TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Development of a Multicomponent Adsorption Solver in OpenFOAM

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs unter der Leitung von

Ass.Prof. Dipl.-Ing. Dr. Michael Harasek

und der Betreuung von

Projektass. Bahram Haddadi Sisakht, MSc.

und

Projektass. Dipl.-Ing. Christian Jordan

am

E166 Institut für Verfahrenstechnik, Umwelttechnik und Technische Biowissenschaften

eingereicht an der

Technischen Universität Wien
Fakultät für Maschinenwesen und Betriebswissenschaften

von

Clemens Gößnitzer
Matrikelnummer 1126267
Große Neugasse 22–24/1/15, 1040 Wien

Wien, im März 2016

C.Gößnitzer

Abstract

The aim of this thesis is to implement multicomponent adsorption models in the custom OpenFOAM computational fluid dynamics solver `adsorpFoam` developed at Vienna University of Technology. This includes equilibrium and kinetics models. For this, two multicomponent equilibrium models, the Extended Langmuir Model ELM and the Ideal Adsorbed Solution Theory IAST, are used. They solely depend on single-component isotherm data. For interspecies-dependent kinetics, a diffusion-based approach is chosen.

As the results of a zero-dimensional model show, the quality of prediction of equilibria is dependent on the chosen system of species. The model predictions are compared with experimental data of six multicomponent systems taken from literature.

If experimental data are available, a simple extension to the ELM is possible. This is done by introducing empirical interaction coefficients to account for competitive adsorption, which improves the prediction of most systems. For this approach, data of multicomponent adsorption experiments have to be obtained.

OpenFOAM, an open-source suite of CFD programs, is used in this thesis. The main reasons for this choice are its openness and extensibility.

The implementation in OpenFOAM includes the adaptation of the governing equations, calculation of adsorption equilibrium loading and rate of adsorption. Additionally, the released heat of adsorption increases the temperature distribution of the adsorbing walls. If the calculated rate of adsorption leads to nonphysical results, e.g. more mass adsorbing in one cell than available, limiters are applied.

At the end, a working multicomponent adsorption model was included in the solver `adsorpFoam`. It allows to define multiple adsorbing sites with different parameters per species and site. This implementation is a first step towards multicomponent mass transfer and serves as a basis for further work on the three-dimensional simulation of multicomponent adsorption.

Kurzfassung

In dieser Arbeit wird die Implementierung von Mehrkomponenten-Adsorptionsmodellen in den Löser für numerische Strömungssimulation (Computational fluid dynamics CFD) `adsorpFoam`, entwickelt an der Technischen Universität Wien, beschrieben. Es werden Gleichgewichts- und Kinetik-Modelle präsentiert. Die zwei verwendeten Modelle für das Gleichgewicht von Mehrkomponenten-Adsorption sind einerseits das Extended Langmuir Model ELM, und andererseits die Ideal Adsorbed Solution Theory IAST. Diese beiden Modelle basieren auf Einzelkomponenten-Isothermen. Um gegenseitige Beeinflussung der Spezies bei der Adsorption zu berücksichtigen, wurde ein Kinetik-Modell entwickelt, welches auf Diffusion basiert.

Die Ergebnisse eines null-dimensionalen Modells zeigen, dass die Abweichung zwischen Modellvorhersage und experimentellen Ergebnis der ermittelten Gleichgewichte von den beteiligten Molekülen abhängt. Die Simulationen wurden mit sechs Mehrkomponenten-Systemen verglichen. Die Daten zu den Experimenten wurden der Literatur entnommen. Wenn experimentelle Daten zur Verfügung stehen, kann das Extended Langmuir Model um sogenannte Interaktionskoeffizienten erweitert werden. Diese empirischen Parameter werden aus den Messergebnissen berechnet und berücksichtigen die gegenseitige Beeinflussung bei der Adsorption. Für diese Erweiterung benötigt man Daten aus Mehrkomponenten-Adsorptionsversuchen.

Das verwendete Programm OpenFOAM ist eine frei verfügbare Sammlung von CFD-Lösern. Die freie Verfügbarkeit des Quelltexts und Erweiterbarkeit waren ausschlaggebend, dieses Programm zu verwenden.

Die Implementierung in OpenFOAM umfasst u.a. die Anpassung der Erhaltungsgleichungen, Berechnungen von Gleichgewichtsbeladungen und Adsorptionraten. Zusätzlich erhöht die freigesetzte Adsorptionswärme die Temperaturverteilung der adsorbierenden Wände. Wenn die berechnete Adsorptionsrate zu physikalisch inkorreten Ergebnissen führen würde, müssen Limiter angewandt werden.

Es wurde ein funktionierendes Modell für Mehrkomponenten-Adsorption in den Löser `adsorpFoam` implementiert. Es ist möglich, mehrere adsorbierende Oberflächen zu definieren, und die Parameter pro Komponente und Oberflächen zu wählen. Dies ist der erste Schritt hin zu einer allgemeinen Lösung für Stoffübergang in der CFD und die Implementierung dient als Basis, um weitere Modelle in der Zukunft hinzufügen zu können.

Danksagung

Ich möchte mich an dieser Stelle bei all jenen bedanken, die diese Diplomarbeit ermöglicht haben:
Meine Betreuer Bahram und Christian haben mich stets unterstützt, wenn ich einmal nicht weiter wusste. Danke für die interessanten Gespräche abseits von CFD, Programmieren und Adsorption und für das gute Essen im Büro. Christian hat sich weiters die Mühe gemacht, diese Arbeit Korrektur zu lesen.
Mein Dank gilt auch Michael, der zwar immer beschäftigt ist, trotzdem aber Zeit für mich gefunden hat, um mir wertvolle Hinweise zu geben.
Danke auch an Benjamin, der mich bei der intensiven Fehlersuche unterstützt hat. Zwei Paar Augen sehen mehr als eines.
Weiters möchte ich den Kollegen am Institut für Strömungsmechanik und Wärmeübertragung, insbesondere Georg, Herbert und Johannes, dafür danken, dass sie mein Interesse an Thermodynamik, Strömungsmechanik und der Linux-Kommandozeile geweckt und verstärkt haben.
Respekt und Anerkennung gebührt meinen Eltern, die mich immer unterstützen und ohne deren Hilfe und Motivation mein Leben ganz anders verlaufen wäre. Danke für alles!

Contents

Abstract b

Kurzfassung c

Danksagung d

1. Introduction 6

2. Thermodynamics of Adsorption 8
2.1. Adsorption Equilibrium . . . 10
2.1.1. Single-component Adsorption . . . 12
2.1.2. Multicomponent Adsorption . . . 14
2.2. Adsorption Kinetics . . . 18
2.2.1. Linear Driving Force . . . 19
2.2.2. Diffusion-based Kinetics . . . 19

3. Implementation in Octave 21
3.1. Calculation of Adsorption Coefficients . . . 21
3.2. Extended Langmuir Model . . . 21
3.3. Extended Langmuir Model with Interaction Coefficients . . . 21
3.4. Ideal Adsorbed Solution Theory . . . 22
3.5. Diffusion Kinetics . . . 24

4. Validation and Results in Octave 25
4.1. Validation . . . 25
4.2. Equilibrium Models . . . 25
4.2.1. Binary Systems . . . 26
4.2.2. Ternary Systems . . . 32
4.2.3. Comparison between ELM and ELM with IAC . . . 36
4.3. Kinetics Models . . . 36

5. Computational Fluid Dynamics 40
5.1. Mathematical Fundamentals . . . 40
5.1.1. Discretisation . . . 41
5.1.2. Finite-difference Method . . . 45
5.1.3. Weighted Residual Methods . . . 45
5.2. Finite-volume Method . . . 47
5.2.1. Interpolation . . . 47

5.3. Conservation Equations . . . 49
5.3.1. Total Mass Balance . . . 50
5.3.2. Partial Mass Balance . . . 50
5.3.3. Momentum Balance . . . 50
5.3.4. Energy Balance . . . 51
5.4. Pressure-velocity Coupling . . . 52
5.4.1. PISO Algorithm . . . 52
5.4.2. SIMPLE Algorithm . . . 53
5.4.3. PIMPLE Algorithm . . . 53
5.5. Prediction of Material Properties . . . 53
5.5.1. Diffusion Coefficients in Gas Mixtures . . . 55

6. Introduction to OpenFOAM 56
6.1. User Side . . . 56
6.1.1. Preprocessing . . . 57
6.1.2. Starting the Simulation . . . 57
6.1.3. Postprocessing . . . 57
6.2. Programming Side . . . 57
6.2.1. General Structure of a Solver . . . 57
6.2.2. File Input and Output . . . 58
6.2.3. Data Types . . . 61
6.2.4. Partial Differential Equations . . . 61
6.2.5. Turbulence Modelling . . . 61
6.3. Solver for Flows with Chemical Reactions . . . 61
6.4. Solver for Flows with Single-component Henry Adsorption . . . 63
6.4.1. Adaptation of Conservation Equations . . . 63
6.4.2. Temperature and Species Boundary and Initial Conditions . . . 65

7. Implementation in OpenFOAM 66
7.1. Reading Input Parameters . . . 66
7.2. Adsorption Calculations . . . 67
7.2.1. Preparations . . . 67
7.2.2. Calculating the Adsorption Equilibrium . . . 68
7.2.3. Calculating the Rate of Adsorption . . . 68
7.2.4. Applying Limiters . . . 69
7.2.5. Division by Area . . . 69
7.2.6. Adsorption Enthalpy . . . 69
7.2.7. Pitfalls . . . 70
7.3. Adaptation of Conservation Equations . . . 71
7.4. Information Output . . . 71
7.5. Boundary Conditions . . . 72
7.6. Example Case Setup . . . 72
7.6.1. The adsorptionProperties Dictionary . . . 72

8. Validation and Results in OpenFOAM 75
8.1. Validation . . . 75
8.2. Test Cases . . . 77
8.2.1. Cuboid . . . 77
8.2.2. Packed Bed . . . 78

9. Summary, Discussion and Outlook **91**

Bibliography **93**

A. Octave **i**
A.1. Implementation Code i
A.2. Results of Implementation in Octave ix

B. Example Case Setup in OpenFOAM **xvi**
B.1. 0 directory xvi
B.2. constant directory xx
B.3. system directory xxvii

Index **xxxi**

List of Figures

2.1. Nomenclature of adsorption 8
2.2. The six types of physisorption after IUPAC 9
2.3. Four different single-component adsorption isotherms 13

3.1. Flowchart of the algorithm for solving IAST 23

4.1. Adsorbed amount for the system CH_4–CO 26
4.2. absolute error of the mole fraction of CO for the system CH_4–CO_2 27
4.3. absolute error of the adsorbed amount for the system CH_4–CO_2 27
4.4. Adsorbed amount for the system CH_4–CO_2 28
4.5. absolute error of the mole fraction of CH_4 for the system CH_4–CO_2 28
4.6. absolute error of the adsorbed amount for the system CH_4–CO_2 29
4.7. Adsorbed amount for the system CO–H_2 29
4.8. absolute error of the mole fraction for the system CO–H_2 30
4.9. absolute error of the adsorbed amount for the system CO–H_2 30
4.10. Adsorbed amount for the system CO_2–CO 31
4.11. absolute error for the system CO_2–CO 31
4.12. absolute error of the adsorbed amount for the system CO_2–CO 32
4.13. Adsorbed amount for the system CH_4–CO–H_2 33
4.14. absolute error of the mole fraction for the system CH_4–CO_2–H_2 33
4.15. absolute error of the adsorbed amount for the system CH_4–CO–H_2 34
4.16. Adsorbed amount for the system CH_4–CO_2–H_2 34
4.17. absolute error of the mole fraction for the system CH_4–CO_2–H_2 35
4.18. absolute error of the adsorbed amount for the system CH_4–CO_2–H_2 . . . 35
4.19. Adsorbed amount for the system CH_4–CO–H_2 36
4.20. Adsorbed amount for the system CO_2–CO 37
4.21. Diffusion-based kinetics for the system CH_4–H_2–CO_2–CO 37
4.22. Diffusion-based kinetics for the system CH_4–CO_2–CO 38
4.23. Diffusion-based kinetics for the system CH_4–CO 38
4.24. Diffusion-based kinetics with initial loading for the system CH_4–CO . . . 39

5.1. Three different types of grids 42
5.2. Three different discretisation schemes and the actual derivative 43
5.3. The finite-volume method on a two-dimensional grid 47
5.4. Three different interpolation schemes 48
5.5. Flowchart of the PISO algorithm 52
5.6. Flowchart of the SIMPLE algorithm 53
5.7. Flowchart of the PIMPLE algorithm 54

6.1. Flowchart of a generic OpenFOAM solver 59
6.2. Two-dimensional representation of data points for different field variable types 62

7.1. Flowchart of the adsorption implementation 70

8.1. Test and validation case with three cells 75
8.2. Outline of the square tunnel 77
8.3. Dynamic behaviour of the square tunnel simulation with ELM 78
8.4. Dynamic behaviour of the square tunnel simulation with IAST 78
8.5. Geometry of the packed bed 79
8.6. Results for pressure and velocity magnitude of the steady-state calculation 81
8.7. Distribution of the gas mass fraction of carbon monoxide in the packed bed 82
8.8. Distribution of the gas mass fraction of methane in the packed bed 82
8.9. Distribution of the gas mass fraction of carbon dioxide in the packed bed 83
8.10. Distribution of the gas mass fraction of hydrogen in the packed bed . . . 83
8.11. Velocity magnitude profile of the packed bed 84
8.12. Distribution of the pressure inside the packed bed 85
8.13. Pressure drop in the packed bed 85
8.14. Temperature distribution in the packed bed 86
8.15. Distribution of the temperature inside the packed bed 86
8.16. Distribution of the adsorbed amount of carbon monoxide in the packed bed 87
8.17. Distribution of the adsorbed amount at equilibrium of carbon monoxide in the packed bed 87
8.18. Distribution of the adsorbed amount of methane in the packed bed 88
8.19. Distribution of the adsorbed amount of carbon dioxide in the packed bed 88
8.20. Adsorbed amount of the three components in the packed bed 89
8.21. Relative total continuity errors for the packed bed simulation 90

List of Tables

8.1. Boundary conditions for the packed bed simulation 80

A.1. Adsorbed amount according to experiment, ELM and IAST for the system CH_4–CO x
A.2. Adsorbed amount according to experiment, ELM and IAST for the system CH_4–CO_2 xi
A.3. Adsorbed amount according to experiment, ELM and IAST for the system CO–H_2 xii
A.4. Adsorbed amount according to experiment, ELM and IAST for the system CO_2–CO xiii
A.5. Adsorbed amount according to experiment, ELM and IAST for the system CH_4–CO–H_2 xiv
A.6. Adsorbed amount according to experiment, ELM and IAST for the system CH_4–CO_2–H_2 xv

List of Symbols

A	area in m^2
C	adsorption loading in $\mathrm{mol\,m}^{-2}$
C_m	mole-based monomolecular layer capacity in $\mathrm{mol\,m}^{-2}$
C_m^m	mass-based monomolecular layer capacity in $\mathrm{kg\,m}^{-2}$
C_{eq}	equilibrium adsorption loading in $\mathrm{mol\,m}^{-2}$
D	diffusion coefficient in $\mathrm{m}^2\,\mathrm{s}^{-1}$
F	Helmholtz free energy in J
G	Gibbs free enthalpy in J
J	flux in $\mathrm{mol\,s}^{-1}\,\mathrm{m}^{-1}$
K_e	Henry coefficient in $\mathrm{mol\,m}^{-2}\,\mathrm{Pa}^{-1}$
K_i	linear driving force coefficient of order $i > 0$ in $\mathrm{s}^{-1}\,\mathrm{mol}^{1-\mathrm{i}}\,\mathrm{m}^{2\mathrm{i}-2}$
L	mobility coefficient in $\mathrm{s\,mol\,kg}^{-1}$
M	molar mass in $\mathrm{kg\,mol}^{-1}$
R	gas constant in $\mathrm{J\,mol}^{-1}\,\mathrm{K}^{-1}$
R_a	rate of adsorption in $\mathrm{mol\,s}^{-1}$
R_d	rate of desorption in $\mathrm{mol\,s}^{-1}$
R_i	change of mass of component i in $\mathrm{kg\,s}^{-1}$
S	entropy in $\mathrm{J\,K}^{-1}$
T	temperature in K
U	internal energy in J
V	adsorbed amount in $\mathrm{Ncm}^3\,\mathrm{g}^{-1}$
Ω_D	non-dimensional diffusion collision integral
Θ	relative uptake
α	thermal diffusion coefficient in $\mathrm{m}^2\,\mathrm{s}^{-1}$

δ	relative error
ϵ_i	characteristic energy of component i in J
η	interaction coefficient in the extended Langmuir model
γ	activity coefficient
μ	viscosity in $\mathrm{Pa\,s}$
μ_i	chemical potential of component i in $\mathrm{J\,mol^{-1}}$
ϕ	surface potential of the adsorbed phase per mass of the adsorbent in $\mathrm{J\,kg^{-1}}$
π	spreading pressure in $\mathrm{N\,m^{-1}}$
π^r	reduced spreading pressure in $\mathrm{mol\,kg^{-1}}$
σ	molar area of the adsorbed phase in $\mathrm{m^2\,mol^{-1}}$
σ_i	characteristic length of component i in m
$\mathbf{D}$	diffusion kinetics coupling matrix
$\mathbf{T}$	stress tensor
ζ	bulk viscosity in $\mathrm{Pa\,s}$
b	ratio of adsorption and desorption parameter in $\mathrm{Pa^{-1}}$
f	fugacity in Pa
g	molar Gibbs free enthalpy in $\mathrm{J\,mol^{-1}}$
k	Boltzmann constant in $\mathrm{J\,K^{-1}}$
k_a	adsorption rate coefficient in $\mathrm{mol\,s^{-1}\,Pa^{-1}}$
k_d	desorption rate coefficient in $\mathrm{mol\,s^{-1}}$
l	unit length in m
n	number of moles in mol
n^0	number of adsorbed moles per mass of the adsorbent in $\mathrm{mol\,kg^{-1}}$
n_F	Freundlich coefficient
p	pressure in Pa
r_i	change of mass of component i per volume in $\mathrm{kg\,m^{-3}\,s^{-1}}$
w	mass fraction
x	mole fraction of the adsorbed phase
y	mole fraction of the gas phase
z	non-dimensional difference of the surface potential to the reference state
C	Courant number
STP	standard temperature and pressure at $273.15\,\mathrm{K}$ and $10^5\,\mathrm{Pa}$

CHAPTER 1

Introduction

With today's need to find new sources of fuels for industrial and personal use, adsorption will become more and more important, for its ability of separating and cleaning synthesis gas, or selectively removing carbon dioxide from a feed gas. This requires the prediction and design of adsorption apparatus and leads to the necessity for advanced simulation techniques. However, a complete implementation for detailed simulation like computational fluid dynamics, also known as CFD, was not freely available before.
Also, more general models of mass transfer to account for multiple phenomena from a fluid phase to a solid in CFD are not available. One step to solve this problem was already done by members of the thermal process engineering group at Vienna University of Technology. By adding single-component adsorption capability to an already available open-source solver, they established the `adsorpFoam` framework as described in [Haddadi et al., 2014] and [Haddadi et al., 2015b]. An extension to support multi-region is also available [Haddadi et al., 2015a]. In this thesis, version 1.3.2 of the solver was used. This version does not provide multi-region support. The aim of this thesis is to take the next step and account for more than one adsorbing species. The long-term goal is to greatly generalise mass transfer, only needing one application for many different kinds of phenomena.
Starting with simple one-component Henry adsorption, which was already available in the research group [Haddadi et al., 2014], this thesis will show ways, explanation and testing for multicomponent adsorption on different adsorbents. In the end, there should be a modular, general solver which provides multiple equilibrium and kinetics models which should be run-time selectable.
However, it is not feasible to account for every detail in the context of one single master thesis. The main focus is on finding already used models in literature, and including them as library into the working solver. Therefore, some basic simplifications and assumptions will be made during modelling. Furthermore, the aim is to find and implement already existing models in literature, and not to develop new ones.
Adsorption simulation models are already available in commercial process simulation tools like Aspen Adsim [Aspen, 2016]. However, these tools have to be paid for. Its models are often zero- or one-dimensional, without offering the capabilities and degree of detail CFD can provide. The advantages of CFD are as follows: Not only is the adsorption phenomenon considered, but also the flow and concentration gradients can be made visible. It accounts for pressure drop based on the Navier-Stokes equation and provides a detailed, three-dimensional geometry. Also, detailed temperature distribution can be shown and channelling effects can be recognised and avoided.

This thesis will be divided into nine chapters. In chapter 2, the general problem of adsorption will be described, starting with equilibrium and kinetics for a single adsorbing component. Then, the currently available models for more than one adsorbing component will be presented. Next, a diffusion-based kinetics model will be presented, which is adapted from literature.
In chapter 3, the implementation of the models in Octave, a Matlab-like high-level programming language for numerical computations, will be shown. Some peculiarities will be outlined which will become more important at a later stage.
In chapter 4, the validation and results of the Octave models are presented. Comparison of six multicomponent systems with experimental data from literature are given. The results of the diffusion-based kinetics model will be given.
In chapter 5, an overview of CFD will be given and the fundamental mathematical concepts will be introduced. Afterwards, the finite-volume method and the conservation equations will be outlined in more detail. Additionally, the problem of pressure-velocity coupling and prediction of material properties – a crucial, but often neglected issue – will be addressed.
In chapter 6, the main tool used in this thesis, OpenFOAM, will be introduced. The fundamental aspects of this program will be given. The main focus lies on the programming of new features which will become important later on. Then, the already available solvers suitable for implementing adsorption models are presented.
In chapter 7, the implementation of the adsorption models into the custom OpenFOAM solver `adsorpFoam` is described. The different approaches for each model are reasoned. Next, the necessary adaptations to the conservation equations are explained and some pitfalls which have to be considered when running and extending this solver are given.
In chapter 8, the final results of the implementation in OpenFOAM are validated using simple geometries and small grid size. Then, some more complex case setups are presented.
In chapter 9, the concepts, perceptions and results are summed up and an outlook for possible additional work and refinements in the future is given.

CHAPTER 2

Thermodynamics of Adsorption

Adsorption is the attachment of fluid molecules on the surface of a solid. It is an exothermic process that takes place at the surface of a solid as illustrated in figure 2.1. The molecules of the fluid adsorb on the surface of a solid, called adsorbent, to form a layer or adsorbed phase, also called adsorbate. The bond between adsorbate and adsorbent can be either physical or chemical, called physisorption and chemisorption, respectively. In this chapter, only physisorption of gases will be considered, although some concepts may be applicable to chemisorption and liquid-phase adsorption as well.

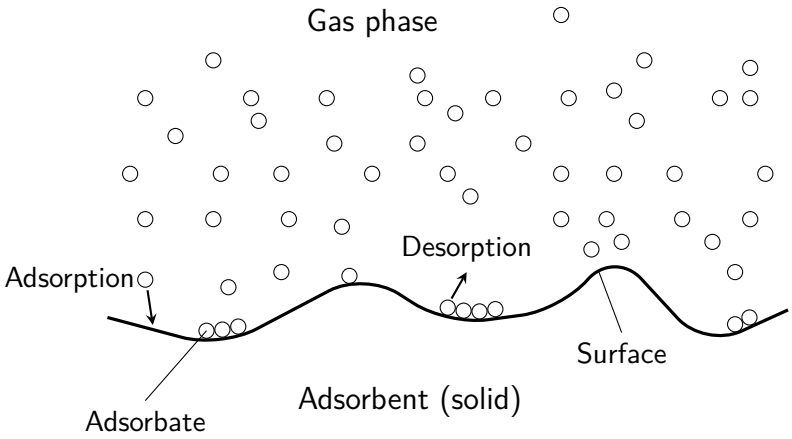


Figure 2.1.: Nomenclature of adsorption.

Since adsorption is a surface-driven phenomenon, open pores, i.e. pores visible to the surface, can significantly increase the capacity of an adsorbent. Depending on the size of its diameter, there are three different kind of pores [Do, 1998]:

- Micropores with a diameter less than 2 nm,
- mesopores with a diameter between 2 nm to 50 nm,
- and macropores with a diameter greater than 50 nm.

Almost all physisorption isotherms can be grouped in one of six basic types classified by the International Union of Pure and Applied Chemistry IUPAC as shown in figure 2.2. At sufficiently low pressures, all types show linear behaviour which is called Henry's Law region. The types of physisorption are as follows [Sing, 1985]:

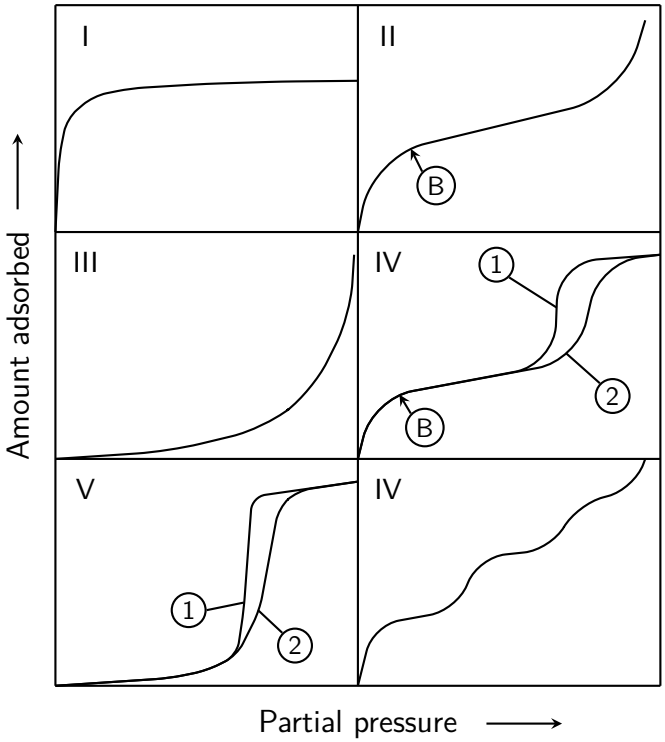


Figure 2.2.: The six types of physisorption after IUPAC, 1: adsorption, 2: desorption, B: beginning of multimolecular layers adsorption (adapted from [Sing, 1985]).

- Type I: reversible with a concave to the partial pressure curve and adsorption limit. This type is normally referred to as Langmuir isotherm.
- Type II: reversible, multilayer adsorption on a non-porous or macroporous adsorbent. Point B indicates the beginning of multimolecular layers adsorption.
- Type III: the curve of the isotherm is convex to the partial pressure curve. Those kind of isotherms are rather uncommon.
- Type IV: adsorption with hysteresis, for capillary condensation taking place in mesopores. Again, point B indicates the beginning of multimolecular layers adsorption and there is an adsorption limit.
- Type V: similar to type III, with an adsorption limit and hysteresis. Like type III, this type is not very common.
- Type VI: stepwise multimolecular layers adsorption takes place at this type, on a uniform non-porous surface. The height of each steps represents the monomolecular layer capacity.

The used adsorbents vary, depending on the adsorbing species, pressure and temperature. Alumina can be used in industrial areas for drying gas streams from moisture. It has a specific surface of $200\,\mathrm{m^2\,g^{-1}}$ to $300\,\mathrm{m^2\,g^{-1}}$ [Do, 1998].
Another widely used adsorbent is silica gel. It is a coagulation of very small silicic acid particles. As alumina, it is used to remove moisture from air flows, for its high affinity to

water. The typical specific surface exceeds the one from alumina, with maximum values as high as $900\,\mathrm{m^2\,g^{-1}}$ [Do, 1998].
Activated carbon is probably the most used adsorbent. It is rather cheap, and has a very high specific surface of up to $1200\,\mathrm{m^2\,g^{-1}}$. Furthermore, its pore size distribution favours adsorption and it has many functional groups containing oxygen at the surface. Those groups are a result of the production process which typically involve oxygen-rich raw materials [Do, 1998].
Another common adsorbent are zeolithes. Although there are natural types of zeolithes, most used adsorbents are made synthetically. It is possible to make many types of zeolithes, to satisfy different requirements [Ruthven, 1984].
For separation of gases, two different dynamic adsorption procedures are known: Pressure-Swing Adsorption PSA and Thermal Swing Adsorption TSA. For continuous operation, both require at least two adsorption beds and are working by a shift of equilibrium. PSA is a technique where the pressure is decreased, and therefore, desorption will take place if the bed is in equilibrium before. So, it is possible to selectively remove one or more components which have a higher affinity to the adsorbent than the rest.
With TSA, temperature is increased to lower the equilibrium loading. PSA is suitable for rapid changes of adsorption and desorption cycles, and does not require heat input. Sometimes, a combination of both is used to maximise performance.
A packed bed adsorber is most commonly used in industry. The adsorbent is loosely packed inside a column. For being able to run continuously, more than one adsorption column is necessary.

2.1. Adsorption Equilibrium

Equilibrium in adsorption always implies a dynamic equilibrium, meaning that rate of adsorption and rate of desorption are equal. Therefore, from a macroscopic point of view, there occurs no visible change at equilibrium. Normally, isotherm conditions are assumed when describing adsorption equilibria.
If not stated otherwise, the assumptions for the subsequent considerations are as follows:

- Localised adsorption: each adsorbed molecule takes up the same area and cannot move on the surface. The surface can be divided into adsorption sites.
- Flat, homogeneous surface: everywhere on the surface, the affinity for the adsorbing molecules is the same. No capillary condensation takes place.
- Monomolecular layer: one adsorption site can take up only one molecule simultaneously.

In order to describe adsorption equilibria, the equation for the change of the Helmholtz free energy F is modified. It reads as follows [Stephan and Mayinger, 1999]:

$$\mathrm{d}F = -S\mathrm{d}T - p\mathrm{d}V + \sum_{i=1}^{N} \mu_i \mathrm{d}n_i, \tag{2.1}$$

with temperature T, pressure p, volume V, entropy S, chemical potential μ_i and number of moles n_i of component i. The total number of components is N.

Now, the so-called spreading pressure π is introduced. It can be interpreted as difference between the surface tension of a clean surface and a surface covered with adsorbate and is defined as follows [Ruthven, 1984]:

$$\pi = -\left(\frac{\partial U}{\partial A}\right)_{S,V,n}. \tag{2.2}$$

with the internal energy U and surface area A. Comparing the above definition of the spreading pressure with the definition of the pressure, the similarity becomes more clear:

$$p = -\left(\frac{\partial U}{\partial V}\right)_{S,V,n}. \tag{2.3}$$

Now, the pressure is replaced by the spreading pressure [Do, 1998]:

$$\mathrm{d}F = -S\mathrm{d}T - \pi\mathrm{d}A + \sum_{i=1}^{N} \mu_i \mathrm{d}n_i. \tag{2.4}$$

Integration at constant π, T and μ gives:

$$F = -\pi A + \sum_{i=1}^{N} \mu_i n_i. \tag{2.5}$$

Now, the above equation is differentiated:

$$\mathrm{d}F = -\pi\mathrm{d}A - A\mathrm{d}\pi + \sum_{i=1}^{N} \mu_i \mathrm{d}n_i + \sum_{i=1}^{N} n_i \mathrm{d}\mu_i. \tag{2.6}$$

Subtracting equation (2.4) from (2.6) gives the Gibbs equation for a plane surface:

$$-A\mathrm{d}\pi + S\mathrm{d}T + \sum_{i=1}^{N} n_i \mathrm{d}\mu_i = 0. \tag{2.7}$$

For equilibrium, isotherm conditions are assumed, $\mathrm{d}T = 0$:

$$-A\mathrm{d}\pi + \sum_{i=1}^{N} n_i \mathrm{d}\mu_i = 0. \tag{2.8}$$

The above equation connects the chemical potential with the spreading pressure and is used in the subsequent section to derive an equation of state for the adsorbed phase.

2.1.1. Single-component Adsorption

For one component, equation (2.8) reduces to:

$$-A\mathrm{d}\pi + n\mathrm{d}\mu = 0. \tag{2.9}$$

Assuming ideal gas behaviuor, with the chemical potential $\mu_g = \mu_g^0 + RT \ln p$, the gas constant R and the chemical potential at reference state μ_g^0, the isotherm Gibbs equation for one component reads as:

$$\left(\frac{\mathrm{d}\pi}{\mathrm{d}\ln p}\right)_T = CRT, \tag{2.10}$$

with the number of adsorbed moles per area $C = nA^{-1}$.

Henry Type Adsorption

For an ideal surface at infinite dilution, equation (2.10) becomes

$$\pi\sigma = RT, \tag{2.11}$$

with the molar area $\sigma = C^{-1}$. This is comparable to the ideal gas equation. There is a linear dependence between pressure and number of adsorbed molecules:

$$C(p) = K_e p, \tag{2.12}$$

with the Henry coefficient K_e. This type of isotherm is only valid at low pressures. Figure 2.3(a) shows a Henry adsorption isotherm with $K_e = 5 \times 10^{-3}\,\mathrm{mol\,g^{-1}\,bar^{-1}}$.

Freundlich Type Adsorption

The Freundlich isotherm is an empirical extension to the Henry adsorption with an additional parameter, the so-called Freundlich parameter $0 < n_F < 1$:

$$C(p) = K_e p^{n_F}. \tag{2.13}$$

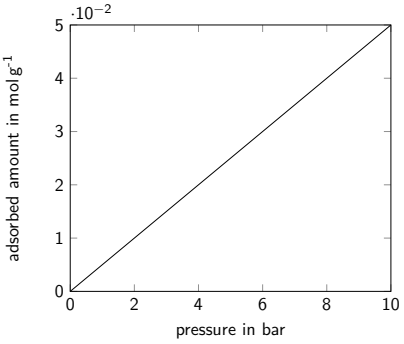
Figure 2.3(b) shows a Freundlich adsorption isotherm with $K_e = 5 \times 10^{-3}\,\mathrm{mol\,g^{-1}\,bar^{-1}}$ and $n_F = 0.85$.

Langmuir Type Adsorption

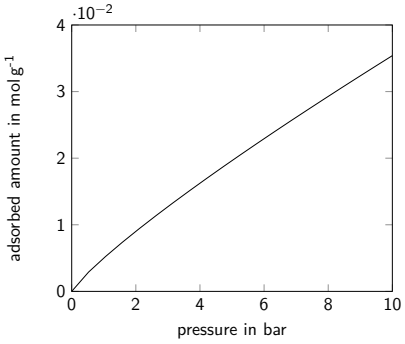
This type of adsorption isotherm was first described by [Langmuir, 1918]. Assuming the following equation of state [Do, 1998]:

$$\pi\sigma = RT\frac{\sigma}{\sigma_0}\ln\frac{\sigma}{\sigma - \sigma_0}, \tag{2.14}$$

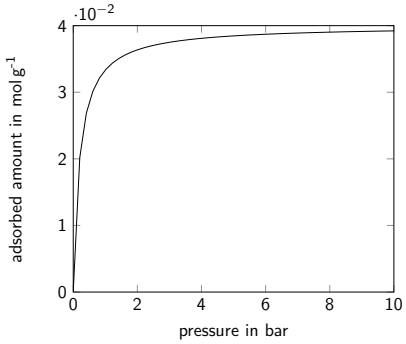
the isotherm reads as follows:



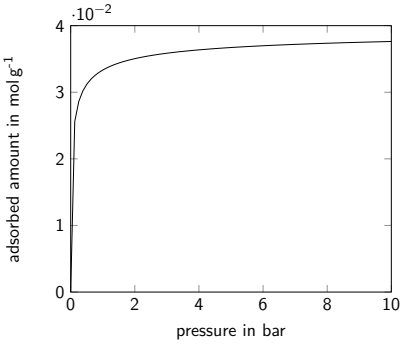
(a) Henry adsorption isotherm.



(b) Freundlich adsorption isotherm.



(c) Langmuir adsorption isotherm.



(d) Freundlich-Langmuir adsorption isotherm.

Figure 2.3.: Four different single-component adsorption isotherms.

$$C(p) = C_m \frac{bp}{1+bp}, \tag{2.15}$$

with the temperature-dependent parameter b and the monomolecular layer loading C_m. This adsorption type is limited in terms of adsorbed amount at high pressure, where the adsorbed amount C approaches C_m. The two parameters can be calculated using the following relations:

$$b(T) = b_0 \exp \frac{T_0}{T}, \tag{2.16}$$

$$C_m(T) = C_m^0 + C_m^1 T, \tag{2.17}$$

with suitable parameters b_0, T_0, C_m^0 and C_m^1. C_m^1 is almost always negative. Therefore, a higher temperature will decrease b and C_m, and the equilibrium loading.
Figure 2.3(c) shows a Langmuir adsorption isotherm with $C_m = 4 \times 10^{-2}\,\mathrm{mol\,g^{-1}}$ and $b = 5\,\mathrm{bar}^{-1}$.

Freundlich-Langmuir Type Adsorption

As the Freundlich isotherm, this is an empirical extension to the Langmuir isotherm, with $0 < n_F < 1$. Thermodynamic consistency is lost, since Langmuir adsorption assumes a fixed number of adsorption sites. This is not the case anymore if the Freundlich parameter is introduced:

$$C(p) = C_m \frac{bp^{n_F}}{1+bp^{n_F}}, \tag{2.18}$$

Figure 2.3(d) shows a Freundlich-Langmuir adsorption isotherm with $C_m = 4 \times 10^{-2}\,\mathrm{mol\,g^{-1}}$, $b = 5\,\mathrm{bar}^{-1}$ and $n_F = 0.85$.

2.1.2. Multicomponent Adsorption

Although actual single-component adsorption does not often occur as most gases consist of more than one species, the extension to multicomponent adsorption is not very well covered in literature. Often, the diluting gases do not adsorb very well. The presented models in this section were developed many decades ago. The aim of the presented models is to predict multicomponent adsorption without the need of experimental data. However, introducing empirical so-called interaction coefficients from experiments will improves the results, as shown in later chapters.

Extended Langmuir Model (ELM)

The extended Langmuir model ELM is derived from the same assumptions as the single-component Langmuir isotherm. Here, it is obtained by assuming a dynamic equilibrium, and not an equation of state. The rate of adsorption $R_{a,i}$ of component i is proportional to the fraction of vacant sites and the partial pressure $p_i = y_i p$ with the mole fraction of the gas phase y_i. It can be written as [Do, 1998]:

$$R_{a,i} = k_{a,i} p_i \left(1 - \sum_{j=1}^{N} \frac{C_j}{C_{m,j}}\right), \tag{2.19}$$

with the adsorption rate coefficient $k_{a,i}$. The index j is introduced to differentiate between the rate of adsorption for one component, denoted with the index i, and the sum of all species, denoted with j. The rate of desorption $R_{d,i}$ of component i is proportional to the fraction of occupied sites and is unaffected by the partial pressure:

$$R_{d,i} = k_{d,i} \frac{C_i}{C_{m,i}}, \tag{2.20}$$

with the desorption rate coefficient $k_{d,i}$. At equilibrium, adsorption and desorption rate have to be equal. Therefore, the following must hold:

$$b_i p_i \left(1 - \sum_{j=1}^{N} \frac{C_j}{C_{m,j}}\right) = \frac{C_i}{C_{m,i}}, \tag{2.21}$$

with $b_i = k_{a,i}/k_{d,i}$ as the ratio of adsorption and desorption parameter. Summing over all species, it reads as:

$$\sum_{i=j}^{N} \frac{C_j}{C_{m,j}} = \frac{\sum_{j=1}^{N} b_j p_j}{1 + \sum_{j=1}^{N} b_j p_j}, \tag{2.22}$$

or, considering only one component:

$$C_i(p) = C_{m,i} \frac{b_i p_i}{1 + \sum_{j=1}^{N} b_j p_j}. \tag{2.23}$$

The extended Langmuir model is widely used, but lacks thermodynamic consistence unless the monomolecular layer capacities C_m of all species are the same. The reason for this condition is that Langmuir adsorption assumes a fixed number of adsorption sites, where molecules can attach. In practice, this condition is rarely met and models which preserve such consistency are described in literature [Bai and Yang, 2001].

Extended Langmuir Model with Interaction Coefficients (ELMIAC)

An extension to the ELM is the introduction of empirical interaction coefficients IAC. Those IAC can be calculated from experimental data and were first proposed by [Schay, 1956]. The isotherm reads as follows:

$$C_i(p) = C_{m,i} \frac{(b_i/\eta_i) p_i}{1 + \sum_{j=1}^{N} (b_j/\eta_j) p_j}. \tag{2.24}$$

with the interaction coefficient η_i of component i. The IAC may be calculated using experimental data [Ritter and Yang, 1987]:

$$\ln \eta_i = \ln b_i p_i - \ln \frac{C_{mes,i}/C_{m,i}}{1 - \sum_{j=1}^{N} C_{mes,j}/C_{m,j}}. \tag{2.25}$$

Ideal Adsorbed Solution Theory (IAST)

The ideal adsorbed solution theory IAST can predict multicomponent adsorption equilibria using only the single-component isotherms of all species. This isotherm can be of any type. It is also possible to use experimental data without modelling the isotherm. The Gibbs free enthalpy per mole of the adsorbate is [Do, 1998]:

$$g = \sum_{i=1}^{N} g_i^0 + g_m, \tag{2.26}$$

with the Gibbs free enthalpy at reference state g_i^0 and the molar free enthalpy of mixing defined as follows:

$$g_m = RT \sum_{i=1}^{N} x_i \ln \gamma_i x_i, \tag{2.27}$$

with the activity coefficient γ_i of component i and the mole fraction x_i of component i in the adsorbed phase. Combining equations (2.26) and (2.27) and using the definition of the Gibbs free enthalpy $G = U - TS$, the following equation is obtained:

$$\frac{U - TS}{\sum_{i=1}^{N} n_i} = \sum_{i=1}^{N} x_i g_i^0 + RT \sum_{i=1}^{N} x_i \ln \gamma_i x_i. \tag{2.28}$$

The starting thermodynamics equations for the adsorbed phase and the gas phase, respectively, read as follows [Do, 1998]:

$$U - TS - \phi m - \sum_{i=1}^{N} \mu_i n_i = 0, \tag{2.29}$$

$$U - TS + pV - \sum_{i=1}^{N} \mu_i n_i = 0. \tag{2.30}$$

m denotes the mass of the adsorbent.
Combining equations (2.28) and (2.29) gives:

$$\frac{\phi m}{\sum_{i=1}^{N} n_i} + \sum_{i=1}^{N} x_i \mu_i = \sum_{i=1}^{N} x_i g_i^0 + RT \sum_{i=1}^{N} x_i \ln \gamma_i x_i = 0. \tag{2.31}$$

with the surface potential of the adsorbate ϕ per unit mass. The spreading pressure π and the specific surface potential ϕ are linked with the area, $\pi A = \phi$. Using the definition of the chemical potential for an ideal gas with fugacity f, equation (2.31) can be rewritten as:

$$\frac{\phi}{n^0} + RT \sum_{i=1}^{N} x_i \ln \frac{f_i}{\gamma_i x_i} + \sum_{i=1}^{N} x_i \left(\mu_i^0 - g_i^0 \right) = 0. \tag{2.32}$$

n^0 is the sum of all adsorbed moles divided by the mass of the adsorbent m.
The molar Gibbs free enthalpy expressed in terms of pressure of the pure component reads as follows [Do, 1998]:

$$g_i^0 = \frac{\phi_i^0}{n_i^0} + \mu_i^0. \tag{2.33}$$

Combining equations (2.32) and (2.33), the fundamental equation for the mixture is obtained:

$$RT \sum_{i=1}^{N} x_i \ln \frac{f_i}{f_i^0 \gamma_i x_i \exp z_i} + \phi \left(\frac{1}{n^0} - \sum_{i=1}^{N} \frac{x_i}{n_i^0} \right) = 0, \tag{2.34}$$

with the fugacity f_i^0 of component i at reference state, and z_i as non-dimensional difference of surface potential to reference state:

$$z_i = -\frac{\phi - \phi_i^0}{n_i^0 RT}.$$

For an ideal adsorbed solution, the following must hold ($\gamma_i = 1$):

$$f_i = f_i^0 x_i \exp z_i. \tag{2.35}$$

This is equal to setting all terms to zero in equation (2.34) and yields:

$$\frac{1}{n^0} = \sum_{i=1}^{N} \frac{x_i}{n_i^0}. \tag{2.36}$$

With $f_i = p y_i$ and a hypothetical pressure p_i^0, the second equation for an ideal adsorbed solution is obtained:

$$p y_i = p_i^0 x_i \exp z_i. \tag{2.37}$$

Using equation (2.10), the reduced spreading pressure is obtained. The reduced pressure is equal to the surface potential divided by temperature and molar gas constant. It reads for the pure component i as:

$$\pi_0^r = \frac{\pi_0 A}{RT} = \frac{\phi_0}{RT} = -\int_0^{p_i^0} \frac{n_i^0(\tilde{p}_i^0)}{\tilde{p}_i^0} \, \mathrm{d}\tilde{p}_i^0, \tag{2.38}$$

where $n_i^0(p_i^0)$ is the adsorbed amount for the single species i per mass adsorbent. The adsorbed amount can be calculated from any type of isotherm, e.g. Langmuir isotherm, or even from experimental data.
[Myers and Prausnitz, 1965] suggested that the surface potential of the mixture is the same as the surface potential of all pure components. Therefore, z_i is zero and the following set of equations is obtained:

$$\frac{\phi}{RT} = \frac{\phi_i^0}{RT} = -\int_0^{p_i^0} \frac{n_i(\tilde{p}_i^0)}{\tilde{p}_i^0}\,\mathrm{d}\tilde{p}_i^0, \tag{2.39}$$

$$p y_i = p_i^0 x_i, \tag{2.40}$$

$$\sum_{i=1}^{N} \frac{x_i}{n_i(p_i^0)} = \frac{1}{n}, \tag{2.41}$$

$$\sum_{i=1}^{N} x_i = 1. \tag{2.42}$$

Equation (2.40) can be interpreted as Raoult's law for gas-adsorbed phase equilibrium. Furthermore, the sum of all mole fractions in the gas phase and the adsorbed phase must be equal to one.
The set of equations (2.39) to (2.42) provides $2N + 1$ relations:

- $N - 1$ relations with equation (2.39),
- N relations with equation (2.40),
- one relation with equation (2.41),
- and one relation with equation (2.42).

Normally, the gas mole fractions, total pressure and temperature are given. This yields $2N + 1$ unknowns and therefore, zero degrees of freedom. The unknowns are:

- N mole fractions in the adsorbed phase x_i,
- N hypothetical pressures p_i^0,
- and the total number of adsorbed moles n.

However, the inverse problem can be posed as well: In that case, the adsorbed mole fractions and the total adsorbed amount are given. The corresponding total pressure and gas mole fractions have to be calculated:

- N mole fractions in the gas phase y_i,
- N hypothetical pressures p_i^0,
- and the total pressure p.

2.2. Adsorption Kinetics

In chemistry, the knowledge of the equilibrium of e.g. a reaction does not explain the problem completely. Often, chemical reactions are hindered by kinetic restrictions. Therefore, kinetics have to be considered as well.

2.2.1. Linear Driving Force

The adsorption rate is assumed to be only dependent from the difference of the equilibrium and the currently adsorbed amount. This yields the linear driving force model [Sircar and Hufton, 2000]:

$$\frac{\partial C_i}{\partial t} = \sum_{j=1}^{\infty} \left[\operatorname{sgn}\left(C_{i,eq} - C_i\right)\right]^{j+1} K_j \left(C_{i,eq} - C_i\right)^j , \tag{2.43}$$

with suitable coefficients K_j. $C_{eq,i}$ denotes the equilibrium loading of component i. The first term is introduced to prevent the loss of the algebraic sign due to even powers, which is important if desorption occurs.

2.2.2. Diffusion-based Kinetics

The driving force of adsorption is the gradient of the chemical potential. Assuming an ideal gas, the flux J_i of component i is given by:

$$\boldsymbol{J}_i = -L_i C_i \nabla \mu_i, \tag{2.44}$$

where L_i is the mobility coefficient of component i. Multiplying equation (2.44) with the surface normal vector leads to:

$$J_i = -L_i C_i \frac{\partial \mu_i}{\partial x}. \tag{2.45}$$

Inserting the chemical potential for an ideal gas, it reads as:

$$J_i = -L_i R T C_i \frac{\partial \ln p_i}{\partial x}. \tag{2.46}$$

$-L_i RT$ can be replaced by the diffusivity of component i in the mixture, $D_{m,i}$. Taking a closer look at the gradient of partial pressure and applying the chain rule, it can be rewritten as:

$$\frac{\partial \ln p_i}{\partial x} = \frac{1}{p_i}\frac{\partial p_i}{\partial x} = \frac{1}{p_i}\sum_{j=1}^{N} \frac{\partial p_i}{\partial C_j}\frac{\partial C_j}{\partial x}. \tag{2.47}$$

Putting this system of equations in matrix-vector form, the following is obtained:

$$\boldsymbol{J} = -\mathbf{D} \cdot \frac{\partial \boldsymbol{C}}{\partial x}, \tag{2.48}$$

with:

$$\mathbf{D} = \left\{ D_{ij} = D_{m,i}\frac{C_i}{p_i}\frac{\partial p_i}{\partial C_j} \right\}.$$

The original model presented in [Do, 1998] is for intraparticle diffusion of the adsorbate. However, the gradient can be modelled with the difference of the equilibrium adsorption concentration for the pressure in the gas phase and the current adsorbed concentration divided by length. So, the change of adsorbed moles per unit length l is given by:

$$\frac{\partial \boldsymbol{C}}{\partial t} = -\frac{\boldsymbol{J}}{l} = \frac{\mathbf{D}}{l} \cdot \frac{\boldsymbol{C}_{eq} - \boldsymbol{C}}{\Delta x}. \tag{2.49}$$

CHAPTER 3

Implementation in Octave

Octave is a freely available, Matlab-like, interpreted language for numerical computations. One of its advantages is that the calculations are vector and matrix-based, and therefore, doing computations for more than one species simultaneously is easy to write. The models described in section 2 are implemented using a zero-dimensional model. This means that only adsorption equilibria are calculated and infinite supply of gas phase is assumed, without considering mass balance, flow properties or control volumes. The main purpose of this implementation is to display the results of the various adsorption models, organise and test the equations required for the implementation of the adsorption models and to serve as a validation for the OpenFOAM implementation.
The code presented in this chapter was written by the author, can be found in appendix A and is freely available under the MIT License [OSI, 2016].

3.1. Calculation of Adsorption Coefficients

The Langmuir parameters are calculated using equations (2.16) and (2.17). In this thesis, the four necessary adjustment parameters are taken from [Ritter and Yang, 1987].

3.2. Extended Langmuir Model

The extended Langmuir model is implemented using equation (2.23). It is shown in listing 3.1.

Listing 3.1: Implementation of the ELM in Octave.

```
1 function C = ELM (p, y, Cm, b)
2   C = Cm .* b .* y * p / (1 + sum (b .* y * p));
3 end
```

3.3. Extended Langmuir Model with Interaction Coefficients

Here, the calculations of IAC based on experimental data from [Ritter, 1985] is done as described by equation (2.25).

Listing 3.2: Calculation of the IAC in Octave.

```
1 function eta = IAC (p, y, Cm, b, C_mes)
2   eta = (1 - sum (C_mes ./ Cm)) * p * y .* b .* Cm ./ C_mes;
3 end
```

3.4. Ideal Adsorbed Solution Theory

First, an estimation of the reduced spreading pressure π^r is calculated. Assuming that the single-component isotherm is of Langmuir type, the following is valid [Do, 1998]:

$$\pi^r_{est} = \frac{\pi_{est} A}{RT} = \frac{\phi_{est}}{RT} = \frac{\sum_{i=1}^{N} C_{m,i}}{N} \ln\left(1 + \sum_{i=1}^{N} b_i p_i\right). \tag{3.1}$$

If Langmuir single-component isotherm is assumed, the virtual pressure p_i^0 can be calculated analytically without iterating:

$$p_i^0 = \frac{1}{b_i}\left(\exp\frac{\pi^r}{C_{m,i}} - 1\right). \tag{3.2}$$

If other isotherms are used, equation (2.39) has to be solved numerically. Next, the mole fractions of all species can be calculated using equation (2.37):

$$x_i = \frac{p_i}{p_i^0}. \tag{3.3}$$

The convergence criterion is that the sum of mole fractions of the adsorbed species must be unity or very close to unity:

$$\left|1 - \sum_{i=1}^{N} x_i\right| \leq \text{ conv. limit}, \tag{3.4}$$

where the convergence limit is set by the user. In this thesis, it is set to an order of magnitude of around 10^{-6} to 10^{-8}. If this limit is met, the computation may continue. If it is not met, a new spreading pressure of the $(n+1)^{\text{th}}$ iteration has to be calculated using the value of the $(n)^{\text{th}}$ iteration:

$$\pi^r_{n+1} = \pi^r_n \sum_{i=1}^{N} x_i. \tag{3.5}$$

For stability reasons, there are some limiters and a relaxation factor applied in the actual implementation, as seen in lines 7 to 15 of listing 3.3. Then, the adsorbed amount is calculated as follows:

$$n_i = \frac{x_i}{\sum_{j=1}^{N} x_j / n_j(p_j^0)}. \tag{3.6}$$

The flowchart of the algorithm is shown in figure 3.1.

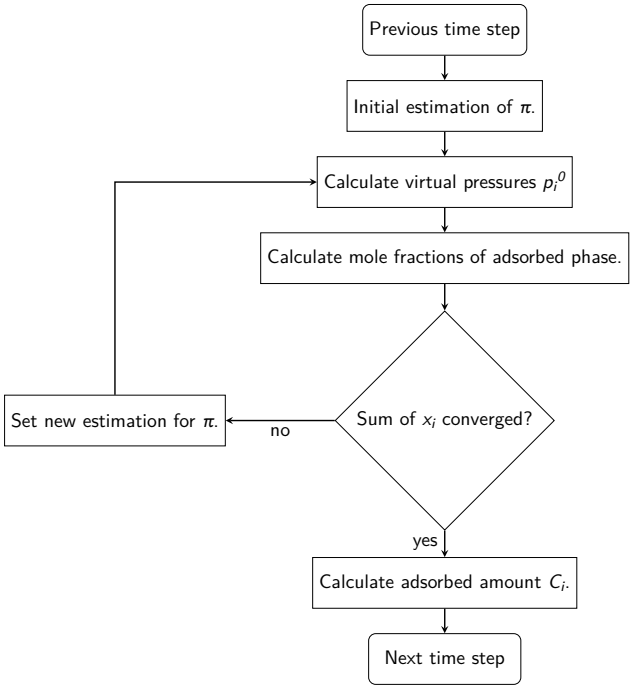


Figure 3.1.: Flowchart of algorithm for solving IAST.

Listing 3.3: Implementation of the IAST in Octave.

```
function C = IAST (p, y, params)
  counter = 0;
  correct = 1;
  z_est = 1;

  do
    if (correct < 10 && correct > 0.1)
      z_est *= 0.75 * (correct - 1) + 1;
    else
      if (correct > 1)
        z_est *= 8;
      else
        z_est *= 0.125;
      end
    end

    switch (params.isotherm)
    case "langmuir"
      p0 = 1 ./ params.b .* (exp (z ./ params.Cm) .- 1);
    otherwise
      error ("no recognised single-component isotherm specified");
    end

    x = p * y ./ p0;
    correct = sum (x);
    ++counter;

    if (counter > params.maxIter)
      break;
    end
  until (abs (sum (x) - 1) < params.convergence)

  n0 =  params.Cm .* params.b .* p0 ./ (1 .+ params.b .* p0);

  C = x / sum (x ./ n0);
end
```

3.5. Diffusion Kinetics

First, the diffusion coefficients for all species for the mixture are calculated as described in section 5.5.1. Next, the coupling matrix defined in equation 2.49 is evaluated, assuming Langmuir adsorption. It implies:

$$\{D_{ij}\} = \begin{cases} D_{m,i} \frac{C_i/C_{m,j}}{1-\sum_{k=1}^{N} C_k/C_{m,k}} & \text{if } i \neq j; \\ D_{m,i} \left(1 + \frac{C_i/C_{m,i}}{1-\sum_{k=1}^{N} C_k/C_{m,k}}\right) & \text{if } i = j. \end{cases} \tag{3.7}$$

With this coupling matrix, the change of loading can be calculated:

$$\frac{\partial \boldsymbol{C}}{\partial t} = \frac{\mathbf{D}}{l} \cdot \frac{\boldsymbol{C} - \boldsymbol{C}_{eq}}{\Delta x}, \tag{3.8}$$

where l denotes the unit length. The denominator has to be determined by comparing experimental data with model predictions. In this thesis, it was chosen arbitrarily. Using this equation, the loading for the new time step is obtained:

$$\boldsymbol{C}(t + \Delta t) = \boldsymbol{C}(t) + \frac{\partial \boldsymbol{C}(t)}{\partial t} \Delta t. \tag{3.9}$$

If $\mathbf{D}$ is a diagonal matrix, the diffusion-based kinetics model simplifies to a linear driving force model.

Listing 3.4: Implementation of the diffusion-bases kinetics in Octave.

```
function rate = diffusion_kinetics (Dm, C, Ceq, Cm)
  for i = 1:length (C)
    for j = 1:length (C)
      D(i, j) = Dm(i) * ((i == j) + C(i) / (Cm(j) * (1 - sum (C ./ Cm))));
    end
  end

  delta_C = Ceq .- C;
  rate = (D * delta_C')';
end
```

CHAPTER 4

Validation and Results in Octave

The results obtained with Octave serve as validation for the OpenFOAM implementation. The source of experimental data was the master thesis of J.A. Ritter [Ritter, 1985] and a paper in which the results were summed up [Ritter and Yang, 1987]. The units used in that paper are volume at standard temperature and pressure per gram adsorbent and pound-force per square inch. For clarity, the pressure is transformed to Pascal, the unit of adsorbed amount is unchanged in this chapter.

4.1. Validation

Validation was done by calculating the average interaction coefficients and comparing them to the values given in the paper [Ritter and Yang, 1987]. The IAC can be calculated using the following:

$$\eta_i = b_i p_i \frac{C_{mes,i}}{C_{m,i}} \left(1 - \sum_{j=1}^{N} \frac{C_{mes,j}}{C_{m,j}} \right). \tag{4.1}$$

Since the obtained interaction coefficients are the same for the implementation in Octave and in the paper, the models are assumed to be implemented correctly. Furthermore, certain characteristics, like the very low equilibrium adsorption loading of hydrogen are displayed correctly by the implementation. A total of six multicomponent systems were checked with data from literature.

4.2. Equilibrium Models

The pressure is in a range of 7 bar to 28 bar and the temperature is in a range of 290 K to 298 K, respectively.
Systems with H_2S are not considered, for its known non-ideal behaviour. The relative error is defined as follows:

$$\delta = 100 \, \frac{V_{calc} - V_{meas}}{V_{meas}}, \tag{4.2}$$

where V_{calc} denotes the adsorbed volume by the model, and V_{meas} represents the measured adsorbed volume.

The standard deviation of the quantity ϕ with a mean value of $\bar{\phi}$ is defined as follows:

$$\sigma_\phi = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\phi_i - \bar{\phi}_i\right)}. \tag{4.3}$$

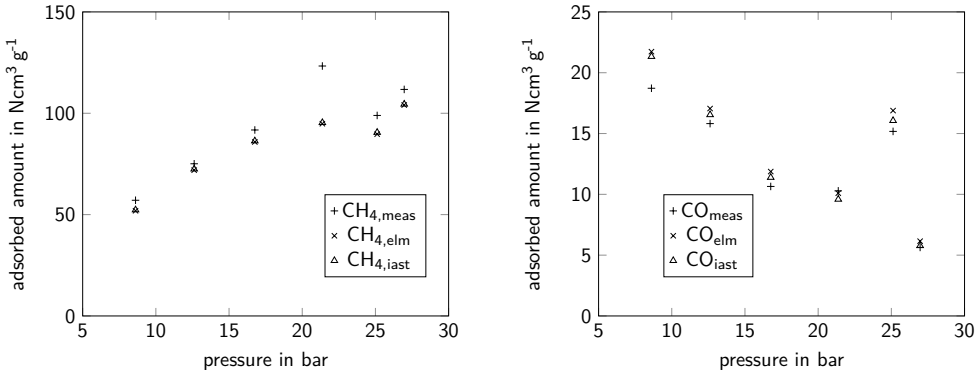
The interaction coefficients improve the extended Langmuir model substantially; however, experimental data is necessary for calculation those parameters. All errors are calculated using the absolute deviation. In the figures in sections 4.2.1 and 4.2.2, the index 'meas' denotes measured values reported by [Ritter and Yang, 1987], whereas the indices 'elm', 'iac' and 'iast' represent the values calculated by the models ELM, ELMIAC and IAST implemented in Octave, respectively.

4.2.1. Binary Systems

Binary systems containing CO_2 are not very well predicted. Generally, IAST predicts mole fractions and adsorbed amount at least slightly better. Since the sum of all mole fractions must add up to unity, the mean and maximum errors and standard deviations of the mole fractions of the adsorbed phase are equal for both components.

CH_4–CO

The system CH_4–CO is predicted rather well by both ELM and IAST, respectively. There is one outlier at around 22 bar, where the reported measured adsorbed loading is probably not correct, since the values at higher pressure are lower, which is physically not correct. The range of pressure is 9 bar to 27 bar, with six data sets available. Figure 4.1 shows the adsorbed amount of ELM and IAST, compared with the measurement. Figures 4.2 and 4.3 show the absolute error of the mole fraction and adsorbed amount, respectively.



(a) CH_4. (b) CO.

Figure 4.1.: Comparison of measurement and model of adsorbed amount for the system CH_4–CO.

The ELM predicts the mole fractions of the adsorbed phase with a mean absolute error of 2.3 $\%_{mol}$, and a maximum of 4.8 $\%_{mol}$. The standard deviation is 1.4 $\%_{mol}$. The equilibrium loading is predicted with an absolute mean absolute error of 10.0 $Ncm^3\,g^{-1}$

for CH_4 and 1.3 $Ncm^3\,g^{-1}$ for CO, respectively. The standard deviation is 9.3 $Ncm^3\,g^{-1}$ and 1.0 $Ncm^3\,g^{-1}$, respectively, with an maximum error of 28 $Ncm^3\,g^{-1}$for CH_4 and of 3.0 $Ncm^3\,g^{-1}$ for CO, respectively. Not considering the outlier, the mean absolute error of predicted adsorbed amount is 6.2 $Ncm^3\,g^{-1}$ for CH_4 and 1.5 $Ncm^3\,g^{-1}$ for CO, respectively. The maximum error is 9.2 $Ncm^3\,g^{-1}$ and 3.0 $Ncm^3\,g^{-1}$, and the standard deviation is 2.3 $Ncm^3\,g^{-1}$ and 0.9 $Ncm^3\,g^{-1}$, respectively.

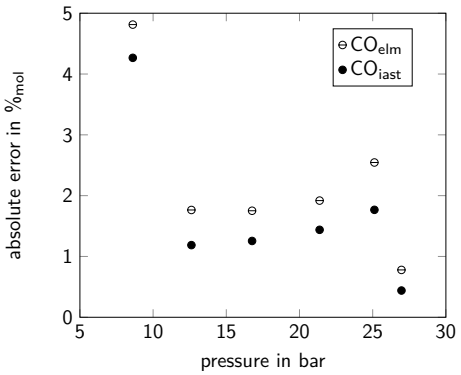
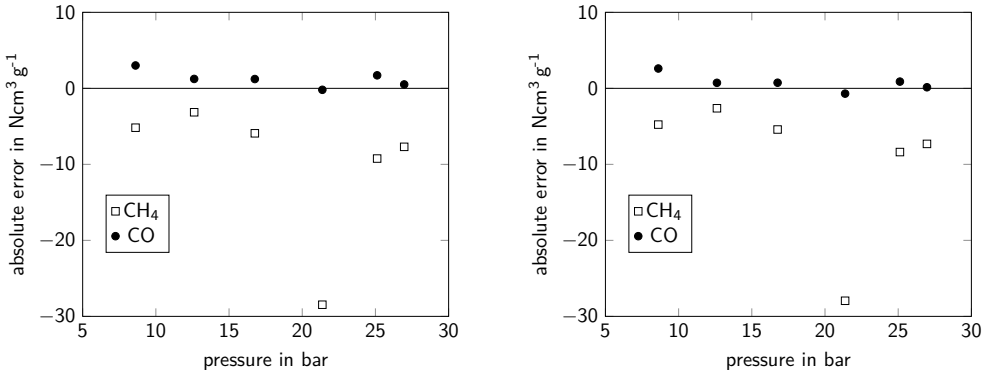


Figure 4.2.: absolute error of the mole fraction of CO for the system CH_4–CO.

The IAST shows a mean absolute error of the mole fractions of 1.7 $\%_{mol}$, a maximum of 4.3 $\%_{mol}$ and a standard deviation of 1.3 $\%_{mol}$. The adsorbed amount is predicted with a mean absolute error of 9.4 $Ncm^3\,g^{-1}$ for CH_4 and 1.0 $Ncm^3\,g^{-1}$ for CO, respectively. The standard deviation is 9.3 $Ncm^3\,g^{-1}$ and 0.8 $Ncm^3\,g^{-1}$, respectively, with a maximum error of 28 $Ncm^3\,g^{-1}$ and 2.6 $Ncm^3\,g^{-1}$, respectively. Without the outlier, the mean absolute error of adsorbed amount is 5.7 $Ncm^3\,g^{-1}$ for CH_4 and 1.0 $Ncm^3\,g^{-1}$ for CO, with a maximum error of 8.4 $Ncm^3\,g^{-1}$ and 2.6 $Ncm^3\,g^{-1}$ and a standard deviation of 2.2 $Ncm^3\,g^{-1}$ and 0.9 $Ncm^3\,g^{-1}$, respectively.



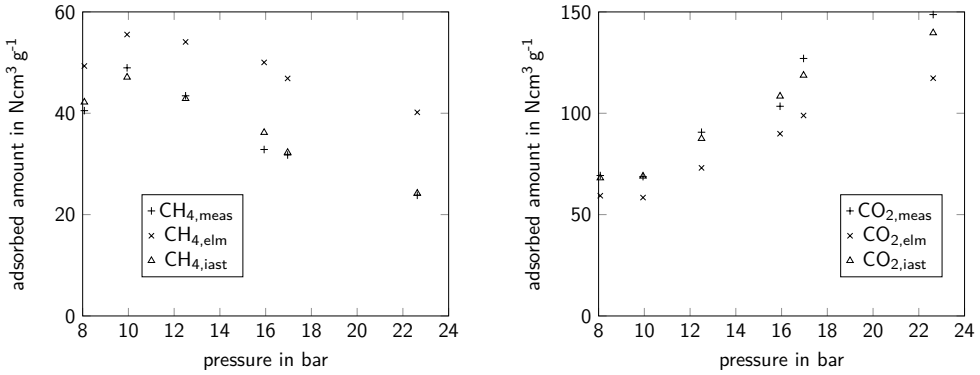
(a) ELM. (b) IAST.

Figure 4.3.: absolute error of the adsorbed amount for the system CH_4–CO.

CH_4–CO_2

The system CH_4–CO_2 is predicted better by IAST than ELM. Figure 4.4 shows the adsorbed amount measured and predicted by both models. There are six data sets with

a pressure range of 8 bar to 23 bar. Figures 4.5 and 4.6 show the absolute error of the mole fraction and adsorbed amount, respectively.



(a) CH_4. (b) CO_2.

Figure 4.4.: Comparison of measurement and model of adsorbed amount for the system CH_4–CO_2.

The mean absolute error of the mole fractions of the adsorbed phase with ELM is 10.2 $\%_{mol}$, with a maximum error of 12.2 $\%_{mol}$ and a standard deviation of 2.0 $\%_{mol}$. The adsorbed amount at equilibrium is predicted with a mean absolute error of 12.4 $Ncm^3\,g^{-1}$ for CH_4 and 18.5 $Ncm^3\,g^{-1}$ for CO_2, respectively. The maximum errors are 17.2 $Ncm^3\,g^{-1}$ and 31.4 , and the standard deviations are 4.4 $Ncm^3\,g^{-1}$ and 9.2 $Ncm^3\,g^{-1}$, respectively.

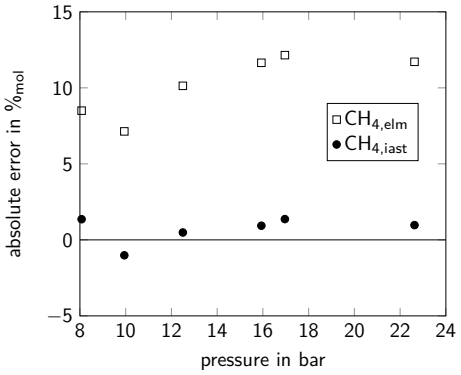
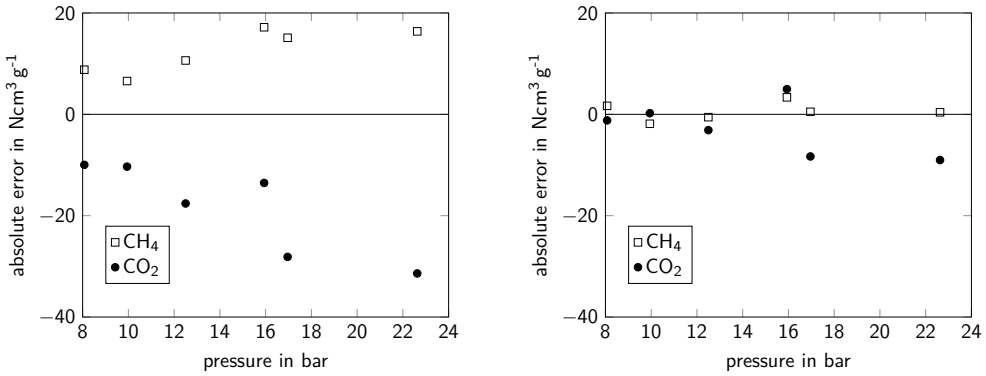


Figure 4.5.: absolute error of the mole fraction of CH_4 for the system CH_4–CO_2.

IAST is predicting this system better than ELM, with a mean absolute error of the mole fractions of 1.0 $\%_{mol}$, a maximum error of 1.3 $\%_{mol}$ and a standard deviation of 0.3 $\%_{mol}$. The adsorbed loading at equilibrium is predicted well with a mean absolute error of 1.4 $Ncm^3\,g^{-1}$ for CH_4 and 4.5 $Ncm^3\,g^{-1}$ for CO_2, respectively. The maximum errors are 3.4 $Ncm^3\,g^{-1}$ and 9.0 $Ncm^3\,g^{-1}$, with a standard deviation of 1.2 $Ncm^3\,g^{-1}$ and 4.5 $Ncm^3\,g^{-1}$, respectively.

CO–H_2

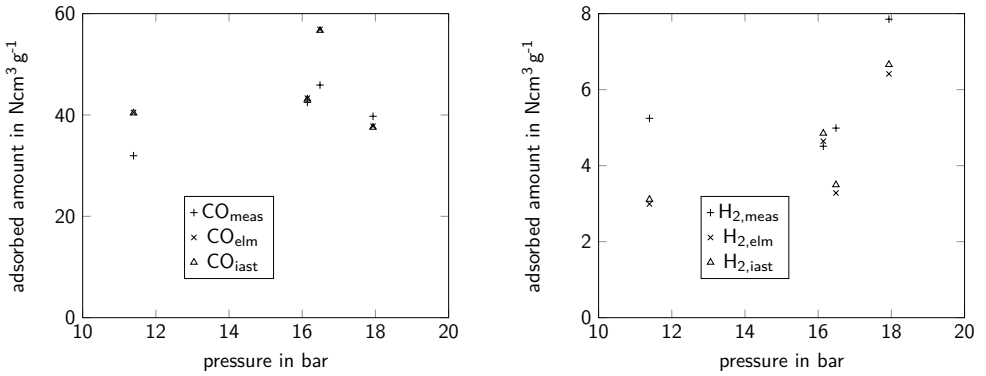
IAST produces better results than ELM at almost all pressures. Figure 4.7 shows the measured equilibrium loading, and the predictions with ELM and IAST. The pressure



(a) ELM. (b) IAST.

Figure 4.6.: absolute error of the adsorbed amount for the system CH_4–CO_2.

varies between 11 bar to 18 bar, with four available data sets. Figures 4.8 and 4.9 show the errors for mole fraction and adsorbed amount at equilibrium, respectively.



(a) CO. (b) H_2.

Figure 4.7.: Comparison of measurement and model of adsorbed amount for the system CO–H_2.

ELM predicts the mole fraction of the adsorbed phase with a mean absolute error of $3.4\,\%_{\mathrm{mol}}$ and a maximum error of $7.2\,\%_{\mathrm{mol}}$. The standard deviation is $3.1\,\%_{\mathrm{mol}}$. The adsorbed amount is calculated resulting in a mean absolute error of $5.6\,\mathrm{Ncm^3\,g^{-1}}$ for CO and $1.4\,\mathrm{Ncm^3\,g^{-1}}$ for H_2, respectively. Here, the maximum error is $11.0\,\mathrm{Ncm^3\,g^{-1}}$ and $2.2\,\mathrm{Ncm^3\,g^{-1}}$, and the standard deviation is $5.0\,\mathrm{Ncm^3\,g^{-1}}$ and $0.9\,\mathrm{Ncm^3\,g^{-1}}$, respectively. When using IAST, the results show a mean absolute error of $2.9\,\%_{\mathrm{mol}}$ for the mole fraction of the adsorbed phase, with a maximum error of $6.9\,\%_{\mathrm{mol}}$ and a standard deviation of $2.9\,\%_{\mathrm{mol}}$. The adsorbed amount at equilibrium is predicted with a mean absolute error of $5.5\,\mathrm{Ncm^3\,g^{-1}}$ for CO and $1.3\,\mathrm{Ncm^3\,g^{-1}}$ for H_2, respectively. Here, the maximum error is $10.8\,\mathrm{Ncm^3\,g^{-1}}$ and $2.1\,\mathrm{Ncm^3\,g^{-1}}$ and the standard deviation is $4.9\,\mathrm{Ncm^3\,g^{-1}}$ and $0.7\,\mathrm{Ncm^3\,g^{-1}}$, respectively.

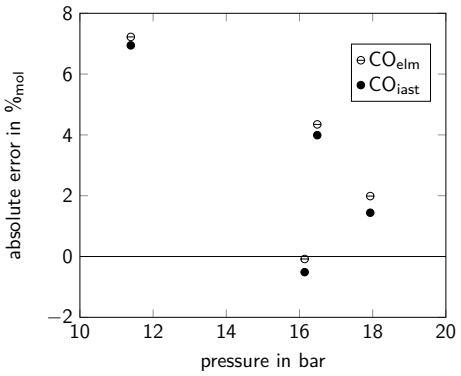
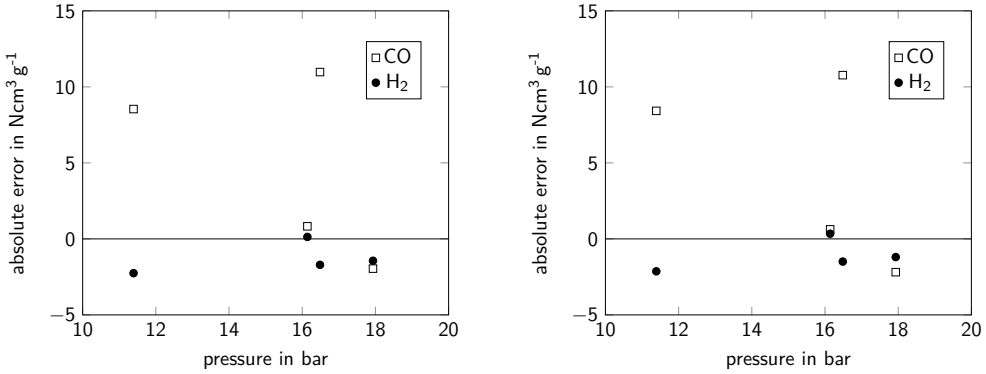


Figure 4.8.: absolute error of the mole fraction for the system CO–H_2.

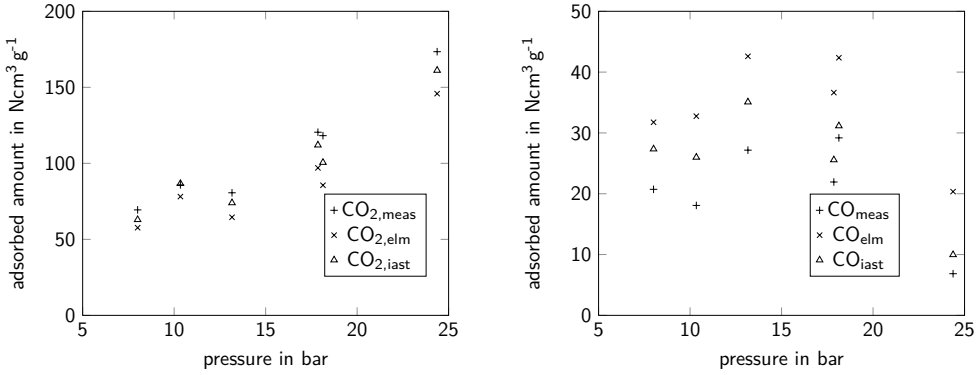


(a) ELM. (b) IAST.

Figure 4.9.: absolute error of the adsorbed amount for the system CO–H_2.

CO_2–CO

Like system CH_4–CO_2, this system is also significantly better predicted by IAST than ELM. Figure 4.10 shows the measured loading at equilibrium, and the predictions of ELM and IAST. Here, a pressure range of 8 bar to 24 bar is available, with six data sets. Figures 4.11 and 4.12 show the errors for mole fraction and adsorbed amount at equilibrium, respectively.



(a) CO_2. (b) CO.

Figure 4.10.: Comparison of measurement and model of adsorbed amount for the system CO_2–CO.

ELM predicts the mole fraction of the adsorbed phase with a mean absolute error of 12.2 $\%_{mol}$, a maximum error of 14.6 $\%_{mol}$ and a standard deviation of 2.1 $\%_{mol}$. The equilibrium loading is calculated resulting in a mean absolute error of 19.9 $Ncm^3\,g^{-1}$ for CO_2 and 13.8 $Ncm^3\,g^{-1}$ for CO, respectively. The maximum error is 32.6 $Ncm^3\,g^{-1}$ and 15.4 , and the standard deviation is 9.6 $Ncm^3\,g^{-1}$ and 1.6 $Ncm^3\,g^{-1}$, respectively.

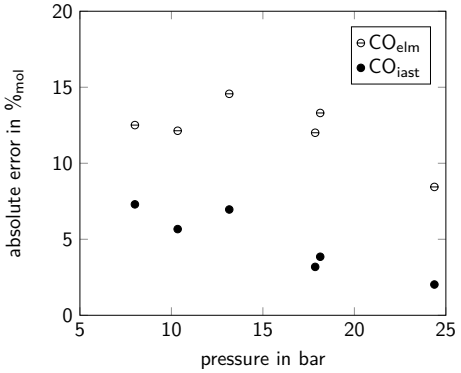
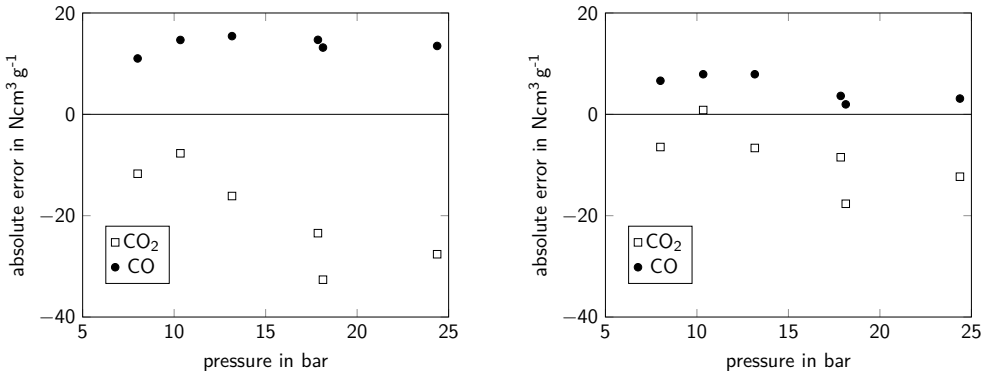


Figure 4.11.: absolute error of the mole fraction for the system CO_2–CO.

As stated, IAST produces better results, with a mean absolute error of the mole fraction of the adsorbed phase of 4.8 $\%_{mol}$, a maximum error of 7.3 $\%_{mol}$ and a standard deviation of 2.1 $\%_{mol}$. The adsorbed amount is predicted with a mean absolute error of 8.7 $Ncm^3\,g^{-1}$ for CO_2 and 5.2 $Ncm^3\,g^{-1}$ for CO, respectively. The maximum error is 17.6 $Ncm^3\,g^{-1}$ and 7.9 $Ncm^3\,g^{-1}$ and the standard deviation is 5.7 $Ncm^3\,g^{-1}$ and 2.6 $Ncm^3\,g^{-1}$, respectively.



(a) ELM. (b) IAST.

Figure 4.12.: absolute error of the adsorbed amount for the system CO_2–CO.

4.2.2. Ternary Systems

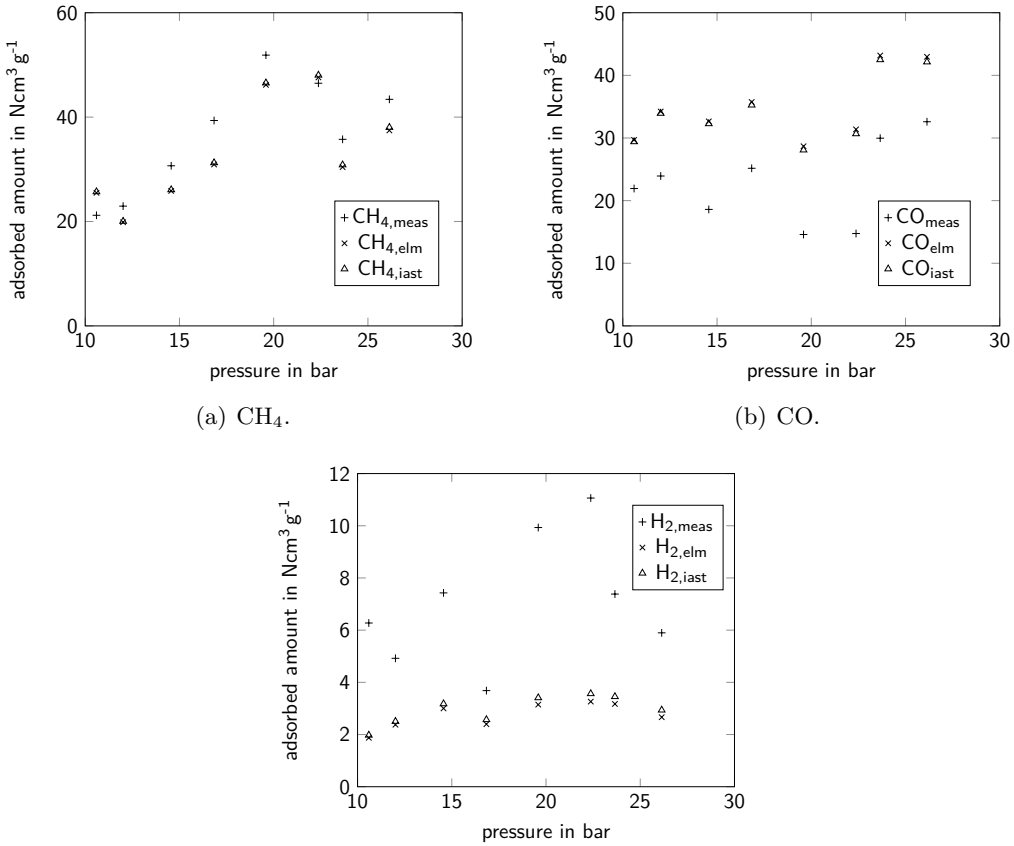
Two ternary systems were reported by [Ritter and Yang, 1987]. The first system is predicted not well by both models. IAST predicts the second system significantly better than ELM.

CH_4–CO–H_2

The system CH_4–CO–H_2 is predicted not well by ELM and IAST. Figure 4.13 shows a comparison between measurement, ELM and IAST. The pressure ranges between 11 bar to 26 bar, with eight data sets available. Figures 4.14 and 4.15 show the errors for mole fraction and adsorbed amount at equilibrium, respectively.
ELM predicts the mole fractions of the adsorbed phase with a mean absolute error of 8.5 $\%_{mol}$ for CH_4, 15.0 $\%_{mol}$ for CO and 6.9 $\%_{mol}$ for H_2, respectively. They have a maximum error of 12.9 $\%_{mol}$, 20.3 $\%_{mol}$ and 11.3 $\%_{mol}$ and a standard deviation of 3.4 $\%_{mol}$, 3.9 $\%_{mol}$ and 3.1 $\%_{mol}$, respectively. The adsorbed amount at equilibrium is calculated resulting in a mean absolute error of 4.8 $Ncm^3\,g^{-1}$ for CH_4, 12.1 $Ncm^3\,g^{-1}$ for CO and 4.3 $Ncm^3\,g^{-1}$ for H_2, respectively. The maximum errors are 8.4 $Ncm^3\,g^{-1}$, 16.6 $Ncm^3\,g^{-1}$ and 7.8 $Ncm^3\,g^{-1}$ and the standard deviations are for each component 2.1 $Ncm^3\,g^{-1}$, 2.9 $Ncm^3\,g^{-1}$ and 2.1 $Ncm^3\,g^{-1}$, respectively.
The calculations using IAST show a mean absolute error of predicted mole fraction of 8.2 $\%_{mol}$ for CH_4, 14.2 $\%_{mol}$ for CO and 6.6 $\%_{mol}$ for H_2, respectively. The maximum error is 12.5 $\%_{mol}$, 19.7 $\%_{mol}$ and 11.0 $\%_{mol}$, and the standard deviation is 3.3 $\%_{mol}$, 3.9 $\%_{mol}$ and 3.1 $\%_{mol}$, respectively. The adsorbed amount is calculated resulting in a mean absolute error of 4.7 $Ncm^3\,g^{-1}$ for CH_4, 11.6 $Ncm^3\,g^{-1}$ for CO and 4.1 $Ncm^3\,g^{-1}$ for H_2, respectively. The maximum error is 8.0 $Ncm^3\,g^{-1}$, 15.9 $Ncm^3\,g^{-1}$ and 7.5 $Ncm^3\,g^{-1}$, and the standard deviation is 1.9 $Ncm^3\,g^{-1}$, 2.8 $Ncm^3\,g^{-1}$ and 2.1 $Ncm^3\,g^{-1}$, respectively.

CH_4–CO_2–H_2

Like some systems before, the system CH_4–CO_2–H_2 is predicted better by IAST than by ELM. Here, the pressure range is 15 bar to 24 bar, with five data sets. Figure 4.16 shows a comparison between measurement, ELM and IAST. Figures 4.17 and 4.18 show the errors for mole fraction and adsorbed amount at equilibrium, respectively.

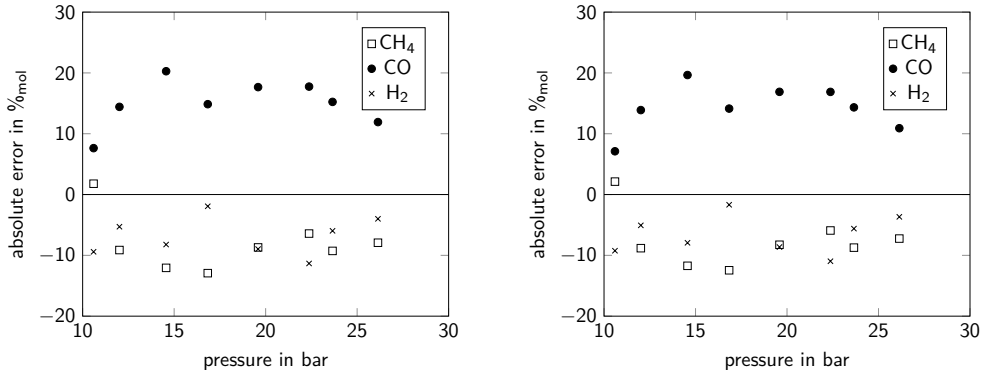


(a) CH_4.

(b) CO.

(c) H_2.

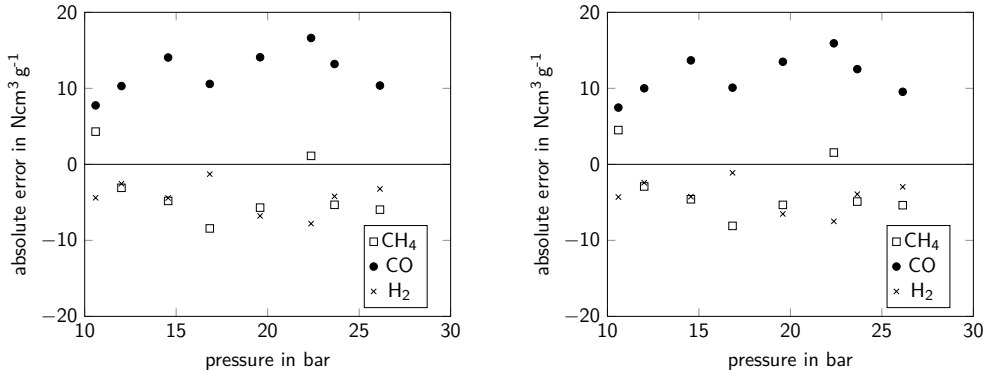
Figure 4.13.: Comparison of measurement and model of adsorbed amount for the system CH_4–CO–H_2.



(a) ELM.

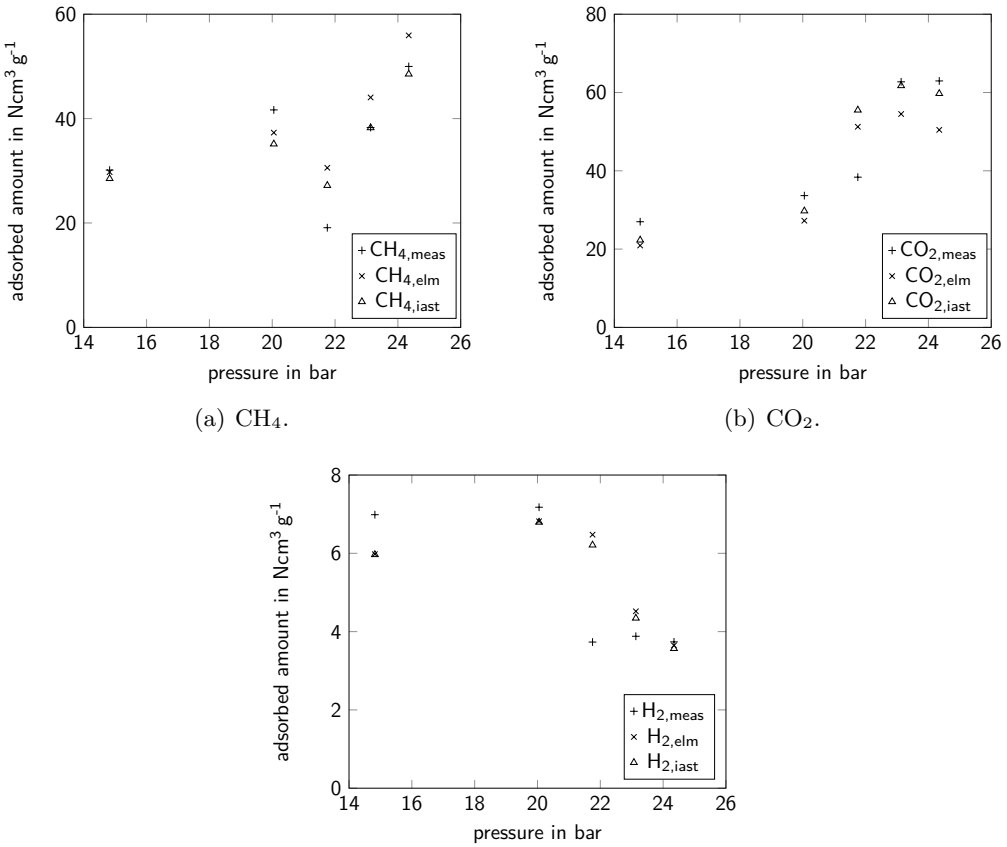
(b) IAST.

Figure 4.14.: absolute error of the mole fraction for the system CH_4–CO–H_2.



(a) ELM. (b) IAST.

Figure 4.15.: absolute error of the adsorbed amount for the system CH_4–CO–H_2.



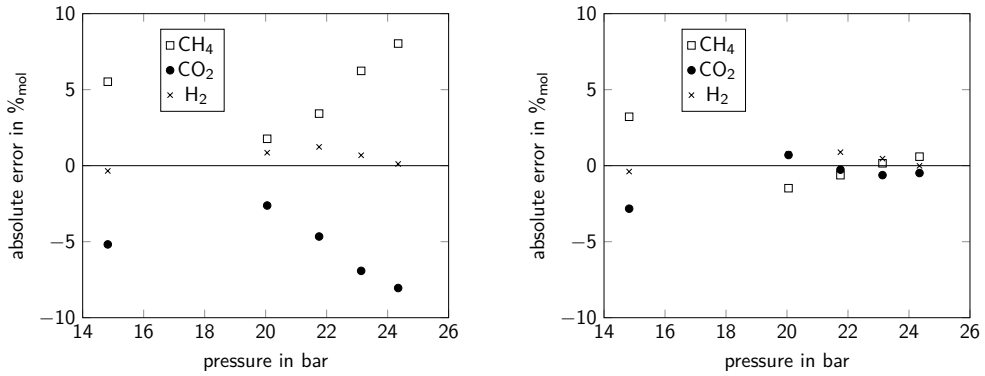
(a) CH_4.

(b) CO_2.

(c) H_2.

Figure 4.16.: Comparison of measurement and model of adsorbed amount for the system CH_4–CO_2–H_2.

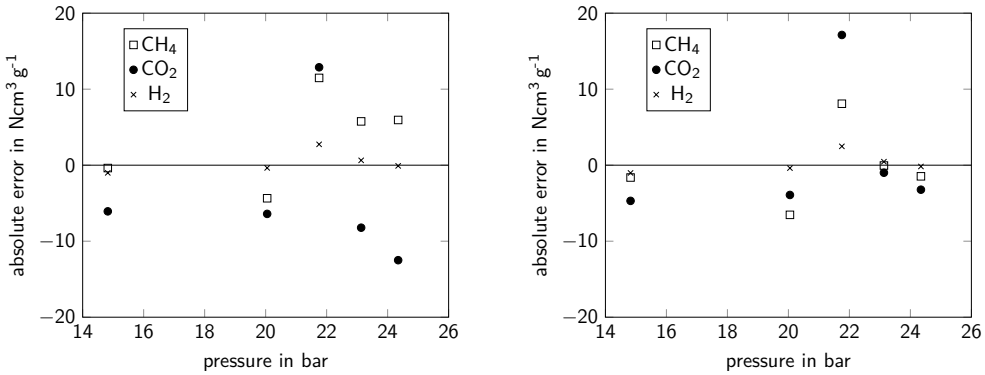
ELM predicts the mole fraction of the adsorbed phase with a mean absolute error of 5.0 $\%_{mol}$ for CH_4, 5.5 $\%_{mol}$ for CO_2 and 0.6 $\%_{mol}$ for H_2, respectively. The results show a maximum error of 8.0 $\%_{mol}$, 8.0 $\%_{mol}$ and 1.2 $\%_{mol}$, and a standard deviation of 2.4 $\%_{mol}$, 2.1 $\%_{mol}$ and 0.4 $\%_{mol}$, respectively. The adsorbed amount is calculated with a mean absolute error of 5.6 $Ncm^3\,g^{-1}$ for CH_4, 9.2 $Ncm^3\,g^{-1}$ for CO_2 and 1.0 $Ncm^3\,g^{-1}$ for H_2, respectively. There is a maximum error of 11.5 $Ncm^3\,g^{-1}$, 12.9 $Ncm^3\,g^{-1}$ and 2.7 $Ncm^3\,g^{-1}$ and a standard deviation of 4.0 $Ncm^3\,g^{-1}$, 3.3 $Ncm^3\,g^{-1}$ and 1.0 $Ncm^3\,g^{-1}$, respectively.



(a) ELM. (b) IAST.

Figure 4.17.: absolute error of the mole fraction for the system CH_4–CO_2–H_2.

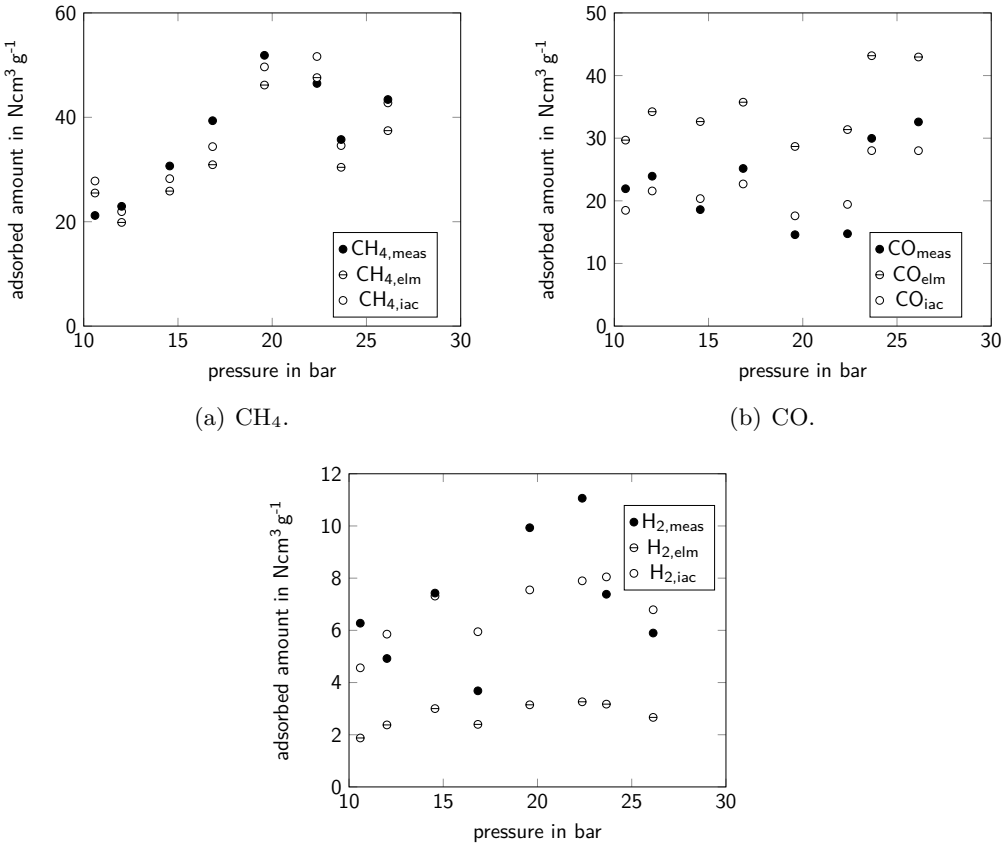
The results obtained with IAST have a mean absolute error of the mole fraction of the adsorbed phase of 1.2 $\%_{mol}$ for CH_4, 1.0 $\%_{mol}$ for CO_2 and 0.5 $\%_{mol}$ for H_2, respectively. The maximum error is 3.2 $\%_{mol}$, 2.8 $\%_{mol}$ and 0.9 $\%_{mol}$, and the standard deviation is 1.2 $\%_{mol}$, 1.0 $\%_{mol}$ and 0.3 $\%_{mol}$, respectively. The adsorbed amount at equilibrium is predicted with a mean absolute error of 3.6 $Ncm^3\,g^{-1}$ for CH_4, 6.0 $Ncm^3\,g^{-1}$ for CO_2 and 0.9 $Ncm^3\,g^{-1}$ for H_2, respectively. The maximum error is 8.1 $Ncm^3\,g^{-1}$, 17.1 $Ncm^3\,g^{-1}$ and 2.5 $Ncm^3\,g^{-1}$, and the standard deviations are for each component 3.5 $Ncm^3\,g^{-1}$, 6.4 $Ncm^3\,g^{-1}$ and 0.9 $Ncm^3\,g^{-1}$, respectively.



(a) ELM. (b) IAST.

Figure 4.18.: absolute error of the adsorbed amount for the system CH_4–CO_2–H_2.

4.2.3. Comparison between ELM and ELM with IAC



(a) CH_4.

(b) CO.

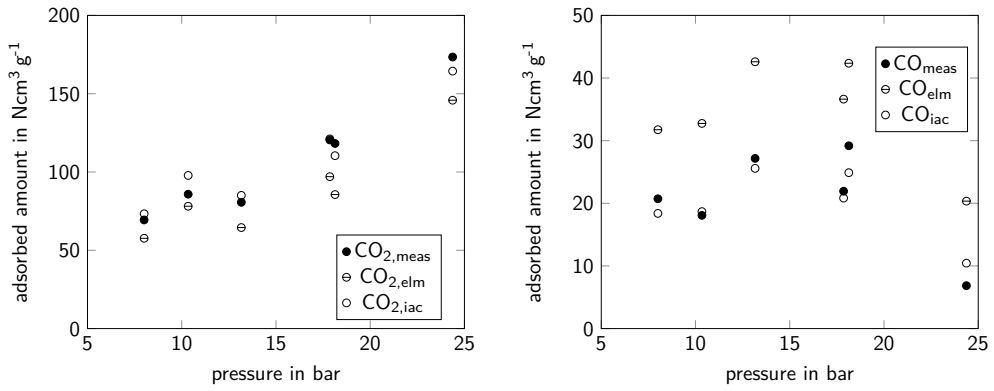
(c) H_2.

Figure 4.19.: Comparison of measurement and model of adsorbed amount for ELM and ELM with IAC for the system CH_4–CO–H_2.

As previously stated, the interaction coefficients improve the prediction of equilibria significantly. The IAC are averaged for one system of species over the entire pressure range. In order for this approach to work, an over- or under-prediction of one species should stay constant, meaning it is either over- or under-predicted. This is not the case for e.g. CH_4 in the system CH_4–CO–H_2, as shown in figure 4.19. The system CO_2–CO meets this condition, and therefore, the IAC improve the prediction noticeable as shown in figure 4.20.

4.3. Kinetics Models

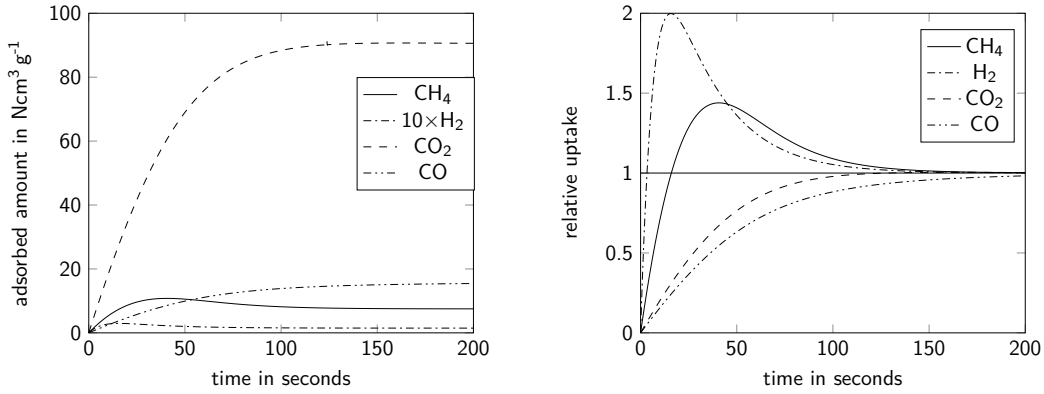
Since the linear driving force model is already available in `adsorpFoam`, only the diffusion-based kinetics are implemented in Octave. The results of the implementation in Octave show an overshoot of the relative uptake of species with low concentration. This behaviour is also reported in literature [Do, 1998]. The relative uptake is defined as:



(a) CO_2. (b) CO.

Figure 4.20.: Comparison of measurement and model of adsorbed amount for ELM and ELM with IAC for the system CO_2–CO.

$$\Theta = \frac{C_i}{C_{eq,i}}. \tag{4.4}$$



(a) Adsorbed amount. Hydrogen is scaled with a factor of ten.

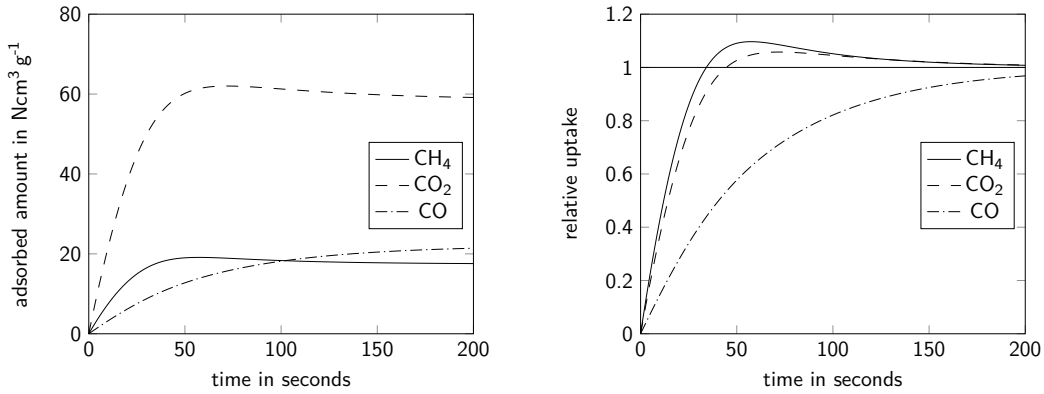
(b) Relative uptake.

Figure 4.21.: Diffusion-based kinetics for the system CH_4–H_2–CO_2–CO.

The simulations are done at a pressure of 100 kPa and a temperature of 298 K. The used equilibrium model is the IAST, and the single-component isotherms are assumed to be of Langmuir type. The two parameters for this isotherm are taken from [Ritter and Yang, 1987]. The denominator for the gradient is chosen arbitrarily as $10^{-4}\,\mathrm{m}^2$. It is chosen so, that the rates are in the same magnitude as those of a linear driving force. This parameter can be adapted later on, using experimental data. The time step is set at 10^{-2} s. For all but the last shown system, no initial loading is assumed. The four systems are chosen arbitrarily and do not match any of the previously discussed systems for equilibrium comparison.
Following gas mole fractions are used for the four-component system:

- $y_{CH_4} = 0.1$, $y_{H_2} = 0.06$, $y_{CO_2} = 0.34$, $y_{CO} = 0.5$

The four-component system shows an overshoot for hydrogen. This is acceptable, since the total adsorbed amount of this species is very low and almost not visible in figure 4.21(a).



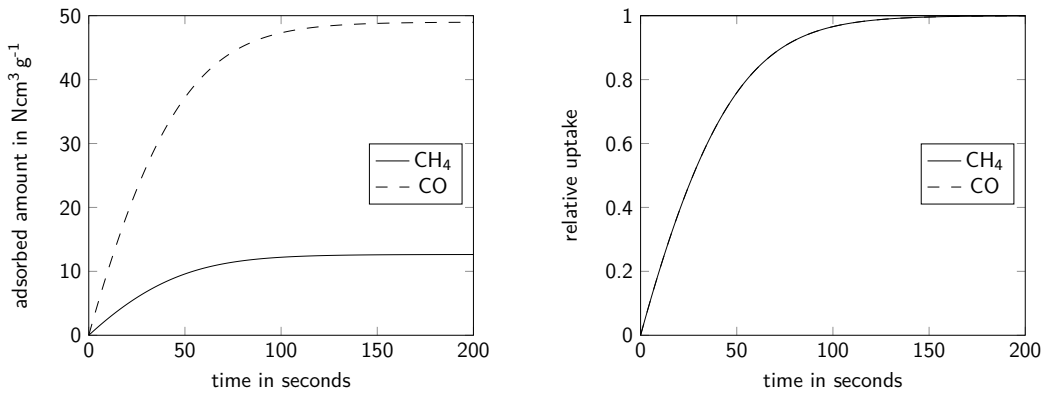
(a) Adsorbed amount. (b) Relative uptake.

Figure 4.22.: Diffusion-based kinetics for the system CH_4–CO_2–CO.

Following gas mole fractions are used for the three-component system:

- $y_{CH_4} = 0.2$, $y_{CO_2} = 0.2$, $y_{CO} = 0.6$

As shown in figure 4.22, the three-component system, containing methane, carbon dioxide and carbon monoxide, is an overshoot visible as well, but not as distinct as before.



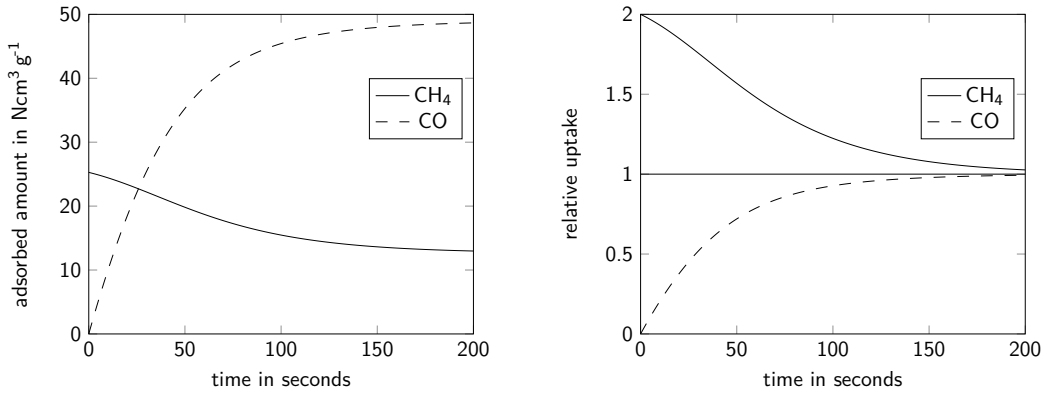
(a) Adsorbed amount. (b) Relative uptake.

Figure 4.23.: Diffusion-based kinetics for the system CH_4–CO.

Following gas mole fractions are used for the two-component systems:

- $y_{CH_4} = 0.1$, $y_{CO} = 0.9$

The relative uptake of the binary system shows hardly any difference between the two species as shown in figure 4.20.
To show the effect of the coupling matrix, one component was chosen to be already present with a loading double of the equilibrium amount. With a linear driving force model, no inflexion point would be expected. However, with diffusion-based kinetics, the effect of the coupling matrix is visible in the first 30 s as shown in figure 4.24.



(a) Adsorbed amount.

(b) Relative uptake.

Figure 4.24.: Diffusion-based kinetics with initial loading for the system CH_4–CO.

CHAPTER 5

Computational Fluid Dynamics

Computational fluid dynamics CFD "focuses on the construction and solution of the governing equations for the different categories of fluid dynamics and the study of various approximations to those equations" (from [Fletcher, 1991]). Since the equations describing flows are non-linear partial differential equations PDE, there is no general analytic solution. Therefore, they have to be solved numerically. With increasing computer performance over the last decades, complex CFD simulations become more and more available to users.
Creating a mathematical model, e.g. a set of PDE which describes a flow, is always the starting point of CFD. Here, the properties of the flow have to be characterised, and simplifications can be made, e.g. neglecting viscosity. The next step is discretising the set of equations in space and time. Depending on the problem, one algorithm might have advantages over the other, for its better capability of solving a specific problem at hand. Normally, these methods have already been implemented in a CFD suite. Next, a grid must be created and the boundary and initial conditions have to be set. Now, with the use of convergence criteria, the simulation may be started. After finishing the calculations, the results can be made visible [Ferzinger and Perić, 2002]. Sometimes, CFD is referred to as 'Colourful fluid dynamics', for its colourful pictures of e.g. flows or pressure drop.

5.1. Mathematical Fundamentals

CFD requires the numerical solution of a PDE. Generally, a linear, second-order, two-dimensional PDE can be written as [Fletcher, 1991]:

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial u^2}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0, \tag{5.1}$$

with constants A to G and $u = u(x, y)$. This type of PDE can be classified into three categories, for their different behaviour of the solutions:

- elliptic PDE with $B^2 - 4AC < 0$,
- parabolic PDE with $B^2 - 4AC = 0$,
- and hyperbolic PDE with $B^2 - 4AC > 0$.

Those types of equations can be solved analytically. However, the equations in CFD are non-linear and cannot be generally solved analytically.

5.1.1. Discretisation

In order to be able to solve an equation numerically, it has to be discretised. In CFD, all calculations are done at points, in volumes or on faces of a mesh. Three different types of grids can be distinguished [Ferzinger and Perić, 2002]:

- Structured or regular grid: this grid type consists of groups of grid lines, and each line in one group does not cross lines from the same group and crosses all members of other groups only once. The grid points can be enumerated and there are efficient algorithms only applicable to structured grids. Consequently, it can only be used on simple geometries. Structured grids can be divided into grids with and without constant grid size. Computations on the former type are sometimes more accurate, since the truncation error made during discretisation and interpolation can be less on grids with constant grid size. Figure 5.1(a) shows an example of a structured grid.
- Block-structured grid: this grid type consists of a number of subdomains which in itself are structured grids. In the interfaces of the subdomains, special care must be taken during calculations. With this type of grid, refinements for e.g. near-wall treatment and more complex geometries are possible to mesh. Figure 5.1(b) shows an example of a block-structured grid.
- Unstructured grid: this grid type does not follow any of the previous rules. Therefore, it is the most flexible, used in automatic meshing and for complex geometries. More complex and generalised algorithms are necessary to solve discretised equations on such grids. Figure 5.1(c) shows an example of an unstructured grid.

With Taylor series expansion, any infinitely differentiable function $f(x)$ can be calculated in proximity to the point x_0:

$$f(x) = \sum_{k=0}^{\infty} \frac{(x - x_0)^k}{k!} \frac{\partial^k f(x_0)}{\partial x^k}. \tag{5.2}$$

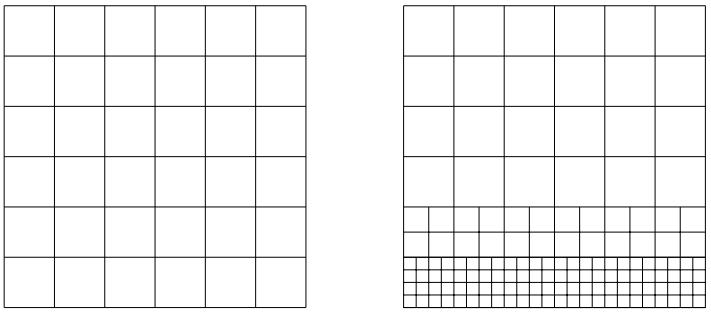
So, the function f can be expressed in the proximity of x_0 as:

$$f(x_0 + \Delta x) = f(x_0) + \Delta x \frac{\partial f(x_0)}{\partial x} + \frac{\Delta x^2}{2} \frac{\partial^2 f(x_0)}{\partial x^2} + \mathcal{O}(\Delta x^3). \tag{5.3}$$

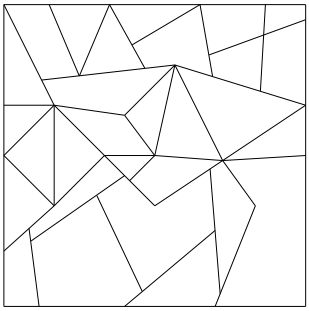
Using Taylor series expansion, the first derivative with respect to x can be discretised as follows:

$$\frac{\partial f}{\partial x} = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} + \mathcal{O}(\Delta x), \tag{5.4}$$

which is a forward-difference scheme of first order, see figure 5.2(a). First order means that the discretisation error scales linear with the step size Δx. This derivative can



(a) A structured grid. (b) A block-structured grid.



(c) An unstructured grid.

Figure 5.1.: Three different types of grids.

also be discretised using a first-order backward-difference discretisation scheme, see figure 5.2(b), and reads as follows:

$$\frac{\partial f}{\partial x} = \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} + \mathcal{O}(\Delta x), \tag{5.5}$$

or, with a second-order central-difference scheme, see figure 5.2(c):

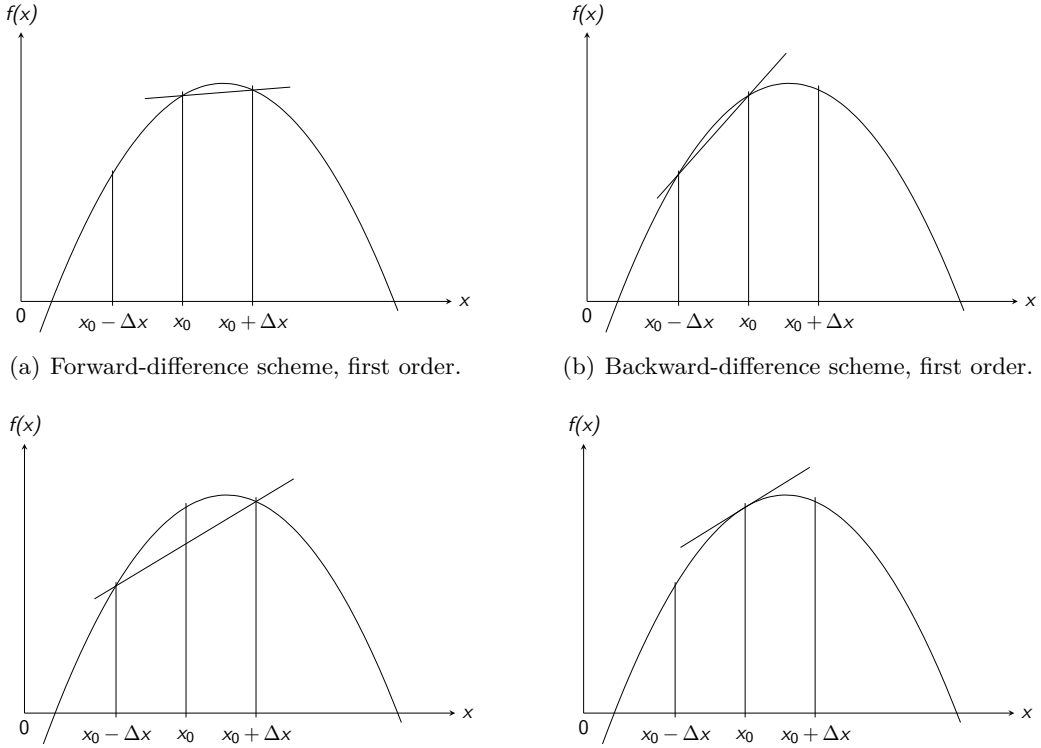
$$\frac{\partial f}{\partial x} = \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x} + \mathcal{O}(\Delta x^2). \tag{5.6}$$

Figure 5.2 shows that the second-order scheme is much better at predicting the actual derivative, shown in figure 5.2(d), than the two first-order schemes.
Generally, a discretisation of the n^{th} derivative of order m with suitable coefficients a_k can be expressed as follows:

$$\frac{\partial^n f}{\partial x^n} = \sum_{k=-\infty}^{\infty} a_k f(x_0 + k\Delta x) + \mathcal{O}(\Delta x^m). \tag{5.7}$$

Normally, a_k takes the following shape, with a rational number a'_k:

$$a_k = \frac{a'_k}{\Delta x^n}. \tag{5.8}$$



(a) Forward-difference scheme, first order.

(b) Backward-difference scheme, first order.

(c) Central-difference scheme, second order.

(d) Actual derivative.

Figure 5.2.: Three different discretisation schemes and the actual first derivative in the point x_o. The straight line in proximity to x_0 represents the discretisation and actual derivative, respectively.

For example, the transient, one-dimensional heat conduction in a solid of length l shall be discretised. It is described by the following equation:

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0, \tag{5.9}$$

with the boundary conditions $T(0,t) = F_1(t)$, $T(l,t) = F_2(t)$ and the initial condition $T(x,0) = G(x)$. This is equivalent to applying a differential operator $\mathcal{H}(*)$ to the function:

$$\mathcal{H}(*) = \frac{\partial(*)}{\partial t} - \alpha \frac{\partial^2(*)}{\partial x^2} = 0. \tag{5.10}$$

The operator $\mathcal{H}(*)$ depends on the form of the PDE. Replacing the differential by differences using Taylor series expansion yields a discretised equation:

$$\frac{T(x,t+\Delta t) - T(x,t)}{\Delta t} - \alpha \frac{T(x-\Delta x,t) - 2T(x,t) + T(x+\Delta x,t)}{\Delta x^2} = 0. \tag{5.11}$$

The above discretisation is called FTCS scheme which stands for "forward in time, centred in space" [IITM, 2016]. It is an explicit scheme, meaning that the unknown values of one time step can be calculated using only already known values from previous time steps and boundary conditions. In contrast, an implicit scheme implies that the unknown values are interdependent and have to be calculated solving a linear system of equations. Implicit schemes are computational more demanding, but generally allow a bigger time step for their better stability.

Convergence

If the solution of the discretised problem tends to the exact solution with the grid spacing approaching zero, the numerical method is called convergent. The Lax-Richtmyer equivalence theorem states, that "a consistent finite difference scheme for a [linear] partial differential equation for which the initial value problem is well-posed is convergent if and only if it is stable" (from [Strikwerda, 2004], with authors note). Therefore, convergence for linear problems implies consistency and stability.
Consistency means that the discretisation should become the original equation in the limit of the discretisation step to zero. Stability implies that a small error, e.g. a round-off error, should decline and not grow.
There are a number of schemes to show stability [Peiró and Sherwin, 2005]. For fluid dynamics, the Courant-Friedrichs-Lewy condition must be satisfied in order to have a stable solution. The so-called Courant number is an indicator of how far a virtual particle can travel during one time step, in terms of the grid size. It is defined as:

$$\mathrm{C} = \frac{u\Delta t}{\Delta x}. \tag{5.12}$$

An explicit scheme for hyperbolic PDE is stable if $\mathrm{C} \leq 1$ [Fletcher, 1991].

Boundedness

Most physical properties are subject to certain restrictions, e.g. the density must never be negative or the mass fraction of one component cannot exceed a value of one. However, very few discretisation methods guarantee such a boundedness. Only some first-order schemes are known to be bounded. All higher-order schemes can produce unbounded results and numerical diffusion, which occurs mostly on coarse grids. Also, unbounded algorithms are prone to having stability and convergence problems [Ferzinger and Perić, 2002].

5.1.2. Finite-difference Method

The finite-difference method solves the discretised PDE at points of a mesh. For example, equation (5.9) can be discretised with the explicit FTCS scheme and rearranged to:

$$T_j^{n+1} = sT_{j-1}^n + (1-2s)\,T_j^n + sT_{j+1}^n, \tag{5.13}$$

where $s = \alpha\Delta t\Delta x^{-2}$, the superscript n denotes the discretisation in time and the subscript j in space, respectively.

5.1.3. Weighted Residual Methods

Weighted residual methods WRM assume an analytically representable, approximate solution which takes the following form for the one-dimensional heat conduction [Fletcher, 1991]:

$$T(x,t) = \sum_{j=0}^{J} a_j(t)\phi_j(x). \tag{5.14}$$

$J > 0$ can be chosen arbitrarily. The higher its value, the more accurate the numerical solution will be and the more computational effort is required. This is an ansatz for $T(x,t)$. In an alternate representation and three dimensions, it reads as:

$$T(\boldsymbol{x},t) = T_0(\boldsymbol{x},t) + \sum_{j=1}^{J} a_j(t)\phi_j(\boldsymbol{x}), \tag{5.15}$$

where T_0 is chosen to satisfy as many boundary and initial conditions as possible.
The functions $\phi_j(\boldsymbol{x})$ can be chosen at will, e.g. polynomials or trigonometric functions. Depending on the problem, the testing functions ϕ_j can be chosen so that they suit the problem. The coefficients $a_j(t)$ are unknown and will be calculated by solving a system of equations.
If the differential operator $\mathcal{H}(*)$ is applied on the the exact solution the equation is equal to zero:

$$\mathcal{H}(T_{exact}) = 0. \tag{5.16}$$

However, doing the same with the numerically obtained solution will yield an error, called residual R:

$$\mathcal{H}(T_{Ansatz}) = R. \tag{5.17}$$

Weighting this residual with a function $W_m(\boldsymbol{x})$ over the computational domain D yields the WRM [Fletcher, 1991]:

$$\iiint_D W_m(\boldsymbol{x}) R \,\mathrm{d}V = 0. \tag{5.18}$$

The weighting function W_m may be chosen freely and some special choices are introduced in the subsequent sections.

Subdomain Method

Splitting up the domain in M subdomains D_m and choosing the weighting function as

$$W_m(\boldsymbol{x}) = 1 \quad \text{for} \quad \boldsymbol{x} \in D_m, \tag{5.19}$$
$$W_m(\boldsymbol{x}) = 0 \quad \text{else}, \tag{5.20}$$

yields the subdomain method. This method is similar to the finite-volume method.

Collocation Method

Choosing certain discretisation points $\boldsymbol{x}_m$ and the weighting function as

$$W_m(\boldsymbol{x}) = \delta\left(\boldsymbol{x} - \boldsymbol{x}_m\right), \tag{5.21}$$

with the Dirac delta function δ, is called collocation method.

Least-square Method

Choosing the weighting function as

$$W_m = \frac{\partial R}{\partial a_m}, \tag{5.22}$$

yields the least-square method. As the name implies, this is equivalent to minimising

$$\iiint_D R^2 \,\mathrm{d}V. \tag{5.23}$$

Galerkin Method

Choosing the weighting function the same as the testing functions:

$$W_m(\boldsymbol{x}) = \phi_m(\boldsymbol{x}), \tag{5.24}$$

yields the Galerkin method, which is similar to the so-called finite-element method.

5.2. Finite-volume Method

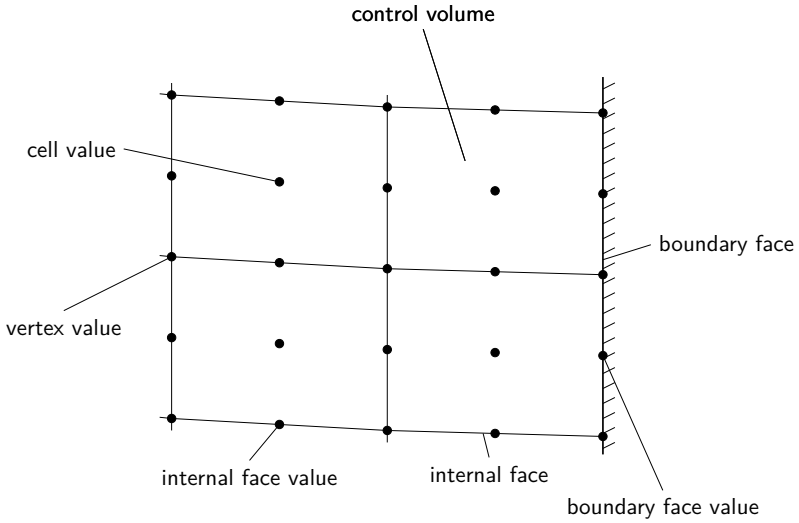


Figure 5.3.: The finite-volume method on a two-dimensional grid (adapted from [CNR, 2016]).

As previously stated, the finite-volume method FVM is a type of WRM. It is similar to the subdomain method, differing only in the absence of an introduction of an approximate solution like equation (5.15). It is chosen for many fluid mechanics problems, since the conservation properties of a physical quantity are preserved [Fletcher, 1991].
In figure 5.3, a two-dimensional finite volume grid is shown. Normally, values of physical properties are stored at the cell centre, called cell value. Some quantities, like temperature, are stored additionally at the boundary faces, called boundary face values.

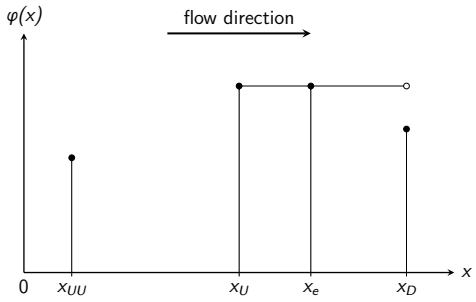
5.2.1. Interpolation

For being able to do calculations at cell faces or points, the values of any physical quantity ϕ defined in the cell centre have to be interpolated. For examples, all surface integrals require physical quantities at the faces of the cells. In figure 5.4, UU, U and D denote the second upstream, first upstream and downstream node, respectively.

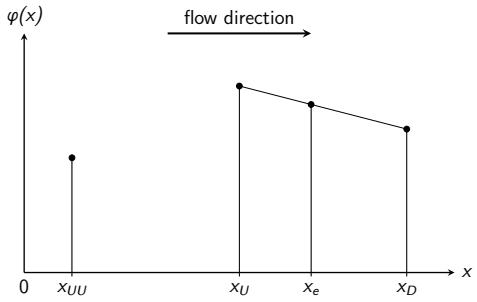
Upwind Interpolation

As the name implies, this schemes is dependent on the flow direction. It is either a backward- or forward-difference approximation for the first derivative. It takes the value for the face from the first cell which is in the direction the flow is coming from. The reason behind this is that it is assumed that some physical properties travel with the flow. It is defined as follows [Ferzinger and Perić, 2002]:

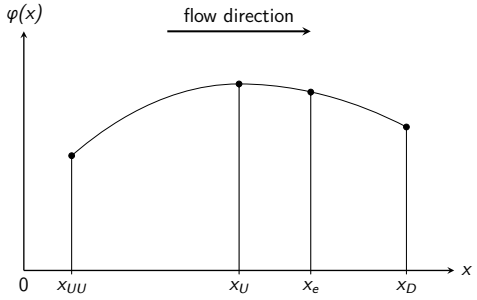
$$\phi_e = \begin{cases} \phi_D, & \text{if } (\boldsymbol{u} \cdot \boldsymbol{n})_e > 0; \\ \phi_U, & \text{if } (\boldsymbol{u} \cdot \boldsymbol{n})_e < 0. \end{cases} \tag{5.25}$$



(a) Upwind interpolation scheme.



(b) Linear interpolation scheme.



(c) QUICK interpolation scheme.

Figure 5.4.: Three different interpolation schemes.

This interpolation scheme does not result in oscillation of the solution and satisfies boundedness, but is prone to being numerically diffusive. It is of first order. A graphical representation of this interpolation scheme is shown in figure 5.4(a).

Linear Interpolation

As the name implies, the value for the face is linearly interpolated between the two nearest nodes:

$$\phi_e = \phi_U + \lambda_e \left(\phi_D - \phi_U\right), \tag{5.26}$$

with the linear interpolation factor λ_e being geometrically defined as:

$$\lambda_e = \frac{x_e - x_U}{x_D - x_U}. \tag{5.27}$$

This schemes is of second order and may produce oscillatory solutions [Ferzinger and Perić, 2002]. A graphical representation of this interpolation scheme is shown in figure 5.4(b).

Quadratic Upwind Interpolation

The quadratic upwind interpolation, also called 'QUICK' which stands for 'quadratic upstream interpolation for convective kinematics', approximates the quantity by a parabola. For a parabola, three points are necessary. Therefore, not only the bordering

neighbours are used, but also at least one additional value is needed. Depending on flow direction, it can be obtained by:

$$\phi_e = \phi_U + g_1(\phi_D - \phi_U) + g_2(\phi_U - \phi_{UU}), \tag{5.28}$$

where D, U and UU denote the downstream, first upstream and second upstream node, respectively [Ferzinger and Perić, 2002]. The coefficients are as follows:

$$g_1 = \frac{(x_e - x_U)(x_e - x_{UU})}{(x_D - x_U)(x_D - x_{UU})}, \tag{5.29}$$

$$g_2 = \frac{(x_e - x_U)(x_D - x_e)}{(x_U - x_{UU})(x_D - x_{UU})}. \tag{5.30}$$

The quadratic upwind interpolation scheme has a truncation error of third order. However, used together with the mid-point rule for calculating surface integrals, the total approximation is of second-order accuracy [Ferzinger and Perić, 2002]. A graphical representation of this interpolation scheme is shown in figure 5.4(c).

5.3. Conservation Equations

The first law of thermodynamics states, that energy must not be created or destroyed. Similar rules apply for momentum and mass. Therefore, every flow has to satisfy some conservation equations.
The change of a physical property $E(t)$ in a material element $V(t)$, i.e. an element which no material enters or leaves, can be described as follows [Kuhlmann, 2007]:

$$\frac{\mathrm{d}E(t)}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \epsilon(t)\,\mathrm{d}V, \tag{5.31}$$

with $\epsilon = E/V$. V_0 denotes a stationary volume and A_0 its boundary. The physical property can be either a scalar or a vectorial quantity. Using the Leibnitz integral rule yields the so-called Reynolds transport theorem:

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V(t)} \epsilon(t)\,\mathrm{d}V = \int_{V_0} \frac{\partial \epsilon}{\partial t}\,\mathrm{d}V + \int_{A_0} \epsilon \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{A}. \tag{5.32}$$

If the physical property E meets a conservation condition, equation (5.32) can be written as

$$\int_{V_0} \frac{\partial \epsilon}{\partial t}\,\mathrm{d}V + \int_{A_0} \epsilon \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{A} = S, \tag{5.33}$$

with a possible source term S.
The first term of equation (5.33) represents the change of the physical property with time inside V_0, the second term describes the change of the physical quantity due to transport over the boundaries of V_0. The latter term is also called convective term.

Applying the Gauß's theorem to equation (5.33) and letting V_0 approach zero, the Reynolds transport theorem can be expressed in differential form:

$$\frac{\partial \epsilon}{\partial t} + \nabla \cdot (\epsilon \boldsymbol{u}) = s, \tag{5.34}$$

with $s = \rho S$.
The integral version of a conservation equation is always valid, whereas the differential version requires the physical property to be continuously differentiable. This condition is not met for e.g. shock waves.

5.3.1. Total Mass Balance

Setting $\epsilon = \rho$ to the density and $s = 0$ yields the mass balance, often called continuity equation [Kuhlmann, 2007]:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0. \tag{5.35}$$

or, with $S = 0$:

$$\int_{V_0} \frac{\partial \rho}{\partial t} \, \mathrm{d}V + \int_{A_0} \rho \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{A} = 0, \tag{5.36}$$

The above equation states, that any mass change inside of the control volume – described with the first term – has to be caused by convective mass transport over the boundaries of V_0 – described with the second term, with no source term possible.

5.3.2. Partial Mass Balance

Setting $\epsilon = \rho_i$ and $s = \nabla \cdot (D \nabla \rho_i) + r_i$, with the diffusion coefficient D and change of mass r_i of component i per volume due to chemical processes, and applying Fick's Law for diffusion, the partial mass balance is obtained:

$$\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i \boldsymbol{u}) = \nabla \cdot (D \nabla \rho_i) + r_i, \tag{5.37}$$

or, with $S = \int_{A_0} D \nabla \rho_i \cdot \mathrm{d}\boldsymbol{A} + R_i$:

$$\int_{V_0} \frac{\partial \rho_i}{\partial t} \, \mathrm{d}V + \int_{A_0} \rho_i \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{A} = \int_{A_0} D \nabla \rho_i \cdot \mathrm{d}\boldsymbol{A} + R_i. \tag{5.38}$$

5.3.3. Momentum Balance

Setting $\epsilon = \rho \boldsymbol{u}$ and $s = -\nabla p + \rho \boldsymbol{f}$, with all external forces per volume $\boldsymbol{f}$, e.g. gravity acceleration, yields the momentum equation of an inviscid fluid [Kuhlmann, 2007]:

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \boldsymbol{u}) = -\nabla p + \rho \boldsymbol{f}, \tag{5.39}$$

or, with $S = -\int_{A_0} p \, \mathrm{d}\boldsymbol{A} + \boldsymbol{F}$:

$$\int_{V_0} \frac{\partial \rho \boldsymbol{u}}{\partial t} \, \mathrm{d}V + \int_{A_0} \rho \boldsymbol{u}\boldsymbol{u} \cdot \mathrm{d}\boldsymbol{A} = -\int_{A_0} p \, \mathrm{d}\boldsymbol{A} + \boldsymbol{F}, \tag{5.40}$$

$$\tag{5.41}$$

Equation (5.39) can be transformed to the so-called Euler equation for inviscid and incompressible, i.e. constant density, flow [Kuhlmann, 2007]:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = \frac{1}{\rho} \nabla p + \boldsymbol{f}. \tag{5.42}$$

If friction is considered, equation (5.40) can be written as:

$$\int_{V_0} \frac{\partial \rho \boldsymbol{u}}{\partial t} \, \mathrm{d}V + \int_{A_0} \rho \boldsymbol{u}\boldsymbol{u} \cdot \mathrm{d}\boldsymbol{A} = \int_{A_0} \mathbf{T} \cdot \mathrm{d}\boldsymbol{A} + \boldsymbol{F}, \tag{5.43}$$

with the stress tensor $\mathbf{T}$, which contains the gradient of pressure.
Assuming a linear dependency of shear rate and shear stress, i.e. Newtonian fluid, the Navier-Stokes equation is obtained, with constant viscosity [Kuhlmann, 2007]:

$$\rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \left(\zeta + \frac{\mu}{3} \right) \nabla \left(\nabla \cdot \boldsymbol{u} \right) + \rho \boldsymbol{f}, \tag{5.44}$$

with the bulk viscosity ζ. The left-hand side describes acceleration. The first term on the right-hand side denotes the change of momentum due to a pressure gradient. The second term stands for the friction of an incompressible fluid. The third term is an additional friction term to account for compressibility effects, and the last term describes all external forces, such as gravity acceleration.

5.3.4. Energy Balance

Setting $\epsilon = \rho \boldsymbol{u}^2/2 + \rho e + \rho g z$ and $s = \dot{q}$, the externally supplied power per volume without the power due to a pressure gradient, and using the definition of enthalpy for an ideal gas $h = e + p\rho^{-1}$ yields the energy balance equation [Kuhlmann, 2007]:

$$\frac{\partial}{\partial t} \left[\rho \left(\frac{\boldsymbol{u}^2}{2} + e + g z \right) \right] + \nabla \cdot \left[\rho \boldsymbol{u} \left(\frac{\boldsymbol{u}^2}{2} + h + g z \right) \right] = \rho \dot{q}, \tag{5.45}$$

or, with $S = \dot{Q}$:

$$\int_{V_0} \frac{\partial}{\partial t} \left[\rho \left(\frac{\boldsymbol{u}^2}{2} + e + g z \right) \right] \mathrm{d}V + \int_{A_0} \rho \left(\frac{\boldsymbol{u}^2}{2} + h + g z \right) \boldsymbol{u} \cdot \mathrm{d}\boldsymbol{A} = \dot{Q}. \tag{5.46}$$

In CFD, all conservation equations and an equation of state have to be solved. Additional equations, e.g. considering chemical reactions, can be added and sometimes, the energy balance is disregarded and the problem is considered to be isotherm.

5.4. Pressure-velocity Coupling

The momentum and continuity equations have to be solved for a flow. Both connect pressure and velocity, making it impossible to solve for one of those two quantities solely. For the sake of clarity, an incompressible and isotherm flow without any external forces is assumed. The two governing equations for this problem are the continuity and the momentum equation [Exeter, 2016]:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{5.47}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} \tag{5.48}$$

For calculating p, $\boldsymbol{u}$ has to be known and vice versa. A couple of algorithms have been developed to solve for both quantities simultaneously or in succession. Here, three algorithms are presented which are of the latter type and used in OpenFOAM.

5.4.1. PISO Algorithm

The pressure-implicit with splitting of operators PISO algorithm solves for a transient flow. It takes an initial guess for pressure p and flux φ, most times the values from the previous time step, and calculates the velocity. Then, a new pressure field can be calculated, using the newly obtained velocity and flux. A flowchart of the algorithm is shown in figure 5.5.

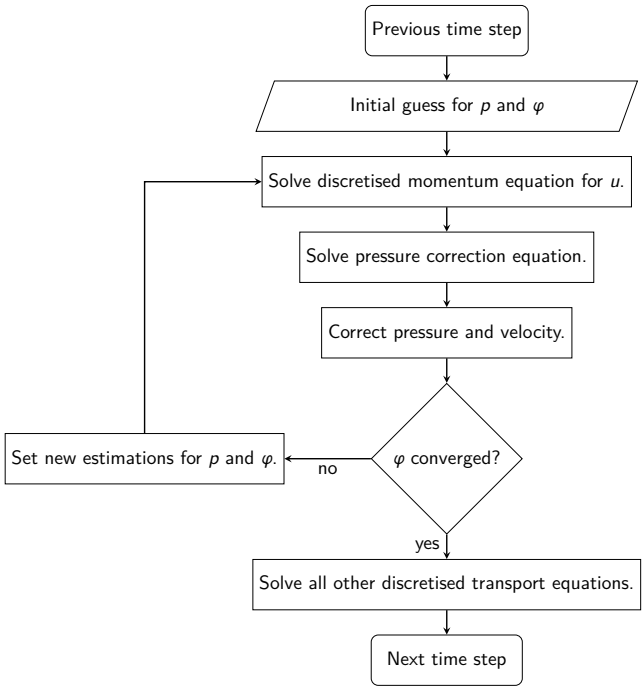


Figure 5.5.: Flowchart of the PISO algorithm (adapted from [Exeter, 2016]).

5.4.2. SIMPLE Algorithm

The semi-implicit method for pressure-linked equations SIMPLE algorithm solves for a steady-state flow. It takes an initial guess for pressure p. With said pressure field, the velocity field is calculated and a pressure correction can be obtained. Using this correction to update a new pressure field leads to a new iteration loop [Patankar, 1980]. A flowchart of the algorithm is shown in figure 5.6.

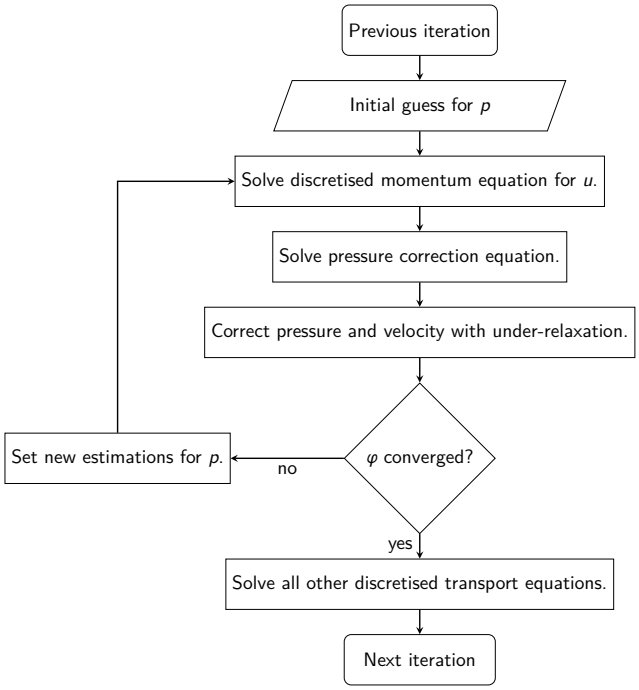


Figure 5.6.: Flowchart of the SIMPLE algorithm (adapted from [Exeter, 2016]).

5.4.3. PIMPLE Algorithm

The merged PISO-SIMPLE (PIMPLE) algorithm combines both PISO and SIMPLE for transient or steady-state flow. It has an outer PISO and an inner SIMPLE loop, allowing the use of bigger time steps and faster simulations [Jasak, 1996]. A flowchart of the algorithm is shown in figure 5.7. The convergence criteria for both loops are often a fixed number of iteration loops.

5.5. Prediction of Material Properties

Since all conservation equations need material properties, they have to be known or calculated. Most of those quantities are dependent on pressure and temperature, and knowing the dependency allows for an approximate calculation. Often, the dependency of pressure is low and can be neglected.

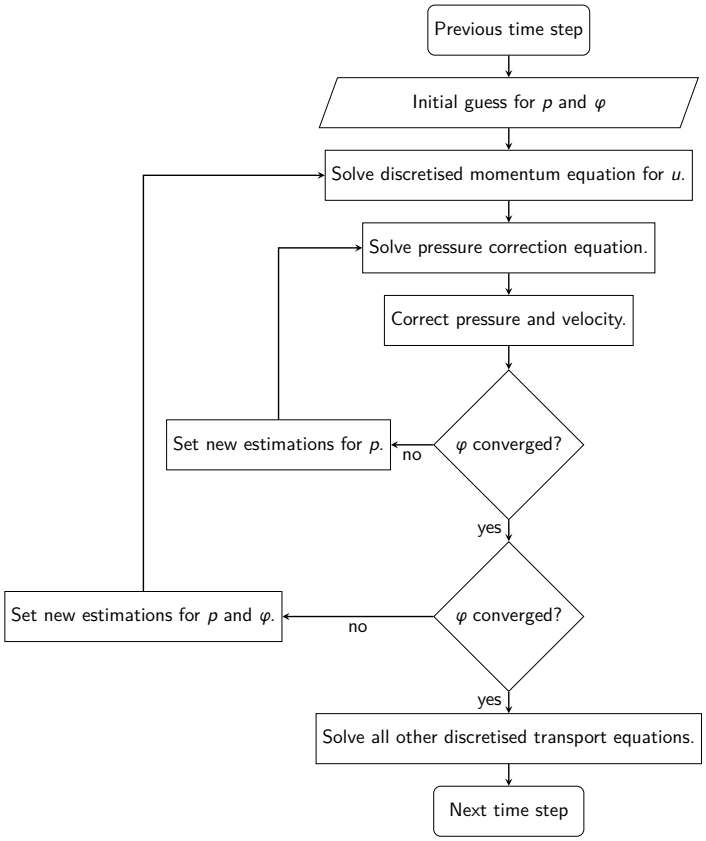


Figure 5.7.: Flowchart of the PIMPLE algorithm [Holzinger, 2014]. Often, the convergence criteria are a fixed number of iteration loops.

5.5.1. Diffusion Coefficients in Gas Mixtures

The diffusion in a gas mixture with more than two components can be estimated using the kinetic theory of gases. First, the diffusion coefficient for a binary mixture can be obtained as follows [Reid et al., 1987]:

$$D_{AB} = \frac{3}{16} \frac{(4\pi kT/M_{AB})^{1/2}}{n\pi\sigma_{AB}^2 \Omega_D} f_D, \tag{5.49}$$

k denotes the Boltzmann constant and n the number of moles. The correction term f_D which accounts for different molar masses is usually close to unity and can be neglected. The averaged characteristic length σ_{AB}, the averaged molar mass M_{AB} and the diffusion collision integral Ω_D are calculated using the following relations:

$$\begin{aligned}
M_{AB} &= \frac{2}{1/M_A + 1/M_B}, \\
\sigma_{AB} &= \frac{\sigma_A + \sigma_B}{2}, \\
\epsilon_{AB} &= (\epsilon_A \epsilon_B)^{1/2}, \\
T^* &= \frac{kT}{\epsilon_{AB}}, \\
\Omega_D &= \frac{A}{(T^*)^B} + \frac{C}{\exp(DT^*)} + \frac{E}{\exp(FT^*)} + \frac{G}{\exp(HT^*)},
\end{aligned}$$

with ϵ_i being the characteristic energy and σ_i being the characteristic length of species i. The assumption of equal binary diffusion coefficients, meaning $D_{AB} = D_{BA}$, is not true e.g. in the case of hydrogen, since it is a small molecule. Therefore, it diffuses much faster than most other gases.
For a mixture of N gases, the diffusion coefficients for each component can be calculated from the binary coefficients:

$$D_{m,i} = \left(\sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{x_j}{D_{ij}} \right)^{-1}. \tag{5.50}$$

CHAPTER 6

Introduction to OpenFOAM

OpenFOAM is an open-source software for solving computational fluid dynamics problems. It is freely available on the Internet under the terms of the GNU General Public License and consists of many small programs, called solvers, tools and libraries. One solver typically deals with the solution for e.g. an inviscid flow or scalar transport due to convection and diffusion. Besides solvers, there are also shared libraries which implement types of boundary conditions or physical models, e.g. for turbulence modelling or chemical reactions. This collection of programs is entirely controlled via a command-line interface, with no default graphical user interface.
Since OpenFOAM is structured using object-oriented programming paradigm, custom solvers and models can be implemented by taking the solver or model which is most similar to the problem at hand, and adapting it. This is also the approach for the work done in this thesis. The OpenFOAM version used in this thesis is 2.4.0 and all source code files shown in this chapter are taken from the official OpenFOAM homepage [OpenFOAM, 2016a].

6.1. User Side

An OpenFOAM case is configured with text files called dictionaries and consists of at least three folders [Greenshields, 2015b]:

- `0`: Here, all the initial and boundary conditions are defined. For example, for a laminar, incompressible and isotherm flow, there would be a `p` and `U` file in this directory. So-called patches represent faces with equal boundary conditions.
- `constant`: Here, configuration files specific to the flow or species are placed, e.g. the chosen turbulence model, chemical reactions or material properties. Also, the mesh is defined in the subdirectory `polyMesh`.
- `system`: Here, time and residual controls, algorithm options and discretisation schemes are specified. Residual and algorithm controls have to be specified in the `fvSolution` file. The discretisation schemes are defined in the `fvSchemes` file. Time step, write intervals and other important general settings are put into the `controlDict` file.

6.1.1. Preprocessing

Before starting the solver, a mesh must be created. OpenFOAM has the ability to create or import one. Creating a mesh is possible using `blockMesh` or meshing an already existing geometry from a file using `snappyHexMesh`. The use of either of those commands requires the existence of an appropriate dictionary, and sometimes other files as well. Additionally, meshes created with other applications can be imported using the `*ToFoam` command family.
Initialising physical quantities can either be done by editing the files in the `0` directory by hand, copying them, or by using the commands `setFields` or `mapFields` [Holzinger, 2014].
When running big geometries with millions of cells, it is often required to parallelise calculations. OpenFOAM provides such a capability by decomposing the computational domain. This is done using the `decomposePar` command.

6.1.2. Starting the Simulation

Once the case is set up, the calculations can be started by typing the name of the chosen solver in the terminal, e.g. `reactingFoam`. Depending on the size of the mesh and complexity of the case, the computation may take seconds to weeks. During calculations, the output is written to time directories as specified in the `controlDict` dictionary.

6.1.3. Postprocessing

After finishing the calculations postprocessing can begin. If more than one processor core was used, the mesh has to be reconstructed using the `reconstructPar` tool. By creating an empty `foam.foam` file in the root case directory or converting the results to the VTK format with the `foamToVTK` command, the results can be displayed and analysed using ParaView.
ParaView is the standard tool for graphical postprocessing. It can display geometries, make physical quantities visible and slice the computational domain to show e.g. pressure drop along the axis of a pipe or mass fraction distribution over a cross section.

6.2. Programming Side

Developing a new solver takes place at a high level of OpenFOAM. Since it is modular and class-based, taking an already existing solver that has the most similarities to the problem at hand is usually the best way to start programming.

6.2.1. General Structure of a Solver

Taking a very simple application like `laplacianFoam`, the general structure of a solver becomes clear. The source code for the main function can be read in listing 6.1. Before the main functions, some basic header files are included in lines 32 and 33 which provide the fundamental CFD and FVM capabilities of OpenFOAM. The main function starts in line 37.

Listing 6.1: structure of `laplacianFoam.C`

```
32 #include "fvCFD.H"
33 #include "simpleControl.H"
34
35 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
36
37 int main(int argc, char *argv[])
38 {
39     #include "setRootCase.H"
40
41     #include "createTime.H"
42     #include "createMesh.H"
43     #include "createFields.H"
44
45     simpleControl simple(mesh);
46
47     // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
48
49     Info<< "\nCalculating temperature distribution\n" << endl;
50
51     while (simple.loop())
52     {
53         Info<< "Time = " << runTime.timeName() << nl << endl;
54
55         while (simple.correctNonOrthogonal())
56         {
57             solve
58             (
59                 fvm::ddt(T) - fvm::laplacian(DT, T)
60             );
61         }
62
63         #include "write.H"
64
65         Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
66             << "  ClockTime = " << runTime.elapsedClockTime() << " s"
67             << nl << endl;
68     }
69
70     Info<< "End\n" << endl;
71
72     return 0;
73 }
```

Mesh and time are created in lines 39 to 42 and input and configuration files, the dictionaries, are read in line 43. Next, the time loop is started (line 51), and the diffusive transport equation of the scalar T is solved using the SIMPLE algorithm in line 59. For non-orthogonal meshes, the PDE can be solved multiple times as shown in line 55, for a better outcome. Afterwards, the results can be written to file, and if the end time is not reached, the next time loop starts. A general overview of the structure of an OpenFOAM solver for transient flow is shown in figure 6.1. If the flow is assumed to be steady-state, the criterion for ending the simulation is whether it is converged, and the time step is replaced by an iteration step.

6.2.2. File Input and Output

OpenFOAM writes the results of flow properties to files in time directories. For example, if a `writeInterval` of 2 was specified in the `controlDict`, then there would be folders named `2`, `4` and so on, in the root folder. Inside each time directory, every property will be written in a file. Subsequently, if there are pressure and velocity to write at each time step, there will be those two files called `p` and `U` in each time directory.

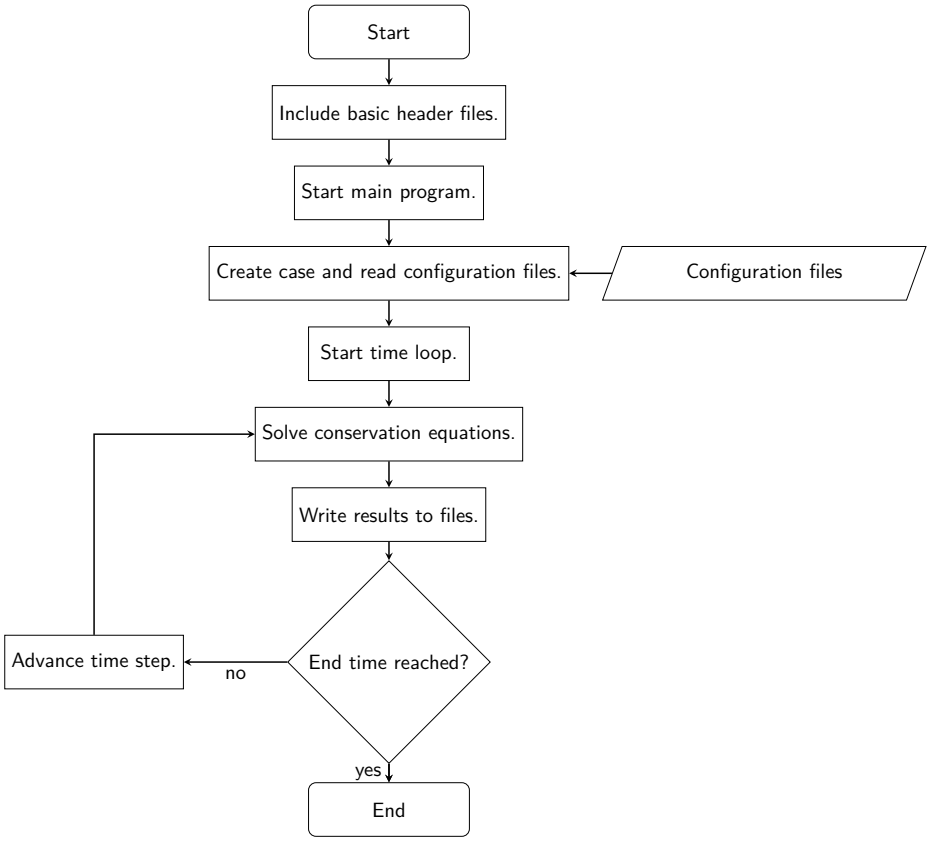


Figure 6.1.: Flowchart of a generic OpenFOAM solver.

Creating field variables and defining which will be printed to file is normally done in the `createFields.H` source file. It is specific to each solver and specifies type and file input/output properties for each necessary field. Taking a look at listing 6.2, the initialisation becomes clear. First, a `volScalarField` which is a type definition for `volField<scalar>`, called `T` is created in lines 3 to 14. Line 8 specifies that this variable has to be read or written to the already mentioned time folders. On lines 10 and 11, the input and output options are specified, respectively. The following statements are valid:

- `MUST_READ`: this file must be present in the time, `constant` or `system` folder.
- `NO_READ`: this file will not be read, and has to be initialised by the program.
- `READ_IF_PRESENT`: if present, this field will be read, otherwise it has to be initialised by the program.
- `MUST_READ_IF_MODIFIED`: if the field has been modified since the last time it was read, it has to be read again. Otherwise, the old values are used.
- `AUTO_WRITE`: this file will be written to file in each time folder.
- `NO_WRITE`: this file will not be written to file and therefore discarded.

Listing 6.2: structure of `createFields.H` for `laplacianFoam`

```
1  Info<< "Reading field T\n" << endl;
2
3  volScalarField T
4  (
5      IOobject
6      (
7          "T",
8          runTime.timeName(),
9          mesh,
10         IOobject::MUST_READ,
11         IOobject::AUTO_WRITE
12     ),
13     mesh
14 );
15
16 Info<< "Reading transportProperties\n" << endl;
17
18 IOdictionary transportProperties
19 (
20     IOobject
21     (
22         "transportProperties",
23         runTime.constant(),
24         mesh,
25         IOobject::MUST_READ_IF_MODIFIED,
26         IOobject::NO_WRITE
27     )
28 );
29
30 Info<< "Reading diffusivity DT\n" << endl;
31
32 dimensionedScalar DT
33 (
34     transportProperties.lookup("DT")
35 );
```

In lines 18 to 28, a new `IOdictionary` is created. As stated previously, dictionaries are used for configuration and setting variables and are usually read from the `constant`

folder, as seen in line 8. Here, the diffusion `DT` is read from the `transportProperties` dictionary in lines 32 to 35.

6.2.3. Data Types

Although the standard C++ data types are available, its use is discouraged. OpenFOAM redefines all variable types for easier changes in future release. Furthermore, the use of arrays is also abstracted.
The datatype `label` is used to store integer data, such as counters and indices of arrays. Dependent whether the program runs on a 32- or 64-bit architecture, it can store a maximum value of either 2×10^9 or 9×10^{18}, respectively [OpenFOAM, 2016b]. A `scalar` is used to store a single floating point number, whereas a `dimensionedScalar` is used for floats with a physical dimension. A `word` is used for storing strings.
Arrays in OpenFOAM are handled with so-called `List<Type>` or `PtrList<Type>`, to store a multitude of variables of a generic type. Properties that are defined in the centre of each cell are stored in a `volField<Type>`. Most physical properties, such as temperature, pressure and mass fraction, are stored in such a field. It is defined in the centres of a cell as shown in figure 6.2(a). Properties that are defined in the centre of each face are stored in a `surfField<Type>`. Therefore, it is defined on cell faces as shown in figure 6.2(b) and is mostly used for intermediate storage of values to evaluate surface integrals. Properties that are defined at points of the mesh are stored in a `pointField<Type>`. Incidentally, it is defined on cell vertices, as shown figure 6.2(c). Additionally, every field contains the values of the boundary faces, as shown in figure 6.2. OpenFOAM uses C++ type definitions to abbreviate variable declaration. Amongst many others, a `volField<scalar>` and `volScalarField` are the same. This makes programming easier and the code is better readable.

6.2.4. Partial Differential Equations

Thanks to object-oriented programming and high abstraction, the formulation of a PDE in OpenFOAM is simple and straightforward. For example, the transient diffusion of a scalar T (Fick's second law) can be expressed by the following PDE, which is also put in code in listing 6.1 on line 59:

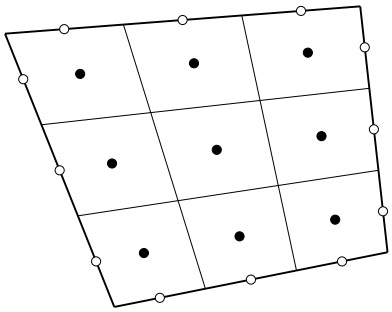
$$\frac{\partial T}{\partial t} - D_t \frac{\partial^2 T}{\partial x^2} = 0. \tag{6.1}$$

6.2.5. Turbulence Modelling

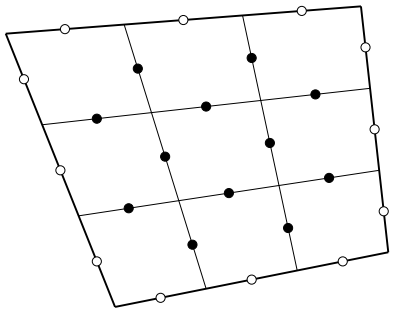
Turbulence modelling is done in a very abstract way, meaning that at the time where the solver was written it is not necessary to know whether and what turbulence model shall be used for the simulation. Also, there is a turbulence model called 'laminar' which disables turbulence modelling.

6.3. Solver for Flows with Chemical Reactions

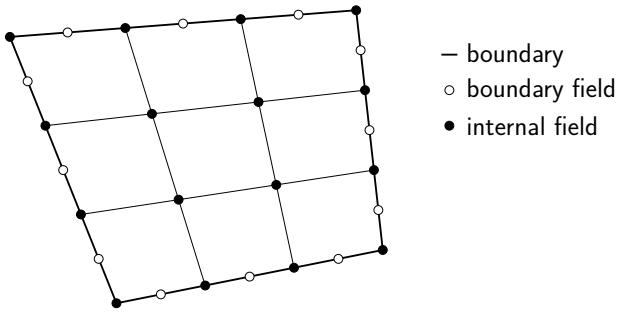
A standard solver for flows with chemical reactions is called `reactingFoam`. According to the official OpenFOAM user guide, `reactingFoam` is a "solver for combustion with



(a) Data points of a `volField<Type>` field.



(b) Data points of a `surfField<Type>` field.



(c) Data points of a `pointField<Type>` field.

Figure 6.2.: Two-dimensional representation of data points for different field variable types in OpenFOAM (adapted from [Greenshields, 2015a]).

chemical reactions" [Greenshields, 2015b]. Turbulence modelling is supported. This solver is also used as basis for `adsorpFoam` [Haddadi et al., 2014] and its extension to multicomponent adsorption `generalMultiAdsorpFoam`. It is capable of solving all conservation equations numerically and can additionally solve for a chemical reaction taking place in the computational domain.
The following files are necessary for running a case with `reactingFoam`:

- `0` directory:
  - The files `p`, `T` and `U` set boundary and initial conditions for pressure, temperature and velocity, respectively.
  - If turbulence modelling is chosen, the appropriate files have to be present.
  - For each species, one file has to be present, e.g. `CH4` and `CO`.
- `constant` directory:
  - The file `chemistryProperties` defines the settings for the chemistry solver.
  - In the file `combustionProperties`, the properties for the combustion model are set.
  - In the file `g`, the gravity acceleration is specified.
  - In the file `reactions`, the names of all species are set and the reactions taking place are defined.
  - In the file `thermo.compressibleGas`, the molar mass, coefficients for calculating the specific heat capacity and transport parameters are put.

  - In the file `thermophysicalProperties`, the thermodynamic models are chosen. Also, the inert species is specified here.
  - In the file `turbulenceProperties`, the chosen turbulence model is put. Additional files are necessary unless 'laminar' is chosen.

- `system` directory:
  - In the files `controlDict`, `fvSchemes` and `fvSolution` general solver settings, like convergence criteria or start and end time, are specified.

6.4. Solver for Flows with Single-component Henry Adsorption

A solver for single-component Henry adsorption called `adsorpFoam`, version 1.3.2, was already developed in the research group [Haddadi et al., 2014]. It has the capability of adsorbing one species. Three additional header files, called `createAdsorptionFields.H`, `adsorption.H` and `adsorptionHeat.H` were written. In the first file, the additional necessary fields and parameters to account for adsorption are created and read, if applicable. In the second file, the actual adsorption equilibrium is calculated. In the third file, the released heat of adsorption and the enthalpy change due to the removal or addition of mass from an adsorbing face into the bordering cell is modelled.
In this version, and also in the multicomponent implementation described later, only adsorption on the surface is considered, without actually modelling the surface in detail with pores or regarding diffusion inside the solid. The height of the layer of the adsorbate is not considered in the mesh, but modelled with a virtual height specified by the user and only relevant for calculating the wall temperature.

6.4.1. Adaptation of Conservation Equations

The actual adsorption calculations are done inside the time loop and before the conservation equations are solved. The results of these calculations, like adsorbed amount, enthalpy change and released heat of adsorption, are introduced to the conservation equations as sink or source terms and therefore, the conservation equations are adapted In the total mass balance, the removal or addition of mass due to adsorption and desorption, respectively, has to be considered. This is done by introducing a term on the right-hand side, as shown in line 7 of listing 6.3.

Listing 6.3: Adapted total mass balance.

```
1 fvScalarMatrix rhoEqn
2 (
3       fvm::ddt(rho)
4     + fvc::div(phi)
5     ==
6       fvOptions(rho)
7     + volAdsorption
8 );
```

In the momentum equation, the removed mass due to adsorption is introduced as an implicit source term with `Foam::Sp` [Greenshields, 2015a], as seen in line 8 of of listing 6.4.

Listing 6.4: Adapted momentum equation.

```
1 fvVectorMatrix UEqn
2 (
```

```
3     fvm::ddt(rho, U)
4   + fvm::div(phi, U)
5   + turbulence->divDevRhoReff(U)
6   ==
7     rho*g
8   + fvm::Sp(volAdsorption, U)
9   + fvOptions(rho, U)
10 );
```

The partial mass balance is solved $N-1$ times, where N is the total number of species. The N^{th} species is the so-called inert species and is calculated indirectly by using the condition that the sum of all mass fractions has to be unity. In this version of `adsorpFoam`, only one species is adsorbing. Its partial mass balance is adapted by introducing the removed mass due to adsorption, as shown in line 15 of listing 6.5. Therefore, the adsorbing species must not be set as the inert species.

Listing 6.5: Adapted partial mass balance.

```
1  fvScalarMatrix YiEqn
2  (
3      fvm::ddt(rho, Yi)
4    + mvConvection->fvmDiv(phi, Yi)
5    - fvm::laplacian(turbulence->muEff(), Yi)
6    ==
7      reaction->R(Yi)
8    + fvOptions(rho, Yi)
9  );
10
11 if (Y[i].name() == adsorpSpecie)
12 {
13     solve
14     (
15         YiEqn == volAdsorption,
16         mesh.solver("Yi")
17     );
18 }
19 else //other species
20 {
21     solve
22     (
23         YiEqn,
24         mesh.solver("Yi")
25     );
26 }
```

The energy balance has to account for the enthalpy loss due to the removal of mass. This is done in line 18 of listing 6.6. The energy balance can either be expressed in terms of enthalpy or internal energy as seen in line 6. The user can select which will be used, and this can remedy some stability problems in certain cases.

Listing 6.6: Adapted energy balance.

```
1  fvScalarMatrix EEqn
2  (
3      fvm::ddt(rho, he) + mvConvection->fvmDiv(phi, he)
4    + fvc::ddt(rho, K) + fvc::div(phi, K)
5    + (
6          he.name() == "e"
7        ? fvc::div
8          (
9              fvc::absolute(phi/fvc::interpolate(rho), U),
10             p,
11             "div(phiv,p)"
12         )
13           : -dpdt
14     )
```

```
15   - fvm::laplacian(turbulence->alphaEff(), he)
16   ==
17     reaction->Sh()
18   + adsorptionEnthalpyChange
19   + fvOptions(rho, he)
20 );
```

The PIMPLE algorithm is used to solve the pressure-velocity coupling. To account for pressure changes due to removal of mass, a sink term is introduced, as shown in line 8 of listing 6.7.

Listing 6.7: Adapted pressure equation for the PIMPLE algorithm.

```
1 fvScalarMatrix pEqn
2 (
3     fvm::ddt(psi, p)
4   + fvc::div(phiHbyA)
5   - fvm::laplacian(rho*rAU, p)
6   ==
7     fvOptions(psi, p, rho.name())
8   + volAdsorption
9 );
```

6.4.2. Temperature and Species Boundary and Initial Conditions

For declaring a wall as adsorbing, the keyword `adsorpWall` has to be set as boundary condition in the species files. The file `adsorption_*` must be present in the appropriate time directory because it will be read. If there is a loading set, it will be used as initial condition.

CHAPTER 7

Implementation in OpenFOAM

Based on `adsorpFoam` version 1.3.2, which was introduced in section 6.4, a multicomponent solver is developed. This version of `adsorpFoam` can only account for the Henry adsorption of one single species. The new solver is called `generalMultiAdsorpFoam`. 'general' indicates that more than one model is available, whereas 'multi' implies the capability of adsorbing multiple species at different boundaries. The coefficients for each species can be defined per boundary patch. The boundary condition files remain unchanged, the file `adsorptionHeat.H` was changed slightly and optimised. The file `createAdsorptionFields.H` was greatly extended.
The implementation of the adsorption model can be structured in two parts: reading the additional parameters as described in the next section, and calculating equilibrium loading and rate of adsorption. In the general structure of an OpenFOAM solver, the adsorption calculations take place at the beginning of each time step. After the rate of adsorption is calculated, it may be introduced to the conservation equations.

7.1. Reading Input Parameters

In order to account for more than one species, the file `createAdsorptionFields.H` is adapted. The coefficients and some fields are stored in arrays, called `PtrList<Type>`. The following arrays for storing coefficients are created and read from the dictionary `adsorptionProperties`:

- `adsorptionType` and `kineticsType` are of type `word` and store the names of the used equilibrium and kinetics model, respectively.
- `Ke` is the Henry coefficient, `b0`, `T0`, `Cm0` and `Cm1` are coefficients for calculating the temperature-dependent Langmuir parameter. All of these variables store values for each species and adsorbing patch and are of type `PtrList<dimensionedScalar>`.
- `iac` is of type `scalarList` and stores the interaction coefficients for ELM per species and adsorbing patch. This parameter is optional; if not present, it will be set to unity, which simply disables IAC.
- `K1`, `K2` and `K3` are the coefficients for the linear driving force model for adsorption kinetics. They are of type `PtrList<dimensionedScalar>` and store values for each species and each adsorbing patch.

- `sigma`, `epsilon` and `diffusionDeltaZ` are used for the diffusion-based adsorption kinetics model. The first two are of type `PtrList<dimensionedScalar>` and store values for each species, the latter is of type `dimensionedScalar`.
- `adsorptionDeltaH` is of type `PtrList<dimensionedScalar>` and specifies the heat of adsorption. It stores values for each species and each adsorbing patch. `adsorbentCp` is the specific heat capacity at constant pressure of the adsorbent and `adsorbentDensity` is the density of the solid adsorbent. `adsorbentLayerH` is the virtual height of the adsorbed layer. All three are of type `dimensionedScalar`. With this parameters, the temperature of the adsorbing walls is calculated.

The following variables are fields and store calculated adsorption values:

- `adsorption` stores the currently adsorbed amount per species. It is not reset at the beginning of `adsorption.H`. `eqAdsorption` is used to store the equilibrium loading of each species. `surfAdsorption` is used to store the adsorption rate per species. All those variables are of type `PtrList<volScalarField>` and the values are defined at the middle of each adsorbing patch face.
- `volAdsorption` is used to store the adsorption rate per volume per species. The field `cellAdsorption` is used to store the adsorbed amount during one time step in the volume per species. Both are defined in the centre of each cell bordering an adsorbing patch and are of type `PtrList<volScalarField>`.
- `adsorptionEnthalpy` is used to store the released heat of adsorption per time and volume for all species. `removalEnthalpy` stores the change of enthalpy due to the removal or addition of mass. Both are of type `PtrList<volScalarField>` and the first is defined in the centre of each adsorbing face, the latter is defined in the centre of each cell bordering an adsorbing patch.
- `totVolAdsorption` and `totRemovalEnthalpy` are of type `volScalarField` and store the total adsorption rate per volume and the total enthalpy change due to removal of mass per volume, respectively. The values stored are defined in the cell centre at each cell which borders an adsorbing patch.

`adsorpPatches` and `adsorpPatchesLabels` are helper variables to store the names of the adsorbing patches and their number in the list of all patches, respectively. They are of type `wordList` and `labelList`, respectively. The only adsorption fields written to file are `adsorption` and `eqAdsorption`. All other fields are only used internally for calculations.

7.2. Adsorption Calculations

The file `adsorption.H` serves as interface for the equilibrium and kinetics calculation files and handles initialisation and applies limiters. Its flowchart is shown in figure 7.1.

7.2.1. Preparations

First, the mole fractions of the gas phase are calculated in all cells bordering an adsorbing patch. If the ELM or IAST with Langmuir type as single-component isotherm is selected,

the parameters for b and C_m^m according to equations (2.16) and (2.17) are calculated on all adsorbing patch faces.

7.2.2. Calculating the Adsorption Equilibrium

For flexibility, each algorithm for calculating adsorption equilibria is implemented in a separate file. This makes it possible to easily add a new model later on. After the calculations of the adsorption equilibria are done, the field variable `eqAdsorption` has to be set for all species to the absolute adsorbed amount in kg. Dividing by area will be done before calculating the released heat of adsorption, for this greatly simplifies some calculations and checks.

Henry Adsorption

If selected, the equilibrium according to Henry is calculated. The algorithm is implemented in the file `henry.H`. It first loops over all species, and then over all faces of all adsorbing patches. The Henry isotherm was kept from the original implementation and this implementation remains in the code for historical purposes.

Langmuir Adsorption

It would be possible to implement single-component Langmuir adsorption for all species, meaning no coupling between the isotherms of the species. Since the ELM is available, this implementation was not done in the frame of this thesis.

Extended Langmuir Model

If selected, the equilibrium according to the extended Langmuir model is calculated. The algorithm is implemented in the file `elm.H`. It first loops over all species, and then over all faces of all adsorbing patches.

Ideal Adsorbed Solution Theory

If selected, the equilibrium according to the ideal adsorbed solution theory is calculated. The algorithm is implemented in the file `iast-langmuir.H`. As the name of the file implies, the single-component isotherm used is of Langmuir type. The parameter C_m^m is converted to be mole-based. It first loops over all faces of all adsorbing patches, and then over all species, because this simplifies the implementation of the algorithm. The resulting mole-based equilibria are converted to mass-based ones.

7.2.3. Calculating the Rate of Adsorption

Again, each algorithm for calculating the rate of adsorption is implemented in a separate file. The appropriate file is selected by specifying a model. After the calculations are done, the field variables `surfAdsorption` and `cellAdsorption` have to be set for all species in $\mathrm{kg\,s^{-1}}$ and kg, respectively. The stored values are not divided by volume yet, which will be done later.

Linear Driving Force Kinetics

In the file `ldf.H`, the algorithm to calculate the adsorption rate using the linear driving force kinetics model is implemented. First, it loops over all species, then over all faces of all adsorbing patches.

Diffusion-base Kinetics

In the file `diffusion.H`, the algorithm to calculate the adsorption rate according to the diffusion-based kinetics model is implemented. It is described in section 3.5. First, the diffusivity of each species in the mixture is computed in a separate file called `calcDiffusivity.H`. Next, it loops over all faces of all adsorbing patches, then over all species.

7.2.4. Applying Limiters

After the rate of adsorption is calculated, two circumstances may occur which have to be avoided. To remedy this problem, limiters have to be applied and the time step has to be reduced manually in order to get physically correct results. The possible cases are:

- Too high rate of adsorption: if the amount adsorbed during one time step exceeds the available mass in the cell bordering the adsorption faces in question, a so-called positive adsorption limiter has to be applied. The amount adsorbed on the faces in this time step is set to the available mass inside the cell, and a warning is issued.
- Too high rate of desorption: if the currently adsorbed amount exceeds the equilibrium loading, desorption takes place. It may occur that the amount desorbing tops the available mass on the adsorbing faces. In that case, a so-called negative adsorption limiter is applied and the desorbed amount is reduced to the maximum available. Then, a warning is issued.

Ignoring these warnings will generate physically incorrect results.

7.2.5. Division by Area

To finalise results, the field variables are divided by area, since the adsorbed amount is stored as value relative to the surface and not absolute value. Not diving by area during the equilibrium calculations allows to add the absolute values.

7.2.6. Adsorption Enthalpy

The next step is to calculated the released heat of adsorption, the change of temperature of the faces of adsorbing patches and the heat flux from the wall to the fluid. All those calculations are done in the file `adsorptionHeat.H`. First, the released adsorption enthalpy is calculated and stored per species in the field variable `adsorptionEnthalpy`. Next, the heat flux from the wall to the volume due to a possible temperature gradient is computed. This is necessary for the following step in which the temperature of the faces of adsorbing patches is updated. Last, the enthalpy change due to removal or addition of mass is calculated and stored in the field variable `totRemovalEnthalpy`.

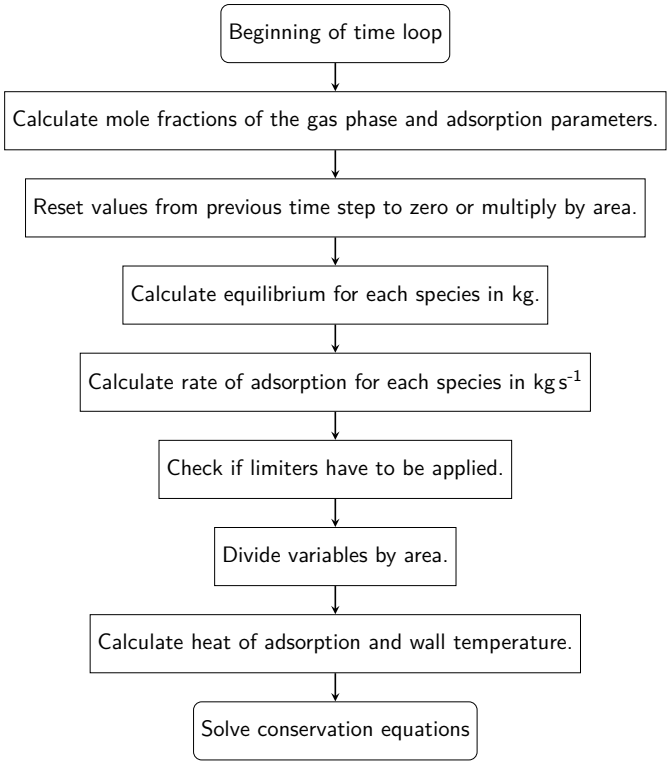


Figure 7.1.: Flowchart of the adsorption implementation.

7.2.7. Pitfalls

Some pitfalls have to be considered during programming and using the solver. Those are:

- Different face sizes of adsorbing patches: the calculations of equilibrium are all done in absolute values. The reason for this is, that loadings cannot be added if the face sizes vary in area. This makes the calculations of equilibrium less error-prone since the division by area is done after the equilibrium and rate of adsorption calculations.
- One cell with more than one adsorbing face: to decide whether to apply a limiter or not, the total adsorbed amount per species is needed for each cell bordering an adsorbing face. Since the rate of adsorption computations are done by looping over faces, the check whether to apply a limiter has to be done after the calculation of the rate of adsorption in an extra loop and cannot be merged.
- User input check: if the user input is not correct, e.g. setting some coefficients to zero, there will be a warning issued and variables that would cause a crash, e.g. dividing by zero, are ignored and set to default values. However, disabling adsorption for one species can be done safely by setting all of its equilibrium parameters to zero. In that case, no warning will be issued.
- Disabling one species: if one species should not be adsorbing on one patch, but on all other adsorbing patches, it does not suffice to disable kinetics. The equilibrium of said species would have an impact on the other species which is probably not

wanted in most cases. Therefore, the equilibrium parameters have to be set to zero if a species should not adsorb on a specific patch. For ELM, it does not suffice setting the monomolecular layer capacity to zero. The ratio of adsorption and desorption parameter has to be zero as well.

- Setting diffusion-based kinetics in combination with Henry adsorption does not make sense and is not allowed. This is because there is no interdependence between the species with Henry adsorption, which would mean that diffusion-based kinetics would degenerate to linear driving force kinetics. Therefore, only a linear driving force model can be selected.
- The temperature change between two time steps is considered very small and negligible. Therefore, this allows the use of isotherms and simplifies the governing equations for the diffusion-based kinetics. But this implies that the time step is sufficiently low.

7.3. Adaptation of Conservation Equations

The conservation equations are adapted in a similar way as described in section 6.4, with some minor differences. First, not only the adsorbed mass of one, but all components is used. Second, all partial mass balances are adapted with the loss of mass due to adsorption of the according single species, as shown in listing 7.1.

Listing 7.1: Adapted partial mass balance.

```
1  fvScalarMatrix YiEqn
2  (
3      fvm::ddt(rho, Yi)
4    + mvConvection->fvmDiv(phi, Yi)
5    - fvm::laplacian(turbulence->muEff(), Yi)
6    ==
7      reaction->R(Yi)
8    + volAdsorption[i]
9    + fvOptions(rho, Yi)
10 );
```

7.4. Information Output

At the end of each time step, a summary of the adsorption properties is printed to standard output. An example is shown in listing 7.2. This is done in the `adsorptionInfo.H` file. The following information is shown:

- Each adsorbing patch with number and name. For each patch, the following patch-averaged data for all species are shown:
  - The name of the single species.
  - The rate of adsorption per single species in $\mathrm{kg\,s^{-1}\,m^{-2}}$.
  - The current adsorption loading per single species in $\mathrm{kg\,m^{-2}}$.
  - The equilibrium adsorption loading per single species in $\mathrm{kg\,m^{-2}}$.
  - The released heat of adsorption per single species in $\mathrm{W\,m^{-2}}$.

Listing 7.2: An example clipping of a log output.

```
Patch 1 (adsDown)
Component CH4
Adsorption rate = 2.93139e-05 kg/(s*m2)
Adsorption loading = 9.27656e-07 kg/m2
Equilibrium adsorption loading = 0.00663936 kg/m2
Adsorption enthalpy = 29.3139 W/m2

Component CO
Adsorption rate = 1.02823e-05 kg/(s*m2)
Adsorption loading = 3.2539e-07 kg/m2
Equilibrium adsorption loading = 0.00393219 kg/m2
Adsorption enthalpy = 10.2823 W/m2
```

For more detailed information, the files in the time directories have to be regarded.

7.5. Boundary Conditions

The same boundary conditions for temperature and species as for `adsorpFoam` have to be set, see section 6.4. Now, all species have to be set adsorbing by specifying the boundary condition as `adsorpWall` for the adsorbing patches. Furthermore, the files `adsorption_*` do not have to be present in the appropriate time directory. However, it is possible to specify loadings in those files, and the solver will use them as initial conditions.

7.6. Example Case Setup

Here, only the main differences to setting up a case for `reactingFoam` are pointed out. To setup an example case for `generalMultiAdsorpFoam`, the following files have to be present in the directories:

- `0`: the boundary conditions for the species and temperature have to be changed for each adsorbing patch to `adsorpWall` and `adsorpAdiabaticWall`, respectively. Valid choices for the first are `henry`, `elm` and `iast-langmuir` and for the latter `ldf` and `diffusion`, respectively.
- `constant`: additionally to the dictionaries necessary for running `reactingFoam`, the `adsorptionProperties` dictionary has to be present.
- `system`: no changes to a normal `reactingFoam` case setup are necessary.

The boundary conditions for the species and temperature have to me modified. All patches that are adsorbing have to be of type `adsorpWall` in all species files, and of type `adsorpAdiabaticWall` for the temperature.

7.6.1. The `adsorptionProperties` Dictionary

In the `adsorptionProperties` dictionary, the following parameters have to be defined:

- `adsorptionType` and `kineticsType` have to be set to the name of the appropriate equilibrium and kinetics model, respectively. Valid options for the former are `henry`, `elm` and `iast-langmuir`. Available kinetics models are `ldf` and `diffusion`.

- `adsorbentCp`, `adsorbentDensity` and `adsorbentLayerH` have to be set as dimensionful quantities.
- If the diffusion-based kinetics model is selected, `diffusionDeltaZ` has to be specified as dimensionful quantity.
- For each species, the following parameters have to be set:
  - If the diffusion-based kinetics model is selected, the parameters `sigma` and `epsilon` have to be set. They are dimensionful quantities.
  - For all adsorbing patches, the following parameters have to be specified:
    * If ELM or IAST with Langmuir type single-component isotherm is selected, the parameters `Cm0`, `Cm1`, `b0` and `T0` have to be set as dimensionful quantities.
    * If Henry adsorption is selected, the Henry coefficient `Ke` has to be specified as dimensionful quantity.
    * If the linear driving force kinetics model is selected, the kinetic parameters `K1` to `K3` have to be set as dimensionful quantities.
    * If the ELM is chosen, the interaction coefficients can be set per species and adsorbing patch by specifying `iac` in the dictionary. If not present or set to zero, they will be ignored.

An example dictionary with two adsorbing patches called `adsUp` and `adsDown`, two species and IAST and diffusion-based kinetics is shown in listing 7.3. A full set of working configuration files can be found in appendix B.

Listing 7.3: An example clipping of an `adsorptionProperties` dictionary.

```
1  adsorptionType iast-langmuir;
2  kineticsType   diffusion;
3
4  adsorbentCp          adsorbentCp          [0 2 -2 -1 0 0 0]    1e3;
5  adsorbentDensity     adsorbentDensity     [1 -3 0 0 0 0 0]     1e3;
6  adsorbentLayerH      adsorbentLayerH      [0 1 0 0 0 0 0]      1e-3;
7  diffusionDeltaZ      diffusionDeltaZ      [0 1 0 0 0 0 0]      1e-2;
8
9  CH4
10 {
11     adsUp
12     {
13         Cm0      Cm0      [ 1 -2 0 0 0 0 0]   0.15259;
14         Cm1      Cm1      [ 1 -2 0 -1 0 0 0]  -1.9851e-4;
15         b0       b0       [ -1 1 2 0 0 0 0]   5.5259e-9;
16         T0       T0       [ 0 0 0 1 0 0 0]    1730.0;
17
18         adsorptionDeltaH adsorptionDeltaH [0 2 -2 0 0 0 0] 1e6;
19     }
20
21     adsDown
22     {
23         Cm0      Cm0      [ 1 -2 0 0 0 0 0]   0.15259;
24         Cm1      Cm1      [ 1 -2 0 -1 0 0 0]  -1.9851e-4;
25         b0       b0       [ -1 1 2 0 0 0 0]   5.5259e-9;
26         T0       T0       [ 0 0 0 1 0 0 0]    1730.0;
27
28         adsorptionDeltaH adsorptionDeltaH [0 2 -2 0 0 0 0] 1e6;
29     }
30
31     sigma    sigma    [0 1 0 0 0 0 0]     3.758e-10;
```

```
32      epsilon epsilon [0 0 0 1 0 0 0]      148.6;
33 }
```

CHAPTER 8

Validation and Results in OpenFOAM

The results and models implemented in Octave serve as a validation case for OpenFOAM. However, the calculations in the latter consider flow and temperature change, and therefore, the results of these two implementations will diverge with increasing time. Here, the validation case and two more complex cases are shown.

8.1. Validation

In order to test the implementation, a simple three-cell geometry with a total of five adsorbing faces is created. This allows for easier debugging, since all values can be printed to standard output without clogging the log. Furthermore, taking a look at the output files in the time directory is also feasible, since there are only five adsorbing faces in total.

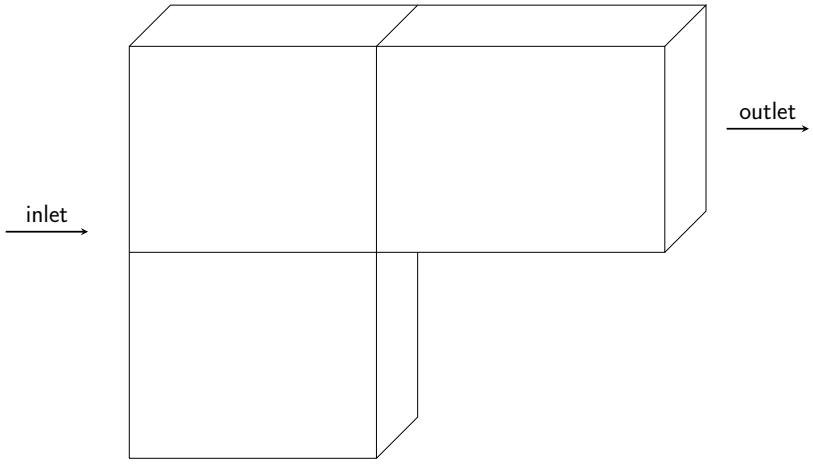


Figure 8.1.: Test and validation case with three cells.

Figure 8.1 shows the geometry of the test case. The three faces at the bottom are adsorbing and belong to the patch `adsDown`, the two faces at the top are also adsorbing and belong to the patch `adsUp`. The cell at the bottom borders two adsorbing faces with different size, and the cell near the outlet has two adsorbing faces which belong to different patches. Those are two special circumstances which were debugged and validated thoroughly.

With this geometry, many test cases showed bugs and incorrect behaviour, and new features like per-patch definition of adsorption parameters were implemented. As seen in listing 8.1, the adsorbed amount can easily be monitored with only five faces. In line 37 and 42, the relevant information can be read. This output was then compared to the results obtained with the implementation in Octave.

Listing 8.1: Adsorbed amount of CH_4 after 0.2 s.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       volScalarField;
13     location    "0.2";
14     object      adsorption_CH4;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 dimensions      [1 -2 0 0 0 0 0];
19
20 internalField   uniform 0;
21
22 boundaryField
23 {
24     inlet
25     {
26         type            calculated;
27         value           uniform 0;
28     }
29     outlet
30     {
31         type            calculated;
32         value           uniform 0;
33     }
34     wall
35     {
36         type            calculated;
37         value           nonuniform List<scalar> 2(5.07276801924e-06
       ↪ 5.07095057904e-06);
38     }
39     adsorptwall
40     {
41         type            calculated;
42         value           nonuniform List<scalar> 3(5.07216439916e-06
       ↪ 5.07216425826e-06 5.07095057904e-06);
43     }
44     frontAndBack
45     {
46         type            calculated;
47         value           uniform 0;
48     }
49 }
50
51
52 // ************************************************************************* //
```

Furthermore, the parameters from [Ritter and Yang, 1987] are adapted, since they are volume- and mole-based, respectively, and OpenFOAM uses mass-based values.

8.2. Test Cases

Here, the results of two test cases are shown. The first, a simple cuboid, demonstrates the capability of CFD coupled with adsorption. Also, the computational effort of the different models is compared and the dynamic behaviour is outlined. The second case, a packed bed, shows the possibility to calculate breakthrough curves and temperature distribution in a more complex geometry.

8.2.1. Cuboid

A square tunnel with 10 cm side length and a total length of 50 cm is used to show a gradient in concentration. In total, the grid consists of 5000 cells. Inlet and outlet have a fixed velocity with a magnitude of $5 \times 10^{-3}\,\mathrm{m\,s^{-1}}$. This yields a Reynolds number of less than 100, which satisfies the condition for laminar flow. The input velocity is $5 \times 10^{-3}\,\mathrm{m\,s^{-1}}$ and the mass fractions of the gas phase at the inlet are $w_{\mathrm{CH_4}} = 0.3$, $w_{\mathrm{CO_2}} = 0.3$ and $w_{\mathrm{CO}} = 0.4$, respectively. The adsorption parameters are shown in listing B.7 in appendix B, as well as all other configuration files for setting up this case.

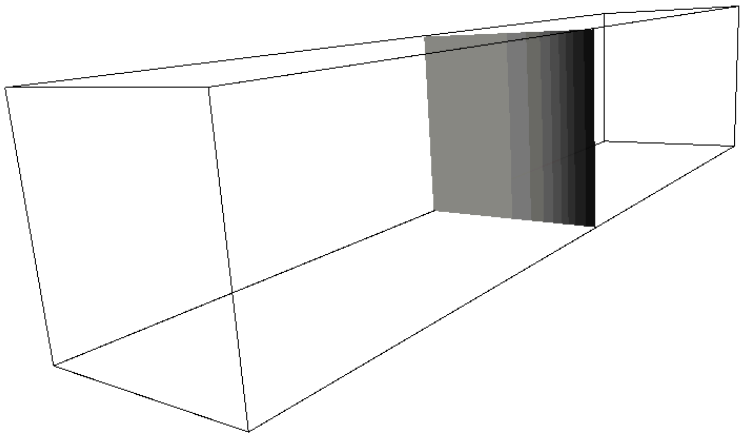


Figure 8.2.: Outline of the square tunnel with the concentration profile of CO.

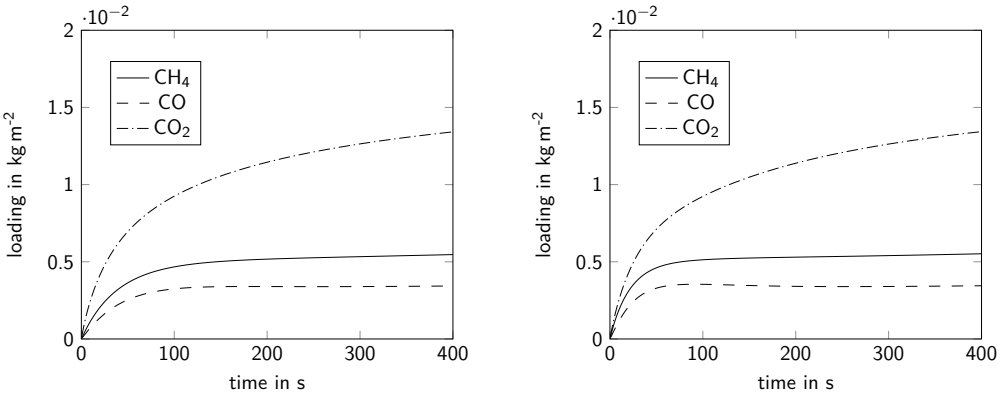
Computational Effort

As expected, the IAST and diffusion-based kinetics model are computational more demanding than the ELM and linear driving force kinetics model, respectively. The following executing times are measured when running the above case for 400 s of simulated time:

- ELM with linear driving force model: 248 s.
- ELM with diffusion-based model: 262 s.
- IAST with linear driving force model: 270 s.
- IAST with diffusion-based model: 285 s.

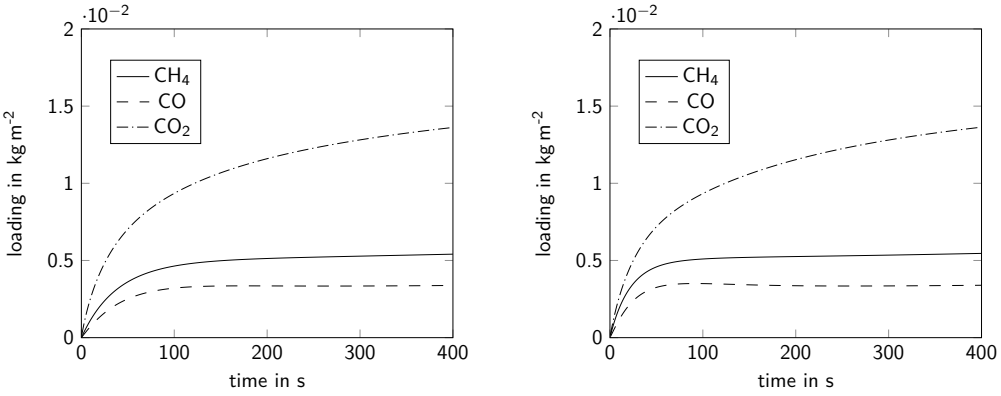
Dynamic Behaviour

The dynamic behaviour of the linear driving force model and the diffusion-based model differ. The latter shows a slight overshoot, as reported before, for CO as seen in figures 8.3(b) and 8.4(b). The data used to plot figures 8.3 and 8.4 are patch-averaged.



(a) Linear driving force kinetics model. (b) Diffusion-based kinetics model.

Figure 8.3.: Dynamic behaviour of the square tunnel simulation with ELM.



(a) Linear driving force kinetics model. (b) Diffusion-based kinetics model.

Figure 8.4.: Dynamic behaviour of the square tunnel simulation with IAST.

The other species are adsorbed rather similar, and the differences between ELM and IAST are negligible. Since OpenFOAM considers energy transfer, the temperature of the adsorbing faces is changing, and therefore, the equilibrium loading changes as well.

8.2.2. Packed Bed

A cylindrical packed bed with a height of 13 cm and a diameter of 3.2 cm is simulated as shown in figure 8.5. The mesh was provided by the research group [Haddadi et al., 2016], and not created by the author. It is an unstructured grid with about 1.3×10^6 cells. The geometry is shown in figure 8.5. The packing has a total adsorbing surface of $6.34 \times 10^{-2}\,\mathrm{m}^2$.

Figure 8.5.: Geometry of the packed bed.

The simulation was done in two steps: First, a steady-state solution with one component and without adsorption was obtained. Then, the adsorption was simulated. A velocity of $0.1\,\mathrm{m\,s^{-1}}$ in the positive z direction was set as boundary condition at the inlet. All species are set to be adsorbing on the packing. The pressure at the outlet was set to 100 kPa. Table 8.1 provides a comprehensive list of the used boundary conditions for the adsorption simulation.

Steady-state Solution

The program `rhoSimpleFoam` was used to obtain a steady-state pressure and velocity field. According to the OpenFOAM user guide, this solver is for steady-state, laminar or turbulent RANS flow with one single species [Greenshields, 2015b]. The convergence criteria for the residuals were set at 10^{-6} for pressure and velocity, respectively, and 10^{-5} for the enthalpy. Figure 8.6 shows the pressure and velocity magnitude field, respectively. As illustrated, the pressure drop over the column is about 11 Pa, and the maximum velocity reported is as $1.85\,\mathrm{m\,s^{-1}}$. The temperature profile did not change and therefore, it was uniform at 300 K.

Adsorption Simulation

The steady-state fields for pressure and velocity were used as initial conditions for the next part of the simulation. In this step, adsorption was simulated using the solver `generalMultiAdsorpFoam`. For this, a mixture of hydrogen, methane, carbon monoxide and carbon dioxide is set at the inlet. The mass fraction of hydrogen is 0.1 corresponding to a mole fraction of 0.58, whereas all other components have a mass fraction of 0.3 at the

Table 8.1.: Boundary conditions for the packed bed simulation.

property name	patch name			
	inlet	outlet	walls	packing
CH4	fixedValue: 0.3	zeroGradient	zeroGradient	adsorpWall
CO	fixedValue: 0.3	zeroGradient	zeroGradient	adsorpWall
CO2	fixedValue: 0.3	zeroGradient	zeroGradient	adsorpWall
H2	fixedValue: 0.1	zeroGradient	zeroGradient	adsorpWall
p	zeroGradient	fixedValue: 1e5	zeroGradient	zeroGradient
T	fixedValue: 300	zeroGradient	zeroGradient	adsorpAdiabaticWall: 300
U	fixedValue: (0 0 0.1)	zeroGradient	fixedValue: (0 0 0)	fixedValue: (0 0 0)

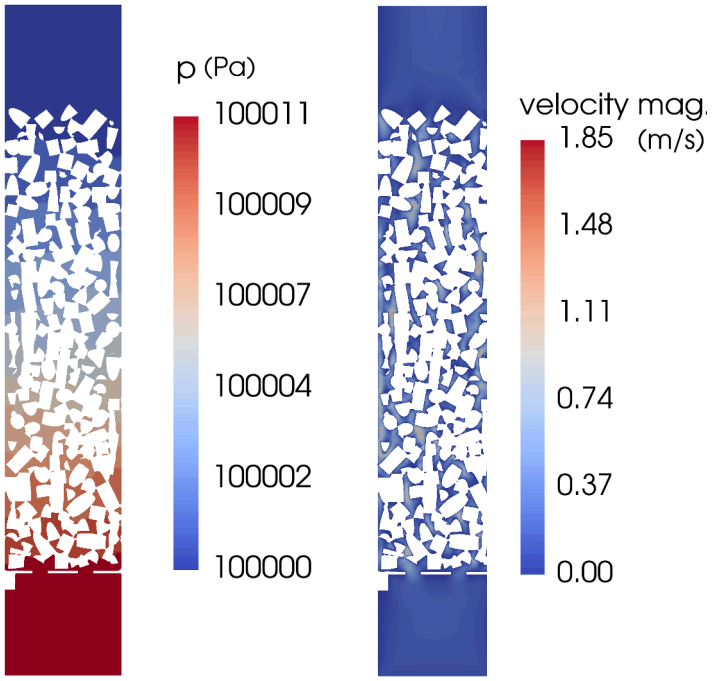
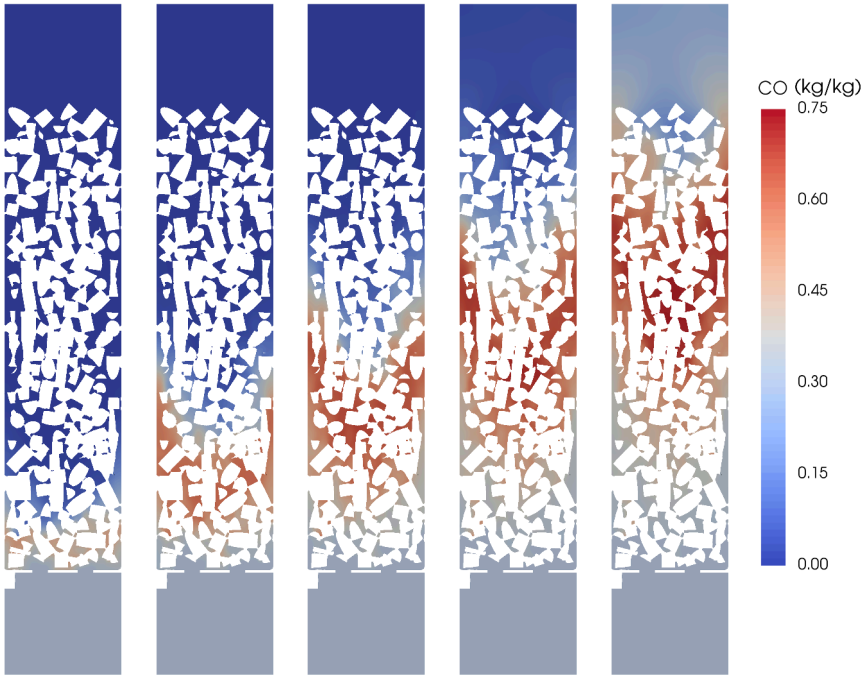


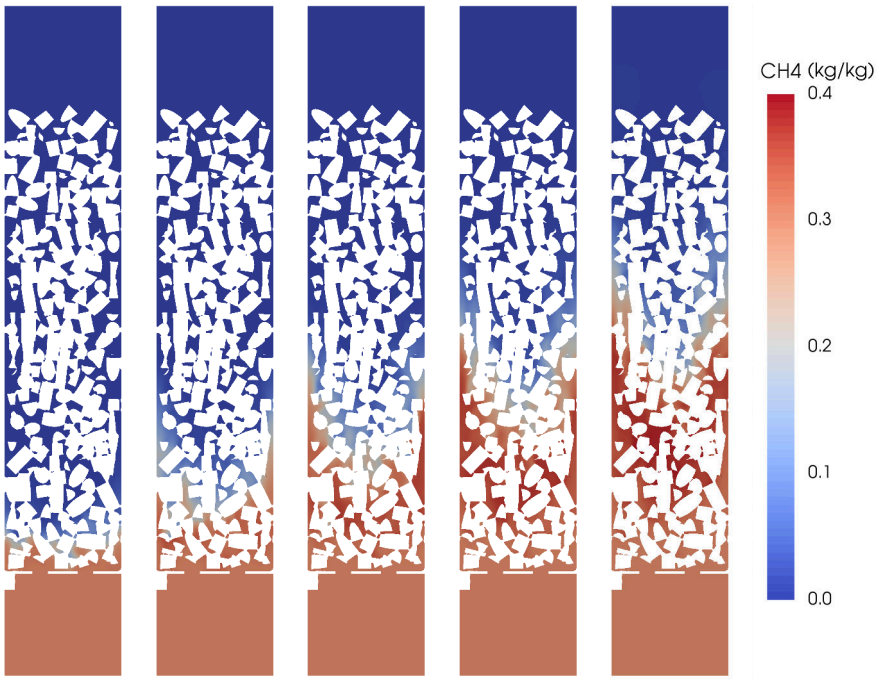
Figure 8.6.: Results for (a) pressure and (b) velocity magnitude of the steady-state calculation.

inlet. This yields 0.22, 0.12 and 0.08 for the mole fractions of methane, carbon monoxide and carbone dioxide, respectively. Initially, the column is filled with hydrogen which is assumed to be non-adsorbing. Therefore, all equilibrium parameters for hydrogen are set to zero. The simulated time was 10 s and the simulation took a little over seven days to complete. The simulation was run in parallel with 24 cores used.
The ELM was used for equilibrium calculations with diffusion-based kinetics. The parameter in the kinetics model was chosen arbitrarily as $10^{-5}\,\mathrm{m}^2$, since no experimental data were available. The equilibrium parameters used for the three adsorbing species were taken from [Ritter and Yang, 1987] and can be viewed in listing B.7 in the appendix. As expected, carbon monoxide is the component to break through first, as shown in figure 8.7. This is due to the fact that its equilibrium loading of $2.5 \times 10^{-3}\,\mathrm{kg\,m}^{-2}$ is about 75 % of that of methane, and about 30 % of that of carbon dioxide.
The next component to break through after carbon monoxide is methane. Its equilibrium loading of $3.3 \times 10^{-3}\,\mathrm{kg\,m}^{-2}$ is about 40 % of that of carbon dioxide as illustrated in figure 8.8. Note that the simulated time of 10 s was too short to actually show any breakthrough of methane.
The last component to break through is carbon monoxide, with an equilibrium loading of $8.8 \times 10^{-3}\,\mathrm{kg\,m}^{-2}$. As with methane, the actual breakthrough was not simulated due to the too short simulated time, as shown in figure 8.9.
To show the effect of the breakthrough of one component, the mass fraction distribution of hydrogen is shown in figure 8.10. After 7 s of simulated time, its mass fraction decreases in the outlet section. This indicates that the adsorber is not separating the inlet gas anymore. This indicates that the packed bed has to be recuperated by decreasing the pressure or increasing the temperature.



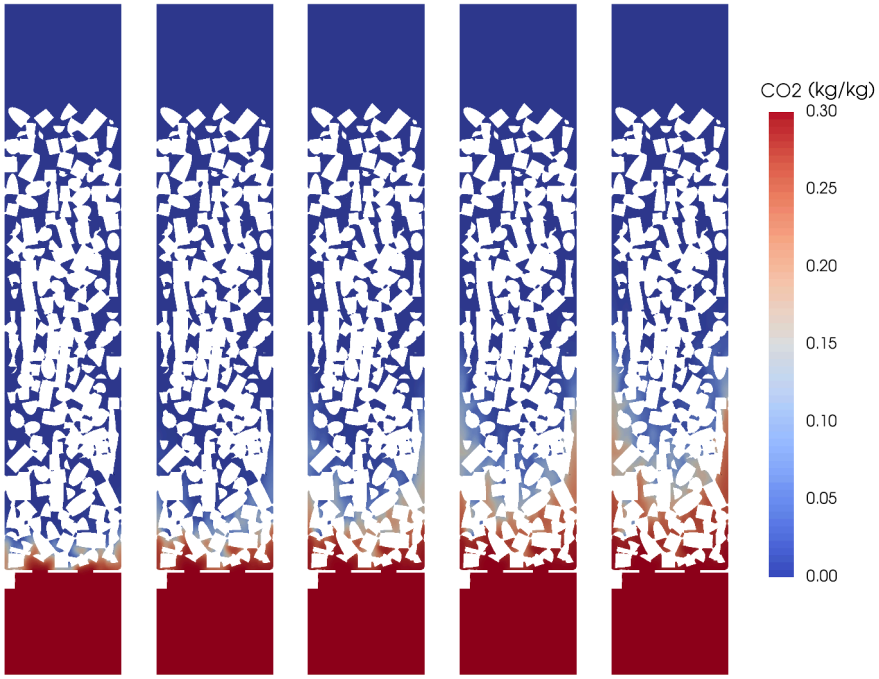
(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.7.: Distribution of the gas mass fraction of carbon monoxide in the packed bed for different simulation times.



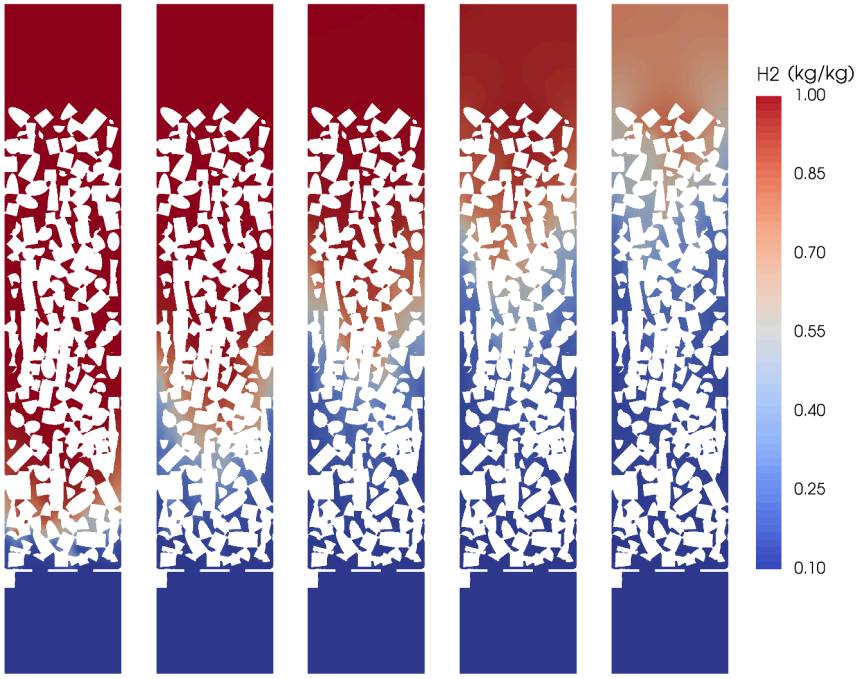
(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.8.: Distribution of the gas mass fraction of methane in the packed bed for different simulation times.



(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

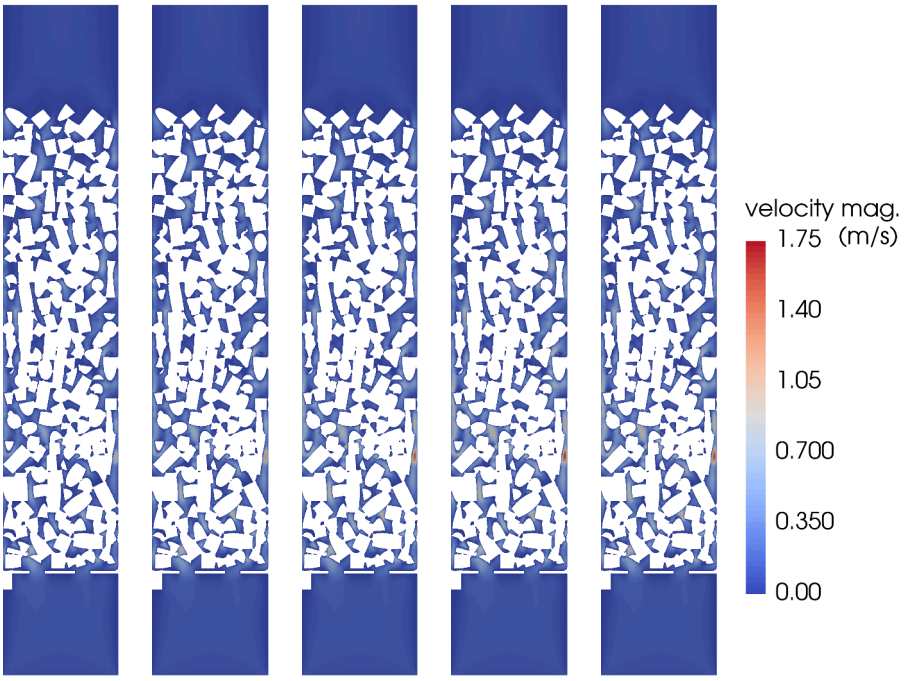
Figure 8.9.: Distribution of the gas mass fraction of carbon dioxide in the packed bed for different simulation times.



(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.10.: Distribution of the gas mass fraction of hydrogen in the packed bed for different simulation times.

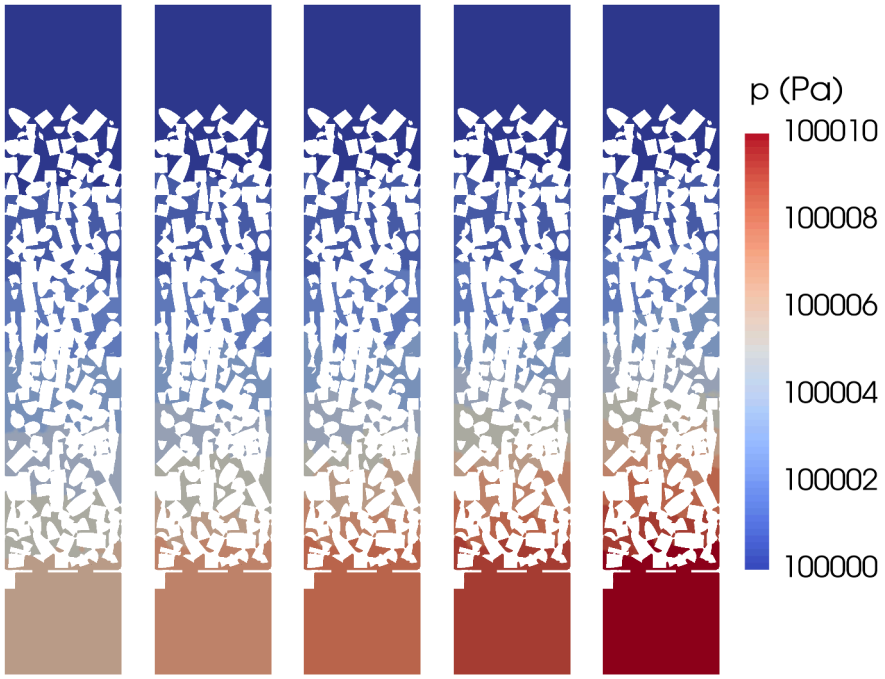
Since adsorption is taking place, the velocity magnitude will be lower than that of the steady-state solution. This effect is illustrated in figure 8.11. As a consequence, the pressure drop decreases as well as shown in figures 8.12 and 8.13. Only integer values are shown in the pressure drop plot. In case of laminar flow, pressure drop and velocity show a linear dependence. With increasing time, the velocity magnitude and pressure drop increased as well. This is due to the fact that no mass is adsorbed anymore in the lower section of the packed bed.



(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.11.: Velocity magnitude profile of the packed bed for different simulation times.

Adsorption is an exothermic process, i.e. heat is released. This also implies that the temperature will increase with time, which is shown in figure 8.14 for the flow and figure 8.15 for the packing, respectively. As an effect of the temperature increase, the equilibrium loading will decrease as shown later.
The loading of carbon monoxide in the packed bed demonstrates the temperature-dependence of the equilibrium. As shown in figure 8.17, the equilibrium loading is lower in the first third of the column. This corresponds to the temperature distribution showed before. Additionally, carbon dioxide and methane are replacing carbon monoxide. The difference between current loading and equilibrium loading is not visible to the eye in figures 8.16 and 8.17. Therefore, only the current loading will be shown for the other two components. As expected, methane and carbon dioxide did not adsorb in the upper part of the packed bed, as shown in figures 8.18 and 8.19.
Figure 8.20 illustrates the adsorbed mass for each species at equilibrium and currently adsorbed. A linear increase of both quantities with time is expected, until the component breaks through. Here, this behaviour is observed: Methane and carbon dioxide show a linear slope, since both did not reach the top of the column yet. Carbon monoxide already broke through before 10 s of simulated time. This explains the decreasing slope. Interestingly, the adsorbed mass per single species does not vary much until carbon



(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.12.: Distribution of the pressure inside the packed bed for different simulation times.

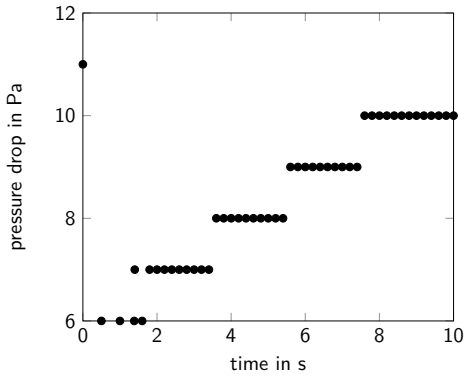
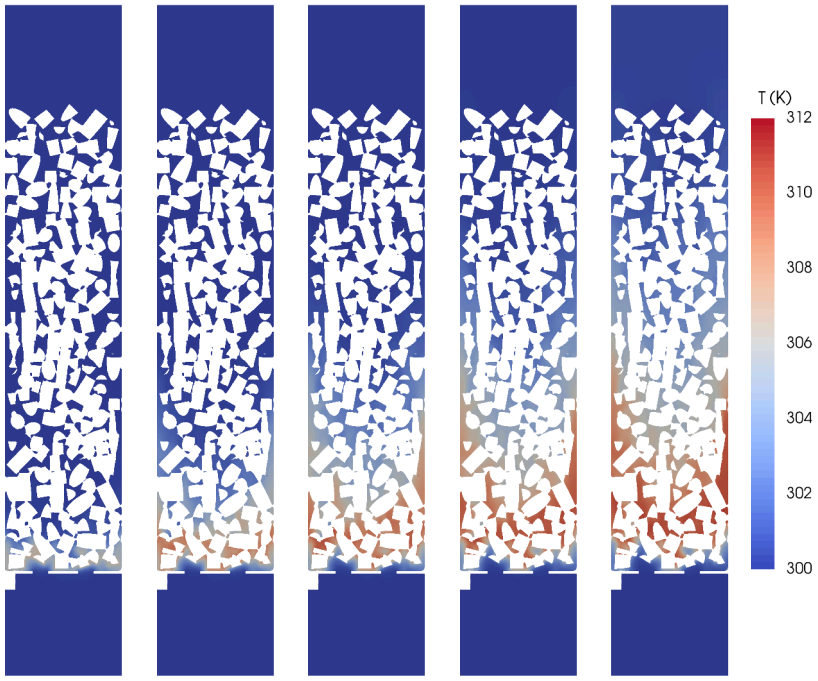
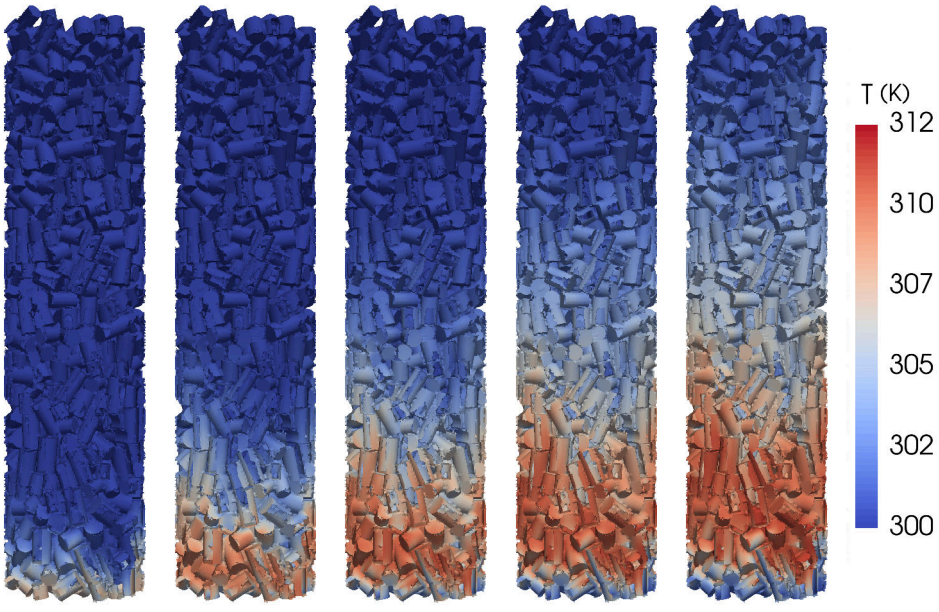


Figure 8.13.: Pressure drop in the packed bed for different simulation times. The pressure drop at 0 s represents the steady-state solution.



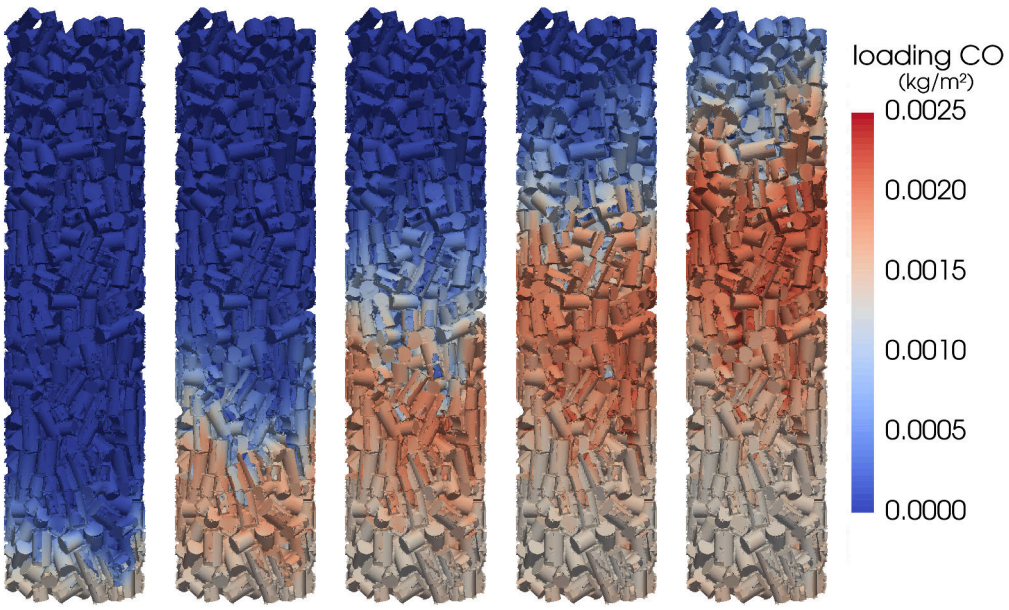
(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.14.: Temperature distribution in the packed bed for different simulation times.



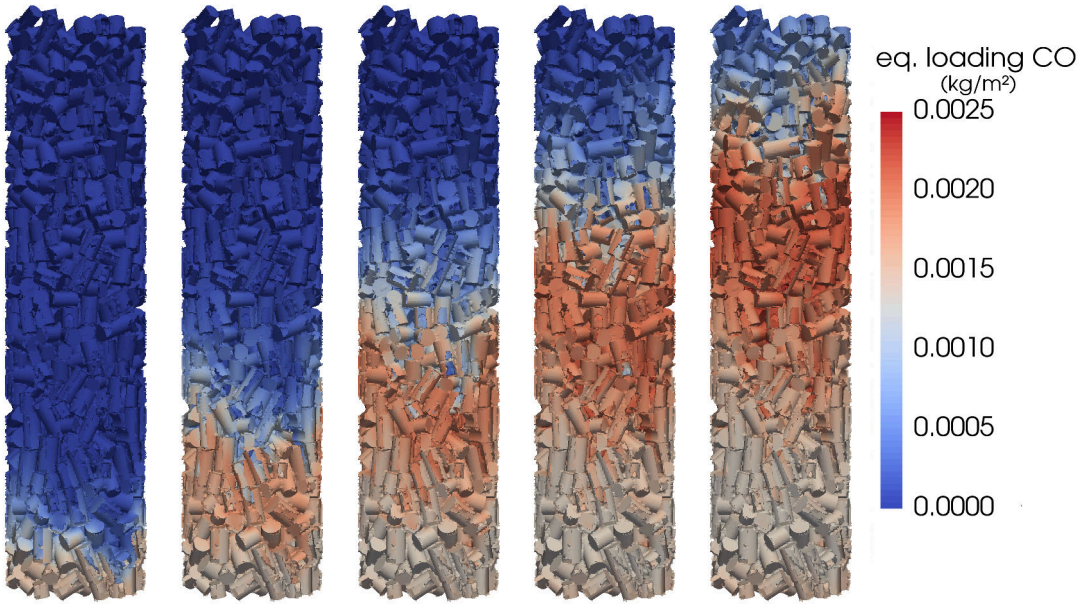
(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.15.: Distribution of the temperature inside the packed bed for different simulation times.



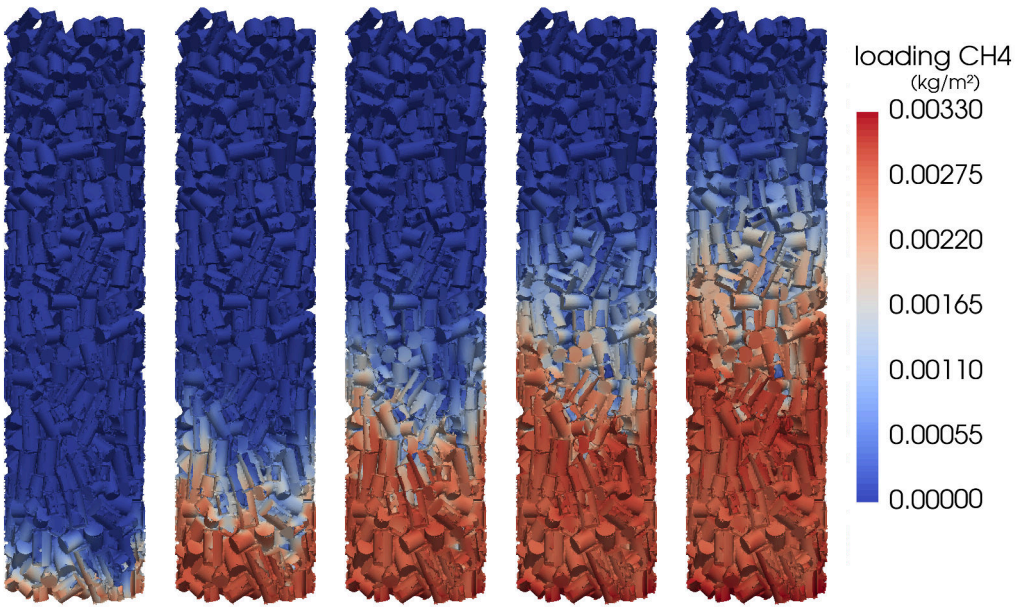
(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.16.: Distribution of the adsorbed amount of carbon monoxide in the packed bed for different simulation times.



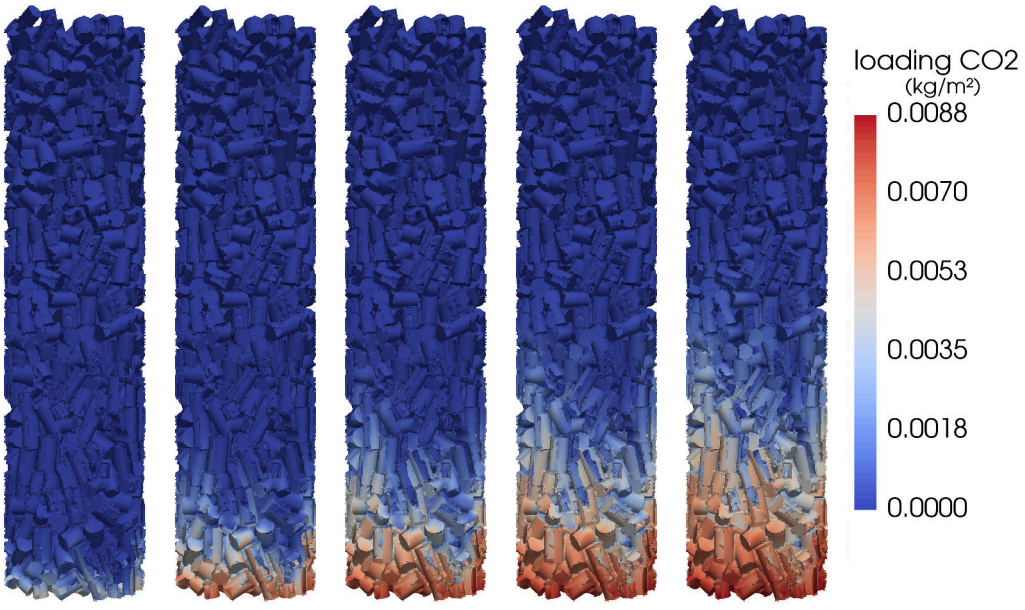
(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.17.: Distribution of the adsorbed amount at equilibrium of carbon monoxide in the packed bed for different simulation times.



(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

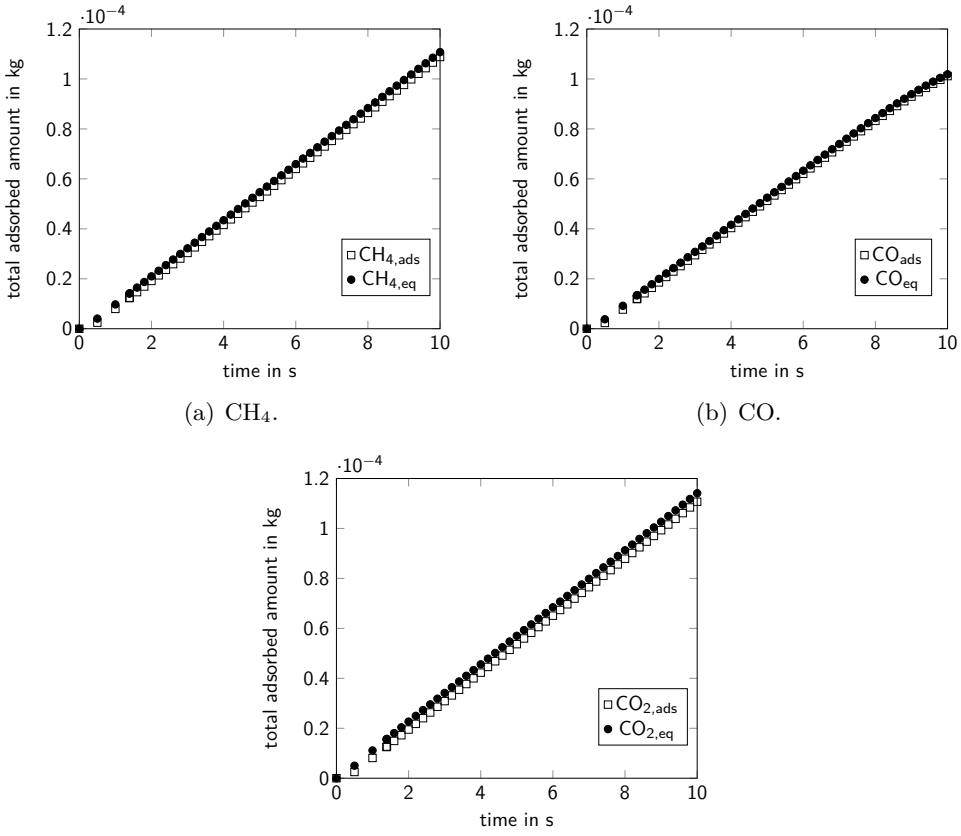
Figure 8.18.: Distribution of the adsorbed amount of methane in the packed bed for different simulation times.



(a) 1 s. (b) 3 s. (c) 5 s. (d) 7 s. (e) 9 s.

Figure 8.19.: Distribution of the adsorbed amount of carbon dioxide in the packed bed for different simulation times.

monoxide breaks through. The very small difference between actual adsorbed amount and amount at equilibrium can be explained by the arbitrarily chosen paramter of the diffusion-based kinetics.
If the simulation was run until the bed is saturated, the total adsorbed mass would be 1.6×10^{-4} kg for carbon monoxide, 2.1×10^{-4} kg for methane and 5.6×10^{-4} kg for carbon dioxide, respectively. This would yield an adsorbed volume of 0.13 Ndm^3 for carbon monoxide, 0.3 Ndm^3 for methane and 0.29 Ndm^3 for carbon dioxide, respectively.



(a) CH_4.

(b) CO.

(c) CO_2.

Figure 8.20.: Adsorbed amount of the three components in the packed bed.

To show the physical correctness of the simulation results, the mass balance errors are regarded. In this case, an unsteady and compressible continuity equation has to be formulated:

$$\dot{m}_{in} = \dot{m}_{out} + \Delta\dot{m} + \dot{m}_{ads}. \tag{8.1}$$

The above equation states that the mass flow inside the control volume $\dot{m}_{in}$ has to be equal to the sum of the mass flow out of the control volume $\dot{m}_{out}$, the change of mass in the control volume due to compressibility effects $\Delta\dot{m}$ and the adsorbed mass flow $\dot{m}_{ads}$. If the regarded time step is low enough, above equation can be integrated by multiplying it with said time step. This yields:

$$m_{in} = m_{out} + \Delta m + m_{ads}. \tag{8.2}$$

The post-processing of the packed bed adsorption simulation showed that the unsteady mass balance cannot be solved manually. This is due to a couple of reasons. The major reason is assumed to be lack of data. The results were written to file every 0.2 s, which proved to be too big to be able to integrate as explained above. Furthermore, no monitors were set at the outlet. This yields an uncertainty of the mass flow out of the column. However, OpenFOAM calculates so-called time step continuity errors, which can be regarded to show that the mass balance is solved correctly. They are relative to the total mass inside the computational domain. In this simulation, they are acceptably low in the order of magnitude of 10^{-10}, as illustrated in figure 8.20.

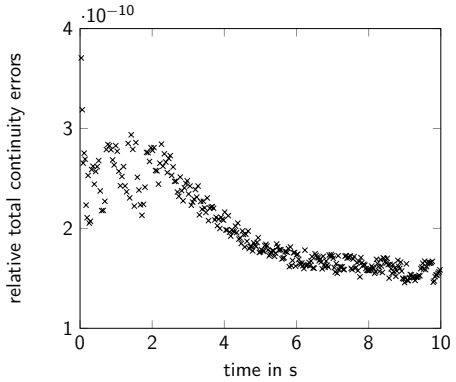


Figure 8.21.: Relative total continuity errors for the packed bed simulation.

CHAPTER 9

Summary, Discussion and Outlook

In this thesis, the implementation of multicomponent adsorption models in the custom OpenFOAM solver `adsorpFoam` in a new solver, called `generalMultiAdsopFoam`, was shown. The used models for equilibrium, extended Langmuir model ELM and ideal adsorbed solution theory, solely depend on single-component adsorption isotherms. First, the models were implemented in Octave, later in OpenFOAM. The deviations are acceptable for most systems, as shown in chapter 4.
The validation of the implementation in OpenFOAM was done by comparing the results of the adsorption calculations of a simple, three cell geometry test case with those of the implementation in Octave.
This is the first implementation of multicomponent adsorption in CFD in the research group. Therefore, it is subject to further improvement. The standard models in OpenFOAM are implemented making use of object-oriented programming paradigm. This was not done in the frame of this thesis, and further abstraction can be done in the future.
Optimisation and reducing the computational effort were not the main focus of the current implementation. Therefore, future releases of `generalMultiAdsopFoam` will most likely perform better and take less memory.
In this thesis, the adsorbed mass is divided by area before written to file. This made it easier to compare results with the Octave model and integrate the loading over the patch area. However, this approach might not be useful in the future. Therefore, this approach might be changed.
Another area for improvement is the choice of time step. OpenFOAM offers the capability to dynamically adapt the time step by setting a maximum allowed Courant number. Additionally, a reference time for adsorption can be defined using the adsorption limiters. In case of a positive limiter, an equivalent Courant number of adsorption for each single species could be defined. It is the fraction of (theoretically) adsorbed mass of said species divided by available mass of said species in all cells bordering an adsorbing patch. As with the traditional Courant number, each cell bordering an adsorbing face would have its own number. Additionally, all species would have their own adsorption Courant number. An adsorption Courant number of zero would mean that no adsorption is taking place. If the adsorption Courant number is close to unity, almost all mass in the cell will be adsorbed, and if it is over one, a positive limiter has to be applied. Similar considerations are valid for desorption. This would allow for dynamic adaptation of the time step.
Currently, the surface of the adsorbing patches is modelled as flat and without pores.

Furthermore, heat conduction in the solid particles of a packed bed is not considered at this point. This leaves room for further improvement.
Some assumptions of the used models have to be questioned critically: The necessity of the same monomolecular layer capacity for all species in the extended Langmuir model is rarely met. However, this model is the most common used in literature. The main reason for this is its simplicity.
Furhtermore, the binary diffusion coefficients are assumed to be equal. However, if the molecular size differs greatly, this assumption may not be valid anymore. Ideal behaviour assumed in the ideal adsorbed solution theory may be valid at low pressure and for some species.
In all presented models, the formation of multimolecular layers and capillary condensation are not considered.
As the name implies, the ideal adsorbed solution theory assumes ideal behaviour. This is not always the case, and can be extended with activity and fugacity coefficients [Sochard et al., 2010].
The goals aet at the beginning of this thesis are met: There is a first implementation of multicomponent adsorption available to the research group. In the future, this solver will serve as foundation for complex simulations and further mass transport phenomena solvers. As illustrated in chapter 8, the breakthrough curves, mass transfer zone MTZ and length of unused bed LUB can be estimated. The capability of setting multiple, different adsorbing patches with different parameters makes it possible to simulate mixtures of different adsorbents. Breakthrough curves, MTZ and LUB could be calculated and displayed automatically in future releases. Furthermore, channelling effects can be made visible and avoided.
For setting up a simulation, the following procedure has proven to work well: First, a steady-state solution of velocity and pressure field should be calculated. Then, said fields serve as initial condition for the adsorption simulation. Another point worth mentioning is the initial condition for the loading: Either the species which is initially present in the computational domain is assumed to be non-adsorbing, or its initial loading is set to its equilibrium loading. Failing to do so will result in high velocity magnitudes in proximity to the adsorbing patch at the beginning of the simulation and a possibly diverging simulation. Monitoring the outlet mass flow for all species is also recommended, so that the unsteady mass balance can be evaluated manually.
The simulations presented in chapter 8.2.2 rely solely on equilibrium parameters from literature and an arbitrarily chosen parameter for the diffusion-based kinetics model. For future simulations, data for the used adsorbents should be available, and the parameter in the diffusion-based kinetics can be estimated by conducting experiments.

Bibliography

[Aspen, 2016] Aspen (2016). Swing Adsorption Modeling – Aspen Adsorption. `http://www.aspentech.com/products/engineering/aspen-adsorption/`. Accessed on 15th February 2016.

[Bai and Yang, 2001] Bai, R. and Yang, R. T. (2001). A thermodynamically consistent langmuir model for mixed gas adsorption. *Journal of Colloid and Interface Science*, 239(2):296 – 302.

[CNR, 2016] CNR (2016). Finite Volume Methods (using vertex reconstructions and DDFV). `http://arturo.imati.cnr.it/~marco/Research/Finite_Volumes/index.html`. Accessed on 8th February 2016.

[Do, 1998] Do, D. D. (1998). *Adsorption Analysis: Equilibria and Kinetics*, volume 2. World Scientific.

[Exeter, 2016] Exeter (2016). CFD solution algorithms. `http://projects.exeter.ac.uk/fluidflow/ComputationalFluidDynamics/notes3web/notes3se1.html`. Accessed on 5th February 2016.

[Ferzinger and Perić, 2002] Ferzinger, J. H. and Perić, M. (2002). *Computational Methods for Fluid Dynamics*. Springer-Verlag, third edition.

[Fletcher, 1991] Fletcher, C. (1991). *Computational Techniques for Fluid Dynamics 1. Fundamental and General Techniques*. Springer series in computational physics. Springer-Verlag, second edition.

[Greenshields, 2015a] Greenshields, C. J. (2015a). *OpenFOAM. The Open Source CFD Toolbox: Programmer's Guide*. CFD Direct Ltd.

[Greenshields, 2015b] Greenshields, C. J. (2015b). *OpenFOAM. The Open Source CFD Toolbox: User Guide*. CFD Direct Ltd.

[Haddadi et al., 2015a] Haddadi, B., Jordan, C., and Harasek, M. (2015a). Numerical simulation of adsorption phenomena using multi-region approach. In *VSS VIENNA young SCIENTISTS SYMPOSIUM*, pages 26–27. Vortrag: Vienna Young Scientists Symposium 2015, Wien; 2015-06-25.

[Haddadi et al., 2015b] Haddadi, B., Jordan, C., and Harasek, M. (2015b). Numerische Simulation des Konzentrations- und Strömungsprofiles in einem Festbettadsorber. *Chemie Ingenieur Technik*, 87(8):1040.

[Haddadi et al., 2016] Haddadi, B., Jordan, C., and Harasek, M. (2016). Mesh of a Packed Bed Adsorber. Not published yet. Internal project.

[Haddadi et al., 2014] Haddadi, B., Martinetz, M., Jordan, C., and Harasek, M. (2014). Numerical simulation of adsorption phenomena. In *Proceedings*, pages 78–81. Vortrag: 10. Minisymposium Verfahrenstechnik, Wien; 2014-06-17 – 2014-06-18.

[Holzinger, 2014] Holzinger, G. (2014). *OpenFOAM. A little User-Manual*. CD-Laboratory – Particulate Flow Modelling.

[IITM, 2016] IITM (2016). Parabolic partial differential equations. `https://mat.iitm.ac.in/home/sryedida/public_html/caimna/pde/forth/forth.html`. Accessed on 8th March 2016.

[Jasak, 1996] Jasak, H. (1996). *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, University of London.

[Kuhlmann, 2007] Kuhlmann, H. C. (2007). *Strömungsmechanik*. Pearson Studium.

[Langmuir, 1918] Langmuir, I. (1918). The adsorption of gases on plane surfaces of glass, mica and platinum. *Journal of the American Chemical Society*, 40(9):1361–1403.

[Myers and Prausnitz, 1965] Myers, A. L. and Prausnitz, J. M. (1965). Thermodynamics of mixed-gas adsorption. *AIChE Journal*, 11(1):121–127.

[OpenFOAM, 2016a] OpenFOAM (2016a). OpenFOAM – Open Source CFD. `http://www.openfoam.org/archive/2.4.0/download/`. Accessed on 7th February 2016.

[OpenFOAM, 2016b] OpenFOAM (2016b). Openfoam-2.4.x/label.h at master. `https://github.com/OpenFOAM/OpenFOAM-2.4.x/blob/master/src/OpenFOAM/primitives/ints/label/label.H/`. Accessed on 8th February 2016.

[OSI, 2016] OSI (2016). Open Source Initiative: The MIT License. `https://opensource.org/licenses/MIT`. Accessed on 9th February 2016.

[Patankar, 1980] Patankar, S. V. (1980). *Numerical Heat Transfer and Fluid Flow*. Series in Computational Methods in Mechanics and Thermal Science. Hemisphere Publishing Corporation.

[Peiró and Sherwin, 2005] Peiró, J. and Sherwin, S. (2005). Finite difference, finite element and finite volume methods for partial differential equations. In *Handbook of materials modeling*, pages 2415–2446. Springer-Verlag.

[Reid et al., 1987] Reid, R. C., Prausnitz, J. M., and Poling, B. E. (1987). *The properties of gases and liquids*. McGraw Hill Book Co., New York, fourth edition.

[Ritter, 1985] Ritter, J. A. (1985). Investiagtion on the adsorption of methane, carbon monoxide, carbon dioxide, hydrogen, hydrogen sulfide and their mixtures on activated carbon. Master's thesis, State University of New York and Buffalo.

[Ritter and Yang, 1987] Ritter, J. A. and Yang, R. T. (1987). Equilibrium Adsorption of Multicomponent Gas Mixtures at Elevated Pressures. *Industrial & Engineering Chemistry Research*, 26(8):1679–1686.

[Ruthven, 1984] Ruthven, D. M. (1984). *Principles of adsorption and adsorption processes*. John Wiley & Sons.

[Schay, 1956] Schay, G. (1956). Theorie de l'adsorption physique des gaz du type Langmuir. *Chim. Phys. Hungary*, 53:691.

[Sing, 1985] Sing, K. S. (1985). Reporting physisorption data for gas/solid systems with special reference to the determination of surface area and porosity (recommendations 1984). *Pure and applied chemistry*, 57(4):603–619.

[Sircar and Hufton, 2000] Sircar, S. and Hufton, J. R. (2000). Why does the linear driving force model for adsorption kinetics work? *Adsorption*, 6(2):137–147.

[Sochard et al., 2010] Sochard, S., Fernandes, N., and Reneaume, J.-M. (2010). Modeling of adsorption isotherm of a binary mixture with real adsorbed solution theory and nonrandom two-liquid model. *AIChE Journal*, 56(12):3109–3119.

[Stephan and Mayinger, 1999] Stephan, K. and Mayinger, F. (1999). *Thermodynamik. Band 2: Mehrstoffsysteme und chemische Reaktionen. Grundlagen und technische Anwendungen*, volume 2. Springer-Verlag.

[Strikwerda, 2004] Strikwerda, J. C. (2004). *Finite difference schemes and partial differential equations*. Siam.

A. Octave

A.1. Implementation Code

Here, the code used for comparing experimental results with simulation calculation is shown. The file `get_data_CH4_CO` is one example of the `get_data_*` files.

Listing A.1: File `get_equilibrium_fast.m`.

```
%% Copyright (c) 2015, 2016 C. Goessnitzer <e1126267@student.tuwien.ac.at>
%
% Permission to use, copy, modify, and distribute this software for any
% purpose with or without fee is hereby granted, provided that the above
% copyright notice and this permission notice appear in all copies.
%
% THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
% WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
% ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
% WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
% ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
% OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

function C = get_equilibrium_fast (p, y, eq_model, params)
  b = params.b;
  Cm = params.Cm;

  switch (eq_model)
  case "elm"
    C = Cm .* b .* y * p / (1 + sum (b .* y * p));
  case "iast"
    max_iter = params.max_iter;
    tolerance = params.tolerance;
    relax = params.relax;
    iast_single = params.iast_single;
    disp_iter = params.disp_iter;

    counter = 0;
    correct = 1;

    switch (iast_single)
    case "langmuir"
      zEst = sum (Cm) / length (Cm) * log (1 + sum (b .* y * p));
      p0 = (exp (zEst ./ Cm) .- 1) ./ b;
      do
        if (correct < 10 && correct > 0.1)
          zEst *= relax * (correct - 1) + 1;
        else
          if (correct > 1)
            zEst *= 8;
          else
            zEst /= 8;
          end
        end

```

```
47          p0 = (exp (zEst ./ Cm) .- 1) ./ b;
48          x = p * y ./ p0;
49          correct = sum (x);
50
51          if (++counter >= max_iter)
52            warning ([ "no convergence after ", num2str(counter), ...
53              " iterations: ", num2str(abs (sum (x) - 1))]);
54            break;
55          end
56        until (abs (correct - 1) < tolerance)
57
58        if disp_iter
59          disp (["number of iterations: ", num2str(counter)]);
60        end
61      otherwise
62        error (["undefined iast_single model: ", num2str(eq_model)]);
63      end
64
65      C0 =  Cm .* b .* p0 ./ (1 .+ b .* p0);
66      Ct = 1 / sum (x ./ C0);
67      C = Ct .* x;
68    otherwise
69      error (["undefined eq_model: ", num2str(eq_model)]);
70    end
71  end
```

Listing A.2: File `get_pressure_fast.m`.

```
1  %% Copyright (c) 2015, 2016 C. Goessnitzer <e1126267@student.tuwien.ac.at>
2  %
3  % Permission to use, copy, modify, and distribute this software for any
4  % purpose with or without fee is hereby granted, provided that the above
5  % copyright notice and this permission notice appear in all copies.
6  %
7  % THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
8  % WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
9  % MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
10 % ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
11 % WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
12 % ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
13 % OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
14
15 function p = get_pressure_fast (Vt, x, Vm, b, eq_model, params)
16   switch (eq_model)
17   case "elm"
18     p = Vt * x ./ (b .* Vm * (1 - sum (Vt .* x ./ Vm)));
19   case "iast"
20     max_iter = params.max_iter;
21     tolerance = params.tolerance;
22     relax = params.relax;
23     iast_single = params.iast_single;
24     disp_iter = params.disp_iter;
25
26     z_est = 1;
27     correct = 1;
28     counter = 0;
29
30     switch (iast_single)
31     case "langmuir"
32       do
33         z_est *= correct;
34         p0_est = 1 ./ b .* (exp (z_est ./ Vm) .- 1);
35         V0_est = Vm .* b .* p0_est ./ (1 .+ b .* p0_est);
36         inv_Vt_est = sum (x ./ V0_est);
37         correct = Vt * inv_Vt_est;
38
39         if (++counter >= max_iter)
40           warning ([ "no convergence after ", num2str(counter), ...
41             " iterations: ", num2str(abs(Vt * inv_Vt_est - 1))]);
```

```
42            break;
43          end
44        until (abs (Vt * inv_Vt_est - 1) < tolerance)
45
46        if disp_iter
47          disp (["number of iterations: ", num2str(counter)]);
48        end
49      end
50
51      p = p0_est .* x;
52    end
53  end
```

Listing A.3: File general-equilibrium.m.

```
1  %% Copyright (c) 2015, 2016 C. Goessnitzer <e1126267@student.tuwien.ac.at>
2  %
3  % Permission to use, copy, modify, and distribute this software for any
4  % purpose with or without fee is hereby granted, provided that the above
5  % copyright notice and this permission notice appear in all copies.
6  %
7  % THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
8  % WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
9  % MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
10 % ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
11 % WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
12 % ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
13 % OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE
14
15 %clc;
16 clear all;
17 close all;
18
19 print_stdio = 0;
20 plotting = 0;
21 params.paper = 1;
22
23 psi = 1;
24 atm = 0;
25
26 psiToPa = 6894.76;
27 atmToBar = 1.01325;
28
29 % Equilibrium Adsorption of Multicomponent Gas Mixtures at Elevated Pressures
30 %
31 % temperature range: 290-298 K
32 % pressure range:    7-28 bar
33
34 % CH4/CO/H2
35  [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CH4_CO_H2();
36
37 % CH4/CO
38 % [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CH4_CO();
39
40 % CH4/CO2/CO/H2/H2S
41 % [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CH4_CO2_CO_H2_H2S();
42
43 % CH4/CH2/CO/H2S
44 % [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CH4_CO2_CO_H2S();
45
46 % CH4/CO2/H2
47 % [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CH4_CO2_H2();
48
49 % CH4/CO2
50 % [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CH4_CO2();
51
52 % CH4/H2/H2S
53 % [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CH4_H2_H2S();
54
```

```
% CO/H2
% [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CO_H2();

% CO2/CO
% [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CO2_CO();

% CO2/H2S
% [ names, p, T, y, xMes, CmMes, etaAvgPaper ] = get_data_CO2_H2S();


p *= psiToPa; % Pa
pBar = 1e-5*p;
params = struct ();

params.iast_single = "langmuir";
params.max_iter = 500;
params.tolerance = 1e-10;
params.relax = 0.5;
params.disp_iter = false;
params.paper = true;

C_mes = CmMes .* xMes;

for i = 1:length(p)
  for j = 1:length(names)
    params.b(j) = get_b (names{j}, T(i)); % 1 / Pa
    params.Cm(j) = get_Vm (names{j}, T(i)); % cc(STP) / g
  end

  params.b = params.b;
  params.Cm = params.Cm;
%  pBar = p(i)*1e-5;
%  TCurr = T(i);

  eq_model = "elm";
  C_elm(:,end+1) = get_equilibrium_fast (p(i), y(:, i)', eq_model, params);
  x_elm(:,end+1) = C_elm(:,end) / sum (C_elm(:,end));

  eq_model = "iast";
  C_iast(:,end+1) = get_equilibrium_fast (p(i), y(:, i)', eq_model, params);
  x_iast(:,end+1) = C_iast(:,end) / sum (C_iast(:,end));

  summe(end+1) = (1 - sum (C_mes(:,i) ./ params.Cm'));
  eta(:,end+1) = (1 - sum (C_mes(:,i) ./ params.Cm')) * p(i) * y(:,i) .*
    ↪ params.b' .* params.Cm' ./ C_mes(:,i);
end

names
C_mes_ = C_mes';
C_elm_ = C_elm';
C_iast_ = C_iast';
C__ = [ 1e-5*p', C_mes', C_elm', C_iast' ];
eta;
C_mes;
summe;
etaAvg = sum (eta') / length (eta');

for i = 1:length(p)
  for j = 1:length(names)
    params.b(j) = get_b (names{j}, T(i)); % 1 / Pa
    params.Cm(j) = get_Vm (names{j}, T(i)); % cc(STP) / g
  end

  C_elmiac(:,end+1) = params.Cm .* params.b ./ etaAvg .* y(:,i)' * p(i) / ...
  (1 + sum (params.b ./ etaAvg .* y(:,i)' * p(i)));

  x_elmiac(:,end+1) = C_elmiac(:,end) / sum (C_elmiac(:,end));
end

```

```
123 C_mes_
124 C_elm_
125 C_iast_
126 return
127 %C_elmiac_ = C_elmiac';
128
129 pBar
130 numberOfDataPoints = length (p)
131
132 absErrorX_elm = [ 1e-5*p', 100*(x_elm .- xMes)' ]
133 absErrorX_elmiac = [ 1e-5*p', 100*(x_elmiac .- xMes)' ];
134 absErrorX_iast = [ 1e-5*p', 100*(x_iast .- xMes)' ]
135
136 absErrorV_elm = [ 1e-5*p', (C_elm .- C_mes)' ]
137 absErrorV_elmiac = [ 1e-5*p', (C_elmiac .- C_mes)' ];
138 absErrorV_iast = [ 1e-5*p', (C_iast .- C_mes)' ]
139
140 relErrorX_elm = [ 1e-5*p', (100 * (x_elm .- xMes) ./ xMes)'];
141 relErrorX_elmiac = [ 1e-5*p', (100 * (x_elmiac .- xMes) ./ xMes)'];
142 relErrorX_iast = [ 1e-5*p', (100 * (x_iast .- xMes) ./ xMes)'];
143
144 relErrorV_elm = [ 1e-5*p', (100 * (C_elm .- C_mes) ./ C_mes)'];
145 relErrorV_elmiac = [ 1e-5*p', (100 * (C_elmiac .- C_mes) ./ C_mes)'];
146 relErrorV_iast = [ 1e-5*p', (100 * (C_iast .- C_mes) ./ C_mes)'];
147
148 %return;
149
150 absMeanX_elm = mean (abs (absErrorX_elm(:,2:end)))
151 absStdDevX_elm = std (abs (absErrorX_elm(:,2:end)))
152 absMaxErX_elm = max (absErrorX_elm(:,2:end))
153 absMinErX_elm = min (absErrorX_elm(:,2:end))
154 absMeanV_elm = mean (abs (absErrorV_elm(:,2:end)))
155 absStdDevV_elm = std (abs (absErrorV_elm(:,2:end)))
156 absMinErV_elm = min (absErrorV_elm(:,2:end))
157 absMaxErV_elm = max (absErrorV_elm(:,2:end))
158
159 %absMeanX_elmiac = mean (abs (absErrorX_elmiac(:,2:end)))
160 %absStdDevX_elmiac = std (abs (absErrorX_elmiac(:,2:end)))
161 %absMaxErX_elmiac = max (absErrorX_elmiac(:,2:end))
162 %absMinErX_elmiac = min (absErrorX_elmiac(:,2:end))
163 %absMeanV_elmiac = mean (abs (absErrorV_elmiac(:,2:end)))
164 %absStdDevV_elmiac = std (abs (absErrorV_elmiac(:,2:end)))
165 %absMinErV_elmiac = min (absErrorV_elmiac(:,2:end))
166 %absMaxErV_elmiac = max (absErrorV_elmiac(:,2:end))
167
168 absMeanX_iast = mean (abs (absErrorX_iast(:,2:end)))
169 absStdDevX_iast = std (abs (absErrorX_iast(:,2:end)))
170 absMinErX_iast = min (absErrorX_iast(:,2:end))
171 absMaxErX_iast = max (absErrorX_iast(:,2:end))
172 absMeanV_iast = mean (abs (absErrorV_iast(:,2:end)))
173 absStdDevV_iast = std (abs (absErrorV_iast(:,2:end)))
174 absMinErV_iast = min (absErrorV_iast(:,2:end))
175 absMaxErV_iast = max (absErrorV_iast(:,2:end))
176
177 %relMeanX_elm = mean (abs (relErrorX_elm(:,2:end)))
178 %relStdDevX_elm = std (abs (relErrorX_elm(:,2:end)))
179 %relMeanV_elm = mean (abs (relErrorV_elm(:,2:end)))
180 %relStdDevV_elm = std (abs (relErrorV_elm(:,2:end)))
181
182 %relMeanX_elmiac = mean (abs (relErrorX_elmiac(:,2:end)))
183 %relStdDevX_elmiac = std (abs (relErrorX_elmiac(:,2:end)))
184 %relMeanV_elmiac = mean (abs (relErrorV_elmiac(:,2:end)))
185 %relStdDevV_elmiac = std (abs (relErrorV_elmiac(:,2:end)))
186
187 %relMeanX_iast = mean (abs (relErrorX_iast(:,2:end)))
188 %relStdDevX_iast = std (abs (relErrorX_iast(:,2:end)))
189 %relMeanV_iast = mean (abs (relErrorV_iast(:,2:end)))
190 %relStdDevV_iast = std (abs (relErrorV_iast(:,2:end)))
191
```

```
192 if (print_stdio)
193   names
194   pBar
195   T
196   y
197   if (params.paper)
198     xMes
199   end
200   x_elm
201   x_elmiac
202   x_iast
203   if (params.paper)
204     CmMes
205   end
206   C_elm
207   C_elmiac
208   C_iast
209 end
210
211 if (plotting)
212   number = 1:length(p);
213   for i = 1:length(names)
214     names_legend{end+1} = [ names{i}, ",elm" ];
215     names_legend{end+1} = [ names{i}, ",iast" ];
216   end
217
218   line_x = [ 0, (length(p)+2) ];
219   line_y = zeros(size(line_x));
220
221   figure(1);
222   for i = 1:length(names)
223     plot(number, 100 .*(x_elm(i, :) .- xMes(i, :)) ./ xMes(i, :), "x",
      ↪ "markersize", 10);
224     hold all;
225     plot(number, 100 .*(x_iast(i, :) .- xMes(i, :)) ./ xMes(i, :), "+",
      ↪ "markersize", 13);
226     hold all;
227   end
228
229   plot(line_x, line_y, "k");
230   xlabel("number of measurement");
231   ylabel("relative error in percent");
232
233   axis([ 0 (length(p)+2) ]);
234   title("relative errors of calculated mole fraction");
235   legend(names_legend);
236
237   figure(2);
238   for i = 1:length(names)
239     plot(number, (x_elm(i, :) .- xMes(i, :)), "x", "markersize", 10);
240     hold all;
241     plot(number, (x_iast(i, :) .- xMes(i, :)), "+", "markersize", 13);
242     hold all;
243   end
244
245   plot(line_x, line_y, "k");
246   xlabel("number of measurement");
247   ylabel("absolute error");
248
249   axis([ 0 (length(p)+2)]);
250   title("absolute errors of calculated mole fraction");
251   legend(names_legend);
252
253   figure(3);
254   plot(number, 100 .*(C_elm .- C_mes) ./ C_mes, "x", "markersize", 10);
255   hold all;
256
257   plot(number, 100 .*(C_iast .- C_mes) ./ C_mes, "+", "markersize", 10);
258   hold all;
```

```
259
260   plot(line_x, line_y, "k");
261
262   xlabel("number of measurement");
263   ylabel("relative error in percent");
264   title("relative errors of calculated adsorbed amount");
265   legend("elm", "iast");
266 end
```

Listing A.4: File `get_data_CH4_CO.m`.

```
1  %% Copyright (c) 2015 C. Goessnitzer <e1126267@student.tuwien.ac.at>
2  %
3  % Permission to use, copy, modify, and distribute this software for any
4  % purpose with or without fee is hereby granted, provided that the above
5  % copyright notice and this permission notice appear in all copies.
6  %
7  % THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
8  % WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
9  % MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
10 % ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
11 % WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
12 % ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
13 % OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
14
15 % Reference:
16 % Ritter, J. A. and Yang, R.T.: Equilibrium Adsorption of Multicomponent Gas
17 %                               Mixtures at Elevated Pressures, 1987.
18
19 function [ names, P, T, Y, X_mes, V_t_mes, eta_avg ] = get_data_CH4_CO()
20
21 names = { "CH4-yang87", "CO-yang87" };
22
23 eta_avg = [ 0.77, 0.93 ];
24
25 P    (  end + 1) = 124.9;            % psi
26 T    (  end + 1) = 293;              % K
27 Y    (:,end + 1) = [ 0.506, 0.494 ]; % mol / mol
28 X_mes(:,end + 1) = [ 0.753, 0.247 ];
29 V_t_mes(end + 1) = 75.8;             % cm3(STP) / g
30
31 P    (  end + 1) = 183.0;            % psi
32 T    (  end + 1) = 293;              % K
33 Y    (:,end + 1) = [ 0.644, 0.356 ]; % mol / mol
34 X_mes(:,end + 1) = [ 0.826, 0.174 ];
35 V_t_mes(end + 1) = 90.9;             % cm3(STP) / g
36
37 P    (  end + 1) = 243.1;            % psi
38 T    (  end + 1) = 295;              % K
39 Y    (:,end + 1) = [ 0.758, 0.242 ]; % mol / mol
40 X_mes(:,end + 1) = [ 0.896, 0.104 ];
41 V_t_mes(end + 1) = 102.4;            % cm3(STP) / g
42
43 % problems with calculating eta!!!
44 %P    (  end + 1) = 310.0;            % psi
45 %T    (  end + 1) = 294;              % K
46 %Y    (:,end + 1) = [ 0.802, 0.198 ]; % mol / mol
47 %X_mes(:,end + 1) = [ 0.923, 0.077 ];
48 %V_t_mes(end + 1) = 133.6;            % cm3(STP) / g
49
50 P    (  end + 1) = 364.3;            % psi
51 T    (  end + 1) = 294;              % K
52 Y    (:,end + 1) = [ 0.696, 0.304 ]; % mol / mol
53 X_mes(:,end + 1) = [ 0.867, 0.133 ];
54 V_t_mes(end + 1) = 114.1;            % cm3(STP) / g
55
56 P    (  end + 1) = 391.1;            % psi
57 T    (  end + 1) = 295;              % K
58 Y    (:,end + 1) = [ 0.880, 0.120 ]; % mol / mol
```

```
X_mes(:,end + 1) = [ 0.952, 0.048 ];
V_t_mes(end + 1) = 117.4;            % cm3(STP) / g

end
```

Listing A.5: File `get_Vm.m`.

```
%% Copyright (c) 2015 C. Goessnitzer <e1126267@student.tuwien.ac.at>
%
% Permission to use, copy, modify, and distribute this software for any
% purpose with or without fee is hereby granted, provided that the above
% copyright notice and this permission notice appear in all copies.
%
% THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
% WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
% MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
% ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
% WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
% ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
% OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

% Reference:
% Ritter, J. A. and Yang, R.T.: Equilibrium Adsorption of Multicomponent Gas
%                               Mixtures at Elevated Pressures, 1987.

function Vm = get_Vm (name, temperature)
  R = 8.3144598; % J / mol K
  T = 273.15; % K
  p = 1e5; % Pa

  % Reference:
  % Ritter, J.A. and Yang, R.T.: Equilibrium Adsorption of Multicomponent Gas
  %                               Mixtures at Elevated Pressures, 1987.
  A_CH4 = -0.281; % Vm3(STP) / (g K)
  A_CO  = -0.299;
  A_CO2 = -0.557;
  A_H2  = -0.433;
  A_H2S = -0.599;

  B_CH4 = 216; % Vm3 (STP) / g
  B_CO  = 214;
  B_CO2 = 378;
  B_H2  = 283;
  B_H2S = 420;

  switch (name)
  case "CH4-yang87"
    Vm = B_CH4 + A_CH4 * temperature; % cc(STP) / g
  case "CO-yang87"
    Vm = B_CO  + A_CO  * temperature;
  case "CO2-yang87"
    Vm = B_CO2 + A_CO2 * temperature;
  case "H2-yang87"
    Vm = B_H2  + A_H2  * temperature;
  case "H2S-yang87"
    Vm = B_H2S + A_H2S * temperature;
  end
end
```

Listing A.6: File `get_b.m`.

```
%% Copyright (c) 2015 C. Goessnitzer <e1126267@student.tuwien.ac.at>
%
% Permission to use, copy, modify, and distribute this software for any
% purpose with or without fee is hereby granted, provided that the above
% copyright notice and this permission notice appear in all copies.
%
% THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
% WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
```

```
% MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
% ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
% WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
% ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
% OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

function b_i = get_b (name, temperature)
  psiToPa = 6894.76;
  atmToPa = 101325;

  % Reference:
  % Ritter, J.A. and Yang, R.T.: Equilibrium Adsorption of Multicomponent Gas
  %                              Mixtures at Elevated Pressures, 1987.
  b0_CH4 = 3.81e-5; % 1 / psi
  b0_CO  = 8.42e-5;
  b0_CO2 = 3.73e-5;
  b0_H2  = 1.47e-5;
  b0_H2S = 11.6e-5;

  e0_CH4 = 1730; % K
  e0_CO  = 1266;
  e0_CO2 = 1885;
  e0_H2  = 918.1;
  e0_H2S = 1723;

  switch (name)
    case "CH4-yang87"
      b_i = b0_CH4 * exp(e0_CH4 / temperature) / psiToPa; % 1 / bar
    case "CO-yang87"
      b_i = b0_CO  * exp(e0_CO  / temperature) / psiToPa;
    case "CO2-yang87"
      b_i = b0_CO2 * exp(e0_CO2 / temperature) / psiToPa;
    case "H2-yang87"
      b_i = b0_H2  * exp(e0_H2  / temperature) / psiToPa;
    case "H2S-yang87"
      b_i = b0_H2S * exp(e0_H2S / temperature) / psiToPa;
  end
end
```

A.2. Results of Implementation in Octave

In this section, all results used for plotting in chapter 4. The unit of pressure is bar and the unit of adsorbed amount is standard volume per gram adsorbent ($\mathrm{Ncm^3\,g^{-1}}$).

Table A.1.: Adsorbed amount according to experiment, ELM, IAST and ELMIAC for the system CH_4–CO.

pressure	Measurment		ELM		IAST		ELMIAC	
	CH_4	CO	CH_4	CO	CH_4	CO	CH_4	CO
8.6116	57.0774	18.7226	51.9004	21.7308	52.3087	21.3306	63.6606	22.2597
12.6174	75.0834	15.8166	71.9366	17.0547	72.4617	16.5430	83.5003	16.5320
16.7612	91.7504	10.6496	85.8354	11.8734	86.3341	11.3896	96.2583	11.1196
21.3738	123.3128	10.2872	94.8460	10.0940	95.3661	9.5911	103.9382	9.2377
25.1176	98.9247	15.1753	89.6880	16.8870	90.5423	16.0617	98.6616	15.5135
26.9654	111.7648	5.6352	104.0696	6.1487	104.4564	5.7759	111.6922	5.5110

Table A.2.: Adsorbed amount according to experiment, ELM, IAST and ELMIAC for the system CH_4–CO_2.

pressure	Measurment		ELM		IAST		ELMIAC	
	CH_4	CO_2	CH_4	CO_2	CH_4	CO_2	CH_4	CO_2
8.0738	40.5162	69.2838	49.3255	59.3203	42.1913	68.0870	37.2270	74.3474
9.9422	48.9632	68.7368	55.5371	58.4190	47.1043	68.9601	42.5550	74.3358
12.5002	43.4484	90.6516	54.0766	73.0598	42.8885	87.5295	40.7412	91.4069
15.9338	32.8483	103.4517	50.0308	89.9184	36.2020	108.4224	36.7899	109.8036
16.9611	31.7600	127.0400	46.8698	98.9063	32.2612	118.7263	33.9561	118.9941
22.6286	23.8050	148.6950	40.1784	117.2957	24.2062	139.6678	28.2376	136.8972

Table A.3.: Adsorbed amount according to experiment, ELM, IAST and ELMIAC for the system CO–H_2.

pressure	Measurment		ELM		IAST		ELMIAC	
	CO	H_2	CO	H_2	CO	H_2	CO	H_2
11.3901	31.9548	5.2452	40.4944	2.9907	40.3777	3.1119	35.8445	3.9200
16.1406	42.4880	4.5120	43.3150	4.6431	43.1149	4.8517	38.3767	6.0915
16.4854	45.9118	4.9882	56.8856	3.2819	56.6828	3.4962	51.3758	4.3890
17.9333	39.7460	7.8540	37.7902	6.4155	37.5563	6.6585	33.1434	8.3318

Table A.4.: Adsorbed amount according to experiment, ELM, IAST and ELMIAC for the system CO_2–CO.

pressure	Measurment		ELM		IAST		ELMIAC	
	CO_2	CO	CO_2	CO	CO_2	CO	CO_2	CO
8.0738	40.5162	69.2838	49.3255	59.3203	42.1913	68.0870	37.2270	74.3474
9.9422	48.9632	68.7368	55.5371	58.4190	47.1043	68.9601	42.5550	74.3358
12.5002	43.4484	90.6516	54.0766	73.0598	42.8885	87.5295	40.7412	91.4069
15.9338	32.8483	103.4517	50.0308	89.9184	36.2020	108.4224	36.7899	109.8036
16.9611	31.7600	127.0400	46.8698	98.9063	32.2612	118.7263	33.9561	118.9941
22.6286	23.8050	148.6950	40.1784	117.2957	24.2062	139.6678	28.2376	136.8972

Table A.5.: Adsorbed amount according to experiment, ELM, IAST and ELMIAC for the system CH_4–CO–H_2.

pressure	Measurment			ELM			IAST			ELMIAC		
	CH_4	CO	H_2	CH_4	CO	H_2	CH_4	CO	H_2	CH_4	CO	H_2
10.5972	21.1926	21.9336	6.2738	25.4985	29.6924	1.8758	25.6988	29.4004	1.9754	27.8026	18.4813	4.5638
12.0107	22.9474	23.9316	4.9210	19.8616	34.2303	2.3762	20.0377	33.9360	2.5033	21.9300	21.5750	5.8544
14.5617	30.6747	18.5976	7.4277	25.8656	32.6530	3.0013	26.0786	32.2763	3.1781	28.2448	20.3543	7.3129
16.8370	39.3514	25.1658	3.6828	30.9200	35.7456	2.3967	31.2558	35.2545	2.5695	34.3864	22.6927	5.9475
19.5880	51.8756	14.5924	9.9320	46.1825	28.6793	3.1465	46.5427	28.0839	3.4085	49.6556	17.6026	7.5489
22.3735	46.4889	14.7492	11.0619	47.6046	31.3751	3.2621	48.0388	30.6718	3.5650	51.6503	19.4324	7.8976
23.6559	35.7459	29.9710	7.3831	30.4235	43.1695	3.1723	30.8488	42.4940	3.4522	34.5881	28.0164	8.0476
26.1380	43.4070	32.5962	5.8968	37.4468	42.9620	2.6644	38.0309	42.1415	2.9371	42.7630	28.0063	6.7893

Table A.6.: Adsorbed amount according to experiment, ELM, IAST and ELMIAC for the system CH_4–CO_2–H_2.

pressure	Measurment			ELM			IAST			ELMIAC		
	CH_4	CO_2	H_2	CH_4	CO_2	H_2	CH_4	CO_2	H_2	CH_4	CO_2	H_2
14.8237	30.1270	26.9861	6.9869	29.7499	20.9123	5.9785	28.5046	22.2911	5.9663	25.5155	22.4254	5.1447
20.0500	41.6625	33.6600	7.1775	37.3105	27.2530	6.8191	35.1298	29.7428	6.7943	32.3563	29.5503	5.9335
21.7530	19.0944	38.3724	3.7332	30.5762	51.2520	6.4766	27.1801	55.5045	6.2113	26.1765	54.8603	5.5632
23.1319	38.2885	62.7302	3.8813	44.0491	54.5070	4.5186	38.2432	61.7380	4.3440	38.3497	59.3330	3.9472
24.3454	49.9904	62.9552	3.7376	55.9383	50.4556	3.6466	48.5243	59.7313	3.5673	49.5089	55.8346	3.2382

B. Example Case Setup in OpenFOAM

All necessary files for setting up a case for `generalMultiAdsorpFoam` are shown in this section.

B.1. 0 directory

Listing B.1: File `CH4`.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      CH4;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 0 0 0 0];

internalField   uniform 0.3;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform 0.3;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform 0.3;
        value           uniform 0.3;
    }
    adsorptwall
    {
        type            adsorpWall;
    }
    wall
    {
        type            zeroGradient;
    }
    frontAndBack
    {
```

```
45         type            zeroGradient;
46     }
47 }
48
49 // ************************************************************************* //
```

Listing B.2: File CO2.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       volScalarField;
13     location    "0";
14     object      CO2;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 dimensions      [0 0 0 0 0 0 0];
19
20 internalField   uniform 0.3;
21
22 boundaryField
23 {
24     inlet
25     {
26         type            fixedValue;
27         value           uniform 0.3;
28     }
29     outlet
30     {
31         type            inletOutlet;
32         inletValue      uniform 0.3;
33         value           uniform 0.3;
34     }
35     adsorptwall
36     {
37         type            adsorpWall;
38     }
39     wall
40     {
41         type            zeroGradient;
42     }
43     frontAndBack
44     {
45         type            zeroGradient;
46     }
47 }
48
49 // ************************************************************************* //
```

Listing B.3: File CO.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
```

```
10     version     2.0;
11     format      ascii;
12     class       volScalarField;
13     location    "0";
14     object      CO2;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 dimensions      [0 0 0 0 0 0 0];
19
20 internalField   uniform 0.4;
21
22 boundaryField
23 {
24     inlet
25     {
26         type            fixedValue;
27         value           uniform 0.4;
28     }
29     outlet
30     {
31         type            inletOutlet;
32         inletValue      uniform 0.4;
33         value           uniform 0.4;
34     }
35     adsorptwall
36     {
37         type            adsorpWall;
38     }
39     wall
40     {
41         type            zeroGradient;
42     }
43     frontAndBack
44     {
45         type            zeroGradient;
46     }
47 }
48
49 // ************************************************************************* //
```

Listing B.4: File p.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       volScalarField;
13     location    "0";
14     object      p;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18
19 dimensions      [1 -1 -2 0 0 0 0];
20
21 internalField   uniform 1e5;
22
23 boundaryField
24 {
25     inlet
26     {
```

```
27          type            zeroGradient;
28      }
29      outlet
30      {
31          type            fixedValue;
32          value           uniform 1e5;
33      }
34      adsorptwall
35      {
36          type            zeroGradient;
37      }
38      wall
39      {
40          type            zeroGradient;
41      }
42      frontAndBack
43      {
44          type            zeroGradient;
45      }
46  }
47
48  // ************************************************************************* //
```

Listing B.5: File T.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       volScalarField;
13     location    "0";
14     object      T;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 dimensions      [0 0 0 1 0 0 0];
19
20 internalField   uniform 300;
21
22 boundaryField
23 {
24     inlet
25     {
26         type            fixedValue;
27         value           uniform 300;
28     }
29     outlet
30     {
31         type            inletOutlet;
32         inletValue      uniform 300;
33         value           uniform 300;
34     }
35     adsorptwall
36     {
37         type            adsorpAdiabaticWall;
38         value           uniform 300;
39     }
40     wall
41     {
42         type            zeroGradient;
43     }
44     frontAndBack
```

```
    {
        type            zeroGradient;
    }
}

// ************************************************************************* //
```

Listing B.6: File U.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0.005 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value           uniform (0.005 0 0);
    }
    outlet
    {
        type            zeroGradient;
    }
    adsorptwall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    wall
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
    frontAndBack
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
}

// ************************************************************************* //
```

B.2. constant directory

Listing B.7: File adsorptionProperties.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      adsorptionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//adsorptionType iast-langmuir;
adsorptionType elm;
//adsorptionType henry;

kineticsType   ldf;
//kineticsType   diffusion;

adsorbentCp             adsorbentCp       [0 2 -2 -1 0 0 0]     1e3;
adsorbentDensity    adsorbentDensity    [1 -3 0 0 0 0 0]          1e3;
adsorbentLayerH     adsorbentLayerH   [0 1 0 0 0 0 0]     1e-3;
diffusionDeltaZ     diffusionDeltaZ   [0 1 0 0 0 0 0]     1e-3;

CH4
{
    adsorptwall
    {
        iac     1;

        Cm0     Cm0     [ 1 -2 0 0 0 0 0]   0.15259;
        Cm1     Cm1     [ 1 -2 0 -1 0 0 0]  -1.9851e-4;
        b0      b0      [ -1 1 2 0 0 0 0]   5.5259e-9;
        T0      T0      [ 0 0 0 1 0 0 0]    1730.0;

        K1      K1      [0 0 -1 0 0 0 0]    0.02;
        K2      K2      [-1 2 -1 0 0 0 0]   0.01;
        K3      K3      [-2 4 -1 0 0 0 0]   0.001;

        Ke      Ke      [0 -1 2 0 0 0 0]    3e-8;

        adsorptionDeltaH adsorptionDeltaH [0 2 -2 0 0 0 0] 1e6;
    }

    sigma   sigma   [0 1 0 0 0 0 0]     3.758e-10;
    epsilon epsilon [0 0 0 1 0 0 0]     148.6;
}

CO
{
    adsorptwall
    {
        iac     1;

        Cm0     Cm0     [ 1 -2 0 0 0 0 0]  0.26395;
        Cm1     Cm1     [ 1 -2 0 -1 0 0 0] -3.6878e-4;
        b0      b0      [ -1 1 2 0 0 0 0]  1.2212e-8;
        T0      T0      [ 0 0 0 1 0 0 0]   1266;

        K1      K1      [0 0 -1 0 0 0 0]   0.02;
        K2      K2      [-1 2 -1 0 0 0 0]  0.01;
        K3      K3      [-2 4 -1 0 0 0 0]  0.001;

        Ke      Ke      [0 -1 2 0 0 0 0]   3e-6;

        adsorptionDeltaH  adsorptionDeltaH  [0 2 -2 0 0 0 0]  1e6;
    }
```

```
73
74      sigma    sigma    [0 1 0 0 0 0 0]     3.69e-10;
75      epsilon  epsilon  [0 0 0 1 0 0 0]     91.7;
76  }
77
78  CO2
79  {
80      adsorptwall
81      {
82          iac      1;
83
84          Cm0      Cm0      [ 1 -2 0 0 0 0 0]  0.73254;
85          Cm1      Cm1      [ 1 -2 0 -1 0 0 0] -1.0794e-3;
86          b0       b0       [ -1 1 2 0 0 0 0]  5.4099e-9;
87          T0       T0       [ 0 0 0 1 0 0 0]   1885;
88
89          Ke       Ke       [0 -1 2 0 0 0 0]   3e-6;
90
91          K1       K1       [0 0 -1 0 0 0 0]   0.02;
92          K2       K2       [-1 2 -1 0 0 0 0]  0.01;
93          K3       K3       [-2 4 -1 0 0 0 0]  0.001;
94
95          adsorptionDeltaH  adsorptionDeltaH  [0 2 -2 0 0 0 0]  1e6;
96      }
97
98      sigma    sigma    [0 1 0 0 0 0 0]     3.941e-10;
99      epsilon  epsilon  [0 0 0 1 0 0 0]     195.2;
100 }
101
102 // ************************************************************************* //
```

Listing B.8: File `chemistryProperties`.

```
1  /*---------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "constant";
14     object      chemistryProperties;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 chemistryType
19 {
20     chemistrySolver   ode;
21     chemistryThermo   psi;
22 }
23
24 chemistry           off;
25
26 initialChemicalTimeStep 1e-07;
27
28 sequentialCoeffs
29 {
30     cTauChem        0.001;
31 }
32
33 EulerImplicitCoeffs
34 {
35     cTauChem        0.05;
36     equilibriumRateLimiter off;
```

```
}

odeCoeffs
{
    solver          SIBS;
    eps             0.05;
    scale           1;
}

// ************************************************************************* //
```

Listing B.9: File `combustionProperties`.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      combustionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
combustionModel  laminar<psiChemistryCombustion>;

active  true;

laminarCoeffs
{
}
// ************************************************************************* //
```

Listing B.10: File `g`.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           ( 0 0 0 );


// ************************************************************************* //
```

Listing B.11: File `blockMeshDict`.

```
/*---------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
4 |  \\    /   O peration     | Version:  2.4.0                                 |
5 |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6 |    \\/     M anipulation  |                                                 |
7 \*---------------------------------------------------------------------------*/
8 FoamFile
9 {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     object      blockMeshDict;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
16
17 convertToMeters 0.1;
18
19 vertices
20 (
21     (0 0 0) //0
22     (5 0 0) //1
23     (5 1 0) //2
24     (0 1 0) //3
25     (0 0 1) //4
26     (5 0 1) //5
27     (5 1 1) //6
28     (0 1 1) //7
29 );
30
31 blocks
32 (
33     hex (0 1 2 3 4 5 6 7) (50 10 10) simpleGrading (1 1 1)
34 );
35
36 edges
37 (
38 );
39
40 boundary
41 (
42     inlet
43     {
44         type patch;
45         faces
46         (
47             (0 3 7 4)
48         );
49     }
50     outlet
51     {
52         type patch;
53         faces
54         (
55             (5 6 2 1)
56         );
57     }
58     wall
59     {
60         type wall;
61         faces
62         (
63             (4 5 1 0)
64             (0 1 2 3)
65             (4 7 6 5)
66         );
67     }
68     adsorptwall
69     {
70         type wall;
71         faces
72         (
```

```
            (6 7 3 2)
        );
    }
);

mergePatchPairs
(
);

// ************************************************************************* //
```

Listing B.12: File `reactions`.

```
species
(
    CH4
    CO
    CO2
);

reactions
{}
```

Listing B.13: File `thermo.compressibleGas`.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      thermo.compressibleGas;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

CO2
{
    specie
    {
        nMoles          1;
        molWeight       44.01;
    }
    thermodynamics
    {
        Tlow            200;
        Thigh           5000;
        Tcommon         1000;
        highCpCoeffs    ( 4.45362 0.00314017 -1.27841e-06 2.394e-10 -1.66903e-14
    ↪ -48967 -0.955396 );
        lowCpCoeffs     ( 2.27572 0.00992207 -1.04091e-05 6.86669e-09
    ↪ -2.11728e-12 -48373.1 10.1885 );
    }
    transport
    {
        As              1.67212e-06;
        Ts              170.672;
    }
}

CH4
{
    specie
```

```
43      {
44          nMoles          1;
45          molWeight       16.043;
46      }
47      thermodynamics
48      {
49          Tlow            200;
50          Thigh           6000;
51          Tcommon         1000;
52          highCpCoeffs    ( 1.91179 0.00960268 -3.38388e-06 5.38797e-10
        ↪ -3.19307e-14 -10099.2 8.48242 );
53          lowCpCoeffs     ( 5.14826 -0.0137002 4.93749e-05 -4.91952e-08 1.70097e-11
        ↪ -10245.3 -4.63323 );
54      }
55      transport
56      {
57          As              1.67212e-06;
58          Ts              170.672;
59      }
60  }
61
62  CO
63  {
64      specie
65      {
66          nMoles          1;
67          molWeight       28.0106;
68      }
69      thermodynamics
70      {
71          Tlow            200;
72          Thigh           6000;
73          Tcommon         1000;
74          highCpCoeffs    ( 3.04849 0.00135173 -4.85794e-07 7.88536e-11
        ↪ -4.69807e-15 -14266.1 6.0171 );
75          lowCpCoeffs     ( 3.57953 -0.000610354 1.01681e-06 9.07006e-10
        ↪ -9.04424e-13 -14344.1 3.50841 );
76      }
77      transport
78      {
79          As              1.67212e-06;
80          Ts              170.672;
81      }
82  }
83
84  // ************************************************************************* //
```

Listing B.14: File `thermophysicalProperties`.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "constant";
14     object      thermophysicalProperties;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 thermoType
19 {
20     type            hePsiThermo;
```

```
    mixture         reactingMixture;
    transport       sutherland;
    thermo          janaf;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}

inertSpecie CO2;

chemistryReader foamChemistryReader;

foamChemistryFile "$FOAM_CASE/constant/reactions";

foamChemistryThermoFile "$FOAM_CASE/constant/thermo.compressibleGas";

// ************************************************************************* //
```

Listing B.15: File `turbulenceProperties`.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType  laminar;


// ************************************************************************* //
```

B.3. `system` directory

Listing B.16: File `controlDict`.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.4.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     generalMultiAdsorpFoam;

startFrom       startTime;

```

```
22 startTime       0;
23
24 stopAt          endTime;
25
26 endTime         400;
27
28 deltaT          1e-1;
29
30 writeControl    adjustableRunTime;
31
32 writeInterval   1;
33
34 purgeWrite      0;
35
36 writeFormat     ascii;
37
38 writePrecision  6;
39
40 writeCompression off;
41
42 timeFormat      general;
43
44 timePrecision   6;
45
46 runTimeModifiable true;
47
48 adjustTimeStep  true;
49
50 maxCo          0.5;
51
52 libs ("libAdsorptionBoundaryConditions.so");
53 // ************************************************************************* //
```

Listing B.17: File `fvSchemes`.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      fvSchemes;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 ddtSchemes
19 {
20     default         Euler;
21 }
22
23 gradSchemes
24 {
25     default         Gauss linear;
26     grad(p)         Gauss linear;
27 }
28
29 divSchemes
30 {
31     default         none;
32
33     div(phi,U)      Gauss limitedLinearV 1;
34     div(phi,Yi_h)   Gauss limitedLinear01 1;
```

```
35     div(phi,h)      Gauss limitedLinear 1;
36     div(phi,K)      Gauss limitedLinear 1;
37     div(phid,p)     Gauss limitedLinear 1;
38     div(phi,epsilon) Gauss limitedLinear 1;
39     div(phi,k) Gauss limitedLinear 1;
40     div((muEff*dev2(T(grad(U))))) Gauss linear;
41 }
42
43 laplacianSchemes
44 {
45     default         Gauss linear uncorrected;
46     laplacian(muEff,U) Gauss linear uncorrected;
47     laplacian(mut,U) Gauss linear uncorrected;
48     laplacian(DkEff,k) Gauss linear uncorrected;
49     laplacian(DepsilonEff,epsilon) Gauss linear uncorrected;
50     laplacian((rho*(1|A(U))),p) Gauss linear uncorrected;
51     laplacian(alphaEff,h) Gauss linear uncorrected;
52 }
53
54 interpolationSchemes
55 {
56     default         linear;
57 }
58
59 snGradSchemes
60 {
61     default         uncorrected;
62 }
63
64 fluxRequired
65 {
66     default         no;
67     p;
68 }
69
70
71 // ************************************************************************* //
```

Listing B.18: File `fvSolution`.

```
1  /*--------------------------------*- C++ -*----------------------------------*\
2  | =========                 |                                                 |
3  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
4  |  \\    /   O peration     | Version:  2.4.0                                 |
5  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
6  |    \\/     M anipulation  |                                                 |
7  \*---------------------------------------------------------------------------*/
8  FoamFile
9  {
10     version     2.0;
11     format      ascii;
12     class       dictionary;
13     location    "system";
14     object      fvSolution;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17
18 solvers
19 {
20     rho
21     {
22         solver          PCG;
23         preconditioner  DIC;
24         tolerance       1e-06;
25         relTol          0;
26     }
27
28     rhoFinal
29     {
```

```
30          $rho;
31          tolerance       1e-06;
32          relTol          0;
33      }
34
35      p
36      {
37          solver           PCG;
38          preconditioner   DIC;
39          tolerance        1e-6;
40          relTol           0;
41      }
42
43      pFinal
44      {
45          $p;
46          tolerance        1e-6;
47          relTol           0;
48      }
49
50      "(U|h|k|epsilon)"
51      {
52          solver          PBiCG;
53          preconditioner  DILU;
54          tolerance       1e-06;
55          relTol          0;
56      }
57
58      "(U|h|k|epsilon)Final"
59      {
60          solver          PBiCG;
61          preconditioner  DILU;
62          tolerance       1e-06;
63          relTol          0;
64      }
65
66      Yi
67      {
68          $hFinal;
69      }
70  }
71
72  PIMPLE
73  {
74      momentumPredictor yes;
75      nOuterCorrectors  1;
76      nCorrectors     1;
77      nNonOrthogonalCorrectors 0;
78  }
79
80
81  // ************************************************************************* //
```

Index

activity, 15
adsorption, 7
  isotherms, 7
  kinetics, 17
    diffusion-based, 18
    linear driving force, 18
  multicomponent, 13
    extended Langmuir model, 13
    extended Langmuir model with interaction coefficients, 14
    ideal adsorbed solution theory, 15
  single-component, 11
    Freundlich, 11
    Freundlich-Langmuir, 13
    Henry, 11
    Langmuir, 11
  types of, 7

conservation equations, 48
consistency, 43
continuity equation, 49
convergence, 43
Courant number, 43
Courant-Friedrichs-Lewy condition, 43

diffusion coefficient, 52
discretisation, 40

energy balance, 50
Euler equation, 49

finite-difference method, 44
finite-volume method, 46
fugacity, 15

Gibbs equation for a plane surface, 10
Gibbs free enthalpy, 15
grid, 40
  block-structured, 40
  structured, 40
  unstructured, 40

Helmholtz free energy, 9
Henry's Law, 7

International Union of Pure and Applied Chemistry, 7

mass balance
  partial, 49
  total, 49
material property, 52
momentum balance, 49

Navier-Stokes equation, 50

OpenFOAM, 55
  adsorpFoam, 62
  boundary condition, 55
    adsorpAdiabaticWall, 71
    adsorpWall, 71
  diffusion.H, 68
  file input and output, 57
  generalMultiAdsorpFoam, 65
    adsorptionProperties, 65
    createAdsorptionFields.H, 65
    elm.H, 67
    henry.H, 67
    iast-langmuir.H, 67
    ldf.H, 68
    limiters, 68
  reactingFoam, 60

partial differential equation, 39
pressure-velocity coupling, 50
  PIMPLE algorithm, 52
  PISO algorithm, 51
  SIMPLE algorithm, 51

residual, 44
Reynolds number, 76

Reynolds transport theorem, 48

stability, 43

Taylor series expansion, 40

weighted residual method, 44
    collocation method, 45
    Galerkin method, 45
    least-square method, 45
    subdomain method, 45