Computational Physics of Low-Temperature Plasma Simulation

Citation for published version (APA):
Schoutrop, C. E. M. (2023). *Computational Physics of Low-Temperature Plasma Simulation: Stoichiometric transformations, Krylov methods and improved quadrature rules*. [Phd Thesis 1 (Research TU/e / Graduation TU/e), Applied Physics and Science Education]. Eindhoven University of Technology.

Document status and date:
Published: 01/02/2023

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

Computational Physics of Low-Temperature Plasma Simulation

Stoichiometric transformations, Krylov methods and improved quadrature rules

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Technische Universiteit
Eindhoven, op gezag van de rector magnificus prof.dr.ir. F.P.T. Baaijens,
voor een commissie aangewezen door het College voor Promoties, in het
openbaar te verdedigen op woensdag 1 februari 2023 om 16:00 uur

door

Chris Erik Maria Schoutrop

geboren te Maastricht

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	Prof. Dr. C. Storm
promotoren:	Dr. Ir. J. van Dijk
	Dr. Ir. J.H.M. ten Thije Boonkkamp
leden:	Prof. Dr. R.P. Brinkmann (Ruhr-Universität Bochum)
	Prof. Dr. Ir. B. Koren
	Prof. Dr. F. Toschi
adviseur:	Dr. J. Wehner

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Computational Physics of Low-Temperature Plasma Simulation

Stoichiometric transformations, Krylov methods and improved quadrature rules

Chris Schoutrop

This thesis was prepared with the LaTeX documentation system.
Printed by: Proefschriftspecialist, The Netherlands

This research is supported by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) and The Netherlands eScience Center in the framework of the JCER program,

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-833062-1-6

Copyright © 2023 C.E.M. Schoutrop

Summary

Wind and solar power generation are highly variable. In order to address this variability, these energy sources require a method to store the generated energy. One possible avenue is to dissociate CO_2 and use the produced CO to synthesize carbon-based fuel, storing energy in a carbon-neutral manner. To gain insight into dissociating CO_2 with plasmas, numerical simulations are an indispensible tool. However, performing such simulations requires vast computational resources, especially in the case of time-resolved three-dimensional multiphysics problems. In this light, the main objective of this thesis is to improve computational efficiency and reliability of plasma simulations.

One of the main governing equations in plasma simulations is the Advection-Diffusion-Reaction (ADR) equation. A specific form of the ADR equation is the multicomponent particle balance. We conduct an analysis of the Stefan-Maxwell model for diffusion used in such equations. To solve the ADR equation for multicomponent mixtures such as plasmas, we use a Finite Volume Method (FVM), most notably the complete flux scheme, which is an extension of the exponential flux scheme. After discretization with the FVM, typically a large sparse linear system is obtained. We investigate iterative methods, specifically Krylov subspace methods, to solve such linear systems efficiently.

The physics resulting from a simulation is only as complete as the physics that was included in the model. To ensure exact conservation of physical invariants such as conservation of mass and quasi-neutrality, we derive a linear transformation of the multicomponent particle balance. This transformation is based on the stoichiometry of the chemical reaction network. Additional benefits of this method are a reduction in the number of unknowns and an improved condition number of the system matrix. To expand on this, we write the solution as a superposition of the chemical equilibrium solution and a deviation from this equilibrium. This superposition method allows one to first solve an algebraic system of equations to obtain the equilibrium solution, and then solve a potentially simpler system of partial differential equations for the deviation (from equilibrium).

Three-dimensional plasma simulations require us to solve large linear systems. Iterative Krylov subspace methods are among the most efficient solvers for such systems. A detailed analysis of several Krylov subspace methods is presented, where we specifically investigate the convergence and reliability of the methods when applied to the three-dimensional ADR equation.

Additionally, we have contributed three Krylov subspace methods to the well-known Eigen C++ library for linear algebra. In this thesis, we describe in detail the implementation choices that have been fundamental to our BiCGStab(L), IDR(S) and IDR(S)Stab(L) codes.

Finally, we take a look at MagnumPI, a toolbox for computing integrals to obtain scattering angles and cross sections, starting from atomic interaction potentials. We present a detailed analysis on adaptive quadrature schemes, and demonstrate quadrature schemes that improve over existing codes for these problems. Such scattering angles and cross sections are valuable input data for numerical simulations.

To conclude, we have critically analyzed the Stefan-Maxwell model of diffusion, produced efficient open-source Krylov subspace solvers, improved the reliability and efficiency of multicomponent mixture simulations using a novel linear transformation, and enhanced adaptive integration methods.

Cover of this thesis

Contrary to the expectation of many students, linear is not synonymous with easy. The cover shows a so-called “linear cellular automaton” (CA). This kind of CA has been classified by the famous Stephen Wolfram [1, p. 51], who is also known to many students from a website that shares his last name.

The CA shown on this cover is “rule 165” according to Wolfram’s classification. If we assign a value of 1 to black squares, a value of 0 to white squares, and the letters p, q, r to three consecutive squares. Then the color of the square below square q is determined by the rule $(1 + p + r) \bmod 2$ where $(p, q, r = 0, 1)$.

Even though this CA can be described with an astonishingly simple expression, a complicated pattern emerges from applying it repeatedly. Linear algebra plays a central role in this thesis, including iteratively solving linear systems, an undertaking that appears simple, but gives rise to great complexity.

The top row appears random, however, it is not. For the real puzzle solvers, there is a message hidden in this row. This message is unlikely to be easy to crack, but it is easy to verify.

Contents

Summary v

Cover of this thesis vii

I Introduction 1

1 General introduction 3

2 Physics of multicomponent transport in plasmas 5
- 2.1 From Boltzmann to the particle balance 5
  - 2.1.1 Integral form of the particle balance 7
- 2.2 The Stefan-Maxwell equations 7
  - 2.2.1 Definition of the pressure tensor according to Holt and Whittaker 11
  - 2.2.2 Simplifying the momentum balance 14

3 Finite Volume and Krylov methods 17
- 3.1 Discretization with the Finite Volume Method 18
- 3.2 Solving sparse linear systems 22

II Reduction and equilibrium 27

4 Exploiting stoichiometry 29
- 4.1 Introduction 30
- 4.2 Mathematical model 33
  - 4.2.1 Continuity equations for plasmas as multicomponent mixtures 33
  - 4.2.2 Diffusion velocities 36
- 4.3 Stoichiometric transformation 43
  - 4.3.1 Three-component argon system 44
  - 4.3.2 Transforming a generic system 50

4.4 Discretization . . . 54
4.4.1 Finite Volume Method . . . 55
4.4.2 Flux approximation . . . 55
4.5 Results and discussion . . . 58
4.5.1 Three-species argon system . . . 60
4.5.2 Aluminum-argon-hydrogen system . . . 63
4.6 Conclusions and outlook . . . 66
4.A On the positive definiteness of the ambipolar diffusion matrix . . . 67
4.B Reaction list for the 8-species system . . . 69
4.C Obtaining a basis for a 9-species system . . . 70

5 Near-equilibrium description of plasmas based on the stoichiometric transformation 73
5.1 Governing equations . . . 74
5.2 Solving nonlinear systems using Newton-Krylov . . . 76
5.3 Atomic hydrogen model . . . 77
5.4 Computing the chemical equilibrium solution . . . 79
5.5 Stability and uniqueness of the chemical equilibrium . . . 81
5.6 Results . . . 82
5.7 Discussion and conclusion . . . 83
5.8 Outlook . . . 84
5.8.1 Enhanced procedure for finding the chemical equilibrium . 85
5.8.2 Applications . . . 86
5.8.3 Extensions . . . 86

III Krylov methods 89

6 Reliability investigation of BiCGStab and IDR solvers for the advection-diffusion-reaction equation 91
6.1 Introduction . . . 93
6.2 Solver implementation . . . 96
6.2.1 Choice of shadow residual . . . 98
6.2.2 Reliable update scheme . . . 100
6.2.3 Keeping the lowest residual solution . . . 102
6.2.4 Propagation of the initial residual . . . 102
6.3 The discrete advection-diffusion-reaction equation . . . 103
6.3.1 Dimensionless form of the ADR equation . . . 104
6.3.2 Discretization . . . 106
6.4 Results and discussion . . . 109
6.4.1 Convergence as function of Péclet and Damköhler numbers 110
6.4.2 Comparing modified solvers . . . 112

6.4.3 Comparing performance for specific Péclet and Damköhler numbers 114
6.4.4 Propagation of the residual for LMR and BiCGStab 116
6.4.5 Preconditioned BiCGStab 120
6.5 Conclusions 122
6.A Derivation LMR step size and residual 124
6.B Positive definite discretization matrices 126

7 From LMR to BiCGStab 129
7.1 Local Minimum Residual (LMR) 131
7.1.1 Properties of the LMR algorithm 132
7.1.2 LMR iteration as discretized time-dependent linear PDE . . 135
7.1.3 Extended Local Minimal Residual (LMR(2)) 135
7.2 Full Orthogonalization Method (FOM) 136
7.3 Conjugate gradient method (CG) 138
7.4 Bi-Conjugate Gradient (BiCG) 144
7.5 Bi-Conjugate Gradient Stabilized (BiCGStab) 148

8 Implementation of three new Krylov solvers for the Eigen library 151
8.1 Bi-Conjugate Gradient Stabilized(L) (BiCGStab(L)) 152
8.1.1 Choice of argmin method 153
8.1.2 Choice of the shadow residual vector 154
8.1.3 Reliable computation of the residual 158
8.1.4 Right preconditioning 159
8.1.5 Choice of default L-value 162
8.2 Induced Dimension Reduction (IDR(S)) 163
8.2.1 Choice of shadow space 163
8.2.2 Mitigation of numerical issues for small ω 163
8.2.3 Optional residual smoothing 166
8.3 Induced Dimension Reduction Stabilized(L) (IDR(S)Stab(L)) . . . 166
8.3.1 Choice of S and L 169
8.3.2 FOM construction of initial U-matrix 169

IV Magnum PI 171

9 Numerical integration of atomic potentials 173
9.1 Atomic scattering model 174
9.2 Adaptive quadrature 175
9.3 Local error estimate 177
9.3.1 Fractal rule 177
9.3.2 Simpson's rule 178
9.3.3 Bisection based error estimate 178

9.3.4 Gaussian quadrature methods 179
9.3.5 Error analysis of Gauss-Legendre quadrature 181
9.3.6 Gauss-Kronrod methods 182
9.3.7 Error analysis of Gauss-Kronrod quadrature rules 183
9.4 Computation of the scattering angle 184
9.5 Results and discussion 186
9.5.1 Benchmark problem Colonna 186
9.5.2 Lennard-Jones potential 187
9.6 Conclusions 190

10 General conclusions, discussion and outlook **193**
10.1 Part I: Introduction 193
10.2 Part II: Reduction and equilibrium 193
10.3 Part III: Krylov methods 194
10.4 Part IV: Magnum PI 194

Bibliography **195**

Acknowledgments **209**

Curriculum Vitae **211**

List of publications **213**

Part I

Introduction

Chapter 1

General introduction

Net anthropogenic CO_2-emissions from fossil fuel combustion and industrial processes have increased by 67% between 1990 and 2019 [2, p. 7]. Furthermore, reduction in CO_2-emission from fossil fuels and industrial processes has been offset by an increase in emissions caused by an increase in industry, energy production, transport, agriculture and building use [2, p. 8]. To limit global temperature increase, the CO_2-emissions have to be decreased significantly [2, p. 105].

One significant obstacle to introducing renewable power sources such as wind and solar is the intermittent power production of these sources [3]. Another obstacle is a mismatch in space; ideal locations for solar and wind farms may not be where the energy demand is largest. These two obstacles require an expansion of the energy transmission system and a storage solution [4]. Electrochemical storage in the form of solid-electrode batteries is not an ideal solution, such batteries have two orders of magnitude too little energy to power ratio to be suitable for storing intermittent renewable power, making this method excessively expensive [3].

An alternative method is to capture CO_2, and dissociate it into CO and oxygen. The produced CO can then be used in the Fisher-Tropsch process [5] to produce carbon-based fuel. If the energy required for this process is generated in a renewable fashion, these methods can be used to store excess energy as carbon-based fuel. This has the advantage that existing infrastructure for carbon-based fuel can be utilized.

One of the most energy efficient methods to dissociate CO_2 is by using plasmas [6]. To optimize the process further, and gain more insight in the physics of CO_2 decomposition by plasmas, numerical simulations are an indispensable tool. The objective of this research is to enhance the simulation efficiency of dissociating CO_2 into CO and O_2 using plasmas. However, simulating plasmas is not a trivial task. A wide range of physics has to be included in the models, ranging from electrodynamics, fluid dynamics, chemical processes and statistical physics. In the most expensive case, a fully-resolved three-dimensional, possibly also time dependent, simulation has to be set up.

Numerically, this results in complications such as stiffness, a large ratio between the timescales of fast and slow processes [7, p. 129]. The number of equations to be solved is one for each species present in the plasma, multiplied by the number of grid points. These factors combined lead to a significant computational load, especially for three-dimensional simulations on a fine grid, which have a substantial number of grid points.

To improve the computational efficiency of plasma simulations, several topics are addressed. One of the largest costs is associated with solving large, sparse, linear systems, resulting from a discretization of the governing equations. This thesis discusses the use of Krylov subspace iterative solvers for solving these linear systems. Krylov methods are a class of iterative solvers that approximate the solution to the linear system as a linear combination of a few of the previous iterates, Part III. Another important aspect is the development of reliable transport algorithms, ensuring there are no undesirable numerical artifacts in the final result of a simulation. More specifically, we focus on the Advection-Diffusion-Reaction (ADR) equation, and discretization schemes using the Finite Volume Method (FVM). Results are presented for one, two and three-dimensional discretizations of the ADR equation throughout this thesis.

One of the methods used to avoid numerical artifacts in this thesis is by exploiting stoichiometric relations. The set of governing equations is transformed, allowing a simplification of the chemical reaction term, and an exact reproduction of invariants such as total mass and net charge, Chapter 4. Additionally, a novel method is presented for plasmas that computes a deviation from an equilibrium solution, rather than starting from a rough initial guess. This can be performed without including any additional approximations that would influence the final result, Chapter 5. Finally, we examine a tool to produce valuable input data, starting from quantum mechanical interaction potentials. Specifically, we look into efficient integration schemes for computing cross sections. We determine the efficiency of these techniques for a benchmark problem and compare our computed cross sections with data from literature, Chapter 9.

Chapter 2

Physics of multicomponent transport in plasmas

Plasma is a state of matter for which a significant number of molecules are ionized [8, p. 1]; a gas composed of ions, neutral atoms, neutral molecules, and negatively charged electrons. In terms of applications, it is used in manufacturing semiconductors [9], producing coatings [10], welding, chemical processing, metallurgy [11] and various medical applications [12].

To model plasmas and perform numerical simulations, we have to obtain a set of governing equations. In the coming sections we obtain the particle balance in differential and integral form, starting from the Boltzmann equation, Section 2.1. Subsequently, we derive one of the models for multicomponent diffusion in plasmas, Section 2.2. An analysis is made of the existing derivations of the Stefan-Maxwell equations describing diffusion in multicomponent mixtures.

2.1 From Boltzmann to the particle balance

One of the starting points to describe plasmas is the Boltzmann equation [13, p. 29]

$$\frac{\partial}{\partial t} f + \vec{v} \cdot \nabla_{\vec{r}} f + \frac{\vec{F}}{m} \cdot \nabla_{\vec{v}} f = \left(\frac{\partial}{\partial t} f \right)_{\mathrm{c}} . \tag{2.1}$$

Here the distribution function $f(\vec{r}, \vec{v}, t)$ describes the number of particles in a small volume $\mathrm{d}^3\vec{r}\mathrm{d}^3\vec{v}$ at a time t. The gradient operator in position space is denoted by $\nabla_{\vec{r}}$, and the operator $\nabla_{\vec{v}}$ denotes the gradient in velocity space given in Cartesian coordinates as $\nabla_{\vec{v}} = \begin{bmatrix} \frac{\partial}{\partial v_x} & \frac{\partial}{\partial v_y} & \frac{\partial}{\partial v_z} \end{bmatrix}^{\mathrm{T}}$. To obtain the particle balance from the

Boltzmann equation, we integrate (2.1) over all velocity space;

$$\int \frac{\partial}{\partial t} f \mathrm{d}^3\vec{v} + \int \vec{v} \cdot \nabla_{\vec{r}} f \mathrm{d}^3\vec{v} + \int \frac{\vec{F}}{m} \cdot \nabla_{\vec{v}} f \mathrm{d}^3\vec{v} = \int \left(\frac{\partial}{\partial t} f\right)_{\mathrm{c}} \mathrm{d}^3\vec{v}. \tag{2.2}$$

The first term can be recognized as the time derivative of the number density in a point, which can be evaluated directly by taking the derivative outside the integral, i.e.,

$$\int \frac{\partial}{\partial t} f \mathrm{d}^3\vec{v} = \frac{\partial}{\partial t} \int f \mathrm{d}^3\vec{v} = \frac{\partial}{\partial t} n(\vec{r}), \tag{2.3}$$

where $n(\vec{r})$ is the number density in a point $\vec{r}$. To evaluate the second term, note that since $\vec{v}$ and $\vec{r}$ are independent coordinates, it follows that $\vec{v} \cdot \nabla_{\vec{r}} f = \nabla_{\vec{r}} \cdot (\vec{v} f)$ and the second term simplifies to

$$\int \vec{v} \cdot \nabla_{\vec{r}} f \mathrm{d}^3\vec{v} = \int \nabla_{\vec{r}} \cdot (\vec{v} f) \mathrm{d}^3\vec{v}. \tag{2.4}$$

Next, notice that $\nabla_{\vec{r}}$ can be taken out of the integral, i.e.,

$$\int \nabla_{\vec{r}} \cdot (\vec{v} f) \mathrm{d}^3\vec{v} = \nabla_{\vec{r}} \cdot \int (\vec{v} f) \mathrm{d}^3\vec{v}, \tag{2.5}$$

and finally, the remaining integral can be identified as the particle flux $\vec{\Gamma}(\vec{r}, t)$;

$$\int (\vec{v} f) \mathrm{d}^3\vec{v} = \vec{\Gamma}(\vec{r}, t). \tag{2.6}$$

The third term can be simplified assuming that $\vec{F}$ is $\vec{v}$-independent. This excludes magnetized plasmas, as the Lorentz force depends on velocity. Then

$$\int \frac{\vec{F}}{m} \cdot \nabla_{\vec{v}} f \mathrm{d}^3\vec{v} = \frac{\vec{F}}{m} \cdot \int \nabla_{\vec{v}} f \mathrm{d}^3\vec{v}. \tag{2.7}$$

This integral can be evaluated using the fundamental theorem of calculus for each of the three directions v_x, v_y and v_z. Then, since $f \to 0$ for $|\vec{v}| \to \infty$, as there are no particles with infinite velocity, it follows that this integral vanishes;

$$\int \nabla_{\vec{v}} f \mathrm{d}^3\vec{v} = 0. \tag{2.8}$$

The right hand side of (2.2) is 0 in the case of a single-species mixture, as elastic collisions do not produce or destroy particles, but only change their velocity. However, for a process where particles are produced/destroyed in collisions such as mixtures with multiple species, this term can be seen as a source or sink for particles respectively,

$$\int \left(\frac{\partial}{\partial t} f\right)_{\mathrm{c}} \mathrm{d}^3\vec{v} = s(\vec{r}, t). \tag{2.9}$$

Combining (2.2), (2.3), (2.6), (2.8) and (2.9) results in the differential formulation of the *particle balance*

$$\frac{\partial}{\partial t}n(\vec{r}) + \nabla_{\vec{r}} \cdot \vec{\Gamma}(\vec{r},t) = s(\vec{r},t). \tag{2.10}$$

To obtain a description for the flux $\vec{\Gamma}$, in the context of multicomponent mixtures, we turn to the Stefan-Maxwell equations in Section 2.2. For brevity we denote $\nabla_{\vec{r}}$ as ∇ from hereon.

2.1.1 Integral form of the particle balance

Starting from the differential form of the particle balance (2.10), we integrate over a control volume V to obtain

$$\int_V \frac{\partial}{\partial t} n \mathrm{d}V + \int_V \nabla \cdot \vec{\Gamma} \mathrm{d}V = \int_V s \mathrm{d}V. \tag{2.11}$$

Assuming the control volume V does not change in time, is regular and has a closed, orientable, surface S, then the divergence theorem can be applied to rewrite the second term [14, p. 907] which gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_V n \mathrm{d}V + \oint_S \vec{\Gamma} \cdot \mathrm{d}\vec{S} = \int_V s \mathrm{d}V, \tag{2.12}$$

which is known as the integral formulation of the particle balance. Equation (2.12) can be interpreted as: the change in time of the number of particles in a volume depends on the flux in/out of the volume, and the production/destruction inside the volume.

The integral formulation of the particle balance is used extensively in the discretization schemes discussed in Section 3.1. In the next section we describe a key model used to incorporate diffusion; the Stefan-Maxwell equations.

Note that also a momentum balance can be obtained from the Boltzmann equation. However, in the next section we look at approximating the full momentum balance using the Stefan-Maxwell equations.

2.2 The Stefan-Maxwell equations

The Stefan-Maxwell equations are a model for coupled diffusion in multicomponent mixtures [15, 16, 17]. The properties of the ith species in the mixture can be described with the quantities n_i, m_i, ρ_i and p_i, where n_i is the number density, m_i the mass per particle of a species i, $\rho_i = m_i n_i$ the mass density, $p_i = n_i k_\mathrm{B} T_i$

the partial pressure, with $k_\mathrm{B} = 1.38 \times 10^{-23}$ J/K the Boltzmann constant, and N is the number of components in the mixture. The total number density n, mass density ρ and pressure p of the mixture are then the sum over all species i, i.e.,

$$n = \sum_{i=1}^{N} n_i, \quad \rho = \sum_{i=1}^{N} \rho_i, \quad p = \sum_{i=1}^{N} p_i, \tag{2.13}$$

where it is assumed that the mixture acts as an ideal gas, and that Dalton's law applies. Several common fractional variables can be introduced based on these quantities, viz.

$$x_i = \frac{n_i}{n}, \quad y_i = \frac{\rho_i}{\rho}, \quad z_i = \frac{p_i}{p}, \tag{2.14}$$

which represent the molar fractions x_i, the mass fractions y_i, and the pressure fractions z_i. In the coming section different velocities of the species are discussed:

- The velocity of each species, $\vec{v}_i$.
- The velocity of the Center of Mass (CM), $\vec{v}$, also denoted as mass-averaged or barycentric velocity.
- The diffusion velocity $\vec{u}_i := \vec{v}_i - \vec{v}$.

One-dimensional arrays representing a quantity for all species, for example arrays of densities $\boldsymbol{n} = [n_1, n_2, ..., n_N]^\mathrm{T}$, are denoted in boldface. Physical vectors are denoted like the velocities above. Multidimensional arrays (matrices) are denoted with a boldface capital letter $\boldsymbol{A}$, physical tensors of order two are denoted $\boldsymbol{B}$. In Cartesian coordinates the divergence of a second order tensor $\boldsymbol{B}$ is given by [18, p. 516]

$$\nabla \cdot \boldsymbol{B} = \sum_i \sum_j \frac{\partial}{\partial x_i} B_{ij} \vec{e}_i, \tag{2.15}$$

where $\vec{e}_i$ are the Cartesian unit vectors, and the gradient of a vector is given by [18, p. 514]

$$\nabla \vec{b} = \begin{bmatrix} \frac{\partial}{\partial x} b_x & \frac{\partial}{\partial x} b_y & \frac{\partial}{\partial x} b_z \\ \frac{\partial}{\partial y} b_x & \frac{\partial}{\partial y} b_y & \frac{\partial}{\partial y} b_z \\ \frac{\partial}{\partial z} b_x & \frac{\partial}{\partial z} b_y & \frac{\partial}{\partial z} b_z \end{bmatrix}, \tag{2.16}$$

the dot products of a vector with a tensor are given by

$$\vec{b} \cdot \boldsymbol{B} = \sum_i \sum_j b_i B_{ij} \vec{e}_j^{\,\mathrm{T}}, \quad \boldsymbol{B} \cdot \vec{b} = \sum_i \sum_j B_{ij} b_j \vec{e}_i, \tag{2.17}$$

which follow from the expansions

$$\vec{b} = \sum_k b_k \vec{e}_k, \quad \boldsymbol{B} = \sum_i \sum_j B_{ij} \vec{e}_i \vec{e}_j^{\,\mathrm{T}}, \tag{2.18}$$

where the unit vectors $\vec{e}_i$ form an orthonormal basis.

The derivation of the Stefan-Maxwell equations is based on the work of Whitaker [15]. To start, the continuity equation for a species i is introduced. The species continuity equation is similar to the overall continuity equation, however, it also includes an extra source term

$$\frac{\partial}{\partial t}\rho_i + \nabla \cdot (\rho_i \vec{v}_i) = s_i, \tag{2.19}$$

where s_i is a source term for species i, for example a chemical reaction producing or consuming this species. Equation (2.19) can be obtained from Equation (2.10) by considering the particle balance for a species i and multiplying with the species masses m_i. The momentum balance for a species i in a multicomponent mixture in conservative form is given by

$$\frac{\partial}{\partial t}(\rho_i \vec{v}_i) + \nabla \cdot (\rho_i \vec{v}_i \vec{v}_i^{\mathrm{T}}) = \rho_i \vec{b}_i + \nabla \cdot \boldsymbol{T}_i + \sum_{j=1}^{N} \vec{p}_{ij} + s_i \vec{v}_i^{*}, \tag{2.20}$$

where $\rho_i \vec{b}_i$ is a body force, for example gravity. Surface forces are captured in the pressure tensor $\boldsymbol{T}_i$. The quantity $\vec{p}_{ij}$ is a force due to pairwise elastic interactions between particles of different species, and $s_i \vec{v}_i^{*}$ is a source of momentum due to inelastic collisions. Note, that the terms $\sum_{j=1}^{N} \vec{p}_{ij}$ and $s_i \vec{v}_i^{*}$ would be absent in a single-component fluid. By multiplying the continuity equation, (2.19), with the species velocity $\vec{v}_i$, subtracting the result from equation (2.20), and using the product rule of differentiation, i.e.,

$$\begin{aligned} \frac{\partial}{\partial t}(\rho_i \vec{v}_i) &= \vec{v}_i \frac{\partial}{\partial t}\rho_i + \rho_i \frac{\partial}{\partial t}\vec{v}_i, \\ \nabla \cdot ((\rho_i \vec{v}_i)\vec{v}_i^{\mathrm{T}}) &= \vec{v}_i \nabla \cdot (\rho_i \vec{v}_i) + (\rho_i \vec{v}_i) \cdot \nabla \vec{v}_i, \end{aligned} \tag{2.21}$$

equation (2.20) can be cast in the non-conservative form

$$\rho_i \left(\frac{\partial}{\partial t}\vec{v}_i + \vec{v}_i \cdot \nabla \vec{v}_i \right) = \rho_i \vec{b}_i + \nabla \cdot \boldsymbol{T}_i + \sum_{j=1}^{N} \vec{p}_{ij} + s_i (\vec{v}_i^{*} - \vec{v}_i). \tag{2.22}$$

Note that due to the second relation in (2.21) the transpose of $\vec{v}_i$ is no longer present in (2.22). Additionally, it can be observed that the first term in (2.22) is the material derivative $\rho_i \frac{\mathrm{D}}{\mathrm{D}t}\vec{v}_i$. Equation (2.22) is the momentum balance for the *species* velocity $\vec{v}_i$ in non-conservative form. This form is used shortly to derive a momentum balance in terms of the diffusion velocities $\vec{u}_i$.

The next step is to obtain the total momentum balance, and subtract it from the momentum balance for each species, leading to an expression for the diffusion velocities. The total momentum balance is attained by summing (2.20) over all species i and noting that

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\vec{p}_{ij} = \mathbf{0}, \quad \sum_{i=1}^{N} s_i(\vec{v}_i^{\,*} - \vec{v}_i) = \vec{0}, \tag{2.23}$$

as neither elastic nor inelastic collisions can change the overall momentum. Furthermore, it is used that $\sum_{i=1}^{N}\rho_i\vec{b}_i = \rho\vec{b}$. The total momentum balance is then obtained as

$$\frac{\partial}{\partial t}\sum_{i=1}^{N}\rho_i\vec{v}_i + \nabla\cdot\sum_{i=1}^{N}\rho_i\vec{v}_i\vec{v}_i^{\mathrm{T}} = \rho\vec{b} + \nabla\cdot\sum_{i=1}^{N}\boldsymbol{T}_i. \tag{2.24}$$

Since no net mass transport can result from diffusion, it holds that $\sum_{i=1}^{N}\rho_i\vec{u}_i = \vec{0}$, which allows us to rewrite the second term into the form

$$\sum_{i=1}^{N}\rho_i\vec{v}_i\vec{v}_i^{\mathrm{T}} = \rho\vec{v}\vec{v}^{\mathrm{T}} + \sum_{i=1}^{N}\rho_i\vec{u}_i\vec{u}_i^{\mathrm{T}}, \tag{2.25}$$

and consequently the total momentum balance is obtained as

$$\frac{\partial}{\partial t}(\rho\vec{v}) + \nabla\cdot(\rho\vec{v}\vec{v}^{\mathrm{T}}) = \rho\vec{b} + \nabla\cdot\sum_{i=1}^{N}\boldsymbol{T}_i - \nabla\cdot\sum_{i=1}^{N}\rho_i\vec{u}_i\vec{u}_i^{\,\mathrm{T}}. \tag{2.26}$$

To simplify the obtained total momentum balance, the continuity equation

$$\frac{\partial}{\partial t}\rho + \nabla\cdot(\rho\vec{v}) = 0, \tag{2.27}$$

is multiplied by $\vec{v}$ and subtracted from (2.26), which results in a form that can now be used to obtain the momentum balance in terms of diffusion velocities,

$$\rho\left(\frac{\partial}{\partial t}\vec{v} + \vec{v}\cdot\nabla\vec{v}\right) = \rho\vec{b} + \nabla\cdot\sum_{i=1}^{N}\boldsymbol{T}_i - \nabla\cdot\sum_{i=1}^{N}\rho_i\vec{u}_i\vec{u}_i^{\mathrm{T}}. \tag{2.28}$$

To derive the governing equations for the diffusion velocities, equation (2.28) is multiplied by the mass fraction y_i, and subtracted from (2.22). To simplify the advection term we use

$$\begin{aligned}\vec{v}_i\cdot\nabla\vec{v}_i - \vec{v}\cdot\nabla\vec{v} &= \vec{v}_i\cdot\nabla\vec{v}_i - (\vec{v}_i - \vec{u}_i)\cdot\nabla\vec{v}\\ &= \vec{v}_i\cdot\nabla(\vec{u}_i + \vec{v}) - (\vec{v}_i - \vec{u}_i)\cdot\nabla\vec{v}\\ &= \vec{v}_i\cdot\nabla\vec{u}_i + \vec{u}_i\cdot\nabla\vec{v},\end{aligned} \tag{2.29}$$

this leads to the following form of the momentum balance

$$\begin{aligned}\rho_i\left(\frac{\partial}{\partial t}\vec{u}_i+\vec{v}_i\cdot\nabla\vec{u}_i+\vec{u}_i\cdot\nabla\vec{v}\right)&=\rho_i(\vec{b}_i-\vec{b})+\nabla\cdot\boldsymbol{T}_i-\\ y_i\nabla\cdot\sum_{j=1}^{N}&\left(\boldsymbol{T}_j-\rho_j\vec{u}_j\vec{u}_j^{\mathrm{T}}\right)+\sum_{j=1}^{N}\vec{p}_{ij}+s_i(\vec{v}_i^{*}-\vec{v}_i).\end{aligned}\tag{2.30}$$

To further simplify equation (2.30), assumptions about the underlying physical properties of the fluid have to be made. It is assumed that the species in the mixture behave as Newtonian fluids; the viscous stresses are linearly related to the local strain rate, that the fluid is isotropic and when strain rates are zero the pressure tensor must reduce to $\boldsymbol{T}=-p\boldsymbol{I}$ [19, p. 65]. Additionally, it is assumed that

$$\boldsymbol{T}=\sum_i\boldsymbol{T}_i=-p\boldsymbol{I}+\boldsymbol{\tau},\quad \boldsymbol{T}_i=-p_i\boldsymbol{I}+\boldsymbol{\tau}_i,\tag{2.31}$$

where Dalton's law is assumed to hold; the total pressure is the sum of partial pressures,

$$p=\sum_{i=1}^{N}p_i.\tag{2.32}$$

Furthermore, it is assumed that the viscous stress tensor $\boldsymbol{\tau}$ can be written as a superposition of the viscous stress tensor for each species

$$\boldsymbol{\tau}=\sum_{i=1}^{N}\boldsymbol{\tau}_i=\mu(\nabla\vec{v}+\nabla\vec{v}^{\mathrm{T}})+(\tfrac{3}{2}\mu-\kappa)(\nabla\cdot\vec{v})\boldsymbol{I},\tag{2.33}$$

with μ the dynamic viscosity and κ the dilatational viscosity [20, p. 19]. With the additional assumption of a Newtonian fluid the momentum balance becomes

$$\begin{aligned}\rho_i\left(\frac{\partial}{\partial t}\vec{u}_i+\vec{v}_i\cdot\nabla\vec{u}_i+\vec{u}_i\cdot\nabla\vec{v}\right)&=\rho_i(\vec{b}_i-\vec{b})-\nabla p_i+y_i\nabla p+\sum_{j=1}^{N}\vec{p}_{ij}+\\ \nabla\cdot\boldsymbol{\tau}_i-y_i\nabla\cdot\boldsymbol{\tau}&+s_i(\vec{v}_i^{*}-\vec{v}_i)+y_i\nabla\cdot\left(\sum_{j=1}^{N}\rho_j\vec{u}_j\vec{u}_j^{\mathrm{T}}\right).\end{aligned}\tag{2.34}$$

Shortly several additional assumptions are made to further simplify the momentum balance. However, we first take a closer look at the definition of the pressure tensor $\boldsymbol{T}$ in the next Subsection.

2.2.1 Definition of the pressure tensor according to Holt and Whittaker

There exists a discrepancy in the definition of the pressure tensor $\boldsymbol{T}$. References [15, 21][22, p. 264] state that the pressure tensor should be as defined in (2.31), with the resulting equation for the momentum balance as in (2.30). However, [23, p. 51] [24, p. 14], [8, p. 168] claim that the pressure tensor $\boldsymbol{T}$ should include the extra term

$$\sum_{i=1}^{N} \rho_i \vec{u}_i \vec{u}_i^{\mathrm{T}}, \tag{2.35}$$

such that the momentum balance (2.30) reads

$$\begin{gathered} \rho_i \left(\frac{\partial}{\partial t} \vec{u}_i + \vec{v}_i \cdot \nabla \vec{u}_i + \vec{u}_i \cdot \nabla \vec{v} \right) = \\ \rho_i (\vec{b}_i - \vec{b}) + \nabla \cdot \boldsymbol{T}_i - y_i \nabla \cdot \sum_{j=1}^{N} \boldsymbol{T}_j + \sum_{j=1}^{N} \vec{p}_{ij} + s_i (\vec{v}_i^{*} - \vec{v}_i). \end{gathered} \tag{2.36}$$

To keep the notation consistent with [8], in the remainder of this subsection, subscripts indicate components of a tensor, and the superscript (s) indicates a species.

The difference arises from the claim in [8, p. 168], that the components of the pressure tensor for a species (s) are given by

$$T_{ij}^{(s)} = n^{(s)} m^{(s)} \langle V_i^{(s)} V_j^{(s)} \rangle, \tag{2.37}$$

where $V_i^{(s)}$ is the velocity of species (s) in direction i defined as

$$V_i^{(s)} := v_i^{(s)} - \langle v_i^{(s)} \rangle, \tag{2.38}$$

with $\langle v_i^{(s)} \rangle$ the averaged velocity, $v_i^{(s)}$ the actual velocity, and $V_i^{(s)}$ the velocity relative to the averaged velocity. The averaging operator is defined in terms of the distribution function f which satisfies the Boltzmann equation given by (2.1), to compute the average of a variable ϕ [8, p. 108] defines the averaging operator $\langle\rangle$ as

$$\langle \phi \rangle := \frac{\int \phi f \mathrm{d}^3 \vec{v}}{\int f \mathrm{d}^3 \vec{v}}. \tag{2.39}$$

Reference [8, p. 168] argues that when writing the equations for a plasma moving with the mass-averaged velocity v, one should calculate the pressure relative to v. A new “peculiar velocity” or “random velocity” is introduced,

$$U_i^{(s)} = v_i^{(s)} - v_i, \tag{2.40}$$

which measures the velocity of a species (s) relative to the mass-averaged velocity. Note that this velocity $U_i^{(s)}$ is identical to the species diffusion velocities $u_i^{(s)}$ as defined previously. The total pressure tensor according to [8] is then given as

$$T_{ij}^{*} = \sum_{s=1}^{N} n^{(s)} m^{(s)} \langle u_i^{(s)} u_j^{(s)} \rangle, \tag{2.41}$$

measured with respect to the velocity v_i^*. Then using the relation

$$\langle u_i^{(s)} u_j^{(s)} \rangle = \langle V_i^{(s)} V_j^{(s)} \rangle + \langle u_i^{(s)} \rangle \langle u_j^{(s)} \rangle, \tag{2.42}$$

the resulting pressure tensor becomes

$$T_{ij}^* = \sum_{s=1}^{N} T_{ij}^{(s)} + \sum_{s=1}^{N} n^{(s)} m^{(s)} \langle u_i^{(s)} \rangle \langle u_j^{(s)} \rangle, \tag{2.43}$$

where $\boldsymbol{T}^{(s)}$ is measured relative to the *averaged* velocities $\langle \vec{v} \rangle$, and $\boldsymbol{T}^*$ relative to the *mass-averaged* velocities v_i.

If the pressure tensor is isotropic, reference [8, p. 168] argues that the hydrostatic pressure p is defined as $p = \frac{1}{3} \sum_{i=1}^{N} \sum_{j=1}^{N} T_{ij}^* \delta^{ij}$, where δ^{ij} is the Kronecker delta. The interpretation of the pressure tensor is that, $\boldsymbol{T}^* \cdot \vec{a}$ is the flux of molecular momentum, of all species across a surface with unit normal $\vec{a}$, where this surface moves with a velocity $\vec{v}$. This implies that (2.41) can be interpreted as a pressure tensor, due to diffusion velocities, if we follow the definition of the pressure tensor [8, p. 111] and the derivation of [8, p. 169]. Note that this does not include pressure due to motion of the CM in this term, since the CM does not move in this reference frame.

A possible explanation of the difference between [8, p. 168] and [15] is that, [15] uses a *thermodynamic* pressure in a lab frame of reference, whereas [8, p. 168] uses a *hydrostatic* pressure in a reference frame moving along with the CM. Similarly, reference [21], which arrives at the same pressure tensor as [15] refers to a “local equilibrium pressure”. However, since the pressure tensor is not a scalar this difference in definition of the pressure may not completely explain the discrepancy, except in the case of zero strain Newtonian fluids, since in that case $\boldsymbol{T} = -p\boldsymbol{I}$.

Another possible cause of this discrepancy is the absence of the averaging operator $\langle\rangle$ in [15], which may lead to misidentifying (2.41) and (2.43).

However, note that the velocity $\vec{V_i}$ is the same as the mass-averaged velocity $\vec{v}$ in single-component fluids, or if all species masses are identical. In that case this discrepancy would not arise, since there is no difference between the average velocity and mass-averaged velocity frames of reference. As a result, (2.37) and (2.41) would be identical since $V_i = u_i$ in this situation. Considering the masses of heavy particles in a plasma are typically of the same order of magnitude, the effect of this extra term in the pressure tensor is expected to be small. Furthermore, if the plasma is weakly ionized the low mass electrons may not have a large contribution either.

Additionally, note that this extra term does not impact the further derivation of the Stefan-Maxwell equations, since the additional term is assumed to be negligible in the next steps.

2.2.2 Simplifying the momentum balance

It is shown in [15] that $|\vec{v}_i^* - \vec{v}_i| \approx |\vec{u}_i|$, if the species masses are all the same order of magnitude. Note, in plasmas the electron is $3-5$ orders of magnitude lighter than other species, nevertheless to continue the derivation of the Stefan Maxwell equations, this assumption is made. Additionally, it is assumed in [15] that:

1. $|\rho_i \frac{\partial}{\partial t}\vec{u}_i| \ll |\nabla p_i|$; quasi-steady state,
2. $|\rho_i(\vec{v}_i \cdot \nabla \vec{u}_i + \vec{u}_i \cdot \nabla \vec{v})| \ll |\nabla p_i|$; inertial effects on the diffusion velocities are negligible,
3. $|y_i \nabla \cdot \sum_{j=1}^{N} \rho_j \vec{u}_j \vec{u}_j^{\mathrm{T}}| \ll |\nabla p_i|$; the diffusive stress is negligible,
4. $|(\nabla \cdot \boldsymbol{\tau}_i - y_i \nabla \cdot \boldsymbol{\tau})| \ll |\nabla p_i|$; viscous stresses are negligible,
5. $|s_i(\vec{v}_i^* - \vec{v}_i)| \ll |\nabla p_i|$; momentum source due to chemical reactions is negligible.

With these assumptions, and using $\nabla p_i = p \nabla z_i + z_i \nabla p$, equation (2.34) simplifies to

$$\sum_{j=1}^{N} \vec{p}_{ij} = \rho_i(\vec{b} - \vec{b}_i) - (y_i - z_i)\nabla p + p \nabla z_i. \tag{2.44}$$

Assuming thermal diffusion is also negligible, as it creates small fluxes [25], the pairwise interactions $\vec{p}_{ij}$ can be modeled as

$$\vec{p}_{ij} = p \frac{z_i z_j}{D_{ij}}(\vec{u}_i - \vec{u}_j), \tag{2.45}$$

with D_{ij} the binary diffusion coefficients. Finally, dividing by p we obtain the Stefan-Maxwell equations

$$\sum_{j=1}^{N} \frac{z_i z_j}{D_{ij}}(\vec{u}_i - \vec{u}_j) = -\nabla z_i + (y_i - z_i)\frac{\nabla p}{p} + \frac{\rho_i}{p}(\vec{b} - \vec{b}_i), \tag{2.46}$$

i.e., the linear combination of diffusion velocities is equal to the combined effects of the gradient in the pressure fractions, the gradient in pressure itself, and the difference in external forces between species. Note that $(\vec{u}_i - \vec{u}_j) = (\vec{v}_i - \vec{v}_j)$, therefore the Stefan-Maxwell equations can also be interpreted to describe the emergence of a difference in species velocities. If the pressure is constant, and if there are no external forces, equation (2.46) can be simplified to the final form of the Stefan-Maxwell equations

$$\sum_{j=1}^{N} \frac{z_i z_j}{D_{ij}} (\vec{u}_i - \vec{u}_j) = -\nabla z_i. \tag{2.47}$$

The Stefan-Maxwell equations are used in chapters 4 and 5, as a model for diffusion in plasmas as multicomponent fluids. In those chapters the Stefan-Maxwell equations are used to obtain a diffusion matrix $\mathcal{E}$.

Chapter 3

Finite Volume and Krylov methods

The main system of equations used in this thesis is of the form

$$\frac{\partial}{\partial t}\boldsymbol{\varphi} + \frac{\partial}{\partial x}(\boldsymbol{U}\boldsymbol{\varphi} - \boldsymbol{\mathcal{E}}\frac{\partial}{\partial x}\boldsymbol{\varphi}) = \boldsymbol{s}, \tag{3.1}$$

equation (3.1) is the so-called (coupled) Advection-Diffusion-Reaction (ADR) equation. Here $\boldsymbol{\varphi}$ is the array of unknowns, such as the mass fractions of all species. The vector $\vec{u}$ is a known velocity field, $\boldsymbol{\mathcal{E}}$ is the symmetric, positive definite diffusion matrix, and $\boldsymbol{s}$ a source. In this thesis $\boldsymbol{\mathcal{E}}$ is obtained via the Stefan-Maxwell equations introduced in Section 2.2. The source $\boldsymbol{s}$ is used to model chemical reactions involving the species present in the plasma. A linearly transformed version of the coupled ADR system (3.1) is used in Chapter 4. Another variant of the coupled ADR equation is used in Chapter 5, there an extension is introduced to describe the solution vector $\boldsymbol{\varphi}$ relative to the chemical equilibrium.

We also cover the scalar, three-dimensional variant

$$\frac{\partial}{\partial t}\varphi + \nabla \cdot (\vec{u}\varphi - \mathcal{E}\nabla\varphi) = s. \tag{3.2}$$

In Chapter 6 the scalar version (3.2) is used in one, two and three-dimensional form to investigate the convergence of some Krylov subspace methods.

To the best of our knowledge, there is no general solution to the ADR equations. In Section 3.1 we consider the FVM, a method where the domain of interest is covered with disjunct control volumes. For each of the control volumes an approximation is set up, after which we obtain a set of algebraic equations describing the relation between the variables in neighboring control volumes.

Solving the resulting set of algebraic equations then yields an approximate solution.

Solving the set of algebraic equations resulting from the discretization typically involves solving a large linear system. Classical methods such as Gaussian elimination may not scale favorably with the number of unknowns in the problem, requiring excessive CPU time. In Section 3.2 we consider methods for solving large linear systems iteratively, more specifically Krylov subspace methods.

3.1 Discretization with the Finite Volume Method

In this section the Finite Volume Method (FVM) is introduced. The FVM is applied to the one-dimensional, scalar ADR equation by covering the domain of interest by disjunct control volumes, and obtaining an approximation in each of them. We start with

$$\frac{\partial}{\partial t}\varphi + \frac{\partial}{\partial x}\Gamma = s(x,\varphi), \quad \Gamma(x,\varphi) = u\varphi - \mathcal{E}\frac{\partial}{\partial x}\varphi, \tag{3.3}$$

where Γ is the flux density of the variable φ. The next step is to integrate (3.3) over the control volume, also referred to as a cell, $[x_{i-1/2}, x_{i+1/2}]$ centered at x_i. We also define $\Delta x = x_{i+1} - x_i$ as the distance between two nodes x_{i+1} and x_i. Since the second term can be integrated in a straightforward manner we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_{i-1/2}}^{x_{i+1/2}} \varphi \,\mathrm{d}x + \Gamma(x_{i+1/2},t) - \Gamma(x_{i-1/2},t) = \int_{x_{i-1/2}}^{x_{i+1/2}} s \,\mathrm{d}x, \tag{3.4}$$

which describes the accumulation of φ in a control volume due to a net influx and production due to chemical reactions. In this thesis we are mostly interested in the stationary ADR equation, therefore steady state is assumed and the time dependence is neglected from hereon in this section.

Our objective is to derive an equation for $\varphi_i \approx \varphi(x_i)$ in terms of φ at the neighboring control volumes, φ_{i-1} and φ_{i+1};

$$a_{\mathrm{E}}\varphi_{i+1} + a_{\mathrm{C}}\varphi_i + a_{\mathrm{W}}\varphi_{i-1} = s_i\Delta x, \tag{3.5}$$

where the coefficients $a_{\mathrm{E}}, a_{\mathrm{C}}$ and a_{W} have to be determined. In general the coefficients $a_{\mathrm{E}}, a_{\mathrm{C}}$ and a_{W} also depend on the position, and should have an index i. However, for brevity we assume in this Chapter that $\vec{u}$ and $\boldsymbol{\mathcal{E}}$ do not depend on position, and therefore also these coefficients do not depend on position. The integral over the source term has been approximated using the midpoint rule. In the FVM we determine the coefficients $a_{\mathrm{E}}, a_{\mathrm{C}}$ and a_{W} by approximating the physical flux $\Gamma(x_{i+1/2})$ with a numerical flux $\Gamma_{i+1/2}$ at each of the control volume boundaries.

Here we consider schemes that describe $\Gamma_{i+1/2}$ as a function of φ_i and φ_{i+1}. The discrete conservation law in terms of numerical fluxes is thus given by

$$\Gamma_{i+1/2} - \Gamma_{i-1/2} = s_i \Delta x. \tag{3.6}$$

Four guidelines for discretizing equations of the type (3.3) are discussed in the book of Patankar, [26, p. 36]:

1. Conservative at the control volume faces; the flux from cell 1 to cell 2, must be the same as the flux from cell 2 to cell 1 with opposite sign.
2. Signs of the coefficients must be such that $a_\mathrm{W} a_\mathrm{C} \leq 0$, $a_\mathrm{E} a_\mathrm{C} \leq 0$. Consider a source-free case with three cells, where a Dirichlet boundary condition fixes the values of φ on the outermost cells, then the value at the center cell must be between the values set at the outermost two cells. Since in this case $a_\mathrm{C}\varphi_i = -a_\mathrm{E}\varphi_{i+1} - a_\mathrm{W}\varphi_{i-1}$, with φ_{i+1} and φ_{i-1} set by the boundary conditions, we demand that $a_\mathrm{E} \leq 0$, $a_\mathrm{W} \leq 0$ and $a_\mathrm{C} \geq 0$.
3. The source term must only be linearized if it is a sink, otherwise a linearization can lead to unphysical results; a detailed discussion can be found in [26, p. 48] and [26, p. 143].
4. Coefficients must add up to zero, more specifically $a_\mathrm{W} + a_\mathrm{E} = -a_\mathrm{C}$. Assuming constant u, $\mathcal{E}$ and $s = 0$, given a solution φ to (3.3) and ignoring boundary conditions, then $\varphi + c$ for any constant $c \neq 0$ is also a solution to (3.3). The discrete scheme gives $a_\mathrm{E}\varphi_{i+1} + a_\mathrm{C}\varphi_i + a_\mathrm{W}\varphi_{i+1} = 0$, however, since $\varphi + c$ is also a solution $a_\mathrm{E}c + a_\mathrm{C}c + a_\mathrm{W}c = 0$ must hold for any c. Therefore, $a_\mathrm{E} + a_\mathrm{C} + a_\mathrm{W} = 0$ for a good discretization scheme.

Next, we present several flux approximations $\Gamma_{i+1/2}$, discuss the resulting numerical scheme and verify if these schemes satisfy the criteria set out in [26].

A first idea to estimate the flux Γ at $x_{i+1/2}$ would be to approximate $\varphi_{i+1/2}$ by linearly interpolating φ_{i+1} and φ_i, which results in the flux approximation

$$\Gamma^{\mathrm{CD}}_{i+1/2} = \tfrac{1}{2} u_{i+1/2}(\varphi_{i+1} + \varphi_i) - \mathcal{E}_{i+1/2}\frac{\varphi_{i+1} - \varphi_i}{\Delta x}. \tag{3.7}$$

Assuming constant u and $\mathcal{E}$, the resulting numerical scheme is

$$\tfrac{1}{2} u(\varphi_{i+1} - \varphi_{i-1}) - \mathcal{E}\frac{\varphi_{i+1} - 2\varphi_i + \varphi_{i-1}}{\Delta x} = s_i \Delta x. \tag{3.8}$$

The scheme (3.8) is referred to as the Central Differencing (CD) scheme. An alternative representation of this scheme is given by

$$a^{\mathrm{CD}}_\mathrm{E}\varphi_{i+1} + a^{\mathrm{CD}}_\mathrm{C}\varphi_i + a^{\mathrm{CD}}_\mathrm{W}\varphi_{i-1} = s_i \Delta x, \tag{3.9}$$

where the coefficients are given by

$$a_{\mathrm{E}}^{\mathrm{CD}} = \tfrac{1}{2}u - \tfrac{\mathcal{E}}{\Delta x}, \quad a_{\mathrm{C}}^{\mathrm{CD}} = 2\tfrac{\mathcal{E}}{\Delta x}, \quad a_{\mathrm{W}}^{\mathrm{CD}} = -\tfrac{1}{2}u - \tfrac{\mathcal{E}}{\Delta x}. \tag{3.10}$$

Referencing to the four rules of a good discretization scheme from [26, p. 36], it is observed that indeed $a_{\mathrm{W}}^{\mathrm{CD}} + a_{\mathrm{E}}^{\mathrm{CD}} = -a_{\mathrm{C}}^{\mathrm{CD}}$. However, for $\frac{1}{2}|u| > \frac{\mathcal{E}}{\Delta x}$ the coefficients $a_{\mathrm{W}}^{\mathrm{CD}}$ and $a_{\mathrm{E}}^{\mathrm{CD}}$ will be of opposite sign, which violates the second rule of [26, p. 36]. It can indeed be shown that in this case the CD scheme can yield unphysical oscillations.

An alternative to the CD scheme is the upwind scheme, which computes the numerical flux at $x_{i+1/2}$ by approximating

$$\varphi_{i+1/2} = \begin{cases} \varphi_i & \text{if} \quad u \geq 0 \\ \varphi_{i+1} & \text{if} \quad u < 0 \end{cases}, \tag{3.11}$$

for the diffusive term we again use central differencing. Taking u constant and $u > 0$, this results in the approximation of the flux as

$$\Gamma_{i+1/2} = u\varphi_i - \frac{\mathcal{E}}{\Delta x}(\varphi_{i+1} - \varphi_i). \tag{3.12}$$

The resulting upwind (UW) scheme uses the upwind value of φ as an approximate value for $\varphi(x_{i+1/2})$, instead of the linear profile assumed by the CD scheme. The resulting scheme has coefficients given by

$$a_{\mathrm{E}}^{\mathrm{UW}} = -\tfrac{\mathcal{E}}{\Delta x}, \quad a_{\mathrm{C}}^{\mathrm{UW}} = u + 2\tfrac{\mathcal{E}}{\Delta x}, \quad a_{\mathrm{W}}^{\mathrm{UW}} = -u - \tfrac{\mathcal{E}}{\Delta x}, \tag{3.13}$$

where $u > 0$, and indeed $a_{\mathrm{W}}^{\mathrm{UW}} + a_{\mathrm{E}}^{\mathrm{UW}} = -a_{\mathrm{C}}^{\mathrm{UW}}$, and $a_{\mathrm{E}}^{\mathrm{UW}} \leq 0$, $a_{\mathrm{W}}^{\mathrm{UW}} \leq 0$ and $a_{\mathrm{C}}^{\mathrm{UW}} \geq 0$. The upwind scheme does not produce unphysical oscillations, however, this comes at the cost of adding extra numerical diffusion, since the upwind scheme can also be obtained if an extra term $\frac{1}{2}|u|\Delta x$ is added to the diffusion coefficient in (3.10). Additionally, the UW scheme is less accurate, as this scheme is only first order in Δx, whereas the CD scheme is second order.

Another flux approximation can be derived from a local homogeneous constant coefficient boundary value problem, viz.,

$$\begin{gathered} \frac{\mathrm{d}}{\mathrm{d}x}\left(u\varphi - \mathcal{E}\frac{\mathrm{d}}{\mathrm{d}x}\varphi\right) = 0, \quad x_i < x < x_{i+1}, \\ \varphi(x_i) = \varphi_i, \quad \varphi(x_{i+1}) = \varphi_{i+1}, \end{gathered} \tag{3.14}$$

where u and $\mathcal{E}$ are assumed constant on $x_i < x < x_{i+1}$. This leads to a flux approximation scheme on the control volume interface at $x_{i+1/2}$ as

$$\Gamma_{i+1/2}^{\mathrm{HF}} = \frac{\mathcal{E}}{\Delta x}\Big(\mathrm{B}(-P)\varphi_i - \mathrm{B}(P)\varphi_{i+1}\Big), \tag{3.15}$$

with P the grid Péclet number and $\mathrm{B}(z)$ the Bernoulli function defined as

$$P := \frac{u\Delta x}{\mathcal{E}}, \quad \mathrm{B}(z) := \frac{z}{e^z - 1}, \quad \mathrm{B}(0) = 1. \tag{3.16}$$

This flux approximation leads to a scheme known as the exponential scheme [26, p. 87], or the Homogeneous Flux (HF) scheme [27]. The resulting scheme has coefficients

$$a_{\mathrm{E}}^{\mathrm{HF}} = -\tfrac{\mathcal{E}}{\Delta x}\mathrm{B}(P), \quad a_{\mathrm{C}}^{\mathrm{HF}} = \tfrac{\mathcal{E}}{\Delta x}(\mathrm{B}(P) + \mathrm{B}(-P)), \quad a_{\mathrm{W}}^{\mathrm{HF}} = -\tfrac{\mathcal{E}}{\Delta x}\mathrm{B}(-P). \tag{3.17}$$

$a_{\mathrm{E}}^{\mathrm{HF}}, a_{\mathrm{W}}^{\mathrm{HF}} < 0$ and $a_{\mathrm{W}}^{\mathrm{HF}} > 0$ since the Bernoulli function $\mathrm{B}(z) > 0$, and these coefficients thus satisfy criterion four. It can be verified that the HF scheme reduces to the CD scheme in the limit of $P \to 0$, and to the UW scheme if $|P| \to \infty$. As a result, the HF scheme does not have the extra numerical diffusion of the UW scheme for small P, and unlike the CD scheme, the HF scheme does not yield unphysical oscillations for large P. The scalar HF scheme is also applied to discretize a three-dimensional conservation law in Chapter 6. Extending the HF scheme to two and three dimensions is straightforward and the reader is referred to [28].

Another extension is to include a piecewise constant source term in the flux approximation, taking the scalar version of the scheme given in [29], yielding the Complete Flux (CF) scheme originally described in [30],

$$a_{\mathrm{E}}^{\mathrm{HF}}\varphi_{i+1} + a_{\mathrm{C}}^{\mathrm{HF}}\varphi_i + a_{\mathrm{W}}^{\mathrm{HF}}\varphi_{i-1} = \Delta x \left(b_{\mathrm{E}}^{\mathrm{CF}} s_{i+1} + b_{\mathrm{C}}^{\mathrm{CF}} s_i + b_{\mathrm{W}}^{\mathrm{CF}} s_{i-1}\right), \tag{3.18}$$

where the coefficients for the source term part are given by

$$b_{\mathrm{E}}^{\mathrm{CF}} = -\tfrac{1}{2}Q(1-\sigma), \quad b_{\mathrm{C}}^{\mathrm{CF}} = 1 - Q\sigma, \quad b_{\mathrm{W}}^{\mathrm{CF}} = \tfrac{1}{2}Q(1+\sigma), \tag{3.19}$$

and the functions, σ and Q are defined as

$$\sigma = \mathrm{sgn}(P), \quad Q = \tfrac{1}{2} - \mathrm{W}(P), \quad \mathrm{W}(z) = \frac{e^z - 1 - z}{z(e^z - 1)}, \quad \mathrm{W}(0) = 0, \tag{3.20}$$

with sgn the signum function such that $\mathrm{sgn}(0) = 1$. These coefficients are constructed such that only the source term in the upwind direction is included in the numerical flux. The CF scheme is discussed in more detail in Section 4.4. Unlike the HF scheme, the CF scheme does not reduce to the UW scheme in the limit of large P. The CF scheme can be extended by including a linearized source term in the derivation of the discretization scheme, this is presented in [27] and [31].

Since we are mainly interested in describing plasmas as multicomponent systems, we require a coupled version of the discretization scheme. The derivation of such a scheme is performed in [29], of which we cite the results here. For coupled systems the HF scheme becomes

$$\boldsymbol{a}_{\mathrm{E}}^{\mathrm{HF}}\boldsymbol{\varphi}_{i+1} + \boldsymbol{a}_{\mathrm{C}}^{\mathrm{HF}}\boldsymbol{\varphi}_{i} + \boldsymbol{a}_{\mathrm{W}}^{\mathrm{HF}}\boldsymbol{\varphi}_{i-1} = \boldsymbol{s}_i\Delta x, \tag{3.21}$$

where the coefficients are now matrices

$$\boldsymbol{a}_{\mathrm{E}}^{\mathrm{HF}} = -\tfrac{\boldsymbol{\mathcal{E}}}{\Delta x}\mathrm{B}(\boldsymbol{P}), \quad \boldsymbol{a}_{\mathrm{C}}^{\mathrm{HF}} = \tfrac{\boldsymbol{\mathcal{E}}}{\Delta x}(\mathrm{B}(\boldsymbol{P}) + \mathrm{B}(-\boldsymbol{P})), \quad \boldsymbol{a}_{\mathrm{W}}^{\mathrm{HF}} = -\tfrac{\boldsymbol{\mathcal{E}}}{\Delta x}\mathrm{B}(-\boldsymbol{P}). \tag{3.22}$$

The Péclet number is replaced by the Péclet matrix $\boldsymbol{P} = \boldsymbol{\mathcal{E}}^{-1}\boldsymbol{U}\Delta x$, with $\boldsymbol{\mathcal{E}}$ the diffusion matrix and $\boldsymbol{U}$ a matrix containing the advection velocities for each of the species. In this thesis we take $\boldsymbol{U}$ to be a scalar matrix $\boldsymbol{U} = u\boldsymbol{I}$. Assuming $\boldsymbol{P}$ has a complete set of eigenvectors, the matrix function $\mathrm{B}(\boldsymbol{P})$ can be computed using the spectral decomposition $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$, where $\boldsymbol{V}$ is the eigenvector matrix and $\boldsymbol{\Lambda}$ a diagonal matrix containing the corresponding eigenvalues. Then $\mathrm{B}(\boldsymbol{P}) = \boldsymbol{V}\mathrm{B}(\boldsymbol{\Lambda})\boldsymbol{V}^{-1}$, where $\mathrm{B}(\boldsymbol{\Lambda})$ is a diagonal matrix consisting of the Bernoulli function applied to each eigenvalue. $\mathrm{B}(-\boldsymbol{P})$ and other matrix functions can be computed similarly.

The CF scheme for coupled systems becomes

$$\boldsymbol{a}_{\mathrm{E}}^{\mathrm{HF}}\boldsymbol{\varphi}_{i+1} + \boldsymbol{a}_{\mathrm{C}}^{\mathrm{HF}}\boldsymbol{\varphi}_{i} + \boldsymbol{a}_{\mathrm{W}}^{\mathrm{HF}}\boldsymbol{\varphi}_{i-1} = \Delta x(\boldsymbol{b}_{\mathrm{E}}^{\mathrm{CF}}\boldsymbol{s}_{i+1} + \boldsymbol{b}_{\mathrm{C}}^{\mathrm{CF}}\boldsymbol{s}_{i} + \boldsymbol{b}_{\mathrm{W}}^{\mathrm{CF}}\boldsymbol{s}_{i-1}), \tag{3.23}$$

with the matrix coefficients of the source part given by

$$\boldsymbol{b}_{\mathrm{E}}^{\mathrm{CF}} = -\tfrac{1}{2}\boldsymbol{Q}(\boldsymbol{I} - \boldsymbol{\sigma}), \quad \boldsymbol{b}_{\mathrm{C}}^{\mathrm{CF}} = \boldsymbol{I} - \boldsymbol{Q}\boldsymbol{\sigma}, \quad \boldsymbol{b}_{\mathrm{W}}^{\mathrm{CF}} = \tfrac{1}{2}\boldsymbol{Q}(\boldsymbol{I} + \boldsymbol{\sigma}), \tag{3.24}$$

where the matrices $\boldsymbol{Q}$ and $\boldsymbol{\sigma}$ are

$$\boldsymbol{Q} = \tfrac{1}{2}\boldsymbol{I} - \boldsymbol{\mathcal{E}}\mathrm{W}(\boldsymbol{P})\boldsymbol{\mathcal{E}}^{-1}, \quad \boldsymbol{\sigma} = \mathrm{sgn}(\boldsymbol{P}). \tag{3.25}$$

The finite volume CF scheme is applied to a one-dimensional system of coupled conservation laws in Chapter 4; a more detailed discussion of the coupled scheme can also be found in that chapter. The coupled HF scheme is used in Chapter 5 on a multicomponent system.

3.2 Solving sparse linear systems

Solving sparse linear systems is featured in Chapters 4, 5, 6, 7 and 8. These linear systems stem from discretization of the ADR-equation. In the current chapter we introduce the concept of sparsity, discuss the shortcomings of direct methods that perform a matrix decomposition, and introduce several iterative methods.

After the discretization step a set of algebraic equations is obtained. If the set of algebraic equations is nonlinear, a linearization step is performed, or a nonlinear solver such as Newton or Picard iteration has to be applied. This results in a set of linear equations of the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{3.26}$$

where $\boldsymbol{A}$ is the discretization matrix, $\boldsymbol{x}$ contains the unknown quantity of interest, for example $\boldsymbol{\varphi}$ in every cell, $\boldsymbol{b}$ typically contains the source term and boundary values of the unknowns.

The resulting matrices typically contain a vast number of zeros, as only a relation between neighboring cells is implemented in the discretization step. One might define an $N \times N$ matrix $\boldsymbol{A}$ as sparse if

$$\lim_{N\to\infty} \frac{\mathrm{nnz}(\boldsymbol{A})}{N^2} = 0, \tag{3.27}$$

where $\mathrm{nnz}(\boldsymbol{A})$ is the number of nonzero elements of $\boldsymbol{A}$. However, [32, p. 75] regards a matrix as sparse if there is a linear algebra technique that can take advantage of the large number of zero elements and their locations in the matrix. An example that would not satisfy (3.27), but does satisfy the definition of [32, p. 75] would be an upper-triangular matrix $\boldsymbol{U}$ with a full upper-triangular part. Such an upper-triangular system can be efficiently solved using back substitution, however, $\lim_{N\to\infty} \frac{\mathrm{nnz}(\boldsymbol{U})}{N^2} = 1/2$. In the coming paragraphs we will introduce several methods that exploit the sparsity, in the sense of Equation (3.27), of matrices generated by discretization of PDEs.

Even though after discretization, the matrix $\boldsymbol{A}$ is typically sparse, the number of unknowns can be large, especially for three-dimensional problems. As an example, consider a domain where M cells are used in each direction, then the number of unknowns scales as M^3.

A starting point for solving linear systems would be the well-known LU decomposition [32, p. 96]. This method factorizes the matrix $\boldsymbol{A}$ into a lower triangular matrix $\boldsymbol{L}$ and an upper triangular matrix $\boldsymbol{U}$ such that

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{L}\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}, \tag{3.28}$$

where the factorized system is then solved by subsequently solving two systems, viz.,

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}, \quad \boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}. \tag{3.29}$$

It should be noted that $\boldsymbol{L}$ and $\boldsymbol{U}$ are never inverted explicitly, but rather forward/backward substitution is used to solve the triangular systems in (3.29). The main downside of LU decomposition for sparse matrices, is that the factors

$\boldsymbol{L}$ and $\boldsymbol{U}$ are not guaranteed to be sparse, even though $\boldsymbol{A}$ is sparse. This fill-in comes with a cost in terms of both memory and time complexity. The time complexity of factorizing $\boldsymbol{A}$ for the CD discretization of the three-dimensional Poisson equation is $\mathcal{O}(N^{7/3})$ [33]. After the factorization is complete, solving the two triangular systems has a complexity of $\mathcal{O}(N^{5/3})$ [33], thus the factorization is more expensive than the two triangular solves.

An alternative strategy for solving linear systems is to use iterative methods. The idea of iterative methods for linear systems is shown in Algorithm 1. In this algorithm an iterand $\boldsymbol{x}_i$ is updated every iteration, and the quality of the solution is estimated based on the corresponding residual $\boldsymbol{r}_i := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$.

Algorithm 1 An iterative procedure for solving linear systems.

```
function SOLVE($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}_i, tolerance$)
    Set $i = 0$
    Start with an initial guess for $\boldsymbol{x}_i$
    Compute initial residual $\boldsymbol{r}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$
    while $\|\boldsymbol{r}_i\| > tolerance$ do
        Compute $\boldsymbol{u}_i$, an update to $\boldsymbol{x}_i$
        Compute the new solution $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{u}_i$
        Compute the new residual $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \boldsymbol{A}\boldsymbol{u}_i$
        Increment $i$ by one
    end while
    return $\boldsymbol{x}_i$
end function
```

One of the simplest iterative methods is Richardson iteration, which simply takes $\boldsymbol{u}_i = \boldsymbol{r}_i$. Even though this method avoids the extra fill-in of the LU decomposition, it only converges if $\rho(\boldsymbol{I}-\boldsymbol{A}) < 1$ [32, p. 116], with ρ the spectral radius of $\boldsymbol{A}$.

A more modern iterative method for positive definite, symmetric $\boldsymbol{A}$ is the Conjugate Gradient (CG) method. This method is a Krylov subspace method which produces an approximate solution $\boldsymbol{x}_k$ as a projection on the affine space $\boldsymbol{x}_k \in \boldsymbol{x}_0 + \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0) = \boldsymbol{x}_0 + \operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \boldsymbol{A}^2\boldsymbol{r}_0, ..., \boldsymbol{A}^{k-1}\boldsymbol{r}_0\}$.

CG is especially efficient compared to the LU decomposition for linear systems obtained from the discretization of three-dimensional elliptic PDEs. As an example, for a discretized Poisson problem in three dimensions, with a grid spacing proportional to $N^{-1/3}$, the CG method requires $\mathcal{O}(N^{4/3})$ FLoating point OPerations (FLOPs) to reach a given tolerance ϵ [33], whereas the LU decomposition requires $\mathcal{O}(N^{7/3})$ for such systems. Additionally, CG is an optimal Krylov method in the sense that in each iteration the error is minimized over the $\boldsymbol{A}$-norm

[34]. CG also is a so-called "short recurrence" method. Short recurrence methods express the next residual and solution approximation in terms of a fixed (practically small) number of previous residuals and solutions respectively, as opposed to long recurrence methods where the number of terms in the recurrence grows with the iteration count. The CG method is discussed in more detail in Section 7.3.

For more general invertible matrices the Faber-Manteuffel theorem [35] states that there is no Krylov subspace method using $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$ which has short recurrences and optimality [36]. Therefore, for a general invertible matrix $\boldsymbol{A}$ one has to either, drop the optimality requirement, use long recurrences, use a space different from $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$, or settle for a compromise. This leads to a wide variety of Krylov methods, each with different tradeoffs.

One possible extension of CG to solve non-symmetric systems is the BiCG algorithm originally described in [37]. The idea of BiCG is to use a second Krylov subspace denoted as the "shadow space"; $\mathcal{K}_k(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0)$, where the vector $\widetilde{\boldsymbol{r}}_0$ is arbitrary but commonly chosen as $\boldsymbol{r}_0$. In Section 7.4 we take a closer look at the derivation of BiCG.

Another common method is the well-known BiCGStab algorithm. This method was originally described in [38] and has since then been incorporated in many popular linear algebra packages such as MATLAB [39] and Eigen [40], and in simulation software such as PLASIMO [41] and COMSOL [42]. BiCGStab combines one iteration of BiCG with a Local Minimal Residual (LMR) step [43]; this step can also be seen as performing one iteration of the GMRES method [44]. LMR chooses the current residual as the search direction, and takes a step in this direction such that the next residual is minimized, which results in a smoother decrease of the residual norm of BiCGStab compared to BiCG. Even though implementing Krylov subspace methods appears straightforward, Section 7.5 addresses several pitfalls and implementation choices for BiCGStab.

Other Krylov subspace methods are the BiCGStab(L), IDR(S) and combined IDR(S)Stab(L) methods. The BiCGStab(L) method combines L steps of BiCG with a GMRES(L) step [45]. GMRES(L) constructs an orthogonal L-dimensional Krylov subspace over which it minimizes the residual. The recent IDR(S) algorithm can be seen as using S different shadow spaces; $\mathcal{K}_k(\boldsymbol{A}, \widetilde{\boldsymbol{R}})$ with $\widetilde{\boldsymbol{R}}$ an $N \times S$ matrix. IDR(S) is constructed such that the residuals are forced into a subspace that decreases in dimension each iteration [46]. Even though not as widespread as BiCGStab, IDR(S) outperforms BiCGStab for a wide class of problems [46, 47]. Finally, IDR(S)Stab(L) combines the ideas of both BiCGStab(L) and IDR(S) [48].

In this thesis we investigate the convergence of the LMR, BiCGStab and

IDR(S) Krylov subspace methods in the context of the Advection-Diffusion-Reaction equation in Chapter 6. A simplified derivation of the LMR, CG and BiCG methods is presented in Chapter 7. Chapter 8 discusses in detail the implementations of BiCGStab(L) [49], IDR(S) [50] and IDR(S)Stab(L) [51] that have been contributed in the framework of this project to the Eigen C++ linear algebra library [40].

Part II

Reduction and equilibrium

Chapter 4

Multicomponent transport in plasmas;exploiting stoichiometry

This chapter is based on:
Multicomponent transport in plasmas; exploiting stoichiometry
Schoutrop, C., van Dijk, J., and Ten Thije Boonkkamp, J.,Journal of Computational Physics, 428, p.109979, 2021

Abstract. A system of mass balance equations is set up for plasma-chemical simulations; we derive the diffusion velocities from Stefan-Maxwell theory. An algorithm to linearly transform the system of mass balance equations in the context of non-LTE plasmas is described. This transformation is derived from the reaction set, and eliminates part of the chemical source term, enforces invariants such as conservation of mass and quasi-neutrality up to machine precision, provides a reduction in the number of unknowns, and significantly improves conditioning of the discretized system. The MATLAB code used in the numerical experiments is available on GitLab: `https://gitlab.com/ChrisSchoutrop/stoichiometric-transformation`.

4.1 Introduction

Multicomponent, chemically reacting flows are ubiquitous in nature and technology, for example in combustion, chemical reactors and plasma physical applications [52, 53, 54]. For such systems multiphysics models have to be considered, to account for example for the flow field, EM field, particle balances, energy balances and chemical reactions. In particular when the number of components (chemical species) and reactions is large, the governing equations lead to an increasingly large set of (nonlinear) equations and typically a wide range of time scales due to chemical reactions. Both the large number of equations, and the spread in time scales result in a significant computational load, by requiring solutions to large linear systems and excessively small time steps. In addition, physical invariants must be respected throughout the simulation. An important consequence of the bottom-up approach is apparent in the modeling of plasmas; if invariants such as conservation of charge and mass are not properly taken into account, they are not guaranteed.

Even if such invariants are satisfied by the governing equations, the discretized approximation must also respect the same invariants [55]. This sparked discretization schemes such as the complete flux scheme presented in [56]. For the complete flux scheme, if the continuous governing equations satisfy conservation of charge and mass on the boundaries, then they will be satisfied throughout the domain in exact arithmetic.

However, even in the continuous governing equations, care has to be taken when setting up the original equations describing the problem. When modelling coupled diffusion via the Stefan-Maxwell relations, it is important to be aware of singular diffusion matrices arising in the model as identified by Giovangigli [16], additionally small binary diffusion coefficients can lead to numerical difficulty [57]. Another challenge that presents itself is the inclusion of finite rate chemistry, which manifests itself primarily as a nonlinear source term. However, in such chemical reactions there are fundamental invariants that can be exploited; a chemical reaction cannot create net electric charge, nor change the number of chemical elements present for example. Such invariants are implicit in the stoichiometry of each chemical reaction. This chapter provides a linear-algebraic perspective on multicomponent transport problems, and derives a linear transformation of the governing equations based on the stoichiometry. Using this approach physical invariants can be enforced by directly imposing them on the transformed system. This results in conservation of mass and charge, accurate up to machine precision.

To close the set of mass balance equations, a relation is needed to express diffusion in terms of gradients of mass fractions. The well-known work of

Giovangigli [16] presents a description of Stefan-Maxwell diffusion, that treats all species in the mixture on an equal footing. An important physical law is that diffusion does not transport net mass. Additionally, when working with mass fractions, it is important that these must sum to 1. This necessarily implies that not all mass fractions are linearly independent. A possible remedy is to single out one of the species to ensure the sum is always 1. However, such an approach leads to numerical issues when this species is a trace species. In the work of Giovangigli, all species are treated in the same way, none of the species is singled out. This treatment eventually leads to a singular diffusion matrix, which must be regularized. However, in this procedure there is a free, positive parameter which can be chosen arbitrarily. A similar regularization procedure is required in the context of ambipolar diffusion, where the charges of species have a significant impact on the diffusion velocities. In our work we present an approach to derive these parameters from an optimization problem, by choosing the parameter such that the condition number of the resulting diffusion matrix and the ambipolar diffusion matrix is minimized. Another important step is to derive a relation between the gradients of mass, and mole fractions. Here we show that this relation is not unique, and we provide an alternative derivation that leads to a different linear relation between these two quantities. An essential difference is that in our work we show that after linearly transforming the original set of mass balance equations one can single out one species, which ultimately leads to improved conservation of mass in numerical experiments.

The main contents of this chapter describes a linear transformation of the governing equations based on the stoichiometry of the reactions present in the system. An early example of exploiting stoichiometry to simplify the governing equations has been elaborated and applied in the work of Butler and Brokaw [58] to aid the calculation of the thermal conductivity for systems in local thermodynamic equilibrium (LTE). This early example of exploiting the dependency between species in a multicomponent mixture presents a detailed analysis of computing the thermal conductivity of a mixture in the presence of chemical reactions. This work makes the observation that in every chemical reaction set, it is possible to identify independent and dependent species, where species can be constructed from the other species present in the mixture, for example $H_2O = CO_2 - CO + H_2$, in a linear-algebraic way using the species CO_2, CO and H_2 as a basis to construct H_2O. This formalism has the benefit that only the independent contributions of the particle flux have to be considered. However, to complete their analysis chemical equilibrium is assumed.

In the context of chemically reacting flows under LTE, the concept of constructing a linear transformation has been exploited extensively in the work of Rini [59]. Under the assumption of LTE there exist formation reactions that allow every species to be constructed from a set of species chosen as a "basis". Here a matrix, referred to as the stoichiometric matrix, is introduced which defines the number of basis species present in the remaining species. The structure of this matrix is then taken advantage of to obtain a homogeneous mass balance for the species chosen as building blocks. This simplified system is then used to describe the mass diffusion flux of elements in terms of gradients of the elemental mass fractions and temperature, eventually leading to a system described by the conventional Navier-Stokes equations complemented by advection-diffusion equations for the chemical elements present in the mixture. In our work we show that the stoichiometric matrix of Rini can also be extended to improve the numerical properties of the solution methods in the context of *non*-LTE systems. The key difference is that in LTE all reactions with correct stoichiometry exist, whereas in the non-LTE case incomplete reaction sets have to be considered, and the "basis species" have to be derived from the species in the mixture.

Linearly transforming the set of governing equations is common in porous media [60, 61, 62]. One such application of linearly transforming the equations describing multicomponent transport-reaction in porous media is presented in [60]. In essence in this work the authors derive two projection matrices based on the stoichiometry, which have the effect of changing the basis of the chemical source vector. The new (linear-algebraic) basis is constructed such that part of the source term is eliminated. Even though this strategy is effective in eliminating part of the source term, the transformation is *not* unique. We argue in our work that the non-uniqueness of such transformation strategies can also be applied to enhance the numerical properties of the transformed system in plasmas as well.

Another study in the context of porous media was conducted in [62] by Fan et al., where a reactive-transport problem was investigated for the application of CO_2-storage. In the work of Fan et al. a multicomponent transport problem in the presence of chemical reactions was investigated. Here a quasi-element description of chemical reactions is taken advantage of to eliminate part of the source term in the system of mass balance equations. It is here where the definition of an "element" is coined as: "an atom or a compound that does not partition into smaller entities by chemical reactions in the system under consideration". Importantly, this implies a broader notion of element, than just the chemical elements present on the periodic table. This broad notion of element is what allows a "quasi-element basis" to be constructed, even if pure chemical elements are not present in the mixture under consideration. Summarizing, our model of multi-species diffusion includes the following;

1. A transformation of the governing equations in order to derive the quasi-element mass fractions balance equations. These quasi-elements are built from the available reaction mechanism closely related to a classical elemental

decomposition. This allows for partial elimination of the chemical source vector.

2. Various extra assumptions may be easily implemented using the reformulated equations, like quasi-neutrality or constant quasi-elemental conditions and this may lead to a reduction of the number of unknowns.

3. Minimization of the condition number for matrices involved in Stefan-Maxwell diffusion based by optimizing the regularization procedure.

4. Decreased condition number of the resulting discretization matrices for the reformulated system.

This chapter is organized as follows. We start by introducing the governing conservation laws for plasmas as multicomponent mixtures in Section 4.2. Next, in Section 4.3, the core part of our work is presented on obtaining a stoichiometric transformation to improve the numerical properties of the model. To discretize the governing equations, we summarize the discretization scheme presented in [56]. We report and discuss the results of our numerical experiments in Section 4.5. We close with concluding remarks in the final section. The MATLAB code used in the numerical experiments is available on GitLab: https://gitlab.com/ChrisSchoutrop/stoichiometric-transformation.

4.2 Mathematical model

The difference between a gas and a plasma is the significant effects of charged particles. Even simple plasmas containing only one type of neutral species, positively charged ions and electrons have to be regarded as multicomponent mixtures, first, due to the difference in charge of each species, and second due to the substantial difference in mass; on one hand the heavy ions and neutrals, on the other hand the light electrons. To describe plasmas, the mass and charge balances are introduced. To close the set of equations we derive a relation for the diffusion velocities from the Stefan-Maxwell equations, where we also take the effects of ambipolar diffusion into account.

4.2.1 Continuity equations for plasmas as multicomponent mixtures

Multicomponent mixtures are described by conservation laws, such as particle, momentum and energy balances. We first introduce the mass balances in Section 4.2.1.1, next the charge balance in Section 4.2.1.2 to account for charged species. The main focus of this chapter is on transforming the mass balances, therefore we consider a multicomponent mixture with a constant flow field, and diffusion via the Stefan-Maxwell relations. For isothermal mixtures the energy balance does not have to be taken into account.

4.2.1.1 Mass balances

For each species i there is a balance equation describing how its number density n_i changes as a function of position and time, given by

$$\partial_t n_i + \nabla \cdot (n_i \vec{v}_i) = \omega_i. \tag{4.1}$$

Here $\vec{v}_i$ is the velocity of species i and ω_i is the production/consumption rate for this species. Here an arrow is used to denote a spatial vector, for example the electric field $\vec{E}$ or the velocity $\vec{v}$. Boldface lower case letters denote arrays of variables associated with the species in the mixture, for example $\boldsymbol{n} = [n_1, ..., n_{N_\mathrm{s}}]^\mathrm{T}$ is the array containing the number densities for all N_s species.

Multiplying the particle balance for species i with its mass m_i, the well-known mass balance is obtained

$$\partial_t(\rho y_i) + \nabla \cdot (\rho y_i \vec{v}_i) = m_i \omega_i, \tag{4.2}$$

with the corresponding mass density ρ and mass fractions y_i defined as

$$\rho := \textstyle\sum_i \rho_i, \quad \rho_i := m_i n_i, \quad y_i := \rho_i/\rho, \tag{4.3}$$

where the summation runs over all species i. We also introduce the mass-averaged velocity $\vec{v}$

$$\vec{v} := \textstyle\sum_i y_i \vec{v}_i, \tag{4.4}$$

this allows for the introduction of diffusion velocities $\vec{u}_i := \vec{v}_i - \vec{v}$; the velocity of each species relative to the mass-averaged velocity. Using this definition, the second term in the mass balance (4.2) can be split into a bulk and a diffusive contribution;

$$\partial_t(\rho y_i) + \nabla \cdot (\rho y_i \vec{v}) + \nabla \cdot (\rho y_i \vec{u}_i) = m_i \omega_i. \tag{4.5}$$

The formulation of the particle balance in terms of mass fractions yields three invariants. First, by their definition in (4.3) the mass fractions sum to unity. Second, since the diffusion velocities $\vec{u}_i$ are defined relative to the mass-averaged velocity $\vec{v}$, we have

$$\textstyle\sum_i y_i \vec{u}_i = \vec{0}, \tag{4.6}$$

i.e., diffusion does not transport net mass. Finally, in chemical reactions the total mass must be conserved;

$$\textstyle\sum_i m_i \omega_i = 0. \tag{4.7}$$

As a result, the sum over all mass balances yields the overall continuity equation, i.e.,

$$\partial_t \rho + \nabla \cdot (\rho \vec{v}) = 0. \tag{4.8}$$

For later use we also need the diffusive mass fluxes, defined as

$$\vec{\phi}_i = \rho y_i \vec{u}_i, \tag{4.9}$$

and the mole fractions

$$\chi_i = \frac{n_i}{\sum_j n_j}. \tag{4.10}$$

Equation (4.5) is the system that will be solved numerically. However, to include the effects of charged species first the charge balance has to be introduced.

4.2.1.2 Charge balance

Lieberman [13, p. 6] defines a plasma as "a collection of free charged particles moving in random directions that is, on average, electrically neutral". One key aspect of a plasma is thus the significant presence of charged species, most notably positive ions and electrons. To incorporate the effect of these charged species Maxwell's equations should be solved. More specifically, Gauss' law should be used to calculate the electrostatic field $\vec{E}$ induced by the particles

$$\nabla \cdot (\epsilon \vec{E}) = \rho_{\mathrm{c}}, \tag{4.11}$$

where ρ_{c} is the charge density and ϵ the electric permittivity of the medium. However, in the continuum approach it is preferable to assume the plasma is quasi-neutral. Such a situation arises when charged particles are effective in shielding the electric field of other particles. The characteristic length scale over which significant charge densities can exist is the Debye length λ_{D}[13, p. 38], i.e., when the length scales of interest are much larger than λ_{D} then the assumption of quasi-neutrality is valid. This is generally the case for high pressure plasmas, or in the core of a plasma.

Another effect is that each charged species has an impact on the diffusion velocities of all charged species. For example consider a positive ion and an electron moving apart, the electric field directed from the ion to the electron will accelerate the ion, while at the same time decelerating the electron. Consequently, electrons will diffuse slightly slower than they would in a neutral environment, whereas ions will diffuse slightly faster. This effect is known as *ambipolar diffusion*, and must be reflected by the diffusion velocities $\vec{u}_i$ [13, p. 135]. The electric field that is generated in this effect is known as the *ambipolar electric field* $\vec{E}_{\mathrm{amb}}$ and will play a key role in describing ambipolar diffusion in Section 4.2.2.2.

Similar to the mass balances (4.5), one can obtain the charge balances. Multiplying (4.1) with the charge q_i of each species and introducing the decomposition $\vec{v}_i = \vec{v} + \vec{u}_i$, we find

$$\partial_t(q_i n_i) + \nabla \cdot (q_i n_i \vec{v}) + \nabla \cdot (q_i n_i \vec{u}_i) = q_i \omega_i. \tag{4.12}$$

Due to the assumption of quasi-neutrality, the net charge in the continuum limit is zero

$$\textstyle\sum_i q_i n_i = 0. \tag{4.13}$$

Additionally, chemical reactions cannot produce net electric charge;

$$\textstyle\sum_i q_i \omega_i = 0. \tag{4.14}$$

The total current density $\vec{j}$ is given by a summation over all individual contributions;

$$\vec{j} := \textstyle\sum_i q_i n_i \vec{v}_i. \tag{4.15}$$

As a consequence of the invariants (4.13) and (4.14), summing (4.12) over all species a constraint on the current density $\vec{j}_i := q_i n_i \vec{v}_i$ is obtained;

$$\nabla \cdot \left(\textstyle\sum_i \vec{j}_i \right) = 0, \tag{4.16}$$

i.e., the total current density $\vec{j}$ must be divergence-free. However, note that for quasi-neutral plasmas $\sum_i q_i n_i \vec{v} = \vec{v} \sum_i q_i n_i = \vec{0}$ and (4.15) reduces to $\vec{j} = \sum_i q_i n_i \vec{u}_i$. Therefore, any currents present in the plasma due to boundary conditions can only be described by the diffusive contribution to the species velocities;

$$\vec{j} = \textstyle\sum_i \vec{j}_i = \sum_i q_i n_i \vec{u}_i = \vec{j}_{\text{ext}}, \tag{4.17}$$

i.e., the charge transported by diffusive processes must be equal to the external current density $\vec{j}_{\text{ext}}$, see [55]. Thus, without externally driven currents, diffusion does not transport net charge. Assuming no currents on the boundaries, this corresponds to a zero-current approximation. Since quasi-neutrality is also imposed on the boundaries of the system, there is no net charge flux. For details we refer to [55, 63].

4.2.2 Diffusion velocities

In the previous section the particle balances have been introduced, as well as its relation to the charge and mass balance. To close the system of the particle balance equations the Stefan-Maxwell description for multicomponent diffusion is used to obtain the diffusion velocities as function of the mass fractions. The Stefan-Maxwell description of diffusion is an active area of research, with both recent developments in numerical strategies [17, 64, 65, 66, 67] and theoretical work

[68]. Here, we follow the approach of Giovangigli [16]. Alternatively, [69] presents a DAE-like idea with an augmented matrix formalism. However, in the method presented here we do not have to solve DAEs. Generally DAEs are more difficult to solve than ODEs. To a certain extent enforcing properties such as conservation of mass and quasi-neutrality can be seen as imposing algebraic constraints, in addition to the governing conservation laws (a system of PDEs). These algebraic constraints are present in the system without the need to explicitly solve a set of DAEs.

4.2.2.1 Stefan-Maxwell diffusion

To close the model a relation for the diffusion velocities $\vec{u}_i$ is required. To do this the Stefan-Maxwell equations are introduced based on the work of Giovangigli [16]

$$\textstyle\sum_j f_{ij}(\vec{u}_i - \vec{u}_j) = -\vec{d}_i, \tag{4.18}$$

where f_{ij} are the friction coefficients and $\vec{d}_i$ the so-called driving forces. This variable $\vec{d}_i$ can incorporate effects such as concentration, pressure, thermal and ambipolar diffusion. However, here only the contributions of concentration and ambipolar diffusion will be taken into account. For clarity only the one-dimensional case is considered here, since for non-magnetized plasmas the spatial components of the diffusion velocities are decoupled [70] and each of the spatial directions can be viewed independently. Therefore, it suffices to consider only one spatial component, the x-direction. Equation (4.18) then reduces to

$$\textstyle\sum_j f_{ij}(u_i - u_j) = -d_i. \tag{4.19}$$

The friction coefficients f_{ij} in (4.19) are expressed as

$$f_{ij} = \frac{z_i z_j}{\mathcal{D}_{ij}} > 0, \tag{4.20}$$

with $\mathcal{D}_{ij}$ the binary diffusion coefficients [71] and $z_i := p_i / \sum_j p_j$ the pressure fractions, where p_i indicates the partial pressure of a species i. The diffusion coefficients are adopted from Ramshaw [71], which introduces a correction factor to the earlier work in [72] to account for the effects of charged particles to the (thermal) diffusion coefficients. The models in [71, 72] are based on a first order approximation of Chapman-Enskog theory. Assuming the ideal gas law, it follows for an isothermal plasma that $z_i = \chi_i$. Since the friction coefficients (4.20) describe pairwise interactions between species, $f_{ij} = f_{ji}$ and $\mathcal{D}_{ij} = \mathcal{D}_{ji}$. Using these properties of f_{ij}, the original set (4.19) can be compactly written as a linear system;

$$\boldsymbol{F}\boldsymbol{u} = -\boldsymbol{d}, \tag{4.21}$$

with $\boldsymbol{F}$ the friction matrix

$$\boldsymbol{F} = (F_{ij}), \quad F_{ij} := \begin{cases} -f_{ij} & i \neq j \\ \sum_{k\neq i} f_{ik} & i = j \end{cases}, \tag{4.22}$$

where $\boldsymbol{u} = [u_1, ..., u_{N_\mathrm{S}}]^\mathrm{T}$ is the array of diffusion velocities for each species, and $\boldsymbol{d} = [d_1, ..., d_{N_\mathrm{S}}]^\mathrm{T}$ describes the driving forces for each species. It should be noted, since diffusion is always relative to the mass-averaged velocity of the flow, that the driving forces must sum to zero. Otherwise, a net force would be present, and diffusion would also affect the mass-averaged velocity v. The criterion $\mathbf{1}^\mathrm{T}\boldsymbol{d} = 0$ is required for the system (4.21) to be consistent.

From the definition in (4.22) it can be seen that each row and column of $\boldsymbol{F}$ sums to 0, since $f_{ij} \equiv f_{ji}$. This implies that $\mathbf{1} := [1, .., 1]^\mathrm{T}$ is a left eigenvector of $\boldsymbol{F}$ with eigenvalue 0, and therefore $\boldsymbol{F}$ is singular; $\mathbf{1}^\mathrm{T}\boldsymbol{F} = \mathbf{0}^\mathrm{T}$. As a consequence system (4.21) may not have a solution for $\boldsymbol{u}$. However, since $\mathbf{1}^\mathrm{T}\boldsymbol{d} = 0$, the vector $\boldsymbol{d}$ resides in the range of $\boldsymbol{F}$ denoted as $\mathcal{R}(\boldsymbol{F})$. This implies that if the dimension of the nullspace, $\mathcal{N}(\boldsymbol{F})$, of $\boldsymbol{F}$ is 1, then (4.21) does have a solution. To prove that $\boldsymbol{d} \in \mathcal{R}(\boldsymbol{F})$ such that eq. (4.21) has a solution. We start by proving that $\boldsymbol{F}$ is symmetric, nonnegative definite, cf. [16]. Given a nonzero vector $\boldsymbol{v}$, it follows after some basic manipulations, and by using $f_{ij} = f_{ji}$ that

$$\boldsymbol{v}^\mathrm{T}\boldsymbol{F}\boldsymbol{v} = \textstyle\sum_i \sum_{j>i} f_{ij}(v_i - v_j)^2 \geq 0, \tag{4.23}$$

for $f_{ij} > 0$. Therefore $\boldsymbol{F}$ is symmetric, nonnegative definite. Note $\boldsymbol{v}^\mathrm{T}\boldsymbol{F}\boldsymbol{v} = 0$ if all elements of $\boldsymbol{v}$ are equal, i.e., $\boldsymbol{v} = \lambda\mathbf{1}$. Next, we show that $\mathcal{N}(\boldsymbol{F}) = \{\mathbf{1}\}$ using proof by contraction. Obviously $\mathbf{1} \in \mathcal{N}(\boldsymbol{F})$. Assume there exists a vector other than $\boldsymbol{v} = \lambda\mathbf{1}$ in the nullspace of $\boldsymbol{F}$. Then $\boldsymbol{F}\boldsymbol{v} = \mathbf{0}$ and $\boldsymbol{v}^\mathrm{T}\boldsymbol{F}\boldsymbol{v} = 0$ which contradicts the earlier finding that $\boldsymbol{v}^\mathrm{T}\boldsymbol{F}\boldsymbol{v} = 0$ if all elements of $\boldsymbol{v}$ are equal, from which we concluded $\boldsymbol{v} = \lambda\mathbf{1}$, therefore $\mathcal{N}(\boldsymbol{F}) = \{\mathbf{1}\}$. To complete the proof we apply the rank-nullity theorem; $\dim\mathcal{N}(\boldsymbol{F}) + \dim\mathcal{R}(\boldsymbol{F}) = N_\mathrm{s}$ therefore $\dim\mathcal{R}(\boldsymbol{F}) = N_\mathrm{s} - 1$. Moreover, since $\boldsymbol{F}$ is symmetric $\mathcal{R}(\boldsymbol{F}) = \mathcal{N}(\boldsymbol{F})^\perp = \{\mathbf{1}\}^\perp$. Clearly $\boldsymbol{d} \in \{\mathbf{1}\}^\perp$ since $\mathbf{1}^\mathrm{T}\boldsymbol{d} = 0$.

To obtain this solution $\boldsymbol{u}$ from (4.21), we use the diffusive mass-flux constraint (4.6), written in matrix-vector form as;

$$\boldsymbol{y}^\mathrm{T}\boldsymbol{u} = 0. \tag{4.24}$$

The idea is then to combine the linear system (4.21), with (4.24) to obtain a regularized friction matrix $\widetilde{\boldsymbol{F}}$. This regularization can be performed by adding the diadic product $\alpha\boldsymbol{y}\boldsymbol{y}^\mathrm{T}$ with $\alpha > 0$ to $\boldsymbol{F}$ in equation (4.21);

$$\widetilde{\boldsymbol{F}}\boldsymbol{u} = \left(\boldsymbol{F} + \alpha\boldsymbol{y}\boldsymbol{y}^\mathrm{T}\right)\boldsymbol{u} = -\boldsymbol{d}. \tag{4.25}$$

Since $\alpha(\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}})\boldsymbol{u} = \alpha\boldsymbol{y}(\boldsymbol{y}^{\mathrm{T}}\boldsymbol{u}) = \boldsymbol{0}$ this addition does not change $\boldsymbol{u}$.

Now the velocities $\boldsymbol{u}$ can be obtained by inverting the regularized matrix $\widetilde{\boldsymbol{F}}$ as

$$\boldsymbol{u} = -\widetilde{\boldsymbol{F}}^{-1}\boldsymbol{d} = -\widetilde{\boldsymbol{D}}\boldsymbol{d}, \tag{4.26}$$

where we have introduced the matrix $\widetilde{\boldsymbol{D}} := \widetilde{\boldsymbol{F}}^{-1}$. This matrix is symmetric and positive definite for $\alpha > 0$ [16], as can be shown from the definition of $\boldsymbol{F}$ and the positivity of f_{ij}. In [73] it is suggested to choose $\alpha = 1/\max(\mathcal{D}_{ij})$, such that elements of the matrices $\boldsymbol{F}$ and $\alpha\boldsymbol{y}\boldsymbol{y}$ will have the same order of magnitude. Another choice for α is presented in [74], where a (slightly different) friction matrix was regularized, there the choice $\alpha = \max(\boldsymbol{F})/(\boldsymbol{y}^{\mathrm{T}}\boldsymbol{1})$ was made. However, it was also noted that this choice may not always be the optimal one. An alternative approach would be to minimize the condition number κ of the friction matrix. We choose to determine α from a minimization problem;

$$\alpha = \operatorname*{argmin}_{\alpha>0} \kappa(\boldsymbol{F} + \alpha\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}}). \tag{4.27}$$

Such minimization requires only a few iterations of the function `fminbnd` in Matlab 2019a [39]. A low condition number is a useful property to have since the matrix $\widetilde{\boldsymbol{F}}$ has to be inverted to obtain the diffusion matrix $\widetilde{\boldsymbol{D}}$. In the next section $\widetilde{\boldsymbol{D}}$ will be modified to include the effects of ambipolar diffusion. Alternatively, there exist cheap approximate inversions of Stefan-Maxwell matrices for plasmas [57].

4.2.2.2 Ambipolar diffusion

From the previous section an expression for the diffusion velocities $\boldsymbol{u}$ in terms of the driving forces $\boldsymbol{d}$ has been obtained. Following the approach of Peerenboom et al. [55] and the earlier work presented by Giovangigli in [63], we include the effect of ambipolar diffusion The total driving force is given by

$$\boldsymbol{d} = \boldsymbol{d}_{\mathrm{con}} + \boldsymbol{d}_{\mathrm{amb}}, \tag{4.28a}$$

where $\boldsymbol{d}_{\mathrm{con}}$ is the driving force due to concentration diffusion and $\boldsymbol{d}_{\mathrm{amb}}$ the ambipolar diffusion contribution, given by

$$\boldsymbol{d}_{\mathrm{con}} = \partial_x\boldsymbol{\chi}, \quad \boldsymbol{d}_{\mathrm{amb}} = -\boldsymbol{\rho}_{\mathrm{c}}\frac{E_{\mathrm{amb}}}{p}, \tag{4.28b}$$

where $\boldsymbol{\rho}_{\mathrm{c}}$ is the charge density array with components n_iq_i and E_{amb} the x-component of the ambipolar electric field. Due to ambipolar diffusion, charged particles cannot diffuse independently of each other, as argued in Section 4.2.1.2. Additionally, we conclude from equation (4.16) that the diffusive current density $\vec{j}$

must be divergence-free. However, if there is no current due to an external electric field, then it follows as a special case from equation (4.17) that

$$\textstyle\sum_i j_i \equiv \boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\boldsymbol{u} = 0. \tag{4.29}$$

Combining equation (4.26) with the driving forces given in (4.28) and left-multiplying with $\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}$ it follows that

$$\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\boldsymbol{u} = -\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\widetilde{\boldsymbol{D}}\Big(\partial_x\boldsymbol{\chi} - \frac{E_\mathrm{amb}}{p}\boldsymbol{\rho}_\mathrm{c}\Big) = 0. \tag{4.30}$$

From (4.30) an expression for the ambipolar electric field E_amb can be obtained;

$$\frac{1}{p}E_\mathrm{amb} = \frac{\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\widetilde{\boldsymbol{D}}\partial_x\boldsymbol{\chi}}{\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_\mathrm{c}}, \tag{4.31}$$

and as a result, using associativity of matrix multiplication,

$$\boldsymbol{u} = -\boldsymbol{D}_\mathrm{amb}\partial_x\boldsymbol{\chi}, \quad \boldsymbol{D}_\mathrm{amb} := \widetilde{\boldsymbol{D}} - \frac{\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_\mathrm{c}\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\widetilde{\boldsymbol{D}}}{\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_\mathrm{c}}, \tag{4.32}$$

with $\boldsymbol{D}_\mathrm{amb}$ the (singular) ambipolar diffusion matrix. Combining equation (4.30) and (4.32) we see that the matrix $\boldsymbol{D}_\mathrm{amb}$ is singular, since $\boldsymbol{\rho}_\mathrm{c}^\mathrm{T}\boldsymbol{D}_\mathrm{amb} = \boldsymbol{0}^\mathrm{T}$. To regularize the matrix, a similar procedure is applied as in (4.25)

$$\widetilde{\boldsymbol{D}}_\mathrm{amb} := \boldsymbol{D}_\mathrm{amb} + \beta\boldsymbol{q}\boldsymbol{q}^\mathrm{T}, \tag{4.33}$$

with $\beta > 0$. To justify the addition of the dyadic product $\beta\boldsymbol{q}\boldsymbol{q}^\mathrm{T}$ the constraint set by the quasi-neutrality approximation (4.13) is used. Since $\boldsymbol{q}^\mathrm{T}\boldsymbol{n} = 0$ implies $\boldsymbol{q}^\mathrm{T}\boldsymbol{\chi} = 0$, it becomes clear that the diffusion velocities can be obtained from

$$\boldsymbol{u} = -\widetilde{\boldsymbol{D}}_\mathrm{amb}\partial_x\boldsymbol{\chi}. \tag{4.34}$$

Since the addition of $\beta\boldsymbol{q}\boldsymbol{q}^\mathrm{T}$ does not change the solution to (4.34), the value for β in (4.33) is again chosen to minimize the condition number;

$$\beta = \operatorname*{argmin}_{\beta>0}\kappa(\boldsymbol{D}_\mathrm{amb} + \beta\boldsymbol{q}\boldsymbol{q}^\mathrm{T}). \tag{4.35}$$

It can be shown that for $\beta > 0$ the matrix $\widetilde{\boldsymbol{D}}_\mathrm{amb}$ is symmetric positive definite and thus regular, this is shown in Section 4.A. Again, we opt for minimizing the condition number, as later on in the discretization the inverse of the diffusion matrix will be needed.

4.2.2.3 Diffusion in terms of mass fluxes

The next step is to relate the spatial variation in the mass fractions to the spatial variation in mole fractions. To do this $\partial_x \boldsymbol{\chi}$ has to be related to $\partial_x \boldsymbol{y}$, and for this purpose we seek a matrix $\widetilde{\boldsymbol{\mathcal{M}}}$ such that

$$\partial_x \boldsymbol{\chi} = \widetilde{\boldsymbol{\mathcal{M}}} \partial_x \boldsymbol{y}. \tag{4.36}$$

Previously, in the work of Giovangigli [16] it is argued that imposing $\sum_i \chi_i = 1$ and $\sum_i y_i = 1$ a priori leads to a singular matrix $\widetilde{\boldsymbol{\mathcal{M}}}$. In this work, the problem was overcome by defining

$$(\textstyle\sum_i \chi_i) W = \sum_i \chi_i m_i, \quad W := \dfrac{\sum_i y_i}{\sum_i y_i/m_i}, \tag{4.37}$$

which leads to $\sum_i \chi_i = \sum_i y_i$, as opposed to imposing $\sum_i \chi_i = \sum_i y_i = 1$ directly. In our work we start with χ_i and y_i defined from number densities and species masses, from this we indeed obtain a singular matrix relating $\partial_x \boldsymbol{\chi}$ and $\partial_x \boldsymbol{y}$. However, we show that this matrix can be regularized by adding a dyadic product.

Recall that the mole and mass fractions introduced earlier in (4.10) and (4.3) can both be defined in terms of densities;

$$\chi_i := n_i \Big(\textstyle\sum_j n_j \Big)^{-1}, \quad y_i := m_i n_i \Big(\textstyle\sum_j m_j n_j \Big)^{-1}, \tag{4.38}$$

from which a direct relation between χ_i and y_i can be found

$$\chi_i = \frac{y_i}{m_i} \Big(\textstyle\sum_j \frac{y_j}{m_j} \Big)^{-1}. \tag{4.39}$$

Taking the derivative of χ_i with respect to the spatial coordinate x gives

$$\partial_x \chi_i = \frac{1}{m_i} \partial_x y_i \Big(\textstyle\sum_j \frac{y_j}{m_j} \Big)^{-1} - \frac{y_i}{m_i} \Big(\textstyle\sum_j \frac{y_j}{m_j} \Big)^{-2} \Big(\textstyle\sum_j \frac{1}{m_j} \partial_x y_j \Big). \tag{4.40}$$

To write this more compactly we introduce the number averaged molecular weight $\overline{m}$, defined by

$$\overline{m} := \textstyle\sum_j m_j \chi_j = \Big(\textstyle\sum_j \frac{y_j}{m_j} \Big)^{-1}. \tag{4.41}$$

Using $\overline{m}$, equation (4.40) can be simplified to

$$\partial_x \chi_i = \frac{\overline{m}}{m_i} \partial_x y_i - \frac{y_i}{m_i} \overline{m}^2 \textstyle\sum_j \frac{1}{m_j} \partial_x y_j. \tag{4.42}$$

To obtain (4.42) in matrix form, the Kronecker delta δ_{ij} is introduced. With the Kronecker delta and introducing a vector $\boldsymbol{\mu}$ with components $\mu_i := \overline{m}/m_i$, equation (4.42) can be written as

$$\partial_x \chi_i = \mu_i \textstyle\sum_j \big(\delta_{ij} - y_i \mu_j \big) \partial_x y_j, \tag{4.43}$$

which in matrix-vector form reads

$$\partial_x \boldsymbol{\chi} = \operatorname{diag}(\boldsymbol{\mu}) \left(\boldsymbol{I} - \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}}\right) \partial_x \boldsymbol{y}. \tag{4.44}$$

However, this linear transformation is not invertible, which is due to the constraint on the mass fractions. Since $\sum_i y_i = 1$, it is clear that

$$\boldsymbol{1}^{\mathrm{T}} \partial_x \boldsymbol{y} = \partial_x (\boldsymbol{1}^{\mathrm{T}} \boldsymbol{y}) = 0. \tag{4.45}$$

This phenomenon arises here since for N_{s} species the vector $\partial_x \boldsymbol{y}$ has only $N_{\mathrm{s}} - 1$ independent components. From the definition in (4.41) it is obvious that $\boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{y} = 1$. Consequently,

$$\boldsymbol{1}^{\mathrm{T}} \operatorname{diag}(\boldsymbol{\mu}) \left(\boldsymbol{I} - \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}}\right) = \mu^{\mathrm{T}} \left(\boldsymbol{I} - \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}}\right) = \boldsymbol{0}, \tag{4.46}$$

hence $\operatorname{diag}(\boldsymbol{\mu}) \left(\boldsymbol{I} - \boldsymbol{y}\mu^{\mathrm{T}}\right)$ is singular. To regularize the transformation in (4.44) we seek a new transformation of the form

$$\partial_x \boldsymbol{\chi} = \operatorname{diag}(\boldsymbol{\mu}) \left(\boldsymbol{I} - \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}} + \gamma \boldsymbol{a}\boldsymbol{a}^{\mathrm{T}}\right) \partial_x \boldsymbol{y}, \tag{4.47}$$

for some scalar $\gamma > 0$ to be determined shortly and a vector $\boldsymbol{a}$ that is chosen such that

$$\boldsymbol{a}^{\mathrm{T}} \partial_x \boldsymbol{y} = 0, \tag{4.48}$$

such that the regularization will not change $\partial_x \boldsymbol{\chi}$. Using the constraint (4.45) this vector $\boldsymbol{a}$ can be identified as $\boldsymbol{a} = \boldsymbol{1}$. The resulting regularized system is then given by:

$$\partial_x \boldsymbol{\chi} = \operatorname{diag}(\boldsymbol{\mu}) \left(\boldsymbol{I} - \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}} + \gamma \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}\right) \partial_x \boldsymbol{y}. \tag{4.49}$$

To determine γ we left-multiply (4.49) by $\boldsymbol{1}^{\mathrm{T}}$, resulting in

$$\boldsymbol{1}^{\mathrm{T}} \partial_x \boldsymbol{\chi} = \boldsymbol{\mu}^{\mathrm{T}} \left(\boldsymbol{I} - \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}} + \gamma \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}\right) \partial_x \boldsymbol{y}. \tag{4.50}$$

Using $\boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{y} = 1$ and the requirement that $\boldsymbol{1}^{\mathrm{T}} \partial_x \boldsymbol{\chi} = \boldsymbol{1}^{\mathrm{T}} \partial_x \boldsymbol{y}$ we obtain

$$\boldsymbol{1}^{\mathrm{T}} \partial_x \boldsymbol{\chi} = \gamma \boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{1}\boldsymbol{1}^{\mathrm{T}} \partial_x \boldsymbol{y}, \tag{4.51}$$

from which it is obvious that

$$\gamma = \frac{1}{\boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{1}} > 0. \tag{4.52}$$

The regularized relation between $\partial_x \boldsymbol{\chi}$ and $\partial_x \boldsymbol{y}$ is thus given by

$$\partial_x \boldsymbol{\chi} = \widetilde{\boldsymbol{\mathcal{M}}} \partial_x \boldsymbol{y} = \operatorname{diag}(\boldsymbol{\mu}) \left(\boldsymbol{I} - \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}} + \frac{\boldsymbol{1}\boldsymbol{1}^{\mathrm{T}}}{\boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{1}}\right) \partial_x \boldsymbol{y}. \tag{4.53}$$

It can be shown by computing the inverse relation by applying the Sherman-Morrison formula [75] twice and using $\boldsymbol{y}^{\mathrm{T}} \boldsymbol{1} = 1$ that the matrix $\widetilde{\boldsymbol{M}}$ in (4.53) is indeed regular, and the inverse relation reads;

$$\partial_x \boldsymbol{y} = \left(\boldsymbol{I} + \left(1 + \frac{N_{\mathrm{s}}}{\boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{1}}\right) \boldsymbol{y}\boldsymbol{\mu}^{\mathrm{T}} - \boldsymbol{y}\boldsymbol{1}^{\mathrm{T}} - \frac{\boldsymbol{1}\boldsymbol{\mu}^{\mathrm{T}}}{\boldsymbol{\mu}^{\mathrm{T}} \boldsymbol{1}}\right) \operatorname{diag}(\boldsymbol{\mu})^{-1} \partial_x \boldsymbol{\chi}. \tag{4.54}$$

Finally, we want to express the diffusive mass flux $\boldsymbol{\phi}$ in the form $\boldsymbol{\phi} = -\boldsymbol{\mathcal{E}}\partial_x \boldsymbol{y}$, where $\boldsymbol{\mathcal{E}}$ is referred to as the diffusion matrix. To that purpose, we have to combine the relations (4.34) for the diffusion velocity $\boldsymbol{u}$ and (4.53), relating the derivatives $\partial_x \boldsymbol{\chi}$ and $\partial_x \boldsymbol{y}$. This way we obtain

$$\begin{aligned} \boldsymbol{\phi} =& \ \mathrm{diag}(\rho \boldsymbol{y})\boldsymbol{u} \\ =& - \mathrm{diag}(\rho \boldsymbol{y})\widetilde{\boldsymbol{D}}_{\mathrm{amb}}\partial_x \boldsymbol{\chi} \\ =& - \mathrm{diag}(\rho \boldsymbol{y})\widetilde{\boldsymbol{D}}_{\mathrm{amb}}\widetilde{\boldsymbol{\mathcal{M}}}\partial_x \boldsymbol{y} \\ =& -\boldsymbol{\mathcal{E}}\partial_x \boldsymbol{y} \end{aligned} . \tag{4.55}$$

Finally, the mass balance for a species i is then given by

$$\partial_t(\rho y_i) + \partial_x\big(\rho v y_i + \textstyle\sum_j \mathcal{E}_{ij}\partial_x y_j\big) = m_i \omega_i, \tag{4.56}$$

which in matrix-vector form reads

$$\partial_t(\rho \boldsymbol{y}) + \partial_x(\rho v \boldsymbol{y} + \boldsymbol{\mathcal{E}}\partial_x \boldsymbol{y}) = \mathrm{diag}(\boldsymbol{m})\boldsymbol{\omega}. \tag{4.57}$$

4.3 Stoichiometric transformation

Fortunately, for chemical reactions, some source terms in the system of mass balances (4.56) can be eliminated using linear transformations. The idea is to use reaction invariants, such as $\boldsymbol{1}^{\mathrm{T}}\boldsymbol{y} = 1$ and $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n} = 0$ (assuming a quasi-neutral plasma), to linearly transform the source vector. These two properties can also be exploited to simplify the system.

This section elaborates on constructing a linear transformation that eliminates part of the source term, and allows for the exploitation of $\boldsymbol{1}^{\mathrm{T}}\boldsymbol{y} = 1$ and $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n} = 0$. First, an example for a simple argon system is presented in Section 4.3.1. Second, in Section 4.3.2, this example is generalized to a generic set of species and reactions. To construct the linear transformation, we propose a method that combines techniques presented in [61] by Kräutle et al. and in [59] by Rini et al. In the work of Kräutle a linear transformation was used in the context of porous media, to simplify the system of equations and partly eliminate the source vector. However, here we expand on a transformation similar to what was presented in the context of LTE systems by Rini to eliminate source terms. Here we use LTE as defined in [76], section 3.1.5 and section 4.4.4 from [77]. LTE requires that collisional processes are in local equilibrium. I.e. four conditions have to be locally satisfied;

1. The translational energy distribution is Maxwellian.
2. The population of excited states is described by a Boltzmann distribution.

3. Ionization processes can be described by the Saha equation.
4. The chemical equilibrium is described by a dissociation balance, e.g. Guldberg-Waage [78].

In the present work, we follow the LTE method for calculating the quasi-elemental fractions, but the system of equations for the mass fractions is completed by additional mass balances, rather than by a local application of thermodynamic equilibrium relations.

4.3.1 Three-component argon system

4.3.1.1 Eliminating part of the source term

To motivate the idea of linearly combining the conservation laws, we consider the particle balance for an argon system with three species; Ar^+, Ar and e^-, and the following (net) reactions:

$$\mathrm{Ar} \rightleftharpoons \mathrm{Ar}^+ + \mathrm{e}^-. \tag{4.58}$$

For this simple argon system, the particle balances are given by the set of equations,

$$\frac{\partial n_{\mathrm{Ar}^+}}{\partial t} + \frac{\partial \mathit{\Gamma}_{\mathrm{Ar}^+}}{\partial x} = \omega_{\mathrm{Ar}^+}, \tag{4.59a}$$

$$\frac{\partial n_{\mathrm{Ar}}}{\partial t} + \frac{\partial \mathit{\Gamma}_{\mathrm{Ar}}}{\partial x} = \omega_{\mathrm{Ar}}, \tag{4.59b}$$

$$\frac{\partial n_{\mathrm{e}^-}}{\partial t} + \frac{\partial \mathit{\Gamma}_{\mathrm{e}^-}}{\partial x} = \omega_{\mathrm{e}^-}, \tag{4.59c}$$

where we have introduced $\mathit{\Gamma}_i$ as the particle flux of species i. Note, however, that there are restrictions on the source terms, viz. conservation of charge and conservation of number of argon nuclei:

1. Conservation of charge:

$$\omega_{\mathrm{Ar}^+} = \omega_{\mathrm{e}^-}, \tag{4.60a}$$

for every positively charged argon ion introduced in the system, a corresponding electron must be produced as well.

2. Conservation of argon nuclei:

$$\omega_{\mathrm{Ar}^+} = -\omega_{\mathrm{Ar}}, \tag{4.60b}$$

in the reaction set (4.58) argon can only be converted between its ionic state Ar^+ and normal state Ar, but there is no net production of the chemical element argon.

It is possible to take linear combinations of the equations in (4.59) such that two terms on the right-hand side vanish. For the remaining equation we can take any linear combination, as long as it does not result in a linear dependency between the new set of equations. A possible combination that achieves this is given by,

$$\frac{\partial}{\partial t}(n_{\mathrm{Ar}^+} - n_{\mathrm{Ar}} + n_{\mathrm{e}^-}) + \frac{\partial}{\partial x}(\mathit{\Gamma}_{\mathrm{Ar}^+} - \mathit{\Gamma}_{\mathrm{Ar}} + \mathit{\Gamma}_{\mathrm{e}^-}) = \omega_{\mathrm{Ar}^+} - \omega_{\mathrm{Ar}} + \omega_{\mathrm{e}^-}, \tag{4.61a}$$

$$\frac{\partial}{\partial t}(n_{\mathrm{Ar}^+} + n_{\mathrm{Ar}}) + \frac{\partial}{\partial x}(\mathit{\Gamma}_{\mathrm{Ar}^+} + \mathit{\Gamma}_{\mathrm{Ar}}) = \omega_{\mathrm{Ar}^+} + \omega_{\mathrm{Ar}} = 0, \tag{4.61b}$$

$$\frac{\partial}{\partial t}(-n_{\mathrm{Ar}^+} + n_{\mathrm{e}^-}) + \frac{\partial}{\partial x}(-\mathit{\Gamma}_{\mathrm{Ar}^+} + \mathit{\Gamma}_{\mathrm{e}^-}) = -\omega_{\mathrm{Ar}^+} + \omega_{\mathrm{e}^-} = 0, \tag{4.61c}$$

by convention the homogeneous equations are positioned last. Note that the linear combination that leads to the elimination of two source terms is *not* unique. Any linear combination of (4.61b) and (4.61c), still results in a zero source term. Neither is the choice of (4.61a) unique.

Note that the conversion of (4.59) into (4.61) can be formulated as a matrix multiplication of the original system of equations. I.e. we multiply the system

$$\frac{\partial \boldsymbol{n}}{\partial t} + \frac{\partial \boldsymbol{\Gamma}}{\partial x} = \boldsymbol{\omega}, \tag{4.62}$$

from the left by the constant matrix

$$\boldsymbol{M} = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \tag{4.63}$$

where $\boldsymbol{n}$, $\boldsymbol{\Gamma}$ and $\boldsymbol{s}$ are the vector of particle densities, fluxes and chemical source terms respectively;

$$\boldsymbol{n} = \begin{bmatrix} n_{\mathrm{Ar}^+} \\ n_{\mathrm{Ar}} \\ n_{\mathrm{e}^-} \end{bmatrix}, \quad \boldsymbol{\Gamma} = \begin{bmatrix} \mathit{\Gamma}_{\mathrm{Ar}^+} \\ \mathit{\Gamma}_{\mathrm{Ar}} \\ \mathit{\Gamma}_{\mathrm{e}^-} \end{bmatrix}, \quad \boldsymbol{\omega} = \begin{bmatrix} \omega_{\mathrm{Ar}^+} \\ \omega_{\mathrm{Ar}} \\ \omega_{\mathrm{e}^-} \end{bmatrix}. \tag{4.64}$$

The matrix $\boldsymbol{M}$ is constructed such that the last two rows of $\boldsymbol{M}$ constitute an orthogonal complement to the source vector $\boldsymbol{\omega}$;

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_1^{\mathrm{T}} \\ \boldsymbol{M}_2^{\mathrm{T}} \\ \boldsymbol{M}_3^{\mathrm{T}} \end{bmatrix}, \tag{4.65}$$

where by construction $\langle \boldsymbol{\omega} \rangle^{\perp} = \langle \boldsymbol{M}_2, \boldsymbol{M}_3 \rangle$, and consequently $\boldsymbol{M}_2^{\mathrm{T}} \boldsymbol{\omega} = 0$ and $\boldsymbol{M}_3^{\mathrm{T}} \boldsymbol{\omega} = 0$. Using this transformation matrix $\boldsymbol{M}$, a transformed system of particle balance equations can be introduced;

$$\frac{\partial}{\partial t}\left(\boldsymbol{M}\boldsymbol{n}\right)+\frac{\partial}{\partial x}\left(\boldsymbol{M}\boldsymbol{\Gamma}\right)=\boldsymbol{M}\boldsymbol{\omega}. \tag{4.66}$$

To construct the matrix $\boldsymbol{M}$, in (4.63), the concept of a "quasi-element basis" must be introduced, similar to what was presented in Rini [59]. However, we use the broader notion of element as inspired by the definition of Fan et al. in [62], who defined an element as "an atom, or a compound that does not partition into smaller compounds by chemical reactions in the system under consideration". In this work we define a quasi-element as: "an atom, or a compound present in the mixture, of which the amount is unchanged by all chemical reactions in the system under consideration".

We add to this definition the concept of a "quasi-element basis"; "if all non-quasi-element species present in a mixture can be constructed in chemical reactions from the set of quasi-elements, then this set of quasi-elements is called a quasi-element basis." Spectator particles without reactions are necessarily quasi-elements. An algorithm for finding such a basis, from an arbitrary chemical reaction system is elaborated in Section 4.3.2. Note that our definition allows a transformation similar to the one introduced in [59] also for *non-LTE*-systems, where formation reactions allow every species to be converted into other species as long as the stoichiometry is correct. To illustrate why choosing chemical elements as basis species would not suffice in general, consider the example mixture from [58] containing HF and $(\mathrm{HF})_6$. The species HF and $(\mathrm{HF})_6$ can be converted via the reaction

$$\mathrm{HF} \rightleftarrows (\mathrm{HF})_6, \tag{4.67}$$

indeed there would be no (net) production of H nor of F. However, this may lead one to the conclusion that two source terms could be eliminated, which is not the case.

To elaborate on these definitions, consider the mixture with $\mathrm{Ar}^+, \mathrm{Ar}, \mathrm{e}^-$ which can be used as a simple model for argon. Note that if we choose Ar and e^- as quasi-elements, the remaining species can be constructed from these species

$$\mathrm{Ar}^+ \rightleftarrows \mathrm{Ar}-\mathrm{e}^-. \tag{4.68}$$

Since all species can be constructed from quasi-elements via chemical reactions, the chosen set of quasi-elements forms a quasi-element basis. As such the species Ar^+ can be seen as the quasi-element Ar minus e^-. Since by definition quasi-elements cannot be produced or consumed in chemical reactions, one can conclude that for any chemistry set which has Ar and e^- as its only quasi-elements;

$$\{\boldsymbol{\omega}\} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{\mathrm{Ar}^+} \\ \omega_{\mathrm{Ar}} \\ \omega_{\mathrm{e}^-} \end{bmatrix} = \mathbf{0}, \tag{4.69}$$

i.e., the chemical sources for quasi-elemental are zero. Here quasi-element quantities are denoted by $\{\}$, for example $\{n\}$ would be the number density of a quasi-element. A useful consequence of this is that it is possible to also construct a linear relation to obtain the total number density of each quasi-element;

$$\{\boldsymbol{n}\} = \begin{bmatrix} \{n_{\mathrm{Ar}}\} \\ \{n_{\mathrm{e}^-}\} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_{\mathrm{Ar}^+} \\ n_{\mathrm{Ar}} \\ n_{\mathrm{e}^-} \end{bmatrix}. \tag{4.70}$$

The first entry, $\{n_{\mathrm{Ar}}\}$, is the total amount of “quasi-elemental Ar” present in all species combined. The second entry is the total amount of “quasi-elemental e^-”. We use the convention that basis species are put at the end of a vector, and define N_e as the number of quasi-elements. In the argon example $N_\mathrm{e} = 2$.

Note that in LTE the chemical composition is given by thermodynamic equilibrium relations. These relations give the ratio between the concentrations of two species, irrespective of the specific reaction path between them. However, in non-LTE the specific reaction path and reaction rate is relevant. The reaction network does not have to include the formation reactions, nor linear combinations of specific formation reactions. As a consequence the reaction network does not have to provide a possible path to convert two species.

For example, consider a mixture at 293K with H_2, O_2 a trace amount of H_2O and one reaction $2H_2 + O_2 \rightarrow 2H_2O$. Even though this reaction is allowed, the reaction rate may be negligible at this temperature. In such a case the modeler may choose to not include this reaction as part of a chemical reduction technique. By omitting this reaction, the reaction network is “incomplete”, as there is no available conversion between H_2, O_2 and H_2O.

In terms of linear algebra, if the reaction is included the source vector exists in a one-dimensional subspace of $\mathrm{span}\{\omega_\mathrm{H}\vec{e}_1, \omega_\mathrm{O}\vec{e}_2, \omega_{\mathrm{H_2O}}\vec{e}_3\}$ for an arbitrary set of basis vectors $\vec{e}_1, \vec{e}_2$ and $\vec{e}_3$ in $\mathbb{R}^3$. If the reaction is removed the source vector exists in a zero-dimensional subspace, i.e., no chemical reactions.

The challenge of completing non-LTE reaction sets is founded in that non-LTE requires the specific reaction mechanism, and corresponding reaction rates to be known. First, a presented reaction set may not be valid under all conditions, for example the mechanism described in [79] is only valid for temperatures between 950 and 2500K. Second, several reaction mechanisms may exist for similar chemistry sets, for example the 25 species CO_2 network presented in [80] and the 72 species CO_2 network in [81, 82]. Third, complex reaction mechanisms can lead to excessive computational load for one, two and three-dimensional studies and chemical reduction techniques have to be applied. A practical example would be the dissociation of CO_2; in LTE the ratio of CO_2 to CO would be determined by a thermodynamic equilibrium relation like Guldberg-Waage. However, in the case of non-LTE the dissociation process involves a complex reaction mechanism involving the vibrational states of CO_2 [81].

4.3.1.2 Exploiting spatial invariants

With the transformation matrix $\boldsymbol{M}$ constructed, part of the source-vector can be eliminated. However, it was also noted that the construction of such a transformation matrix is *not* unique. One can left-multiply the bottom two rows of the matrix $\boldsymbol{M}$ constructed in the argon example by any arbitrary, non-singular matrix of size 2×2 without destroying the property that the last two source terms are eliminated.

We exploit this additional degree of freedom by altering $\boldsymbol{M}$ such that one of the transformed variables becomes $\mathbf{1}^{\mathrm{T}}\boldsymbol{y}$, and another one becomes; $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n}$. This reformulation can then be used to enforce $\mathbf{1}^{\mathrm{T}}\boldsymbol{y} = 1$ and $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n} = 0$ throughout the domain. To simplify the analysis we consider the case with only one charged species in the set of quasi-elements.

If we are working in terms of mass fractions instead of number densities the transformation matrix $\boldsymbol{M}$ has to be scaled first to account for this. This can be seen by comparing the particle balances, equation (4.1), and the mass balances, equation (4.5). The transformation matrix when working with mass fractions becomes

$$\boldsymbol{T} := \operatorname{diag}(\boldsymbol{m})\boldsymbol{M}\operatorname{diag}(\boldsymbol{m})^{-1}, \tag{4.71}$$

this is equivalent to first transforming the source vector for mass fractions to the one required for number densities, next applying matrix $\boldsymbol{M}$, and finally converting back to mass fractions. The transformed system in terms of mass fractions is given in matrix-vector form as

$$\frac{\partial}{\partial t}(\rho \boldsymbol{T}\boldsymbol{y}) + \frac{\partial}{\partial x}(\boldsymbol{T}\boldsymbol{\Gamma}_{\mathrm{y}}) = \boldsymbol{T}\operatorname{diag}(\boldsymbol{m})\boldsymbol{\omega}, \tag{4.72}$$

with $\boldsymbol{\Gamma}_{\mathrm{y}}$ the mass fluxes. Continuing with the argon example, with Ar and e^- chosen as quasi-elements, we get the transformation matrix $\boldsymbol{T}$ as follows;

$$\boldsymbol{T} = \begin{bmatrix} 1 & -\frac{m_{\mathrm{Ar}+}}{m_{\mathrm{Ar}}} & \frac{m_{\mathrm{Ar}+}}{m_{\mathrm{e}-}} \\ \frac{m_{\mathrm{Ar}}}{m_{\mathrm{Ar}+}} & 1 & 0 \\ -\frac{m_{\mathrm{e}-}}{m_{\mathrm{Ar}+}} & 0 & 1 \end{bmatrix}. \tag{4.73}$$

To uncover an interpretation of the transformed variables, we first carry out the matrix product $\boldsymbol{Ty}$, from which it can be seen that

$$[\boldsymbol{Ty}]_3 = -\frac{m_{\mathrm{e}^-}}{m_{\mathrm{Ar}^+}} y_{\mathrm{Ar}^+} + y_{\mathrm{e}^-}, \tag{4.74}$$

which can be identified as the condition for quasi-neutrality;

$$[\boldsymbol{Ty}]_3 = \frac{m_{\mathrm{e}^-}}{\rho}(-n_{\mathrm{Ar}^+} + n_{\mathrm{e}^-}), \tag{4.75}$$

the term in parentheses is proportional to the charge density, and since we assume a quasi-neutral plasma $[\boldsymbol{Ty}]_3 = 0$ throughout the domain. In Section 4.3.2.3 it will be shown that the charge density of the quasi-elements must equal the charge density of all species. Hence, if quasi-neutrality is assumed, the charge density of the quasi-elements must also be zero.

A second spatial invariant is present in the system, namely, conservation of mass. Starting from the quasi-element number densities

$$\{n_{\mathrm{Ar}}\} = n_{\mathrm{Ar}^+} + n_{\mathrm{Ar}}, \quad \{n_{\mathrm{e}^-}\} = -n_{\mathrm{Ar}} + n_{\mathrm{e}^-}, \tag{4.76}$$

we obtain the quasi-element mass fractions

$$\{y_{\mathrm{Ar}}\} = \frac{m_{\mathrm{Ar}}}{m_{\mathrm{Ar}^+}} y_{\mathrm{Ar}^+} + y_{\mathrm{Ar}}, \quad \{y_{\mathrm{e}^-}\} = -\frac{m_{\mathrm{e}^-}}{m_{\mathrm{Ar}^+}} y_{\mathrm{Ar}^+} + y_{\mathrm{e}^-}, \tag{4.77}$$

from which it can be seen that

$$\{y_{\mathrm{Ar}}\} + \{y_{\mathrm{e}^-}\} = y_{\mathrm{Ar}^+} + y_{\mathrm{Ar}} + y_{\mathrm{e}^-} = 1, \tag{4.78}$$

which is also constant throughout the entire domain. To achieve a transformation matrix $\widetilde{\boldsymbol{T}}$ that results in a variable representing $\sum_i y_i = 1$, we multiply $\boldsymbol{T}$ from the left by another matrix that performs a summation of the quasi-element mass fractions $\{y_{\mathrm{Ar}}\} + \{y_{\mathrm{e}^-}\}$;

$$\widetilde{\boldsymbol{T}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{T} = \begin{bmatrix} 1 & -\frac{m_{\mathrm{Ar}^+}}{m_{\mathrm{Ar}}} & \frac{m_{\mathrm{Ar}^+}}{m_{\mathrm{e}^-}} \\ \frac{m_{\mathrm{Ar}}}{m_{\mathrm{Ar}^+}} - \frac{m_{\mathrm{e}^-}}{m_{\mathrm{Ar}^+}} & 1 & 1 \\ -\frac{m_{\mathrm{e}^-}}{m_{\mathrm{Ar}^+}} & 0 & 1 \end{bmatrix}. \tag{4.79}$$

This operation only affects the second variable, which now reads,

$$[\widetilde{\boldsymbol{T}}\boldsymbol{y}]_2 = \left(\frac{m_{\mathrm{Ar}}}{m_{\mathrm{Ar}^+}} - \frac{m_{\mathrm{e}^-}}{m_{\mathrm{Ar}^+}}\right) y_{\mathrm{Ar}^+} + y_{\mathrm{Ar}} + y_{\mathrm{e}^-}, \tag{4.80}$$

then, since $m_{\mathrm{Ar}} - m_{\mathrm{e}^-} = m_{\mathrm{Ar}^+}$, it follows that $[\widetilde{\boldsymbol{T}}\boldsymbol{y}]_2 = 1$. I.e. $\{y_{\mathrm{Ar}}\} + \{y_{\mathrm{e}^-}\} = 1$ throughout the domain. It will be shown in Section 4.3.2.3 that this property generalizes to a generic set of species as well. With this we arrive at the final form of the transformation for the argon system;

$$\frac{\partial}{\partial t}\left(\rho\widetilde{\boldsymbol{T}}\boldsymbol{y}\right)+\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{T}}\boldsymbol{\Gamma}_{\mathrm{y}}\right)=\widetilde{\boldsymbol{T}}\,\mathrm{diag}(\boldsymbol{m})\boldsymbol{\omega}, \tag{4.81}$$

where now two components of the source term have been eliminated and two transformed variables are known beforehand and can be set constant throughout the domain. This results in a modest reduction of the number of equations. Additionally, by choosing the transformation in this way it is possible to enforce both $\mathbf{1}^{\mathrm{T}}\boldsymbol{y}=1$ and $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n}=0$ explicitly.

An example where even further reduction can be achieved by exploiting spatial invariants is in the context of atmospheric re-entry studies, here it is common to assume the nitrogen and oxygen element fractions as constant [83].

4.3.2 Transforming a generic system

4.3.2.1 Constructing the transformation matrix

A generic reaction r in a mixture with N_{s} species X_p can be written as

$$\sum_{p=1}^{N_{\mathrm{S}}}\nu_{\mathrm{f}}(p,r)X_p\rightarrow\sum_{p=1}^{N_{\mathrm{S}}}\nu_{\mathrm{b}}(p,r)X_p. \tag{4.82}$$

The coefficients $\nu_{\mathrm{f}}(p,r)$ and $\nu_{\mathrm{b}}(p,r)$ represent the number of particles of type X_p that are consumed or produced by the reaction r, respectively. The two sets of coefficients for a collection of N_{r} reactions form two matrices $\boldsymbol{\nu}_{\mathrm{f}}=(\nu_{\mathrm{f}}(p,r))$, $\boldsymbol{\nu}_{\mathrm{b}}=(\nu_{\mathrm{b}}(p,r))\in\mathbb{Q}^{N_{\mathrm{S}}\times N_{\mathrm{r}}}$. The coefficients of reaction r are located in column r of these matrices. The net number of particles of type X_p that is produced in reaction r is given by the number $\nu(p,r)=\nu_{\mathrm{b}}(p,r)-\nu_{\mathrm{f}}(p,r)$.

Let us now define a matrix with coefficients $M_{\mathrm{e}}(i,p)$ as the number of quasi-elements of type i that is present in a species of type p. These coefficients $M_{\mathrm{e}}(i,p)$ are the stoichiometric coefficients of the species p with respect to the quasi-elements chosen as a basis. These coefficients form a matrix $\boldsymbol{M}_{\mathrm{e}}\in\mathbb{Q}^{N_{\mathrm{e}}\times N_{\mathrm{S}}}$. The matrix $\boldsymbol{M}_{\mathrm{e}}$ can always be decomposed as

$$\boldsymbol{M}_{\mathrm{e}}=\left[\begin{array}{c|c}\boldsymbol{B} & \boldsymbol{I}_{N_{\mathrm{e}}}\end{array}\right], \tag{4.83}$$

where $\boldsymbol{B}$ is a matrix with N_{e} rows and $N_{\mathrm{s}}-N_{\mathrm{e}}$ columns and $\boldsymbol{I}_{N_{\mathrm{e}}}$ is the identity matrix of rank N_{e}. To obtain the square transformation matrix $\boldsymbol{M}$, we use an orthogonal complement to $\boldsymbol{M}_{\mathrm{e}}$ to ensure $\boldsymbol{M}$ is invertible;

$$\boldsymbol{M}=\begin{bmatrix}\boldsymbol{I}_{N_{\mathrm{S}}-N_{\mathrm{e}}} & -\boldsymbol{B}^{\mathrm{T}}\\ \boldsymbol{B} & \boldsymbol{I}_{N_{\mathrm{e}}}\end{bmatrix}. \tag{4.84}$$

Note, that this specific choice also offers a clear interpretation to the top part of $\boldsymbol{M}$, i.e., each row describes which combination of quasi-elements it takes to create one of the remaining species. Additionally, this specific form is used in [59] to link species enthalpies to reaction enthalpies.

For a reaction r, with coefficients given by $\nu_\mathrm{f}(:,r)$ and $\nu_\mathrm{b}(:,r)$, the rth column of the matrices $\boldsymbol{\nu}_\mathrm{f}$ and $\boldsymbol{\nu}_\mathrm{b}$ indicates the reaction components. The total number of quasi-elements that are consumed or produced by one occurrence of the reaction r are given by the column vectors $\boldsymbol{M}_\mathrm{e}\boldsymbol{\nu}_f(:,r)$ and $\boldsymbol{M}_\mathrm{e}\nu_\mathrm{b}(:,r)$, respectively. However, since quasi-elements are conserved in reactions we conclude that for every reaction r

$$\boldsymbol{M}_\mathrm{e}\boldsymbol{\nu}_f(:,r) = \boldsymbol{M}_\mathrm{e}\boldsymbol{\nu}_b(:,r), \tag{4.85}$$

and as a consequence, in terms of the matrix $\boldsymbol{\nu} = \boldsymbol{\nu}_\mathrm{b} - \boldsymbol{\nu}_\mathrm{f}$,

$$\boldsymbol{M}_\mathrm{e}\boldsymbol{\nu} = \boldsymbol{0}. \tag{4.86}$$

The net source obtained by taking all reactions into account is compactly written as

$$\boldsymbol{\omega} = \boldsymbol{\nu}\boldsymbol{Z}, \tag{4.87}$$

where the vector $\boldsymbol{Z}$ describes the volumetric rate of each reaction. Multiplying (4.87) with $\boldsymbol{M}_\mathrm{e}$ we get the quasi-elemental production rates; since matrix multiplication is associative and $\boldsymbol{M}_\mathrm{e}(\boldsymbol{\nu}_\mathrm{b} - \boldsymbol{\nu}_\mathrm{f}) = \boldsymbol{0}$ we again find the result that

$$\boldsymbol{M}_\mathrm{e}\boldsymbol{\omega} = \boldsymbol{M}_\mathrm{e}\boldsymbol{\nu}\boldsymbol{Z} = \boldsymbol{0}. \tag{4.88}$$

Relation (4.88) expresses once more that no quasi-elements are produced or destroyed in reactions. This completes the derivation that for generic systems a matrix $\boldsymbol{M}$ can be used to partially eliminate the source terms.

4.3.2.2 Constructing a quasi-element basis

To find a suitable set of quasi-elements $\mathcal{E}$ in the basis, Algorithm 2 can be applied. First we construct the set $\mathcal{U}$ of all species present in the mixture, including spectator particles that do not take part in any chemical reaction. Next, we reduce the original reactions in the system to their net form. For example the reaction

$$\mathrm{e}^- + \mathrm{Ar} \to 2\mathrm{e}^- + \mathrm{Ar}^+, \tag{4.89a}$$

would be reduced to the net reaction

$$\mathrm{Ar} \to \mathrm{e}^- + \mathrm{Ar}^+. \tag{4.89b}$$

We then write the reactions as a collection of unordered sets $\mathcal{S}$ and run Algorithm 2. After it completes, the set $\mathcal{E}$ will contain a suitable set of quasi-elements, the set $\mathcal{R}$ will contain all remaining species. Note that this algorithm does not provide a unique result, since the step "Choose a species i from $\mathcal{U}$" allows for an arbitrary choice. An example for a 9-species mixture is provided in Section 4.C. Once the basis is determined, the construction of $\boldsymbol{M}$ then follows a similar procedure as elaborated in [59]. However, in the procedure in [59] chemical elements are chosen as a basis and all conversions are assumed to exist in the context of LTE. In this work, Algorithm 2 allows a quasi-element basis to be obtained even when certain conversions are not allowed. The main characteristics of this method can be summarized as follows:

1. There is freedom of choice of basis species; cf. line 9 in Algorithm 2. This allows for further exploitation of the transformation.
2. This method automatically identifies linear dependence between species.
3. (Non-trivial) Molecules may also be chosen as quasi-elements. For example, adding the species ArH with reaction $2\mathrm{Ar} + \mathrm{H}_2 \rightarrow 2\mathrm{ArH}$ to the reaction set given in Appendix 4.C, results in a possible basis $\mathcal{E} = \{\mathrm{Au}, \mathrm{ArH}, \mathrm{e}^-, \mathrm{H}_2, \mathrm{Al}^+\}$.
4. It is possible to identify a basis for incomplete reaction sets with spectator species.

Algorithm 2 Proposed method for finding a quasi-element basis from a given reaction set.

```
 1: $\mathcal{U} = \{$All species$\}, \mathcal{R} = \{\}, \mathcal{E} = \{\}, \mathcal{S} = \{$Collection of reaction sets$\}$
 2: for each set $j$ in the collection $\mathcal{S}$ do        ▷ Spectator particles are always quasi-elements
 3:     if $j$ contains exactly 1 species then
 4:         Add this species to $\mathcal{E}$
 5:         Remove this species from $\mathcal{U}$
 6:     end if
 7: end for
 8: while $\mathcal{U} \neq \{\}$ do
 9:     Choose a species $i$ from $\mathcal{U}$
10:     Add $i$ to $\mathcal{E}$
11:     Remove $i$ from $\mathcal{U}$
12:     while there exist sets in $\mathcal{S}$, which have exactly 1 species $k$ that is not in $\mathcal{R} \cup \mathcal{E}$ do
13:         Add species $k$ to $\mathcal{R}$
14:         Remove species $k$ from $\mathcal{U}$
15:     end while
16: end while
```

4.3.2.3 quasi-elemental properties

Rini et al. [59] showed that quasi-element diffusion fluxes can be obtained from the full array of diffusion fluxes. Similarly, for a generic set of species, with a corresponding quasi-elemental basis we can relate the number density of the quasi-elements to the number density of the species, i.e., $\boldsymbol{\{n\}} = \boldsymbol{M}_{\mathrm{e}}\boldsymbol{n}$ and also the masses of the species chosen as quasi-elements, $\boldsymbol{\{m\}}$, to masses of the species; i.e.

$$m_p = \textstyle\sum_i M_b(i,p)\boldsymbol{\{m\}}_i, \tag{4.90}$$

which follows from the definition of $\boldsymbol{M}_{\mathrm{e}}$. From the definition of the mass density ρ in (4.3) we can derive that

$$\boldsymbol{\rho} = \boldsymbol{m}^{\mathrm{T}}\boldsymbol{n} = (\boldsymbol{M}_{\mathrm{e}}^{\mathrm{T}}\boldsymbol{\{m\}})^{\mathrm{T}}\boldsymbol{n} = \boldsymbol{\{m\}}^{\mathrm{T}}(\boldsymbol{M}_{\mathrm{e}}\boldsymbol{n}) = \boldsymbol{\{m\}}^{\mathrm{T}}\boldsymbol{\{n\}}, \tag{4.91}$$

i.e., the mass density of the quasi-elements and the species is the same. The quasi-elemental mass densities are defined as $\{\rho\}_i = \{m\}_i\{n\}_i$ and follow directly from transforming the vector of species mass densities;

$$\begin{aligned}\boldsymbol{\{\rho\}} &= \operatorname{diag}(\boldsymbol{\{m\}})\boldsymbol{\{n\}} \\ &= \operatorname{diag}(\boldsymbol{\{m\}})\boldsymbol{M}_{\mathrm{e}}\boldsymbol{n} \\ &= \operatorname{diag}(\boldsymbol{\{m\}})\boldsymbol{M}_{\mathrm{e}}\operatorname{diag}(\boldsymbol{m})^{-1}\boldsymbol{\rho} \\ &= \boldsymbol{T}\boldsymbol{\rho}.\end{aligned} \tag{4.92}$$

As a result the sum of the quasi-elemental mass densities is given by

$$\mathbf{1}^{\mathrm{T}}\boldsymbol{\{\rho\}} = \boldsymbol{\{m\}}^{\mathrm{T}}\boldsymbol{M}_{\mathrm{e}}\operatorname{diag}(\boldsymbol{m})^{-1}\boldsymbol{\rho} = \boldsymbol{m}^{\mathrm{T}}\operatorname{diag}(\boldsymbol{m})^{-1}\boldsymbol{\rho} = \mathbf{1}^{\mathrm{T}}\boldsymbol{\rho}. \tag{4.93}$$

This shows that the total mass density equals the sum of the quasi-elemental mass densities.

The quasi-elemental mass fractions are given by the relation $\{y\}_i = \{\rho\}_i/(\mathbf{1}^{\mathrm{T}}\boldsymbol{\rho})$. Combining equations (4.92) and (4.93) we find the expected relations

$$\boldsymbol{\{y\}} = \operatorname{diag}(\boldsymbol{\{m\}})\boldsymbol{M}_{\mathrm{e}}\operatorname{diag}^{-1}(\boldsymbol{m})\boldsymbol{y}. \tag{4.94}$$

Moreover, left-multiplying the relation $\boldsymbol{\{y\}} = \boldsymbol{\{\rho\}}/(\mathbf{1}^{\mathrm{T}}\boldsymbol{\rho})$ with $\mathbf{1}^{\mathrm{T}}$ and applying (4.93) we obtain

$$\mathbf{1}^{\mathrm{T}}\boldsymbol{\{y\}} = 1. \tag{4.95}$$

Analogous results can be obtained for mixtures in which one or more of the quasi-elements carry electric charge. Then the charge of each species can be obtained from the charge of the quasi-elements via

$$\boldsymbol{q} = \boldsymbol{M}_{\mathrm{e}}^{\mathrm{T}}\boldsymbol{\{q\}}. \tag{4.96}$$

The charge density ρ_c of the mixture can be written as

$$\rho_{\mathrm{c}} = \boldsymbol{q}^{\mathrm{T}}\boldsymbol{n} = \{\boldsymbol{q}\}^{\mathrm{T}}\{\boldsymbol{n}\}, \tag{4.97}$$

in a derivation similar to what was shown for (4.95), where we replace $\{\boldsymbol{m}\}$ and $\boldsymbol{m}$ with $\{\boldsymbol{q}\}$ and $\boldsymbol{q}$, respectively. With this result we arrive at the conclusion that the sum of the quasi-element charge densities is equal to the sum of the species charge densities.

Therefore, by invoking relations (4.95) and (4.97) we can construct a matrix $\widetilde{\boldsymbol{T}}$ similar to the one shown for the argon example in (4.79), such that two of the transformed variables are constant.

4.4 Discretization

As in general no analytical solution is known to the full multi-component particle balance set, a numerical approximation has to be computed instead. A generic one-dimensional system of conservation laws can be written as

$$\frac{\partial \boldsymbol{\varphi}}{\partial t} + \frac{\partial \boldsymbol{\Gamma}}{\partial x} = \boldsymbol{s}, \tag{4.98a}$$

with the flux vector given by;

$$\boldsymbol{\Gamma} := \boldsymbol{U}\boldsymbol{\varphi} - \boldsymbol{\mathcal{E}}\frac{\partial \boldsymbol{\varphi}}{\partial x}. \tag{4.98b}$$

In the case of multicomponent diffusion $\boldsymbol{\varphi} = \boldsymbol{y}$ or $\boldsymbol{\varphi} = \widetilde{\boldsymbol{T}}\boldsymbol{y}$, the original, or transformed mass fractions, respectively. Note that the components of $\boldsymbol{\varphi}$ are coupled via the diffusion matrix $\boldsymbol{\mathcal{E}}$ and via the source terms. The flux vector $\boldsymbol{\Gamma}$ is a generic flux vector corresponding to $\boldsymbol{\varphi}$. One can choose to transform the discretized system, or choose to discretize the transformed system. In this work we first transform, then discretize, as it requires two transformations less. Both options yield the same discrete system of equations.

Many strategies exist to spatially discretize such partial differential equations, including finite element, finite difference, finite volume and spectral methods. Finite Volume Methods (FVM) are based on the integral formulation of the conservation law, which can be obtained by integrating (4.98a) over an arbitrary interval $[a, b]$;

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_a^b \boldsymbol{\varphi}(x,t)\,\mathrm{d}x + \boldsymbol{\Gamma}(b,t) - \boldsymbol{\Gamma}(a,t) = \int_a^b \boldsymbol{s}(x,t)\,\mathrm{d}x. \tag{4.99}$$

The concept of the FVM is to approximate the flux vectors $\boldsymbol{\Gamma}$ by a numerical flux $\boldsymbol{F}$ at the edges of each control volume. To compute this numerical flux vector we closely follow the derivation in reference [56].

4.4.1 Finite Volume Method

In the FVM method the domain is subdivided into a finite number of disjunct intervals; referred to as control volumes. Here a cell-centered approach is applied, as shown in Fig. 4.1. In such a configuration $\boldsymbol{\varphi}$ has to be computed in the nodal points x_j. Control volumes are defined around this nodal point, the j-th control volume extends over $[x_{j-1/2}, x_{j+1/2}]$, where $x_{j\pm1/2} := \frac{1}{2}(x_j + x_{j\pm1})$ and the width of this volume is defined as the grid size $\Delta x := x_{j-1/2} - x_{j+1/2}$.

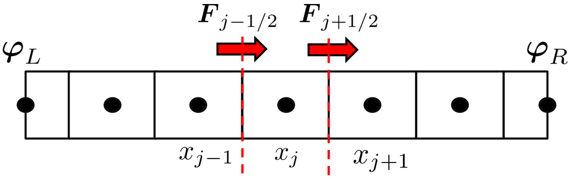


Figure 4.1: Cell centered grid used for discretization. The unknown $\boldsymbol{\varphi}$ has to be determined at each grid point x_j and the flux vector at the edges $x_{j\pm1/2}$ of an interval $I_j = [x_{j-1/2}, x_{j+1/2}]$.

Choose $[a, b] = I_j$ in (4.99), then approximating the integrals by the midpoint rule yields the semi-discrete conservation law;

$$\dot{\boldsymbol{\varphi}}_j(t)\Delta x = -(\boldsymbol{F}_{j+1/2}(t) - \boldsymbol{F}_{j-1/2}(t)) + \boldsymbol{s}_j(t)\Delta x. \tag{4.100}$$

Here $\boldsymbol{\varphi}_j(t)$ is the numerical approximation of $\boldsymbol{\varphi}(x_j, t)$, similarly $\boldsymbol{s}_j(t) := \boldsymbol{s}(x_j, t)$. Equation (4.100) states that the rate at which $\boldsymbol{\varphi}$ changes is given by the net influx into the control volume, plus the local production/consumption via the source term s_j. Alternatively this can be viewed in terms of fluxes,

$$\boldsymbol{F}_{j+1/2}(t) - \boldsymbol{F}_{j-1/2}(t) = (\boldsymbol{s}_j(t) - \dot{\boldsymbol{\varphi}}_j(t))\,\Delta x, \tag{4.101}$$

where the factor $\boldsymbol{s}_j(t) - \dot{\boldsymbol{\varphi}}_j(t)$ can be perceived as a modified source term, which we denote by $\hat{\boldsymbol{s}}_j$. In the following the explicit time-dependence (t) is omitted. Section 4.4.2 outlines the numerical approximation for the flux vectors $\boldsymbol{\Gamma}(x_{j\pm1/2})$ in terms of $\boldsymbol{\varphi}_j$ and $\boldsymbol{\varphi}_{j\pm1}$ and the sources $\hat{\boldsymbol{s}}_j$ and $\hat{\boldsymbol{s}}_{j\pm1}$.

4.4.2 Flux approximation

The next step is to find an expression for the fluxes; for each cell face at $x_{j+1/2}$ we seek to approximate the flux $\boldsymbol{\Gamma}(x_{j+1/2})$ by a "numerical flux" $\boldsymbol{F}_{j+1/2}$ in terms of the neighboring unknowns and neighboring source terms;

$$\boldsymbol{F}_{j+1/2} = \boldsymbol{\alpha}_{j+1/2}\boldsymbol{\varphi}_j - \boldsymbol{\beta}_{j+1/2}\boldsymbol{\varphi}_{j+1} + \Delta x(\boldsymbol{\gamma}_{j+1/2}\hat{\boldsymbol{s}}_j + \boldsymbol{\delta}_{j+1/2}\hat{\boldsymbol{s}}_{j+1}). \tag{4.102}$$

Assume the advection and diffusion matrices $\boldsymbol{U}$ and $\boldsymbol{\mathcal{E}}$ are piecewise constant on each interval $[x_j, x_{j+1}]$. Then the matrix coefficients $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta}$ in (4.102) are also piecewise constant and only depend on $\boldsymbol{U}$ and $\boldsymbol{\mathcal{E}}$. The matrix $\boldsymbol{U}$ is often a constant, scalar matrix (i.e., $\boldsymbol{U} = u\boldsymbol{I}$), however, for the sake of completeness we define $\boldsymbol{U}_{j+1/2}$ as the advection matrix on the cell face located at $x_{j+1/2}$. The values of $\boldsymbol{U}_{j+1/2}$ and $\boldsymbol{\mathcal{E}}_{j+1/2}$ are estimated using the arithmetic average of $\boldsymbol{\varphi}$ and evaluating the functions for $\boldsymbol{U}$ and $\boldsymbol{\mathcal{E}}$ on the position $x_{j+1/2}$; $\boldsymbol{U}_{j+1/2} \coloneqq \boldsymbol{U}(\frac{1}{2}(\boldsymbol{\varphi}_j + \boldsymbol{\varphi}_{j+1}), x_{j+1/2})$ and $\boldsymbol{\mathcal{E}}_{j+1/2} \coloneqq \boldsymbol{\mathcal{E}}(\frac{1}{2}(\boldsymbol{\varphi}_j + \boldsymbol{\varphi}_{j+1}), x_{j+1/2})$ when needed. The idea is to compute the numerical flux from the local system boundary value problem

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}x}(\boldsymbol{\Gamma}) = \frac{\mathrm{d}}{\mathrm{d}x}\left(\boldsymbol{U}_{j+1/2}\boldsymbol{\varphi} - \boldsymbol{\mathcal{E}}_{j+1/2}\frac{\mathrm{d}\boldsymbol{\varphi}}{\mathrm{d}x}\right) = \hat{\boldsymbol{s}}, \quad x_j < x < x_{j+1}, \\ &\boldsymbol{\varphi}(x_j) = \boldsymbol{\varphi}_j, \quad \boldsymbol{\varphi}(x_{j+1}) = \boldsymbol{\varphi}_{j+1}, \end{aligned} \tag{4.103}$$

and use this to approximate the flux at the cell edge. Here we give a brief outline of the derivation for the flux approximation, for more details see [56]. The numerical approximation for the flux vector $\boldsymbol{\Gamma}$ is the superposition of the flux approximations computed from the homogeneous and inhomogeneous systems;

$$\boldsymbol{\Gamma}(x_{j+1/2}) \approx \boldsymbol{F}_{j+1/2} = \boldsymbol{F}^{\mathrm{h}}_{j+1/2} + \boldsymbol{F}^{\mathrm{i}}_{j+1/2}, \tag{4.104}$$

where $\boldsymbol{F}^{\mathrm{h}}_{j+1/2}$ is the flux corresponding to the homogeneous advection-diffusion system, and $\boldsymbol{F}^{\mathrm{i}}_{j+1/2}$ is the inhomogeneous flux vector, obtained by talking into account $\boldsymbol{s}$ in the flux approximation. To simplify notation we define the variables

$$\boldsymbol{A} \coloneqq \boldsymbol{\mathcal{E}}^{-1}\boldsymbol{U}, \quad \boldsymbol{P} \coloneqq \Delta x\boldsymbol{A}, \quad \boldsymbol{S}(x) \coloneqq \int_{x_{j+1/2}}^{x} \boldsymbol{s}(\xi)\,\mathrm{d}\xi. \tag{4.105}$$

To compute all matrix functions needed, it is assumed that the matrix $\boldsymbol{A}$ has only real eigenvalues λ and a complete set of eigenvectors $\boldsymbol{v}$, which satisfy the generalized eigenvalue problem

$$(\boldsymbol{U} - \lambda\boldsymbol{\mathcal{E}})\boldsymbol{v} = \boldsymbol{0}, \tag{4.106}$$

such that $\boldsymbol{A}$ has the spectral decomposition

$$\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}, \tag{4.107}$$

with $\boldsymbol{V}$ containing the eigenvectors and Λ a diagonal matrix containing the corresponding eigenvalues. The assumption of real eigenvalues will be used to determine the upwind value of the source term, to be specified in equation (4.115). Two important cases for which this assumption holds are: first, $\boldsymbol{U}$ is a scalar matrix and second this condition is also satisfied if $\boldsymbol{\mathcal{E}}$ is symmetric positive definite as elaborated in [56].

To obtain an expression for the numerical flux, we integrate equation (4.103) from $x_{j+1/2}$ to $x \in (x_j, x_{j+1})$, which gives

$$\boldsymbol{\Gamma}(x) - \boldsymbol{\Gamma}(x_{j+1/2}) = \boldsymbol{S}(x). \tag{4.108}$$

Since $\boldsymbol{A}$ is assumed piecewise constant, the flux vector given in equation (4.103) in this interval can be written, using the integrating factor, as

$$\boldsymbol{\Gamma}_{j+1/2} = -\boldsymbol{\mathcal{E}}_{j+1/2} e^{x\boldsymbol{A}_{j+1/2}} \frac{\mathrm{d}}{\mathrm{d}x}\left(e^{-x\boldsymbol{A}_{j+1/2}}\boldsymbol{\varphi}\right). \tag{4.109}$$

Substituting (4.109) into (4.108), isolating the derivative, subsequently integrating from x_j to x_{j+1} and applying the boundary conditions given (4.103) results in the following expression for the flux;

$$\begin{aligned}\boldsymbol{\Gamma}(x_{j+1/2}) = \boldsymbol{\mathcal{E}}_{j+1/2}&\left[\int_{x_j}^{x_{j+1}} e^{-x\boldsymbol{A}_{j+1/2}}\,\mathrm{d}x\right]^{-1}\\ \times&\left[e^{-x_j\boldsymbol{A}_{j+1/2}}\boldsymbol{\varphi}_j - e^{-x_{j+1}\boldsymbol{A}_{j+1/2}}\boldsymbol{\varphi}_{j+1} - \int_{x_j}^{x_{j+1}} e^{-x\boldsymbol{A}_{j+1/2}}\boldsymbol{\mathcal{E}}_{j+1/2}^{-1}\boldsymbol{S}\,\mathrm{d}x\right].\end{aligned} \tag{4.110}$$

If the source term is set to $\boldsymbol{0}$, this results in the expression for the homogeneous flux;

$$\boldsymbol{F}^{\mathrm{h}}_{j+1/2} = \frac{\boldsymbol{\mathcal{E}}}{\Delta x}\Big(B(-\boldsymbol{P}_{j+1/2})\boldsymbol{\varphi}_j - B(\boldsymbol{P}_{j+1/2})\boldsymbol{\varphi}_{j+1}\Big), \tag{4.111}$$

where B is the Bernoulli function defined by;

$$B(z) := \frac{z}{e^z - 1}. \tag{4.112}$$

To derive the approximation for the inhomogeneous flux, the integrals in equation (4.110) are evaluated exactly using a Green's matrix assuming $\boldsymbol{s}$ is constant. For more details see ref. [56]. This results in

$$\boldsymbol{F}^{\mathrm{i}}_{j+1/2} = \Delta x\left(\tfrac{1}{2}\boldsymbol{I} - W(\widetilde{\boldsymbol{P}}_{j+1/2})\right)\hat{\boldsymbol{s}}_{\mathrm{u},j+1/2}, \quad \widetilde{\boldsymbol{P}}_{j+1/2} = \Delta x \boldsymbol{U}_{j+1/2}\boldsymbol{\mathcal{E}}^{-1}_{j+1/2}. \tag{4.113}$$

The function $W(z)$ used here is given by

$$W(z) := \frac{e^z - 1 - z}{z(e^z - 1)}, \tag{4.114}$$

and the upwind value of the source, $\hat{\boldsymbol{s}}_{\mathrm{u},j+1/2}$, is given by the linear combination

$$\begin{aligned}\hat{\boldsymbol{s}}_{\mathrm{u},j+1/2} &= \tfrac{1}{2}(\boldsymbol{I} + \boldsymbol{\sigma}_{j+1/2})\hat{\boldsymbol{s}}_j + \tfrac{1}{2}(\boldsymbol{I} - \boldsymbol{\sigma}_{j+1/2})\hat{\boldsymbol{s}}_{j+1},\\ \boldsymbol{\sigma}_{j+1/2} &= \boldsymbol{\mathcal{E}}_{j+1/2}\mathrm{sgn}(\boldsymbol{A}_{j+1/2})\boldsymbol{\mathcal{E}}^{-1}_{j+1/2}.\end{aligned} \tag{4.115}$$

The matrix function $\operatorname{sgn}(\boldsymbol{A})$ is defined via the spectral decomposition of $\boldsymbol{A}$ as

$$\operatorname{sgn}(\boldsymbol{A}) = \boldsymbol{V}\operatorname{sgn}(\boldsymbol{\Lambda})\boldsymbol{V}^{-1}, \quad \operatorname{sgn}(\boldsymbol{\Lambda}) = \operatorname{diag}(\operatorname{sgn}(\lambda_1), \operatorname{sgn}(\lambda_2), ..., \operatorname{sgn}(\lambda_m)), \tag{4.116}$$

with definition $\operatorname{sgn}(0) = 1$. The complete flux approximation is then given by the superposition

$$\boldsymbol{F}_{j+1/2} = \boldsymbol{F}^{\mathrm{h}}_{j+1/2} + \boldsymbol{F}^{\mathrm{i}}_{j+1/2}. \tag{4.117}$$

Substituting the expressions for the homogeneous (4.111) and inhomogeneous (4.113) flux vectors in the semi-discrete conservation law (4.100), the resulting semi-discrete scheme is given by;

$$\begin{aligned}\Delta x\big(\boldsymbol{\gamma}_{j-1/2}\dot{\boldsymbol{\varphi}}_{j-1} + (\boldsymbol{I} - \boldsymbol{\gamma}_{j+1/2} + \boldsymbol{\delta}_{j-1/2})\dot{\boldsymbol{\varphi}}_j - \boldsymbol{\delta}_{j+1/2}\dot{\boldsymbol{\varphi}}_{j+1}\big)\\ -\boldsymbol{\alpha}_{j-1/2}\boldsymbol{\varphi}_{j-1} + (\boldsymbol{\alpha}_{j+1/2} + \boldsymbol{\beta}_{j-1/2})\boldsymbol{\varphi}_j - \boldsymbol{\beta}_{j+1/2}\boldsymbol{\varphi}_{j+1}\\ = \Delta x\big(\boldsymbol{\gamma}_{j-1/2}\boldsymbol{s}_{j-1} + (\boldsymbol{I} - \boldsymbol{\gamma}_{j+1/2} + \boldsymbol{\delta}_{j-1/2})\boldsymbol{s}_j - \boldsymbol{\delta}_{j+1/2}\boldsymbol{s}_{j+1}\big),\end{aligned} \tag{4.118}$$

with the coefficient matrices defined by

$$\begin{aligned}\boldsymbol{\alpha}_{j+1/2} &= \frac{1}{\Delta x}\boldsymbol{\mathcal{E}}_{j+1/2}B(-\boldsymbol{P}_{j+1/2}),\\ \boldsymbol{\beta}_{j+1/2} &= \frac{1}{\Delta x}\boldsymbol{\mathcal{E}}_{j+1/2}B(\boldsymbol{P}_{j+1/2}),\\ \boldsymbol{\gamma}_{j+1/2} &= \frac{1}{2}\boldsymbol{Q}_{j+1/2}(\boldsymbol{I} + \boldsymbol{\sigma}_{j+1/2}),\\ \boldsymbol{\delta}_{j+1/2} &= \frac{1}{2}\boldsymbol{Q}_{j+1/2}(\boldsymbol{I} - \boldsymbol{\sigma}_{j+1/2}),\\ \boldsymbol{Q}_{j+1/2} &= \frac{1}{2}I - \boldsymbol{\mathcal{E}}_{j+1/2}W(\widetilde{\boldsymbol{P}}_{j+1/2})\boldsymbol{\mathcal{E}}^{-1}_{j+1/2}.\end{aligned} \tag{4.119}$$

4.5 Results and discussion

To illustrate the claims made in Section 4.3 the transformation is applied to two example systems, the first of which is the three-species argon system as introduced earlier, the second a more complex system with 8 species and 10 reactions. To obtain a steady state solution to the semi-discrete system given by (4.118), we first discretize the time derivative with implicit Euler. Next, we consider the limit for $t \to \infty$, equivalent to solving (4.118) with $\dot{\boldsymbol{\varphi}} = 0$. This results in a nonlinear algebraic system of the form

$$\boldsymbol{F}(\boldsymbol{\psi}) = \boldsymbol{b}(\boldsymbol{\psi}) - \boldsymbol{A}(\boldsymbol{\psi})\boldsymbol{\psi} = \boldsymbol{0}, \tag{4.120}$$

where $\boldsymbol{\psi}$ is the vector of unknowns containing $\boldsymbol{\varphi}$ for each grid point. Both the transformed, and untransformed systems yield a nonlinear system of the form (4.120). Such systems can be solved with a variety of methods such as Newton or

Picard iteration. Here we opt for Picard iteration to solve the resulting nonlinear algebraic system of equations. For each Picard iteration a linear system of the form,

$$\boldsymbol{A}(\boldsymbol{\psi}^k)\boldsymbol{\psi}^{k+1} = \boldsymbol{b}(\boldsymbol{\psi}^k), \tag{4.121}$$

has to be solved where $\boldsymbol{\psi}^k$ is the k-th iterate, a vector containing $\boldsymbol{\varphi}$ for each grid point, at iteration number k. Since the system matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ depend on $\boldsymbol{\psi}$, both have to be recomputed for each iteration. We iterate without under-relaxation, and as an initial guess we take a linear interpolation between the imposed boundary conditions. To solve each resulting linear system, we use the `mldivide` functionality provided by MATLAB [39]; for these systems `mldivide` utilizes LAPACK's linear solver for a general band matrix provided by the Intel(R) Math Kernel Library Version 2018.0.3 Product Build 20180406 for Intel(R) 64 architecture applications. Linear Algebra PACKage Version 3.7.0. We continue the Picard iteration until the relative residual satisfies the criterion

$$\frac{\|\boldsymbol{b}(\boldsymbol{\psi^k}) - \boldsymbol{A}(\boldsymbol{\psi}^{k+1})\boldsymbol{\psi}^{k+1}\|_2}{\|\boldsymbol{b}(\boldsymbol{\psi^k})\|_2} < 10^{-7}. \tag{4.122}$$

We also compute the condition number $\kappa(\boldsymbol{A})$, which gives an upper bound on the ratio of the relative error in $\boldsymbol{\psi}$ and the error in the right-hand side $\boldsymbol{b}$. As a rule of thumb, if the condition number $\kappa(\boldsymbol{A}) = 10^p$, one can expect to lose p digits of precision in solving (4.121) [84, p. 321].

It will be shown that, in our numerical experiments, the condition number of the transformed system has significantly decreased, compared to the original system. However, the condition number can be further reduced by rescaling the rows and columns of each linear system. Inspired by the procedure applied in [85] we perform a straightforward row and column scaling to the matrix $\boldsymbol{A}$. First we left-multiply the matrix $\boldsymbol{A}$, with $\boldsymbol{Q} = \mathrm{diag}(q_i)$ with coefficients

$$q_i := \left(\sum_{j=1}^{N} |A_{ij}|\right)^{-1}, \tag{4.123}$$

which scales the system matrix such that the absolute row sums become equal to 1. Subsequently, a similar scaling is applied to the columns of $\boldsymbol{QA}$ with the matrix $\boldsymbol{P} = \mathrm{diag}(p_j)$, with

$$p_j = \left(\sum_{i=1}^{N} \left|(\boldsymbol{QA})_{ij}\right|\right)^{-1}. \tag{4.124}$$

As a final result this gives a scaled coefficient matrix $\boldsymbol{QAP}$, and the resulting, scaled system reads

$$\big(\boldsymbol{QA}(\boldsymbol{\psi^k})\boldsymbol{P}\big)\big(\boldsymbol{P}^{-1}\boldsymbol{\psi}^{k+1}\big) = \boldsymbol{Qb}(\boldsymbol{\psi^k}). \tag{4.125}$$

4.5.1 Three-species argon system

To illustrate the stoichiometric transformation we recall the argon example from Section 4.3.1, and apply the transformation $\widetilde{\boldsymbol{T}}$ from equation (4.79) such that two out of three transformed variables are constant.

This numerical experiment is governed by the mass balances as given by equation (4.5). We compute a solution over the domain extending from $x = 0$ to $x = L$, with the length of the domain given by $L = 1 \times 10^{-4}$ m. Here we take a constant ion and electron temperature, $T_\mathrm{i} = 300$ K and $T_\mathrm{e} = 7500$ K, respectively. The pressure has been fixed at 10^3 Pa, and there is a constant advection of -0.1 $\mathrm{kgm^{-2}s^{-1}}$. On both boundaries Dirichlet data have been imposed, in such a way that quasi-neutrality and $\sum_i y_i = 1$ holds on both boundaries. First the values for y_Ar and y_{Ar^+} are set, then the electron mass fraction is computed from the quasi-neutrality requirement, finally each mass fraction is divided by $\sum_i y_i$, such that the resulting boundary conditions satisfy $\sum_i y_i = 1$ and quasi-neutrality. The starting values for y_Ar and y_{Ar^+} are given by

$$\boldsymbol{y} = \begin{bmatrix} y_{\mathrm{Ar}^+} \\ y_\mathrm{Ar} \end{bmatrix}, \quad \boldsymbol{y}_\mathrm{L} := \boldsymbol{y}(x=0) = \begin{bmatrix} 0.04 \\ 0.96 \end{bmatrix}, \quad \boldsymbol{y}_\mathrm{R} := \boldsymbol{y}(x=L) = \begin{bmatrix} 0.96 \\ 0.04 \end{bmatrix}. \tag{4.126}$$

Two reactions are included, an ionization and a recombination reaction;

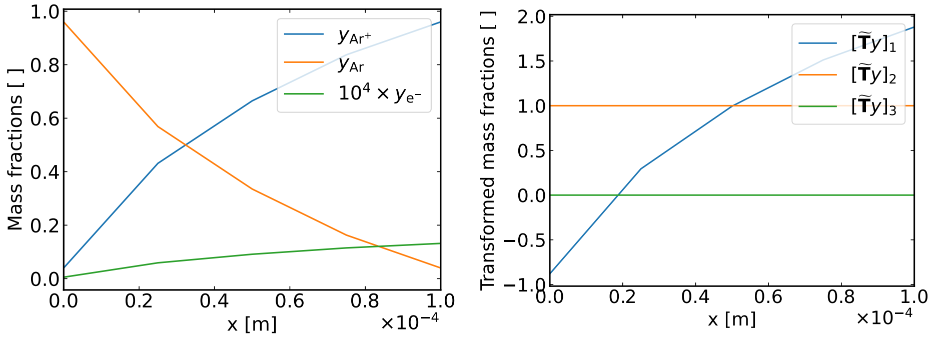
$$\mathrm{Ar} + \mathrm{e}^- \xrightarrow{k_\mathrm{i}} \mathrm{Ar}^+ + 2\mathrm{e}^-, \tag{4.127a}$$

$$\mathrm{Ar}^+ + 2\mathrm{e}^- \xrightarrow{k_\mathrm{r}} \mathrm{Ar} + \mathrm{e}^-, \tag{4.127b}$$

where k_i and k_r are the reaction rates given by an Arrhenius-type relation, detailed in Section 4.B.

The effect of the transformation can be seen in Figure 4.2, before transformation none of the three variables are constant, after transformation it can be observed that $(\widetilde{\boldsymbol{T}}\boldsymbol{y})_2 = 1$ and $(\widetilde{\boldsymbol{T}}\boldsymbol{y})_3 = 0$ as expected from the analysis in Section 4.3.1. These two variables represent $\sum_i y_i = 1$ and quasi-neutrality respectively. It will be shown shortly to which degree these variables are constant.

Since two out of three transformed variables are known, these do not need to be solved for, resulting in a smaller system. As a result the linear system that has to be solved in each Picard iteration has N instead of $3N$ degrees of freedom. The cost of computing the transformation itself is modest, similar to the cost of computing the coefficient matrices required by the discretization scheme.

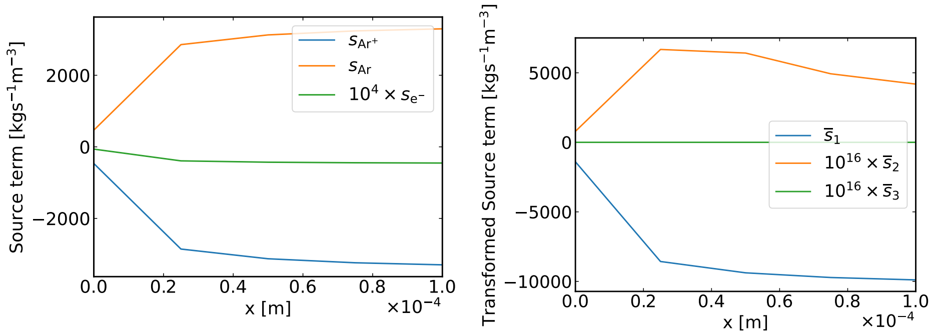


(a) Original system

(b) Transformed system

Figure 4.2: Computed mass fractions for the argon system, using 5 grid points.

A second effect of the transformation is that the two source terms corresponding to quasi-elements vanish, which can be seen in Figure 4.3. After the transformation the second and third source term have been reduced to effectively zero. This is expected based on the meaning of the variables $[\widetilde{\boldsymbol{Ty}}]_2 = 1$ and $[\widetilde{\boldsymbol{Ty}}]_3 = 0$, since chemical reactions cannot produce mass, nor (net) electric charge.

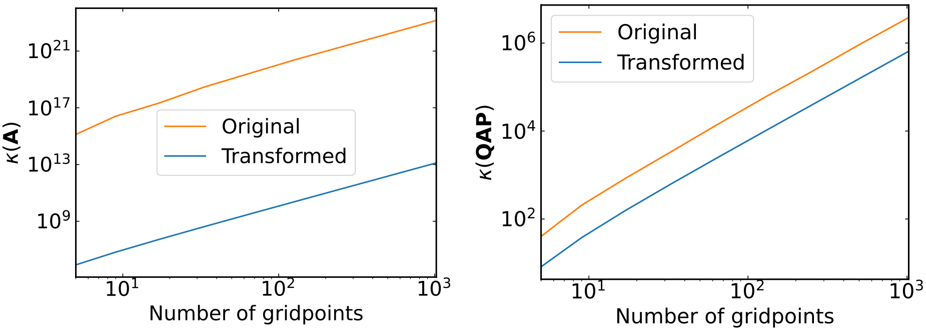


(a) Original system

(b) Transformed system

Figure 4.3: Computed source terms for the argon system.

Next, we report the worst condition number of the system matrix encountered in Picard iteration, as function of the number of grid points, see Figure 4.4. Applying the rescaling shows a strong reduction in condition number, see Figure 4.4. After scaling the transformed system also shows a significant reduction in condition number compared to the original system. The slope of both the original and transformed systems are identical, approximately 2, implying that the condition number increases proportional to N^2. Given the reduction in condition number, the scaled variant (4.125) is used in the Picard iteration for all numerical experiments shown here.



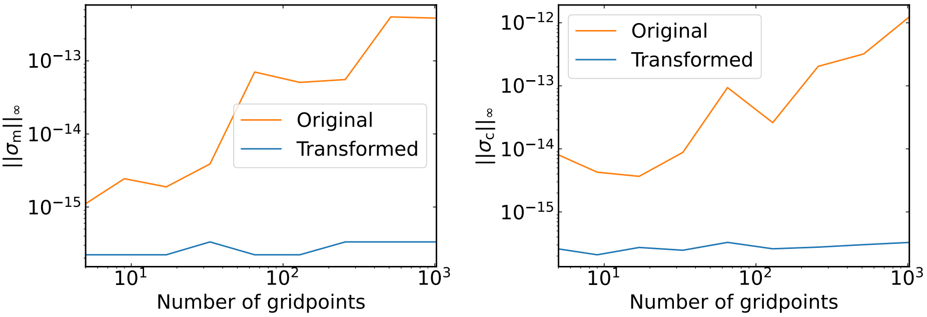
(a) Before scaling (b) After scaling

Figure 4.4: Effect of scaling the transformed and original system matrices.

Finally, a comparison is made between the transformed and original systems for how well conservation of mass, $\mathbf{1}^{\mathrm{T}}\boldsymbol{y} = 1$ and quasi-neutrality $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n} = 0$, are satisfied. To quantify these invariants we introduce

$$\sigma_{\mathrm{m}} := |1 - \mathbf{1}^{\mathrm{T}}\boldsymbol{y}|, \quad \sigma_{\mathrm{c}} := \left|\frac{\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n}}{e\mathbf{1}^{\mathrm{T}}\boldsymbol{n}}\right|, \tag{4.128}$$

where e is the elementary charge. For every grid cell in the domain we compute σ_{m} and σ_{c}, then we take the infinity norm to determine the maximum deviation. This is reported in Figures 4.5a and 4.5b, respectively. It can be seen that the transformed system is able to represent both quantities more accurately. However, not exactly since there is an apparent error in transforming back from $\widetilde{\boldsymbol{T}}\boldsymbol{y}$ to $\boldsymbol{y}$ of the order of the rounding errors.



(a) Conservation of mass (b) Quasi-neutrality

Figure 4.5: Effect on conservation of mass and quasi-neutrality, here the largest deviation for each simulation is plotted for the argon system.

4.5.2 Aluminum-argon-hydrogen system

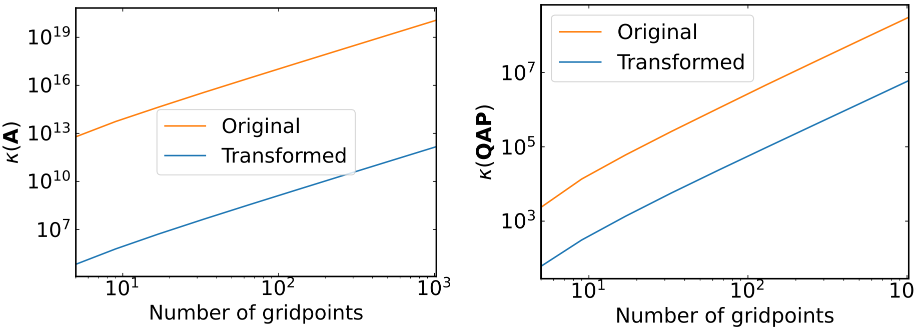
Next, the transformation is applied to a larger system involving 8 species and 2 ions. This system involves the species Ar^+, Ar^*, Al^+, H, e^-, Ar, Al and H_2. Here the last four of these can be used as quasi-elements to construct all other species. Applying the construction as proposed in Section 4.3 yields the non-integer transformation matrix $\boldsymbol{M}$, given by

$$\boldsymbol{M} = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -\frac{1}{2} \\ \hline -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 1 \end{array}\right]. \tag{4.129}$$

The final transformation matrix $\widetilde{\boldsymbol{T}}$ is constructed such that $[\widetilde{\boldsymbol{T}}\boldsymbol{y}]_5 = 0$ represents $\boldsymbol{q}^{\mathrm{T}}\boldsymbol{n} = 0$ and $[\widetilde{\boldsymbol{T}}\boldsymbol{y}]_6 = 1$ represents $\boldsymbol{y}^{\mathrm{T}}\boldsymbol{1} = 1$. We solve this system over a domain from $x = 0$ to $x = L$, with $L = 10^{-3}$ m. Similar to the argon experiment, we set the mass fraction values for all species, except y_{e^-}, then the electron mass fraction is computed from the quasi-neutrality requirement, finally each mass fraction in the resulting array is divided by $\sum_i y_i$ such that again both $\sum_i y_i = 1$ and quasi-neutrality are satisfied on the boundaries. The starting values for this system are given by

$$\boldsymbol{y} = \begin{bmatrix} y_{\mathrm{Ar}^+} \\ y_{\mathrm{Ar}^*} \\ y_{\mathrm{Al}^+} \\ y_{\mathrm{H}} \\ y_{\mathrm{Ar}} \\ y_{\mathrm{Al}} \\ y_{\mathrm{H}_2} \end{bmatrix}, \boldsymbol{y}_{\mathrm{L}} := \boldsymbol{y}(x=0) = \begin{bmatrix} 0.16 \\ 0.11 \\ 0.13 \\ 0.13 \\ 0.19 \\ 0.22 \\ 0.046 \end{bmatrix}, \boldsymbol{y}_{\mathrm{R}} := \boldsymbol{y}(x=L) = \begin{bmatrix} 0.0081 \\ 0.11 \\ 0.094 \\ 0.26 \\ 0.22 \\ 0.25 \\ 0.054 \end{bmatrix}. \tag{4.130}$$

Similar to the argon system, it can be seen in Figures 4.6 that scaling the system matrix has a strong impact on the condition number. However, for both the scaled and unscaled systems the transformed system is significantly better conditioned than the original. As is the case with the argon system of Section 4.5.1, the system matrix for the transformed system is slightly smaller. The transformed system has $6N$ degrees of freedom, whereas the original system has $8N$.

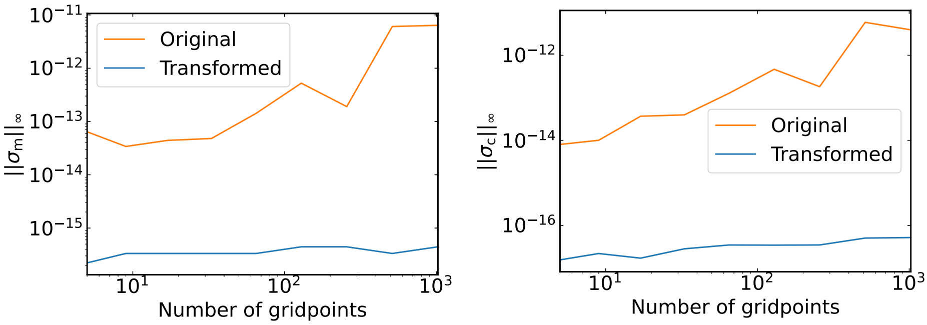


(a) Before scaling

(b) After scaling

Figure 4.6: Effect of transforming and scaling on the condition number.

It can also be seen in Figures 4.7a and 4.7b that both conservation of mass, and quasi-neutrality are improved for the transformed system compared to the original. The only source of error in these quantities that remains is rounding errors.



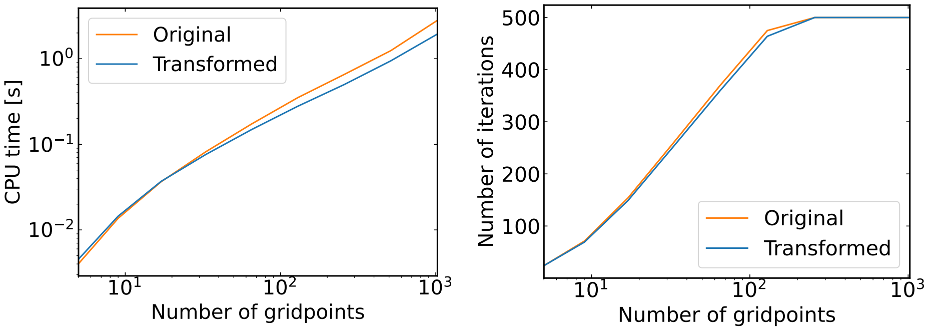
(a) Conservation of mass

(b) Quasi-neutrality

Figure 4.7: Effect on the conservation of mass and quasi-neutrality, here the largest deviation for each simulation is plotted for the 8 species system.

The total time required to solve all linear systems involving the system matrix is given in Figure 4.8a. Here it can be seen that the transformed system does provide a speedup over the original. For more than 20 gridpoints, the slopes are close, indicating that both the original and transformed system have the same asymptotic complexity.

The total number of iterations required before the system converged is slightly, but not significantly affected by the transformation. This can be seen in Figure 4.8b. The systems with $N \geq 256$ reached the maximum allowed number of 500 iterations.



(a) Time to solve all linear systems.

(b) Number of Picard iterations.

Figure 4.8: Effect of the transformation on the total CPU time for the Picard iteration.

To summarize, for the argon, and the argon-aluminum-hydrogen systems, applying the stoichiometric transformation we can eliminate part of the source term, reduce the condition number of both the scaled and unscaled system matrix, and improve on both conservation of mass and quasi-neutrality. Using conservation of mass and quasi-neutrality, allows for a reduction of the linear system involved. Even though this does not reduce the asymptotic complexity, numerical experiments show a speedup for the transformed system.

4.6 Conclusions and outlook

Starting from the mass balances and the Stefan-Maxwell relations [16, 55], we have proposed an alternative choice of regularization parameter in the derivation of the diffusion matrix. We have shown that an alternative derivation exists to link the gradients of mass and mole fractions needed for computing the diffusion velocities.

By combining ideas from porous media [62, 61], and the idea of taking linear combinations of species to derive a stoichiometric transformation similar, but not identical to the one proposed by Rini [59], an algorithm is introduced to identify a basis set of quasi-elements, given a mixture where only certain reactions are allowed. For an argon example the stoichiometric transformation has been derived explicitly, which is then generalized to an arbitrary chemistry set. By introducing a matrix $\widetilde{\boldsymbol{T}}$ it is possible to obtain a transformation such that two of the transformed variables are invariants of the system. We are able to show in numerical experiments that applying this transformation results in exact enforcement of the invariants quasi-neutrality and mass conservation.

The number of iterations required to achieve the tolerance set is only mildly affected by the transformation. The time to solve all linear systems in the Picard iterations combined is reduced. This is caused by two effects. First, the decrease in the number of iterations and second, each linear system for the transformed system has $2N$ fewer unknowns. An additional effect is that the transformation is able to significantly reduce the condition number, both before and after scaling. This is expected to first of all reduce the error, but may also allow for faster convergence of Krylov-subspace iterative methods. For example, the rate at which the well-known Conjugate Gradient method decreases the error is proportional to $1/\sqrt{\kappa}$ [32, p. 215][32, p260-p261].

The main objective of our work is to illustrate the stoichiometric transformation for plasma-chemical systems. Based on the results presented in Section 4.5, this method is able to impose the constraints quasi-neutrality and conservation of mass up to the machine precision. The slight reduction in CPU time, combined with the decreased condition number indicate that this method may be more advantageous in 2D and 3D simulations, for which the cost of solving linear systems is more significant.

A possible improvement to reduce the number of iterations required to solve (4.120) would be to use Newton iteration instead of Picard iteration. An additional modification, would be a strategy similar to Gummel iteration [86], where first the source-free part would be solved, providing an improved guess for the part that does not have a zero source. To improve convergence, it would be an idea to incorporate the method of false transients to obtain the steady state solution, possibly in conjunction with Gummel iteration. Since the stoichiometric transformation guarantees that chemical reactions do not produce quasi-elements, it may also be an option to combine the stoichiometric transformation with chemical reduction techniques for complex chemistries, for example, by combining the timescale analysis of CSP [87] with the stoichiometric transformation and approximating a subset of the transformed variables as constant.

Appendix

4.A On the positive definiteness of the ambipolar diffusion matrix

The ambipolar diffusion matrix is defined in equation (4.32) as the real matrix;

$$\boldsymbol{D}_{\text{amb}} := \widetilde{\boldsymbol{D}} - \frac{\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_{\text{c}}(\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_{\text{c}})^{\text{T}}}{\boldsymbol{\rho}_{\text{c}}^{\text{T}}\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_{\text{c}}}, \tag{4.131}$$

where $\widetilde{\boldsymbol{D}} = \widetilde{\boldsymbol{F}}^{-1}$ is Symmetric Positive Definite (SPD) [16]. The regularized, ambipolar diffusion matrix

$$\widetilde{\boldsymbol{D}}_{\text{amb}} := \boldsymbol{D}_{\text{amb}} + \beta\boldsymbol{q}\boldsymbol{q}^{\text{T}},\ \beta > 0,\ \boldsymbol{q} \neq \boldsymbol{0}, \tag{4.132}$$

is symmetric, as can be seen from (4.131). Giovangigli derived an ambipolar diffusion matrix in [63], which is then shown to be symmetric positive semidefinite using a generalized inner product. Here, we use a similar approach to prove that the regularized ambipolar diffusion matrix (4.132) is regular.

Our goal is to show positive definiteness of $\widetilde{\boldsymbol{D}}_{\text{amb}}$ from the definition;

$$\forall\, \boldsymbol{a} \neq \boldsymbol{0} :\ \boldsymbol{a}^{\text{T}}\widetilde{\boldsymbol{D}}_{\text{amb}}\boldsymbol{a} > 0. \tag{4.133}$$

Note that since $\widetilde{\boldsymbol{D}}$ is SPD, one can define an inner product for two real vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, as

$$\langle \boldsymbol{a}, \boldsymbol{b}\rangle_{\widetilde{\boldsymbol{D}}} := \boldsymbol{a}^{\text{T}}\widetilde{\boldsymbol{D}}\boldsymbol{b}, \tag{4.134}$$

since it adheres to the properties that define an inner product:

1. Symmetry:
$$\langle \boldsymbol{a}, \boldsymbol{b}\rangle_{\widetilde{\boldsymbol{D}}} = \langle \boldsymbol{b}, \boldsymbol{a}\rangle_{\widetilde{\boldsymbol{D}}}. \tag{4.135a}$$
2. Linearity in the first argument, for real scalars α, γ and a vector $\boldsymbol{d}$:
$$\langle \alpha\boldsymbol{a} + \gamma\boldsymbol{d}, \boldsymbol{b}\rangle_{\widetilde{\boldsymbol{D}}} = \alpha\langle \boldsymbol{a}, \boldsymbol{b}\rangle_{\widetilde{\boldsymbol{D}}} + \gamma\langle \boldsymbol{d}, \boldsymbol{b}\rangle_{\widetilde{\boldsymbol{D}}}. \tag{4.135b}$$
3. Positive-definiteness:
$$\forall\, \boldsymbol{a} \neq \boldsymbol{0} :\; \boldsymbol{a}^{\mathrm{T}}\widetilde{\boldsymbol{D}}\boldsymbol{a} > 0. \tag{4.135c}$$

Second, is the inner product can be used to define a norm

$$\boldsymbol{a}^{\mathrm{T}}\widetilde{\boldsymbol{D}}\boldsymbol{a} =: \|\boldsymbol{a}\|^2_{\widetilde{\boldsymbol{D}}}, \tag{4.136}$$

moreover, for inner products the Cauchy-Schwarz inequality holds

$$\langle \boldsymbol{a}, \boldsymbol{b}\rangle_{\widetilde{\boldsymbol{D}}} \leq \|\boldsymbol{a}\|_{\widetilde{\boldsymbol{D}}}\|\boldsymbol{b}\|_{\widetilde{\boldsymbol{D}}}. \tag{4.137}$$

Left-multiplying $\widetilde{\boldsymbol{D}}_{\mathrm{amb}}$ with $\boldsymbol{a}^{\mathrm{T}}$ and right-multiplying with $\boldsymbol{a}$ results in the quadratic form

$$\boldsymbol{a}^{\mathrm{T}}\widetilde{\boldsymbol{D}}_{\mathrm{amb}}\boldsymbol{a} = \boldsymbol{a}^{\mathrm{T}}\widetilde{\boldsymbol{D}}\boldsymbol{a} - \frac{\boldsymbol{a}^{\mathrm{T}}\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_{\mathrm{c}}\boldsymbol{\rho}_{\mathrm{c}}^{\mathrm{T}}\widetilde{\boldsymbol{D}}\boldsymbol{a}}{\boldsymbol{\rho}_{\mathrm{c}}^{\mathrm{T}}\widetilde{\boldsymbol{D}}\boldsymbol{\rho}_{\mathrm{c}}} + \beta\boldsymbol{a}^{\mathrm{T}}\boldsymbol{q}\boldsymbol{q}^{\mathrm{T}}\boldsymbol{a}. \tag{4.138}$$

Then, using the Cauchy-Schwarz inequality on the second term it can be concluded that

$$\boldsymbol{a}^{\mathrm{T}}\widetilde{\boldsymbol{D}}_{\mathrm{amb}}\boldsymbol{a} = \underbrace{\|\boldsymbol{a}\|^2_{\widetilde{\boldsymbol{D}}}}_{>0} - \underbrace{\frac{\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}}\rangle^2_{\widetilde{\boldsymbol{D}}}}{\|\boldsymbol{\rho}_{\mathrm{c}}\|^2_{\widetilde{\boldsymbol{D}}}}}_{\leq\|\boldsymbol{a}\|^2_{\widetilde{\boldsymbol{D}}}} + \underbrace{\beta\|\boldsymbol{a}^{\mathrm{T}}\boldsymbol{q}\|^2_2}_{\geq 0} \geq 0, \tag{4.139}$$

where $\|\cdot\|_2$ indicates the 2-norm. Therefore $\boldsymbol{a}^{\mathrm{T}}\widetilde{\boldsymbol{D}}_{\mathrm{amb}}\boldsymbol{a} > 0$ for all $\boldsymbol{a} \neq \boldsymbol{0}$, except if

$$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{q} = 0 \quad \wedge \quad \langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}}\rangle^2_{\widetilde{\boldsymbol{D}}} = \|\boldsymbol{a}\|^2_{\widetilde{\boldsymbol{D}}}\|\boldsymbol{\rho}_{\mathrm{c}}\|^2_{\widetilde{\boldsymbol{D}}}, \tag{4.140}$$

i.e., $\boldsymbol{a}$ is orthogonal to $\boldsymbol{q}$, and $\boldsymbol{a}$ and $\boldsymbol{\rho}_{\mathrm{c}}$ are linearly dependent. In any case $\widetilde{\boldsymbol{D}}_{\mathrm{amb}}$ is symmetric, *semi*-positive definite. To complete the proof that $\widetilde{\boldsymbol{D}}_{\mathrm{amb}}$ is SPD it must be shown that the criterion in (4.140) never holds. To do this we first show under which conditions $\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}}\rangle^2_{\widetilde{\boldsymbol{D}}} = \|\boldsymbol{a}\|^2_{\widetilde{\boldsymbol{D}}}\|\boldsymbol{\rho}_{\mathrm{c}}\|^2_{\widetilde{\boldsymbol{D}}}$, then show that this is incompatible with $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{q} = 0$.

To investigate which $\boldsymbol{a}$ leads to

$$\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}}\rangle^2_{\widetilde{\boldsymbol{D}}} = \|\boldsymbol{a}\|^2_{\widetilde{\boldsymbol{D}}}\|\boldsymbol{\rho}_{\mathrm{c}}\|^2_{\widetilde{\boldsymbol{D}}}, \tag{4.141}$$

we turn to the inner-product notation again;

$$\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}^2 = \langle \boldsymbol{a}, \boldsymbol{a} \rangle_{\widetilde{\boldsymbol{D}}} \langle \boldsymbol{\rho}_{\mathrm{c}}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}, \tag{4.142}$$

and define a scalar λ given by

$$\lambda := \frac{\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}}{\langle \boldsymbol{\rho}_{\mathrm{c}}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}}. \tag{4.143}$$

Then equation (4.142) becomes after some intermediate steps

$$\begin{aligned} 0 &= \langle \boldsymbol{a}, \boldsymbol{a} \rangle_{\widetilde{\boldsymbol{D}}} - \frac{\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}^2}{\langle \boldsymbol{\rho}_{\mathrm{c}}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}} \\ &= \langle \boldsymbol{a}, \boldsymbol{a} \rangle_{\widetilde{\boldsymbol{D}}} - 2\frac{\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}^2}{\langle \boldsymbol{\rho}_{\mathrm{c}}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}} + \frac{\langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}^2}{\langle \boldsymbol{\rho}_{\mathrm{c}}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}} \quad . \\ &= \langle \boldsymbol{a}, \boldsymbol{a} \rangle_{\widetilde{\boldsymbol{D}}} - 2\lambda \langle \boldsymbol{a}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}} + \lambda^2 \langle \boldsymbol{\rho}_{\mathrm{c}}, \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}}, \\ &= \langle \boldsymbol{a} - \lambda \boldsymbol{\rho}_{\mathrm{c}}, \boldsymbol{a} - \lambda \boldsymbol{\rho}_{\mathrm{c}} \rangle_{\widetilde{\boldsymbol{D}}} \\ &= \| \boldsymbol{a} - \lambda \boldsymbol{\rho}_{\mathrm{c}} \|_{\widetilde{\boldsymbol{D}}}^2 \end{aligned} \tag{4.144}$$

Therefore, the equality in equation (4.141) is satisfied if $\boldsymbol{a} = \lambda \boldsymbol{\rho}_{\mathrm{c}}$. Next, we show that this constraint yields that $\boldsymbol{a}^{\mathrm{T}} \boldsymbol{q} \neq 0$. Using the definition of $\boldsymbol{\rho}_{\mathrm{c}}$,

$$\rho_{c,i} := n_i q_i, \tag{4.145}$$

and substituting $\boldsymbol{a} = \lambda \boldsymbol{\rho}_{\mathrm{c}}$ we get

$$\lambda \boldsymbol{\rho}_{\mathrm{c}}^{\mathrm{T}} \boldsymbol{q} = \textstyle\sum_i q_i^2 n_i \neq 0, \tag{4.146}$$

as long as there is any charged species present with a nonzero number density, this condition is equivalent to $\boldsymbol{\rho}_{\mathrm{c}} \neq \boldsymbol{0}$. In conclusion, (4.140) never holds. Therefore, $\boldsymbol{a}^{\mathrm{T}} \widetilde{\boldsymbol{D}}_{\mathrm{amb}} \boldsymbol{a} > 0$ and thus $\widetilde{\boldsymbol{D}}_{\mathrm{amb}}$ is SPD. Note that this also implies that the matrix $\widetilde{\boldsymbol{D}}_{\mathrm{amb}}$ is regular for $\beta > 0$.

4.B Reaction list for the 8-species system

The reaction rates are given by an Arrhenius-type relation,

$$k = c\Big(\tfrac{T}{T_{\mathrm{ref}}}\Big)^q \exp(-E_{\mathrm{a}}/(k_{\mathrm{B}} T)), \tag{4.147}$$

where the coefficients are given by Table 4.B.1 and k_{B} is the Boltzmann-constant. For all reactions, except the dissociation of hydrogen, the temperature in (4.147) is the electron temperature. Table 4.B.1 shows a list of the Arrhenius coefficients for the 8-species system, for the reactions in this table the electron temperature is used. Reaction 11 only relates hydrogen and uses the ion temperature instead of electron temperature.

Table 4.B.1: Table of coefficients in the Arrhenius relation (4.147) for reaction rates of the 8-species system. Here X is the number of reactants present in the reaction.

Reaction	Equation	c (m^{3X}s^{-1})	q	E_a (eV)	Source
Argon ionization	$\mathrm{Ar} + \mathrm{e}^- \xrightarrow{k_1} \mathrm{Ar}^+ + 2\mathrm{e}^-$	2.3×10^{-14}	0.68	15.76	[88]
Argon recombination	$\mathrm{Ar}^+ + 2\mathrm{e}^- \xrightarrow{k_2} \mathrm{Ar} + \mathrm{e}^-$	8.75×10^{-39}	−4.5	0	[89]
Argon excitation	$\mathrm{Ar} + \mathrm{e}^- \xrightarrow{k_3} \mathrm{Ar}^* + \mathrm{e}^-$	5.0×10^{-15}	0.74	11.56	[90]
Ionization of excited argon	$\mathrm{Ar}^* + \mathrm{e}^- \xrightarrow{k_4} \mathrm{Ar}^+ + 2\mathrm{e}^-$	6.8×10^{-15}	0.67	4.20	[91]
De-excitation of argon	$\mathrm{Ar}^* + \mathrm{e}^- \xrightarrow{k_5} \mathrm{Ar} + \mathrm{e}^-$	4.3×10^{-16}	0.74	0	[88]
Radiative de-excitation of argon	$\mathrm{Ar}^* \xrightarrow{k_6} \mathrm{Ar} + \hbar\nu$	3.15×10^{8}	0	0	[92]
Aluminum ionization	$\mathrm{Al} + \mathrm{e}^- \xrightarrow{k_7} \mathrm{Al}^+ + 2\mathrm{e}^-$	1.23×10^{-13}	0	7.23	[93]
Penning ionization	$\mathrm{Ar}^* + \mathrm{Al} \xrightarrow{k_8} \mathrm{Al}^+ + \mathrm{e}^- + \mathrm{Ar}$	5.9×10^{-16}	0	0	[94]
Charge exchange	$\mathrm{Ar}^+ + \mathrm{Al} \xrightarrow{k_9} \mathrm{Al}^+ + \mathrm{Ar}$	1.0×10^{-15}	0	0	[94]
Dissociation of hydrogen	$\mathrm{H}_2 \xrightarrow{k_{10}} 2\mathrm{H}$	5.610×10^{-15}	0.419	4.478	[95]

4.C Obtaining a basis for a 9-species system

Starting from the reaction set given in Section 4.B.1, where one spectator particle Au and an aluminum recombination reaction are added, we first construct the net reactions, Table 4.C.1. We want to find some set of quasi-element species $\mathcal{E}$, such that all species in $\mathcal{U}$ can be constructed either directly, or indirectly via the reactions given in Table 4.C.1. Applying Algorithm 2 results in a basis set $\mathcal{E}$, for example $\{\mathrm{Ar}, \mathrm{e}^-, \mathrm{Al}, \mathrm{H}, \mathrm{Au}\}$, $\{\mathrm{Au}, \mathrm{H}_2, \mathrm{Ar}^+, \mathrm{Ar}, \mathrm{Al}^+\}$ or $\{\mathrm{Au}, \mathrm{H}_2, \mathrm{Ar}^*, \mathrm{e}^-, \mathrm{Al}^+\}$ would suffice. However, $\{\mathrm{Au}, \mathrm{H}_2, \mathrm{Ar}^*, \mathrm{Ar}, \mathrm{Al}^+\}$ would not, since reactions 3 and 5 in Table 4.C.1 would allow for conversion between two quasi-elements in a chemical reaction, which contradicts the definition of a quasi-element. First we define the set $\mathcal{U}$ containing all species in the mixture;

$$\mathcal{U} = \{\mathrm{Ar}, \mathrm{Ar}^*, \mathrm{Ar}^+, \mathrm{Al}, \mathrm{Al}^+, \mathrm{H}, \mathrm{H}_2, \mathrm{e}^-, \mathrm{Au}\}. \tag{4.148}$$

Next, we write the reactions as a collection of unordered sets $\mathcal{S}$;

$$\begin{aligned}\mathcal{S} = \{&\{\mathrm{Ar}, \mathrm{Ar}^+, \mathrm{e}^-\}, \{\mathrm{Ar}^+, \mathrm{e}^-, \mathrm{Ar}\}, \{\mathrm{Ar}, \mathrm{Ar}^*\}, \{\mathrm{Ar}^*, \mathrm{Ar}^+, \mathrm{e}^-\},\\ &\{\mathrm{Ar}^*, \mathrm{Ar}\}, \{\mathrm{Ar}^*, \mathrm{Ar}\}, \{\mathrm{Al}, \mathrm{Al}^+, \mathrm{e}^-\}, \{\mathrm{Ar}^*, \mathrm{Al}, \mathrm{Al}^+, \mathrm{Ar}, \mathrm{e}^-\},.\\ &\{\mathrm{Ar}^+, \mathrm{Al}, \mathrm{Al}^+, \mathrm{Ar}\}, \{\mathrm{H}_2, \mathrm{H}\}, \{\mathrm{Au}\}\}\end{aligned} \tag{4.149}$$

Finally, we apply Algorithm 2.

Table 4.C.1: Net reactions for the 8-particle system with Au as a spectator particle.

Reaction	Equation
Argon ionization	$\mathrm{Ar} \xrightarrow{k_1} \mathrm{Ar}^+ + \mathrm{e}^-$
Argon recombination	$\mathrm{Ar}^+ + \mathrm{e}^- \xrightarrow{k_2} \mathrm{Ar}$
Argon excitation	$\mathrm{Ar} \xrightarrow{k_3} \mathrm{Ar}^*$
Ionization of excited argon	$\mathrm{Ar}^* \xrightarrow{k_4} \mathrm{Ar}^+ + \mathrm{e}^-$
De-excitation of argon	$\mathrm{Ar}^* \xrightarrow{k_5} \mathrm{Ar}$
Radiative de-excitation of argon	$\mathrm{Ar}^* \xrightarrow{k_6} \mathrm{Ar}$
Aluminum ionization	$\mathrm{Al} \xrightarrow{k_7} \mathrm{Al}^+ + \mathrm{e}^-$
Penning ionization	$\mathrm{Ar}^* + \mathrm{Al} \xrightarrow{k_8} \mathrm{Al}^+ + \mathrm{e}^- + \mathrm{Ar}$
Charge exchange	$\mathrm{Ar}^+ + \mathrm{Al} \xrightarrow{k_9} \mathrm{Al}^+ + \mathrm{Ar}$
Dissociation of hydrogen	$\mathrm{H}_2 \xrightarrow{k_{10}} 2\mathrm{H}$
Spectator species	Au

Chapter 5

Near-equilibrium description of plasmas based on the stoichiometric transformation

In this chapter we discuss a method to solve the Advection-Diffusion-Reaction (ADR) equation for plasmas that are close to local chemical equilibrium. This is achieved by computing the chemical equilibrium solution, and solving the ADR equation for the deviation from chemical equilibrium.

If the Damköhler number is large then the plasma rapidly tends toward the chemical equilibrium solution, for example if there is no transport in the plasma, or the chemical timescales are much shorter than the transport timescales. The near-equilibrium description presented in this chapter can also be interpreted as computing the deviation from a no-transport solution. Note that this method is broader than changing the initial guess, since the equilibrium solution can also be changed during the simulation. For example, in systems where the temperature varies over time, the chemical equilibrium solution would also vary over time.

Another potential advantage of the near-equilibrium description is that the chemical equilibrium solution may be strongly dependent on position. For example if there is a strong temperature gradient in the plasma, the deviation from chemical equilibrium may still be relatively slowly varying in space if transport terms are small.

Our near-equilibrium description method is based on the Stoichiometric Transformation Method (STM) described in [96]. In this paper, a linear transformation is applied to the governing set of ADR equations. The method is based on the chemical reactions present in the plasma. The result is that invariants such as mass and charge can be enforced up to machine precision. Additionally, part of the chemical source term in the resulting set of PDEs is eliminated.

Here, we take the STM as a starting point, and add an additional step to the method. The idea is to approximate the non-quasi-element part based on the chemical equilibrium solution, and compute deviations from this chemical equilibrium.

The idea of the method presented here is to approximate the non-quasi-element part based on the chemical equilibrium solution, with a perturbation. Application to a large test problem is left as future research. In the next section we present the governing equations in the near-equilibrium description.

5.1 Governing equations

The particle balance for a multi-species mixture in one dimension is given by

$$\frac{\partial}{\partial t}\boldsymbol{\varphi} + \frac{\partial}{\partial x}\left(\boldsymbol{U}\boldsymbol{\varphi} - \boldsymbol{\mathcal{E}}\frac{\partial}{\partial x}\boldsymbol{\varphi}\right) = \boldsymbol{s}(\boldsymbol{\varphi}). \tag{5.1}$$

Like in Chapter 4,[96], we again apply the stoichiometric transformation to obtain a new set of equations

$$\frac{\partial}{\partial t}\widetilde{\boldsymbol{\varphi}} + \frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{\varphi}} - \widetilde{\boldsymbol{\mathcal{E}}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\varphi}}\right) = \widetilde{\boldsymbol{s}}(\widetilde{\boldsymbol{\varphi}}), \tag{5.2}$$

where the transformation for a vector $\boldsymbol{v}$ is defined as $\widetilde{\boldsymbol{v}} := \boldsymbol{M}\boldsymbol{v}$ and for a matrix $\boldsymbol{A}$ as $\widetilde{\boldsymbol{A}} := \boldsymbol{M}\boldsymbol{A}\boldsymbol{M}^{-1}$. To simplify further analysis we assume a steady state solution. Here $\boldsymbol{M}$ is both regular and constant.

Unlike the STM as described in Chapter 4,[96], here the transformation matrix is computed using Gaussian elimination. Given the stoichiometric matrix $\boldsymbol{\nu}$, the transformation matrix $\boldsymbol{M}$ is obtained by

$$\mathrm{rref}(\boldsymbol{\nu}|\boldsymbol{I}) = [\boldsymbol{\nu}_{\mathrm{rref}}|\boldsymbol{M}], \tag{5.3}$$

where rref computes the row reduced echelon form, | is a concatenation operator and $\boldsymbol{I}$ is the identity matrix with rank equal to the number of rows in the matrix $\boldsymbol{\nu}$.

To describe the plasma in terms of a chemical equilibrium and a deviation, we split the variable $\widetilde{\boldsymbol{\varphi}}$ in two parts, one part containing the quasi-elements, $\widetilde{\boldsymbol{\varphi}}_{\mathrm{q}}$, and another part containing the non-quasi-elements, $\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}$. Similarly, the matrices $\widetilde{\boldsymbol{U}}$ and $\widetilde{\boldsymbol{\mathcal{E}}}$ are split into four blocks;

$$\widetilde{\boldsymbol{\varphi}} = \begin{bmatrix} \widetilde{\boldsymbol{\varphi}}_{\mathrm{n}} \\ \widetilde{\boldsymbol{\varphi}}_{\mathrm{q}} \end{bmatrix}, \quad \widetilde{\boldsymbol{\mathcal{E}}} = \begin{bmatrix} \widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nn}} & \widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nq}} \\ \widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qn}} & \widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qq}} \end{bmatrix}. \tag{5.4}$$

Substituting these split variables in equation (5.2) results in a block-system,

$$\frac{\partial}{\partial x}\left(\begin{bmatrix}\widetilde{\boldsymbol{U}}_{\mathrm{nn}} & \boldsymbol{0}\\ \boldsymbol{0} & \widetilde{\boldsymbol{U}}_{\mathrm{qq}}\end{bmatrix}\begin{bmatrix}\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}\\ \widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}\end{bmatrix}-\begin{bmatrix}\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nn}} & \widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nq}}\\ \widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qn}} & \widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qq}}\end{bmatrix}\frac{\partial}{\partial x}\begin{bmatrix}\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}\\ \widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}\end{bmatrix}\right)=\begin{bmatrix}\widetilde{\boldsymbol{s}}_{\mathrm{n}}\\ \boldsymbol{0}\end{bmatrix}, \tag{5.5}$$

note that $\widetilde{\boldsymbol{s}}_{\mathrm{e}} = \boldsymbol{0}$ since quasi-elements, by definition, cannot be produced or destroyed in chemical reactions. The block-system (5.5) is then split into two parts, (5.6a) and (5.6b); the non-quasi-element part

$$\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{U}}_{\mathrm{nn}}\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}-\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nn}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}-\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nq}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}\right)=\widetilde{\boldsymbol{s}}_{\mathrm{n}}, \tag{5.6a}$$

and the quasi-element part

$$\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{U}}_{\mathrm{qq}}\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}-\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qn}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}-\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qq}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}\right)=\boldsymbol{0}. \tag{5.6b}$$

The idea is to write $\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}$ as a superposition of the chemical equilibrium solution $\widetilde{\boldsymbol{\gamma}}$ and a perturbation $\widetilde{\boldsymbol{\delta}}$, i.e.,

$$\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}} = \widetilde{\boldsymbol{\gamma}} + \widetilde{\boldsymbol{\delta}}, \tag{5.7}$$

where the chemical equilibrium solution only depends on the quasi-elements and possibly the position

$$\widetilde{\boldsymbol{\gamma}} = \widetilde{\boldsymbol{\gamma}}(\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}, x). \tag{5.8}$$

The chemical equilibrium solution $\widetilde{\boldsymbol{\gamma}}$ can be obtained by solving $\widetilde{\boldsymbol{s}}_{\mathrm{n}}(\widetilde{\boldsymbol{\gamma}}, \widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}, \widetilde{\boldsymbol{\delta}} = \boldsymbol{0}) = \boldsymbol{0}$ for $\widetilde{\boldsymbol{\gamma}}$, which corresponds with finding $\boldsymbol{s}(\boldsymbol{\varphi}) = \boldsymbol{0}$. The process of finding the chemical equilibrium solution is discussed in Section 5.4.

With these newly introduced variables the ‘’block-system” of equations (5.5) becomes

$$\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{U}}_{\mathrm{nn}}\widetilde{\boldsymbol{\delta}}-\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nn}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\delta}}\right)=\widetilde{\boldsymbol{s}}_{\mathrm{n}}-\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{U}}_{\mathrm{nn}}\widetilde{\boldsymbol{\gamma}}-\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nn}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\gamma}}\right)+\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{nq}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}\right), \tag{5.9a}$$

and

$$\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{U}}_{\mathrm{qq}}\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}-\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qq}}\frac{\partial}{\partial x}\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}\right)=\frac{\partial}{\partial x}\left(\widetilde{\boldsymbol{\mathcal{E}}}_{\mathrm{qn}}\frac{\partial}{\partial x}(\widetilde{\boldsymbol{\gamma}}+\widetilde{\boldsymbol{\delta}})\right). \tag{5.9b}$$

The idea is to solve (5.9) iteratively, by first obtaining a new approximation to $\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}}$ from (5.9b), then solving (5.9a) for $\widetilde{\boldsymbol{\delta}}$.

In general, equations (5.9a) and (5.9b), and their discretized counterparts, are nonlinear. In the next section we describe an application of the Newton method for solving such systems, in a matrix-free method.

5.2 Solving nonlinear systems using Newton-Krylov

For a scalar nonlinear function $f(x)$ the well-known Newton method can be derived by first computing the Taylor expansion of f around the current estimate x^k,

$$f(x^{k+1}) = f(x^k) + f'(x^k)(x^{k+1} - x^k) + \mathcal{O}((x^{k+1} - x^k)^2), \tag{5.10}$$

then by setting the right-hand side to zero, dropping the higher order terms and solving for x^{k+1}, which yields the recursive relation

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}. \tag{5.11}$$

This procedure is repeated until $|f(x^{k+1})|$ and $|x^{k+1} - x^k|$ are sufficiently small, or a maximum number of iterations is reached. Newton's method can be extended to vector-valued functions $\boldsymbol{f}(\boldsymbol{x})$ as well.

To do this, we again start by computing the first-order Taylor expansion about the current estimate $\boldsymbol{x}^k$, i.e.,

$$\boldsymbol{f}(\boldsymbol{x}^{k+1}) \approx \boldsymbol{f}(\boldsymbol{x}^k) + \boldsymbol{J}(\boldsymbol{x}^k)(\boldsymbol{x}^{k+1} - \boldsymbol{x}^k), \tag{5.12}$$

where $\boldsymbol{J}(\boldsymbol{x}^k)$ is the Jacobi-matrix of $\boldsymbol{f}$ with elements given by

$$J_{ij} = \frac{\partial f_i}{\partial x_j}. \tag{5.13}$$

Then the next approximation $\boldsymbol{x}^{k+1}$ becomes

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \boldsymbol{J}^{-1}(\boldsymbol{x}^k)\boldsymbol{f}(\boldsymbol{x}^k) = \boldsymbol{x}^k - \boldsymbol{\zeta}^k, \tag{5.14}$$

where the linear system,

$$\boldsymbol{J}(\boldsymbol{x}^k)\boldsymbol{\zeta}^k = \boldsymbol{f}(\boldsymbol{x}^k), \tag{5.15}$$

has to be solved to obtain the next iterand $\boldsymbol{x}^{k+1}$. If the Jacobi-matrix is singular for some $\boldsymbol{x}^k$, there do exist modifications to Newton's method that work around this issue [97] and can still maintain near-quadratic convergence.

Explicitly computing $\boldsymbol{J}$ is expensive as it requires $\Omega(Nm^2)$ partial derivatives for a discretized nonlinear PDE problem with N grid points and m species. However, it is possible to approximate $\boldsymbol{J}\boldsymbol{v}$ for any vector $\boldsymbol{v}$ without computing N^2m^2 partial derivatives [98] by using the forward difference approximation

$$\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{v} \approx \frac{1}{h}(\boldsymbol{f}(\boldsymbol{x}_k + h\boldsymbol{v}) - \boldsymbol{f}(\boldsymbol{x}_k)), \tag{5.16}$$

or by the more accurate, and twice as expensive, central difference approximation

$$\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{v} \approx \frac{1}{2h}(\boldsymbol{f}(\boldsymbol{x}_k + h\boldsymbol{v}) - \boldsymbol{f}(\boldsymbol{x}_k - h\boldsymbol{v})), \tag{5.17}$$

for some step size $h > 0$. The idea is then to solve the linear system (5.15) with a Krylov subspace method. Such methods have the advantage that they only require matrix-vector products to solve linear systems, and do not need the matrix itself. Additionally, these methods solve a linear system iteratively, which allows early termination if the required tolerance is reached. Krylov subspace methods are described extensively in Part III.

A commonly used Krylov subspace method is GMRES [44], which is an optimal method in the sense that it requires the fewest possible number of matrix-vector products to achieve a given tolerance out of all Krylov subspace methods.

Note that if we want to compute the derivative of a function that takes as argument a vector of mass fractions, $\boldsymbol{f}(\boldsymbol{y})$, it is not correct to use any finite difference approximation such as (5.16), since this can violate $\mathbf{1}^{\mathrm{T}}\boldsymbol{y} = 1$. Consider a case similar to (5.16), where now $\boldsymbol{f}$ depends on all mass fractions $\boldsymbol{y}$, i.e.,

$$\boldsymbol{J}(\boldsymbol{y})\boldsymbol{v} \approx \frac{1}{h}(\boldsymbol{f}(\boldsymbol{y} + h\boldsymbol{v}) - \boldsymbol{f}(\boldsymbol{y})). \tag{5.18}$$

Since the input argument of $\boldsymbol{f}$ is an array of mass fractions, it has to sum to 1. Therefore, we require

$$\mathbf{1}^{\mathrm{T}}(\boldsymbol{y} + h\boldsymbol{v}) = 1, \tag{5.19}$$

however, note that since $\mathbf{1}^{\mathrm{T}}\boldsymbol{y} = 1$, it follows that $\mathbf{1}^{\mathrm{T}}\boldsymbol{v} = 0$. When applying a Newton-Krylov method, the vector $\boldsymbol{v}$ cannot be chosen arbitrarily, as it is dictated by the Krylov method, which makes it unlikely that $\mathbf{1}^{\mathrm{T}}\boldsymbol{v} = 0$ for all Krylov iterations. However, if we first transform the system, and only wish to compute the derivative of the non-element part, this issue is avoided.

In the next section we describe the model to illustrate the near-equilibrium method.

5.3 Atomic hydrogen model

To demonstrate the near-equilibrium approximation, we use a simple atomic hydrogen model. In this model the atomic hydrogen ground state, the hydrogen ion and the electron are included; this system consists of the species H, H^+ and e^- with the following reactions;

$$\mathrm{e}^- + \mathrm{H} \xrightarrow{R_{\mathrm{f}}} \mathrm{H}^+ + 2\mathrm{e}^-, \tag{5.20a}$$

$$\mathrm{H}^+ + 2\mathrm{e}^- \xrightarrow{R_\mathrm{b}} \mathrm{e}^- + \mathrm{H}, \tag{5.20b}$$

for some reaction rates $R_\mathrm{f} \geq 0$ and $R_\mathrm{b} \geq 0$, to be determined shortly.

For the electron impact ionization rate we use Equation (19) from [99], which for hydrogen is given by as a function of the electron temperature and the energy level of the hydrogen atom, i.e.,

$$R_\mathrm{f}(T_\mathrm{q}, p) = 8\pi a_0^2 p^4 f_p \sqrt{\frac{2k_\mathrm{B}T_\mathrm{q}}{\pi m_\mathrm{q}}} u(T_\mathrm{q}, p)\psi(T_\mathrm{q}, p), \tag{5.21}$$

where p is the principal quantum number determining the energy level, the variables u and ψ are obtained from an integration of the cross section modeled in [99], resulting in

$$u(T_\mathrm{q}, p) = \frac{E_p}{k_B T_\mathrm{q}}, \quad \psi(T_\mathrm{q}, p) = \int_{t=u}^{\infty} (1 - t/u)e^{-t} \ln(1.25t/u)\mathrm{d}t. \tag{5.22}$$

The values of f_p are $f_1 = 0.665$, $f_2 = 0.71$, $f_3 = 0.81$, $f_4 = 0.94$ $f_{\geq 5} = 1$. The recombination reaction is also based on the principle of detailed balancing [99], and is given by

$$R_\mathrm{b}(T_\mathrm{q}, p) = \frac{g_{\mathrm{neutral},p}}{2g_\mathrm{ion}} \frac{h^3}{(2\pi m_\mathrm{q} k_\mathrm{B} T_\mathrm{q})^{3/2}} R_\mathrm{f}(T_\mathrm{q}, p) \exp(u(T_\mathrm{q}, p)), \tag{5.23}$$

where h is Planck's constant, $g_{\mathrm{neutral},n}$ and g_ion are the statistical weights of the neutral and ionized states, respectively. For this reaction set, which only includes the ground state and the hydrogen ion both $g_{\mathrm{neutral},1} = 1$ and $g_\mathrm{ion} = 1$.

It can be shown that the stoichiometric transformation again leads to two transformed variables which are constant. We start by ordering the species as follows

$$\boldsymbol{y} = \begin{bmatrix} y_\mathrm{H} \\ y_{\mathrm{H}^+} \\ y_{\mathrm{e}^-} \end{bmatrix}, \tag{5.24}$$

and the reactions are ordered as in (5.20). The stoichiometric matrix $\boldsymbol{\nu}$ is then given by

$$\boldsymbol{\nu} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}, \tag{5.25}$$

and a stoichiometric transformation matrix by

$$\boldsymbol{M} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}, \tag{5.26}$$

for which it can be verified that $\boldsymbol{M\nu}$ results in a matrix with the bottom two rows equal to zero.

The scaled stoichiometric matrix $\boldsymbol{T}$ as elaborated in Chapter 4 and [96] is given by

$$\boldsymbol{T} = \begin{bmatrix} 0 & 0 & m_{\mathrm{H}}/m_{\mathrm{e}^-} \\ m_{\mathrm{H}^+}/m_{\mathrm{H}} & 0 & m_{\mathrm{H}^+}/m_{\mathrm{e}^-} \\ 0 & m_{\mathrm{e}^-}/m_{\mathrm{H}^+} & -1 \end{bmatrix}. \tag{5.27}$$

We show that two of the transformed mass fractions are constant. To do this we compute $\boldsymbol{Ty}$ and expand the bottom two rows of the result,

$$\boldsymbol{Ty} = \begin{bmatrix} \frac{m_{\mathrm{H}}}{m_{\mathrm{e}^-}} y_{\mathrm{e}^-} \\ \frac{m_{\mathrm{H}^+}}{m_{\mathrm{H}}} y_{\mathrm{H}} + \frac{m_{\mathrm{H}^+}}{m_{\mathrm{e}^-}} y_{\mathrm{e}^-} \\ \frac{m_{\mathrm{e}^-}}{m_{\mathrm{H}^+}} y_{\mathrm{H}^+} - y_{\mathrm{e}^-} \end{bmatrix}. \tag{5.28}$$

It then follows that the bottom component, $[\boldsymbol{Ty}]_3$, is zero if the plasma is quasi-neutral i.e., $n_{\mathrm{H}^+} = n_{\mathrm{e}^-}$,

$$[\boldsymbol{Ty}]_3 = \frac{m_{\mathrm{e}^-}}{\rho}(n_{\mathrm{H}^+} - n_{\mathrm{e}^-}) = 0. \tag{5.29}$$

The second component can be shown to be constant by using the result that $[\boldsymbol{Ty}]_3 = 0$, using the relation between the masses of the species $m_{\mathrm{H}} = m_{\mathrm{H}^+} + m_{\mathrm{e}^-}$, and the definition that the mass fractions sum to one, i.e., $y_{\mathrm{H}} + y_{\mathrm{H}^+} + y_{\mathrm{e}^-} = 1$;

$$[\boldsymbol{Ty}]_2 = \frac{m_{\mathrm{H}^+}}{m_{\mathrm{H}}}\left(y_{\mathrm{H}} + \frac{m_{\mathrm{H}^+} + m_{\mathrm{e}^-}}{m_{\mathrm{e}^-}} y_{\mathrm{e}^-}\right) = \frac{m_{\mathrm{H}^+}}{m_{\mathrm{H}}}. \tag{5.30}$$

This result is used in the numerical experiments to eliminate two of the transformed variables. In the next section it is described how the chemical equilibrium solution for this model is obtained.

5.4 Computing the chemical equilibrium solution

To obtain the chemical equilibrium solution we integrate (5.2) as if it was a perfectly stirred reactor;

$$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\boldsymbol{\varphi}} = \widetilde{\boldsymbol{s}}, \tag{5.31}$$

over a sufficiently large time span to reach steady state, similar to the method of false transients. If the chemical equilibrium is unique, then the time integration should converge to the same equilibrium regardless of the initial value of $\widetilde{\boldsymbol{\varphi}}_{\mathrm{n}}$.

The idea of evaluating an ODE system instead of directly using a (constrained) nonlinear solver is to arrive at a solution with a stable chemical equilibrium; the same solution the actual chemical system would reach if it were to evolve in time. The transformed system is solved to guarantee the conservation of elements for every time step of the ODE solver, since the element quantities are source free, and thus do not change in time.

Chemical reaction rates can vary over a wide range of timescales, leading to potentially high stiffness, therefore we use an ODE solver designed for stiff ODEs, in this case the MATLAB solver `ode15s` [100].

Since this stiff solver employs an implicit method, a nonlinear algebraic system has to be solved every time step. To guarantee that the concentration of quasi-elements is constant, we explicitly include

$$\frac{\mathrm{d}}{\mathrm{d}t}\widetilde{\boldsymbol{\varphi}}_{\mathrm{e}} = \mathbf{0}, \tag{5.32}$$

to enforce this. Finally, a refinement step is performed, where we use the previously computed solution of (5.31) as an initial guess for another round of integration of the system (5.31).

The resulting chemical equilibrium solution as function of temperature for the simple hydrogen system is shown in Figure 5.4.1. Here it can be seen that most of the hydrogen is in atomic state at low temperatures, whereas around 10^4 K the chemical equilibrium shifts toward mostly ionized. In the next section it is shown that the chemical equilibrium solution is both unique and stable.

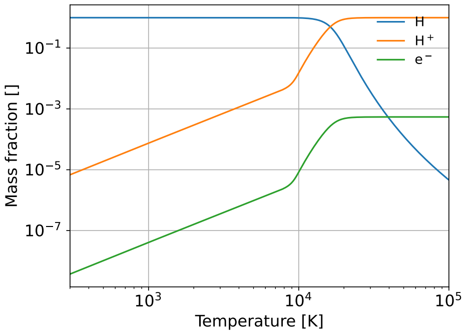


Figure 5.4.1: Chemical equilibrium solution as function of temperature.

5.5 Stability and uniqueness of the chemical equilibrium

It can be shown for a simple three-species hydrogen system, under certain conditions, that the chemical equilibrium point is stable and unique.

The chemical source term in terms of number densities is given by

$$\boldsymbol{S}(\boldsymbol{n}) = \boldsymbol{\nu} \begin{bmatrix} R_\mathrm{f} n_\mathrm{H} n_{\mathrm{e}^-} \\ R_\mathrm{b} n_{\mathrm{e}^-}^2 n_{\mathrm{H}^+} \end{bmatrix}, \tag{5.33}$$

with $\boldsymbol{\nu}$ given in (5.25). Assuming $R_\mathrm{f} > 0$ and $R_\mathrm{b} > 0$ there are three situations for which $\boldsymbol{S}(\boldsymbol{n}) = \boldsymbol{0}$, namely:

1. $n_{\mathrm{e}^-} = 0$, no electrons to ionize or recombine.
2. $n_\mathrm{H} = 0$ and $n_{\mathrm{H}^+} = 0$, a system consisting of only electrons.
3. $\frac{R_\mathrm{f}}{R_\mathrm{b}} = \frac{n_{\mathrm{e}^-} n_{\mathrm{H}^+}}{n_\mathrm{H}}$, a balance between the forward and backward reactions.

Given that all $n > 0$, the only remaining solution for $\boldsymbol{S}(\boldsymbol{n}) = \boldsymbol{0}$ is when there is a balance between ionization and recombination;

$$\frac{R_\mathrm{f}}{R_\mathrm{b}} = \frac{n_{\mathrm{e}^-} n_{\mathrm{H}^+}}{n_\mathrm{H}}. \tag{5.34}$$

To show that this chemical equilibrium is a stable equilibrium, we expand $\boldsymbol{S}(\boldsymbol{n})$ in a Taylor-series up to first order, about the chemical equilibrium, then substitute (5.34)

$$\boldsymbol{S}(\boldsymbol{n}) \approx \boldsymbol{J}(\boldsymbol{n}_\mathrm{eq})(\boldsymbol{n} - \boldsymbol{n}_\mathrm{eq}), \tag{5.35}$$

and compute the eigenvalues of the Jacobi matrix $\boldsymbol{J}(\boldsymbol{n}_\mathrm{eq})$. It can be shown in a straightforward manner that the eigenvalues λ_i of $\boldsymbol{J}(\boldsymbol{n}_\mathrm{eq})$ are given by;

$$\lambda_1 = 0, \quad \lambda_2 = 0 \quad \lambda_3 = -R_\mathrm{b} \frac{n_{\mathrm{e}^-}}{n_\mathrm{H}} (n_{\mathrm{e}^-} n_\mathrm{H} + n_{\mathrm{e}^-} n_{\mathrm{H}^+} + n_\mathrm{H} n_{\mathrm{H}^+}), \tag{5.36}$$

since $R_\mathrm{f} > 0$, $R_\mathrm{b} > 0$ and $n > 0$, it can be concluded that $\lambda_3 < 0$ and thus this equilibrium is stable. Therefore, the only chemical equilibrium with $n > 0$ is stable.

To show that an equilibrium is stable, if the eigenvalues are negative, consider (5.35) and the eigenvalue decomposition of $\boldsymbol{J}(\boldsymbol{n}_\mathrm{eq}) = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$ given by

$$\boldsymbol{V} = \begin{bmatrix} \frac{n_\mathrm{H}}{n_{\mathrm{e}^-}} & \frac{n_\mathrm{H}}{n_{\mathrm{H}^+}} & -1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}, \tag{5.37}$$

where $\boldsymbol{V}$ is invertible, as it has a strictly positive determinant;

$$\det(\boldsymbol{V}) = 1 + n_{\mathrm{H}} \left(\frac{1}{n_{\mathrm{e}^-}} + \frac{1}{n_{\mathrm{H}^+}} \right) > 0. \tag{5.38}$$

Since the eigenvalue decomposition exists, we can write

$$\frac{\mathrm{d}\boldsymbol{n}}{\mathrm{d}t} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}(\boldsymbol{n} - \boldsymbol{n}_{\mathrm{eq}}), \tag{5.39}$$

then we obtain a decoupled ODE system,

$$\frac{\mathrm{d}\boldsymbol{V}^{-1}\boldsymbol{n}}{\mathrm{d}t} = \boldsymbol{\Lambda}\boldsymbol{V}^{-1}(\boldsymbol{n} - \boldsymbol{n}_{\mathrm{eq}}). \tag{5.40}$$

Since $\boldsymbol{\Lambda}$ is diagonal, the solution for each of the components of $\boldsymbol{V}^{-1}\boldsymbol{n}$ is given by

$$[\boldsymbol{V}^{-1}\boldsymbol{n}]_i = C_i \exp(\lambda_i t) + [\boldsymbol{V}^{-1}\boldsymbol{n}_{\mathrm{eq}}]_i, \tag{5.41}$$

with C_i a constant of integration. Therefore, if the real part of each λ_i is negative, the solution exponentially tends toward chemical equilibrium, invariants for which $\lambda_i = 0$ remain constant.

In the next section the results of the simple hydrogen model are presented for a one-dimensional ADR problem.

5.6 Results

We compute a solution over the domain extending from $x = 0$ to $x = L$, with the length of the domain $L = 0.1$ m, and use 50 grid points. The HF scheme has been used to discretize Equations (5.9). The pressure has been fixed at 10^5 Pa, and there is no advection. Dirichlet boundary conditions are used given by

$$\boldsymbol{y}(x = 0) = \begin{bmatrix} 0.49986 \\ 0.49986 \\ 2.722 \times 10^{-4} \end{bmatrix}, \quad \boldsymbol{y}(x = L) = \begin{bmatrix} 0.9999 \\ 9.999 \times 10^{-5} \\ 5.445 \times 10^{-8} \end{bmatrix}, \tag{5.42}$$

these are chosen such that at both boundaries quasi-neutrality and $\mathbf{1}^{\mathrm{T}}\boldsymbol{y} = 0$ hold. For simplicity the electron and ion temperatures have been set equal. The temperature profile shown in Figure 5.6.1 has been chosen such that there is a large drop in temperature, which causes the equilibrium solution to also change significantly.

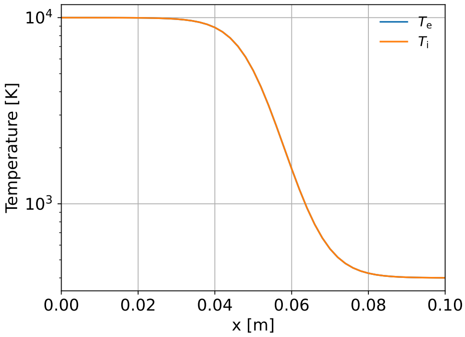
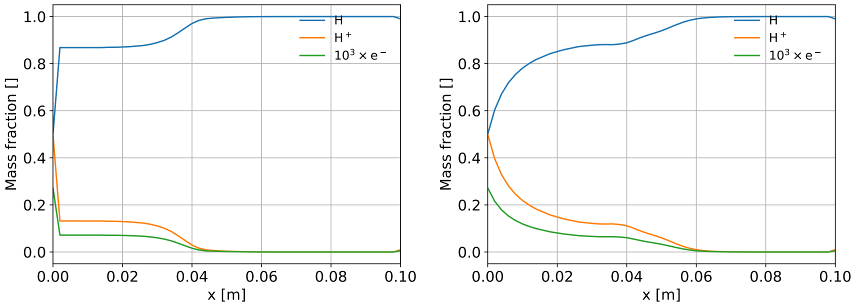


Figure 5.6.1: Temperature distribution



(a) Chemical equilibrium solution

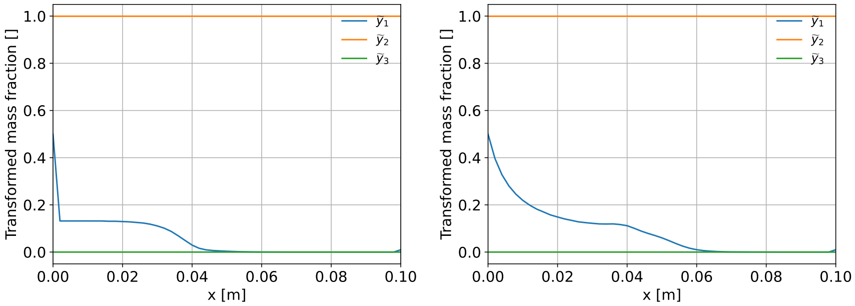
(b) Actual solution

Figure 5.6.2: Comparison of the chemical equilibrium solution and the actual solution for the atomic hydrogen model.

It can be seen in Figure 5.6.2 that the equilibrium solution qualitatively shows the same behavior as the actual solution. However, the actual solution is smoother due to transport. In Figure 5.6.3 it can be seen that for both the chemical equilibrium solution and the transformed solution, y_2 and y_3 are constant. Furthermore, it can also be seen that qualitatively y_1 is similar for both solutions.

5.7 Discussion and conclusion

A hydrogen model has been set up to illustrate the method of the near-equilibrium approximation. Firstly, the set of governing equations has been linearly transformed based on the stoichiometry of the problem. Secondly, the non-element variable has been written as a superposition of the chemical equilibrium solution, and a deviation from this chemical equilibrium.



(a) Transformed chemical equilibrium solution

(b) Transformed actual solution

Figure 5.6.3: Comparison of the transformed chemical equilibrium and actual solutions for the atomic hydrogen model.

As a result, two of the three unknowns can be replaced by constants, as these represent quasi-neutrality and conservation of mass up to a constant factor. For the remaining equation, we illustrate the idea of using a Newton-Krylov method. In this method the Jacobian is estimated using a finite difference formula. In particular the stoichiometric transformation ensures that even though the Jacobian is inexact, the invariants such as quasi-neutrality and conservation of mass are still satisfied exactly.

The chemical equilibrium solution has been obtained by integrating a zero-dimensional coupled ODE system. This was done as it did not get stuck in unstable (near)equilibria, which are an issue when a nonlinear root finding method is used to solve $\boldsymbol{S}(\boldsymbol{y}) = \boldsymbol{0}$. Similar to the one-dimensional model, the stoichiometric transformation has been used here to ensure all invariants are satisfied. Furthermore, the invariants reduce the number of unknowns to the extent where we only have to tabulate the $\boldsymbol{S}(\boldsymbol{y})$ as a function of temperature.

In conclusion, the stoichiometric transformation can be used to eliminate invariants, ensure exact conservation of invariants, and simplify tabulation of the chemical equilibrium solution. The near-equilibrium approximation extension can be used to get an improved estimate of the true solution.

5.8 Outlook

Even though the stoichiometric transformation matrix is relatively straightforward to compute if the stoichiometric matrix is known, the number of variables for which the source term is eliminated may be modest. For example the CO_2 model of [101] has 72 species and 5732 reactions, however, the number of quasi-elements for this reaction system likely does not exceed 3, since the only chemical elements present are C and O. Similarly for the 78 species argon model of [92], which has only two quasi-elements.

Furthermore, computing the chemical equilibrium solution is challenging for complex chemistries. Since it is not known a priori to which time the zero-dimensional model has to be integrated. Additionally, if the electron and heavy species temperatures are not equal, or there is another similar input that is required, tabulating the chemical equilibrium becomes much more expensive.

Nevertheless, testing the near-equilibrium extension for a complex model may be a next step, possibly in combination with some reduction method such as pathway analysis presented in [24]. Additionally, the Newton-Krylov method is expected to show more favorable convergence over the previously used Picard iteration used in [96].

Another possibly novel extension could be to apply the near-equilibrium approximation to time dependent models. For such models the chemical equilibrium solution would also evolve in time, whereas simply choosing a different initial guess would not allow for this flexibility.

Finally, another outlook would be to do a timescale analysis, and investigate the effect of the near-equilibrium approximation on stiffness. Typically, the timescales of chemical reactions in plasmas span a wide range. By starting from the chemical equilibrium solution, fast reactions may be less prevalent. Compared to starting from a linear interpolation between the boundary conditions, which is likely to lead to strong reactions that equilibrate on a short timescale.

5.8.1 Enhanced procedure for finding the chemical equilibrium

One of the possible improvements for this method, would be to find more efficient methods of finding the chemical equilibrium. In Section 5.4 we found the chemical equilibrium for a three-species hydrogen example. There have been attempts to find the chemical equilibrium for a 20 species atomic hydrogen model. However, the runtime of finding the chemical equilibrium with the method proposed in Section 5.4 is far too great to be of practical use ($>$ 137 hours). This is likely due to the time-integration requiring a time step that is too small to make meaningful progress. Additionally, the method does not guarantee that every mass fraction is in the range $[0, 1]$, only that they add up to 1, and that quasi-neutrality is achieved.

An alternative idea would be to use a constrained nonlinear optimization procedure, for example MATLAB's `fmincon`. However, the attempts made to use such a procedure for this thesis did not lead to satisfactory chemical equilibria results. Instead, issues such as the solver breaking down, returning a local optimum, returning a potentially unstable equilibrium, returning a solution that does not satisfy the constraints, were encountered.

A possibly better approach would be to assume a type of Saha/Boltzmann equilibrium for some species and use an analytical expression. However, this leaves an open problem: what to do with the other species?

Finally, a more advanced method of finding the chemical equilibrium could be implemented, for example the strategy presented in [102].

5.8.2 Applications

An important application for the near-equilibrium method could be plasmas with a strong temperature gradient and little transport. For such plasmas the chemical equilibrium solution would largely determine the real solution. Solving the derivation relative to the chemical equilibrium may be faster, as the chemical equilibrium is a good reference.

Since the near-equilibrium method starts at $\boldsymbol{\delta} = \mathbf{0}$, i.e., it starts at the chemical equilibrium, the chemical reactions may be less extreme compared to starting from a linear interpolation between the boundary conditions. Similar to what is shown in the context of ILDM, where near equilibrium only the longer timescales remain [89]. This may lead to a numerically more favorable approach.

5.8.3 Extensions

An alternative approach to solving the system (5.9), would be to solve (5.9b) for $\widetilde{\boldsymbol{\delta}}$, and solve (5.9a) for $\widetilde{\varphi}_{\mathrm{e}}$. This way there is no advection operator applied to the variable we want to solve for.

To check if the near-equilibrium method is applicable, an idea would be to consider only the diffusion part near equilibrium

$$-\boldsymbol{\mathcal{E}}\frac{\partial^2 \boldsymbol{\varphi}}{\partial x^2} = \boldsymbol{J}(\boldsymbol{\varphi}_{\mathrm{eq}})(\boldsymbol{\varphi} - \boldsymbol{\varphi}_{\mathrm{eq}}), \tag{5.43}$$

then if we introduce a length scale L, a dimensionless length $\widetilde{x} := x/L$, and multiply both sides with $\boldsymbol{\mathcal{E}}^{-1}$, this results in

$$-\frac{\partial^2 \boldsymbol{\varphi}}{\partial \widetilde{x}^2} = L^2 \boldsymbol{\mathcal{E}}^{-1} \boldsymbol{J}(\boldsymbol{\varphi}_{\mathrm{eq}})(\boldsymbol{\varphi} - \boldsymbol{\varphi}_{\mathrm{eq}}). \tag{5.44}$$

Here the matrix $L^2\boldsymbol{\mathcal{E}}^{-1}\boldsymbol{J}$ can be identified as a Damköhler matrix $\mathbf{Da}$. Assuming the Damköhler matrix has the spectral decomposition $\mathbf{Da} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}$, and $\boldsymbol{V}^{-1}$ is “slowly varying” in x, we can solve (5.44) as

$$[\boldsymbol{V}^{-1}\varphi]_k = c_1 \sin\left(x\sqrt{\lambda_k}\right) + c_2 \cos\left(x\sqrt{\lambda_k}\right) + [\boldsymbol{V}^{-1}\boldsymbol{\varphi}_{\mathrm{eq}}]_k. \tag{5.45}$$

One extension to the near-equilibrium method would then be to use (5.45) as a reference solution.

Presumably the near-equilibrium method works better if the eigenvalues of $\mathbf{Da}$ are large. As large Damköhler numbers indicate that the reaction rates are much more significant than transport rates. Investigating the eigenvalues of $\mathbf{Da}$ may thus give a measure of quality for the chemical equilibrium solution.

Part III

Krylov methods

Chapter 6

Reliability investigation of BiCGStab and IDR solvers for the advection-diffusion-reaction equation

This chapter is based on:
Reliability investigation of BiCGStab and IDR solvers for the advection-diffusion-reaction equation
Schoutrop, C., Van Dijk, J. and Ten Thije Boonkkamp, J., Communications in Computational Physics, vol 32, p. 156-188. doi: 10.4208/cicp.OA-2021-0182.

Abstract. The reliability of BiCGStab and IDR solvers for the exponential scheme discretization of the advection-diffusion-reaction equation is investigated. The resulting discretization matrices have real eigenvalues. We consider BiCGStab, IDR(S), BiCGStab(L) and various modifications of BiCGStab, where S denotes the dimension of the shadow space and L the degree of the polynomial used in the polynomial part. Several implementations of BiCGStab exist which are equivalent in exact arithmetic, however, not in finite precision arithmetic. The modifications of BiCGStab we consider are; choosing a random shadow vector, a reliable updating scheme, and storing the best intermediate solution. It is shown that the Local Minimal Residual algorithm, a method similar to the "minimize residual" step of BiCGStab, can be interpreted in terms of a time-dependent advection-diffusion-reaction equation with homogeneous Dirichlet boundary conditions for the residual, which plays a key role in the convergence analysis. Due to the real eigenvalues, the benefit of BiCGStab(L) compared to BiCGStab is shown to be modest in numerical experiments. Non-sparse (e.g. uniform random) shadow residual turns out to be essential for the reliability

of BiCGStab. The reliable updating scheme ensures the required tolerance is truly achieved. Keeping the best intermediate solution has no significant effect. Recommendation is to modify BiCGStab with a random shadow residual and the reliable updating scheme, especially in the regime of large Péclet and small Damköhler numbers. An alternative option is IDR(S), which outperforms BiCGStab for problems with strong advection in terms of the number of matrix-vector products. The MATLAB code used in the numerical experiments is available on GitLab: `https://gitlab.com/ChrisSchoutrop/krylov-adr`, a C++ implementation of IDR(S) is available in the Eigen linear algebra library: `http://eigen.tuxfamily.org`.

6.1 Introduction

Advection-diffusion-reaction (ADR) equations and their discrete approximations are ubiquitous in the modeling of physical systems [103, 104, 105, 106, 107]. A wide variety of discretization schemes for the ADR equation, such as finite difference, (pseudo)spectral, finite element and finite volume methods exist. For the solution to be representative, the discretization scheme must not only be convergent in the limit of infinitesimally fine grids, but also yield representative results for more pragmatic grid sizes. A counterexample, is the discretization of the ADR equation in the presence of strong advection where the central differencing scheme yields spurious oscillations if the grid is too coarse [26, p. 83]. This discrepancy is reflected by the eigenvalues of the exact ADR-operator and the discretized operator, i.e., the exact operator has *real* eigenvalues, whereas the discretized version has *complex* eigenvalues. However, for the exponential discretization scheme of [26, 108] which we use here, the discretized version also has real eigenvalues.

After the discretization step a linear system is obtained which must be solved to obtain the approximate solution. Such linear systems are of the type $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, with $\boldsymbol{A}$ a generally sparse, asymmetric, but invertible matrix of size $N \times N$. In this chapter we mainly consider 3D equations. Note that even with a modest $M = 102$ grid points per direction this results in a linear system with $N = 10^6$ unknowns. A robust method for solving such linear systems is by factorizing $\boldsymbol{A}$ into a pair of lower and upper triangular matrices using the well-known LU decomposition[32, p. 96], and subsequently computing $\boldsymbol{x}$ by solving two triangular systems using backward and forward substitution. The main downside of LU decomposition is that for a sparse system matrix there can be significant fill-in; the factors $\boldsymbol{L}$ and $\boldsymbol{U}$ are not guaranteed to be sparse. As a result the time complexity of factorizing the resulting $\boldsymbol{A}$ for discretized three-dimensional ADR-equations is in general $\mathcal{O}(N^{7/3})$ [33].

Another approach are iterative methods, most commonly the Krylov subspace methods such as the Conjugate Gradient (CG) method. Such methods seek successive approximations to the solution as a projection on the linear subspace $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0) = \operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \boldsymbol{A}^2\boldsymbol{r}_0, ..., \boldsymbol{A}^{k-1}\boldsymbol{r}_0\}$ with $\boldsymbol{r}_0 := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ the residual generated by some initial guess $\boldsymbol{x}_0$. For linear systems obtained from discretizing a second order PDE with a grid spacing proportional to $N^{-1/3}$, the CG method requires $\mathcal{O}(N^{4/3})$ flops to reach a given tolerance ϵ [33], saving a factor N compared to the LU decomposition. A second benefit is that the memory requirement of $\mathcal{O}(N)$ is modest, as only the matrix itself and a constant number of vectors of size N have to be stored. Additionally, CG is an optimal Krylov method in the sense that in each iteration the error is minimized over the

$\boldsymbol{A}$-norm [34].

The main drawback of CG is the requirement of a symmetric, positive definite matrix $\boldsymbol{A}$. For more general invertible matrices the Faber-Manteuffel theorem [35] states that there is no Krylov subspace method using $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$ that has short recurrences and optimality [36]. Short recurrence methods express the next residual and solution approximation in terms of a fixed (practically small) number of previous residuals and iterands. However, in long recurrence methods the number of terms in the recurrence grows with the iteration count. Optimality here refers to the minimization of the error $\boldsymbol{e}_k := \boldsymbol{x} - \boldsymbol{x}_k \in \text{span}\{\boldsymbol{x} - \boldsymbol{x}_0\} + \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$ in some norm. Therefore, for a general invertible matrix $\boldsymbol{A}$ one has to either, drop the optimality requirement, use long recurrences, use a space different than $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$, or settle for a compromise. This leads to a wide variety of Krylov methods, each with different tradeoffs.

One possible extension of CG to solve non-symmetric systems is the BiCG algorithm originally described in [37]. The idea of BiCG is to use a second Krylov subspace denoted as the "shadow space"; $\mathcal{K}_k(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0)$, where the vector $\widetilde{\boldsymbol{r}}_0$ is arbitrary but commonly chosen as $\boldsymbol{r}_0$. Unfortunately, the BiCG algorithm has several drawbacks as well. First, unlike CG the residuals produced by the BiCG algorithm are not guaranteed to decrease each iteration in some norm. Second, additional computational effort is needed in matrix vector products involving $\boldsymbol{A}^{\mathrm{T}}$ [32, p. 247]. Third, matrix-vector products of the form $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{v}$ may be prohibitively expensive to compute, making the BiCG method unsuitable for some applications such as Newton-Krylov methods [32, 104, p. 241].

To alleviate these issues the well-known BiCGStab algorithm was developed. This method was originally described in [38] and has since then been incorporated in many popular linear algebra packages such as MATLAB [39] and Eigen [40], and in simulation software such as PLASIMO [41] and COMSOL [42]. BiCGStab combines one iteration of BiCG with a Local Minimal Residual (LMR) step [43]; this step can also be seen as performing one iteration of the GMRES method [44]. LMR chooses the current residual as a search direction, and takes a step in this direction such that the next residual is minimized, which results in a smoother decrease of the residual norm of BiCGStab compared to BiCG. Additionally, BiCGStab does not require operations involving $\boldsymbol{A}^{\mathrm{T}}$. However, this combination of LMR and BiCG leads to several subtleties which are discussed in Section 6.2.

To compare BiCGStab with other Krylov methods, we also include BiCGStab(L) and IDR(S). The BiCGStab(L) method, combines L steps of BiCG with a GMRES(L) step [45]. Another method we include is the recent IDR(S) algorithm, which can be seen as using S different shadow spaces; $\mathcal{K}_k(\boldsymbol{A}, \widetilde{\boldsymbol{R}})$ with $\widetilde{\boldsymbol{R}}$ an $N \times S$ matrix. IDR(S) is constructed such that the residuals are forced into a subspace that decreases in dimension each iteration [46]. Even though not as widespread as BiCGStab, IDR(S) outperforms BiCGStab for a wide class of problems [46, 47]. We focus on BiCGStab, BiCGStab(L), IDR(S), and most importantly the differences in reliability arising in the implementations of the BiCGStab algorithm.

In this chapter we focus on specific implementations of the BiCGStab algorithm, and show that small differences in these implementations can have a significant impact on the reliability of this solver. Here we place the effects of these differences in a physical context by considering the advection-diffusion-reaction equation. Furthermore, we do not use the central differencing discretization scheme, but apply the exponential scheme for which the common critique that BiCGStab cannot handle complex eigenvalues does not apply. We then show that even though the eigenvalues are real, there are still pitfalls for a widely used BiCGStab implementation, and we present possible mitigations. Additionally we show that even for preconditioned systems these modifications are still applicable. To summarize, the objective and novelty of this chapter is:

1. Investigate the reliability of BiCGStab, BiCGStab(L) and IDR(S) for advection-diffusion-reaction equations discretized with the exponential scheme, resulting in matrices with real eigenvalues.
2. Discuss the subtleties in different BiCGStab implementations for model linear systems.
3. Discuss pitfalls and remedies for several BiCGStab implementations.
4. Highlight the critical effect of the choice of a shadow residual $\widetilde{\boldsymbol{r}}$ in BiCGStab.
5. Illustrate that for the LMR method, the residual propagates similarly to the solution of a time-dependent advection-diffusion equation. This phenomenon is used to obtain insight into the convergence of BiCGStab.
6. Present numerical results suggesting that the residual for BiCGStab is transported similarly to the solution of a time-dependent advection-diffusion problem to support the observation regarding the shadow residual $\widetilde{\boldsymbol{r}}$.
7. Show that even for preconditioned systems the proposed modifications are still relevant.

Typically, large sparse linear systems are solved using preconditioned iterations. We investigate the effect of Jacobi, incomplete LU and geometric multigrid preconditioners for the MATLAB implementation of BiCGStab. Such preconditioners do indeed significantly reduce the number of required matrix-vector products, however, we show that it does not alleviate all shortcomings. To simplify the analysis, and work toward a robust BiCGStab variant that converges for a broad range of systems we mainly consider unpreconditioned systems in the coming sections.

The contents of this chapter are the following. First, the BiCGStab algorithm and different implementations are discussed in Section 6.2. Second, the model ADR-system along with the discretization and resulting eigenvalues are introduced in Section 6.3. Third, several numerical experiments are presented and discussed in Section 6.4. Finally, we end with concluding remarks in Section 6.5.

In conclusion, we recommend to use BiCGStab with a random shadow residual $\widetilde{\boldsymbol{r}}$ and reliable updating scheme, or alternatively the efficient IDR(S) algorithm. The MATLAB code used in the numerical experiments is available on GitLab: `https://gitlab.com/ChrisSchoutrop/krylov-adr` a C++ implementation of IDR(S) is available in the Eigen library [40].

6.2 Solver implementation

One of the main goals of this chapter is to compare BiCGStab and IDR. However, this is not as straightforward as one might expect. For example, the BiCGStab algorithm has several variants, which are not identical for finite precision arithmetic. A common alternative to BiCGStab is BiCGStab(L), which modifies BiCGStab to include a higher order stabilizing polynomial step and reduces to BiCGStab for $L = 1$. The higher order polynomial step makes BiCGStab(L) more robust for matrices having eigenvalues with large imaginary parts [45]. It is shown in [109] that BiCGStab(L) can be implemented in different ways, affecting both stability and computational complexity. There also exist other algorithms which reduce to BiCGStab for specific parameters, such as IDR(S) [46] and IDR(S)Stab(L) [47, 110]. In addition, there are several variants for specific situations, such as a version of BiCGStab geared specifically for parallel computers; see e.g. [111] and [112], a block-version [113] specialized for solving systems with multiple right-hand sides, and a recycling version [114] optimized for solving a sequence of linear systems.

The BiCGStab algorithm which we list here for the sake of completeness is based on Algorithm 1 of [115] with $L = 1$, given here as the baseline BiCGStab Algorithm 3. In this chapter $\langle \boldsymbol{a}, \boldsymbol{b} \rangle := \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$ denotes the inner product of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. In Algorithm 3 the MATLAB notation is used, i.e., $\widehat{\boldsymbol{r}}_{:,1}$ denotes the entire first column of the matrix $\widehat{\boldsymbol{r}}$. Unless mentioned explicitly all arithmetic is conducted over the real numbers.

As mentioned in the introduction, BiCGStab combines BiCG with LMR. In Algorithm 3, specifically in Lines $2-6$, the initial residual $\widehat{\boldsymbol{r}}_{:,1}$, the search directions $\widehat{\boldsymbol{u}}$ and the shadow residual $\widetilde{\boldsymbol{r}}$ are initialized. The next part (lines $8-15$) performs one BiCG iteration, and the lines $16-19$ perform the LMR-step. The LMR-step can be seen as choosing the search direction equal to the residual $\widehat{\boldsymbol{r}}_{:,1}$, and computing ω such that the norm of the next residual, $\|\widehat{\boldsymbol{r}}_{:,1} - \omega\widehat{\boldsymbol{r}}_{:,2}\|_2$, is minimized. Note that $\widehat{\boldsymbol{r}}_{:,1}$ contains the residual and $\widehat{\boldsymbol{r}}_{:,2}$ is an auxiliary vector used in the LMR step.

To illustrate some of the differences between several BiCGStab implementations, we focus on three specific modifications that require little alteration to the baseline BiCGStab algorithm, i.e.,

1. Random shadow residual, $\widetilde{\boldsymbol{r}}$, Section 6.2.1.
2. Reliable update scheme from [116], Section 6.2.2.
3. Storing solution with the best residual thus far, Section 6.2.3.

In the next sections we illustrate shortcomings of baseline BiCGStab using small model problems, and comment on the effect of these three modifications. In Section 6.2.4 it is shown how the initial residual is propagated similarly to the solution of a time-dependent advection equation, which is argued to be the main reason for the effectiveness of choosing a random shadow residual later in this chapter.

Algorithm 3 Baseline BiCGStab

```
 1: function BASELINE BICGSTAB($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol$)
 2:     $\rho_0 \leftarrow 1, \alpha \leftarrow 1, \omega \leftarrow 1, \widehat{\boldsymbol{r}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{N\times 2}, \widehat{\boldsymbol{u}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{N\times 2}$
 3:     $\widehat{\boldsymbol{r}}_{:,1} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
 4:     $\widetilde{\boldsymbol{r}} \leftarrow \widehat{\boldsymbol{r}}_{:,1}$                    ▷ Arbitrary, such that $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle \neq 0$
 5:     $\boldsymbol{x}' \leftarrow \boldsymbol{x}, \quad \boldsymbol{x} \leftarrow \boldsymbol{0}$
 6:     $\zeta \leftarrow \|\widehat{\boldsymbol{r}}_{:,1}\|$
 7:     while $\zeta > tol\|\boldsymbol{b}\|$ do
 8:         $\rho_0 \leftarrow -\rho_0\omega$
 9:         $\rho_1 \leftarrow \langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle$
10:         $\beta \leftarrow \alpha(\rho_1/\rho_0), \quad \rho_0 \leftarrow \rho_1$
11:         $\widehat{\boldsymbol{u}}_{:,1} \leftarrow \widehat{\boldsymbol{r}}_{:,1} - \beta\widehat{\boldsymbol{u}}_{:,1}$
12:         $\widehat{\boldsymbol{u}}_{:,2} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{u}}_{:,1}$
13:         $\alpha \leftarrow \rho_1/\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{u}}_{:,2} \rangle$
14:         $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\widehat{\boldsymbol{u}}_{:,1}$
15:         $\widehat{\boldsymbol{r}}_{:,1} \leftarrow \widehat{\boldsymbol{r}}_{:,1} - \alpha\widehat{\boldsymbol{u}}_{:,2}$
16:         $\widehat{\boldsymbol{r}}_{:,2} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{r}}_{:,1}$
17:         $\omega \leftarrow \text{argmin}_\omega \|\widehat{\boldsymbol{r}}_{:,1} - \omega\widehat{\boldsymbol{r}}_{:,2}\| = \langle \widehat{\boldsymbol{r}}_{:,2}, \widehat{\boldsymbol{r}}_{:,1} \rangle / \langle \widehat{\boldsymbol{r}}_{:,2}, \widehat{\boldsymbol{r}}_{:,2} \rangle$
18:         $\boldsymbol{x} \leftarrow \boldsymbol{x} + \omega\widehat{\boldsymbol{r}}_{:,1}$
19:         $\widehat{\boldsymbol{r}}_{:,1} \leftarrow \widehat{\boldsymbol{r}}_{:,1} - \omega\widehat{\boldsymbol{r}}_{:,2}$
20:         $\widehat{\boldsymbol{u}}_{:,1} \leftarrow \widehat{\boldsymbol{u}}_{:,1} - \omega\widehat{\boldsymbol{u}}_{:,2}$
21:         $\zeta \leftarrow \|\widehat{\boldsymbol{r}}_{:,1}\|$
22:     end while
23:     $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{x}'$ return $\boldsymbol{x}, \zeta/\|\boldsymbol{b}\|$
24: end function
```

6.2.1 Choice of shadow residual

There is quite some freedom in the implementation of BiCGStab, one example is the choice for $\widetilde{\boldsymbol{r}}$ in Algorithm 3. This vector of the BiCGStab algorithm is the so-called "shadow residual" and can be chosen almost arbitrarily. By default, $\widetilde{\boldsymbol{r}}$ is chosen as the first residual $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$. However, as will be demonstrated for several model systems, the specific choice of $\widetilde{\boldsymbol{r}}$ can strongly influence the convergence of BiCGStab.

An interesting alternative choice for $\widetilde{\boldsymbol{r}}$ stems from the IDR(S) algorithm in [46], i.e., choosing $\widetilde{\boldsymbol{r}}$ random and *complex*. To illustrate this, consider linear systems involving a matrix with eigenvalues that have a large imaginary part compared to the real part. For such systems BiCGStab breaks down as a result of $|\omega| \ll 1$ in line 17. Such systems were one of the main motivations to develop BiCGStab(L) [45]. The simplest system for which BiCGStab will stagnate due to

$\omega = 0$ is a system involving a $\pi/2$-rotation matrix;

$$\boldsymbol{A} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{6.1}$$

The matrix in (6.1) has purely imaginary eigenvalues $\pm i$. It is straightforward to show that for any real-valued $\widehat{\boldsymbol{r}}_{:,1}$, the operation $\widehat{\boldsymbol{r}}_{:,2} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{r}}_{:,1}$ results in $\langle \widehat{\boldsymbol{r}}_{:,1}, \widehat{\boldsymbol{r}}_{:,2} \rangle = 0$, since $\widehat{\boldsymbol{r}}_{:,1} \perp \widehat{\boldsymbol{r}}_{:,2}$ and thus $\omega = 0$. One way to avoid this is to use a version of BiCGStab with higher-order stabilizing polynomials, as is the idea behind BiCGStab(L) with $L > 1$. Alternatively, note that this issue does not arise in complex arithmetic. This is achieved by choosing $\widetilde{\boldsymbol{r}}$ random and complex, since this will also result in a complex-valued $\widehat{\boldsymbol{r}}_{:,1}$. Consequently, as the inner product to determine ω also involves complex conjugation, it can be shown that $\langle \widehat{\boldsymbol{r}}_{:,1}, \widehat{\boldsymbol{r}}_{:,2} \rangle$ is purely imaginary. In this case BiCGStab does most likely not break down, however, choosing complex $\widetilde{\boldsymbol{r}}$ comes at the cost of replacing real by complex arithmetic.

But, there also exist systems with *real* eigenvalues for which the default choice of $\widetilde{\boldsymbol{r}}$ will cause BiCGStab to break down. Surprisingly, this occurs for the following system

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{6.2}$$

It can be verified that the default choice of $\widetilde{\boldsymbol{r}}$ leads to $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{u}}_{:,2} \rangle = 0$ and the algorithm stalls while not having updated $\boldsymbol{x}$ even once. The idea of choosing a random $\widetilde{\boldsymbol{r}}$ is to avoid any correlation between the generated residuals $\widehat{\boldsymbol{r}}$ and search directions $\widehat{\boldsymbol{u}}$, making it exceedingly unlikely for BiCGStab to break down or stagnate due to any of these quantities having a vanishingly small inner product with $\widetilde{\boldsymbol{r}}$. This can be achieved by choosing every element of $\widetilde{\boldsymbol{r}}$ in the interval $(0, 1)$ from a uniform random distribution, i.e., $\widetilde{r}_i \in U(0, 1)$.

Inspired by the system presented in Example 3.3 of [117], another system for which BiCGStab will break down, and which can be avoided by a different choice of $\widetilde{\boldsymbol{r}}$ reads

$$\boldsymbol{A} = \begin{bmatrix} \lambda & 0 & 0 \\ -\lambda & \lambda & 0 \\ 0 & -\lambda & \lambda \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \lambda > 0, \tag{6.3}$$

where again all eigenvalues are real, and are equal to λ. This system can be seen as the discretization with the upwind scheme of a one-dimensional problem on a grid with only a few grid cells. Note that for problems involving strong advection the exponential scheme reduces to the upwind discretization scheme. Note that each iteration the boundary value on the left is propagated one grid cell. For the default choice of $\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0$ this is fatal, since $\boldsymbol{r}_0$ only has the first element nonzero, however, the later residuals produced have a vanishing first element, leading to $\rho_1 = \langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle = 0$ on line 9 in Algorithm 3. Because of this, combined with the property that $\boldsymbol{A}\boldsymbol{e}_i = \lambda(\boldsymbol{e}_i - \boldsymbol{e}_{i+1})$ for $i < 3$, α on line 13 becomes undefined due to a zero by zero division. A straightforward calculation shows that α becomes undefined in the second iteration of Algorithm 3. It is shown in Section 6.2.4 that $\langle \boldsymbol{r}_0, \boldsymbol{r}_k \rangle$ decreases exponentially for the LMR method; it is also observed later in Section 6.4.4 that a similar phenomenon occurs for BiCGStab and LMR when applied to discretization matrices obtained from two-dimensional ADR problems.

The idea of choosing a random $\widetilde{\boldsymbol{r}}$ is inspired by several sources. First, the effect of choosing a random initial guess (and thus a random $\widetilde{\boldsymbol{r}}$) was shown in [118] for BiCG algorithms to significantly impact which model problems could be solved. Second, similar to BiCGStab, IDR(S) uses an S-dimensional shadow space. In [46] in the context of the IDR(S) algorithm it was noted that choosing *all* S vectors at random is essential for robustness. Note that IDR(1) is equivalent to BiCGStab. Finally, it was shown in two numerical experiments of acoustics problems that choosing a random $\widetilde{\boldsymbol{r}}$ improved convergence [119] for CGS, and in [120] for both CGS and BiCGStab. Here CGS is the well-known Conjugate Gradient Squared method, a derivative of BiCG that does not require operations with $\boldsymbol{A}^{\mathrm{T}}$.

6.2.2 Reliable update scheme

Note that in baseline BiCGStab, the residual $\boldsymbol{r} := \widehat{\boldsymbol{r}}_{:,1}$ is only directly computed from the definition $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ at the initialization step in line 3. For all subsequent iterations both $\boldsymbol{r}$ and $\boldsymbol{x}$ are computed recursively from the results of previous iterations. This opens an avenue through which finite precision arithmetic can lead to discrepancies between the true residual $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ and the recursively computed residual $\boldsymbol{r}$. This discrepancy between the true residual and the recursively computed residual is called the *residual gap*.

Assuming the only error is introduced by k updates of the form

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \boldsymbol{w}_{j+1}, \quad \boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \boldsymbol{A}\boldsymbol{w}_{j+1}, \tag{6.4}$$

it is shown in [109] that the maximum residual gap due to accumulating rounding errors is bounded by

$$\big| \|\boldsymbol{r}_k\| - \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\| \big| \leq 2k n_{\mathrm{A}} \epsilon_{\mathrm{m}} \| |\boldsymbol{A}| \| \|\boldsymbol{A}^{-1}\| \max_{j \leq k} \|\boldsymbol{r}_j\|. \tag{6.5}$$

Here n_A is the maximum number of nonzeros per row and ϵ_{m} the machine precision, $\| |\boldsymbol{A}| \|$ is the 2-norm of the element-wise absolute value of $\boldsymbol{A}$. An example of a problematic system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is given by

$$\boldsymbol{A} = \begin{bmatrix} 1 & -\gamma & 0 \\ -\gamma & 1 & -\gamma \\ 0 & -\gamma & 1 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}. \tag{6.6}$$

For this system it can be shown, in particular if $\gamma \geq \sqrt{2}$, that

$$\| |\boldsymbol{A}| \| = 1 + \sqrt{2}\gamma, \quad \|\boldsymbol{A}^{-1}\| = 1, \tag{6.7}$$

where the matrix-norms are the 2-norm. After the first update to the residual we obtain

$$\widehat{\boldsymbol{r}}_{:,1} = \begin{bmatrix} 0 \\ 2\gamma \\ 0 \end{bmatrix}, \tag{6.8}$$

therefore the term $\max_{j \leq k} \|\boldsymbol{r}_j\|$ in (6.5) is at least 2γ. Using this result it follows that a conservative estimate for the right-hand side in (6.5) is γ^2, more specifically it scales as $\Omega(\gamma^2)$ in the limit $\gamma \to \infty$ for this system. Therefore, depending on γ the upper bound for the residual gap can be arbitrarily large.

By computing the true residual from the definition $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ the gap is reduced to zero. However, this is expensive as it costs an additional Matrix-Vector product (MV). In [116] it is suggested to compute the true residual if one of the following conditions holds:

1. $\|\widehat{\boldsymbol{r}}_{:,1}\| < 0.01\|\boldsymbol{b}\|$ and $\|\boldsymbol{b}\| \leq \max_j \|\widehat{\boldsymbol{r}}^j_{:,1}\|$,
2. $\|\boldsymbol{b}\| \leq 0.01 \max_j \|\widehat{\boldsymbol{r}}^j_{:,1}\|$ and $\|\widehat{\boldsymbol{r}}_{:,1}\| < \max_j \|\widehat{\boldsymbol{r}}^j_{:,1}\|$,

where the maximum is taken over all residuals since the last computation of the true residual. By computing the true residual the residual gap can be significantly decreased with only a few extra MV. Similarly, $\boldsymbol{x}$ is updated in a “group wise” manner, collecting all the updates into a temporary vector then applying several updates at once to $\boldsymbol{x}$ in order to minimize the effects of round-off errors. For more details regarding the derivation of the reliable updating strategy and the precise conditions for the groupwise updates, the reader is referred to [109, 116].

An alternative strategy is applied in MATLAB’s implementation of BiCGStab. If $\|\boldsymbol{r}\|/\|\boldsymbol{b}\|$ has decreased below the specified tolerance, the recursive residual is replaced by the true residual. This has the benefit of ensuring the residual is accurate, however, replacing the recursively computed residual by the true residual can destroy the BiCG process [109]. We will indeed show in numerical experiments that MATLAB’s implementation can break down after this residual replacement.

6.2.3 Keeping the lowest residual solution

A final idea is applied in MATLAB's variant of BiCGStab [39]. One major difference between the baseline and MATLAB's variant of BiCGStab is keeping track of the $\boldsymbol{x}$ with the lowest residual. After every update of $\boldsymbol{x}$ and $\widehat{\boldsymbol{r}}_{:,1}$ the new $\boldsymbol{x}$ is stored separately until a solution is computed with an even lower residual. This idea has its merits, since the convergence of BiCGStab is in general not monotonic. If the method breaks down, the solution $\boldsymbol{x}$ computed last may be worse than a previously computed solution. Nevertheless, it is important to keep in mind that, due to the residual gap, a previously computed $\boldsymbol{x}$ may not actually have the smallest true residual, even though it could have the smallest recursively computed residual as pointed out in Section 6.2.2. The effects of this modification are discussed in more detail in Section 6.4.2. In the next section we analyze the way the initial residual of the LMR method changes with each iteration.

6.2.4 Propagation of the initial residual

To illustrate how the initial residual is affected by the BiCGStab process we investigate a simpler Krylov subspace method. The LMR algorithm as presented in [43] is given in Algorithm 4. In essence this method makes a step in the direction of the residual, and determines a step size such that the norm of the next residual is minimal. This method can also be interpreted as GMRES(1), or the second half of BiCGStab (Line 16-19 in baseline BiCGStab).

Algorithm 4 LMR algorithm

```
function LMR($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol$)
    $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
    while $\|\boldsymbol{r}\|_2 > tol$ do
        $\omega \leftarrow \frac{\langle \boldsymbol{A}\boldsymbol{r}, \boldsymbol{r} \rangle}{\langle \boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}\boldsymbol{r} \rangle}$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + \omega \boldsymbol{r}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - \omega \boldsymbol{A}\boldsymbol{r}$
    end while
    return $\boldsymbol{x}$
end function
```

To illustrate the LMR process, we consider a system of arbitrary size similar to the example shown in (6.3) with $\lambda = 1$. It is shown in Appendix 6.A that for a linear system involving a bidiagonal Toeplitz matrix $\boldsymbol{A}$ of size $N \times N$, with -1 on the first sub-diagonal and 1 on the diagonal the residual propagates toward the right. Such a bidiagonal Toeplitz matrix is a model for a one-dimensional advection equation with an upwind discretization. More specifically, given a right-hand side $\boldsymbol{e}_1$ and initial guess $\boldsymbol{x} = \boldsymbol{0}$ the k-th LMR residual has elements that follow a binomial distribution;

$$r_i^k = \binom{k}{i-1}(\tfrac{1}{2})^k, \quad i = 1, 2, ..., k+1 < N, \tag{6.9}$$

with $\boldsymbol{r}^0 = \boldsymbol{e}_1$. Importantly, note that

$$\langle \boldsymbol{r}^0, \boldsymbol{r}^k \rangle = \frac{1}{2^k}, \tag{6.10}$$

i.e., the inner product of the first and the k-th residual decreases *exponentially* with k.

Another interpretation of the LMR algorithm exists if $\boldsymbol{A}$ is positive definite, since for such matrices $\omega > 0$. Line 6 of Algorithm 4 can then be written as

$$\frac{\boldsymbol{r}^{k+1} - \boldsymbol{r}^k}{\omega} = -\boldsymbol{A}\boldsymbol{r}^k, \tag{6.11}$$

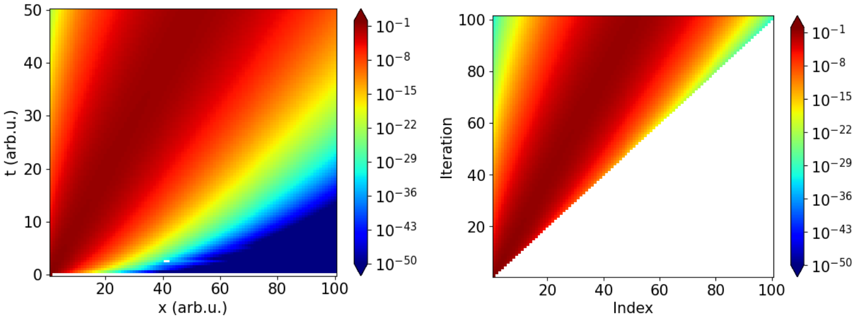
which is the forward Euler discretization with step size ω of the ODE system

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = -\boldsymbol{A}\boldsymbol{r}. \tag{6.12}$$

The LMR algorithm can be seen as time integration method for a linear ODE system. For ADR problems the matrix $\boldsymbol{A}$ represents a discretized ADR operator, and $\boldsymbol{r}$ a discrete approximation to a boundary value problem with homogeneous Dirichlet boundary conditions. The exact solution to (6.12) is compared with the LMR residual in Figure 6.2.1 for the model upwind discretization resulting in a bidiagonal Toeplitz matrix of size 100. It is shown in Appendix 6.B that the discretization matrices obtained later in this chapter are indeed positive definite. From Figure 6.2.1 it can be concluded that the residual starts primarily on the left and propagates toward the interior of the domain. Additionally, it can be seen that the LMR residual indeed resembles the exact solution of the ODE system.

6.3 The discrete advection-diffusion-reaction equation

To investigate the performance of baseline BiCGStab and the modifications presented in Section 6.2, we introduce the advection-diffusion-reaction equation as a model problem. First, the scalar advection-diffusion-reaction equation is converted to dimensionless form, to investigate the iterative solvers for the entire parameter space. Second, the discretization scheme is outlined to obtain a set of difference equations, yielding a linear system. Third, the eigenvalues of the resulting linear system are obtained to show that the obtained matrix has real, positive eigenvalues.



(a) Exact solution of (6.12) (b) LMR residual

Figure 6.2.1: Comparison of the exact solution of the time dependent ODE system (6.12) and the LMR solution (forward Euler approximation) for the bidiagonal Toeplitz matrix of size 100. Note that for the first 100 iterations $\omega = \frac{1}{2}$. Importantly, this shows that the residual is advected throughout the domain, one grid cell per iteration.

6.3.1 Dimensionless form of the ADR equation

The scalar advection-diffusion-reaction equation, used here as a test problem, is relevant for a wide variety of physical systems, describing for example Fickian diffusion in a binary mixture or describing a temperature distribution. Starting from the general ADR equation given by Equation (2.13) in reference [26], with a linearized source term the equation reads

$$\frac{\partial \xi \varphi}{\partial t} + \nabla \cdot (\vec{u}\varphi - \epsilon \nabla \varphi) = s_{\mathrm{C}} - s_{\mathrm{P}}\varphi. \tag{6.13}$$

Here $\vec{u}$ is the velocity field, $\epsilon > 0$ a diffusion coefficient, s_{C} and $s_{\mathrm{P}}\varphi$ a constant and linear reaction term, respectively, and φ the quantity of interest. For clarity $\vec{u}$, ϵ, s_{C} and s_{P} are taken to be constant, furthermore $s_{\mathrm{C}}, s_{\mathrm{P}} > 0$. In absence of transport, $s_{\mathrm{C}} - s_{\mathrm{P}}\varphi_{\mathrm{eq}} = 0$ where φ_{eq} is the (chemical) equilibrium value of φ. Then after the substituting $\varphi_{\mathrm{eq}} := s_{\mathrm{C}}/s_{\mathrm{P}}$, the resulting model equation is given by

$$\frac{\partial \xi \varphi}{\partial t} + \nabla \cdot (\vec{u}\varphi - \epsilon \nabla \varphi) = -s_{\mathrm{P}}(\varphi - \varphi_{\mathrm{eq}}). \tag{6.14}$$

The coefficient $\xi > 0$ denotes the responsiveness of the system; for small ξ the solution can vary more rapidly compared to large values of ξ. Physically ξ can, for example, take the role of a specific heat capacity or mass density.

Consider a Cartesian three-dimensional coordinate system, then Equation

(6.14) can be expanded as follows

$$\frac{\partial \xi\varphi}{\partial t} + \sum_{i=1}^{3} \frac{\partial}{\partial x_i}\left(u_i\varphi - \epsilon\frac{\partial \varphi}{\partial x_i}\right) = -s_{\mathrm{P}}(\varphi - \varphi_{\mathrm{eq}}), \tag{6.15}$$

with x_i the i-th Cartesian coordinate, and u_i the corresponding component of the advection velocity. Next, we aim to make (6.15) dimensionless. To commence, a scaling is performed; we scale the time t by a characteristic timescale T, and the coordinates x_i are scaled by the length scales L_i for the i-th direction;

$$t^* := t/T, \quad x_i^* := (x_i - x_{i,\min})/L_i. \tag{6.16}$$

The values of the characteristic variables are to be determined later. Using the newly introduced quantities T and L_i, the partial derivatives can be written as

$$\frac{\partial}{\partial t} = \frac{1}{T}\frac{\partial}{\partial t^*}, \quad \frac{\partial}{\partial x_i} = \frac{1}{L_i}\frac{\partial}{\partial x_i^*}. \tag{6.17}$$

Assuming constant ξ, and combining with the differentiation rules from (6.17) Equation (6.15) can be simplified to

$$\frac{\xi}{T}\frac{\partial \varphi}{\partial t^*} + \epsilon\sum_{i=1}^{3} \frac{1}{L_i^2}\frac{\partial}{\partial x_i^*}\left(\frac{u_iL_i}{\epsilon}\varphi - \frac{\partial \varphi}{\partial x_i^*}\right) = -s_{\mathrm{P}}(\varphi - \varphi_{\mathrm{eq}}). \tag{6.18}$$

Furthermore, by introducing Péclet numbers for each of the Cartesian directions

$$\mathrm{Pe}_i := \frac{u_iL_i}{\epsilon}, \tag{6.19}$$

we can simplify the notation. These Péclet numbers can be interpreted as the relative strength of advection in the direction $\vec{e}_i$ compared to diffusion. After some algebraic manipulations the advection-diffusion-reaction equation becomes

$$\frac{\partial \varphi}{\partial t^*} + \frac{T\epsilon}{\xi}\sum_{i=1}^{3} \frac{1}{L_i^2}\frac{\partial}{\partial x_i^*}\left(\mathrm{Pe}_i\varphi - \frac{\partial \varphi}{\partial x_i^*}\right) = -s_{\mathrm{P}}(\varphi - \varphi_{\mathrm{eq}})\frac{T}{\xi}. \tag{6.20}$$

We introduce the transport frequencies $\nu_{\mathrm{t},i}$ and the reaction frequency ν_{r}, defined as

$$\nu_{\mathrm{t},i} := \frac{\epsilon}{\xi L_i^2}, \quad \nu_{\mathrm{r}} := \frac{s_{\mathrm{P}}}{\xi}. \tag{6.21}$$

We then obtain the advection-diffusion-reaction equation in terms of frequencies $\nu_{\mathrm{t},i}$ and ν_{r} as;

$$\frac{\partial \varphi}{\partial t^*} + T\sum_{i=1}^{3} \nu_{\mathrm{t},i}\frac{\partial}{\partial x_i^*}\left(\mathrm{Pe}_i\varphi - \frac{\partial \varphi}{\partial x_i^*}\right) = -\nu_{\mathrm{r}}T(\varphi - \varphi_{\mathrm{eq}}). \tag{6.22}$$

Next, by taking T such that $\nu_\mathrm{r} T = 1$ the Damköhler numbers Da_i can be introduced as follows,

$$\nu_{\mathrm{t},i} T = \frac{\nu_{\mathrm{t},i}}{\nu_\mathrm{r}} =: \frac{1}{\mathrm{Da}_i}, \quad \mathrm{Da}_i = \frac{s_\mathrm{P} L_i^2}{\epsilon}. \tag{6.23}$$

Finally, the dimensionless time-dependent advection-diffusion-reaction equation is obtained by dividing both sides by a reference value φ_ref, such that $\varphi^* := \varphi/\varphi_\mathrm{ref}$, which results in

$$\frac{\partial \varphi^*}{\partial t^*} + \sum_{i=1}^{3} \frac{1}{\mathrm{Da}_i} \frac{\partial}{\partial x_i^*} \left(\mathrm{Pe}_i \varphi^* - \frac{\partial \varphi^*}{\partial x_i^*} \right) = -(\varphi^* - \varphi_\mathrm{eq}^*). \tag{6.24}$$

If the characteristic length scales $L_i = L$ are the same in each of the Cartesian directions, and assuming a stationary solution, Equation (6.24) reduces to

$$\sum_{i=1}^{3} \frac{\partial}{\partial x_i^*} \left(\mathrm{Pe}_i \varphi^* - \frac{\partial \varphi^*}{\partial x_i^*} \right) = -\mathrm{Da}(\varphi^* - \varphi_\mathrm{eq}^*), \tag{6.25}$$

where now all Damköhler numbers are the same; Da. For ease of notation, * is omitted in the following sections. In the next section we discretize Equation (6.25) and obtain a linear system which can be used to approximate the exact solution of (6.25).

6.3.2 Discretization

6.3.2.1 One-dimensional scheme

To discretize (6.25), for ease of exposition, we start with the one-dimensional equivalent with $\varphi_\mathrm{eq} = 0$, since φ_eq will only affect the right-hand side of the resulting linear system. The ADR-equation can then be written as

$$\frac{\mathrm{d}}{\mathrm{d}x} \mathit{\Gamma}(\varphi(x)) = -\mathrm{Da}\,\varphi(x), \tag{6.26a}$$

where $\mathit{\Gamma}$ is the flux, given by

$$\mathit{\Gamma}(\varphi) := \mathrm{Pe}\,\varphi - \frac{\mathrm{d}\varphi}{\mathrm{d}x}. \tag{6.26b}$$

To discretize (6.26a), we use the Finite Volume Method (FVM). In the FVM method the domain is subdivided into a finite number of disjunct intervals; referred to as control volumes. Here a cell-centered approach is applied on a uniform grid; in such a configuration φ has to be computed at the nodal points x_i. Control volumes are defined around these nodal points; the i-th control volume extends over $[x_{i-1/2}, x_{i+1/2}]$, where $x_{i\pm 1/2} := \frac{1}{2}(x_i + x_{i\pm 1})$ and the width of this volume is defined as the grid size $\Delta x := x_{i+1/2} - x_{i-1/2}$.

Previously, the Péclet and Damköhler numbers were defined with respect to a length scale L, in the following we take the length of each grid cell as the respective length scale, that is;

$$L = \Delta x =: h, \tag{6.27}$$

defining them as the grid Péclet and Damköhler numbers. Integrating (6.26a) over a control volume and approximating the integral over the source term with the midpoint rule we obtain the discrete conservation law for each interval;

$$F_{i+1/2} - F_{i-1/2} = -h\mathrm{Da}\,\varphi_i, \tag{6.28}$$

with φ_i a numerical approximation of $\varphi(x_i)$. Several expressions for the numerical flux $F_{i+1/2}$ at $x_{i+1/2}$ exist, depending on the discretization scheme used. Here we use the exponential flux given in [26, p. 86];

$$F_{i+1/2} = \frac{1}{h}(\mathrm{B}(-\mathrm{Pe})\varphi_i - \mathrm{B}(\mathrm{Pe})\varphi_{i+1}), \tag{6.29}$$

where $\mathrm{B}(z)$ is the generating function for the Bernoulli numbers; in short the Bernoulli function, defined as [121, p. 40][122, p. 804]

$$\mathrm{B}(z) := \begin{cases} \frac{z}{\exp(z)-1} & z \neq 0, \\ 1 & z = 0. \end{cases} \tag{6.30}$$

The resulting exponential scheme is obtained from the discrete conservation law (6.28) resulting in

$$\frac{1}{h}(-\mathrm{B}(\mathrm{Pe})\varphi_{i+1} + (\mathrm{B}(-\mathrm{Pe}) + \mathrm{B}(\mathrm{Pe}))\varphi_i - \mathrm{B}(-\mathrm{Pe})\varphi_{i-1}) = -h\mathrm{Da}\,\varphi_i. \tag{6.31}$$

Assuming Dirichlet boundary conditions, the discretization matrix without linear source term, is given by a tridiagonal Toeplitz matrix. For the exponential scheme the following discretization matrix is obtained;

$$\boldsymbol{A}_x = \frac{1}{h}\mathrm{tridiag}(-\mathrm{B}(-\mathrm{Pe}), \mathrm{B}(-\mathrm{Pe}) + \mathrm{B}(\mathrm{Pe}), -\mathrm{B}(\mathrm{Pe})), \tag{6.32}$$

with $\mathrm{tridiag}(a, b, c)$ indicating a tridiagonal Toeplitz matrix with a on the sub-diagonal, b on the diagonal and c on the super-diagonal. Note that for strong positive advection ($\mathrm{Pe} \to \infty$) the matrix $\boldsymbol{A}_x$ becomes bi-diagonal, similar to the small model system shown in Equation (6.3), and the problem discussed in Section 6.2.4. Since this is a tridiagonal Toeplitz matrix a closed form expression for the eigenvalues is known [123], namely, they are given by

$$\begin{aligned} \lambda_k(\boldsymbol{A}_x) = \frac{1}{h}\Big(\mathrm{B}(-\mathrm{Pe}) + \mathrm{B}(\mathrm{Pe}) + 2\sqrt{\mathrm{B}(\mathrm{Pe})\mathrm{B}(-\mathrm{Pe})}\cos\Big(\frac{k\pi}{M-1}\Big)\Big), \\ k = 1, 2, ..., M-2, \end{aligned} \tag{6.33}$$

with M the number of grid points. Note that since $\mathrm{B}(\mathrm{Pe}) > 0$ and the cosine factor is greater than -1, we can obtain a strict lower bound, as

$$\begin{aligned} \lambda_k &> \frac{1}{h}\Big(\mathrm{B}(-\mathrm{Pe}) + \mathrm{B}(\mathrm{Pe}) - 2\sqrt{\mathrm{B}(\mathrm{Pe})\mathrm{B}(-\mathrm{Pe})}\Big) \\ &= \frac{1}{h}\Big(\sqrt{\mathrm{B}(-\mathrm{Pe})} - \sqrt{\mathrm{B}(\mathrm{Pe})}\Big)^2 \geq 0. \end{aligned} \tag{6.34}$$

6.3.2.2 Three-dimensional scheme

To extend the discussion of Section 6.3.2.1 to three-dimensional problems, the Kronecker sum is used. The Kronecker sum $\oplus$, is given by [124, p. 268]

$$\boldsymbol{P} \oplus \boldsymbol{Q} \coloneqq \boldsymbol{I}_n \otimes \boldsymbol{P} + \boldsymbol{Q} \otimes \boldsymbol{I}_m, \tag{6.35}$$

with $\boldsymbol{P} \in \mathbb{R}^{m\times m}$, $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{I}_m$, $\boldsymbol{I}_n$ the identity matrices of size $m \times m$ and $n \times n$, respectively. Here $\otimes$ is the well-known Kronecker product (tensor product) defined for two matrices $\boldsymbol{P} \in \mathbb{R}^{N\times N}$ and $\boldsymbol{Q}$ as

$$\boldsymbol{P} \otimes \boldsymbol{Q} = \begin{bmatrix} p_{1,1}\boldsymbol{Q} & \cdots & p_{1,N}\boldsymbol{Q} \\ \vdots & \ddots & \vdots \\ p_{N,1}\boldsymbol{Q} & \cdots & p_{N,N}\boldsymbol{Q} \end{bmatrix}. \tag{6.36}$$

It is then possible to construct the matrix $\boldsymbol{A}_{\mathrm{3D}}$ from three one-dimensional discretization matrices using the Kronecker sum, similar to the procedure in [125].

For a three-dimensional problem on a cube of sides $(M-1)h$ with constant coefficients and cubical cells with sides of length h, the discretization matrix excluding the source term can be written as

$$\boldsymbol{A}_{\mathrm{3D}} = (\boldsymbol{A}_x \oplus \boldsymbol{A}_y \oplus \boldsymbol{A}_z)h^2, \tag{6.37}$$

where $\boldsymbol{A}_x$, $\boldsymbol{A}_y$ and $\boldsymbol{A}_z$ are the discretization matrices corresponding to the one-dimensional problem for each direction as laid out in Section 6.3.2.1. Applying (6.35) twice, and using the associative property of the Kronecker sum it can be shown that

$$\boldsymbol{A}_x \oplus \boldsymbol{A}_y \oplus \boldsymbol{A}_z = \boldsymbol{I} \otimes \boldsymbol{I} \otimes \boldsymbol{A}_x + \boldsymbol{I} \otimes \boldsymbol{A}_y \otimes \boldsymbol{I} + \boldsymbol{A}_z \otimes \boldsymbol{I} \otimes \boldsymbol{I}, \tag{6.38}$$

where each of the identity matrices is of size $M-2$, the number of grid cells in each direction.

Next, we show how the eigenvalues of $\boldsymbol{A}_{\mathrm{3D}}$ can be obtained. Consider a vector $\boldsymbol{v}$ as right-eigenvector of $\boldsymbol{P}$ with eigenvalue λ and $\boldsymbol{w}$ a right-eigenvector of

$\boldsymbol{Q}$ with eigenvalue μ, then by using the mixed-product property of tensor products it follows that [124, p. 244]

$$\begin{aligned}(\boldsymbol{I}_n \otimes \boldsymbol{P} + \boldsymbol{Q} \otimes \boldsymbol{I}_m)(\boldsymbol{w} \otimes \boldsymbol{v}) &= \boldsymbol{I}_n \boldsymbol{w} \otimes \boldsymbol{P}\boldsymbol{v} + \boldsymbol{Q}\boldsymbol{w} \otimes \boldsymbol{I}_m \boldsymbol{v} \\ &= \boldsymbol{w} \otimes \lambda \boldsymbol{v} + \mu \boldsymbol{w} \otimes \boldsymbol{v} \\ &= (\lambda + \mu)(\boldsymbol{w} \otimes \boldsymbol{v}).\end{aligned} \tag{6.39}$$

Thus $\boldsymbol{w} \otimes \boldsymbol{v}$ is an eigenvector of $\boldsymbol{P} \oplus \boldsymbol{Q}$ with eigenvalue $\lambda + \mu$, conform Theorem 4.4.5 in [124, p. 268]. Consequently, if λ_x is an eigenvalue of $\boldsymbol{A}_x$, λ_y an eigenvalue of $\boldsymbol{A}_y$ and λ_z an eigenvalue of $\boldsymbol{A}_z$, then $\lambda_x + \lambda_y + \lambda_z$ is an eigenvalue of $\boldsymbol{A}_{3\mathrm{D}}$, resulting in an expression for the eigenvalues of $\boldsymbol{A}_{3\mathrm{D}}$;

$$\lambda_{(i,j,k)}(\boldsymbol{A}_{3\mathrm{D}}) = (\lambda_i(\boldsymbol{A}_x) + \lambda_j(\boldsymbol{A}_y) + \lambda_k(\boldsymbol{A}_z))h^2, \quad i, j, k = 1, 2, ..., M - 2. \tag{6.40}$$

Finally, the effect of the linear source term has to be taken into account, namely, a shift of the eigenvalues by $h^3\mathrm{Da}$ resulting in the final set of eigenvalues for the discretization matrix;

$$\lambda_{(i,j,k)}(\boldsymbol{A}) = (\lambda_i(\boldsymbol{A}_x) + \lambda_j(\boldsymbol{A}_y) + \lambda_k(\boldsymbol{A}_z))h^2 + h^3\mathrm{Da}, \quad i, j, k = 1, 2, ..., M - 2. \tag{6.41}$$

This shows that all eigenvalues of $\boldsymbol{A}$ are positive and real for $\mathrm{Da} \geq 0$. As a result, for all values of Pe and Da which will be used in the numerical experiments later, the linear system is invertible and only has positive, real eigenvalues. Furthermore, in Appendix 6.B it is shown that $\boldsymbol{A}$ is also positive definite.

In [45] an example was shown involving an advection problem where the system matrix following from discretization with the central differencing scheme contains eigenvalues with large imaginary parts. In such a case BiCGStab($L > 1$) is shown to outperform BiCGStab. However, it should also be noted that for problems with strong advection the central difference scheme yields spurious oscillations in the solution. Therefore, in our experiments we consider the exponential scheme which does not result in unphysical behavior for strong advection.

In the numerical experiments of Section 6.4 we illustrate that both of MATLAB's versions of BiCGStab and BiCGStab(2) do not converge for advection dominated problems with the exponential scheme used here. Moreover, we also investigate modifications to the baseline BiCGStab algorithm and show a potential mitigation by using a different choice of $\widetilde{\boldsymbol{r}}$.

6.4 Results and discussion

In this section we investigate the reliability of BiCGStab, IDR and their various implementations. To do this an ADR-equation is discretized in 3D and the true relative residuals $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\|/\|\boldsymbol{b}\|$ are compared over the entire range of Péclet and Damköhler numbers. Next, convergence as function of the number of Matrix-Vector products (MV) is compared for four specific combinations of Péclet and Damköhler numbers. Finally, a numerical experiment is conducted to elaborate on the effect of the residual propagating through the domain, and the effect of choosing a random $\widetilde{\boldsymbol{r}}$ in BiCGStab.

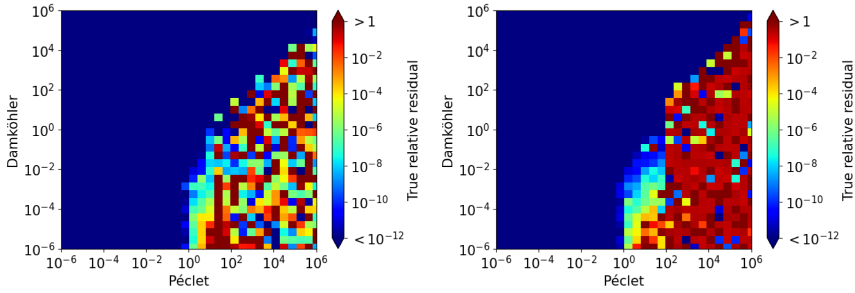
6.4.1 Convergence as function of Péclet and Damköhler numbers

To investigate the convergence of BiCGStab and IDR, a model problem is set up to investigate the entire range of Péclet and Damköhler numbers. We solve (6.25) for φ on the unit cube. To investigate the entire range of Péclet and Damköhler numbers we choose

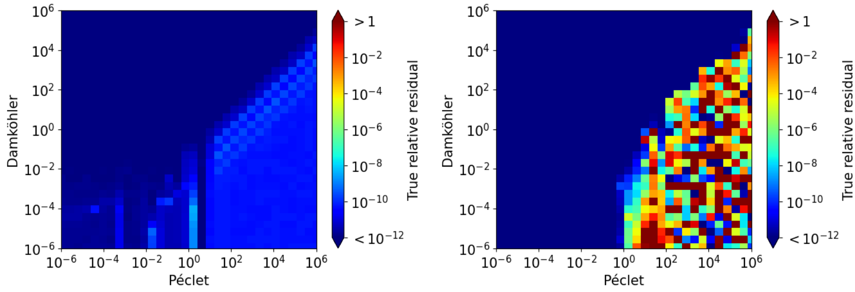
$$\begin{aligned} \vec{u} = \frac{u}{\sqrt{3}}(\vec{e}_x + \vec{e}_y + \vec{e}_z), \quad \epsilon = 1, \quad \varphi_{\mathrm{eq}} = 0, \\ \varphi(0,y,z) = \varphi(x,1,z) = \varphi(x,y,1) = 1, \\ \varphi(1,y,z) = \varphi(x,0,z) = \varphi(x,y,0) = 0, \end{aligned} \tag{6.42}$$

for varying u, and s_{P}. The grid contains $M = 101$ grid cells for each of the three Cartesian directions such that $h = 1/100$. After discretization with the exponential scheme described in Section 6.3.2 a linear system with $99^3 = 970,299$ unknowns is obtained. No preconditioner is used and the initial guess is set to zero. Both Pe and Da use the grid spacing h as the characteristic length scales. The grid Péclet and Damköhler numbers in (6.25) are varied over the range $[10^{-6}, 10^{6}]$.

Each of the iterative solvers is given a maximum of 10^4 MV, and the tolerance is set such that the relative residual $\|\boldsymbol{r}\|/\|\boldsymbol{b}\|$ is smaller than 10^{-12}. We will show that the most efficient solver in terms of MV requires less than 10% of this maximum to converge. I.e., if a solver does not converge, increasing the maximum number of iterations in an attempt to enforce convergence would be inefficient, and may not resolve the issue either. After completion of the iteration we compare the true residual for all four methods in Figure 6.4.1.



(a) MATLAB's BiCGStab (b) MATLAB's BiCGStab(2)



(c) IDR(4) (d) Baseline BiCGStab

Figure 6.4.1: True residual of the boundary value problem given by (6.25) and (6.42). Comparison between several solvers.

It can be seen in Figure 6.4.1a, 6.4.1b and 6.4.1d that none of the BiCGStab variants are successful in solving the discretized advection-diffusion-reaction problem for all Péclet and Damköhler numbers. However, Figure 6.4.1c shows that IDR(4) is close to achieving the prescribed tolerance for a wide range of Péclet and Damköhler numbers.

BiCGStab(2) does not show improved reliability compared to BiCGStab. This is expected based on the eigenvalues, since the eigenvalues of the discretization matrix are real as pointed out in Section 6.3.1. The ability to handle matrices with complex eigenvalues efficiently is one of the main advantages of BiCGStab(2) over BiCGStab. However, for real eigenvalues this advantage is not relevant. As a result there is little difference between the regions where MATLAB's implementations of BiCGStab and BiCGStab(2) converge.

In addition, it can be seen that MATLAB's implementation of BiCGStab performs similarly to the baseline BiCGStab algorithm. This is due to a common issue related to propagation of the initial residual. Both of MATLAB's implementation and the baseline algorithm share this issue, which is discussed in more detail in Section 6.4.4. Most importantly, neither solver converges reliably for problems with dominant advection, especially when only a weak source term is present.

In the next section we show the effects of the modifications presented in Section 6.2 when applied to the baseline BiCGStab algorithm.

6.4.2 Comparing modified solvers

To show the effects of the modifications to BiCGStab as discussed in Section 6.2, we first show a version of BiCGStab with all modifications enabled;

- random shadow residual $\widetilde{\boldsymbol{r}}$,
- reliable updating scheme,
- keeping the best intermediate solution.

The true residual computed from the solutions returned by the modified BiCGStab algorithm with all three modifications *enabled* is shown in Figure 6.4.2a. It can be seen in this figure that for all Péclet and Damköhler numbers the modified version of BiCGStab converges to the prescribed tolerance. Next, we *enable* or *disable* one of the modifications in separate figures to see their effect on the final residual.

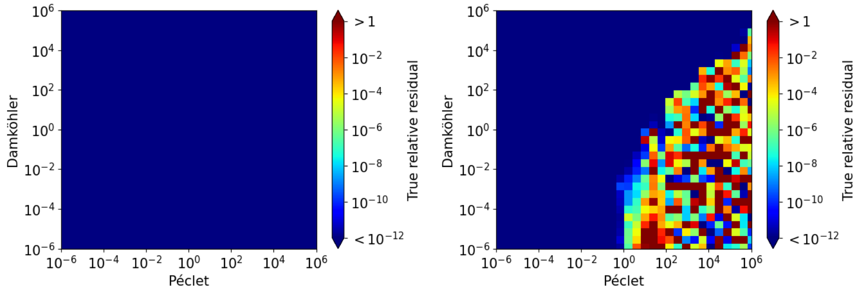
Figure 6.4.2b shows that the effect of storing the solution corresponding to the lowest residual thus far, $\boldsymbol{r}_{\min}$, compared with the baseline algorithm shown in Figure 6.4.1d is negligible. Furthermore, if the solver were to converge, the benefit of this modification is limited as well. If the solver converges, modified BiCGStab always returns the solution computed in the last iteration. Thus, if the recursively computed residual is accurate, and the solver does not stagnate, then keeping track of the best intermediate solution is redundant. However, this may have an unwanted effect if the maximum number of iterations set is small. If there is no improvement over the initial residual, BiCGStab would return the initial guess as a solution.

In Figure 6.4.2c we *disable* the use of a random $\widetilde{\boldsymbol{r}}$, and use the default choice $\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$. Here it can be observed that the choice of $\widetilde{\boldsymbol{r}}$ has a substantial effect on the final residual for the region roughly given by $\mathrm{Pe} > 10^2$ and $\mathrm{Da} < \mathrm{Pe}$. Compared to Figure 6.4.2a it is clear that for these problems the choice of $\widetilde{\boldsymbol{r}}$ plays a significant role.

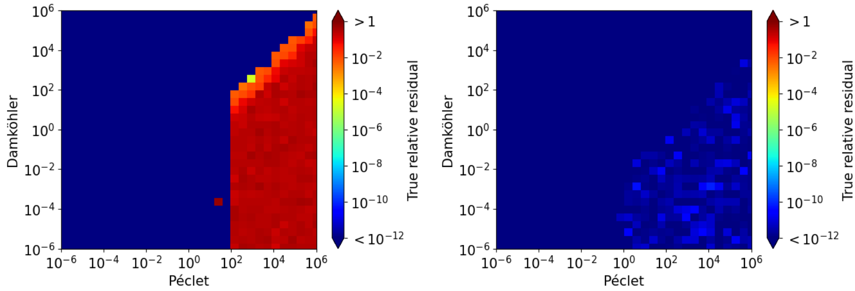
The cause of the convergence issues when using the default choice of $\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0$ is a manifestation of the three-dimensional equivalent of the scenario presented in

Equation (6.3). More specifically, the residual starts near one of the boundaries and propagates in a wave-like manner through the domain. The important consequence of this is that the initial residual only contains nonzero elements near one of the boundaries and gradually these nonzero elements decrease in further iterations. This leads to the issue that $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle \to 0$ and indirectly $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{u}}_{:,2} \rangle \to 0$. Reliable updating offers no improvement here, as the issue occurs even with accurate residuals. This effect is shown in more detail in Section 6.4.4.

The effect of the reliable updating scheme is shown in Figure 6.4.2d. In this figure we show BiCGStab with the modifications of random $\widetilde{\boldsymbol{r}}$ and keeping the best intermediate solution *enabled*, and reliable updating *disabled*. For a wide range of the Péclet and Damköhler numbers the solver converges to the required tolerance of 10^{-12}, except for some cases in the region $\mathrm{Pe} > 1$ and $\mathrm{Da} < 10^3$. Even though the effect of reliable updating is not as dramatic as choosing a different $\widetilde{\boldsymbol{r}}$, a good solver should not stop iterating prematurely. Therefore, the reliable updating scheme is nevertheless a beneficial addition to ensure the required tolerance has truly been achieved.



(a) Modified BiCGStab (b) Baseline BiCGStab with $\boldsymbol{r}_{\min}$.



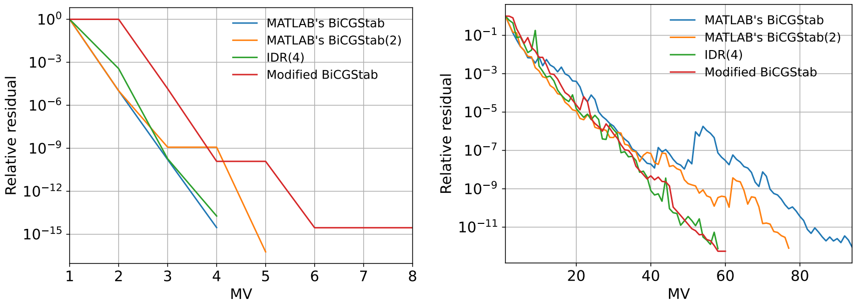
(c) Modified BiCGStab without random $\widetilde{\boldsymbol{r}}$. (d) Modified BiCGStab without reliable update.

Figure 6.4.2: True residual of the system boundary value problem given by (6.25) and (6.42). Comparison between BiCGStab and several modified variants.

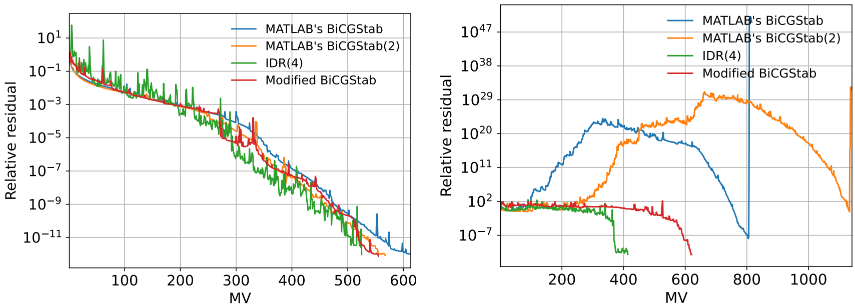
In the next section we investigate the evolution of the residual as function of the number of MV, for four selected combinations of Péclet and Damköhler numbers.

6.4.3 Comparing performance for specific Péclet and Damköhler numbers

The convergence as function of the number of matrix-vector products for four combinations of Péclet and Damköhler numbers is investigated. Figure 6.4.3 shows the relative residual (as computed by the solver) $\|\boldsymbol{r}\|/\|\boldsymbol{b}\|$ as function of number of matrix-vector products. Here the same model problem is used as in the previous sections.



(a) $\mathrm{Pe} = 10^{-5}$, $\mathrm{Da} = 10^{5}$ (b) $\mathrm{Pe} = 10^{5}$, $\mathrm{Da} = 10^{5}$



(c) $\mathrm{Pe} = 10^{-5}$, $\mathrm{Da} = 10^{-5}$ (d) $\mathrm{Pe} = 10^{5}$, $\mathrm{Da} = 10^{-5}$

Figure 6.4.3: Relative residual of the boundary value problem given by (6.25) and (6.42) as function of the number of matrix-vector products (MV). Note that this Figure shows the convergence behavior corresponding to four locations shown in Figures 6.4.1 and 6.4.2.

For large Damköhler and small Péclet numbers it can be seen in Figure 6.4.3a that all methods converge within a few MV. When both the Péclet and Damköhler numbers are large it can be seen in Figure 6.4.3b that the methods perform similarly for the first 40 MV, however, after this point MATLAB's solvers start to slow down. For a problem with dominant diffusion, Figure 6.4.3c, all solvers converge smoothly and perform about the same. However, the number of MV required is much larger than the case with negligible diffusion shown in Figure 6.4.3b.

Figure 6.4.3d highlights one of the points of interest with MATLAB's implementations. First, it can be seen that the residual *increases* significantly above the starting value for MATLAB's solvers. Second, the residual can be seen to sharply increase at approximately 800 MV and 1200 MV for MATLAB's versions of BiCGStab and BiCGStab(L), respectively. The cause of this is that after a certain number of iterations the recursively computed residual does decrease, however, it strongly deviates from the true residual. When MATLAB's implementation detects stagnation, or when the recursively computed residual norm is below the tolerance, the true residual is computed. After this step the recursive residual is replaced by the true residual. However, shortly after this replacement of the recursively computed residual by the true residual, the solvers stagnate.

It can be seen in Figure 6.4.3d that IDR(4) converges in the least number of matrix-vector products. Next, note that for the discretized problem, it takes at least 300 MV to propagate the boundary conditions throughout the domain if every MV only spreads the solution to neighboring cells, similar to what was discussed for LMR in Section 6.2.4. Interestingly, the convergence graph for IDR(4) shows a plateau for the first approximately 300 MV. Similarly, it can be seen in Figure 6.4.3c that there appears to be a speedup in convergence around 300 MV. For Figure 6.4.3a it should be noted that the system matrix is strongly diagonally dominant, nearly becoming a multiple of the identity matrix, causing the fast convergence. Figure 6.4.3b does not show a plateau region, presumably since the initial guess of zeros is already a good approximation throughout the domain.

6.4.4 Propagation of the residual for LMR and BiCGStab

In this section we investigate the hypothesis on the significant impact of replacing the default choice of $\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0$ with a random $\widetilde{\boldsymbol{r}}$. It is conjectured that, similar to the model problem discussed in Section 6.2.4, for 2D and 3D problems the residual propagates into the interior domain, following the advection direction.

To show the effect of the residual propagating through the domain, as was suggested by Equation (6.12), we consider a two-dimensional ADR problem on the unit square with the following parameters;

$$\begin{aligned} \vec{u} = \frac{u}{\sqrt{2}}(\vec{e}_x + \vec{e}_y), \quad \epsilon = 1, \quad \varphi_{\text{eq}} = 0 \\ \varphi(0,y) = \varphi(x,1) = 1, \\ \varphi(1,y) = \varphi(x,0) = 0, \end{aligned} \tag{6.43}$$

where again both u, and s_{P} are varied to investigate specific Péclet and Damköhler numbers. The system is discretized using the exponential scheme using 101 grid points per Cartesian direction. The Péclet number is set to 10^5 and the Damköhler number to 10^{-5}; the advection dominated regime. The resulting linear system is solved using a zero initial guess for both LMR and baseline BiCGStab, modified to include a random $\widetilde{\boldsymbol{r}}$.

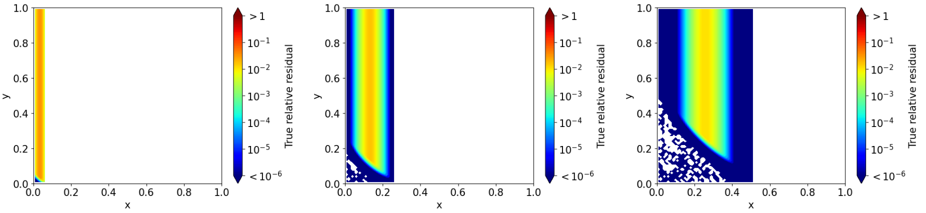
In Figure 6.4.4 two main effects can be seen. First, the nonzero residual is propagating in the x-direction. Second, a wave-like structure traveling in the $\vec{e}_x + \vec{e}_y$ direction. Since in the two-dimensional discretization scheme cells are coupled using a 5-point stencil, every element in the residual vector can affect at most 4 other elements per MV, viz. their direct neighbors. Therefore to have the grid cell representing the residual at $x = 0, y = 0$ affect the grid cell at $x = 1, y = 0$ would take $M - 2$ MV. Similarly, it would take $2(M - 1)$ MV to have the grid cell at $x = 0, y = 0$ affect the one at $x = 1, y = 1$, since the propagation follows a staircase pattern.

It can be seen in Figure 6.4.4 that for LMR the initial residual is nonzero close to $x = 0$ and zero throughout the rest of the domain, similar to the one-dimensional case shown in Figure 6.2.1. For consecutive iterations the residual can be seen to propagate similar to a wave traveling in the $\vec{e}_x + \vec{e}_y$ direction. This is consistent with the interpretation of LMR solving a time-dependent version of the advection equation, as suggested by Equation (6.12).

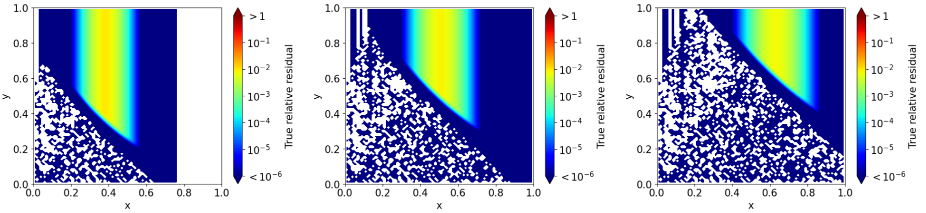
For BiCGStab a similar effect is observed, however, a dispersion-like phenomenon is also present; see Figure 6.4.5. The residual is pushed into the direction $\vec{e}_x + \vec{e}_y$ as well, however, unlike LMR, the wavefront is not as sharp. It can be seen that the residual propagates in a wave-like manner through the domain prone to dispersion. Note that the initial residual is only nonzero on the left-most grid cells.

Importantly, since the residual propagates into the interior domain this results in $\langle \boldsymbol{r}_0, \boldsymbol{r}_k \rangle \to 0$ on line 9 of Algorithm 3. Consequently, $\beta \to 0$, $\widehat{\boldsymbol{u}}_{:,1} \to \widehat{\boldsymbol{r}}_{:,1}$. Additional multiplications by $\boldsymbol{A}$ only shift the residual even more, $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{A}\widehat{\boldsymbol{u}}_{:,1} \rangle \to 0$ and α becomes undetermined on line 13. Note, however, that since BiCGStab performs 2 MV per iteration, one might expect the wave traveling in the $\vec{e}_x + \vec{e}_y$ to have crossed through the entire domain after 50 iterations. This does not seem to be the case, suggesting that only the LMR-part is responsible for the wave propagation.

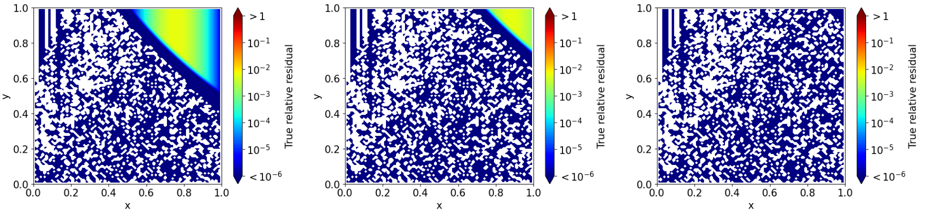
The effect of propagating residuals is the suspected cause of BiCGStab failing to converge without a random $\widetilde{\boldsymbol{r}}$, which can be seen by comparing Figure 6.4.2a and Figure 6.4.2c. Note that this reason is completely different from the issue due to BiCGStab failing for complex eigenvalues which result from central difference discretization. Additionally, the propagating residuals suggest that the LMR method needs $\mathcal{O}(M)$ iterations to get through the first phase of convergence (the plateau in Figure 6.4.6). It is suspected that this effect also relates to the conjecture in [126] which states that the first phase of convergence is determined by the longest streamline for residual-minimizing Krylov subspace methods, in this case $2(M - 2)$ cells, as it takes $2(M - 2)$ iterations for the residual to propagate from $x = 0, y = 0$ to $x = 1, y = 1$, following only the grid cells in the direction of the advection velocity.



(a) 5 iterations (b) 25 iterations (c) 50 iterations

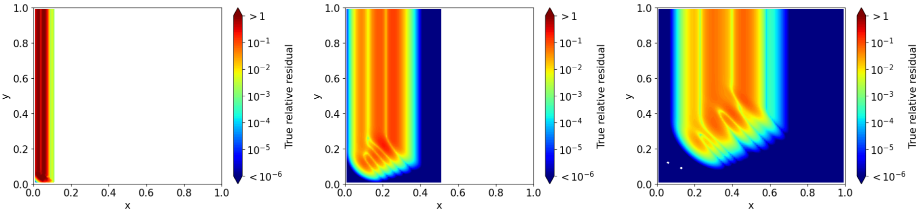


(d) 75 iterations (e) 100 iterations (f) 125 iterations

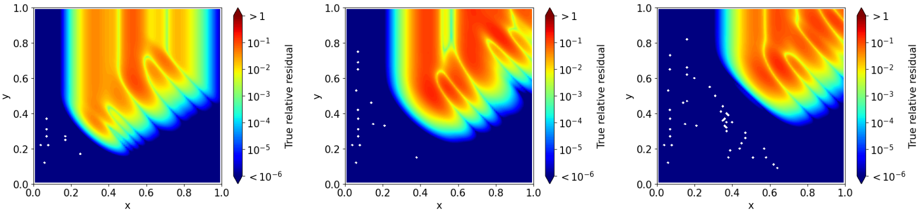


(g) 150 iterations (h) 175 iterations (i) 200 iterations

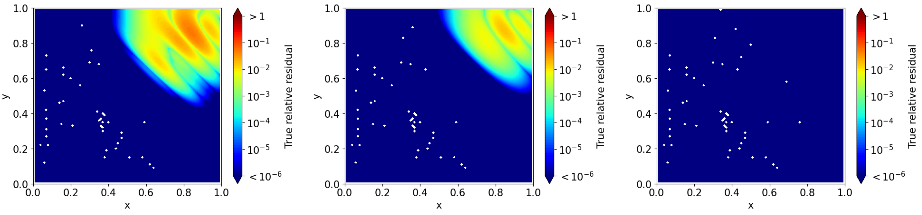
Figure 6.4.4: Relative residuals for LMR plotted in the $x - y$ plane for the model problem given by the parameters in (6.43). White in these figures indicates a residual of exactly 0 at a given position. Here it can be seen that the residual propagates in the direction of $\vec{u}$.



(a) 5 iterations (b) 25 iterations (c) 50 iterations



(d) 75 iterations (e) 100 iterations (f) 125 iterations



(g) 150 iterations (h) 175 iterations (i) 200 iterations

Figure 6.4.5: Relative residuals plotted in the $x - y$ plane for baseline BiCGStab with random $\widetilde{\boldsymbol{r}}$ for the problem described in (6.43) ($\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0$ breaks down). White in these figures indicates a residual of exactly 0 at a given position.

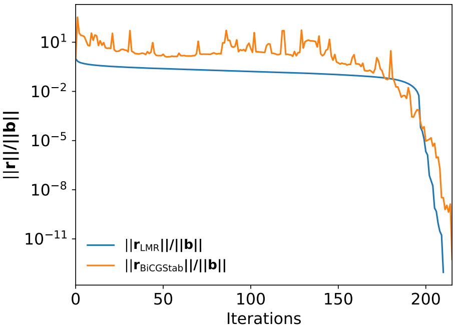


Figure 6.4.6: Relative residuals for LMR and baseline BiCGStab with random $\widetilde{\boldsymbol{r}}$ for the problem described in (6.25) and (6.43) as function of the number of iterations. Note that BiCGStab performs 2 MV per iteration, one in the BiCG part, one in the LMR part.

6.4.5 Preconditioned BiCGStab

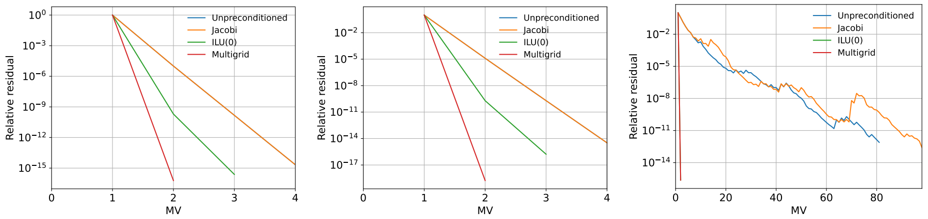
Since linear systems are typically solved using preconditioned iterations, several numerical experiments using MATLAB's BiCGStab, have been performed for a number of preconditioners.

Specifically, we investigate if the use of a preconditioner can alleviate the convergence issues of MATLAB's BiCGStab, displayed in Figure 6.4.1, in the regime of large Péclet and small Damköhler numbers. MATLAB's BiCGStab was chosen as it supports the use of functions as a preconditioner, and has a larger user base than our modified implementation of BiGCStab. Furthermore, our modified implementation converges for all Péclet and Damköhler numbers without requiring a preconditioner.

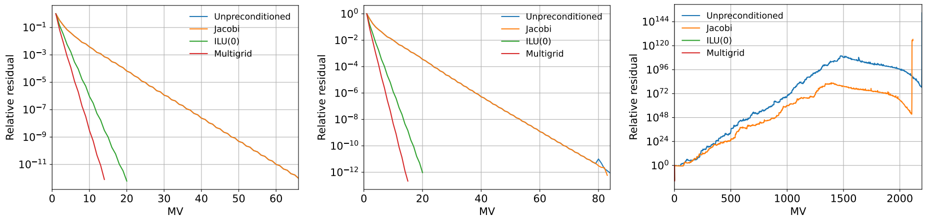
These experiments use the same parameters as presented in Section 6.4.1, however, with $M = 256$. Several convergence plots similar to the ones shown in Figure 6.4.3 are presented. The preconditioners used here are Jacobi, ILU(0) and a V-cycle geometric multigrid preconditioner with a depth of 4 and Gauss-Seidel smoothing in the direction of the advection [127, p. 95], the convergence plots of which are shown in Figure 6.4.7. The results show that multigrid is most efficient in terms of matrix-vector products, followed closely by ILU(0). Both multigrid and ILU(0) converge especially quickly for strong advection, since in this limit the system matrix becomes approximately lower-triangular. The Jacobi preconditioner gives no significant speedup, if any at all.

Note, however, that the issues with MATLAB's BiCGStab are not completely resolved by adding a preconditioner. As can be seen in Figure 6.4.7h, the issue related to the deviation between the estimated and the true residuals is still present, as can be seen by the sudden increase in residual after the estimated residual has reached the tolerance of 10^{-12}.

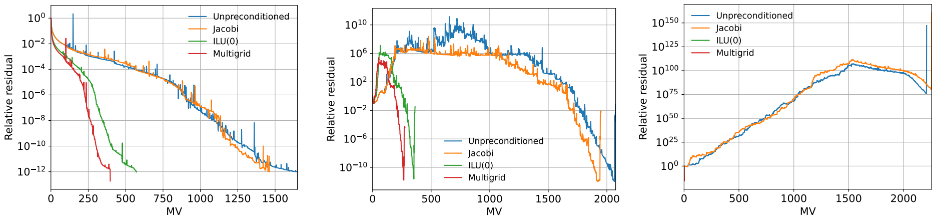
Even though a good preconditioner does significantly lower the required number of matrix-vector products as expected, it does not provide a remedy for all convergence issues. Both a good preconditioner *and* a robust iterative method are required for efficient and reliable convergence.



(a) $\mathrm{Pe} = 10^{-5}, \mathrm{Da} = 10^{5}$ (b) $\mathrm{Pe} = 1, \mathrm{Da} = 10^{5}$ (c) $\mathrm{Pe} = 10^{5}, \mathrm{Da} = 10^{5}$



(d) $\mathrm{Pe} = 10^{-5}, \mathrm{Da} = 1$ (e) $\mathrm{Pe} = 1, \mathrm{Da} = 1$ (f) $\mathrm{Pe} = 10^{5}, \mathrm{Da} = 1$



(g) $\mathrm{Pe} = 10^{-5}, \mathrm{Da} = 10^{-5}$ S (h) $\mathrm{Pe} = 1, \mathrm{Da} = 10^{-5}$ (i) $\mathrm{Pe} = 10^{5}, \mathrm{Da} = 10^{-5}$

Figure 6.4.7: Comparison of the estimated residual as function of number of MV for MATLAB's BiCGStab with various preconditioners with $M = 256$. For $\mathrm{Pe} = 10^{5}$ ILU(0) and Multigrid both converge in less than 5 MV.

6.5 Conclusions

As mentioned in the original reference of BiCGStab [38], there are many possible variants of BiCGStab which are all equivalent in exact arithmetic, but may have different behavior in finite precision arithmetic. In this chapter we have reported on the reliability of different implementations of BiCGStab and IDR for advection-diffusion-reaction (ADR) problems with real eigenvalues of the discretization matrix.

First, a baseline BiCGStab implementation has been presented, for which it has been shown that even for small matrices this variant can exhibit complications in converging to a solution. Three modifications to the baseline BiCGStab algorithm have been implemented, i.e., the choice of shadow residual $\widetilde{\boldsymbol{r}}$, the reliable updating scheme and keeping the solution with the smallest residual thus far.

A model ADR problem has been set up and converted into dimensionless form. Using this dimensionless form the entire parameter space of Péclet and Damköhler numbers can be investigated. The ADR-equation has been discretized using the Finite Volume Method in combination with the exponential scheme of [26]. Unlike the central difference scheme, the exponential scheme yields a discretization matrix with *real* eigenvalues, even for strong advection.

Since the eigenvalues are real, there is no improvement in reliability of BiCGStab(2) compared to BiCGStab. MATLAB's implementations of BiCGStab and BiCGStab(2) performed no better than the baseline BiCGStab algorithm. IDR(4) performed very well for practically all Péclet and Damköhler numbers, clearly outperforming the modified BiCGStab implementations, in terms of the number of matrix-vector products for problems with large Péclet number. For roughly $\mathrm{Pe} > 1$ and $\mathrm{Da} < \mathrm{Pe}$ MATLAB's solvers and the baseline BiCGStab method do not converge. Choosing a random shadow residual, $\widetilde{\boldsymbol{r}}$, is essential in the numerical experiments for the reliability of BiCGStab starting from a zero initial guess. It is hypothesized that this is due to the residual moving through the domain, eventually leading to $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle \to 0$ and a breakdown of the BiCG part.

It is shown for matrices resulting from discretized ADR equations that the LMR method can be interpreted as a time integration method for the advection equation subject to homogeneous Dirichlet boundary conditions. This is explicitly shown for a one-dimensional advection equation in Section 6.2.4. However, since the investigated discretization matrices are positive definite implying $\omega > 0$, this time stepping property also holds for these problems. Furthermore, it is demonstrated that BiCGStab shows similar behavior in numerical experiments. It is suspected that the effect of the residual moving as a wave in the advection direction is related to the conjecture in [126], where it was argued that the first phase (initial plateau where the residual remains relatively constant) lasts as long as the longest streamline takes to traverse the grid with the flow for residual-minimizing Krylov subspace methods.

The reliable updating scheme included in BiCGStab adds an extra step to ensure the computed residual truly achieves the prescribed tolerance. As long as the solver converges to the tolerance required, there is no benefit in storing the best solution thus far. If the method stalls the effect is still not significant in our experiments.

We recommend to modify BiCGStab with a random shadow residual, in conjunction with a reliable updating scheme, most notably if the initial residual is sparse, for example when it is only nonzero near the boundaries of the domain. Moreover IDR(4) is an excellent candidate, achieving the tolerance for a large number of cases. Additionally, for strong advection, IDR(4) uses significantly less matrix-vector products compared to BiCGStab.

In practice one cannot know beforehand which linear solver is optimal for a given problem. We have shown in this chapter that even specific implementations of BiCGStab have a significant impact on convergence. For example, the seemingly arbitrary parameter $\widetilde{\boldsymbol{r}}$ has a major impact on the robustness of the solver in our experiments. This observation complicates the applicability of specific iterative methods even further. Nevertheless, we conclude, based on the numerical results, that modified BiCGStab and IDR(S) would be preferable over standard BiCGStab for ADR problems. A practical benefit of the modified BiCGStab variant presented here is that choosing a different $\widetilde{\boldsymbol{r}}$ is trivial to implement in existing codes and should be sufficient to largely mitigate issues relating to the vanishing of $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle$.

In this chapter, the analysis is only applied to the linear advection-diffusion-reaction equation. However, for nonlinear problems such as the nonlinear ADR equation and the Navier-Stokes equations a linear system is still obtained after linearization using Newton or Picard iteration. Alternatively, one may choose to linearize before discretizing, for example by linearizing the source term of the ADR equation which was considered in this chapter. We believe that even for nonlinear PDEs the results shown here can still be relevant.

Appendix

6.A Derivation LMR step size and residual

In this appendix we show that under certain conditions the step size $\omega = \frac{1}{2}$; additionally we derive an expression for the k-th residual generated by the LMR algorithm for the problem given in Section 6.2.4.

First, the step size ω in the LMR algorithm is given by

$$\omega := \frac{\langle \boldsymbol{Ar}, \boldsymbol{r} \rangle}{\langle \boldsymbol{Ar}, \boldsymbol{Ar} \rangle}, \tag{6.44}$$

where the matrix $\boldsymbol{A}$ is the tridiagonal Toeplitz matrix

$$\boldsymbol{A} := \text{tridiag}(-1, 1, 0). \tag{6.45}$$

With these definitions, we will show that $\omega = \frac{1}{2}$ for any vector $\boldsymbol{r}$ with last element equal to zero given this specific matrix $\boldsymbol{A}$. We start by computing the numerator in (6.44);

$$\langle \boldsymbol{Ar}, \boldsymbol{r} \rangle = \sum_{i=1}^{N} (\boldsymbol{Ar})_i r_i = \sum_{i=1}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big) r_i, \tag{6.46}$$

then by splitting off the first term of the sum over i, and substituting the values for the elements of $\boldsymbol{A}$ we arrive at

$$\sum_{j=1}^{N} a_{1j} r_j r_1 + \sum_{i=2}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big) r_i = r_1^2 + \sum_{i=2}^{N} (-r_{i-1} + r_i) r_i. \tag{6.47}$$

The expression in (6.47) can be seen as the squared 2-norm of $\boldsymbol{r}$ with a rest term, since after expanding the brackets it is equal to

$$\sum_{i=1}^{N} r_i^2 - \sum_{i=2}^{N} r_{i-1} r_i = \|\boldsymbol{r}\|^2 - \sum_{i=1}^{N-1} r_i r_{i+1}. \tag{6.48}$$

The denominator in (6.44) is computed via a similar procedure;

$$\langle \boldsymbol{Ar}, \boldsymbol{Ar} \rangle = \sum_{i=1}^{N} (\boldsymbol{Ar})_i^2 = \sum_{i=1}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big)^2, \tag{6.49}$$

after again splitting off the first term of the i-sum and substituting the values for $\boldsymbol{A}$ we obtain

$$\Big(\sum_{j=1}^{N} a_{1j} r_j \Big)^2 + \sum_{i=2}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big)^2 = r_1^2 + \sum_{i=2}^{N} (-r_{i-1} + r_i)^2. \tag{6.50}$$

Then, expanding the squared term in the summation and relabeling the index, the denominator becomes

$$\begin{aligned} r_1^2 + \sum_{i=2}^{N}(r_i^2 + r_{i-1}^2 - 2r_{i-1}r_i) &= \|\boldsymbol{r}\|^2 + \sum_{i=2}^{N} r_{i-1}^2 - 2\sum_{i=2}^{N} r_{i-1}r_i \\ &= 2\|\boldsymbol{r}\|^2 - r_N^2 - 2\sum_{i=1}^{N-1} r_i r_{i+1}. \end{aligned} \tag{6.51}$$

By dividing (6.48) and (6.51) it is clear that

$$\omega = \tfrac{1}{2}, \quad \text{if} \quad r_N = 0. \tag{6.52}$$

Second, we derive an expression for the k-th residual produced by LMR starting from an initial residual of $\boldsymbol{e}_1$. The LMR recurrence for the residual, given $\omega = \frac{1}{2}$, is the following

$$\boldsymbol{r}^{k+1} = (\boldsymbol{I} - \omega\boldsymbol{A})\boldsymbol{r}^k = (\boldsymbol{I} - \tfrac{1}{2}\boldsymbol{A})\boldsymbol{r}^k. \tag{6.53}$$

Next, the matrix $\boldsymbol{B}$ is introduced as

$$\boldsymbol{B} := (\boldsymbol{I} - \tfrac{1}{2}\boldsymbol{A}) = \tfrac{1}{2}\text{tridiag}(1, 1, 0). \tag{6.54}$$

Note that the matrix $\boldsymbol{B}$ can be interpreted as a smoothing matrix, taking two neighboring elements of a vector and computing the average value. We then define

$$r_0^k := 0 \quad \text{for all} \quad k, \tag{6.55}$$

and a special case of the binomial coefficients as

$$\binom{0}{0} := 1. \tag{6.56}$$

Then for any vector $\boldsymbol{r}^k$ Equation (6.53) gives the recurrence

$$r_i^{k+1} = \tfrac{1}{2}(r_i^k + r_{i-1}^k), \quad i = 1, 2, ..., N. \tag{6.57}$$

We will prove that the k-th residual

$$r_i^k = \binom{k}{i-1}(\tfrac{1}{2})^k, \quad i = 1, 2, ..., k+1 < N, \tag{6.58}$$

with the other elements given by

$$r_i^k = 0, \quad i > k+1. \tag{6.59}$$

To prove (6.58) holds for any k, we use a proof by induction; we start by showing the base case $k = 0$, then for a given k we derive the case for $k + 1$.

Starting from the first residual vector $\boldsymbol{r}^0 = \boldsymbol{e}_1$;

$$\boldsymbol{r}_1^0 = 1, \quad r_i^0 = 0, \quad i > 1. \tag{6.60}$$

The LMR recurrence yields the next residual as

$$\boldsymbol{r}^1 = \boldsymbol{B}\boldsymbol{e}_1 = \frac{1}{2}\begin{bmatrix}1 & 1 & 0 & \dots & 0\end{bmatrix}^{\mathrm{T}}, \tag{6.61}$$

which agrees with (6.58). Next, we have to show that for any k, i

$$r_i^{k+1} = \binom{k+1}{i-1}(\tfrac{1}{2})^{k+1}, \quad i = 1, 2, ..., k+2 < N. \tag{6.62}$$

Substituting r_i^k from (6.58) into the recurrence (6.57) yields

$$r_i^{k+1} = \tfrac{1}{2}(r_i^k + r_{i-1}^k) = \tfrac{1}{2}\left[\binom{k}{i-1}(\tfrac{1}{2})^k + \binom{k}{i-2}(\tfrac{1}{2})^k\right], \quad i = 1, 2, ..., k+2 < N. \tag{6.63}$$

This can be rewritten to

$$r_i^{k+1} = \left[\binom{k}{i-1} + \binom{k}{i-2}\right](\tfrac{1}{2})^k = \binom{k+1}{i-1}(\tfrac{1}{2})^{k+1}. \tag{6.64}$$

Since (6.64) is equal to (6.58) after relabeling the index, it follows by induction that for any k Equation (6.58) satisfies the relation (6.53), completing the proof.

6.B Positive definite discretization matrices

A positive definite (PD) matrix has the property that $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} > 0$ for all vectors $\boldsymbol{x} \neq \boldsymbol{0}$ [75, p. 140]. First, we start by showing if $\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}} + \boldsymbol{A})\boldsymbol{x} > 0$ then $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} > 0$. Second, we show that the discretization matrices obtained for the one-dimensional ADR problem are PD. Finally, we show that the two-dimensional and three-dimensional ADR problems also yield PD discretization matrices.

First, note that

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x})^{\mathrm{T}} = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x}, \tag{6.65}$$

and consequently

$$\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}})\boldsymbol{x} = 2\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}, \tag{6.66}$$

therefore if

$$\boldsymbol{x}^{\mathrm{T}}(\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}})\boldsymbol{x} > 0, \quad \text{then}, \quad \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} > 0. \tag{6.67}$$

Second, we show that $\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}}$ with $\boldsymbol{A}_x$ given by (6.68) shortly, is PD. To do this, note that the matrix $\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}}$ is symmetric, therefore if all eigenvalues are positive then this matrix is PD [124, p.246]. The matrix $\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}}$ is given by

$$\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}} = \frac{1}{h}\mathrm{tridiag}(-\mathrm{B}(\mathrm{Pe}) - \mathrm{B}(-\mathrm{Pe}), 2\mathrm{B}(-\mathrm{Pe}) + 2\mathrm{B}(\mathrm{Pe}), -\mathrm{B}(-\mathrm{Pe}) - \mathrm{B}(\mathrm{Pe})), \tag{6.68}$$

which can be written as

$$\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}} = \frac{1}{h}(\mathrm{B}(\mathrm{Pe}) + \mathrm{B}(-\mathrm{Pe}))\mathrm{tridiag}(-1, 2, -1). \tag{6.69}$$

Since the eigenvalues of $\mathrm{tridiag}(-1, 2, -1)$ are positive for any number of grid points, $\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}}$ is PD. It can be shown similarly that also for positive Damköhler numbers the discretization matrix is PD, since the addition of a positive Damköhler number term only shifts all eigenvalues toward the right on the real axis.

To extend this derivation to the two-dimensional ADR problems, note that

$$(\boldsymbol{A}_x \oplus \boldsymbol{A}_y)^{\mathrm{T}} = (\boldsymbol{I} \otimes \boldsymbol{A}_x + \boldsymbol{A}_y \otimes \boldsymbol{I})^{\mathrm{T}} = (\boldsymbol{I} \otimes \boldsymbol{A}_x)^{\mathrm{T}} + (\boldsymbol{A}_y \otimes \boldsymbol{I})^{\mathrm{T}}, \tag{6.70}$$

and using the property $(\boldsymbol{I} \otimes \boldsymbol{A})^{\mathrm{T}} = \boldsymbol{I} \otimes \boldsymbol{A}^{\mathrm{T}}$ [128, p. 40] it follows that

$$(\boldsymbol{A}_x \oplus \boldsymbol{A}_y) + (\boldsymbol{A}_x \oplus \boldsymbol{A}_y)^{\mathrm{T}} = \boldsymbol{I} \otimes (\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}}) + (\boldsymbol{A}_y + \boldsymbol{A}_y^{\mathrm{T}}) \otimes \boldsymbol{I}. \tag{6.71}$$

By using corollary 4.2.13 [124, p.246] it is known that if $(\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}})$ is symmetric positive definite (SPD) then $\boldsymbol{I} \otimes (\boldsymbol{A}_x + \boldsymbol{A}_x^{\mathrm{T}})$ is also SPD. Via a similar argument it follows that $(\boldsymbol{A}_y + \boldsymbol{A}_y^{\mathrm{T}}) \otimes \boldsymbol{I}$ is also SPD. Since the element-wise sum of two SPD matrices yields another SPD matrix, we conclude that $(\boldsymbol{A}_x \oplus \boldsymbol{A}_y)$ is PD.

The matrices obtained from the three-dimensional discretization of the ADR equation with the exponential scheme can be shown to be PD similarly.

Chapter 7

From LMR to BiCGStab

Linear systems are ubiquitous in computational physics, ranging from simulations in fluid dynamics [129] to quantum mechanics [130], acoustics [119], electromagnetism [131], plasmas [96] and many more. A linear system is described by the equation

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{7.1}$$

where $\boldsymbol{A}$ is a real $N \times N$ matrix, $\boldsymbol{x}$ and $\boldsymbol{b}$ are real vectors of size N.

There is a wide variety of methods to solve such linear systems for $\boldsymbol{x}$, many of which rely on a decomposition of the matrix $\boldsymbol{A}$, such as LU decomposition, Cholesky decomposition etc. More specialized decompositions make use of the sparsity of the matrix. Such methods are classified as direct methods, which deliver the exact solution after a known number of arithmetic operations. There also exist iterative methods, which produce a sequence of successive approximations by repeating some update rules, until some termination criterion is reached [132].

The methods of interest in this thesis are the iterative methods called Krylov subspace methods; such methods seek to approximate the inverse of $\boldsymbol{A}$ with a matrix polynomial $P(\boldsymbol{A})$ [32, p157-p158] such that

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \approx P(\boldsymbol{A})\boldsymbol{b}. \tag{7.2}$$

To find such a polynomial, Krylov subspace methods compute projections on the linear subspace $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$;

$$\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v}) = \operatorname{span}\left\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}^2\boldsymbol{v}, ..., \boldsymbol{A}^{k-1}\boldsymbol{v}\right\}, \tag{7.3}$$

this subspace $\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{v})$ is the so-called Krylov subspace, named after Aleksei Nikolaevich Krylov who published on eigenvalue problems [133]. A wide variety of Krylov subspace methods exist, including, but not limited to the following:

1. Local Minimum Residual (LMR) [134].
2. Conjugate Gradient (CG) [135].
3. Conjugate Gradient Squared (CGS) [136].
4. BiConjugate Gradient (BiCG) [137].
5. BiConjugate Gradient Stabilized (BiCGStab) [38].
6. BiConjugate Gradient Stabilized with a higher order minimum residual step (BiCGStab(L)) [138].
7. Induced Dimension Reduction(S) (IDR(S)) [46].
8. Induced Dimension Reduction Stabilized with a higher order minimum residual step (IDR(S)Stab(L)) [48].
9. Minimum Residual (MinRes)[139].
10. Generalized Minimal Residual (GMRES) [44].
11. Quasi-Minimal Residual (QMR) [140].

Additionally, a lot of variants exist for these methods, for example for BiCGStab there exist:

1. The original description of [38].
2. A version adapted for efficient parallelization [112].
3. A method designed to recycle previously constructed Krylov subspaces for sequences of linear systems [114].
4. A block version designed for systems with multiple right hand sides [113].
5. An implementation designed for computers with distributed memory [111].
6. Adaptations for improving convergence when using finite precision arithmetic [141].
7. Variants for matrices with complex spectra [142].

In this thesis we cover only LMR, CG, BiCG, BiCGStab, BiCGStab(L), IDR(S) and IDR(S)Stab(L), and discuss variants of BiCGStab. This chapter covers the main concepts of Krylov methods, and discusses several pitfalls in their application. In the coming sections we start from the relatively simple LMR method, and work towards a description of our Eigen-implementations of IDR(S) [50], BiCGStab(L) [49] and IDR(S)Stab(L) [51] algorithms in the next chapter.

7.1 Local Minimum Residual (LMR)

In this section the Local Minimum Residual (LMR) method is introduced. LMR is one of the simplest Krylov subspace methods, which we use to introduce several key concepts. The aim of Krylov subspace methods is to solve the linear system (7.2) using an iterative procedure. Since the linear system is solved iteratively, we need a way to estimate the accuracy of the obtained solution. Ideally, one would use the error $\boldsymbol{e}_k$, defined as

$$\boldsymbol{e}_k := \boldsymbol{x} - \boldsymbol{x}_k, \tag{7.4}$$

as a measure for the quality of the obtained solution. However, the error is typically not known. An alternative measure can be obtained by left-multiplying (7.4) with the matrix $\boldsymbol{A}$,

$$\boldsymbol{A}\boldsymbol{e}_k = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k, \tag{7.5}$$

and defining the residual

$$\boldsymbol{r}_k := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k. \tag{7.6}$$

Starting with some initial guess $\boldsymbol{x}_0$ and corresponding residual $\boldsymbol{r}_0$, the LMR method seeks an update to $\boldsymbol{x}_k$ of the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \omega_k \boldsymbol{r}_k, \tag{7.7}$$

where ω_k is a coefficient that has to be determined at each iteration. In the LMR method, the value of ω_k is chosen such that the next residual is minimized in the 2-norm. Using (7.6) and (7.7), the next residual, $\boldsymbol{r}_{k+1}$, is obtained as

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x}_k + \omega_k \boldsymbol{r}_k), \tag{7.8}$$

which can be simplified to

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \omega_k \boldsymbol{A}\boldsymbol{r}_k. \tag{7.9}$$

To obtain ω_k which minimizes $\|\boldsymbol{r}_{k+1}\|_2$, both sides of (7.9) are left-multiplied by $\boldsymbol{r}_{k+1}^{\mathrm{T}}$,

$$\boldsymbol{r}_{k+1}^{\mathrm{T}} \boldsymbol{r}_{k+1} = (\boldsymbol{r}_k^{\mathrm{T}} - \omega_k \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{A}^{\mathrm{T}})(\boldsymbol{r}_k - \omega_k \boldsymbol{A}\boldsymbol{r}_k). \tag{7.10}$$

After expanding (7.10), and using $\boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{A}^{\mathrm{T}} \boldsymbol{r}_k = \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{A}\boldsymbol{r}_k$, we obtain a quadratic polynomial in ω_k

$$\|\boldsymbol{r}_{k+1}\|_2^2 = \|\boldsymbol{r}_k\|_2^2 - 2\omega_k \boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{A}\boldsymbol{r}_k + \omega_k^2 (\boldsymbol{A}\boldsymbol{r}_k)^{\mathrm{T}} \boldsymbol{A}\boldsymbol{r}_k, \tag{7.11}$$

which has a minimum at

$$\omega_k = \frac{\boldsymbol{r}_k^{\mathrm{T}} \boldsymbol{A}\boldsymbol{r}_k}{(\boldsymbol{A}\boldsymbol{r}_k)^{\mathrm{T}} (\boldsymbol{A}\boldsymbol{r}_k)}. \tag{7.12}$$

This choice of ω_k then minimizes $\|\boldsymbol{r}_{k+1}\|_2$, hence the name of the algorithm; LMR. With ω_k given in (7.12), the LMR algorithm can be set up, see Algorithm 5.

Algorithm 5 Basic LMR algorithm

```
function LMR($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol$)
    $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
    while $\|\boldsymbol{r}\|_2 > tol$ do
        $\omega \leftarrow \frac{\boldsymbol{r}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{r}}{(\boldsymbol{A}\boldsymbol{r})^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{r})}$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + \omega\boldsymbol{r}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - \omega\boldsymbol{A}\boldsymbol{r}$
    end while
    return $\boldsymbol{x}$
end function
```

7.1.1 Properties of the LMR algorithm

Even though the LMR algorithm will never diverge, as by construction the next residual is at most as big as the current residual (taking $\omega = 0$), it is possible to construct matrices for which convergence is arbitrarily slow. Consider the rotation matrix given by

$$\boldsymbol{A} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}, \tag{7.13}$$

and the effect it has on the coefficient ω_k in equation (7.12). For any $\boldsymbol{r}$, it can be shown that

$$\omega_k = \cos(\theta), \tag{7.14}$$

which indicates that the reduction of the residual in equation (7.9) can be made arbitrarily small. More specifically it can be shown that the residual vector is updated via the recurrence

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \cos(\theta)\boldsymbol{A}\boldsymbol{r}_k, \tag{7.15}$$

it can be shown using $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \boldsymbol{I}$ and $\boldsymbol{r}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{r}_k = \cos(\theta)\|\boldsymbol{r}_k\|^2$ for $\boldsymbol{r}_k \neq 0$ that

$$\frac{\|\boldsymbol{r}_{k+1}\|^2}{\|\boldsymbol{r}_k\|^2} = 1 - \cos(\theta)^2, \tag{7.16}$$

therefore starting from $\boldsymbol{r}_0$, after k iterations the LMR algorithm produces a residual with norm

$$\|\boldsymbol{r}_{k+1}\|^2 = \|\boldsymbol{r}_0\|^2(1 - \cos(\theta)^2)^{k+1}. \tag{7.17}$$

As expected, the residual will never increase, since $|1 - \cos(\theta)^2| = |\sin^2(\theta)| \leq 1$. However, for $\theta = \pi/2$ the LMR algorithm stalls. The number of iterations needed for convergence of LMR as function of θ is shown in Figure 7.1.1a. Here it can be seen that the number of iterations required sharply increases near $\theta = \pi/2$ and $\theta = 3\pi/2$.

This result can also be understood from an eigenvalue analysis. Firstly, note that if $\boldsymbol{r}_k$ is an eigenvector of $\boldsymbol{A}$ with eigenvalue λ, then choosing $\omega_k = 1/\lambda$ leads to $\boldsymbol{r}_{k+1} = \boldsymbol{0}$. Secondly, note that if $\boldsymbol{r}_k$ is real, then ω_k is also real. Thirdly, consider an expansion of $\boldsymbol{r}_k$ in terms of the eigenvectors of $\boldsymbol{A}$,

$$\boldsymbol{r}_k = \xi_1 \boldsymbol{v}_1 + \xi_2 \boldsymbol{v}_2, \tag{7.18}$$

where ξ_i are the expansion coefficients and $\boldsymbol{v}_i$ the eigenvectors of $\boldsymbol{A}$. Then $\boldsymbol{r}_{k+1}$ is given by

$$\boldsymbol{r}_{k+1} = \xi_1 \boldsymbol{v}_1 + \xi_2 \boldsymbol{v}_2 - \omega_k \boldsymbol{A}(\xi_1 \boldsymbol{v}_1 + \xi_2 \boldsymbol{v}_2), \tag{7.19}$$

then by using $\boldsymbol{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, with λ_i the eigenvalue corresponding to $\boldsymbol{v}_i$ we obtain

$$\boldsymbol{r}_{k+1} = \xi_1(1 - \omega_k \lambda_1)\boldsymbol{v}_1 + \xi_2(1 - \omega_k \lambda_2)\boldsymbol{v}_2. \tag{7.20}$$

Note that the eigenvalues of (7.13) are $e^{\pm i\theta} = \cos(\theta) \pm i\sin(\theta)$, and that $\omega_k = \cos(\theta)$ is a real number. Furthermore, note if $\omega_k = 1/\lambda_1$ then the $\boldsymbol{v}_1$ component of the residual is eliminated. This suggests that, complex ω may greatly improve convergence, since choosing $\omega_1 = 1/\lambda_1$ and $\omega_2 = 1/\lambda_2$ achieves $\boldsymbol{r}_3 = \boldsymbol{0}$. An alternative to complex ω is to consider a vector $\boldsymbol{\omega}$, this extension is discussed in Section 7.1.3. The convergence issues for $\omega \approx 0$ are similar in nature to the convergence issues BiCGStab [38] faces when the system matrix has eigenvalues with a large imaginary component. The extension to a vector $\boldsymbol{\omega}$ to work around this issue, is similar to the idea of BiCGStab(L), which is presented later.

Another example that illustrates the convergences of LMR can be obtained by applying the LMR to the linear system with the matrix

$$\boldsymbol{A} = \boldsymbol{I} + \boldsymbol{C}, \quad \boldsymbol{C} = \begin{bmatrix} 0 & c \\ -c & 0 \end{bmatrix}, \tag{7.21}$$

where $\boldsymbol{C}$ represents the anti-symmetric part of $\boldsymbol{A}$. It is straightforward to show that for such systems ω_k is given by

$$\omega_k = \frac{1}{1 + c^2}, \tag{7.22}$$

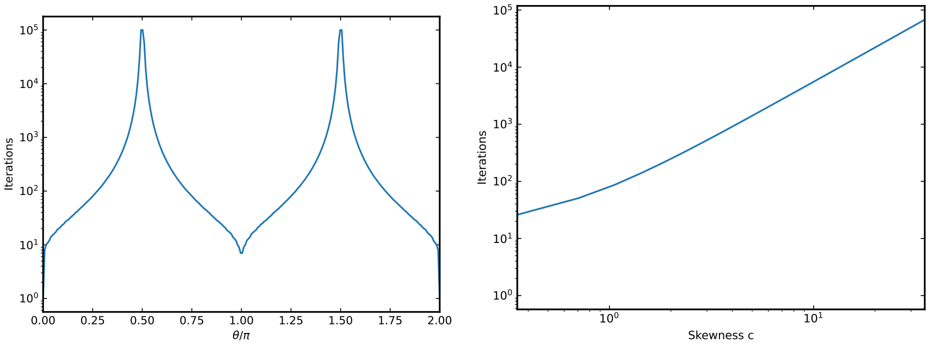
and that the recurrence relation for the residual becomes

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \frac{1}{1 + c^2}(\boldsymbol{r}_k + \boldsymbol{C}\boldsymbol{r}_k). \tag{7.23}$$

By computing the norm of $\boldsymbol{r}_{k+1}$ it follows that the residual decreases as

$$\frac{\|\boldsymbol{r}_{k+1}\|^2}{\|\boldsymbol{r}_0\|^2} = \left(1 - \frac{1}{1 + c^2}\right)^k, \tag{7.24}$$

this shows that the convergence of LMR slows down with increasing c, suggesting that for asymmetric matrices LMR may converge slowly. The effect of increasing the skewness of $\boldsymbol{A}$ is shown in Figure 7.1.1b.



(a) Rotation matrix (7.13)

(b) Skew symmetric matrix (7.21)

Figure 7.1.1: Number of iterations required for LMR to converge to $\|\boldsymbol{r}\| < 10^{-12}$ starting from an initial guess of $\boldsymbol{x}_0 = \boldsymbol{0}$ and right hand side $\boldsymbol{b} = \boldsymbol{1}$.

7.1.2 LMR iteration as discretized time-dependent linear PDE

Note that the LMR recurrence given by Equation (7.9) can be rearranged to become

$$\frac{\boldsymbol{r}_{k+1} - \boldsymbol{r}_k}{\omega_k} = -\boldsymbol{A}\boldsymbol{r}_k. \tag{7.25}$$

If $\omega_k > 0$ the LMR iteration can be interpreted as a time integration method for the linear, first-order ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r} = -\boldsymbol{A}\boldsymbol{r}. \tag{7.26}$$

For positive definite $\boldsymbol{A}$ it can be shown that $\omega_k > 0$ for all k. In that case (7.25) is a time-discretization of (7.26) using forward Euler and a variable time-step ω_k. Additionally, due to the local minimization property, the "variable timestep" ω_k is such that forward Euler does not diverge, since $\|\boldsymbol{r}_{k+1}\|_2 \leq \|\boldsymbol{r}_k\|_2$.

In Chapter 6 a discretization of the ADR equation

$$\frac{\partial}{\partial t}\varphi = -\nabla \cdot (\vec{u}\varphi - \epsilon\nabla\varphi), \tag{7.27}$$

was considered, which (depending on the discretization scheme) results in a positive definite, but not necessarily symmetric, matrix $\boldsymbol{A}$. This has the benefit that previous experience with equations of the type (7.27) can be extended to insights regarding the propagation of the LMR residual; knowledge of the PDE can be applied to investigate convergence of LMR as demonstrated in Chapter 6.

On the one hand LMR is straightforward to investigate analytically, easy to implement, and never diverges. On the other hand, the number of iterations can be excessively large, even for systems with just two unknowns. Therefore, in the coming sections we consider extensions and alternatives to the LMR method.

The LMR recurrence can be written as a polynomial of the form

$$\boldsymbol{r}_{k+1} = (\boldsymbol{I} - \omega_k\boldsymbol{A})\boldsymbol{r}_k, \tag{7.28}$$

an extension to LMR would be to use a polynomial of degree two, similar to the higher order polynomial step in BiCGStab(L) [45], such as

$$\boldsymbol{r}_{k+1} = (\boldsymbol{I} - \omega_{k,1}\boldsymbol{A} - \omega_{k,2}\boldsymbol{A}^2)\boldsymbol{r}_k, \tag{7.29}$$

this idea is investigated in the next section.

7.1.3 Extended Local Minimal Residual (LMR(2))

As sketched in equation (7.29), one possible extension to the LMR algorithm is to consider a second order polynomial in $\boldsymbol{A}$. The idea is again to choose the coefficients such that $\|\boldsymbol{r}_{k+1}\|_2^2$ is minimized. To achieve this minimization, we introduce a vector $\boldsymbol{\omega}_k$,

$$\boldsymbol{\omega}_k := \begin{bmatrix} \omega_{k,1} \\ \omega_{k,2} \end{bmatrix}. \tag{7.30}$$

In order to minimize $\boldsymbol{r}_{k+1}$, we have to find $\boldsymbol{\omega}_k$ which solves the following least-squares problem

$$\boldsymbol{r}_{k+1} = \begin{bmatrix} \boldsymbol{A}\boldsymbol{r}_k & \boldsymbol{A}^2\boldsymbol{r}_k \end{bmatrix} \boldsymbol{\omega}_k. \tag{7.31}$$

This least-squares problem can be solved with a variety of methods, most notably by orthogonalizing $\boldsymbol{A}\boldsymbol{r}_k$ and $\boldsymbol{A}^2\boldsymbol{r}_k$. The resulting LMR(2) algorithm is given in Algorithm 6. Section 7.2 discusses an extension to LMR(m), by orthogonalizing an arbitrary number of vectors to minimize $\|\boldsymbol{r}_{k+1}\|_2$. Note that in some references on Krylov subspace methods [143, 32, 47], these least-squares problems are denoted in terms of argmin, i.e.,

$$\underset{\boldsymbol{\omega}}{\operatorname{argmin}} \|\boldsymbol{r} - [\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}]\boldsymbol{\omega}\|_2, \tag{7.32}$$

which denotes finding $\boldsymbol{\omega}$ such that it minimizes $\|\boldsymbol{r} - [\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}]\boldsymbol{\omega}\|_2$.

Algorithm 6 LMR(2) algorithm

```
function LMR(2)($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol$)
    $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
    while $\|\boldsymbol{r}\|_2 > tol$ do
        $\boldsymbol{\omega} \leftarrow \operatorname{argmin}_{\boldsymbol{\omega}} \|\boldsymbol{r} - [\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}]\boldsymbol{\omega}\|_2$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + [\boldsymbol{r}, \boldsymbol{A}\boldsymbol{r}]\boldsymbol{\omega}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - [\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}]\boldsymbol{\omega}$
    end while
    return $\boldsymbol{x}$
end function
```

If $\boldsymbol{A}$ is of size 2×2, and the vectors $\boldsymbol{A}\boldsymbol{r}$ and $\boldsymbol{A}^2\boldsymbol{r}$ are linearly independent, LMR(2) will converge to the exact solution within one iteration. To extend this favorable property to larger systems, the next section discusses the Full Orthogonalization Method (FOM). A method that is guaranteed to converge in N iterations for a linear system with N unknowns.

7.2 Full Orthogonalization Method (FOM)

In this section FOM is discussed. In essence, FOM extends the LMR(2) algorithm to LMR(m), where the argmin step is computed by explicit orthogonalization of the space spanned by $\{\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}, \ldots, \boldsymbol{A}^m\boldsymbol{r}\}$.

Algorithm 7 LMR(m) algorithm

```
function LMR($m$)($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol, m$)
    $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
    while $\|\boldsymbol{r}\|_2 > tol$ do
        $\boldsymbol{\omega} \leftarrow \operatorname{argmin}_{\boldsymbol{\omega}} \|\boldsymbol{r} - [\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}, \ldots, \boldsymbol{A}^m\boldsymbol{r}]\boldsymbol{\omega}\|_2$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + [\boldsymbol{r}, \boldsymbol{A}\boldsymbol{r}, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{r}]\boldsymbol{\omega}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - [\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}, \ldots, \boldsymbol{A}^m\boldsymbol{r}]\boldsymbol{\omega}$
    end while
    return $\boldsymbol{x}$
end function
```

In principle, it is possible to compute $\boldsymbol{\omega}$ without orthogonalization of $\boldsymbol{B} = [\boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}, \ldots, \boldsymbol{A}^m\boldsymbol{r}]$, for example by solving a system of normal equations $\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{\omega} = \boldsymbol{B}\boldsymbol{r}$. However, this is problematic as the columns of $\boldsymbol{B}$ tend towards the largest eigenvector of $\boldsymbol{A}$, and thus tend to become nearly linearly dependent. This leads to conditioning issues, which are amplified by the property that $\kappa(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}) = \kappa(\boldsymbol{B})^2$ where κ is the condition number.

In the FOM algorithm, one iteration of LMR(m) is computed by explicit orthogonalization using (modified) Gram-Schmidt. The resulting update to $\boldsymbol{x}$ is then obtained by solving a different linear system. The orthogonalization procedure is similar to computing the QR-decomposition of $\boldsymbol{B}$ using Gram-Schmidt. However, instead of a triangular matrix $\boldsymbol{R}$ FOM produces a Hessenberg matrix $\boldsymbol{H}$. In the special case of orthogonalizing a Krylov subspace like $[\boldsymbol{r}, \boldsymbol{A}\boldsymbol{r}, \ldots, \boldsymbol{A}^{m-1}\boldsymbol{r}]$ this orthogonalization procedure is also known as Arnoldi iteration.

The FOM algorithm is given in Algorithm 8, based on Algorithm 6.4 of [32]. A slightly modified version of Algorithm 8 is used in our IDR(S)Stab(L) implementation, which will be covered in Section 8.3. Like LMR, FOM will never diverge due to the residual minimization property. However, the cost of FOM scales as $\mathcal{O}(m^2)$, since every new vector $\boldsymbol{v}_{j+1}$ has to be made orthogonal to every previous vector $\boldsymbol{v}_i$ with $i \leq j$.

In the next section we discuss a Krylov method, which builds an orthogonal basis while only using one previous vector.

Algorithm 8 Full Orthogonalization Method (FOM)

```
function FOM($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, m$)
    $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
    $\boldsymbol{v}_1 \leftarrow \boldsymbol{r}/\|\boldsymbol{r}\|_2$
    $\boldsymbol{H} = \boldsymbol{0}$                                    ▷ $\boldsymbol{H} \in \mathbb{R}^{m \times m}$
    for $j = 1, 2, \dots, m$ do
        $\boldsymbol{w} \leftarrow \boldsymbol{A}\boldsymbol{v}_j$
        for $i = 1, \dots, j$ do
            $H_{ij} = \langle \boldsymbol{w}, \boldsymbol{v}_i \rangle$
            $\boldsymbol{w} \leftarrow \boldsymbol{w} - H_{ij}\boldsymbol{v}_i$
        end for
        $H_{j+1,j} = \|\boldsymbol{w}\|_2$
        if $H_{j+1,j} = 0$ then
            $j \leftarrow m$
            $\boldsymbol{y} \leftarrow \|\boldsymbol{r}_0\|_2 \boldsymbol{H}^{-1}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{r}_0$
            $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{V}\boldsymbol{y}$
            return $\boldsymbol{x}$
        end if
    end for
    $\boldsymbol{y} \leftarrow \|\boldsymbol{r}\|_2 \boldsymbol{H}^{-1}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{r}_0$          ▷ $\boldsymbol{V} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_m]$
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{V}\boldsymbol{y}$
    return $\boldsymbol{x}$
end function
```

7.3 Conjugate gradient method (CG)

Ideally one would like to combine the short recurrence property of the LMR algorithm, Section 7.1, with the guaranteed convergence and optimality properties of the FOM algorithm, in Section 7.2. However, algorithms with these two properties do not exist for an arbitrary matrix $\boldsymbol{A}$, by the Faber-Manteuffel theorem [35, 144]. Nevertheless, for an important class of matrices such an algorithm *can* be constructed. In the case of symmetric, positive definite matrices the well-known Conjugate Gradient algorithm is both a short recurrence method, and optimal in the sense that the error is minimized over some norm [135, 34, 32]. The derivation of CG presented here is based on [34].

The goal of the Conjugate Gradient method is to solve a linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{7.33}$$

by starting from an initial guess $\boldsymbol{x}_0$, and iteratively improving on this guess. Here the matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, and the vectors $\boldsymbol{x} \in \mathbb{R}^{N \times 1}$ and $\boldsymbol{b} \in \mathbb{R}^{N \times 1}$. After k

iterations an approximate solution $\boldsymbol{x}_k$ is obtained, with some error $\boldsymbol{e}_k$ defined as

$$\boldsymbol{e}_k := \boldsymbol{x} - \boldsymbol{x}_k. \tag{7.34}$$

Typically, the error is unknown, however, knowing the error would allow for a trivial computation of the exact solution; $\boldsymbol{x} = \boldsymbol{x}_k + \boldsymbol{e}_k$. Therefore, a residual associated with an iterate $\boldsymbol{x}_k$ is used as a quantitative measure for the obtained solution;

$$\boldsymbol{r}_k := \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{A}\boldsymbol{e}_k. \tag{7.35}$$

Similar to FOM, in the CG-method we want to compute an orthogonal basis, and use it to obtain successive approximations for the solution $\boldsymbol{x}$. A first idea would be to use some set of orthogonal vectors $\{\boldsymbol{d}_0, \dots, \boldsymbol{d}_{N-1}\}$ such that

$$\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{d}_j = 0, \quad \text{if} \quad i \neq j, \tag{7.36}$$

and expand $\boldsymbol{x}$ in terms of this basis

$$\boldsymbol{x} = \sum_{j=0}^{N-1} \gamma_j \boldsymbol{d}_j. \tag{7.37}$$

Note that, because of the orthogonality property (7.36), the coefficients γ_j can be obtained by left-multiplying (7.37) with the transposed vectors from the set $\{\boldsymbol{d}_0, \dots, \boldsymbol{d}_{N-1}\}$ to obtain;

$$\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{x} = \sum_{j=0}^{N-1} \gamma_j \boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{d}_j = \gamma_i \boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{d}_i, \tag{7.38}$$

which results in an expression for the expansion coefficients γ_i,

$$\gamma_i = \frac{\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{x}}{\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{d}_i}. \tag{7.39}$$

Unfortunately, computing the coefficients γ_i using (7.39) requires the exact solution $\boldsymbol{x}$.

However, it is possible to obtain an orthogonal expansion for which the expansion coefficients can be computed without requiring $\boldsymbol{x}$. The idea is to start with a basis $\{\boldsymbol{p}_0, \dots, \boldsymbol{p}_{N-1}\}$, the properties of which will be discussed shortly. Then again expand $\boldsymbol{x}$ in terms of this basis,

$$\boldsymbol{x} = \sum_{j=0}^{N-1} \alpha_j \boldsymbol{p}_j, \tag{7.40}$$

left-multiply both sides by $\boldsymbol{A}$ and use that $\boldsymbol{Ax} = \boldsymbol{b}$. This results in the expression

$$\boldsymbol{Ax} = \boldsymbol{b} = \sum_{j=0}^{N-1} \alpha_j \boldsymbol{A}\boldsymbol{p}_j. \tag{7.41}$$

To compute the expansion coefficients α_j, we assume $\boldsymbol{A}$ is symmetric positive definite, which allows us to define an inner product

$$\langle \boldsymbol{p}_i, \boldsymbol{p}_j \rangle_{\boldsymbol{A}} := \boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{A}\boldsymbol{p}_j. \tag{7.42}$$

It is then possible to define orthogonal vectors with respect to this inner product. The vectors $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ are said to be $\boldsymbol{A}$-orthogonal if

$$\langle \boldsymbol{p}_i, \boldsymbol{p}_j \rangle_{\boldsymbol{A}} = 0, \quad \text{if} \quad i \neq j. \tag{7.43}$$

Then a similar concept as used in (7.37) can be applied to (7.41); by left-multiplying with vectors from the set $\{\boldsymbol{p}_0, \dots, \boldsymbol{p}_{N-1}\}$ the expansion coefficients α_j can be obtained;

$$\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{b} = \sum_{j=0}^{N-1} \alpha_j \boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{A}\boldsymbol{p}_j = \alpha_i \boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{A}\boldsymbol{p}_i. \tag{7.44}$$

This yields an expression for the expansion coefficients,

$$\alpha_i = \frac{\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{b}}{\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{A}\boldsymbol{p}_i}, \tag{7.45}$$

which can be obtained if one has a suitable set of $\boldsymbol{A}$-orthogonal vectors $\{\boldsymbol{p}_0, \dots, \boldsymbol{p}_{N-1}\}$.

Next, we will construct a set of $\boldsymbol{A}$-orthogonal vectors from an arbitrary, but linearly independent set of vectors $\{\boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1}\}$. Similar to the Gram-Schmidt procedure of generating orthogonal vectors, a variant of this procedure can also be used to generate $\boldsymbol{A}$-orthogonal vectors;

$$\boldsymbol{p}_k = \boldsymbol{u}_k - \sum_{j=0}^{k-1} \frac{\langle \boldsymbol{u}_k, \boldsymbol{p}_j \rangle_{\boldsymbol{A}}}{\langle \boldsymbol{p}_j, \boldsymbol{p}_j \rangle_{\boldsymbol{A}}} \boldsymbol{p}_j = \boldsymbol{u}_k - \sum_{j=0}^{k-1} \frac{\boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{A}\boldsymbol{p}_j}{\boldsymbol{p}_j^{\mathrm{T}} \boldsymbol{A}\boldsymbol{p}_j} \boldsymbol{p}_j. \tag{7.46}$$

In principle one can construct an iterative algorithm using (7.46) and (7.45) similar to the FOM algorithm in Section 7.2, by starting from *any* set of vectors $\{\boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1}\}$, $\boldsymbol{A}$-orthogonalizing, computing the expansion coefficients α, and computing $\boldsymbol{x}$ using equation (7.40). However, note that every new vector $\boldsymbol{p}_k$ has to be made $\boldsymbol{A}$-orthogonal to *every* $\boldsymbol{p}_{<k}$; with each additional vector the algorithm becomes significantly more expensive. However, using the properties of the $\boldsymbol{A}$-orthogonal basis a more efficient algorithm can be constructed.

To optimize the $\boldsymbol{A}$-orthogonalization procedure some additional insight is needed, for brevity we take $\boldsymbol{x}_0 = \boldsymbol{0}$. Note that the iterative procedure produces, after $k+1$ iterations, an approximation $\boldsymbol{x}_{k+1}$

$$\boldsymbol{x}_{k+1} = \sum_{j=0}^{k} \alpha_j \boldsymbol{p}_j, \tag{7.47}$$

with an associated error given by

$$\boldsymbol{e}_{k+1} = \boldsymbol{x} - \boldsymbol{x}_{k+1} = \sum_{j=0}^{N-1} \alpha_j \boldsymbol{p}_j - \sum_{j=0}^{k} \alpha_j \boldsymbol{p}_j = \sum_{j=k+1}^{N-1} \alpha_j \boldsymbol{p}_j. \tag{7.48}$$

Left-multiplying (7.48) by $\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{A}$ and using $\boldsymbol{A}\boldsymbol{e}_{k+1} = \boldsymbol{r}_{k+1}$ yields;

$$\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{r}_{k+1} = \sum_{j=k+1}^{N-1} \alpha_j \boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{A} \boldsymbol{p}_j. \tag{7.49}$$

Then, due to the $\boldsymbol{A}$-orthogonality of the basis vectors $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}\}$ it follows that

$$\boldsymbol{p}_k^{\mathrm{T}} \boldsymbol{r}_{k+1} = 0. \tag{7.50}$$

Equation (7.50) thus implies that the $(k+1)$-th residual is orthogonal to all $\boldsymbol{p}_{\leq k}$.

Using (7.47), it is clear that one can obtain a recurrence for obtaining the next approximation of $\boldsymbol{x}$;

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k, \tag{7.51}$$

from which it follows that

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k - \alpha_k \boldsymbol{p}_k. \tag{7.52}$$

Using equations (7.51) and (7.52), an explicit expression for α_k can be obtained. Firstly, note that the vectors $\boldsymbol{p}_k$ are constructed to be $\boldsymbol{A}$-orthogonal. Secondly, α_k can be chosen such that the next error $\boldsymbol{e}_{k+1}$ is $\boldsymbol{A}$-orthogonal to $\boldsymbol{p}_k$. This implies that the error has no component remaining in the direction $\boldsymbol{p}_k$. This implies at least a local optimality property, where an error or a residual is minimized each iteration.

Since all vectors of the set $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}\}$ are $\boldsymbol{A}$-orthogonal to each other, this implies the error cannot increase (in some norm) and the exact solution is obtained after N iterations, as after N iterations $\{\boldsymbol{p}_0, \dots, \boldsymbol{p}_{N-1}\}$ spans $\mathbb{R}^N$.

To construct this α_k, left-multiply (7.52) by $\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}$ and impose $\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}_{k+1} = 0$;

$$\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}_{k+1} = 0 = \boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}_k - \alpha_k \boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k, \tag{7.53}$$

this results in an expression for α_k,

$$\alpha_k = \frac{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}_k}{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k} = \frac{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{r}_k}{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}. \tag{7.54}$$

Finally, a recurrence for the next residual can be obtained as follows;

$$\boldsymbol{r}_{k+1} = \boldsymbol{A}\boldsymbol{e}_{k+1}, \tag{7.55a}$$
$$\boldsymbol{r}_{k+1} = \boldsymbol{A}(\boldsymbol{e}_k - \alpha_k \boldsymbol{p}_k), \tag{7.55b}$$
$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{p}_k. \tag{7.55c}$$

It can be shown that this choice of α_k as constructed in (7.54) minimizes $\|\boldsymbol{e}_{k+1}\|_{\boldsymbol{A}} := \boldsymbol{e}_{k+1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{e}_{k+1}$. Starting from (7.52), the choice of α_k that minimizes $\|\boldsymbol{e}_{k+1}\|_{\boldsymbol{A}}$ can be obtained by solving

$$\frac{\mathrm{d}}{\mathrm{d}\alpha}\|\boldsymbol{e}_{k+1}\|_{\boldsymbol{A}}^2 = -2\boldsymbol{e}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k + 2\alpha_k \boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k = 0, \tag{7.56}$$

which results in α_k as given in (7.54).

Next, we use the $\boldsymbol{A}$-orthogonality to make computing the vectors $\{\boldsymbol{p}_0, \dots, \boldsymbol{p}_{N-1}\}$ significantly more efficient. The idea is to take $\boldsymbol{u}_i = \boldsymbol{r}_i$ in equation (7.46), obtain an expression for $\{\boldsymbol{p}_0, \dots, \boldsymbol{p}_{N-1}\}$, then use the orthogonality property (7.50) to eliminate most terms in the summation of (7.46). This also yields that $\boldsymbol{p}_0 = \boldsymbol{r}_0$.

Note that equation (7.55c) shows that $\boldsymbol{r}_{k+1}$ is a linear combination of the previous residual $\boldsymbol{r}_k$, and $\boldsymbol{A}\boldsymbol{p}_k$. Additionally, after k iterations the spans of the vectors $\boldsymbol{r}$ and $\boldsymbol{p}$ have the property that

$$\mathcal{P}_k = \operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{r}_1, ..., \boldsymbol{r}_{k-1}\} \tag{7.57a}$$
$$= \operatorname{span}\{\boldsymbol{p}_0, \boldsymbol{p}_1, ..., \boldsymbol{p}_{k-1}\} \tag{7.57b}$$
$$= \operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, ..., \boldsymbol{A}^{k-1}\boldsymbol{r}_0\} \tag{7.57c}$$
$$= \operatorname{span}\{\boldsymbol{p}_0, \boldsymbol{A}\boldsymbol{p}_0, ..., \boldsymbol{A}^{k-1}\boldsymbol{p}_0\}. \tag{7.57d}$$

With the observation in (7.57) a significant efficiency improvement can be obtained. Firstly, note that

$$\mathcal{P}_{k+1} = \operatorname{span}\{\boldsymbol{p}_0, \boldsymbol{p}_1, ..., \boldsymbol{p}_k\} = \operatorname{span}\{\boldsymbol{p}_0, \boldsymbol{A}\boldsymbol{p}_0, ..., \boldsymbol{A}^k\boldsymbol{p}_0\}. \tag{7.58}$$

Secondly, from the orthogonality property (7.50) it follows that the next residual, $\boldsymbol{r}_{k+1} \perp \mathcal{P}_{k+1}$. Thirdly, we use that

$$\boldsymbol{A}\mathcal{P}_k = \operatorname{span}\{\boldsymbol{A}\boldsymbol{p}_0, \boldsymbol{A}\boldsymbol{p}_1, ..., \boldsymbol{A}\boldsymbol{p}_{k-1}\} = \operatorname{span}\{\boldsymbol{A}\boldsymbol{p}_0, \boldsymbol{A}^2\boldsymbol{p}_0, ..., \boldsymbol{A}^{k-1}\boldsymbol{p}_0\}, \tag{7.59}$$

is a subspace of $\mathcal{P}_{k+1}$. Since $\boldsymbol{r}_{k+1} \perp \mathcal{P}_{k+1}$ it follows that $\boldsymbol{r}_{k+1} \perp \boldsymbol{A}\mathcal{P}_k$, i.e., $\boldsymbol{r}_{k+1}$ is $\boldsymbol{A}$-orthogonal to $\mathcal{P}_k$. This leads to the conclusion

$$\boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_j = 0, \quad \text{if} \quad j < k, \tag{7.60}$$

therefore the residual $\boldsymbol{r}_{k+1}$ is already $\boldsymbol{A}$-orthogonal to all previous search directions. This result of equation (7.60) greatly simplifies the Gram-Schmidt procedure used to construct the $\boldsymbol{A}$-orthogonal basis in equation (7.46);

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \sum_{j=0}^{k} \frac{\boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_j}{\boldsymbol{p}_j^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_j}\boldsymbol{p}_j = \boldsymbol{r}_{k+1} - \frac{\boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}\boldsymbol{p}_k. \tag{7.61}$$

Then using equations (7.51), (7.54), (7.55) and (7.61) the complete Conjugate Gradient algorithm becomes Algorithm 9.

Even though this algorithm uses only short recurrences, as it only takes one previous basis vector to construct an orthogonal basis CG also guarantees to minimize the error in some norm. However, this algorithm strongly relies on the concept of $\boldsymbol{A}$-orthogonality. Unfortunately this property can only be exploited if $\boldsymbol{A}$ is symmetric, positive definite.

In the next sections we present iterative methods that do not have this restriction on $\boldsymbol{A}$. However, by the Faber-Manteuffel theorem [35] there exists no optimal Krylov subspace method with short recurrences for every matrix $\boldsymbol{A}$, since in general it is not possible to generate an orthogonal basis of Krylov subspaces via only short recurrences[144]. Therefore, these methods will either have to sacrifice optimality, short recurrence, use a different space altogether, or find a compromise.

Algorithm 9 Conjugate Gradient method

```
function CG($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}_0, tol$)
    $\boldsymbol{r}_0 \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$
    $\boldsymbol{p}_0 \leftarrow \boldsymbol{r}_0$
    $k \leftarrow 0$
    while $\|\boldsymbol{r}_k\|_2 > tol$ do
        $\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{p}_k$
        $\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{r}_{k+1} - \frac{\boldsymbol{r}_{k+1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}\boldsymbol{p}_k$
        $\alpha_k \leftarrow \frac{\boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{r}_k}{\boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}$
        $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$
        $k \leftarrow k + 1$
    end while
    return $\boldsymbol{x}$
end function
```

7.4 Bi-Conjugate Gradient (BiCG)

The derivation of the BiCG algorithm as presented in this section is based on reference [145]. In Section 7.3 it was shown that CG constructs residuals such that

$$\boldsymbol{r}_{k+1} \perp \operatorname{span}\{\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_{\mathbf{0}}, ..., \boldsymbol{A}^k\boldsymbol{r}_0\}, \tag{7.62}$$

where it constructs these residuals efficiently by exploiting $\boldsymbol{A}$-orthogonality. However, CG requires that $\boldsymbol{A}$ is symmetric positive definite.

For a general matrix $\boldsymbol{A}$ it is expensive to construct an orthogonal basis of the Krylov subspace due to the Faber-Manteuffel theorem, therefore a different approach has to be taken. The idea of BiCG is to construct *two* Krylov subspaces, and exploit *bi*-orthogonality, i.e., the property for two indexed families of vectors $\boldsymbol{v}_i$ and $\widetilde{\boldsymbol{v}}_j$ such that,

$$\langle \boldsymbol{v}_i, \widetilde{\boldsymbol{v}}_j \rangle = 0 \quad \text{if} \quad i \neq j. \tag{7.63}$$

To achieve this, BiCG constructs a sequence of residuals $\boldsymbol{r}_k$, and a sequence of "shadow" residuals $\widetilde{\boldsymbol{r}}_k$ such that

$$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k \in \boldsymbol{r}_0 + \boldsymbol{A}\mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0), \quad \widetilde{\boldsymbol{r}}_k \in \widetilde{\boldsymbol{r}}_0 + \boldsymbol{A}^{\mathrm{T}}\mathcal{K}_k(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0), \tag{7.64}$$

with a bi-orthogonality requirement dictating that the residuals satisfy

$$\boldsymbol{r}_k \perp \mathcal{K}_k(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0), \quad \widetilde{\boldsymbol{r}}_k \perp \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0). \tag{7.65}$$

The initial shadow vector $\widetilde{\boldsymbol{r}}_0$ is arbitrary, as long as $\widetilde{\boldsymbol{r}}_j^{\mathrm{T}}\boldsymbol{r}_j \neq 0$. Note that there do exist choices of $\widetilde{\boldsymbol{r}}_0$ which can significantly affect convergence. The effect of choosing a different $\widetilde{\boldsymbol{r}}_0$ in the context of an algorithm related to BiCG (BiCGStab), was elaborated in Section 6. To start developing an algorithm for which the residuals satisfy (7.64) and (7.65), note that the residuals $\boldsymbol{r}_{i+1}$, with $i = 0, ..., k-1$, can be written as a polynomial

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_0 + \boldsymbol{A}\sum_{j=0}^{i} b_j^{(i)} \boldsymbol{A}^j \boldsymbol{r}_0. \tag{7.66}$$

Similarly the shadow residual $\widetilde{\boldsymbol{r}}_{i+1}$ can also be written as a polynomial

$$\widetilde{\boldsymbol{r}}_{i+1} = \widetilde{\boldsymbol{r}}_0 + \boldsymbol{A}^{\mathrm{T}}\sum_{j=0}^{i} b_j^{(i)} (\boldsymbol{A}^{\mathrm{T}})^j \widetilde{\boldsymbol{r}}_0. \tag{7.67}$$

The coefficients $b_j^{(i)}$ can be obtained from a linear system obtained by exploiting the orthogonality requirement $\boldsymbol{r}_{i+1}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}})^j\widetilde{\boldsymbol{r}}_0 = 0$ as given in Equation (7.65). Since $\boldsymbol{r}_k \perp \mathcal{K}_k(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0)$, $\boldsymbol{r}_{i+1}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}})^k\widetilde{\boldsymbol{r}}_0 = 0$ for $k \leq i$; one can left-multiply (7.66) by $\boldsymbol{r}_{i+1}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}})^k$ to obtain

$$\boldsymbol{r}_{i+1}^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}})^k\widetilde{\boldsymbol{r}}_0 = \boldsymbol{r}_0^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}})^k\widetilde{\boldsymbol{r}}_0 + \sum_{j=0}^{i} \boldsymbol{r}_0^{\mathrm{T}}(\boldsymbol{A}^{\mathrm{T}})^{j+1+k}\widetilde{\boldsymbol{r}}_0 b_j^{(i)} = \boldsymbol{0}. \tag{7.68}$$

Furthermore, since $\widetilde{\boldsymbol{r}}_k \perp \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$, $\widetilde{\boldsymbol{r}}_{i+1}^{\mathrm{T}}\boldsymbol{A}^k\boldsymbol{r}_0 = 0$ for $k \leq i$, we can also left-multiply (7.67) by $\widetilde{\boldsymbol{r}}_{i+1}^{\mathrm{T}}\boldsymbol{A}^k$ to obtain another relation for $b_j^{(i)}$,

$$\widetilde{\boldsymbol{r}}_{i+1}^{\mathrm{T}}\boldsymbol{A}^k\boldsymbol{r}_0 = \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^k\boldsymbol{r}_0 + \sum_{j=0}^{i} \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^{j+1+k}\boldsymbol{r}_0 b_j^{(i)} = \boldsymbol{0}. \tag{7.69}$$

Note that both (7.68) and (7.69) result in the same linear system for the expansion coefficients $b_j^{(i)}$, since $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{b} = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{a}$ for any $\boldsymbol{a}$ and $\boldsymbol{b}$. The linear system for the coefficients $b_j^{(i)}$ is given by

$$-\begin{bmatrix} \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{r}_0 \\ \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}\boldsymbol{r}_0 \\ \vdots \\ \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^i\boldsymbol{r}_0 \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}\boldsymbol{r}_0 & \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^2\boldsymbol{r}_0 & \dots & \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^{i+1}\boldsymbol{r}_0 \\ \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^2\boldsymbol{r}_0 & \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^3\boldsymbol{r}_0 & \dots & \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^{i+2}\boldsymbol{r}_0 \\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^{i+1}\boldsymbol{r}_0 & \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^{i+2}\boldsymbol{r}_0 & \dots & \widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{A}^{2i+1}\boldsymbol{r}_0 \end{bmatrix} \begin{bmatrix} b_0^{(i)} \\ b_1^{(i)} \\ \vdots \\ b_i^{(i)} \end{bmatrix}. \tag{7.70}$$

We do not need these coefficients $b_j^{(i)}$ explicitly, however, it is shown in [145] if the matrix in (7.70) is regular, then $\boldsymbol{r}_{i+1}$ exists and is unique, $\widetilde{\boldsymbol{r}}_{i+1}$ exists and is unique, $\boldsymbol{r}_j$ and $\widetilde{\boldsymbol{r}}_j$ are linearly independent, and the residuals span a Krylov subspace,

$$\operatorname{span}\{\boldsymbol{r}_0, \dots, \boldsymbol{r}_i\} = \mathcal{K}_{i+1}(\boldsymbol{A}, \boldsymbol{r}_0), \quad \operatorname{span}\{\widetilde{\boldsymbol{r}}_0, \dots, \widetilde{\boldsymbol{r}}_i\} = \mathcal{K}_{i+1}(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0). \tag{7.71}$$

Since the residuals span a Krylov subspace, the polynomials in (7.66) and (7.67) can be written in terms of a linear combination of the previous residuals;

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \boldsymbol{A}\sum_{j=0}^{i} \gamma_j^{(i)} \boldsymbol{r}_j, \quad \widetilde{\boldsymbol{r}}_{i+1} = \widetilde{\boldsymbol{r}}_i + \boldsymbol{A}^{\mathrm{T}}\sum_{j=0}^{i} \gamma_j^{(i)} \widetilde{\boldsymbol{r}}_j, \tag{7.72}$$

where the coefficients $\gamma_j^{(i)}$ may depend on i, which would require all coefficients to be recalculated every iteration. To avoid this potential issue these polynomials have to be slightly altered. The idea is to write the linear combinations (7.72) in a way similar to the Horner form of a polynomial;

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + \gamma_i^{(i)}\boldsymbol{A}\Big(\boldsymbol{r}_i + \frac{\gamma_{i-1}^{(i)}}{\gamma_i^{(i)}}\Big(\boldsymbol{r}_{i-1} + \frac{\gamma_{i-2}^{(i)}}{\gamma_{i-1}^{(i)}}\Big(\boldsymbol{r}_{i-2} + \dots\frac{\gamma_1^{(i)}}{\gamma_2^{(i)}}\Big(\boldsymbol{r}_1 + \frac{\gamma_0^{(i)}}{\gamma_1^{(i)}}\boldsymbol{r}_0\Big)\dots\Big)\Big)\Big). \tag{7.73}$$

It is shown in [145] that even though $\gamma_j^{(i)}$ depend on i, the ratios $\gamma_{j-1}^{(i)}/\gamma_j^{(i)}$ do not. Therefore, once we have computed $\gamma_{j-1}^{(i)}/\gamma_j^{(i)}$ for a given i, it remains valid for all future iterations. Furthermore, it can be shown that $\gamma_j^{(i)} \neq 0$ if both: $\widetilde{\boldsymbol{r}}_j^{\mathrm{T}}\boldsymbol{r}_j \neq 0$, and the matrix in equation (7.70) is regular.

Then by defining $\alpha_i = -\gamma_i^{(i)}$ and $\beta_{i-1} = \gamma_{i-1}/\gamma_i$ it becomes clear that $\boldsymbol{r}_{i+1}$ can be written as a recursive relation, since

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i\boldsymbol{A}(\boldsymbol{r}_i + \beta_{i-1}(\boldsymbol{r}_{i-1} + \beta_{i-2}(\boldsymbol{r}_{i-2} + \dots\beta_1(\boldsymbol{r}_1 + \beta_0\boldsymbol{r}_0)\dots))). \tag{7.74}$$

Additionally, the search direction $\boldsymbol{p}$ has to be identified, where $\boldsymbol{p}_0 = \boldsymbol{r}_0$ and $\boldsymbol{p}_{j+1} = \boldsymbol{r}_{j+1} + \beta_j\boldsymbol{p}_j$. By using the recursive expression for the search directions $\boldsymbol{p}_{i+1}$ we obtain the recurrence relation for the residual as

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i\boldsymbol{A}\boldsymbol{p}_i, \tag{7.75}$$

from which it follows that $\boldsymbol{x}_{i+1}$ is given by the recurrence

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i\boldsymbol{p}_i, \tag{7.76}$$

since $\boldsymbol{r}_{i+1} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{i+1}$. The recurrence for the search directions $\boldsymbol{p}_i$ is given by

$$\boldsymbol{p}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i\boldsymbol{p}_i. \tag{7.77}$$

Similarly, a recursive relation for the shadow residual can be obtained

$$\widetilde{\boldsymbol{r}}_{i+1} = \widetilde{\boldsymbol{r}}_i - \alpha_i\boldsymbol{A}^{\mathrm{T}}\widetilde{\boldsymbol{p}}_i, \tag{7.78}$$

where $\widetilde{\boldsymbol{p}}_0 = \widetilde{\boldsymbol{r}}_0$, and the recurrence for $\widetilde{\boldsymbol{p}}$ is given by

$$\widetilde{\boldsymbol{p}}_{j+1} = \widetilde{\boldsymbol{r}}_{j+1} + \beta_j\widetilde{\boldsymbol{p}}_j. \tag{7.79}$$

Note that a “shadow solution” $\widetilde{\boldsymbol{x}}_{i+1}$ recurrence can also be obtained

$$\widetilde{\boldsymbol{x}}_{i+1} = \widetilde{\boldsymbol{x}}_i + \alpha_i \widetilde{\boldsymbol{p}}_i, \tag{7.80}$$

where $\boldsymbol{A}^{\mathrm{T}} \widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}}$ and $\widetilde{\boldsymbol{r}}_0 = \widetilde{\boldsymbol{b}} - \boldsymbol{A}^{\mathrm{T}} \widetilde{\boldsymbol{x}}_0$, thus BiCG can be seen as solving both $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{A}^{\mathrm{T}} \widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}}$ simultaneously[32, p. 247].

The next step is to derive a relation for the coefficients α_i and β_i. To obtain a relation for α_i we left-multiply equation (7.75) with $\widetilde{\boldsymbol{p}}_i^{\mathrm{T}}$ to obtain

$$\widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{r}_{i+1} = \widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{r}_i - \alpha_i \widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{A} \boldsymbol{p}_i, \tag{7.81}$$

then as $\widetilde{\boldsymbol{p}}_i \in \mathcal{K}_i(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0)$, and $\boldsymbol{r}_{i+1} \perp \mathcal{K}_i(\boldsymbol{A}^{\mathrm{T}}, \widetilde{\boldsymbol{r}}_0)$ it follows that $\widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{r}_{i+1} = 0$, and therefore α_i is given by

$$\alpha_i = \frac{\widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{r}_i}{\widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{A} \boldsymbol{p}_i}. \tag{7.82}$$

Finally, it can be shown from equation (7.79), and $\widetilde{\boldsymbol{p}}_{i-i}^{\mathrm{T}} \boldsymbol{r}_i = 0$ that $\widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{r}_i = \widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{r}_i$, after which the final form of α_i becomes

$$\alpha_i = \frac{\widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{r}_i}{\widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{A} \boldsymbol{p}_i}. \tag{7.83}$$

To derive the relation for β_i, we start with (7.77), and left-multiply by $\widetilde{\boldsymbol{r}}_i^{\mathrm{T}}$. Then by using $\widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{r}_{i+1} = 0$, $\widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{p}_i = \widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{r}_i$, we obtain

$$\beta_i = \frac{\widetilde{\boldsymbol{r}}_{i+1}^{\mathrm{T}} \boldsymbol{r}_{i+1}}{\widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{r}_i}, \tag{7.84}$$

where we used that $\widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{p}_{i+1} = \widetilde{\boldsymbol{r}}_{i+1}^{\mathrm{T}} \boldsymbol{r}_{i+1}$, since

$$\begin{aligned} \widetilde{\boldsymbol{r}}_{i+1}^{\mathrm{T}} \boldsymbol{r}_{i+1} &= \widetilde{\boldsymbol{r}}_{i+1}^{\mathrm{T}} \boldsymbol{p}_{i+1} \\ &= \widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{p}_{i+1} - \alpha_i \widetilde{\boldsymbol{p}}_i^{\mathrm{T}} \boldsymbol{A} \boldsymbol{p}_{i+1} \\ &= \widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{p}_{i+1} + \frac{\alpha_i}{\alpha_{i+1}} \widetilde{\boldsymbol{p}}_i^{\mathrm{T}} (\boldsymbol{r}_{i+2} - \boldsymbol{r}_{i+1}) \\ &= \widetilde{\boldsymbol{r}}_i^{\mathrm{T}} \boldsymbol{p}_{i+1}, \end{aligned} \tag{7.85}$$

where in the second to last step we use that $\boldsymbol{A}\boldsymbol{p}_{i+1} = \frac{1}{\alpha_{i+1}}(\boldsymbol{r}_{i+2} - \boldsymbol{r}_{i+1})$, which can be obtained from (7.75). This completes the BiCG algorithm as stated in Algorithm 10.

Note that Algorithm 10 can be modified to solve $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ and $\boldsymbol{A}^{\mathrm{T}} \widetilde{\boldsymbol{x}} = \widetilde{\boldsymbol{b}}$ simultaneously. However, also note that if the solution to the transposed system is not required, BiCG nevertheless requires two MV per iteration. Furthermore, if $\widetilde{\boldsymbol{r}}_0 = \boldsymbol{r}_0$ and $\boldsymbol{A}$ is SPD, then BiCG produces the same sequence of $\boldsymbol{x}_k$ as CG, albeit at twice the number of MV.

Algorithm 10 Bi-Conjugate Gradient method

```
function BiCG($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}_0, tol$)
    $\boldsymbol{r}_0 \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$,
    Choose $\widetilde{\boldsymbol{r}}_0$ such that $\widetilde{\boldsymbol{r}}_0^{\mathrm{T}}\boldsymbol{r}_0 \neq 0$
    $\boldsymbol{p}_0 \leftarrow \boldsymbol{r}_0$, $\widetilde{\boldsymbol{p}}_0 \leftarrow \widetilde{\boldsymbol{r}}_0$
    $k \leftarrow 0$
    while $\|\boldsymbol{r}_k\|_2 > tol$ do
        $\alpha_k \leftarrow \frac{\boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{r}_k}{\widetilde{\boldsymbol{p}}_k^{\mathrm{T}}\boldsymbol{A}\boldsymbol{p}_k}$
        $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k\boldsymbol{p}_k$
        $\boldsymbol{r}_{k+1} \leftarrow \boldsymbol{r}_k - \alpha_k\boldsymbol{A}\boldsymbol{p}_k$
        $\widetilde{\boldsymbol{r}}_{k+1} \leftarrow \widetilde{\boldsymbol{r}}_k - \alpha_k\boldsymbol{A}^{\mathrm{T}}\widetilde{\boldsymbol{p}}_k$
        $\beta_k = \frac{\widetilde{\boldsymbol{r}}_{k+1}^{\mathrm{T}}\boldsymbol{r}_{k+1}}{\widetilde{\boldsymbol{r}}_k^{\mathrm{T}}\boldsymbol{r}_k}$
        $\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{r}_{k+1} + \beta_k\boldsymbol{p}_k$
        $\widetilde{\boldsymbol{p}}_{k+1} \leftarrow \widetilde{\boldsymbol{r}}_{k+1} + \beta_k\widetilde{\boldsymbol{p}}_k$
        $k \leftarrow k + 1$
    end while
    return $\boldsymbol{x}$
end function
```

7.5 Bi-Conjugate Gradient Stabilized (BiCGStab)

Two of the shortcomings of the BiCG algorithm are the requirement of multiplications with $\boldsymbol{A}^{\mathrm{T}}$, and the relative inefficiency of solving two systems simultaneously. The requirement of having $\boldsymbol{A}^{\mathrm{T}}$ is that BiCG may not be usable for so-called "matrix-free" applications, where often $\boldsymbol{A}^{\mathrm{T}}$ may be excessively expensive to obtain. One such application was presented in Section 5.2, where Krylov methods are used to solve the linear system obtained from Newton's method. Furthermore, unlike LMR, the residuals produced by BiCG can increase in size.

An alternative to the BiCG algorithm that does not require $\boldsymbol{A}^{\mathrm{T}}$ is the widely used Bi-Conjugate Gradient *Stabilized* (BiCGStab) method. This method aims to provide both a smoother convergence compared to BiCG, and does not require MV involving $\boldsymbol{A}^{\mathrm{T}}$ [32, p. 244]. BiCGStab combines the idea of building orthogonal spaces presented in BiCG, with the smooth convergence provided by LMR. First a BiCG step is computed, then an LMR step.

The BiCGStab algorithm given here is based on the baseline BiCGStab algorithm given in Algorithm 3. However, for the sake of clarity, it has been simplified, and additional comments have been added. It is important to note that there is a wide variety of different implementations of BiCGStab

[38, 112, 114, 113, 111, 141, 142], even though they should produce the same sequence of residuals in theory, in finite precision arithmetic these may not be identical.

Note that even though BiCGStab looks relatively straightforward to implement, there is a variety of edge cases and pitfalls that have to be taken into account, namely:

1. When dealing with matrices that have imaginary eigenvalues, BiCGStab breaks down as $\omega = 0$ leads to a division by zero on line 9.

2. Some choices of $\widetilde{\boldsymbol{r}}$ lead to $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle \to 0$ on line 8, which in turn leads to $\beta \to 0$, $\boldsymbol{p} \to \boldsymbol{r}$. It has been shown in Chapter 6 that for a class of matrices $\boldsymbol{A}$ this also leads to $\alpha \to 0$.

3. There can be an accumulation of round-off errors, since $\boldsymbol{x}$ is never explicitly used to calculate the residual $\boldsymbol{r}$, except on line 3.

4. If BiCGStab converges in the BiCG part, that is $\boldsymbol{r} = \boldsymbol{0}$ on line 13, then there is a division by zero on line 15.

5. If $\boldsymbol{b} = \boldsymbol{0}$ the solution to $\boldsymbol{Ax} = \boldsymbol{b}$ is trivial, nevertheless if $\boldsymbol{x}_0 \neq \boldsymbol{0}$ BiCGStab can get stuck in an infinite loop, due to rounding errors, $\|\boldsymbol{r}\|$ may never reach 0 exactly. This results in the criterion on line 5 to never be false, and thus results in an infinite loop.

6. There are several possible optimizations, for example caching of the vectors $\boldsymbol{Ap}$ and $\boldsymbol{Ar}$, or skipping the update on line 18 if $\|\boldsymbol{r}\| \leq tol\|\boldsymbol{b}\|$ after line 17. Furthermore the two updates of $\boldsymbol{x}$ on lines 12 and 16 can be combined by introducing a temporary vector variable.

7. Another implementation detail is the argmin calculation, which can be done by solving a system of normal equations by a variety of different factorizations, or via one of several different methods to compute a QR decomposition.

Implementation aspects like the above, and others are discussed in more detail in Chapter 8, where we present our C++ implementations of the Krylov subspace methods IDR(S), BiCGStab(L) and IDR(S)Stab(L).

Algorithm 11 Bi-Conjugate Gradient Stabilized (BiCGStab) method

```
function BiCGStab($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol$)
    $\rho_0 \leftarrow 1, \alpha \leftarrow 1, \omega \leftarrow 1, \boldsymbol{p} \leftarrow \boldsymbol{0}$
    $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
    $\widetilde{\boldsymbol{r}} \leftarrow \boldsymbol{r}$                                ▷ Arbitrary, such that $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle \neq 0$
    while $\|\boldsymbol{r}\| > tol\|\boldsymbol{b}\|$ do
                                                   ▷ BiCG part
        $\rho_0 \leftarrow -\rho_0\omega$
        $\rho_1 \leftarrow \langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle$
        $\beta \leftarrow \alpha(\rho_1/\rho_0), \quad \rho_0 \leftarrow \rho_1$
        $\boldsymbol{p} \leftarrow \boldsymbol{r} - \beta\boldsymbol{p}$
        $\alpha \leftarrow \rho_1/\langle \widetilde{\boldsymbol{r}}, \boldsymbol{A}\boldsymbol{p} \rangle$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\boldsymbol{p}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - \alpha\boldsymbol{A}\boldsymbol{p}$
                                                   ▷ LMR part
        $\omega \leftarrow \mathrm{argmin}_{\omega} \|\boldsymbol{r} - \omega\boldsymbol{A}\boldsymbol{r}\| = \langle \boldsymbol{A}\boldsymbol{r}, \boldsymbol{r} \rangle / \langle \boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}\boldsymbol{r} \rangle$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + \omega\boldsymbol{r}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - \omega\boldsymbol{A}\boldsymbol{r}$
        $\boldsymbol{p} \leftarrow \boldsymbol{p} - \omega\boldsymbol{A}\boldsymbol{p}$                  ▷ Update $\boldsymbol{p}$ for the next BiCG step
    end while
    return $\boldsymbol{x}$
end function
```

Chapter 8

Implementation of three new Krylov solvers for the Eigen library

The Eigen C++ library is a template library for linear algebra. Eigen is used in a wide variety of other projects, at the time of writing 90 projects are listed on the main page of the project [40], including simulation toolboxes, machine learning frameworks, computer vision and several mobile applications.

Eigen also provides several methods for solving linear systems, including direct methods such as the well-known Cholesky, QR and LU decompositions. In the context of this thesis we are interested in iterative solvers. Such solvers are also included in Eigen, specifically the following Krylov subspace methods are included in Eigen:

1. CG
2. BiCGStab
3. Least squares CG
4. Unsupported: DGMRES
5. Unsupported: GMRES
6. Unsupported: MinRes
7. Our contribution: BiCGStab(L)
8. Our contribution: IDR(S)
9. Our contribution: IDR(S)Stab(L)

In Chapter 7 we described the main concepts of the underlying linear algebra, and several pitfalls of Krylov subspace methods. In this chapter we discuss our contributed Krylov subspace methods, and implementation choices. To the best of our knowledge, our implementation of IDR(S)Stab(L) is the first and only open-source C++ version.

8.1 Bi-Conjugate Gradient Stabilized(L) (BiCGStab(L))

BiCGStab(L) extends the BiCGStab algorithm to incorporate a higher order LMR step. One of the main motivations of adding a higher order LMR step, is that a first-order LMR step can lead to stagnation if the matrix has eigenvalues with a large imaginary part [138].

To do this, BiCGStab(L) performs L steps of BiCG, and applies LMR(L) to the residuals generated by the BiCG part. One possible implementation of BiCGStab(L) is given in Algorithm 12. However, there are a variety of different implementations and extensions to BiCGStab(L) [45, 109, 48, 115, 109]. In this section we discuss the choices made in our Eigen-implementation of BiCGStab(L) [49].

Algorithm 12 BiCGStab(L) adapted from reference [47]. $\boldsymbol{R} = [\boldsymbol{R}, \boldsymbol{a}]$ concatenates a column vector $\boldsymbol{a}$ to the right side of $\boldsymbol{R}$. Subscripts of matrices indicate indices of a column, starting at 0. Ranges of indices are denoted with a semicolon (:). For example, $\boldsymbol{R}_{1:3}$ is a matrix consisting of columns 1, 2 and 3 of $\boldsymbol{R}$. Note that the sizes of the matrices $\boldsymbol{R}$ and $\boldsymbol{U}$ are not constant.

```
 1: function BiCGStab($L$)($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol, L$)
 2:     $\boldsymbol{R} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
 3:     $\widetilde{\boldsymbol{r}} \leftarrow \boldsymbol{R}_0$
 4:     $\boldsymbol{U}_{0:1} \leftarrow [\boldsymbol{R}_0, \boldsymbol{A}\boldsymbol{R}_0]$
 5:     while $\|\boldsymbol{R}_0\| > tol$ do
 6:         for $j = 1, ..., L$ do
 7:                                                        ▷ The BiCG step
 8:             $\sigma \leftarrow \widetilde{\boldsymbol{r}}^{\mathrm{T}}\boldsymbol{U}_j$
 9:             $\alpha \leftarrow \sigma^{-1}\widetilde{\boldsymbol{r}}^{\mathrm{T}}\boldsymbol{R}_{j-1}$
10:             $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\boldsymbol{U}_0$
11:             $\boldsymbol{R} \leftarrow \boldsymbol{R} - \alpha\boldsymbol{U}_{1:j}$
12:             $\boldsymbol{R} \leftarrow [\boldsymbol{R}, \boldsymbol{A}\boldsymbol{R}_{j-1}]$
13:             $\beta \leftarrow \sigma^{-1}\widetilde{\boldsymbol{r}}^{\mathrm{T}}\boldsymbol{R}_j$
14:             $\boldsymbol{U} \leftarrow \boldsymbol{R} - \beta\boldsymbol{U}$
15:             $\boldsymbol{U} \leftarrow [\boldsymbol{U}, \boldsymbol{A}\boldsymbol{U}_j]$
16:         end for
17:                                   ▷ The polynomial step (LMR step)
18:         $\boldsymbol{\gamma} \leftarrow \operatorname{argmin}_{\boldsymbol{\gamma}} \|\boldsymbol{R}_0 - \boldsymbol{R}_{1:L}\boldsymbol{\gamma}\|_2$
19:         $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{R}_{0:(L-1)}\boldsymbol{\gamma}$
20:         $\boldsymbol{R} \leftarrow \boldsymbol{R}_0 - \boldsymbol{R}_{1:L}\boldsymbol{\gamma}$
21:         $\boldsymbol{U} \leftarrow [\boldsymbol{U}_0 - \boldsymbol{U}_{1:L}\boldsymbol{\gamma}, \boldsymbol{U}_1 - \boldsymbol{U}_{2:(L+1)}\boldsymbol{\gamma}]$
22:     end while
23: end function
```

8.1.1 Choice of argmin method

There are a variety of methods available to perform the argmin step in line 18 of Algorithm 12. In effect, this step comes down to solving the following least squares problem for $\boldsymbol{\gamma}$, i.e.,

$$\boldsymbol{R}_0 = \boldsymbol{R}_{1:L}\boldsymbol{\gamma}. \tag{8.1}$$

Eigen provides ten algorithms to do this, four of which solve the normal equations using a decomposition of $\boldsymbol{R}_{1:L}^{\mathrm{T}}\boldsymbol{R}_{1:L}$:

1. `PartialPivLU`, LU decomposition with partial pivoting.
2. `FullPivLU`, LU decomposition with full pivoting.
3. `LLT`, standard Cholesky decomposition.

4. `LDLT`, robuster version of the Cholesky decomposition that includes pivoting.

and six perform some decomposition without forming the normal equations:

1. `HouseholderQR`, QR decomposition using Householder transformations
2. `ColPivHouseholderQR`, QR decomposition using Householder transformations, including possible column swaps.
3. `FullPivHouseholderQR`, QR decomposition using Householder transformations, including possible column and row swaps.
4. `CompleteOrthogonalDecomposition`, computes a decomposition of a matrix $\boldsymbol{A}$ as $\boldsymbol{AP} = \boldsymbol{QTZ}$ where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{Q}$ and $\boldsymbol{Z}$ are unitary and $\boldsymbol{T}$ is an upper triangular matrix.
5. `BDCSVD`, computes the Singular Value Decomposition (SVD) using a "bidiagonal divide and conquer" method.
6. `JacobiSVD`, computes the SVD using the Jacobi method.

To eliminate some of the available methods, note that the normal equations result in a linear system with a symmetric, positive semi-definite matrix. Therefore we can rule out `PartialPivLU` and `FullPivLU`, since they do not exploit the symmetry of the resulting matrix. Furthermore, it is stated in the documentation [146] that `LLT` is not stable for the semi-definite case. This only leaves `LDLT` as an option out of the algorithms that use the normal equations.

It is expected that the least squares problem is "thin", i.e., the number of rows of $\boldsymbol{R}_{1:L}$ far exceeds the number of columns. A benchmark of decomposition methods in Eigen [147] shows that for a test problem with 10,000 rows and 8 columns the computation times are as given in Table 8.1.1. It can be seen that the methods that solve the normal equations are the fastest, followed by QR methods, and finally the SVD methods.

It is shown in [109] that solving the argmin step accurately, can also reduce the number of MV needed. To avoid issues such as badly conditioned normal equations, and still maintain good performance of the argmin step, we choose the `HouseholderQR` method. Since it is the cheapest of the methods that do not solve the normal equations. Providing a balance between saving MV, and cost to solve the argmin step.

8.1.2 Choice of the shadow residual vector

We have shown in Chapter 6 that a random $\widetilde{\boldsymbol{r}}$ avoids some stagnation problems in BiCGStab. Similarly to BiCGStab, the choice of $\widetilde{\boldsymbol{r}}$ is arbitrary in BiCGStab(L) as

Table 8.1.1: Execution times to solve a $10,000 \times 8$ least squares problem. For methods that solve the normal equations this includes the time needed to set up the system of normal equations.

Algorithm	Time [ms]
LLT	6.79
LDLT	6.81
PartialPivLU	6.81
FullPivLU	6.83
HouseholderQR	34.26
ColPivHouseholderQR	36.05
CompleteOrthogonalDecomposition	35.75
FullPivHouseholderQR	69.38
JacobiSVD	113.81
BDCSVD	110.53

well. However, $\widetilde{\boldsymbol{r}}$ must be chosen such that it is not orthogonal to any intermediate residual $\boldsymbol{U}_j$, otherwise this would lead to a division by zero on line 9 of Algorithm 12.

To avoid problems related to certain $\widetilde{\boldsymbol{r}}$, as demonstrated in Chapter 6, $\widetilde{\boldsymbol{r}}$ is chosen at random in our implementation of BiCGStab(L). More specifically, each element of $\widetilde{\boldsymbol{r}}$ is chosen from a uniform random distribution in $[-1, 1]$; this is the default when generating random numbers using Eigen's random number generator. In Chapter 6 the elements were drawn randomly from $(0, 1)$, the default distribution used by MATLAB. To our knowledge, there is no known advantage of choosing either distribution, or if there are other distributions that would be preferable.

It is important to note that the vector $\widetilde{\boldsymbol{r}}$ is not normalized, unlike in our IDR(S) [50] and IDR(S)Stab(L) [51] implementations. To argue the benefit of not normalizing $\widetilde{\boldsymbol{r}}$, we argue that in high dimensions the inner product of two randomly chosen normalized vectors tends to zero, whereas the inner product of two randomly chosen non-normalized vectors does not.

To show this, consider two randomly chosen vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ of size N, with elements from a uniform random real distribution over the range $[-1, 1]$. We also define $\widehat{\boldsymbol{a}} := \boldsymbol{a}/\|\boldsymbol{a}\|$ and $\widehat{\boldsymbol{b}} := \boldsymbol{b}/\|\boldsymbol{b}\|$. The 2-norm squared of $\boldsymbol{a}$ is given by

$$\|\boldsymbol{a}\|^2 = \sum_{i=1}^{N} a_i^2, \tag{8.2}$$

then by using equation (5.27) from [148, p. 258], i.e.,

$$\mathrm{E}\left[a_1 + a_2 + ... + a_N\right] = \mathrm{E}\left[a_1\right] + \mathrm{E}\left[a_2\right] + ... + \mathrm{E}\left[a_N\right], \tag{8.3}$$

and computing the mean of a uniform random variable as in [148, p. 156], it follows that the expected value of the squared norm is

$$\|\boldsymbol{a}\|^2 = \mathrm{E}\left[\sum_{i=1}^{N} a_i^2\right] = \sum_{i=1}^{N} \mathrm{E}\left[a_i^2\right] = N\int_{-1}^{1} \frac{x^2}{2}\mathrm{d}x = \frac{N}{3}. \tag{8.4}$$

The expected value $\mathrm{E}\left[\langle \boldsymbol{a}, \boldsymbol{b}\rangle\right] = 0$, since $\mathrm{E}\left[a_i\right] = \mathrm{E}\left[b_i\right] = 0$ and

$$\mathrm{E}\left[a_i b_i\right] = \mathrm{E}\left[a_i\right]\mathrm{E}\left[b_i\right] = 0, \tag{8.5}$$

where it is used that a_i and b_i are independent [148, p. 258]. Even though the expected value of $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ is zero, it can be shown that the variance of $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ *increases* with N, whereas the variance of $\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}\rangle$ *decreases* with N. The variance of $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ is given as

$$\begin{aligned} \mathrm{var}(\langle \boldsymbol{a}, \boldsymbol{b}\rangle) = \mathrm{var}\left(\sum_{i=1}^{N} a_i b_i\right) = \sum_{i=1}^{N} \mathrm{var}(a_i b_i) \\ = \sum_{i=1}^{N} \mathrm{E}\left[(a_i b_i - \mathrm{E}\left[a_i b_i\right])^2\right] = \sum_{i=1}^{N} \mathrm{E}\left[(a_i b_i)^2\right] = \sum_{i=1}^{N} \mathrm{E}\left[a_i^2\right]\mathrm{E}\left[b_i^2\right] = \frac{N}{9}, \end{aligned} \tag{8.6}$$

where it was used that a_i and b_i are independent to take the sum out of the variance function. To show that the variance of $\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}\rangle$ decreases with N, we use the property that [148, p. 110]

$$\mathrm{var}(\alpha x) = \alpha^2 \mathrm{var}(x), \tag{8.7}$$

where α is a constant and x a random variable, to argue that

$$\mathrm{var}(\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}}\rangle) = \mathrm{var}\left(\frac{\langle \boldsymbol{a}, \boldsymbol{b}\rangle}{\|\boldsymbol{a}\|\|\boldsymbol{b}\|}\right) \approx \frac{\mathrm{var}(\langle \boldsymbol{a}, \boldsymbol{b}\rangle)}{\mathrm{E}\left[\, \|\boldsymbol{a}\|\|\boldsymbol{b}\| \,\right]^2} = \frac{\mathrm{var}(\langle \boldsymbol{a}, \boldsymbol{b}\rangle)}{(N^2/9)} = \frac{9}{N}, \tag{8.8}$$

and thus decreases with N.

The probability density function for a normally distributed stochastic variable X with mean μ and variance σ^2 is given by the Gaussian,

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\tfrac{1}{2}\Big(\frac{x-\mu}{\sigma}\Big)^2\Big), \tag{8.9}$$

and the probability $\mathrm{P}(|x-\mu| \leq \epsilon)$ with $\epsilon > 0$ can be obtained by integrating

$$\mathrm{P}(|x-\mu| \leq \epsilon) = \int_{\mu-\epsilon}^{\mu+\epsilon} p(x)\mathrm{d}x = \operatorname{erf}\left(\frac{\epsilon}{\sqrt{2}\sigma}\right), \tag{8.10}$$

where $\operatorname{erf}(z)$ is the error function. Assuming the distributions produced by $\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle$ and $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ are normal, and since $\operatorname{erf}(z)$ monotonically increases from 0 to 1 over $z \in [0, \infty)$, we conclude that

$$\lim_{N\to\infty} \mathrm{P}(|\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle| \leq \epsilon) = 1, \tag{8.11}$$

i.e., for sufficiently large N the absolute value of the inner product $|\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle|$ is *smaller* than ϵ with high probability. However, in the non-normalized case

$$\lim_{N\to\infty} \mathrm{P}(|\langle \boldsymbol{a}, \boldsymbol{b} \rangle| \leq \epsilon) = 0, \tag{8.12}$$

and therefore for sufficiently large N the absolute value of the inner product $|\langle \boldsymbol{a}, \boldsymbol{b} \rangle|$ is *larger* than ϵ with high probability as $\mathrm{P}(|\langle \boldsymbol{a}, \boldsymbol{b} \rangle| > \epsilon) = 1 - \mathrm{P}(|\langle \boldsymbol{a}, \boldsymbol{b} \rangle| < \epsilon)$. These findings have also been tested in a numerical experiment, the results of which are shown in Figure 8.1.1. In this numerical experiment for each N, 10^3 random vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ were generated. Subsequently, the mean and variance of $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ and $\langle \widehat{\boldsymbol{a}}, \widehat{\boldsymbol{b}} \rangle$ were calculated for each N.

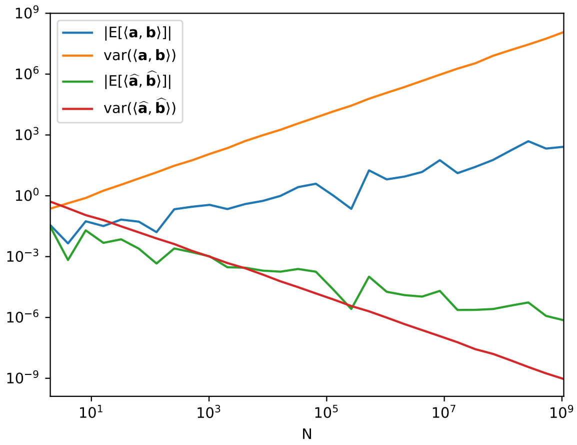


Figure 8.1.1: Plot of the expected value and variance for both normalized and non-normalized inner products.

Since nothing is known a priori about $\boldsymbol{r}$, apart from the size N, we choose to not normalize the random vector $\widetilde{\boldsymbol{r}}$ in order to avoid the issue of vanishing inner product between normalized vectors.

In principle one can pick any $\widetilde{\boldsymbol{r}}$, and compute the inner product with $\boldsymbol{r}$ to check if $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle \neq 0$. It is possible to restart if $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle$ is too small, as is done in the Eigen implementation of BiCGStab. However, by choosing non-normalized $\widetilde{\boldsymbol{r}}$, the inner product $|\langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle|$ is unlikely to be small, especially for large N. Therefore, we opted to keep this part of the implementation simple and only implement a random $\widetilde{\boldsymbol{r}}$, without further checks.

8.1.3 Reliable computation of the residual

Iterative methods such as BiCGStab(L) aim to minimize the residual

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}. \tag{8.13}$$

To save on the number of MV, the residual is typically not computed using (8.13), but rather from a recursive relation of the form

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{A}\boldsymbol{p}_k, \tag{8.14}$$

which does not require an extra MV as the MV $\boldsymbol{A}\boldsymbol{p}_k$ is used in multiple steps, for example in BiCGStab Algorithm 11 on lines 11, 13 and 18. In exact arithmetic the recursively computed residual (8.14) and the true residual (8.13) are equal. However, in finite precision arithmetic there may be a significant discrepancy between the two methods of computing the residual as we will argue shortly, and was demonstrated numerically in Chapter 6.

Importantly, the norm of the residual is typically used as a stopping criterion. If the true and updated residuals drift apart, then the method may stop prematurely, or perform unnecessary iterations. If the method stops prematurely, it is an option to restart the solver. However, restarting also means the super linear rate of convergence may be lost [116].

In reference [116] methods are provided to improve the accuracy of the computed residual, without losing speed of convergence. The idea is to:

1. Compute the true residual on some strategically chosen iterations.
2. Accumulate updates to $\boldsymbol{x}$ by using “group-wise” updates.
3. Perform “flying restarts” to maintain speed of convergence when restarting.

An upper bound on the residual gap, the difference between the true and recursively computed residual, can be obtained. Iterative methods typically use two recurrences,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{p}_k \quad \boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{A}\boldsymbol{p}_k. \tag{8.15}$$

Even if $\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k$ exactly, in finite precision arithmetic there is no guarantee that $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{k+1} = \boldsymbol{r}_{k+1}$. Here we investigate the arithmetic error introduced by the MV $\boldsymbol{A}\boldsymbol{p}_k$, given as $\Delta_{\mathrm{A}}\boldsymbol{p}_k$. Following [116], if we consider an error of the form

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \boldsymbol{A}\boldsymbol{p}_k - \Delta_{\mathrm{A}}\boldsymbol{p}_k, \quad \|\Delta_{\mathrm{A}}\| \leq n_{\mathrm{A}}\epsilon_{\mathrm{m}}\||\boldsymbol{A}|\|, \tag{8.16}$$

where n_{A} is the maximum number of nonzeros in a row of $\boldsymbol{A}$, and ϵ_{m} is the machine precision, $\epsilon_{\mathrm{m}} \approx 10^{-16}$ for double-precision floating point numbers. Furthermore, if we only consider errors of the form (8.16), then the upper bound for residual gap over k iterations can be estimated by equation (6) in [116];

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{r}_k\| \leq 2n_{\mathrm{A}}\epsilon_{\mathrm{m}}\||\boldsymbol{A}|\|\|\boldsymbol{A}^{-1}\| \sum_{j \leq k} \|\boldsymbol{r}_j\|. \tag{8.17}$$

Even though (8.17) only provides an upper bound on the residual gap, we can make several observations;

1. The residual gap is expected to increase with the number of nonzeros per row.
2. If all entries of $\boldsymbol{A}$ are positive, then the upper bound is proportional to the condition number $\|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$.
3. Large residuals increase the upper bound.
4. If convergence is slow, the upper bound is larger due to the sum over the residuals.
5. As expected, the residual gap vanishes in infinite precision arithmetic where $\epsilon_{\mathrm{m}} = 0$. This shows the residual gap is a numerical artifact.

The version of BiCGStab(L) with shift, group-wise updating and reliable updating is shown in Algorithm 13.

8.1.4 Right preconditioning

A few types of preconditioning can be implemented, most notably left-preconditioning and right-preconditioning. There do exist preconditioning schemes such as split preconditioning which utilize an incomplete LU-factorization of $\boldsymbol{A}$ [32, p. 285]. However, in our implementation we have to support a generic preconditioner. More specifically, the Eigen-API provides a function

Algorithm 13 BiCGStab(L) adapted from reference [47]. $\boldsymbol{R} = [\boldsymbol{R}, \boldsymbol{a}]$ concatenates a column vector $\boldsymbol{a}$ to $\boldsymbol{R}$. Subscripts of matrices indicate indices of a column, starting at 0, $\boldsymbol{R}_{1:3}$ is a matrix consisting of columns 1, 2 and 3 of $\boldsymbol{R}$. Note that the sizes of the matrices $\boldsymbol{R}$ and $\boldsymbol{U}$ are not constant. Here both the shift and the reliable updating scheme have been added to the reference implementation.

```
function BiCGStab($L$)($\boldsymbol{A}, \widehat{\boldsymbol{b}}, \boldsymbol{x}, tol, L$)
    $\boldsymbol{x} \leftarrow \boldsymbol{x}_0, \quad \boldsymbol{b} \leftarrow \widehat{\boldsymbol{b}} - \boldsymbol{A}\boldsymbol{x}_0$
    $\boldsymbol{y} \leftarrow \boldsymbol{0}, \quad \boldsymbol{R} \leftarrow \boldsymbol{b}, \quad \widetilde{\boldsymbol{r}} \leftarrow \boldsymbol{R}_0$
    $\boldsymbol{U}_{0:1} \leftarrow [\boldsymbol{R}_0, \boldsymbol{A}\boldsymbol{R}_0]$
    while $\|\boldsymbol{R}_0\| > tol$ do
        for $j = 1, ..., L$ do
                                                        ▷ The BiCG step
            $\sigma \leftarrow \widetilde{\boldsymbol{r}}^{\mathrm{T}}\boldsymbol{U}_j$
            $\alpha \leftarrow \sigma^{-1}\widetilde{\boldsymbol{r}}^{\mathrm{T}}\boldsymbol{R}_{j-1}$
            $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\boldsymbol{U}_0$
            $\boldsymbol{R} \leftarrow \boldsymbol{R} - \alpha\boldsymbol{U}_{1:j}$
            $\boldsymbol{R} \leftarrow [\boldsymbol{R}, \boldsymbol{A}\boldsymbol{R}_{j-1}]$
            $\beta \leftarrow \sigma^{-1}\widetilde{\boldsymbol{r}}^{\mathrm{T}}\boldsymbol{R}_j$
            $\boldsymbol{U} \leftarrow \boldsymbol{R} - \beta\boldsymbol{U}$
            $\boldsymbol{U} \leftarrow [\boldsymbol{U}, \boldsymbol{A}\boldsymbol{U}_j]$
        end for
                                        ▷ The polynomial step (LMR step)
        $\boldsymbol{\gamma} \leftarrow \mathrm{argmin}_{\boldsymbol{\gamma}} \|\boldsymbol{R}_0 - \boldsymbol{R}_{1:L}\boldsymbol{\gamma}\|_2$
        $\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{R}_{0:(L-1)}\boldsymbol{\gamma}$
        $\boldsymbol{R} \leftarrow \boldsymbol{R}_0 - \boldsymbol{R}_{1:L}\boldsymbol{\gamma}$
        $\boldsymbol{U} \leftarrow [\boldsymbol{U}_0 - \boldsymbol{U}_{1:L}\boldsymbol{\gamma}, \boldsymbol{U}_1 - \boldsymbol{U}_{2:(L+1)}\boldsymbol{\gamma}]$
                                                ▷ Reliable updating part
        if $\|\boldsymbol{r}\| \leq 0.01\|\widehat{\boldsymbol{b}}\|$ and $\|\widehat{\boldsymbol{b}}\| \leq \max_j \|\boldsymbol{r}_j\|$ then
            $flying\_restart \leftarrow true$
        end if
        if $(\|\boldsymbol{r} < 0.01 \max_j \|\boldsymbol{r}_j\|\|$ and $\|\widehat{\boldsymbol{b}}\| \leq \max_j \|\boldsymbol{r}_j\|)$ then
            $compute\_residual \leftarrow true$
        end if
        if $compute\_residual$ or $flying\_restart$ then
            $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{y}$              ▷ Compute the true residual.
        end if
        if $flying\_restart$ then
            $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{y}, \quad \boldsymbol{y} \leftarrow \boldsymbol{0}, \quad \boldsymbol{b} \leftarrow \boldsymbol{r}$   ▷ Add accumulated updates into $\boldsymbol{x}$ and shift the problem.
        end if
        $compute\_residual \leftarrow false, \quad flying\_restart \leftarrow false$
    end while
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{y}$                          ▷ Add accumulated updates into $\boldsymbol{x}$.
end function
```

`preconditioner.solve(y)` which has the effect of applying a preconditioner to a vector $\boldsymbol{y}$. In terms of linear algebra this can be modeled as applying a matrix $\boldsymbol{M}^{-1}\boldsymbol{y}$. The specific preconditioner function used is completely determined by the user, and may use $\boldsymbol{y}$ in a complicated way, for example a multigrid preconditioner.

Left-preconditioning replaces the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by left-multiplying both sides with $\boldsymbol{M}^{-1}$,

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}, \tag{8.18}$$

the residual of the preconditioned system is then given by

$$\boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k). \tag{8.19}$$

Alternatively, right-preconditioning uses that $\boldsymbol{M}^{-1}\boldsymbol{M} = \boldsymbol{I}$, and works on the system

$$\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{x} = \boldsymbol{b}, \tag{8.20}$$

the residual of this preconditioned system is the same as the unpreconditioned system, since

$$\boldsymbol{b} - \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{M}\boldsymbol{x}_k. \tag{8.21}$$

Note, when implementing a right-preconditioned Krylov method, one does not need $\boldsymbol{M}$, since

$$\boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{u} = \boldsymbol{b}, \tag{8.22}$$

can be solved for $\boldsymbol{u}$ and subsequently

$$\boldsymbol{M}^{-1}\boldsymbol{u} = \boldsymbol{x}. \tag{8.23}$$

The initial residual is the same for preconditioned and unpreconditioned iterations. In practice implementing right-preconditioning broadly speaking boils down to

1. Replace $\boldsymbol{A}$ with $\boldsymbol{A}\boldsymbol{M}^{-1}$, except when calculating $\boldsymbol{r}_0$.
2. One operation with $\boldsymbol{M}^{-1}$ at the end to obtain the accumulated updates to $\boldsymbol{x}_0$.

It is important to note that both $\boldsymbol{A}\boldsymbol{M}^{-1}$, $\boldsymbol{M}^{-1}\boldsymbol{A}$ have the same spectrum. However, there is a difference due to the difference in residuals, in left-preconditioned GMRES $\boldsymbol{M}^{-1}\boldsymbol{r}$ is minimized, in right-preconditioned GMRES $\boldsymbol{r}$ is minimized. For both variants of the preconditioning, the same number of $\boldsymbol{M}^{-1}$ operations are required.

The main motivation for choosing right-preconditioning in our implementation of BiCGStab(L) is that operations are performed which aim to minimize the residual $\boldsymbol{r}$, not $\boldsymbol{M}^{-1}\boldsymbol{r}$. Even though this choice may only yield a modest difference in convergence behavior, except if $\boldsymbol{M}^{-1}$ is ill-conditioned [32, p. 287], we aim to make the BiCGStab(L) implementation as broadly applicable as possible. Choosing right-preconditioning may resolve some issues when a user uses an ill-conditioned $\boldsymbol{M}^{-1}$.

8.1.5 Choice of default L-value

One of the defining parameters of the BiCGStab(L) algorithm is the variable L. This parameter determines the number of BiCG steps and the number of MV per iteration. Importantly, it also determines the size of γ used in the LMR part.

It can be seen in line 18 of Algorithm 12 that the LMR step attempts to minimize $\boldsymbol{R}_0$ in some space spanned by $\boldsymbol{R}_{1:L}$, where the columns of $\boldsymbol{R}$ are generated from the BiCG step.

On the one hand, choosing a small L, such as $L = 1$, BiCGStab(L) reduces to a variant of BiCGStab, which brings back issues such as breakdown in the LMR step for matrices with eigenvalues that have a large imaginary part.

On the other hand, choosing a large value of L is inefficient and may not combine the advantages of BiCG and LMR(L), since for large L, BiCGStab(L) first computes L BiCG steps, which may lead to increasing residuals, and only then performs an LMR(L) step. This is also expensive in terms of vector operations, since LMR(L) for large L would have to be performed by full orthogonalization of some space of dimension L to ensure stability. Another problem is that the least squares problem may become badly conditioned for large L. This issue is addressed in [109], where different variants of BiCGStab(L) are described that work around this issue for large L. However, it is difficult to determine which of the presented workarounds should be used. Therefore we did not go the route of implementing one of these variants, and instead chose a modest value of L.

Finally, based on benchmarks presented in Table 5.1 of [45] $L = 2$ and $L = 4$ seem to perform similarly. There is a case to be made for $L = 4$, based on the results shown in [109], where BiCGStab(4) converges in fewer MV than BiCGStab(2). However, this may not be reflected in terms of CPU-time. For a related algorithm, IDR(S)Stab(L) a benchmark for a three-dimensional convection-diffusion problem shown in [47] reports a 15% speed advantage of IDR(S)Stab(2) compared to IDR(S)Stab(4) for all S.

The choice was made to set $L = 2$ as the default, since it resolves all issues BiCGStab has with complex eigenvalues, appears to be the fastest option and avoids badly conditioned least squares problems as much as possible. Furthermore, MATLAB's implementation of BiCGStab(L) also defaults to $L = 2$.

8.2 Induced Dimension Reduction (IDR(S))

The Induced Dimension Reduction method IDR(S) is a robust and efficient Krylov subspace method for square, sparse, large asymmetric linear systems.

The IDR(S) method is a short-recurrence, limited memory, finite termination method [46]; IDR(S) converges in at most $N + N/S$ iterations in exact arithmetic, and uses a fixed number of $3S + 5$ vectors. In comparison, BiCGSTAB terminates in $2N$ iterations and uses 7 vectors. The finite termination method GMRES requires a maximum of N iterations, and uses $I + 3$ vectors, with I the number of iterations. Restarting GMRES limits memory usage, but destroys the finite termination property [32, p. 199]. Note that without restarting, each successive GMRES iteration becomes more expensive in terms of time as well.

In the numerical experiments shown in [48], for indefinite systems IDR(4) outperforms both BiCGStab and BiCGStab(4). Additionally, in [46] IDR(6) outperformed BiCGStab(8) by a factor two, and a factor six compared to the original BiCGStab for a three-dimensional complex valued Helmholtz problem. Furthermore, IDR(S) can handle real matrices with complex eigenvalues more efficiently than BiCGStab [46].

The implementation of the IDR(S) algorithm in Eigen can be found in [50]. This implementation is based on [46] and a reference MATLAB code provided by Martin van Gijzen and Peter Sonneveld. In the next sections we discuss several implementation choices and options that are made in our C++ implementation. Schematically, the IDR(S) algorithm is given in Algorithm 14.

8.2.1 Choice of shadow space

The matrix $\widetilde{\boldsymbol{P}}_0$ of size $N \times S$ is arbitrary. Here we choose every element from a random uniform distribution over $[-1, 1]$ and orthonormalize the columns. This was suggested for IDR(S)Stab(L) in [110] and it was mentioned in [46] that choosing all columns randomly improved robustness. To orthonormalize the columns we use the HouseholderQR method built into Eigen.

Another choice is to chose every element of $\widetilde{\boldsymbol{P}}_0$ random and complex, as suggested in [46]. This improves convergence for matrices with large imaginary eigenvalues, however, this comes at the cost of more expensive arithmetic.

8.2.2 Mitigation of numerical issues for small ω

One of the differences between the Eigen implementation and the reference MATLAB code, is the computation of ω. The reference gives this computation as

Algorithm 14 IDR(S) adapted from reference [47]

```
function IDR($(S)(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol)$)
    $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
                                        ▷ Generate $\boldsymbol{P} = [\boldsymbol{r}, \boldsymbol{A}\boldsymbol{r}, ..., \boldsymbol{A}^{S-1}\boldsymbol{r}]$
    for $q = 1, ..., S$ do
        if $q = 1$ then
            $\boldsymbol{v} \leftarrow \boldsymbol{r}$
        else
            $\boldsymbol{v} \leftarrow \boldsymbol{A}\boldsymbol{v}$
        end if
        $\boldsymbol{P}_{:,q} \leftarrow \boldsymbol{v}$
    end for
    while $\|\boldsymbol{r}\|_2 > tol$ do
                                        ▷ The IDR Step
        $\boldsymbol{\alpha} \leftarrow (\widetilde{\boldsymbol{P}}_0^{\mathrm{T}} \boldsymbol{A}\boldsymbol{P})^{-1} \widetilde{\boldsymbol{P}}_0^{\mathrm{T}} \boldsymbol{r}$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{P}\boldsymbol{\alpha}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - \boldsymbol{A}\boldsymbol{P}\boldsymbol{\alpha}$
        for $q = 1, ..., S$ do
            if $q = 1$ then
                $\boldsymbol{v}_0 \leftarrow \boldsymbol{r}$
            else
                $\boldsymbol{v}_0 \leftarrow \boldsymbol{v}_1$
            end if
            $\boldsymbol{v}_1 \leftarrow \boldsymbol{A}\boldsymbol{v}_0$
            $\boldsymbol{\beta} \leftarrow (\widetilde{\boldsymbol{P}}_0^{\mathrm{T}} \boldsymbol{A}\boldsymbol{P})^{-1} \widetilde{\boldsymbol{P}}_0^{\mathrm{T}} \boldsymbol{A}\boldsymbol{v}_0$
            $\boldsymbol{v}_0 \leftarrow \boldsymbol{v}_0 - \boldsymbol{P}\boldsymbol{\beta}$
            $\boldsymbol{V}_{:,q} \leftarrow \boldsymbol{v}_0$
            $\boldsymbol{v}_1 \leftarrow \boldsymbol{v}_1 - \boldsymbol{A}\boldsymbol{P}\boldsymbol{\beta}$
        end for
                                        ▷ The polynomial step (LMR step)
        $\omega \leftarrow \mathrm{argmin}_\omega \|\boldsymbol{r} - \omega \boldsymbol{A}\boldsymbol{r}\|_2$
        $\boldsymbol{x} \leftarrow \boldsymbol{x} + \omega \boldsymbol{r}$
        $\boldsymbol{r} \leftarrow \boldsymbol{r} - \omega \boldsymbol{A}\boldsymbol{r}$
        $\boldsymbol{P} \leftarrow \boldsymbol{V} - \omega \boldsymbol{A}\boldsymbol{V}$
    end while
    return $\boldsymbol{x}$
end function
```

Algorithm 15, and the Eigen version is given by Algorithm 16. One of the failure modes of LMR is when $\omega = 0$. In algorithms like BiCGStab $|\omega| \ll 1$ can lead to breakdown. To circumvent this issue in the implementation of IDR(S), a check is in place to replace ω if it is too small. In the reference implementation of IDR(S) this check is given by Algorithm 15.

Algorithm 15 Calculation of ω in the reference implementation

```
function Omega($\boldsymbol{t}, \boldsymbol{s}, angle = 0.7$)
    $ns \leftarrow \|\boldsymbol{s}\|$
    $nt \leftarrow \|\boldsymbol{t}\|$
    $ts \leftarrow \boldsymbol{t}^{\mathrm{H}}\boldsymbol{s}$
    $\rho \leftarrow |ts/(nt \cdot ts)|$
    $\omega \leftarrow ts/(nt \cdot nt)$
    if $\rho < angle$ then
        $\omega \leftarrow \omega \cdot angle/\rho$
    end if
    return $\omega$
end function
```

It was pointed out during the code review process by the Eigen team that the reference implementation could produce an undefined ω if $\rho = 0$. The difference between Algorithms 15 and 16 is that in the reference implementation there is no precaution for vanishingly small ρ, which could lead to a potential division by zero. For small ρ, the reference implementation computes ω as

$$\omega = ts \cdot \frac{ts}{nt \cdot nt} \cdot \frac{angle}{\rho}. \tag{8.24}$$

To avoid the potential division by zero, in the Eigen implementation this is rewritten as

$$\omega = angle \cdot \frac{ns}{nt} \cdot \frac{ts}{|ts|} = angle \frac{ns}{nt} \cdot \operatorname{sgn}(ts), \tag{8.25}$$

where sgn is the complex valued signum function, which is well-behaved for all ts, except $ts = 0$. Note that the potential division by nt cannot lead to a division by zero, since if $nt = 0$ then $ts = 0$.

Algorithm 16 Calculation of ω in the Eigen implementation

```
function OMEGA($\boldsymbol{t}, \boldsymbol{s}, angle = 0.7$)
    $ns \leftarrow \|\boldsymbol{s}\|$
    $nt \leftarrow \|\boldsymbol{t}\|$
    $ts \leftarrow \boldsymbol{t}^{\mathrm{H}}\boldsymbol{s}$
    $\rho \leftarrow |ts/(nt \cdot ns)|$
    if $\rho < angle$ then
        if $ts = 0$ then return 0
        else
            return $angle \cdot (ns/nt) \cdot (ts/|ts|)$
        end if
    end if
end function
```

8.2.3 Optional residual smoothing

One of the options provided in this implementation of IDR(S) is residual smoothing. At a modest cost of a couple vector operations the residual can be made to decrease monotonically. However, this does not significantly influence the rate of convergence.

Since the convergence rate is essentially the same [149], by default this option is turned off. Nevertheless, if smooth decrease of the residual is required, it can be provided by our IDR(S) implementation.

8.3 Induced Dimension Reduction Stabilized(L) (IDR(S)Stab(L))

The implementation of the IDR(S)Stab(L) algorithm in Eigen can be found at [51]. This implementation is based on [110, 48]. Here we specifically implemented the variant from [110], as it offers a reduced residual gap and requires less CPU time than the variant of [48] as shown in the numerical experiments of [110]. This algorithm is given in Algorithm 17.

IDR(S)Stab(L) combines IDR(S) with a higher order polynomial step such as in BiCGStab(L).

The matrix $\widetilde{\boldsymbol{R}}_0$ is chosen randomly with elements from the uniform random distribution $[-1, 1]$, the default random distribution in Eigen. The columns of $\widetilde{\boldsymbol{R}}_0$ are then orthonormalized as suggested in [110].

A major difference between IDR(S)Stab(L) presented in [110] and the Eigen-implementation is the use of matrix-concatenations. For example, lines 16, 18 and 19 of the reference implementation shown in Algorithm 17 concatenate matrices. In our implementation the space is pre-allocated, and parts of the pre-allocated matrices are accessed directly. This reduces the number of memory allocations.

Several variables can be precomputed or rewritten in Algorithm 17, for example $(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{R}_0)^{\mathrm{T}} = \boldsymbol{R}_0^{\mathrm{T}}\boldsymbol{A}$. Since $\boldsymbol{R}_0$ is constant the result of this product can be cached.

Another difference is that we use modified Gram-Schmidt for the steps on lines 9-10, 27-37, to improve the robustness compared to the reference implementation which uses regular Gram-Schmidt.

Algorithm 17 IDR(S)Stab(L) based on [110]

```
function IDR($S$)STAB($L$)($\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}, tol, \widetilde{\boldsymbol{R}}_0$)
    $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}, \quad \boldsymbol{r} = [\boldsymbol{r}_0]$
    for $q = 1, 2, ..., S$ do
        if $q = 1$ then
            $\boldsymbol{u}_0 = \boldsymbol{r}_0$
        else
            $\boldsymbol{u}_0 = \boldsymbol{A}\boldsymbol{u}_0$
        end if
        $\boldsymbol{\mu} = (\boldsymbol{U}_{0,:,1:q-1})^{\mathrm{T}}\boldsymbol{u}_0, \quad \boldsymbol{u}_0 = \boldsymbol{u}_0 - \boldsymbol{U}_{0,:,1:q-1}\boldsymbol{\mu}$
        $\boldsymbol{u}_0 = \boldsymbol{u}_0/\|\boldsymbol{u}_0\|_2, \quad \boldsymbol{U}_{0,:,q} = \boldsymbol{u}_0$
    end for
    while $\|\boldsymbol{r}_0\| > tol$ do
        for $j = 1, 2, ..., L$ do                                   ▷ The IDR step
            $\boldsymbol{\sigma} = (\boldsymbol{A}^{\mathrm{T}}\boldsymbol{R}_0)^{\mathrm{T}}\boldsymbol{U}_{j-1}$
            if $j = 1$ then
                $\boldsymbol{\alpha} = \boldsymbol{\sigma}^{-1}(\widetilde{\boldsymbol{R}}_0^{\mathrm{T}}\boldsymbol{r}_0)$
            else
                $\boldsymbol{\alpha} = \boldsymbol{\sigma}^{-1}((\boldsymbol{A}^{\mathrm{T}}\widetilde{\boldsymbol{R}}_0)^{\mathrm{T}}\boldsymbol{r}_{j-2})$
            end if
            $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{U}_0\boldsymbol{\alpha}, \quad \boldsymbol{r}_0 = \boldsymbol{r}_0 - \boldsymbol{A}(\boldsymbol{U}_0\boldsymbol{\alpha})$
            for $i = 1, 2, ..., j - 2$ do
                $\boldsymbol{r}_i = \boldsymbol{r}_i - \boldsymbol{U}_{i+1}\boldsymbol{\alpha}$
            end for
            if $j > 1$ then
                $\boldsymbol{r} = [\boldsymbol{r}; \boldsymbol{A}\boldsymbol{r}_{j-2}]$
            end if
            for $q = 1, 2, ..., S$ do
                if $q = 1$ then
                    $\boldsymbol{u} = \boldsymbol{r}$
                else
                    $\boldsymbol{u} = [\boldsymbol{u}_1; \boldsymbol{u}_2; ...; \boldsymbol{u}_j]$
                    $\boldsymbol{\beta} = \boldsymbol{\sigma}^{-1}((\boldsymbol{A}^{\mathrm{T}}\widetilde{\boldsymbol{R}}_0)^{\mathrm{T}}\boldsymbol{u}_{j-1})$
                    $\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{U}\boldsymbol{\beta}, \quad \boldsymbol{u} = [\boldsymbol{u}; \boldsymbol{A}\boldsymbol{u}_{j-1}]$
                    $\boldsymbol{\mu} = (\boldsymbol{V}_{j,:,1:q-1})^{\mathrm{T}}\boldsymbol{u}_j, \quad \boldsymbol{u} = \boldsymbol{u} - \boldsymbol{V}_{(:,1:q-1)}\boldsymbol{\mu}$
                    $\boldsymbol{u} = \boldsymbol{u}/\|\boldsymbol{u}_j\|_2, \quad \boldsymbol{V}_{(:,q)} = \boldsymbol{u}$
                end if
            end for
            $\boldsymbol{U} = \boldsymbol{V}$
        end for
        $\boldsymbol{r} = [\boldsymbol{r}; \boldsymbol{A}\boldsymbol{r}_{L-1}]$              ▷ The polynomial step (LMR step)
        $\boldsymbol{\gamma} = [\gamma_1; \gamma_2; ...; \gamma_L] = \mathrm{argmin}_{\boldsymbol{\gamma}} \|\boldsymbol{r}_0 - [\boldsymbol{r}_1, \boldsymbol{r}_2, ..., \boldsymbol{r}_L]\boldsymbol{\gamma}\|_2$
        $\boldsymbol{x} = \boldsymbol{x} + [\boldsymbol{r}_0, \boldsymbol{r}_1, ..., \boldsymbol{r}_{L-1}]\boldsymbol{\gamma}, \quad \boldsymbol{r}_0 = \boldsymbol{r}_0 - \boldsymbol{A}([\boldsymbol{r}_0, \boldsymbol{r}_1, ..., \boldsymbol{r}_{L-1}]\boldsymbol{\gamma})$
        $\boldsymbol{U} = [\boldsymbol{U}_0 - \sum_{j=1}^{L} \gamma_j \boldsymbol{U}_j]$
    end while
end function
```

8.3.1 Choice of S and L

Based on numerical experiments presented in [110], $S = 4, L = 2$ is the most efficient in terms of CPU time. In terms of Matrix-Vector products (MV) and AXPY operations ($\alpha\boldsymbol{x} + \boldsymbol{y}$), a comparison between the IDR(S)Stab(L) variants of [48] and [110] is presented in Table 8.3.1. It can be seen that even though [110] requires 30% more MV for $S = 4, L = 2$ compared to [48], the variant of [110] is cheaper than [48] in terms of AXPY operations, [48] requires 78% more AXPY operations.

To avoid edge cases when dealing with small matrices, we add a check to make sure that $S < N$ and $L < N$. If this check fails, we solve the linear system directly using an LU decomposition.

Table 8.3.1: Comparison of the number of MV and AXPY operations between two IDR(S)Stab(L) formulations.

Reference	MV	AXPY
[48]	$L(S+1)$	$\frac{1}{2}LS(L+1)(S+1) + L(S^2+3S+2)$
[110]	$L(S+1)+L+1$	$\frac{1}{2}LS(L+1)(S+1) + \frac{3}{2}L + S + \frac{1}{2}$
[48],$S=4,L=2$	10	120
[110],$S=4,L=2$	13	67.5

8.3.2 FOM construction of initial U-matrix

In reference [110], lines 3-7 of Algorithm 1 represent the Arnoldi process to generate S orthogonal vectors in the Krylov subspace $\mathcal{K}_S\{\boldsymbol{r}_0, \boldsymbol{A}\}$. In the Eigen implementation we combined this Arnoldi process with the FOM method to provide a possible early exit. The complete FOM algorithm is discussed in Section 7.2.

The idea of modifying the Arnoldi construction to include FOM is that:

1. The extra CPU cost is negligible, as only a small least squares problem has to be solved and no extra MV are required.
2. The extra memory cost is negligible, only an extra matrix of $\mathcal{O}(S^2)$ elements has to be stored.
3. If the initial guess is sufficiently accurate, for example when solving time-dependent problems, the solution produced by FOM can provide an early exit, saving an entire IDR(S)Stab(L) cycle.

Part IV

Magnum PI

Chapter 9

Numerical integration of atomic potentials

MagnumPI [150] is a numerical integration toolbox for computing scattering angles, cross sections and collision integrals, from basic atomic data such as an interaction potential describing the collision between two particles.

One of the key publications in integrating atomic data is [151], where a strategy is elaborated for computing a scattering angle, starting from an interaction potential.

Several physical parameters can be derived from the interaction potential, firstly, a scattering angle describing the asymptotic behavior of two particles after a collision, secondly, using the scattering angle an interaction cross section can be computed. Such cross sections can be used as input data to calculate chemical reaction rates. Finally, a so-called collision integral can be computed, which gives macroscopic physical parameters such as viscosity and diffusivity [152].

The main focus of this chapter is to investigate the integration strategy used in [151], which is an adaptive quadrature strategy referred to as the "fractal" method in this reference. Other adaptive quadrature schemes are presented as well. We present an error analysis of each scheme, perform numerical experiments showing the relative efficiency, discuss convergence behavior of the schemes and present an outlook on the integration of interaction potentials.

The current working version of MagnumPI can be found at `https://gitlab.com/magnumpi/magnumpi`.

9.1 Atomic scattering model

The scattering angle Θ as function of the impact parameter b is given in [151] by

$$\Theta(b) = \pi - 2b \int_{r_\mathrm{m}}^{\infty} \left[1 - \frac{b^2}{r^2} - \frac{V(r)}{E}\right]^{-1/2} \frac{\mathrm{d}r}{r^2}, \tag{9.1}$$

where r_m is defined as the largest root of the term in square brackets, $V(r)$ is the interaction potential and E the energy of the incoming particle. The interpretation of r_m is that it is the distance of closest approach. Geometrically the scattering process is shown in Figure 9.1.1.

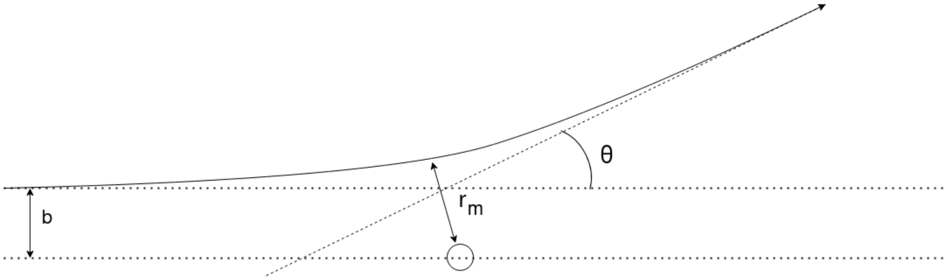


Figure 9.1.1: Geometry of the atomic interaction.

The interaction cross section $Q^{(L)}(E)$ can be obtained by integrating over all impact parameters;

$$Q^{(L)}(E) = 2\pi \int_0^{\infty} b(1 - \cos^L(\Theta))\mathrm{d}b, \tag{9.2}$$

with L a positive integer. Note that for large impact parameters, b, the distance of closest approach, r_m, is also large as is evident from Figure 9.1.1. For physically relevant interaction potentials $\lim_{r\to\infty} V(r) = 0$. In this limit Θ approaches 0, and the integrand of (9.2) also tends to zero in this limit. This is consistent with the physical interpretation, as without any interaction potential, the interaction cross section should be zero. This can also be shown from (9.1) by setting $V(r) = 0$, which results in $\Theta = 0$.

When dealing with attractive potentials, at specific energies and values of the impact parameter an incoming particle can be nearly trapped in the potential, similar to an asteroid approaching earth, completing several orbits and escaping again. This is referred to as orbiting. In such cases the scattering angle Θ will vary strongly with b. As a consequence the cosine term in (9.2) leads to a strongly oscillatory integrand.

Note that both (9.1) and (9.2) are improper integrals. The strategy used in MagnumPI is discussed in Section 9.4. To demonstrate the numerical schemes for (9.2), a model problem is presented in [151];

$$\int_{10^{-8}}^{2} \sin(\pi \ln(x))\mathrm{d}x, \tag{9.3}$$

which we use here as a starting point. We show here that, while maintaining a similar adaptive quadrature strategy as in [151], faster convergence can be achieved than for the previously mentioned "fractal integration method" of [151].

9.2 Adaptive quadrature

The idea of the "fractal method", as dubbed in [151], is to estimate a definite integral over $[a, b]$ using some quadrature rule, and estimate the error over this interval. If the error exceeds a defined tolerance, the interval is bisected. For each of the subintervals the quadrature rule is applied recursively.

The motivation for this method is that the function f is sampled more efficiently compared to methods with an equidistant distribution of nodes. Methods with nodes based on a local error estimate are known as adaptive quadrature methods [153]. A template algorithm that performs such integration strategy is given in Algorithm 18.

Algorithm 18 Template adaptive quadrature integration of a function f on $[a, b]$.

```
function INTEGRATE(f, $a$, $b$, $tol$)
    [$integral$, $error\_estimate$] ← Quadrature(f,a,b)
    if $error\_estimate > tol$ then
        $integral_1$ ← Integrate(f, $a$, $(a + b)/2$)
        $integral_2$ ← Integrate(f, $(a + b)/2$, $b$)
        $integral$ ← $integral_1 + integral_2$
    end if
    return $integral$
end function
```

The function `Quadrature` is any quadrature rule applied on the interval $[a, b]$, and returns an estimate for the integral with corresponding error estimate. Several quadrature rules and methods to estimate the error exist, which are discussed shortly.

First, we define $Q_{\mathrm{X}}(a, b)$ as the approximation to the exact integral $I(a, b) := \int_a^b f(x)\mathrm{d}x$ computed with a quadrature rule X. Next, we also define $\mathcal{E}_{\mathrm{X}}$ as the error in the approximation; $\mathcal{E}_{\mathrm{X}}(a, b) := I(a, b) - Q_{\mathrm{X}}(a, b)$. Additionally, $h := b - a$, the size of an interval $[a, b]$, and m the midpoint of the interval $m := (b + a)/2$. Unless explicitly denoted otherwise, $Q_{\mathrm{X}} = Q_{\mathrm{X}}(a, b)$, $\mathcal{E}_{\mathrm{X}} = \mathcal{E}_{\mathrm{X}}(a, b)$ and $I = I(a, b)$.

The first quadrature rule we discuss is the midpoint rule [154, 155], given by;

$$Q_{\mathrm{M}} = hf\left(m\right), \tag{9.4}$$

with corresponding error term

$$\mathcal{E}_{\mathrm{M}} = \frac{h^3}{24} f''(m) + \frac{h^5}{1920} f^{(4)}(m) + \mathcal{O}(h^7). \tag{9.5}$$

The second quadrature rule is the trapezoidal rule [154, 155] given by:

$$Q_{\mathrm{T}} = \frac{h}{2}\left(f(b) + f(a)\right), \tag{9.6}$$

with an error of

$$\mathcal{E}_{\mathrm{T}} = -\frac{h^3}{12} f''(m) - \frac{h^5}{480} f^{(4)}(m) + \mathcal{O}(h^7). \tag{9.7}$$

The fractal quadrature rule given in [151] combines the midpoint and trapezoidal rules by taking the arithmetic average of the two rules;

$$Q_{\mathrm{F}} = \tfrac{1}{2}(Q_{\mathrm{M}} + Q_{\mathrm{T}}), \tag{9.8}$$

where the error in the combined rules is then

$$\mathcal{E}_{\mathrm{F}} = (\mathcal{E}_{\mathrm{M}} + \mathcal{E}_{\mathrm{T}})/2 = -\frac{h^3}{48} f''(m) - \frac{h^5}{1280} f^{(4)}(m) + \mathcal{O}(h^7), \tag{9.9}$$

which is better than both the midpoint and trapezoidal rules by a factor two. However, the leading term in the error is still $\mathcal{O}(h^3)$.

Note that by taking a weighted average of Q_{M} and Q_{T} one can eliminate the $\mathcal{O}(h^3)$ term in equation (9.9). Define a weighted rule Q_{S} as

$$Q_{\mathrm{S}} := \alpha Q_{\mathrm{M}} + \beta Q_{\mathrm{T}}, \tag{9.10}$$

with $\alpha + \beta = 1$. The error in Q_{S} is given by the weighted average of the error terms;

$$\begin{aligned} \mathcal{E}_{\mathrm{S}} = \alpha &\left(\frac{h^3}{24} f''(m) + \frac{h^5}{1920} f^{(4)}(m) + \mathcal{O}(h^7) \right) \\ +\beta &\left(-\frac{h^3}{12} f''(m) - \frac{h^5}{480} f^{(4)}(m) + \mathcal{O}(h^7) \right). \end{aligned} \tag{9.11}$$

By taking $\alpha = \frac{2}{3}$ and $\beta = \frac{1}{3}$ the $\mathcal{O}(h^3)$-term can be eliminated, which results in the quadrature rule

$$Q_{\mathrm{S}} = \frac{2}{3} Q_{\mathrm{M}} + \frac{1}{3} Q_{\mathrm{T}}, \tag{9.12}$$

or more explicitly:

$$Q_{\mathrm{S}} = \frac{h}{6}(f(a) + 4f(m) + f(b)), \tag{9.13}$$

better known as Simpson's rule [154]. Similar to the "fractal method", Simpson's rule can also be applied adaptively, and has a more favorable error term. The exact error in the Simpson's rule is given by [154, 155]

$$\mathcal{E}_{\mathrm{S}}(a,b) = f^{(4)}(m) \frac{h^5}{2880} + \mathcal{O}(h^7). \tag{9.14}$$

In the next section we seek a method of estimating the local error; the error in a subinterval. Using the local error, the adaptive algorithm can decide whether to bisect a given interval further. The error estimates are discussed in the next section.

9.3 Local error estimate

9.3.1 Fractal rule

The fractal rule of reference [151] can be written as a linear combination of the trapezoidal and the midpoint rule;

$$Q_{\mathrm{F}} = \frac{1}{2} Q_{\mathrm{T}} + \frac{1}{2} Q_{\mathrm{M}} = \frac{h}{4} \left(f(a) + f(b) + 2f(m) \right), \tag{9.15}$$

with an error $\mathcal{E}_{\mathrm{F}}$ given by equation (9.9).

The error estimation strategy gives an approximation of the true error $\mathcal{E}_{\mathrm{F}}$, denoted as ϵ_{F}. To obtain the estimate ϵ_{F} reference [151] computes $\epsilon_{\mathrm{F}} = Q_{\mathrm{F}} - Q_{\mathrm{T}}$, which results in

$$\epsilon_{\mathrm{F}} = -\frac{h}{4} \left(f(a) - 2f(m) + f(b) \right), \tag{9.16}$$

the term inside brackets is reminiscent of the central difference estimate for the second derivative. The estimate for the error,

$$\epsilon_{\mathrm{F}} = -\frac{h^3}{4}\left(\frac{1}{h^2}(f(b) - 2f(m) + f(a))\right) = -\frac{h^3}{4}f''(m) + \mathcal{O}(h^5), \tag{9.17}$$

has the same order as the actual error, $\mathcal{O}(h^3)$, and it estimates $f''(m)$ in the actual error (9.9) with the second order central difference method. As a consequence the error estimated by the fractal rule is indeed representative of the true error.

9.3.2 Simpson's rule

Simpson's rule is a linear combination of the trapezoidal and midpoint rules, in such a way that the $\mathcal{O}(h^3)$ term in the error cancels. One method to obtain an error estimate for Simpson's rule is analogous to Section 9.3.1, namely,

$$\epsilon_{\mathrm{S}} = Q_{\mathrm{S}} - Q_{\mathrm{T}} = -\frac{2}{3}Q_{\mathrm{T}} + \frac{2}{3}Q_{\mathrm{M}} = -\frac{1}{3}h^3\left[\frac{1}{h^2}\left(f(a) - 2f(m) + f(a)\right)\right]. \tag{9.18}$$

Such an estimate has two mismatches with the real leading order term in the error. Firstly, the order of the estimated error shows $\mathcal{O}(h^3)$ behavior, whilst the actual error is $\mathcal{O}(h^5)$. Secondly, the term inside square brackets approximates the *second* derivative, whilst the real error is proportional to the *fourth* derivative.

9.3.3 Bisection based error estimate

A suggestion for computing an error estimate is given in reference [153];

$$\epsilon_{\mathrm{S}} = Q_{\mathrm{S}} - (Q_{\mathrm{S}}(a,m) + Q_{\mathrm{S}}(m,b)). \tag{9.19}$$

To investigate the properties of this estimate consider the expansion $Q_{\mathrm{X}} = I - \mathcal{E}_{\mathrm{X}}$;

$$\epsilon_{\mathrm{S}} = I - \mathcal{E}_{\mathrm{S}} - (I(a,m) - \mathcal{E}_{\mathrm{S}}(a,m) + I(m,b) - \mathcal{E}_{\mathrm{S}}(m,b)), \tag{9.20}$$

where ϵ_{S} approximates the error $\mathcal{E}_{\mathrm{S}}$. Since I represents the exact value of the integral, equation (9.20) can be simplified to

$$\epsilon_{\mathrm{S}} = -\mathcal{E}_{\mathrm{S}} + \mathcal{E}_{\mathrm{S}}(a,m) + \mathcal{E}_{\mathrm{S}}(m,b). \tag{9.21}$$

Then using the error estimate presented in equation (9.14), the estimated error based on bisection is

$$\epsilon_{\mathrm{S}} = -\frac{h^5}{2880}f^{(4)}(m) + \frac{1}{2^5}\frac{h^5}{2880}(f^{(4)}(m_1) + f^{(4)}(m_2)), \tag{9.22}$$

where $m_1 = \frac{1}{4}(3a+b)$, $m_2 = \frac{1}{4}(a+3b)$ and $\mathcal{O}(h^7)$-terms are ignored. Equation (9.22) seems to be a much more representative estimate than equation (9.18). First, equation (9.22) has the same order as the actual error term, i.e., $\mathcal{O}(h^5)$. Second, the leading term in the error estimate is the actual error. Third, the deviation from the actual error, the second term in (9.22), is relatively small due to the factor 2^{-5}. Finally, even the deviation is proportional to the fourth derivative. Note that the bisection method as presented here is *not* related to the root-finding method with the same name.

9.3.4 Gaussian quadrature methods

Based on reference [156, p. 514] in this section Gaussian quadrature rules are discussed. Such quadrature rules approximate an integral over the interval $[-1, 1]$ by a summation

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx \sum_{i=1}^{n} w_{\mathrm{G},i} f(x_{\mathrm{G},i}), \tag{9.23}$$

where $w_{\mathrm{G},i}$ are the weights and $x_{\mathrm{G},i}$ the nodes. Note that there are $2n$ degrees of freedom which can be exploited to integrate polynomials up to degree $2n-1$ exactly, using properties of orthogonal polynomials. The idea of the Gaussian quadrature method is to choose the n nodes and weights such that a polynomial of a degree as large as possible can be integrated exactly. It turns out that with n nodes, a polynomial of degree $2n-1$ can still be integrated exactly. Implicitly, the Gaussian quadrature method thus approximates f as a polynomial of degree $2n-1$, and integrates the result exactly. To apply the Gaussian quadrature rules over the interval $[a, b]$ we use a linear transformation of the form

$$\int_{a}^{b} f(x)\mathrm{d}x = \int_{-1}^{1} f\left(\frac{b-a}{2}y + \frac{b+a}{2}\right)\frac{b-a}{2}\mathrm{d}y. \tag{9.24}$$

Consider the division of a polynomial $P_{2n-1}(x)$ of degree $2n-1$, by the Legendre polynomial $L_n(x)$ of degree n:

$$P_{2n-1}(x) = q(x)L_n(x) + r(x), \tag{9.25}$$

where $q(x)$ is the quotient, a polynomial of degree $n-1$, and $r(x)$ is the remainder of degree at most $n-1$. Integrating equation (9.25) yields:

$$\int_{-1}^{1} P_{2n-1}(x)\mathrm{d}x = \int_{-1}^{1} q(x)L_n(x)\mathrm{d}x + \int_{-1}^{1} r(x)\mathrm{d}x. \tag{9.26}$$

It is important to note that the Legendre polynomials of degree n are orthogonal to all monomials x^k where $k < n$ [156, p. 514]. Since $q(x)$ is of degree $n-1$ the orthogonality property of the Legendre polynomials gives

$$\int_{-1}^{1} q(x)L_n(x)\mathrm{d}x = 0. \tag{9.27}$$

Thus, the integration as presented in (9.26) yields

$$\int_{-1}^{1} P_{2n-1}(x)\mathrm{d}x = \int_{-1}^{1} r(x)\mathrm{d}x. \tag{9.28}$$

The idea is to choose the nodes $x_{\mathrm{G},i}$ in equation (9.23) such that they correspond with the roots of $L_n(x)$. This way the first term in the right-hand side of (9.26) is integrated exactly, since

$$\sum_{i=1}^{n} w_{\mathrm{G},i} q(x_{\mathrm{G},i}) L_n(x_{\mathrm{G},i}) = 0, \tag{9.29}$$

as $L_n(x_{\mathrm{G},i}) = 0$. The weights $w_{\mathrm{G},i}$ can be exploited to exactly integrate the remainder $r(x)$, since $r(x)$ is a polynomial of degree $\leq n-1$. Due to the linearity of integration it is sufficient to have all monomials; $1, x^1, x^2, ..., x^{n-1}$ be exactly represented by the approximation. To do this, we require that

$$\sum_{i=1}^{n} w_{\mathrm{G},i} x_{\mathrm{G},i}^{k} = \int_{-1}^{1} x^k \mathrm{d}x = \frac{1+(-1)^k}{1+k}, \quad k = 0, 1, ..., n-1. \tag{9.30}$$

This can be compactly represented as a linear system of equations;

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_{\mathrm{G},1} & x_{\mathrm{G},2} & \cdots & x_{\mathrm{G},(n-1)} \\ x_{\mathrm{G},1}^2 & x_{\mathrm{G},2}^2 & \cdots & x_{\mathrm{G},(n-1)}^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_{\mathrm{G},1}^{n-1} & x_{\mathrm{G},2}^{n-1} & \cdots & x_{\mathrm{G},(n-1)}^{n-1} \end{bmatrix} \begin{bmatrix} w_{\mathrm{G},1} \\ w_{\mathrm{G},2} \\ \vdots \\ w_{\mathrm{G},(n-1)} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \vdots \\ (1+(-1)^{n-1})/n \end{bmatrix}. \tag{9.31}$$

With the weights obtained from this linear system all polynomials up to degree $2n-1$ are integrated exactly on the interval $[-1, 1]$. The matrix shown in (9.31) is a so-called "Vandermonde matrix" [157, p. 94]; such matrices are typically ill-conditioned [157, p. 95]. However, the nodes and weights only have to be computed once, for which there are accurate methods [157, p. 184]. Common Gaussian quadrature nodes and weights are tabulated as well [154, p. 147].

As an example consider the three-point Gauss-Legendre quadrature rule. The 3rd degree Legendre polynomial is given by

$$L_3(x) = \frac{1}{2}\left(5x^3 - 3x\right), \tag{9.32}$$

with roots

$$x_{\mathrm{G},1} = -\sqrt{\frac{3}{5}}, \quad x_{\mathrm{G},2} = 0, \quad x_{\mathrm{G},3} = +\sqrt{\frac{3}{5}}. \tag{9.33}$$

To determine the weights the linear system from equation (9.31) for the case $n = 3$;

$$\begin{bmatrix} 1 & 1 & 1 \\ -\sqrt{\frac{3}{5}} & 0 & \sqrt{\frac{3}{5}} \\ -\frac{3}{5} & 0 & \frac{3}{5} \end{bmatrix} \begin{bmatrix} w_{\mathrm{G},1} \\ w_{\mathrm{G},2} \\ w_{\mathrm{G},3} \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 2/3 \end{bmatrix}, \tag{9.34}$$

has to be solved, which results in

$$w_{\mathrm{G},1} = \frac{5}{9}, \quad w_{\mathrm{G},2} = \frac{8}{9}, \quad w_{\mathrm{G},3} = \frac{5}{9}. \tag{9.35}$$

The three-point rule has the special property that it shares the center node with the one-point Gauss-Legendre rule ($x_{\mathrm{G},1} = 0, w_{\mathrm{G},1} = 2$) as well. Such embedded rules can be used to obtain an error estimate, as will be shown in Section 9.3.7. A generalization of the construction of such embedded rules is done in the derivation of Gauss-Kronrod quadrature. These methods will be discussed in Section 9.3.6. In the next section we investigate the error estimate and the true estimate for the Gauss-Legendre quadrature rules.

9.3.5 Error analysis of Gauss-Legendre quadrature

For an n-point Gauss-Legendre quadrature rule the error is given by [154]

$$\mathcal{E}_{\mathrm{G},n} = h^{2n+1} \frac{(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi), \quad \xi \in (a,b). \tag{9.36}$$

To simplify the notation, define

$$R_{\mathrm{G},n} := \frac{(n!)^4}{(2n+1)[(2n)!]^3}. \tag{9.37}$$

The bisection method from Section 9.3.3 can also be applied to the general n-point Gauss-Legendre quadrature;

$$\epsilon_{\mathrm{G},n} = Q_{\mathrm{G},n} - \left(Q_{\mathrm{G},n}\left(a,m\right) + Q_{\mathrm{G},n}\left(m,b\right)\right), \tag{9.38}$$

then by substituting $Q_{\mathrm{G},n} = I_{\mathrm{G},n} + \mathcal{E}_{\mathrm{G},n}$, and simplifying we obtain

$$\epsilon_{\mathrm{G},n} = -\mathcal{E}_{\mathrm{G},n} + \mathcal{E}_{\mathrm{G},n}\left(a,m\right) + \mathcal{E}_{\mathrm{G},n}\left(m,b\right). \tag{9.39}$$

The estimated error is then obtained as

$$\epsilon_{\mathrm{G},n} = -h^{2n+1} R_{\mathrm{G},n} f^{(2n)}(\xi) + \left(\frac{h}{2}\right)^{2n+1} R_{\mathrm{G},n} f^{(2n)}(\xi_1) + \left(\frac{h}{2}\right)^{2n+1} R_{\mathrm{G},n} f^{(2n)}(\xi_2), \tag{9.40}$$

where $\xi_1 \in [m, b]$ and $\xi_2 \in [a, m]$. This can be rewritten to obtain the bisection-based estimated error for Gauss-Legendre methods;

$$\epsilon_{\mathrm{G,n}} = -h^{2n+1} R_{\mathrm{G},n} f^{(2n)}(\xi) + h^{2n+1} \frac{R_{\mathrm{G},n}}{2^{2n+1}} \left(f^{(2n)}(\xi_1) + f^{(2n)}(\xi_2) \right). \tag{9.41}$$

While the true error is the first term in (9.41), viz.,

$$\mathcal{E}_{\mathrm{G,n}} = h^{2n+1} R_{\mathrm{G},n} f^{(2n)}(\xi), \tag{9.42}$$

similar to what has been obtained for Simpson's rule in Section 9.3.3. However, note that for the 2-point Gauss-Legendre rule the error is

$$\mathcal{E}_{\mathrm{G,2}} = \frac{h^5}{4320} f^{(4)}(\xi), \tag{9.43}$$

thus it can integrate polynomials up to degree 3 exactly with 2 function evaluations. Simpson's rule can also integrate polynomials up to degree 3 exactly, however, it requires 3 function evaluations and has a larger coefficient in the error term;

$$\mathcal{E}_{\mathrm{S}} = \frac{h^5}{2880} f^{(4)}(\xi). \tag{9.44}$$

On the other hand, since Simpson's rule uses equidistant nodes, function evaluations could be reused between bisections.

9.3.6 Gauss-Kronrod methods

The Gauss-Kronrod methods are derived similarly to the Gauss-Legendre quadrature methods as elaborated in Section 9.3.4. However, they have the property that for a given set of nodes, two quadrature rules can be obtained.

The Gauss-Kronrod method uses Stieltjes polynomials E_n [158], such that the polynomial division similar to equation (9.25) becomes

$$P_{3n}(x) = q(x) L_n(x) E_{n+1}(x) + r(x). \tag{9.45}$$

Here the quotient $q(x)$ is a polynomial of degree $n-1$, and the remainder $r(x)$ a polynomial of at most degree $n-1$. The Stieltjes polynomials can be obtained from the orthogonality relations

$$\int_{-1}^{1} x^k L_n(x) E_{n+1}(x) \mathrm{d}x = 0, \quad k = 0, 1, ..., n-1. \tag{9.46}$$

From equation (9.46), and an arbitrary normalization condition for E_{n+1} another system of equations is obtained, which can be solved to obtain the coefficients of the polynomials E_{n+1}. With the normalization defined as

$$\int_{-1}^{1} E_{n+1}^2(x) \mathrm{d}x := 1, \tag{9.47}$$

the first few Stieltjes polynomials are given by:

$$\begin{aligned}
E_0(x) &= \frac{1}{\sqrt{2}},\\
E_1(x) &= \sqrt{\frac{3}{2}}x,\\
E_2(x) &= \frac{5}{2\sqrt{2}}x^2 - \frac{3}{2\sqrt{2}},\\
E_3(x) &= 7\sqrt{\frac{5}{22}}x^3 - 3\sqrt{\frac{10}{11}}x,
\end{aligned} \tag{9.48}$$

even though $E_0(x)$ is not used in the derivation of the Gauss-Kronrod methodes, it is listed for completeness. Similarly to the derivation of the Gauss-Legendre quadrature rules, here the $2n+1$ nodes $x_{\mathrm{K},i}$ are chosen at the nodes of $L_n(x)$ and $E_{n+1}(x)$. Where $q(x)$ and $r(x)$ are polynomials of at most degree n. The weights are chosen such that $\int_{-1}^{1} r(x)\mathrm{d}x$ is integrated exactly.

Note that the roots of $E_2(x)$ are $\pm\sqrt{\frac{3}{5}}$, which are the same as two of the nodes in the three-point Gauss-Legendre method of equation (9.33). Thus, the three-point Gauss-Legendre quadrature rule is the same as the one-point Gauss-Legendre rule extended with 2 additional Kronrod points. The nested one-point Gauss-Legendre quadrature is simply the midpoint method.

9.3.7 Error analysis of Gauss-Kronrod quadrature rules

Gauss-Kronrod methods are nested methods to compute an integral that runs over an interval $[-1, 1]$. The weights and nodes of this method are such that a lower order rule is embedded in a higher order rule. For example, the three-point Gauss-Kronrod rule has weights and nodes given by Table 9.3.1.

Table 9.3.1: Nodes $x_{\mathrm{K},i}$ and weights $w_{\mathrm{K},i}$ for the three-point Gauss-Kronrod rule.

$x_{\mathrm{K},i}$	$w_{\mathrm{K},i}$	$w_{\mathrm{G},i}$
$-\sqrt{\frac{3}{5}}$	$\frac{5}{9}$	0
0	$\frac{8}{9}$	2
$+\sqrt{\frac{3}{5}}$	$\frac{5}{9}$	0

Using the coefficients in Table 9.3.1 one can obtain two integral estimates;

$$Q_{\mathrm{G},1} = 2f(0), \tag{9.49}$$

and

$$Q_{\mathrm{K},3} = \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right). \tag{9.50}$$

However, note that one only needs three function evaluations to compute both, since $f(0)$ can be reused. One can then obtain an error estimate similar to what is shown in reference [154] for a 15-point Gauss-Kronrod rule, by taking the absolute difference between the two integral estimates;

$$\epsilon_{\mathrm{GK},3} = |Q_{\mathrm{K},3} - Q_{\mathrm{G},1}|. \tag{9.51}$$

This produces an error estimate proportional to $h^3 f''(\xi)$ where $\xi \in (a, b)$, since $Q_{\mathrm{G},1}$ is the largest contributor to the error here. A general expression for the error in a GK-rule is unknown [156], however, it *is* known that it can integrate polynomials up to order $3n + 1$ exactly, with n the order of the embedded Gaussian rule. So the error has to be at least proportional to the $(3n + 2)$th derivative. *If* the error of the Gauss-Kronrod rule can be written in a form $\mathcal{E}_{\mathrm{K},n} = h^{3n+3} Q_{\mathrm{K},n} f^{(3n+2)}(\xi)$, similarly to the Gauss-Legendre error, then the actual error of $Q_{\mathrm{K},3}$ is proportional to $h^6 f^{(5)}$, not proportional to $h^3 f''$ as given by (9.51).

With the bisection method one can then obtain an error estimate

$$\epsilon_{\mathrm{K},n} = -h^{3n+3} R_{\mathrm{K},n} \left(f^{(3n+2)}(\xi) - \frac{1}{2^{3n+3}} \left(f^{(3n+2)}(\xi_1) + f^{(3n+2)}(\xi_2) \right) \right), \tag{9.52}$$

where $\xi_2 \in [a, m]$ and $\xi_1 \in [m, b]$. The bisection method is more expensive to evaluate than (9.51). However, it may have more favorable properties. As the error estimate provided by the bisection method is proportional to the same derivative as the actual error, and has the same power of h as the actual error.

9.4 Computation of the scattering angle

The integral for the scattering angle in (9.1) is improper in two ways, the upper bound is infinite, and there is a singularity of the integrand at the lower bound. Additionally, the lower bound r_{m} has to be obtained via a root finding process; r_{m} is the largest root of the term inside square brackets. In practice there is no analytical solution, and $V(r)$ may be in table form, further complicating analytical investigation. The idea of this section is to treat the integral in (9.1) as a Terminal Value Problem (TVP). We then use physical arguments to choose a finite upper bound, and integrate towards smaller r. When the term inside square brackets is negative, or r is negative we reject the step.

A common scheme to integrate ODEs numerically is the Dormand-Prince

5(4) Embedded Runge-Kutta method with adaptive step size [157, p. 910]. To convert (9.1) into a TVP, let $\aleph(r)$ be a primitive of the integrand in (9.1), then

$$\aleph(\infty) - \aleph(r_\mathrm{m}) = \int_{r_\mathrm{m}}^{\infty} \left[1 - \frac{b^2}{r^2} - \frac{V(r)}{E}\right]^{-1/2} \frac{\mathrm{d}r}{r^2}, \tag{9.53}$$

which allows us to obtain the TVP for $\aleph(r)$,

$$\frac{\mathrm{d}\aleph}{\mathrm{d}r} = \left[1 - \frac{b^2}{r^2} - \frac{V(r)}{E}\right]^{-1/2} \frac{1}{r^2}, \quad \aleph(\infty) = 0. \tag{9.54}$$

Since r_m is not known a priori, but has to be obtained from a root finding problem, we integrate from $r = \infty$ in decreasing r-direction, until

$$1 - \frac{b^2}{r^2} - \frac{V(r)}{E} \leq 0. \tag{9.55}$$

Since the potentials $V(r)$ are atomic interactions, it is expected that at large distances there will be little effect on the scattering angle. To completely rule out any significant contribution we start integrating from $r_\infty = 10$ m. Then we use an adaptive step size to integrate efficiently over a wide range of length scales.

To integrate from large r_∞ towards r_m we take a step size $h < 0$ in the Runge-Kutta scheme. This step size is adjusted as described in [157, p. 911]. A fourth and fifth order accurate numerical solution is computed, and the error is estimated by taking the difference between these estimates.

Since the right-hand side of (9.54) does not depend on $\aleph$, the scheme from [157, p. 911] can be simplified to,

$$k_1 = h\aleph'(r_n), \tag{9.56a}$$
$$k_3 = h\aleph'(r_n + 3h/10), \tag{9.56b}$$
$$k_4 = h\aleph'(r_n + 4h/5), \tag{9.56c}$$
$$k_5 = h\aleph'(r_n + 8h/9), \tag{9.56d}$$
$$k_6 = h\aleph'(r_n + h), \tag{9.56e}$$

k_2 is not needed and $k_7 = k_6$, since $\aleph'$ is independent of $\aleph$. The next values $\aleph_{n+1} \approx \aleph(r_{n+1})$ are then given by

$$\aleph^{(4)}_{n+1} = \tfrac{5179}{57600}k_1 + \tfrac{7571}{16695}k_3 + \tfrac{393}{640}k_4 - \tfrac{92097}{339200}k_5 + \tfrac{187}{2100}k_6 + \tfrac{1}{40}k_7, \tag{9.57a}$$
$$\aleph^{(5)}_{n+1} = \tfrac{35}{384}k_1 + \tfrac{500}{1113}k_3 + \tfrac{125}{192}k_4 - \tfrac{2187}{6784}k_5 + \tfrac{11}{84}k_6, \tag{9.57b}$$

where $\aleph^{(5)}_{n+1}$ is a fifth and $\aleph^{(4)}_{n+1}$ a fourth order estimate. The error in this step can then be estimated by

$$\epsilon = |\aleph^{(5)}_{n+1} - \aleph^{(4)}_{n+1}|. \tag{9.58}$$

The next step size is then chosen as

$$h_{n+1} = 0.9 h_n \left| \frac{tol}{\max(\epsilon, \epsilon/|\aleph^{(5)}_{n+1}|)} \right|^{1/5}, \tag{9.59}$$

where changes in the step size are limited such that

$$\frac{1}{2} \leq \frac{h_{n+1}}{h_n} \leq 2. \tag{9.60}$$

9.5 Results and discussion

9.5.1 Benchmark problem Colonna

The first numerical experiment is the same as presented in [151]; we integrate

$$\int_{10^{-8}}^{2} \sin(\pi \ln(x)) \mathrm{d}x, \tag{9.61}$$

numerically using an adaptive integration strategy. Here we repeat the experiment for a number of quadrature rules.

To test the quadrature schemes, we set a tolerance, then require that the local value of the absolute error is less than this tolerance. For a given tolerance we then approximate the integral shown in (9.61), and store the number of integrand evaluations. The results are shown in Figure 9.5.1.

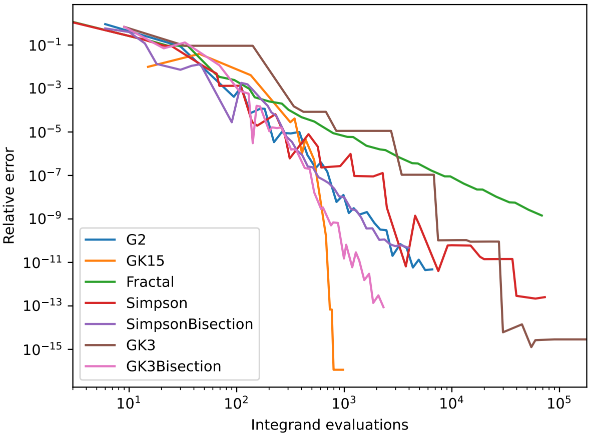


Figure 9.5.1: Relative error as function of the number of integrand evaluations for various adaptive integration strategies for the integral in (9.61).

It can be seen in Figure 9.5.1 that the methods that use the bisection based error estimate (G2, SimpsonBisection and GK3Bisection) from Section 9.3.3 converge more smoothly compared to the methods that use an embedded estimate (Simpson and GK3). The exceptions to this are GK15, which converges much faster than the other methods, and the fractal method. The error estimate used in the fractal method is representative of the true error, and thus converges smoothly. For the GK3 method, the error estimate is $\mathcal{O}(h^3)$, whereas the true error is $\mathcal{O}(h^6)$. This is likely the cause of the staircase shape in the convergence graph.

The GK15 method converges by far the fastest for this problem, for strict error requirements. Even though the fractal method converges very smoothly, the rate of convergence is subpar.

9.5.2 Lennard-Jones potential

Here we use the Lennard-Jones potential as a model to compare our integration strategy with results from the literature. In reference [152] the Lennard-Jones potential has the form

$$V(r) = 4\epsilon\left(\left(\frac{d}{r}\right)^{12} - \left(\frac{d}{r}\right)^{6}\right), \tag{9.62}$$

where ϵ and d are experimental parameters that depend on the atoms of interest. Here d/r can be seen as a dimensionless length. A dimensionless energy β is introduced which links the interaction strength ϵ with the energy of the incoming particle E,

$$\beta = \frac{\epsilon}{2E}. \tag{9.63}$$

Our implementation splits (9.2), and integrates up to a finite upper bound, i.e.,

$$Q^{(L)}(E) \approx 2\pi \int_0^{5\times10^{-9}} b(1-\cos^L(\Theta))\mathrm{d}b + 2\pi \int_{5\times10^{-9}}^{10^{-8}} b(1-\cos^L(\Theta))\mathrm{d}b. \tag{9.64}$$

A plot of the absolute value of the integrand of (9.64) is shown in Figure 9.5.2. The integrand reaches a minimum at $b = 10^{-8}$ m, and does not tend to zero for $b \to \infty$. For $b \gtrsim 10^{-7}$ m the integrand is dominated by rounding error, since $1 - \cos(\Theta)$ is not exactly zero.

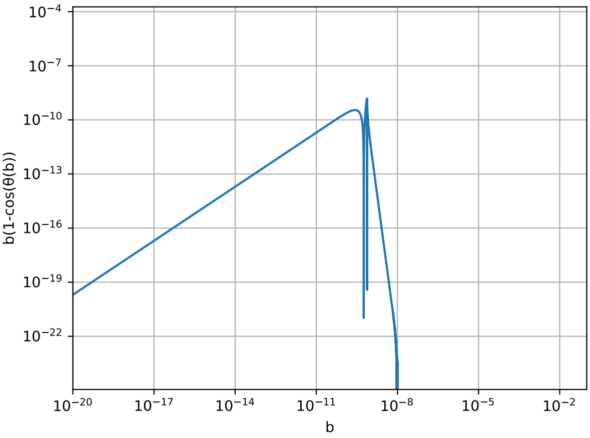
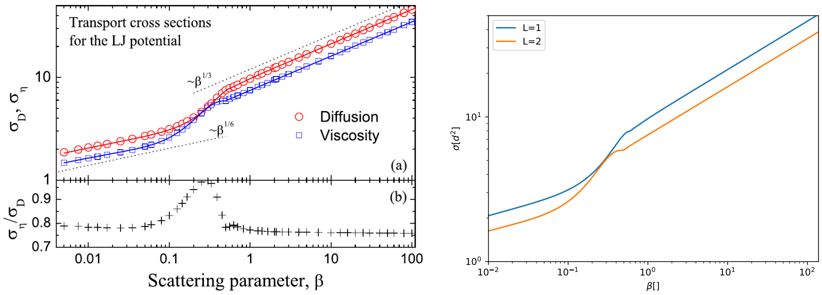


Figure 9.5.2: $|b(1-\cos^L(\Theta))|$ as function of b for the LJ problem, for $\beta = 1$ and $L = 1$.

A potentially more robust approach would be to use a strategy similar to the one suggested in [151]; solve (9.2) as an initial value problem, whenever orbiting is encountered apply another scheme in a small region. Ideally, a method specifically constructed for oscillating integrands such as the Levin and Filon methods [159, p. 29]. It can be seen in Figure 9.5.3 that for both $L = 1$ and $L = 2$ the computed cross sections for the Lennard-Jones potential agree with reference [152]. The



(a) Cross section data from [152] in units of d^2.

(b) Cross sections computed using MagnumPI using GK15.

Figure 9.5.3: Comparison of our numerical results with values from literature. The diffusion cross section corresponds to $L = 1$, the viscosity cross section to $L = 2$.

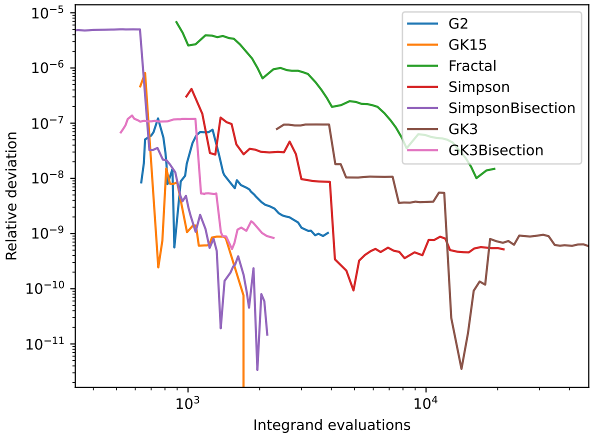


Figure 9.5.4: Relative error as function of the number of integrand evaluations for various adaptive integration strategies for the Lennard Jones problem.

convergence rates for the LJ-problem are shown in Figure 9.5.4. Since we do not have an exact solution, the error is computed relative to GK15 with the lowest setting for the absolute tolerance of 1.1×10^{-28}. For this numerical experiment $d = 3.949$ Å and $E = 19.49$ meV, which are values for xenon [160]. The convergence graphs in Figure 9.5.4 are not as smooth, as the ones for the test problem shown in Figure 9.5.1. One of the conjectured causes for this is that there is no direct control over the number of integrand evaluations, we can only change the tolerance.

9.6 Conclusions

In conclusion we have shown that the adaptive quadrature method, the fractal method, of [151] can be significantly improved without large modifications. By changing two weights the fractal rule can be turned into an adaptive Simpson rule, which has an error of $\mathcal{O}(h^5)$, compared to $\mathcal{O}(h^3)$ for the fractal rule.

We have extensively discussed other quadrature rules, such as Gaussian quadrature and Gauss-Kronrod quadrature. Additionally, the error estimate used to determine when an adaptive method should bisect is discussed. In our numerical experiments it is shown that the bisection method of [153] shows a smoother convergence for GK3 and Simpson's rule compared to the nested error estimates.

To compute the scattering angle, a Runge-Kutta scheme was used with adaptive step size. Computing the scattering angle involves an integral that is improper in two ways, the upper bound is infinite and there is a singularity at the lower bound. Here, we integrate from some initial large r-value towards zero until the singularity is reached. It is argued, based on physical arguments, that this initial r is sufficiently large for any atomic interaction. However, the current method does not *guarantee* that it stops at $r = r_{\mathrm{m}}$, as the term in square brackets in (9.1) may have multiple roots and a step may skip over some roots. In practice $V(r)$ may be tabulated data, which makes it difficult to devise a method to guarantee the correct root is found.

It has been shown that the current MagnumPI implementation using our Runge-Kutta method, and quadrature schemes agree with data shown in the literature [152].

An outlook is to compute the cross section in a way similar to the scattering angle, by treating it as an initial value problem and integrating from $b = 0$ to some large b. There do exist analytical expressions for small scattering angle [13, p. 53], which can be used to estimate the contribution from large b.

Similarly to the idea in [151], if the integrand for the cross section is strongly oscillatory another scheme may be better. As outlook, methods such as the Levin and Filon methods [159, p. 29] could be considered in future versions.

Chapter 10

General conclusions, discussion and outlook

10.1 Part I: Introduction

- The Stefan-Maxwell equations can be derived from Navier-Stokes, however, this requires a significant number of assumptions.
- There appears to be a discrepancy between Whittaker [15] and Holt [8] regarding the pressure tensor.
- Discretization schemes such as the complete flux can be extended to systems of ADR equations.
- An exact definition of sparse linear systems is not straightforward.
- There is a need for efficient solvers for large sparse linear systems, however, there is no optimal Krylov method for every linear system.

10.2 Part II: Reduction and equilibrium

- We have introduced a linear transformation of the system of ADR equations; the Stoichiometric Transformation Method (STM).
- The STM can be used to enforce invariants such as conservation of mass and quasi-neutrality.
- The STM can be extended to a method which computes deviations from equilibrium.
- Using the STM, Newton-Krylov methods can be used to solve systems of ADR equations, without violating $\mathbf{1}^{\mathrm{T}}\boldsymbol{y} = 1$.

- Obtaining the chemical equilibrium efficiently is an important aspect of extending the near equilibrium method.

10.3 Part III: Krylov methods

- We have identified several pitfalls in the implementation of BiCGStab.
- It has been shown that BiCGStab, for problems with strong advection and real eigenvalues, can still break down if the shadow vector is not chosen adequately.
- Numerical experiments have shown that LMR solves a discretized version of the ADR equation.
- An extensive discussion has been presented to improve the implementation of Krylov subspace methods
- The specific design choices for BiCGStab(L), IDR(S) and IDR(S)Stab(L) have been presented, these robust methods are included in the Eigen library.
- A follow-up would be an optimization study, which repeatedly solves a given linear system using BiCGStab, IDR(S) or IDR(S)Stab(L), and aims to find the shadow matrix $\widetilde{\boldsymbol{R}}$ that can solve this system in the smallest number of iterations.

10.4 Part IV: Magnum PI

- We have presented an overview of adaptive quadrature rules and their error estimates.
- Alternatives to the fractal scheme of Colonna have been presented.
- Future work can be to implement a scheme that integrates the cross section as an initial value problem, and integrates parts with orbiting with a scheme designed specifically for oscillatory integrands such as the Levin or Filon schemes.

Bibliography

[1] Stephen Wolfram and M Gad-el Hak. A new kind of science. Appl. Mech. Rev., 56(2), B18–B19, 2003.

[2] Intergovernmental Panel on Climate Change (IPCC). Mitigation of climate change. Contribution of working group III to the sixth assessment report of the intergovernmental panel on climate change, 2022.

[3] Jason Rugolo and Michael J Aziz. Electricity storage for intermittent renewable sources. Energy & Environmental Science, 5(5), 7151–7160, 2012.

[4] Rolando A Rodriguez, Sarah Becker, Gorm B Andresen, Dominik Heide, and Martin Greiner. Transmission needs across a fully renewable european power system. Renewable Energy, 63, 467–476, 2014.

[5] Mark E Dry. The fischer–tropsch process: 1950–2000. Catalysis today, 71(3-4), 227–241, 2002.

[6] Ramses Snoeckx and Annemie Bogaerts. Plasma technology–a novel solution for co 2 conversion? Chemical Society Reviews, 46(19), 5805–5863, 2017.

[7] Robert MM Mattheij, Sjoerd W Rienstra, and JHM Ten Thije Boonkkamp. Partial differential equations: modeling, analysis, computation. SIAM, 2005.

[8] E Howard Holt, Richard E Haskell, and Cap Haskell. Foundations of plasma dynamics. Macmillan, 1965.

[9] Uwe Stamm. Extreme ultraviolet light sources for use in semiconductor lithography—state of the art and future development. Journal of Physics D: Applied Physics, 37(23), 3244, 2004.

[10] Delphine Merche, Nicolas Vandencasteele, and François Reniers. Atmospheric plasmas for thin film deposition: A critical review. Thin Solid Films, 520(13), 4219–4236, 2012.

[11] N Venkatramani. Industrial plasma torches and applications. Current Science, pages 254–262, 2002.

[12] Julia Heinlin, Georg Isbary, W Stolz, G Morfill, M Landthaler, T Shimizu, B Steffes, T Nosenko, JL Zimmermann, and S Karrer. Plasma applications in medicine with a special focus on dermatology. Journal of the European Academy of Dermatology and Venereology, 25(1), 1–11, 2011.

[13] Michael A. Lieberman and Allan J. Lichtenberg. Principles of Plasma Discharges and Materials Processing. A John Wiley & Sons inc., 2nd edition, 2005.

[14] Robert A Adams and Christopher Essex. Calculus: a complete course, volume 7. Pearson, 2010.

[15] Stephen Whitaker. Derivation and application of the stefan-maxwell equations. Revista mexicana de ingeniería química, 8(3), 213–243, 2009.

[16] Giovangigli, Vincent. Mass conservation and singular multicomponent diffusion algorithms. IMPACT of Computing in Science and Engineering, 2(1), 73–97, 1990.

[17] Geiser, Jürgen. Iterative solvers for the Maxwell–Stefan diffusion equations: Methods and applications in plasma and particle transport. Cogent Mathematics & Statistics, 2(1), 1092913, 2015.

[18] Kenneth Kuan-yun Kuo and Ragini Acharya. Applications of turbulent and multiphase combustion. John Wiley & Sons, 2012.

[19] Frank M White and Joseph Majdalani. Viscous fluid flow, volume 3. McGraw-Hill New York, 2006.

[20] R Byron Bird. Transport phenomena. Appl. Mech. Rev., 55(1), R1–R4, 2002.

[21] Richard J Bearman and John G Kirkwood. Statistical mechanics of transport processes. xi. equations of transport in multicomponent systems. The Journal of Chemical Physics, 28(1), 136–145, 1958.

[22] John C Slattery. Advanced transport phenomena. Cambridge University Press, 1999.

[23] Jesper Frederikus Jacobus Janssen. Equilibrium and transport in molecular plasmas. PhD thesis, 2016.

[24] Samaneh Tadayon Mousavi. Modeling of microwave induced plasmas. PhD thesis, Department of Applied Physics, 2020.

[25] Stephen Whitaker and RL Pigford. Thermal diffusion in liquids. Industrial & Engineering Chemistry, 50(7), 1026–1032, 1958.

[26] S. Patankar. Numerical Heat Transfer and Fluid Flow. Taylor & Francis, 2018.

[27] ten Thije Boonkkamp, JHM and Kumar, BV Rathish and Kumar, Sunil and Pargaei, M. Complete flux scheme for conservation laws containing a linear source. In Numerical Mathematics and Advanced Applications ENUMATH 2015, pages 23–31. Springer, 2016.

[28] A. Sivas, M. Manguoglu, J.H.M. ten Thije Boonkkamp, M.J.H. Anthonissen. Discretization and parallel iterative schemes for advection-diffusion-reaction problems. In Numerical Mathematics and Advanced Applications ENUMATH 2015, pages 275–283. Springer, 2016.

[29] J.H.M. ten Thije Boonkkamp, J. van Dijk, L. Liu, K.S.C. Peerenboom. Extension of the complete flux scheme to systems of conservation laws. Journal of Scientific Computing, 53(3), 552–568, 2012.

[30] JHM ten Thije Boonkkamp and MJH Anthonissen. The finite volume-complete flux scheme for advection-diffusion-reaction equations. Journal of Scientific Computing, 46(1), 47–70, 2011.

[31] J van Dijk, RAM van Gestel, Chris EM Schoutrop, and JHM Boonkkamp. Novel flux approximation schemes for systems of coupled advection-diffusion-reaction equations. In Numerical Mathematics and Advanced Applications ENUMATH 2019, pages 313–321. Springer, 2021.

[32] Y. Saad. Iterative Methods for Sparse Linear Systems: Second Edition. SIAM, 2003.

[33] H.A. van der Vorst. Iterative Krylov Methods for large linear Systems, volume 13. Cambridge University Press, 2003.

[34] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

[35] V. Faber, T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. SIAM Journal on Numerical Analysis, 21(2), 352–362, 1984.

[36] J. Liesen, Z. Strakoˇs. On optimal short recurrences for generating orthogonal Krylov subspace bases. SIAM review, 50(3), 485–503, 2008.

[37] C. Lanczos. Solution of systems of linear equations by minimized iterations. J. Res. Nat. Bur. Standards, 49(1), 33–53, 1952.

[38] H.A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 13(2), 631–644, 1992.

[39] MATLAB. R2019a. The MathWorks Inc., 2019.

[40] G. Guennebaud, B. Jacob, et al. Eigen. URl: http://eigen.tuxfamily.org, 2010.

[41] Van Dijk, Jan and Peerenboom, Kim and Jimenez, Manuel and Mihailova, Diana and Van der Mullen, Joost. The plasma modelling toolkit PLASIMO. Journal of Physics D: Applied Physics, 42(19), 194012, 2009.

[42] COMSOL Multiphysics® v. 5.6. www.comsol.com. COMSOL AB, Stockholm, Sweden.

[43] Sleijpen, G.L.G. and Van der Vorst, H.A. Optimal iteration methods for large linear systems of equations. Notes on Numerical Fluid Mechanics, 45, 291–291, 1993.

[44] Y. Saad, M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 7(3), 856–869, 1986.

[45] G.L.G. Sleijpen, H.A. van der Vorst, D.R. Fokkema. BiCGstab (ell) for linear equations involving unsymmetric matrices with complex spectrum. Electronic Transactions on Numerical Analysis., 1, 11–32, 1993.

[46] P. Sonneveld, M.B. van Gijzen. IDR(s): A Family of Simple and Fast Algorithms for Solving Large Nonsymmetric Systems of Linear Equations. SIAM Journal on Scientific Computing, 31, 1035–1062, 2008.

[47] G.L.G. Sleijpen, M.B. van Gijzen. Exploiting BiCGstab (l) strategies to induce dimension reduction. SIAM Journal on Scientific Computing, 32(5), 2687–2709, 2010.

[48] G.L.G. Sleijpen, M.B. van Gijzen. Exploiting BiCGstab(L) Strategies to Induce Dimension Reduction. SIAM Journal on Scientific Computing, 32, 1035–1062, 2010.

[49] Chris EM Schoutrop, Jens Wehner, Jan van Dijk, and Adithya Vijaykumar. Open source BiCGStab(L) implementation. 2021.

[50] Chris EM Schoutrop, Jens Wehner, and Jan van Dijk. Open source IDR(S) implementation. 2021.

[51] Chris EM Schoutrop, Mischa Senders, Lex Kuijpers, Jens Wehner, Jan van Dijk, and Adithya Vijaykumar. Open source IDR(S)Stab(L) implementation. 2021.

[52] Pope, Daniel N and Gogos, George. A new multicomponent diffusion formulation for the finite-volume method: Application to convective droplet combustion. Numerical Heat Transfer, Part B: Fundamentals, 48(3), 213–233, 2005.

[53] Riedel, Uwe. A finite volume scheme on unstructured grids for stiff chemically reacting flows. Combustion Science and Technology, 135(1-6), 99–116, 1998.

[54] Wang, Weizong and Snoeckx, Ramses and Zhang, Xuming and Cha, Min Suk and Bogaerts, Annemie. Modeling plasma-based CO2 and CH4 conversion in mixtures with N2, O2, and H2O: the bigger plasma chemistry picture. The Journal of Physical Chemistry C, 122(16), 8704–8723, 2018.

[55] K.S.C. Peerenboom, J. van Dijk, W.J. Goedheer, J.J.A.M. van der Mullen. On the ambipolar constraint in multi-component diffusion problems. Journal of Computational Physics, 230, 3651–3655, 2011.

[56] ten Thije Boonkkamp, JHM and van Dijk, Jan and Liu, Lei and Peerenboom, KSC. Extension of the complete flux scheme to systems of conservation laws. Journal of Scientific Computing, 53(3), 552–568, 2012.

[57] Giovangigli, Vincent. Multicomponent transport algorithms for partially ionized mixtures. Journal of Computational Physics, 229(11), 4117–4142, 2010.

[58] James N. Butler and Richard S. Brokaw. Thermal conductivity of gas mixtures in chemical equilibrium. The Journal of Chemical Physics, 26(6), 1636–1643, 1957.

[59] Rini, P. D. vanden Abeele, G. Degrez. Closed form for the equations of chemically reacting flows under local thermodynamic equilibrium. Phys. Rev. E, 72, 011204, Jul 2005.

[60] S. Kräutle, P. Knabner. A reduction scheme for the coupled multicomponent transport-reaction problems in porous media: Generalization to problems with heterogeneous equilibrium reactions. Water Resources Research, 43, 244–258, Mar 2007.

[61] Kräutle, S. and Knabner, P. A new numerical reduction scheme for fully coupled multicomponent transport-reaction problems in porous media. Water resources research, 41(9), 2005.

[62] Fan, Yaqing and Durlofsky, Louis J and Tchelepi, Hamdi A. A fully-coupled flow-reactive-transport formulation based on element conservation, with application to CO2 storage simulations. Advances in Water Resources, 42, 47–61, 2012.

[63] Giovangigli, V and Graille, B. Asymptotic stability of equilibrium states for ambipolar plasmas. Mathematical models and methods in applied sciences, 14(09), 1361–1399, 2004.

[64] Geiser, Juergen and Hueso, Jose L and Martinez, Eulalia. Adaptive Iterative Splitting Methods for Convection-Diffusion-Reaction Equations. Mathematics, 8(3), 302, 2020.

[65] Geiser, Juergen. Multicomponent and multiscale systems: Theory, methods, and applications in engineering. Springer, 2015.

[66] Geiser, Jürgen. Multiscale modelling of solute transport through porous media using homogenization and splitting methods. Mathematical and Computer Modelling of Dynamical Systems, 22(3), 221–243, 2016.

[67] Geiser, Jürgen. Mobile and immobile fluid transport: Coupling framework. International journal for numerical methods in fluids, 65(8), 877–922, 2011.

[68] Boudin, Laurent and Grec, Bérénice and Salvarani, Francesco. The Maxwell-Stefan diffusion limit for a kinetic model of mixtures. Acta Applicandae Mathematicae, 136(1), 79–90, 2015.

[69] Krishna, Rajamani. Diffusion in multicomponent electrolyte systems. The Chemical Engineering Journal, 35(1), 19–24, 1987.

[70] K.S.C. Peerenboom, J. van Dijk, W.J. Goedheer, G. Degrez, J.J.A.M. van der Mullen. A finite volume model for multi-component diffusion in magnetically confined plasmas. Journal of Physics D: Applied Physics, 44(19), 194006, 2011.

[71] J.D. Ramshaw. Simple Approximation for Thermal Diffusion in Ionized Gas Mixtures. 1996.

[72] Ramshaw, John D. Short communication simple approximation for thermal diffusion in gas mixtures. J Non Equilib Thermodyn, 21, 99–101, 1996.

[73] K.S.C. Peerenboom. Modeling of magnetized expanding plasmas. PhD thesis, Department of Applied Physics, 2012.

[74] K. Peerenboom, J. Van Boxtel, J. Janssen, J. and Van Dijk. A conservative multicomponent diffusion algorithm for ambipolar plasma flows in local thermodynamic equilibrium. Journal of Physics D: Applied Physics, 47(42), 425202, 2014.

[75] C.F. van Loan, G.H. Golub. Matrix Computations. Johns Hopkins University Press, 1983.

[76] Fridman, Alexander. Plasma chemistry. Cambridge university press, 2008.

[77] Maher I.. Boulos and Fauchais, Pierre and Pfender, Emil. Thermal plasmas: fundamentals and applications. 1994.

[78] van der Heijden, Harm. Modelling of radiative transfer in light sources. Technische Universiteit Eindhoven, 2003.

[79] Gerasimov, G Ya and Shatalov, OP. Kinetic mechanism of combustion of hydrogen–oxygen mixtures. Journal of Engineering Physics and Thermophysics, 86(5), 987–995, 2013.

[80] R. Aerts, T. Martens, A. Bogaerts. Influence of vibrational states on CO2 splitting by dielectric barrier discharges. The Journal of Physical Chemistry C, 116(44), 23257–23273, 2012.

[81] Kozák, Tomáˇs and Bogaerts, Annemie. Splitting of CO2 by vibrational excitation in non-equilibrium plasmas: a reaction kinetics model. Plasma Sources Science and Technology, 23(4), 045004, 2014.

[82] Koelman, Peter and Heijkers, Stijn and Tadayon Mousavi, Samaneh and Graef, Wouter and Mihailova, Diana and Kozak, Tomas and Bogaerts, Annemie and van Dijk, Jan. A Comprehensive Chemical Model for the Splitting of CO2 in Non-Equilibrium Plasmas. Plasma Processes and Polymers, 14(4-5), 1600155, 2017.

[83] Henderson, Sean J and Menart, James A. Equilibrium properties of high-temperature air for a number of pressures. Journal of thermophysics and heat transfer, 22(4), 718–726, 2008.

[84] E.W. Cheney, D.R. Kincaid. Numerical mathematics and computing. Cengage Learning, 2012.

[85] Ruiz, Daniel. A scaling algorithm to equilibrate both rows and columns norms in matrices. Technical report, CM-P00040415, 2001.

[86] H.K. Gummel. A self-consistent iterative scheme for one-dimensional steady state transistor calculations. IEEE Transactions on electron devices, 11(10), 455–465, 1964.

[87] Zagaris, Antonios and Kaper, Hans G and Kaper, Tasso J. Fast and slow dynamics for the computational singular perturbation method. Multiscale Modeling & Simulation, 2(4), 613–638, 2004.

[88] Ashida, S. and Lee, C. and Lieberman, M.A. Spatially averaged (global) model of time modulated high density argon plasmas. Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films, 13(5), 2498–2507, 1995.

[89] T. Rehman. Studies on Plasma-Chemical Reduction. PhD thesis, Eindhoven University of Technology, 2018.

[90] Tachibana, K. Excitation of the 1s5, 1s4, 1s3, and 1s2 levels of argon by low-energy electrons. Physical Review A, 34(2), 1007, 1986.

[91] Lee, C and Lieberman, MA. Global model of Ar, O2, Cl2, and Ar/O2 high-density plasma discharges. Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films, 13(2), 368–380, 1995.

[92] Graef, WAAD. Zero-dimensional models for plasma chemistry. 2012.

[93] Hopwood, Jeffrey A. The role of ionized physical vapor deposition in integrated circuit fabrication. In Thin Films, volume 27, pages 1–7. Elsevier, 2000.

[94] Lu, Junqing and Kushner, Mark J. Effect of sputter heating in ionized metal physical vapor deposition reactors. Journal of Applied Physics, 87(10), 7198–7207, 2000.

[95] Ahmad, Z. and Goedheer, W.J. Optimization and characterization of a Pilot-PSI cascaded arc with non-LTE numerical simulation of Ar, H2 gases. Plasma Sources Science and Technology, 18(1), 015008, 2008.

[96] C. Schoutrop, J. van Dijk, J. ten Thije Boonkkamp. Multicomponent transport in plasmas; exploiting stoichiometry. Journal of Computational Physics, 428, 109979, 2021.

[97] José L Hueso, Eulalia Martínez, and Juan R Torregrosa. Modified newton's method for systems of nonlinear equations with singular jacobian. Journal of Computational and Applied Mathematics, 224(1), 77–83, 2009.

[98] Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. Journal of Computational Physics, 193(2), 357–397, 2004.

[99] HW Drawin and F Emard. Instantaneous population densities of the excited levels of hydrogen atoms and hydrogen-like ions in plasmas. Physica B+ C, 85(2), 333–356, 1976.

[100] Lawrence F Shampine and Mark W Reichelt. The matlab ode suite. SIAM journal on scientific computing, 18(1), 1–22, 1997.

[101] P. Koelman. Chemical aspects of CO2 plasma modelling. PhD thesis, Department of Applied Physics, 4 2019. Proefschrift.

[102] Gianpiero Colonna and A D'angola. A hierarchical approach for fast and accurate equilibrium calculation. Computer physics communications, 163(3), 177–190, 2004.

[103] L. Martinez, A. Dhruv, L. Lin, E. Balaras, M. Keidar. Interaction between a helium atmospheric plasma jet and targets and dynamics of the interface. Plasma Sources Science and Technology, 28(11), 115002, 2019.

[104] D.A. Knoll, P.R. McHugh. Newton-Krylov methods applied to a system of convection-diffusion-reaction equations. Computer Physics Communications, 88(2-3), 141–160, 1995.

[105] J. Lin, S. Reutskiy, C. Chen, J. Lu. A novel method for solving time-dependent 2D advection-diffusion-reaction equations to model transfer in nonlinear anisotropic media. Communications in Computational Physics, 26(1), 233–264, 2019.

[106] B. Fischer, A. Ramage, D.J. Silvester, A.J. Wathen. On parameter choice and iterative convergence for stabilised discretisations of advection–diffusion problems. Computer Methods in Applied Mechanics and Engineering, 179(1-2), 179–195, 1999.

[107] A. Singh, S. Das, S.H. Ong, H. Jafari. Numerical solution of nonlinear reaction–advection–diffusion equation. Journal of Computational and Nonlinear Dynamics, 14(4), 2019.

[108] D.L. Scharfetter, H.K. Gummel. Large-signal analysis of a silicon read diode oscillator. IEEE Transactions on electron devices, 16(1), 64–77, 1969.

[109] G.L.G. Sleijpen, H.A. van der Vorst, D.R. Fokkema. BiCGstab (l) and other hybrid Bi-CG methods. Numerical Algorithms, 7(1), 75–109, 1994.

[110] K. Aihara, K. Abe, E. Ishiwata. A variant of IDRstab with reliable update strategies for solving sparse linear systems. Journal of Computational and Applied Mathematics, 259, 244–258, Mar 2014.

[111] L.T. Yang, R.P. Brent. The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures. In Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings., pages 324–328. IEEE, 2002.

[112] S. Cools, W. Vanroose. The communication-hiding pipelined BiCGStab method for the parallel solution of large unsymmetric linear systems. Parallel Computing, 65, 1–20, 2017.

[113] El Guennouni, A and Jbilou, Khalide and Sadok, Hassane. A block version of BiCGSTAB for linear systems with multiple right-hand sides. Electronic Transactions on Numerical Analysis, 16(129-142), 2, 2003.

[114] K. Ahuja, P. Benner, E. de Sturler, L. Feng. Recycling BiCGSTAB with an application to parametric model order reduction. SIAM Journal on Scientific Computing, 37(5), 429–446, 2015.

[115] Fokkema, Diederik R. Enhanced Implementation of BiCGstab (l) for Solving Linear Systems of Equations. 1996.

[116] G.L.G. Sleijpen and H.A. van der Vorst. Reliable updated residuals in hybrid Bi-CG methods. Computing, 56(2), 141–163, 1996.

[117] J. Liesen, P. Tichỳ. Convergence analysis of Krylov subspace methods. GAMM-Mitteilungen, 27(2), 153–173, 2004.

[118] W. Joubert. Lanczos methods for the solution of nonsymmetric systems of linear equations. SIAM Journal on Matrix Analysis and Applications, 13(3), 926–943, 1992.

[119] Y. Yasuda. S. Sakamoto, Y. Kosaka, T. Sakuma, N. Okamoto, T. Oshima. Numerical analysis of large-scale sound fields using iterative methods part I: application of Krylov subspace methods to boundary element analysis. Journal of Computational Acoustics, 15(04), 449–471, 2007.

[120] N. Okamoto, R. Tomiku, T. Otsuru, Y. Yasuda. Numerical analysis of large-scale sound fields using iterative methods part II: application of Krylov subspace methods to finite element analysis. Journal of Computational Acoustics, 15(04), 473–493, 2007.

[121] J. Spanier, K.B. Oldham. An Atlas of Functions. Hemisphere Publishing Corporation New York, 1987.

[122] M. Abramowitz and I.A. Stegun. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, volume 55. US Government printing office, 1964.

[123] Noschese, Silvia and Pasquini, Lionello and Reichel, Lothar. Tridiagonal Toeplitz matrices: properties and novel applications. Numerical Linear Algebra with Applications, 20(2), 302–326, 2013.

[124] Horn, Roger A and Johnson, Charles R. Topics in Matrix Analysis, 1991. Cambridge University Presss, Cambridge, 37, 39, 1991.

[125] Khoromskaia, Venera and Khoromskij, Boris N and Otto, Felix. Numerical study in stochastic homogenization for elliptic partial differential equations: Convergence rate in the size of representative volume elements. Numerical Linear Algebra with Applications, 27(3), 2296, 2020.

[126] O.G. Ernst. Residual-minimizing Krylov subspace methods for stabilized discretizations of convection-diffusion equations. SIAM Journal on Matrix Analysis and Applications, 21(4), 1079–1101, 2000.

[127] Howard C Elman, David J Silvester, and Andrew J Wathen. Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics. Numerical Mathematics and Scie, 2014.

[128] F. Hiai, D. Petz. Introduction to Matrix Analysis and Applications. Springer Science & Business Media, 2014.

[129] David Kay, Daniel Loghin, and Andrew Wathen. A preconditioner for the steady-state navier–stokes equations. SIAM Journal on Scientific Computing, 24(1), 237–256, 2002.

[130] Yan-Fei Jing, Ting-Zhu Huang, Yong Duan, and Bruno Carpentieri. A comparative study of iterative solutions to linear systems arising in quantum mechanics. Journal of Computational Physics, 229(22), 8511–8520, 2010.

[131] Bruno Carpentieri, Iain S Duff, Luc Giraud, and M Magolu monga Made. Sparse symmetric preconditioners for dense linear systems in electromagnetism. Numerical linear algebra with applications, 11(8-9), 753–771, 2004.

[132] Meurant, Gérard A and Tebbens, Jurjen Duintjer. Krylov Methods for Nonsymmetric Linear Systems: From Theory to Computations, volume 57. Springer Nature, 2020.

[133] Aleksei Nikolaevich Krylov. On the numerical solution of the equation by which, in technical matters, frequencies of small oscillations of material systems are determined (in russian). (4), 491–539, 1931.

[134] G. L. G. Sleijpen. Optimal iteration methods for large linear systems of equations. In W. Hiller and E. Krausse, editors, Workshop on Aspects of Parallelization regarding Finite Elements and Ocean Modelling, volume 52 of Berichte aus dem Fachbereich Physik, Bremerhaven, 1994. Alfred Wegener Inst.

[135] Hestenes, Magnus Rudolph and Stiefel, Eduard. Methods of conjugate gradients for solving linear systems, volume 49. NBS Washington, DC, 1952.

[136] Peter Sonneveld. Cgs, a fast lanczos-type solver for nonsymmetric linear systems. SIAM journal on scientific and statistical computing, 10(1), 36–52, 1989.

[137] Fletcher, Roger. Conjugate gradient methods for indefinite systems. In Numerical analysis, pages 73–89. Springer, 1976.

[138] G.L.G. Sleijpen, D.R. Fokkema. BiCGstab (l) for linear equations involving unsymmetric matrices with complex spectrum. Electronic Transactions on Numerical Analysis, 1(11), 2000, 1993.

[139] C.C. Paige, M.A. Saunders. Solution of sparse indefinite systems of linear equations. SIAM journal on numerical analysis, 12(4), 617–629, 1975.

[140] Roland W Freund and Noël M Nachtigal. Qmr: a quasi-minimal residual method for non-hermitian linear systems. Numerische mathematik, 60(1), 315–339, 1991.

[141] G.L.G. Sleijpen, H.A. van der Vorst. Maintaining convergence properties of BiCGstab methods in finite precision arithmetic. Numerical algorithms, 10(2), 203–223, 1995.

[142] M.H. Gutknecht. Variants of BiCGStab for matrices with complex spectrum. SIAM journal on scientific computing, 14(5), 1020–1033, 1993.

[143] Aihara, Kensuke and Abe, Kuniyoshi and Ishiwata, Emiko. A variant of IDRstab with reliable update strategies for solving sparse linear systems. Journal of Computational and Applied Mathematics, 259, 244–258, 2014.

[144] V. Faber, J. Liesen, P. Tichỳ. The Faber–Manteuffel theorem for linear operators. SIAM Journal on Numerical Analysis, 46(3), 1323–1337, 2008.

[145] Petr Tichỳ and Jan Zítko. Derivation of bicg from the conditions defining lanczos' method for solving a system of linear equations. Applications of Mathematics, 43(5), 381–388, 1998.

[146] Eigen documentation for the cholesky decomposition. https://eigen.tuxfamily.org/dox/classEigen_1_1LLT.html.

[147] Benchmark of dense decompositions. https://eigen.tuxfamily.org/dox/group_DenseDecompositionBenchmark.html.

[148] Alberto Leon-Garcia. Probability and random processes for electrical engineering. Pearson Education India, 1994.

[149] Martin B Gijzen and Peter Sonneveld. IDR(s) MATLAB implementation manual.

[150] J. van Dijk S. Verhoeven C. Schoutrop D. Boer R. Bude J.Janssen, W. Graef. MagnumPI https://gitlab.com/magnumpi/magnumpi, 2022.

[151] G. Colonna, A. Laricchiuta. General numerical algorithm for classical collision integral calculation. Computer Physics Communications, 178(11), 809–816, 2008.

[152] Sergey A Khrapak. Accurate transport cross sections for the lennard-jones potential. The European Physical Journal D, 68(10), 1–6, 2014.

[153] P. Gonnet. A review of error estimation in adaptive quadrature. ACM Computing Surveys, 08 2012.

[154] D. Kahaner, C. Moler, S. Nash. Numerical methods and Software. Prentice Hall, 1989.

[155] K.E. Atkinson. An introduction to numerical analysis. John Wiley and Sons, 2 edition, 1989.

[156] Newman, M. Computational Physics. CreateSpace Independent Publishing Platform, 2012.

[157] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. Numerical recipes 3rd edition: The art of scientific computing. Cambridge university press, 2007.

[158] Sven Ehrich. Stieltjes polynomials and the error of gauss-kronrod quadrature formulas. In Applications and computation of orthogonal polynomials, pages 57–77. Springer, 1999.

[159] Deaño, Alfredo and Huybrechs, Daan and Iserles, Arieh. Computing highly oscillatory integrals. SIAM, 2017.

[160] Gábor Rutkai, Monika Thol, Roland Span, and Jadran Vrabec. How well does the lennard-jones potential represent the thermodynamic properties of noble gases? Molecular Physics, 115(9-12), 1104–1121, 2017.

[161] Denis Demidov. Amgcl: An efficient, flexible, and extensible algebraic multigrid implementation. Lobachevskii Journal of Mathematics, 40(5), 535–546, 2019.

Acknowledgments

I would like to thank my highschool mathematics teacher Erich Mommers for being the inspiration to continue my quest of exploring the exact sciences. Furthermore, I would also like to thank both Jan van Dijk and Jan ten Thije Boonkkamp for their continued support, the numerous insightful discussions and the extensive proofreading.

An inspiration for best practices in C++ and soft-skills is Jens Wehner. It was a pleasure to work together on completing the Krylov subspace solvers IDR(S), BiCGStab(L), IDR(S)Stab(L) and several related topics.

Next, I would also like to thank my colleagues over the years with whom I've had the fortune to share an office; Peter Koelman, Samaneh Tadayon Mousavi, Thijs Hazenberg, Rick Budé, Daan Boer, Jim Gulpen, Robert van Gestel and Anne van Gils. For both the insightful discussions and social interactions. Furthermore I'd like to thank the members of the PLASIMO team to allow me to partake in the development.

Important contributors over the years have been several Bachelor candidates who assisted me in my research; Mischa Senders, Joris van Laanen, Lex Kuijpers, Bastiaan van Hoorn, Niels Meijers, Stan van Kraaij.

Additionally, I would like to thank my uncle Frank Delbressine for the several occasions I have been stranded in Eindhoven, and for his advice on completing a PhD.

Most importantly, my gratitude goes out to my parents, by providing the foundation which enabled me to pursue my academic adventures.

As a final note I would like to express my gratitude to Hoepert with whom I've made the most epic of journeys in this academic adventure.

Curriculum Vitae

Chris Schoutrop was born on 23 November 1994 in Maastricht, the Netherlands. He finished his high school studies in 2012, focussing on the STEM subjects.

He continued to study Applied Physics at the Eindhoven University of Technology. Here he became interested in computational physics. He obtained his Bachelor of Science degree cum laude, after completing a thesis at the Turbulence and Vortex Dynamics group (WDY).

Afterward he continued his studies on the track of Plasma Physics and Radiation Technology, with several courses from the track on Transport Physics. He obtained his Master of Science after finishing his thesis on Electron beam generation by field dependent photoionization at the Coherence and Quantum Technology group (CQT).

He started a PhD project in 2018 in the group Elementary Processes in Gas Discharges (EPG) with the objective to make complex plasma-physical computations more efficient. The results of the PhD project are presented in this dissertation.

List of publications

Peer-reviewed journals

1. Schoutrop, C., van Dijk, J., & ten Thije Boonkkamp, J. (2021). Multicomponent transport in plasmas; exploiting stoichiometry. Journal of Computational Physics.
2. Schoutrop, C., van Dijk, J., & ten Thije Boonkkamp, J. (2022). Reliability Investigation of BiCGStab and IDR Solvers for the Advection-Diffusion-Reaction Equation. Communications in Computational Physics.
3. van Dijk, J., van Gestel, R., Schoutrop, C., & ten Thije Boonkkamp, J. (2021). Novel Flux Approximation Schemes for Systems of Coupled Advection-Diffusion-Reaction Equations. In Numerical Mathematics and Advanced Applications ENUMATH 2019 (pp. 313-321). Springer, Cham.

Selected peer-reviewed conference papers

1. Schoutrop, C., ten Thije Boonkkamp, J. H., Graef, W., & van Dijk, J. (2019, March). Stoichiometric projection methods for plasma simulations. In 24th International Symposium on Plasma Chemistry (ISPC 24), June 9-14, 2019, Naples, Italy. International Plasma Chemistry Society (IPCS).

Oral presentations

1. Improving reliability of BiCGStab. WOPMAS 2022, Eindhoven.
2. Near-equilibrium approximation for advection-diffusion-reaction systems in plasmas. JMBC symposium 2022, Lunteren.
3. Exact element conservation for advection-diffusion-reaction systems. Physics at Veldhoven 2022, Veldhoven.
4. Multicomponent transport in plasmas; exploiting stoichiometry. JMBC symposium-2019, Lunteren.

5. Linearly transformed schemes for numerical simulations. WELTPP-2019, Kerkrade.
6. Linearly transformed discretization schemes for plasma simulations. WOPMAS 2019, Bochum (Germany).

Selected posters

1. Complex computational plasmas: an outlook. WELTPP 2018, Kerkrade.
2. Stoichiometric projection methods for plasma simulations. ISPC24 2019, Naples (Italy).
3. An open source implementation of the IDR(S)Stab(L) solver. ACOS-2019, Eindhoven.

Student reports

1. Van Hoorn, B.M.V. (2020). Using linear computational singular perturbation to simplify 1D diffusion-advection-reaction problems.
2. Van Laanen, J.C.A. (2020). Multidimensional Intrinsic Low-Dimensional Manifolds in Plasma Modelling.
3. Kuijpers, L. (2019). Systematic Comparison of Iterative Solvers.
4. Senders, M.J. (2019). Implementing and Testing a New Variant of IDRStab.
5. Meijers, N.F.T. (2020). Interpolation of Tabulated Atomic Potentials.
6. Van Kraaij, S.A.T. (2021). Systematic comparison of preconditioned Krylov Subspace Methods for an Electron Cyclotron Resonance problem.

Code contributions to open source projects

1. IDR(S) Krylov subspace solver in Eigen [50].
2. BiCGStab(L) Krylov subspace solver in Eigen [49].
3. IDR(S)Stab(L) Krylov subspace solver in Eigen [51].
4. Deprecation of Eigen::MappedSparseMatrix in AMGCL [161].
5. Various additions to MagnumPI [150].