Efficient Implementation of Advanced Molecularly-based Equation of State Models out of the Documentation Level

vorgelegt von
Dipl.-Ing.
Victor Alejandro Merchan Restrepo
geb. in Gießen

von der Fakultät III - Prozesswissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing.-

genehmigte Dissertation

Promotionsausschuss:
Vorsitzender: Prof. Dr.-Ing. Felix Ziegler
Gutachter: Prof. Dr.-Ing. Jens-Uwe Repke
Gutachter: Prof. Dr.-Ing. Günter Wozny
Gutachter: Prof. Ph.D. Flavio Manenti

Tag der wissenschaftlichen Aussprache: 14.12.2018

Berlin 2019

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fachgebiet Dynamik und Betrieb technischer Anlagen der Technischen Universität Berlin. Die wissenschaftlichen Fragenstellungen resultierten aus meiner Mitarbeit am Sonderforschungsbereich Transregio 63 InPROMPT „Integrierte chemische Prozesse in flüssigen Mehrphasensystemen“, in dem ich als assoziiertes Mitglied aktiv war.

Mein Dank gilt an dieser Stelle meinem Doktorvater, Herrn Professor Dr.-Ing Günter Wozny. Seine Begeisterungsfähigkeit und bedingungslose Unterstützung haben erheblich zum Gelingen dieser Arbeit beigetragen. Besonders dankbar bin ich für die Möglichkeit, neben den Lehraufgaben an der TU Berlin, diverse Modellierungskurse im Ausland aktiv mitzugestalten. Prof. Dr.-Ing. Jens-Uwe Repke, Erstgutachter dieser Arbeit, danke ich für erste wertvolle Einblicke in das wissenschaftliche Arbeiten, die er mir während meiner Studienzeit gegeben hat, und für seine tatkräftige Unterstützung nach seiner Übernahme des Fachgebiets. Weiterhin danke ich Prof. Ph.D. Flavio Manenti für die interessanten Diskussionen über numerische Themen und für seine Unterstützung als Gutachter, und Prof. Dr.-Ing. Felix Ziegler für die Übernahme des Prüfungsvorsitzes. Herrn Prof. Dr.-Ing. Harvey Arellano-Garcia bin ich für seine Unterstützung in der Anfangsphase meiner Arbeit am Fachgebiet ebenfalls sehr dankbar.

Bei allen ehemaligen Kolleginnen und Kollegen am Fachgebiet bedanke ich mich für das gute Arbeitsklima. Ich werde alle gemeinsame Unternehmungen in sehr guter Erinnerung behalten. Vornehmlicher Dank gilt Stefan Kuntsche, Robert Kraus, Tilman Barz, Diana Carolina Lopez Cardenas, Sandra Fillinger, Erik Esche und Gregor Tolksdorf. Meinen studentischen Mitarbeitern, sowie meinen Masterarbeiter/-innen und Praktikanten Christoph Walowski, Dominik Krämer, Maria Fernanda Gutierrez Sanchez, Julian Agudelo, Jielian Zhao, Marten Schlüter und Lía de la Paz Rodriguez Pizarro danke ich sehr für ihren Einsatz und ihre sehr gute Arbeit.

Diese Arbeit ist meinen Eltern, meiner Frau Ada, und meinen Töchtern Wanda und Ina gewidmet. Meine Eltern haben mir meinen Wunsch ermöglicht, in Deutschland zu studieren, und damit den Weg für die vorliegende Arbeit geebnet. Meiner Frau danke ich für ihre Geduld und ihre liebevolle Unterstützung in dieser besonderen Phase unserer gemeinsamen Reise.

Burghausen, im Januar 2019 Victor Alejandro Merchan

Kurzfassung

Im Angesicht der zunehmenden Anzahl moderner thermodynamischer Modelle für ingenieurtechnische Anwendungen und deren immer größer werdenden algebraischen Komplexität besteht in der Prozessindustrie und in der Forschung ein großer Bedarf an Methoden, die die Verbreitung und Wiederverwendung der neuen Modelle effizienter gestalten, und somit die Vorteile dieser Modelle schneller zugänglich machen können. Die bisher gängige Praxis, Modellgleichungen als Anhang von Publikationen mitzuliefern, ist ohne Zweifel verbesserungswürdig, nicht zuletzt wegen häufig auftretender Inkonsistenzen zwischen der von den Autoren zur Verfügung gestellten Modelldokumentation und der von ihnen tatsächlich benutzten softwaretechnischen Implementierung. Vor diesem Hintergrund stellt die Nutzung von dokumentationsbasierten Modellen, wie sie von der Modellierungsumgebung MOSAICmodeling unterstützt werden, einen wertvollen Ansatz dar, da diese nicht nur Konsistenz zwischen Dokumentation und Modellimplementierung garantieren, sondern auch die Möglichkeit bieten, mittels Code-Generierung Modellimplementierungen in beliebigen Programmier- und Modelliersprachen zu erzeugen.

Im Rahmen dieser Arbeit werden Methoden entwickelt und evaluiert, die das Ziel haben, den Implementierungsaufwand moderner thermodynamischer Modelle zu reduzieren und gleichzeitig die Verbreitung dieser Modelle zu fördern. Der Kern der Konzepte basiert auf der Anwendung rechnergestützter Methoden für strukturelle Analyse und Ableitungsberechnung, insbesondere algorithmische Differentiation, auf gleichungsbasierte, dokumentationsnahe Modelle der freien Helmholtz Energie. Die resultierenden Implementierungen sind nicht nur für Sprachen und Modellierungsumgebungen gültig, welche sequentielle Auswertungen zulassen. Geeignete Implementierungen werden auch für Tools generiert, welche vollständig gleichungsbasiert arbeiten.

Es konnte insgesamt gezeigt werden, wie die entwickelten Methoden nicht nur die Implementierung moderner thermodynamischer Modelle erheblich erleichtern, sondern auch zu hocheffizienten rechnergestützten Modellimplementierungen führen können. Darüber hinaus wurde im Rahmen einer vergleichenden Analyse der Einfluss der thermodynamischen Modelle auf die Qualität der Prozesssimulation am Beispiel eines homogenkatalytischen Prozesses bewertet. Hinsichtlich der Beschreibung des Phasenverhaltens mit dem heterosegmentierten PCP-SAFT-Modell konnte, unter Erweiterung des Suchraumes, ein neuer Parametersatz gefunden werden, welcher eine deutlich verbesserte Beschreibung aldehydhaltiger Systeme ermöglicht und sich somit für den Einsatz in rigoroser Prozesssimulation eignet. Eine vergleichbar gute Beschreibung der Phasengleichgewichte konnte mit einem entsprechend parametrierten EoS/g^E Modell mit niedrigerem Rechenaufwand erreicht werden. Hinsichtlich der Dichteberechnung ist jedoch mit dem EoS/g^E-Modell erwartungsgemäß eine größere Abweichung in Kauf zu nehmen.

Abstract

Considering the increasing number of modern thermodynamic models for engineering applications and their continuously growing algebraic complexity, in the chemical process industry and research there is a high demand for methods and mechanisms that enhance the dissemination and reuse of these models in an efficient way. Experience shows that the common practice of providing model equations as annexes of publications does not lead to a successful spread of the models. Coding complex algebraic expressions can be highly challenging and prone to error, in particular when considering inconsistencies that frequently appear between model documentation provided by the authors and their actual software implementations. In light of the above, the use of documentation based models, as supported by the modeling environment MOSAICmodeling, appears to be a promising approach. This type of models guarantees the consistency between model documentation and implementation and builds the basis of model implementations in arbitrary programming or modeling languages.

This work focuses on the development and evaluation of methodologies conceived to ease the implementation of modern molecularly based equation of state models while enhancing their dissemination. The core of the concept is the use of computational methods for structural analysis and derivative evaluation, in particular algorithmic differentiation. These are applied on an equation-oriented model implementation of the free Helmholtz energy, originally developed on the documentation level. The resulting model implementations can be used on different modeling environments, including those supporting sequential evaluations and explicit functions, and those supporting fully equation-oriented systems.

In general, it was shown that the developed methods not only considerably facilitate the implementation of modern thermodynamic models; in fact, the resulting implementations proved to be highly efficient in terms of computational performance. Furthermore, within this thesis a comparative evaluation of thermodynamic models was performed in order to quantify the influence of the model selection on the quality of the process simulation of a homogeneous catalytic process. By extending the search space for the heterosegmented PCP-SAFT equation of state model, a new set of parameters was estimated which leads to a significant improvement in the description of the phase behaviour of systems containing long chain aldehydes. With a properly parameterized EoS/g^E model a successful description of phase equilibrium of comparable quality could be reached as well, however, as expected, with relatively large deviations in the density calculations.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AC-SAMMM	The AaChen platform for Structured Automatic Manipulation of Mathematical Models
ACM	Aspen Custom Modeler [see 10]
AD	Algorithmic differentiation, also known as automatic differentiation or computational differentiation
AE	Algebraic equation system
ARD	Average relative deviation (here for variable y)[%]

$$\mathrm{ARD} = 100 \cdot \frac{1}{NM} \cdot \sum_{n=1}^{NM} \left| \frac{y_n^{cal} - y_n^{exp}}{y_n^{exp}} \right|$$

BBD	Bordered block diagonal (decomposition)
BD	Block diagonal (decomposition)
BIPs	Binary interaction parameters
CAPE	Computer-aided process engineering
CAS	Computer algebra system
CFD	Computational fluid dynamics
CO	CAPE-OPEN, Carbon monoxide
CO-LaN	CAPE-OPEN Laboratories Network
CPA	Cubic-Plus-Association (EoS model)
CPR	Curtis-Powell-Reid decomposition algorithm [see 39]
CPU	Central processing unit
DAE	Differential algebraic equation system
`dcc`	Derivative source compiler, AD tool with support of FM and RM implementing ST approach
DMF	N,N-Dimethylformamide
EFCE	European Federation of Chemical Engineering
EoS	Equation of state
EoS/g^E	Cubic EoS model with mixing rule based on model for excess Gibbs energy
ESO	Equation Set Object (Component of CAPE-OPEN's *numerical solvers*)
FD	Finite differences
FM	Forward mode of AD
GC-SAFT	Group contribution approach within SAFT framework

G^E	Excess free enthalpy
GLE	Gas-liquid equilibrium
H2	Hydrogen
HC	Hand coded (mostly in context of derivatives)
hs-PCP-SAFT	PCP-SAFT considering heterosegmented chains
iC12en	iso-dodecene
IPDAEs	Integro-partial differential equation system
LJ-SAFT	Lennard-Jones SAFT by Kraska and Gubbins [110]
LLE	Liquid-liquid equilibrium
max. RD	Maximum relative deviation [%] $$\max.\mathrm{RD} = 100 \cdot \max \left\lvert \frac{y_n^{cal} - y_n^{exp}}{y_n^{exp}} \right\rvert$$
MathML	Mathematical Markup Language
MERSHQ	MERSHQ equations - Material balances, Energy balance, mass and heat-transfer Rate equations, Summation equations, Hydraulic equations for pressure drop, eQuilibrium equations
MESH	MESH equations - Material balances, Equilibrium relations, Summation equations, Heat balance
MHV2	Modified Huron/Vidal mixing rules of second order
NLP	Nonlinear programming
NRTL	Non-random two-liquid local composition model
nC12en	1-dodecene
nC13al	n-tridecanal
ODE	Ordinary differential equation system
OED	Optimal experimental design
OO	Operator overloading (implementation type of AD)
PCIP-SAFT	Perturbed chain induced-polar SAFT
PC-SAFT	Perturbed chain SAFT
PCP-SAFT	Perturbed chain polar-SAFT
PDAE	Partial differential algebraic equation system
PSE	Process systems engineering
pT-Flash	Equilibrium stage model with specified pressure and temperature
PVT	Pressure-volume-temperature
RM	Reverse mode of AD
SAFT	Statistical associating fluid theory
SAFT-VR	SAFT for potentials of variable range
SD	Symbolic derivative
SLE	Solid-liquid equilibrium
s-PC-SAFT	Simplified PC-SAFT by von Solms et al. [215]
SRK	Soave-Redlich-Kwong (EoS model)
SRK-MHV2	SRK EoS model with MHV2 mixing rules
s. t.	Subject to
ST	Source transformation (implementation type of AD)
TMS	Thermomorphic multicomponent solvent
tPC-PSAFT	Truncated perturbed chain-polar SAFT
TPDF	Tangent plane distance function
UNIFAC-DO	UNIFAC Dortmund activity coefficient model
VLE	Vapor-liquid equilibrium

VLLE	Vapor-liquid-liquid equilibrium
XML	Extended Markup Language

Greek letters

ε	Depth of pair potential [J]
ε/k	PC-SAFT pure component parameter [K]
η	Packing fraction [–]
γ	Activity coefficient [–]
μ	Chemical potential [$\mathrm{J\,mol^{-1}}$], Dipole moment [D]
ν	Stochiometric coefficient [–]
Φ	Phase fraction [–]
φ	Fugacity coefficient [–]
π	Number of phases $\pi \in \mathbb{N}$
χ	Arbitrary variable $\chi \in \mathbb{R}$
ψ	Arbitrary variable $\psi \in \mathbb{R}$
ρ	Total number density of molecules [$\mathrm{Å^{-3}}$]
$\hat{\rho}$	Molar density [$\mathrm{kmol\,m^{-3}}$]
σ	Segment diameter [Å], (PC-SAFT parameter)

Indices and numbered superscripts

α	Segment number $\alpha \in \mathbb{N}$
β	Segment number $\beta \in \mathbb{N}$
c	Component number $c \in \mathbb{N}$
i	Component number i, i^{th} elementary function $\in \mathbb{N}$
j	Component number j, j^{th} entry of a vector, j^{th} subsystem, $j \in \mathbb{N}$
k	Number of elementary function in which $\mathbf{G}(\mathbf{u})$ can be decomposed, $k \in \mathbb{N}$
n	Measurement number $n \in \mathbb{N}$
s	Stream number $s \in \mathbb{N}$

Latin letters

A	Helmholtz free energy [J]
a	Helmholtz free energy related to the total number of molecules, $a = A/N$
$\tilde{a}$	Reduced Helmholtz free energy [–]
$\hat{a}$	Molar Helmholtz free energy [$\mathrm{J\,mol^{-1}}$]
$a_{j,i}$	Asymmetric BIP in Eq. (B.1)
$b_{j,i}$	Asymmetric BIP in Eq. (B.1)
C( )	Same as cost( )
$c_{j,i}$	Asymmetric BIP in Eq. (B.1)
cost( )	Computational cost related to evaluation of expressions inside ( ), mostly in t_{CPU} [s]
$d_{j,i}$	Asymmetric BIP in Eq. (B.1)

$e_{j,i}$	Asymmetric BIP in Eq. (B.1)
$\mathbf{F}(\mathbf{x})$	Vector of expressions of algebraic equation system $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, see Eq. (4.1)
$\mathbf{F_j}$	j^{th} equation subsystem of $\mathbf{F}(\mathbf{x})$, $\mathbf{j} \in 1, 2, 3$
F	Molar flow [mol s^{-1}] or [kmol s^{-1}], or mol number [mol]
$\mathbf{f}(t, \mathbf{x}, \mathbf{z})$	Right-hand side of differential equations
$f()$	Right-hand side of general scalar functional relationship
f	Fugacity [Pa]
$f_{j,i}$	Asymmetric BIP in Eq. (B.1)
$\mathbf{G}(\mathbf{u})$	Right-hand side of algebraic (scalar or vectorial) expressions
$\mathbf{G}'(\mathbf{u})$	Jacobian matrix of $\mathbf{G}(\mathbf{u})$, see Eq. (3.2)
$\mathbf{G}'_{\mathbf{v}}(\mathbf{u})$	Directional derivative of $\mathbf{G}(\mathbf{u})$ along $\mathbf{v}$, see Eq. (3.5)
G	Gibbs free energy [J]
$\mathbf{g}(t, \mathbf{x}, \mathbf{z})$	Right-hand side of algebraic equations
H	Enthalpy [J]
h	Magnitude of the perturbation for approximation of derivatives
$\hat{h}$	Molar enthalpy [J mol^{-1}]
Im[ ]	Imaginary part of complex number inside [ ]
i	Unit imaginary number. $i^2 = -1$
k	Boltzmann constant $k = 1.38066\,10^{-23}$ J K^{-1}
$k_{i,\alpha,j,\beta}$	BIP between segment α of component i and segment β of component j
m	Number of segments per chain $m \in \mathbb{R}$ [–] (PC-SAFT parameter)
$\mathbb{N}$	Set of natural numbers
N	Total number of molecules [–]
N	Number or amount of something, e.g. N_{eval} denotes the number of evaluations [–]
NC	Number of components [–]
NG	Maximum number of phases in equilibrium according to Gibbs' phase rule [–]
NM	Number of measurements [–]
$NGLE$	Number of gas solubility measurements [–]
NP	Number of phases [–]
$\mathbf{n}$	Vector of mole numbers in a mixture, $\mathbf{n} = [n_1, \ldots, n_{n_\mathbf{n}}]^T$
n	Mole numbers [–]
nC12al	n-dodecanal
OF	General objective function to be minimized
$\mathbf{p}$	Vector of variables w.r.t. which derivatives are built,$\mathbf{p} = [p_1, \ldots, p_{n_\mathbf{p}}]^T$
p	Pressure [Pa]
Q	Heat flow [J s^{-1}], or heat amount [J]
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Set of n-tuples of real numbers
R	Universal gas constant $R = 8.314$ J mol^{-1} K^{-1}
r	Reaction rate [kmol s^{-1}]
$\mathbf{S}$	Seed matrix to build directional derivatives
S	Entropy [J K^{-1}]
$\hat{s}$	Molar entropy [J K^{-1} mol^{-1}]
T	Temperature [K]
t	Time [s]
U	Internal energy [J]
$\mathbf{u}$	Vector of input (fixed) variables, $\mathbf{u} = [u_1, \ldots, u_{n_\mathbf{u}}]^T$

V	Volume [m^3]
$\mathbf{v}$	Vector along which the directional derivative $\mathbf{G}'(\mathbf{u}) \cdot \mathbf{v}$ is built, $\mathbf{v} = [v_1, \ldots, v_{n_\mathbf{v}}]^T$ where $n_\mathbf{v} = n_\mathbf{u}$
v	Molecular volume [$Å^3$]
$\hat{v}$	Molar volume [$m^3\,mol^{-1}$]
$\mathbf{w}$	Vector along which the directional derivative $\mathbf{w}^T \cdot \mathbf{G}'(\mathbf{u})$ is built, $\mathbf{w} = [w_1, \ldots, w_{n_\mathbf{w}}]^T$ where $n_\mathbf{w} = n_\mathbf{y}$
w	Mass fraction [$kg\,kg^{-1}$]
X	Conversion
$\mathbf{x}$	Vector of differential variables, $\mathbf{x} = [x_1, \ldots, x_{n_\mathbf{x}}]^T$
$\mathbf{x}$	Vector of mixture molar fractions, $\mathbf{x} = [x_1, \ldots, x_{n_\mathbf{x}}]^T$
$\mathbf{x_j}$	Vector of algebraic variables that first appear in equation subsystem $\mathbf{F_j}$, $j \in 1, 3$
$\tilde{\mathbf{x}}_\mathbf{j}$	Vector of any kind of variables that appear in the equation subsystem $\mathbf{F_j}$, $j \in 1, 2, 3$
x	Molar fraction [$mol\,mol^{-1}$]
Y	Yield
$\mathbf{y}$	Resulting value of evaluation of $\mathbf{G}(\mathbf{u})$, $\mathbf{y} = [y_1, \ldots, y_{n_\mathbf{y}}]^T$
Z	Compressibility factor [–]
$\mathbf{z}$	Vector of algebraic variables, $\mathbf{z} = [z_1, \ldots, z_{n_\mathbf{z}}]^T$

Subscripts and superscripts

$assoc$	Contribution due to association forces
$C10an$	Decane
$C12en$	General dodecene consisting of 1-dodecene and iso-dodecene
CPU	Central processing unit (used to denote CPU time t_{CPU})
cal	Calculated value
$chain$	Contribution due to chain formation out of individual segments
DMF	$N,N-$dimethylformamide
dd	Number of directional derivatives
$dipole$	Contribution due to dipole/dipole interactions
$disp$	Contribution due to dispersive attraction
$eval$	Related to evaluations
exp	Experimental value
FM	Forward mode of AD
GAS	Gaseous component
GLE	Related to gas solubility measurements
hc	Contribution of hard chain system
(j)	Numbered superscript to characterize corresponding phase, $j \in \mathbb{N}$
L	Liquid phase
LLE	Related to liquid liquid equilibrium measurement
LV	Related to vapor liquid equilibrium
NC	Number of components
$NLLE$	Number of LLE measurements
NPM	Number of pure compound measurements
$nC12al$	n-dodecanal
$nC12al - head$	Head segment of n-dodecanal

$nC12al-tail$	Tail segment of n-dodecanal
$nC12en$	1-dodecene
$n_{\mathbf{u}}$	Number of elements of vector $\mathbf{u}$
$n_{\mathbf{v}}$	Number of elements of vector $\mathbf{v}$
$n_{\mathbf{w}}$	Number of elements of vector $\mathbf{w}$
$n_{\mathbf{x}}$	Number of elements of vector $\mathbf{x}$
$n_{\mathbf{y}}$	Number of elements of vector $\mathbf{y}$
$n_{\mathbf{z}}$	Number of elements of vector $\mathbf{z}$
org	Related to an organic phase
p	At constant pressure (used in context of heat capacity)
pol	Related to a polar phase
$pure$	Related to a pure compound property
res	Residual term (deviation from ideal gas property)
RM	Reverse mode of AD
Sat	Saturated property (e. g. at boiling or dew curve)
seg	Dispersion-repulsion contribution of single segment
sin	Single
sol	Solution
$*$	Property of an ideal gas
syngas	Synthesis gas
T	Transpose of a vector, At a fixed temperature T
V	Vapor phase
v	At constant volume (used in context of the heat capacity)
0	At reference state 0

Chapter 1

Introduction

1.1 Motivation

Within the development of novel processes and the improvement of existing ones there is a high demand for the application of model-based methods, for example in form of process simulation, process design or process optimization studies, since these offer systematic ways of finding potential process improvements while reducing the need for complex and costly experiments. A particular high success can be expected from such methods if used considering high-quality predictive thermodynamic models capable of covering the studied ranges of operation conditions.

Taking into account the most recent efforts in the development of general thermodynamic models for engineering applications [106], molecularly based equation of state models,[1] such as the ones based on the Statistical Associating Fluid Theory (SAFT), are perhaps the most promising candidates to expand the ranges of applications of currently used thermodynamic models [84, 104] and hence to enhance the successful application of model-based methods as well. Due to their sound theoretical basis, which explicitly allows the consideration of different types of molecular interactions, molecularly based models are particularly suitable to describe thermodynamic behavior over wide ranges of operation conditions.

The big impact of the original SAFT model has triggered the development of numerous modifications that account for different types of molecular interactions and different approaches to describe these. Though several of these developments have shown impressive performance on the description of single complex systems [see e. g. 76, 104, 209], in view

[1] A more detailed connotation for the term "'molecularly based equation of state models"' as used in this thesis is given by "'advanced equations of state based on statistical mechanics that incorporate ideas from perturbation, chemical or lattice theories"'.

of the very large number of developed models the overall dissemination of models of this family is not only low but slow as well. While few models, such as the PC-SAFT and CPA equation of state have made it into commercial process simulators and physical property packages and are nowadays well established in academic and industrial applications, a significant part of new model developments and extensions of currently available models rarely gets the possibility of being thoroughly evaluated for potential applications. The relatively high effort related to the implementation of new models coupled with the common situation that requirements on evaluations of thermodynamic models are mostly short term is only one of the main reasons that explain the deficits in the dissemination of new model developments. This fundamental problem is also recognized and discussed in a study of the European Federation of Chemical Engineers (EFCE) on industrial requirements for thermodynamics and transport properties [84], which also proposes possible approaches to remedy it. In particular Hendriks et al. [84] postulate the necessity of standardized model implementations with extensive model documentation and recommend the publication of the source code used for the evaluation studies.

In view of the big potentials which result from the use of advanced molecularly based equation of state model in particular and of models of arbitrary high algebraic complexity in general, it appears to be essential to develop approaches that reduce the complexity of the model implementation step while providing means to enhance the dissemination of the models.

1.2 Goals of This Thesis

This thesis focuses on the development and evaluation of methods conceived to ease the implementation and enhance the dissemination of molecularly based equation of state models. For this purpose, the concept of modeling on the documentation level,[2] as introduced by Kuntsche [113] and supported by the web-based modeling environment MOSAICmodeling[3], is extended with advanced computational approaches based on algorithmic differentiation and automatic structural analysis. The developments introduce means of providing standard (reference) model implementations with a consistent model documentation and show a possible way to expand the use of documentation-based models in rigorous process modeling applications.

An important reason that motivates the development of the aforementioned methods is given by the high algebraic complexity of the underlying models, which makes their implementation a challenging and highly error-prone process. Since a detailed analysis of the

[2] In this thesis and elsewhere this concept is also known as documentation-based modeling.

[3] `www.mosaic-modeling.de`

models shows that a significant part of the model equations that need to be implemented consists of partial derivatives of expressions defined by other model equations, in a first stage it should be evaluated how computational methods for derivative evaluation can be used to reduce the implementation effort. Within this context, it should also be evaluated how the selection of derivative evaluation methods has an influence on the resulting overall computational effort. Particularly interesting is the comparison of advanced computational methods against hand coded derivatives, as they are normally made available in model documentation provided in publications.

Since the inherent equation-oriented nature of model implementations based on the documentation level and its modifications may give rise to difficulties when solving engineering problems with such implementations, for example due to the necessity of a large amount of high-quality starting values, in a second stage, alternative methods should be considered which open the possibility of generating more efficient sequential evaluation procedures out of the equation-oriented documentation-based implementation. For this purpose, structural analysis techniques based on the block diagonal decomposition of the Jacobian of the equation system are taken into account. In particular it is interesting to study the potential of combining the information provided by structural analysis with algorithmic differentiation methods. A modified type of purely equation-oriented implementation based on the generation of a linear sensitivity equation system is studied as well.

Independent of the development of methods to ease the implementation and enhance the dissemination of advanced equation of state (EoS) models, in this thesis a comparative evaluation of thermodynamic models is considered as well. Considering a modern molecularly based EoS model, the heterosegmented PCP-SAFT EoS model, and a cubic EoS model with mixing rules based on an activity coefficient model, the SRK-MHV2 EoS model with NRTL as underlying activity coefficient model, not only the correlative capabilities of the models are evaluated but also their influence on the quality of a process wide simulation. The evaluated process, the hydroformylation of 1-dodecene in a thermomorphic solvent system, is a good example for processes that take place on complex, multiphase systems, a type of process that can significantly profit from the consideration of advanced thermodynamic models and model-based methods.

1.3 Overview of This Thesis

Chapter 2 summarizes the modeling fundamentals treated within this thesis. Section 2.1 introduces the fundamental elements of rigorous process models putting a particular focus on the classification of model implementation types. Section 2.2 presents fundamentals on popular approaches for tool-independent model implementations, a concept that is

supported and extended by documentation-based models, which are introduced in detail in Section 2.3. Section 2.4 summarizes the thermodynamic fundamentals treated in this thesis, putting particular emphasis on the role of derivatives, flash calculations and the thermodynamic models considered in the case study of the hydroformylation process.

Chapter 3 summarizes the fundamentals on the derivative evaluation methods that are considered and evaluated in order to reduce the implementation effort for molecularly based equation of state models. In Chapter 4 a short analysis is shown that makes clear the complexity of molecularly based EoS models, before introducing in detail the developed approaches to reduce the model implementation effort based on the concepts introduced in the previous chapters. In Chapter 5 the results are provided. Section 5.1 first gives a short introduction to the hydroformylation process, which is the basis of most evaluations, before the influences of different derivative evaluation methods and implementation approaches are discussed in Section 5.2 and 5.3. Section 5.4 presents the results of a comparative evaluation of thermodynamic models related to the considered hydroformylation process. These include parameter estimations studies, description of LLE, GLE, and the validation of the thermodynamic models against experimental data of a miniplant with full recycle. Section 5.5 presents results of process wide simulations considering different thermodynamic models.

Finally, Chapter 6 summarizes the main results of this thesis and Chapter 7 gives an outlook to the open questions and future research works that to the author's opinion are worth to be investigated in further detail.

Chapter 2

Modeling Fundamentals

From the many steps of the model development task [60], that are worth to do some research in (model conceptualization, model construction, model validation), this thesis is mainly a contribution to the model construction/model implementation step/task of customized user models. Accordingly, the modeling fundamentals provided in this chapter are limited to this field, and the chemical process treated in the case studies. Excellent reviews on process modeling, including detailed descriptions of modeling work flows, are given among others by Hangos and Cameron [79], Marquardt [124], Preisig [168], Heitzig [83] and Gani et al. [60]. Also, several extensive reviews on modeling tools used in process and chemical engineering have been published, see for example the reviews by Marquardt [125], von Wedel et al. [219] or Merchan et al. [132].

The content of this chapter is organized as follows: first, the main components of rigorous process models are introduced, putting a focus on their general mathematical formulation, and on common approaches for model implementation. Second, the concept of documentation-based models, as characterized by Kuntsche [113], is presented and its current implementation is evaluated with respect to the requirements of rigorous process modeling. Finally, thermodynamic fundamentals are provided due to their central importance in this thesis. On the one hand thermodynamic models, as common (and usually complex) part of advanced models, offer an interesting field for the study of the integration of advanced derivative generation techniques in the model development process.[1] On the other hand, the evaluation and selection of proper thermodynamic models is an indispensable part of the process analysis and simulation presented in Section 5.1.

[1] This is discussed in detail in Chapter 4. Note that the fundamentals provided in Chapter 3 are necessary as well in order to understand this discussion.

2.1 Fundamentals of Rigorous Process Models

The definition of process model considered in this thesis is based on Matzopoulos [130]. Accordingly, "*A process model is considered to be a set of mathematical relationships and data that describe the behavior of a process system, capable of being implemented within a process modeling environment in such a way that they can be used to generate further knowledge about a system through simulation, optimization, and other related activities*". In addition to the above definition, from the author's point of view documentation is an integral part of process models. Information regarding the origin of the models, their underlying assumptions, and a consistent nomenclature have a decisive impact on the quality and usability of the process models.

Process models can be classified according to several criteria, for example their mathematical structure (algebraic, dynamic, with lumped parameters, distributed parameters, linear, nonlinear, discrete, continuous, etc.), the appearance and grade of randomness (stochastic vs deterministic), or according to the grade of physical foundation of the model equations.[2] Regarding this last point, models are usually classified in rigorous models (also called first principle, mechanistic, phenomenological, or white-box models), empirical models (also called statistical, data-driven, or black-box models), and semi-empirical models (also called semi-mechanistic, hybrid, or grey-box model) [94]. Though the methods developed within this thesis, are equally applicable to any of the aforementioned type of models, if the models can be formulated as equation system, in this thesis emphasis is placed on rigorous models, as they offer interesting applications for the implementation of physical properties and thermodynamic models.

Depending on the followed modeling goal and resulting modeling depth, several systematic approaches define the model equations that are required to model the system, that is, the building blocks of the considered rigorous process model. Examples are systematic approaches that recommend *setting up* the MESH equations (Material balances, Equilibrium relations, Summation equations, Heat balance) for equilibrium modeling, or the MERSHQ equations (Material balances, Energy balance, mass and heat-transfer Rate equations, Summation equations, Hydraulic equations for pressure drop, eQuilibrium equations) for non-equilibrium modeling [223]. Further systematic approaches for the development of rigorous process models are described, among others, by Marquardt [124], Hangos and Cameron [79], or Preisig [168].

Independent of the followed systematic approach for model development, a coarser classification of the building blocks of rigorous process models only distinguishes between balance

[2] More detailed classifications of model types including some examples are given by Cameron [31] or by Gani et al. [60].

equations, which follow from conservation laws, and constitutive equations that include all further required equation systems [79].

An important group of constitutive equations is given by the model equations that describe physical properties or (mostly mixture) properties required for equilibrium calculations. Though in principle this type of equations may have the same mathematical (algebraic) nature as any model equation, physical property and thermodynamic models show some peculiarities that make them worth to be studied separately. Therefore, in the next subsections the general model implementation of physical properties are treated separately from the remaining model equations.

2.1.1 Mathematical model formulation

Depending on the modeling goal and the considered modeling depth, the equation system resulting from the modeling step may be of different mathematical nature. Steady state analysis and simulation of lumped parameter systems leads to general algebraic equation systems (AEs), whereas differential-algebraic equation systems (DAEs) or ordinary differential equations systems (ODEs) usually arise from modeling lumped parameter dynamic systems or one dimensional, steady state distributed parameter systems. Process models considering spatial distribution or additional internal coordinates, as in the case of population balance applications, can lead to partial-differential-algebraic equations systems (PDAEs) or integro-partial differential algebraic equation systems (IPDAEs).[3]

Considering the previous discussion and since AEs and ODEs are special cases of DAEs, it becomes clear why most mathematical models used in PSE applications fit into general model formulations for DAEs. A common representation for DAEs is given by the so-called *semi-explicit* DAEs model formulation, see Eq. (2.1).

$$\mathbf{x}' = \mathbf{f}(t, \mathbf{x}, \mathbf{z}), \tag{2.1a}$$

$$\mathbf{0} = \mathbf{g}(t, \mathbf{x}, \mathbf{z}), \tag{2.1b}$$

where t is the independent time variable, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ is the vector of differential variables and $\mathbf{z} \in \mathbb{R}^{n_\mathbf{z}}$ is the vector of algebraic variables. Note that the most general *fully implicit* DAE formulation

$$\mathbf{F}(t, \mathbf{y}, \mathbf{y}') = \mathbf{0}, \text{where } \mathbf{y} = (\mathbf{x}^T, \mathbf{z}^T)^T \tag{2.2}$$

[3] In PSE applications PDAEs and IPDAEs are usually restricted to standard geometries. Besides that, they can be converted in standard DAEs by applying automatic discretization techniques, see for example [49].

can always be written in the semi-explicit form (Eq. (2.1)) [9, 80].[4] As already remarked, the semi-explicit DAE formulation of Eq. (2.1) also represents a superset that includes the formulation of general ODEs $\mathbf{x}' = \mathbf{f}(t, \mathbf{x})$ and general AEs $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ as special cases.

Model implementation/representation

In terms of information technology *equation-based* modeling, as done when building rigorous process models from the scratch, "*is a declarative approach for problem specification*" [163]. That is, the resulting model equations only show relations between model variables without prescribing any order of evaluation. In order to solve a given simulation or optimization problem a model implementation is required. For the model implementation, the chosen model representation is of big relevance. This later can be available in an *open-form* or in a *closed-form* [126].

An open-form model implementation/representation results from transferring a given equation system (for example Eq. (2.1)) into a generic modeling language and does not specify any model inputs, outputs, or state variables. It is important to emphasize that equations, the core element of open-form model implementations (e. g. $g_1(\mathbf{z}) = 0$), only represent an acausal relation among variables $z_1, z_2, \ldots, z_{n_z}$. That means, they neither specify which variables are inputs or outputs nor they define an order of evaluation [58]. For a model-based task to be solved, the open-form model implementation needs to be coupled to a feasible variable specification and a general-purpose numerical solver. Though at this stage the input/output causality becomes clear, this cannot always be exploited during the evaluation procedure, since the causality cannot always be expressed explicitly in form of *assignments*, particularly not when the model is described by nonlinear expressions.

On the other hand, closed-form models are designed to provide fixed model outputs for given model inputs and commonly consist of assignments, (e. g $z_{n_z} := z_1 - z_2$ for known z_1 and z_2), control loops, and other common elements of procedural programming languages. In case of a general AE $\mathbf{g}(\mathbf{z}) = \mathbf{0}$ a closed-form calculation can be represented according to Eq. (2.3)

$$\mathbf{y} := \mathbf{G}(\mathbf{u}), \tag{2.3}$$

where the vector of algebraic variables $\mathbf{z} = (\mathbf{y}^T, \mathbf{u}^T)^T$ is split in known input variables $\mathbf{u}$ and the required output variables $\mathbf{y}$. Accordingly, the closed-form formulation is related

[4] Since the current implementation of documentation-based models treated in this thesis only accepts the formulation of DAE system in the semi-explicit form, the focus is laid only on this type of formulation.

with the open-form equation system (see Eq. (2.4)).[5]

$$\mathbf{g}(\mathbf{z}) = \mathbf{y} - \mathbf{G}(\mathbf{u}) = \mathbf{0} \tag{2.4}$$

Closed form models can be realized in several ways, for example as sequential evaluation procedure consisting of assignments/explicit equations, as problem-tailored algorithms (as in flash calculations or distillation columns models), or as open-form model implementation coupled with general-purpose numerical solution algorithms.

The information, whether a mathematical model is available in an open-form or a closed-form formulation is central for the solution approach chosen and for the integration of models of different origins (see next section). Note that not only the chosen solution approach is affected by this, but also the computational efficiency for derivative evaluation (see Chapter 3).

2.1.2 Physical properties

The need of evaluating physical pure compound and mixture properties is one of the main characteristics of chemical process modeling in comparison to other application fields of modeling. Besides the description of pure compound thermophysical properties, there is a major need to describe mixture properties properly [61]. It is a well-known and accepted fact, that a correct mathematical description of the physical properties of a given material system is indispensable for the successful application of process simulation [32] and further, more advanced model-based tasks applied for synthesis, design and optimization of chemical process systems [16, 18]. Undoubtedly, the big success of commercial flowsheet simulators in industrial applications is highly related to the availability of extensive physical property databases and thermodynamic models integrated therein. In [66] Gernaey et al. claim that the low impact of generic tools such as MATLAB/Simulink and Modelica in dynamic process simulation is primarily due to their limited ability to access thermodynamic and physical property data.

From a practical perspective, physical property calculations can appear in similar forms as discussed above in Section 2.1.1, either as closed-form or open-form implementations.

Closed-form implementations commonly available as physical property subroutines or calls, not only offer the advantage of allowing re-use of validated code from thermodynamic models and extensive physical property databases, but also permit the use of tailored solution algorithms for the evaluation of physical properties. An example of this last point is given

[5] Closed form formulations for dynamic systems are discussed by Schopfer [185].

through the evaluation of properties calculated from cubic EoS models, where algorithmic implementations directly allow the evaluation of the properties of a desired phase. Disadvantages of closed-form implementations are given by their tool dependency, since some interface may be required for the physical properties calls to interact with the remaining part of the model, and, in many cases, by the lack of exact/analytic derivative information [61]. The evaluation of derivative information of the physical property functions may have a critical importance for the successful application of model-based techniques on models containing the physical property calls.

Open-form implementations of physical property models, on the other hand, can directly be implemented as a sub-model within the overall model using the same modeling language (and hence without necessity of further interfaces), and make it relatively simple, to obtain derivative information of first and higher order. Besides the positive effects on the implementation and generation of derivative information, general difficulties common to all open-form implementations also appear: depending on the implemented model, not only the size of the overall mathematical model can grow significantly with the additional equations, but also its grade of nonlinearity and non-convexity. The higher number of equations is additionally linked with a higher number of variables, for which good initial values/guesses may be indispensable. As usual, general solution algorithms work the best when the initial guess lies near the solution, so that proper initialization strategies are required. These latter, however, can usually be best obtained based on some specific algorithm. A further important aspect of open-form implementations is given by the algebraic complexity of the model and the quality of the available sources. Transferring a published model can be a highly error prone task.

2.1.3 Evaluation of implementation strategies

From the aforementioned strategies, the integration of closed-form physical property calculations or thermodynamic models is nowadays considered as the most efficient approach when it comes to the robust process *simulation* of rigorous process models [204]. Accordingly, this strategy is not only supported by the very successful sequential-modular modeling tools such as Aspen Plus [11] or CHEMCAD [36], which rely on closed-form implementations, but also by all major commercial domain specific equation-oriented modeling tools such as gPROMS [170] or Aspen Custom Modeler (ACM) [10]. These latter provide interfaces for physical property calls and even supply own physical property databases or access to well-known databases. Standardization efforts significantly simplify the workflow required for integrating closed-form implementations of physical property calls. For example, using the IK-CAPE interface [50] or CAPE-OPEN property packages, see also

Section 2.2.2 for more details on the latter, it is possible to integrate physical property packages of different sources within different process modeling tools. It should be remarked that such packages not only provide function values for physical properties and thermodynamic state functions. Derivatives of these functions with respect to temperature, pressure and compositions can be provided as well.

Motivated by the trend towards large-scale equation-oriented optimization [43] and since modern large-scale optimization algorithms are designed to work the best with fully open, equation-oriented models with exact first order and second order derivatives [17], recently, significant progress has been done towards the efficient solution of classical thermodynamic models implemented as open-form models. As an example, Kamath et al. [95] present an efficient equation-oriented model formulation for the solution of cubic EoS models, which, among others, has been implemented in the framework for large-scale flowsheet optimization described by Dowling and Biegler [43]. Despite this progress, it is not yet evident, whether these types of approaches based on open-form formulations of *complex* models will prevail or whether other techniques will promote the use of closed-form formulations. As for today, the considerations presented by Tolsma et al. [204] back in 2002 regarding equation-oriented modeling are still valid: *"the use of external procedures within software packages will be necessary for quite some time"*.

A further important challenge is related to the fact that many complex physical property or thermodynamic models are not available through any software package. In light of the continuously growing number of promising models of continuously growing complexity,[6] it becomes important to find or provide means that allow modelers to develop efficient model implementations with low implementation effort. Besides the high algebraic complexity of those models, which is almost inevitably linked with an error prone, tedious coding, there is an additional important aspect that may limit or restrict the usability of the models. Whether a model is available as closed-form or open-form implementation, there is usually a tool dependency. Model implementations coded in a particular programming or modeling language can usually only be used with that language. Model reuse within other environments would usually imply recoding, that is redundant modeling, or the creation of additional interfaces.

2.2 Tool-Independent Implementation Approaches for Open-Form Models

Since the early days of process simulation, when problem specific subroutines were implemented using general-purpose, high level programming languages such as Fortran or

[6] Considered for example the molecular based EoS models discussed in section 2.4.4.

C, the need of providing better support for model development and management has led to the development and establishment of several powerful generic and domain specific software tools. Though these latter have contributed significantly to the acceptance and further development of model-based methods within the field of PSE, tool specific concepts make it difficult or impossible to exchange models across different software platforms and model-based applications [219]. Standardization efforts are an important contribution to overcome these difficulties. Regarding open-form model implementations, most commonly used for customized user models, these efforts are either based on the modeling language Modelica, MathML/XML technologies, or CAPE-OPEN's Equation Set Object (ESO). These concepts are treated in the next subsections. A further important issue that significantly simplifies the model reusability is given by the model documentation. Documentation-based modeling approaches are discussed in Section 2.3.

2.2.1 Modelica

Pursuing the target of standardizing development, reuse and exchange of dynamic system models and model libraries, and based on the experiences gained through the development and use of numerous generic (also called domain-neutral or multi-domain) modeling tools (Dymola, ObjectMath, Smile, etc.), Modelica has emerged as a comprehensive domain-neutral, object-oriented modeling language for physical systems [58].[7] Modelica supports several modern concepts, such as object orientation, equation-based modeling, and hierarchical structuring. If elementary model components are available, either as own development or obtained through model libraries, complex compound models can be easily built by either browsing, dragging, dropping and connecting icons together, or by use of proper language elements. Extensive libraries of several domains permit easy reuse and development of interdisciplinary models including, among others, mechanical, hydraulic or electrical model elements.

The high popularity of Modelica has also been promoted through the availability of several academic and commercial modeling and simulations environments. Some of the most popular being OpenModelica, JModelica.org and Dymola. A central task of these environments is the compilation of the models. Classical compiler techniques can drastically reduce the complexity of user defined models to allow an efficient model evaluation within simulations or optimization. However, it should be considered that this characteristic represents a tool-dependent issue. Not all modeling environments have equally strong symbolic engines or compilers.

[7] The Modelica language is maintained and developed by the Modelica Association, which also provides the open source Modelica Standard Library.

Despite its continuously growing popularity, in comparison with other domains, Modelica has had a relatively low impact in rigorous process modeling as done in process systems engineering or computer-aided process engineering. As already mentioned in Section 2.1.2, an important reason for this fact may be related to challenges regarding the implementation/integration of the calculation of multi-component fluid properties and phase equilibria. Though some research has been and is being done in this direction [e.g. 179, 207], in this regard, other comparable, domain specific tools such as ACM or gPROMS are still clearly ahead.

2.2.2 CAPE OPEN's Equation Set Object

CAPE-OPEN is an industry standard for interoperability between process simulation software [211]. It defines standardized interfaces that enable the exchange and use of process modeling components across process modeling tools from different vendors or developers. While the most plausible exchangeable process modeling components in the context of process simulation, *unit operation models*, *thermodynamic/physical properties models*, and *numerical solvers*, are the only components considered by the early version 0.93 of the standard [37], the version 1.0 additionally considers further components, such as *chemical reaction*, *optimization*, *parameter estimation and data reconciliation*, among others [38], and is thus more suitable for more advanced applications.[8] For a detailed insight on the motivation and development process of the CAPE-OPEN standard, starting from the EU-funded project of the same name, towards the follow up organization CO-LaN [199], following sources are recommended [15, 26, 211, 218].

In spite of the diversity of process modeling components supported by CAPE-OPEN interfaces in version 1.0, most implementations and applications are still restricted to interfaces for unit operations and thermodynamics [see 211]; for exchange and reuse of models commonly available as *closed-form* implementations. Regarding *open-form* model implementations, the Equation Set Object (ESO), a component of the *Numerical Solvers* [14, 157], constitutes a software abstraction of general AE or DAE systems, as required for the numerical solution of equation-based models with generic AE solvers or DAE integrators. The standard ESO interface gives access to the structure of the system, among others, the number of equations and variables, and permits the evaluation of the residuals and partial derivatives of the equation system.

Though the primary aim of the ESO concept was the support of exchangeable, CO compliant general *Numerical Solvers* as developed by Belaud et al. [14] or Schopfer et al. [186], it

[8] The only difference between CAPE-OPEN standard version 1.0 and the most recent version 1.1 concerns changes in the interface for *Thermodynamic and Physical Properties*.

has become the core concept that enables ESO supporting modeling environments, such as gPROMS [170], to act as *model servers*. Accessing the model structure through the ESO interface, algorithm developers can test their programs on advanced model implementations originally developed using the advantages of gPROMS [21, 116, 121, 186]. Schopfer [185] presents examples on modeling tool integration of equation-based models accessible through the ESO interface. In spite of its advantages, the CAPE OPEN's ESO is very rarely used. This may be due to the fact that the adaption of a model implementation to fulfill the ESO interface is a highly complex task.

2.2.3 XML/MathML technologies

In recent years, the eXtensible Markup Language (XML) and further Markup languages derived from it have become the key technology for *systematic* exchanging of structured data in the internet. They enable vendor-neutral, software-neutral, version-neutral ways of storing information [103] and are hence particularly helpful for collaborative work. Also, in the field of Chemical Engineering XML-based technologies have led to the development of several languages that clearly enhance the dissemination and reusability of important data. Some interesting examples include, among others, thermoML [57], an approach for storage and exchange of thermophysical and thermochemical physical property data, and PlantData XML [196], an electronic data exchange standard for the process industry. Efforts to extract the domain specific knowledge of modeling of chemical process systems into an XML-based language led to the development of CapeML [217]. CapeML is an XML-based domain-specific language for describing model equations in process engineering [163]. Its main goal is to support the development and reuse of hierarchical model structures.

The Mathematical Markup Language (MathML) provides means to describe the representation (presentation MathML) and meaning (content MathML) of mathematical expressions, as they appear in model equations of open-form model implementations. Zerry et al. [228, 229] discuss the feasibility of exchanging equation-oriented process models coded in content MathML. An important point of criticism regarding the use of MathML as basic technology for model implementation is given by the fact that it does not suit the engineering needs to create hierarchically structured models as required for complex models consisting of several components [163]. In fact this issue motivated the development of CapeML. Alternatively, as extensively discussed in Kuntsche [113], the introduction of modular modeling elements such as notations and connectors (notation translators) can be used to build highly complex models while keeping MathML as the core technology. Besides modularity concepts, Kuntsche introduced concepts that enable the use of XML based documentation-based models. These are discussed in more detail in Section 2.3.1.

2.3 Integration of Model Documentation in Modeling Tasks

Though tool independent model implementations, as discussed above, significantly extend the reusability of equation-oriented (open-form) models, important shortcomings, common to all types of open-form implementations, are still present. As pointed out by Marquardt in 1996 [125] "*equation-oriented languages do not assist the user in developing models from using engineering concepts, nor is there support for the documentation of the modeling process during the life cycle of a process or for proper design and documentation of the model library. Reuse of validated unit models by a group of simulator users is therefore almost impossible and redundant modeling is unavoidable*". While the first aspect, assistance in model development from engineering concepts, is particularly helpful for beginners in the field of modeling, it may conflict with creativity, a fundamental element of the modeling process. Besides that, a restriction of engineering concepts to a particular field, as done in ProcessModeller [168] with concepts limited to the field of Chemical Engineering, may make it impossible to develop empirical or even semi-empirical models [94].

The second aspect regarding the missing support of documentation within the life cycle of a mathematical model is a central problem that not only complicates the model reuse (even when using the same modeling language or tool) but the application of model-based methods in general. Detailed comments on the need for and challenges of model documentation in the modeling process are given in [113]. It is for this reason that several approaches have been developed in order to integrate documentation steps within modeling tasks.[9]

A comprehensive approach for integration of documentation in the process modeling life cycle is discussed in [24] for the ModKit modeling tool. A more common approach to keep consistency between model implementation and documentation consists on putting both elements in one and the same document. This approach, also known as *in-line documentation* [113], is supported in engineering math software as PTC Mathcad [171] or common computer algebra systems (CASs) as Maple [122]. In those tools, mathematical commands can be evaluated interactively in a worksheet and enriched with documentation elements such as text, tables and images. Recently this type of applications has become very popular through free web platforms such as IPython Notebooks [162] or more recently SageMathCloud [177], which additionally offers novel collaborative computing capabilities through cloud computing. In spite of the success of these approaches, which is particularly big in academia, in most cases they are not suited to address the requirements common to industrial-scale process systems modeling, for example the call of external physical

[9] The simplest method for model documentation through comments on some code file is too unstructured and unspecific to be treated here in more detail.

property models, or support of the construction of large, hierarchical models.

Despite the great advantages of this type of approaches regarding consistent model implementations and documentation, these approaches do not resolve the challenges previously discussed in Section 2.1. Tools like PTC Mathcad and Maple work with proprietary formats and cannot simply communicate with other arbitrary tools. Additionally, the workflow supported by the corresponding worksheets is rather appropriate for sequential evaluation of single commands and hence not so proper for declarative model formulation, as commonly required for flexible use of model-based applications. Furthermore, it is difficult to reuse existent model elements to build complex compound models which often appear in PSE applications. It can be concluded that approaches are required that resolve tool dependency issues in conjunction with a consistent model documentation while considering peculiarities common to PSE (or CAPE) applications.

An interesting approach that fulfills most of the previously mentioned points, but the tool dependency, is presented by Alloula et al. [5]. They show the explicit integration and interaction of CAPE software with model documentation available as a Microsoft Word file. Kuntsche et al. [115] introduced a novel approach that combines a highly consistent model documentation with code generation in any arbitrary programming or modeling languages. The modeling process takes place on the documentation level [113]. Concepts common in process modeling are supported as well. In this thesis, models developed based on the concepts by Kuntsche are denominated as documentation-based models. A more detailed presentation of these concepts and its current software realization is given in Section 2.3.1.

2.3.1 Documentation-based models - Modeling on the documentation level

Following the definition introduced in [135], *"a documentation-based model is a model that guarantees consistency between the language-specific implementation and its documentation. A language-specific realization can be extracted directly from a general (tool independent) model documentation"*. A crucial feature of this approach is the capability of generating (source) code out of model equations given in a documentation format.

Motivated by new possibilities for modeling tasks in collaborative and distributed environments using web-technologies, Zerry et al. [229], [228] showed the feasibility of using *content MathML* for coding and exchanging equation-based models over the internet in a *proof-of-concept* software implementation. Executable model implementations could be obtained out of the MathML model definition through automatic code generation of Fortran or Java code. With his efforts of further reducing the disparities between general model documentations and implementations Kuntsche [113] went one step further. He recognized

the accepted common practice of separating meaning and presentation of mathematical content as a potential problem. As a simple example, consider that for people not familiarized with common notations used in Chemical Engineering it is not clear whether the term LV in the variable p^{LV} corresponds to an exponent or to a superscript. The fact that *"correctly specified mathematical content can be specified in an ambiguous way"* [113], can lead to differences between model documentation and implementation.

As a solution to this problem Kuntsche et al. [113, 114, 115] introduced an unambiguous mathematical notation as a mandatory model component. By these means it is possible to obtain consistency between documentation and model implementations working with a subset of presentation MathML. Fully usability of consistently documented models in arbitrary modeling or programming languages is guaranteed with the additional concept of user defined language specifications. This concept allows modelers to generate code to any arbitrary language supporting equations with common mathematical operators.

Documentation-based models are defined in a declarative way, typical for equation-oriented modeling. Accordingly, the most straightforward approach for code generation results in an open-form model implementation that can be directly used for any type of model-based application. Additionally, documentation-based models are fully domain-independent and applicable for any type of model, either rigorous, semi-empirical or empirical, as long as the models can be expressed by equation systems.

Based on the previously discussed ideas and further modularity concepts suitable for the development of large compound systems, Kuntsche developed the web-based modeling, simulation environment MOSAICmodeling described in [113]. Since then MOSAICmodeling has further evolved to a web-based collaborative platform for integrated model and data management [111, 112, 132], that also supports the formulation and solution of optimization problems as well. Some characteristics of the current implementation relevant to this thesis, including the limitations, are discussed in Section 2.3.2.

2.3.2 Implementation in MOSAICmodeling

In the following, an overview of some aspects and limitations of the current MOSAICmodeling implementation, that are important for this thesis, are given. A first comparison of MOSAICmodeling against other tools developed in academia as for 2011 is provided by Heitzig [83]. It should be noted that almost all limitations of MOSAICmodeling listed in [83] have been remedied in later MOSAICmodeling versions. A more recent comparison of MOSAICmodeling with other contemporary modeling tools is given in [132]. In spite of current limitations listed below, at its current state of development, MOSAICmodeling

can be seen as a suitable tool for large-scale equation-oriented flowsheet optimization, a current hot topic in PSE [43].

Supported mathematical notation and operators

A central aspect to keep the proximity between model documentation as common in scientific papers and model implementation is to allow the definition of so-called two-dimensional variable names when coding the model. An example of two-dimensional variable names is $c_{p,i,tr}^{L,m}$, which can be used to denote the mass specific (superscript m) heat capacity at constant pressure (base name c with subscript p) of the liquid component number i in tray number tr (superscript L and indices i and tr). In this example the corresponding one-dimensional variable name could be `C_Lm_pitr`, which is certainly not as good readable as the two-dimensional one. By comparing the full variable name against the mandatory notation, the meaning of the variable can be easily extracted by the reader. Since the variable naming conventions used in scientific papers are often ambiguous, a convention for variable naming is necessary. The Association of German Engineers (German: Verein Deutscher Ingenieure (VDI)) and the Working Party on Distillation, Absorption and Extraction of the EFCE introduced in the VDI directive 2761 an unambiguous standard notation for mathematical modeling of thermal separation processes. This notation is available in MOSAICmodeling.[10]

Besides the aforementioned example regarding exponents and superscripts, a mechanism is needed to distinguish subscripts from indices. While subscripts represent an unchanged part of the name, for example p to distinguish c_p from c_v (the specific heat capacity at constant volume), indices are used to denominate elements of a given set, which are characterized by different values of i. Details on the implemented variable naming convention supported by MOSAICmodeling are provided in [115].[11]

Besides the naming definitions, a clearly defined set of mathematical operators is supported in order to write the model equations. Besides the standard (binary) arithmetic operators, the most common mathematical functions, such as sine, cosine, exponential function, natural logarithm, exponentiation or summation are supported [see 115].

There is a large number of mathematical functions that may appear in engineering models but are not directly supported by MOSAICmodeling. For arbitrary variables χ, $\psi \in \mathbb{R}$

[10] It should be remarked, that the VDI directive 2761 has been withdrawn. A comparable notation for a wider field of sciences has been introduced in DIN 1304.

[11] While this convention ensures clarity in the variable naming, it may prohibit well defined variable names as available in the literature. For example instead of writing $c_{p,i,tr}^{L,m}$, a modeler could write $c_{p,m,i,tr}^{L}$. The latter name is, however, not supported by MOSAICmodeling. While an arbitrary number of superscripts and indices are permitted, pro variable only one subscript is allowed.

and the vector $\mathbf{z} \in \mathbb{R}^{n_\mathbf{z}}$ consider for example $\sinh(\chi)$, the absolute value $|\chi|$, $\max(\chi, \psi)$ or $\prod_{i=1}^{n_\mathbf{z}} z_i$. The limitation to a basic set of operators/operations is related to the fact, that not every language for which code generation is supported, supports these operators and functions. Furthermore, some of those functions, for example (max) or the absolute value are not smooth and are best implemented with some type of smooth approximation in order to avoid possible convergence problems. This later point is particularly important for optimization applications with gradient based methods.

Considering the conventions for variable namings and the supported operators, relatively complex, but still highly readable models can be formulated with low effort using a clearly defined subset of the LaTeX mathematical notation called MosaicLatex [113]. Though relatively complex models, such as those of multiple-stage separation systems can be easily written down with the supported MOSAICmodeling conventions, the algebraic input capabilities supported by MOSAICmodeling are clearly limited in comparison to mathematical programming languages such as GAMS [29] or AMPL [54]. Without doubt, the high flexibility regarding the algebraic input, including flexible definition of sets and numerous set operators ($\cup$, $\cap$, $\subset$, $\not\subset$) is one of the main reasons for the popularity of the aforementioned languages. The introduction of set operations in the model definition not only enhances the model readability but can also help to reduce the implementation effort and perhaps the model size.

Regarding the variable naming conventions currently implemented in MOSAICmodeling, there is an important difference in the way indices are evaluated. While in classical, matrix-based notations the order of the indices is enough to interpret the meaning of the variables, in MOSAICmodeling this is not the case. In MOSAICmodeling the index becomes part of the name and the meaning follows from the name. To keep the aforementioned example, in order to express the mass specific heat capacity of the liquid component 2 in tray number 3 with MOSAICmodeling's conventions the variable can be written either as $c^{L,m}_{p,i=2,tr=3}$ or as $c^{L,m}_{p,tr=3,i=2}$. On the other hand, note that in a classical matrix-based notation in general $c^{L,m}_{p,2,3} \neq c^{L,m}_{p,3,2}$.

Because of these peculiarities, particular care should be taken when transferring equation systems with multiple indices defined with classical notations into MOSAICmodeling. Expressions with many indices may need to be implemented in a different order to represent the same information as in the classical matrix-based representation. Recently, some efforts have been made to expand the algebraic input limitations available on indices [25].

Supported model types

As previously mentioned in Section 2.1.1, for a modeling tool or language to treat most model-based application in process systems engineering, it is enough to offer support for the formulation of differential algebraic equation systems (DAEs). MOSAICmodeling supports the direct implementation of semi-explicit DAEs as presented in Eq. (2.1). Recent further developments are concerned with means to directly implement PDAEs, that can be automatically discretized with some user-defined discretization method [49], hence leading to larger scale algebraic equation systems. Having the possibility of entering models as IPDAEs, it results obvious that with the considered documentation-based approach it is possible to input model equations, describing physical phenomena at any scale. However, it should be noted that there is no direct support for entering multiscale models directly. Prototypical tools for realizing such conceptual models and execution of those simulations are presented by Zhao et al. [231].

Code generation

Within the context of this thesis *code generation* is understood as the capability of transforming some kind of computer-based mathematical representation into a code that can be executed or evaluated for solution purposes (e. g. for simulation or optimization). In this sense, any *platform and solver independent modeling tool* coupled with a modeling language intrinsically needs some kind of code generation that transfers the mathematical representation defined by the modeling language into the requirements required by a particular solver. Code generation is hence a key feature of systems supporting the Modelica modeling language, or general algebraic mathematical programming languages like those provided with the GAMS or AMPL environments. Modelica models, independently of the way they have been defined by the users, are flattened[12] and usually translated into C files that are compiled to perform efficient simulations [58]. The AMPL system converts the mathematical model defined by its problem formulation language into so-called .nl files, which are read by AMPL's solver-interface library for evaluation purposes [54]. GAMS works in a similar way.

An important part of the code generation and evaluation of equation-based model implementations, in particular for large-scale models, is concerned with performing model structure analyses, simplifications and symbolic manipulations. Among others, these are intended to reduce the problem size and complexity of the models and to enhance the

[12] A flattened Modelica model, denotes a model implementation in which all Modelica object-oriented features, such as hierarchies and connections, are removed so that only a purely mathematical representation as equation system remains. In MOSAICmodeling's terms it corresponds to an instantiated model.

robustness of the solution methods. This process is known as preprocessing or presolving and includes several techniques such as elimination of trivial or repeated equations and variables, identification of nonlinear subsystems that can be solved sequentially, and identification and separate solution of linear sub-systems of equations, among others (see for example [46, 65]). A disadvantage of implementations of these model analysis techniques is given by the fact that these either work on vendor-specific formats, generated by vendor-specific environments, or are part of vendor specific compiler techniques. Though most analyses are certainly based on well-known and available algorithms and codes, reimplementation and interfacing is not a trivial task.

Compared with the modeling environments discussed above and other popular modeling environments, MOSAICmodeling's implementation of code generation is much richer since it is not limited to environment dependent solver interfaces, but it can cover any language that supports the concept of equations. MOSAICmodeling's code generation creates by default a text-based model implementation in a scalar format, that means a model in which all indexed equations and variables are expanded (or flattened) and all variables and equations can be seen as scalar contents. A comparable code generation is also done by the CONVERT solver available in GAMS. A general advantage of scalar formats is that they permit an easy translation into other languages, since no special concepts or set operators, that could appear in the original model definition, need to be translated separately. Considering that MOSAICmodeling's model formulation does not support advanced features for constructing and manipulating sets, this advantage has the potential to be exploited in a more efficient way [25].

When translating MOSAICmodeling's mathematical representation into the corresponding *code* of other modeling environments, e. g. Modelica, GAMS or AMPL, it is possible to use the model analysis tools inherently available in these environments, for solution purposes, so that it may not appear to be necessary to implement these analyses in MOSAICmodeling. However, in order to support modeling languages and environments that do not offer these capabilities intrinsically, for example C or C++ code linked with numerical libraries such as C-GSL or BzzMath [30], and to support alternative methods to exploit the structural information of the model, it seems useful to realize these analysis at MOSAICmodeling's tool independent level.

In this context, Kraus [111] introduced in MOSAICmodeling methods that exploit the structural information of the model to allow for a more efficient code generation. Some applications are discussed in [28]. These methods work based on decomposition algorithms which are applied on the incidence matrix of the equation system, the block diagonal (BD) decomposition and bordered block diagonal (BBD) decomposition. The BD decomposition decomposes a given equation system into sets of equation systems that can be solved

sequentially or provides information on structural problems that result from wrong (design) specifications. The BBD decomposition recognizes key equations and key variables which prevent a higher degree of decomposition, hence it can help to recognize potential *tear* variables and equations, that may help to automatically generate efficient initialization strategies or nested solution approaches.

2.3.3 Extension on documentation-level concepts discussed in this thesis

The previous discussion has shown that there is a great potential to extend the application of documentation-based models. This potential is seen through the chance of generating algorithmic evaluation methods obtained out of the basis of the structural information present in the open-form model implementation. Considering that algorithmic evaluations or closed-form models still play a major role in PSE model-based applications, in particular when working with rigorous process models, and that those evaluations can be obtained out of open-form formulations, it seems reasonable to explore more possibilities of obtaining efficient closed-form formulations out of equation-oriented formulations.

This thesis evaluates the application of structural analysis features together with algorithmic differentiation techniques in order to offer an efficient code generation of advanced molecularly based EoS models with a low implementation effort. The resulting approach can be seen as a means of providing standard (reference) model implementations with a consistent model documentation [84] and shows a possible way to extend the use documentation-based models in rigorous process simulation applications.

An additional extension to the concept of documentation-based models that was intensively evaluated, but not developed within this thesis consists on the integration of external function calls in documentation-based models. Though the integration of external physical property calls into equation-oriented formulation represents the state-of-the-art approach supported by domain-specific process modeling environments, these concepts were originally not available in the basic conception of modeling in the documentation level [113] limiting significantly the suitability of this type of models for process modeling applications.

2.4 Thermodynamic Fundamentals

As previously remarked in the introduction, one of the main objectives of this thesis is to provide improved ways to efficiently implement advanced thermodynamic equation of state models for their use in rigorous process modeling applications. This section summarizes the most important aspects regarding the calculation of thermodynamic properties and

phase equilibria with equation of state models as required for engineering applications. Important facts regarding challenges of modeling of multiphase liquid systems and the current state of the art equation of state models are discussed as well.

Contribution of classical thermodynamics

Classical thermodynamics provide the fundamental relations required for the evaluation of properties of pure components and mixtures. Based on *fundamental property relations* (Smith et al. [192]) it can be shown that the corresponding *fundamental equations* completely describe the thermodynamic state of a system [194]. As an example, for a general open system Eq. (2.5) and Eq. (2.6) show the fundamental property relations of the internal energy U

$$\mathrm{d}U = T\mathrm{d}S - p\mathrm{d}V + \sum_i \mu_i \mathrm{d}n_i, \tag{2.5}$$

and the Helmholtz free energy A

$$\mathrm{d}A = -S\mathrm{d}T - p\mathrm{d}V + \sum_i \mu_i \mathrm{d}n_i, \tag{2.6}$$

where T is the temperature, S the entropy, p the pressure, V the volume, μ_i the chemical potential of component i and n_i the number of moles of component i. The corresponding fundamental equations are mathematical expressions of the form $U = f(S, V, \mathbf{n})$ and $A = f(T, V, \mathbf{n})$ respectively. Given a fundamental equation $A = f(T, V, \mathbf{n})$ for a system characterized by fixed values of temperature T, volume V and mixture mole numbers $\mathbf{n}$, all other thermodynamic state variables of that system, such as pressure p, specific heat capacity at constant volume c_V, enthalpy h, entropy S, etc. can be *easily* obtained (see Section 2.4.1).

Classical Thermodynamics also provide the fundamental relations required for the calculation of phase equilibria. In particular, following necessary conditions for phase equilibrium result [144, 167, 192]:

$$T^{(j)} = T^{(\alpha)}, \qquad j \neq \alpha \tag{2.7}$$

$$p^{(j)} = p^{(\alpha)}, \qquad j \neq \alpha \tag{2.8}$$

$$\mu_i^{(j)} = \mu_i^{(\alpha)}, \qquad j \neq \alpha \text{ and } i = 1, \ldots, NC. \tag{2.9}$$

In the equations above the superscript in parenthesis (e.g. (j)) refers to the phase j. The chemical potential μ_i is given either by Eq (2.10)

$$\mu_i = \left(\frac{\partial U}{\partial n_i}\right)_{S,V,n_j}, \tag{2.10}$$

or from the corresponding equivalent equations that result from the respective derivatives of the thermodynamic state functions enthalpy H, Gibbs free energy G, or the Helmholtz free energy A as shown in Eq. (2.11).

$$\mu_i = \left(\frac{\partial H}{\partial n_i}\right)_{S,P,n_j} = \left(\frac{\partial G}{\partial n_i}\right)_{T,P,n_j} = \left(\frac{\partial A}{\partial n_i}\right)_{T,V,n_j} \tag{2.11}$$

Though in principle the conditions required for the calculation of phase equilibria are already given through the equality of the chemical potentials (Eq. (2.9)), for practical, among others numerical reasons [165, 192], it is convenient to define the equilibrium conditions through the equality of an equivalent thermodynamic property, the fugacity of the component i, denoted f_i. Accordingly, for practical applications Eq. (2.12) is used instead of Eq. (2.9). Approaches for calculation of the fugacity are summarized in Section 2.4.2.

$$f_i^{(j)} = f_i^{(\alpha)}, \qquad j \neq \alpha \text{ and } i = 1, \ldots, NC. \tag{2.12}$$

Classical thermodynamics provide the necessary (Eq. (2.7)) and sufficient conditions for phase equilibrium, and even provide relations to calculate the chemical potential, and further thermodynamic state variables, that could *easily* be derived from a fundamental equation such as $A = f(T, V, \mathbf{n})$ or $G = f(T, p, \mathbf{n})$. However, classical thermodynamics do not provide analytic expressions for these fundamental equations for the description of real matter. These depend on the molecular behavior of matter and are therefore related to Molecular Physics and Statistical Mechanics.[13]

Reference state selection and residual properties

Within the calculation of thermodynamic state variables, the focus lies on changes of the values of the variables and not on their absolute values. Since the choice of a reference state is a matter of convenience, it appears practical to choose the ideal gas state (marked

[13] The efforts towards the description of the behavior of real matter can be mainly categorized in EoS approaches ($A(T, V, \mathbf{n})$) and activity coefficient models ($G(T, p, \mathbf{n})$) [194]. In this work the focus lies on the EoS approach.

by *) as a reference state. According to Mollerup and Michelsen the Helmholtz free energy of an ideal gas mixture $A^*(T, V, \mathbf{n})$ is given by Eq. (2.13)

$$A^*(T, V, \mathbf{n}) = \sum_i n_i \left(\mu_i^*(T, p_0) + RT \log(\frac{n_i RT}{p_0 V}) - RT \right), \tag{2.13}$$

where p_o is a fixed reference pressure. Having an expression for the Helmholtz free energy of an ideal gas mixture the remaining task is providing an expression for the difference between the Helmholtz free energy of the real mixture and the Helmholtz free energy of the ideal gas mixture under similar conditions $(T, V, \mathbf{n})$. In general, such a difference between a real mixture property and the equivalent ideal gas mixture property is denoted a residual property. The residual Helmholtz free energy is defined by Eq. (2.14).

$$A^{res}(T, V, \mathbf{n}) = A(T, V, \mathbf{n}) - A^*(T, V, \mathbf{n}) \tag{2.14}$$

As pointed out by Mollerup and Michelsen (Page 60): *"The expressions for the residual Helmholtz (free) energy is the key equation in equilibrium thermodynamics because other residual properties are calculable as partial derivatives in the independent variables T, V, and n"*. The next section draws on engineering approaches towards the derivation of such expressions.

Approaches for description of real matter

Initial efforts for the description of real matter, in particular regarding the behavior of real fluids, focused mainly on the description of volumetric behavior; in the development of quantitative relations between pressure, volume and temperature at given composition, also known as pressure-volume-temperature (PVT) relations. As shown by Eq. (2.15), based on an analytic expression of the form $p = f(T, V, \mathbf{n})$ or $V = f(T, p, \mathbf{n})$), it is possible to obtain an analytic expression for the residual Helmholtz free energy of real fluid (mixtures)

$$A^{res}(T, V, \mathbf{n}) = -\int_\infty^V \left(p - \frac{n RT}{V'} \right) \mathrm{d}V' = \int_0^p \left(V - \frac{n RT}{p'} \right) \mathrm{d}p', \tag{2.15}$$

where n, the mole number, is given by the sum of all elements from vector $\mathbf{n}$. It is also common to formulate or find PVT relations in terms of the independent variables $(T, \hat{\rho}, \mathbf{x})$, hence as pressure explicit equation of the form $p = f(T, \hat{\rho}, \mathbf{x})$, where $\hat{\rho} = n/V$ is

the molar density. Considering the compressibility factor $Z = P/(\hat{\rho}\,R\,T)$ the corresponding expression for the residual Helmholtz free energy is given by Eq. (2.16).

$$A^{res}(T, \hat{\rho}, \mathbf{x}) = n\,R\,T \int_0^{\hat{\rho}} \frac{Z(T, \hat{\rho}, \mathbf{x}) - 1}{\hat{\rho}} \mathrm{d}\hat{\rho} \tag{2.16}$$

Among the large amount of available PVT relations semi-empirical cubic equations of state, such as the Soave-Redlich-Kwong [193] or the Peng-Robinson EoS models [158] have a distinguished position. Being relatively simple analytic expressions applicable to high-pressures and hydrocarbon-gas mixtures, cubic EoS were and are still nowadays an omnipresent working tool for the oil and gas industry [84]. Also in the chemical industries, not least because of advances in mixing rules [42, 210], cubic EoS have had a very important role. However, regarding macromolecular components or systems with strong intermolecular interactions, as they appear with strong polar or associating components and their mixtures (e. g. in complex multiphase liquid systems), cubic EoS seem to have reached the limit of their applicability.

As for today, Statistical Mechanics provide the means for the development of proper engineering equation of state models able to describe more complex systems. On the basis of the description of the molecular structure of a fluid and the interaction between a relative low number of molecules, analytic expressions for the Helmholtz free energy are obtained, that can then be used to describe the properties of macroscopic systems [42].[14] These resulting analytic expressions for the residual Helmholtz free energy are *commonly* related to the total number of molecules N (yielding A^{res}/N) and *commonly* use $(T, v, \mathbf{x})$ or $(T, \rho, \mathbf{x})$ as independent variables. $v = V/N$ is the extensive volume divided by the total number of molecules, $\rho = 1/v$ is the corresponding total number density of molecules. The expressions for the Helmholtz free energy are commonly published as dimensionless, reduced residual energy $\tilde{a}^{res}(T, \rho, \mathbf{x}) = A^{res}/(N\,k\,T)$, where k is the Boltzmann constant.

2.4.1 Calculation of thermodynamic properties with EoS models

Depending on the origin and implementation of EoS models, different model representations are often found in literature. Mollerup and Michelsen [140, 144, 145] clearly recommend and provide the formulation and implementation of Helmholtz free energy models with the independent variables $T, V, \mathbf{n}$ and even recommend in general to avoid molar fractions $\mathbf{x}$ as independent variables. According to them, derivatives with respect to molar fractions miss many important symmetry properties, lead to a higher risk of errors and inconsistencies in the model implementation and in addition, lead to more complex expressions. On the other hand, Prausnitz [167] recommends the set of independent variables

[14] The resulting EoS models are therefore called molecularly based equation of state models.

$(T, \hat{\rho}, \mathbf{x})$ due to considerable computational simplification which is discussed in detail in [205]. Topliss [205] recognizes the variable set $(T, V, \mathbf{n})$ as convenient for defining thermodynamic relations but claims that the implementation effort for a specific EoS model and its derivative is unnecessary high in comparison with implementations based on the set of independent variables $(T, \hat{\rho}, \mathbf{x})$. Although these very well-known groups come to contrary recommendations, they both agree in the fact that the exploitation of common subexpressions is the key to an efficient computational implementation of thermodynamic models.

For both variable sets important derivatives for the generation of the mostly used thermodynamic properties are available in the literature. For $(T, V, \mathbf{n})$ in [144] extensive tables of derivatives of A^{res} and $A^{res}/(RT)$ are provided. In [205] thermodynamic properties are obtained as derivatives of expressions for the dimensionless reduced residual Helmholtz free energy $\tilde{a}^{res} = A^{res}/(N\,k\,T) = A^{res}/(n\,R\,T) = \hat{a}^{res}/(R\,T)$. Though in [205] the variables $(T, \hat{\rho}, \mathbf{x})$ are considered as independent variables the relations are valid as well when considering ρ (the total number density) instead of $\hat{\rho}$ (the molar density).

In the remaining of this section, it is shown how some important thermodynamic properties can be derived from expressions for the reduced residual Helmholtz free energy $\tilde{a}^{res}$.[15] In accordance to most formulations found in literature, the considered expressions are written either as $\tilde{a}^{res}(T, \hat{v}, \mathbf{x})$ or as $\tilde{a}^{res}(T, \rho, \mathbf{x})$. Since the thermodynamic properties are derived from a residual property, the presented properties also refer in most cases to residual properties. The contributions of the ideal gas mixture, which is additionally required to calculate the properties of real mixtures, can be evaluated according to relations provided in standard textbooks on physical properties or thermodynamics [e.g. 172, 192].

Further, it should be noted that the focus lies on the resulting expressions required for the evaluation of thermophysical properties and phase equilibrium calculations and not on where these relations come from. For more details on the derivations of the presented equations consider e. g. [72].

Pressure and compressibility factor

The pressure can be directly obtained from the fundamental relation for the Helmholtz free energy (Eq. (2.6))

$$p = -\left(\frac{\partial A}{\partial V}\right)_{T,\mathbf{n}}. \tag{2.17}$$

[15] Since in the considered partial derivatives only one independent variable is varied at a time, the other (fixed) independent variables are not written as subscripts of the partial derivatives (in contrast to the usual notation in most thermodynamic books).

An alternative way of calculating the pressure is by obtaining it from the compressibility factor $Z = p\,V/(n\,R\,T)$ according to Eq. (2.18)

$$p = \frac{Z\,R\,T}{\hat{v}} = Z\,k\,T\,\rho\left(10^{10}\frac{\text{Å}}{\text{m}}\right)^3, \tag{2.18}$$

where the compressibility factor Z is calculated according to Eq. (2.19)

$$Z = 1 - \hat{v}\left(\frac{\partial\tilde{a}^{res}}{\partial\hat{v}}\right) = 1 + \rho\left(\frac{\partial\tilde{a}^{res}}{\partial\rho}\right). \tag{2.19}$$

As it will be shown henceforth, the compressibility factor Z plays a major role in the evaluation of almost all thermodynamic properties discussed in this contribution. Hence the derivatives of $\tilde{a}^{res}$ with respect to ρ or or $\hat{v}$ are of major importance.

Entropy and enthalpy

As indicated by the fundamental relation of the Helmholtz free energy (Eq. (2.6)) the entropy is obtained by Eq. (2.20)

$$S = -\left(\frac{\partial A}{\partial T}\right)_{V,\mathbf{n}} \tag{2.20}$$

Following the definition equation of the Helmholtz free energy $A = H - T\,S$ the enthalpy is given by Eq. (2.21)

$$H = A - T\left(\frac{\partial A}{\partial T}\right)_{V,\mathbf{n}}. \tag{2.21}$$

For typical engineering calculations involving energy balances the focus lies on the evaluation of molar enthalpies $\hat{h}$. For a given model of the reduced residual Helmholtz free energy $\tilde{a}^{res}$ the residual molar enthalpy can be calculated by Eq. (2.22) [74].

$$\hat{h}^{res} = -R\,T^2\left(\frac{\partial\tilde{a}^{res}}{\partial T}\right) + R\,T\,(Z-1) \tag{2.22}$$

Though in this thesis no entropy evaluation is required, Eq. (2.23) shows how the residual molar entropy $\hat{s}^{res}$ can be calculated out of $\tilde{a}^{res}$. It is well noticeable that the additional effort for evaluating the entropy is comparably low.

$$\hat{s}^{res}(\hat{v},T) = -R\,T\left[\left(\frac{\partial\tilde{a}^{res}}{\partial T}\right) + \frac{\tilde{a}^{res}}{T}\right] \tag{2.23a}$$

$$\hat{s}^{res}(p,T) = \hat{s}^{res}(\hat{v},T) + R\,\ln Z \tag{2.23b}$$

Chemical potential and fugacity coefficient

As indicated by the fundamental relation of the Helmholtz free energy (Eq. (2.6)) the chemical potential of a component i is given by Eq. (2.24)

$$\mu_i = \left(\frac{\partial A}{\partial n_i}\right)_{T,V,n_{j\neq i}} \quad \text{or} \quad \mu_i = \left(\frac{\partial A}{\partial N_i}\right)_{T,v,N_{j\neq i}} \tag{2.24}$$

As mentioned before, for practical evaluation of the equilibrium conditions it is common to use the isofugacity conditions (Eq. (2.12)) instead of the equality of the chemical potentials (Eq. (2.9)). Hence expressions for the fugacity f_i are required.

When modeling the behavior of real matter with EoS models, it is common to define the real behavior as a deviation with respect to the corresponding state of an ideal gas mixture. Based on the fact that the fugacity of a component i in an ideal gas mixture is given by $f_i^{iG} = p\,x_i$, a factor is defined, which addresses this deviation. The definition of this factor, known as fugacity coefficient, is given by (2.25).

$$\varphi_i = \frac{f_i}{p\,x_i} \tag{2.25}$$

According to Topliss [205] the fugacity coefficient of component i in a mixture can be obtained by evaluation of Eq (2.26). Note that $n = \sum_{i=1}^{NC} n_i$.

$$\ln\varphi_i = \left(\frac{\partial n\,\tilde{a}^{res}}{\partial n_i}\right)_{T,\rho,n_{j\neq i}} + Z - 1 - \ln Z \tag{2.26}$$

Replacing the derivative with respect to the molar numbers appearing in Eq. (2.26) with derivatives with respect to molar fractions, as shown among others in [175], results in Eq. (2.27)). Note that the derivative with respect to molar fraction of compound i as it appears in Eq. (2.27) describes a "true" partial derivative, in the sense, that all other molar fractions are held constant, regardless of the summation equation $\sum_{i=1}^{NC} x_i = 1$.

$$\ln\varphi_i = \tilde{a}^{res} + \left(\frac{\partial\tilde{a}^{res}}{\partial x_i}\right) - \sum_{j=1}^{NC} x_j \cdot \left(\frac{\partial\tilde{a}^{res}}{\partial x_j}\right) + Z - 1 - \ln Z \tag{2.27}$$

As a comparison, using a formulation in terms of $(T, V, \mathbf{n})$ as proposed by Mollerup and Michelsen leads to Eq. (2.28)

$$\ln \varphi_i = \left(\frac{\partial A^{res}/(RT)}{\partial n_i} \right)_{T,V,n_j} - \ln Z. \tag{2.28}$$

2.4.2 Phase equilibrium calculations

One of the most important tasks of thermodynamics of phase equilibria is the determination of the equilibrium phase distribution, that is, the number of phases in equilibrium and their compositions that result from applying fixed external conditions, such as pressure p or temperature T, to a given mixture of known overall compositions.

For given pressure p and temperature T the phase equilibrium distribution of a non-reactive system with an initial composition given by the vector of mole numbers $\mathbf{n^0}$ is characterized by the state that globally minimizes the total Gibbs free energy $G^{Total}(\mathbf{n}, p, T)$. That is by the values of $\mathbf{n}$ and NP resulting from the solution of the optimization problem defined by Eq. (2.29).[16]

$$\begin{aligned} \min_{\mathbf{n},NP} \quad & G^{Total}(\mathbf{n}, p, T) = \sum_{j=1}^{NP} \sum_{i=1}^{NC} n_{i,j} \mu_{i,j}(\mathbf{n_j}, p, T) \\ \text{s. t.} \quad & \left(\sum_{j=1}^{NP} n_{i,j} \right) - n_i^0 = 0, \ i = 1, \ldots, NC, \\ & n_{i,j} \in [0, n_i^0], \ i = 1, \ldots, NC \ ; \ j = 1, \ldots, NP; \\ & NP \in [0, NG] \subset \mathbb{N}; \end{aligned} \tag{2.29}$$

where $\mathbf{n}$ is a $NP \times NC$ matrix, whose elements $n_{i,j}$ represent the molar amount of component i in the phase j, $\mathbf{n_j}$ is a column vector containing the NC compositions of phase j, and NG stands for the maximal number of phases that can exist at phase equilibrium. The upper bound of the latter variable is determined by Gibbs phase rule.

While finding the global solution of the Gibbs free energy minimization as shown in Eq. (2.29) is the most safe and rigorous way of solving phase equilibrium calculations, for several numerical and computational reasons, it is common practice to obtain phase equilibrium distributions from the solution of general equations of phase equilibrium through so-called equation solving techniques [197]. These latter are often formulated as the general equations of an equilibrium stage listed below.

[16] Due to its completeness, the formulation of the optimization problem is taken from Pereira et al. [159].

Independently of the proposed solution method one of the most challenging issues related to the solution of phase equilibrium problems is given by the fact that the number of phases at equilibria NP is not known a priory. Probably, the most common approach to handle this issue when using equation solving techniques or Gibbs free energy minimization with local optimization methods is the integration of stability analysis checks within phase equilibrium calculations as proposed in [12] and [137].

The probably most common approach to evaluate phase stability is by means of the Gibbs tangent plane criterion. Therein, for fixed temperature, pressure and composition, the tangent plane distance function (TPDF) is evaluated over the whole composition space to check if the tangent plane distance function becomes negative at some position (or region) in the composition space. If this is the case, then the considered state is not stable and hence the amount of considered phases at equilibrium is wrong. See [144] for more details.

As an alternative to methods that couple the equation solving techniques or Gibbs energy minimization and stability analysis in separate steps, equation-solving techniques are available that include a simultaneous test of phase stability through the consideration of inequality constraints [e.g. 77]. Such techniques have also been successfully used with advanced molecularly based EoS models such as PCP-SAFT [6]. For the purposes of this thesis it is not necessary to introduce the stability analysis within the phase equilibrium calculations, since the phase behavior of the considered multicomponent system (and its components) is well known experimentally and the considered models reproduce the measurements properly.

Only for simulation results gained using model parameters obtained within own parameter estimation studies, see Section 5.4.2, additional stability analysis studies are realized so as to test that the predicted phase equilibria correspond to stable thermodynamic states. For these cases stability analysis is performed according to the formulation by Xu et al. [224].

General equations of an equilibrium stage

The general equations for an equilibrium stage are formulated next for a mixture of NC components yielding NP phases in equilibrium. It is assumed that there is one single (single-phase) inlet flow, which is denoted with the integer superscript 0. The following equations can be considered either for a closed or for an open system, depending on how the variable F is interpreted.

Material balances Independent of the number of phases resulting in equilibrium, the amount given by the overall composition (or feed) is preserved. This is stated by Eq. (2.30)

$$F^{(0)} \cdot x_i^{(0)} = \sum_{j=1}^{NP} F^{(j)} \cdot x_i^{(j)}, \qquad i = 1 \dots NC, \tag{2.30}$$

Equilibrium relations As previously explained, the necessary criteria for phase equilibria are given by the equality of pressure p ((2.31)), temperature T (Eq. (2.32)) and fugacity of each component f_i in all phases. Following the description of matter with an EoS model approach and considering Eq. (2.25), the isofugacity criterion can be written according to Eq. (2.33).

$$p^{(j)} = p^{(j+1)}, \qquad j = 1, \dots, NP-1 \tag{2.31}$$

$$T^{(j)} = T^{(j+1)}, \qquad j = 1, \dots, NP-1 \tag{2.32}$$

$$x_i^{(j)} \cdot \varphi_i^{(j)} = x_i^{(j+1)} \cdot \varphi_i^{(j+1)} \qquad j = 1, \dots, NP-1 \text{ and } i = 1, \dots, NC \tag{2.33}$$

In Eq. (2.33) the fugacity coefficients $\varphi_i^{(j)}$ are considered as a function of temperature T, pressure p and composition $\mathbf{x}$ of the corresponding phase j. Due the fact, that the expressions for fugacity coefficients, as shown in Eq. (2.27), are given as function of temperature, density and composition, it is necessary to find the density that corresponds to the given pressure. This is obtained from the solution of the so-called density root problem discussed below.

Summation equations Due to their nature, summation of molar fractions of any phase must equal 1. This is explicitly demanded in Eq. (2.34)

$$\sum_{i=1}^{NC} x_i^{(j)} = 1, \qquad j = 0, \dots, NP \tag{2.34}$$

Energy balance Neglecting energy contributions due to kinetic or potential energy, as usually in most chemical engineering applications, and considering $\hat{h}^{(j)}$ as specific enthalpy of a phase or stream j and Q as supplied/consumed heat flow the energy balance is expressed by Eq. (2.35).

$$F^{(0)} \cdot \hat{h}^{(0)} = \sum_{j=1}^{NP} F^{(j)} \cdot \hat{h}^{(j)} \pm Q, \tag{2.35}$$

Like the fugacity coefficients in Eq. (2.33), here the enthalpies are considered as functions of the temperature, pressure and composition of the respective stream j.

Degree of freedom of equilibrium stage

Counting the variables and equations listed from Eq. (2.30) to (2.35) a degree of freedom analysis can be realized. Table 2.1 presents the results of the degree of freedom analysis considering that the fugacity coefficients φ_i and molar enthalpies $\hat{h}$ are evaluated by functions that depend on the temperature, pressure, and composition of the corresponding streams.

Table 2.1: Degree of freedom analysis of general equilibrium stage model with NC components and NP phases in equilibrium

Variables		Amount	Equations		Amount
Compositions	$x_i^{(0)}, \ldots, x_i^{(NP)}$	$NC(NP+1)$	Material balance	(2.30)	NC
Mole (Flow)	$F^{(0)}, \ldots, F^{(NP)}$	$(NP+1)$	Mech. equilibrium	(2.31)	$NP-1$
Pressures	$p^{(0)}, \ldots, p^{(NP)}$	$(NP+1)$	Therm. equilibrium	(2.32)	$NP-1$
Temperatures	$T^{(0)}, \ldots, T^{(NP)}$	$(NP+1)$	Isofugacity	(2.33)	$(NP-1)\,NC$
Heat	Q	1	Summation relation	(2.34)	$NP+1$
			Energy balance	(2.35)	1
$\sum$		$(NC+3)(NP+1)+1$			$(NC+3)(NP)$
Degree of freedom					$NC+4$

As observed, independent of the number of compounds NC or number of phases NP in equilibrium, the equilibrium stage has a degree of freedom of $NC+4$. For fixed overall (inlet) conditions, i.e. for known overall molar amount (or feed flow) $F^{(0)}$, temperature $T^{(0)}$, pressure $p^{(0)}$ and $NC-1$ molar fractions $x_i^{(0)}$ this leads to the well-known fact that the degree of freedom of the equilibrium stage is two. Hence, equilibrium stage calculations, also known as flash calculation, are categorized according to the selected specification, leading for example to pT-flash, pQ-flash, pH-flash, among others. Because of its high practical relevance and application in this thesis the pT-flash specification is discussed in more detail below.

Though in most applications it is common to fix the overall (feed) compositions, the general equation-oriented formulation, as provided above, allows to make other specifications that may be helpful for specific cases. As an example, when treating coupled reaction/separation systems with full solvent recycle, as discussed later in Chapter 5, it may be convenient to specify outlet flows together with some outlet molar fractions in order to robustly solve the process with full recycle. When specifying outlet molar fractions, however, particular care should be taken, not to violate Gibbs' phase rule. This latter defines the maximum number of molar fractions at equilibrium that can be fixed as design values.

pT-Flash/ isothermal flash calculation

Among the many available possibilities of selecting the two degrees of freedom required to fully specify the equilibrium stage model [see 138], the specification of temperature T and pressure p, also known as isothermal (, isobaric) flash calculation, also called pT-flash calculation, leads to the most prominent phase equilibrium evaluation. The pT-flash is not only convenient for practical reasons, since both variables can relatively easy be measured and/or adjusted, but it also has computational advantages. As Mollerup and Michelsen [144] point out, it is the flash *"calculation for which a robust and reliable algorithm is most easily written"*.

In this thesis, pT-Flash calculations are used both for process simulation and parameter estimation purposes. Since particular attention is given to isothermal, two-phase pT-flash calculations we introduce next a popular alternative formulation of the general phase equilibrium equations for a system with two phases in equilibrium [see 144, 221]. The two phases in equilibrium are denoted by the superscripts (1) and (2). A general formulation for an arbitrary number of phases can be found in [144] or in the Appendix of [221]. A simplified version of the flash calculation for pure compounds, which is considered in Section 5.4.2 for the calculation of vapor pressure and saturated liquid density is given in Appendix B.2.

Material balances The formulation of the material balance results from dividing Eq. (2.30) by $F^{(0)}$, with $\Phi^{(1)} = F^{(1)}/F^{(0)}$ as the phase fraction of phase 1 and additionally implying that the overall mass balance $F^{(0)} = F^{(1)} + F^{(2)}$ is fulfilled.

$$x_i^{(0)} = \Phi^{(1)} \cdot x_i^{(1)} + (1 - \Phi^{(1)}) \cdot x_i^{(2)} \qquad i = 1, \ldots, NC \tag{2.36}$$

Equilibrium relations For a system in equilibrium it can implicitly be assumed that the equality of temperature and pressure is fulfilled, hence wherever the variables $T^{(j)}$ or $p^{(j)}$ appear in the equilibrium stage equation system they can be replaced by one single T and p respectively. Further, considering p and T as specifications, it is obvious that the mechanical and thermal equilibrium do not need to be formulated additionally. The resulting equilibrium relations are given by Eq. (2.37).

$$x_i^{(1)} \cdot \varphi_i^{(1)}(T, p, \mathbf{x^{(1)}}) = x_i^{(2)} \cdot \varphi_i^{(2)}(T, p, \mathbf{x^{(2)}}) \qquad i = 1, \ldots, NC \tag{2.37}$$

Summation equations Considering that the overall mass balance is included explicitly in the mass balances given by Eq. (2.36), in order to avoid linear dependency, the summation equations for all phases ($j \in \mathbb{N}$) but one, need to be considered. The considered summation equations (Eq. (2.38) and Eq. (2.39)) are shown below. Instead of removing one summation equation, it is common practice to combine two summation equations to one relation (Eq. (2.39)). In the common case where the overall (feed) composition is known, Eq. (2.38) is neglected.

$$0 = 1 - \sum_{i=1}^{NC} x_i^{(0)} \tag{2.38}$$

$$0 = \sum_{i=1}^{NC} x_i^{(2)} - x_i^{(1)} \tag{2.39}$$

Energy balance When pressure and temperature are selected as design variables, the energy balance provides the information, on how big the heat (flow) Q needs to be to fulfill Eq. (2.35). Since the energy balance can be solved independently of the further equations and the goals treated in this work focus on the calculation of equilibrium compositions, the energy balance can be neglected.

Density root problem

Following the requirements of the equilibrium stage calculation for specified temperature T and pressure p, it is necessary to evaluate fugacity coefficients for these independent variables. However, as shown in Eq. (2.27) the fugacity coefficients and its composing elements do not depend explicitly on pressure but on the density. For a system with given temperature, pressure and composition the corresponding density (or volume) can be obtained from the solution of the so-called density root problem. This consists of Eq. (2.40) in combination with Eq. (2.19).[17]

$$p = Z \cdot k \cdot T \cdot \rho = Z \cdot \frac{R \cdot T}{\hat{v}} \tag{2.40}$$

Depending on the type of EoS model used, the effort required for finding the solution of Eq. (2.40) can vary from trivial to very high. While in case of cubic EoS models simple analytic procedure as the Cardano's method can be used to obtain all existing roots,[18] in

[17] For the sake of a better readability, we desist to write any engineering units and superscripts denoting phases at this point.

[18] Interestingly, for some applications the analytic solution is not recommended due to round off errors related to the evaluation of trigonometric functions as approximated by digital computers [146].

case of complex, non-cubic EoS models, robust algorithms or at least good starting values for some general iterative procedure may be required. Topliss et al. [206] introduced a powerful algorithm strongly based on derivative information. A challenge related to the solution of the density root problem is related to the appearance of multiple roots (i.e. multiple volumes or densities) for given pressure and temperature. In the case of cubic EoS the resulting densities can be easily assigned to a liquid or vapor phase or be nonphysical. For higher-than-cubic EoS models, however, the situation is much more complex. For example Koak et al. [101] show it is possible to obtain up to 5 different volume roots using the SAFT EoS model [34] for values of pressure and temperature that lie in a practical region of interest. Similar results are obtained by Privat et al. [169] for the PC-SAFT EoS model [74].

2.4.3 Challenges related to complex multiphase liquid systems

Important current trends of chemical industry point toward replacing traditional production chains based on petroleum with more sustainable feedstock (e.ġ. biomass) and in general toward the development of processes and products characterized by highly complex thermophysical behavior, e. g. process with simultaneous electrolyte and association phenomena involving charged macromolecules or biomolecules. In this thesis the focus lies on the description of multiphase liquid systems, as they appear in the hydroformylation of 1-dodecene in a thermomorphic solvent system consisting of 1-decane and DMF (see Section 5.1). This system can be characterized as a highly temperature dependent gas-liquid, size asymmetric reacting system containing highly polar and nonpolar components at a relatively high pressure level of 2 MPa [136].

Model-based methods offer a systematic way of finding process and product improvements while reducing the effort for complex and expensive experiments. They have been very successfully applied for systems involving low-molecular weight components with rather weak interactions, that is, for systems whose thermophysical properties and phase behavior are properly described by classical thermodynamic models. In order to successfully apply model-based methods for more complex (industrial) applications, such as those taking place in *complex* multiphase liquid systems, it is thus necessary to have a proper description of thermophysical behavior and phase equilibrium, e. g. through a properly parameterized EoS model.

The challenges related to modeling and simulation of multiphase liquid systems are in principle twofold: either related to modeling challenges or to computational challenges. These two different types of challenges are discussed below. Additionally, a further important issue that mainly concerns the modeler or the model end user interested in implementing or simply testing a new model, is additionally discussed, namely the model availability.

Modeling challenges

Lack of understanding of phenomena of real matter at the molecular level manifests itself as thermodynamic challenges related to the description and analysis of macroscopic systems. In some extensive lists, Kontogeorgis and Folas [106] address some of the open thermodynamic challenges for the 21st century. Regarding the petroleum and chemical industries, the main application fields of interest for this work, some of the considered challenges are listed below:

- Prediction of multicomponent, multiphase equilibria, based exclusively on binary parameters.
- Simultaneous prediction of phase equilibria (e. g. VLE, LLE, SLE, VLLE) over extended temperature ranges with a common set of parameters.
- Need of highly predictive models for a wide range of applications, including calculation of thermophysical properties and phase equilibria.

Currently, the general strategy to address these issues is to develop EoS models that explicitly account for the underlying molecular phenomena. A common approach to achieve this is to consider the intermolecular forces as additive contributions to the Helmholtz free energy of a system. Since most challenges concern the description of highly associating or polar molecules and their mixtures, particular attention is given to the intermolecular forces of associative (self-association, cross-association) and polar (dipole, quadrupole, induced dipole) nature. The introduction of terms for explicit evaluation of association phenomena and polar interactions have enabled the successful modeling of several processes involving complex polar/associating fluids over extensive temperature and pressure ranges [e.g. 4, 104, 209]. While several approaches have been successfully developed for the characterization of polar contributions [73, 76, 93, 97, 100], as pointed in Kontogeorgis [104], there is still no consensus on the importance and the role of polar effects in the context of association models.[19] This is particularly the case for multicomponent system for which both, association and polar forces appear simultaneously.

Despite some deficiencies in describing some highly complex systems, e. g. multifunctional associating chemicals, electrolytes, among others [104], regarding the aforementioned challenges, EoS models based on association theories seem to be a step into the right direction. Particularly impressive are highly accurate descriptions of complex VLE over large pressure and temperature ranges with no BIPs or with very low values thereof. In spite of these

[19] von Solms et al. [216] and Folas et al. [52] show the feasibility of describing some polar compounds as self-associating. However, for highly polar mixtures it is rather recommended to explicitly consider polar terms [106].

impressive results, that more or less show the applicability of certain models for single application fields (or processes), modern association theories are not yet at the point to be considered as all-purpose models. For example, some LLE that can be quantitatively well fitted with activity coefficient models, are not necessarily well fitted by EoS models based on association theories (see for example [150, 181]), at least not when using a moderate number of fitted parameters, as usual for models based on association theories.

Computational challenges

Despite the (supposedly given) capability of a given thermodynamic model to correlate, estimate or predict complex thermophysical properties and phase equilibria with high fidelity, numerical or computational challenges may prevent the use of such models for practical applications. These issues can either prevent proper solutions to be found or lead to prohibitive large computational effort. In their review article Hendriks et al. [84] point up new appearing challenges regarding the fast and robust calculations for reactive and non-reactive multiphase flashes and stability tests as major industrial requirements for thermodynamics. Though in opinion of Hendriks et al. these issues are not of active interest to the thermodynamic community, the development of reliable methods for phase equilibria and stability analysis remains an active research area [e.g. 142, 143, 159, 160, 161, 176].

Most open challenges regarding robust flash calculations and stability analysis are related to the necessity of finding the **global minimum** of the Gibbs free energy function or the tangent plane distance function or the necessity of finding **all stationary points** of the latter. Particularly challenging is the fact that for advanced thermodynamic models the required expressions for the Gibbs free energy and the TPDF are highly nonlinear and non-convex.

Finding all stationary points or the global optimum of complex algebraic expressions using standard local methods, as commonly done, requires extensive evaluations over the search space and thus very often a prohibitive computational effort.[20] Alternatively, advanced computational techniques and optimization methods exist (and are still being developed) that provide a mathematical and computational guarantee of reliability of finding all roots or the global minimum. Several methods have been developed and applied to well established thermodynamic models such as standard activity coefficient models [131, 198] and cubic EoS models [81, 87]. Regarding (advanced) molecularly based EoS models, the work by Xu et al. [224] is particularly promising. A detailed review on the use of global optimization for phase equilibrium calculations with a strong focus on stochastic methods is given by Zhang et al. [230].

[20] As pointed out by Hendriks et al. [84] the stability test is the most time-consuming step of multiphase calculations.

As previously mentioned, in this thesis the phase behavior of the considered system, including the number of available phases at equilibrium, is well known a priori, so that stability analyses are not required in most cases and equation-solving techniques are enough for the considered cases. Though in this situation the issues of global optimality and phase stability do not play such a strong role as in other applications, it is still a challenging task to converge phase equilibrium calculations. A common issue here is the **trivial solution**. Considering the formulation of the pT-Flash/Isothermal Flash Calculation as formulated in Eq. (2.36) to (2.38) in Section 2.4.2 the trivial solution corresponds to the solution $x_i^{(2)} = x_i^{(1)} = x_i^{(0)}$ for $i = 1 \dots NC$. Though the trivial solution may appear in any equilibrium calculation, since it can be always fulfilled, independent of thermodynamic model used or phase equilibrium described, it commonly appears when the flash equation system is solved by (some generic) nonlinear solution approach such as the Newton's method and the starting values are not very close to the solution. These issues are particularly common for LLE calculations, for which starting guesses need to be extremely close to the solution in order to converge. To avoid the trivial solution, it is recommended to calculate the first steps with some specialized flash algorithm such as successive substitution, before changing to a standard Newton based generic solver with a better convergence rate.

Model availability

Besides the problems listed above, a very serious issue, certainly not only limited to complex multiphase liquid systems, is related to the availability of the models. In light of the rapidly growing amount and complexity of EoS models,[21] there is the necessity to find proper ways to make the new developments available to modelers/users willed to test the new developments for their applications. Though commercial process simulators and thermodynamic property packages try to keep pace with current trends and novel models, advanced users may have particular interest in applying some of the newest, sometimes recently published, developments. Unfortunately, many models are only available through equations provided in publications, and hence need to be implemented by the interested modeler on his own. As an example, in order to implement the heterosegmented PCP-SAFT EoS model it is necessary to extract the model equations from two Ph.D. theses [182, 208] and one paper [76].[22] Among other factors, the high algebraic complexity of the model equations, notation inconsistencies within publications and the apparently unavoidable typing errors make the model implementation a particularly laborious and error-prone task. In general, due to missing model availability and the big effort related to

[21] As pointed out by Kontogeorgis in [104] already, back in 2013 there were many more SAFT models than there have ever been UNIFAC versions.

[22] Additionally, it is necessary to build the partial derivatives of the polar contribution term.

the implementation of each single model it is often difficult to evaluate alternative models properly.

Looking for a solution to the model availability issues, Hendriks et al. [84] postulate the necessity of providing standardized model implementations with extensive documentation on the considered model equations and even recommend providing the source code of the model implementations. Some interesting efforts in this direction that partially follow these recommendations include the software PE 2000 by Pfohl et al. [166] and the publication and supplementary materials provided by Martín et al. [128]. PE 2000 makes available many well-known and modern thermodynamic models and gives the user the possibility of relatively easily integrating own thermodynamic models. For this latter feature the users needs to implement Fortran subroutines for the evaluation of pressure $p(T, \rho, \mathbf{x})$ and fugacity coefficient of a component i in a mixture $\varphi_i(T, \rho, \mathbf{x})$. In [128] MATLAB source code of advanced EoS models is made available. While these latter contributions certainly represent steps in the right direction, they still represent tool dependent solutions, so that the reusability is significantly restricted.

The present work is intended to provide an important contribution toward the easier implementation and standardization of advanced (molecularly based) EoS models. Following the documentation-based approach, as discussed in Section 2.2, the considered approach addresses the apparently unavoidable issue regarding differences in model documentation and implementation and avoids commonly appearing tool dependency. Instead of requiring the implementation of subroutines for pressure and fugacity coefficients, which are based on derivatives of the Helmholtz free energy, it suffices to write the expressions that define the Helmholtz free energy as an equation system.

2.4.4 Considered thermodynamic models

The best known and most used classical general models for the description of matter and phase equilibrium are cubic EoS models and activity coefficient models for the description of Gibbs free energy. Cubic EoS models [158, 193] have proven to be highly suitable and reliable tools for the description of process with gas and hydrocarbons also at high pressures [106, 153]. Among the activity coefficient models, local composition models for the excess Gibbs free energy with adjustable binary parameters such as NRTL [173] and UNIQUAC [1] are very strong tools for correlating highly complex phase behavior (e. g. LLE or SLE), however, they show rather weak predictive capabilities of multicomponent and multiphase systems when using binary parameters obtained from binary data only. On the other hand, the predictive group contribution approach UNIFAC [55] and its derivatives show remarkably good results for low-molecular weight mixtures of non-ideal compounds, but

again misses to perform well on highly non-ideal and in particular long chain molecules. Though the aforementioned models have a large amount of application fields, as remarked by Kontogeorgis and Folas [106] many classical models may have reached their limit of application, unless process-specific parameters are used.

The modeling application considered in this thesis, a hydroformylation process of 1-dodecene in a thermomorphic solvent system, introduced later in Section 5.1, is perhaps an example of an application that goes beyond those limits. In order to properly describe the process it is not only important to have a good description of complex highly temperature dependent liquid-liquid equilibrium at operation pressure around 2 MPa, but also allow the consideration of supercritical components including gas solubility. For the description of such complex multiphase equilibria, so-called EoS/g^E models, and models based on association theories [148], such as the cubic plus association equation of State (CPA EoS) [109] or the perturbed chain statistical associating fluid theory (PC-SAFT) [74] should perform well [182], [214]. In the following a short description of this model categories is given.

EoS/g^E models

A thermodynamically consistent extension of the two aforementioned types of thermodynamic models is given by so-called EoS/g^E models [105]. These combine a cubic EoS with a mixing rule based on a model for the excess enthalpy. Roughly speaking, these models consist of the classical expressions that define cubic EoS, with the main difference that the model for the excess free enthalpy becomes part of the mixing rule of the energy parameters (mostly called a). Depending on the fundamental assumptions made at the model derivations/development different expressions result that account for a different influence of the activity coefficient model in the overall models, that is for different mixing rule.

In principle EoS/g^E models combine the advantages and disadvantages of both main model sources. Like g^E models they are suitable for correlating complex liquid-liquid equilibrium, while additionally providing the strength of cubic EoS to consider potential pressure effects. According to Valderrama [210] back in 2003, they were perhaps the most appropriate for modeling mixtures with highly asymmetric components. From today's scientific view EoS/g^E models can be seen as a mature technology, one of the mostly followed topics of Chemical Engineering Thermodynamics of the mid 80s of the 20^{th} century. Despite well-known limitations EoS/g^E models belong nowadays to the best (conventional general) models available. The possibility of being combined with predictive models such as UNIFAC and its derivatives makes this family of models very popular. They type of models keep on being used for modeling complex systems such as Biodiesel [152] or CO_2-expanded-liquid systems [225]). Although EoS/g^E models in comparison with advanced

association models normally require a much larger number of binary parameters, they also provide in many cases excellent correlative capabilities. A further important advantage is given by the fact, that this latter type of model is currently available in most commercial process simulators.

SRK-MHV2 EoS model Within the modeling task pursued in this work, an "approximate zero reference pressure models", the Huron-Vidal mixing rules of second order, also denoted as MHV2 mixing rules [40] come into consideration. Well known drawbacks of zero reference pressure models, like relatively low performance in description of size asymmetric systems containing gases with hydrocarbons of different lengths, are often related to the fact that mixing rules use very often group contribution methods as its underlying activity coefficient model.

Preliminary studies [78] indicated that the SRK EoS model in combination with the Huron-Vidal mixing rules of second order and the Non-Random-Two-Liquid (NRTL) activity coefficient model is suitable to model the phase equilibria that appear in the process described in Section 5.1. It is perhaps interesting to remark that Kontogeorgis and Folas see today's "true value" of the NRTL model when used as a mixing rule of a cubic EoS [106]. The considered model equations describing the SRK-MHV2 model are given in Section 5.2.2 or in MOSAICmodeling's database as remarked in Appendix A.4. Note that whenever we refer to the SRK-MHV2 model within this contribution, NRTL is implied as underlying activity coefficient model.

Association models

Association models refer to a family of thermodynamic models that account explicitly for the formation of hydrogen bonds. The models developed for describing these systems are commonly categorized in three different groups: chemical, lattice (quasi-chemical) and perturbation theories [106]. Although these theories are derived based on different approaches, the chemical theory being the most empirical and perturbation theories the most theoretical one, it is well known that all of them lead to expressions with similar mathematical structure [45] and are hence able to describe complex pure and multicomponent systems with comparable success. While nowadays, chemical theories are rather seen as powerful correlative tools that need to be used judiciously [148, 167], due to their sound theoretical basis and the higher extrapolability based therein, association models based on perturbation theories are currently the most followed approach for the development of reliable EoS models suitable for engineering practice.

Following the terms used by Kontogeorgis [104], a less strict definition of association models refers to models that have specially been developed for handling the description of highly

polar and associating molecules and that can be integrated into equation of state models. Note however, that the application of these models is not strictly limited to associating fluids. The models from the current probably most popular family of molecularly-based EoS models based on perturbation theories, the Statistical Associating Fluid Theory (SAFT) and its derivatives, can be equally well applied to general associating fluids or to nonassociating chain fluids [148].[23]

Within the classical SAFT approach, as originally proposed by Chapman, Jackson, and Gubbins [35, 90] and later refined by Huang and Radosz [88, 89], there are three major contributions to the total intermolecular potential of a considered molecule. These are the dispersion-repulsion contribution of individual segments (seg), the contribution due to chain formation out of the individual segments ($chain$) and the contribution due to association of segments of some molecule with segment of other molecules ($assoc$) [148]. Accordingly, the resulting expression for the Helmholtz free energy consists of these three major contributions (see Eq. (2.41)).[24]

$$A^{res} = A^{seg} + A^{chain} + A^{assoc} \tag{2.41}$$

The big success of the original SAFT model triggered the development of numerous SAFT modifications that account for different types of molecular interactions and different approaches to describe these. Interestingly, despite some significant differences with respect to the original model, it should be noted that most modifications use the same chain formation and association contributions of the original SAFT model by Huang and Radosz [88]. Some of the most prominent SAFT modifications include simplified SAFT by Fu and Sandler [59], Lennard-Jones SAFT (LJ-SAFT) by Kraska and Gubbins [110], soft-SAFT by Blas and Vega [23], SAFT for potentials of variable range (SAFT-VR) by Gil-Villegas et al. [67] and perturbed chain SAFT (PC-SAFT) by Gross and Sadowski [74]. More details regarding the characteristics of these models can be found in [106] or in the original publications. Further former modifications that did not reach a similar impact as the aforementioned are also mentioned in Economou [44]. Another very important and more empirical approach is given by the so-called Cubic plus Association Model [107, 108, 109] which uses an association term as SAFT but considers a standard cubic EoS model for the description of the segment and the chain contributions of Eq. (2.41).

Among the aforementioned models, SAFT-VR, CPA and PC-SAFT have found perhaps

[23] As a matter of fact, the contribution to the Helmholtz free energy due to chain formation, which is common part of all models of the SAFT family results from applying association concepts to build chain fluids out of spherical segments.

[24] Newer modifications of the model include the consideration of further contributions such as multipolar forces or Coulomb interactions.

the most diverse application fields and led to the development of a significant number of further specialized models. For example, for the PC-SAFT EoS, several modification have appeared such as the simplified PC-SAFT (s-PC-SAFT) by von Solms et al. [215], several implementations with different contributions for multipolar forces as polar PC-SAFT by Tumakaka et al. [209], PC-polar SAFT (PCP-SAFT) by Gross and Vrabec, [73, 76], PC induced-polar SAFT (PCIP-SAFT) by Kleiner and Gross [100] or the truncated PC-polar SAFT (tPC-PSAFT) by Karakatsani et al. [96]. A further important group of SAFT modifications are related to the description of heterosegmented molecules, that is, molecules that do not consist of the same type of spherical segments (as in the classical chain approach) but of different chain segments, and each chain consists of spherical segments of the same type. Such a concept is particularly valuable for the description of polymers, complex long chain molecules (e. g. *n*-dodecanal or *n*-tridecanal) or for the transfer of the concept of group contribution methods into the SAFT framework. This latter topic, the development of so-called GC-SAFT approaches, is a very hot topic in current research, as different research groups develop their own approaches, see for example Peters et al. [164] , Paduszyński and Domańska [154], Sauer et al. [180] for GC-SAFT approaches using the PC-SAFT EoS model as basis. In the near future these methods should play a major role for the calculation, in particular *prediction* of phase equilibria.

Despite the big progress done up to now, modern association models cannot still be used as general thermodynamic models for all types of applications.[25] Hence, a detailed and critical evaluation of the thermodynamic model is a ubiquitous part of the process modeling process. Taking into consideration the large amount of available publications that basically praise the capabilities of new model developments without providing any source code, modern (computational) methods for efficient implementation of such models are of high relevance.

Since in the case studies discussed later in Chapter 5 a particular association model, the heterosegmented PCP-SAFT [181], will be used, this model will be shortly introduced in the following.

Heterosegmented PCP-SAFT Being a modification of the statistical associating fluid theory (SAFT) by Chapman et al. [34], PC-SAFT, developed by Gross and Sadowski [74], is a particularly successful EoS model, which has shown impressive capabilities in modeling thermophysical properties of pure components and phase equilibria of multicomponent systems. PC-SAFT provides an expression for the residual Helmholtz free energy A^{res} which consists of several additive contributions accounting for different intermolecular forces.

[25] In fact quoting an important review from industrial perspective Hendriks et al. [84] "Association models such as SAFT and CPA are gaining ground".

Besides the hard chain contribution (*hc*) that accounts for repulsive interactions, relevant attractive forces such as London dispersion (*disp*), dipole-dipole interaction (*dipole*) and association (*assoc*) are usually considered (Eq. (2.42)). The use of different expressions for the dipole-dipole contribution A^{dipole} leads to different denominations of the PC-SAFT approach. Using the dipole interaction term proposed by Gross and Vrabec [76] leads to the so-called perturbed chain polar SAFT (PCP-SAFT) EoS.

$$A^{res} = A^{hc} + A^{disp} + A^{dipole} + A^{assoc} \tag{2.42}$$

In [181] and [214] it was found that for the description of the considered hydroformylation mixtures explained in more detail in Section 5.1(see also Table 5.1) dipole-dipole interactions need to be considered, but no association forces. Based on this, the resulting expression for the reduced residual Helmholtz free energy $\tilde{a}^{res} = A^{res}/(N \cdot k \cdot T)$ that will be considered in this work is given by Eq. (2.43).

$$\tilde{a}^{res}(\rho, \mathbf{x}, T) = \tilde{a}^{hc} + \tilde{a}^{disp} + \tilde{a}^{dipole}, \tag{2.43}$$

The notation in $\tilde{a}^{res}(\rho, \mathbf{x}, T)$ indicates the computational implementation of $\tilde{a}^{res}$ as function of the total number density of molecules of the phase ρ, the component molar fractions of the mixture $\mathbf{x}$ and the temperature T (see Section 2.1.1). In order to evaluate the expressions given in Eq. (2.43) following the hs-PCP-SAFT approach, for each segment α of a nonpolar component i, three pure component parameters are needed, the number of segments per chain $m_{i,\alpha}$, the segment diameter $\sigma_{i,\alpha}$ and the segment dispersion energy $\epsilon_{i,\alpha}/k$. In case of polar components, additionally to the three previously mentioned parameters the dipole moment $\mu_{i,\alpha}$ is required. For the calculation of mixtures, mixing rules are required. As usual the Lorentz-Berthelot mixing rules are applied, whereas in this case they refer to the interaction of a segment α of component i with a segment β of component j (Eqs. (2.44) and (2.45)).

$$\sigma_{i,\alpha,j,\beta} = \frac{1}{2} \cdot (\sigma_{i,\alpha} + \sigma_{j,\beta}) \tag{2.44}$$

$$\epsilon_{i,\alpha,j,\beta} = (1 - k_{i,\alpha,j,\beta}) \cdot \sqrt{\epsilon_{i,\alpha}/k \cdot \epsilon_{j,\beta}/k} \tag{2.45}$$

The model equations building the hs-PCP-SAFT approach, as considered in this work, can be found in [208] and [182]. Minor typing errors appearing in [181, 182] can be located proofing their consistency with the polar term published in [76]. The corresponding analytic expressions for the reduced Helmholtz free energy are available through the MOSAICmodeling database as indicated in Appendix A.4.

UNIFAC Dortmund

In addition to the already referred hs-PCP-SAFT and SRK-MHV2 models, for some LLE evaluations at ambient pressure the UNIFAC Dortmund (UNIFAC-DO) [91] model was used. As found out in preliminary studies of Gutierrez-Sanchez [78], among the UNIFAC variants available through Aspen Plus V8.2, UNIFAC-DO fits the available LLE experimental data, for the considered hydroformylation process, the best (see Chapter 5.1). Flash calculations with UNIFAC-DO were done with Aspen Plus V8.2 or using a CAPE-OPEN property package generated by it.

Chapter 3

Derivative Evaluation Techniques

It is a well-known fact that the accurate and efficient evaluation of derivative information of mathematical process models plays a major role in the field of computer-aided process engineering [202]. Most types of generic solution algorithms handling nonlinear systems require some kind of evaluation of derivative information. While basic simulation tasks concerning the solution of mostly nonlinear algebraic systems (AEs) and dynamic systems (ODEs or DAEs) primarily require first order derivatives,[1] most more advanced model-based applications require by default second and higher order derivatives. Some examples include general nonlinear programming (NLP) problems, which require second order derivatives if second-order gradient-based NLP solver are used, or optimal experimental design (OED) for parameter estimation, which requires up to third-order derivatives if second-order gradient-based NLP solvers are used [8, 127]. But not only solution algorithms require derivatives. As already shown in Section 2.4.1 for the evaluation of standard thermodynamic properties, such as pressure or fugacity coefficients, derivatives of thermodynamic state functions may be required as well. Accordingly, model-based methods including such models require derivatives of even higher order.

The evaluation of derivative information of high quality is, however, not only linked to high robustness or better rate of convergence, but it has also an important influence on the computational effort required for the evaluation and solution of given models and model-based tasks. This effort depends strongly on the method considered for derivative evaluation.

This chapter provides the necessary concepts in order to understand the integration of automatic derivative evaluation methods in the model development discussed in Chapter 4. After giving a short overview of the most commonly required derivatives within process

[1] These appear mostly in form of Jacobian matrices. In case of larger systems, where the solution of linear equation systems with direct methods becomes critical, in form of directional derivatives.

simulation and the most popular techniques for derivative evaluation in Sections 3.1 and 3.2 respectively, a more detailed, but still short description on algorithmic differentiation (AD) is provided in Section 3.3. This includes an overview about popular AD tools, their use and some special characteristics of AD of equation-oriented model implementations. Section 3.4 describes the state of the art on the use of derivative information in PSE and thermodynamic applications.

3.1 Short Overview of Commonly Required Derivatives

Following explanations show the relation between the Jacobian matrix and directional derivatives and are important to understand the concepts introduced later in Section 3.2.3. For the sake of simplicity, some of the following remarks refer to a general closed-form model formulation, representing the sufficiently smooth multivariate nonlinear function $\mathbf{G} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$:

$$\mathbf{y} = \mathbf{G}(\mathbf{u}) \tag{3.1}$$

Jacobian matrix

For the solution of simulation problems with mathematical models described by the vector valued function $\mathbf{G}$, Eq. (3.1), the corresponding Jacobian matrix $\mathbf{G}'(\mathbf{u})$, also called Jacobian, is required. It is the matrix of all first order partial derivatives defined according to Eq. (3.2)

$$\mathbf{G}'(\mathbf{u}) = \frac{\partial \mathbf{y}}{\partial \mathbf{u}} = \left[\frac{\partial \mathbf{y}}{\partial u_1} \quad \cdots \quad \frac{\partial \mathbf{y}}{\partial u_{n_u}} \right] = \begin{bmatrix} \frac{\partial y_1}{\partial u_1} & \cdots & \frac{\partial y_1}{\partial u_{n_u}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_{n_y}}{\partial u_1} & \cdots & \frac{\partial y_{n_y}}{\partial u_{n_u}} \end{bmatrix} \tag{3.2}$$

where the j^{th} column of the Jacobian, the partial derivative with respect to u_j, is defined according to Eq. (3.3)

$$\frac{\partial \mathbf{y}}{\partial u_j} = \frac{\partial \mathbf{G}}{\partial u_j} = \lim_{h \to 0} \left(\frac{\mathbf{G}(\mathbf{u} + e_j\, h) - \mathbf{G}(\mathbf{u})}{h} \right). \tag{3.3}$$

In Eq. (3.3) e_j stands for the Cartesian basis vector and h is the perturbation variable. Note that the partial derivative defined above is a particular case of a directional derivative

Directional derivatives

The directional derivative of $\mathbf{G}$ along a vector $\mathbf{v}$ is defined by following limit

$$\mathbf{G}'_{\mathbf{v}}(\mathbf{u}) = \lim_{h \to 0} \left(\frac{\mathbf{G}(\mathbf{u} + \mathbf{v}\, h) - \mathbf{G}(\mathbf{u})}{h} \right), \tag{3.4}$$

and can be obtained from the matrix-vector multiplication of the Jacobian matrix $\mathbf{G}'(\mathbf{u})$ with the direction $\mathbf{v}$ according to Eq. (3.5)

$$\mathbf{G}'_{\mathbf{v}}(\mathbf{u}) = \mathbf{G}'(\mathbf{u}) \cdot \mathbf{v} \tag{3.5}$$

In addition to matrix-vector products, commonly called as tangents, directional derivatives are also found as vector-matrix product, also called adjoint directional derivatives [71, 151] (See also Section 3.3.1).

Practical importance of directional derivatives

The previously mentioned categorization of directional derivatives is of big importance for the later subsections, in which methods will be introduced that are specialized in the generation of such directional derivative information. For example, using the forward mode of algorithmic differentiation, explained in Section 3.3.1, directional derivative characterized through Eq. (3.12) or Eq. (3.5) can be evaluated with machine-accuracy without explicitly evaluating the Jacobian $\mathbf{G}'(\mathbf{u})$. It should be remarked that this kind of directional derivatives is not only valuable for solution algorithms which explicitly require such[2] but they can also be used to generate full Jacobian or Hessian information. In general, it is not necessary to build a full Jacobian out of n_u Cartesian directional derivatives. The number of directional derivatives required to build the full Jacobian can significantly be reduced by applying compression algorithms such as the Curtis-Powell-Reid (CPR) algorithm [39] to the sparsity pattern of the system. As an example, in [133] it is shown, how the full Jacobian of a distillation column model with 280 equations can be evaluated by 14 directional derivatives, instead of 280. The distillation column model is a modification of a benchmark example, originally published in [147], more popularly known as Hydrocarbon-20.

It should be noted that the use of directional derivatives is not limited to generate derivatives required by solution algorithms. As shown in Section 2.4.1 first order derivatives of

[2] These are mostly algorithms for truly large-scale systems, for which the underlying solution of the resulting linear systems is made with iterative methods. As an example, Vetukuri et al. [212] present an efficient optimization algorithm that works with Jacobian-vector and Hessian-vector products instead of full (dense) Jacobian and Hessian matrices.

the Helmholtz free energy function with respect to its independent variables, its gradients, are necessary to build any other mixture property. Later in Section 3.3.1, it will be shown that gradients can be evaluated particularly efficient with the reverse mode of algorithmic differentiation.

3.2 Overview of Common Derivative Evaluation Techniques

This section introduces the derivative generation/evaluation methods that are used in this thesis. General information regarding the accuracy of derivatives, computational effort and applicability are discussed. Some definitions required for this discussion are introduced in the next sections. A more detailed review on methods for computational evaluation of derivatives is given by Tolsma and Barton [202]. A quantitative evaluation of the computational efficiency of these different methods for the generation of gradients required for the implementation of advanced thermodynamic models is provided in Section 5.2.

3.2.1 Derivatives of equations: Hand-coded and symbolic derivatives

Though these methods used for derivative evaluation are fully different, in this section they are listed together since both are obtained from applying differentiation rules on expressions formulated as equations and the resulting expressions are equations as well. While hand-coded derivatives result from applying differentiation rules "'by hand"', as taught in school, symbolic derivatives result from recursively applying the same rules on a computational tree representation [134, 156]. These both types of derivative generation methods have also in common, that resulting derivative values are *exact*.[3]

Nowadays, it is a common practice to use symbolic derivatives provided through CAS tools for the validation of carefully developed, hand coded derivatives (see Section 3.4).

Hand-coded derivatives

This type of derivatives is obtained through the application of differentiation rules on mathematical expressions by hand and the subsequent coding of the resulting derivative expressions. If carefully coded, that means, if expressions common to the original equations and their derivatives are properly reused (exploited), hand-coded derivatives can be highly efficient, perhaps as efficient as possible. In fact, the discussion from Section 2.4.1 regarding the most proper approach for the selection of independent variables for the evaluation of

[3] Within this context, exact means that the only source of errors are round off errors, which are unavoidable.

the Helmholtz free energy is based on the question, which formulation allows the coder to easily find and reuse common expressions at the model implementation stage.

In spite of these advantages, when working with complex mathematical expressions, the implementations of hand-coded derivatives is in general a very time-consuming and error prone process. An implementation of hand-coded derivatives appears to be reasonable for two completely different situations: either for highly trivial models, or if the best performance possible is required.

Symbolic derivatives

Symbolic derivatives refer to the generation of derivatives as supported by computer algebra software like Maple [122]. For a given mathematical expression, building symbolic derivative with respect to one of its variables, leads to a further equation as it would result from applying the differentiation rules by hand.

When building commonly required derivative information, for example gradients, Jacobians, or Hessians, the common strategy inherent to symbolic derivatives is to build the single elements, that means single partial derivatives, one by one. It is obvious that such an approach may significantly reduce the chances of reusing common subexpressions and hence is not particularly efficient. Additionally, it is a well-known fact that symbolic derivatives of complex equations are in some cases much more complex than the original equations so that the effort for the evaluation of the derivatives can be much larger than the effort for the evaluation of the original function. Examples that show these and further deficiencies of symbolic derivatives are discussed among others by Griewank [69] or Thiéry [200]. A common issue that appears in this context is known as expression swelling [151].

Well-known process modeling tools such as ACM or gPROMS build symbolic derivatives of models defined in their respective modeling languages for solution purposes. These tools do not offer access to the single derivative expressions but work with internal equation representations that exploit common subexpressions of the residuals and its derivatives [see 155, 156]. This enables highly efficient evaluations of derivative information.

In spite of the availability of efficient implementations of symbolic derivative evaluations that allow the reuse of common expressions, there are applications for which symbolic derivatives are not suitable. In case of simulation of truly large-scale and dense systems, for which direct methods are not the first choice for the solution of linear equation systems, no full Jacobians may be needed but directional derivatives may be preferred. In such cases, it may be a better idea to perform algorithmic differentiation to obtain the required directional derivative (see Section 3.2.3) than to build the directional derivative as a product of a full Jacobian with a vector direction.

3.2.2 Derivative approximation methods: Finite differences and complex step derivative approximation

A different way to obtain numerical values of derivatives of a *mathematical expression* follows from the truncation of a Taylor series expansion of that expression. In this section two approaches of this class are discussed: the well-known finite differences approximation (FD) and the complex step derivative approximation, which has only recently become widespread. Due to their nature as approximations, numerical calculated derivative values obtained by these methods not only have round off errors but also truncation errors.

Finite differences

Though neither particularly accurate, nor computational efficient, finite differences (FD) is perhaps one of the most used methods for derivative evaluation. The main reasons for this popularity are given by its straightforward implementation and by the fact that FD can always be applied, even when the model is only available as a black box, for example as an executable closed-form implementation. FD provides a simple way to approximate partial derivatives or directional derivatives. Using forward difference approximations the partial derivatives defined by Eq. (3.3) can be evaluated by a difference quotient defined in Eq. (3.6).[4]

$$\frac{\partial \mathbf{y}}{\partial u_j} = \frac{\partial \mathbf{G}}{\partial u_j} \approx \left(\frac{\mathbf{G}(\mathbf{u} + e_j\, h) - \mathbf{G}(\mathbf{u})}{h} \right) \tag{3.6}$$

A critical issue concerning all types of FD approximations is the selection of the perturbation variable h. If a too large value of h is chosen, large truncation errors are the consequence. If, however, h is chosen too small, subtractive cancellation errors appearing in the numerator of Eq. (3.6), become dominant. In critical applications, where the best possible accuracy is required but for specific reasons, FD is the only feasible way to evaluate derivatives, it may become necessary to execute additional evaluations so as to find the optimal perturbation h with the minimal error. But even if the optimal value of h is selected, as pointed out by Tolsma and Barton [202] "*the number of significant values in the derivatives is much less than that of the original function value*". According to Griewank [70] "*one must expect a halving in the number of significant digits under the best of circumstances*". While in some cases, the loss of accuracy for the first order derivative resulting from FD, is acceptable, for higher order derivatives this is mostly not the case.

[4] Another well-known, and more accurate approach is the central difference approximation. This is twice as expensive as forward difference approximation.

As pointed out by Naumann [151] *"The impact of cancellation and rounding errors becomes even more dramatic if second derivatives are approximated using second-order finite differences".*

Complex-step derivative approximation

Keeping the notation used in Section 3.1 for the description of a general *closed-form* algebraic procedure $\mathbf{y} = \mathbf{G}(\mathbf{u})$, the complex-step derivative approximation of $\mathbf{y}$ with respect to the j-th element of $\mathbf{u}$ is given by Eq. (3.7)

$$\frac{\partial \mathbf{y}}{\partial u_j} = \frac{\partial \mathbf{G}}{\partial u_j} = \lim_{h \to 0} \frac{\mathrm{Im}\left[(\mathbf{G}(\mathbf{u} + e_j\, i\, h)\right]}{h}. \tag{3.7}$$

where i represents the imaginary unit, $i = \sqrt{-1}$, and Im[ ] denotes the imaginary part of a complex number inside the square brackets. Details on the theoretical foundations of this method and some practical aspects are provided by Martins et al. [129]. Regarding their implementation, support for the evaluation of complex analysis is necessary. Hence, the used programming language must support complex arithmetic. In [129], details on Fortran and C/C++ implementations are provided. The algorithmic differentiation tool for MATLAB ADiMat [20] (see Section 3.3.3) also provides an easy to use interface to use the complex-step derivative approximation. An important difference of this method with respect to FD approximations is that no subtraction appears in the numerator of Eq. (3.7), and hence no subtractive cancellation errors can appear. As a consequence, if very small values of h are selected, derivative values can achieve the accuracy of the function evaluations and become perturbation-size insensitive. No additional efforts are required to find an optimal perturbation h. In his Ph.D. thesis Al-Saifi [3] analyses the influence of derivative values of advanced molecularly based EoS models required for the solution of phase equilibrium calculations. He concludes that differences caused by evaluating derivatives with the complex-step approximation instead of analytic expressions are negligible.

A significant shortcoming of this approach is that high accuracy is only reached for first-order derivatives [118]. A different, still not so widespread method that enables the calculation of high-order derivatives of high accuracy is based on the concept of multi-complex numbers as proposed by Lantoine et al. [118]. Alternatively, n-th order derivative information of high quality can be obtained with the complex-step derivative approximation, if $(n-1)$-th order is available from any other approach.

Following the evaluating results by Martins et al. [129] it can be concluded that in comparison with FD and algorithmic differentiation the complex-step derivative approximation

represents a good compromise between ease of implementation and algorithmic efficiency. The former result is based on the opinion that code needs less changes than when applying AD.

3.2.3 Algorithmic differentiation

Algorithmic differentiation (AD) [71, 151], also commonly known as automatic differentiation or computational differentiation, denotes a set of computational techniques for evaluation of truncation-error-free (directional) derivative information of computer programs. Within this context, a computer program denotes source code written in some high-level programming language (e. g. Fortran, C++, MATLAB) with common program elements such as IF-ELSE statements, FOR loops and complex hierarchies of subroutine calls [203].

Given a program for the evaluation of an arbitrary function, the application of AD techniques leads to a modified or new program that returns the values of the function and its derivatives with machine accuracy. Since a computer program, can be used equally well to express acausal relations (e. g., open-form implementation in form of residuals) or causal relations (e.ġ., algorithmic closed-form evaluation) AD can be used to fulfill the requirements of derivative information of any type of model implementation. The only precondition is the availability of source code.

Since a deeper understanding on AD and its practical implementations, at least at the level of an advanced user, is required to understand some of the ideas developed in this thesis, in the next section a more detailed, but still short introduction to AD is given. It is mainly based on explanations given by Slawig [191], Hannemann-Tamás [80] and Willkomm et al. [222]. Other short, but more detailed descriptions are provided in the Ph.D. theses of Hannemann-Tamás [80] and Andersson [7] or in [191]. Extensive introductions to AD are provided in the books of Griewank and Walther [71] and Naumann [151].

3.3 A More Detailed but Still Short Introduction to Algorithmic Differentiation

The key aspect of AD is that any mathematical function/expression available as a program can be decomposed into a sequence of unary and binary operations, for which partial derivatives are well known and can easily be evaluated. For example, the general mathematical function defined in Eq. (2.3), alternatively expressed by Eq. (3.8)

$$\mathbf{G} : \mathbf{u} \mapsto \mathbf{y}, \tag{3.8}$$

can be thought to consist of the concatenation of k different elementary functions G_i [5]

$$\mathbf{G} = \mathbf{G}_k \circ \mathbf{G}_{k-1} \cdots \circ \mathbf{G}_2 \circ \mathbf{G}_1. \tag{3.9}$$

This concatenation can be equivalently expressed by the list of evaluation instructions shown below in Eq. (3.10). Therein the variables $\mathbf{x}_1$ to $\mathbf{x}_k$ are denoted as intermediate variables that keep the values of the evaluation of the intermediate operations $\mathbf{G}_i$

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{u} \\ \mathbf{x}_i &= \mathbf{G}_i(\mathbf{x}_{i-1}), \qquad i = 1, \ldots, k \\ \mathbf{y} = \mathbf{G}(\mathbf{u}) &= \mathbf{x}_k \end{aligned} \tag{3.10}$$

Depending on the way partial derivatives are accumulated to build the required derivative information, two different modes of AD exist, the forward mode (FM) and the reverse mode (RM). In both cases common subexpressions appearing in both, the original evaluation and its corresponding derivatives, are exploited for a more efficient evaluation. Regarding the tool support of AD, two different main types of implementations exist, operator overloading (OO) and source code transformation (ST). In the following section closer details are given regarding the most important modes and implementation types of algorithmic differentiation.

3.3.1 Algorithmic differentiation modes

Considering mathematical functions as a concatenation of elemental functions as in Eq. (3.9), the corresponding derivative information can easily be obtained from the application of the chain rule of differentiation. This results in a product of the single derivative components (Eq. (3.11))

$$\mathbf{G}'(\mathbf{u}) = \mathbf{G}'_k(\mathbf{G}_{k-1}(\cdots(\mathbf{G}_1(\mathbf{u})))) \cdots \mathbf{G}'_2(\mathbf{G}_1(\mathbf{u})) \cdot \mathbf{G}'_1(\mathbf{u}) \tag{3.11}$$

which, considering Eq. (3.10), alternatively can be expressed by Eq. (3.12)

$$\mathbf{G}'(\mathbf{u}) = \mathbf{G}'_k(\mathbf{x}_{k-1}) \cdots \mathbf{G}'_2(\mathbf{x}_1) \cdot \mathbf{G}'_1(\mathbf{x}_0). \tag{3.12}$$

Note that Eq. (3.12) represents *full derivative information* as defined in Eq. (3.2). If any particular type of directional derivative information is required, for example directional

[5] The vector notation is kept in order to keep generality.

derivatives of the form $\mathbf{G}'(\mathbf{u}) \cdot \mathbf{v}$ with $\mathbf{v} \in \mathbb{R}^{n_u}$, or $\mathbf{w}^T \cdot \mathbf{G}'(\mathbf{u})$ with $\mathbf{w} \in \mathbb{R}^{n_y}$, the matrix product given by Eq. (3.12) needs to be expanded accordingly.

Regarding the evaluation of such directional derivatives, a very important aspect is given by the fact that matrix multiplication is associative.[6] For example, $\mathbf{G}'(\mathbf{u}) \cdot \mathbf{v}$ can be either evaluated according to Eq. (3.13)

$$\mathbf{G}'(\mathbf{u}) \cdot \mathbf{v} = \mathbf{G}'_k \cdot (\cdots \mathbf{G}'_2 \cdot (\mathbf{G}'_1 \cdot \mathbf{v}) \cdots), \tag{3.13}$$

with an effort given by k matrix-vector products or according to Eq. (3.14) with a larger effort given by $k-1$ matrix-matrix multiplications and 1 matrix-vector multiplication.

$$\mathbf{G}'(\mathbf{u}) \cdot \mathbf{v} = ((\cdots (\mathbf{G}'_k \cdot \mathbf{G}'_{k-1}) \cdot \mathbf{G}'_{k-2} \cdots) \cdot \mathbf{G}'_1) \cdot \mathbf{v} \tag{3.14}$$

The evaluation order shown in Eq. (3.13), where the derivative of the first operation is performed first and the derivative of the last operation is performed last, corresponds to the *forward mode* of algorithmic differentiation and is especially suited for directional derivatives of the form $\mathbf{G}'(\mathbf{u}) \cdot \mathbf{v}$. On the other hand, the evaluation order shown in Eq. (3.14) corresponds to the *reverse mode* of algorithmic differentiation. This latter approach can be particularly efficient when evaluating directional derivatives of the form $\mathbf{w}^T \cdot \mathbf{G}'(\mathbf{u})$, a type of directional derivatives better known as adjoints.

In summary, AD can be seen as a set of advanced computational techniques to perform highly efficient directional derivatives. The different modes provide suitable methods to evaluate different types of directional derivatives and can have a significant influence on the computational effort required to evaluate these. For a given function $\mathbf{G} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_y} : \mathbf{y} = \mathbf{G}(\mathbf{u})$, the most important factor that indicates the suitability of the modes of AD are the number of independent variables n_u and the number of dependent variables n_y. The most important facts regarding the characteristics of both modes of AD are presented next.[7]

Forward mode

For a given mathematical function $\mathbf{G} : \mathbb{R}^{n_u} \to \mathbb{R}^{n_y} : \mathbf{y} = \mathbf{G}(\mathbf{u})$ available as computer program and a *seed* matrix $\mathbf{S}_{FM} \in \mathbb{R}^{n_u \times n_{dd}}$ the forward mode of AD provides means to

[6] For given matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ of compatible dimensions associativity means that $(\mathbf{A} \cdot \mathbf{B}) \cdot \mathbf{C} = \mathbf{A} \cdot (\mathbf{B} \cdot \mathbf{C})$

[7] The following explanations are based on Willkomm et al. [222].

efficiently evaluate $\mathbf{G}'(\mathbf{u}) \cdot \mathbf{S}_{FM}$. The most important facts are summarized in Eq. (3.15).

$$n_y \left\{ \underbrace{\begin{bmatrix} \\ \mathbf{G}'(\mathbf{u}) \\ \\ \end{bmatrix}}_{n_u} \cdot \underbrace{\begin{bmatrix} \\ \mathbf{S}_{FM} \\ \\ \end{bmatrix}}_{n_{dd} \equiv \text{costs}} \right\} \text{for free} \tag{3.15}$$

The main information provided through Eq. (3.15) is related to the computational effort. Commonly used approaches to evaluate this effort compare the cost of evaluating the required directional derivatives against the cost of evaluating the original functions. For example, the cost of evaluating $\mathbf{G}'(\mathbf{u}) \cdot \mathbf{S}_{FM}$ is bounded above by around 3 times n_{dd} the cost of evaluating $\mathbf{G}(\mathbf{u})$:

$$\frac{\text{cost}(\mathbf{G}'(\mathbf{u}) \cdot \mathbf{S}_{FM})}{\text{cost}(\mathbf{G}(\mathbf{u}))} \leq 3 \cdot n_{dd}. \tag{3.16}$$

Note that these results, though a bit worse, are comparable with FD approximation but provide truncation free derivatives. If the structure (the sparsity pattern) of the system permits it, the seed matrix can be chosen in such a way that the full Jacobian information can be evaluated with only a small number of directions n_{dd}, as already discussed in 3.1. Note that these advantages can be exploited by FD and the complex-step derivative approximation method as well.

Reverse mode

Considering the same mathematical function as previously discussed and a seed matrix $\mathbf{S}_{RM} \in \mathbb{R}^{n_{dd} \times n_y}$, the reverse mode of AD provide means to efficiently evaluate $\mathbf{S}_{RM} \cdot \mathbf{G}'(\mathbf{u})$. The most important facts are summarized in Eq. (3.17).

$$n_{dd} \equiv \text{costs} \left\{ \underbrace{\begin{bmatrix} \mathbf{S}_{RM} \end{bmatrix}}_{\text{for free}} \cdot \underbrace{\begin{bmatrix} \\ \mathbf{G}'(\mathbf{u}) \\ \\ \end{bmatrix}}_{n_u} \right\} n_y \tag{3.17}$$

In RM, the ratio of the cost of evaluating $\mathbf{S}_{RM} \cdot \mathbf{G}'(\mathbf{u})$ and the cost of evaluating the original function $\mathbf{G}(\mathbf{u})$ is given by Eq. (3.18)

$$\frac{\text{cost}(\mathbf{S}_{RM} \cdot \mathbf{G}'(\mathbf{u}))}{\text{cost}(\mathbf{G}(\mathbf{u}))} \leq 3 \cdot n_{dd} \tag{3.18}$$

and is hence fully independent of the number of independent variables n_u of $\mathbf{G}(\mathbf{u})$. Though this seems to be simply an analogy to the costs of the FM (see Eq. (3.16)), it is a highly remarkable result. According to this, it is possible to build the gradient of a multivariate function $G : \mathbb{R}^{n_u} \rightarrow \mathbb{R}$ with an arbitrary number of independent variables with a remarkable low effort. In general, the reverse mode appears to be the most efficient way to build gradients automatically, or in general, when the number of dependent variables is significantly smaller than the number of independent variables $n_y << n_u$.

3.3.2 Algorithmic differentiation implementations

Besides the aforementioned AD modes that can be used to accumulate derivative information (FM or RM), for the user of AD it is important to know the two main different types of software realizations of AD. While normal AD users cannot do much about AD implementations types, but to accept them, knowledge about them is still important to know how to integrate derivative generation in the modeling workflow. The two most important ways of implementing AD are source code transformation (ST) and operator overloading (OO). Both are shortly introduced next. A detailed comparison and discussion is given by Bischof and Bücker [19].

Source code transformation (ST)

Source code transformation, also called code source-to-source transformation, denotes a set of techniques that transforms the source code for evaluating a function in a source code that explicitly evaluates the derivative of the original source code. If the generated code uses standard data types and standard operators of the considered programming language[8] derivative information of higher order can easily be built. Besides that, the derivative evaluation can immensely profit from optimizations that the compiler can perform on the resulting code.

Regarding the workflow required for integration of ST derivative information, there are no standardized methods. The effort required to generate derivative information with a ST tool depends completely on the AD tool. While many tools require different steps to create and integrate the derivative information, including the definition of additional files, or the creation of a new main program, some other permit the creation and integration of derivatives with one single call. In either case, the required modifications can be automatized through code generation techniques. Some well-known tools that implement ST techniques for derivative generation of code written in different languages are summarized in Section 3.3.3.

[8] This is not always the case. Consider for example ADiMat [20].

Operator overloading

The second classical way to implement AD is the operator overloading (OO) approach. It takes advantage of the overloading capabilities provided by modern computer languages, that is, the possibility to redefine the meaning of elementary operators. Operators and data types can be defined in such a way that not only the required elementary operation is evaluated but also its associated derivative data. For example, if the assignment `y = sin(x)` is evaluated in runtime, the associated derivative assignments `d_y = cos(x) * d_x` are evaluated and further propagated as well.

In general, the integration of OO for derivative generation is an easy task, since almost no modification of the source code is required. On the other hand, debugging and the evaluation of higher order derivatives are much less intuitive. Additionally, since the evaluation of derivatives takes place *on runtime* there is no room for faster execution through compiler optimizations. Some well-known tools that implement OO techniques for derivative generation of code written in different languages are summarized in Section 3.3.3.

3.3.3 Overview of well-known AD tools

AD tools for high-level programming languages

Besides the previously discussed AD modes (FM and RM) and implementation strategies (OO and ST) a further important criterion of categorization of AD tools is given by the language for which the tool provides AD capabilities. Table 3.1 lists some well-known AD tools that have been considered, tested or evaluated within this thesis. A more detailed, though not very up-to-date comparison of different academic and commercial AD tools is provided by Slawig [191]. An extensive and continuously growing list of AD tools is provided in `www.autodiff.org`, the portal of the automatic/algorithmic differentiation community.

For the evaluation of the type of problems treated in this thesis, particular focus was put on tools available for free academic use that support both modes of AD. An additional important point is the support for higher order derivatives, which is fulfilled by most tools listed above, at least for the evaluation of second order derivatives.

AD tools for modeling languages

While the tools listed in Table 3.1 have the main goal of enabling the user-tailored generation of derivatives for given input programs, usually written in high-level, general-purpose programming languages, AD techniques have also found propagation in general-purpose

Table 3.1: Overview of popular AD tools considered in this thesis

Name	ADOL-C	ADIC*	ADiMat	TOMLAB/MAD	CasADi
Source	[220]	[22]	[20]	[53]	[7]
Language	C/C++	C	MATLAB	MATLAB	C/C++,Python, MATLAB
Implementation	OO	ST	ST	OO	ST**
AD Mode	FM, RM	FM	FM, RM	FM	FM, RM
Free academic use	✓	✓	✓		✓
Easy-to-use-drivers***	✓		✓		✓

* This refers to the ADIC version 1.2. The newer ADIC 2 is not yet fully available

** Though it originally supported OO and ST, currently only ST is supported

*** This point is based on the subjective opinion of the author as an AD user

mathematical programming systems such as AMPL or GAMS, and domain-specific modeling environments like Jacobian® (RES Group Inc). By means of AD [see 63, 64] AMPL generates derivative information required by the linked solvers. A similar approach is supported by Jacobian®, in which the derivative information required by solution algorithms is provided by AD techniques described in [201, 204].

These latter AD implementations tools are specially tailored to fulfill the requirements of given solver interfaces and do not offer the possibility of using or reusing the generated derivative information as part of a model. The generality and flexibility given by open-form equation-oriented modeling is not given. Therefore, recently efforts have been done to apply AD methods to equation-based modeling languages, while keeping the inherent generality of the equation-based approach. In Section 3.3.4, challenges related to these efforts are discussed and current implementations are discussed.

3.3.4 Characteristics related to equation-oriented model representations

In order to understand the peculiarities of AD of modeling tools that work with equation-oriented model implementation in comparison to classical AD tools, it is recommendable to recall the concepts of *closed-form* and *open-form* implementations previously discussed in Section 2.1.1. In few words, it can be summarized that classical AD tools have been specially designed for the differentiation of computer programs/mathematical models that consist of assignments, control loops, and further constructs common to procedural/imperative languages, hence for model implementations available in a closed-form.

As discussed in Section 3.3 the key concept of classical AD tools is that the computer program/mathematical model to be differentiated is decomposed into elementary unary and binary operations and assignments that are evaluated sequentially. Through the introduction of intermediate or temporary assignments that save the values of the elementary operations, it is particularly straightforward/obvious to exploit common subexpressions

that permit an efficient evaluation of the original function and its derivatives. At this level, the decomposition of the original function in assignments with a clear causal relation between them plays a key role. Note however, that these causal relations are in general not available when considering an equation-oriented formulation. For that reason, for the implementation of equation-based AD other algorithmic concepts are required. Some of the key concepts are discussed in [163] and [47]. Of particular importance is the need to keep the equation-based nature of the models in the generated derivative information, that means the resulting derivative information should be seen as derivative equations. As previously discussed, equations are more general than assignment statements.

In the discussion on the fundamentals of modeling led in Chapter 2 tool-independent implementation approaches for open-form models were introduced as an important contribution to enhance the generality and reusability of open-form models. For the sake of completeness, in the following we list corresponding efforts to implement AD techniques within this equation-oriented languages and tools.

Modelica - ADModelica

ADModelica is a tool developed by Elsheikh and Wiechert [48] [46] that has been designed for the generation of sensitivity equation systems of models formulated as DAEs. It implements the source code transformation approach in forward mode on XML representations of flattened/expanded Modelica equations and generates sensitivity equation systems that result (are translated back) in additional Modelica equations. An advantage of applying AD on the equation level is that the generation of derivatives is independent of the Modelica compiler and that the generated derivatives can be accessed by any Modelica compliant simulation environment. Note further, that the generated sensitivity equation system can be further optimized by standard Modelica compiler techniques available through the used simulation environment.

CAPE OPEN's Equation Set Object (ESO) - AC-SAMMM [9]

The CAPE OPEN's Equation Set Object represents a *convenient* and *practical* interface to access, transfer or exchange model information required for the solution of simulation tasks involving systems described by AEs and DAEs. A reasonable way to spread the advantages of the ESO interface towards more advanced optimization-based applications would require extending the ESO specification with support of higher order derivative information and to couple the ESO supporting model implementations with methods for generation of high-order derivative information. Such an approach has been done within

the development of AC-SAMMM (The AaChen platform for Structured Automatic Manipulation of Mathematical Models) [80, 184].

AC-SAMMM is not an AD-tool but a software platform that has the goal of generating and providing highly optimized model and derivative code out of a model representation written in high-level equation-based modeling languages [184]. The implementation shown in [80, 184] applies these ideas on Modelica model implementations. Practical applications have been done in the context of high order sensitivity analysis of DAE systems.

In contrary to the other two approaches introduced in this section, ADModelica and ADiCape, which are based on AD techniques for mathematical models represented through equation systems, AC-SAMMM follows a so-called compilation-AD approach [47]. That means, the original mathematical model implementation is translated in a C/C++ compliant procedural/imperative code representation of the residuals on which classical AD methods are applied. The AD step is performed by `dcc`, a derivative code compiler that works with source transformation techniques [151]. The big strength of `dcc` lies in the possibility easily generating higher order derivative information. This is possible since the code generated by `dcc` only uses the same operators and data types as the original code, defined by a reduced set of the C programming language. This is an example of the advantage given by the support of a small set of operators and programming language elements.

Though this approach is applied on models, that are originally available as open-form (equation-oriented) formulation, it should be remarked that the equation-oriented nature of the original model formulation gets lost with the application of the classical AD techniques. This, however, is not in conflict with the philosophy of the Equation Set Object. This latter only requires that the residuals and corresponding derivative information are evaluated properly. It is not of relevance, whether their evaluation follows from single algebraic expressions given by equations or through sequences of assignments.

XML/MathML technologies - ADiCape

ADiCape, a tool for automatic differentiation of mathematical models written in the CapeML modeling language (see Section 2.2.3), is another example that shows the feasibility of the application of AD techniques in an equation-based modeling paradigm [21, 163]. It is interesting to remark that some of the limitations or shortcomings of ADiCape, which are discussed in detail in [163], are related to the simplicity of the algebraic input supported by CapeML.

[9] See Section 2.2.2 for a more detailed explanation on CAPE-OPEN's Equation Set Objects.

ADiCape transforms a given model expression coded in CapeML into equations for derivatives also expressed in CapeML, so that the equation-oriented nature, and the advantages of CapeML are kept in the augmented version of the process model. Besides developing the AD code transformation by using eXtensible Style-sheet Language Transformations (XSLT) templates, Petera [163] developed methods for the evaluation of CapeML models and derivatives for simulation and optimization purposes. For this purpose, an executable ESO (object) is created out of the XML representation of the model and its derivatives. The information contained by the ESO object can then be used by the interfaced software.[10]

3.4 Derivative Evaluation Methods in Process Systems Engineering and Thermodynamics

This section discusses how the previously discussed derivative evaluation methods are commonly used in general PSE applications and in calculations involving thermodynamic models. Though thermodynamic models can be considered as just one possible family of models treated within PSE applications, due to their big importance in rigorous process modeling and some peculiarities, they are discussed separately.

It is important to remark that the use of derivative information in PSE and thermodynamic applications is not limited to the generation of derivative information required for solution algorithms but also in many cases derivative information are parts of the models as well.

3.4.1 Derivatives in process engineering applications

In model-based applications that appear in PSE all types of derivative evaluation methods presented in Section 3.2 may appear.[11] Perhaps the most decisive factor that influences the availability or selection of a particular derivative evaluation type is the nature of the model implementation:

- If the model is only available in form of executable binaries, finite differences (FD) approximations become the only feasible approach.

[10] Analogously to the implementation of AC-SAMMM, the considered ESO object does not refer to the one defined by CAPE OPEN's specification [157] but to a modified version that includes second order derivative information.

[11] The following discussion focuses on derivative information required by solution algorithms. A common use of derivative evaluation methods for model generation purposes appears within sensitivity analysis of dynamic systems [see 48, 80, 133].

- On the other hand, if the models are available in form of equation systems coded in the modeling language of some modern equation-based modeling environments, it is common to work with truncation-error free derivatives that are automatically provided by the modeling environments themselves. These are obtained either through symbolic differentiation (e. g. gPROMS, ACM) or AD techniques (e. g. Jacobian®, AMPL, GAMS).

- If, however, the model implementation is written in a high-level general-purpose language, the model developer has the chance of implementing any type of derivative evaluation technique and hence to influence the computational performance. This is however coupled with an additional effort related to the integration of further libraries or toolboxes for derivative evaluation. In spite of additional efforts, there are still many reasons for developing customized user models with high-level general purpose languages. For best possible computational performance and the use of high-end numerical libraries, it is common to write models in Fortran or C/C++. For fast development of prototype model implementations and advanced debugging features, general purpose numerical environments like MATLAB or Python/Numpy are very well suited. In such a case, it is particularly valuable to select the method for derivative evaluation that promises high computational efficiency.

Regarding computational efficiency issues, several authors from different fields have compared different computational derivative evaluation methods. From the field of process systems engineering Tolsma and Barton [202] provide an extensive review and evaluation considering typical Chemical Engineering models. Though they come to the clear conclusion that AD is superior to the other techniques,[12] there are still cases where other approaches perform better (e. g. *SD with linear and bilinear mathematical terms*). It should further be considered that the very strong performance predictions for AD (see Section 3.3.1) are based on theoretical considerations and that this *theoretical* performance is not necessarily reached by all tools. In order to obtain a convincing performance it may be reasonable to evaluate and compare different derivative evaluation and different tools.

On this basis Merchan et al. [133] propose an approach to include evaluation of derivative evaluation methods in the context of model development and realization of simulation tasks. The principle considered to offer generation of derivative information of different types for one and the same model source is shown in Figure 3.1. Starting with a model formulated in the documentation level using LaTeX syntax as input either symbolic derivatives of arbitrary order or AD derivatives can be generated. Through the integration of

[12] These results are also obtained by other authors as well. See for example Griewank [69], Griewank and Walther [71], or Naumann [151].

these concepts in standard modeling workflows the user gets the chance to evaluate the influence of derivative evaluation methods on the performance of his tasks.

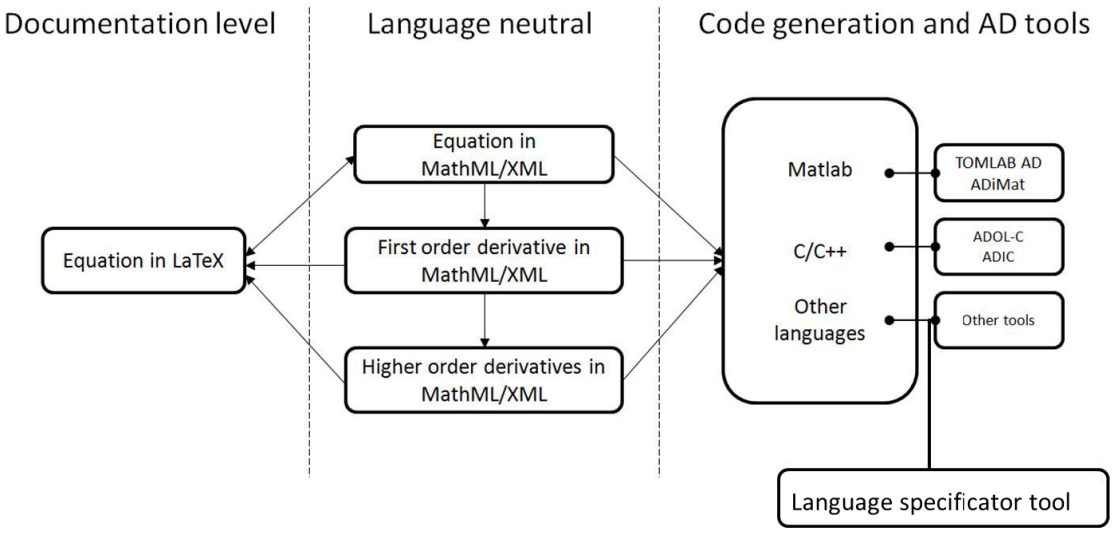

Figure 3.1: Concept for generation of derivative information out of the documentation level[14]

When working with rigorous process models, an important issue is given by the appearance of external function calls (e.g. *physical property calls*) within the models. Their derivatives can be obtained in the simplest case by FD approximations. Alternatively, modern, mostly commercial libraries specialized in physical properties provide derivatives that result from hard coded analytic expressions. However, it should be noted that this derivative information is mostly restricted to first order derivatives and hence not suitable for optimization purposes. Tolsma et al. [204] uses AD techniques to incorporate external function written in Fortran, and their derivatives into the equation-based environment ABACUSS, the predecessor of Jacobian®.[15]

In general, the calculation of derivatives of thermodynamic properties is a decisive and limiting aspect in the solution of complex systems described by rigorous process models. A peculiarity of derivative evaluation concerning thermodynamic models is related to the fact that derivative evaluation is already important at the level of model definition, as clearly shown by the theory presented in Section 2.4.1. More details on these aspects are given in the following Section 3.4.2.

[14]Reprinted from Merchan, Kraus, Barz, Arellano-Garcia, and Wozny [133] with permission from Elsevier.

[15]Within this context, it is interesting to remark that even back in the 80s domain-specific modeling environments such as SpeedUp [155], the predecessor of Aspen Custom Modeler and gPROMS, used by default the CPR algorithm [39] to reduce the computational effort related to the FD approximations of model parts defined by external functions.

3.4.2 Derivative calculations with thermodynamic models

For Chemical Engineering model-based applications, obtaining derivative information of thermodynamic models is not only an issue of high importance for the model implementation and solution processes, but it may also represent a major challenge. Challenges are mostly related to the nature of common implementations, mostly given in *closed-form*, and to their high algebraic complexity.[16]

Regarding the importance of derivative values of external subroutines for the solution of process simulation problems, Macchietto et al. [120] show the overall positive influence of *exact* derivatives at all levels and propose a method for the generation of *exact* derivatives of models available as procedures. Kilakos and Kalitventzeff [99] present the advantages of using analytic/exact derivative of thermodynamic property models and present a systematic approach to obtain partial derivatives of thermodynamic functions with respect to any thermodynamic state variable based on few explicitly *hand coded* derivatives. Mischler et al. [141] were perhaps the first authors from the field of computer-aided process engineering/process systems engineering to discuss the use of AD techniques in their field. They apply AD techniques on thermodynamic property calculation routines and come to conclusions that fit well with the state of the art knowledge of AD, for example that the reverse mode is perhaps the best to build derivatives of functions with scalar outputs such as scalar properties of multicomponent mixtures.

It should be remarked that all contributions discussed in the previous lines and sections concern the generation of derivative information required for numerical solution methods. Derivative information required as part of the model definitions, for example, for the calculation of compressibility factors (see Eq. (2.19)) or fugacity coefficients (see Eq. (2.27) or (2.28)) is mostly supposed to be somehow efficiently hand coded. An interesting point in the discussion presented in Section 2.4.1 regarding the *right set* of independent variables for the expressions of Helmholtz free energy, either $(T, V, \mathbf{n})$ (recommended by Mollerup and Michelsen [144]) or $(T, v, \mathbf{x})$ (recommended by Prausnitz [167]), is that all authors claim to provide an approach which is particularly suitable to exploit common subexpressions when hand coding such derived properties, and hence which not only allow for a more efficient but a better readable implementation.

An important challenge related to the implementation of derivative expressions that are part of the model definition is related to the continuously increasing theoretical depth and thus algebraic complexity of modern thermodynamic models. While for cubic EoS models the implementation of the corresponding derivative expressions does not lead to very

[16] As a matter of fact, the high algebraic complexity, which is often linked with a large model size, is one of the main reasons for implementing models as *closed-form*. Another important reason is the reusability aspect (see also Section 2.1.2).

complex expressions, the situation is different for modern molecularly based EoS models as those discussed in Section 2.4.4. In general, the implementation of such models can become a challenging and highly-error prone task. Specialists interested in disseminating their model developments frequently fill lots of pages of publication appendixes with expressions for the corresponding derivatives of their own developed models. To call a few, for example Gross and Sadowski [74], Tumakaka [208], Xu et al. [224] include extensive appendices with derivative information. While the publication of such expressions is highly appreciated by model implementers and users, as a consequence of the models' high algebraic complexity, the counterproductive situation often appears that those expressions very often contain errors or typos. Other approaches for model dissemination discussed in Section 2.4.3 such as PE 2000 [166] reduce the need of (re)implementing complex EoS models, but still require the model developer to *hand/hard code* the complex derivative expressions and restrict the model user to the evaluation of models supported by the tool.

Besides hand coding, approaches and tools have been developed to enable the automatic generation of the required derivatives, mostly based on symbolic differentiation. On the basis of the computer algebra system REDUCE, Silva and Castier [190] introduced a computational package that generates Fortran subroutine for the evaluation of activity coefficients out of the expressions for the excess Gibbs free energy for an arbitrary number of components. Taylor [195] discusses possible applications of computer algebra software in Chemical Engineering Thermodynamics and introduces a Maple package that among others, supports the generation of expressions for activity coefficients from g^E models and fugacity coefficients from cubic EoS models. Castier [33] presents a comparable package implemented in the Mathematica® programming language. In spite of the practical relevance of these works, it should be noted that the examples treated in the aforementioned works are mostly limited to well-known g^E and (mostly cubic) EoS models, which have a much lower level of complexity than modern molecularly based EoS models. Motivated by the practical limitations of general-purpose CAS software regarding more complex models and the generation of higher order derivatives,[17] Thiéry [200] introduces a C++ library specially designed for the direct generation of derivatives up to the 4^{th} order for models given as analytic expressions of the Helmholtz free energy. Instead of obtaining high order derivatives as analytic expressions (let's say of 3^{rd} order) through sequential application of first order derivatives (obtaining first 1^{st}, then 2^{nd} and finally 3^{rd} order derivatives), numerical values of the required high order derivatives are obtained from the evaluation of *hard-coded* formulas for higher order derivatives, which exploit common subexpressions by default. In spite of these advantages coding the model is rather complex since the Helmholtz free energy is entered in its tree representation.

[17] These limitations are basically the expression swelling for higher order derivatives and the fact that common subexpressions are not efficiently exploited.

Some newer approaches for the generation of exact/analytic derivatives of thermodynamic function of arbitrary high order are currently being followed by the group under Prof. Haug-Warberg at the Norwegian University of Science and Technology (NTNU). While well-known techniques for symbolic differentiation have been designed to produce scalar derivatives, based on the introduction of commutative multidimensional array operations Løvfall [119] introduced a novel method to calculate symbolic gradients (of order) of thermodynamic models while keeping the classical mathematical structure of algebraic models. Though his tool offers the feature to generate C-code of the thermodynamic models and derivatives the model definition is rather cumbersome and restricted to a rather limited set of operators. Moreover, the input of thermodynamic models with implicit terms is not supported, so that the association term common to PC-SAFT or the CPA-EoS cannot be coded. Selvaag [187] developed a light-weight computer algebra system specially designed to handle multidimensional algebraic objects. Though the model definition is much less cumbersome than in the work by Løvfall, the implementation is far from the standard documentation formats. It should be noted that the studies presented in [119] and [187] do not consider advanced EoS models, as such based on association theories.

Though the approaches treated/developed so far appear to be steps in the right direction, to the author's opinion they are still far from fulfilling the vision by Hendriks et al. [84], regarding standardized, well documented model implementations. This thesis presents a new approach that to the author's opinion fulfills this vision more properly. Joining the concepts of documentation-based models, (in particular the **closeness** to the original documentation and the tool independency regarding model implementation and use), together with advanced techniques for derivative evaluations computational efficient model instances of thermodynamic models can be obtained with low implementation effort.

Chapter 4

Embedding Automatic Derivative Generation Techniques within the Development of Documentation-based Models

As previously discussed in Section 2.3, documentation-based models have proven to be a highly suitable approach for coding and using customized user models as they often appear in advanced PSE applications. While in its original implementation by Kuntsche et al. [113, 115] this approach is particularly well suited for the implementation of purely equation-oriented models, the extensions described by Merchan et al. [135] concerning physical property calls and port definitions made the concept also suitable for typical Chemical Engineering problems involving complex thermodynamics. Of course, this approach can only successfully be applied if a proper thermodynamic model is available and accessible.

As the algebraic complexity and amount of potential thermodynamic models increasingly grows; among others, as a consequence of the intense efforts put in the development of advanced molecularly based EoS models (see Section 2.4.4), the modeler concerned with selecting and using proper thermodynamic models is confronted with the necessity of integrating these models in own applications and, if possible, with the necessity of comparing against concurring model alternatives and/or experimental data. Unfortunately, the implementation of such models in most cases turns out to be a highly complex and error-prone task. Limited time resources may prevent modelers to implement and seriously evaluate the models.

An important factor responsible for difficulties in the model implementation is given by the high algebraic complexity of the original expressions for the reduced, residual Helmholtz

free energy and the even higher algebraic complexity of its derivatives which are required for the model implementation as well.[1] In spite of the availability of advanced derivative evaluation methods and tools, as discussed in Section 3.2 and Section 3.3.3 it is still common practice to hand code derivative information that is part of the model. Considering that the implementation of such a single model is not just error prone but very time-consuming task and that the process may need to be done for different concurring models and model implementations, it is desirable to find a way to make this implementation process less complex and expensive.

In this chapter a novel approach for an efficient implementation of advanced EoS models with low coding/programming effort is presented. It combines concepts and advantages of documentation-based models with advanced derivative evaluation techniques. Though the described methodology is completely application independent and could, in principle, be also applied for the generation of general sensitivity equation systems, the main focus is put on advanced EoS models because of their big significance for rigorous process modeling and their current importance. This chapter is organized as follows: On the basis of the hs-PCP-SAFT EoS model, in Section 4.1 the algebraic complexity of advanced EoS models is quantified in the context of two-phase pT-flash calculations. Further, structural issues are discussed that motivate the use of automatic derivative evaluation methods for a reduction of the implementation effort of such models. Section 4.2 then shortly introduces a general approach for generation of derivative information (sensitivities) of an arbitrary open-form model implementation before presenting its application in novel approaches for EoS models formulated as expression of the Helmholtz free energy in Section 4.3. In order to keep the generality and tool independency offered by documentation-based models and MOSAICmodeling, two different methods are discussed. One that enables the work in fully equation-oriented environments (see Section 4.3.1) and one that leads to efficient closed-form implementations (see Section 4.3.2).

4.1 Illustrative Case Study for Advanced EoS Models

As an illustrative example to quantify the complexity of model equations given by derivatives of other equations, in this section a fully equation-oriented formulation of the two-phase pT-flash model formulated in Section 2.4.2 (Eqs. (2.36), (2.37) and (2.38)) is considered. In addition to the aforementioned three generic equations, further generic equations are required for the density root problem of each phase (Eq. (2.19) and Eq. (2.40)), and for the calculation of the fugacity coefficients of each component in each phase (Eq. (2.27)).

[1] Some interesting facts on the complexity of advanced EoS models provided as expressions of the Helmholtz free energy are discussed in [200].

While these six generic equations are fully independent of the thermodynamic model, for the thermodynamic model additional equations are required to define the reduced residual Helmholtz free energy $\tilde{a}^{res}$, and its derivatives $\partial \tilde{a}^{res}/\partial \rho$ and $\partial \tilde{a}^{res}/\partial x_j$. Section 4.1.1 describes an approach for the quantification of the algebraic complexity of the hs-PCP-SAFT EoS model. Section 4.1.2 shows structural properties of the two-phase pT-flash calculation in conjunction with the advanced EoS model which indicate the possibilities of using automatic techniques for derivative evaluation in order to reduce the effort related to the implementation of such models.

4.1.1 Complexity evaluation of the heterosegmented PCP-SAFT EoS model

For the example presented in this section, the heterosegmented PCP-SAFT EoS model, which was shortly introduced in Section 2.4.4, is considered. Though this model presents a high algebraic complexity, mostly due to the heterosegmented character,[2] it should be noted that higher levels of algebraic complexity are also possible, for example, when association or quadrupolar terms are considered as well. In order to implement all algebraic expressions that define this model several sources need to be consulted, in particular the Ph.D. theses of Gross [72], Tumakaka [208] and Schäfer [182]. Due to some inconsistencies in the polar term provided by Schäfer, it is additionally necessary to carefully compare it against the original homosegmented version provided by Gross and Vrabec [76]. Particular care should be taken when bringing together model elements of different sources in order to keep consistent variable namings. Additionally, it should be noted that the derivatives of the polar contribution term with respect to the density and the molar fractions are not published in the open literature. For the following complexity evaluation, the aforementioned derivatives were obtained applying symbolic derivatives and coded by hand.

To allow for a first quantification of the complexity of derivative expressions in comparison to the original model, Table 4.1 compares the number of generic equations and number of characters required to enter the expressions for the Helmholtz free energy and its derivatives with respect to the molar fractions in MOSAICmodeling. Note that the derivatives which are required for the evaluation of compressibility factors and fugacity coefficients are hand coded equations obtained from applying symbolic derivatives to the generic equations that define the reduced, residual Helmholtz free energy $\tilde{a}^{res}$. While the number of generic expressions for derivatives is comparable with the original expressions for $\tilde{a}^{res}$, the number of characters clearly shows that the algebraic expressions that define derivatives are much larger and thus more complex than the original equations. The number of characters

[2] Among others it contains sixfold and fivefold embedded sums and variables with up to 8 different indices.

refers to the effort for entering the model equations as in the original documentation. As an example consider the variable η_k (see Eq. (5.46)) that denotes the packing fraction of the phase k and is coded in MOSAICmodeling as `\eta_{k}`, requiring thus 8 characters for an unambiguous two-dimensional variable representation.

Table 4.1: Comparison of algebraic complexity of expressions for reduced residual Helmholtz free energy against its derivatives

Expressions for	Number of equations	Number of characters
$\tilde{a}^{res}$	32	4833
Z (Eq. (2.19))	14	2855
$\frac{\tilde{a}^{res}}{\partial x_j}$	18	5598

While considering the implementation of generic equations gives a good idea on the complexity and implementation effort, another perspective on the complexity of the model appears when considering the resulting expanded/flattened equation system for calculation of phase equilibria for systems with varying number of components. For the sake of simplicity, this analysis limits to systems consisting of two phases in equilibrium. Table 4.2 gives an overview of the number of equations that result when instantiating/flattening a generic pT-flash equation system for different number of components and segments (see Section 2.4.4). For the considered application a maximal number of two segments is required.

Table 4.2: Complexity evaluation of instantiated/expanded equation-oriented pT-flash equation system

Number of components	Number of equations 1 segment	2 segments
1	194	366
2	447	1603
3	995	4952
4	2037	11949

The numbers in Table 4.2 make clear why a purely equation-oriented model implementation, as it could be automatically generated from MOSAICmodeling, is not expected to be feasible for simulation purposes. Besides the fact that the equation-oriented solution of flash calculations can be very challenging, in particular in case of LLE calculations that strongly tend to converge to the trivial solution, the need to find a large amount of proper starting values makes the usability of equation-oriented solution approaches questionable. However, as common for mostly equation-oriented implementation, if good starting values, that is initial guesses are available, such a fully equation-oriented implementation may come into consideration, in particular in conjunction with environments that auto-

matically generate derivative information. An example of the successful use of a fully equation-oriented formulation of the standard (homosegmented) PCP-SAFT using AMPL is presented in [85] for GLE calculations. In Section 5.3 results of equilibrium calculations are presented that were obtained based on an equation-oriented formulation of the heterosegmented PCP-SAFT EoS model.

An important influence on the obtained model size is related to the way how indices are evaluated by MOSAICmodeling. For example, if there were flexible ways to define index set relations, many unnecessary equations could be ignored. Consider for example ways to make clear that $k_{i=1,a=2,j=2,b=1} = k_{i=2,a=1,j=1,b=2}$ without explicitly writing it as an equation. With a more flexible algebraic model input including index set operations, it should be possible to obtain more compact model implementations.[3] Additional details on the model implementation of the heterosegmented PCP-SAFT EoS model specific to MOSAICmodeling are discussed in Appendix A.1.

4.1.2 Structural properties of two-phase pT-Flash model implementation in conjunction with advanced EoS models

Considering the algebraic expressions defining the reduced, residual Helmholtz free energy and its derivatives as equations, an equation-oriented two-phase pT-Flash model implementation with the heterosegmented PCP-SAFT EoS model consists of 69 generic equations: five pT-Flash equations (Eqs. (2.36), (2.37), (2.39), (2.18) and (2.27)) already listed in Section 2.4.2 and 64 additional equations defining $\tilde{a}^{res}$ and expressions derived from it. Following the description provided in Section 2.4.2, an additional equation appears to be missing, namely Eq. (2.19) that describes how the compressibility factor results based on partial derivatives of the reduced, residual Helmholtz free energy. In fact, this later equation does not directly appear in the considered equation system, since, algebraic expressions for Z are directly provided in the model documentation.

Analyzing the previously mentioned two-phase pT-flash model equations from a different viewpoint, it is possible to categorize its generic model equations in three different subsets:

1. 32 generic equations that only contain algebraic variables: equation system for reduced, residual Helmholtz free energy $\tilde{a}^{res}$.
2. 32 generic equations for the (explicit) definition of variables that can be defined as derivatives of equation from the first subset of equations with respect to other variables of the first subset: equation system for the evaluation of the derivatives

[3] With the update to MOSAICmodeling version 2.1 new possibilities of using indices have been introduced. See `http://www.mosaic-modeling.de/?p=3042`

of $\tilde{a}^{res}$ with respect to ρ and x_j. Since this group of equations is linked with the highest implementation effort, it is of particular interest to automate or somehow replace the generation of these equations.

3. 5 generic equations that represent relations between variables from the two previous subsets (such as Eq. (2.27) or Eq. (2.19) if applied) and further potential variables as well. Note that if Eq. (2.19) is counted to this group, this latter then consists of 6 instead of 5 generic equations. In such case, the second group would contain one equation less.

In accordance with Table 4.1, the biggest part of the model consists of equations that define derivative evaluations. On the basis of this general categorization obtained from a specific case, in Section 4.2 a general approach is developed to automatically generate derivative information from models provided as open-form/equation-based implementations. In Section 4.3 this approach will be used for the automatic generation of derivative information of advanced EoS models provided as open-form implementations of the corresponding Helmholtz free energy function. One of the resulting approaches enables to automatically obtain an efficient *procedural model implementation* including derivative information out of the open-form documentation-based formulation.

4.2 General Approach to Generate Derivative Information out of an Equation-Oriented Model Implementation of a Part of a Model

On the basis of the previously discussed two-phase pT-flash equation system and the general mathematical structure recognized and discussed in Section 4.1.2, in the following a system independent approach is discussed to automatically generate derivative information from a part of a model. Generally speaking, it is an application of the implicit function theorem.

The starting point of the considerations is a general and already specified nonlinear equation system represented by the multivariate function $\mathbf{F} : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_x}$ (Eq. (4.1).

$$\mathbf{F}(\mathbf{x}) = \mathbf{0} \tag{4.1}$$

According to the previous considerations (see Section 4.1.2) $\mathbf{F}$ consists of three different types of subsystems, each of which has different properties. For the sake of simplicity, these are simply called $\mathbf{F_1}(\tilde{\mathbf{x}}_\mathbf{1})$, $\mathbf{F_2}(\tilde{\mathbf{x}}_\mathbf{2})$, and $\mathbf{F_3}(\tilde{\mathbf{x}}_\mathbf{3})$, so that

$$\mathbf{F}(\mathbf{x}) = [\mathbf{F_1}(\tilde{\mathbf{x}}_\mathbf{1}), \mathbf{F_2}(\tilde{\mathbf{x}}_\mathbf{2}), \mathbf{F_3}(\tilde{\mathbf{x}}_\mathbf{3})]^T \tag{4.2}$$

The vector of unknown variables $\mathbf{x}$ can contain different types of variables depending on the considered equation subsystem: purely algebraic variables, variables that represent derivatives of (potentially unknown) variables with respect to other (potentially unknown) variables, or variables that represent relations between the two former types of variables. In the examples used above, $\tilde{\mathbf{x}}_\mathbf{j}$ with $\mathbf{j} \in 1, 2, 3$ is used to denote any group or combination of variables that appears in the respective equation subsystem $\mathbf{j}$.

According to the aforementioned considerations the first equation subsystem $\mathbf{F_1}$ consists only of algebraic variables. These can be further classified into two different variable classes: variables that may become differentiated ($\mathbf{x_1}$) and variables with respect to which derivatives are built $\mathbf{p}$, so that $\tilde{\mathbf{x}}_\mathbf{1} = [\mathbf{x_1}, \mathbf{p}]^T$ and consequently $\mathbf{F_1} = \mathbf{F_1}(\mathbf{x_1}, \mathbf{p})$. The subscript $\mathbf{1}$ in $\mathbf{x_1}$ indicates that it refers to variables that can appear in the subsystem $\mathbf{F_1}$.

The second equation subsystem contains the equations that explicitly define the derivatives of some algebraic variables with respect to others, or following the notation discussed above $\partial\,\mathbf{x_1}/\partial\,\mathbf{p}$. This information can follow from differentiation of $\mathbf{F_1}$ with respect to $\mathbf{p}$. Applying the chain rule to $\mathbf{F_1}(\mathbf{x_1}, \mathbf{p}) = 0$ this results in Eq. (4.3).[4]

$$\mathbf{J}_{\mathbf{1},\mathbf{x_1}} \cdot \frac{\partial \mathbf{x_1}}{\partial \mathbf{p}} + \mathbf{J}_{\mathbf{1},\mathbf{p}} = 0 \tag{4.3}$$

so that the subsystem $\mathbf{F_2}(\tilde{\mathbf{x}}_\mathbf{2})$ can be more generally written as

$$\mathbf{F_2}\left(\mathbf{x_1}, \frac{\partial \mathbf{x_1}}{\partial \mathbf{p}}, \mathbf{p}\right) = 0. \tag{4.4}$$

In Eq. (4.3) $\mathbf{J}_{\mathbf{1},\mathbf{x_1}}$ denotes the Jacobian of $\mathbf{F_1}$ with respect to $\mathbf{x_1}$, $\mathbf{J}_{\mathbf{1},\mathbf{x_1}} = \partial\,\mathbf{F_1}/\partial\,\mathbf{x_1}$, and $\mathbf{J}_{\mathbf{1},\mathbf{p}}$ the corresponding Jacobian containing derivatives with respect to $\mathbf{p}$. Eq. (4.3) is equivalent to a sensitivity equation system. That means the required derivative information can be obtained as the solution of a matrix linear equation system given by Eq. (4.5). Since the model $\mathbf{F_1}$ is available as open-form formulation the required Jacobians can easily be obtained by some of the methods described in Section 3.2.2. However, since the values of the Jacobian have an influence on model equations, which still need to be solved, it is preferable to generate derivative information free of truncation errors.

$$\mathbf{J}_{\mathbf{1},\mathbf{x}} \cdot \frac{\partial \mathbf{x_1}}{\partial \mathbf{p}} = -\mathbf{J}_{\mathbf{1},\mathbf{p}} \tag{4.5}$$

[4] It is important that the variables are properly sorted, so that $\dim(\mathbf{x_1}) = \dim(\mathbf{F_1})$. It should be considered that $\mathbf{p}$ may contain variables not listed as unknown variables of Eq. (4.1) $\mathbf{x}$, e.g. design/specified variables with respect to which derivatives are required. For example, temperature T or pressure p of a pT-flash calculation.

Following the general problem structure derived from the study of the two-phase pT-flash equation system, the third equation subsystem $\mathbf{F_3}(\tilde{\mathbf{x}}_\mathbf{3})$ is nothing more than a general algebraic equation system that may combine all different types of variables discussed up to now with additional variables that build relations with the former. This is more clearly described by Eq. (4.6)

$$\mathbf{F_3}(\tilde{\mathbf{x}}_\mathbf{3}) = \mathbf{F_3}\left(\mathbf{x_1}, \mathbf{p}, \frac{\partial \mathbf{F_1}}{\partial \mathbf{p}}, \mathbf{x_3}\right) \tag{4.6}$$

where $\mathbf{x_3}$ denotes the vector of algebraic variables that only appear in the third equation subsystem $\mathbf{F_3}$.

Following all considerations discussed above, a method results to automate the generation of derivative information that builds a significant part of a considered model $\mathbf{F}(\mathbf{x})$, and that otherwise would be obtained through a combination of differentiation rules and hand coding. As made clear by Eq. (4.7) there is no need to code equations that define derivatives, since the corresponding derivative values can be obtained from the solution of the sensitivity equation of $\mathbf{F_1}$. This sensitivity equation can be automatically generated and its solution corresponds to the solution of a matrix linear equation system. Note that the computational effort for the solution of the sensitivity equation system is almost independent of the number of required derivatives, that is the dimension of $\mathbf{p}$, since one and the same L-U-decomposition of $\mathbf{J_{1,x}}$ can be used for the solution of the different linear equation systems with the different right-hand sides. These latter are characterized by the different columns of $\mathbf{J_{1,p}}$.

$$\mathbf{F}(\mathbf{x}) = \begin{bmatrix} \mathbf{F_1}\left(\mathbf{x_1}, \mathbf{p}\right) \\ \mathbf{F_2}\left(\mathbf{x_1}, \frac{\partial \mathbf{x_1}}{\partial \mathbf{p}}, \mathbf{p}\right) \\ \mathbf{F_3}\left(\mathbf{x_1}, \mathbf{p}, \frac{\partial \mathbf{F_1}}{\partial \mathbf{p}}, \mathbf{x_3},\right) \end{bmatrix} = \begin{bmatrix} \mathbf{F_1}\left(\mathbf{x_1}, \mathbf{p}\right) \\ \mathbf{J_{1,x}} \cdot \frac{\partial \mathbf{x_1}}{\partial \mathbf{p}} + \mathbf{J_{1,p}} \\ \mathbf{F_3}\left(\mathbf{x_1}, \mathbf{p}, \frac{\partial \mathbf{F_1}}{\partial \mathbf{p}}, \mathbf{x_3},\right) \end{bmatrix} = \mathbf{0} \tag{4.7}$$

On the basis of the analysis of a fully equation-oriented (open-form) implementation of a two-phase pT-Flash model including rigorous thermodynamics, a general approach has been found that can significantly reduce the implementation effort of model equations defined by complex derivatives of other variables of the model. In Section 4.3, a specific application of this method and an extension thereof are discussed for the implementation of advanced EoS models provided as expressions of the reduced, residual Helmholtz free energy.

4.3 Approaches for Automatic Generation of Derivatives of Advanced Thermodynamic Models

As indicated by the analysis of the structure of the two-phase pT-flash model coupled with the hs-PCP-SAFT EoS model discussed in Section 4.1.1, advanced molecular based EoS models or any other EoS model characterized by highly complex algebraic expressions of the Helmholtz free energy, offer a big potential for the evaluation of advanced methods for generation of derivative information.

The general approach for generation of derivative information presented in Section 4.2 is based on an open-form formulation and generates a matrix linear equation system whose solution provides the required derivative information. Accordingly, it can be categorized as fully equation-oriented approach. For a more detailed description and evaluation see Section 4.3.1. From the perspective of a modeler working with any type of equation-oriented modeling implementation, including documentation-based models, this may appear as a reasonable approach, since the effort for the model implementation is significantly reduced. The fact that the open-form implementation is supported by both, high-level general-purpose languages and domain-specific tools, is an important advantage. However, there are three main reasons that motivate the development of other, perhaps more efficient alternatives:

1. Though working with purely equation-oriented model implementations of complex thermodynamic models is feasible when using good initial values (starting guesses), in the general case, it is rather an interesting, emerging research topic (see e.g. [43, 95]). The state of the art, and most robust approach to work with such models, is still given through the integration of external subroutines.

2. The fully equation-oriented approach from Section 4.2 requires the evaluation of full and, if possible exact, Jacobian information. Depending on the number of considered components/segments and phases of the system the Jacobian $\mathbf{J}_{\mathbf{1},\mathbf{x_1}}$ can become very large.

   The generation of a full Jacobian obtained by conventional symbolic derivatives would generate each analytic expression for each nonzero Jacobian entry in a one-by-one manner, so that common subexpressions will most probably not be exploited properly. Some early works, among others from Topliss et al. [206] and Michelsen and Mollerup [140], claim however, that this is the most decisive aspect for efficient derivative evaluation. An approach, whose success is inherently related with the evaluation of common subexpressions is given by algorithmic differentiation. Though the

structure of the equation system defining $\tilde{a}^{res}$, that is $\mathbf{F_1}$, is the most decisive factor when it comes to efficiently evaluating the Jacobian with AD (see Section 3.3.1), at least at a first sight, it does not appear possible to exploit the advantages of algorithmic differentiation. It is well known, that the advantages of the main modes of AD (the forward and the reverse) mode are particularly strong if the number of dependent and independent variables differ strongly. One of the few restrictive properties of Eq. (4.3) is however, that $\mathbf{J_{1,x_1}}$ must be quadratic.

3. As previously explained in Chapter 3, the classical, and best developed methods of algorithmic differentiation work the best on model implemented as programs with clearly defined evaluation sequences, not in the declarative approach typical for equations, where there is no specification of the order/flow of the computations.

Carefully analyzing these three points it becomes clear, that the way of providing the required type of model and derivative evaluation would require the implementation of the reduced residual Helmholtz free energy as a function or as a sequential procedure with $\tilde{a}^{res}$ as output value. While this seems to conflict with the equation-based nature of an open-form implementation resulting from the documentation-based approach, within the scope of this thesis an approach is developed which remedies this apparent conflict.

By applying structural analysis techniques to the subsystem that defines $\tilde{a}^{res}$, it is possible to convert the equation system in a sequential evaluation approach for $\tilde{a}^{res}$. In other words, it is possible to convert an open-form model formulation in a closed-form formulation which is then converted in a *procedural model implementation*. The resulting model implementation would correspond to a function with more inputs (e.g. T, ρ, $x_1, x_2, \ldots, x_{nc}$) than outputs ($\tilde{a}^{res}$), a system for which the reverse mode of AD should perform the best. In fact, if a function for the evaluation of the Helmholtz free energy is available the required derivatives for the evaluation of compressibility factors, fugacity coefficients or enthalpies all follow from gradients, a derivative element which is known to be efficiently evaluated by the reverse mode of AD.

On the basis of the previous observations, in this thesis two approaches are proposed for the generation of advanced thermodynamic models and its derivatives out of an open-form, documentation-based formulation of the Helmholtz free energy. The basic idea and workflow of these two approaches is explained by Figure 4.1 for a pT-flash equation system. Some further details are discussed in Section 4.3.1 for the fully equation-oriented approach and in Section 4.3.2 for the alternative approach including structural analysis.

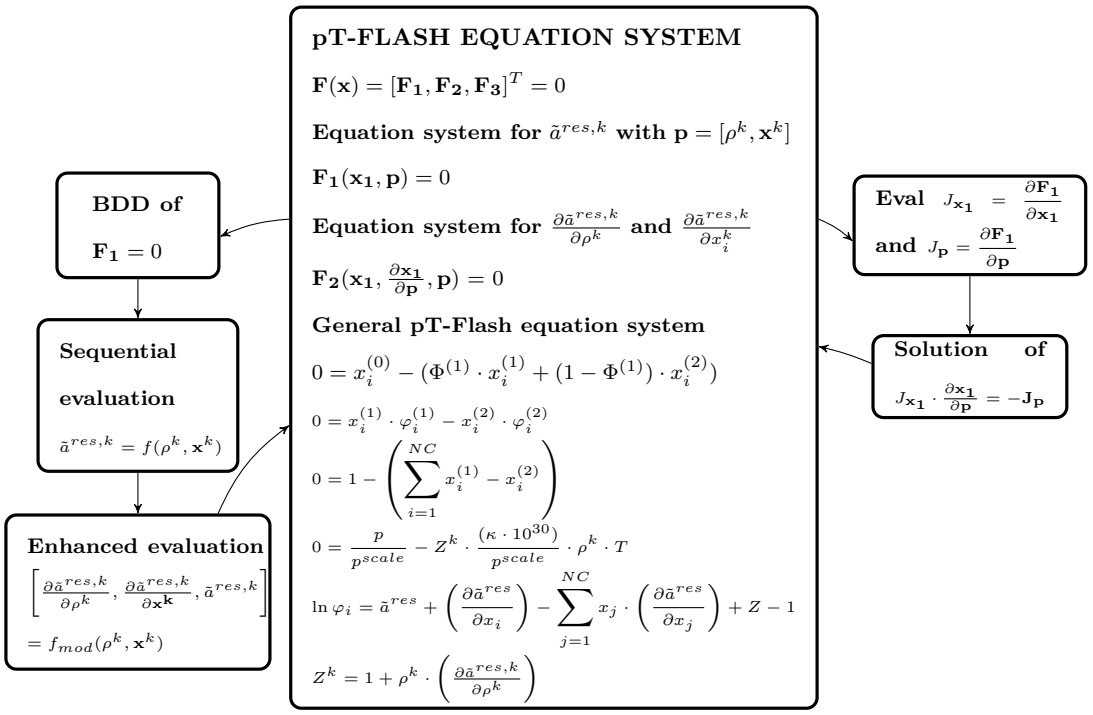

Figure 4.1: Proposed approaches for reduced implementation effort of phase equilibrium calculations shown for a two-phase pT-Flash. Left path: building sequential evaluation with BDD and derivatives with reverse mode of AD. Right path: fully equation-oriented implementation with implicit function theorem[6]

4.3.1 Fully equation-oriented approach based on implicit function theorem

The principles of the fully equation-oriented approach are explained in detail in Section 4.2. The previously discussed categorization of the model equations in three different groups is indicated in Figure 4.1. To make clear the selection of variables $\mathbf{x_1}$ and $\mathbf{p}$ a specific simulation case needs to be specified. For the sake of simplicity, a simplified pure compound flash model is considered. This has the consequence that the considered model is no longer a pT-flash model since only one of the two variables pressure p and temperature T can be fixed. In the following considerations the simplified flash model is considered to evaluate the vapor pressure and the liquid and vapor density ρ^L and ρ^V at a given temperature.

Regarding the variable groups introduced in Section 4.2, the vector $\mathbf{x_1}$ contains all unknowns of the equation system that describes $\tilde{a}^{res}$ for both considered phases implicitly, so that $\mathbf{x_1} = [\tilde{a}^{res}_{k=1}, \tilde{a}^{res}_{k=2}, \dots]^T$, where the index k denotes the two different phases. On the other hand, the vector $\mathbf{p}$ contains the variables with respect to which derivatives are built, $\mathbf{p} = [\rho_{k=1}, \rho_{k=2}, x_{k=1,i=1}, x_{k=2,i=1}]^T$. In this context, the index i denotes the num-

[6] Due to lack of space the expressions for fugacity coefficient are written without a superscript to denote the corresponding phase.

ber of the considered component. The group of variables that build $\partial \mathbf{x_1}/\partial \mathbf{p}$ follows as an obvious consequence of the selection of $\mathbf{x_1}$ and $\mathbf{p}$. Classical examples of variables that only appear in the third group of equations include the pressure, compressibility factors, fugacity coefficients and any other variable that can be derived from $\tilde{a}^{res}$, so that for example $\mathbf{x_3} = [p, Z_{k=1}, Z_{k=2}, \varphi_{k=1}, \dots]$.

As previously discussed for this model implementation, the modeler only needs to code the subsystem $\mathbf{F_1}$ and $\mathbf{F_3}$, since the additional derivative definition equations can be automatically generated. In order for this principle to work, the modeling environment must provide some method to generate the required Jacobians from the available equations and to integrate these generated derivatives as part of the enhanced sensitivity equation system. Though the generation of exact, mostly symbolic, derivatives is usually one of the core functions of advanced equation-oriented modeling environments, these latter were designed to use these derivatives as internal objects interfaced with solution algorithms and hence do not offer straightforward methods to integrate these additional equations in the models. A general approach to overcome these shortcomings is by offering the automatic generation of sensitivity equations at the level of code generation. By doing this, an enhanced model can be supported by any language supported by MOSAICmodeling's code generation.

As it can be seen from the structure of Eq. (4.7), the sensitivity equation corresponds to a linear equation system. Depending on the language or platform selected for code generation this fact can be considered to ease the code generation and solution process. For example, for languages, libraries or environments that have syntax elements for the direct solution of linear equation systems (e.g. MATLAB), code can be generated in such a way that the solution of the linear system is embedded directly in the problem formulation, thus reducing the problem size significantly. Other modeling environments or the solvers linked with these, will automatically recognize linear elements during the analysis or solution procedure and adapt the solution procedure. In the most general cases, however, the considered platform or solver will simply treat the complete problem as a large nonlinear algebraic system without recognizing the linear character of a significant part of the equation system.

To summarize these considerations in few words, it can be said, that the successful solution of the resulting model implementation, which was partially generated automatically, highly depends on the considered platform selected for code generation.

4.3.2 Automatic structure analysis and algorithmic differentiation

The second approach for generation of efficient model implementations of advanced thermodynamic models, shown on the left side of Figure 4.1 relies on two main components:

1. Automatic generation of a procedural model implementation for the evaluation of the reduced residual Helmholtz Free energy $\tilde{a}^{res}$ out of a corresponding equation-oriented, documentation-based model formulation.
2. Advanced derivative evaluation techniques applicable to the procedural model implementation, in particular reverse mode of algorithmic differentiation.

The first point is inspired by the work of Gani et al. [62], which demonstrates how structural/numerical analysis techniques applied on the incidence matrix of an open-form model formulation can be used to find convenient ways of coding (and solving) mathematical problems which include common (property)/thermodynamic models. Kraus [111] describes the implementation of the block diagonal decomposition (this corresponds to the algorithm used in [62]) and the bordered block decomposition of the system's incidence matrix as approaches to generate alternative solution procedures of general algebraic system and discusses its application on the solution of VLE equilibrium calculation with a fully hand coded PC-SAFT EoS model. In particular he shows that the bordered block decomposition gives hints on how the solution approach can be divided in inner and outer loops. This latter aspect is equivalent to the common practice of decoupling the density root problem from the phase equilibrium calculation.

For the generation of the procedural model implementation, the first step is the same as for the fully equation-oriented approach. That means, it is necessary to identify the equation systems that defines the Helmholtz free energy and to classify the variables properly, so as to obtain $\mathbf{F_1}(\mathbf{x_1}, \mathbf{p})$. In the next step the block decomposition is applied on the incidence matrix of $\mathbf{F_1}(\mathbf{x_1}, \mathbf{p})$. For this step to be performed correctly, the variables of the vector $\mathbf{p}$ must be considered as design specifications, independently on the fact whether these are the actual unknowns of the full equation system $\mathbf{F}(\mathbf{x})$ or not.

From the application of the block decomposition on the incidence matrix of $\mathbf{F_1}$ vectors result that indicate in which order which variable (or variable groups) can be obtained from the solution of which equation (or group of equations), so as to solve the full system through the sequential solution of single subsystems. While a classical implementation of the resulting solution procedure would consist on the application of a general (nonlinear) solver to each single subsystem (solution block) as considered by Kraus [111], an alternative and convenient approach would exploit the possibility of explicitly obtaining the solution

through algebraic manipulation of the single equations (equation blocks). Through the identification of equations that can explicitly be solved, the number of unknown elements of $\mathbf{x_1}$ can be reduced and in the best possible case a sequential evaluation procedure for $\tilde{a}^{res}$ can be obtained. This optimal case is indeed fulfilled when applying the procedure to the open-form formulation of the hs-PCP-SAFT EoS model as considered in this work. Challenges resulting from equation (subsystems) that cannot be solved explicitly, for example for equations of association terms, which are normally solved iteratively, can be solved by modifying the code generation in such a way that the solution of the implicit subsystem is done by some general iterative procedure and the corresponding derivative information is obtained from the implicit function theorem.

Having generated a procedural evaluation, the next step consists on the application of a proper method for derivative generation. The steps and effort required to do this depend on the considered language for code generation and the used AD tool. A comparison of the derivative evaluation methods introduced in Section 3.1 applied in the context of advanced thermodynamic models is provided in Section 5.2.1.

While the generation of a procedural model implementation and its derivative information according to this approach can be seen as restrictive in comparison to the fully tool independent equation-oriented approach, taking into account the state of the art methodology of coding and using complex thermodynamic models, this approach can be seen as a significant contribution towards fulfilling the vision of Hendriks et al. [84] towards standard, well documented model implementations. Keeping the principles of the documentation-based approach, it is even possible to generate standard implementation for different languages based on one and the same documentation equivalent model formulation. Considering that all thermodynamic variables can be obtained from derivatives of the Helmholtz free energy, this approach may guide to a partially automated generation of thermodynamic property packages. In the light of well developed, efficient techniques for derivative evaluation it does not appear reasonable to keep on spending a big effort in the development and coding of derivative information by hand.

Chapter 5

Results - Applications to Multiphase Liquid Systems

The methods developed and evaluated within this thesis are motivated by challenges that follow from the application of rigorous process modeling and model-based methods to complex multiphase systems. In particular, within the scope of this contribution a homogeneous catalytic process that takes place in liquid multiphase systems, the rhodium catalyzed hydroformylation of 1-dodecene in a DMF-decane thermomorphic solvent system, is considered in further detail. A short introduction to this process is given in Section 5.1.

Section 5.2 discusses and quantifies some aspects related to the derivative evaluation techniques introduced in the previous chapters. In particular, it is discussed how the selection of derivative evaluation methods and the selection of the independent set of variables of the Helmholtz free energy have an influence on the computational effort required to evaluate EoS models of different levels of complexity. Section 5.3 presents results that show the feasibility of using documentation-based models in conjunction with advanced derivative evaluation methods as a possible approach for the formulation and solution of rigorous process simulation tasks. An evaluation of the different paths shown in Figure 4.1 is provided. Section 5.4 provides a detailed comparative evaluation of thermodynamic models that can be used to describe the phase equilibria that appear in the considered hydroformylation process. Available data and modeling approaches are evaluated and alternative models and parameterizations are proposed. Finally, Section 5.5 presents results related to the plant-wide simulation of the hydroformylation process putting an emphasis on the influence of the selection of the thermodynamic models.

As previously indicated and as it will be shown, in this section several results are presented, which are related to the evaluation of computational effort. It should be remarked that all measurements related to these aspects are performed on a laptop computer with Intel

Core i7-4510U CPU @ 2.00 GHz 2.60 GHz CPU and 12 GB RAM running Windows 7 operating system. More details on the used tools are given in the single subsections.

Also in this section several models are referred which are available in the web-based modeling environment MOSAICmodeling. A detailed list of the location of the models within MOSAICmodeling is given in Appendix A.4

5.1 Rhodium-Catalyzed Hydroformylation of 1-Dodecene in DMF-Decane Thermomorphic Solvent System: Process Description

The use of tunable solvent systems such as micellar solvent systems [82, 149], ionic liquids [174, 189], supercritical CO_2 [92, 102], or thermomorphic solvent systems (TMS) [13, 188] is a promising approach towards the development of highly efficient processes based on rhodium-based homogeneous catalysis. The fundamental idea behind these processes is the unification of the main advantages of Rhodium-based catalysis, namely high selectivities at relatively mild operation conditions, with an efficient recovery of the extremely valuable rhodium catalyst. The considered solvent system work in such a way that the required reaction and separation conditions can be set/tuned by an operation parameter, such as temperature or pressure.

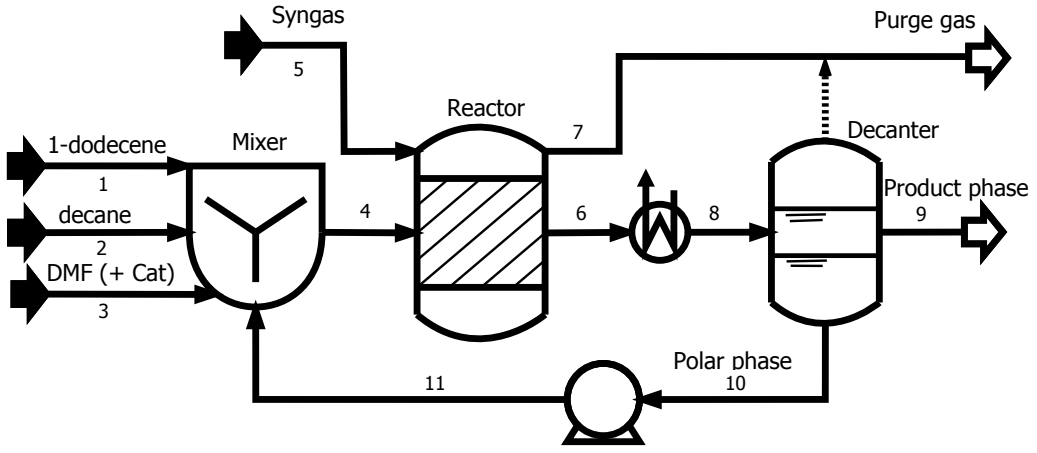

Figure 5.1: Simplified flowsheet of considered hydroformylation process with stream numeration for process modeling purposes. Dotted line makes clear that no decanter gas output is considered in the model[2]

In this work a rhodium-catalyzed hydroformylation process of 1-dodecene in a thermomorphic solvent system is considered. The solvent system consists of the polar solvent N,N-dimethylformamide (DMF) and the nonpolar solvent n-decane. It should be noted that the used BIPHEPHOS modified rhodium catalyst is mainly soluble in the polar phase.

At high, reaction relevant temperatures around 90 °C the multicomponent liquid system, which is completed by the resulting products (see Table 5.1), builds a homogeneous liquid phase that allows a reaction without mass transfer limitations on the liquid side. At lower temperatures, between 5 °C and 25 °C, a phase split takes place so that the liquid mixture leaving the reactor separates in a DMF-rich polar phase (recycle phase) and an organic phase rich on decane and the produced aldehyde. The feasibility of this process was first discussed in [181] based on separate batch experiments and a successful realization of a continuous process at miniplant scale is discussed in [226, 227]. Figure 5.1 provides a simplified flowsheet of the process presented in [226, 227]. The polar and hence catalyst rich phase resulting from the phase separation is recycled to the reactor, while the organic phase (product phase) can be further purified in subsequent downstream steps.

Table 5.1 lists the main components that appear in the considered process. As justified in [136], all appearing isomeric alkenes and aldehydes were respectively lumped into single pseudo-components designated as iso-dodecene and iso-aldehyde. Dodecane results from the hydration of 1-dodecene, which is also considered in the reaction network [98, 123].

Table 5.1: Considered components appearing in the hydroformylation of 1-dodecene in decane/DMF TMS system

Component number	Component name	Component short name
1	1-dodecene	nC12en
2	*n*-tridecanal	nC13al
3	iso-dodecene	iC12en
4	iso-tridecanal	iC13al
5	dodecane	nC12an
6	*N*, *N*-dimethylformamide	DMF
7	*n*-decane	C10an
8	hydrogen	H2
9	carbon monoxide	CO
10	Active catalyst	cat
11	Cat precursor $Rh(acac)(CO)_2$	prec
12	BIPHEPHOS	–

5.2 Influence of Derivative Evaluation Techniques on Computational Efficiency of Advanced EoS Models

In Section 2.4.1 it is shown, which derivatives are required to obtain thermodynamic state variables from an expression of the Helmholtz free energy. Several methods for generation

[2]Reprinted with permission from Merchan and Wozny [136]. Copyright 2016 American Chemical Society.

and evaluation of derivative information have been presented in Section 3.2.

In this section, quantitative evaluations of the computational efficiency of those methods are discussed in different case studies, each of which has a different focus. In Section 5.2.1 a quantitative evaluation of the derivative evaluation methods is given for the generation of gradients of an expression of the Helmholtz free Energy. The examples treated in Section 5.2.2 give a try to bring light into the discussion regarding the proper set of independent variables required for Helmholtz free energy models (see Section 2.4.1). Both aforementioned cases consider implementations of the Helmholtz free energy that are provided as a sequential evaluation procedure, as it can be obtained from applying the **BD**-decomposition to a corresponding equation-oriented formulation. While the results presented in the Sections 5.2.1 and 5.2.2 are mainly focused on computational aspects that can be discussed with selected tool, the contents of Section 5.3 put a higher focus on the multi-tool aspect, one of the main strengths of the documentation-based approaches.

5.2.1 Computational effort for evaluation of gradients

Subject of this first comparison is the generation of gradients of $\tilde{a}^{res}(T, v, \mathbf{x})$, as they are required to build such thermodynamic expressions as compressibility factors Z, or fugacity coefficients of a component i in a mixture φ_i. In order to evaluate the computational effort of different methods, we follow the approach already discussed in Section 3.3.1, by comparing the cost of evaluating the gradients, a directional derivative, against the cost of evaluating the original expression of the Helmholtz free energy, see for example Eq. (3.16) or (3.18). As measure for computational effort the CPU time is considered, this denotes the amount of time that the program keeps the CPU busy [117]. In the following, this time is obtained by using corresponding commands in the considered environments, e.g. `cputime` in MATLAB or `clocktime()` from Python's `time` package.[3]

The results presented in this section are based on the MOSAICmodeling's implementation of the heterosegmented PCP-SAFT EoS model. This implementation is based on model equations provided in different sources [72, 76, 182, 208] and its mathematical input agrees to a very high extent with the notation used in the original publications (see also Appendix A.1). Besides the fundamental expressions for the description of the different contributions to the reduced residual Helmholtz free energy, the fully analytic model implementation available in MOSAICmodeling includes equations that explicitly define compressibility fractions and fugacity coefficients. These latter equations contain some expressions that can be considered as hand coded/symbolic derivatives of the original expressions for the reduced residual Helmholtz free energy.

[3]Note that there are several further time measurement concepts such as elapsed time (also called wall clock time) or user time [see 117], for which other commands exist.

Though additional equations and computational effort regarding the evaluations of the different contributions to the compressibility factor Z can be directly related to the derivatives with respect to density or specific volume (see Eq. (2.19)), for the sake of exactness, in the following evaluations only partial derivatives with respect to molar fractions are considered. In order to quantify the influence of the number of independent variables the computational effort is measured for a varying number of components. All results were obtained using MATLAB release 2013b (The MathWorks, Inc.), since for this environment well developed packages are available, which allow a straightforward generation and evaluation of the derivative methods discussed in Section 3.2. The algorithmic differentiation evaluations are done using the CasADi package version 3.0.0 [7], the derivative approximation methods finite differences (central differences) and complex step derivative approximations using special drivers available in ADiMat [20]. As previously mentioned, the fundamental model for the reduced residual Helmholtz free energy is a sequential evaluation procedure coded in MATLAB obtained from an equation-oriented MOSAICmodeling model. The hand coded/symbolic derivatives are also part of the original MOSAICmodeling model implementation. The gradient information using the forward mode was obtained using an identity matrix as seed matrix. For the reverse mode a 1 was used as seed matrix. Note that the explicit call of the reverse mode leads to the same results as using the driver for gradient evaluation. The results of the evaluation of the computational effort are summarized in Figure 5.2.

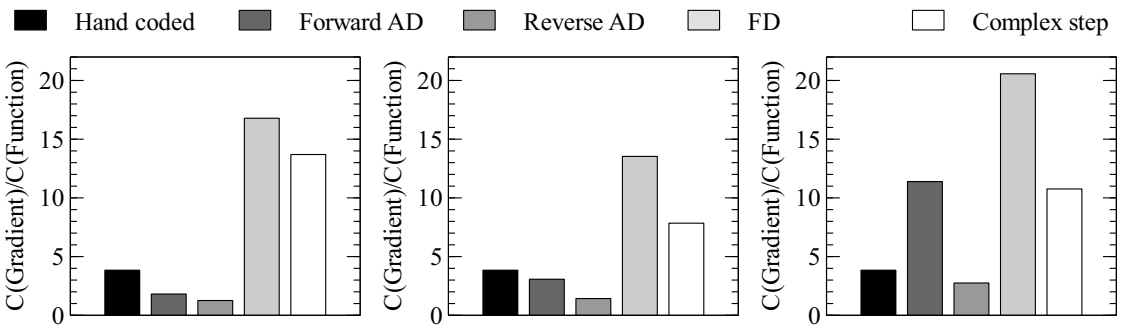

Figure 5.2: Computational overhead of evaluation of directional derivatives versus evaluation of original function. Left: mixture of three components. Middle: mixture of 6 components. Right: mixture of 10 components. See Eq. (3.18)

The overall results for mixtures with 6 (Figure 5.2 middle) and 10 components (Figure 5.2 right) show a very good accordance with the theory discussed previously in Section 3.2. In particular, the forward mode of AD and both approximation methods show a clear proportionality with the number of independent variables, while the overhead of the reverse mode does not depend on it. Regarding the comparison with a mixture of 3 components (Figure 5.2 left), both approximation methods, finite differences and the complex step method,

show a much higher additional effort than expected. This issue is however not related to the methods but to their implementation. As described in the ADiMat documentation,[4] this issue is common for relatively small functions, for which the computational effort is not limited by the size of the code, but by additional operations, inherent to the AD implementation. Though not shown here, this is verified by evaluating alternative FD evaluation modes. For the cases with 6 or 10 components, changing the method from central differences to forward or backward differences leads to a reduction of the evaluation effort by the factor of two. On the other hand, for the case with 3 components, changing the mode of finite differences does not lead to any significant difference. Note that this issue also appears for the algorithmic differentiation methods provided by ADiMat, whose results are not listed here.

In summary, these results clearly show the high potential of using AD, in particular the reverse mode, for an efficient evaluation of gradient information of advanced thermodynamic models. The forward mode of algorithmic differentiation and the complex step approximation also appear as interesting alternatives to obtain truncation error free derivatives as well, but at a much higher computational effort for increasing number of independent variables.

Independent of the fact, whether a particular derivative evaluation method is efficient in comparison with hand coded derivatives, it should be considered that the use of alternative derivative evaluation methods is in general less error prone and requires a lower implementation effort than the implementation of hand-coded derivatives. The latter aspect is particularly true when the derivative evaluation method is integrated through automatic code generation features.

5.2.2 Influence of the chosen set of independent variables : $(T, V, \mathbf{n})$ versus $(T, \rho, \mathbf{x})$ model formulation

As previously mentioned in the beginning of Section 2.4, in literature it is an open question whether $T, V, \mathbf{n}$ [140, 144, 145] or $T, \rho, \mathbf{x}$ [167, 205, 206] should be chosen as set of independent variables for the implementations of Helmholtz free energy function models. The supporters of each variable set claim that their corresponding approaches are particularly suitable in facilitating the exploitation of common subexpressions when hand coding derived properties, thus leading to less complex expressions and a reduced implementation effort.

Based on the fact that algorithmic differentiation (AD) methods inherently exploit common subexpressions in sequential evaluation procedures [71], in this section AD is used

[4] See `http://adimat.sc.informatik.tu-darmstadt.de/doc/`

to compare both implementation approaches. In order to allow for a general statement, several different thermodynamic models of increasing complexity are considered: a simple cubic EoS model (Soave-Redlich-Kwong EoS [193]), a cubic EoS model with mixing rules based on a g^E model (SRK-MHV2 [40] with NRTL [173] as underlying g^E model), and a simplified association model considering only chain and dispersion contributions (s-PC-SAFT [215]). For the quantification of computational effort the CPU-time for the evaluation of fugacity coefficients of a multicomponent mixture phase will be considered. In addition to considering the natural equation selection corresponding to the chosen set of independent variables, namely Eq. (2.28) for $(T, V, \mathbf{n})$ and Eq. (2.27) for $(T, \hat{\rho}, \mathbf{x})$, it will be evaluated, how these two latter equations can be used for a model originally formulated in a different set of independent variables.

On the basis of the experience with several AD tools, including the results presented in Section 5.2.1, the reverse mode from CasADi version 3.0.0 was chosen to generate the required gradients and the fugacity coefficients. For a better comparability the models that are fully analytically available are also evaluated with CasADi. This means that the sequential expressions are evaluated as a CasADi object. The results discussed next were obtained with Python model implementations.[5]

In the following, model formulations and obtained results are presented for each single considered thermodynamic model. Though all considered models are implemented and hence documented in MOSAICmodeling (see Appendix A.4), for the sake of a better understanding, all model equations are listed below.[6] At the end of this section a summarizing discussion of the most important results is provided.

Soave-Redlich-Kwong EoS model (SRK)

Due to the relatively low algebraic complexity of this model and the fact that recommended analytic expressions are available for both sets of independent variables, this model permits a detailed evaluation of the computational effort for the evaluation of compressibility factor Z and fugacity coefficients φ_i.

Analytic expressions for reduced Helmholtz free energy with independent variables $(T, V, \mathbf{n})$

In the following, analytic expressions are given for the reduced residual Helmholtz free energy of a mixed phase k of NC components as proposed by Mollerup and Michelsen [144, 145] $F^{red,res} = A^r/(R \cdot T) = n \cdot \tilde{a}^{res}$. It should be noted that the order in which

[5] Python 2.7.9, provided with Anaconda 2.2.0 (64 bit).

[6] Since the model equations in Section 5.2.2 are all available in MOSAICmodeling, the corresponding variable names are not separately considered in the nomenclature of this thesis. See Appendix A.4 for further details.

the model equations are introduced next is different than the order required to build a sequential evaluation procedure. The reduced Helmholtz free energy of a phase k is given by Eq. (5.1)

$$F_k^{red,res} = n_k \cdot \ln(\frac{V_k}{V_k - B_k}) - \frac{D_k}{B_k \cdot R \cdot T} \cdot \ln(\frac{V_k + B_k}{V_k}), \tag{5.1}$$

where T stands for the temperature, V_k for the extensive volume and n_k for the total mole numbers in phase k ($n_k = \sum_{i=1}^{NC} n_{k,i}$). The mixture parameter B_k (Eq. (5.2)) results from a linear mixture rule of the covolume parameter b_i^{pure} (Eq. (5.3)), which depends on the critical temperature T_i^c and pressure p_i^c.

$$B_k = \sum_{i=1}^{NC} n_{k,i} \cdot b_i^{pure} \tag{5.2}$$

$$b_i^{pure} = 0.08664 \cdot \frac{R \cdot T_i^c}{p_i^c} \tag{5.3}$$

The mixture parameter D_k is given by Eq. (5.4)

$$D_k = \sum_{i=1}^{NC} \sum_{j=1}^{NC} n_{k,i} \cdot n_{k,i=j} \cdot a_{i,j}, \tag{5.4}$$

where $a_{i,j} = (a_i^{pure} \cdot a_{i=j}^{pure})^{0.5} \cdot (1 - k_{i,j})$ denotes the mixing rule of the EoS energy parameter in physical term a_i^{pure} (Eq. (5.5))

$$a_i^{pure} = 0.42747 \cdot \frac{(R \cdot T_i^c)^2}{p_i^c} \cdot \Omega_i^{pure}. \tag{5.5}$$

Ω_i^{pure} from Eq. (5.5) stands for the commonly called alpha-function given by Eq. (5.6).[7]

$$\Omega_i^{pure} = [1 + m_i^{pure} \cdot (1 - (T/T_i^c)^{0.5})]^2, \tag{5.6}$$

where m_i denotes a pure component parameter that only depends on the component's acentric factor ω_i according to Eq. (5.7).

$$m_i^{pure} = 0.480 + 1.574 \cdot \omega_i - 0.176 \cdot (\omega_i)^2 \tag{5.7}$$

[7] Ω (Omega) was used in the notation to avoid ambiguity with the variable $\alpha = a/(b \cdot R \cdot T)$, which also often appears in the context of cubic EoS models. Note also that the expression given here is only valid for components at undercritical state.

Note that all equations listed above regarding pure component parameters are valid as well for formulations with a different set of independent variables and that all equations above are sufficient to obtain derivatives with AD. The analytic expressions considered for the evaluation of compressibility factors and fugacity coefficients are provided next.

Additional analytic expressions for compressibility factors and fugacity coefficients $(T, V, \mathbf{n})$
In addition to the previously presented expressions, Mollerup and Michelsen [145] provide following general expressions for the calculation of the compressibility factor

$$Z_k = \frac{P \cdot V_k}{n_k \cdot R \cdot T} \tag{5.8}$$

where the pressure is given by Eq. (5.9)

$$P = -R \cdot T \cdot \left(\frac{\partial F_k}{\partial V_k}\right)_{T,\mathbf{n}} + \frac{n_k \cdot R \cdot T}{V_k} \tag{5.9}$$

and the partial derivative required in Eq. (5.9) ($F_V = \left(\frac{\partial F_k}{\partial V_k}\right)_{T,\mathbf{n}}$) is given by Eq. (5.10)

$$F_V = n_k \cdot \left(\frac{1}{V_k} - \frac{1}{V_k - B_k}\right) - \frac{D_k}{B_k \cdot R \cdot T}\left(\frac{1}{V_k + B_k} - \frac{1}{V_k}\right) \tag{5.10}$$

Although these formulations are claimed to be particularly efficient, since many common subexpressions (mainly given in the form of partial derivatives) are evaluated only once, it should be remarked that there is still potential to reduce the computational effort through proper simplifications or reformulations. For example, instead of evaluating Eq. (5.8), considering Eq. (5.9) and Eq. (5.10), it is more efficient to obtain the compressibility factor by directly evaluating Eq. (5.11). Eq. (5.11) results from inserting Eq. (5.9) in Eq. (5.8) and application of algebraic simplifications.

$$Z_k = \frac{V_k}{V_k - B_k} - \frac{D_k}{n_k \cdot R \cdot T \cdot (V_k + B_k)} \tag{5.11}$$

The fugacity coefficients of a component i in a mixture given by a phase k can be obtained by evaluation of Eq. (5.12)

$$\varphi_{k,i} = \exp(F_{n,k} + F_{B,k} \cdot b_i^{pure} + F_{D,k} \cdot D_{i,k} - \ln(Z_k)). \tag{5.12}$$

Most terms that appear in Eq. (5.12) denote partial derivative of the reduced Helmholtz free energy expression: $F_{n,k}$ (Eq. (5.13)) the partial derivative of $F_k^{red,res}$ with respect

to $n_{k,i}$, $F_{B,k}$ (Eq. (5.14)) and $F_{D,k}$ (Eq (5.15)) denote the partial derivative of $F_k^{red,res}$ with respect to B_k and D_k. $D_{i,k}$ (Eq. (5.16)) denotes the partial derivative of the mixing parameter D_k (Eq. (5.4)) with respect to $n_{k,i}$.

$$F_{n,k} = \ln(\frac{V_k}{V_k - B_k}) \tag{5.13}$$

$$F_{B,k} = \left[\frac{n_k}{V_k - B_k} - \frac{D_k}{B_k \cdot R \cdot T} \cdot (-\frac{1}{B_k} \cdot \ln(\frac{V_k + B_k}{V_k}) + \frac{1}{V_k + B_k})\right] \tag{5.14}$$

$$F_{D,k} = -\frac{1}{B_k \cdot R \cdot T} \cdot \ln(\frac{V_k + B_k}{V_k}) \tag{5.15}$$

$$D_{i,k} = 2 \cdot \sum_{j=1}^{NC} n_{i=j,k} \cdot a_{i,j} \tag{5.16}$$

Analytic expressions with independent variables $(T, v, \mathbf{x})$ Following expressions represent the (dimensionless) reduced residual Helmholtz free energy $a^{red,res} = \tilde{a}^{res} = A^r/(N \cdot k \cdot T)$.[8] In comparison with the expressions shown above that depend on the set $(T, V, \mathbf{n})$, it is only necessary to introduce expressions for $a_k^{red,res}$ itself (Eq. (5.17)) and for the mixture parameters a_k (Eq. (5.18)) and b_k (Eq. (5.19)). All further expressions for pure component properties and further expressions remain the same as shown above.

$$a_k^{red,res} = \ln(\frac{v_k}{v_k - b_k}) - \frac{a_k}{b_k \cdot R \cdot T} \cdot \ln(\frac{v_k + b_k}{v_k}) \tag{5.17}$$

$$a_k = \sum_{i=1}^{NC} \sum_{j=1}^{NC} x_{k,i} \cdot x_{k,i=j} \cdot a_{i,j} \tag{5.18}$$

$$b_k = \sum_{i=1}^{NC} x_{k,i} \cdot b_i^{pure} \tag{5.19}$$

Additional analytic expressions for compressibility factors and fugacity coefficients $(T, v, \mathbf{x})$ Following analytic expressions are considered as provided in [165]. The compressibility fraction is given by Eq. (5.20)

$$Z_k = \frac{v_k}{v_k - b_k} - \frac{a_k}{R \cdot T \cdot (v_k + b_k)} \tag{5.20}$$

[8] The variable $a^{red,res}$ is introduced in place of $\tilde{a}^{res}$ since the tilde ˜ above a^{res} is not supported by MOSAICmodeling.

The fugacity coefficient of component i in a mixed phase k is given by Eq. (5.21).

$$\varphi_{k,i} = \exp(\frac{b_i^{pure}}{b_k} \cdot (Z_k - 1) - \ln\left(Z_k \cdot (1 - \frac{b_k}{v_k})\right) + \frac{1}{b_k \cdot R \cdot T} \cdot \left(\frac{a_k \cdot b_i^{pure}}{b_k} - 2 \cdot (a_k \cdot a_i^{pure})^{0.5}\right) \cdot \ln\left(1 + \frac{b_k}{v_k}\right)) \quad (5.21)$$

Following comparison of Eq. (5.12) against Eq. (5.21) it appears that the evaluation of Eq. (5.21) requires less operations. In particular for large amount of components the evaluation of $D_{i,k}$ (Eq. (5.16)) appears to require a significant amount of operations.

Changing the set of independent variables through additional expressions

Regardless of the set of independent variables originally chosen to formulate a model of the reduced residual Helmholtz free energy, it may sometimes be desirable to obtain derivative information with respect to a different set of variables. Following the recommendations of different research groups with respect to the "optimal" set of independent variables, this may even be one of the first steps when implementing models that have been originally made available with a different set of variables.

Using a common workflow, such a task would require replacing some variables, for example the molar fraction of a component i in a phase k ($x_{k,i}$), by different expressions defined with the alternative set of variables ($x_{k,i} := n_{k,i} / \sum_{i=1}^{NC} n_{k,i}$), and additional algebraic manipulations and simplifications in order to obtain new expressions with reduced complexity or a better exploitation of common subexpressions. However, following the philosophy of algorithmic differentiation, instead of introducing and replacing variables and expressions by hand, it is only necessary to define further expressions that define the conversion from one set of variables to the desired one and to put this at the right place of a sequential evaluation procedure. The exploitation of common subexpressions is automatically provided by the computational methods of AD.

It should be considered that advantages in computational performance may not only appear due to a potential reduction of computational effort through derivative evaluation with respect to a given set of variables, but by the fact that the different sets of independent variables may lead to expressions of different levels of complexity. As an example, for known values of the required derivatives, it appears obvious that evaluating a fugacity coefficient with Eq. (2.28) requires significantly less operations than evaluating Eq. (2.27). Additionally, it is worth noticing, that the solution approaches considered in this thesis imply that the density root problem is a part of the equation system to be solved. Recent

work by Pereira et al. [159] on the solution of pT-Flash calculations show that an additional reduction of computational cost can be obtained by using model formulations and solution algorithms that incorporate the volume as an explicit variable. As pointed out in [159], in the context of *higher-than-cubic* advanced EoS models, "'it may be beneficial to adopt formalisms which rely on the more natural $\mathbf{n}, V, T$ canonical ensemble"'.

In the following, the additional sets of equations are listed which are required to convert the considered equation system from a set of independent variables into a different one.

Evaluation of derivatives with respect to v and $\mathbf{x}$ for model originally available for V and $\mathbf{n}$ For an arbitrary set of model equations describing the reduced residual Helmholtz free energy $F^{red,res} = A^r/(R \cdot T)$ with $(T, V, \mathbf{n})$ as set of independent variables, following equations are required to obtain $(T, v, \mathbf{x})$ as independent variables

$$n_{i,k} = n_k \cdot x_{i,k} \tag{5.22}$$

$$V_k = n_k \cdot v_k \tag{5.23}$$

$$a_k^{red,res} = \frac{F_k^{red,res}}{n_k} \tag{5.24}$$

Note that the considered set of equations is sufficient if the total number of moles of phase k (n_k)is known. Otherwise an additional equation is required to define n_k ($n_k = \sum_{i=1}^{NC} n_{k,i}$).

Evaluation of derivatives with respect to V and $\mathbf{n}$ for model originally available for v and $\mathbf{x}$ For an arbitrary set of model equations describing the reduced residual Helmholtz free energy $\tilde{a}^{res} = a^{red,res} = A^r/(n \cdot R \cdot T)$ with $(T, v, \mathbf{x})$ as independent variables, following equations are required to treat $(T, V, \mathbf{n})$ as independent variables.

$$n_k = \sum_{i=1}^{NC} n_{k,i} \tag{5.25}$$

$$x_{k,i} = \frac{n_{k,i}}{n_k} \tag{5.26}$$

$$v_k = \frac{V_k}{n_k} \tag{5.27}$$

$$F_k^{red,res} = a_k^{red,res} \cdot n_k \tag{5.28}$$

Evaluation of computational effort for SRK EoS model

Table 5.2 gives an overview of the considered evaluations. Since this type of evaluation is used for the different EoS models a detailed explanation is provided: the first column is introduced to number each of the single considered cases. The second column indicates the set of independent variables and the considered expressions that define the Helmholtz free energy. The third column indicates, whether Eq. (2.27) or (2.28) is used for the evaluation of the fugacity coefficient and hence indicates, whether additional expressions are required in order to change the set of independent variables. Finally, the last column shows the considered method for the evaluation of derivatives, in this case either hand coded/symbolic derivatives (HC/SD) or algorithmic differentiation (AD) in reverse mode.

Table 5.2: Considered cases for the computational evaluation of the influence of the set of independent variables using the SRK EoS model

Case	Helmholtz Free Energy	Fugacity Coefficient	Derivative Method
1	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,\rho,\mathbf{x})$	HC/SD
2	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,\rho,\mathbf{x})$	AD
3	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,V,\mathbf{n})$	AD
4	$F^{red,res}(T,V,\mathbf{n})$	$\varphi_i(T,V,\mathbf{n})$	HC/SD
5	$F^{red,res}(T,V,\mathbf{n})$	$\varphi_i(T,V,\mathbf{n})$	AD
6	$F^{red,res}(T,V,\mathbf{n})$	$\varphi_i(T,\rho,\mathbf{x})$	AD

As a measure for the evaluation of the computational effort required for each type of mathematical formulation the CPU time required for the evaluation of fugacity coefficients is evaluated. In order to obtain a significant time from the measurement, the equation system is not only evaluated for a large number of components, in this case 50 compounds, but also with a high amount of repetitions, namely ten thousand.

In order to evaluate the fugacity coefficients, an easily verifiable system was selected. The mixture of 50 components is thought to be of 25 components with the properties of methane and further 25 components with the properties of ethane. Since the SRK EoS model does not exhibit the so-called Michelsen-Kistenmacher syndrome [139], the fugacity coefficients, and as a matter of fact any arbitrary property of a mixture phase, has the same value of a binary mixture of comparable component distribution.

The obtained overall results are given in Figure 5.3. According to these, the use of the available analytic expression leads to the lowest computational effort, clearly showing the high quality of the provided expressions. On the other hand, the methods that exploit derivative evaluation with algorithmic differentiation show comparable results while the cases in which the fugacity coefficients are evaluated by Eq. (2.28) seem advantageous in comparison to those evaluated with Eq. (2.27). This last fact is particularly clear when comparing the second and the third bars: in spite of introducing additional equations to

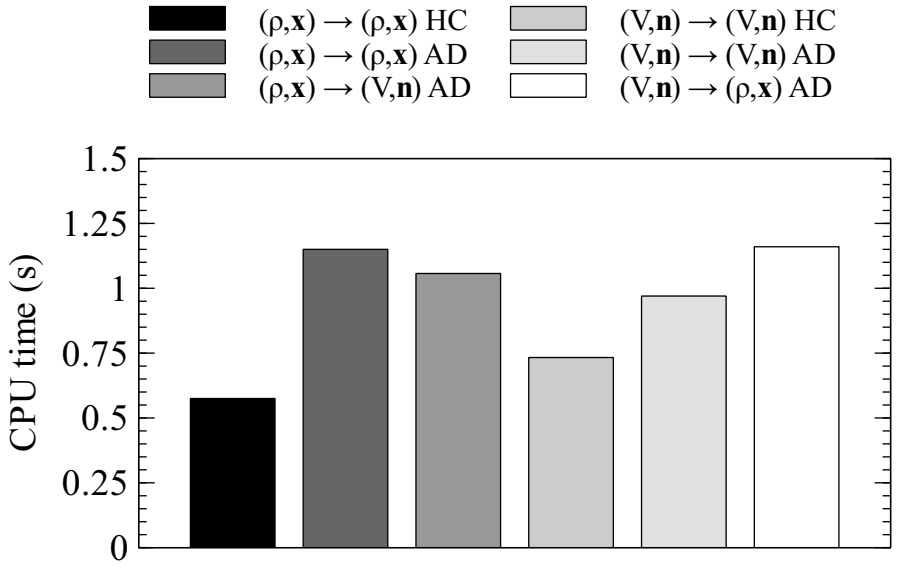

Figure 5.3: CPU times for 10000 evaluations of fugacity coefficients of a mixture consisting of 50 components using the SRK EoS model with different sets of independent variables for $\tilde{a}^{res}$ and φ_i (Results obtained with python)

convert the molar fractions and the molar density in the corresponding extensive variables (third column) the overall effort is lower compared to the second bar, where no additional expressions are required but the fugacity coefficient is evaluated with Eq. (2.27).

Soave-Redlich-Kwong with Huron-Vidal mixing rules of second order (SRK-MHV2) and NRTL as underlying g^E model

The SRK-MHV2 is one of the so-called EoS/g^E models, a type of model that combines a "mostly cubic" EoS model with a g^E model. In case of the SRK-MHV2 model all equations are the same as the original SRK model formulation except for the equation that defines the mixture parameter a_k. This latter results from the largest solution of a quadratic equation that, among others, depends on an explicit evaluation of a g^E model. In the following explanations only the analytic model formulation as published in [40] is considered. This latter considers $(T, v, \mathbf{x})$ as the set of independent variables. In accordance with the practical use of the SRK-MHV2 EoS model in this work, the Non-Random-Two-Liquid model (NRTL) [173] is considered as underlying g^E model.

Analytic expressions with independent variables $(T, v, \mathbf{x})$ In the model implementation Eq. (5.29) defines the reduced residual Helmholtz free energy, where $\alpha_k = a_k/(b_k \cdot R \cdot T)$. Note that

Eq. (5.29) is fully equivalent to Eq. (5.17).

$$a_k^{red,res} = -\ln(1 - \frac{b_k}{v_k}) - \alpha_k \cdot \ln(1 + \frac{b_k}{v_k}) \tag{5.29}$$

According to [40] the value of α_k results from the largest solution of the quadratic equation given in Eq. (5.30), where $\alpha_i^{pure} = a_i^{pure}/(b_i^{pure} \cdot R \cdot T)$, q1 = -0.4783 and q2 = -0.0047. $g_k^{E,red}$ ($g_k^{E,red} = g_k^E/(R \cdot T)$) stands for the reduced or normalized Excess Gibbs Energy of a phase k.

$$0 = q2 \cdot (\alpha_k)^2 + q1 \cdot \alpha_k - g_k^{E,red} - \left(\sum_{i=1}^{NC} x_{k,i} \cdot \ln(\frac{b_k}{b_i^{pure}})\right) - \left(q1 \cdot \sum_{i=1}^{NC} x_{k,i} \cdot \alpha_i^{pure}\right) - \left(q2 \cdot \sum_{i=1}^{NC} x_{k,i} \cdot (\alpha_i^{pure})^2\right), \tag{5.30}$$

In order to ensure that the largest root of Eq. (5.30) is obtained, it is convenient to write equations that define each single root separately and to provide an additional expression to get the largest one. Introducing the variable $abbv1$ to denote the constant term in a quadratic equation

$$abbv1 = -g_k^{E,red} - \left(\sum_{i=1}^{NC} x_{k,i} \cdot \ln(\frac{b_k}{b_i^{pure}})\right) - \left(q1 \cdot \sum_{i=1}^{NC} x_{k,i} \cdot \alpha_i^{pure}\right) - \left(q2 \cdot \sum_{i=1}^{NC} x_{k,i} \cdot (\alpha_i^{pure})^2\right), \tag{5.31}$$

the solutions of Eq. (5.30) are given by Eq. (5.32), where $n_{g=1} = 1$ and $n_{g=2} = 2$.

$$\alpha_{k,g} = \frac{-q1 + [(-1)^{n_g} \cdot ((q1)^2 - 4 \cdot q2 \cdot abbv1)^{0.5}]}{2 \cdot q2},\ g = 1,\ 2 \tag{5.32}$$

To obtain the largest root of Eq. (5.30), it would suffice to evaluate $\max(\alpha_{k,g=1}, \alpha_{k,g=2})$. However, since the $\max(x, y)$ function is not available in MOSAICmodeling, and for numerical reasons, in the considered model implementation a smooth approximation is coded according to Eq. (5.33). *st* denotes a parameter that can be modified to adapt the smoothness of the model approximation. For the application within this thesis, the value of *st* was chosen in such a way that no significant difference between results obtained by the

analytic approach and the approximation appeared. The selected value is available in the model documentation (see Appendix A.4).

$$\alpha_k = \frac{\ln(\exp(st \cdot \alpha_{k,g=1}) + \exp(st \cdot \alpha_{k,g=2}))}{st} \tag{5.33}$$

From the above equations the only expressions that needs to be closed correspond to the reduced model for the excess energy $g_k^{E,red}$. In the current work the NRTL model is considered. Following MOSAICmodeling's notation this model is coded according to Eq. (5.34)

$$g_k^{E,red} = \sum_{i=1}^{NC} \left[x_{k,i} \cdot \frac{\sum_{j=1}^{NC} \tau_{j,i} \cdot G_{j,i} \cdot x_{k,i=j}}{\sum_{l=1}^{NC} G_{j=l,i} \cdot x_{k,i=l}} \right], \tag{5.34}$$

where the temperature dependent energy interaction parameters between j-i pairs of molecules, $\tau_{j,i}$, $G_{j,i}$ and $\alpha_{j,i}$, are considered as in their Aspen Plus default implementation [11], hence with Eqs. (5.35), (5.36) and (5.37) respectively.

$$\tau_{j,i} = a_{j,i} + b_{j,i}/T + e_{j,i} \cdot \ln(T) + f_{j,i} \cdot T \tag{5.35}$$

$$G_{j,i} = \exp(-\alpha_{j,i} \cdot \tau_{j,i}) \tag{5.36}$$

$$\alpha_{j,i} = c_{j,i} + d_{j,i} \cdot (T - 273.15) \tag{5.37}$$

Additional analytic expressions for compressibility factor and fugacity coefficients $(T, v, \mathbf{x})$ Similar to the SRK EoS presented above, additional analytic expressions are required for the compressibility factor and the fugacity coefficients. The compressibility factor is given by Eq. (5.38).

$$Z_k = \frac{v_k}{v_k - b_k} - \alpha_k \cdot \left(\frac{b_k}{v_k + b_k}\right) \tag{5.38}$$

According to Dahl and Michelsen [40] the fugacity coefficient of a component i in a phase k is given by Eq. (5.39), where $n\alpha_{ni,k,i}$ represents the partial derivative of $n\alpha_k$ ($n\alpha_k = n_k \cdot \alpha_k$) with respect to the mole number of component i in phase k ($n\alpha_{ni,k,i} = \frac{\partial(n\alpha_k)}{\partial n_{k,i}}$).[9] To avoid

[9] In the original publication [40] there is a typo in this equation. Erroneously in [40] the variable $n\alpha_{ni,i}$ appears as an additional factor inside the logarithm.

confusion with the notation of Eq. (5.39), it should be remarked that the expression n_i in Eq. (5.39) is a subscript and not an index [115].

$$\varphi_{k,i} = \exp(\frac{b_i^{pure}}{b_k} \cdot (Z_k - 1) - \ln(Z_k \cdot (1 - \frac{b_k}{v_k})) - \ln(\frac{v_k + b_k}{v_k}) \cdot n\alpha_{ni,k,i}) \tag{5.39}$$

The analytic expression that defines $n\alpha_{ni,k,i}$, Eq. (5.40), results from multiplying Eq. (5.30) times n_k and applying implicit differentiation with respect to $n_{i,k}$. While the resulting expression clearly does not seem to have a very high degree of algebraic complexity, it should be remarked that obtaining these expressions requires a relatively large amount of algebraic manipulations. As an example, in [68] the derivation of analytic expressions for the fugacity coefficient of a component i in a mixture with the Predictive SRK (PSRK) EoS model fills several pages.[10]

$$n\alpha_{ni,k,i} = \frac{1}{q1 + 2 \cdot \alpha_k \cdot q2} \cdot (q1 \cdot \alpha_i^{pure} + q2 \cdot ((\alpha_k)^2 + (\alpha_i^{pure})^2) + \ln(\gamma_{k,i}) + \ln(\frac{b_k}{b_i^{pure}}) + \frac{b_i^{pure}}{b_k} - 1) \tag{5.40}$$

The only variable appearing in Eq. (5.40) that needs to be further specified is the activity coefficient $\gamma_{k,i}$. Considering NRTL as underlying g^E model, using MOSAICmodeling's algebraic input this latter variable is defined by Eq. (5.41).

$$\gamma_{k,i} = \exp(\frac{\sum_{j=1}^{NC} x_{k,i=j} \cdot \tau_{j,i} \cdot G_{j,i}}{\sum_{l=1}^{NC} x_{k,i=l} \cdot G_{j=l,i}} + \sum_{j=1}^{NC} (\frac{x_{k,i=j} \cdot G_{j=i,i=j}}{\sum_{l=1}^{NC} x_{k,i=l} \cdot G_{j=l,i=j}} \cdot (\tau_{j=i,i=j} - \frac{\sum_{m=1}^{NC} x_{k,i=m} \cdot \tau_{j=m,i=j} \cdot G_{j=m,i=j}}{\sum_{l=1}^{NC} x_{k,i=l} \cdot G_{j=l,i=j}}))) \tag{5.41}$$

Evaluation of computational effort for SRK-MHV2 model Table 5.3 gives an overview of the considered evaluations. The meaning of the single columns of Table 5.3 is the same as explained previously for Table 5.2. In comparison with the SRK EoS model a much lower number of cases is evaluated, since no formulation was found considering the set $(T, V, \mathbf{n})$. As the results indicate, this small set of evaluations still provides interesting insights.

Similarly to the discussion presented previously for the SRK EoS model, the CPU time required for the evaluation of fugacity coefficients is evaluated. For similar reasons listed previously, the equation system is evaluated 10000 times for a mixture of 50 components. As easily verifiable system a mixture of DMF and decane was selected. From the 50

[10] PSRK is a different EoS/g^E model of very similar algebraic complexity.

Table 5.3: Considered cases for the computational evaluation of the influence of the set of independent variables using the SRK-MHV2 EoS model

Case	Helmholtz Free Energy	Fugacity Coefficient	Derivative Method
1	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,\rho,\mathbf{x})$	HC/SD
2	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,\rho,\mathbf{x})$	AD
5	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,V,\mathbf{n})$	AD

components, 25 have the properties of DMF and further 25 the properties of decane. The values of the NRTL binary interaction parameters are taken from Table 5.14. The properties of the mixture phase have the same values as a binary mixture of comparable component distribution.

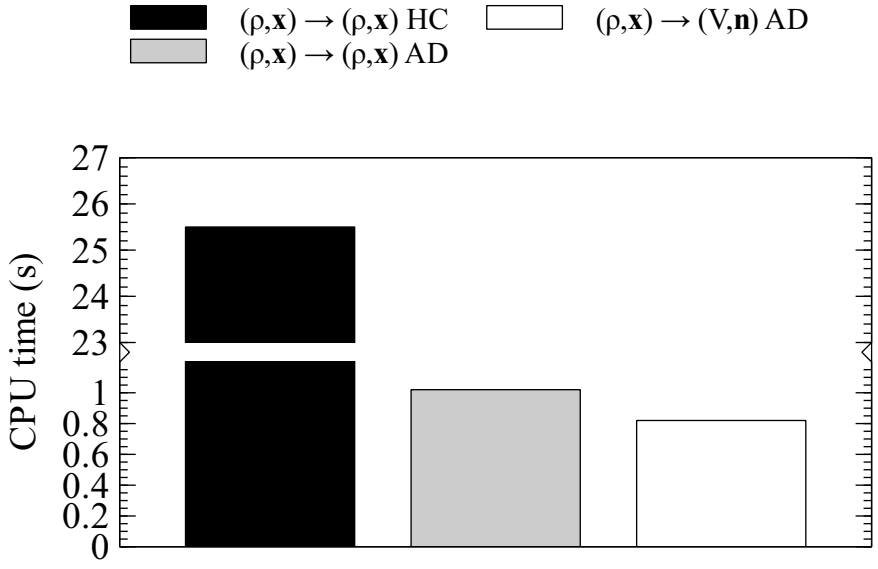

Figure 5.4: CPU times for 10000 evaluations of fugacity coefficients of a mixture consisting of 50 components using the SRK-MHV2 (NRTL) EoS model with different sets of independent variables for $\tilde{a}^{res}$ and φ_i

The obtained overall results are given in Figure 5.4. Unlike the results obtained with the SRK EoS model, in this case the symbolic expressions provided for the evaluation of compressibility factors and fugacity coefficients cause a significantly higher computational effort. Clearly, the evaluation of the additional symbolic expressions required by Eq. (5.39), namely Eq. (5.40) and the very complex Eq. (5.41), require a much higher number of operations than using Eq. (2.27) or Eq. (2.28) in conjunction with gradients gained by the reverse mode of algorithmic differentiation.

As in the case concerning the SRK EoS model, a comparison of the AD formulations

shows that considering the independent variables $(T, V, \mathbf{n})$ for the evaluation of fugacity coefficients leads to the lowest computational effort. This is particularly interesting, since it occurs despite the fact that the model is originally available with the set of independent variables $(T, v, \mathbf{x})$ and that the conversion to the set $(T, V, \mathbf{n})$ requires the introduction of additional expressions (Eq. (5.25)) to Eq. (5.28)). Since the evaluation of the computational time required to (only) evaluate the gradients, leaving out the explicit evaluations of fugacity coefficients, does not show a significant difference between the different sets of independent variables, it follows in this case that the advantages of using $(T, V, \mathbf{n})$ lie mainly in the fact that the evaluation of Eq. (2.28) requires less operations than the evaluation of Eq. (2.27). It is expectable that these differences may become more pronounced for mixtures with a higher number of compounds.

Simplified PC-SAFT (s-PC-SAFT)

In comparison with the original PC-SAFT model of Gross and Sadowski [74], the simplified PC-SAFT EoS model of von Solms et al. [215], s-PC-SAFT EoS, can be seen as a theoretically well-founded modification with simpler algebraic structure but comparable correlative capabilities.[11] The s-PC-SAFT EoS model consists of algebraic expressions for the description of different contributions to the reduced residual Helmholtz free Energy. For the sake of simplicity, in the following only the terms of the hard chain and the dispersion contributions are considered.

Analytic expressions with independent variables $(T, \rho, \mathbf{x})$ The expressions for the reduced residual Helmholtz free energy considered in this model formulation (Eq. (5.42)) are taken from the original publication [215].

$$a_k^{res,red} = a_k^{hc,red} + a_k^{disp,red} \tag{5.42}$$

To evaluate the hard chain contribution of the mixture $a_k^{hc,red}$, Eq. (5.43) is considered. Additional expressions are required for the mean segment number in the mixture mm_k (Eq. (5.44)), the hard sphere contribution of the reduced residual Helmholtz free energy $a_k^{hs,red}$ (Eq. (5.45)), the mean segment diameter of the mixture d_k (Eq. (5.47)), the segment diameter of the pure components d_i (Eq. (5.48)), the packing fraction η_k defined by Eq. (5.46) and the radial distribution function of the hard-sphere fluid g_k^{hs} (Eq. (5.49)).

[11] Within this thesis the s-PC-SAFT EoS model has not been used to describe the thermomorphic solvent system treated in Section 5.1. The considered polar terms evaluated in this work have only been applied to the original PC-SAFT EoS model of Gross and Sadowski [74].

Additional details on the meaning of further variables can be taken from the original literature.

$$a_k^{hc,red} = mm_k \cdot a_k^{hs,red} - \ln(g_k^{hs}) \cdot \sum_{i=1}^{NC} x_{i,k} \cdot (m_i - 1) \tag{5.43}$$

$$mm_k = \sum_{i=1}^{NC} x_{i,k} \cdot m_i \tag{5.44}$$

$$a_k^{hs,red} = \frac{4 \cdot \eta_k - 3 \cdot (\eta_k)^2}{(1 - \eta_k)^2} \tag{5.45}$$

$$\eta_k = \frac{\pi}{6} \cdot \rho_k \cdot mm_k \cdot (d_k)^3 \tag{5.46}$$

$$d_k = (\frac{\sum_{i=1}^{NC} x_{k,i} \cdot m_i \cdot (d_i)^3}{\sum_{i=1}^{NC} x_{k,i} \cdot m_i})^{1/3} \tag{5.47}$$

$$d_i = \sigma_i \cdot (1 - 0.12 \cdot \exp(-3 \cdot \frac{\epsilon_i}{T})) \tag{5.48}$$

$$g_k^{hs} = \frac{1 - \eta_k/2}{(1 - \eta_k)^3} \tag{5.49}$$

Eq. (5.50) defines the dispersion contribution to the reduced residual Helmholtz free energy. Additional expressions are required for the single power series in density that substitute the integrals of the perturbation theory $I1_k$ (Eq. (5.51)) and $I2_k$ (Eq. (5.52)) and the corresponding coefficients of these power series $a_{k,p-1}^{basic}$ (Eq. (5.53)) and $b_{k,p-1}^{basic}$ (Eq. (5.54)), for the abbreviations $abbv1_k$ (Eq. (5.55)), $abbv2_k$ ((5.56)) and $C1_k$ (Eq. (5.57)), and finally for the mixing rules of the segment diameter $\sigma_{i,j}$ Eq. (5.58) and of the depth of pair potential $\epsilon_{i,j}$ defined in Eq. (5.59).

$$a_k^{disp,red} = -2 \cdot \pi \cdot \rho_k \cdot I1_k \cdot abbv1_k - \pi \cdot \rho_k \cdot mm_k \cdot C1_k \cdot I2_k \cdot abbv2_k \tag{5.50}$$

$$I1_k = \sum_{p=1}^{Np} a_{k,p-1}^{basic} \cdot (\eta_k)^{n_{p-1}} \tag{5.51}$$

$$I2_k = \sum_{p=1}^{Np} b_{k,p-1}^{basic} \cdot (\eta_k)^{n_{p-1}} \tag{5.52}$$

$$a_{k,p-1}^{basic} = a_{p-1,ui=0}^{basic} + \frac{mm_k - 1}{mm_k} \cdot a_{p-1,ui=1}^{basic} + \frac{mm_k - 1}{mm_k} \cdot \frac{mm_k - 2}{mm_k} \cdot a_{p-1,ui=2}^{basic} \tag{5.53}$$

$$b_{k,p-1}^{basic} = b_{p-1,ui=0}^{basic} + \frac{mm_k - 1}{mm_k} \cdot b_{p-1,ui=1}^{basic} + \frac{mm_k - 1}{mm_k} \cdot \frac{mm_k - 2}{mm_k} \cdot b_{p-1,ui=2}^{basic} \tag{5.54}$$

$$abbv1_k = \sum_{i=1}^{NC} \sum_{j=1}^{NC} x_{i,k} \cdot x_{i=j,k} \cdot m_i \cdot m_{i=j} \cdot (\frac{\epsilon_{i,j}}{T}) \cdot (\sigma_{i,j})^3 \tag{5.55}$$

$$abbv2_k = \sum_{i=1}^{NC} \sum_{j=1}^{NC} x_{i,k} \cdot x_{i=j,k} \cdot m_i \cdot m_{i=j} \cdot (\frac{\epsilon_{i,j}}{T})^2 \cdot (\sigma_{i,j})^3 \tag{5.56}$$

$$C1_k = (1 + mm_k \cdot \frac{8 \cdot \eta_k - 2 \cdot (\eta_k)^2}{(1 - (\eta_k))^4} + (1 - mm_k) \cdot \frac{20 \cdot \eta_k - 27 \cdot (\eta_k)^2 + 12 \cdot (\eta_k)^3 - 2 \cdot (\eta_k)^4}{[(1 - \eta_k) \cdot (2 - (\eta_k))]^2})^{-1} \tag{5.57}$$

$$\sigma_{i,j} = 0.5 \cdot (\sigma_i + \sigma_{i=j}) \tag{5.58}$$

$$\epsilon_{i,j} = (\epsilon_i \cdot \epsilon_{i=j})^{0.5} \cdot (1 - k_{i,j}) \tag{5.59}$$

Additional analytic expressions for compressibility factor and fugacity coefficients $(T, \rho, \mathbf{x})$ The compressibility factor Z_k consists, as expected, of the corresponding terms of the ideal gas ($Z_k^{IG} = 1$) and further hard chain and dispersion contributions (Eq. (5.60)).

$$Z_k = 1 + Z_k^{hc} + Z_k^{disp} \tag{5.60}$$

Since no analytic expressions were found that define the hard chain contribution, these have been built by applying differentiation rules by hand. The resulting expressions for the hard sphere and the hard chain terms are given by Eq. (5.62) and Eq. (5.61) respectively.

$$Z_k^{hc} = mm_k \cdot Z_k^{hs} - \frac{\eta_k}{g_k^{hs}} \cdot (\frac{2.5 - \eta_k}{(1 - \eta_k)^4}) \cdot \sum_{i=1}^{NC} x_{i,k} \cdot (m_i - 1) \tag{5.61}$$

$$Z_k^{hs} = \frac{2 \cdot \eta_k \cdot (\eta_k - 2)}{(\eta_k - 1)^3} \tag{5.62}$$

All analytic expressions for the evaluation of the dispersion distribution of the compressibility factor are taken from the original PC-SAFT publication [74] and are listed from Eq. (5.63) to Eq. (5.66). It should be noted that minor deviations in comparison with the original notation appear due to inconveniences of the original notation.[12] Such inconveniences made it necessary to reformulate some variable names in a systematic way. In general, such kind of issues could be avoided by using unambiguous notations that follow the recommendations proposed by Kuntsche [113].

$$Z_k^{disp} = -2 \cdot \pi \cdot \rho_k \cdot dI1_k \cdot abbv1_k - \pi \cdot \rho_k \cdot mm_k \cdot [C1_k \cdot dI2_k + C2_k \cdot \eta_k \cdot I2_k] \cdot abbv2_k \quad (5.63)$$

$$dI1_k = \sum_{p=1}^{Np} a_{k,p-1}^{basic} \cdot (n_p) \cdot (\eta_k)^{n_{p-1}} \quad (5.64)$$

$$dI2_k = \sum_{p=1}^{Np} b_{k,p-1}^{basic} \cdot (n_p) \cdot (\eta_k)^{n_{p-1}} \quad (5.65)$$

$$C2_k = -(C1_k)^2 \cdot (mm_k \cdot \frac{-4 \cdot (\eta_k)^2 + 20 \cdot \eta_k + 8}{(1-\eta_k)^5} + (1 - mm_k) \cdot \frac{2 \cdot (\eta_k)^3 + 12 \cdot (\eta_k)^2 - 48 \cdot \eta_k + 40}{((1-\eta_k) \cdot (2-\eta_k))^3}) \quad (5.66)$$

Since in the original publication [215] the set of independent variables $(T, \rho, \mathbf{x})$ is chosen, the fugacity coefficient is evaluated with Eq. (2.27). In MOSAICmodeling's implementation this is coded as Eq. (5.67).

$$\varphi_{i,k} = \exp(a_k^{res,red} + a_{xk,i,k}^{res,red} - \sum_{j=1}^{NC} x_{i=j,k} \cdot a_{xk,i=j,k}^{res,red} + Z_k - 1 - \ln(Z_k)) \quad (5.67)$$

The partial derivative of the residual reduced Helmholtz free energy with respect to the molar fraction $a_{xk,i,k}^{res,red}$ consists of the derivative of the different contributions (Eq. (5.68)). The meaning of the variables is self-explanatory. Similarly to the implementation of the SRK-MHV2, here xk is introduced as subscript to denote derivatives with respect to molar fractions.

$$a_{xk,i,k}^{res,red} = a_{xk,i,k}^{hc,red} + a_{xk,i,k}^{disp,red} \quad (5.68)$$

[12] For example in the original notation there are variable names that would lead to very long variable names in the generated code. These may be too long to be properly interpreted by some modeling environments. Another issue is given by the fact that expressions containing divisions are not supported by MOSAICmodeling as valid variable names.

The corresponding derivatives of the hard sphere term and further expressions that appear therein are provided by Eq. (5.69) to Eq. (5.73). Since these expressions were not found in the open literature, they were hand coded based on application of symbolic derivatives.

$$a_{xk,i,k}^{hc,red} = m_i \cdot a_k^{hs,red} + mm_k \cdot a_{xk,i,k}^{hs,red} - \left(\frac{1}{g_k^{hs}} \cdot g_{xk,i,k}^{hs} \cdot \sum_{i=1}^{NC} [x_{k,i} \cdot (m_i - 1)] + \ln(g_k^{hs}) \cdot (m_i - 1)\right) \quad (5.69)$$

$$a_{xk,i,k}^{hs,red} = \frac{2 \cdot (\eta_k - 2)}{(\eta_k - 1)^3} \cdot \eta_{xk,i,k} \quad (5.70)$$

$$\eta_{xk,i,k} = \frac{\pi \cdot \rho_k}{6} \cdot (m_i \cdot (d_k)^3 + mm_k \cdot 3 \cdot (d_k)^2 \cdot d_{xk,i,k}) \quad (5.71)$$

$$d_{xk,i,k} = \frac{1}{3} \cdot (d_k)^{-2} \cdot \left(\frac{(m_i \cdot (d_i)^3 \cdot mm_k) - (\sum_{i=1}^{NC} x_{i,k} \cdot m_i \cdot (d_i)^3) \cdot m_i}{(mm_k)^2}\right) \quad (5.72)$$

$$g_{xk,i,k}^{hs} = \left(\frac{2.5 - \eta_k}{(1 - \eta_k)^4}\right) \cdot \eta_{xk,i,k} \quad (5.73)$$

The corresponding derivatives of the dispersion contribution and the further expressions that appear therein are provided by Eq. (5.74) to Eq. (5.81). Here again, the meaning of the variables is self-explanatory and provided in the nomenclature of the corresponding model implementation in MOSAICmodeling.

$$a_{xk,i,k}^{disp,red} = -2 \cdot \pi \cdot \rho_k \cdot (I1_{xk,i,k} \cdot abbv1_k + I1_k \cdot abbv1_{xk,i,k}) - \pi \cdot \rho_k \cdot mm_k \cdot C1_k \cdot (I2_{xk,i,k} \cdot abbv2_k + I2_k \cdot abbv2_{xk,i,k}) - \pi \cdot \rho_k \cdot (m_i \cdot C1_k + mm_k \cdot C1_{xk,i,k}) \cdot I2_k \cdot abbv2_k \quad (5.74)$$

$$I1_{xk,i,k} = \sum_{p=1}^{Np} [a_{k,p-1}^{basic} \cdot (n_p - 1) \cdot \eta_{xk,i,k} \cdot (\eta_k)^{n_p - 2} + a_{xk,i,k,p-1}^{basic} \cdot (\eta_k)^{n_p - 1}] \quad (5.75)$$

$$a_{xk,i,p-1,k}^{basic} = \left(\frac{m_i}{(mm_k)^2} \cdot a_{p-1,ui=1}^{basic} + \frac{m_i}{(mm_k)^2} \cdot \left(3 - \frac{4}{mm_k}\right) \cdot a_{p-1,ui=2}^{basic}\right) \quad (5.76)$$

$$abbv1_{xk,i,k} = 2 \cdot m_i \cdot \sum_{j=1}^{NC} x_{i=j,k} \cdot m_{i=j} \cdot \left(\frac{\epsilon_{i,j}}{T}\right) \cdot (\sigma_{i,j})^3 \quad (5.77)$$

$$I2_{xk,i,k} = \sum_{p=1}^{Np} [b^{basic}_{k,p-1} \cdot (n_p - 1) \cdot \eta_{xk,i,k} \cdot (\eta_k)^{n_p-2} + b^{basic}_{xk,i,k,p-1} \cdot (\eta_k)^{n_p-1}] \tag{5.78}$$

$$b^{basic}_{xk,i,k,p-1} = (\frac{m_i}{(mm_k)^2} \cdot b^{basic}_{p-1,ui=1} + \frac{m_i}{(mm_k)^2} \cdot (3 - \frac{4}{mm_k}) \cdot b^{basic}_{p-1,ui=2}) \tag{5.79}$$

$$abbv2_{xk,i,k} = 2 \cdot m_i \cdot \sum_{j=1}^{NC} x_{i=j,k} \cdot m_{i=j} \cdot (\frac{\epsilon_{i,j}}{T})^2 \cdot (\sigma_{i,j})^3 \tag{5.80}$$

$$C1_{xk,i,k} = C2_k \cdot \eta_{xk,i,k} - (C1_k)^2 \cdot (m_i \cdot \frac{8 \cdot \eta_k - 2 \cdot (\eta_k)^2}{(1 - (\eta_k))^4} - m_i \cdot \frac{20 \cdot \eta_k - 27 \cdot (\eta_k)^2 + 12 \cdot (\eta_k)^3 - 2 \cdot (\eta_k)^4}{[(1 - \eta_k) \cdot (2 - (\eta_k))]^2}) \tag{5.81}$$

Analytic expressions with independent variables $(T, V, \mathbf{n})$ In the following, analytic expressions are presented which provide the reduced residual Helmholtz free energy according to Michelsen $F^{res,red} = A^r/(R \cdot T) = n \cdot \tilde{a}^{res}$ as function of the independent variables $(T, V, \mathbf{n})$. Though this set of independent variables is considered in mathematical expressions of the s-PC-SAFT EoS model provided in [106], it should be remarked that the expressions considered in this work do not correspond with the expressions from [106], since the latter were found to be inconsistent with their engineering units. For the sake of simplicity only the expressions are listed which differ from the ones presented above for the set of independent variables $T, \rho, \mathbf{x}$.

As for common association models, the reduced residual Helmholtz free energy consists of different contributions (Eq. (5.82)).

$$F^{res,red}_k = F^{hc,red}_k + F^{disp,red}_k \tag{5.82}$$

The expressions for the hard chain contribution are given in Eq. (5.83), where n_k is the total number of moles in phase k ($n_k = \sum_{i=1} n_{,i,k}$), $n_{i,k}$ is the number of moles of component i in phase k ($n_{i,k} = n_k \cdot x_{i,k}$) and mm_k the mean segment number in the mixture (Eq. (5.86)).

$$F^{hc,red}_k = mm_k \cdot n_k \cdot a^{hs,red}_k - \sum_{i=1}^{NC} n_{i,k} \cdot (m_i - 1) \cdot \ln(g^{hs}_k) \tag{5.83}$$

Most variables that appear in Eq. (5.83) can be taken as they originally appear in the model formulation for the set of independent variables $(T, \rho, \mathbf{x})$. However, it should be

noted that the packing fraction η_k that appears in many expressions is now defined by Eq. (5.84), where N_{av} is the Avogadro number, and V_k the volume in [m^3].

$$\eta_k = \frac{\pi \cdot (d_k)^3}{6} \cdot (N_{av} \cdot (10)^{-30}) \cdot (\frac{n_k}{V_k}) \cdot mm_k, \tag{5.84}$$

$$d_k = (\frac{\sum_{i=1}^{NC} n_{k,i} \cdot m_i \cdot (d_i)^3}{\sum_{i=1}^{NC} n_{k,i} \cdot m_i})^{1/3} \tag{5.85}$$

$$mm_k = \frac{1}{n_k} \cdot \sum_{i=1}^{NC} n_{i,k} \cdot m_i \tag{5.86}$$

The modified expressions for the dispersion contribution are given in Eq. (5.87) to Eq. (5.89). All other variables such as $I1_k$, $I2_k$, or $C1_k$ expressions can be considered as defined above.

$$F_k^{disp,red} = -2 \cdot \pi \cdot (\frac{N_{av} \cdot (10)^{-30}}{V_k}) \cdot I1_k \cdot abbv1_k - \pi \cdot (\frac{N_{av} \cdot (10)^{-30}}{V_k}) \cdot mm_k \cdot C1_k \cdot I2_k \cdot abbv2_k \tag{5.87}$$

$$abbv1_k = \sum_{i=1}^{NC} \sum_{j=1}^{NC} n_{i,k} \cdot n_{i=j,k} \cdot m_i \cdot m_{i=j} \cdot (\frac{\epsilon_{i,j}}{T}) \cdot (\sigma_{i,j})^3 \tag{5.88}$$

$$abbv2_k = \sum_{i=1}^{NC} \sum_{j=1}^{NC} n_{i,k} \cdot n_{i=j,k} \cdot m_i \cdot m_{i=j} \cdot (\frac{\epsilon_{i,j}}{T})^2 \cdot (\sigma_{i,j})^3 \tag{5.89}$$

Evaluation of derivatives with respect to V and $\mathbf{n}$ for model originally available for ρ and $\mathbf{x}$
In order to convert the model to work with the independent variables $n_{i,k}$ and V_k, when originally formulated with $x_{i,k}$ and ρ_k, additional equations are required. Eq. (5.90) is used to obtain the total number of moles n_k. This later is required to obtain the molar fractions $x_{i,k}$ with Eq. (5.91) and the total number density with Eq. (5.92).

$$n_k = \sum_{i=1}^{NC} n_{i,k} \tag{5.90}$$

$$x_{i,k} = \frac{n_{i,k}}{n_k} \tag{5.91}$$

$$\rho_k = \frac{n_k}{V_k} \cdot 6.022 \cdot (10)^{-7} \tag{5.92}$$

Finally, in order to evaluate derivatives with respect to V_k and $n_{i,k}$, Eq. (5.93) is used to convert the reduced residual dimensionless Helmholtz free energy into the form proposed by Michelsen.

$$F_k^{res,red} = n_k \cdot a_k^{res,red} \tag{5.93}$$

Evaluation of derivatives with respect to ρ and $\mathbf{x}$ for model originally available for V and $\mathbf{n}$ In order to convert the model to work with the independent variables $x_{i,k}$ and ρ_k, when originally formulated with $n_{i,k}$ and V_k, additional equations are required. For a given total number of moles n_k, the mole numbers of each component result from Eq. (5.94) and the extensive volume (in m^3) from Eq. (5.95).

$$n_{i,k} = n_k \cdot x_{i,k} \tag{5.94}$$

$$V_k = \frac{n_k}{\rho_k} \cdot (N_{av} \cdot (10)^{-30}) \tag{5.95}$$

Finally, in order to evaluate derivatives with respect to ρ_k and $x_{i,k}$, Eq. (5.96) is used to convert the reduced expressions as used by Michelsen into the corresponding dimensionless reduced variables.

$$a_k^{red,res} = \frac{F_k^{red,res}}{n_k} \tag{5.96}$$

Evaluation of computational effort for s-PC-SAFT Table 5.4 gives an overview of the different considered evaluations. A detailed explanation on the meaning of the columns is given previously in the corresponding description of the SRK EoS model.

Table 5.4: Considered cases for the computational evaluation of the influence of the set of independent variables using the s-PC-SAFT EoS model

Case	Helmholtz Free Energy	Fugacity Coefficient	Derivative Method
1	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,\rho,\mathbf{x})$	HC/SD
2	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,\rho,\mathbf{x})$	AD
3	$\tilde{a}^{red,res}(T,\rho,\mathbf{x})$	$\varphi_i(T,V,\mathbf{n})$	AD
4	$F^{red,res}(T,V,\mathbf{n})$	$\varphi_i(T,V,\mathbf{n})$	AD
5	$F^{red,res}(T,V,\mathbf{n})$	$\varphi_i(T,\rho,\mathbf{x})$	AD

As for the two EoS models discussed previously, SRK and SRK-MHV2, in order to quantify the computational effort required for the different mathematical formulations, the CPU

time required for the evaluation of fugacity coefficients is evaluated. It is obtained from the ten thousandfold evaluation for a mixture of 50 components. An easily verifiable system was selected. The mixture of 50 components is thought to consist of 25 components with the properties of methane and further 25 components with the properties of ethane. Since the s-PC-SAFT EoS model does not exhibit the so-called Michelsen-Kistenmacher syndrome [139], the fugacity coefficients have the same values as a binary mixture of comparable component distribution.

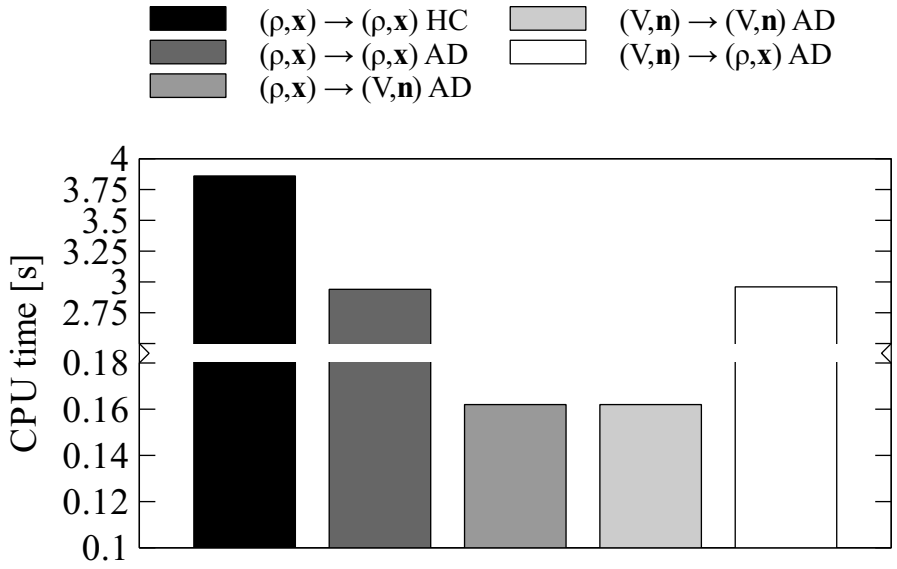

Figure 5.5: CPU times for 10000 evaluations of fugacity coefficients of a mixture consisting of 50 components using the s-PC-SAFT EoS model with different sets of independent variables for $\tilde{a}^{res}$ and φ_i

The results are given in Figure 5.5. They clearly show an influence of the considered set of independent variables on the computational effort. It is not only shown that derivatives obtained with AD lead to lower CPU times as hand coded symbolic derivatives, but also that the evaluation of fugacity coefficients according to Eq. (2.28) (with $(T, V, \mathbf{n})$ as independent variables) has advantages against the set $(T, \rho, \mathbf{x})$.

Particularly interesting is the fact that the introduction of additional expressions to change the set of independent variables has a major influence in this case. While the results indicate that the use of AD for derivative evaluation leads to a comparable effort as the evaluation of published analytic expressions (see first and second bar of Fig. 5.5), it becomes clear that the introduction of additional equations to change the set of dependent variables has a stronger influence. As the third bar of Fig. 5.5 indicates, it is of great advantage to build derivatives with $(T, V, \mathbf{n})$ as a set of variables. On the other hand,

the fifth bar shows the opposite for $(T, \rho, \mathbf{x})$. A more detailed analysis not shown here indicates that the differences in the aforementioned cases are in fact a consequence of the effort related to the derivative evaluations. Unlike the case with the SRK-MHV2 model, the differences in computational effort that appear through the evaluation of Eq. (2.28) or Eq. (2.27), are not decisive in this case.

Summarizing discussion on selection of the set of independent variables

On the basis of the evaluations previously discussed, it becomes clear that the selection of the set of independent variables indeed can have a significant influence on the resulting computational effort when working with EoS models. This finding can be seen as a confirmation to the fact propagated by the research groups, which support a particular set of independent variables. In spite of this, a comparison of the results also makes clear that general statements valid for all models and the different sets of independent variables are difficult to obtain. While the results regarding the SRK EoS model show the superiority of hand coded derivative information over derivatives obtained with algorithmic differentiation, the results concerning the SRK-MHV2 and s-PC-SAFT models show the opposite.[13] This first fact underlines the quality of the analytic expressions available for cubic EoS in comparison to analytic expressions provided for other models which are provided as hand coded derivatives. The factor by which the CPU times is augmented or reduced when using AD instead of hand coded derivatives is different for each considered model.

A very interesting finding that concern all models is the fact that all cases in which fugacity coefficients are evaluated with (2.28), using derivatives with respect to the mole numbers of component, lead to lower CPU times than the corresponding cases in which fugacity coefficients are evaluated with Eq. (2.27), using derivatives with respect to the molar fractions of the components. This reduction in the CPU times is independent of the fact, whether the core of the model considers $(T, \rho, \mathbf{x})$ or $(T, V, \mathbf{n})$ as independent variables. Particularly interesting is specially the fact, that no variable transformation involving complex algebraic manipulations seems to be necessary to profit from the advantages of using $(T, V, \mathbf{n})$ instead of $(T, \rho, \mathbf{x})$. As the results indicate, it is enough to introduce simple additional equations highlighting the relations between the different sets of independent variables (see for example Eq. (5.25) to Eq. (5.28)). In the case of the s-PC-SAFT EoS model, a comparable performance is obtained when introducing the dependency of $(T, V, \mathbf{n})$ through adding further relations (Eq. (5.25) to Eq. (5.28)) as in the case in which this dependence is added through algebraic manipulations (See third bar against fourth bar of Figure 5.5).[14]

[13] This last information is also valid for the hs-PCP-SAFT EoS model as shown in Figure 5.2.

[14] The introduction of algebraic manipulations refers to the process of *manually* eliminating the set of

Regarding the discussion related to the recommendable set of independent variables, this study clearly shows, that using $(T, V, \mathbf{n})$ leads in most cases to lower CPU times. However, as the results of SRK EoS show, in case of highly optimized model formulations differences may not be significant. Perhaps, a more interesting result of the shown analysis is the indirect finding that the combination of documentation-based implementations of thermodynamic models in conjunction with structural analysis and reverse mode of algorithmic differentiation can easily lead to model formulations that can profit from advantages of using a particular set of independent variables. As discussed, it is only necessary to expand the available model formulation by additional relations defining conversions between different sets of independent variables (see for example Eq. (5.25) to Eq. (5.28) or Eq. (5.22) to Eq. (5.24)) in order to obtain the required derivative information. There is no need to modify the core of the considered model implementation.

Finally, it should be noted that the advantages of a given set of independent variables may not only be given by related CPU times, but by the fact that particular algorithms or software may be available for a particular set of independent variables. It should also be taken into account that the results discussed above are strongly related to the considered formulations of the EoS models and to the tools used. It is a well-known fact that the quality of derivatives obtained by algorithmic differentiation is directly related to the quality of the implementation of the function whose derivatives are built.

5.3 Quantification of Influence of Derivative Evaluation Methods and Their Implementation: Hydroformylation Case Studies

The focus of the evaluations presented in Sections 5.2.1 and 5.2.2 has highlighted the computational effort when working with EoS models of the reduced residual Helmholtz free energy. In Section 5.2.1 the focus was laid on the gradient evaluation with various methods for derivative evaluation including hand coded derivatives, complex step approximation, and forward and reverse mode of AD. In Section 5.2.2 the influence of choosing a particular set of independent variables for the evaluation of derived thermodynamic properties is quantified for different EoS models of increasing complexity.

While the aforementioned results are concerned with derivative evaluation techniques and their efficiency, in this section some results are presented that emphasize the multi-tool aspect which is made possible by considering documentation-based models as the basis of the implementations. Not only different formulations and tools are considered for the evaluation of the models and their derivatives, but the code generations are chosen in such a

independent variables $(T, \rho, \mathbf{x})$ by introducing the relations Eq. (5.25) to Eq. (5.28).

way that all paths shown in Figure 4.1 are supported. Besides considering code generation that supports sequential evaluations and exploit such for derivative evaluation with AD techniques, implementations are taken into account which rely on a fully equation-oriented formulation.

A further feature of the following examples is given by the fact that the model implementations are used for real applications using the heterosegmented PCP-SAFT EoS model, namely for phase equilibrium calculations considering compounds that appear in the hydroformylation process introduced in Section 5.1. Table 5.5 gives an overview of the considered equilibrium calculations. The second column lists the compounds for which the equilibrium calculations are made, the third column the sought quantities in each case and the fourth column the variable range of independent variables that are changed to obtain the corresponding phase diagrams summarized in Figure 5.6.

Table 5.5: Considered phase equilibrium calculations of hydroformylation process

Case	Compounds	Sought quantity	Variable range
1	n-dodecanal	p^{LV}, ρ_L^{LV}, ρ_V^{LV}	$T = 450 \cdots 520$ K
2	1-dodecene/DMF	$\mathbf{x}^{org}$,$\mathbf{x}^{pol}$	$T = 288.15 \ldots 326$ K
3	1-dodecene/DMF/ decane	$\mathbf{x}^{org}$,$\mathbf{x}^{pol}$	$x = [0.08, 0.52, 0.4]$ $\ldots$ [0.3, 0.65, 0.05]
4	1-dodecene/n-dodecanal/ DMF/decane, H_2, CO	x_{syngas}^{org},x_{syngas}^{pol}	$p = 0.1 \ldots 12$ MPa T = 363.9 K

While the realization of equilibrium calculations as shown in Figure 5.6 indicates the feasibility of considering documentation-based model implementations for real applications with complex thermodynamic models, additional valuable information can be obtained by analyzing the computational effort related to the evaluation of different model formulations that can be generated automatically. In the following, the computational effort related to the generation of phase diagrams is evaluated. With the exception of the equilibrium calculations for pure n-dodecanal, in which the equilibrium is calculated for fixed values of temperature, in all cases the equilibrium compositions are obtained from the solution of the pT-flash equation system as introduced in Section 2.4.2. The generation of phase diagrams not only has a high practical relevance but is a rather simple application which requires multiple solutions and hence multiple evaluation of the considered equation systems. The considered evaluation approach is based on taking a valid solution point as initial guess for a following evaluation with slightly modified independent variables. This independent variable is given by the third column of Table 5.5. Note, that this approach can also be seen as the application of a homotopy method to obtain a difficult solution at the end of the considered interval. Further details on the considered variations are described next. All given CPU times refer to the same hardware as described in the introduction of Chapter 5.

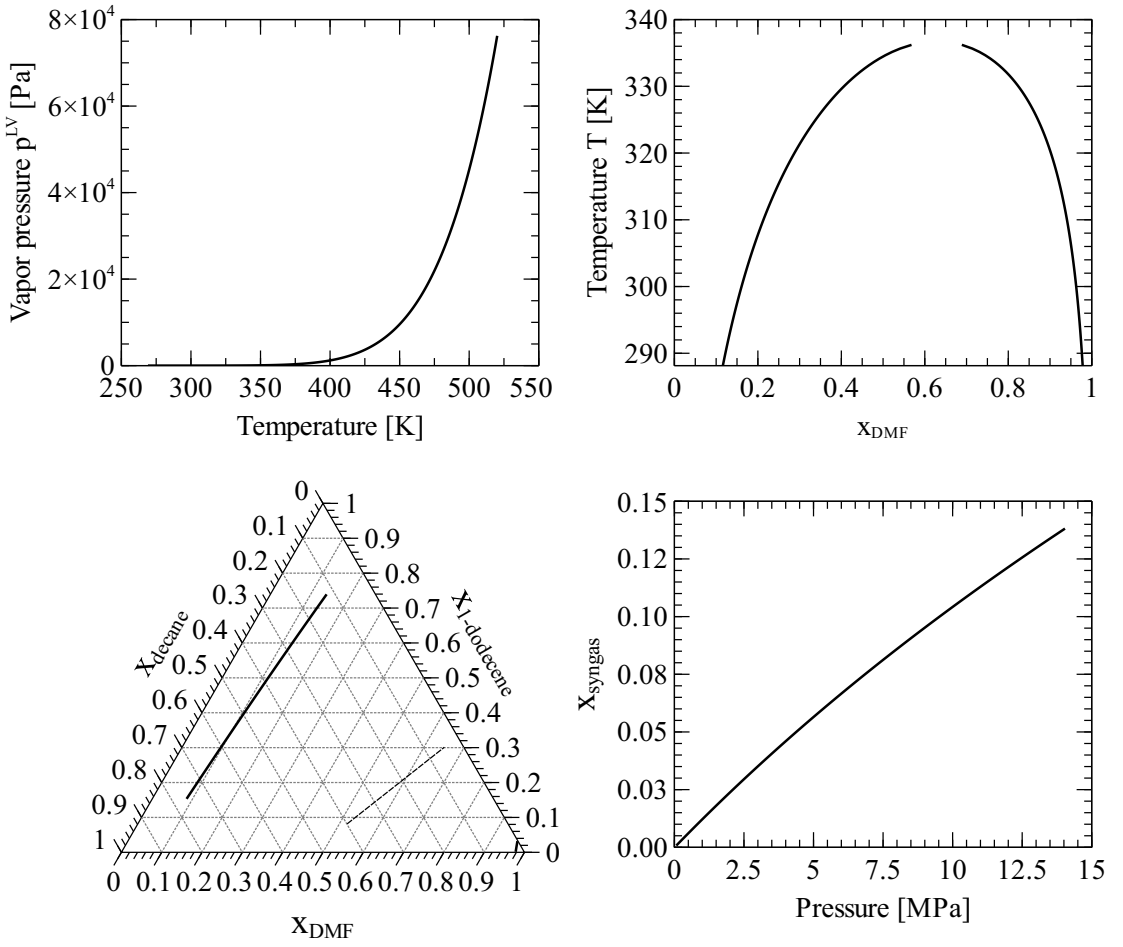

Figure 5.6: Phase diagrams based on calculations described in Table 5.5. Upper left: vapor pressure of n-dodecanal. Upper right: LLE of 1-dodecene/DMF mixture at 1 bar. Lower left: LLE of 1-dodecene/DMF/decane mixture at 298.15 K and 1 bar. Lower right: GLE of syngas $x_{syngas} = x_{CO} + x_{H2}$ in 1-dodecene/n-dodecanal/DMF/decane

Considered tools for evaluation of flash calculations

To emphasize on the multi-tool aspect gained by the use of documentation-based models, in this study different tools are considered, each of which can be seen as an exponent of a particular type of modeling approach. As a representative of general-purpose programming languages C++ code coupled with the numerical library BzzMath [30] is considered. MATLAB, which is used in conjunction with the package CasADi, represents a general-purpose numerical programming environment. Both aforementioned languages support a flexible model implementation, allowing sequential, equation-oriented, or hybrid approaches (see Section 2.1.1).

MATLAB and CasADi The code is generated for and evaluated by MATLAB release 2013b (The MathWorks, Inc.). CasADi version 3.0 is used not only for the evaluation of derivatives but also for the evaluation of all types of analytic expressions. In all cases, expressions previously analyzed by CasADi are called as CasADi function objects. This leads to a faster code execution compared to the default execution. The standard Newton solver provided with CasADi is used with default settings.

BzzMath and ADOL-C The generated C++ code is compiled and evaluated in the integrated development environment Microsoft Visual Studio 2010. The required derivative information in the case evaluating algorithmic differentiation is generated using the ADOL-C library version 2.5.2 [220] obtained as precompiled libraries from the website of Prof. Andrea Walther.[15] For the solution of the phase equilibrium problems the code of the system of nonlinear equations is additionally linked against the BzzMath library version 7.1 [30]. Though the BzzMath library gives the possibility of exploiting shared memory parallel computing through the OpenMP library, in the presented results this option was deactivated. This was necessary to get the used ADOL-C version to work properly with the BzzMath library.

Considered model implementations

While in all considered equilibrium calculations the pT-Flash model has to be fulfilled, hence, the equations (2.36), (2.37) and (2.39) are part of the equation system to be solved,[16] the difference between the evaluated model implementations is given by the way how the analytic expressions for the Helmholtz free energy and their derivatives are implemented or evaluated. Following implementation types are considered:

1. All expressions that define the Helmholtz free energy and its derivatives are available as equations. Since this implementation requires that analytic expressions for the derivatives are available (usually as hand coded derivatives), this only represents a reference case for fully equation-oriented implementations. As previously discussed in Section 4.1.1, in this case the pT-Flash equations are only a small part of the overall equation system.

2. This case is similar to the first one but with the difference that a structural analysis is used to generate sequential evaluation steps/procedures for the evaluation of the Helmholtz free energy and its derivatives. The actual number of equations

[15] https://math.uni-paderborn.de/ag/mathematik-und-ihre-anwendungen/software/

[16] Note that Eq. (2.38) is not listed here since the phase equilibria are solved for a given feed concentration.

and unknowns is given by the pT-flash equations. This case can also be seen as a reference case for programs/model implementations that support explicit sequential evaluations.

3. This model implementation corresponds to the one indicated by the left path in Figure 4.1. The equation system defining the expressions for the reduced residual Helmholtz energy is reordered on the basis of the structural analysis and the derivatives are obtained with reverse mode of algorithmic differentiation. Similarly to case 2 the actual number of equations is given by the pT-flash equations. The thermodynamic functions and derivatives can be seen as external calls.

4. The fourth and last evaluation type corresponds to the right path of Figure 4.1 which can be implemented in any arbitrary equation-oriented environment. The approaches chosen to evaluate the Jacobians ($\mathbf{J_{1,x}}$ and $\mathbf{J_1}, \mathbf{p}$ in Eq. (4.5)) and to solve their linear sensitivity equation system (Eq. (4.5)) represent at least two additional degrees of freedom in practical implementations. To obtain a first qualitative impression of the influence given by these approaches, different implementations are considered for the single programming languages. For the MATLAB implementation the derivatives and the solution of the linear equation system are directly made using corresponding CasADi commands. In case of the C++/BzzMath implementation the Jacobians are generated by symbolic derivatives as supported by MOSAICmodeling (equation by equation and without any additional effort to exploit common expressions). The resulting sensitivity linear equation system is solved using the `solve` method of the class `BzzFactorizedGauss`.

   Alternatively to directly executing calls for the solution of linear equations systems, as done with the proposed approaches, it would have been feasible to write out the linear equations explicitly. In fact, at first glance this latter approach seems to match better with the concepts of equation-oriented modeling and solving. However, explicitly writing out single linear equations does not seem necessary since the separate solution or even automatic recognition of linear subsystems has become a common feature of advanced environments that support the solution nonlinear equation systems.

Evaluated factors

The variety of model implementations gives a wide list of factors that are interesting to be evaluated. Although the generation of the phase diagrams is based on the simple strategy of taking a valid solution as initial guess for the following, slightly modified conditions,

the selection of the model implementation approach can have a major influence on the solution efficiency. Following factors are considered in the following evaluations:

1. **CPU-time** (t_{CPU}) Overall time required to solve the evaluations needed to generate the phase diagrams shown in Figure 5.6.

2. **Total number of solutions required** (N_{eval}) This denotes how many times the equation system needs to be solved in order to obtain the corresponding phase equilibrium diagram. Note that the solution of the model equations is obtained with so-called equation solving techniques [197], that means using general (nonlinear) solvers on the pT-flash equation systems. The number of solutions is given by the number of elements considered to cover the range given by the 4^{th} column of Table. 5.6. As a hypothetical example, consider the vapor pressure calculation of n-dodecanal, which should cover a temperature range between 450 and 520 K. If it would turn out to be necessary to solve the equation system (see Appendix B.2) at least for each *integer valued* temperature, in order to obtain stable solutions, then N_{eval} would have value of 71. Considering equal distances between each different value of the independent design variables, as done in this study, this is fully equivalent to the finding that the distance between each of the considered values of the design variable is one.

   Following strategy is considered to find out a proper value of N_{eval}. Starting with a feasible solution at the beginning or the end of the range, a very small change of the modifiable design variable (4^{th} column of Table. 5.6) is made that leads to convergence. Then the factor by which this change is made, is sequentially multiplied by a factor of two until convergence problems appear. The highest acceptable step size is of interest, since it leads to a smaller value of N_{eval}. However, it should be noted that there is a lower bound to N_{eval}. Accordingly, at least ten evaluations should be done. The reason behind this decision is based on the necessity to keep a high enough number of evaluations so as to obtain a significant time measurement.

3. **Time per solution step** (t_{sol}^{sin}) Though related with both measures listed above, $t_{sol}^{sin} = t_{CPU}/N_{eval}$, the average time for the solution of a single equilibrium calculation provides additionally a coarse measure of the influence of the problem size on the solution efficiency.

Results and discussion

On the basis of the modeling tools and model formulations discussed above, the overall results are given in Table 5.6. Perhaps as numerous as the amount of single results listed

Table 5.6: Overview of CPU times of phase equilibria calculations shown in Figure 5.6 with different derivative evaluation methods to build a part of the model

Case	Impl	Number of	MATLAB			BzzMath		
(Tab. 5.5)	Type	Equations [–]	t_{CPU} [s]	N_{eval} [–]	t_{sol}^{sin} [ms]	t_{CPU} [s]	N_{eval} [–]	t_{sol}^{sin} [ms]
1	1	366	0.24	10	24	0.75	10	75
	2	3	19.31	3500	5.52	1.28	3500	0.37
	3	3	19.03	3500	5.44	18.6	3500	5.31
	4	222	0.07	10	7.0	6.5	10	650
2	1	451	0.33	10	33	1.59	10	159
	2	7	55.18	7600	7.26	5.73	7600	0.75
	3	7	49.67	7600	6.54	70.5	7600	9.28
	4	214	0.31	40	7.75	28.97	40	724.25
3	1	1002	0.64	10	64	5.28	10	528
	2	9	0.18	20	9	0.07	10	7
	3	9	0.15	20	7.5	0.41	10	41
	4	490	0.1	10	10	52.1	10	5210
4	1	6645	4.39	20	219.5	2477.22	20	123 860.95
	2	15	0.59	40	14.75	0.37	40	9.25
	3	15	0.37	40	9.25	*	*	*
	4	2653	0.5	20	25	26 169.43	20	1 308 471.45

* Due to the big size of the problem, this could not be compiled with the considered configuration

in it, is the number of different conclusions that can be drawn from it. In the following an attempt will be made to summarize some of the most relevant findings.

A first check against some expectable, rather obvious results is used to proof the correctness of the calculations. As an example, regardless of which model implementation or which tool is evaluated, all results clearly show that the computational effort required for the solution of a single well converging phase equilibrium calculation t_{sol}^{sin} grows with the number of components of the considered mixture, hence with the size of the equation system. A further expected result follows from the comparison of t_{sol}^{sin} of the model implementation 2. Without considering external methods and tools for derivative evaluation and additional significant efforts concerning the evaluation of the derivative evaluation of large equation-oriented implementations, the results indicate the faster execution of compiled code (C++/BzzMath implementation) in comparison with the MATLAB/CasADi script code.[17] Also it seems reasonable that the single solution times t_{sol}^{sin} of equation systems that contain only few actual equations and many explicit evaluations are shorter as those which fully consist of equations without explicit order of evaluation.

A first interesting result results by comparing the CPU-times of the implementation types 2

[17]However, it should be remarked that the execution of MATLAB code coupled with CasADi is faster than the execution of comparable plain MATLAB code.

and 3 for all considered cases. The goal of this comparison is to quantify the influence of using derivatives generated with algorithmic differentiation against hand coded derivatives. In this case, not only an influence of the use of algorithmic differentiation is shown but mostly an influence of the considered algorithmic differentiation tool. While the MATLAB/CasADi implementation 3 using reverse mode of algorithmic differentiation leads to a faster execution than the corresponding implementation 2 with analytic expressions, the BzzMath implementation with ADOL-C has a clearly worse performance than the corresponding analytic case. In general, differences in performance and deviations from the expected theoretical performance are common when evaluating different AD tools. In this case the AD tool based on source transformation, CasADi, clearly outperforms the very popular operator overloading tool ADOL-C. In general, a careful selection and detailed evaluation of the considered AD tool is mandatory to properly exploit the possibilities of these methods. In spite of this effort, it must be considered, that in the context of code generation such an analysis only needs to be done once for each combination of programming language and AD tool.

Regarding the equation-oriented implementation approaches, comparable results, as discussed above, can be obtained by comparing all results of the implementation types 1 and 4. Again, significant differences appear depending on the considered tool. The MATLAB/CasADi implementation type 4 shows a great potential with clearly better results than the fully equation-oriented implementation type 1 and single solution times t_{sol}^{sin} comparable with the implementation types with explicit evaluations. On the other hand, the results obtained with the C++/BzzMath evaluation show the opposite behavior, where the implementation type 4 causes significantly higher solution times than implementation type 1. However, it should be remarked that the differences in the results of the implementations 1 and 4 are not only a consequence of the used programming language but additionally result from the method chosen to build the Jacobians (AD vs symbolic derivatives without exploitation of common expressions) and from solution of the linear sensitivity equation system. While a detailed analysis of these factors would be highly interesting, that would clearly go beyond the scope of this work. In the following, however, additional results are presented which easily follow from analyzing Table 5.6.

First, an analysis is provided that was conceived to quantify the performance of different derivative evaluation methods at the solution stage. As previously mentioned, a comparison of the times per single solution t_{sol}^{sin} of the implementation type 2 for both, MATLAB and C++ shows the advantages of using high-level-programming languages, which is mainly given by a faster code execution in comparison with the use of script languages. However, when comparing the results obtained for the implementation type 1, the latter considerations appear not to be valid. Here a different aspect comes into play. When

solving larger equation systems with many variables, the evaluation of the Jacobian of the system, as required by most Newton based solution methods may become critical. This is particularly the case when no analytic Jacobian is provided explicitly but approximated by some finite differences approach. This is the reason why the solution of fully equation-oriented formulations with MATLAB/CasADi has lower CPU times than the one using the BzzMath library: while the nonlinear solver provided with CasADi automatically exploits structural information to build exact Jacobians under use of compression techniques, hence reducing the effort for the evaluation of the Jacobian (see Section 3.1), the considered solver class of the BzzMath library `BzzNonLinearSystemObject` uses by default finite difference approximations. Consider for example the matrices shown in Figure 5.7. On the left-hand side, the sparsity pattern of the full equation-oriented implementation, a 6645x6645 matrix, is shown, making clear that a naive approach would consider a 6645x6645 identity seed matrix, implying 6646 function evaluations to build the full Jacobian at each iteration. The right-hand side of Figure 5.7 shows the seed matrix that results from applying the CPR algorithm [39] to the aforementioned 6645x6645 matrix. The resulting 6645x315 matrix indicates that a full Jacobian can be built out of 315 directional derivatives instead of 6645. At this point it should be remarked that the reduction of CPU time through the reduction of directions required to approximate the full Jacobian can also be supported by the BzzMath library. In order to do that, it would be necessary to use the `BzzNonLinearSystemSparse` class, which also supports sparse unstructured matrices, in place of the standard `BzzNonLinearSystemObject`. Additionally, the sparsity information would need to be provided. Since in MOSAICmodeling equation objects keep an overview of equations available in it, the sparsity information could be easily provided as part of a documentation-based model.

All findings discussed up to this point follow from the evaluation of the times per single solution t_{sol}^{sin} provided in Table. 5.6. Further information can be obtained by additionally evaluating the overall CPU times for solving the equation system in the desired variable range (t_{CPU}) in combination with the number of required evaluations (N_{eval}). The overall CPU times for the generation of the phase equilibrium plots clearly show that there are cases for which using larger equation systems without explicit evaluations have advantages over the systems, where a large part of the variables result from explicit evaluations. This is particularly the case for the calculation case 1 regarding the pure compound VLE of *n*-dodecanal and the case 2, the LLE of the DMF/1-dodecene binary system (Table 5.5). In these cases, using the fully equation-oriented implementation type 1 and the implementation with automatically generated sensitivity linear system (type 4), it is possible to obtain stable solutions of the phase equilibrium calculations over wide ranges of the independent variables with a relatively low number of evaluations and, in most cases, with an overall lower CPU time. On the other hand, when using the implementation types 2

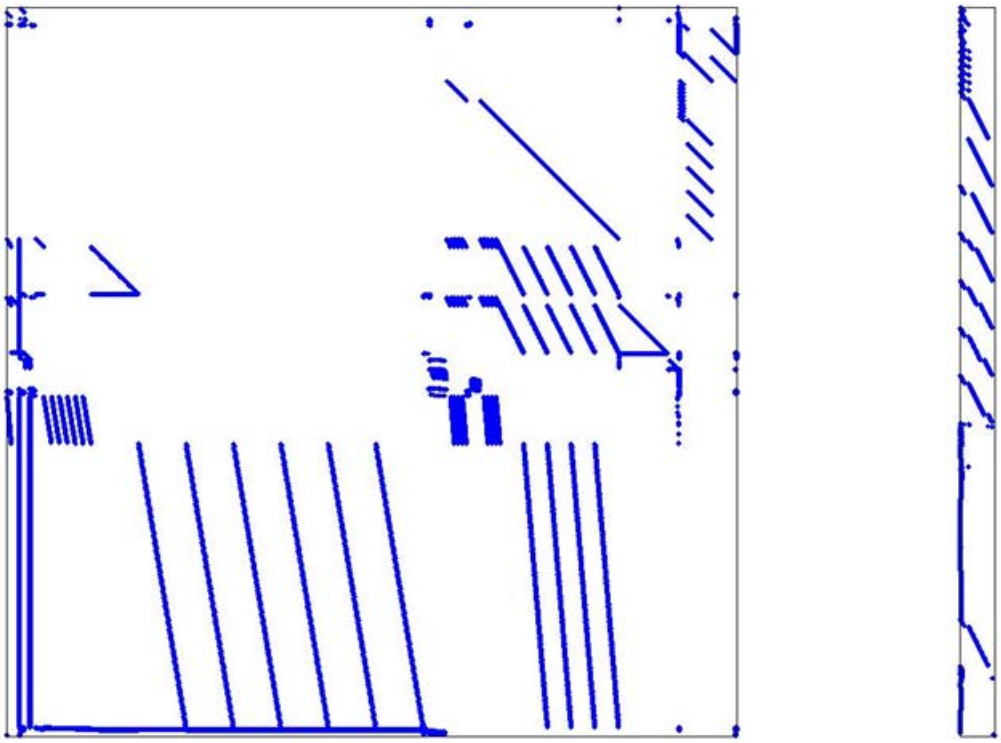

Figure 5.7: Incidence matrix of the Jacobian of the PCP-SAFT EoS model for mixture of 6 compounds (Implementation type 1 of Table 5.6). Left: 6645x6645 incidence matrix. Right: corresponding 6645x315 seed matrix obtained with CPR algorithm [39] applied to the matrix on the left-hand side

and 3, the cases with a significant lower number of equations, it is necessary to do very small changes in order to stay on a converging path. The phase equilibrium calculation cases 3 and 4 regarding the system with 3 and 6 compounds show a different behavior. For all implementation types, it is possible to cover the interesting ranges for phase equilibrium calculations with a relatively low amount of calculations. Accordingly, for these cases 3 and 4 the overall calculation times t_{CPU} of the implementation types 2 and 3 are significantly lower than the corresponding ones of implementation types 1 and 4.

Regarding the performance differences between fully equation-oriented implementations and those containing explicit evaluations,[18] probably, simple analytic and numerical measures such as equation and variable scaling or automatic equation reformulations could be automatically introduced to avoid such big performance differences. Such measures, however, were not studied in the current work. The result in Table. 5.6 rather reflect the potential that follows from directly using the same model formulations as available in the original model documentation. In spite of the potential given through the use of automatic model reformulations, in accordance with Hendriks et al. [84] it seems preferable

[18] These are particularly noticeable in the overall CPU times t_{CPU}.

if the authors of the EoS models directly provide a model formulation with convenient solution behavior.

All results clearly show the feasibility of using documentation-based model implementations of advanced EoS models as the basis for their application in real simulation tasks. It was shown that the selection of derivative evaluation methods and tools can have a major influence on the solution efficiency for both, fully equation-oriented model formulations and those including sequential evaluations. Among the obtained results, the ones based on the implicit function theorem with MATLAB/CasADi show the very high potential of equation-oriented formulations in combination with the implicit function theorem. This latter approach not only has short solution times but also a good numerical stability. In the context of equation-oriented implementations (types 1 and 4) the central importance of the Jacobians required by the solution algorithms was also shown.

In summary, all results clearly indicate that the proper use of automatic derivative evaluation methods not only leads to a reduced model implementation effort but that it can lead to highly efficient computational implementations that perform better than fully hand coded implementations.

5.4 Evaluation of Thermodynamic Models / Phase Equilibrium Calculations

Independent of the methods developed and applied in this work concerning documentation-based models, a significant contribution of the work done within the scope of this thesis consists of providing a rigorous plant wide simulation of the hydroformylation process described in Section 5.1. An important prerequisite to fulfill this goal consists of the evaluation and selection of proper models for the description of phase equilibria and further properties of mixtures. In this section concurrent thermodynamic models and corresponding parameterizations are evaluated in their suitability to correlate and estimate the phase equilibria that appear in the hydroformylation process.

After summarizing the available experimental data and modeling approaches in Section 5.4.1, in Section 5.4.2 an alternative parameterization approach for the hs-PCP-SAFT is proposed along with the parameterization using an alternative model, the SRK-MHV2 EoS model. Section 5.4.3 summarizes the results of the parameterization described in Section 5.4.2.

5.4.1 Available equilibrium data / current modeling approaches

On the basis of the considered compounds already listed in Table 5.1, Table 5.7 summarizes the availability of experimental data that can be used to obtain binary interaction parameters (BIPs) required by the thermodynamic models. For the sake of clarity, available LLE data, GLE data, and missing/unavailable data are considered or discussed separately.

Table 5.7: Inventory of available and missing experimental/modeling data regarding single binary pairs. ✓indicates pairs for which binary equilibrium data are directly available. ✔indicates binary interactions that can be obtained from ternary LLE data. ⊛ indicates binary pair information that can be approximated by parameters of the linear (n) components.× indicates pair for which there is no information at all

Component	i/j	1	2	3	4	5	6	7	8	9
1-dodecene	1		×	×	×	×	✓	✔	✓	✓
n-tridecanal*	2	×		×	×	×	✔	✔	✓	✓
iso-dodecene	3	×	×		×	×	⊛	⊛	⊛	⊛
iso-tridecanal	4	×	×	×		×	⊛	⊛	⊛	⊛
dodecane	5	×	×	×	×		✓	×	✓	✓
DMF	6	✓	✔	⊛	⊛	✓		✓	✓	✓
decane	7	✔	✔	⊛	⊛	×	✓		✓	✓
H_2	8	✓	✓	⊛	⊛	✓	✓	✓		×
CO	9	✓	✓	⊛	⊛	✓	✓	✓	×	

∗ Available n-dodecanal is considered as n-tridecanal

Liquid-liquid equilibrium data and current modeling approaches

Table 5.8: Considered LLE experimental data

Mixture type	Components	Ref.	Temperature [K]	Points given*
Binary	DMF/decane	[181]**	283.15 – 347.65	15
Binary	DMF/1-dodecene	[181]	288.15 – 333.15	6
Binary	DMF/dodecane	[2]***	288 – 318	4
Ternary	DMF/decane/ dodecene	[181]	298.15	6
			333.15	7
			343.15	10
Ternary	DMF/decane/ *n*-dodecanal	[181]	283.15	8
			288.15	6
			298.15	9
Ternary	DMF/decane/ *n*-tridecanal	[183]	298.15	12

* Points given per phase

** Considering only measurements made with the analytic method

*** Only binary measurements were considered

Table 5.8 provides an overview of the published LLE measurements considered in this contribution. With the exception of the data for the binary pair DMF/dodecane [2], all experimental data are taken from Schäfer and Sadowski [181, 182, 183]. In addition to their experimental work, Schäfer and Sadowski also showed the feasibility of satisfactory modeling most LLE measured by them with the PCP-SAFT EoS model [74, 76] using a relatively low number of BIPs (see also Table 5.16). Since the LLEs of ternary systems containing long chain aldehydes could not be estimated with a satisfactory accuracy using the standard (homonuclear or homosegmented) PCP-SAFT EoS model, Schaefer et al. [181] introduced the concept of the copolymer or heterosegmented PC-SAFT EoS model [75] to describe the polar long chain aldehydes. Figure 5.9 visualizes the additional degrees of freedom that result from introducing the long-chain aldehyde as a heterosegmented component.[19]

Though the introduction of the modeling approach based on hs-PCP-SAFT [181] represents a remarkable improvement in the correlation of the measurement data in comparison to the homosegmented approach, a detailed comparison of measurements and simulation, as shown by Figure 5.8, makes clear that the heterosegmented approach along with the proposed BIPs does not fulfill the expectations put on a high-fidelity predictive model as required by a rigorous plant-wide model.[20] The simulated mixing gaps are in general larger than those in the experiments and an acceptable fitting is only given at low aldehyde molar fractions. Furthermore, the information gained from the temperature dependence is purely qualitative.

The comparably large deviations between experiments and simulation for systems containing aldehydes suggest the need of either finding a better parameterization for the hs-PCP-SAFT approach or using an alternative thermodynamic model that gives a better fitting, while allowing application for multicomponent systems. In Section 5.4.2 both suggestions are considered. Alternative parameterization of the hs-PCP-SAFT model are considered and the SRK-MHV2 EoS model with NRTL as underlying g^E model is evaluated in its possibilities to fit the experimental data.

Gas-liquid equilibrium data and current modeling approach

Vogelpohl et al. provide in a series of papers [213, 214] a detailed experimental study to gas solubilities relevant to the TMS system considered in this work over wide ranges of

[19] Though not noticeable from Figure 5.9, it should be noted that the pure compound parameters of the resulting segments may be seen as additional degrees of freedom as well, which not only need to fulfill the LLE data but also available pure compound properties.

[20] The results generated by UNIFAC-Dortmund (UNIFAC-DO) are provided for the sake of completeness. Among the popular modifications of the UNIFAC model available in Aspen Plus, in preliminary studies [78] UNIFAC Dortmund showed the best agreement for the considered TMS.

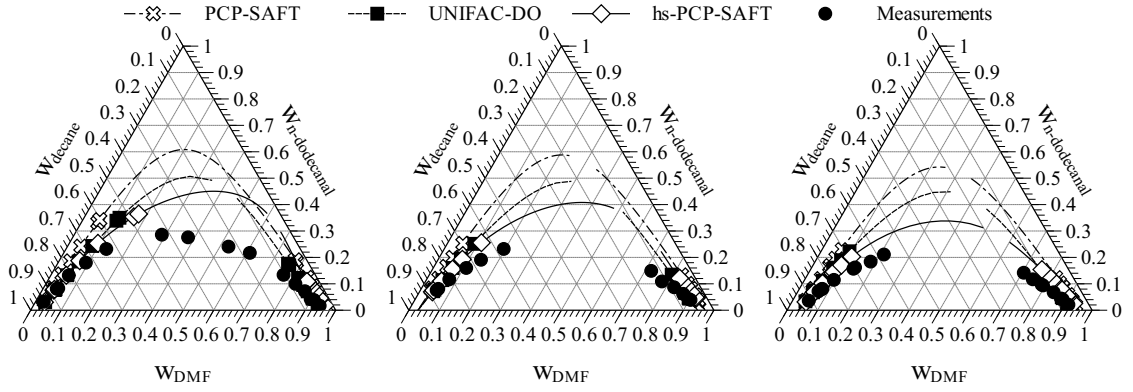

Figure 5.8: Liquid-liquid equilibrium experimental data and modeling results of *n*-dodecanal/DMF/decane system. Left: at 283.15 K. Center: at 288.15 K. Right: at 298.15 K. Symbols represent either experimental data from [181] (black circles) or LLE calculations with the respective models where the feed lies in the center of the experimental tie lines. Lines are binodal curves calculated using the respective model.

temperature and pressure. In addition to the experimental work, the gas solubilities were successfully modeled with the (homosegmented) PCP-SAFT EoS model. It was shown that using one single temperature independent BIP per gas/liquid pair, it is possible to estimate the temperature dependent gas solubility of multicomponent systems that consist of up to 4 liquid compounds and two gases with a satisfactory accuracy [214].

Leaving out very few measurements of CO in a *n*/*iso*-tridecanal mixture, *n*-dodecanal was treated as the product aldehyde. Similar to the LLE case, the solubilities of H_2, CO, and syngas in isomeric liquid components have not been studied. From the available BIPs that can be taken from the two last rows of Table 5.7 all but the ones for the H_2/dodecane pair are provided in [213, 214]. BIPs for H_2/dodecane can be obtained from the measurements published in [51].

Due to the high quality of the parameterization found by Vogelpohl et al. it does not appear to be necessary to look for an alternative parameter sets or for an alternative model. However, since not all LLE estimations obtained with hs-PCP-SAFT appear to be satisfactory, it is evaluated as well whether an alternative model, in this case the SRK-MHV2 model with NRTL as underlying g^E model can fit LLE and GLE satisfactorily.

Remarks on unavailable/missing data

Some remaining sources of uncertainty or challenges due to missing information should be remarked:

1. **Dodecanal/tridecanal issue** The overall goal of providing a thermodynamic con-

sistent, temperature dependent description of the phase behavior is significantly hindered by the fact that almost no measurements are available for systems containing n-tridecanal. Instead, almost all measurements were done with n-dodecanal. Since the very few available LLE and GLE measurements with n-tridecanal show a very similar phase behavior to systems with n-dodecanal, in this work n-tridecanal will be considered to have the same (temperature dependent) phase behavior as n-dodecanal. However, this does not mean that the BIPs obtained from systems that contain n-dodecanal should be used for systems with n-tridecanal. Results of phase equilibrium calculations with EoS models do namely not only depend on the values of BIPs but also on the values of pure compound parameters. Accordingly, different values of pure compound parameters should normally lead to different results of the parameter estimation, and hence to different BIPs.

2. **Iso-compounds** The iso-compounds listed in Table. 5.1 represent lumped compounds. This is already a major simplification of the true conditions in the considered process, since experiments show that that isomerization reactions lead to a broad variety of isomers of alkenes and aldehydes[123]. The possibility of rightly quantifying the influence of such compounds in the process is additionally reduced by the fact that no experimental phase equilibria data relevant to the considered hydroformylation are available at all. Regarding the evaluation of the concurring SRK-MHV2 and hs-PCP-SAFT EoS models, this unavailability does not represent any drawback, since only available information can be used for the model evaluation or comparison. The issue of unavailable information on iso-compounds may however have a higher importance for a rigorous plant-wide simulation, for which BIPs of the iso-compounds should be provided as well. Several approaches appear to be thinkable in this regard:

   - Set for all BIPs regarding interaction with an iso-compound the same values as for interactions with corresponding linear n-compounds.
   - Set for all BIPs regarding interaction with an iso-compound values obtained from a predictive approach such as UNIFAC. This evaluation is mainly thought to check how sensitive the model reacts with respect to small changes in BIPs. It is in particular interesting for comparison with the cases where BIPs are taken from the corresponding linear components.
   - Set the value of zero for these BIPs. This is again particularly interesting to compare this extreme case against the results of other approaches.

   While the results of such approaches have a high uncertainty, the comparison of all cases is interesting to quantify the influence of these BIPs in conjunction with the

amount of iso-compounds. In other words, it can answer the question whether the amounts of iso-compounds are high enough to have a notable influence on the overall calculations.

3. **Missing binary information** Though for most liquid compounds that appear in larger amounts (1-dodecene, aldehyde, DMF and decane) and for most gas/liquid pairs binary information is available, besides the isomeric compounds there are still binary pairs for which no binary information is available at all. Perhaps, the most important missing information binary information concerns the 1-dodecene/n-aldehyde pair. Here again similar alternatives to those discussed previously appear to be appropriate to quantify the missing binary information. Either neglecting the interaction or considering the interaction as predicted by the UNIFAC-Dortmund approach.

5.4.2 Alternative parameterization/model structure

Following the above explanations on the availability of experimental data and modeling approaches it becomes clear that the requirements for parameter estimation depend on the considered type of phase equilibrium available. Accordingly, the parameter requirements for LLE and GLE are treated separately. Before going into the details of the different types of phase equilibria some details on the model implementations are listed below.

Model implementation for simulation and parameter estimation

The basis for the parameter estimation and simulation studies described in this and the next sections is built by the possibility to evaluate phase equilibrium calculations for fixed inlet concentration, pressure and temperature.[21] This is done for the hs-PCP-SAFT and SRK-MHV2 EoS models. As shown in Section 2.4.2 for general multicomponent systems and in Appendix B.2 for pure components, the equations that define the pT-flash itself are rather simple and trivial to code or implement. The challenge lies in the correct implementation of the expressions for the residual, reduced Helmholtz free energy and its derivatives.

For the simulations and parameter estimation tasks presented in this section, pT-flash calculations including the corresponding thermodynamic models have been implemented in MATLAB release 2013b (The MathWorks, Inc.). For both models the expressions for the reduced, residual Helmholtz free energy have been coded as sequential evaluation procedures based on the results of a structural analysis as discussed in Section 4.3.2. The

[21] This is also called pT-flash calculation or isothermal, isobaric phase equilibrium problem.

hs-PCP-SAFT EoS model expressions for the Helmholtz free energy have been implemented according to Eq. (2.43) as a function of the total number density of molecules and molar fractions, while the SRK-EoS model has been implemented according to Eq. (5.17) as a function of the molar volume and the molar fractions. The partial derivatives required to obtain the compressibility factors and fugacity coefficients as indicated by Eq. (2.19) and (2.27) are obtained by the automatic differentiation package ADiMat[20].

All phase equilibrium calculations are initialized with the direct substitution method [144] and switched to a Newton based method after few iterations. Newton based solutions are carried out with the MATLAB's standard function for the solution of general nonlinear equation systems "fsolve", using the trust region dogleg algorithm with default settings.

Though the pT-flash calculations were implemented and used for the hs-PCP-SAFT and the SRK-MHV2 EoS models, flash calculations and parameter estimation studies related to the SRK-MHV2 model were evaluated with the software Aspen Plus Version 8.2 as well. Additional details on the parameter estimation of a particular type of data or for a particular type of model are described in the next sections.

LLE parameter estimation studies

Following the explanations of Section 5.4.1 and as highlighted in Figure 5.8 the ternary system *n*-dodecanal-DMF-decane is the only mixture treated by Schaefer et al. [181] whose phase equilibrium is not properly described by the hs-PCP-SAFT approach. Hence the focus of the further parameter estimation using the hs-PCP-SAFT EoS model is limited to the description of this ternary system. On the other hand, the description of all LLEs listed in Table 5.8 with the SRK-MHV2 EoS model requires the estimation of all BIPs that can be obtained from the available experimental data. In accordance with the different requirements for the parameter estimations, both tasks are treated separately.

hs-PCP-SAFT The set of BIPs used for the description of the LLE of the *n*-aldehyde containing systems provided in [181, 182] results from an effort to simultaneously fit pure compound properties (vapor pressure and saturated liquid density) and ternary LLEs (*n*-aldehyde-DMF-decane) for a homologous series of aldehydes of different chain lengths. For example, in [181, 182] the aldehydes are supposed to consist of a nonpolar tail group with the properties of an alkane and a head group containing the carbonyl group. While this strategy reduces the number of possible parameters, in particular when considering *n*-dodecanal,[22] the results shown in Figure 5.8 make clear that the obtained fitting is

[22] This is due to the fact that in the heterosegmented approach the nonpolar tail segment of n-dodecanal is defined by the properties of decane, a component for which BIPs with other compounds are already available. See also Figure 5.9.

not as satisfactory as required for rigorous process simulation. By giving up assumptions related to the heterosegmented structure of the aldehyde it should be possible to find better parameterizations that allow for a better correlation of the available LLE data.

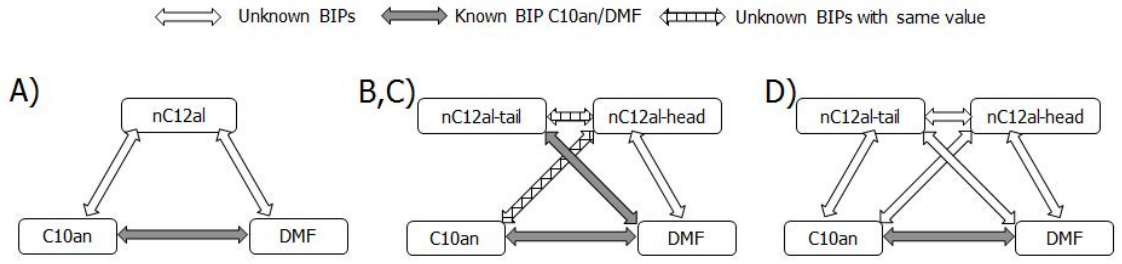

Figure 5.9: Schematic representation of BIPs obtained from LLE parameterization cases A, B, C and D using the hs-PCP-SAFT EoS model

Therefore, in this section additional parameterizations are proposed that let expect a better correlation of the LLE data. In addition to a pure homosegmented approach (parameterization case A) and the heterosegmented approach by Schaefer et al. [181] (parameterization case B) two additional parameterization cases are considered, which respectively consider different raw data and additional binary interactions. Figure 5.9 provides a schematic representation of the BIPs considered by the parameterizations. Note that the parameterization cases B and C share the same representation but the difference between them is given by the considered measurements. While case B considers data related of a homologous series of aldehydes, case C only considers data for n-dodecanal. Table 5.9 summarizes the parameters that are estimated in each case and the data sources.

Table 5.9: Considered LLE-parameterization cases evaluating PCP-SAFT and its derivatives

Case	Estimated parameters/Considered data sources
A	m_{nC12al}, σ_{nC12al} and ϵ_{nC12al}/k from P^{LV} and ρ_L^{Sat} data $k_{nC12al,C10an}$ and $k_{nC12al,DMF}$ from LLE data. Fixed μ_{nC12al}
B	$m_{nC12al-head}$, $\sigma_{nC12al-head}$, $\epsilon_{nC12al-head}/k$, $\mu_{nC12al-head}$, $k_{nC12al-head,nC12al-tail}$, $k_{nC12al-head,DMF}$ from pure compound data and LLE of homologous series of aldehydes [182]
C	Same as Case B but using only n-dodecanal data
D	$m_{nC12al-head}$, $\sigma_{nC12al-head}$, $\epsilon_{nC12al-head}/k$, $\mu_{nC12al-head}$ $m_{nC12al-tail}$, $\sigma_{nC12al-tail}$, $\epsilon_{nC12al-tail}/k$, $k_{C10an,nC12al-tail}$, $k_{C10an,nC12al-head}$, $k_{nC12al-tail,nC12al-head}$, $k_{nC12al-tail,DMF}$, $k_{nC12al-head,DMF}$ from P^{LV}, ρ_L^{Sat} and LLE data

As indicated by Figure 5.9 the possibility of defining additional BIPs with modified n-dodecanal segments is linked with the necessity of finding proper pure compound parameters for these new/modified segments. Table 5.9 lists the pure compound properties and the BIPs that are obtained in each parameterization case. Accordingly, the objective func-

tion (OF) that needs to be minimized within the considered parameter estimation problem consist of two different parts (Eq. (5.97)), one that denotes the pure compound properties and one that denotes the LLE. The terms of the resulting least square objective function are given by Eq. (5.98) and Eq. (5.99) respectively.

$$OF = OF^{pure} + OF^{LLE} \tag{5.97}$$

$$OF^{pure} = \sum_{n=1}^{NPM} \left(\left(\frac{p_n^{LV,exp} - p_n^{LV,cal}}{p_n^{LV,exp}} \right)^2 + \left(\frac{\rho_{L,n}^{Sat,exp} - \rho_{L,n}^{Sat,cal}}{\rho_{L,n}^{Sat,exp}} \right)^2 \right) \tag{5.98}$$

$$OF^{LLE} = \sum_{n=1}^{NLLE} \sum_{i=1}^{NC} \left(\left(\frac{x_{n,i}^{org,exp} - x_{n,i}^{org,cal}}{x_{n,i}^{org,exp}} \right)^2 + \left(\frac{x_{n,i}^{pol,exp} - x_{n,i}^{org,cal}}{x_{n,i}^{pol,exp}} \right)^2 \right) \tag{5.99}$$

The required LLE measurement data $x_{n,i}^{org,exp}$ and $x_{n,i}^{org,exp}$, are directly taken from [182], the pure component data for vapor pressure $p_n^{LV,exp}$ and saturated liquid density $\rho_{L,n}^{Sat,exp}$ of *n*-dodecanal were obtained as *evaluated results* using the NIST ThermoData Engine version 7.1 [56] in a temperature range between 320 and 570 K.[23] The calculated equilibrium compositions $x_{n,i}^{org,exp}$ and $x_{n,i}^{pol,exp}$ result from the solution of pT-flash calculations as described in Section 2.4.2, where the feed composition was arbitrary chosen to lie in the center of the experimental tie lines. The calculated vapor pressure and saturated liquid density result from the solution of the flash calculation for pure compounds as described in Appendix B.2.

The parameter estimation problem, the minimization of Eq. (5.97), is solved with MATLAB's "lsqnonlin" function using the trust region reflective algorithm. In all cases the parameter values published in [182] were used as initial guesses. Additionally, several further initial guesses were tried, which led to the same solutions. The results obtained by simulation studies using the obtained parameter sets were successfully checked against stability analysis. For this latter point, the formulation proposed in [224] was considered, but the solution was obtained using local optimization methods.

SRK-MHV2 with NRTL Based on theoretical reasons already presented in Section 2.4.4, the SRK-MHV2 EoS model in combination with the NRTL g^E model appears to be a promising candidate for the correlation of the given LLE and GLE data. Since in the literature there are no available BIPs for any of the interesting binary pairs, all available information regarding binary and ternary LLEs summarized in Table 5.8 should be used

[23] The NIST ThermoData Engine version 7.1 is available through Aspen Plus version 8.2.

to find as many potential significant BIPs as possible. As common for EoS/g^E models, the BIPs to be fitted correspond to the BIPs of the g^E model. The strategy for parameter estimation is as follows: first, BIPs are obtained that can be determined from binary measurements. The determined parameters are fixed for the parameter estimations based on ternary measurements, which are evaluated to obtain the missing BIPs. 1-dodecene-decane BIPs are obtained from the ternary system 1-dodecene-DMF-decane, while *n*-dodecanal-decane and *n*-dodecanal-DMF BIPs are obtained from LLE data of the ternary system *n*-dodecanal-DMF-decane.

Similar to the parameter estimation based on the LLE data of the ternary system *n*-dodecanal-DMF-decane with the hs-PCP-SAFT EoS model, for which the influence of the number of estimated parameters on the fitting quality is studied, a comparable study is made with the SRK-MHV2 EoS model. It should be evaluated how the number of BIPs considered to quantify the binary interaction between the pairs *n*-dodecanal/decane and *n*-dodecanal/DMF has an influence on the quality of the fitting. Considering the value of the BIPs for the pair DMF/decane as already known, according to Appendix B.1 there is a maximum of up to 20 BIPs that can be considered. Following the estimation procedure proposed in Appendix B.1.1 three different converging parameterization cases were found. The estimated parameters are summarized in Table 5.10.

Table 5.10: Considered parameterizations using SRK-MHV2 with NRTL for description of ternary system *n*-dodecanal (1), DMF (2), decane (3) system. BIPs for (1) and (2) were previously obtained from binary LLE data

Case	Estimated Parameters
A1	$b_{1,2}$, $b_{1,3}$, $b_{2,1}$, $b_{3,1}$
B1	$a_{1,2}$, $a_{1,3}$, $a_{3,1}$, $b_{1,2}$, $b_{1,3}$, $b_{2,1}$, $b_{3,1}$
C1	$a_{1,2}$, $b_{1,2}$, $b_{1,3}$, $b_{2,1}$, $b_{3,1}$, $c_{1,3}$, $f_{1,2}$, $f_{1,3}$, $f_{2,1}$, $f_{3,1}$

.

The parameter estimations regarding the SRK-MHV2 EoS model were done with Aspen Plus data regression system [11] considering the Maximum-Likelihood objective function and further default settings. The methodological approach to estimate each BIP is explained in Appendix B.1. In addition to parameter estimation with Aspen, parameter estimations with the MATLAB implementation have been done using Eq. (5.99) as objective function. Comparable results were obtained.

GLE parameter estimation studies

The requirements for GLE parameter estimation are different to those of the LLE cases. Since all done parameter estimation studies using the PCP-SAFT approach led to satisfactory results, there is no need to find any better parameterization, but it is only necessary to find BIPs of binary pairs that have not been considered in the open literature up to now. In case of the SRK-MHV2 EoS model with NRTL as underlying g^E model, BIPs need to be found for all gas-liquid binary pairs.

PCP-SAFT Since the parameterizations for all considered binary pairs show a good agreement between measurement and calculations[213, 214] only additional parameterizations for missing pairs are required. Hence, a parameter estimation for the missing pair H_2-dodecane is required. Since, both [213] and [214] provide contradictory BIP-values for the CO-DMF pairs, additionally own parameter estimation for CO-DMF are considered as well.

The implementation and solution of the required parameter estimation problem is analogous to the one for the LLE case described above for the hs-PCP-SAFT EoS model, but considering the objective function given by Eq. (5.100),

$$OF^{GLE} = \sum_{n=1}^{NGLE} \left(\frac{x_{GAS,n}^{L,exp} - x_{GAS,n}^{L,cal}}{x_{GAS,n}^{L,exp}} \right)^2 \tag{5.100}$$

where $n = 1 \ldots NGLE$ denotes the set of all available measured solubilities for a given gas/liquid pair for all available pressure and temperature levels. $x_{GAS}^{L,cal}$ denotes the calculated molar fraction of the gaseous compound denoted by GAS, in the liquid phase L. Accordingly, only one of the four molar fractions obtained from the underlying flash calculation is required by the parameter estimation procedure. The corresponding experimental values $x_{GAS}^{L,exp}$ for H_2 in dodecane are taken from [51]. For the pair DMF/CO experimental values are taken from [213].

SRK-MHV2 with NRTL For a wide pressure range of around 0.1 up to 12 MPa [24] preliminary studies described in Gutierrez-Sanchez [78] showed that a qualitatively right description of the GLE of all gas/liquid pairs involved in the considered process can be obtained by fitting the SRK-MHV2 EoS model against simulation results (*simulated measurements*) generated with the PCP-SAFT EoS model using parameters provided in [213, 214].

A more detailed quantitative evaluation reveals, however, that the accuracy of the simulation results using BIPs obtained over a wider pressure range are not satisfactory. More

[24] As considered by Vogelpohl et al. in their studies.

satisfactory, quantitatively correct results, can be obtained if smaller pressure ranges, for example between 0.1 and 3.5 MPa, are considered [78]. Since this smaller pressure range coincides with the interesting operation range of the process, the parameter estimation regarding GLE with the SRK-MHV2 model only concerns this pressure range.

In analogy to the LLE case, the GLE parameter estimation is carried out with Aspen plus data regression system[11] using the strategy described in Appendix B.1.1. The GLE measurements required for the parameter estimation are generated by evaluating pT-flash calculations with the PCP-SAFT EoS model in the pressure range from 0.1 to 3.5 MPa at the temperatures for which experimental data are available. The pure compound parameters and BIPs are considered as provided in [213, 214] or as estimated in Section 5.4.3.

5.4.3 Results of parameter estimation studies

As described in Section 5.4.2 the goals of each of the single considered parameter estimation studies depend on the type of phase equilibrium and the thermodynamic model. In the following, the results are presented according to these criteria. Comparative evaluations of the correlative capabilities of the models for each equilibrium type, LLE and GLE, are given as well.

Results of LLE parameter estimation studies using hs-PCP-SAFT

In accordance with the parameter estimation cases considered to obtain a better description of the liquid-liquid equilibrium of the ternary system 1-dodecene-DMF-*n*-dodecanal with the hs-PCP-SAFT EoS model, see Figure 5.9 and Table 5.9, results are available for both, pure compound parameters and binary interaction parameters.

Table 5.11 lists the pure compound parameters obtained from the different parameterization cases and provides the corresponding average relative deviation (ARD) and maximum relative deviation (max. RD).[25] While the results show that the homosegmented case A, the only case where pure compound parameters and BIPs are estimated separately, has the lowest deviations and hence provides the best fit, it can be seen, that the deviation obtained from all other cases lie in an acceptable range.

The binary interaction parameters obtained from the corresponding parameterization cases are given in Table 5.12. It should be remarked that some of the values of BIPs provided therein for parameterization case B do not coincide with the values published in [181, 183], but coincide with the values published in the Ph.D. thesis of Schäfer [182]. A detailed

[25] The used definitions of the ARD and max. RD can be found in the nomenclature.

Table 5.11: Pure compound parameters of n-dodecanal from different parameterization cases from Table 5.9 and their influence on pure component properties estimation

					Vapor pressure		Sat. liq. density	
Case see Table 5.9	m [–]	σ [Å]	ϵ/k [K]	μ [D]	ARD [%]	max. RD [%]	ARD [%]	max. RD [%]
A*	6.3108	3.5401	252.73	2.88	0.36	1.08	0.10	0.40
B								
–tail	4.6627	3.8384	243.87		4.72	5.19	8.70	9.51
–head	1.5599	3.0601	220.56	2.88				
C								
–tail	4.6627	3.8384	243.87		1.62	2.59	16.12	17.71
–head	1.6906	3.4410	175.78	3.42				
D								
–tail	4.5433	3.9165	243.78		4.95	12.99	4.44	6.55
–head	1.6906	2.9447	167.45	3.59				

∗ New fit due to major deviations when using parameters from [181], probably due to different sources of pure compound data

evaluation of the consequence of modeling the LLE with the corresponding BIPs is given in Figure 5.10. This shows the average relative deviation of the calculated molar fractions of each component in each phase on a percentage basis for single temperature levels.

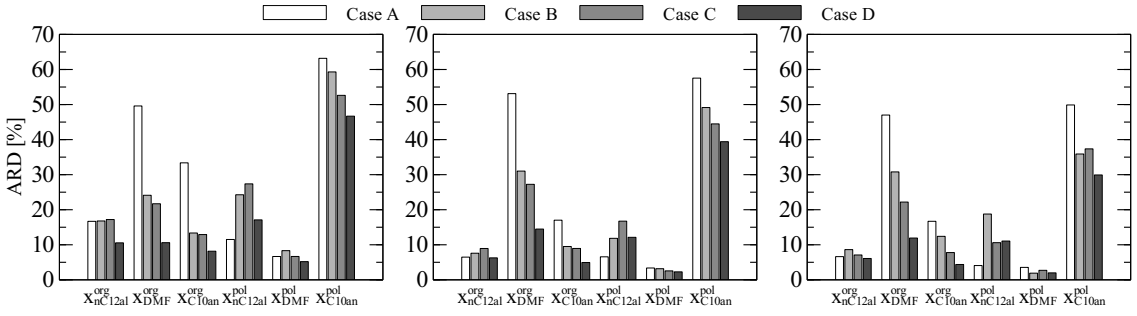

Figure 5.10: Average relative deviation of molar fractions of each component in each phase based on different parameterization cases from Table 5.9. Left: T = 283.15 K. Center: T = 288.15 K. Right: T = 298.15 K

From Figure 5.10 it can clearly be seen that the heterosegmented approach as introduced in [181] (case B) represents a significant improvement in comparison with the homosegmented approach (case A). Analogously the parameterization case C and in particular the parameterization case D represent an additional improvement in comparison with case B. The improvement in the fitting introduced through the additional parameterization cases is particularly noticeable for the molar fractions related to the organic phase. The molar fractions related to the polar phase still show relatively large deviations. Additionally, it can be recognized that at lower temperatures a worse fitting is obtained. This latter

Table 5.12: Binary interaction parameters from different parameterization cases listed in Table 5.9

Case see Table 5.9		Binary interaction parameters DMF	decane	*n*-dodecanal tail
A	DMF		$0.1159 - 0.000315 \cdot T$	
	n-dodecanal	0.01	0.015	
B	DMF		$0.1159 - 0.000315 \cdot T$	$0.1159 - 0.000315 \cdot T$
	n-dodecanal head	0.0075	-0.032	-0.031
C	DMF		$0.1159 - 0.000315 \cdot T$	$0.1159 - 0.000315 \cdot T$
	n-dodecanal head	$-0.561 + 0.0017 \cdot T$	-0.107	-0.107
D	DMF		$0.1159 - 0.000315 \cdot T$	-0.002
	n-dodecanal head	- 0.006	-0.075	0.044
	n-dodecanal tail	-0.002	-0.074	

issue, which is particularly clear for the temperature level of 283.15 K, should not be seen as a particular weakness of the model at low temperatures, but is due to the fact that at 283.15 K there is a relatively large number of measurements near to the critical point, which are particularly difficult to fit.

A different representation of the aforementioned results, that will be used to compare the correlative capabilities of the hs-PCP-SAFT EoS model against the SRK-MHV2 EoS model for further binary and ternary systems, is given by evaluating the average of the ARD of the molar fractions of all components and all phases at each single temperature according to Eq. (5.101). The resulting values are given in Table 5.13.

$$\mathrm{ARD}_T^{org,pol} = \frac{100}{2 \cdot NM \cdot NC} \cdot \sum_{n=1}^{NM} \sum_{i=1}^{NC} \left(\frac{\left| x_{n,i}^{org,exp} - x_{n,i}^{org,cal} \right|}{x_{n,i}^{org,exp}} + \frac{\left| x_{n,i}^{pol,exp} - x_{n,i}^{pol,cal} \right|}{x_{n,i}^{pol,exp}} \right) \quad (5.101)$$

For the sake of clearness, it should additionally be remarked that the results of the alternative parameterization of *n*-dodecanal and the alternative BIPs should not be seen as improved pure compound parameters and BIPs in comparison to previous approaches. The results rather indicate the correlative potential of the model structure that results from introducing *n*-dodecanal as a flexible heterosegmented component with a polar head segment. Although some of the obtained BIPs ($k_{i,\alpha,j,\beta}$) show relatively large values in comparison with BIP ranges common in other LLE modeling applications with PCP-SAFT, for fitting purposes the resulting values seem to be acceptable.

Table 5.13: ARD analysis of different PCP-SAFT parameterization at different temperatures (see Eq. (5.101))

Case	$ARD_T^{org,pol}$ [%]		
see Table 5.9	$T = 283.15\,K$	$T = 288.15\,K$	$T = 298.15\,K$
A	30.17	24.02	21.30
B	24.36	18.71	18.06
C	23.08	18.15	14.62
D	16.80	12.44	10.18

Results of LLE parameter estimation studies using SRK-MHV2 model

As mentioned in Section 5.4.2 regarding the SRK-MHV2 EoS model, it is necessary to determine the required NRTL BIPs of all binary pairs for which experimental LLE data are available (see Table 5.7). The following results will first focus on the evaluation of all binary and ternary systems that have been successfully modeled with the PCP-SAFT approach. Subsequently, and similar to the previous study using the hs-PCP-SAFT, it will be evaluated how the quality of the parameter estimation of the ternary system *n*-dodecanal-DMF-decane is influenced by the number of estimated parameters.

Following the parameter estimation approach described in Section 5.4.2 and the estimation procedure described in Appendix B.1.1, the NRTL BIPs listed in Table 5.14 are obtained.[26]

Table 5.14: Binary NRTL parameters regressed from data listed in Table 5.8

i	decane	dodecene	DMF	decane	decane	n-dodecanal
j	DMF	DMF	dodecane	dodecene	n-dodecanal	DMF
$a_{i,j}$	−5.964	8.659	−85.842	83.136	8.442	−8.292
$a_{j,i}$	20.282	0.421	15.699	83.136	−6.610	0
$b_{i,j}$	2243.971	0.000	12 813.500	−13 048.291	−1380.816	1824.345
$b_{j,i}$	0	702.031	0	−13 048.291	1171.226	681.649
$c_{i,j}$	0.2	0.2	0.2	0.3	0.3	0.2
$d_{i,j}$	0	0	0	0	0	0
$e_{i,j}$	0	0	0	0	0	0
$e_{j,i}$	−3.077	0	0	0	0	0
$f_{i,j}$	0	−0.026	0.156	−0.131	0	0
$f_{j,i}$	0	0	−0.049	−0.131	0	0
T^{low}	283.15	288.15	288	298.15	283.15	283.15
T^{up}	347.65	333.15	318	343.15	298.15	298.15

Table 5.15 lists the results of the parameter estimation obtained on the basis of ternary LLE data for the system *n*-dodecanal-DMF-decane and different parameter sets as listed

[26] The results for the pairs *n*-dodecanal/decane and *n*-dodecanal/DMF are discussed below (see Table 5.15).

in Table 5.10. More details on the quality of these results, in comparison with the ones of the PCP-SAFT approach, are given below in the comparative evaluation of both models.

Table 5.15: Binary interaction NRTL parameters obtained considering ternary LLE *n*-dodecanal/DMF/decane data resulting from parameterization cases A1, B1 and C1 (see Table 5.10)

	Case A1		Case B1		Case C1	
i	decane	n-dodecanal	decane	n-dodecanal	decane	n-dodecanal
j	n-dodecanal	DMF	n-dodecanal	DMF	n-dodecanal	DMF
$a_{i,j}$	0	0	8.442	−8.292	0	0
$a_{j,i}$	0	0	−6.610	0	0	−57.317
$b_{i,j}$	1081.269	736.526	−1380.816	1824.345	−1537.812	−2407.539
$b_{j,i}$	−744.742	−601.003	1171.226	681.649	389.332	9854.132
$c_{i,j}$	0.3	0.2	0.3	0.2	0.492	0.2
$d_{i,j}$	0	0	0	0	0	0
$e_{i,j}$	0	0	0	0	0	0
$e_{j,i}$	0	0	0	0	0	0
$f_{i,j}$	0	0	0	0	0.030	0.046
$f_{j,i}$	0	0	0	0	−0.011	0.070

Comparative evaluation of the parameter estimation results using hs-PCP-SAFT and the SRK-MHV2 EoS model

Considering the definition of the average relative deviation of the calculated molar fractions introduced by Eq. (5.101), Table 5.16 provides a comparison of the correlative capabilities of PCP-SAFT against the SRK-MHV2 EoS model with the estimated NRTL BIPs. Table 5.16 also provides the number of BIPs considered by each estimation and, for the sake of completeness, provides additionally the corresponding values obtained from the evaluation with the predictive UNIFAC Dortmund model.

The results of Table 5.16 show that the fits obtained with the SRK-MHV2 EoS model have smaller deviations than the corresponding ones using the PCP-SAFT EoS model. However, it should be considered that PCP-SAFT provides very good results in spite of a very low number of estimated BIPs. This is surely partially due to the fact that the goal in the original publication [181] was to show the feasibility of rightfully modeling the relevant LLE with few BIPs, and not to provide the best possible fitting. Additionally, in case of the ternary system 1-dodecene-DMF-decane, it should be considered that using PCP-SAFT a very good description of the real system is reached without the need to introduce BIPs for the pair decane/dodecene. Regarding the UNIFAC Dortmund model, it can be summarized that it provides relatively good results for the binary system over the considered temperature range and even good results for the ternary system at 298.15 K.

Table 5.16: Modeling results for DMF-decane, 1-dodecene-DMF and 1-dodecene-DMF-decane and number of required BIPs

Mixture	Temperature	$ARD_T^{org,pol}$		
	[K]	PCP-SAFT	SRK-MHV2	UNIFAC-DO
DMF/decane	283.15 – 347.65	9.15[a]	2.67[b]	10.55
1-dodecene/DMF	288.15 – 333.15	7.74[a]	2.28[b]	14.43
1-dodecene/DMF/	298.15	12.67[b]	4.70[c]	5.81
decane	333.15	2.89	5.45	23.07
	343.15	12.03	4.19	26.84

[a] Total number of required BIPs: 2
[b] Total number of required BIPs: 4
[c] Total number of required BIPs: 11

However, the predicted temperature dependence does not fulfill the requirements to rightfully describe a thermomorphic solvent system. All in all, it can be summarized that a good correlation of the appearing LLE can be reached by considering the PCP-SAFT pure compound and binary interaction parameters provided in [181] and the SRK-MHV2 NRTL-BIPs provided in Table 5.14.

Regarding the ternary system *n*-dodecanal-DMF-decane, Figure 5.11 provides a comparison of the correlative capabilities of the three parameterization cases designed for the SRK-MHV2 EoS model, see Table 5.10, against the best results obtained with hs-PCP-SAFT (Case D of Table 5.9). Additionally, and in order to summarize the influence of the number of BIPs on the correlative capabilities of both models, Figure 5.12 provides comparisons of calculated LLEs against experimental measurements. Particularly interesting in Figure 5.12 is the fact that it gives an impression on how good the temperature dependency of the liquid-liquid equilibrium is predicted.

The results clearly show that all considered parameterizations cases based on the SRK-MHV2 EoS model agree much better with the measurements than the results obtained with the modified hs-PCP-SAFT approaches. This is in particular the case for the molar fractions of the polar phase. On the other hand, both models show a comparable accuracy for the molar fractions of the organic phase. Considering the ARD values resulting from the parameterization cases, it appears that all results obtained with the SRK-MHV2 model are comparably satisfactory. However, by checking Figure 5.12 carefully, it becomes clear that only the cases B1 and C1 indicate a quantitatively relevant temperature dependence of the liquid-liquid equilibrium. Since no major difference appears between the cases B1 and C1, for practical reasons, the results obtained from the case B1 will be used for the

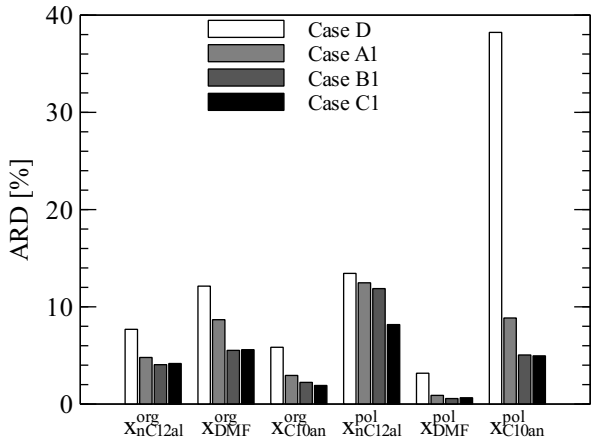

Figure 5.11: ARD analysis of molar fractions of each component in each phase weighted over all temperatures. Comparison of best hs-PCP-SAFT parameterization (Case D) against NRTL parameterizations from Table 5.10 [28]

process simulation studies.

Results of GLE parameter estimation studies using SRK-MHV2 EoS model and comparative evaluation against PCP-SAFT

As previously described in Section 5.4.2, only for the gas/liquid pair with missing BIPs, H_2/dodecane, or the pair with contradictory BIPs, CO/DMF, parameter estimations with the PCP-SAFT EoS model are required. On the other hand, for the aforementioned and all other gas/liquid pairs NRTL BIPs need to be found to properly describe the gas-liquid equilibria in the pressure range between 0.1 and 3.5 MPa and in the temperature range covered by the available experimental data of the respective publications.

Table 5.17 provides the results of all estimated NRTL BIPs for all liquid components with carbon monoxide. Analogous results for hydrogen are given in Table 5.18. A comparison of these BIPs against the results of the LLE data regression shows that for a successful description of the gas-liquid equilibrium a comparable number of binary interaction parameters is required as for the LLE cases. The main difference is perhaps given by the fact that for the description of the gas-liquid equilibria it is necessary to consider other values of the variable $\alpha_{j,i}$, or rather $c_{j,i}$ and $d_{j,i}$ (see Eq. (B.3)), than the recommended default values for common VLE and LLE description.

An evaluation of the quality of the data regression for each single gas-liquid pair is given in Table 5.19. The corresponding average relative deviations clearly show that satisfactory

[28] Reprinted with permission from Merchan and Wozny [136]. Copyright 2016 American Chemical Society.

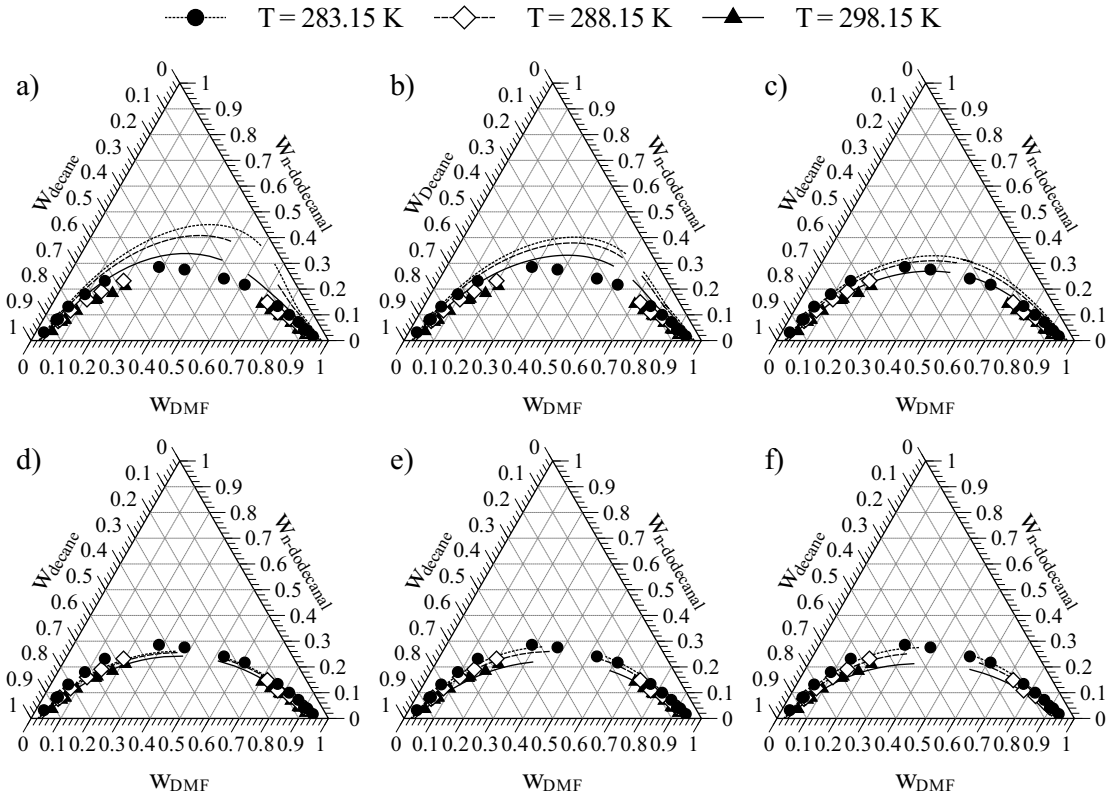

Figure 5.12: Comparison of correlative capabilities of considered parameterizations for the hs-PCP-SAFT (a-c) and the SRK-MHV2 (d-f) EoS Models. Diagrams a, b, and c correspond to parameterizations cases B, C and D (see Table. 5.9), diagrams d, e, and f correspond to cases A1, B1, and C1 (see Table 5.10)

fits are obtained in all cases. Table 5.19 also provides the $k_{i,j}$ values[29] that were used to generate the simulation data that were used to fit the SRK-MHV2 EoS model, and the number of NRTL BIPs required to fit the GLE data satisfactorily. While the PCP-SAFT EoS model comes out with a single BIP, the SRK-MHV2 approach requires in most cases at least 4 BIPs. This clearly shows the superiority of the PCP-SAFT approach to describe the concerning gas-liquid equilibria.

While the results of Table 5.19 clearly show that binary gas liquid equilibrium of all considered gas-liquid binary pairs can be successfully modeled with the SRK-MHV2 EoS model, it is still an open question whether this model in conjunction with BIPs from Table 5.17 and 5.18 is suitable for modeling gas solubilities in multicomponent liquid mixtures. In order to evaluate this issue, Figure 5.13 presents a comparison of the calculated syngas solubilities in liquid mixtures of either DMF and decane or *n*-dodecanal and 1-dodecene. The results are given for different ratios of the liquid components and pressure levels of

[29]This is a common naming for the BIPs considered by the PCP-SAFT EoS model.

Table 5.17: Binary NRTL parameters regressed from gas-liquid equilibrium of CO/pure solvent pairs generated with PCP-SAFT in a range of 0.1 to 3.5 MPa

i	CO	CO	CO	CO	CO
j	dodecene	n-dodecanal	dodecane	DMF	decane
$a_{i,j}$	−37.649	87.892	−43.075	1.289	−273.121
$a_{j,i}$	20.328	−71.456	3.949	0	−0.770
$b_{i,j}$	0	−17 634.380	0	0	0
$b_{j,i}$	0	5468.303	0	0	0
$c_{i,j}$	0.189	0.187	0.217	−0.551	0.148
$d_{i,j}$	0	0	0	0.007	0
$e_{i,j}$	0	0	0	0	51.033
$e_{j,i}$	−3.505	9.591	0	0	0
$f_{i,j}$	0.190	−0.024	0.194	0	0
$f_{j,i}$	0	0	−0.011	0	

Table 5.18: Binary NRTL parameters regressed from gas-liquid equilibrium of H_2/pure solvent pairs generated with PCP-SAFT in a range of 0.1 to 3.5 MPa

i	H2	H2	H2	H2	H2
j	dodecene	n-dodecanal	dodecane	DMF	decane
$a_{i,j}$	84.483	−709.795	184.551	−75.444	−416.921
$a_{j,i}$	−2.334	6.467	4.159	−4.304	−1.491
$b_{i,j}$	0	0	−36 486.050	3320.349	0
$b_{j,i}$	543.313	0	0	1841.103	0
$c_{i,j}$	0.044	0.108	0.072	0.250	0.048
$d_{i,j}$	0.000 18	0	0	0	0
$e_{i,j}$	0	131.331	0	11.310	86.051
$e_{j,i}$	0	0	0	0	0
$f_{i,j}$	0	0	0	0	0
$f_{j,i}$	0	−0.035	−0.011	0	0

1, 10 and 20 bar, at a temperature of 363.15 K. For the full specification of the pT-Flash calculation, the feed compositions of CO and H_2 are given by molar fraction of 0.05.

Figure 5.13 provides several interesting details. While the results regarding the pair 1-dodecene/*n*-dodecanal show a very good accordance between the models for all considered pressures and compositions, and very low changes in solubility due to the presence of a second compound, the results concerning the DMF/decane system require a more detailed analysis.[30] As a rather trivial result from Figure 5.13, it can be seen that the accordance of the models regarding the gas solubility in the pure liquids is as good as expected according to Table 5.19. Also, it is clearly shown that the syngas solubility in DMF is much lower than in decane. However, some larger deviations appear when considering the solubility in DMF/decane mixtures. Though the fact that a growing amount of DMF leads to

[30] This latter point is not only justified by the fact that these two compounds have the highest concentrations in reactor and decanter (see Figure 5.1).

Table 5.19: Considered Gas/Liquid Binary Interaction Pairs and Resulting Absolute Relative Deviation of Gas Solubilities of Binary Pairs

Binary pair	PCP-SAFT $k_{i,j}$ used	Number of BIPS required by EoS/g^E	ARD %
H_2/dodecene	0.065	5	3.07
H_2/dodecanal	−0.008	5	1.62
H_2/DMF	−0.087	6	0.19
H_2/decane	0.101	4	3.41
H_2/dodecane	0.149*	5	3.27
CO/dodecene	0.093	5	2.55
CO/dodecanal	0.065	7	1.03
CO/DMF	0.0092*	3	2.00
CO/decane	0.089	4	2.94
CO/dodecane	0.125	5	1.75

* denotes $k_{i,j}$ values obtained in this work. All others are from [214]

a reduction of the syngas solubility in the DMF/decane mixture is qualitatively rightly predicted by the SRK-MHV2 EoS model, the quantitative differences of the resulting molar fractions are in most cases slightly larger than 10 %. The largest deviations appear in cases where both liquid compounds are available in comparable amounts. This result indicates that care must be taken when evaluating and analyzing results of model-based methods applied for multicomponent GLE calculations made with the SRK-MHV2 EoS model. For example, based on the fact that the reactor of the considered process is operated at a temperature of 363.15 K and a pressure of 20 bar [226], and on the basis of the results shown in Figure 5.13, it would not be recommendable to use the reactor kinetic published by Kiedorf et al. [98] using solubilities calculated with the SRK-MHV2-model.[31] If, however, the influence of gas solubility calculations is not expected to have a major influence on the decision to be supported by model-based methods, the errors introduced by the model may be acceptable.[32]

For the sake of completeness, and in order to indicate how the model selection could influence a rigorous decanter model, Figure 5.14 shows the pressure dependent syngas solubility at 283.15 K and 363.15 K for three different DMF/decane ratios, as considered in [214]. The lower row of Figure 5.14 basically confirms the results from Figure 5.13 showing on one hand the lower syngas solubilities as a consequence of the higher concentrations of DMF, and on the other hand, that the SRK-MHV2 EoS model, in general, predicts lower solubilities as the PCP-SAFT EoS model. More interesting perhaps are the results of the

[31] It should however be noted, that this would even be not recommended for the PCP-SAFT EoS model, since the kinetic parameters by Kiedorf were estimated based on fixed solubilities calculated with a Henry approach.

[32] For example for a three-phase flash calculation as it would appear when rigorously modeling the decanter unit (see Figure 5.1 and Figure 5.14).

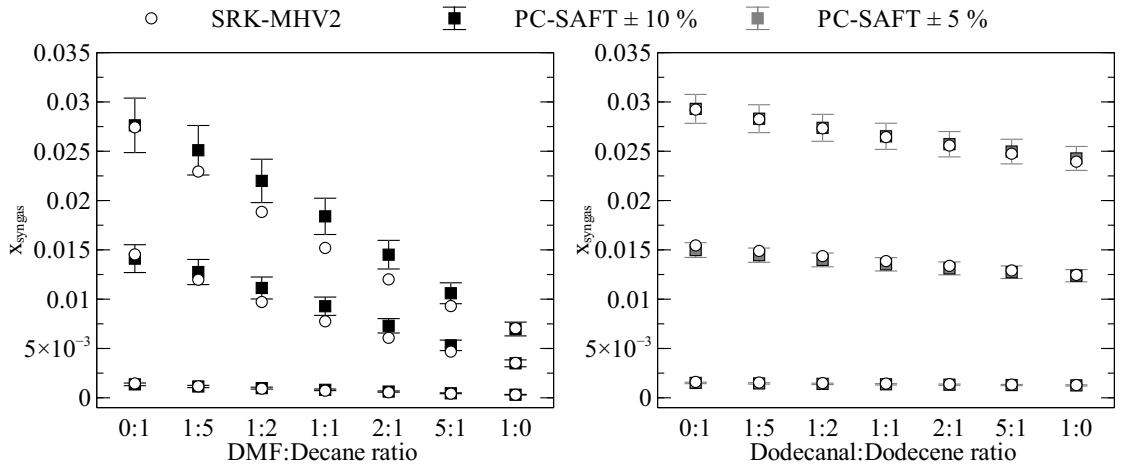

Figure 5.13: Composition dependence of syngas solubility in binary liquid mixtures: SRK-MHV2 vs PCP-SAFT at 1, 10 and 20 bars at 363.15 K: the bars provided with the squares are given to enclose a region of the ± 5 or ± 10 % around the PCP-SAFT prediction

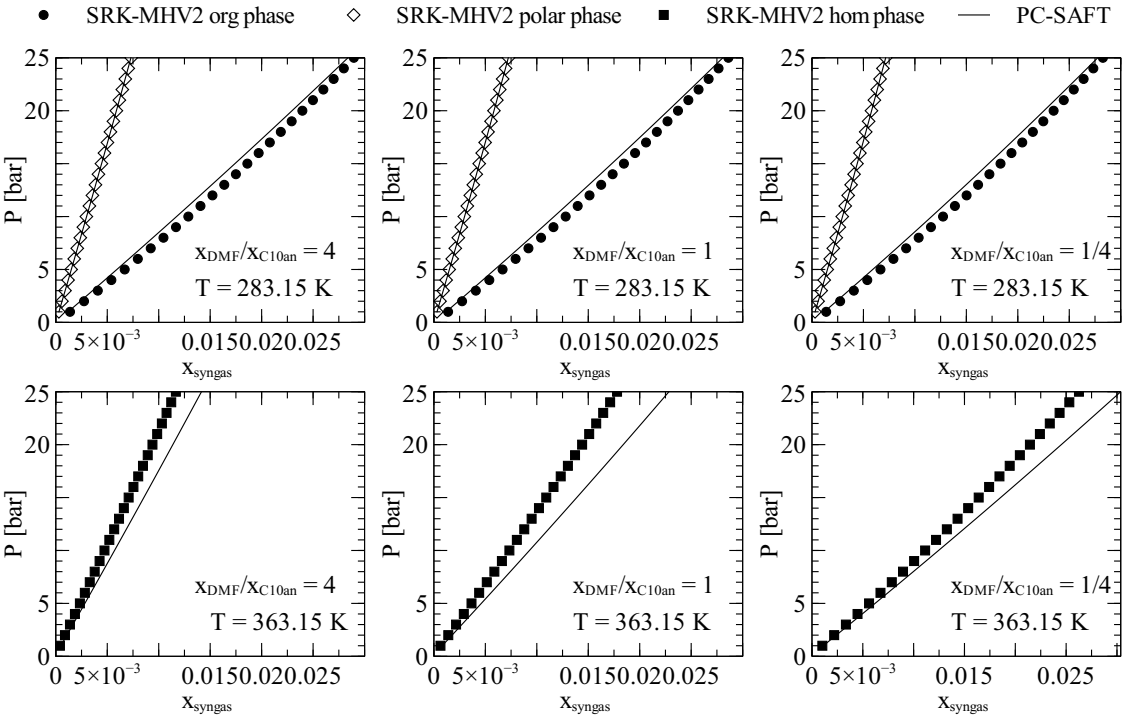

Figure 5.14: Pressure dependent syngas solubility in a DMF/n-decane mixture at different temperatures and feed compositions. Results are obtained from flash calculations for a mixed feed with $x_{H2} = x_{CO} = 0.05$

first row of Figure 5.14, the results at 283.15 K, since these show the gas solubilities in two liquid phases that are in equilibrium with each other. To the knowledge of the author

these are the first published gas solubility results obtained with the PCP-SAFT EoS model for a total of three phases in equilibrium. The equilibrium results were obtained from the solution of a multiphase equilibrium stage model as formulated in [144] or in the Appendix of [221]. In general, the results show a very good agreement between the calculations of the PCP-SAFT and SRK-MHV2 EoS models independent of the considered feed distribution. In fact, the feed composition does not appear to have an influence on the calculated syngas solubility. These two main characteristics of the results can be indirectly explained by the results of the LLE equilibrium of the binary DMF/decane system.[33] First, for a given temperature, the LLE molar fractions at equilibrium are independent of the feed composition. The same appears to be valid for the calculated syngas compositions. As a second aspect, it should be considered that, at the considered temperature, the composition of the liquid phases in equilibrium is very close to composition of pure phases. Since a very good accordance between the models is given for the gas solubility in pure compounds, it also makes sense that a good accordance is reached for almost pure phases, as they are calculated from the LLE for DMF/decane.

As expected, it appears that the additional presence of gaseous components does not have a significant influence on available liquid phases in equilibrium. The influence of dissolved gases is expected to have a decisive influence on the hydroformylation reaction but not on phase equilibrium calculations, where no reactions are considered.

5.5 Plant Wide Process Simulation: Simplified Process Model with Rigorous Liquid-Liquid Phase Equilibrium Calculations

As previously described, one of the main goals of this thesis is to provide a rigorous plant wide simulation of the hydroformylation process presented in Section 5.1.

In the former sections, particularly in Section 5.4, a relatively large effort has been put in the thermodynamic consistent description of the phase equilibria that appear in the considered hydroformylation process. In the following, based on the thermodynamic models studied previously, it should be evaluated how the quality of the description of the phase equilibria, in particular the liquid liquid equilibria that take place in the decanter, affects the overall predicted performance of the coupled multicomponent reaction/separation process with full recycle (Figure 5.1). Besides introducing a method to evaluate the influence of the selected thermodynamic model, this simulation study should allow to validate the resulting process models against experimental data from a real mini plant presented in

[33] That LLE behavior is shown in Figure 1 of [181]. It is qualitatively comparable with the LLE of 1-dodecene and DMF shown in Figure 5.6.

[226, 227]. Further details concerning modeling goals, modeling assumptions, and results of simulation studies are given next.

5.5.1 Modeling goals

The simplified nature of the model is mainly given by the fact that the number of considered compounds is reduced as much as possible and that no reaction kinetics, as provided in [98], are considered. There are several goals linked with the development of the simplified process wide model studied in this section. The most important goals are summarized in the following list:

- The simplified model should enable a quantification of the selection of the thermodynamic model on the calculated/predicted overall process performance. In particular, following thermodynamic models should be considered:

  1. UNIFAC-DO: UNIFAC-Dortmund considering the default group distribution and function group parameters as available in Aspen Plus version 8.2.
  2. hs-PCP-SAFT [181]: Heterosegmented PCP-SAFT EoS model with pure compound parameters and BIPs as provided by Schaefer et al. [181].
  3. SRK-MHV2 ($k_{C12en,C12al} = 0$): SRK-MHV2 EoS model with NRTL as underlying g^E model with BIPs from Table 5.14 and neglecting binary interaction between 1-dodecene and *n*-dodecanal.
  4. SRK-MHV2 ($k_{C12en,C12al} \neq 0$): SRK-MHV2 EoS model with NRTL as underlying g^E model with BIPs from Table 5.14. NRTL-BIPs between 1-dodecene and *n*-dodecanal are estimated based on UNIFAC Dortmund.
  5. hs-PCP-SAFT (best fit) : Heterosegmented PCP-SAFT EoS model with pure compound parameters and BIPs obtained from parameterization case D treated in this work (see Table 5.11 and Table 5.12).

- For all considered models the interdependence between reactor performance, process performance, product flow/recycle flow ratio and decanter separation temperature should be studied.

- The model formulation should be chosen in a such a way that the results are independent of magnitudes of feed and the size of reaction and separation devices. This indicates that rigorous phase equilibrium thermodynamics can be considered, since equilibrium state is size independent. However, no rigorous reaction kinetics should be considered, since such would describe size dependent reaction rates.

- Since the main information to be obtained is the influence of phase equilibrium calculations, the developed model should be conceived in such a way that potential weaknesses of the considered thermodynamic model, for example regarding the calculation of volumetric or thermal properties, do not have a negative influence on the considered process wide simulations.
- Finally, the model should enable the evaluation of the validity of the model assumptions listed below. In particular this is done by comparing the process wide simulation results against experimental data.
- Besides providing valuable process information, the simplified model can be seen as a robust method to generate initial guesses for more detailed model implementations containing more phenomena and constraints, such as well developed reaction kinetics.

5.5.2 Main modeling assumptions / reduction of number of compounds

Most assumptions considered for the development of the simplified process wide model are related to the reduction of the number of compounds. These assumptions are listed next:

Catalyst complex is neglected The components 10 to 12 from Table 5.1 are neglected. Though the catalyst and ligand concentration have a decisive influence on the actual reactor performance, this assumption is fully supported by the modeling goals. According to the latter, the reaction performance should adapt in such a way that the overall process works for the conditions defined by the phase equilibrium in the decanter. Besides that, due to the very low amounts of catalyst complex in the reacting mixture it is assumed that these compounds can be neglected without effecting the phase equilibrium calculations. It should be considered that their molar fractions differ from the main components' molar fractions by 3 to 4 orders of magnitude.

Dodecane is neglected Dodecane appears in the process as the product of the hydrogenation of 1-dodecene. Due to the relatively low amounts of dodecane that are generated in the reaction [98] it appears to be a legitimate assumption to neglect it. The fact that almost no BIPs with dodecane are available also indicates that the consideration of dodecane is an additional source of uncertainty.

Linear compounds (n-compounds) and iso-compounds are lumped to single alkenes and aldehydes A major uncertainty in current modeling approaches is given by the fact that the considered iso-compounds, as they appear in Table 5.1, are artificially introduced to represent many different iso-compounds that can appear in low amounts, and by the fact

that no phase equilibrium measurements have been realized to quantify the interactions of these iso-compounds with solvents, reactants and main products.[34] In spite of the uncertainty given by the availability of these iso-compounds inside the liquid phases, a proper approach is still required in order to quantify the influence of these iso-compounds, at least in a qualitatively right way. For example, simply neglecting BIPs for the interaction between DMF and some iso-dodecene, would be equivalent to suppose that these components build an ideal mixture with no phase split. This is certainly not the case.

One feasible approach to quantify the binary interactions of lumped iso-compounds with other compounds is to assume that the BIPs valid for interactions with linear compounds are also valid for the corresponding iso-compounds. In view of the fact that *volumetric* pure compound properties of iso-compounds and their linear counterparts are very similar, the consideration of iso-compounds with equal binary interactions as the linear compounds would lead to the same calculation results that would be obtained when the linear and the corresponding iso-compound would be replaced by a lumped or general compound with the same properties and BIPs.[35]

As an example, consider the hypothetical pT-flash calculation I of a ternary feed consisting of 1-dodecene (nC12en), iso-dodecene (iC12en) and DMF, where 1-dodecene and iso-dodecene are supposed to have the same pure compound properties. The resulting two liquid phases in equilibrium, A and B, are characterized by the equilibrium molar fractions $x^{A,I}_{nC12en}, x^{A,I}_{iC12en}, x^{A,I}_{DMF}$ of the organic phase A and $x^{B,I}_{nC12en}$, $x^{B,I}_{iC12en}$, $x^{B,I}_{DMF}$ of the polar phase B. The superscript I denotes the considered flash calculations. If for a different pT-flash calculation, denoted II, the compounds 1-dodecene and iso-dodecene are added to build a general dodecene (C12en), then for the results of the pT-flash calculation II the relations presented in Eq. (5.102) to Eq. (5.105) must be fulfilled.

$$x^{A,II}_{C12en} = x^{A,I}_{nC12en} + x^{A,I}_{iC12en} \tag{5.102}$$

$$x^{A,II}_{DMF} = x^{A,I}_{DMF} \tag{5.103}$$

$$x^{B,II}_{C12en} = x^{B,I}_{nC12en} + x^{B,I}_{iC12en} \tag{5.104}$$

[34] In view of the large number of possible iso-compounds and the relatively low amount in which they appear, it probably does not even make sense to make a detailed investigation of the relevant phase equilibria.

[35] It should be noted that this is only the case for thermodynamic models that do not exhibit the so-called Michelsen-Kistenmacher syndrome [139].

$$x_{DMF}^{B,II} = x_{DMF}^{B,I} \tag{5.105}$$

The explanations and the example given above make clear that it may be reasonable to reduce the number of compounds in cases where iso-components are expected to have similar binary interactions as the corresponding linear components. An obvious drawback of explicitly neglecting iso-compounds is given by the fact that direct ways to quantify their influence, for example from isomerization reactions or reactions where iso-compounds are involved, get lost. However, as shown by Eq. (5.102) or Eq. (5.104), the phase equilibrium calculation can indirectly consider the influence of such iso-compounds. The general compounds can be thought to consist of different proportions of linear and iso-compounds, which could be a consequence of isomerization reactions or further reactions involving such compounds.

Gases are neglected Though gas solubility has a decisive influence on the performance of the hydroformylation, a detailed quantification thereof is not the primary objective of this work. While [98] shows how process parameters as temperature and pressure have an effect on gas solubilities and hence on the chemical reactions, at the level of the simplified model it is rather of interest, how phase separation conditions affect the process performance and thus the corresponding (theoretical) reactor performance. Independent of the aforementioned justification, additionally, it should be remarked that even at the considered pressure levels around 2 MPa the molar fractions of solved gases in the liquid phases are significantly lower than the ones of the main liquid components. Additionally, and as already shown by the results from Figure 5.14, it should be considered that the presence of gas components has a negligible influence on the liquid phases and the products that leave the decanter.

Having neglected the catalyst complex, dodecane, both gases CO and H_2, and having joined the linear and isomeric compounds to general compounds the total number of components could be reduced from 13 to 4 (see Table 5.1).

5.5.3 Main modeling assumptions / missing important information

In addition to the assumptions that were introduced to reduce the number of compounds, there are further assumptions, whose introduction is related to the fact that important information is missing. These assumptions are listed next.[36]

[36] Based on this definition, the assumption regarding the introduction of general alkenes and aldehydes could have been assigned to this category as well.

Dodecanal is considered instead of tridecanal This assumption is based on the remark on unavailable/missing data, which was already introduced as "dodecanal/tridecanal" issue in Section 5.4.1. The produced aldehyde will be defined to have exactly the same (temperature dependent) phase behavior as dodecanal.

Though there are several valid ways to realize such a demand, for example to fit tridecanal-DMF-decane LLEs against simulation data generated from the best parameterization found with hs-PCP-SAFT or SRK-MHV2 for ternary systems with n-dodecanal, for the sake of a simple model implementation, n-dodecanal will be considered directly as the produced aldehyde. From the perspective of rigorous modeling, such an assumption represents a major error, since the elemental balance would be violated. However, in view of the considered modeling goals, the error seems to be acceptable. Since the main goal of the process lies on quantifying the influence of the phase separation in the decanter on the overall process, it is irrelevant whether tridecanal or dodecanal is considered, since both are considered to have the same phase behavior. Furthermore, the process is not supposed to provide any additional calculation based on any additional compound specific property such as molar mass, etc.

Reaction performance characterized by size independent reaction rate As a consequence of neglecting dodecane and defining general compounds that combine linear and isomeric alkenes and aldehydes into corresponding lumped compounds, there is only the need to model one single reaction, namely the conversion from general dodecene, characterized by the properties of 1-dodecene, into the general aldehyde, characterized by the properties of n-tridecanal(actually dodecanal). Following the modeling goal that the reaction should be quantified independent of size, the single reaction is best described by a single reaction rate r. Based on the flowsheet representation in Figure 5.1 the component specific mass balance for any component c entering or leaving the reactor is given by Equation (5.106)

$$\sum_{s=4}^{5} F_{s,c} \cdot x_{s,c} - \sum_{s=6}^{7} F_{s,c} \cdot x_{s,c} + \nu_c \cdot r = 0 \tag{5.106}$$

Eq. (5.106) is important for the solution of the model of the coupled reactor/decanter system with full recycle. For given feed conditions of the decanter or for given feed conditions of the overall process, the variable r builds the link between the production of aldehyde and the consumption of alkene, closing the gap for finding feasible solutions. This will be explained in more detail in the simulation studies presented below.

Though the single reaction rate r is a valuable variable, not only for the quantification of the reaction but mainly for practical reasons indicated above, there are other variables that can help to better interpret and evaluate the reactor and process performance. These

are the reactor performance X_R defined according to Eq. (5.107), and the process wide overall aldehyde yield $Y_{C13al}^{process}$ defined by Eq. (5.108). Note that the subscript $C13al$ is thought to denote the general aldehyde produced by the reaction, which is described with the properties and phase behavior of n-dodecanal.

$$X_R = \frac{F_{s=4} \cdot x_{s=4,C12en} - F_{s=6} \cdot x_{s=6,C12en}}{F_{s=4} \cdot x_{s=4,C12en}} \tag{5.107}$$

$$Y_{C13al}^{process} = \frac{F_{s=9} \cdot x_{s=9,C13al}}{F_{s=1} \cdot x_{s=1,C12en}} \tag{5.108}$$

From the above performance indicator variables, the process wide yield $Y_{C13al}^{process}$ is of particular importance, since it represents the only variable that can be considered to validate the process model against experimental data from Zagajewski et al. [227]. The fact that, according to the modeling assumptions, no distinction is made between linear and isomeric compounds does not affect the quality of this validation that much, since experimental work, for example [27, 181] and [227], clearly shows that the yield towards isomeric aldehydes is negligible in comparison to the yield towards linear aldehydes.[37]

Though the alkene conversion of the reactor as defined in Eq. (5.107) provides valuable information on the required reactor performance it cannot be taken for model validation, since no detailed experimental information is available on the recycle and hence the actual flow conditions entering the reactor are unknown.

Model implementation and solution strategy

Following all assumptions listed above, the simplified plant wide model results from formulating the phase equilibria of the decanter and the component balances and summation equations of the mixer, the reactor and the decanter. Since besides pressure and temperature in the decanter, which will be considered to be fixed, no further information regarding pressure or temperature of the process are required there is no need to define energy or momentum balances. The resulting model equations for the reactor and decanter are provided in [136]. Note that in [136], for the sake of completeness, equations related to gas compounds are listed, though they could be completely taken out of the equation system without affecting the solution.

The model equations are implemented in MOSAICmodeling. For calculations with UNIFAC-Dortmund and the SRK-MHV2 EoS model gPROMS code is generated with external function calls for fugacity coefficients [135]. The corresponding thermodynamic packages were

[37] This is in fact one of the main characteristics of the considered catalyst/ligand combination.

generated as CAPE-OPEN property packages set up in Aspen Plus Version 8.2. The examples concerning implementations of the hs-PCP-SAFT EoS model were calculated with MATLAB. The fugacity coefficient calls were integrated as external functions, based on an implementation of the Helmholtz free energy, where derivatives required for the evaluation of fugacity coefficients where calculated with automatic differentiation.

Although the considered model represents a very simple and small equation system, which can be solved by a general Newton-based solution approach, depending on the chosen set of design variables, different solution strategies may be beneficial. This will be shortly discussed in the relevant cases below.

5.5.4 Simulation studies with simplified process model: Influence of reactor product flow

This first simulation study based on the simplified model is not only designed to provide interesting information about the process wide operation but also to clarify some characteristics of the model that result from the model assumptions discussed above.

Table 5.20: Considered decanter feed conditions for first simulation case with simplified process-wide model

Feed	w_{C12en}	w_{C13al}	w_{DMF}	w_{C10an}
1	0.01	0.05	0.47	0.47
2	0.03	0.15	0.41	0.41
3	0.05	0.25	0.35	0.35

In this first simulation, the influence of the concentration of a hypothetical product that leaves the reactor on the overall process performance is studied. A possible use of such a simulation could be to provide a first estimation of overall process behavior based on results from batch reactor experiments. For this study, three different reactor product compositions listed in Table 5.20 are considered. The compositions are chosen in such a way that the influence of the amount of produced aldehyde at similar DMF/decane ratios can be studied. In addition to the decanter's feed composition, pressure and temperature, the inlet flow stream of the decanter needs to be specified for the process wide simulation to become fully specified. For given operation conditions of the decanter, including its feed flow, the corresponding outlet stream flows and concentration are obtained. Since the recycled component stream flow of aldehyde is now known, the reaction rate r follows from its component balance around the reactor. With known reaction rate the feed flow of alkene coming into the reactor, and into the overall process can be calculated. The overall feed amounts of DMF/decane follow directly either from component mass balance around

the whole process or from consideration or from mass balances around the reactor and the mixer (see Figure 5.1).

Although the selected value of the feed flow obviously has an influence on the magnitude of the calculated flows and reaction rates, it does not have any influence on the calculated molar fractions and further variables that are considered to evaluate the process, thus fulfilling the modeling goals. In addition to the overall process yield $Y_{C13al}^{process}$ defined in Eq. (5.108), the product/recycle ratio F_{prod}/F_{rec} (Eq. (5.109), and the aldehyde separation efficiency ϵ_{C13al} (Eq. (5.110)) are introduced as process wide performance indicators.

$$F_{prod}/F_{rec} = \frac{F_{s=9}}{F_{s=10}} \tag{5.109}$$

$$\epsilon_{C13al} = \frac{F_{s=9} \cdot x_{s=9,C13al}}{F_{s=8} \cdot x_{s=8,C13al}} \tag{5.110}$$

Figure 5.15 and Figure 5.16 summarize the results of different simulations designed to quantify the introduced performance indicators for different reactor outlets (Table 5.20), decanter temperatures and different thermodynamic models.

Numerous findings can be won from Figure 5.15 and Figure 5.16. Regarding the evaluated process wide performance indicators, it appears that all models provide values of similar magnitude at temperature below 285 K. However, at higher temperatures differences become relevant, in particular for cases with higher aldehyde compositions, feed 2 and 3. With respect to the temperature dependence, it is interesting to note that the hs-PCP-SAFT EoS model as proposed by Schaefer et al. [181] shows in most cases similar trends as the UNIFAC-Dortmund, while both approaches based on the SRK-MHV2 EoS model and the hs-PCP-SAFT approach with parameters found in this work (case D from Tables 5.11 and 5.12) show contrary trends. From the comparison of the different approaches based on the SRK-MHV2 EoS model, it follows that the additional consideration of binary interactions between dodecene and the produced aldehyde does not appear to have any influence at low temperatures and low aldehyde concentrations. However, in regions of high temperature and high aldehyde concentrations the differences can be significant as shown in Figure 5.15 c, making clear that the right description of this binary interaction could represent a bottleneck under certain conditions. In any case, care should be taken when analyzing regions that show strong changes. The fact that positive changes regarding higher product/recycle ratio F_{prod}/F_{rec}, process wide yield $Y_{C13al}^{process}$ and aldehyde separation efficiency ϵ_{C13al} appear at operations regions of high aldehyde concentration and high decanter temperature should not lead to the conclusion that high temperatures in general have a positive impact on the process operation. The strong and sudden changes observed

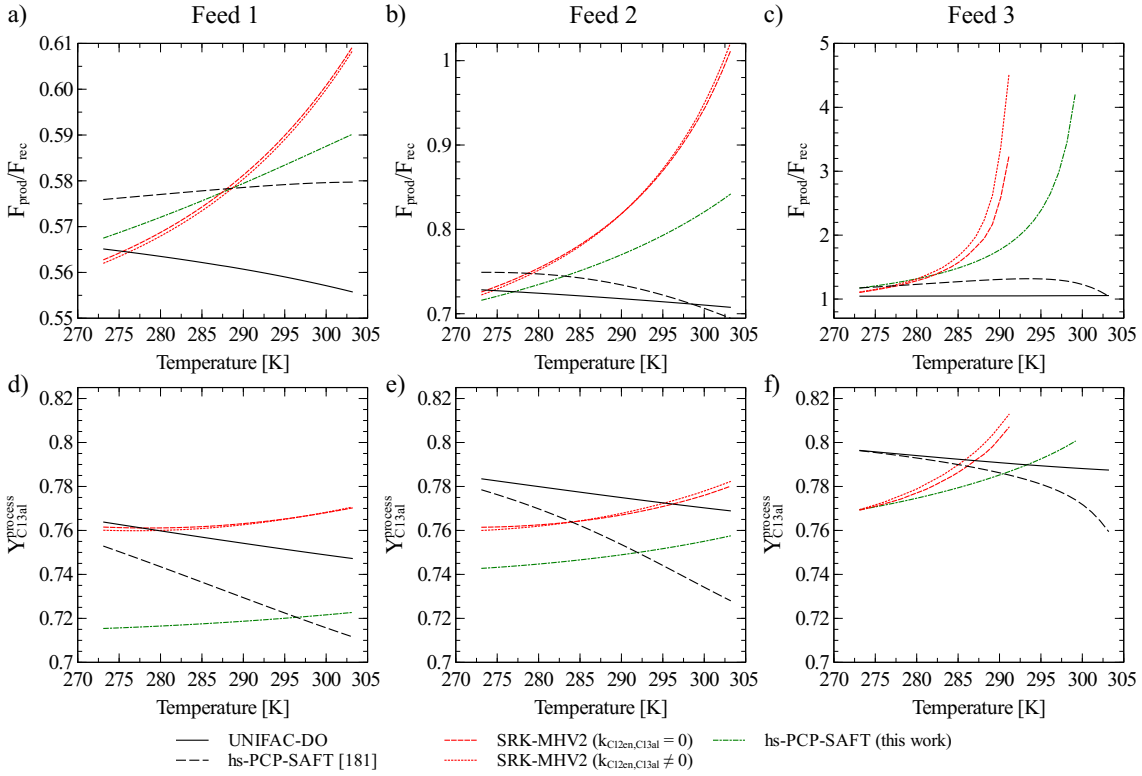

Figure 5.15: Temperature dependence of product/recycle ratio and process wide yield for different reactor product compositions from Table 5.20 and different thermodynamic models

in the process indicators are rather related to the fact that the considered region lies close to the critical region where the mixing gaps ceases to exist. Whether concentration or temperature changes have a positive effect on the overall process depends on the considered decanter feed position and the shape of the phase equilibrium region close to that point.

The high differences in predicted process performance at high aldehyde concentrations are consistent with the big deviations found at ternary systems of *n*-dodecanal, DMF and decane at similar conditions, see Figure 5.8. Generally speaking, the results presented above very strongly depend on the arbitrary fixed DMF/decane ratio at the decanter inlet and hence are not expected to provide a highly valuable information about the true process. However, the example makes already clear that there are highly sensitive operation regions for which small changes can have strongly positive or negative changes on the overall process performance. These findings point out the importance of a quantitatively good description of phase equilibria. Only such a description should be the basis of right decisions based on model-based methods.

Next, the question should be answered, whether the thermodynamic description and the

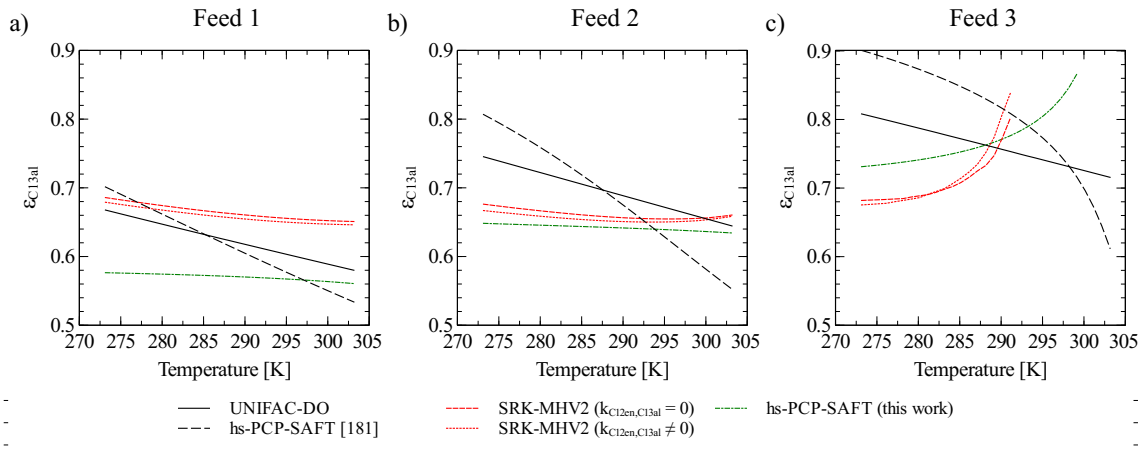

Figure 5.16: Temperature dependence of recovery efficiency of aldehyde for different reactor product compositions from Table 5.20 and different thermodynamic models

model assumptions have led to a simplified model that can reproduce experimental results obtained with the real miniplant.

5.5.5 Simulation studies towards simplified process model: Model validation and influence of density calculation

As already mentioned in the model assumptions, the only available way to validate the process wide simplified model is by comparing the calculated and experimental process wide aldehyde yield $Y_{C13al}^{process}$ obtained for given process feed conditions. The most important experimental feed conditions provided by Zagajewski et al. [227] that are relevant for this simulation case are summarized in Table 5.21. The experimental value of the process wide aldehyde yield is taken as the end value of the time profiles of the n-aldehyde time profiles. It should be remarked that the aforementioned time profiles reach a stable, nearly steady state behavior.

Table 5.21: Important process feed conditions as provided by Zagajewski et al. [227]

$\dot{V}^{feed,org}$	$60\,\mathrm{mLh^{-1}}$
w_{C10an}	0.724
w_{nC12en}	0.276
$\dot{V}^{Make-Up}$	$4.5\,\mathrm{mLh^{-1}}$
w_{DMF}	0.9927*
$T_{decanter}$	278.15 K
$p^{process}$	20 bar

*For the sake of simplicity w_{DMF} is considered to be 1

In order to simulate the process wide behavior based on the provided values from Ta-

ble 5.21, it is necessary to convert the given volumetric flow data into corresponding molar (or mass) flow information. This conversion requires the evaluation of the density of the considered mixtures. Note however, that such additional density calculations, which can be seen as an additional source of uncertainty, would not be necessary if experimental feed values would be available as molar or mass flow information.[38] In the following, for this density evaluations both, the PCP-SAFT and the SRK-MHV2 EoS models, are considered. Table 5.22 summarizes the equivalent total molar and mass feed flows and the corresponding compositions when all feed components are put into a single hypothetical feed stream.

Table 5.22: Process feed conditions from Table 5.21 as total inlet molar and mass flows with corresponding molar/mass fractions. Results presented considering density calculation with PCP-SAFT and SRK-MHV2 EoS models

Density calculated with PCP-SAFT [181] *				Density calculated with SRK-MHV2 (Table 5.14)			
F^{feed}	$0.351\,\mathrm{mol\,h^{-1}}$	$\dot{M}^{feed}$	$47.942\,\mathrm{g\,h^{-1}}$	F^{feed}	$0.281\,\mathrm{mol\,h^{-1}}$	$\dot{M}^{feed}$	$38.929\,\mathrm{g\,h^{-1}}$
x^{feed}_{nC12en}	0.204	w^{feed}_{nC12en}	0.252	x^{feed}_{nC12en}	0.211	w^{feed}_{nC12en}	0.256
x^{feed}_{DMF}	0.162	w^{feed}_{DMF}	0.087	x^{feed}_{DMF}	0.134	w^{feed}_{DMF}	0.071
x^{feed}_{C10an}	0.634	w^{feed}_{C10an}	0.661	x^{feed}_{C10an}	0.655	w^{feed}_{C10an}	0.673

*Since no long chain aldehydes appears in the considered flows, there is no need to consider any parameters estimated in this work

As the results indicate, if inlet flow information is only available as volumetric flow, the right conversion into molar or mass flow information may be a decisive step for a proper model validation. Therefore, in the results presented below, the influence of the conversion of volume flow into molar or mass flows, is considered as well. Regarding the resulting molar and mass flows provided in Table 5.22, it can be said that the calculation results obtained with PCP-SAFT are more realistic. Molar volumes of pure compounds and mixtures concerning the considered thermomorphic solvent system calculated with hs-PCP-SAFT are in very good accordance with results obtained from specialized molar volume models/correlations such as the COSTALD (Corresponding STAte Liquid Density) method. Regarding the capabilities of thermodynamic EoS models to correlate and predict liquid volumetric properties and phase equilibria, a further remark needs to be done. Although the evaluation of volumetric properties is part of the evaluation of fugacity coefficients ($\varphi_i(Z)$, see Eq. (2.27)), a successful correlation of phase equilibria should not lead to the assumption that the values of volumetric properties that are calculated inside phase equilibrium calculations have a comparable quality to the description of phase equilibria. For the sake of a better overall description of volumetric properties and phase

[38] This is not only valid for the simplified model but for any type of model that does not explicitly require volumetric properties or conversions between such and molar/mass specific variables.

equilibria, it may be necessary to give up the wish of calculating all properties with one and the same model. Though not thermodynamic consistent, this is in fact common practice.

Following the same reasoning as in the previous simulation study regarding the influence of the composition of the reactor outlet, in addition to the decanter pressure and temperature, further four specifications need to be made in order to fully specify the equation system to be solved. Perhaps, the most intuitive approach would be to define the liquid feed flow and three molar fractions from Table 5.22. This approach is considered not to be safe, since common experience with reactor/separator systems with full recycle shows that such specification often leads to systems with no feasible physical solution. Instead of fully specifying the system inlets and predicting the recycle behavior, for such systems, it is more common to look for feed conditions that lead to a particular behavior with fixed recycle/ flows conditions.

In this work, an alternative approach is proposed that leads to comparable results as when specifying the feed, but that strongly reduces the risk of not finding proper solutions. Instead of fixing the values of the feed conditions, the overall feed flow F^{feed} and three corresponding molar fractions, two known outlet molar fractions and the molar flows of the process outlet and of the recycle are specified. Note that fixing the outlet flow and the outlet molar fractions of DMF and decane is completely equivalent to fixing the overall inlet flow and the associated molar fractions, since according to mass balance $F^{feed} = F_{s=9}$. The equalities $x_{s=9,DMF} = x^{feed}_{DMF}$ and $x_{s=9,C10an} = x^{feed}_{C10an}$ are valid as well. Having fixed the decanter's pressure and temperature and knowing at least two equilibrium compositions of the phases in equilibrium, according to Gibbs' phase rule all other remaining molar fraction result from an equilibrium calculation. Having specified three variables (without counting the decanter's temperature and pressure), according to the explanations of the previous simulation study, still one additional variable needs to be fixed in order to fully specify the equation system. For practical reasons, in the considered case the recycle flow appears to be a good choice, since it opens the possibility to study the relation between the product/recycle ratio and the reactor performance.

Figure 5.17 shows the results of the calculated process wide yields obtained for the feeds given in Table 5.22 and different thermodynamic models. Additionally, it provides the experimental value from [227] for comparisons. As previously indicated, the calculations are done for fixed decanter operation conditions, 20 bar and 278.15 K, fixed decanter outlet flows ($F_{s=9} = F^{feed}$) and fixed product concentration of DMF and decane ($x_{s=9,DMF} = x^{feed}_{DMF}$ and $x_{s=9,C10an} = x^{feed}_{C10an}$), all taken from Table 5.22. Since no measurements of the recycle flow $F_{s=10}$ are available, there is apparently no reference point for the selection of the value of this variable and hence no possibility of fully specifying the model as in the experiments. Interestingly, a more detailed analysis of the model shows that the selection of

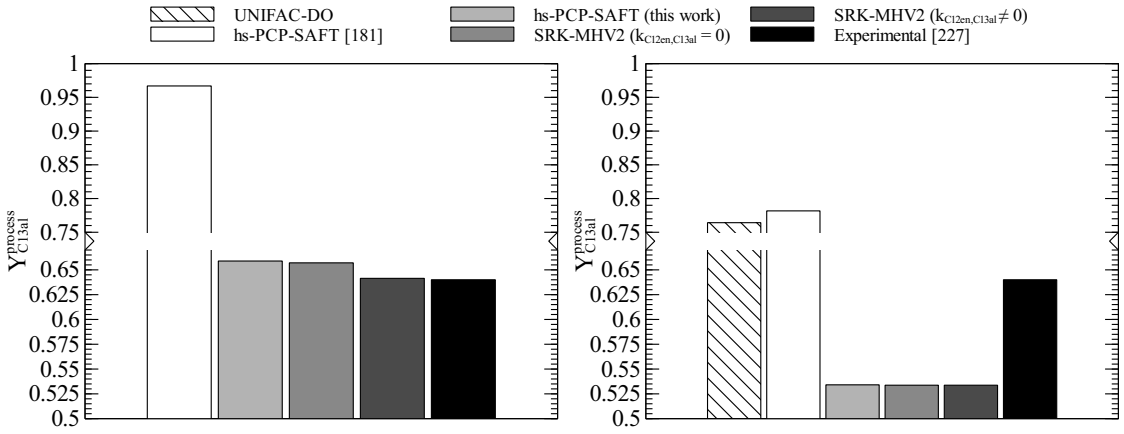

Figure 5.17: Comparison of experimentally obtained aldehyde yield against simulations using different thermodynamic models. Left: density calculation according to PC-SAFT. Right: density calculation with SRKMHV2 EoS model

the recycle flow has no influence on the calculated process wide yield at all. Considering an overall mass balance, the process wide aldehyde yield originally introduced in Eq. (5.108) can be rewritten as Eq. (5.111), hence as a variable whose value automatically results from the feed molar fractions of DMF and decane.

$$Y_{C13al}^{process} = \frac{x_{s=9,C13al}}{x_{s=9,C12en} + x_{s=9,C13al}} \tag{5.111}$$

Numerous facts follow from Figure 5.17. Similar to the results of the previous simulation study regarding the influence of reactor outlet composition, results calculated with hs-PCP-SAFT of Schaefer et al. [181] and UNIFAC-Dortmund on one hand, and with SRK-MHV2 and hs-PCP-SAFT parameterization of this work on the other hand, show a very high similarity among each other. Note that no results of the UNIFAC-Dortmund results are given on the left-hand side of Figure 5.17 since no feasible solution could be found for the given specifications. Regarding the conversion of the given volumetric flow into molar/mass flows (Table 5.22), it becomes clear that the selected approach for density calculation has a major influence on the calculated process wide aldehyde yield. For the feed conditions calculated using the SRK-MHV2 EoS model (Figure 5.17, right) none of the thermodynamic models used for phase equilibrium calculations leads to a good agreement with the experimental yield. On the other hand, considering volumetric inlet conditions obtained with PCP-SAFT EoS model (Figure 5.17, left) satisfactory results are obtained for the SRK-MHV2 approaches and heterosegmented PCP-SAFT with the parameters proposed in this work. It should be remarked that the results by Zagajewski et al. [227] were obtained under consideration of mean densities which are comparable to

those obtained with the PCP-SAFT EoS model.

In order to quantify the influence on the different product/recycle ratios on the process, Figure 5.18 shows the reactor conversions obtained when considering the inlet conditions shown above and different recycle/product ratios. As can be seen, based on such results, feasible pairs of reaction X_R and recycle/product ratios can be found that correspond to a given overall process Yield $Y_{C13al}^{process}$ at fixed amounts of solvents.

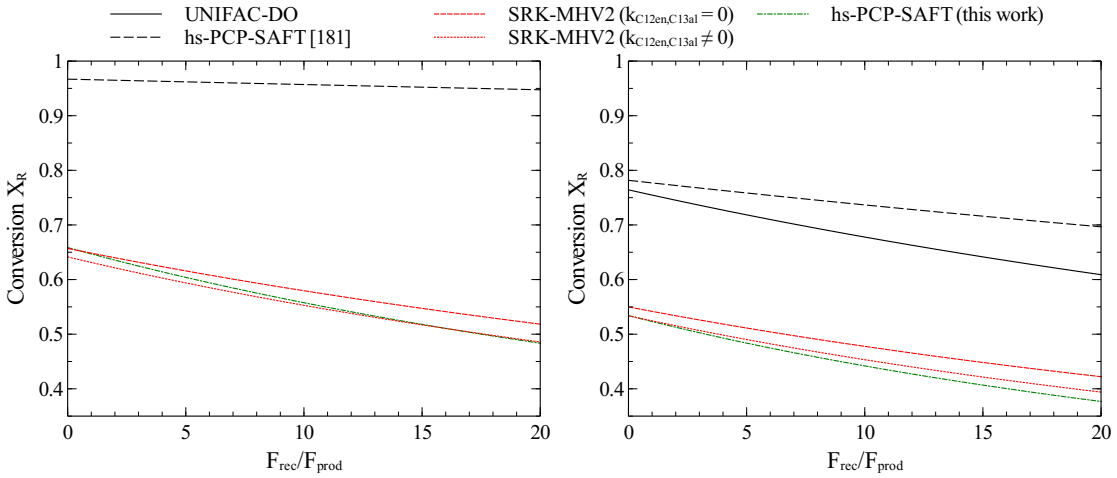

Figure 5.18: Comparison of calculated reactor conversion for solvent flows provided in Table 5.20, different recycle/product ratios and different thermodynamic models. Left: based on density calculation with PCP-SAFT EoS model. Right: density calculation with SRK-MHV2 EoS model

Though not thermodynamic consistent, for the sake of a better overall process description, it appears to make sense to calculate phase equilibria using the SRK-MHV2 EoS model with the parameter sets from Table 5.14, Table 5.17 and Table 5.18, but to calculate the molar volume (or molar density) using an alternative approach that provides better results. To avoid the kind of inconsistencies described above, apparently it would be sufficient to rely on overall mass or molar flow information for process validation, hence, to avoid the necessity of relying on density calculations. However, there are common cases in which density calculations are unavoidable and may have a big influence on the model validation. A common example is given by reaction kinetics, which are commonly given in concentration units.

From the perspective of the simplified model, the selection of different methods for density calculation on the feed conditions is completely equivalent to evaluating the model with different feed compositions. The big differences shown in process performance for the different models used for density calculation, motivates a detailed evaluation of the influence of the DMF/decane feed distributions that may be required to obtain a stable operation

at a particular reactor or process performance. This later point is the main subject of the simulation study presented below.

5.5.6 Simulation studies with simplified process model: Influence of plant feed conditions

The results shown in the previous section in Figure 5.17 and Figure 5.18 clearly indicate the influence on the DMF/decane distribution on the resulting process yield. In order to quantify this influence in a more detailed way, Figure 5.19 shows the corresponding process yields (Eq. (5.108)) that can be obtained in a relatively small range of overall DMF and decane compositions around the operation point given in the left side of Table 5.22. For the sake of completeness, both models are considered that show the best agreement with experimental data, the SRK-MHV2 (left) and the hs-PCP-SAFT EoS model (right). Both, with parameters found in this thesis.[39]

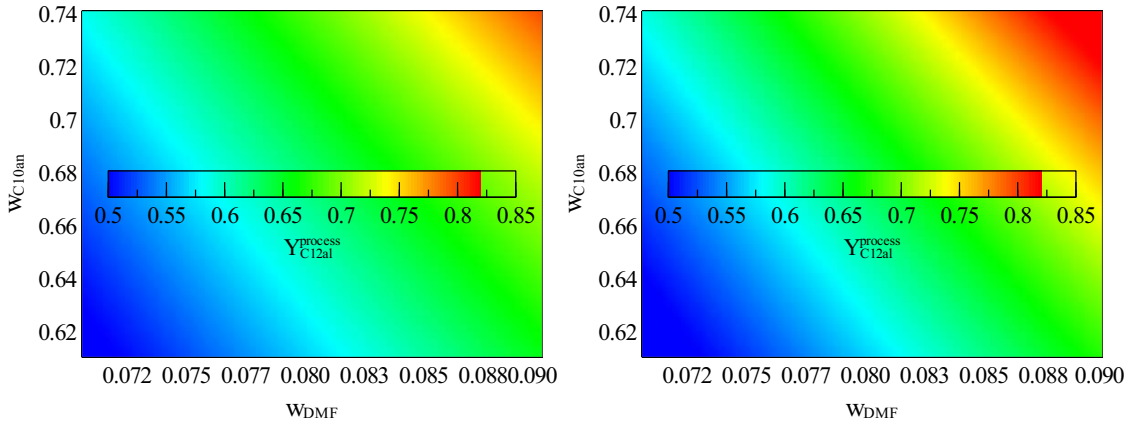

Figure 5.19: Process-wide aldehyde yield as a function of total feed composition: Left: using SRK-MHV2 EoS model. Right: using hs-PCP SAFT EoS model with parameters determined in this thesis

Besides making clear the relatively large quantitative influence of small changes in the feed compositions on the resulting overall process yield, further important facts follow from Figure 5.19. Particularly interesting is the fact that for the shown region, several different combinations of DMF/decane composition pairs can be set to reach one and the same process yield. This result can also be explained by Eq. (5.108), as there are several different combinations of compositions of dodecene and aldehyde that result in the same given value of yield.

The simplicity of Eq. (5.111) also makes clear that high values of the process wide yield

[39] Since compositions are directly considered as specifications there is no need to consider any differences in the density calculations.

are rather a natural consequence of low dodecene concentrations leaving the reactor, than a consequence of particular DMF/decane feed distributions. The theoretical best process yield of 1 only follows from the obvious case of a full conversion of the alkenes into aldehydes.

Chapter 6

Summary

The main goal of this thesis was the development and evaluation of methods designed to reduce the effort related to the computational implementation of molecularly based equation of state models. On the basis of the concept of modeling on the documentation level, the resulting model implementations fulfill to a high extent the request for standardized model implementations with extensive documentation as proposed by the Working Party on Thermodynamics and Transport Properties of the European Federation of Chemical Engineers [84], and hence should have a stimulating effect on the dissemination of new model developments and extensions of molecularly based equation of state models.

In a first step, it was studied how computational techniques for derivative generation can be used to reduce the implementation effort for molecularly based thermodynamic models and how the considered computational methods, forward of reverse mode of algorithmic differentiation on the one hand and complex step and finite differences approximation on the other hand, perform in comparison with the common practice of hand coded derivatives. On the basis of a sequential evaluation procedure of the Helmholtz free energy expressions of the heterosegmented PCP-SAFT EoS model, which was obtained automatically from the structural analysis of a documentation-based implementation, the aforementioned computational derivative evaluation methods were used to obtain the gradients of the Helmholtz free energy. While hand coding of derivatives is time-consuming, error-prone and commonly leads to large algebraic expressions, the integration of computational derivative evaluation methods only requires some additional lines of tool specific code.[1] Regarding the computational performance that follows from the consideration of different derivative evaluation methods,[2] the obtained results are in good accordance with the the-

[1] Note that the fact that tool specific knowledge may be required to evaluate derivatives does not conflict with the tool independency propagated by modeling on the documentation level, since the tool specific derivative evaluation code only takes place at the tool specific code generation stage.

[2] The performance of a particular derivative evaluation method as used in this thesis is given by the ratio

ory, according to which the suitability of a particular method depends on the number of independent and dependent variables of the evaluation procedure. In the considered case of gradient evaluation, as required for obtaining thermodynamic properties of mixtures, the reverse mode of algorithmic differentiation is clearly particularly suitable since the computational effort does not depend on the number of independent variables. For evaluations with a relatively low number of independent variables, for example for some property of a binary or a ternary mixture, all studied methods have acceptable performance. From this first type of evaluation, it clearly follows that the use of computational derivative evaluation techniques not only reduces the implementation effort but it can also lead to model implementations with overall lower computational effort, if the proper method for the derivative evaluation is selected.

Besides the study of computational performance for gradient evaluation with different derivative evaluation methods, algorithmic differentiation was used in an attempt to give an answer to a controversial question in Computational Thermodynamics. When it comes to the computational implementation of models for the Helmholtz free energy, well-known research groups do not agree on the proper selection of the independent variables. While some claim that temperature, extensive volume, and the vector of the mole numbers of all components $(T, V, \mathbf{n})$ is the most proper set of independent variables, their objectors affirm the same for their preferred set, which consists of temperature, intensive volume (or density) and the vector of molar fractions $(T, v \text{ or } \hat{\rho}, \mathbf{x})$. Since algorithmic differentiation intrinsically exploits common subexpressions, it appears to be a promising approach to clarify this issue. Considering of the SRK EoS model, the SRK-MHV2 EoS model with NRTL as underlying activity coefficient model, and the s-PC-SAFT EoS model, it was quantified how big the influence of the chosen set of independent variables on the computational effort is. For the comparison, evaluations of the fugacity coefficients of multicomponent mixtures were considered. The results, which were obtained using the reverse mode of algorithmic differentiation, were also compared against hand coded implementations. Though the obtained results do not provide a general answer valid for all models, they clearly indicate that the selection of a particular set of independent variables can have a relevant influence on the resulting computational effort. While in the case of the SRK EoS model, the differences between the different sets are not very high, and the hand coded implementations show a slightly better performance than the ones obtained with AD, in case of the SRK-MHV2 and the s-PC-SAFT EoS models the differences between the performances of the different sets are larger and better results are obtained for the AD implementations; in particular for those where the fugacity coefficients are evaluated considering $(T, V, \mathbf{n})$ as set of independent variables.

of the CPU-time required for the evaluation of the wanted derivative and the CPU-time for the evaluation of the function whose derivative is considered, see e. g Eq. (3.16) or Eq. (3.18).

Perhaps more interesting than the quantitative results regarding the selection of independent variables is the incidental finding, that it is not necessary to introduce algebraic manipulations to an available model formulation (by hand) in order to benefit from the advantages of a different set of independent variables. When applying structural analysis and reverse mode of algorithmic differentiation on documentation-based implementation of thermodynamic models, it suffices to add equations that define relations between the elements of the different sets of independent variables and to denote the independent variables with respect to which derivatives should be built. Independent of the position where the additional equations are added to the documentation-based formulation, the structural evaluation automatically reorders the single equations in such a way that the information flow between the new independents and the expressions for the Helmholtz free energy are properly set so as to get the required derivative information.

Though the evaluations concerning the more complex thermodynamic models performed better with the set $(T, V, \mathbf{n})$, the question regarding the best set of independent variables is not considered as clarified. In the AD community it is a well-accepted fact, that the quality of the evaluated derivatives is related to the quality of the implementations of the function whose derivative is built [71]. Since the quality of the underlying expressions defining the Helmholtz free energy depends on the code generation capabilities and the algebraic syntax supported by MOSAICmodeling, the resulting code cannot be used to obtain a general statement.

In addition to the previously discussed studies that quantified and evaluated the influence of single derivative evaluation methods and sets of independent variables on the evaluation of gradients, further studies were performed that put an emphasis on the implementation of the developed methods into practical applications while keeping the multi-tool aspect inherent to documentation-based models. As real application the focus is laid on the implementation and solution of GLE and LLE phase equilibrium calculations of mixtures that appear in the hydroformylation process introduced in Section 5.1. The multi-tool aspect is given by the fact that the generated models are available for different tools that support different implementation and solution strategies. Besides evaluating the feasibility of documentation-based models as the basis for real (complex) applications, the relatively large number of potential ways to generate code considering different types of derivative information opened the chance to evaluate single implementation types in more detail. Though all cases were solved with the solution approach commonly known as equation solving techniques [197],[3] they differ by the way how the Helmholtz free energy expressions and its derivatives are obtained and implemented. Following cases were considered:

[3]That means under solution of phase equilibrium equations with some general, mostly Newton based, solver for general algebraic equation system.

1. Implementation of thermodynamic model and its hand coded derivatives as sequential evaluation procedure gained from the documentation-based implementation.

2. Implementation of thermodynamic model as sequential evaluation procedure gained from the documentation-based implementation and derivatives of thermodynamic model from reverse mode of algorithmic differentiation.

3. Fully equation-oriented implementation of the thermodynamic model and its hand coded derivatives.

4. Fully equation-oriented implementation of the thermodynamic model and derivatives obtained from the solution of a linear equation system that follows from the application of the implicit function theorem.

For the sake of simplicity, in the following, the first two cases listed above are denoted as implementations including explicit evaluations, while the last two types are denoted as cases without explicit evaluations or simply as equation-oriented evaluations. While all model implementations successfully led to solutions with acceptable computational effort, hence, clearly demonstrating the suitability of documentation-based models as a basis for modeling applications with molecularly based EoS models, the comparison of the different implementation types provided numerous interesting facts indicating different strengths and shortcomings.

A first, rather trivial result follows from the multi-tool aspect which gives the possibility of evaluating the same theoretical fundamentals with different languages or tools. While the derivative information obtained with the source transformation tool CasADi leads to slightly lower computational efforts than the hand coded ones, the derivatives evaluated by the operator overloading tool ADOL-C showed a significantly lower performance. Though this first result does not lead to any general information, it clearly indicates that the availability and degree of development of potential AD tools can have a major influence on the efficiency of the resulting implementation. A further interesting comparison is related to the different number of equations and unknown variables that result from the different implementation types. Clearly, the implementations in which thermodynamic models and its derivatives are available as sequential evaluation procedures require the lowest computational effort to perform single converging Newton steps. Interestingly, the results also indicate that in some cases the implementations without explicit evaluations show a more robust convergence behavior. In those latter cases the convergence behavior was less sensitive to changes in design specifications leading to an overall faster generation of equilibrium diagrams over wide ranges of an independent variable. In some other cases, however, both types of model implementations showed an equally robust conver-

gence behavior.[4] In those latter cases the evaluation with explicit evaluations lead to a faster evaluation of the phase equilibria required for the generation of phase diagrams. Though a significant part of the appearing convergence issues could be possibly remedied by changes in the model formulation, the cases were designed to directly evaluate the possibility of using model formulations as available in the documentation. From the analysis of results obtained on the basis of equation-oriented model formulations, it additionally becomes clear that another type of derivative information becomes critical if certain problem size is exceeded. If no method or approach is available to evaluate the Jacobian of the overall equation system efficiently, then the evaluation of the Jacobian may become the computational bottleneck. This makes clear the need of tools that support the evaluation of higher order derivatives.

From the four different evaluated implementation types listed above, the second and the fourth are particularly interesting for the goals of this thesis, since both use automatically generated derivatives as part of the model. Particularly, the fourth approach fulfills the tool-independent request in the best possible way, since it is implementable in any arbitrary type of programming language or equation-oriented modeling environment. Depending on the used language or modeling environment used for the code generation, the solution of the linear equation system can result either from a corresponding call for linear systems or from the solver used for the whole equation system. Here it should be considered that modern solvers for nonlinear systems can automatically detect linear sub-systems and solve these separately. Though the solutions of the model formulation considering sequential explicit evaluations do not show in all cases the expected robustness, it should be considered that such formulations are not treated commonly with equation solving techniques. A different potential way of using such models without the drawbacks of general solution methods would be to exclusively generate expressions for the thermodynamic model and its derivatives and to implement these into available solution algorithms.

In addition to the previously discussed tasks, within this thesis a considerable effort was put into the thermodynamic consistent description of phase equilibria phenomena relevant to the hydroformylation process of 1-dodecene in a thermomorphic solvent system of dimethylformamide and decane. Within this context, a comparative evaluation of two thermodynamic models, the heterosegmented Perturbed Chain Polar SAFT EoS model (hs-PCP-SAFT) and the Soave-Redlich-Kwong EoS model with Huron-Vidal mixing rules of second order and the Non-random Two-Liquid local composition model as underlying activity coefficient model (SRK-MHV2-NRTL), was made. This included parameter estimation studies with the goal of correlating available multicomponent LLE and GLE data,

[4] Similar to above the robustness is quantified as low sensitiveness with respect to changes in some design variable.

the quantification of the influence of the calculated LLE on the simulated overall process behavior and the validation of the simulated overall process behavior against available experimental data of a miniplant with full recycle.

Since the quality of the description of the LLE of the ternary system DMF-decane-*n*-tridecanal with the hs-PCP-SAFT EoS model as proposed in the literature [181] is far from satisfactory, in this work a new and better parameterization was found under a stronger exploitation of the available model structure. NRTL binary interaction parameters for the use in the SRK-MHV2 EoS model were also obtained on the basis of available LLE and GLE data. With the obtained parameter sets both considered models provide a satisfactory temperature dependent description of phase equilibria in the studied operation ranges. In general, results gained with the SRK-MHV2-NRTL EoS model provides a better accordance with the available LLE data, while the calculation of GLE with PCP-SAFT have shown to be so good,[5] that calculated solubilities obtained using PCP-SAFT were taken as a basis for the parameter estimation with the SRK-MHV2-NRTL model. Though binary interaction parameters could be found that led to a satisfactory description of the gas solubility of each single gas-liquid pair over the interesting ranges of pressure and temperature, in some cases the quantified differences became more significant when considering multicomponent mixtures of several liquids and gases.

Taking into consideration several extensively discussed assumptions, a simplified process wide model with rigorous phase equilibrium calculations was developed in order to quantify the influence of the quality of the LLE calculations on the calculated overall process performance and in order to allow a model validation against experimental data. The results of the evaluation of the simplified process wide model with different thermodynamic models and parameterizations clearly indicate the large influence of the selected thermodynamic model. The previously available best description of the LLE with the hs-PCP-SAFT EoS model [181] led to similar results as obtained with UNIFAC-Dortmund and to overall large deviations compared to the experimental results. In contrast, results obtained with the parameterizations determined in this thesis for the hs-PCP-SAFT and the SRK-MHV2-NRTL EoS models showed similar quality and a good accordance to the available experimental data. Though both approaches show satisfactory results regarding the description of phase equilibrium phenomena, a more detailed analysis of the thermodynamic models shows a clear shortcoming of the SRK-MHV2-NRTL EoS model when used for density calculations. If, however, no calculation of volumetric or other derived properties is required, as in the model considered in this thesis, the SRK-MHV2-NRTL clearly demonstrate the high potential available in using well developed thermodynamic concepts, such as EoS/g^E models.

[5] Note that the heterosegmented aspect was not considered in [213, 214].

The improved parameterizations of the hs-PCP-SAFT EoS model is a successful example of the methods developed in this thesis. Independent of the fact that the developed methods significantly eased the implementation process, an important contribution resulted from the mere possibility to use a model that so far was only available as a publication and that has only been evaluated by the original model developers. The critical evaluation of published results and their apparent shortcomings led to the consideration of model structures that eventually led to a significantly better process description. The willingness of making model developments easily accessible to potential users is perhaps one of the most decisive factors towards the successful dissemination of new models and model developments.

Chapter 7

Outlook to Future Research Directions and Recommendations for Further Software Developments

The results shown in this thesis point out the potentials and advantages of integrating advanced computational methods for derivative evaluation within the implementation of advanced molecularly based EoS models. Through the chosen approach of integrating these principles into the concept of modeling on the documentation level, the resulting methods not only lead to a reduction of the implementation effort, but also provide means that can have an important influence on enhancing the dissemination of new model developments and model modifications.

While in this work the main focus has been put on evaluating the feasibility of the aforementioned concepts and their application in rather simple model-based applications, mostly limited to flash calculations, the results and the wide range of types of derivative information required by more advanced model-based methods indicate that further research is required in order to extend the field of application of the concepts developed in this thesis. In the following, the recommendations regarding future research work and future software implementations are summarized in two Sections. In Section 7.1 open questions and possible potentials regarding the extension of the methods towards the use of higher order derivatives are discussed. Section 7.2 focuses on potential developments in MOSAICmodeling that could lead to a significantly better model dissemination.

7.1 Developments Towards Integration of Higher Order Derivatives and More Advanced Model-Based Methods

Technically speaking, the methods developed and evaluated in this thesis with the goal of reducing the implementation effort of complex molecularly based EoS models, are concerned with the automatic generation of derivatives of first order on the basis of an equation system that defines the reduced, residual Helmholtz energy. Due to the nature of EoS models available as expressions of the Helmholtz energy, first order derivatives are sufficient to evaluate any other mixture property such as compressibility factor, fugacity coefficients or enthalpy, and hence sufficient for the formulation of mathematical models required for model-based applications such as equilibrium calculations. Exact higher order derivatives of the EoS model, as they could be used to build the Jacobian of the equation system defining the equilibrium calculations, are currently not provided by the methods developed in this thesis.[1]

As pointed out in Griewank [70] regarding the solution of general algebraic equation systems, the quality of the derivatives required by solution algorithms may affect the rate of convergence, but not the solution accuracy itself. For this latter point the accuracy of the residuals is decisive. This assertion points out that the methods developed and evaluated in this work fulfill requirements of process simulation tasks. However, care should be taken when using more advanced model-based methods which rely on the solution of optimization problems. The solution of optimization problems is equivalent to the solution of algebraic equation systems which follow from optimality conditions that must be fulfilled. Since the algebraic equation systems that build the optimality conditions partially consist of higher order derivatives, the quality of these derivative evaluations is decisive for the reliable solution of the underlying optimization problems.

Figure 7.1 indicates schematically the current possibilities of the developed approach and shows feasible applications that could result from considering exact higher order derivatives. For a given equilibrium calculation model characterized by the vector function $\vec{g}$, Figure 7.1 points out which derivatives are required for the solution of tasks related to parameter identification and optimal experimental design. Note that in case of optimal experimental design the objective function is a function of parameter sensitivities, whose evaluation requires derivatives of $\vec{g}$ with respect to the state variables and parameters [86].[2]

[1] However, it should be noted that the equation systems generated and solved using solvers provided with CasADi build exact Jacobians based on algorithmic differentiation, hence taking into consideration of second-order derivatives of the Helmholtz energy. As confirmed by the results from Section 5.3, the availability of these second order derivatives have a significant computational effect at the solver level.

[2] Note that, for the sake of a better readability, the derivatives with respect to state variables, $\frac{\partial \vec{g}}{\partial \vec{z}}$, where

The link to experimental data shown as well in Figure 7.1 points out the possibility of automatizing phase equilibria measurements and its correlation as discussed by Dechambre et al. [41] for popular local composition models. The multiple models listed on the right side of Figure 7.1 indicates the possibility of evaluating several different concurring models, for which only the expressions for the free Helmholtz energy need to be implemented.

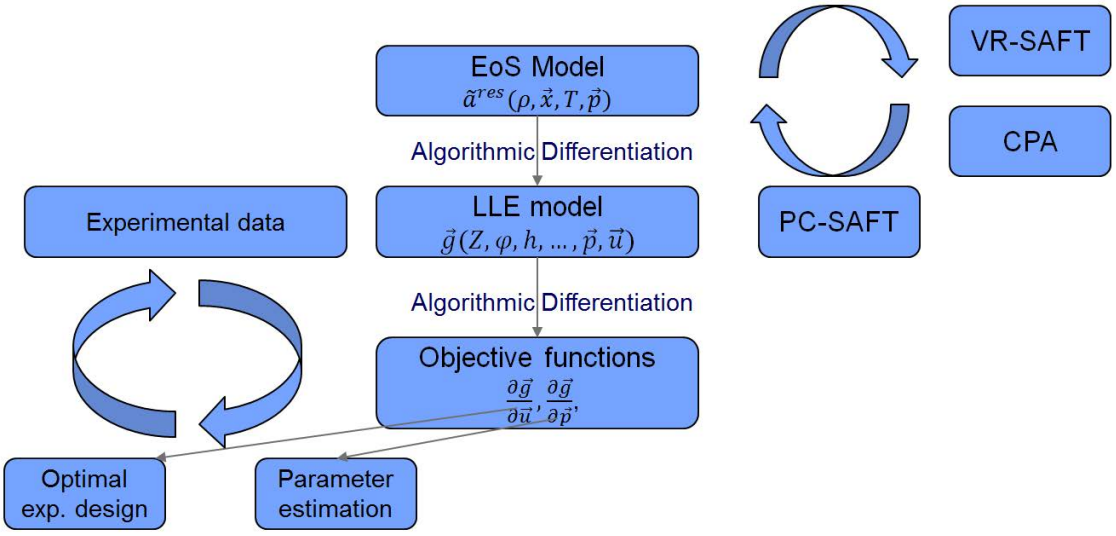

Figure 7.1: Potential applications resulting from automatic generation/evaluation of derivatives of higher order

From a scientific perspective, particularly interesting open questions are related to the use and implementation of structural analysis in order to automatically identify or propose proper combination of derivative evaluation methods so as to efficiently evaluate higher order derivatives. Here it should be considered that many well developed AD tools only support the evaluation of first order derivatives. A further challenging detail is related to the differences between implementations conceived as fully equation-oriented and those converted in sequential evaluation procedures. Equally interesting is the consideration of advances in algorithmic differentiation in an equation-oriented modeling paradigm, so as to work internally with completely tool-independent, yet efficient formulations.

The automatic formulation of advanced model-based methods linked with the possibility of implementing several well documented models with low effort, forms the basis of a potential tool to evaluate the correlative capabilities of concurring thermodynamic models. In conjunction with a standardized set of data, against which all models may be tested, such a tool could have a significant influence on the dissemination of molecularly based EoS models. As pointed out in [84], "'the validation against agreed industry benchmarks could lead to the reduction of cycle time from development to industrial use"' of the model.

$\vec{z} = [Z, \varphi, h, \ldots,]$, are not included in Figure 7.1.

Although the automatic generation of advanced model-based problem formulations, as it can be obtained through the support of higher order derivatives, could significantly reduce the effort related to the implementation of simple and advanced model-based methods, it should be considered that the solution of the aforementioned problems remains a challenging task, see Section 2.4.3. As a result, further efforts should be put on the evaluation and dissemination of algorithms specialized in the solution of this type of problems as well.

7.2 Recommendations on Further Developments in MOSAICmodeling

The application of advanced computational methods and support for proper workflows requires the availability of proper software tools. On the basis of the gained experience, in the author's view, the implementation of following recommendations could significantly improve the possibilities available by the current documentation-based models as supported by MOSAICmodeling.

7.2.1 Recommendations related to automatic code generation

The equilibrium calculations presented in this work show the feasibility of using general equation-solving techniques based on documentation-like, fully equation-oriented model implementations.[3] Though not separately discussed in detail in this work, the solution of the equilibrium calculations was highly challenging. In particular, it was mandatory to consider initial guesses very close to the actual solution in order to obtain convergence for the LLE calculations. This is in accordance with the well-known finding that generic programs for numerical methods are ineffective for phase equilibrium calculations [197]. Additionally, it should be considered that in this work calculations were considered for mixtures, for which experimental phase equilibria are available. Hence, in most cases it was not necessary to apply stability analysis in order to know the number of phases in equilibrium in advance.

A major improvement in the dissemination of advanced thermodynamic models could be given by the possibility of generating and providing code embedding those models within tailored algorithms for phase equilibrium calculations. Of particular high interest are methods capable of finding the right solution despite not knowing the number of phases in equilibrium a priori.

[3] Though in the understanding of most authors, the equation-solving techniques refer to the simultaneous solution of the equation system of an equilibrium stage where the thermodynamic variables are available as function calls, in this contribution equation-solving techniques also refer to the cases in which the equations defining the thermodynamic variables are simultaneously solved with the equations system of the equilibrium stage.

On a basic level, the approaches by Kraus [111] represent small steps into the direction of providing solution algorithms: the block decomposition identifies subsystems of equations that can be solved independently/ sequentially. The common strategy of decoupling the solution of the root density problem from the remaining phase equilibrium conditions follows from the bordered block diagonal decomposition and was implemented by code generation for MATLAB [111]. A further development could be given by directly providing the possibility of manually choosing the solution sequence of single equations and equation systems. Own experience with the direct substitution approach [144] clearly pointed out the big potential available in simple algorithmic evaluations in comparison with generic solution approaches. Initialization of the pT-Flash calculations with direct substitution led in most cases to proper convergence despite of using initial guesses relatively far from the solution.[4] While at first sight such an approach of providing code with correspondent solution algorithms only appears to be suitable for code generation for generic programming languages such as C/C++, Fortran or MATLAB, it should be remarked that purely equation-oriented modeling environments, such as gPROMS or Aspen Custom Modeler offer tool specific workflows for the implementation of algorithmic procedures for initialization purposes. Alternatively, it is possible to divide the large equation system into subsystems which are solved sequentially according to user defined tasks. Though the implementation of such code generation may be a large amount of work and requires a detailed knowledge of advanced tool specific features, the potential benefits are beyond doubt.

Regarding the consideration of cases where the number of phases is not known a priori, and leaving aside approaches which directly rely on the minimization of the Gibbs energy, several different equation-oriented approaches can come into consideration. An interesting approach proposed by Alsaifi and Englezos [6] relies on the possibility of formulating and solving nonlinear equation systems including inequality constraints. This is fully equivalent to the formulation of optimization problems with complementarity constraints as discussed in [16]. According to this, at a simple level, it would suffice to support inequality constraints as equation system elements and to include code generation for a solver supporting those inequalities. Interestingly, despite the equation-oriented nature of the resulting formulation, the solution approach proposed by Alsaifi and Englezos [6] has an algorithmic nature that considers inner and outer loops. An alternative to the consideration of inequality constraints or complementarity constraints is given by considering nonsmooth formulation for modeling the appearance and disappearance of phases [178]. In order for this concept to work, the modeling or programming languages must support the use of nonsmooth operators, such as min() or max(). Hence, taking into considera-

[4] This also was the reason why solutions of the phase equilibrium calculations within parameter estimation studies were initialized with direct substitution before switching to generic solution approaches.

tion proper model formulations, it may be enough to permit nonsmooth operators in the default syntax used for model implementation. In general, this seems to be a reasonable type of method for modeling process with discontinuities. However, it should be noted that finding a first feasible solutions still remains a challenge.

There are further promising approaches that could be implemented via code generation to better solve the most common challenges that appear when using advanced thermodynamic models. Among those, interval analysis seems particularly promising, see also Section 2.4.3. In spite of given highlights of specific approaches, a proper, mostly algorithmic initialization seems in most cases indispensable so that the challenging task of providing such algorithmic possibilities appears to be necessary.

A further possible way of using the advantages of code generation, while not introducing advanced algorithmic manipulations, could be given if only considering mathematical model equations defining the free Helmholtz energy. Since all thermodynamic properties follow from derivatives of the Helmholtz free energy, it is thinkable to generate thermodynamic packages out of the equation-oriented formulation. An automatic generation of a closed-form, CAPE-OPEN compliant thermodynamic model would be a major step toward the dissemination and reuse of new model improvements and developments.

7.2.2 Further recommendation and topics

In addition to changes that can be seen as improvements of the code generation capabilities, there are various modifications that could importantly increase the suitability of MOSAICmodeling as a tool for a better dissemination of advanced thermodynamic models. These potential modifications are explained below.

Algebraic input supported by MOSAICmodeling and notational issues

A series of potential modifications are related to the algebraic input. While with the current implementation in MOSAICmodeling a high similarity with published model documentations is given, as already shown by the example in Appendix A.2, in some cases, the instantiated or flattened equation systems can be unnecessarily large or have an unnecessary large number of variables. In this context, it should be remarked that such inefficient implementations are not only a consequence of limitations in the algebraic input supported by MOSAICmodeling, but result from the fact that the notations used in published model documentations are very often inaccurate and even ambiguous. Additionally, in many cases additional knowledge or even assumptions are required in order to implement the models properly.

A good example of possible improvements in the used notations and MOSAICmodeling's features, concerns the way indexes are handled. In the current implementation, indices are assumed to cover a continuous range of integers starting at one. Besides that, they are only thought to denote variables names and their values cannot be used directly in mathematical operations. A better support could be reached if indices were implemented or treated as general elements of sets and if set operators, such as union ($\cup$) or intersection ($\cap$), were supported as well.[5] In the case that elements of sets represent integers, it would also be helpful to directly use the values of the indices directly for mathematical operations.

Independent of modifications to enhance the mathematical input supported by MOSAICmodeling, the example given in A.2 and the author's own experience underlay the fact that the success of model implementations based on documentation strongly depends on the exactness and quality of the used mathematical notation. Giving model developers the possibility of simultaneously implementing and documenting the model, should help model developers to assimilate the importance of an unambiguous mathematical notation. A model implementation with an unambiguous mathematical notation not only facilitates the transfer into a practical implementation but builds the basis for the realization of an automatic code generation into arbitrary modeling or programming languages.

Regarding the evaluation efficiency of the generated code, for some modeling languages it would be advantageous to offer the possibility of supporting code generation considering vector-valued and multidimensional variables. Systems with a very large number of scalar variables and equations are in some cases too large to be read by some software tools properly.

More advanced features: MOSAICmodeling as dedicated online tool with better Coupling of sequential and simultaneous solution approaches

The potential modifications discussed above have a very broad range of influence on the types of problems and tools supported by MOSAICmodeling. Improvements on the permitted mathematical notation and algebraic input provide advantages to all types of MOSAICmodeling models and applications. Code generation implementing some type of solution algorithm, as shortly suggested in Section 7.2.1, could be of general use, but a large effort may be required to get it to work for modeling environments relying on the equation-oriented solution approach. On the other hand, single solution approaches designed as problem-tailored code generation, though potentially very valuable, have a

[5] In fact, the readability of simple models well supported by MOSAICmodeling could be improved if indices represent general elements of a set and not only integers. For example, the readability of a component index i is better if it directly refers to the names of the components, i = {H_2O, CH_4, CO}, instead of referring to component numbers i = {1, 2, 3 }.

relatively narrow field of application, limited to few particular modeling tools or programming languages. The automatic generation of code including efficiently evaluated higher order derivatives as discussed in Section 7.1 can be considered to belong to this latter category since its use is currently limited to the few languages for which higher order derivatives can be obtained via algorithmic differentiation.

A possible approach to affront the challenges related to tool dependency could be given by further developing MOSAICmodeling as a dedicated tool to handle advanced modeling tasks with advanced thermodynamic models. Instead of providing code generation for selected approaches and selected tools, it seems reasonable to make such rather specific problem solution approaches available online, as it is currently made with the solution of general nonlinear equation systems and semi-explicit DAEs, which can be solved online with the BzzMath library.

Though such kind of approach seems to be against one of the main advantages of MOSAICmodeling, the possibility of working with arbitrary programming or modeling languages, the advantages are also significant. Among others, there is no need for the user to have a particular tool or modeling environment, since everything could be accessed online. Additionally, by offering a limited amount of software tools in conjunction with high level programming languages and compiled code, it is possible to work with highly efficient computational implementations.

The advantages of working offline with compiled code become particularly clear when evaluation derivatives with algorithmic differentiation. Since its efficiency relies on the exploitation of common subexpressions, it profits from the internal representation of the expressions. Andersson [7] demonstrates this clearly with a simple example comparing the internal tree representation of one and the same algebraic expression in MATLAB against the one resulting from CasADi. The efficiency of the considered code and its derivatives not only depends on the resulting algebraic expressions itself but on how these are represented and interpreted internally. This is one of the reasons why it did not seem necessary to support AD techniques on the MOSAICmodeling-MathML level. In spite of generating an efficient set of algebraic equations with some potential AD for equation-oriented systems, at the end the internal representation at the interpreter level is decisive.

A further advantage of keeping all features within one dedicated online tool could be given by the possibility of making accessible sequential (algorithmic) and equation-oriented solution strategies. The possibility of using the results of a previous calculation as initial guess for a further one is currently supported and could be significantly enhanced in conjunction with solvers exploiting the direct evaluation of explicit expressions or some user defined sequential evaluations procedure. Closing the gap between sequential modular and equation-oriented solution approaches would mean a major step towards a wider

dissemination of models formulated on the documentation level.

Appendix A

Implementation Details and Considered Models

A.1 Implementation Details of the Heterosegmented PCP-SAFT EoS Model in MOSAICmodeling

It should be noted that MOSAICmodeling permits a highly readable model implementation close to the original documentation.[1] For example for the following equation published in the original documentation[72, 74] as given by Eq. (A.1).

$$I_1(\eta, \bar{m}) = \sum_{i=0}^{6} a_i(\bar{m})\, \eta^i, \tag{A.1}$$

where I_1 denotes an approximation of an integral from the perturbation theory [74], the corresponding MOSAICmodeling implementation is coded as

$$I1_k = \sum_{p=1}^{Np} a_{k,p-1}^{basic} \cdot (\eta_k)^{n_{p-1}}, \tag{A.2}$$

using following MosaicLatex syntax

```
I1_{k}=\sum_{p=1}^{N_{p}}{a^{basic}_{k,p-1}\cdot(\eta_{k})^{n_{p-1}}}
```

and hence a total of 69 characters. In spite of the obvious similarity of the expressions Eq. (A.1) and Eq. (A.2) the available differences should be discussed in order to stress some differences that can appear between general model documentation and the input supported

[1] Since Eqs. (A.1) and (A.2) are shown in this context as a purely mathematical example, the appearing variable names are not considered in the nomenclature of this thesis.

by MOSAICmodeling. Though some of the differences are due to limited algebraic support offered by MOSAICmodeling, some other are due to ambiguity of notations commonly used in model documentation. In the aforementioned example the main differences are:

1. In the original documentation (Eq. (A.1)) the terms in parentheses $(\eta, \bar{m})$ and $\bar{m}$ are introduced to make clear the dependency on other variables. However, this is not necessary since the dependency is evident from the expressions defining I_1 and a_i itself. Note that the expressions for are a_i not shown here.

2. The expressions of Eq. (A.2) contain an index k which does not appear in the original version. This is due to the fact that the MOSAICmodeling implementation refers to a value of the phase denoted by the index k. Though not available in the documentation an attentive modeler should easily recognize the necessity of the additional index for a practical model implementation.

3. In Eq. (A.2) the variable a appears with the superscript "$basic$". This superscript was additionally introduced in order to distinguish this a from another a appearing in a similar expression due to the polar contribution. This is an example of adaption of notation due to combination of different documentation sources.

4. Explicitly appearance of multiplication ($\cdot \iff$ `\cdot`) in Eq. (A.2): in MOSAICmodeling it is mandatory to write appearance of multiplication explicitly in order to avoid ambiguity. Someone non-familiar with the PC-SAFT EoS model could interpret the expression $a_i(\bar{m})\eta^i$ provided in the original documentation erroneously, e. g. as a product of a_i, $\bar{m}$ and η^i.

5. Both equations use different names for the indices and different bounds of the sum expression: this important difference is indirectly due to the fact that in MOSAICmodeling the names of indices become part of the variable name and that there are no fully numerical indices. For example, in the context of MOSAICmodeling $\sum_{i=1}^{3} a_i$ is a different expression than $\sum_{j=1}^{3} a_j$. In the above case (Eq. (A.1) and Eq. (A.2)) the index p was taken for Eq. (A.2) instead of i since the index i was used elsewhere to denote the component index. Additionally, while the summation index p from Eq. (A.2) could start counting with 0 instead of 1, it was decided to start with 1 since the generic equation that defines $a_{k,p-1}^{basic}$ (not shown here) needs an upper value N_p of 7 to automatically generate $a_{k,p=0}^{basic}$ up to $a_{k,p=6}^{basic}$ in the instantiation/flattening step.[2] Besides that, MOSAICmodeling's rules for the definition of sums do not allow

[2]The actual limitation is that in the flattening/instantiation step the index sets always starts with the value of 1. Because of that it is necessary to write the index expression $p-1$, so that $p=0$ is generated first.

to write an upper limit of the sum as a number. This is separately defined when the evaluation, that is the particular simulation case, is created.

6. Differences in the syntax used to write the power expressions: the parentheses around the basis that appears in Eq. (A.2) result from MOSAICmodeling's conventions to write unambiguous mathematical expressions introduced in [115]. The goal of the parentheses is to distinguish exponents from superscripts, otherwise it is not entirely clear what is meant. The difference in the exponent expression, using n_{p-1} instead of i, is given by the fact that operations on index variables are only allowed at the index level. They cannot be used as common variables. As an example, if $(\eta_k)^{p-1}$ is written, MOSAICmodeling interprets p as a further base name (as long as it is defined in the nomenclature) and not as the index variable name. In order to simulate the behavior given by the original model equation (A.1) p is used as an index to enumerate different exponents $n_{p=0}$, $n_{p=1}$ up to $n_{p=6}$. To obtain the desired behavior for the generated code, it is necessary to fix the values 1, 2 up to 6 as corresponding design specifications.

The effort to code the model into MOSAICmodeling is almost fully equivalent to the effort of creating a model documentation, including the definition of a proper notation as part of the effort. The resulting total number of equations is based on the original model documentation [72, 182, 208], where some variables have been artificially introduced to make the model more readable. Regarding the author's experience with the implementation of advanced EoS models, there is some level of complexity, for example complex mathematical expressions with up to 6 embedded sums and variables names with up to 8 different indices for which the equation-oriented formulation not necessarily enhances the readability of the model.

A.2 Example Showing Necessity of More Flexible Mathematical Notation

As an example, consider a common issue that appears in the heterosegmented PCP-SAFT EoS model, multiple nested summations in which the upper bound of some inner summation depends on the current value of the index of an outer summation. Consider an excerpt of an expression of the polar term as published in [27]:[3]

$$\sum_i \sum_j x_i x_j \cdot \frac{\mu_i^2 \mu_j^2}{m_i m_j} \sum_\alpha \sum_\beta \frac{J_{2,i\alpha j\beta}^{DD}}{\sigma_{i\alpha j\beta}^3 z_{i,\alpha} z_{j\beta}} \tag{A.3}$$

[3]Note that this published equation was found to be wrong, since the polarity should not only be a property of the component, but a property of a segment of a component.

Using the notation and algebraic input currently supported by MOSAICmodeling, Eq. (A.3) can be implemented as shown in Eq. (A.4). Note that NC denotes the total number of components and NS the total number of segments. While it is rather evident that the total number of segments should be a property of the considered component, it can be said with certainty that this information is not explicitly given in the model documentation, neither in the original formulation (Eq. (A.3)) [181] nor in the corresponding MOSAICmodeling implementation (Eq. (A.4)).

$$\sum_{i=1}^{NC}\sum_{j=1}^{NC} x_i \cdot x_{i=j} \frac{(\mu_i)^2 \cdot (\mu_{i=j})^2}{m_i \cdot m_{i=j}} \cdot \sum_{\alpha=1}^{NS}\sum_{\beta=1}^{NS} \frac{J2_{i,\alpha,j,\beta}^{DD}}{(\sigma_{i,\alpha,j,\beta})^3 \cdot z_{i,\alpha} \cdot z_{i=j,\alpha=\beta}} \tag{A.4}$$

Additionally, neither from Eq. (A.3) nor from Eq. (A.4) it becomes clear that there is a relation between the indices i and α on the one hand and between the indices j and β on the other hand. In fact, the missing upper bounds in the summations in Eq. (A.3) could lead to the wrong assumption that the summations cover all elements of the sets of components (i and j) and segments (α and β). In Eq. (A.4) the total number of segments NS is the same for all components $i = 1$ to NC. In case of multicomponent systems with only one or few heterosegmented components, as the aldehyde containing systems considered in Chapter 5, this would lead to unnecessarily large expressions. A possible solution to this issue could be given by allowing expressions as given in Eq. (A.5). Hence, it would suffice to allow indexed upper bounds of summations.

$$\sum_{i=1}^{NC}\sum_{j=1}^{NC} x_i \cdot x_{i=j} \frac{(\mu_i)^2 \cdot (\mu_{i=j})^2}{m_i \cdot m_{i=j}} \cdot \sum_{\alpha=1}^{NS_i}\sum_{\beta=1}^{NS_j} \frac{J2_{i,\alpha,j,\beta}^{DD}}{(\sigma_{i,\alpha,j,\beta})^3 \cdot z_{i,\alpha} \cdot z_{i=j,\alpha=\beta}} \tag{A.5}$$

A further possible modification shows the potential of introducing sets for the definition of indices. A short analysis of the expressions of Eq. (A.3) clearly shows that a relevant contribution, with a value different than zero, only follows if quantifying a pair of polar components (with μ_i and $\mu_j \neq 0$). Accordingly, it is not necessary to evaluate Eq. (A.3) in all cases in which at least a nonpolar component appears. The same argumentation is valid for the expressions that define $J_{2,i\alpha j\beta}^{DD}$ (see Eq. (A.6)), since their only purpose is to be evaluated within Eq. (A.3). A more proper and efficient formulation of Eq. (A.3) and Eq. (A.6) could be reached if the component indices i and j are only expanded for polar components. Such a grouping of indices could efficiently be realized using sets and set operations to characterize polar and nonpolar compounds.

$$J_{2,i\alpha j\beta}^{DD} = \sum_{n=o}^{4} \left(a_{n,i\alpha j\beta} + b_{n,i\alpha j\beta} \frac{\varepsilon_{i\alpha j\beta}}{k\,T} \right) \eta^n \tag{A.6}$$

Component number	Component name
1	dodecene
2	tridecanal
3	DMF
4	decane

A.3 Simplified Plant Wide Model Including Degree of Freedom Analysis

As mentioned in Section 5.5.4 to 5.5.6, a simplified model is used in order to run the simulation studies. Considering the assumptions listed in Section 5.5.2 the process flowsheet shown in Figure 5.1 can be further simplified to the one shown in Figure A.1.

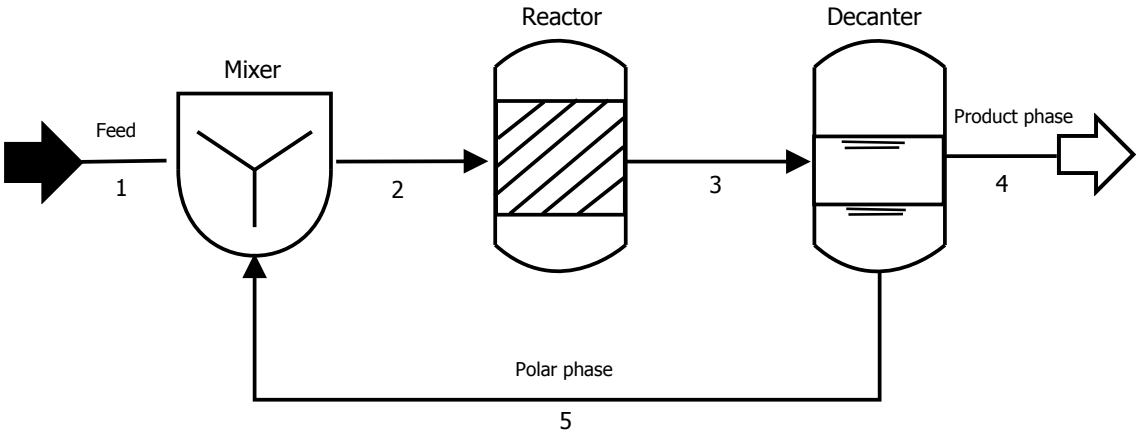

Figure A.1: Simplified flowsheet of simplified model

In the following, the considered model equations of the simplified plant wide model are listed. The stream numbers are according to the numbers in Figure A.1 and denoted with the index s. The index c denotes the component number in accordance with Table A.3.

A.3.1 Model equations

Model equations of mixer

$$0 = F_{s=1} \cdot x_{s=1,c} + F_{s=5} \cdot x_{s=5,c} - F_{s=2} \cdot x_{s=2,c}\,, \; c = 1,2,3,4 \tag{1-4}$$

$$0 = 1 - \left(\sum_{c=1}^{4} x_{s,c} \right) \,, s = 1,2 \tag{5-6}$$

Model equations of reactor

$$0 = F_{s=2} \cdot x_{s=2,c=1} - F_{s=3} \cdot x_{s=3,c=1} - r \tag{7}$$

$$0 = F_{s=2} \cdot x_{s=2,c=2} - F_{s=3} \cdot x_{s=3,c=2} + r \tag{8}$$

$$0 = F_{s=2} \cdot x_{s=2,c} - F_{s=3} \cdot x_{s=3,c}\,, c = 3,4 \tag{9-10}$$

$$0 = 1 - \left(\sum_{c=1}^{4} x_{s=3,c} \right) \tag{11}$$

Model equations of decanter

$$0 = F_{s=3} \cdot x_{s=3,c} - F_{s=4} \cdot x_{s=4,c} - F_{s=5} \cdot x_{s=5,c}\,,\ c = 1,2,3,4 \tag{12-15}$$

$$x_{s=4,c} \cdot \varphi_{s=4,c}(T_D, p_D, x_{s=4,c}) = x_{s=5,c} \cdot \varphi_{s=5,c}(T_D, p_D, x_{s=5,c})\,,\ c = 1,2,3,4 \tag{16-19}$$

$$0 = 1 - \left(\sum_{c=1}^{4} x_{s,c} \right)\,, s = 4,5 \tag{20-21}$$

A.3.2 Degree of freedom analysis

The degree of freedom results from the difference between the numbers of variables and equations. For the model equations shown above, the number of variables N_V results from Eq. (A.7)

$$N_V = \underbrace{N_S \cdot (N_C + 1)}_{F_s,\, x_{s,c}\,,\text{for } s=1\ldots5\,,c=1\ldots4} + \underbrace{3}_{T_D, p_D, r} = 28, \tag{A.7}$$

where N_S denotes the numbers of streams and N_C the number of components. The equality of pressure and temperature of the streams 4 and 5 is implicitly given by the fact that the corresponding fugacity coefficients are evaluated with the same pressure p_D and temperature T_D. Further, the fugacity coefficients $\varphi_{s,c}$ are not counted as variables. They are considered to be evaluated by some kind of external function call.

Considering the 28 variables and the 21 equations listed above, for the simplified plant wide model a degree of freedom of 7 results. Note however that all simulation studies considered in the Sections 5.5.4 to 5.5.6 have additional conditions that further reduce the degree of freedom:

- In all cases, it is considered that the temperature T_D and pressure at the decanter outlet p_D are known.
- Further, there is no aldehyde in the feed flow, stream 1, hence $x_{s=1,c=2} = 0$.

Hence, for the applications of the plant wide model a degree of freedom considered in this work a degree of freedom of 4 results.

A.4 EoS Models Implemented in MOSAICmodeling

In the following, a short overview of the EoS models used in this thesis is given. All equations, equation systems and further MOSAICmodeling elements can be found in the folder `/User/Merchan/Dissertation` of the MOSAICmodeling database of Berlin.

For the sake of simplicity, only information regarding equation systems or evaluations is considered. Besides a short description, only the element identification numbers (Eqsys ID or Evaluation ID) are given, since these give the most direct access to the model without the need to search within the folder structure. It does not appear necessary to provide additional details on single model elements, since this information can be easily obtained from the equation system itself.

In Appendix A.4.1 to A.4.3 the equations systems are listed that were used to study the influence of the set of independent variables in Section 5.2.2. In Appendix A.4.4 evaluations are listed that were used for the evaluation of different methods to evaluate gradients, or to in order to show the feasibility of solving phase equilibrium calculations on the basis of equation-oriented model formulations obtained from the documentation level.

A.4.1 Soave-Redlich-Kwong EoS model

- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables (Eqsys ID: 67177).
- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables and analytic expressions for compressibility factor and fugacity coefficients (Eqsys ID: 67187).
- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables obtained through introduction of additional equations to a system originally formulated with $(T, V, \mathbf{n})$ (Eqsys ID: 68466).

- Reduced residual Helmholtz energy according to Michelsen with $(T, V, \mathbf{n})$ as set of independent variables (Eqsys ID: 67181).
- Reduced residual Helmholtz energy according to Michelsen with $(T, V, \mathbf{n})$ as set of independent variables and analytic Expressions for compressibility factor and fugacity coefficients (Eqsys ID: 70759).
- Reduced residual Helmholtz energy according to Michelsen with $(T, V, \mathbf{n})$ as set of independent variables obtained through introduction of additional equations to a system originally formulated with $(T, v, \mathbf{x})$ (Eqsys ID: 67366).

A.4.2 Soave-Redlich-Kwong EoS model with Huron-Vidal mixing rules of second order

- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables (Eqsys ID: 54058).
- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables and analytic expressions for compressibility factor and fugacity coefficients (Eqsys ID: 71338).
- Reduced residual Helmholtz energy according to Michelsen with $(T, V, \mathbf{n})$ as set of independent variables obtained through introduction of additional equations to a system originally formulated with $(T, v, \mathbf{x})$ (Eqsys ID: 72431).

A.4.3 Simplified PC-SAFT EoS model

- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables (Eqsys ID: 66757).
- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables and analytic expressions for compressibility factor and fugacity coefficients (Eqsys ID: 72866).
- Dimensionless reduced residual Helmholtz energy with $(T, v, \mathbf{x})$ as set of independent variables obtained through introduction of additional equations to a system originally formulated with $(T, V, \mathbf{n})$ (Eqsys ID: 73124).
- Reduced residual Helmholtz energy according to Michelsen with $(T, V, \mathbf{n})$ as set of independent variables (Eqsys ID: 73158).
- Reduced residual Helmholtz energy according to Michelsen with $(T, V, \mathbf{n})$ as set of independent variables obtained through introduction of additional equations to a system originally formulated with $(T, v, \mathbf{x})$ (Eqsys ID: 73040).

A.4.4 Heterosegmented PCP-SAFT EoS model

- Evaluation of dimensionless reduced residual Helmholtz energy with 10 components with parameters of pure 1-dodecene (Evaluation ID: 68114).
- Phase equilibrium calculation for n-dodecanal at 450 K (Evaluation ID: 62384)
- pT-flash calculation for binary system consisting of 1-dodecene and DMF (Evaluation ID: 62399).
- pT-flash calculation for ternary system consisting of 1-dodecene, DMF and decane (Evaluation ID: 65591).
- pT-flash calculation for multicomponent system consisting of 1-dodecene, homosegmented n-dodecanal, DMF, decane, carbon monoxide and hydrogen (Evaluation ID: 65593).

Appendix B

Remarks on Calculations

B.1 Parameter Estimation Strategies for NRTL Binary Interaction Parameters

As already mentioned in Section 5.2.2, when considering the NRTL model the temperature dependent energy interaction parameters between a j-i pair of molecules are quantified by the variables $\tau_{j,i}, G_{j,i}$ and $\alpha_{j,i}$, defined by the equations (5.35), (5.36) and (5.37), respectively. For the sake of a better readability, the equations defining those variables are listed next, Eq. (B.1) to (B.3).

$$\tau_{j,i} = a_{j,i} + b_{j,i}/T + e_{j,i} \cdot \ln(T) + f_{j,i} \cdot T \tag{B.1}$$

$$G_{j,i} = \exp(-\alpha_{j,i} \cdot \tau_{j,i}) \tag{B.2}$$

$$\alpha_{j,i} = c_{j,i} + d_{j,i} \cdot (T - 273.15) \tag{B.3}$$

As can be seen from the upper equations, the temperature dependence of $\tau_{j,i}, G_{j,i}$ and $\alpha_{j,i}$ is in fact defined by the values of the parameters $a_{j,i}$, $b_{j,i}$, $c_{j,i}$, $d_{j,i}$, $e_{j,i}$ and $f_{j,i}$, which need to be estimated. The variable $\tau_{j,i}$ is in general asymmetric, that is $\tau_{j,i} \neq \tau_{i,j}$, and so are the parameters $a_{j,i}$, $b_{j,i}$, $e_{j,i}$ and $f_{j,i}$. On the other hand, the variable $\alpha_{j,i}$ and hence $c_{j,i}$ and $d_{j,i}$ are symmetric. While in principle, all parameters can be varied for fitting purposes, it has become common practice to consider fixed values for $c_{j,i}$ and neglect the temperature dependence given by $d_{j,i}$. Following these latter recommendations, all available LLE were fitted successfully using the estimation procedure given below (Section B.1.1). In order to fit the GLE properly, it was necessary to fit the values of $c_{j,i}$ and $d_{j,i}$ as well.

B.1.1 Estimation procedure

As previously mentioned, for the LLE estimation procedure temperature independent $\alpha_{j,i}$ were considered, that is $d_{j,i}$ is set to zero, and $c_{j,i}$ was either set to 0.2 for all compound pairs containing DMF and to the value of 0.3 for all other pairs that do not show a liquid-liquid split.

After having fixed the values of $c_{j,i}$, the next step is finding out, which model structure supported by Eq. (B.1) is required in order to represent the temperature dependency of the LLE properly. This is done by finding proper values of $a_{j,i}$ and perhaps $a_{i,j}$ that fit the data properly at each single temperature level. Having recollected the values of $a_{j,i}$ and perhaps $a_{i,j}$ for each considered temperature, it is now possible to check which type of profile $\tau_{j,i}(T)$ may lead to a good approximation of the LLE data, and to check how the type of profile can be approximated by Eq. (B.1). Dependent on the form of the temperature dependent $\tau(T)$ profile and the number of data available, a subset of the searched parameters is obtained with multi-linear regression. The results of the multi-linear regression are used as starting values for the parameter estimation carried out in Aspen. On the basis of the parameter standard deviation provided by the Aspen Plus data regression system, statistically insignificant parameters are identified and fixed to zero.

The procedure used to estimate the parameters required to describe the GLE data is similar to the one described above, but additionally the values and temperature dependency of $\alpha_{j,i}$ are considered as well. By setting proper parameter lower and upper bounds, it can be ensured that the values of $\alpha_{j,i}$ lie inside or close to the common range between 0.2 and 0.5.

B.2 Flash Calculation for Pure Components

Considering the formulation of a pT-flash as given in Section 2.4.2 as a starting point, when evaluating the phase equilibrium of a single component in two phases (L) and (V) for a given Temperature T, the resulting flash equation system reduces to an equation system of three equations (Eqs. (B.4) to (B.6)) and three unknown variables, P^{LV}, ρ^L and ρ^V.

$$P^{LV} = Z^L \cdot k \cdot T \cdot \rho^L \cdot \left(10^{10}\frac{\mathring{\mathrm{A}}}{m}\right)^3 \tag{B.4}$$

$$P^{LV} = Z^V \cdot k \cdot T \cdot \rho^V \cdot \left(10^{10}\frac{\mathring{\mathrm{A}}}{m}\right)^3 \tag{B.5}$$

$$\varphi^L = \varphi^V \tag{B.6}$$

It consists of the density root problem for each phase, Eq. (B.4) and Eq. (B.5), and the isofugacity conditions, which is equivalent with the equality of fugacity coefficients, Eq. (B.6). The compressibility factors and fugacity coefficients of the respective phases are calculated by Eq. (B.7) and Eq. (B.8) respectively.

$$Z^k = 1 + \rho^k \cdot \left(\frac{\partial \tilde{a}^{res,k}}{\partial \rho^k} \right) , \; k \in \{L, V\} \tag{B.7}$$

$$\varphi^k = \exp(\tilde{a}^{res,k} + Z^k - 1 - \ln Z^k) , \; k \in \{L, V\} \tag{B.8}$$

While the vapor pressure P^{LV} in [Pa] can be directly taken from the result of the aforementioned equation system, additional calculations may be required to obtain the saturated liquid density (ρ_L^{Sat}) in the required engineering units. As an example, in order to obtain the saturated liquid density in [kg m^{-3}] Eq. (B.9) can be explicitly evaluated

$$\rho_L^{Sat,M} = \frac{M \cdot P^{LV}}{Z^L \cdot R \cdot T}, \tag{B.9}$$

where the M in the superscript makes clear that mass density is meant in this case and the base name M denotes the molecular mass of the considered compound.

Appendix C

Own Publications and Supervised Theses

C.1 Peer-reviewed Journal Publications

Merchan, V. A., E. Esche, S. Fillinger, G. Tolksdorf, and G. Wozny (2016). Computer-Aided Process and Plant Development: A Review of Common Software Tools and Methods and Comparison against an Integrated Collaborative Approach. *Chemie Ingenieur Technik 88*(1-2), 50–69.

Merchan, V. A. and G. Wozny (2015). Comparative evaluation of rigorous thermodynamic models for the description of the hydroformylation of 1-dodecene in a thermomorphic solvent system. *Industrial & Engineering Chemistry Research 55*(1), 293–310.

Müller, D., D. M. Hoang, V. A. Merchan, H. Arellano-Garcia, Y. Kasaka, M. Müller, R. Schomäcker, and G. Wozny (2014). Towards a novel process concept for the hydroformylation of higher alkenes: Mini-plant operation strategies via model development and optimal experimental design. *Chemical Engineering Science 115*, 127–138.

Merchan, V. A., G. Tolksdorf, R. Kraus, and G. Wozny (2014). Extending documentation-based models towards an efficient integration into commercial process simulators. *Chemie-Ingenieur-Technik 86*(7), 1117–1129.

Kraus, R., S. Fillinger, G. Tolksdorf, D. H. Minh, V. A. Merchan, and G. Wozny (2014). Improving Model and Data Integration Using MOSAIC as Central Data Management Platform. *Chemie Ingenieur Technik 86*(7), 1130–1136.

Hoang, D. M., T. Barz, V. A. Merchan, L. T. Biegler, and H. Arellano-Garcia (2013). Simultaneous Solution Approach to Model-Based Experimental Design. *AIChE Journal 59*(11), 4169–4183.

C.2 Peer-reviewed Conference Proceedings

Merchan, V. A., R. Kraus, and G. Wozny (2013). Use of a web-based modeling environment in the education of process engineers. In P. Varbanov, J. J. Klemes, P. Seferlis, A. I. Papadopoulos and S. Vouteakis (Eds.), *16th International Conference on Process Integration, Modeling and Optimization for Energy Saving and Pollution Reduction*, Volume 35 of *Chemical Engineering Transactions*, pp. 673–678. AIDIC Servizi S.r.l.

Merchan, V. A., R. Kraus, T. Barz, H. Arellano-Garcia, and G. Wozny (2012). Generation of first and higher order derivative information out of the documentation level. In I. A. Karimi and R. Srinivasan (Eds.), *11th International Symposium on Process Systems Engineering International Symposium on Process Systems Engineering*, Volume 31 of *Computer Aided Chemical Engineering*, pp. 950–954. Elsevier B.V.

Kraus, R., V. A. Merchan, H. Arellano-Garcia, and G. Wozny (2012). Hierarchical simulation of integrated chemical processes with a web based modeling tool. In I. A. Karimi and R. Srinivasan (Eds.), *11th International Symposium on Process Systems Engineering International Symposium on Process Systems Engineering*, Volume 31 of *Computer Aided Chemical Engineering*, Singapore, pp. 155–159. Elsevier B.V.

Müller, M., V. A. Merchan, H. Arellano-Garcia, R. Schomäcker, and G. Wozny (2011). A novel process design for the hydroformylation of higher alkenes. In E. N. Pistikopoulos, M. Georgiadis and A. C. Kokossis (Eds.), *21 European Symposium on Computer Aided Process Engineering*, Volume 29 of *Computer Aided Chemical Engineering*, pp. 226–230. Elsevier B.V.

Merchan, V. A., S. Kuntsche, H. Arellano-Garcia, and G. Wozny (2011). Symbolic generation of first and higher-order derivatives with MOSAIC. In G. Wozny and L. Hady (Eds.), *International Conference Process Engineering and Chemical Plant Design*, pp. 114–123. Universitäsverlag der TU Berlin.

Kraus, R., V. A. Merchan, S. Kuntsche, F. Manenti, G. Buzzi-Ferraris and G. Wozny (2011). Modelling in the documentation level using MOSAIC and numerical libraries. In S. Pierucci (Ed.), *The tenth International Conference on Chemical and Process Engineering*, Volume 10 of *AIDIC Conference Series*, pp. 207–214. AIDIC Servizi S.r.l.

C.3 Presentations without Publication

(Presenting author is underlined).

Merchan, V. A., R. Kraus, and G. Wozny (2014) Computational aspects of the implementation of SAFT based EoS models using numerical analysis and automatic differentiation. In *20th European Conference on Thermophysical Properties*, 31 August - 4 September 2014, Porto, Portugal.

Hoang, D. M., T. Barz, V. A. Merchan, L. T. Biegler, H. Arellano-Garcia, and G. Wozny (2013). Model based experiment design: A comparison between sequential and simultaneous optimization approaches. In *9th World Congress of Chemical Engineering*, 18-23 August 2013, Seoul, South Korea.

Merchan, V. A., R. Kraus, H. Arellano-Garcia, and G. Wozny (2012) Einsatz einer webbasierten Modellierungsumgebung in der Ausbildung von Prozessingenieuren. In *Jahrestreffen der Fachgemeinschaft Prozess-, Apparate- und Anlagentechnik (PAAT)*, 19-20 November 2012, Dortmund, Germany.

Kraus, R., V. A. Merchan, H. Arellano-Garcia, and G. Wozny (2012) MOSAIC als online Plattform für die integrierte Verwaltung von Modellen und Messdaten. In *Jahrestreffen der Fachgemeinschaft Prozess-, Apparate- und Anlagentechnik (PAAT)*, 19-20 November 2012, Dortmund, Germany.

C.4 Supervised Student Projects

The following student projects and theses were conducted under my supervision during my time as research fellow at Technische Universität Berlin. Selected results from the theses were used in this dissertation.

Zhao, J. (2014) *Implementierung eines komplexen Zustandsgleichungsmodells unter Anwendung von algorithmischer Differentiation.* Diploma thesis.

Gutierrez-Sanchez, M. F. (2014) *Comparative evaluation of rigorous thermodynamic models for the description of the hydroformylation of long chain olefins in thermomorphic solvent systems* . Master's thesis.

Agudelo, J. (2014). *Dynamische Modellierung und Prozesssimulation der Hydroformylierung langkettiger Olefine unter Einsatz von CAPE-OPEN-Stoffdatenpaketen in gPROMS.* Master's thesis.

Rodriguez-Pizarro, L. d. l. P. (2013) *Modelación del equilibrio entre fases en sistemas líquidos multifásicos micelares mediante la minimización de la energía libre de Gibbs, utilizando la plataforma MOSAIC y adaptando la herramiena gráfica TERNPLOT incluida en MATLAB.* Internship.

Beinike, D., C. Bühl., D. Ewers, S. Funk, S., and D. Schaff (2012) *Simulation der Miniplant zur Hydroformylierung von langkettigen Olefinen.* Energy and Process Engineering Project.

Schlüter, M. (2012). *Automatisches Differenzieren in MOSAIC.* Internship.

Fillinger, S., D. Löffler, and J. N. Ratyanake (2011) Modellierung und Simulation mehrphasiger Systeme mit den kommerziellen Tools Aspen Plus und ACM am Beispiel der Hydroformylierung langkettiger Olefine. Energy and Process Engineering Project.

Bibliography

[1] Abrams, D. S. and J. M. Prausnitz (1975). Statistical thermodynamics of liquid mixtures: A new expression for the excess Gibbs energy of partly or completely miscible systems. *AIChE Journal 21*(1), 116–128. doi:10.1002/aic.690210115.

[2] Al-Jimaz, A. S., M. S. Fandary, J. A. Al-Kandary, and M. A. Fahim (2006). Phase equilibria for the extraction of sec-butylbenzene from dodecane with N,N-dimethylformamide. *Fluid Phase Equilibria 240*(1), 79–86. doi:10.1016/j.fluid.2005.12.006.

[3] Al-Saifi, N. M. (2011). *Prediction and Computation of Phase Equilibria in Polar and Polarizable Mixtures Using Theory-based Equations of State*. Ph.D. thesis, The University of British Columbia.

[4] Al-Saifi, N. M., E. Z. Hamad, and P. Englezos (2008). Prediction of vapor-liquid equilibrium in water-alcohol-hydrocarbon systems with the dipolar perturbed-chain SAFT equation of state. *Fluid Phase Equilibria 271*(1-2), 82–93. doi:10.1016/j.fluid.2008.06.015.

[5] Alloula, K., J.-P. Belaud, and J. M. Le Lann (2010). Solving CAPE models from Microsoft Office applications. In S. Pierucci and G. Buzzi-Ferraris (Eds.), *20th European Symposium on Computer Aided Process Engineering*, Volume 28 of *Computer Aided Chemical Engineering*, pp. 679–684. Elsevier B.V.

[6] Alsaifi, N. M. and P. Englezos (2011). Prediction of multiphase equilibrium using the PC-SAFT equation of state and simultaneous testing of phase stability. *Fluid Phase Equilibria 302*(1-2), 169–178. doi:10.1016/j.fluid.2010.09.002.

[7] Andersson, J. (2013). *A General-Purpose Software Framework for Dynamic Optimization*. Ph.D. thesis, Arenberg Doctoral School, KU Leuven.

[8] Arellano-Garcia, H., J. Schöneberger, and S. Körkel (2007). Optimale Versuchsplanung in der Chemischen Verfahrenstechnik. *Chemie Ingenieur Technik 79*(10), 1625–1638.

[9] Ascher, U. and L. R. Petzold (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Philadelphia: Society for Industrial and Applied Mathematics.

[10] AspenTech (2018a). Aspen Custom Modeler. Available from: `https://www.aspentech.com/products/pages/aspen-custom-modeler`.

[11] AspenTech (2018b). Aspen Plus. Available from: www.aspentech.com.

[12] Baker, L. E., A. C. Pierce, and K. D. Luks (1980). Gibbs Energy Analysis of Phase Equilibria. *Society of Petroleum Engineers journal 22*(5), 731–742.

[13] Behr, A. and C. Fängewisch (2002). Temperature-dependent multicomponent solvent systems - An alternative concept for recycling homogeneous catalysts. *Chemical Engineering and Technology 25*(2), 143–147. doi:10.1002/1521-4125(200202)25:2<143::AID-CEAT143>3.0.CO;2-O.

[14] Belaud, J.-P., K. Alloula, J.-M. Le Lann, and X. Joulia (2001). Open software architecture for numerical solvers : design, implementation and validation. In R. Gani and S. B. Jørgensen (Eds.), *European Symposium on Computer Aided Process Engineering - 11, 34th European Symposium of the Working Party on Computer Aided Process Engineering*, Volume 9 of *Computer Aided Chemical Engineering*, Kolding, Denmark, pp. 967–972. Elsevier B.V.

[15] Belaud, J.-P. and M. Pons (2002). Open software architecture for process simulation: The current status of CAPE-OPEN standard. In J. Grieving and V. Schijndel (Eds.), *European Symposium on Computer Aided Process Engineering-12, 35th European Symposium of the Working Party on Computer Aided Process Engineering*, Volume 10 of *Computer Aided Chemical Engineering*, The Hague, The Netherlands, pp. 847–852. Elsevier B.V.

[16] Biegler, L. T. (2010). *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics. doi:10.1137/1.9780898719383.

[17] Biegler, L. T. (2014). Recent Advances in Chemical Process Optimization. *Chemie Ingenieur Technik 86*(7), 943–952. doi:10.1002/cite.201400033.

[18] Biegler, L. T., I. E. Grossman, and A. W. Westerberg (1997). *Systematic Methods of Chemical Process Design*. Prentice Hall International Series in the Physical and Chemical Engineering Sciences. Prentice Hall.

[19] Bischof, C. and H. Bücker (2000). Computing Derivatives of Computer Programs. In J. Grotendoost (Ed.), *Modern Methods and Algorithms of Quantum Chemistry - Proceedings, Second Edition*, Volume 3 of *NIC Series*, Jülich, pp. 315–327. John von Neumann Institute for Computing (NIC). Available from: http://www.john-von-neumann-institut.de/nic/EN/Publications/NICSeries/nic-series-1-10.html?nn=1487798.

[20] Bischof, C., H. Bucker, B. Lang, A. Rasch, and A. Vehreschild (2002). Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs. In *Proceedings. Second IEEE International Workshop on Source Code Analysis and Manipulation*, pp. 65–72. IEEE Comput. Soc. Available from: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1134106.

[21] Bischof, C. H., H. M. Bücker, W. Marquardt, M. Petera, and J. Wyes (2006). Transforming Equation-Based Models in Process Engineering. In M. Bücker, G. F. Corliss, U. Naumann, P. D. Hovland, and B. Norris (Eds.), *Automatic Differentiation: Applications, Theory, and Implementations*, Volume 50 of *Lecture Notes in Computational Science and Engineering*, pp. 189–198. Springer.

[22] Bischof, C. H., L. Roh, and A. J. Mauer-Oats (1997). ADIC: an extensible automatic differentiation tool for ANSI-C. *Software: Practice and Experience 27*(12), 1427–1456. doi:10.1002/(SICI)1097-024X(199712)27:12<1427::AID-SPE138>3.0.CO;2-Q.

[23] Blas, F. J. and L. F. Vega (1998). Prediction of Binary and Ternary Diagrams Using the Statistical Associating Fluid Theory (SAFT) Equation of State. *Industrial & Engineering Chemistry Research 37*(2), 660–674. doi:10.1021/ie970449+.

[24] Bogusch, R., B. Lohmann, and W. Marquardt (2001). Computer-aided process modeling with ModKit. *Computers & Chemical Engineering 25*(7-8), 963–995. doi:10.1016/S0098-1354(01)00626-3.

[25] Bonart, H. (2016). *Source Code Generation for Parallelized Simulations of a Dynamic Nonequilibrium Separation Unit using MOSAIC*. Master' s thesis, Technische Universität Berlin.

[26] Braunschweig, B. L., C. C. Pantelides, H. I. Britt, and S. Sama (2000). Process modeling: the promise of open software architectures. *Chemical Engineering Progress 96*(9), 65–76.

[27] Brunsch, Y. and A. Behr (2013). Temperature-controlled catalyst recycling in homogeneous transition-metal catalysis: Minimization of catalyst leaching. *Angewandte Chemie International Edition 52*(5), 1586–1589. doi:10.1002/anie.201208667.

[28] Bublitz, S., E. Esche, G. Tolksdorf, V. Mehrmann, and J. U. Repke (2017). Analysis and Decomposition for Improved Convergence of Nonlinear Process Models in Chemical Engineering. *Chemie Ingenieur Technik 89*(11), 1503–1514. doi:10.1002/cite.201700041.

[29] Bussieck, M. R. and A. Meeraus (2004). General Algebraic Modeling System (GAMS). In J. Kallrath (Ed.), *Modeling Languages in Mathematical Optimization*, Volume 88 of *Applied Optimization*, Chapter 8, pp. 137–157. Springer US.

[30] Buzzi-Ferraris, G. and F. Manenti (2012). BzzMath: Library Overview and Recent Advances in Numerical Methods. In I. D. Lockhart Bogle and M. Fairweather (Eds.), *22nd European Symposium on Computer Aided Process Engineering*, Volume 30 of *Computer Aided Chemical Engineering*, pp. 1312–1316. Elsevier B.V.

[31] Cameron, I. T. (2011). *Product and process modelling : a case study approach*. Elsevier. doi:10.1016/B978-0-444-53161-2/00016-9.

[32] Carlson, E. C. (1996). Don't Gamble With Physical Properties. *Chemical Engineering and Processing 10*, 35–46.

[33] Castier, M. (1999). Automatic implementation of thermodynamic models using computer algebra. *Computers & Chemical Engineering 21*(9), 1229–1245.

[34] Chapman, W. G., K. E. Gubbins, G. Jackson, and M. Radosz (1989). SAFT: Equation-of-state solution model for associating fluids. *Fluid Phase Equilibria 52*(C), 31–38. doi:10.1016/0378-3812(89)80308-5.

[35] Chapman, W. G., G. Jackson, and K. E. Gubbins (1988). Phase equilibria of associating fluids: chain molecules with multiple bonding sites. *Molecular Physics 65*(5), 1057–1079.

[36] Chemstations (2018). CHEMCAD. Available from: `www.chemstations.com`.

[37] CO-LaN. CAPE-OPEN Business Interfaces version 1.0. Available from: `http://www.colan.org/index-33.html`.

[38] CO-LaN Consortium (2003). CAPE-OPEN Methods & Tools Integrated Guidelines. Available from: `http://www.colan.org/index-33.html`.

[39] Curtis, A. R., M. J. D. Powell, and J. K. Reid (1974). On the Estimation of Spare Jacobian Matrices. *Journal of the Institute of Mathematics and its Applications 13*(1), 117–119.

[40] Dahl, S. and M. L. Michelsen (1990). High-pressure vapor-liquid equilibrium with a UNIFAC-based equation of state. *AIChE Journal 36*(12), 1829–1836. doi:10.1002/aic.690361207.

[41] Dechambre, D., C. Pauls, L. Greiner, K. Leonhard, and A. Bardow (2014). Towards automated characterisation of liquid-liquid equilibria. *Fluid Phase Equilibria 362*, 328–334. doi:10.1016/j.fluid.2013.10.048.

[42] Dohrn, R. (1994). *Berechnung von Phasengleichgewichten*. Grundlagen und Fortschritte der Ingenieurwissenschaften. Vieweg+Teubner Verlag.

[43] Dowling, A. W. and L. T. Biegler (2015). A framework for efficient large scale equation-oriented flowsheet optimization. *Computers & Chemical Engineering 72*, 3–20. doi:10.1016/j.compchemeng.2014.05.013.

[44] Economou, I. G. (2002). Statistical Associating Fluid Theory: A Successful Model for the Calculation of Thermodynamic and Phase Equilibrium Properties of Complex Fluid Mixtures. *Industrial & Engineering Chemistry Research 41*(5), 953–962. doi:10.1021/ie0102201.

[45] Economou, I. G. and M. D. Donohue (1991). Chemical, quasi-chemical and perturbation theories for associating fluids. *AIChE Journal 37*(12), 1875–1894. doi:10.1002/aic.690371212.

[46] Elsheikh, A. (2012). *Modelica-Based Computational Tools for Sensitivity Analysis via Automatic Differentiation*. Ph.D. thesis, RWTH Aachen University.

[47] Elsheikh, A. (2015). An equation-based algorithmic differentiation technique for differential algebraic equations. *Journal of Computational and Applied Mathematics 281*, 135–151. Available from: `http://dx.doi.org/10.1016/j.cam.2014.12.026`, doi:10.1016/j.cam.2014.12.026.

[48] Elsheikh, A. and W. Wiechert (2008). Automatic Sensitivity Analysis of DAE-Systems Generated from Equation-Based Modeling Languages. In C. H. Bischof, H. Bücker, P. D. Hovland, U. Naumann, and J. Utke (Eds.), *Advances in Automatic Differentiation*, Volume 64 of *Lecture Notes in Computational Science and Engineering*, pp. 235–246. Springer.

[49] Esche, E., D. Müller, G. Tolksdorf, R. Kraus, and G. Wozny (2014). MOSAIC: An online modeling platform supporting automatic discretization of partial differential equation systems. In M. R. Eden, J. D. Siirola, and G. P. Towler (Eds.), *Proceedings of the 8th International Conference on Foundations of Computer-Aided Process Design*, Volume 34 of *Computer Aided Chemical Engineering*, pp. 693–698.

[50] Fieg, G., W. Gutermuth, W. Kothe, H. Mayer, S. Nagel, H. Wendeler, and G. Wozny (1995). A standard interface for use of thermodynamics in process simulation. *Computers & Chemical Engineering 19, Sup. 1*, 317 – 320.

[51] Florusse, L. J., C. J. Peters, J. C. Pàmies, L. F. Vega, and H. Meijer (2003). Solubility of Hydrogen in Heavy n-Alkanes: Experiments and SAFT Modeling. *AIChE Journal 49*(12), 3260–3269. doi:10.1002/aic.690491225.

[52] Folas, G. K., G. M. Kontogeorgis, M. L. Michelsen, and E. H. Stenby (2006). Application of the cubic-plus-association equation of state to mixtures with polar chemicals and high pressures. *Industrial & Engineering Chemistry Research 45*(4), 1516–1526. doi:10.1021/ie0509241.

[53] Forth, S. (2006). An efficient overloaded implementation of forward mode automatic differentiation in MATLAB. *ACM Transactions on Mathematical Software (TOMS) 32*(2), 195–222. doi:10.1145/1141885.1141888.

[54] Fourer, R., D. M. Gay, and B. W. Kernighan (2004). Design Principles and New Developments in the AMPL Modeling Language. In J. Kallrath (Ed.), *Modeling Languages in Mathematical Optimization*, Volume 88 of *Applied Optimization*, Chapter 7, pp. 105–135. Springer US.

[55] Fredenslund, A., R. L. Jones, and J. M. Prausnitz (1975). Group-contribution estimation of activity coefficients in nonideal liquid mixtures. *AIChE Journal 21*(6), 1086–1099. doi:10.1002/aic.690210607.

[56] Frenkel, M., R. D. Chirico, V. Diky, X. Yan, Q. Dong, and C. Muzny (2005). ThermoData Engine (TDE): Software implementation of the dynamic data evaluation concept. *Journal of Chemical Information and Modeling 45*(4), 816–838. doi:10.1021/ci050067b.

[57] Frenkel, M., V. Diky, R. D. Chirico, R. N. Goldberg, H. Heerklotz, J. E. Ladbury, D. P. Remeta, J. H. Dymond, A. R. H. Goodwin, K. N. Marsh, W. A. Wakeham,

S. E. Stein, P. L. Brown, E. Königsberger, and P. A. Williams (2011). ThermoML: An XML-based approach for storage and exchange of experimental and critically evaluated thermophysical and thermochemical property data. 5. Speciation and complex equilibria. *Journal of Chemical and Engineering Data 56*(2), 307–316. doi:10.1021/je100999j.

[58] Fritzson, P. and V. Engelson (1998). Modelica-A unified object-oriented language for system modeling and simulation. In E. Jul (Ed.), *ECOOP'98-Object-Oriented Programming*, Volume 1445 of *Lecture Notes in Computer Science*, pp. 67–90. Springer.

[59] Fu, Y.-H. and S. I. Sandler (1995). A Simplified SAFT Equation of State for Associating Compounds and Mixtures. *Industrial & Engineering Chemistry Research 34*(5), 1897–1909. doi:10.1021/ie00044a042.

[60] Gani, R., I. T. Cameron, A. Lucia, G. Sin, and M. Georgiadis (2012). Process Systems Engineering , 2 . Modeling and Simulation. In *Ullmann's Encyclopedia of Industrial Chemistry*. Wiley-VCH.

[61] Gani, R. and J. P. O. Connell (2004). Role of Properties and their Models in Process and Product Design. In R. Gani and G. M. Kontogeorgis (Eds.), *Computer Aided Property Estimation for Process and Product Design*, Volume 19 of *Computer Aided Chemical Engineering*, Chapter 2, pp. 27–41. Elsevier.

[62] Gani, R., N. Muro-Suñé, M. Sales-Cruz, C. Leibovici, and J. P. O'Connell (2006). Mathematical and numerical analysis of classes of property models. *Fluid Phase Equilibria 250*(1-2), 1–32. doi:10.1016/j.fluid.2006.09.010.

[63] Gay, D. M. (1991). Automatic Differentiation of Nonlinear AMPL Models. In A. Griewank and G. F. Corliss (Eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pp. 61–73. Society for Industrial and Applied Mathematics.

[64] Gay, D. M. (1996). More AD of Nonlinear AMPL Models: Computing Hessian Information and Exploiting Partial Separability. In M. Berz, C. H. Bischof, G. F. Corliss, and A. Griewank (Eds.), *Computational Differentiation: Techniques, Applications, and Tools*, pp. 173–184. Philadelphia: Society for Industrial and Applied Mathematics.

[65] Gay, D. M. (2001). Symbolic-Algebraic Computations in a Modeling Language for Mathematical Programming. In G. Alefeld, J. Rohn, S. Rump, and T. Yamamoto (Eds.), *Symbolic Algebraic Methods and Verification Methods*, pp. 99–106. Springer Vienna.

[66] Gernaey, K. V., J. Glassey, S. Skogestad, S. Krämer, A. Weiß, S. Engell, E. N. Pistikopoulos, and D. B. Cameron (2012). Process Systems Engineering , 5 . Process Dynamics , Control , Monitoring , and Identification. In *Ullmann's Encyclopedia of Industrial Chemistry*.

[67] Gil-Villegas, A., A. Galindo, P. J. Whitehead, S. J. Mills, G. Jackson, and A. N. Burgess (1997). Statistical associating fluid theory for chain molecules with attractive potentials of variable range. *The Journal of Chemical Physics 106*(10), 4168. doi:10.1063/1.473101.

[68] Gmehling, J., B. Kolbe, M. Kleiber, and J. Rarey (2012). *Chemical Thermodynaics for Process Simulation*. Wiley-VCH.

[69] Griewank, A. (1989). On Automatic Differentiation. In M. Iri and K. Tanabe (Eds.), *Mathematical Programming. Recent Developments and Applications* (1 ed.), Volume 6 of *Mathematics and its Applications*, pp. 83–107. Springer Netherlands.

[70] Griewank, A. (2009). Complexity of Gradients, Jacobians and Hessians. In C. A. Floudas and P. M. Pardalos (Eds.), *Encyclopedia of Optimization* (2nd ed.)., pp. 425–435. Springer US.

[71] Griewank, A. and A. Walther (2008). *Evaluating derivatives* (2nd rev. ed.). Philadelphia: Society for Industrial and Applied Mathematics.

[72] Gross, J. (2001). *Entwicklung einer Zustandsgleichung für einfach, assoziierende und makromolekulare Stoffe*, Volume 684 of *Fortschritt-Berichte VDI, Reihe 3*. VDI-Verlag.

[73] Gross, J. (2005). An equation-of-state contribution for polar components: quadrupolar molecules. *AIChE Journal 51*(9), 2556–2568.

[74] Gross, J. and G. Sadowski (2001). Perturbed-Chain SAFT: An Equation of State Based on a Perturbation Theory for Chain Molecules. *Industrial & Engineering Chemistry Research 40*(4), 1244–1260. doi:10.1021/ie0003887.

[75] Gross, J., O. Spuhl, F. Tumakaka, and G. Sadowski (2003). Modeling Copolymer Systems Using the Perturbed-Chain SAFT Equation of State. *Industrial & Engineering Chemistry Research 42*(6), 1266–1274. doi:10.1021/ie020509y.

[76] Gross, J. and J. Vrabec (2006). An equation-of-state contribution for polar components: Dipolar molecules. *AIChE Journal 52*(3), 1194–1204. doi:10.1002/aic.10683.

[77] Gupta, A. K., P. Raj Bishnoi, and N. Kalogerakis (1991). A method for the simultaneous phase equilibria and stability calculations for multiphase reacting and non-reacting systems. *Fluid Phase Equilibria 63*(1-2), 65–89. doi:10.1016/0378-3812(91)80021-M.

[78] Gutierrez-Sanchez, M. F. (2014). *Comparative Evaluation of Rigorous Thermodynamic Models for the description of the Hydroformylation of Long Chain Olefins in Thermomorphic Solvent Systems*. Master's thesis, Technische Universität Berlin.

[79] Hangos, K. and I. Cameron (2001). *Process Modelling and Model Analysis*, Volume 4 of *Process Systems Engineering*. London: Academic Press.

[80] Hannemann-Tamás, R. (2013). *Adjoint Sensitivity Analysis for Optimal Control of Non-Smooth Differential-Algebraic Equations*. Shaker Verlag.

[81] Harding, S. T. and C. A. Floudas (2000). Phase stability with cubic equations of state: Global optimization approach. *AIChE Journal 46*(7), 1422–1440. doi:10.1002/aic.690460715.

[82] Haumann, M., H. Koch, P. Hugo, and R. Schomäcker (2002). Hydroformylation of 1-dodecene using Rh-TPPTS in a microemulsion. *Applied Catalysis A: General 225*(1-2), 239–249. doi:10.1016/S0926-860X(01)00869-9.

[83] Heitzig, M. (2011). *Computer-aided modelling for efficient and innovative product-process engineering Computer-aided modelling for efficient and innovative product-process*. Ph.D. thesis, Technical University of Denmark.

[84] Hendriks, E., G. M. Kontogeorgis, R. Dohrn, J. C. De Hemptinne, I. G. Economou, L. F. Zilnik, and V. Vesovic (2010). Industrial requirements for thermodynamics and transport properties. *Industrial & Engineering Chemistry Research 49*(22), 11131–11141. doi:10.1021/ie101231b.

[85] Hentschel, B., A. Peschel, M. Xie, C. Vogelpohl, G. Sadowski, H. Freund, and K. Sundmacher (2014). Model-based prediction of optimal conditions for 1-octene hydroformylation. *Chemical Engineering Science 115*, 58–68. doi:10.1016/j.ces.2013.03.051.

[86] Hoang, M. D., T. Barz, V. A. Merchan, L. T. Biegler, and H. Arellano-Garcia (2013). Simultaneous Solution Approach to Model-Based Experimental Design. *AIChE Journal 59*(11), 4169–4183. doi:10.1002/aic.14145.

[87] Hua, J. Z., J. F. Brennecke, and M. A. Stadtherr (1998). Enhanced interval analysis for phase stability: Cubic equation of state models. *Industrial & Engineering Chemistry Research 37*(4), 1519–1527.

[88] Huang, S. H. and M. Radosz (1990). Equation of state for small, large, polydisperse and associating molecules. *Industrial & Engineering Chemistry Research 29*, 2284–2294. doi:10.1021/ie00107a014.

[89] Huang, S. H. and M. Radosz (1991). Equation of state for small, large, polydisperse, and associating molecules: extension to fluid mixtures. *Industrial & Engineering Chemistry Research 30*(8), 1994–2005.

[90] Jackson, G., W. G. Chapman, and K. E. Gubbins (1988). Phase equilibria of associating fluids of spherical and chain molecules. *International Journal of Thermophysics 9*(5), 769–779. doi:10.1007/BF00503243.

[91] Jakob, A., H. Grensemann, J. Lohmann, and J. Gmehling (2006). Further development of modified UNIFAC (Dortmund): Revision and extension 5. *Industrial & Engineering Chemistry Research 45*(23), 7924–7933. doi:10.1021/ie060355c.

[92] Jin, H. and B. Subramaniam (2004). Homogeneous catalytic hydroformylation of 1-octene in CO 2-expanded solvent media. *Chemical Engineering Science 59*(22-23), 4887–4893. doi:10.1016/j.ces.2004.09.034.

[93] Jog, P. K. and W. G. Chapman (1999). Application of Wertheim's thermodynamic perturbation theory to dipolar hard sphere chains. *Molecular Physics 97*(3), 307–319. doi:10.1080/002689799163703.

[94] Kahrs, O. (2010). *Semi-empirical modeling of process systems*. Fortschritt-Berichte VDI: Reihe 3, Verfahrenstechnik, Berichte aus der Aachener Verfahrenstechnik - Prozesstechnik, RWTH Aachen University, 915. Düsseldorf: VDI-Verlag.

[95] Kamath, R. S., L. T. Biegler, and I. E. Grossmann (2010). An equation-oriented approach for handling thermodynamics based on cubic equation of state in process optimization. *Computers & Chemical Engineering 34*(12), 2085–2096. doi:10.1016/j.compchemeng.2010.07.028.

[96] Karakatsani, E. K., G. M. Kontogeorgis, and I. G. Economou (2006). Evaluation of the truncated perturbed chain-polar statistical associating fluid theory for complex mixture fluid phase equilibria. *Industrial & Engineering Chemistry Research 45*(17), 6063–6074. doi:10.1021/ie060313o.

[97] Karakatsani, E. K., T. Spyriouni, and I. G. Economou (2005). Extended statistical associating fluid theory (SAFT) equations of state for dipolar fluids. *AIChE Journal 51*(8), 2328–2342. doi:10.1002/aic.10473.

[98] Kiedorf, G., D. M. Hoang, A. Müller, A. Jörke, J. Markert, H. Arellano-Garcia, A. Seidel-Morgenstern, and C. Hamel (2014). Kinetics of 1-dodecene hydroformylation in a thermomorphic solvent system using a rhodium-biphephos catalyst. *Chemical Engineering Science 115*, 31–48.

[99] Kilakos, A. and B. Kalitventzeff (1993). A new implementation for analytical derivatives of thermodynamic properties and its beneficial application on dynamic simulation. *Computers & Chemical Engineering 17*(5-6), 441–450. doi:10.1016/0098-1354(93)80035-L.

[100] Kleiner, M. and J. Gross (2006). An equation of state contribution for polar components: Polarizable dipoles. *AIChE Journal 52*(5), 1951–1961.

[101] Koak, N., T. W. de Loos, and R. A. Heidemann (1999). Effect of the power series dispersion term on the pressure - volume behavior of Statistical Associating Fluid Theory. *Industrial & Engineering Chemistry Research 38*(4), 1718–1722. doi:10.1021/ie9804069.

[102] Koch, D. and W. Leitner (1998). Rhodium-catalyzed hydroformylation in supercritical carbon dioxide. *Journal of the American Chemical Society 120*(51), 13398–13404. doi:10.1021/ja980729w.

[103] Köller, J., J.-P. Belaud, M. Jarke, A. Kuckelberg, and T. Teague (2002). Methods & tools for software architecture. In B. L. Braunschweig and R. Gani (Eds.), *Software Architectures and Tools for Computer Aided Process Engineering*, Volume 11 of *Computer Aided Chemical Engineering*, Chapter 4.1, pp. 229–266. Elsevier B.V.

[104] Kontogeorgis, G. M. (2013). Association theories for complex thermodynamics. *Chemical Engineering Research and Design 91*(10), 1840–1858. doi:10.1016/j.cherd.2013.07.006.

[105] Kontogeorgis, G. M. and P. Coutsikos (2012). Thirty years with EoS/GE models-what have we learned? *Industrial & Engineering Chemistry Research 51*(11), 4119–4142. doi:10.1021/ie2015119.

[106] Kontogeorgis, G. M. and G. K. Folas (2009). *Thermodynamic Models for Industrial Applications: From Classical and Advanced Mixing Rules to Association Theories*. John Wiley & Sons. doi:10.1002/9780470747537.

[107] Kontogeorgis, G. M., M. L. Michelsen, G. K. Folas, S. Derawi, N. Von Solms, and E. H. Stenby (2006a). Ten Years with the CPA (Cubic-Plus-Association) equation of state. Part 1. Pure compounds and self-associating systems. *Industrial & Engineering Chemistry Research 45*(14), 4855–4868. doi:10.1021/ie051305v.

[108] Kontogeorgis, G. M., M. L. Michelsen, G. K. Folas, S. Derawi, N. Von Solms, and E. H. Stenby (2006b). Ten Years with the CPA (Cubic-Plus-Association) equation of state. Part 1. Pure compounds and self-associating systems. *Industrial & Engineering Chemistry Research 45*(14), 4855–4868. doi:10.1021/ie051305v.

[109] Kontogeorgis, G. M., E. C. Voutsas, I. V. Yakoumis, and D. P. Tassios (1996). An Equation of State for Associating Fluids. *Industrial & Engineering Chemistry Research 35*(11), 4310–4318.

[110] Kraska, T. and K. E. Gubbins (1996). Phase Equilibria Calculations with a Modified SAFT Equation of State. 1. Pure Alkanes, Alkanols, and Water. *Industrial & Engineering Chemistry Research 35*(12), 4727–4737. doi:10.1021/ie9602320.

[111] Kraus, R. (2015). *Concepts and implementation of Collaborative Modeling in MOSAIC*. Ph.D. thesis, Technische Universität Berlin.

[112] Kraus, R., S. Fillinger, G. Tolksdorf, D. H. Minh, V. A. Merchan, and G. Wozny (2014). Improving Model and Data Integration Using MOSAIC as Central Data Management Platform. *Chemie Ingenieur Technik 86*(7), 1130–1136. doi:10.1002/cite.201400007.

[113] Kuntsche, S. (2013). *Modular Model Specification on the Documentation Level*. Ph.D. thesis, Technische Universität Berlin.

[114] Kuntsche, S., H. Arellano-Garcia, and G. Wozny (2010). A new modeling environment based on internet-standards XML and MathML. In S. Pierucci and G. Buzzi-Ferraris (Eds.), *20th European Conference on Thermophysical Properties*, Volume 28 of *Computer Aided Process Engineering*, pp. 673–678. Elsevier B.V. doi:10.1016/S1570-7946(10)28113-0.

[115] Kuntsche, S., T. Barz, R. Kraus, H. Arellano-Garcia, and G. Wozny (2011). MOSAIC a web-based modeling environment for code generation. *Computers & Chemical Engineering 35*(11), 2257–2273. doi:10.1016/j.compchemeng.2011.03.022.

[116] Lang, Y.-D. and L. Biegler (2005). Large-Scale Nonlinear Programming with a CAPE-OPEN Compliant Interface. *Chemical Engineering Research and Design 83*(6), 718–723. doi:10.1205/cherd.04377.

[117] Langtangen, H. P. (2009). *Python Scripting for Computational Science* (3rd ed.), Volume 3 of *Text in Computational Science and Engineering*. Springer.

[118] Lantoine, G., R. P. Russell, and T. Dargent (2012). Using multicomplex variables for automatic computation of high-order derivatives. *ACM Transactions on Mathematical Software (TOMS) 38*(3), 16:1–16:21.

[119] Løvfall, B. T. (2008). *Computer Realization of Thermodynamic Models Using Algebraic Objects*. Ph.D. thesis, Norwegian University of Science and Technology.

[120] Macchietto, S., G. I. Maduabueke, and R. Szczepanski (1988). Efficient implementation of VLE procedures in equation-oriented simulators. *AIChE Journal 34*(6), 955–963. doi:10.1002/aic.690340608.

[121] Mangold, M., K. D. Mohl, S. Grüner, A. Kienle, and E. D. Gilles (2002). Nonlinear analysis of gPROMS models using DIVA via a CAPE ESO interface. In J. Grievink and J. van Schijndel (Eds.), *European Symposium on Computer Aided Process Engineering-12, 35th European Symposium of the Working Party on Computer Aided Process Engineering*, Volume 10, The Hague, pp. 919–924.

[122] Maplesoft TM (2016). Maple. Available from: `http://www.maplesoft.com/products/maple/`.

[123] Markert, J., Y. Brunsch, T. Munkelt, G. Kiedorf, A. Behr, C. Hamel, and A. Seidel-Morgenstern (2013). Analysis of the reaction network for the Rh-catalyzed hydroformylation of 1-dodecene in a thermomorphic multicomponent solvent system. *Applied Catalysis A: General 462-463*, 287–295. doi:10.1016/j.apcata.2013.04.005.

[124] Marquardt, W. (1995). Towards a Process Modeling Methodolgy. In R. Berver (Ed.), *Methods of Model Based Process Control*, Volume 293 of *NATO ASI Series: Serie E, Applied sciences*, pp. 3–40. Kluwer Acad. Publ.

[125] Marquardt, W. (1996). Trends in computer-aided process modeling. *Computers & Chemical Engineering 20*(6), 591–609. doi:10.1016/0098-1354(95)00195-6.

[126] Marquardt, W. (2000). Perspectives on Lifecycle Process Modeling. In M. F. Malone (Ed.), *Proceedings of the Fifth International Conference on Chemical Process Design*, Volume 96 of *AIChE Symposium Series*, Breckenridge, Colorado, pp. 192–214.

[127] Marquardt, W. (2008). Higher-Order Derivatives in Computational Systems Engineering Problem Solving.

[128] Martín, Á., M. D. Bermejo, F. A. Mato, and M. J. Cocero (2011). Teaching advanced equations of state in applied thermodynamics courses using open source programs. *Education for Chemical Engineers 6*(4), 114–121. doi:10.1016/j.ece.2011.08.003.

[129] Martins, J. R. R. A., P. Sturdza, and J. J. Alonso (2003). The complex-step derivative approximation. *ACM Trans. Math. Softw. 29*(3), 245–262. doi:10.1145/838250.838251.

[130] Matzopoulos, M. (2011). Dynamic Process Modeling: Combining Models and Experimental Data to Solve Industrial Problems. In M. C. Georgiadis, J. R. Banga, and E. N. Pistikopoulos (Eds.), *Process Systems Engineering*, Volume 7 of *Process Systems Engineering*, Chapter 1, pp. 1–33. Weinheim: Wiley-VCH. doi:10.1002/9783527631339.ch1.

[131] McDonald, C. M. and C. A. Floudas (1997). GLOPEQ: A new computational tool for the phase and chemical equilibrium problem. *Computers & Chemical Engineering 21*(1), 1–23. doi:10.1016/0098-1354(95)00250-2.

[132] Merchan, V. A., E. Esche, S. Fillinger, G. Tolksdorf, and G. Wozny (2016). Computer-Aided Process and Plant Development: A Review of Common Software Tools and Methods and Comparison against an Integrated Collaborative Approach. *Chemie Ingenieur Technik 88*(1-2), 50—-69.

[133] Merchan, V. A., R. Kraus, T. Barz, H. Arellano-Garcia, and G. Wozny (2012). Generation of first and higher order derivative information out of the documentation level. In I. A. Karimi and R. Srinivasan (Eds.), *11th International Symposium on Process Systems Engineering International Symposium on Process Systems Engineering*, Volume 31 of *Computer Aided Chemical Engineering*, Singapore, pp. 950–954. Elsevier B.V.

[134] Merchan, V. A., S. Kuntsche, H. Arellano-Garcia, and G. Wozny (2011). Symbolic generation of first and higher-derivatives with MOSAIC. In G. Wozny and L. Hady (Eds.), *18th International Conference Process Engineering and Chemical Plant Design*, Berlin.

[135] Merchan, V. A., G. Tolksdorf, R. Kraus, and G. Wozny (2014). Extending documentation-based models towards an efficient integration into commercial process simulators. *Chemie Ingenieur Technik 86*(7), 1117–1129. doi:10.1002/cite.201400014.

[136] Merchan, V. A. and G. Wozny (2015). Comparative evaluation of rigorous thermodynamic models for the description of the hydroformylation of 1-dodecene in a thermomorphic solvent system. *Industrial & Engineering Chemistry Research 55*(1), 293–310. doi:10.1021/acs.iecr.5b03328.

[137] Michelsen, M. L. (1982). The isothermal flash problem. part I. Stability. *Fluid Phase Equilibria 9*, 1–19.

[138] Michelsen, M. L. (1993). Phase equilibrium calculations. What is easy and what is difficult? *Computers & Chemical Engineering 17*(5-6), 431–439. doi:10.1016/0098-1354(93)80034-K.

[139] Michelsen, M. L. and H. Kistenmacher (1990). On composition-dependent interaction coefficients. *Fluid Phase Equilibria 58*, 229–230.

[140] Michelsen, M. L. and J. Mollerup (1986). Partial derivatives of thermodynamic properties. *AIChE Journal 32*(8), 1389–1392. doi:10.1002/aic.690320818.

[141] Mischler, C., X. Joulia, E. Hassold, A. Galligo, and R. Esposito (1995). Automatic Differentiation applications to computer aided process engineering. *Computers & Chemical Engineering 19*, 779–784. doi:10.1016/0098-1354(95)87129-2.

[142] Mitsos, A. and P. I. Barton (2007). A dual extremum principle in thermodynamics. *AIChE Journal 53*(8), 2131–2147. doi:10.1002/aic.11230.

[143] Mitsos, A., G. M. Bollas, and P. I. Barton (2009). Bilevel optimization formulation for parameter estimation in liquid-liquid phase equilibrium problems. *Chemical Engineering Science 64*(3), 548–559. doi:10.1016/j.ces.2008.09.034.

[144] Mollerup, J. and M. L. Michelsen (2004). *Thermodynamic Models: Fundamentals & Computational Aspects*. Tie-Line Publications.

[145] Mollerup, J. M. and M. L. Michelsen (1992). Calculation of thermodynamic equilibrium properties. *Fluid Phase Equilibria 74*(C), 1–15. doi:10.1016/0378-3812(92)85049-E.

[146] Monroy-Loperena, R. (2012). A note on the analytical solution of cubic equations of state in process simulation. *Industrial & Engineering Chemistry Research 51*(19), 6972–6976. doi:10.1021/ie2023004.

[147] Moré, J. J. (1990). A Collection of Nonlinear Model Problems. In E. L. Allgower and K. Georg (Eds.), *Computational solution of nonlinear systems of equations*, Volume 26 of *Lectures in applied mathematics*, pp. 723–762. American Mathematical Society.

[148] Müller, E. A. and K. E. Gubbins (2001). Molecular-based equations of state for associating fluids: A review of SAFT and related approaches. *Industrial & Engineering Chemistry Research 40*(10), 2193–2211. doi:10.1021/ie000773w.

[149] Müller, M., Y. Kasaka, D. Müller, R. Schomäcker, and G. Wozny (2013). Process design for the separation of three liquid phases for a continuous hydroformylation process in a miniplant scale. *Industrial & Engineering Chemistry Research 52*(22), 7259–7264. doi:10.1021/ie302487m.

[150] Musko, N. E. (2013). *Heterogeneously Catalysed Chemical Reactions in Carbon Dioxide Medium*. Ph.D. thesis, Technical University of Denmark (DTU).

[151] Naumann, U. (2012). *The Art of Differentiating Computer Programs: An Introduction to Algorithmic Differentiation*. Number 24 in Software, Environments and Tools. Philadelphia: Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611972078.

[152] Oliveira, M. B., V. Ribeiro, A. J. Queimada, and J. A. P. Coutinho (2011). Modeling phase equilibria relevant to biodiesel production: A comparison of g E models, cubic EoS, EoS-g E and association EoS. *Industrial & Engineering Chemistry Research 50*(4), 2348–2358. doi:10.1021/ie1013585.

[153] Orbey, H. and S. I. Sandler (1998). *Modeling Vapor-Liquid Equilibria Cubic Equations of State and Their Mixing Rules*. Cambridge series in Chemical Engineering. Cambridge University Press.

[154] Paduszyński, K. and U. Domańska (2012). Heterosegmented perturbed-chain statistical associating fluid theory as a robust and accurate tool for modeling of various alkanes. 1. Pure fluids. *Industrial & Engineering Chemistry Research 51*(39), 12967–12983. doi:10.1021/ie301998j.

[155] Pantelides, C. C. (1988a). SpeedUp-recent advances in process simulation. *Computers & Chemical Engineering 12*(7), 745–755. doi:10.1016/0098-1354(88)80012-7.

[156] Pantelides, C. C. (1988b). *Symbolic and numerical techniques for the solution of large systems of nonlinear algebraic equations*. Ph.D. thesis, Imperial College of Science and Technology London.

[157] Pantelides, C. C., B. Keeping, J. Bernier, and C. Gautreau (1999). Open interface specification numerical solvers. Technical report co-numr-el 03 version 1.08, CAPE-OPEN.

[158] Peng, D.-Y. and D. B. Robinson (1976). A new two-constant equation of state. *Industrial & Engineering Chemistry Fundamentals 15*(1), 59–64. doi:10.1021/i160057a011.

[159] Pereira, F. E., A. Galindo, G. Jackson, and C. S. Adjiman (2014). On the impact of using volume as an independent variable for the solution of P-T fluid-phase equilibrium with equations of state. *Computers & Chemical Engineering 71*, 67–76. doi:10.1016/j.compchemeng.2014.06.009.

[160] Pereira, F. E., G. Jackson, A. Galindo, and C. S. Adjiman (2010). A duality-based optimisation approach for the reliable solution of (P, T) phase equilibrium in volume-composition space. *Fluid Phase Equilibria 299*(1), 1–23. doi:10.1016/j.fluid.2010.08.001.

[161] Pereira, F. E., G. Jackson, A. Galindo, and C. S. Adjiman (2012). The HELD algorithm for multicomponent, multiphase equilibrium calculations with generic equations of state. *Computers & Chemical Engineering 36*(1), 99–118. doi:10.1016/j.compchemeng.2011.07.009.

[162] Pérez, F. and B. E. Granger (2007). IPython: A system for interactive scientific computing. *Computing in Science and Engineering 9*(3), 21–29. Available from: `http://ipython.org`, doi:10.1109/MCSE.2007.53.

[163] Petera, M. A. (2011). *Automatic Differentiation of the CapeML High-Level Language for Process Engineering*. Ph.D. thesis, RWTH Aachen.

[164] Peters, F. T., F. S. Laube, and G. Sadowski (2012). Development of a group contribution method for polymers within the PC-SAFT model. *Fluid Phase Equilibria 324*, 70–79. doi:10.1016/j.fluid.2012.03.009.

[165] Pfennig, A. (2004). *Thermodynamik der Gemische*. Springer. doi:10.1007/978-3-642-18923-4.

[166] Pfohl, O., S. Petkov, and G. Brunner (2000). "PE" Quickly Makes Available the Newest Equations of State via the Internet. *Industrial & Engineering Chemistry Research 39*(11), 4439–4440. doi:10.1021/ie000778t.

[167] Prausnitz, J. M. (1986). *Molecular thermodynamics of fluid-phase equilibrium* (3rd ed.). Prentice Hall International Series in the Physical and Chemical Engineering Sciences. Prentice Hall.

[168] Preisig, H. A. (2010). Constructing and maintaining proper process models. *Computers & Chemical Engineering 34*(9), 1543–1555. doi:10.1016/j.compchemeng.2010.02.023.

[169] Privat, R., R. Gani, and J. N. Jaubert (2010). Are safe results obtained when the PC-SAFT equation of state is applied to ordinary pure chemicals? *Fluid Phase Equilibria 295*(1), 76–92. doi:10.1016/j.fluid.2010.03.041.

[170] Process Systems Enterprise (2018). gPROMS. Available from: `http://www.psenterprise.com/products/gproms`.

[171] PTC (2016). PTC Mathcad. Available from: `http://www.ptc.com/engineering-math-software/mathcad`.

[172] Reid, R. C., J. M. Prasunitz, and B. E. Poling (1987). *The properties of Gases and Liquids* (5 ed.). McGraw-Hill.

[173] Renon, H. and J. M. Prausnitz (1968). Local compositions in thermodynamic excess functions for liquid mixtures. *AIChE Journal 14*(1), 135–144. doi:10.1002/aic.690140124.

[174] Riisager, A., K. M. Eriksen, P. Wasserscheid, and R. Fehrmann (2003). Propene and 1-octene hydroformylation with silica-supported, ionic liquid-phase (SILP) Rh-phosphine catalysts in continuous fixed-bed mode. *Catalysis Letters 90*(3-4), 149–153. doi:10.1023/B:CATL.0000004109.46005.be.

[175] Rowlinson, J. S. and F. L. Swinton (1982). *Liquids and Liquid Mixtures: Butterworths Monographs in Chemistry* (2nd ed.). Butterworth.

[176] Ryll, O., S. Blagov, and H. Hasse (2012). Convex envelope method for the determination of fluid phase diagrams. *Fluid Phase Equilibria 324*, 108–116. doi:10.1016/j.fluid.2012.04.002.

[177] SageMath Inc. (2016). SageMathCloud. Available from: `https://cloud.sagemath.com`.

[178] Sahlodin, A. M., H. A. J. Watson, and P. I. Barton (2016). Nonsmooth model for dynamic simulation of phase changes. *AIChE Journal 62*(9), 3334–3351.

[179] Sandrock, C., P. L. de Vaal, and Jacek Jezowski and Jan Thullie (2009). Dynamic simulation of Chemical Engineering systems using OpenModelica and CAPE-OPEN. In J. Jeżowski and J. Thullie (Eds.), *19th European Symposium on Computer Aided Process Engineering - ESCAPE19*, Volume 26 of *Computer Aided Chemical Engineering*, pp. 859–864. Elsevier B.V.

[180] Sauer, E., M. Stavrou, and J. Gross (2014). Comparison between a homo- and a heterosegmented group contribution approach based on the perturbed-chain polar statistical associating fluid theory equation of state. *Industrial & Engineering Chemistry Research 53*(38), 14854–14864. doi:10.1021/ie502203w.

[181] Schaefer, E., Y. Brunsch, G. Sadowski, and A. Behr (2012). Hydroformylation of 1 - Dodecene in the Thermomorphic Solvent System Dimethylformamide / Decane . Phase Behavior - Reaction Performance - Catalyst Recycling. *Industrial & Engineering Chemistry Research 51*, 10296–10306.

[182] Schäfer, E. (2013). *Simultaneous Calculation of Fluid Phase Equilibria and Interfacial Properties of Aprotic Solvent Mixtures Incorporating Experimental Studies.* Schriftenreihe Thermodynamik. Dr. Hut.

[183] Schäfer, E. and G. Sadowski (2012). Liquid-Liquid Equilibria of Systems with Linear Aldehydes. Experimental Data and Modeling with PCP-SAFT. *Industrial & Engineering Chemistry Research 51*(44), 14525–14534. doi:10.1021/ie301566d.

[184] Schmitz, M., R. Hannemann-Tamas, B. Gendler, M. Förster, W. Marquardt, and U. Naumann (2012). Software for Higher-order Sensitivity Analysis of Parametric DAEs. *SNE Simulation Notes Europe 22*(3-4), 163–168. doi:10.11128/sne.22.tn.10151.

[185] Schopfer, G. (2006). *A Framework for Tool Integration in Chemical Process Modeling.* Fortschritt-Berichte VDI: Reihe 3, Verfahrenstechnik, Bericht aus dem Lehrstuhl für Prozesstechnik, RWTH Aachen,868. VDI-Verlag.

[186] Schopfer, G., J. Wyes, W. Marquardt, and L. von Wedel (2005). A library for equation system processing based on the CAPE-OPEN ESO Interface. In L. Puigjaner and A. Espuña (Eds.), *European Symposium on Computer-Aided Process Engineering-15, 38th European Symposium of the Working Party on Computer Aided Process Engineering*, Volume 20 of *Computer-Aided Chemical Engineering*, Barcelona, Spain, pp. 1573–1578. Elsevier B.V.

[187] Selvaag, K. (2013). *Symbolic Differentiation of Multivariable Functions to Arbitrary Order.* Master's thesis, Norwegian University of Science and Technology.

[188] Shaharun, M. S., B. K. Dutta, H. Mukhtar, and S. Maitra (2010). Hydroformylation of 1-octene using rhodium-phosphite catalyst in a thermomorphic solvent system. *Chemical Engineering Science 65*(1), 273–281. doi:10.1016/j.ces.2009.06.071.

[189] Sharma, A., C. J. Lebigue, R. M. Deshpande, A. a. Kelkar, and H. Delmas (2010). Hydroformylation of 1-octene using [Bmim][PF6]-decane biphasic media and rhodium complex catalyst: Thermodynamic properties and kinetic study. *Industrial & Engineering Chemistry Research 49*(21), 10698–10706. doi:10.1021/ie100222k.

[190] Silva, A. S. and M. Castier (1993). Automatic differentiation and implementation of thermodynamic models using a computer algebra system. *Computers & Chemical Engineering 17*(Supplement 1), 473–478.

[191] Slawig, T. (2005). *Algorithmisches Differenzieren in Simulation und Optimalsteuerung von Differentialgleichungen.* Habilitationsschrift, Technische Universität Berlin.

[192] Smith, J. M., H. C. Van Ness, and M. M. Abbott (2005). *Introduction to Chemical Engineering Thermodynamics* (7 ed.), Volume 27. McGraw-Hill Education. doi:10.1021/ed027p584.3.

[193] Soave, G. (1972). Equilibrium constants from a modified Redlich-Kwong equation of state. *Chemical Engineering Science 27*(6), 1197–1203. doi:10.1016/0009-2509(72)80096-4.

[194] Tassios, D. (1993). *Applied Chemical Engineering Thermodynamics*. Springer.

[195] Taylor, R. (1997). Automatic derivation of thermodynamic property functions using computer algebra. *Fluid Phase Equilibria 129*, 37–47.

[196] Teague, T. (2002). PlantData XML. In B. L. Braunschweig and R. Gani (Eds.), *Software Architectures and Tools for Computer Aided Process Engineering*, Volume 11 of *Computer Aided Chemical Engineering*, Chapter 4.2, pp. 267–291. Elsevier B.V.

[197] Teh, Y. and G. Rangaiah (2002). A Study of Equation-Solving and Gibbs Free Energy Minimization Methods for Phase Equilibrium Calculations. *Chemical Engineering Research and Design 80*(7), 745–759. doi:10.1205/026387602320776821.

[198] Tessier, S. R., J. F. Brennecke, and M. A. Stadtherr (2000). Reliable phase stability analysis for excess Gibbs energy models. *Chemical Engineering Science 55*(10), 1785–1796. doi:10.1016/S0009-2509(99)00442-X.

[199] The CAPE-OPEN Laboratories Network (2016). The CAPE-OPEN Laboratories Network. Available from: `www.colan.org`.

[200] Thiéry, R. (1996). A new object-oriented library for calculating high-order multivariable derivatives and thermodynamic properties of fluids with equations of state. *Computers & Geosciences 22*(7), 801–815.

[201] Tolsma, J. E. (1999). *Analysis of Heteroazeotropic Systems*. Ph.D. thesis, Massachusets Institute of Technology.

[202] Tolsma, J. E. and P. I. Barton (1998). On computational differentiation. *Computers & Chemical Engineering 22*(4-5), 475–490. doi:10.1016/S0098-1354(97)00264-0.

[203] Tolsma, J. E. and P. I. Barton (2002). Numerical Solvers. In B. Braunschweig and R. Gani (Eds.), *Software Architectures and Tools for Computer Aided Process Engineering*, Volume 11 of *Computer Aided Chemical Engineering*, Chapter 3.2, pp. 127–164. Elsevier B.V.

[204] Tolsma, J. E., J. a. Clabaugh, and P. I. Barton (2002). Symbolic Incorporation of External Procedures into Process Modeling Environments. *Industrial & Engineering Chemistry Research 41*(16), 3867–3876. doi:10.1021/ie0107946.

[205] Topliss, R. J. (1985). *Techniques to Facilitate the Use of Equations of State for Complex Fluid-Phase Equilibria*. Ph.D. thesis, University of California, Berkeley.

[206] Topliss, R. J., D. Dimitrelis, and J. M. Prausnitz (1988). Computational aspects of a non-cubic equation of state for phase-equilibrium calculations. Effect of density-dependent mixing rules. *Computers & Chemical Engineering 12*(5), 483–489. doi:10.1016/0098-1354(88)85067-1.

[207] Trapp, C., F. Casella, T. V. D. Stelt, and P. Colonna (2014). Use of External Fluid Property Code in Modelica for Modelling of a Pre-combustion CO2 Capture Process Involving Multi-Component, Two-Phase Fluids. In *Proceedings of the 10th International Modelica Conference*, Lund, Sweden, pp. 1047–1056. doi:10.3384/ecp140961047.

[208] Tumakaka, F. (2004). *Modellierung von Phasengleichgewichten in Copolymer-Systemen mit polaren und assoziierenden Monomeren.* Shaker Verlag.

[209] Tumakaka, F., J. Gross, and G. Sadowski (2005). Thermodynamic modeling of complex systems using PC-SAFT. *Fluid Phase Equilibria 228-229*, 89–98. doi:10.1016/j.fluid.2004.09.037.

[210] Valderrama, J. O. (2003). The State of the Cubic Equations of State. *Industrial & Engineering Chemistry Research 42*(8), 1603–1618. doi:10.1021/ie020447b.

[211] Van Baten, J. and M. Pons (2014). Cape-open: Interoperability in industrial flowsheet simulation software. *Chemie Ingenieur Technik 86*(7), 1052–1064. doi:10.1002/cite.201400009.

[212] Vetukuri, S. R. R., L. T. Biegler, and A. Walther (2010). An Inexact Trust-Region Algorithm for the Optimization of Periodic Adsorption Processes. *Industrial & Engineering Chemistry Research 49*(23), 12004–12013. doi:10.1021/ie100706c.

[213] Vogelpohl, C., C. Brandenbusch, and G. Sadowski (2013). High-pressure gas solubility in multicomponent solvent systems for hydroformylation. Part I: Carbon monoxide solubility. *Journal of Supercritical Fluids 81*, 23–32. doi:10.1016/j.supflu.2013.04.006.

[214] Vogelpohl, C., C. Brandenbusch, and G. Sadowski (2014). High-pressure gas solubility in multicomponent solvent systems for hydroformylation. Part II: Syngas solubility. *Journal of Supercritical Fluids 88*, 74–84. doi:10.1016/j.supflu.2014.01.017.

[215] von Solms, N., M. L. Michelsen, and G. M. Kontogeorgis (2003). Equation of State for Highly Asymmetric and Associating Mixtures. *Industrial & Engineering Chemistry Research 42*(5), 1098–1105.

[216] von Solms, N., M. L. Michelsen, and G. M. Kontogeorgis (2004). Applying Association Theories to Polar Fluids. *Industrial & Engineering Chemistry Research 43*(7), 1803–1806. doi:10.1021/ie034243m.

[217] von Wedel, L. (2002). CapeML - A model exchange language for chemical process modeling. Tech report lpt-2002-16, Lehrstuhl für Prozesstechnik, RWTH Aachen University.

[218] von Wedel, L. (2008). Neuerungen und Erfahrungen aus der Anwendung des CAPE-OPEN Standards. *Chemie Ingenieur Technik 80*(1-2), 119–126. doi:10.1002/cite.200700146.

[219] von Wedel, L., W. Marquardt, and R. Gani (2002). Modelling frameworks. In B. L. Braunschweig and R. Gani (Eds.), *Software Architectures and Tools for Computer Aided Process Engineering*, Volume 11 of *Computer Aided Chemical Engineering*, Chapter 3.1, pp. 89–125. Elsevier B.V.

[220] Walther, A. and A. Griewank (2012). Getting started with ADOL-C. In U. Naumann and O. Schenk (Eds.), *Combinatorial Scientific Computing*, pp. 181–202. Chapman-Hall CRC Computational Science.

[221] Wasylkiewicz, S. K., Y. K. Li, M. A. Satyro, and M. J. Wasylkiewicz (2013). Application of a global optimization algorithm to phase stability and liquid-liquid equilibrium calculations. *Fluid Phase Equilibria 358*, 304–318. doi:10.1016/j.fluid.2013.08.030.

[222] Willkomm, J., C. H. Bischof, and H. M. Bücker (2011). Accurate Derivatives for Computational Engineering with Automatic Differentiation for Matlab (ADiMat). Available from: `http://www.sc.informatik.tu-darmstadt.de/media/fg_bischof/material_2/adimat/willkomm-handout-icce2011.pdf`.

[223] Wozny, G. and H. Wendeler (2006). Methoden und Werkzeuge der Simulationstechnik. In R. Goedecke (Ed.), *Fluidverfahrenstechnik*, Chapter 2.4, pp. 86–165. Wiley-VCH.

[224] Xu, G., J. F. Brennecke, and M. a. Stadtherr (2002). Reliable Computation of Phase Stability and Equilibrium from the SAFT Equation of State. *Industrial & Engineering Chemistry Research 41*(May), 938–952. doi:10.1021/ie0101801.

[225] Ye, K., H. Freund, Z. Xie, B. Subramaniam, and K. Sundmacher (2012). Prediction of multicomponent phase behavior of CO2-expanded liquids using CEoS/GE models and comparison with experimental data. *The Journal of Supercritical Fluids 67*, 41–52. doi:10.1016/j.supflu.2012.03.007.

[226] Zagajewski, M., A. Behr, P. Sasse, and J. Wittmann (2014). Continuously operated miniplant for the rhodium catalyzed hydroformylation of 1-dodecene in a thermomorphic multicomponent solvent system (TMS). *Chemical Engineering Science 115*, 88–94. doi:10.1016/j.ces.2013.09.033.

[227] Zagajewski, M., J. Dreimann, and A. Behr (2014). Verfahrensentwicklung vom Labor zur Miniplant: Hydroformylierung von 1-Dodecen in thermomorphen Lösungsmittelsystemen. *Chemie Ingenieur Technik 86*(4), 449–457. doi:10.1002/cite.201300147.

[228] Zerry, R. (2008). *MOSAIC, eine webbasierte Modellierungs- und Simulationsumgebung für die Verfahrenstechnik*. Shaker Verlag.

[229] Zerry, R., B. Gauss, L. Urbas, and G. Wozny (2004). Web-based object oriented modelling and simulation using mathml. In A. Barbosa-Povoa and H. Matos (Eds.), *European Symposium on Computer-Aided Process Engineering-14, 37th European Symposium of the Working Party on Computer-Aided Process Engineering*, Volume 18 of *Computer Aided Chemical Engineering*, Lisbon, Portugal, pp. 1171–1176. Elsevier B.V.

[230] Zhang, H., A. Bonilla-Petriciolet, and G. P. Rangaiah (2011). A review on global optimization methods for phase equilibrium modeling and calculations. *The open Thermodynamics Journal 5*(S1), 71–92.

[231] Zhao, Y., C. Jiang, and A. Yang (2012). Towards computer-aided multiscale modelling: A generic supporting environment for model realization and execution. *Computers & Chemical Engineering 40*, 45–57. doi:10.1016/j.compchemeng.2012.02.012.