bar{s}} b_i \gamma_{ij} \bar{c}_j = \frac{1}{6m}, \quad \sum_{i=1}^{s}\sum_{j=1}^{\bar{s}} b_i \eta_j(\lambda) \bar{c}_j = \frac{\lambda(\lambda+1)}{2m}, \tag{7.13}$$

and

$$\sum_{i=1}^{\bar{s}}\sum_{j=1}^{s} \bar{b}_i \bar{\gamma}_{ij} c_j = \frac{m}{6}. \tag{7.14}$$

Note that what has been said about MRKI applies to explicit as well as implicit methods, and even a mix of both, like implicit for the latent part and explicit for the active.

From this point onwards one can construct particular MRKI methods. In [19] for example, one constructs a third order explicit MRK-method.

7.3 Stability

Following [20], we adopt the test-problem,

$$\begin{pmatrix} \dot{y}_A \\ \dot{y}_L \end{pmatrix} = \mathcal{A} \begin{pmatrix} y_A \\ y_L \end{pmatrix}, \quad \mathcal{A} = \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix} \in \mathbb{R}^{2\times 2}, \tag{7.15}$$

because multi-rate formulas require at least two equations. Assuming that

$$\alpha_{11}, \alpha_{22} < 0 \quad \text{and} \quad \gamma = \frac{\alpha_{12}\alpha_{21}}{\alpha_{11}\alpha_{22}} < 1, \tag{7.16}$$

ensures that both eigenvalues of $\mathcal{A}$ have negative real parts. The parameter γ can be viewed as a measure for the coupling between the equations. Also a measure κ for the stiffness of the system is introduced, i.e.,

$$\kappa = \frac{\alpha_{22}}{\alpha_{11}}. \tag{7.17}$$

A compound step of the multi-rate Runge-Kutta methods suggested in the previous section applied to (7.15) can be expressed as

$$\begin{pmatrix} y_{A,m} \\ y_{L,1} \end{pmatrix} = \mathcal{K} \begin{pmatrix} y_{A,0} \\ y_{L,0} \end{pmatrix}, \tag{7.18}$$

where the matrix $\mathcal{K}$ depends on the step-sizes H and h, and the coefficients of the method. Observe that the method is asymptotically stable if and only if the eigenvalues of $\mathcal{K}$ are both within the unit disk.

The following scenario will be studied. For both parts, the active and the latent part of $y(t)$, a method has been chosen to integrate the particular subsystem. Step-sizes H and h are chosen such that the uncoupled system ($\alpha_{12} = \alpha_{21} = 0$) is stable. This means that the stability functions $R_A(h\alpha_{11})$ and $R_L(H\alpha_{22}$ are both of absolute value less than one.

Remark that for the case $\alpha_{12} = \alpha_{21} = 0$ the eigenvalues of $\mathcal{K}$ are equal to R_L and R_A^m. Therefore, it seems convenient to express the matrix $\mathcal{K}$ in terms of R_A and R_L, rather than in terms of H and h. Now it is interesting to study the stability region, that is the values γ, R_A and R_L for which eigenvalues of $\mathcal{K}$ are in the unit disk, and how this region varies with increasing m and stiffness ratio κ.

To demonstrate this idea, we use the following example taken from [20]. Suppose we have for the active part the Euler Forward scheme and for the latent part the Euler Backward scheme, i.e.,

$$y_{A,\lambda+1} = y_{A,\lambda} + h\alpha_{11}y_{A,\lambda}, \quad \lambda = 0, 1, 2, \ldots, m-1, \tag{7.19}$$

$$y_{L,1} = y_{L,0} + H\alpha_{22}y_{L,1}. \tag{7.20}$$

Remark that for every step-size H the stability function $|R_L| < 1$ and that $|R_A| < 1$ if and only if $-2 < \alpha_{11}h < 0$. The stability functions R_A and R_L

are related through $H = mh$ as

$$R_L = \frac{1}{1 + m(1 - R_A)\kappa}. \tag{7.21}$$

Using this and a lot of calculations, which are done in [20], one obtains a set of inequalities that determine a set of values for γ and R_A such that the eigenvalues of $\mathcal{K}$ are in the unit disk. The following stability regions are obtained for $\kappa = 1$ and $\kappa = 100$, see Figure 7.1. We can conclude that the stability region becomes smaller with increasing m and increasing κ. In this case the coupling has not much influence, but there are multi-rate methods where 'strong' coupling gives very small stability regions. A modification of the synchronization of the active and latent part, gives a stability region as in 7.2, which is taken from [20]. We observe that in the second case coupling and stiffnes influence the range of step-sizes h that integrate (7.19) in a stable way.

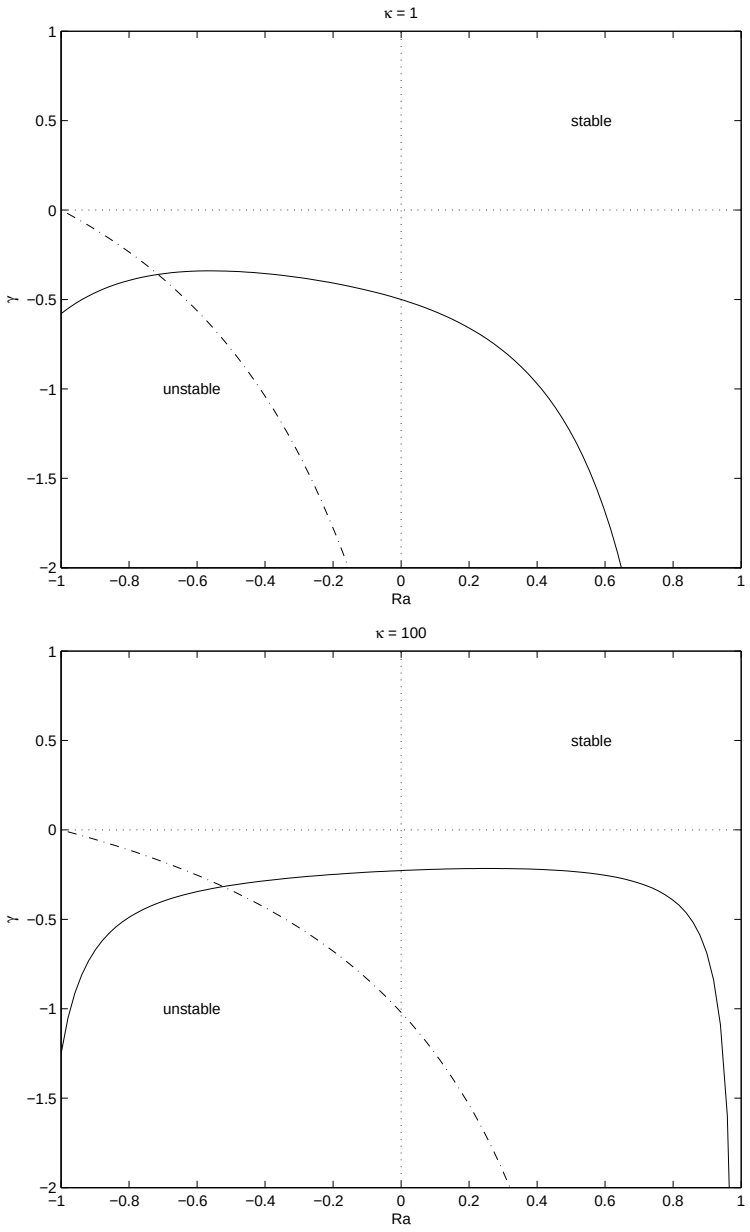

Figure 7.1: Boundaries of the stability regions for $m = 1$ (–)) and $m = 10$ (solid). The methods are stable to the right of the boundaries and unstable to the left.

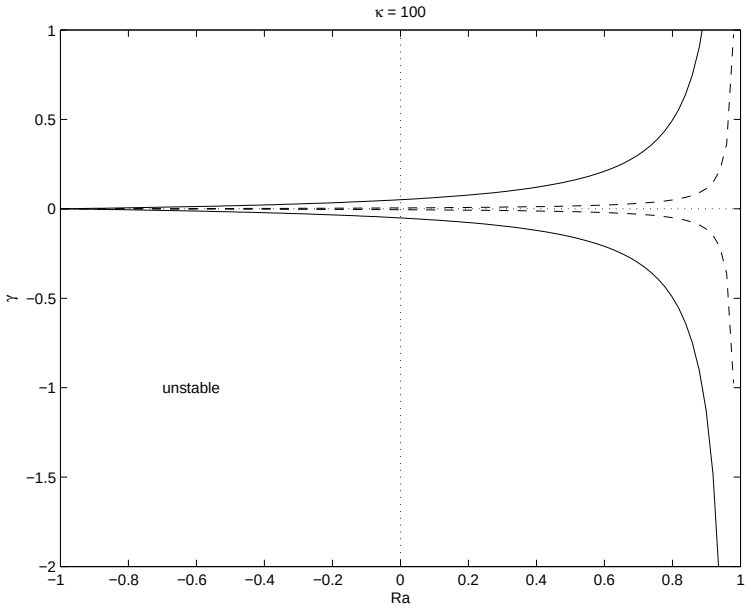

Figure 7.2: Boundaries of the stability regions for $m = 1$ (–)) and $m = 10$ (solid). The methods are stable to the right of the boundaries and unstable to the left.

Part III

Nonlinear Solvers

Chapter 1

Introduction

Solving a partial differential equation numerically gives after spatial and time discretization in general a (huge) system of nonlinear equations. There are several methods to solve the general nonlinear equation

$$F(x) = 0,$$

where $x \in \mathbb{R}^n$ and $F : \mathbb{R}^n \to \mathbb{R}^m, m, n \in \mathbb{N}$. Well known are the fixed point iteration, the Newton iteration, Broyden method and so on. In this part we will treat the most important ones. First we present the Newton iteration in one and more variables and also Broyden's method. In Chapter 3 we discuss the Picard iteration.

Chapter 2

Newton's Method

The contents of this chapter is based on references [15, 23]. We present general properties and facts of Newton's method, which we will also call Newton iteration. This chapter has the following construction. First we present Newton's method in one and more variables and present some general properties. Then, we continue with convergence properties and criteria to terminate the iteration. Furthermore, we pay some attention to modifications of the Newton iteration. Finally we conclude this chapter with possible failures of these nonlinear solvers.

2.1 Newton's Method in One Variable

We shortly discuss the Newton iteration that solves the nonlinear equation

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

where $f(x)$ is a real-valued function in the variable $x \in \mathbb{R}$. The iteration

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}, \quad k = 0, 1, 2, \ldots \tag{2.2}$$

is known as the *Newton Iteration*. This method can also be represented graphically as done in Figure 2.1. Properties and a derivation of this method will be done for the more general case in the next section.

2.2 General Remarks on Newton's Method

In order to solve n dimensional systems $F(x) = 0$, with $F : \mathbb{R}^n \to \mathbb{R}^n$ and $x \in \mathbb{R}^n$, the Newton iteration (2.2) can be generalized into

$$x^{k+1} = x^k - F'(x^k)^{-1} F(x^k). \tag{2.3}$$

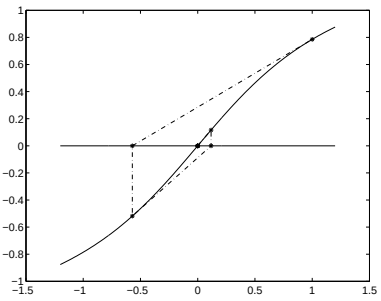

Figure 2.1: Illustration of the Newton method

Importance of the above method rests on the fact that, under certain conditions on F, the error $\|x^{k+1} - x\|$ (remark that x is a solution of $F(x) = 0$) can be estimated by the inequality

$$\|x^{k+1} - x\| \leq c\|x^k - x\|^2. \tag{2.4}$$

Inequality (2.4) yields for a $c \in \mathbb{R}$ and a certain norm defined on $\mathbb{R}^n$. The above inequality states that the $(k+1)^{\text{th}}$ error is proportional to the k^{th} error squared. We will not give an exact proof of this estimate, but only a sketch. For the exact proof we refer to [23, Chapter 10]. The sketch of the proof of inequality (2.4) starts with a sketch of the derivation of the method of Newton.

Assume $F : \mathbb{R}^n \to \mathbb{R}^m$ and that F is Fréchet differentiable[1] in x^k. Then

$$0 = F(x) = F(x^k) + F'(x)(x - x^k) + R(x - x^k), \tag{2.6}$$

where

$$\lim_{h \to 0} \frac{R(h)}{\|h\|} = 0.$$

For a proof we refer to [23]. If x^k is in a neighborhood of x, then it is natural to neglect the term $R(x - x^k)$. Furthermore, we assume that $F'(x^k)$ is non-singular in a neighborhood of x, which implies that $m = n$.

The difference $x - x^k$ can then be approximated by the solution of the system

$$F'(x^k)h = -F(x^k), \quad \text{with} \quad h = x - x^k. \tag{2.7}$$

As a new approximation of x we take

$$\begin{aligned} x^{k+1} &= x^k + h \\ &= x^k - F'(x^k)^{-1}F(x^k). \end{aligned} \tag{2.8}$$

[1]The mapping $F : D \subset \mathbb{R}^n \to \mathbb{R}^m$ is Fréchet differentiable at $x \in \text{int}D$ if there is an $A \in L(\mathbb{R}^n, \mathbb{R}^m)$ (space of linear mappings from $\mathbb{R}^n \to \mathbb{R}^m$) such that

$$\lim_{h \to 0} \frac{\|F(x+h) - F(x) - Ah\|}{\|h\|} = 0. \tag{2.5}$$

The linear operator A is denoted by $F'(x)$ and is called the Fréchet derivative of F at x.

If the second Fréchet derivative is bounded in a neighborhood of x and $F'(x)$ is non-singular, then it can be shown that there exists an α such that

$$\|R(x - x^k)\| \leq \alpha \|x - x^k\|^2. \tag{2.9}$$

For a proof we refer to [23, Theorem 3.3.6]. Subtract (2.8) of (2.6) gives

$$F'(x^k)(x - x^k) = R(x - x^k). \tag{2.10}$$

From the above equality follows

$$\|F'(x^k)\|\|(x - x^k)\| = \|R(x - x^k)\|. \tag{2.11}$$

Using (2.9) we obtain inequality (2.4).

Newton's method is theoretically attractive, but in practice there can be some difficulties. In each step of the iteration a linear system needs to be solved. Especially in practical applications the dimensions of the systems to solve can be up to one million or even larger. Another difficulty is that in each step not only the n components of $F(x^k)$ have to be evaluated, but also the n^2 entries of its Jacobian. In the case that the partial derivatives of each component have a simple functional form the Jacobian can be computed exactly. In all other cases it is more desirable to avoid these computations. Almost all modifications of Newton iterations are constructed in such a way to avoid the explicit computation of (partial) derivatives. An example of such modification is to use instead of the exact Jacobian an approximate of the Jacobian. One can approximate the Jacobian for example by finite differences. The price on has to pay for this modification is that the convergence to a solution will become more slowly, i.e., more iterations are needed to solve the problem. However, the overall cost of solving $F(x) = 0$ is usually significant less, because the computation of the Newton step is less expensive.

Some last general remarks on Newton's method and iteration methods in general. For each iterative method that solves $F(x) = 0$, it is required that the method is norm reducing in the sense that

$$\|F(x^{k+1})\| \leq \|F(x^k)\| \quad k = 0, 1, 2, \ldots, \tag{2.12}$$

holds for some norm defined on $\mathbb{R}^n$. In general the 'standard Newton iteration' (2.3) does not necessarily satisfy this requirement. This problem is solved by the simple modification

$$x^{k+1} = x^k - \omega_k F'(x^k)^{-1} F(x^k), \tag{2.13}$$

where the relaxation-parameter ω_k is chosen such that (2.12) holds. More on this in Section 2.6.

Another general modification of Newton's method is to reevaluate $F'(x)$ occasionally. The iteration scheme then becomes

$$x^{k+1} = x^k - F'(x^{p(k)})^{-1}F(x^k), \quad k = 0, 1, 2, \ldots \tag{2.14}$$

where $p(k)$ is some integer less or equal to k. For $p(k) \equiv k$ we have the 'standard Newton iteration' (2.3) and for $p(k) \equiv 0$ the simplified Newton method or chord method. A disadvantage of the chord method is that we have linear convergence, i.e., $\|x^{k+1} - x\| \leq \zeta \|x^k - x\|$, for a $\zeta \in (0, 1)$.

2.3 Convergence Properties

In this section we give a measure of the rate of convergence of an iterative process. An example of an iterative process is for instance the method of Newton.

Definition 2.1. *Let $\{x^n\}_{n=0}^{\infty}$ be a sequence in $\mathbb{R}^n$ that converges to a fixed $x \in \mathbb{R}$. Further, let $\|\cdot\|$ be a norm defined on $\mathbb{R}^n$. If there exist positive constants $\lambda \in \mathbb{R}$ and $p \in [1, \infty)$ such that*

$$\lim_{n\to\infty} \frac{\|x^{n+1} - x\|}{\|x^n - x\|^p} = \lambda, \tag{2.15}$$

then we say that x^n converges to x with order p and asymptotic constant λ.

Whenever a process converges with order one, $p = 1$, then we call the convergence q-linear. If a process converges with $p = 1$ and $\lambda = 0$, then we call such a process *q-super-linearly* convergent. Processes that converge with $p = 2$ are called q-quadraticly convergent. We remark that q-quadratic convergence is a special case of q-super-linear convergence.[2] In general, every process that converges with order p greater than one is super-linearly convergent.

As seen in the previous section Newton's method converges quadratically. The result is summarized in the next theorem.

Theorem 2.2. *Assume that $F(x) = 0$ has a solution, which we denote by x^*. Further, assume that the Jacobian $F'(x^*)$ is nonsingular and Lipschitz continuous near x^*. If x^0 is sufficiently near x^*, then the Newton sequence*

[2] Assume that our process converges quadratically. Recall that this also means that $\lim_{n\to\infty} \|x^n - x\| = 0$. Then, the fact that the iterative process converges quadratically implies that for n sufficiently large there exist a $K > 0$ such that $\|x^{n+1} - x\| \leq K\|x^n - x\|^2$. It follows that

$$\lim_{n\to\infty} \frac{\|x^{n+1} - x\|}{\|x^n - x\|} \leq \lim_{n\to\infty} \frac{K\|x^n - x\|^2}{\|x^n - x\|} = 0.$$

exists, i.e. $F'(x^n)$ is nonsingular for all $n \geq 0$, and the sequence converges to x^*. Furthermore, there exists a $K > 0$ such that

$$\|x^{n+1} - x\| \leq K\|x^n - x\|^2, \tag{2.16}$$

for n sufficiently large.

Usually in practice it is more efficient to approximate the Newton step in some way. One (obvious) way to do this is to approximate the Jacobian $F'(x^n)$ in such a way that computation of the derivative is avoided. The *secant method* for *scalar equations* approximates the derivative using finite differences. It uses the two most recent iterations to form the difference quotient, i.e.,

$$x^{n+1} = x^n - \frac{F(x^n)(x^n - x^{n-1})}{F(x^n) - F(x^{n-1})}, \tag{2.17}$$

where x^n is the current iteration and x^{n-1} the previous iteration. As general known, if the secant method converges, then it converges with order φ, where φ is the golden ratio.[3] We remark that the secant method must be initialized with two points. One way to do that is to let $x^{-1} = 0.99x^0$. We took this from [15]. Finally we remark that (2.17) cannot be extended to systems of nonlinear equations, because the denominator in the fraction would be a difference of vectors. There are many generalizations to n dimensions of the secant method. We will discuss one of them in Section 2.7.

2.4 Criteria for Termination of the Iteration

In practice we do not like to iterate forever. Therefore, an iterative method should be stopped if the approximate solution is accurate enough. A termination criterion is an essential part of an iterative method. A weak criterion is useless, while a too severe criterion makes the iterative method expensive or, even worser, it might never stop.

While one cannot know the error without knowing the solution, in most cases the norm of $F(x^k)$ can be used as a reliable indicator of the rate of decay of norm of the error $\|e^n\| = \|x - x^n\|$ as the iteration progresses. Termination of the iteration takes place, based on this heuristic, when

$$\|F(x^k)\| \leq \tau_r\|F(x^0)\| + \tau_a, \tag{2.18}$$

where τ_r is the relative error tolerance and τ_a the absolute error tolerance. Both tolerances are important. For instance, it is a poor idea to put $\tau_a = 0$. In the case that the initial iterate is near the solution x, then (2.18) is impossible to satisfy.

[3]The golden ratio φ is the positive root of the equation $\varphi^2 - \varphi - 1 = 0$, meaning that $\varphi = \frac{1+\sqrt{5}}{2}$. The golden ratio also satisfies the recurrence relation $\varphi^n = \varphi^{n-1} + \varphi^{n-2}$. Taking $n = 1$ gives the special case $\varphi = 1 + \varphi^{-1}$.

2.5 Inexact Newton Methods

In the Newton iteration there are two difficulties. The first one is to compute the Jacobian at the current iterate x_n. Secondly, computation of the solution of the equation

$$F'(x_n)s = -F(x_n), \tag{2.19}$$

can be very hard, or even impossible. Instead of solving the Newton step exactly one could also approximate it using some iterative method. More on iterative methods can be found in Part **??**.

Assume that the n^{th} iterate is known and that the Jacobian $F'(x^n)$ as well as $F(x^n)$ are known. Solving (2.19) by using an iterative method gives with a initial start-vector s_0 a sequence of vectors $\{s_n\}_{n\geq 0}$ that converges to the solution s of (2.19). After k iterations the residual r_k is equal to

$$r_k = F'(x^n)s_k + F(x^n). \tag{2.20}$$

A good criterion to stop the iterative process that approximates the solution of (2.19) is

$$\frac{\|F'(x^n)s_k + F(x^n)\|}{\|F(x^n)\|} \leq \eta \quad \eta \in \mathbb{R}. \tag{2.21}$$

For more on stopping criteria for iterative methods we refer to Section 2.4 and Part **??**. Criterion (2.21) rewritten gives the so-called *inexact Newton condition*

$$\|F'(x^n)s_k + F(x^n)\| \leq \eta\|F(x^n)\|, \tag{2.22}$$

where the parameter η is called the *forcing term*. Remark that by choosing a smaller value for η the inexact Newton method makes it more like Newton's method. Secondly we remark that during the Newton iteration the forcing term η can be varied, thus for the n^{th} iterate we have η_n. In the following theorem, taken without proof from [15], we give some convergence results.

Theorem 2.3. *Assume that $F(x) = 0$ has a solution, F' is Lipschitz continuous near this solution and that $F'(x)$ is nonsingular. Then there are δ and $\bar{\eta}$ such that, if $x_0 \in \mathcal{B}(\delta)$, $\{\eta_n\} \subset [0, \bar{\eta}]$, then the inexact Newton iteration*

$$x^{n+1} = x^n + s_n,$$

where

$$\|F'(x^n)s_n + F(x^n)\| \leq \eta_n\|F(x^n)\|, \tag{2.23}$$

converges linearly to x. Moreover, if $\eta_n \to 0$, then the convergence is q-super-linear.

2.6 Global Convergence

In this section we use an example taken from [15] to illustrate the theory. To illustrate that the Newton iteration only converges locally we apply it to the function

$$f(x) = \arctan(x), \tag{2.24}$$

with initial iterate $x^0 = 10$. This initial value is too far from the root, so that the local convergence theory is not valid. The Newton step can be computed easily and is equal to

$$s = \frac{f(x^0)}{f'(x^0)} \approx \frac{1.5}{-0.01} \approx -150. \tag{2.25}$$

Hence, this computed Newton step is in the correct direction, but is far too large in magnitude. By continuing the iteration we obtain the following sequence of iterates

$$\begin{array}{rcl} x^0 & = & 10 \\ x^1 & = & -138 \\ x^2 & = & 2.9 \cdot 10^4 \\ x^3 & = & -1.5 \cdot 10^9 \\ & \vdots & \end{array} \tag{2.26}$$

Since the computed Newton step s points in the correct direction we can apply the following simple modification. We reduce the Newton step by half until $\|f(x)\|$ has been reduced. We then get a convergence behavior as in Figure 2.2

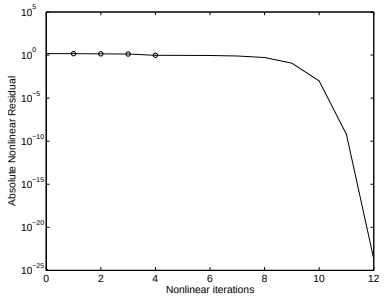

Figure 2.2: The absolute residual on log scale versus the number of iterations. The problem $\arctan(x) = 0$ is solved by a Newton iteration with initial iterate $x^0 = 10$. The Newton step is computed in such a way that (2.28) is satisfied.

We now return to the general case of $F(x) = 0$. In order to describe this artifice in a clear way we define the *Newton direction* as

$$d = -F'(x^n)^{-1} F(x^n), \tag{2.27}$$

and the *Newton step* s as a positive scalar multiple of the Newton direction d. In the local convergence theory there is no difference between the Newton direction and the Newton step, i.e. $s = d$. The Newton step will be determined as follows. Find the smallest integer m such that

$$\|F(x^n + 2^{-m}d)\| < (1 - \alpha 2^{-m})\|F(x^n)\|, \tag{2.28}$$

and let the Newton step be $s = 2^{-m}d$. Condition (2.28) is called the sufficient decrease of $\|F\|$. The parameter α in (2.28) is a small number, chosen in such a way to satisfy the sufficient decrease condition as easy as possible. In [15] $\alpha = 10^{-4}$.

Methods like the one described above are called line search methods. This kind of methods search for a decrease in $\|F\|$ along the segment $[x^n, x^n + d]$. Some problems can be more efficient solved if the step-length reduction is more aggressive, for instance by factors of $\frac{1}{10}$ instead of $\frac{1}{2}$. To make this possible, we can do the following after two reductions by half. Build a quadratic polynomial model of

$$\psi(\lambda) = \|F(x^n + \lambda d)\|^2, \tag{2.29}$$

which is based on interpolation of ψ at the three most recent values of λ. The minimizer of the quadratic model then becomes the next value for λ. As safeguard we claim that the reduction in λ is at least a factor of 2 and at most a factor of 10. This proposed algorithm, taken from [15], generates a sequence of λ_m's with

$$\lambda_0 = 1 \tag{2.30}$$

$$\lambda_1 = \frac{1}{2} \tag{2.31}$$

and λ_m such that

$$\frac{1}{10} \leq \frac{\lambda_m}{\lambda_{m-1}} \leq \frac{1}{2} \quad \text{for all} \quad m \in \{m \in \mathbb{N} | m \geq 2\}. \tag{2.32}$$

The line search will be terminated for the smallest $m \leq 0$ such that

$$\|F(x_n + \lambda_m d)\| < (1 - \alpha\lambda_m)\|F(x_n)\|, \tag{2.33}$$

holds in a certain norm. For more details or other ways to implement a line search we refer to [15].

For completeness the above is sketched in Algorithm 2.1. Finally we remark that for smooth (vector)functions F, there are only three possible scenarios for the iteration process of Algorithm 2.1 to occur:

1. the sequence $\{x_n\}_{n\geq 0}$ converges to a solution of $F(x) = 0$,
2. the sequence $\{x_n\}_{n\geq 0}$ will become unbounded,

Algorithm 2.1 Line Search

Evaluate $F(x)$; $\tau \leftarrow \tau_r \|F(x)\| + \tau_a$.
while $\|F(x)\| > \tau$ **do**
  Find $d = -F'(x)^{-1}F(x)$.
  If no such d can be found, then terminate with failure.
  $\lambda = 1$.
  while $\|F(x + \lambda d)\| > (1 - \alpha\lambda)\|F(x)\|$ **do**
    $\lambda \leftarrow \sigma\lambda$, where $\sigma \in [\frac{1}{10}, \frac{1}{2}]$ is computed by minimizing the polynomial model of $\|F(x + \lambda d)\|^2$.
  end while
  $x \leftarrow x + \lambda d$.
end while

3. the Jacobian $F'(x_n)$ will become singular for a certain x_n.

The line search algorithm as presented here is the most simplest way to find a root of $F(x)$, when the initial iterate is far from a root. There are many alternatives to this line search algorithm that can sometimes overcome stagnation or, in the case of many solutions, find the solution that is appropriate to the physical problem. Three of these other methods are *trust region globalization*, *pseudo-transient continuation* and *homotopy methods*. These alternatives will not be discussed in this report. For more information we refer to [15].

2.7 Extension of Secant Method to n Dimensions

For solving the scalar equation $f(x) = 0$ the *secant method* is given as

$$x^{k+1} = x^k - \frac{f(x^k)(x^k - x^{k-1})}{f(x^k) - f(x^{k-1})}, \tag{2.34}$$

where x^k is the current iteration and x^{k-1} the iteration before. As seen before, the secant method converges with order equal to the golden ratio.

One of the most famous extensions of the secant method to n dimensions is Broyden's method. We will shortly show how this method can be derived from the $(k + 1)$-point sequential secant method, which is derived in [23, Chapter 7]. We remark that this method is not used in practice. Methods that are used in practice can be derived from this method.

The $(n+1)$-point sequential secant method is the n dimensional extension of the scalar secant method and given as

$$x^{k+1} = x^k - A_k^{-1}F(x^k), \quad k = 0, 1, \ldots, \tag{2.35}$$

where

$$A_k = \Gamma_k H_k^{-1}, \quad H_k = [p^{k-1}, \ldots, p^{k-n}], \quad \Gamma_k = [q^{k-1}, \ldots, q^{k-n}], \tag{2.36}$$

with

$$p^i = x^{i+1} - x^i \quad \text{and} \quad q^i = F(x^{i+1}) - F(x^i). \tag{2.37}$$

Remark that this method needs n initial values, e.g., $x^0, \ldots, x^{-n}$. The matrix A_k is an approximation of the Jacobian in the k^{th} iteration. For the next iteration the approximation of the Jacobian will be given by

$$A_{k+1} = A_k - F(x^{k+1})(v^k)^T, \tag{2.38}$$

with v^k the first row of H_{k+1}^{-1}. For a derivation of this result, see [23], Theorem 7.3.1. This result suggests that A_{k+1} is obtained from A_k by addition of a matrix of rank one. Therefore, the more general form of (2.38) is

$$A_{k+1} = A_k + u^k(v^k)^T, \quad \text{for some} \quad u^k, v^k \in \mathbb{R}^n. \tag{2.39}$$

Using the Sherman - Morrison formula[4] gives

$$A_{k+1}^{-1} = A_k^{-1} - \frac{A_k^{-1}u^k(v^k)^T A_k^{-1}}{1 + (v^k)^T A_k^{-1} u^k}, \tag{2.41}$$

where we assumed the denominator to be unequal to zero and A_0 nonsingular. This is the first constraint on v^k and u^k, e.g.,

$$1 + (v^k)^T A_k^{-1} u^k \neq 0. \tag{2.42}$$

From now on it is possible to construct a wide range of methods. Next, we assume that

$$(v^k)^T p^k \neq 0 \quad \text{and} \quad u^k = \frac{F(x^{k+1})}{(v^K)^T p^k} \quad k \in \mathbb{N}. \tag{2.43}$$

The first constraint becomes then, using $p^k + A_k^{-1}F(x^{k+1}) = A^{-1}q^k$,

$$(v^k)^T A_k^{-1} q^k \neq 0. \tag{2.44}$$

Then, (2.41) becomes

$$A_{k+1}^{-1} = A_k^{-1} - \frac{A_k^{-1}F(x^{k+1})(v^k)^T A_k^{-1}}{(v^k)^T A_k^{-1} q^k}. \tag{2.45}$$

Broyden's method [3] is a special case of the method defined by (2.43), in which v^k is taken equal to p^k. We remark that (2.44) may *not* be equal to zero. The corresponding algorithm, see [3], is given as Algorithm 2.2. In Algorithm 2.2 the matrix A_k^{-1} is represented by the matrix B_k. Furthermore, we remark that the update B_{k+1} in State 8 of the Broyden algorithm is in practice not computed in that way. For more information on the Broyden algorithm we refer to [3, 15].

[4]Sherman - Morrison formula: Let the $n \times n$ matrix A be invertible and let $u, v \in \mathbb{R}^n$. Then $A + uv^T$ is invertible if and only if $1 + v^T A^{-1} u \neq 0$, and then

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}. \tag{2.40}$$

Algorithm 2.2 Broyden

1: Obtain an initial estimate x^0 of the solution.
2: Obtain an initial value of the iteration matrix B_0. For instance, this can be done by approximating the Jacobian at x^0 and inverting the result.
3: Compute $F_i = F(x_i)$ $i = 0, 1, 2, \ldots$
4: Compute $p_i = -B_i F_i$.
5: Choose λ_i such that $x^{i+1} = x^i + \lambda_i p_i$ implies $\|F_{i+1}\| < \|F_i\|$.
6: Test $\|F_{i+1}\|$ for convergence.
7: Compute $y_i = F_{i+1} - F_i$.
8: Compute $B_{i+1} = B_i - \frac{B_i F(x^{k+1})(p_i)^T B_i}{(p_i)^T B_i y_i}$.
9: Repeat from Step 4.

2.8 Failures

In this section we discuss possible failures of the nonlinear solvers presented in this chapter. Among others we treat possible causes for failure to converge.

In the case that the problem has no solution, then any solver will have trouble finding a solution. For example, if $f(x) = e^{-x}$, then the Newton iteration will converge to $-\infty$ for any starting point.

2.8.1 Non-smooth Functions

All variations of the Newton iteration which are discussed in this chapter are intended to find roots of $F(x) = 0$ for which $F'(x)$ is Lipschitz continuous. Solving problems with a discontinuous Jacobian causes unpredictable behavior of the codes. An example of a function with a non-smooth Jacobian is the vector norm.

2.8.2 Slow Convergence

In the case one has a nonlinear solver that converges, but not as fast as it should, then probably one of the following three operations is inaccurate :

- Computation of the Jacobian,
- Computation of the Jacobian-vector product,
- Linear solver.

Super-linear convergence which follows from the local convergence theory only holds if the *correct linear system* is solved accurately. In order to get the expected super-linear convergence, one could check the following things.

1. Check the computation of the Jacobian,

2. If you use a iterative linear solver, then make sure that the termination criteria for the linear solver are set tight enough.

2.8.3 No Convergence

If the Newton iteration does not converge, then either the iteration will become unbounded or the Jacobian will become singular. We give possible causes of failure of convergence. The first one is to check whether the problem has a solution or not. In the case your problem has a solution one of the following remarks might cause the failure.

- Inaccurate function evaluation. If the error in the function evaluation is higher than the machine roundoff, then the computed Newton direction, which is computed by a difference Jacobian, can be poor enough for the iteration to fail.
- Singular Jacobian. If the Jacobian becomes (almost) singular, then the step lengths will become zero. In the case that one terminates the iteration when the step length approaches zero and does not check whether F approaches zero, then one concludes incorrectly that the problem has been solved.
- If the line search algorithm as presented in Section 2.6 fails for the problem, one should use one of the alternatives presented in the same section.

2.8.4 Failure of the Line Search

In the case that there is no singular Jacobian and the line search algorithm reduces the step size to an unacceptable small value, then this means that the computed Newton direction is poor. In order to let the line search algorithm perform better, the computation of the Newton direction must become more accurate. Furthermore, we repeat that convergence of the line search algorithm is based on an exact Jacobian. A difference approximation to a Jacobian or Jacobian-vector product is usually, but not always, sufficient.

For more background on failures of Newton's method and the line search algorithm we refer to [15].

Chapter 3

Picard Iteration

Picard iteration is also known as fixed point iteration. We first give the definition of a fixed point.

Definition 3.1. *A fixed point is a point that does not change upon application of a map or a system of differential equations. For example, a point p such that $f(p) = p$ holds for a given map $f : \mathbb{R} \to \mathbb{R}$ is called a fixed point. Points of an autonomous system of ordinary differential equations at which*

$$\begin{cases} \frac{dx_1}{dt} &= f_1(x_1, \ldots, x_n) &= 0 \\ \frac{dx_2}{dt} &= f_2(x_1, \ldots, x_n) &= 0 \\ & \vdots & \\ \frac{dx_n}{dt} &= f_n(x_1, \ldots, x_n) &= 0 \end{cases} \tag{3.1}$$

are also known as fixed points.

In this chapter we restrict ourselves to finding a fixed point of a given map $F : \mathbb{R}^n \to \mathbb{R}^n$. First, we consider the scalar case, e.g., $n = 1$.

3.1 One Dimension

To find a solution of the scalar equation $f(x) = x$, or equivently, to find a fixed point of the given map $f : \mathbb{R} \to \mathbb{R}$ the following approximation can be used. Choose a start-value x_0 and determine x_n by $x_n = f(x_{n-1})$ for $n \geq 1$. If the sequence converges to x and f is continuous, then

$$x = \lim_{n\to\infty} x_n = \lim_{n\to\infty} f(x_{n-1}) = f\left(\lim_{n\to\infty} x_{n-1}\right) = f(x). \tag{3.2}$$

From Equation (3.2) follows that x is a fixed point of $f(x)$. This approximation of the fixed point is called the *fixed point* or *Picard iteration*.

The following theorems give sufficient conditions for existence and uniqueness of a fixed point, and convergence of the sequence $\{x_n\}_{n\geq 0}$ to a fixed point. Both theorems are taken from [38].

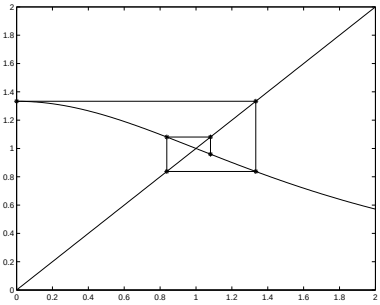

Figure 3.1: Graphical illustration of the fixed point method

Theorem 3.2. *If $f \in C[a,b]$ and $f(x) \in [a,b]$ for all $x \in [a,b]$, then f has a fixed point in $[a,b]$. Furthermore, if $f'(x)$ exists for $x \in [a,b]$ and there exists a positive constant $k < 1$ such that*

$$|f'(x)| \leq k \quad \textit{for all} \quad x \in [a,b], \tag{3.3}$$

then the fixed point in $[a,b]$ is unique.

Proof. See [38], page 86. □

Theorem 3.3. *Assume $f \in C[a,b]$, $f(x) \in [a,b]$, $x \in [a,b]$ and $|f'(x)| \leq k < 1$ for $x \in [a,b]$. The fixed point iteration converges to x for every start-value $x_0 \in [a,b]$.*

Proof. The assumptions imply that f has a unique fixed point in $[a,b]$. Using the mean-value theorem we obtain

$$|x_n - x| = |f(x_{n-1}) - f(x)| = |f'(\xi)||x_{n-1} - x| \leq k|x_{n-1} - x|. \tag{3.4}$$

By induction follows

$$\lim_{n\to\infty} |x_n - x| \leq \lim_{n\to\infty} k^n |x_0 - x| = 0. \tag{3.5}$$

We conclude that x_n converges to x. □

The fixed point iteration is illustrated in Figure 3.1 for the map

$$f(x) = \frac{4}{x^2 + 3}, \tag{3.6}$$

with start-value $x_0 = 0$.

3.2 Higher Dimensions

We start with the definition of a contraction, or contraction mapping.

Definition 3.4. *A mapping $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$ is contractive on a set $D_0 \subset D$ if there is an $\alpha < 1$ such that $\|F(x) - F(y)\| \leq \alpha\|x - y\|$ for all $x, y \in D_0$. We call such a mapping F a contraction mapping, or shorter a contraction on D_0.*

Note that the contractive property as given in the above definition is norm dependent. Clearly, a contraction is Lipschitz-continuous. The following theorem gives a basic result on the existence and uniqueness of a fixed point of a contraction. This result is known as the Banach Fixed Point Theorem.

Theorem 3.5. *Let F be a contraction mapping from a closed subset D of a Banach space S onto D. Then there exists a unique $z \in D$ such that $F(z) = z$.*

Then, the following theorem presents a criterion for convergence of the Picard iteration in more dimensions.

Theorem 3.6. *Assume that F is a contraction on a closed subset D of $\mathbb{R}^n$. The fixed point iteration*

$$x^{k+1} = F(x^k), \tag{3.7}$$

converges to a fixed point x for every start-value x^0, if the spectral radius of the Jacobian of $F(x)$ is less than one.

Proof. We obtain from the mean-value theorem that

$$\|x^{k+1} - x\| = \|F(x^k) - F(x)\| = \|J_F(\xi_{k+1})(x^k - x)\|, \tag{3.8}$$

where $\xi_{k+1} \in (x^k, x)$. For any multiplicative norm $\|\cdot\|$, the norm of $J_F(\xi_{k+1})(x^k - x)$ is less or equal to

$$\|J_F(\xi_{k+1})\|\|(x^k - x)\|. \tag{3.9}$$

By induction we obtain

$$\|x^{k+1} - x\| \leq \|J_F(\xi_{k+1})\|\|J_F(\xi_k)\| \cdots \|J_F(\xi_1)\|\|(x^0 - x)\|. \tag{3.10}$$

In order to have convergence of the fixed point iteration, $\|J_F(\xi_i)\|$ should be less than one for all $i \in [1, k+1]$. Since for every multiplicative norm holds

$$\rho(A) \leq \|A\|, \tag{3.11}$$

a neccesary condition for convergence is $\rho(J_F(x)) < 1$ in a sphere with radius $\|x^0 - x\|$ around x. □

3.3 Some Last Remarks

In this section we give some last remarks considering the fixed point iteration. To terminate the iteration it is sufficient to fulfill the condition

$$\|x^{k+1} - x^k\| \leq tol. \tag{3.12}$$

In the case that the Picard iteration does not converge, i.e., the spectral radius of the Jacobian becomes larger than one, a relaxation parameter ω can be used. The so-called Picard iteration scheme with relaxation parameter is

$$x^{k+1} = x^k + \omega(F(x^k) - x^k), \tag{3.13}$$

with $\omega > 0$.

Part IV

Linear Solvers

Chapter 1

Introduction

In Part III we saw that solving nonlinear equations iteratively needs solutions of linear systems. For example, the standard Newton iteration

$$x^{k+} = x^k - F'(x^k)^{-1}F(x^k), \tag{1.1}$$

needs in each iteration the solution of the system

$$F'(x^k)s^k = F(x^k). \tag{1.2}$$

In this part we discuss methods which solve linear algebraic systems. Generally, we distinguish direct and iterative solution methods. In this chapter we first give some examples of direct solution methods. Then it also will become clear that we focus only on iterative methods. The further contents of this chapter consists of the presentation of the basic ideas behind iterative methods.

Examples of direct solution methods are the LU factorization and the Choleski factorization. The LU factorization is based on the following observation. Suppose we have the system

$$Ax = b, \tag{1.3}$$

with A square $(n \times n)$ and nonsingular. It can be shown that there exist Gauß transformations $M_1, \ldots, M_{n-1}$ such that

$$M_{n-1} \cdots M_1 A = U, \tag{1.4}$$

with U an upper triangular matrix. Then

$$A = (M_{n-1} \cdots M_1)^{-1}U = LU, \tag{1.5}$$

with $L = (M_{n-1} \cdots M_1)^{-1}$. It can be shown that L is a lower triangular matrix.

If the LU factorization of A is obtained, then the solution of $Ax = b$ is easy to compute. The solution of $LUx = b$ can be split into two parts, i.e., first the solution of the lower triangular system $Ly = b$ and then the solution of the upper triangular system $Ux = y$. To compute an LU factorization of A takes $2n^3/3$ flops. Solving $Ly = b$ and $Ux = y$ requires $2n^2$ flops.

In the case that A is symmetric positive definite (SPD), then the factorization

$$A = GG^T, \tag{1.6}$$

with G a lower triangular matrix, exists. This factorization is known as the Choleski factorization and costs $n^3/3$ flops.

Problems arising from discretized PDEs lead in general to large sparse systems of equations. Direct solutions methods can be impractical if A is large and sparse, because the L and U can be dense. This is especially the case for 3D problems. We then need a considerable amount of memory and the solving the triangular system can many floating point operations. Therefore, we restrict ourselves to iterative methods to solve linear systems. The amount of work to solve $Ax = b$ iteratively depends on the number of iterations and the amount of work per iteration.

The basic idea behind solving $Ax = b$, with A invertible, iteratively is the following. Construct a splitting $A = B + (A - B)$, so that B is "easy invertible". The original linear system can then be rewritten as

$$Bx + (A - B)x = b. \tag{1.7}$$

By putting

$$Bx^{i+1} + (A - B)x^i = b, \tag{1.8}$$

we introduce a sequence of approximate solutions $\left(x^i\right)_{i=0}^{\infty}$. Equation (1.8) can be rewritten as

$$x^{i+1} = (I - B^{-1}A)x^i + B^{-1}b, \tag{1.9}$$

or equivalently as

$$x^{i+1} = Qx^i + s, \tag{1.10}$$

with $Q = I - B^{-1}A$ and $s = B^{-1}b$. We call an iterative process *consistent* if the matrix $I - Q$ is non-singular and if $(I - Q)^{-1}s = A^{-1}b$. If the iteration process is consistent, then the equation $(I - Q)x = s$ has the unique solution $x^{i\to\infty} \equiv x$. We call an iterative process *convergent* if the sequence $x^0, x^1, x^2, \ldots$ converges to a limit *independent* of the start-value x^0.

For example, the Jacobi method has the splitting $A = D - C$, with D a diagonal matrix and C a matrix that contains zeros on the main diagonal. By taking B equal to D the Jacobi method reads

$$Dx^{i+1} = Cx^i + b. \tag{1.11}$$

In the Gauß-Seidel method we also split up $C = C_1 + C_2$, with a lower triangular matrix C_1 and an upper triangular matrix C_2. Then, by taking $B = D - C_1$ the Gauß-Seidel method reads

$$(D - C_1)x^{i+1} = C_2 x^i + b. \tag{1.12}$$

In our application of CVD we get after discretization and applying a time integration method a huge nonlinear system to solve. The size of this nonlinear system depends on

1. the spatial dimension d ($d = 1, 2, 3$),
2. the number of chemical species,
3. solving the nonlinear equations coupled or decoupled.

As mentioned above, to solve this nonlinear system also linear systems have to be solved. These are huge and sparse linear systems.

Therefore, iterative solution methods are more efficient than direct methods, especially in the three-dimensional case.

Using these basic iterative methods as described above a class of efficient solution methods can be derived. This class, the class of the so-called Krylov subspace methods will be discussed in the next chapter. We conclude with presenting preconditioner techniques in Chapter 3, which are used to accelerate Krylov subspace methods.

Chapter 2

Krylov Subspace Methods

In Chapter 1 the basic iterative method is defined to approximate the solution of a linear system $Ax = b$ iteratively by

$$x^{i+1} = (I - B^{-1}A)x^i + B^{-1}b, \tag{2.1}$$

where B is an "easy invertible" matrix coming from the splitting $A = B + (A - B)$. We also remark that the iteration process (2.1) needs a start approximation x^0. Rewriting (2.1) gives

$$x^{i+1} = x^i + B^{-1}(b - Ax^i) = x^i + B^{-1}r^i, \tag{2.2}$$

where $r^i = b - Ax^i$ is called the i^{th} residual. Using this notation, the first steps of the iteration process are

$$\begin{aligned}
x^0 &= x^0, \\
x^1 &= x^0 + B^{-1}r^0, \\
x^2 &= x^1 + B^{-1}r^1 = \\
&= x^0 + B^{-1}r^0 + B^{-1}(b - Ax^1) = \\
&= x^0 + B^{-1}r^0 + B^{-1}(b - A(x^0 + B^{-1}r^0)) = \\
&= x^0 + (2B^{-1} - B^{-1}AB^{-1})r^0, \\
&\vdots
\end{aligned}$$

which implies that

$$x^i \in x^0 + \operatorname{span}\left\{B^{-1}r^0, B^{-1}A(B^{-1})r^0, \ldots, (B^{-1}A)^{i-1}(B^{-1})r^0\right\}. \tag{2.3}$$

The subspace $K^i(A, r^0) := \operatorname{span}\{r^0, Ar^0, \ldots, A^{i-1}r^0\}$ is called the Krylov-space of dimension i corresponding to the matrix A and initial residual r^0. Each approximation x^i, $i = 1, 2, \ldots$, that is computed by a basic iterative method is an element of $x^0 + K^i(M^{-1}A, M^{-1}r^0)$.

This chapter is divided into two parts. In the first part the Conjugate Gradients method will be treated, which is a Krylov Subspace method for symmetric matrices. In the other part of this chapter we discuss Krylov subspace methods for general matrices.

2.1 Krylov Subspace Methods for Symmetric Matrices

In this section we present Krylov subspace methods for solving

$$Ax = b, \tag{2.4}$$

where A is a symmetric matrix. The CG method is the most prominent iterative method for solving linear systems (2.4) in the case that A is SPD. CG will be derived from the Lanczos Algorithm, which will be presented in the next section.

2.1.1 Arnoldi's Method and the Lanczos Algorithm

The Arnoldi Algorithm is a procedure for building an orthonormal basis of the Krylov subspace $K^i(A, r^0)$. It is given in Algorithm 2.1, which is taken from [28].

Algorithm 2.1 Arnoldi

Choose a vector v_1 of norm 1,
for $j = 1, 2, \ldots, m$ **do**
  Compute $h_{ij} = (Av_j, v_i)$ for $i = 1, 2, \ldots, j$,
  Compute $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$,
  $h_{j+1,j} = \|w_j\|_2$,
  if $h_{j+1,j} = 0$ **then**
    Stop,
  end if
  $v_{j+1} = w_j / h_{j+1,j}$
end for

A few simple properties of the algorithm are given in Proposition 2.1 and 2.2.

Proposition 2.1. *Assume that Algorithm 2.1 does not stop before the* m^{th} *step. Then the vectors* $v_1, v_2, \ldots, v_m$ *from an orthonormal basis of the Krylov subspace*

$$K^m(A, r^0) = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}. \tag{2.5}$$

Proposition 2.2. *Denote by* V_m *the* $n \times n$ *matrix with column vectors* $v_1, \ldots, v_m$ *, by* $\bar{H}_m$ *the* $(m+1) \times n$ *Hessenberg matrix whose nonzero entries*

h_{ij} are defined by Algorithm 2.1, and by H_m the matrix obtained from $\bar{H}_m$ by deleting its last row. Then the following relations hold:

$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^T && (2.6)\\ &= V_{m+1}\bar{H}_m, && (2.7)\\ V_m^T A V_m &= H_m. && (2.8) \end{aligned}$$

For the proofs of Proposition 2.1 and 2.2 we refer to [28].

In the case that the matrix A is symmetric the Hessenberg H_m will become symmetric tridiagonal, see [28, Theorem 6.2]. The standard notation used to describe the Lanczos Algorithm is obtained by setting

$$\alpha_j \equiv h_{jj}, \quad \beta_j \equiv h_{j-1,j}, \tag{2.9}$$

and if T_m denotes the resulting H_m matrix, it is of the form

$$\begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_1 & \beta_2 & & \\ & & \ddots & & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{pmatrix}. \tag{2.10}$$

This leads to the following variant of Arnoldi's Algorithm (see Algorithm 2.1), which is called the Lanczos Algorithm. It is given as Algorithm 2.2.

Algorithm 2.2 Lanczos Algorithm

Choose an initial vector v_1 of norm 1. Set $\beta_1 \equiv 0$ and $v_0 \equiv 0$,
for $j = 1, 2, \ldots, m$ **do**
  $w_j = AV_j - \beta_j v_{j-1}$,
  $\alpha_j = (w_j, v_j)$,
  $w_j = w_j - \alpha_j v_j$,
  $\beta_{j+1} = \|w_j\|_2$,
  if $\beta_{j+1} = 0$ **then**
    Stop
  end if
  $v_{j+1} = w_j/\beta_{j+1}$
end for

Algorithm 2.2 garantuees that in **exact** algorithmic the vectors v_i, $i = 1, 2, \ldots, m$, are orthogonal. In practice, when we do not compute in exact algorithmic, exact orthogonality of these vectors is only observed at the beginning of the process. At some point the v_i's start to loose their global orthogonality rapidly. For more information on this subject we refer to [28].

2.1.2 The Conjugate Gradient Algorithm

As mentioned before, the Conjugate Gradient Method, shortly CG, is the best known iterative techniques for solving sparse symmetric positive definite linear systems. Shortly described, CG is a realization of an orthogonal projection technique onto the Krylov subspace $K^m(A, r^0)$, where r^0 is the initial residual.

We will derive the CG method form the Lanczos Method for Linear Systems. This Lanczos Method for Linear Systems approximates the solution of $Ax = b$, with $A^T = A$, by an orthogonal projection onto $K^m(A < r^0)$. Given an initial guess x^0 to the linear system $Ax = b$ and the Lanczos vectors $v_1, \ldots, v_m$ together with the tridiagonal matrix T_m, the orthogonal projection is given as

$$x^m = x^0 + V_m y_m, \quad y_m = T_m^{-1}(\beta e_1). \tag{2.11}$$

The actual algorithm is given as Algorithm 2.3.

Algorithm 2.3 Lanczos Method for Linear Systems

1: Compute $r^0 = b - Ax^0$, $\beta = \|r^0\|$ and $v_1 = r^0/\beta$,
2: **for** $j = 1, 2, \ldots, m$ **do**
3: $w_j = AV_j - \beta_j v_{j-1}$, (If $j = 1$ set $\beta_1 v_0 \equiv 0$)
4: $\alpha_j = (w_j, v_j)$,
5: $w_j = w_j - \alpha_j v_j$,
6: $\beta_{j+1} = \|w_j\|_2$,
7: **if** $\beta_{j+1} = 0$ **then**
8: Set $m = j$ and go to 10
9: **end if**
10: Set $T_m = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$ and $V_m = [v_1, \ldots, v_m]$,
11: Compute $y_m = T_m^{-1}(\beta e_1)$ and $x^m = x^0 + V_m y_m$.
12: **end for**

Remark from Algorithm 2.3 that the approximate solution is given as

$$x^m = x^0 + V_m y_m = x^m = x^0 + V_m U_m^{-1} L_m^{-1}(\beta e_1), \tag{2.12}$$

where we used the LU factorization of $T_m = L_m U_m$. Letting

$$P_m = V_m U_m^{-1} \quad \text{and} \quad z_m = L_m^{-1}(\beta e_1), \tag{2.13}$$

gives $x^m = x^0 + P_m z^m$. As remarked in [28], the last column of P_m, denoted by p_m, can be updated from the previous p_i's and v_m by

$$p_m = \frac{1}{\eta_m}(v_m - \beta_m p_{m-1}), \tag{2.14}$$

with

$$\lambda_m = \frac{\beta_m}{\eta_{m-1}} \quad \text{and} \quad \eta_m = \alpha_m - \lambda_m \beta_m. \tag{2.15}$$

In addition, as shown in [28], we have

$$z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} \quad \text{with} \quad \zeta_m = -\lambda_m \zeta_{m-1} \quad \text{and} \quad \zeta_1 = \beta. \tag{2.16}$$

As a result, x_m can be updated at each step as $x_m = x_{m-1} + \zeta_m p_m$. In the next proposition we give some properties of the Lanczos algorithm.

Proposition 2.3. *Let $r^m = b - Ax^m$, $m = 0, 1, \ldots$, be the residual vectors produced by Algorithm 2.3 and p_m be the auxiliary vectors as given above. Then,*

1. *Each residual vector r^m is such that $r^m = \sigma_m v_{m+1}$, where σ_m is a certain scalar. As a result, the residual vectors are orthogonal to each other.*
2. *The auxiliary vectors p_i form an A-conjugate set, i.e., $(Ap_i, p_j) = 0$, for $i \neq j$.* [1]

Proof. For a proof we refer to [28]. □

From the above proposition we can derive a modified version of Algorithm 2.3 by imposing orthogonality and conjugate conditions. This modified version of Algorithm 2.3 is the CG method.

First we remark that

$$x^{j+1} = x^j + \alpha_j x^j. \tag{2.17}$$

Therefore, the residual vectors must satisfy

$$r^{j+1} = r^j - \alpha_j A p^j. \tag{2.18}$$

To have the residual vectors orthogonal, it is necessary to impose $(r^j - \alpha_j A p^j, p^j) = 0$. It follows that α_j must be equal to

$$\alpha_j = \frac{(r^j, r^j)}{Ap^j, r^j)}. \tag{2.19}$$

Using Proposition 2.3 and Equation (2.14) we observe that the next search direction p^{j+1} is a linear combination of r^{j+1} and p^j. After rescaling the p vectors appropriately, it follows that

$$p^{j+1} = r^{j+1} + \beta_j p^j. \tag{2.20}$$

A first consequence of the above relation is

$$(Ap^j, r^j) = (Ap^j, p^j - \beta_{j-1} p^{j-1}) = (Ap^j, p^j), \tag{2.21}$$

[1] To ensure that (Ax, x) is an inner product, A must be SPD

because Ap^j is orthogonal to p^{j-1}. Remark that (2.19) becomes $\alpha_j = (r^j, r^j)/(Ap^j, p^j)$. A second consequence of (2.20) is the following. Since p^{j+1} is orthogonal to Ap^j, we can derive that β_j in (2.20) must be equal to

$$\beta_j = -\frac{(r^{j+1}, Ap^j)}{(p^j, Ap^j)} = \frac{1}{\alpha_j}\frac{(r^{j+1}, (r^{j+1} - r^j))}{(Ap^j, p^j)} = \frac{(r^{j+1}, r^{j+1})}{(r^j, r^j)}. \tag{2.22}$$

Putting these relations together gives Algorithm 2.4, i.e., the CG algorithm.

Algorithm 2.4 Conjugate Gradient

1: Compute $r^0 = b - Ax^0$ and $p^0 = r^0$,
2: **for** $j = 1, 2, \ldots,$ until convergence **do**
3: $\alpha_j = (r^j, r^j)/(Ap^j, p^j)$,
4: $x^{j+1} = x^j + \alpha_j p^j$,
5: $r^{j+1} = r^j - \alpha_j Ap^j$,
6: $\beta_j = (r^{j+1}, r^{j+1})/(r^j, r^j)$,
7: $p^{j+1} = r^{j+1} + \beta_j p^j$.
8: **end for**

2.2 Krylov Subspace Methods for General Matrices

In the previous section we have discussed the CG method. This Krylov subspace method can only be used to solve (sparse) symmetric positive definite linear systems. In this section we discuss the best known Krylov subspace methods for general (real) sparse linear systems, i.e., we consider Krylov subspace methods to solve $Ax = b$, with A an $n \times n$ nonsingular real matrix.

Considering the case of symmetric positive definite linear systems, we have seen that CG has the properties:

- Krylov subspace based on A,
- optimality: the residual is minimized in a certain norm,
- short recurrences; only the results of one foregoing step are necessary, work and memory do not increase for an increasing number of iterations.

In [8] it has been shown that it is impossible to obtain a Krylov subspace method for general matrices, which satisfies the above properties. From this we conclude that a Krylov method for general matrices has either an optimality property but long recurrences, or no optimality and short recurrences. A method that has an optimality property and short recurrences is not a Krylov subspace method.

It appears that there are essentially three different ways to solve non-symmetric real linear systems:

1. Solve the normal equations $A^T Ax = A^T b$ with CG,
2. Construct a basis for the Krylov subspace by a 3-term bi-orthogonality relation,
3. Make all the residuals explicitly orthogonal in order to have an orthogonal basis for the Krylov subspace.

In this report only the last two classes will be treated. For more information on the first class of methods we refer to [22, 28].

2.2.1 BiCG Type Methods

In the previous chapter we saw that the CG method is derived from the Lanczos Algorithm, i.e. Algorithm 2.2. It appears that there are various generalizations of the Lanczos Algorithm for general matrices. One of these generalizations, the Arnoldi procedure (Algorithm 2.1), has already been treated. Another extension of Lanczos to general matrices is the Lanczos Bi-orthogonalization Algorithm. The concept of the Lanczos Bi-orthogonalization Algorithm is different in concept from Arnoldi in the sense that it relies on bi-orthogonal sequences instead of orthogonal sequences.

In this section we shortly present the Lanczos Bi-orthogonalization Algorithm and mention how Bi-Conjugate Gradient (BiCG) can be derived from it. Furthermore, we present the basic ideas behind Conjugate Gradient Squared (CGS) and Bi-Conjugate Gradient STABilized (Bi-CGSTAB) The latter algorithm, i.e. Bi-CGSTAB, is together with GMRES one of the best known iterative methods for general sparse real linear systems.

Lanczos Bi-Orthogonalization

The algorithm proposed by Lanczos for non-symmetric matrices builds a pair of bi-orthogonal bases for the two subspaces

$$K^m(A, v_1) = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}, \tag{2.23}$$

and

$$K^m(A^T, w_1) = \text{span}\{w_1, A^T w_1, \ldots, (A^T)^{m-1}w_1\}. \tag{2.24}$$

The algorithm that achieves this is Algorithm 2.5. There can be proved that if Algorithm 2.5 does not break down before step m, then the vectors v_i and w_j, $i, j = 1, 2, \ldots, m$, form a bi-orthogonal system.

Comparing this algorithm with Arnoldi's method we remark the following. From a practical point of view, the Lanczos Algorithm has an advantage over Arnoldi's method, because it requires only a few vectors of storage, if

Algorithm 2.5 Lanczos Bi-Orthogonalization Procedure

1: Choose two vectors v_1, w_1 such that $(v_1, w_1) = 0$,
2: Set $\beta_1 = \delta_1 = 0$ and $w_0 = v_0 = 0$,
3: **for** $j = 1, 2, \ldots, m$ **do**
4: $\alpha_j = (Av_j, w_j)$,
5: $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$,
6: $\hat{w}_{j+1} = A^T w_j - \alpha_j w_j - \beta_j w_{j-1}$,
7: $\delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1})|^{1/2}$,
8: **if** $\delta_{j+1} = 0$ **then**
9: Stop.
10: **end if**
11: $\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})/\delta_{j+1}$,
12: $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$,
13: $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$,
14: **end for**

no re-orthogonalization is performed. In particular, six vectors of length n and a tridiagonal matrix are needed for storage. This amount of storage is independent of m.

Bi-Conjugate Gradient

The Bi-Conjugate Gradient (BiCG) procedure can be derived from Algorithm 2.5 in exactly the same way as the CG Algorithm was derived from Algorithm 2.2. Implicitly, the BiCG algorithm solves not only the original system $Ax = b$, but also a dual linear system $A^T x^* = b^*$. The dual system is often ignored in the algorithm.

The BiCG procedure is a projection process onto

$$K^m(A, v_1) = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}, \tag{2.25}$$

orthogonally to

$$K^m(A^T, w_1) = \text{span}\{w_1, A^T w_1, \ldots, (A^T)^{m-1} w_1\}. \tag{2.26}$$

The vector v_1 is taken as usual $v_1 = r_0/\|r_0\|_2$. The vector w_1 is arbitrary, provided that $(v_1, w_1) \neq 0$. The algorithm is given as Algorithm 2.6

Conjugate Gradient Squared

Each step of BiCG requires a matrix-vector product with both A and A^T. Secondly, observe that the vectors p_i^* and w_j, generated by A^T do not contribute directly to the solution. Due to this observations, one might ask, is it possible to bypass the use of A^T and still generate iterates related to BiCG ? The answer is given as the Conjugate Gradient Squared (CGS) method.

Algorithm 2.6 Bi-Conjugate Gradient

1: Compute $r_0 = b - Ax_0$. Choose r_0^* such that $(r_0, r_0^*) \neq 0$.
2: Set $p_0 = r_0$ and $p_0^* = r_0^*$,
3: **for** $j = 1, 2, \ldots,$ until convergence **do**
4: $\quad \alpha_j = (r_j, r_j^*)/(Ap_j, p_j^*)$,
5: $\quad x_{j+1} = x_j + \alpha_j p_j$,
6: $\quad r_{j+1} = r_j - \alpha_j A p_j$,
7: $\quad r_{j+1}^* = r_j^* - \alpha_j A^T p_j^*$,
8: $\quad \beta_j = (r_{j+1}, r_{j+1}^*)/(r_j, r_j^*)$,
9: $\quad p_{j+1} = r_{j+1} - \beta_j p_j$,
10: $\quad p_{j+1}^* = r_{j+1}^* - \beta_j p_j^*$,
11: **end for**

The CGS method was developed in 1984 by Sonneveld [31]. In the BiCG method, the residual r_j can be regarded as the product of r_0 and the j^{th} degree polynomial $\mathcal{R}$ in A, i.e.,

$$r_j = \mathcal{R}_j(A) r_0, \tag{2.27}$$

satisfying the constraint $\mathcal{R}_j(0) = 1$. This same polynomial satisfies also

$$r_j^* = \mathcal{R}_j(A^T) r_0^*, \tag{2.28}$$

by construction of the BiCG method. Similarly, the conjugate-direction polynomial $\mathcal{P}_j(t)$ satisfies the following expressions

$$p_j = \mathcal{P}_j(A) r_0 \quad \text{and} \quad p_j^* = \mathcal{P}_j(A^T) r_0^*. \tag{2.29}$$

Also, note that the scalar α_j in BiCG is given as

$$\alpha_j = \frac{(r_j, r_j^*)}{(Ap_j, p_j^*)} = \frac{(\mathcal{R}_j(A) r_0, \mathcal{R}_j(A^T) r_0^*)}{(A\mathcal{P}_j(A) r_0, \mathcal{P}_j(A^T) r_0^*)} = \frac{(\mathcal{R}_j^2(A) r_0, r_0^*)}{(A\mathcal{P}_j^2(A) r_0, r_0^*)}, \tag{2.30}$$

which indicates that if it is possible to obtain a recursion for the vectors $\mathcal{R}_j^2(A) r_0$ and $\mathcal{P}_j^2(A) r_0$, then computing α_j and β_j cause no problem.

Expression (2.30) suggests that if $\mathcal{R}_j(A)$ reduces r_0 to a smaller vector r_j, then it might be advantageous to apply this 'contraction'-operator twice and compute $\mathcal{R}_j^2(A) r_0$. The iteration coefficients can still be recovered from these vectors, and it turns out to be easy to find the corresponding approximations for x. This approach is called the Conjugate Gradient Squared method. See Algorithm 2.7. For a complete derivation we refer to [28].

Observe that in Algorithm 2.7 are no matrix-vector products with A^T. Instead, two matrix-vector products with A are performed in each step. In general, one should expect the resulting algorithm to converge twice as fast as BiCG. Therefore, what has been done is to replace the matrix-vector products with A^T with more useful work.

Algorithm 2.7 Conjugate Gradient Squared

Compute $r_0 = b - Ax_0$. Choose r_0^* arbitrary,
Set $p_0 = u_0 = r_0$,
for $j = 1, 2, \ldots,$ until convergence **do**
  $\alpha_j = (r_j, r_0^*)/(Ap_j, r_0^*)$,
  $q_i = u_j - \alpha_j A p_j$
  $x_{j+1} = x_j + \alpha_j(u_j + q_j)$,
  $r_{j+1} = r_j - \alpha_j A()u_j + q_j)$,
  $\beta_j = (r_{j+1}, r_0^*)/(r_j, r_0^*)$,
  $u_{j+1} = r_{j+1} - \beta_j q_j$,
  $p_{j+1} = u_{j+1}^* - \beta_j(q_j + \beta_j p_j)$,
end for

A drawback of the CGS method is the following. Since the polynomials are squared, rounding errors tend to have a more damaging effect than in the standard BiCG algorithm.

Bi-CGSTAB

The CGS algorithm is based on squaring the residual polynomial, and, in cases of irregular convergence, this may lead to substantial build-up of rounding errors, or possibly even overflow. The Bi-Conjugate Gradient Stabilized (Bi-CGSTAB) Algorithm is a variation of CGS that has a remedy for this difficulty. During the development of Bi-CGSTAB one used a residual of the form

$$r_j = \mathcal{Q}_j(A)\mathcal{R}_j(A)r_0, \tag{2.31}$$

instead of $r_j = \mathcal{R}_j^2(A)r_0$. In (2.31) the polynomial $\mathcal{Q}_j(A)$ is a new polynomial which is defined recursively at each step with the goal of 'stabilizing' or 'smoothing' the convergence behavior of the original algorithm. In particular, $\mathcal{Q}_j$ is defined recursively by

$$\mathcal{Q}_{j+1}(t) = (1 - \omega_j t)\mathcal{Q}_j(t). \tag{2.32}$$

The above recursion is equivalent with

$$\mathcal{Q}_{j+1}(t) = (1 - \omega_0 t)(1 - \omega_1 t)\cdots(1 - \omega_j t), \tag{2.33}$$

where the scalars ω_i, $i = 0, 1, \ldots, j$ have to be determined.

For the determination of the scalars ω_i, $i = 0, 1, \ldots, j$, and the complete derivation of the Bi-CGSTAB algorithm we refer to [28], pp 216-219. The Bi-CGSTAB algorithm is given as Algorithm 2.8.

Algorithm 2.8 Bi-CGSTAB

1: Compute $r_0 = b - Ax_0$. Choose r_0^* arbitrary,
2: Set $p_0 = r_0$,
3: **for** $j = 1, 2, \ldots,$ until convergence **do**
4: $\quad \alpha_j = (r_j, r_0^*)/(Ap_j, r_0^*)$,
5: $\quad s_i = r_j - \alpha_j A p_j$
6: $\quad w_j = (As_j, s_j)/(As_j, As_j)$,
7: $\quad x_{j+1} = x_j + \alpha_j p_j + w_j s_j$,
8: $\quad r_{j+1} = s_j - w_j A s_j$,
9: $\quad \beta_j = \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \frac{\alpha_j}{w_j}$,
10: $\quad p_{j+1} = r_{j+1} + \beta_j (p_j - w_j A p_j)$,
11: **end for**

2.2.2 GMRES Methods

The Generalized Minimum Residual Method (GMRES) is a projection method on the Krylov subspace $K^m(A, v_1)$, with $v_1 = r_0/\|r_0\|_2$. This technique minimizes the residual norm over all vectors in $x_0 + K^m(A, v_1)$. In order to do this, GMRES generates a sequence of orthogonal vector with long recurrences due to the non-symmetry. GMRES uses a variant of Arnoldi's Algorithm (Algorithm 2.1) to find an orthonormal basis. We first give the idea behind the GMRES algorithm.

GMRES Derivation

Any vector x in $x_0 + K^m(A, v_1)$ can be written as

$$x = x_0 + V_m y, \tag{2.34}$$

where y is an m-vector. Define

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2, \tag{2.35}$$

and remark that

$$\begin{aligned} b - Ax &= b - A(x_0 + V_m y) = && (2.36)\\ &= r_0 - AV_m y = && (2.37)\\ &= \beta v_1 - V_{m+1}\bar{H}_m y = && (2.38)\\ &= V_{m+1}(\beta e_1 - \bar{H}_m y), && (2.39) \end{aligned}$$

where we used Proposition 2.2, $\beta = \|r_0\|_2$ and $e_1 = (1, 0, \ldots, 0)^T$. Since the column-vectors of V_{m+1} are orthonormal, we obtain

$$J(y) = \|b - A(x_0 + V_m y)\|_2 = \|\beta e_1 - \bar{H}_m y\|_2. \tag{2.40}$$

The GMRES approximation is the vector of $x_0+K^m(A,v_1)$ which minimizes $J(y)$. Using (2.40) this approximation can easily be obtained by

$$x_m = x_0 + V_m y_m \quad \text{where} \quad y_m = \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2. \tag{2.41}$$

Observe that y_m is the solution of an $(m+1)\times m$ least-squares problem, which is inexpensive to solve (for m typically small).

2.2.3 The GMRES Algorithm

Summarizing the above remarks and results gives the following algorithm.

Algorithm 2.9 GMRES

1: Compute $r^0 = b - Ax^0$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$
2: Define the $(m+1)\times m$ matrix $\bar{H}_m = \{h_{ij}\}_{1\le i\le m+1, 1\le j\le m}$. Set $\bar{H}_m = 0$.
3: **for** $j = 1, 2, \ldots, m$ **do**
4: Compute $w_j = Av_j$,
5: **for** $i = 1, \ldots, j$ **do**
6: $h_{ij} = (w_j, v_i)$,
7: $w_j = w_j - h_{ij} v_i$,
8: **end for**
9: $h_{j+1,i} = \|w_j\|_2$,
10: **if** $h_{j+1,j} = 0$ **then**
11: Set $m = j$ and go to 15,
12: **end if**
13: $v_{j+1} = w_j / h_{j+1,j}$
14: **end for**
15: Compute $y_m = \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_M y_m$.

We remark that in Algorithm 2.9 the lines 3 to 14 represent Arnoldi's Algorithm for finding an orthonormal basis of $K^m(A, v_1)$.

If Algorithm 2.9 is examined carefully, we observe that GMRES has a breakdown when $h_{j+1,j} = 0$ at a given step j. In this situation the next Arnoldi vector cannot be generated. However, in this situation, the residual vector is equal to zero and GMRES will deliver the exact solution at this step. For a proof of this we refer to [28, Proposition 6.10].

The last remark considering Algorithm 2.9 is the following. In the case the iteration number m is large, the GMRES algorithm becomes impractical by lack of memory and increasing computational requirements. This follows directly from the fact that during the Arnoldi steps (lines 3-14 of Algorithm 2.9) the number of vectors to be stored increases. To remedy this problem one can think of *restarting* or *truncating* the algorithm. More information on restarting and truncating can be found in [28].

2.3 Stopcriterium

In most of the iterative methods given in this chapter the following statement is included :

for $j = 1, 2, \ldots,$ until convergence **do**.

To specify the statement 'until convergence' various stop criteria can be chosen. It depends on each practical situation which of them should be chosen for an accurate solution. We give two standard stop criteria. Other criteria can be found in for example [22, 29].

- Criterion 1 : $\|r_j\|_2 \leq \varepsilon$

  The main disadvantage of this stop criterion is that it is not scaling invariant. By this we mean that if $\|r_j\|_2 \leq \varepsilon$, then this does not hold for $\|100 r_j\|_2$, although the accuracy for x_i remains the same,

- Criterion 2 : $\frac{\|r_j\|_2}{\|b\|_2} \leq \varepsilon$

  In this criterion the norm of the residual is small in comparison with the norm of the right-hand side. Replacing ε by $\varepsilon / K_2(A)$ gives that the relative error in x is less than ε, i.e.

$$\frac{\|x - x_i\|_2}{\|x\|_2} \leq K_2(A) \frac{\|r_j\|_2}{\|b\|_2} \leq \varepsilon. \tag{2.42}$$

Chapter 3

Precondition Techniques

Krylov subspace methods for solving (sparse) linear systems are well founded theoretically, but appear to suffer from slow convergence from typical applications as CFD. In this chapter we first describe preconditioning without being specific about the particular preconditioners used. In Section 3.2 we will give some widely used preconditioners as, for example, ILU.

Efficiency and robustness of iterative methods can be improved by using preconditioning techniques. In one sentence, preconditioning can be described as :

> Transforming the original linear system into one which has the same solution, but which is likely to be easier to solve with an iterative solver.

In practical applications, the reliability of iterative solution methods depends more on the quality of the preconditioner than on the particular Krylov subspace accelerator used.

3.1 Preconditioned Iterations

In this section we discuss the preconditioned versions of the Krylov subspace methods of the previous chapter, using a generic preconditioner. We will mainly follow Reference [28]. We start with the Preconditioned Conjugate Gradient method.

3.1.1 Preconditioned Conjugate Gradient

Consider a linear system $Ax = b$, whereby A is symmetric positive definite. Furthermore, we assume that a preconditioner M is available, which approximates A in some yet-undefined sense. This preconditioner M is also to be assumed symmetric positive definite. From a practical point of view, the

only requirement for M is that linear systems $Mx = \tilde{b}$ are inexpensive to solve. This is a neccesary requirement, because preconditioned algorithms will require a linear system solution of $Mx = \tilde{b}$ at each step. Then, we solve the left preconditioned system

$$M^{-1}Ax = M^{-1}b. \tag{3.1}$$

Note that in general this system is not symmetric. To preserve symmetry we introduce the M-inner product.

The M-inner product is defined as

$$(x, y)_M = (Mx, y) = (x, My). \tag{3.2}$$

By observing that

$$\begin{aligned} (M^{-1}Ax, y)_M &= (Ax, y) = (x, Ay) = \\ &= (x, M(M^{-1}A)y) = (x, M^{-1}Ay)_M, \end{aligned} \tag{3.3}$$

we see that $M^{-1}A$ is self-adjoint for the M-inner product. As an alternative, we replace the Euclidean inner product in the CG algorithm by the M-inner product.

We now will rewrite the CG algorithm for the M-inner product. Therefore we distinguish the original residue $r_j = b - Ax_j$ and the preconditioned residual $z_j = M^{-1}r_j$. For the preconditioned system we then obtain the following sequence of parameters (compare with Algorithm 2.4) :

1. $\alpha_j = (z_j, z_j)_M / (M^{-1}Ap_j, p_j)_M$,
2. $x_{j+1} = x_j + \alpha_j p_j$,
3. $r_{j+1} = r_j - \alpha_j Ap_j$ and $z_{j+1} = M^{-1}r_{j+1}$,
4. $\beta_j = (z_{j+1}, z_{j+1})_M / (z_j, z_j)_M$,
5. $p_{j+1} = z_{j+1} + \beta_j p_j$.

The M-inner products $(z_j, z_j)_M$ and $(M^{-1}Ap_j, p_j)_M$ do not have to be computed explicitly, i.e.

$$(z_j, z_j)_M = (r_j, z_j) \quad \text{and} \quad (M^{-1}Ap_j, p_j)_M = (Ap_j, p_j). \tag{3.4}$$

Using the latter observation, we obtain the Preconditioned Conjugate Gradient Algorithm, given as Algorithm 3.1.

Algorithm 3.1 Preconditioned Conjugate Gradient

Compute $r_0 = b - Ax_0$, $z_0 = M^{-1}r_0$ and $p_0 = z_0$,
for $j = 0, 1, \ldots,$ until convergence **do**
  $\alpha_j = (r_j, z_j)/(Ap_j, p_j)$,
  $x_{j+1} = x_j + \alpha_j p_j$,
  $r_{j+1} = r_j - \alpha_j A p_j$,
  $z_{j+1} = M^{-1} r_{j+1}$,
  $\beta_j = (r_{j+1}, z_{j+1})/(r_j, z_j)$,
  $p_{j+1} = z_{j+1} + \beta_j p_j$.
end for

Left, Right and Split Preconditionering

To transform the linear system $Ax = b$ with a preconditioner M, such that the transformed system is asier'to solve, we have three options:

1. Left Preconditioning: Solve $M^{-1}Ax = M^{-1}b$,
2. Right Preconditioning: Solve $AM^{-1}u = b$ with $u = Mx$,
3. Split Preconditioning: If M is SPD, then there exists a matrix L such that $M = LL^T$. Solve $L^{-1}AL^{-T}u = b$ with $x = L^T u$.

The Left Preconditioning is already given in (3.1) and used to determine the Preconditioned CG Algorithm. Right Preconditioning has the advantage that the original residual $r_i = b - Ax_i$ is computed implicitly by $b - AM^{-1}u_i = b - Ax_i$. An advantage of Split Preconditioning is that the linear system is symmetric.

Finally, we remark that the three options given above can be used to construct preconditioned iterative solution methods. In this report we give preconditioned versions of

- (Left) Preconditioned CG,
- Left Preconditioned GMRES,
- Right Preconditioned GMRES, and
- (Left) Preconditioned Bi-CGSTAB.

For more information on preconditioned iterative methods we refer to [28, 29]

3.1.2 Preconditioned GMRES

As for preconditioning the CG method, we have for the GMRES method again the three options for applying the preconditioning operation, i.e. left,

right and split preconditioning. In this section we treat the left and right preconditioning of GMRES. There will be a fundamental difference with respect to the right preconditioning.

Left Preconditioned GMRES

Left Preconditioned GMRES solves the left preconditioned linear system

$$M^{-1}Ax = M^{-1}b, \tag{3.5}$$

with A and M general matrices. The straightforward application of GMRES to (3.5) gives Algorithm 3.2, i.e., the Left Preconditioned GMRES Algorithm.

Algorithm 3.2 Left Preconditioned GMRES

1: Compute $r_0 = M^{-1}(b - Ax_0)$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$
2: **for** $j = 1, 2, \ldots, m$ **do**
3: Compute $w = M^{-1}Av_j$,
4: **for** $i = 1, \ldots, j$ **do**
5: $h_{ij} = (w, v_i)$,
6: $w = w - h_{ij}v_i$,
7: **end for**
8: $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$,
9: **end for**
10: Define $V_m = [v_1, \ldots, v_m]$ and $\bar{H}_m = (h_{i,j})_{1 \le i \le j+1, 1 \le j \le m}$,
11: Compute $y_m = \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$.
12: If satisfied Stop, else set $x_0 = x_m$ and go to 1.

In Algorithm 3.2 an orthogonal basis is constructed for the left preconditioned Krylov subspace

$$\text{span} = \{r_0, M^{-1}Ar_0, \ldots, (M^{-1}A)^{m-1}r_0\}. \tag{3.6}$$

Furthermore, we remark that in Algorithm 3.2 all residuals and their norms correspond to the preconditioned residual $r_m = M^{-1}(b - Ax_m)$. The 'original' residuals $r_{m,\text{original}} = b - Ax_m$ are not computed. In the case that the stop criterion is based on r_m, this can cause some difficulties. A possible solution is to compute the residuals r_m explicitly.

Right Preconditioned GMRES

The Right Preconditioned GMRES Algorithm is based on solving

$$AM^{-1}u = b \quad \text{with} \quad u = Mx. \tag{3.7}$$

In this situation the new variable u never needs to be invoked explicitly. By simple analysis of (3.7), as done in [28], one obtains that one preconditioning

operation is needed at the end of the loop, instead of at the beginning in the case of the left preconditioned version. We obtain then Algorithm 3.3. Algorithm 3.3 builds an orthogonal basis of the right preconditioned Krylov

Algorithm 3.3 Right Preconditioned GMRES

1: Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$
2: **for** $j = 1, 2, \ldots, m$ **do**
3: Compute $w = AM^{-1}v_j$,
4: **for** $i = 1, \ldots, j$ **do**
5: $h_{i,j} = (w, v_i)$,
6: $w = w - h_{i,j}v_i$,
7: **end for**
8: $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$,
9: Define $V_m = [v_1, \ldots, v_m]$ and $\bar{H}_m = (h_{i,j})_{1 \le i \le j+1, 1 \le j \le m}$,
10: **end for**
11: Compute $y_m = \text{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + M^{-1}V_m y_m$.
12: If satisfied Stop, else set $x_0 = x_m$ and go to 1.

subspace

$$\text{span} = \{r_0, AM^{-1}r_0, \ldots, (AM^{-1})^{m-1}r_0\}. \tag{3.8}$$

Note that the residual norm is now relative to the initial system, since the algorithm obtains the residual $b - Ax_m = b - AM^{-1}u_m$, implicitly. This is an essential difference with the left preconditioned GMRES algorithm.

By comparing left and right preconditioning GMRES the following proposition, taken from [28], can be proven.

Proposition 3.1. *The approximate solution obtained by left or right preconditioned GMRES is of the form*

$$x_m = x_0 + s_{m-1}(M^{-1}A)z_0 = x_0 + M^{-1}s_{m-1}(AM^{-1})r_0, \tag{3.9}$$

where $z_0 = M^{-1}r_0$ and s_{m-1} is a polynomial of degree $(m-1)$. The polynomial s_{m-1} minimizes the residual norm $\|b - Ax_m\|_2$ in the right preconditioning case, and the preconditioned residual norm $\|M^{-1}(b - Ax_m)\|$ in the left preconditioning case.

In most practical situations, the difference in the convergence behavior of the two approaches is not significant. The only exception is when M is ill-conditioned which could lead to substantial differences.

3.1.3 Preconditioned Bi-CGSTAB

We restrict ourselves by giving the preconditioned Bi-CGSTAB algorithm. It is given as Algorithm 3.4.

Algorithm 3.4 Preconditioned Bi-CGSTAB

1: Compute $r_0 = b - Ax_0$ and choose r_0^* arbitrary,
2: Set $p_0 = r_0$,
3: **for** $j = 0, 1, 2, \ldots, m$ **do**
4: $\quad \tilde{p}_j = M^{-1} p_j$,
5: $\quad \tilde{s}_j = M^{-1} s_j$,
6: $\quad v_j = A s_j$,
7: $\quad \alpha_j = \tilde{p}_j / (A\tilde{p}_j, \tilde{r}_0)$,
8: $\quad s_j = r_j - \alpha_j A\tilde{p}_j$,
9: $\quad w_j = (v_j, s_j)/(v_j, v_j)$,
10: $\quad x_{j+1} = x_j + \alpha_j \tilde{p}_j + w_j \tilde{s}_j$,
11: $\quad r_{j+1} = s_j - w_j v_j$,
12: $\quad \beta_j = \frac{(r_{j+1}, \tilde{r}_0)}{(r_j, \tilde{r}_0)} \times \frac{\alpha_j}{w_j}$,
13: $\quad p_{j+1} = r_j + \beta_j p_j - w_j A\tilde{p}_j$,
14: **end for**

3.2 Preconditioned Techniques

In this section we give some of the well known precondition techniques. Among others we will treat the Diagonal Preconditioner, ILU, and .....

Finding a good preconditioner to solve a sparse linear system is often viewed as a combination of art and science. Roughly speaking, a preconditioner is a explicit or implicit modification of an original linear system which makes it "easier" to solve by a given iterative method. An example of an explicit form of preconditioning is to scale all rows of the system to make the diagonal elements equal to one. The resulting system can be solved by a Krylov subspace method and may require fewer iterates to converge than with the original system. We remark that this is not guaranteed. As another example, solving the linear system

$$M^{-1}Ax = M^{-1}b, \tag{3.10}$$

where M^{-1} is some complicated mapping that may involve FFT transforms, integral calculations, etc., may be another form of preconditionering. Here, it is unlikely that the matrix M^{-1} and $M^{-1}A$ can be computed explicitly. Instead, the iterative processes operate with A and M^{-1} whenever needed. In practice, the (explicit or implicit) preconditioning operation M^{-1} should be inexpensive to apply to an arbitrary vector.

In this report we only treat matrix based preconditioners. Analytic preconditioners like AILU, SoV, ..., are not suited for the linear systems arising from our application.

3.2.1 Diagonal Scaling

We transform the original system $Ax = b$, with A is SPD, to the transformed system

$$\tilde{A}\tilde{x} = \tilde{b}, \tag{3.11}$$

where $\tilde{A} = P^{-1}AP^{-T}$, $x = P^{-T}\tilde{x}$ and $\tilde{b} = P^{-1}b$. The matrix $M = P^{-1}P^{-T}$ is called the preconditioner.

A simple choice for P is a diagonal matrix with the elements

$$P_{i,i} = \sqrt{A_{i,i}}, \quad \text{for } i = 1, 2, \ldots, n. \tag{3.12}$$

Then, it is easy to derive that

$$\tilde{A}_{i,i} = P_{i,i}^{-1}A_{i,i}P_{i,i}^{-T} = 1, \quad \text{for } i = 1, 2, \ldots, n. \tag{3.13}$$

In [30] it has been shown that for this choice for P the condition number of $\tilde{A}$ is minimized. For this preconditioner it is advantageous to apply CG to $\tilde{A}\tilde{x} = \tilde{b}$, since $\tilde{A}$ is easy to calculate.

3.2.2 Incomplete LU Factorization

One of the simplest ways to obtain a preconditioner is to perform an *incomplete factorization* of the original matrix A. This gives a factorization of the form $A = LU - R$, where L and U have the same non-zero structure as the lower and upper parts of the original matrix A respectively, and R is the residual of the factorization.

In a practical implementation the ILU factorization depends on the implementation of the Gaussian elimination. The general ILU factorization is given in Algorithm 3.5. Here P is the *zero pattern set* such that

$$P \subset \{(i, j) \ : i \neq j; 1 \leq i, j \leq n\}, \tag{3.14}$$

and a_{ij} are the elements of A. If we choose for P the empty set, i.e. $P = \emptyset$,

Algorithm 3.5 General ILU Factorization (IKJ version)

1: **for** $i = 2, \ldots, n$ **do**
2: **for** $k = 1, \ldots, i-1$ and if $(i, j) \notin P$ **do**
3: $a_{ik} = a_{ik}/a_{kk}$,
4: **for** $j = k+1, \ldots, n$ and for $(i, j) \notin P$ **do**
5: $a_{ij} = a_{ij} - a_{ik}a_{kj}$.
6: **end for**
7: **end for**
8: **end for**

then Algorithm 3.5 gives the complete LU decomposition of A.

Zero Fill-In ILU (ILU(0))

Denote by $NZ(A)$ the set of pairs (i,j), $1 \leq i,j \leq n$ such that $a_{ij} \neq 0$ and by $Z(A)$ the set of pairs (i,j), $1 \leq i,j \leq n$ such that $a_{ij} = 0$.

The ILU technique with no fill-in, denoted by ILU(0), consists of taking the zero pattern P to be precisely the zero pattern of A. This defines the ILU factorization in general terms:

> Choose any pair of L and U such that the elements of $A - LU$ are zero in the locations $NZ(A)$.

These constraints do not define the ILU(0) factors in a unique way since there are, in general, infinitely many pairs of L and U which satisfy these requirements. However, the standard ILU(0) technique is defined constructively using Algorithm 3.5 with $P = Z(A)$ and adding the requirement $L_{ii} = 1$. ILU(0) reduces Algorithm 3.5 to Algorithm 3.6. The accuracy

Algorithm 3.6 ILU(0)

1: **for** $i = 2, \ldots, n$ **do**
2: **for** $k = 1, \ldots, i-1$ and if $(i,k) \notin Z(A)$ **do**
3: $a_{ik} = a_{ik}/a_{kk}$,
4: **for** $j = k+1, k+2, \ldots, n$ and for $(i,j) \notin Z(A)$ **do**
5: $a_{ij} = a_{ij} - a_{ik}a_{kj}$.
6: **end for**
7: **end for**
8: **end for**

of the ILU(0) factorization may be insufficient to yield an adequate rate of convergence, see for example [28, Section 10.3.2].

ILU(p)

In order to improve this convergence rate as well as the efficiency, more accurate ILU factorizations can be performed by allowing some fill-ins. This is better known as the ILU(p) method, where p is the level of fill-in. In this section we explain ILU(p) in particular for $p = 1$.

The ILU(1) factorization results form taking P to be zero pattern of the product L_0U_0, where L_0 and U_0 are obtained via the ILU(0) factorization. By doing this, we actually consider a matrix with additional off-diagonal elements which are actually zero in the additional matrix A. The factors L_1 and U_1 of ILU(1) are obtained by performing ILU(0) to this matrix, i.e. the matrix L_0U_0.

By induction, we can define ILU(p) for $p = 2, 3, \ldots$ Remark that for increasing values of p we get a higher amount of fill-in in both L and U, which could be inefficient. Therefore, one should look for the optimal mix between the amount of work and the convergence rate.

Other variants of ILU

There are several variants of ILU(p) which differ from the variant described above. For instance we have the *Modified* ILU (MILU) which attempts to reduce the effect of dropping by compensating for the discard entries. A popular strategy is to add up all the elements that have been dropped at the completion of the k-loop in Algorithm 3.5. Then this sum is subtracted from the diagonal entry in U. The compensation strategy described above is the MILU method.

Another variant of ILU which has a more accurate factorization is ILUT or ILU(tol). Roughly speaking, the ILUT method drops elements which are smaller than a specified value τ. For example, in [28], $\tau = 10^{-4}$.

More details about MILU of ILUT can be found in [28, Chapter 10].

3.3 Incomplete Choleski Factorization

In the case that the matrix A is SPD we are able to construct an Incomplete Choleski Factorization, i.e. $A = LL^T - R$. The matrix L has the same non-zero structure as the lower part of A. Here we described the IC(0) preconditioner. The IC(p) Algorithm for $p \geq 1$, can be constructed in an analogous way as ILU(p) is constructed in the previous section. For more information about Incomplete Choleski Factorization we refer to [29].

3.4 Multigrid

The last remark on preconditioners is that also multigrid can be used as a preconditioning technique. This can easily be done by choosing $M^{-1} = (I - Q)A^{-1}$, where Q is the multigrid iteration operator. See for example [22].

With a *varying* preconditioner, like multigrid with a different cycle in each iteration, a Krylov subspace method in which the preconditioner can change from iteration to iteration is needed. The Flexible GMRES (FGMRES), or the GMRESR method allows such allows such a varying preconditioner. More information on Multigrid and Multigrid as preconditioner can be found in e.g. [22].

Part V

Concluding Remarks

Chapter 1

Summary and Conclusions

The process of CVD is mathematically described by :

1. continuity equation,
2. incompressible Navier-Stokes equations,
3. transport equation for thermal energy,
4. transport equation for gas species, or advection-diffusion-reaction equations,
5. ideal gas law.

The main purpose of this research project is to develop robust and efficient solvers for the heat reaction system, i.e., the coupled system of the transport equation for thermal energy and the advection-diffusion-reaction equations. The first approximation is to solve the test problem, which consists of a small number (say 10) of coupled advection-diffusion-reaction equations.

Using the Method of Lines approach, we first discretize in space using the finite volume approach resulting in a ODE system

$$y' = f(t, y), \quad y \in \mathbb{R}^N \quad \text{and} \quad f(t, y) : \mathbb{R}^{N+1} \to \mathbb{R}^N. \tag{1.1}$$

In (1.1) the integer N is the number of grid cells in the finite volume discretrization. This ODE system (1.1) is a so-called stiff system. The next step is to find a suitable time integration method. In general this method will contain implicit parts, or even it can be fully implicit, which results in nonlinear systems. To solve nonlinear systems, solutions of linear systems are also needed. These remarks are summarized in Figure 1.1.

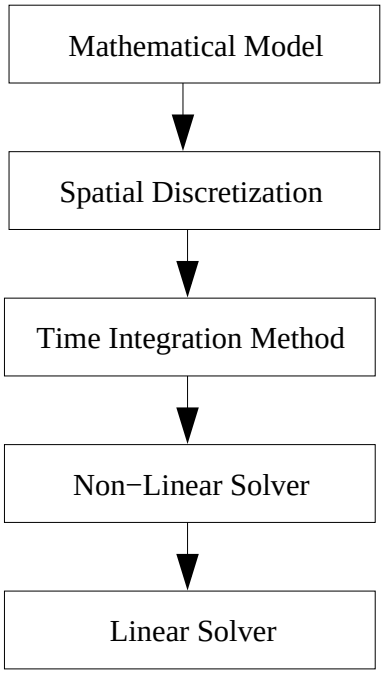

Figure 1.1: Schematic representation of steps to solve the mathematical model of CVD

1.1 Time Integration Methods

In literature one can find many time integration methods that are developed to integrate stiff systems. Nevertheless, the most succesfull ones, in practical applications, are usually the most famous ones as

- Rosenbrock methods,
- BDF, implicit linear multi-step,
- Implicit Runge Kutta methods, like RADAU methods.

We remark that these methods are fully implicit, except Rosenbrock methods which are derived from DIRK methods. The idea behind Rosenbrock methods is not to solve the nonlinear equations resulting from the implicit scheme for the stage approximations, but to compute the intermediate approximations from the linearized nonlinear system. See Part II, Section 2.2.

More recent research in the area of solving advection-diffusion-reaction equations is focused on "splitting up" the ODE system (1.1) into slow and fast varying components. Splitting up an ODE system can be done in two ways, i.e., splitting up into two subsystems and solve them separately or split the system up into a part for explicit and a part for implicit treatment. Examples of the first method are Operator Splitting and Multi-rate (Runge Kutta) methods. IMEX extensions of Runge Kutta (Chebyshev) methods are examples of the second class.

1.2 Nonlinear and Linear Solvers

The iterative methods studied for solving linear and nonlinear systems are all standard techniques, see Part III and IV. The most important question is how to find for a given time integration method the best suitable (with respect to efficiency and robustness) nonlinear solver and (eventually) linear solver.

Chapter 2

Future Research

We formulate the research subjects for future work.

1. 2D Test-problem
2. Selection of time integration methods
3. Investigate how coupling and stiffness influences stability
4. Relation time integration method and (non) linear solvers
5. 3D Test-problem :

   aim is to solve CVD system with 50 species on $50 \times 50 \times 50$ grid

2.1 Test-problem and Time Integration Methods

The first two points of the above enumeration are connected to each other. For the given test problem we will make a selection of time integration methods and compare their efficiency and robustness for the CVD application. Of course, the selection will consist of both well known methods like Rosenbrock, et. and recently developed method like IMEX RKC, etc.

2.2 Stability

By the third point in the above enumeration we mean that for some methods, like for instance Multi rate Runge Kutta methods, both coupling between slow and fast components and the enormous variations in reaction rates(the latter produces stiffness), influence the stability. As shown in [20], a simple multi-rate extension of Backward Euler can have a very small stability region for one of the subsystems. For other multi-rate extensions we could not

find any references on stability. To our knowledge a general theory on stability for multi-rate methods is not known. In the case we apply such time integration methods, we also have to investigate the influence of coupling between subsystems and stifness on stability.

2.3 (Non)-Linear Solvers

In order to solve the stiff ODE system resulting from spatial discretization, some nonlinear and linear systems have to be solved. For this typical CVD application we have to search for the optimal (non)linear solvers. By optimal we mean efficient and robust.

2.4 Extension to Three Dimensions

As formulated in the Preface, the aim of this research project is to solve the coupled heat / reaction system on a three dimensional spatial grid. To have a realistic model for CVD and/or combustion one has to model approximately 50 different chemical species. This results in a nonlinear coupled PDE system of 49 coupled advection-diffusion-reaction equations and one heat equation. The last step is then to integrate the developed solver(s) in a CFD package.

Part VI

Appendices

Appendix A

Collocation Methods

In this part we pay a little attention to *collocation methods*. Collocation is an old an universal concept in numerical analysis. Applied to an ordinary differential equation

$$w'(t) = f(t, w(t)), \tag{A.1}$$

the idea is as follows. Search for a polynomial $u(t)$ of degree s, with s a positive integer, whose derivative coincides at s given points with the vector field of the differential equation (A.1).

Definition A.1. *For s a positive integer and $c_1, \ldots, c_s$ real constants (typically between 0 and 1), the corresponding collocation polynomial $u(x)$ of degree s is defined by*

$$\begin{aligned} u(x_n) &= w_n \quad \textit{initial value} && \text{(A.2)} \\ u'(x_n + c_i h) &= f(x_n + c_i h, u(x_n + c_i h)) \quad i = 1, \ldots, s. && \text{(A.3)} \\ & && \text{(A.4)} \end{aligned}$$

The numerical solution is then given by

$$w_{n+1} = u(x_n + h). \tag{A.5}$$

If some c_i 's coincide, the collocation condition (A.3) will contain higher derivatives and lead to multi-derivative methods, see [10]. We suppose all c_i 's to be different.

Theorem A.2. *The collocation method as defined in the above definition is equivalent to the s-stage Implicit Runge-Kutta method with*

$$\alpha_{ij} = \int_0^{c_i} L_j(t)dt, \quad b_j = \int_0^1 L_j(t)dt \quad \text{and} \quad L_j(t) = \prod_{k \neq j} \frac{t - c_k}{c_j - c_k}. \tag{A.6}$$

Proof. Put $u'(x_n + c_i h) = k_i$ such that

$$u'(x_n + c_i h) = \sum_{j=1}^{s} k_j L_j(t). \tag{A.7}$$

Then integrate

$$u(x_n + c_i h) = w_0 + h \int_0^{c_i} u'(x_n + c_i h) dt,$$

and insert into (A.3) together with (A.6). We then obtain an implicit Runge-Kutta scheme. □

Appendix B

Padé Approximations

Padé approximations are rational functions which, for a given degree of the numerator and the denominator, have highest order of approximation. Their origin lies in the theory of continued fractions and they played a fundamental role in Hermite's proof of the transcendency of e. These optimal approximations can be obtained for the exponential function e^z, which is given in the following theorem. Theorem B.1 is taken from [11].

Theorem B.1. *The (k, j)-Padé approximation to e^z is given by*

$$R_{kj}(z) = \frac{P_{kj}(z)}{Q_{kj}(z)}, \tag{B.1}$$

where

$$\begin{aligned} P_{kj}(z) = 1 + \frac{k}{j+k} z + \frac{k(k-1)}{(j+k)(j+k-1)} \cdot \frac{z^2}{2!} + \cdots \\ \cdots + \frac{k(k-1)\cdots 1}{(j+k)(j+k-1)\cdots(j+1)} \cdot \frac{z^k}{k!}, \end{aligned} \tag{B.2}$$

and

$$\begin{aligned} Q_{kj}(z) = 1 - \frac{j}{k+j} z + \frac{j(j-1)}{(k+j)(k+j-1)} \cdot \frac{z^2}{2!} + \cdots \\ \cdots + \frac{j(j-1)\cdots 1}{(k+j)(k+j-1)\cdots(k+1)} \cdot \frac{z^j}{j!} = P_{jk}(-z). \end{aligned} \tag{B.3}$$

The error $e^z - R_{kj}(z)$ is then given by

$$e^z - R_{kj}(z) = (-1)^j \frac{j!k!}{(j+k)!(j+k+1)!} z^{j+k+1} + \mathcal{O}(z^{j+k+2}). \tag{B.4}$$

It is the unique rational approximation to e^z of order $j + k$, such that the degrees of numerator and denominator are k and j, respectively.

In Table B.1 the first Padé approximations to e^z are given.

	$k=0$	$k=1$	$k=2$
$j=0$	$\frac{1}{1}$	$\frac{1+z}{1}$	$\frac{1+z+\frac{z^2}{2!}}{1}$
$j=1$	$\frac{1}{1-z}$	$\frac{1+\frac{1}{2}z}{1-\frac{1}{2}z}$	$\frac{1+\frac{2}{3}z+\frac{1}{3}\frac{z^2}{2!}}{1-\frac{1}{3}z}$
$j=2$	$\frac{1}{1-z+\frac{z^2}{2!}}$	$\frac{1+\frac{1}{3}z}{1-\frac{2}{3}z+\frac{1}{3}\frac{z^2}{2!}}$	$\frac{1+\frac{1}{2}z+\frac{1}{6}\frac{z^2}{2!}}{1-\frac{1}{2}z+\frac{1}{6}\frac{z^2}{2!}}$

Table B.1: (k,j)-Padé approximations to e^z for $k,j=0,1,2$

Bibliography

[1] M. Bakker, *Analytical Aspects of a Minimax Problem*(in Dutch), Technical Note TN62, Mathematical Center, Amsterdam

[2] R.B. Bird, W.E. Stewart and E.N. Lightfoot, *Transport Phenomena*, John Wiley & Sons, (2002)

[3] C.G. Broyden, *A New Method of Solving Nonlinear Simultaneous Equations*, Computer Journal, 12, pp. 94-99, (1969)

[4] C.F. Curtiss and J.O. Hirschfelder, *Integration of Stiff Equations*, Proc. Nat. Acad. Sci. 38, pp. 235-243, (1952)

[5] G. Dahlquist, *Convergence and Stability in the Numerical Integration of Ordinary Differential Equations*, Math. Scand. 4, pp. 33-53, (1956)

[6] Ch. Engstler and Ch. Lubich, *Multi-rate Extrapolation Methods for Differential Equations with different time scales*, Computing 58, pp. 173 - 185, (1997)

[7] F.C. Eversteijn, P.J.W. Severin, C.H.J. van den Brekel and H.L. Peek *A Stagnant Layer Model for the Epitaxial Growth of Silicon from Silane in a Horizontal Reactor*, J. Electrochem. Soc. 117, pp.925-931, (1970)

[8] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Num. Anal., 21, pp. 356-362, (1984)

[9] A. Guillo and B. Lago, *Domaine de Stabilité Associé aux Formules d'Integration Numérique d'Equations Différentielles, a pas Sépar]'es et pas Liés. Recherche de Formules a Grand Rayon de Stabilité.*, Ier. Congr. assoc. Fran. Calcul, AFCAL, Grenoble, pp.43-56, (1961)

[10] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer Series in Computational Mathematics, 8, Springer, Berlin, (1987)

[11] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Second Edition, Springer Series in Computational Mathematics, 14, Springer, Berlin, (1996)

[12] P.J. van der Houwen and B.P. Sommeijer, *On the Internal Stability of Explicit, m-stage Runge-Kutta Methods for Large m Values*, Z. Angew. Math. Mech., vol 60, pp. 479-485

[13] W. Hundsdorfer and J.G. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, Springer Series in Computational Mathematics, 33, Springer, Berlin, (2003)

[14] K.F. Jensen, *Modeling of Chemical Vapor Deposition Reactors*, in *Modeling of Chemical Vapor Deposition Reactors for Semiconductor Fabrication*, Course notes, University of California Extension, Berkeley, USA, (1988)

[15] C.T. Kelley, *Solving Nonlinear Equations with Newton's Method*, Fundamentals of Algorithms, SIAM, (2003)

[16] C.R. Kleijn, *Transport Phenomena in Chemical Vapor Deposition Reactors*, PhD thesis, Delft University of Technology, (1991)

[17] C.R. Kleijn, *Computational Modeling of Transport Phenomena and Detailed Chemistry in Chemical Vapor Deposition- A Benchmark Solution*, Thin Solid Films, 365, pp. 294-306, (2000)

[18] A.E.T. Kuiper, C.H.K. van den Brekel, J. de Groot and F.W. Veltkamp, *Modeling of Low Pressure CVD Processes*, J. Electrochem. Soc. 129, pp. 2288-2291, (1982)

[19] A. Kværnø and P. Rentrop, *Low Order Multi-rate Runge-Kutta Methods in Electric Circuit Simulation*, Preprint No. 99/1, IWRMM, Universitt Karlsruhe (TH), 1999

[20] A. Kværnø, *Stability of Multi-rate Runge-Kutta Schemes*, Int. J. Differ. Equ. Appl., pp.97-105, (2000)

[21] M. Meyyapan, *Computational Modeling in Semiconductor Processing*, Artech House, Boston, (1995)

[22] C.W. Oosterlee, H. Bijl, H.X. Lin, S.W. de Leeuw, J.B. Perot, C. Vuik, P. Wesseling, *Lecture Notes for Course "Computational Science and Engineering"*, Lecture Notes, Delft University of Technology, (2005)

[23] J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Reprint of the 1970 original, Classics in Applied Mathematics 30, SIAM, Philadelphia, (2000)

[24] S.V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publ. Corp., Washington, (1980)

[25] A. Prothero and A. Robinson, *On the Stability and Accuracy of One-Step Methods for Solving Stiff Systems of Ordinary Differential Equations*, Math. of Comput., 28, pp.145-162, (1974)

[26] H.H. Robertson, *The Solution of a Set of Reaction Rate Equations*, In : J. Walsh ed. : Numer. Anal., an Introduction, Academ. Press, pp. 178-182, (1966)

[27] H.H. Rosenbrock, *Some General Implicit Processes for the Numerical Solution of Differential Equations*, Comp. J. 5, pp. 329-330, (1963)

[28] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, (1996)

[29] A. Segal and C. Vuik, *Computational Fluid Dynamics II*, J.M. Burgerscentrum Course Notes, (2004)

[30] A. van der Sluis, *Conditioning, Equilibration and Pivoting in Linear Algebraic Systems*, Numer. Math., 15, pp. 74-86, (1970)

[31] P. Sonneveld, *CGS : A Fast Lanczos Type Solver for Non-symmetric Linear Systems*, SIAM J. Sci. Stat. Comp., 10, pp. 36-52, (1989)

[32] B. Sportisse, *An Analysis of Operator Splitting in the Stiff Case*, J. Comput. Phys. 161, pp. 140-168, (2000)

[33] G. Strang, *On the Construction and Comparison of Difference Schemes*, SIAM J. Numer. Anal. 5, pp.506-517, (1968)

[34] M. Suzuki, *Fractal Decomposition of Exponential Operators with Applications to Many-Body Theories and Monte Carlo Simulations*, Phys. Lett. A 146, pp. 319-323, (1990)

[35] J.G. Verwer and B. Sportisse, *A Note on Operator Splitting in a Stiff Linear Case*, Report MAS-R9830, CWI, Amsterdam, (1998)

[36] J.G. Verwer and B.P. Sommeijer, *An Implicit-Explicit Runge-Kutta-Chebyshev Scheme for Diffusion-Reaction Equations*, SIAM Journal on Sci. Comp., 25, pp.1824-1835, (2004)

[37] J.G. Verwer, B.P. Sommeijer and W. Hundsdorfer, *RKC Time-Stepping for Advection-Diffusion-Reaction Problems*, Journal of Comp. Physics, 201, pp. 61-79, (2004)

[38] C. Vuik, *Numerieke Methoden voor Differentiaalvergelijkingen*, Lecture Notes, Delft University of Technology, (2000)

[39] H. Yoshida, *Construction of Higher Order Symplectic Integrators*, Phys. Lett. A 150, pp. 262-268, (1990)