Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

Henrik Lia

Computational Aspects of the Two-phase Isothermal Flash

Master's thesis in MTPROD
Supervisor: Even Solbraa
June 2022

NTNU
Norwegian University of
Science and Technology

Henrik Lia

Computational Aspects of the Two-phase Isothermal Flash

Master's thesis in MTPROD
Supervisor: Even Solbraa
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

NTNU
Norwegian University of
Science and Technology

SUMMARY

Ever since the introduction of compositional modeling using equations of state in the 1950s and 1960s, the two-phase isothermal flash calculation has been essential for predicting the phase behavior of petroleum and other multi-component systems. Methods for automating the procedure and decreasing the amount of time have been essential in making compositional calculations feasible in a commercial setting.

The cubic equation of state model has proven to be both accurate and simple enough to be used to describe most traditional petroleum fluids. The theory of cubic equation of state calculations has been well documented in the literature. However, the numerical aspects associated with finite-precision mathematics in modern computers have not been emphasized to the same extent.

The flash calculation is a complex system of equations that are very sensitive to the choice of initial estimates and solution approaches. There is a wide range of heuristics and *rule of thumbs* in the literature that attempts to make the calculation as robust as possible, while still making the calculations as fast as possible. The purpose of this thesis is to (1) compare different methodologies provided in the literature, (2) give a workflow of how to efficiently implement the given methodologies from a computational perspective, and (3) develop novel methodologies where severe numerical errors occur.

First a novel, fast, and robust algorithm for determining the roots of cubic equations of state was developed, that utilizes previously calculated roots to enhance the initial estimates. Then a novel method is described that uses the theory of finite-precision mathematics to avoid severe round-off errors in the solution of the material balance, and guarantees that a numerically accurate solution is found.

A variety of methods for updating the estimated K-values have been compared in this thesis. The methods are divided into (1) accelerated successive substitution methods, and (2) second-order methods. A comparison was made between the standard successive substitution, the Wegstein accelerated method, and two dominant eigenvalue-based

methods. Key points of investigation in this work are computational speed and the ability of the different methods to converge to the correct solution. A second-order multivariate Newton-Raphson method was investigated and compared to the accelerated successive substitution methods.

When developing performance code, the most important aspect is to improve the segments of the code that have the largest impact on performance. A framework for profiling and quantifying performance in a consistent manner was developed using this as the guiding principle.

SAMMENDRAG

Helt siden introduksjonen av komposisjonell modellering med bruk av kubiske tilstandslikninger på 1950- og 1960-tallet, har den to-fase isotermiske flash-beregningen vært essensielt for å predikere faseoppførsel for petroleumsystemer og andre flerkomponent systemer. Metoder for å automatisere prosedyren og redusere tidsbruken, har vært viktig i å bidra til at komposisjonelle beregninger har blitt mulig å bruke i kommersielle sammenhenger.

Kubiske tilstandslikninger har vist seg å være nøyaktig og enkle nok til å bli brukt for å beskrive de fleste tradisjonelle petroleumsfluider. Teorien bak beregninger ved bruk av kubiske tilstandslikninger er veldokumentert i litteraturen, men de numeriske aspektene assosiert med å utføre matematikk med begrenset presisjon har ikke blitt like godt vektlagt.

Flash-beregningen er et komplekst system med likninger som er veldig sensitive til valget av initiale estimater og løsningstilnærminger. Det er mange strategier og tommelfingerregler i litteraturen som prøver å gjøre beregninger så robust som mulig, men som fortsatt gjør beregningene så fort som mulig. Målet ved denne oppgaven er (1) å sammenlikne forskjellige løsningsmetoder funnet i litteraturen, (2) beskrive en metode for hvordan å effektivt implementere de gitte metodene fra et beregningsperspektiv og (3) utvikle nye metoder hvor alvorlige avrundingsfeil skjer.

En ny, rask og robust algoritme for å beregne røttene til en kubisk tilstandslikning har blitt utviklet, som anvender tidligere beregnede røtter for å forbedre initialestimater. En ny algoritme er beskrevet, som bruker flyttallsmatematikk for å unngå alvorlige avrundingsfeil i beregninger av materialbalansen og garanterer at en numerisk nøyaktig løsning er funnet.

Flere metoder for å oppdatere K-verdiestimater har blitt sammenliknet i denne oppgaven. Metodene er delt inn i (1) akselererte suksessiv substutisjonsmetoder og (2) andreordens metoder. En sammenlikning mellom suksessiv substutisjonsmetoden, Wegsteins akselererte

metode og to dominant egenverdibaserte metoder har blitt gjort. De viktigste punktene i sammenlikningen er beregningstid og konvergens til riktig løsning. En andreordens multivariabel Newton-Raphson metode har blitt implementert og sammenliknet med de akselererte suksessiv substutisjonsmetodene.

Når man utvikler ytelseskritiske beregningsprogrammer er det viktig å optimalisere de mest ytelseskritiske segmentene av programmet. En metode for å profilere og kvantifisere ytelse på et konsistent vis har blitt utviklet med dette som hovedprinsipp.

ACKNOWLEDGEMENTS

First and foremost I would like to thank my great friend and colleague at Whitson AS, Markus Hays Nielsen. He invited me to attend the course *Advanced PVT and EOS Fluid Characterization* in the summer of 2021, where I was introduced to the topics of PVT and computational thermodynamics. In the last year, Markus has been my closest collaborator and I would like to thank him for all our interesting discussions and all his help on my thesis work.

A special thanks goes out to my supervisor Even Solbraa for all his help and for showing such a great interest in the various topics related to my thesis work.

I would also like to thank all my great colleagues at Whitson AS, where I have been working part-time for the last two years. Even though I started as a pure software engineer, I have been allowed to work more and more on topics related to thermodynamical engineering, which have culminated in to this thesis. I am very grateful for my time as a part-time employee at Whitson AS and I could not be more excited about my upcoming full-time job there.

Further, I would like to acknowledge Aaron Zick of Zick Technologies, for developing and providing a license to the excellent PVT software PhazeComp.

Finally, I also want to thank my family and friends for helping and supporting me through my 5 years at NTNU. Without them, I would not be the person I am today.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The main objective of this thesis is to investigate the computational aspects of the two-phase isothermal flash calculation and determine the impact of different solution approaches and their implementation on (1) performance with regards to computational time, and (2) the ability for the calculation to converge to the correct solution. This work was initiated through a research project proposed by Whitson AS as part of their academic and commercial work related to phase behavior.

This work aims to aluminate issues related to the implementation of the two-phase isothermal flash calculation that is typically not discussed or disclosed in academic literature. These topics include (1) issues related to finite-precision mathematics such as severe round-off error, (2) modern commercial computer architecture, (3) choice of programming languages, and (4) workflows related to comparing and enhancing different implementations through profiling and benchmarking.

Chapter 2 presents the theory used in the following chapters. The first section covers the phase behavior fundamentals for single- and multi-component systems and calculational aspects related to cubic equations of state. The second section covers the fundamentals of floating-point arithmetic related to round-off errors and catastrophic cancellation. Finally, modern computer architecture is described and highlights of the effect it can have on the performance of the flash calculation are given.

Chapter 3 presents the methodology used in the implementation of different segments of the flash calculation. The first section covers the single-phase density calculation, followed by a section on the vapor-liquid equilibrium calculation. This section covers K-value initial estimates, the material balance solution, phase property calculations and K-value update methods. Finally, data generation and performance testing are described.

Chapter 4 presents the results of the various methods presented in Chapter 3. The different sections present the results of and compare the different methods used for (1) determining

roots of the cubic equation of state, (2) solving the Rachford-Rice equation, (3) updating K-values using accelerated successive substitution methods, and (3) updating K-values using the Newton-Raphson method.

Chapter 5 presents the conclusion based on the results and discussion presented in Chapter 4, followed by the proposed items for further work given in Chapter 6. Appendix A derives a simplification for the calculation of finding the minimum Gibbs energy for multiple roots. Appendix B presents the different equation of state models used in this work and Appendix C presents the different compositions and their respective phase envelopes.

CHAPTER 2

THEORY

2.1 Equation of State Calculations

2.1.1 Equations of State

EOS models commonly relates pressure (p), volume (V), temperature (T), and molar composition (z) of a fluid. The simplest and perhaps most familiar EOS model is the ideal gas law[1], given by

$$pv = RT, \tag{2.1}$$

where R is the universal gas constant, and v is the molar volume defined as volume divided by molar amount, i.e. $v = V/n$. The ideal gas law is inaccurate for gases at high pressures and low temperatures. A quantity for calculating the deviation from the ideal gas law is introduced, called the Z-factor, and is defined by

$$Z = \frac{pv}{RT}. \tag{2.2}$$

Equation (2.2) is called the real gas law. A chart of Z-factors was developed by Standing and Katz[2] that can be used for predicting the volumetric behavior of natural gases based on the corresponding states principle[3]. The chart shows the Z-factor plotted versus reduced temperature (T_r) and reduced pressure (p_r), which are defined by

$$T_r = \frac{T}{T_c}, \tag{2.3a}$$

$$p_r = \frac{p}{p_c}, \tag{2.3b}$$

where T_c and p_c are the temperature and pressure at the critical point of the fluid. If the gas is a mixture of components with different critical properties, pseudo-critical temperature

and pressure are used instead. One possible method to estimate the pseudo-critical conditions is using a linear mixing rule, given by

$$T_{pr} = \sum_{i=1}^{N_c} z_i T_{ci}, \tag{2.4a}$$

$$p_{pr} = \sum_{i=1}^{N_c} z_i p_{ci}, \tag{2.4b}$$

$$\tag{2.4c}$$

where N_c is the number of components and z_i is molar fraction of each component. Several numerical approximations of the Standing-Katz chart have been developed, such as the method described by Hall and Yarborough[4], which uses the Carnahan-Starling hard sphere EOS. Even though the Standing-Katz chart is highly accurate for natural gases, it can not describe liquid phases or multiphase behavior, which means that a more complex EOS model is needed. A popular choice is the cubic EOS model.

2.1.2 Cubic Equations of State

A cubic equations of state is a model that can be expressed in the following way

$$Z^3 + A_2 Z^2 + A_1 Z + A_0 = 0. \tag{2.5}$$

The two most widely used cubic EOS models in petroleum engineering are the Soave-Redlich-Kwong[5] (SRK) and Peng-Robinson[6] (PR) equations[3].

The SRK EOS is given by

$$p = \frac{RT}{v-b} - \frac{a}{v(v+b)}, \tag{2.6}$$

where the terms a and b are defined as

$$a = \Omega_a \frac{R^2 T_c^2}{p_c} \alpha(T_r), \tag{2.7a}$$

$$b = \Omega_b \frac{RT_c}{p_c}, \tag{2.7b}$$

$$\alpha(T) = [1 + m(\omega)(1 - \sqrt{T_r})]^2, \tag{2.7c}$$

where the coefficients Ω_A and Ω_B are listed in Table 2.1 and ω is the acentric factor. The function $m(\omega)$ for the SRK EOS is given by

$$m(\omega)_{SRK} = 0.480 + 1.574\omega - 0.176\omega^2. \tag{2.8}$$

The PR EOS is given by

$$p = \frac{RT}{v-b} - \frac{a}{v(v+b) + b(v-b)}, \tag{2.9}$$

where the terms a and b are also defined by Equation (2.7), but with different values for the coefficients Ω_A and Ω_B. The function $m(\omega)$ for the PR EOS is defined as

$$m(\omega)_{PR} = \begin{cases} 0.37464 + 1.54226\omega - 0.26992\omega^2, & \text{if } \omega \leq 0.49 \\ 0.3796 + 1.485\omega - 0.1644\omega^2 + 0.01667\omega^3, & \text{else} \end{cases} \tag{2.10}$$

Both the SRK and the PR EOS models can be written on a general form given by

$$p = \frac{RT}{v-b} - \frac{a}{(v+\delta_1 b)(v+\delta_2 b)}, \tag{2.11}$$

where δ_1 and δ_2 are different for the SRK and PR models and are listed in Table 2.1. Equation (2.11) can be expressed in the form of Equation (2.5) with

$$A_0 = -\delta_1\delta_2 B^3 - \delta_1\delta_2 B^2 - AB, \tag{2.12a}$$
$$A_1 = (\delta_1\delta_2 - \delta_1 - \delta_2)B^2 - (\delta_1 - \delta_2)B + A, \tag{2.12b}$$
$$A_2 = (\delta_1 + \delta_2 - 1)B - 1, \tag{2.12c}$$

where A and B are the dimensionless versions of the parameters a and b defined as

$$A = \frac{ap}{R^2T^2}, \tag{2.13a}$$
$$B = \frac{bp}{RT}. \tag{2.13b}$$

Table 2.1: EOS constants for the SRK and PR models.

	SRK	PR
δ_1	1	$1+\sqrt{2}$
δ_2	0	$1-\sqrt{2}$
Ω_a	0.42748	0.45724
Ω_b	0.08664	0.07780

2.1.3 Volume Shifts

Peneloux et al. introduced the concept of volume shifts in 1982[7] to improve the accuracy of the predicted volumetric data of cubic EOS models. The volume shift (c) is defined as

$$v = v^{EOS} - c, \tag{2.14}$$

and is often represented by the unitless parameter s, defined as $s = c/b$. A major advantage of the volume shifts is that it will not influence the results of vapor-liquid equilibrium calculations, which are typically highly accurately predicted by both the PR and SRK EOS models for petroleum systems.

2.1.4 Gibbs Energy and Equilibrium Conditions

Calculations using cubic equations of state uses the assumption that the fluid is at equilibrium. When the temperature and pressure of a fluid system are known, the equilibrium conditions are determined by minimizing the Gibbs energy (G)[8]. The Gibbs energy is a state function given by

$$G = \sum_{i=1}^{N_c} \mu_i n_i, \tag{2.15}$$

where n_i is the molar amounts of each component and μ_i is the chemical potential of the different components. When multiple phases are present in a fluid system, the total Gibbs energy is the sum of each phase's Gibbs energy. If two phases are assumed to exist at equilibrium, it can be shown that the equilibrium condition is given by[8]

$$n_i^V \mu_i^V = n_i^L \mu_i^L, \tag{2.16}$$

where the superscripts V and L denote the vapor and liquid phases respectively. The cubic equations of state model the chemical potential in the form

$$\mu_i = RT \ln f_i + \lambda(T), \tag{2.17}$$

where f_i is the component-fugacity and $\lambda(T)$ is a term that is constant for each phase at the same temperature. Using Equation (2.17) it can be shown that the equal chemical potential constraint given by Equation (2.16) reduces to

$$n_i^V f_i^V = n_i^L f_i^L. \tag{2.18}$$

2.1.5 Single-Component Systems

For single-component fluids, the expression for fugacity is obtained using the relation

$$\ln \phi = \frac{1}{RT} \int_V^\infty \left(\left(\frac{dP}{dn_i} \right)_{T,V} - \frac{RT}{V} \right) dV - \ln Z, \tag{2.19}$$

where ϕ is called the fugacity coefficient and is defined as $\phi = \frac{f}{p}$ for single component fluids. Applying Equation (2.19) to a cubic EOS with only one component results in the following expression

$$\ln \phi = Z - 1 - \ln(Z - B) - \frac{1}{\delta_1 - \delta_2} \cdot \frac{A}{B} \ln \left(\frac{Z + \delta_1 B}{Z + \delta_2 B} \right). \tag{2.20}$$

If a pressure and temperature are specified, the parameters A and B can be calculated, and the Z-factor can be obtained by solving Equation (2.5). If the specified pressure and temperature are sub-critical, Equation (2.5) can have three solutions. When this happens,

the middle root is discarded as it is unphysical, and the Z-factor with the lowest Gibbs is chosen as the correct solution. Choosing the largest Z-factor corresponds to having a vapor-like fluid, while the lowest Z-factor corresponds to having a liquid-like fluid.

If both roots have the same Gibbs energy, they are both valid, which means the fluid will split into one liquid-like phase and one vapor-like phase. When this occurs, the specified pressure is said to be the vapor pressure at that given temperature. An example of how the vapor pressure changes with temperature can be seen in Figure 2.1.

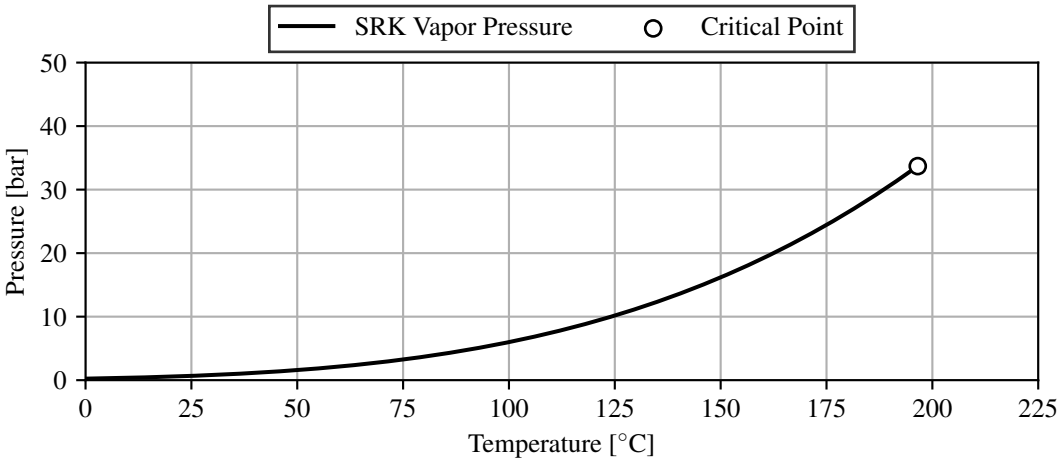


Figure 2.1: Vapor pressure curve for normal pentane using the SRK EOS.

2.1.6 Multi-Component Systems

The transition from a liquid-like to a vapor-like phase for mixture-fluids will generally not occur at only one pressure but gradually over a pressure range. Given a constant temperature, the first pressure at which liquid will form when the pressure increases is called the dewpoint pressure, and the last pressure where vapor is observed is called the bubblepoint pressure. An example of how the dewpoint and bubblepoint pressures changes with temperature, often called a phase envelope, can be seen in Figure 2.2. At a specific temperature, the bubblepoint and dewpoint pressures converge into a critical point, where the properties of both phases converge to the same value. Only dewpoint pressures exist at temperatures greater than the critical temperature, which means that a liquid phase will form and disappear again as the pressure decreases. This phenomenon is called retrograde condensation and can seem somewhat unintuitive.

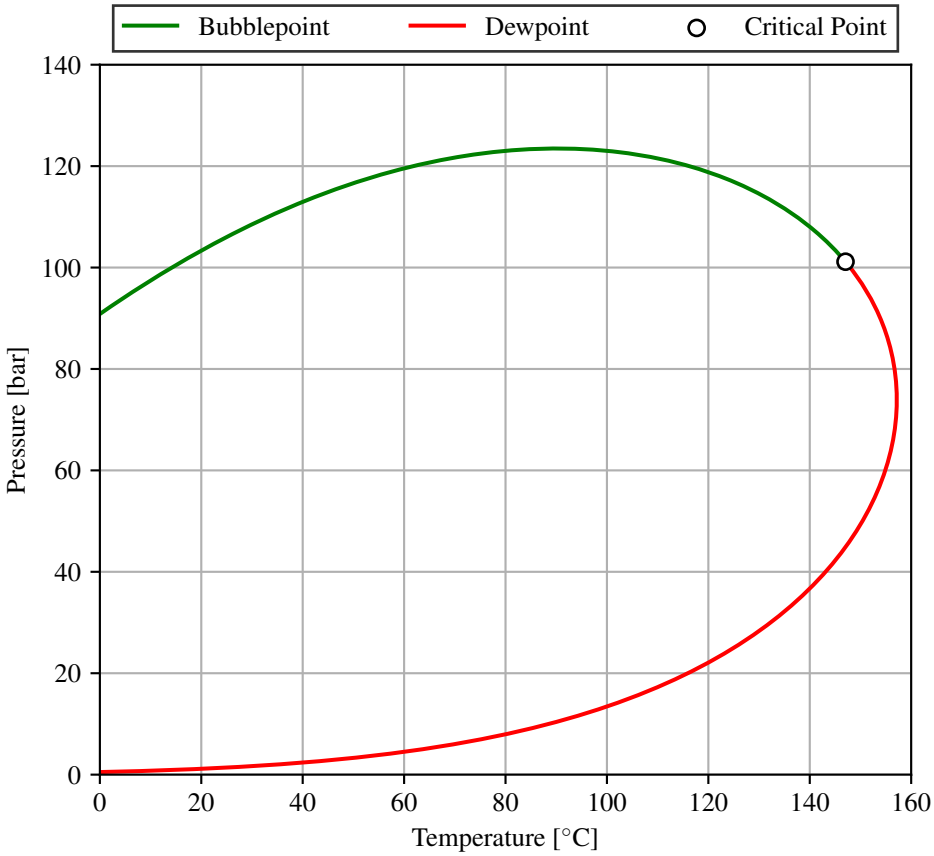


Figure 2.2: Phase envelope of a binary mixture containing equal molar amounts of methane and normal pentane using the PR EOS.

Calculating the fluid properties outside of the two-phase region enclosed by the bubblepoint pressures and the dewpoint pressures is similar to the description of single-component fluids, except that a set of mixing rules are used to get the average parameters A and B for the mixture, given by

$$A = \sum_{i=1}^{N_c} \sum_{j=1}^{N_c} z_i z_j A_{ij}, \tag{2.21a}$$

$$B = \sum_{i=1}^{N_c} z_i B_i, \tag{2.21b}$$

where $A_{ij} = \sqrt{A_i A_j}(1 - k_{ij})$. k_{ij} is a correction parameter often called binary interaction parameters (BIPs). If multiple roots are found, the root with the lowest Gibbs energy is chosen. To calculate the Gibbs energy of a mixture, the component-specific fugacity coefficients are needed, which for the EOS models given by Equation (2.11) Coats[9]

summarized to be

$$\ln \phi_i = \ln \frac{f_i}{u_i p} = \frac{B_i}{B}(Z-1) - \ln(Z-B) + \frac{1}{\delta_1 - \delta_2}\frac{A}{B}\left(\frac{B_i}{B} - \frac{2}{A}\sum_{j=1}^{N_c} u_j A_{ij}\right) \ln\left(\frac{Z+\delta_1 B}{Z+\delta_2 B}\right). \tag{2.22}$$

The volume shifts are calculated using a linear mixing rule, i.e.

$$v = v^{EOS} - \sum_{i=1}^{N_c} z_i c_i. \tag{2.23}$$

Inside the two-phase region, the liquid and vapor phase compositions need to be determined before the fluid properties can be calculated. This calculation is called the two-phase isothermal *flash calculation*.

2.1.7 The Flash Calculation

The two-phase isothermal flash calculation is said to be the most important equilibrium calculation[8]. The calculation inputs are pressure, temperature, the overall composition, and the EOS parameters. The outputs are the equilibrium molar compositions of the two phases and the molar vapor and liquid fractions, which are defined by

$$V = \frac{n_V}{n_V + n_L}, \tag{2.24a}$$

$$L = \frac{n_L}{n_V + n_L}, \tag{2.24b}$$

where n_V and n_L are the molar amounts of the vapor and liquid phases, respectively.

The flash calculation is constrained by equal fugacity as defined in Equation (2.18), and a component material balance. Given an equilibrium vapor composition (y_i) and an equilibrium liquid composition (x_i), the following holds

$$\sum_{i=1}^{N_c} y_i = \sum_{i=1}^{N_c} x_i = 1. \tag{2.25}$$

which can be rearranged to

$$\sum_{i=1}^{N_c} (y_i - x_i) = 0. \tag{2.26}$$

From the material balance $z_i = V y_i + L x_i$ and $L = 1 - V$, the following relationship holds

$$z_i = V y_i + (1-V)\frac{y_i}{Ki} \tag{2.27a}$$

$$z_i = V K_i x_i + (1-V) x_i, \tag{2.27b}$$

where K_i is the equilibrium ratios or K-values, defined by

$$K_i = \frac{y_i}{x_i}. \tag{2.28}$$

These equations can be rearranged to

$$x_i = \frac{z_i}{1 + V(K_i - 1)}. \tag{2.29a}$$

$$y_i = \frac{z_i K_i}{1 + V(K_i - 1)} = K_i x_i, \tag{2.29b}$$

Inserting Equations 2.29a and 2.29b into Equation (2.26) results in the Rachford-Rice equation, given by

$$h(V) = \sum_{i=1}^{N_c} \frac{z_i(K_i - 1)}{1 + V(K_i - 1)}, \tag{2.30}$$

where the system satisfies all constraints if $h(V) = 0$.

If initial estimates of the K-values are given, the Rachford-Rice equation is solved, and the fugacity coefficients of each component for both phases can be calculated using Equation (2.22). If the equal fugacity constraint is not satisfied, the K-values are set to $K_i = \phi_i^L / \phi_i^V$, and the same procedure is repeated until convergence.

2.1.8 Negative Flash

In 1989 Whitson and Michelsen showed that it is possible to perform flash calculations outside the two-phase region, called a *negative flash* calculation[10]. A negative flash calculation corresponds to finding a saddle point in the Gibbs energy surface and will always yield a vapor fraction below zero or above one. However, the equilibrium phase compositions are non-negative, and the material balance and equal fugacity constraints are still satisfied.

Instead of being bounded by the two-phase region, the negative flash is bounded by the converge envelope, which can be seen in figure Figure 2.3. The convergence envelope traces the line where the flash calculation yields a solution where $K_i = 1$, and it can be seen in figure Figure 2.3 that the critical point is a special case of the convergence envelope where it coincides with the phase envelope.

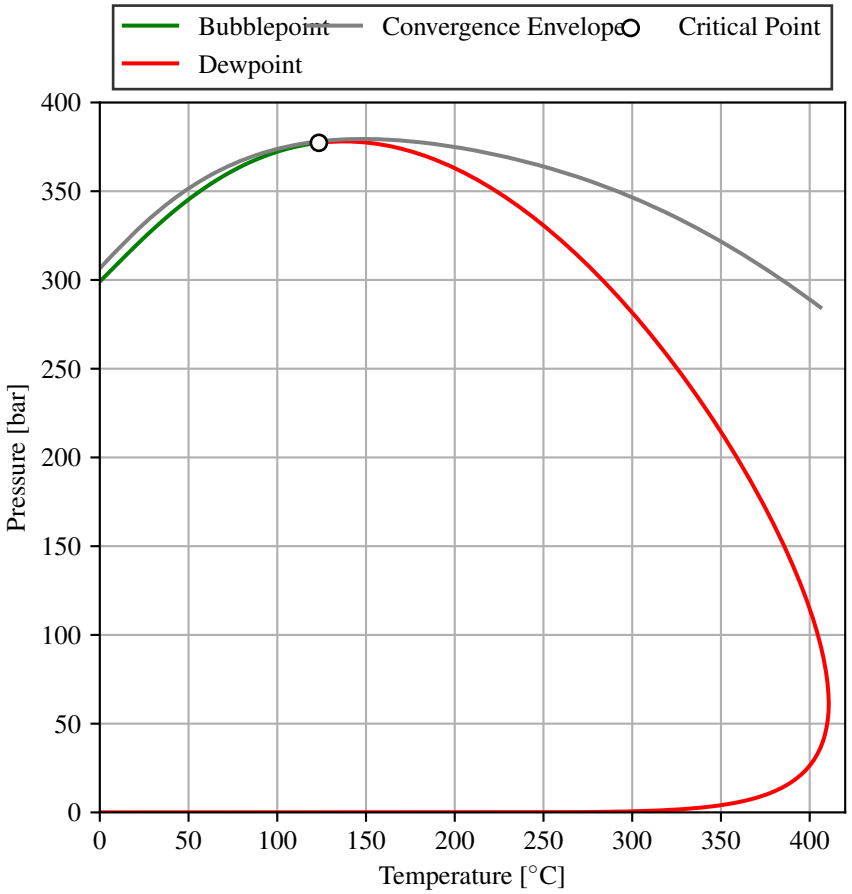


Figure 2.3: An example of a convergence envelope and a phase envelope.

Using the negative flash, Whitson and Michelsen showed that the theoretical bounds for the vapor fraction are given by[10]

$$V_{\text{min}} = \frac{1}{1 - K_{\text{max}}}, \tag{2.31a}$$

$$V_{\text{max}} = \frac{1}{1 - K_{\text{min}}}. \tag{2.31b}$$

Using these bounds, a solution to the Rachford-Rice equation is guaranteed to exist if $K_{\text{min}} < 1$ and $K_{\text{min}} > 0$, which is not true for the positive flash bounds, i.e. $0 < V < 1$. Nichita and Leibovici later showed that a stricter solution bound is given by[11]

$$V_{\text{min}}^{NL} = c_1 + z_1(c_n - c_1), \tag{2.32a}$$

$$V_{\text{max}}^{NL} = c_n - z_n(c_n - c_1), \tag{2.32b}$$

where the subscript 1 corresponds to the component with the greatest K-value and n to the component with the lowest K-value, and

$$c_i = \frac{1}{1 - K_i}. \tag{2.33}$$

2.2 Floating-point Arithmetic

It is important to take into account that all computer calculations are performed using finite-precision mathematics. Most modern computer languages use the IEEE 754 binary floating-point standard for storing and calculating non-integer numbers. An arbitrary floating point number y can be represented by[12]

$$y = \pm m \cdot \beta^e, \tag{2.34}$$

where β is the *base*, usually set to two, e is the *exponent* and m is the *significand*. The significand has a finite number of digits and is defined as $d_0.d_1d_2d_3...d_{p-1}$ where p is called the precision and d_n is an integer in the range $[0, \beta - 1]$. To ensure that all floating-point representations are unique, the significand's most significant digit, d_0, must be set to one. Because all floating-point numbers have a finite precision t, they can only represent a subset of all real numbers exactly, which means that all other numbers are represented using an approximation.

2.2.1 Round-off Errors

The approximate nature of floating-point numbers means that all floating-point calculations can produce a small error. Using the definition from Higham[13], the floating-point error ϵ of any IEEE 754 compliant arithmetic operation satisfies

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + \epsilon), \quad |\epsilon| \leq u, \tag{2.35}$$

where $fl(\cdot)$ is a floating point operation, op is either addition, subtraction, multiplication or division, and u is the machine precision. The machine precision is usually set to $2^{-24} \approx 10^{-7}$ for single-precision arithmetic and $2^{-53} \approx 10^{-16}$ for double-precision arithmetic[13]. The error produced by one arithmetic operation using single- or double-precision mathematics is usually well within the acceptable range. However, significant errors can occur when multiple operations are performed consecutively. Summing over a large set of numbers in sequence is an example of this. It can be shown by applying Equation (2.35) recursively, that adding over a set of N numbers results in

$$fl\left(\sum_{i=1}^{N} x_i\right) = x_1(1+\epsilon)^{N-1} + \sum_{i=2}^{N} x_i(1+\epsilon)^{N+1-i}. \tag{2.36}$$

This equation shows that significant errors can occur if N is large and x_1 or x_2 is one of the largest numbers in the set. There are numerous ways of reducing the errors in Equation (2.36)[13], for instance, by sorting the set in ascending order before summing. However, the example still shows how computer calculations can produce significant errors if the floating-point errors are not properly considered.

2.2.2 Catastrophic Cancellation

Another and often severe type of error is the error produced by *catastrophic cancellation*. If the two numbers a and b are equal up to seven digits but are represented by the

floating-point approximations $\tilde{a}$ and $\tilde{b}$ with a significand of eight, the resulting subtraction $\tilde{a} - \tilde{b}$ would produce an answer with seven leading zeros and thus only have one significant digit. This phenomenon where the subtraction of two floating-point numbers causes a severe loss of significance is called catastrophic cancellation[12]. This means that if $a - b = \Delta$ and $\tilde{\Delta}$ is the best floating-point approximation of Δ, then $\tilde{a} - \tilde{b} \neq \tilde{\Delta}$ because of the round-off error in floating-point numbers. An example of catastrophic cancellation is shown in Table 2.2 where $a - b$ is calculated to be 2.7 times larger with floating-point numbers than with exact numbers. Note that $\beta = 2$ for IEEE 754 floating-point numbers but is set to 10 in Table 2.2 for illustration purposes.

Table 2.2: Example of catastrophic cancellation.

	Exact Number	Approximation
a	$+2.718281828459$	$+2.718282 \cdot 10^{0}$
b	$+2.718281451231$	$+2.718281 \cdot 10^{0}$
$a - b$	$+3.77228 \cdot 10^{-7}$	$+1.000000 \cdot 10^{-6}$

If the notation $\tilde{a} - \tilde{b}_1 = \tilde{\Delta}_1$ is introduced, then there clearly exists a floating-point number $\tilde{b}_2$ such that $(\tilde{a} - \tilde{b}_1) - \tilde{b}_2 = \tilde{\Delta}_2$ where $|\tilde{\Delta}_2 - \tilde{\Delta}| < |\tilde{\Delta}_1 - \tilde{\Delta}|$. Note that this is only true if the order of operations indicated by the parenthesis is respected. If $\tilde{\Delta}_2 \neq \tilde{\Delta}$, the same procedure can by done again by introducing the number $\tilde{b}_3$, which satisfies $((\tilde{a} - \tilde{b}_1) - \tilde{b}_2) - \tilde{b}_3 = \tilde{\Delta}_3$ where $|\tilde{\Delta}_3 - \tilde{\Delta}| < |\tilde{\Delta}_2 - \tilde{\Delta}|$. This procedure can be extended N times until the condition $\tilde{\Delta}_N = \tilde{\Delta}$ is met. Therefore, using this approach makes it possible to perform the subtraction $a - b$ within machine accuracy. This means that the floating-point number $\tilde{\Delta}_N$ can be represented as $\tilde{a} - \sum_{i=1}^{N} \tilde{b}_i$, or written recursively to respect the correct order of operations

$$\tilde{\Delta}_n = \begin{cases} \tilde{a} - \tilde{b}_1, & \text{if } n = 1, \\ \tilde{\Delta}_{n-1} - \tilde{b}_n, & \text{else.} \end{cases} \tag{2.37}$$

A common algorithm that can, under certain circumstances, cause catastrophic cancellation is the quadratic formulae, i.e.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{2.38}$$

If $b \gg 4ac$ then $\sqrt{b^2 - 4ac} \approx |b|$, which means that catastrophic cancellation will happen. However, in some instances, it is possible to reformulate algorithms to avoid catastrophic cancellation. This is true for the described case of Equation (2.38) , which can be reformulated as

$$x_1 = \frac{-b - \operatorname{sgn}(b)\sqrt{b^2 - 4ac}}{2a} \tag{2.39a}$$

$$x_2 = \frac{c}{ax_1}. \tag{2.39b}$$

However, there exists no reformulation of Equation (2.38) which avoids catastrophic cancellation in the case where $b^2 \approx 4ac$[13].

2.3 Cache Storage and Branching

When optimizing a numerical algorithm for performance, there is a wide variety of topics to consider. Although it is important to have a mathematically efficient algorithm, the performance can begin to stall if the algorithm is implemented without taking the computer's memory usage into account. Early computers were much simpler than today and consisted of various components that were all designed to be equally efficient[14]. This is not the case with modern computers, where the memory and storage components have not improved as much as the CPU because of reasons associated with cost[14]. Consequently, modern CPUs read data at higher rates than the main memory can provide, leading to the introduction of cache storage. The cache is a small but highly efficient memory component connected between the main memory and CPU, acting as a high-frequency data buffer. A simplified schematic of how the CPU, cache, and main memory are configured is shown in Figure 2.4. When the CPU attempts to access data, it will first search the cache, and if the requested data is located in the cache already, a *cache hit* is said to have happened. If the requested data is not found in the cache, the CPU has to search through the much slower main memory and load it into the cache before it can be used further, which is referred to as a *cache miss*. When new data is loaded onto the cache, it replaces the data that has been unused for the longest amount of time. This is an important optimization that enables the most frequently used data to always be located in the cache, significantly reducing the amount of cache misses.

Figure 2.4: Simplified schematic of the CPU, cache, and main memory configuration.

2.3.1 Cache Locality

Another important cache optimization is based on the assumption that data is often located near other related data in the main memory. The CPU will therefore load nearby data to the cache when a cache miss occurs in addition to the necessary data. However, how well this optimization performs is highly dependent on how the program is implemented. A program is said to have good *cache locality* if related data is stored in close proximity, which significantly reduces cache misses.

An example of the importance of cache locality is presented by Stroustrup[15]. He compares linked lists and vectors in the C++ standard library, which are both types of dynamically sized containers but are optimized for different use cases. A vector stores all of its elements continuously and in the correct order in the main memory, while a linked list stores its elements in different parts of the memory, which can be seen in Figure 2.5.

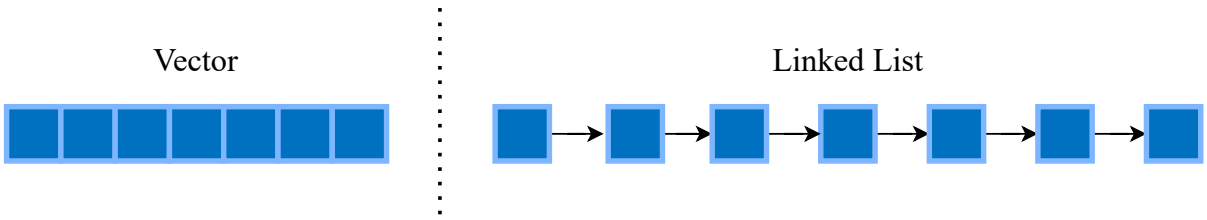


Figure 2.5: Memory layout of a vector and a linked list in C++.

Because the elements of a linked list are stored in different parts of the memory, the linked list will have worse cache locality than a vector, but the linked list can insert and delete intermediate elements with significantly fewer operations than a vector. This is because a vector will have to move all succeeding elements when an element is inserted or deleted, while a linked list can add or remove an element anywhere in the main memory without modifying the memory location of the other elements, which is shown in Figure 2.6. This means that the algorithms for inserting and deleting elements in a linked list are more mathematically efficient than their vector counterparts. To test this theory in practice, Stroustrup constructed a test case where a large set of numbers were inserted and then deleted in random order and found that the vector outperformed the mathematically more efficient linked list in all cases and the difference in execution time only increased with the size of the dataset.

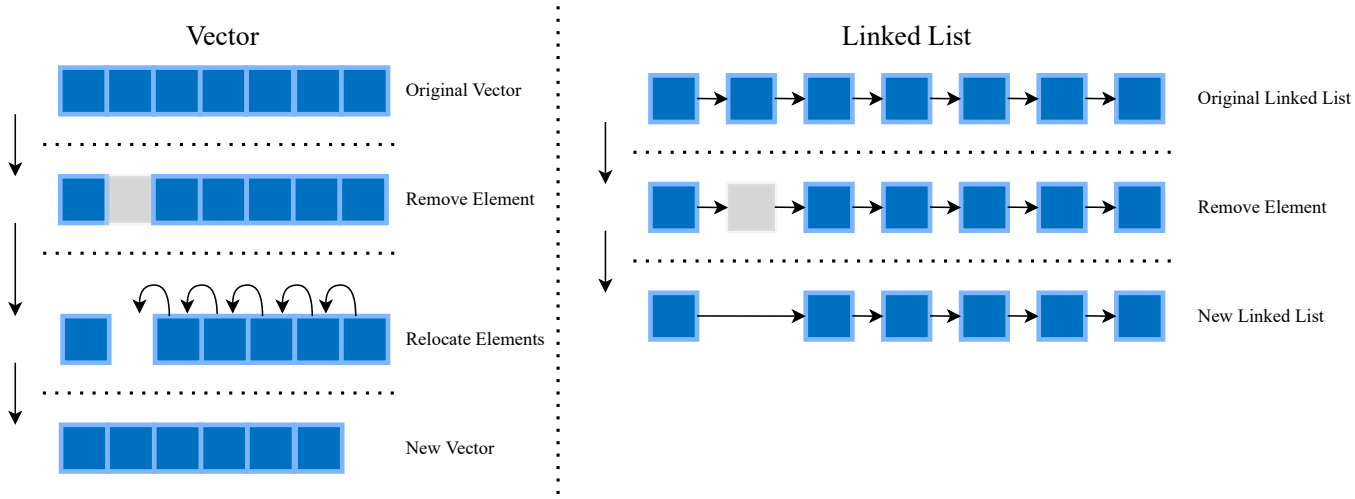


Figure 2.6: Erasing an element in a vector and a linked list.

2.3.2 Branching

Another optimization consideration not related to mathematical efficiency is *branching*. When a computer runs machine code, which is a set of instructions, it will by default execute the given instructions sequentially in order from top to bottom. A *branch* is an instruction that makes the CPU execute other instructions in a different order than

default[16]. Examples of this are functions, which are sets of instructions not located in the default path of the program. Whenever a function is called, the CPU will fetch the instructions from another part of the memory and begin executing those instead. Another example of a branch is an if-block, which conditionally tells the CPU which set of instructions to execute.

Modern CPUs execute multiple instructions in parallel, which means that a new instruction will start to execute before the last instruction is finished. If a program has a conditional branch, the next set of instructions is not always possible to know, which will cause a performance bottleneck. To combat this, modern CPUs will try to predict which branch is most likely to be correct and then begin execution of that branch. If the prediction turns out to be correct, the CPU continues execution as normal, but if the prediction is wrong, the CPU discards the current instructions and begins executing the new branch. If many branch mispredictions occur, multiple unnecessary instructions will be executed, and as a result, the program's overall performance will decrease. Branchless programming is therefore often used as an optimization tool in high-performance programming. An example of branchless programming is using the general function for a cubic EOS models given in Equation (2.11) instead of having separate branches using Equation (2.6) for the SRK EOS model and Equation (2.9) for the PR EOS model. It is worth noting that branch predictors are often highly predictive, and some branches may therefore not hurt the performance at all.

CHAPTER 3

METHODOLOGY

3.1 Single-phase Density Calculation

When a mixture is known to exist as only one phase, the molar volume can be calculated using the Z-factor, which is found by solving the form of the EOS given by Equation (2.5). In general, a cubic polynomial in the form $f(x) = x^3 + A_2x^2 + A_1x + A_0$ can have three different shapes, as shown in Figure 3.1, and will always go from negative to positive as x goes from $-\infty$ to ∞. Cubic polynomials with a saddle point or no extrema are strictly monotonic, meaning there will only be one real and distinct root. However, if a cubic polynomial has two extrema, it can have one, two, or three real roots, as shown in Figure 3.2. Whenever three roots are found when solving Equation (2.5), the middle root is discarded as it is deemed unphysical, and the root with the lowest Gibbs energy is chosen. The condition of lowest Gibbs energy is also used when two roots are found. It is shown in Appendix A that an equivalent expression to checking the root with the lowest Gibbs energy is checking wether $\ln\bar{\phi}(Z_{lowest}) < \ln\bar{\phi}(Z_{largest})$, where $\ln\bar{\phi}$ is defined as

$$\ln\bar{\phi}(Z) = Z - 1 - \ln(Z - B) - \frac{1}{\delta_1 - \delta_2} \cdot \frac{A}{B} \ln\left(\frac{Z + \delta_1 B}{Z + \delta_2 B}\right), \tag{3.1}$$

which is the same as the expression for the single component fugacity coefficient. This equivalent condition is given by Danesh[17] and is less expensive to compute than the full expression for the Gibbs energy.

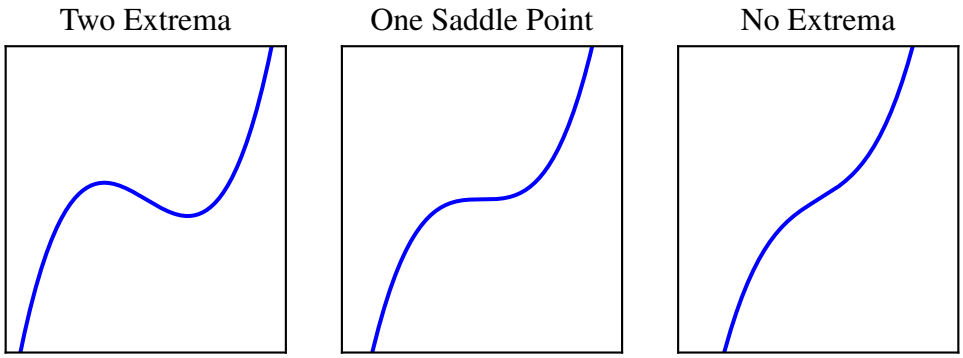


Figure 3.1: Schematic of possible shapes of a cubic polynomial.

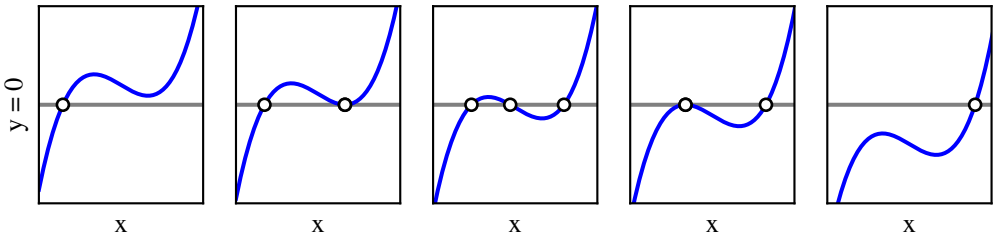


Figure 3.2: Possible locations of roots for a cubic polynomial with two extrema.

One potential issue with calculating the Gibbs energy or the fugacity coefficients and its derivatives is the term $Z - B$, which can cause catastrophic cancellation when $Z \approx B$. However, this problem can be caused by solving for $\hat{Z} = Z - B$ instead of Z. This results in finding the roots of the following cubic polynomial instead

$$\hat{Z}^3 + \hat{A}_2\hat{Z}^2 + \hat{A}_1\hat{Z} + \hat{A}_0, \tag{3.2}$$

where

$$\hat{A}_0 = -\delta_1\delta_2 B^2, \tag{3.3a}$$
$$\hat{A}_1 = \delta_1\delta_2 B^2 - (\delta_1 + \delta_2)B + A, \tag{3.3b}$$
$$\hat{A}_2 = (\delta_1 + \delta_2)B - 1. \tag{3.3c}$$

The Z-factor can then be obtained using $Z = \hat{Z} + B$, which is free of any cancellation as both the Z-factor and the parameter B are positive and $Z > B$.

3.1.1 Analytical Solution

The roots of a cubic polynomial can be found analytically using Cardano's algorithm. The algorithm is given in Numerical Recipes and is formulated as follows[18]:

1. Compute

$$Q = \frac{A_2^2 - 3A_1}{9}, \tag{3.4a}$$

$$R = \frac{2A_2^3 - 9A_2A_1 + 27A_0}{54}. \tag{3.4b}$$

2.a. If $R^2 < Q^3$, the cubic equation has three real roots which is found by calculating

$$x_1 = -2\sqrt{Q}\cos\left(\frac{\theta}{3}\right) - \frac{A_1}{3}, \tag{3.5a}$$

$$x_2 = -2\sqrt{Q}\cos\left(\frac{\theta + 2\pi}{3}\right) - \frac{A_1}{3}, \tag{3.5b}$$

$$x_3 = -2\sqrt{Q}\cos\left(\frac{\theta - 2\pi}{3}\right) - \frac{A_1}{3}, \tag{3.5c}$$

where

$$\theta = \arccos\left(\frac{R}{\sqrt{Q^3}}\right). \tag{3.6}$$

2.b. If $R^2 >= Q^3$, then the cubic equation has only one real root, which is found by calculating

$$x_1 = (A + B) - \frac{A_1}{3}, \tag{3.7}$$

where

$$A = -\text{sgn}(R)\left(|R| + \sqrt{R^2 - Q^3}^{1/3}\right), \tag{3.8a}$$

$$B = \begin{cases} Q/A, & \text{if } A \neq 0 \\ 0, & \text{if } A = 0 \end{cases} \tag{3.8b}$$

$$\tag{3.8c}$$

3. If multiple roots are found, discard the middle root and choose the root with the lowest Gibbs energy using Equation (3.1).

As with the analytical solution of quadratic equations, the explicit formula for solving cubic equations can suffer from catastrophic cancellation. Deiters reports that liquid volumes obtained using Cardano's formula can have relative errors up to 10^{-8} when the theoretical precision is set to approximately 10^{-15}[19]. As a solution, he proposes to follow the calculation of cubic roots with a single Newton-Raphson iteration, i.e.

$$x = x_C - \frac{f(x_C)}{f'(x_C)}, \tag{3.9}$$

where x_C is the solution obtained from Cardano's algorithm, and x is the solution with a reduced round-off error. This step is often referred to as *numerical refinement*. Another potential problem with the analytical formula is the computational cost associated with evaluating expensive trigonometric functions and square roots.

3.1.2 Numerical Solution

A numerical algorithm can potentially be more beneficial when solving cubic equations because of the round-off errors and computational cost of Cardano's algorithm. Deiters proposes the following numerical algorithm[19]:

1. Select an initial guess from the following formula

$$x^{(0)} = \begin{cases} -r, & \text{if } f(x^*) > 0 \\ r, & \text{if } f(x^*) \leq 0 \end{cases}, \tag{3.10}$$

where

$$r = 1 + \max(|A_0|, |A_1|, |A_2|), \tag{3.11}$$

and x^* is the location of the inflection point, which is given by $x^* = -A_2/3$.

2. Find the first root x_1 using Halley's method, which is given by

$$x^{n+1} = x^n - \frac{f(x^n)f'(x^n)}{f'(x^n)^2 - \frac{1}{2}f(x^n)f''(x^n)}. \tag{3.12}$$

3. The remaining roots is found by *deflating* the cubic polynomial, which is an algorithm that finds the quadratic polynomial $h(x)$ that satisfies $f(x) = (x - x_1)h(x)$. This is done by the following formula

$$h(x) = B_2x^2 + B_1x + B_0, \tag{3.13}$$

where

$$B_2 = 1, \tag{3.14a}$$
$$B_1 = B_2x_1 + A_2, \tag{3.14b}$$
$$B_0 = B_1x_1 + A_1. \tag{3.14c}$$

The potential roots of the quadratic polynomial $h(x)$ are then found using the improved quadratic formula given in Equation (2.39).

4. If multiple roots are found, discard the middle root and choose the root with the lowest Gibbs energy using Equation (3.1).

Michelsen shows that a hybrid solution using Cardano's algorithm to find one root and deflating the cubic to obtain the remaining roots is as fast as the numerical approach[20], but does not mention the potential round-off errors that can occur using the analytical expression.

3.1.3 Proposed Algorithm

One potential disadvantage of the aforementioned methods is that they are not able to utilize initial guesses from previous calculations. In a two-phase flash algorithm, the cubic equation is solved once for each phase per iteration, and if the calculated Z-factors do not change substantially from one iteration to the next, an initial guess-based approach using the previous calculated Z-factors can decrease the computational time significantly. A new algorithm utilizing initial guesses is therefore proposed.

1. Find all local extrema of the equation by solving the quadratic equation $f'(x) = 0$ using Equation (2.39).

2.a. If no extrema are found, the cubic will only have one real root. Find the root by using the Newton-Raphson method or Halley's method with the initial guess set to the previously calculated value. If no previous value exists, use the inflection point, i.e. $x^{(0)} = -1/3A_2$

2.b. If the cubic equation has local extrema, it can have multiple solutions. If the left-most extremum is positive, a root is found using the Newton-Raphson method or Halley's method with the initial guess set to

$$x^{(0)} = \begin{cases} x_{\text{previous}}, & \text{if } x_{\text{previous}} < x_{\max} \\ x_{\max} - 1, & \text{else} \end{cases} \tag{3.15}$$

where $x_{\max}$ is the location of the left-most extremum.

If the right-most extremum is negative, a root is found by using the Newton-Raphson method or Halley's method with the initial guess set to

$$x^{(0)} = \begin{cases} x_{\text{previous}}, & \text{if } x_{\text{previous}} > x_{\min} \\ x_{\min} + 1, & \text{else} \end{cases} \tag{3.16}$$

where $x_{\min}$ is the location of the right-most extremum. Note that using $x_{\max} - 1$ and $x_{\min} - 1$ as potential initial guesses is an arbitrary choice, but is guaranteed to converge to the correct root.

3. If multiple roots are found, choose the root with the lowest Gibbs energy using Equation (3.1).

Another advantage of this algorithm is that no computational effort is used to calculate unphysical middle roots. When initial guesses are chosen according to Equation (3.15) or Equation (3.16), the left-most or right-most root is guaranteed to be found[19]. This can be seen in Figure 3.3, where all initial guesses to the left of the local maximum converge to the smallest root, while all initial guesses to the right of the local minimum converge to the largest root.

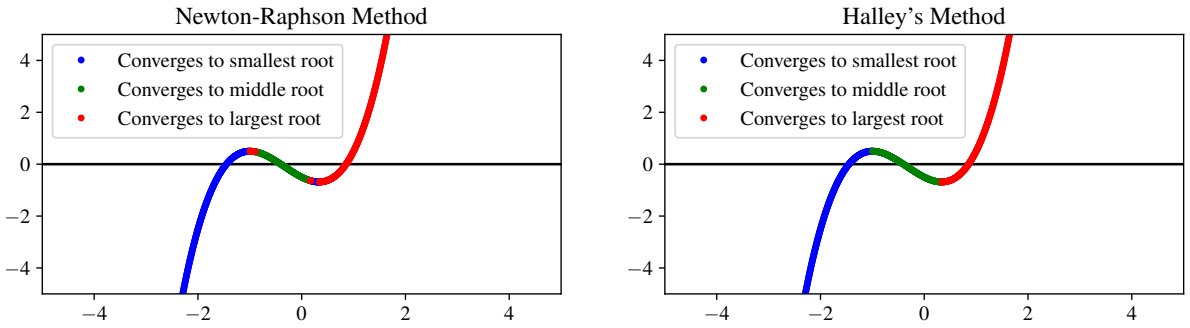


Figure 3.3: Convergence of different initial guesses on the cubic polynomial $x^3 + x^2 - x - 0.5$ using the Newton-Raphson method (left) and Halley's method (right).

3.2 Two-phase Flash Calculation

3.2.1 Solution Strategy

There are several ways to perform an isothermal flash calculation, but the most common solution algorithms available in the literature consist of the following four parts:

1. Obtaining initial K-values
2. Solving a material balance
3. Calculating phase properties
4. Updating the K-values

This procedure is shown in Figure 3.4 and represents the proposed methods by Michelsen[21], Mehra el al.[22], Ammar and Renon[23], and Whitson and Brulé[3]. This procedure is sometimes referred to as the *equation-solving approach*[24] and uses the K-values as primary variables. Other methods also exist, such as the *second-order minimization* method, which integrates the material balance step, and the *reduced flash* method, which requires a small number of non-zero BIPs to be more efficient than the equation-solving approach[25].

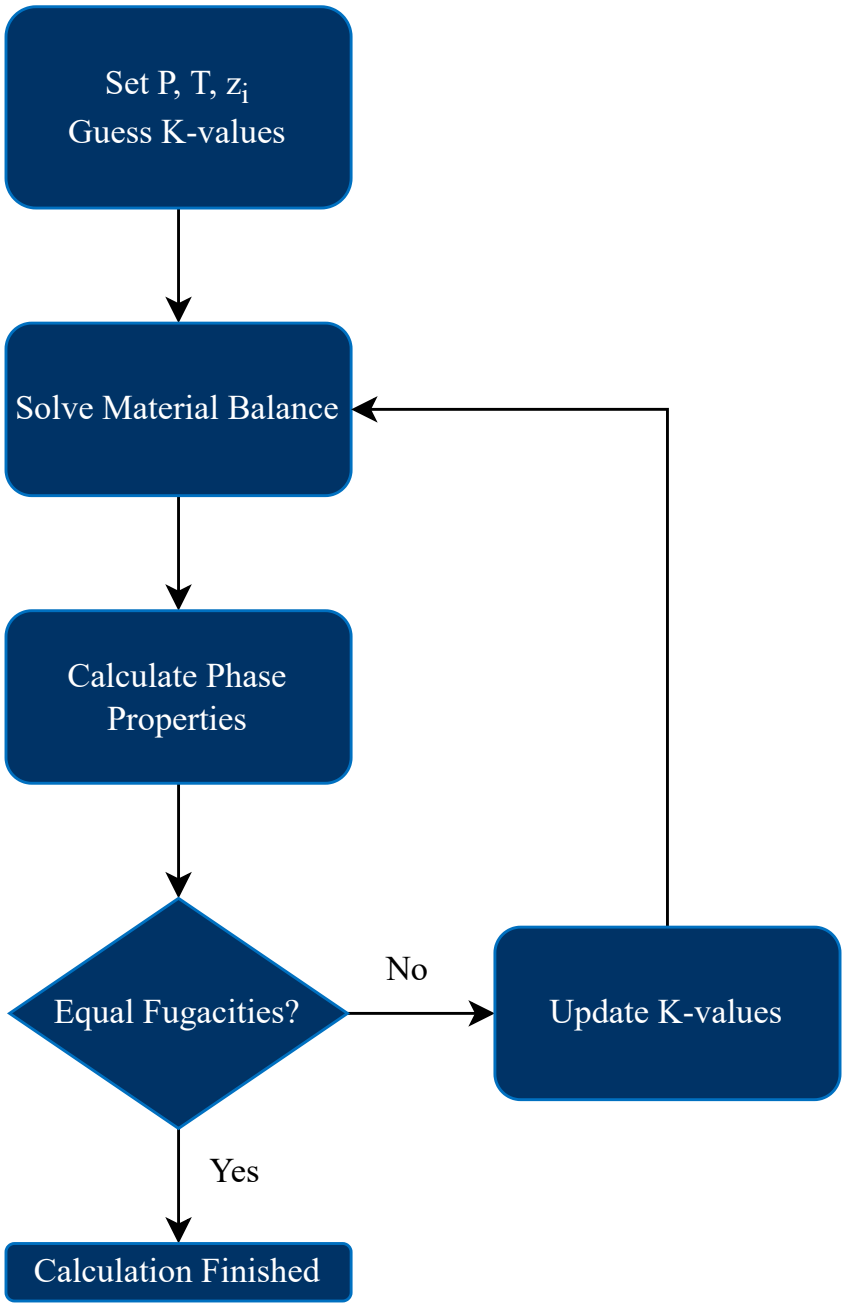


Figure 3.4: Schematic of the K-value based flash calculation procedure.

3.2.2 Initial K-value Estimates

When performing a flash calculation, a set of initial K-values is needed. If the initial K-value guess is too far away from the solution, the calculation can converge to a false trivial solution, which is when $K_i = 1$ for all components. A solution where $K_i = 1$ means that $z_i = x_i = y_i$, which indicates that a phase can form an equilibrium with itself and is therefore not a very useful result. To combat this, different methods of obtaining K-value

initial estimates have been developed.

The most common way of obtaining an initial guess of the K-values is through the use of the Wilson equation[26]

$$K_i = \frac{1}{p_{ri}} \exp\left[5.37(1+\omega_i)\left(1-\frac{1}{T_{ri}}\right)\right]. \tag{3.17}$$

The Wilson equation is inversely proportional with pressure, making it linear on a log-log plot, and is therefore not able to predict the non-linear behavior of K-values at high pressures. This can be seen in Figure 3.5, where the Wilson equation is only able to estimate the K-values somewhat accurately at pressures below 10 bar. Because of this limitation, K-values obtained by Equation (3.17) can cause the flash calculation to converge to a trivial solution when a non-trivial solution exists, i.e. a non-trivial solution.

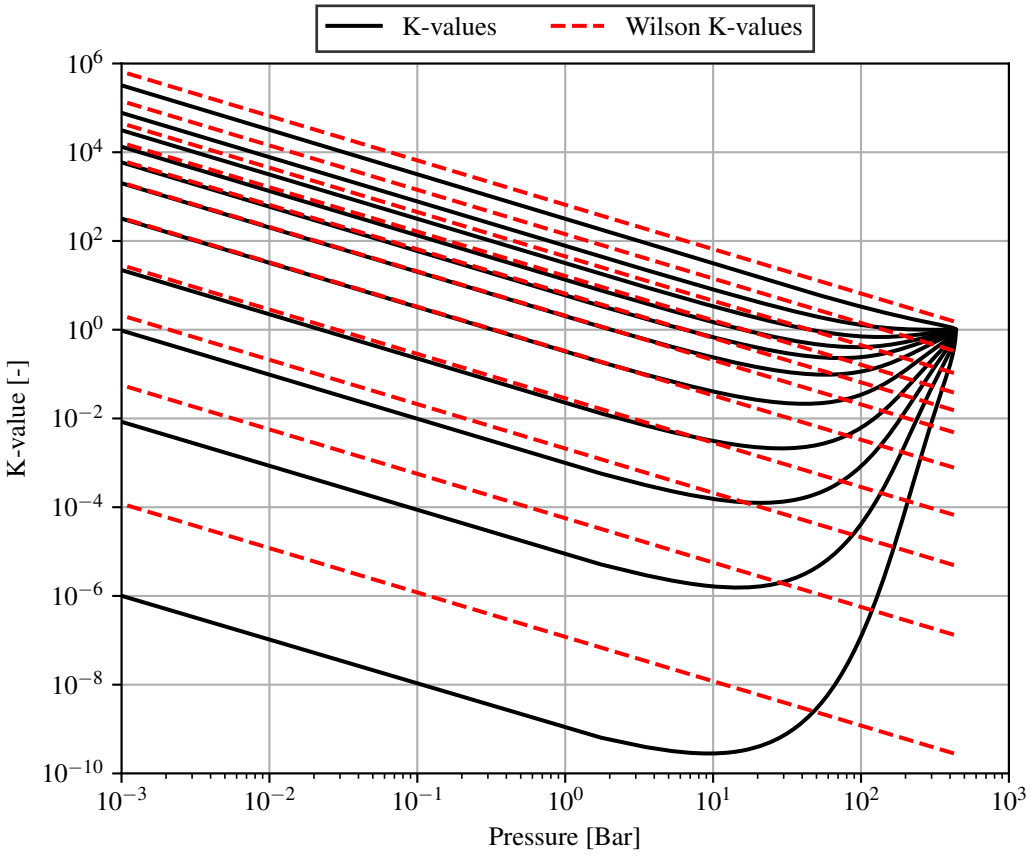


Figure 3.5: An example of K-values calculated with the Wilson equation vs the K-values calculated from the flash calculations using a cubic EOS.

Another approach for obtaining initial K-values is by performing a stability analysis calculation. The stability analysis calculation was first formulated by Michelsen[27] and is based on the Gibbs tangent plane criterion presented by Baker et al.[28]. The stability

analysis calculation determines the correct number of phases at equilibrium but will also provide a good approximation for the K-values. Using the K-values from a stability analysis calculation as the initial K-values of a flash calculation is guaranteed to converge to a non-trivial solution under certain conditions[3]. However, the stability analysis calculation is as complex and can be as computationally intensive as the flash calculation itself and is therefore not mentioned further in this work.

In most situations, the best estimates for the initial K-values are the K-values from a previous flash calculation at nearby pressure and temperature conditions. Using the previous flash calculation K-values are a good assumption n most practical applications of the flash calculations. Examples include most standard PVT depletion experiments, separator calculations and miscibility calculations to mention a few.

3.2.3 Solving the Material Balance

The most common solution approach to obey the material balance is to obtain the vapor molar fraction by solving the Rachford-Rice equation and use the solution to calculate the equilibrium phase compositions (x_i, y_i). Solving the material balance consist of solving the Rachford-Rice equation and then obtaining the molar compositions of the two phases. Muskat and McDowell published an equivalent formulation of Equation (2.30) in 1949[29]. They developed a phase equilibrium solver that replaced the trial and error methods that were used at the time[29]. Their solution for solving the equation was to construct an equivalent electrical circuit where they could vary the voltage until the measured current was zero. Three years later, in 1952, Rachford and Rice published Equation (2.30) and proposed a bisection algorithm that could find its root automatically using a computer[30].

An example of how the Rachford-Rice equation varies with vapor molar fraction is shown in Figure 3.6. If Equation (2.30) is rearranged to

$$h(V) = \sum_{i=1}^{N_c} \frac{z_i}{V - c_i}, \tag{3.18}$$

the vertical asymptotes seen in Figure 3.6 corresponds to all component's c_i values. From this, it is possible to show that the Rachford-Rice equation has N_c asymptotes, and because of the monotonic behavior between asymptotes, there will be $N_c - 1$ roots.

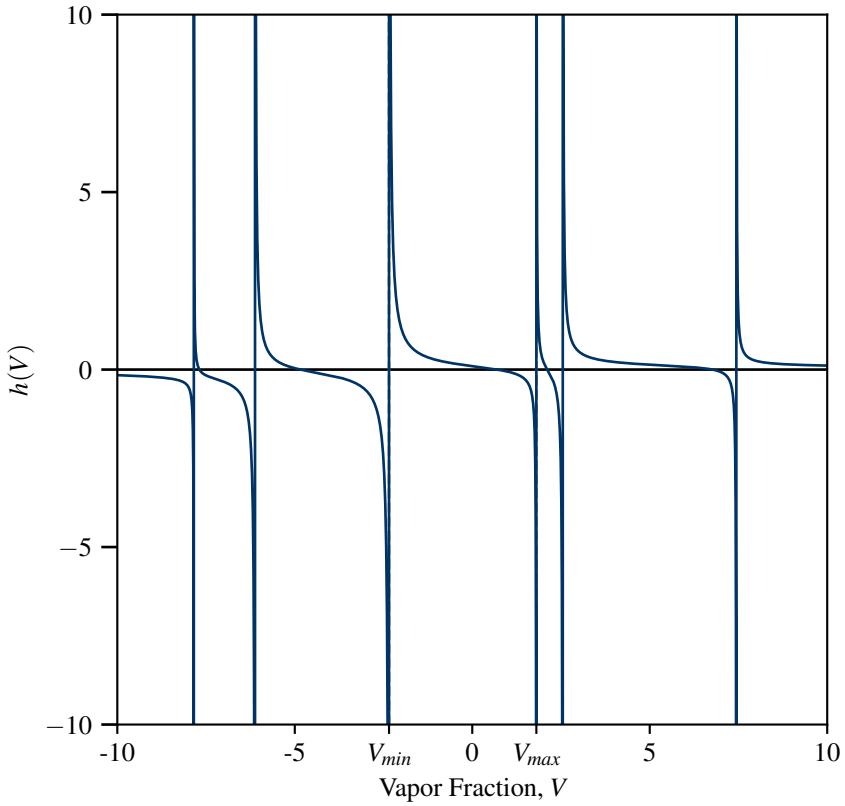


Figure 3.6: Rachford-Rice function for a six-component system.

Equation (3.18) is often solved using the Newton-Raphson method

$$V^{(n)} = V^{(n-1)} - \frac{h(V^{(n-1)})}{h'(V^{(n-1)})}, \tag{3.19}$$

where

$$h'(V) = -\sum_{i=1}^{N_c} \frac{z_i}{(V - c_i)^2}. \tag{3.20}$$

Other higher order root finding methods, such as Halley's method, are also possible to use without increasing the computational cost significantly. This is because higher order derivatives of the Rachford-Rice equation are given by

$$h^k(V) = (-1)^n \sum_{i=1}^{N_c} \frac{z_i}{(V - c_i)^{(k+1)}}, \tag{3.21}$$

where the sum can be efficiently calculated with the same for-loop as the sum in the Rachford-Rice equation. En example of how this can be implemented in the Python programming language is shown in Figure 3.7, where only one extra multiplication and one extra addition for each component are required to calculate a derivative.

Figure 3.7: Efficiently evaluating the Rachford-Rice function, its derivative, and double derivative using Python.

```python
def evalute_rachford_rice(K_values, composition, V):
    h = 0  # Rachford-Rice function
    dh = 0  # Rachford-Rice derivative function
    ddh = 0  # Rachford-Rice double derivative function
    for i in range(len(composition)):
        zi = composition[i]
        Ki = K_values[i]
        common_term = (Ki - 1) / (1 - V * (Ki - 1))

        # Calculate and add RR term
        rachford_rice_term = common_term * zi
        h += rachford_rice_term

        # Calculate and add derivative term
        derivative_term = -rachford_rice_term * common_term
        dh += derivative_term

        # Calculate and add double derivative term
        double_derivative_term = -derivative_term * common_term
        ddh += double_derivative_term

    return h, dh, ddh
```

The number of iterations of the root finding method is dependent on the initial guess $V^{(0)}$. A common initial estimate is the midpoint of the negative flash solution space, given by

$$V^{(0)} = \frac{1}{2}(V_{\text{min}} + V_{\text{max}}). \tag{3.22}$$

More accurate but more computationally heavy initial estimates also have been proposed[11]. However, a good initial guess is often the calculated value of V at the previous iteration of the flash.

The Newton-Raphson method generally requires few iterations but can sometimes overshoot the bounds given by the negative flash region[8]. One way of avoiding this is to do a bisection step if V is predicted below V_{min} or above V_{max} and continuously update the bounds at each iteration. However, this approach can make the solution procedure slow, especially if multiple bisection steps are needed. Michelsen and Mollerup[8] suggest starting over again with a better initial guess if overshooting occurs. If the root-solving method overshoots such that $V < V_{\text{min}}$ a better initial guess is then V_{min}^{NL}, and if overshooting occurs such that $V > V_{\text{max}}$ a new initial guess is then V_{max}^{NL}. It can, in fact, be proven that using the bounds V_{min}^{NL} and V_{max}^{NL} is guaranteed to converge if the Newton-Raphson method is used.

However, while this is true mathematically, it is not necessarily true if the computation is

performed using finite-precision mathematics. An example of the Newton-Raphson failing to converge to within machine-accurate precision is shown in Figure 3.8.

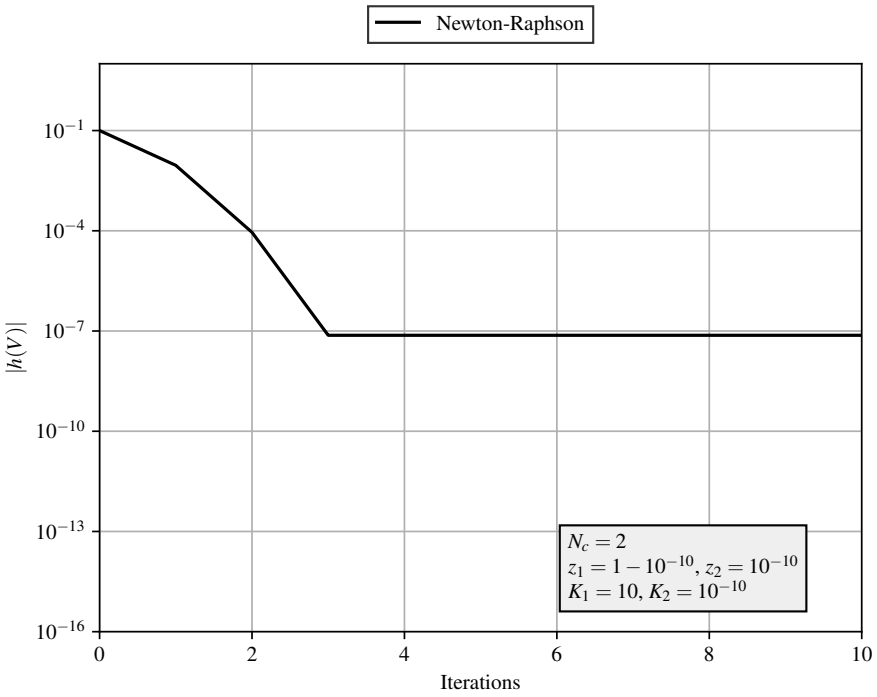


Figure 3.8: Newton-Raphson method for the Rachford-Rice equation with a binary mixture, failing to lower residual past $\approx 10^{-7}$.

This has been reported several times before, most notably in the Rachford-Rice contest from 1995 issued by Curtis Whitson[31]. The contest challenged the participants to solve a set of compositions without violating the material balance and achieve as low computational time as possible. Only two of the submissions beat Zick's algorithm and were also the only two to produce machine-accurate results. However, they were not submitted by students and were also never published.

The only published literature to have claimed to fix the round-off error is the 1997 conference paper by Wang et al.[32]. However, they only considered the round-off errors in the Rachford-Rice equation and not in the calculation of x_i and y_i. Equations 2.29a and 2.29b are used to calculate x_i and y_i, and are also used in the derivation of Equation (2.30). Any round-off issues occurring in Equation (2.30) will therefore occur in the calculation of the equilibrium phase compositions, which can lead to further inaccuracies when calculating phase properties.

The material balance issues in the Rachford-Rice equation are due to Catastrophic cancellation when $V \approx c_i$. A novel solution approach is presented by representing the terms $V - c_i$ in a similar manner as in Equation (2.37). This means that the vapor fraction

needs to be represented as a sum

$$V = \sum_{n=0}^{N} \delta^{(n)}. \tag{3.23}$$

If the Newton-Raphson method converges after N iterations, the vapor molar fraction is set to $V^{(N)}$. Using this fact together with Equation (3.23), the following definition can be introduced

$$\delta^{(n)} = V^{(n)} - V^{(n-1)}, \tag{3.24}$$

and Equation (3.19) can be rewritten as

$$\delta^{(n)} = -h\left(V^{(n-1)}\right)\left[\frac{dh}{dV}\left(V^{(n-1)}\right)\right]^{-1}. \tag{3.25}$$

To make $\delta^{(n)}$ a function $\delta^{(n-1)}$, h can be replaced with

$$h_n(\delta) = \sum_{i=1}^{N_c} \frac{z_i}{\delta - d_i^{(n)}}, \tag{3.26}$$

where

$$d_i^{(n)} = c_i - \sum_{k=0}^{n-1} \delta^{(k)}. \tag{3.27}$$

Equation (3.26) is a Rachford-Rice type equation and has the properties

$$h_n(0) = h(V^{(n-1)}), \tag{3.28a}$$

$$\frac{d^k h_n}{d\delta^k}(0) = \frac{d^k h}{dV^k}(V^{(n-1)}), \tag{3.28b}$$

which means that Equation (3.25) can be written as

$$\delta^{(n)} = -h_n(0)\left[\frac{dh_n}{d\delta}(0)\right]^{-1}. \tag{3.29}$$

This can be presented in the following recursive notation

$$d_i^{(n)} = \begin{cases} c_i, & \text{if } n = 0, \\ d_i^{(n-1)} - \delta^{(n-1)}, & \text{else.} \end{cases} \tag{3.30a}$$

$$\delta^{(n)} = \begin{cases} V^{(0)}, & \text{if } n = 0, \\ -h_n(0)\left[\frac{dh_n}{d\delta}(0)\right]^{-1}, & \text{else.} \end{cases} \tag{3.30b}$$

The recursive Newton-Raphson method given in Equation (3.30a) is written in the same form as Equation (2.37), and will therefore mitigate any errors due to round-off. This can been seen Figure 3.9, where the recursive Newton-Raphson method is applied to the same case as in Figure 3.8 and no round-off issues are seen.

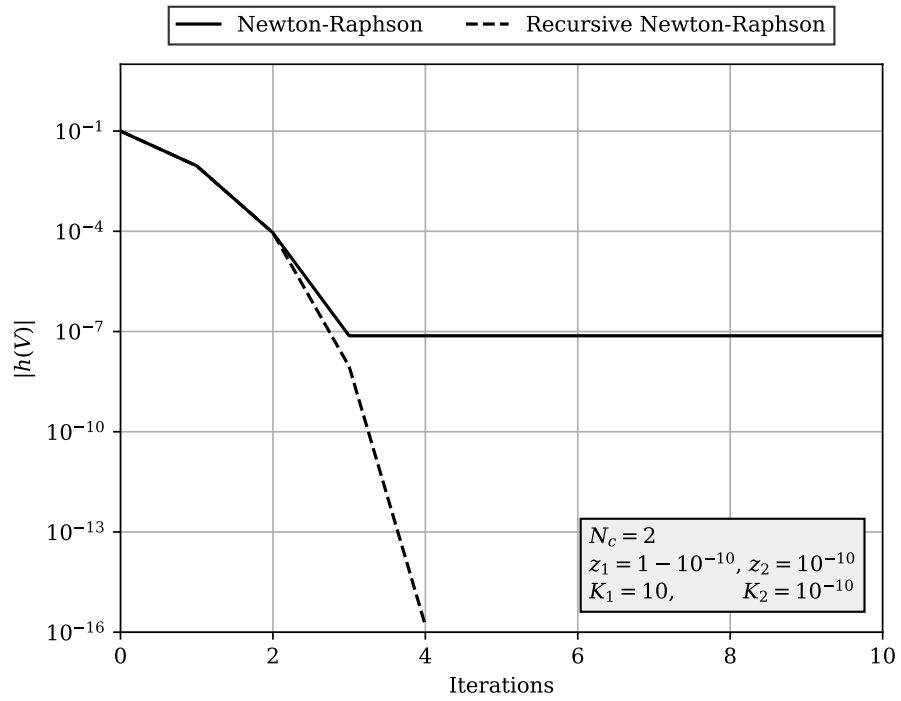


Figure 3.9: Recursive Newton-Raphson method for the Rachford-Rice equation with a binary mixture, where round-off errors are mitigated.

After the recursive Newton-Raphson method have converged, the equilibrium phase compositions can be calculated by

$$x_i = \frac{z_i c_i}{d_i^N}, \tag{3.31a}$$

$$y_i = K_i x_i = K_i \frac{z_i c_i}{d_i^N}. \tag{3.31b}$$

Equations 3.31a and 3.31b contains no additions or subtractions, which means that the calculation of x_i and y_i is correct within machine precision[12].

3.2.4 Phase Property Calculations

The flash calculation is driven by the equal fugacity constraint, therefore the fugacities of each phase must be calculated, given by Equation (2.22). To calculate the fugacity, the Z-factor needs to be calculated using Equation (2.5), which requires the average A and B parameters for each phase. These parameters are obtained using Equation (2.21), and because the temperature and pressure are constant in every iteration of the flash, the parameters A_{ij} and B_i are constant for all iterations. The A parameter is calculated using a quadratic mixing rule, which causes the calculation time to increase quadratically with the number of components. By using the fact that $z_i = Lx_i + Vy_i$, Michelsen showed

that the computation time could be decreased by using the following equation

$$A_i^V = \sum_{j=1}^{N_c} y_j A_{ij} = \frac{1}{V}\left(\sum_{j=1}^{N_c} z_j A_{ij} - L\sum_{j=1}^{N_c} x_j A_{ij}\right) = \frac{1}{V}\left(A_i^z - LA_i^L\right). \tag{3.32}$$

The feed composition z_i is constant, which means that A_i^z is only needed to be calculated once and A_i^V can be calculated directly from Equation (3.32) instead of computing the full sum. A for the vapor and liquid phase, A^V and A^L, can then be calculated by

$$A^V = \sum_{i=1}^{N_c} y_i A_i^V, \tag{3.33a}$$

$$A^L = \sum_{i=1}^{N_c} x_i A_i^L. \tag{3.33b}$$

When the parameters A and B are obtained, the Z-factors for each phase are found using one of the algorithms described in Section 3.1.

3.2.5 Successive Substitution

After the component fugacities are calculated, a new set of K-values can be found using the equation

$$K_i^{(n+1)} = \frac{(\phi_i^L)^{(n)}}{(\phi_i^V)^{(n)}}. \tag{3.34}$$

The *residuals* (r_i^n) of the flash calculation at iteration n can be defined as

$$r_i^{(n+1)} = \Delta \ln K_i^{(n+1)} = (\ln \phi_i^L)^{(n)} - (\ln \phi_i^V)^{(n)} - \ln K_i^{(n)}. \tag{3.35}$$

The equal fugacity constraint cna then be represented by

$$||r_i|| < \varepsilon, \tag{3.36}$$

where ε is an arbitrary convergence criteria set to 1^{-12} in this work. One choice for the vector norm in Equation (3.36) is the infinity norm defined as

$$||X_i|| = \max_i |X_i|. \tag{3.37}$$

If Equation (3.36) is not fulfilled, the material balance is again solved using the K-values from Equation (3.34). This method of updating the K-values is called successive substitution, which is a first-order fixed point method[21]. The rate of convergence of the successive substitution method is determined by the largest eigenvalue (λ_l) of the matrix $\mathbf{S}$[21], defined by

$$S_{ij} = \left(\frac{\partial \ln K_i^{n+1}}{\partial \ln K_j^n}\right)_{n\to\infty}, \tag{3.38}$$

where the rate of convergence becomes worse as $|\lambda_l|$ gets closer to 1. Whitson and Michelsen show that as the pressure and temperature approach critical conditions, two eigenvalues of $\mathbf{S}$ approach 1[10]. This means that the successive substitution method becomes very slow near critical points, where thousands of iterations may be needed[3]. Various accelerated fixed-point methods are therefore often used when updating K-values.

Prausnitz et al. recommends to use Wegstein's method, given by

$$x^{(n+1)} = \frac{x^{(n-1)}g(x^{(n)}) - x^{(n)}g(x^{(n-1)})}{x^{(n-1)} + g(x^{(n)}) - x^{(n)} - g(x^{(n-1)})}, \tag{3.39}$$

where x is the primary variable and $g(x)$ is a function with a fixed point at the solution. In the case of the isothermal flash, $x = \ln K_i$ and $g(x) = (\ln \phi_i^L)^{(n)} - (\ln \phi_i^V)^{(n)} = \ln K_i^{(n)} + r_i^{(n)}$, and Equation (3.39) reduces to

$$r^{(n+1)} = -\frac{r_i^{(n)} r_i^{(n-1)}}{r_i^{(n)} - r_i^{(n-1)}}, \tag{3.40}$$

where the residuals $r_i^{(n)}$ and $r_i^{(n-1)}$ is calculated using the successive substitution method.

Another accelerated method, which is recommended by [21], is the dominant eigenvalue method (DEM) developed by Orbach and Crowe, given by

$$r^{(n+1)} = \frac{r^{(n)}}{1+\mu}, \tag{3.41}$$

where

$$\mu = -\sqrt{\frac{b_{00}}{b_{11}}}, \tag{3.42a}$$

$$b_{ii} = \sum_{j=1}^{N_c} \left(r_j^{(n-i)} \right)^2. \tag{3.42b}$$

Equation (3.41) approximates the result where the successive substitution method is applied infinite times, assuming that the matrix $\mathbf{S}$ is only dominated by one eigenvalue. However, as mentioned previously, there exist two eigenvalues of the same magnitude near the critical point. Because of this, Michelsen suggests that the general dominant eigenvalue method (GDEM) by Nishio and Crowe[33] can be even more efficient[21].

The GDEM method generalizes the DEM to work with multiple dominant eigenvalues and reduces to the following expression if two dominant eigenvalues are assumed.

$$r^{(n+1)} = \frac{r_i^{(n)} - \mu_2 r_i^{(n-1)}}{1 + \mu_1 + \mu_2}, \tag{3.43}$$

where μ_1 and μ_2 are given by

$$\mu_1 = \frac{b_{02}b_{12} - b_{01}b_{22}}{b_{11}b_{22} - b_{12}^2}, \tag{3.44a}$$

$$\mu_2 = \frac{b_{01}b_{12} - b_{02}b_{11}}{b_{11}b_{22} - b_{12}^2}, \tag{3.44b}$$

where

$$b_{ij} = \langle \mathbf{r}^{(n-i)}, \mathbf{r}^{(n-j)} \rangle. \tag{3.45}$$

The symbol $\langle \cdot, \cdot \rangle$ denotes the inner product given by

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^{\mathrm{T}} \mathbf{W} \mathbf{y}, \tag{3.46}$$

where $\mathbf{W}$ is a positive diagonal weighting matrix usually set to the identity matrix[21].

A common problem with accelerated methods is that they can often overshoot, which causes the calculation to converge to a false trivial solution. This is not true when using only successive substitution iterations, and it is therefore important to consider the ratio between the number of accelerated iterations and successive substitution iterations. All the mentioned accelerated successive substitution methods require that the previous iterations are done using successive substitution. The procedure of first performing a number of successive substitution iterations and then doing an accelerated step is often called a *promotion step*. Michelsen and Mollerup recommend performing five successive substitution iterations at each promotion step[8], but it is possible to use any number greater than or equal to four.

3.2.6 The Newton-Raphson Method

Another method of updating K-values is by using the Newton-Raphson method. Michelsen suggests using the following formulation of the Newton-Raphson method

$$J_{ij} \cdot r_i = -g_i, \tag{3.47}$$

where $r_i = \Delta \ln K_i$ and $g_i = \ln f_i^V - \ln f_i^L$ and the jacobian matrix is defined by

$$\mathbf{J} = \mathbf{B}\mathbf{A}^{-1} \tag{3.48}$$

with

$$A_{ij} = \frac{z_i}{x_i y_j}\delta_{ij} - 1, \tag{3.49a}$$

$$B_{ij} = A_{ij} + L\frac{\partial \ln \phi_i^V}{\partial n_j^V} + V\frac{\partial \ln \phi_i^L}{\partial n_j^L}, \tag{3.49b}$$

where δ_{ij} is the Kronecker delta defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{3.50}$$

The fugacity coefficient derivatives in Equation (3.49b) can be found using the approach given by Mollerup and Michelsen[34], and the system of linear equations given in Equation (3.47) can be solved by using the decomposition

$$\mathbf{B} = \mathbf{L}\mathbf{D}\mathbf{L}^T, \tag{3.51}$$

where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{D}$ is a positive diagonal matrix. The two matrices are found by the following formulas[35]

$$D_{ii} = B_{ii} - \sum_{k=0}^{i-1} B_{ik}^2 B_{kk}, \tag{3.52a}$$

$$L_{ij} = \left(B_{ij} - \sum_{k=0}^{i-1} B_{ik} D_{kk} B_{jk} \right) \frac{1}{D_{ii}}. \tag{3.52b}$$

When the two matrices $\mathbf{L}$ and $\mathbf{D}$ are obtained, Equation (3.47) can be solved by

$$\mathbf{r} = -\mathbf{A}\mathbf{L}^{-T}\mathbf{D}^{-1}\mathbf{L}^{-1}\mathbf{g} \tag{3.53}$$

The Newton-Raphson method is a second-order method and will therefore generally converge in fewer iterations than any successive substitution method. However, the Newton-Raphson method is significantly more computationally expensive as it requires storing matrices, performing matrix operations, and calculating fugacity derivatives. In addition, the Newton-Raphson method can fail to converge if the initial guess is too far from the solution because the jacobian matrix might not be positive definite[8]. Michelsen, therefore, advises starting with either two or three accelerated successive substitution promotion steps before the Newton-Raphson method is used[8].

3.3 Data Generation

3.3.1 EOS models and Compositions

Testing of the various methods was performed by using three PR EOS models, which can be found in Appendix B. The first model is a 35-component model, while the two others are lumped versions of that EOS. Lumping an EOS means reducing the number of components, which is done by grouping certain components together into one pseudo-component[9]. For instance, it is common to lump ethane and CO_2 together because they have similar phase properties. Using a lumped EOS will reduce computational time, but the accuracy of the model will generally be worse. The two lumped EOS models have 14

and 9 components and are made using the default lumping method in PhazeComp. To get an indication of how similar the different EOS models are, the phase envelopes of a given test composition are shown in Figure 3.10.

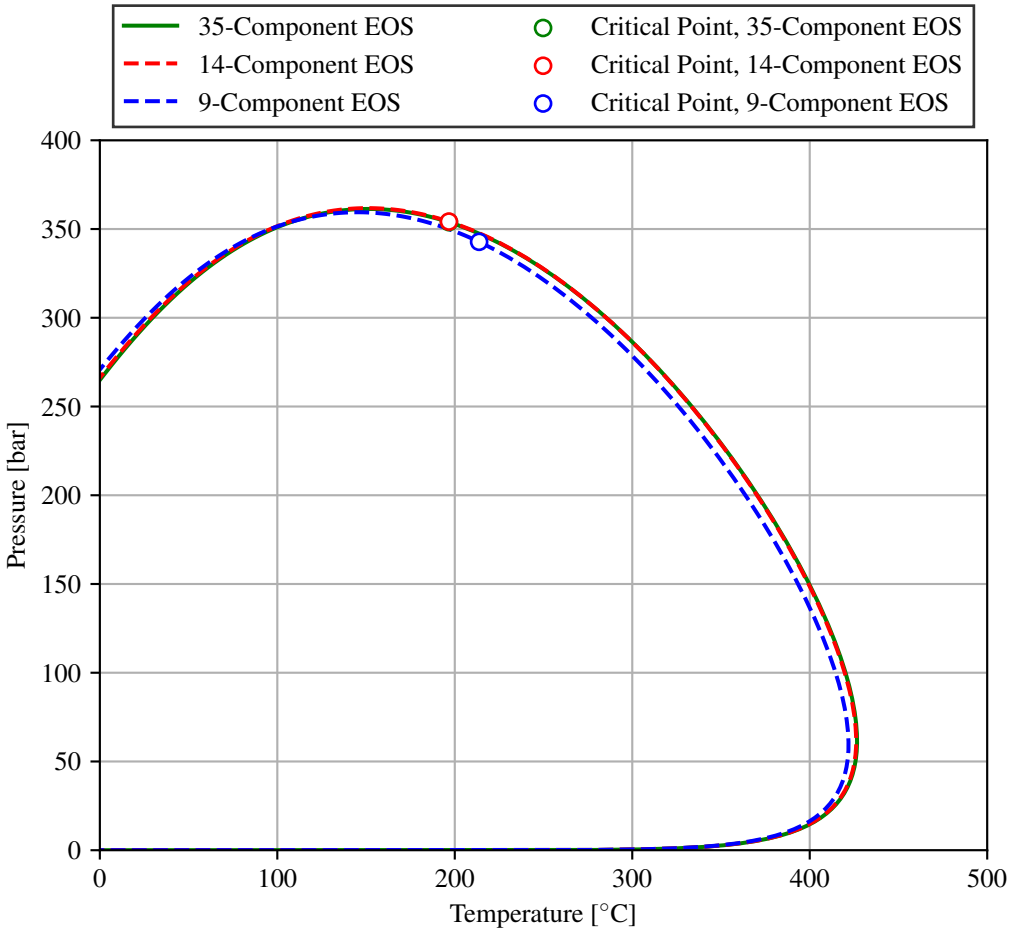


Figure 3.10: Phase envelopes of a test mixture using a 35-component EOS and two lumped EOS models with 14 and 9 components. Note: The calculated phase envelopes using the 35-component and 14-component models coincides almost entirely.

To test how the flash calculations methods perform with different fluid systems, seven synthetic compositions have been generated using whitson+[36]. The different compositions can be found in Appendix C as well as their calculated phase envelopes using the three EOS models.

3.4 Performance Testing

C++ is a compiled programming language, which means that all code is translated to machine instructions before the code is executed. Examples of non-compiled languages

are Python and MATLAB, which are both examples of interpreted languages. They both translate code into machine instructions while the programming is running. This translation is costly, and compiled languages are often faster than interpreted languages because of this. Another benefit of compiled languages such as C++ is that extensive code optimization is performed before code is turned into machine instructions. But because different implementations are easier to optimize than others, it will make it harder to know what will make a program run faster without actually testing it. Therefore, it is beneficial to use performance testing tools when writing performance-critical code in C++.

3.4.1 Profiling - Callgrind

Callgrind is an open-source profiling tool part of the Valgrind tool package, which tracks the function calls of a program and outputs how long each function runs. This enables programmers to identify the functions that are potential performance bottlenecks of a program. For instance, a 20% reduction of a function that uses 40% of the total computation time will reduce the overall run time by 8%, while a 99% reduction of a function that uses 1% of the total computation time will only reduce the overall computation time by 0.99%. When a program is used together with Callgrind, it will run significantly slower, meaning that the reported execution time of each function is not a meaningful metric, and the percentage of the total execution time should be used instead.

3.4.2 Benchmarking - Google Benchmark

Another tool that can be used in tandem with Callgrind is Google's Benchmark C/C++ library. The library can be used to create benchmarks that times a specific part of a program multiple times and that output the average execution time. Multiple benchmarks can run in series, which makes it easy to test different implementations of the same functionality. A code optimization strategy that proved to be very effective when implementing the flash calculation is given below.

1. Make flash calculation benchmarks at some specified temperatures and pressures.
2. Run the benchmarks while running Callgrind and find a function that uses a significant amount of the total execution time.
3. Try to write a more efficient implementation of the function. As mentioned earlier, there is no predictable way of improving the performance of a program, but trying to improve cache locality, removing branches, or changing an algorithm can all potentially lead to a performance increase.
4. Make a benchmark of the new function and run it with Callgrind to see if the new implementation's execution time is lower than that of the old implementation.
5. Run both the old and new benchmarks to see if overall execution time went down. To get accurate execution times, it is important to run the benchmarks without Callgrind.

6. If a performance increase is detected, the new optimized function is used. If no performance benefits are found, the new implementation is discarded if it makes the code harder to read or maintain.

CHAPTER 4

RESULTS AND DISCUSSION

The data used for testing in the following sections were generated by performing multiple flash calculations in a pressure and temperature grid, ranging from 1 bar to 500 bar and 0^{o}C to 500^{o}C with a step size of 2 units for both axes. All flash calculations were initialized with the results of an a priori flash calculation performed at 100^{o}C and 150 bar. This choice will ensure that the successive substitution method will converge to a non-trivial solution for every flash calculation inside the convergence envelope, and will highlight issues related to convergence to false trivial solutions.

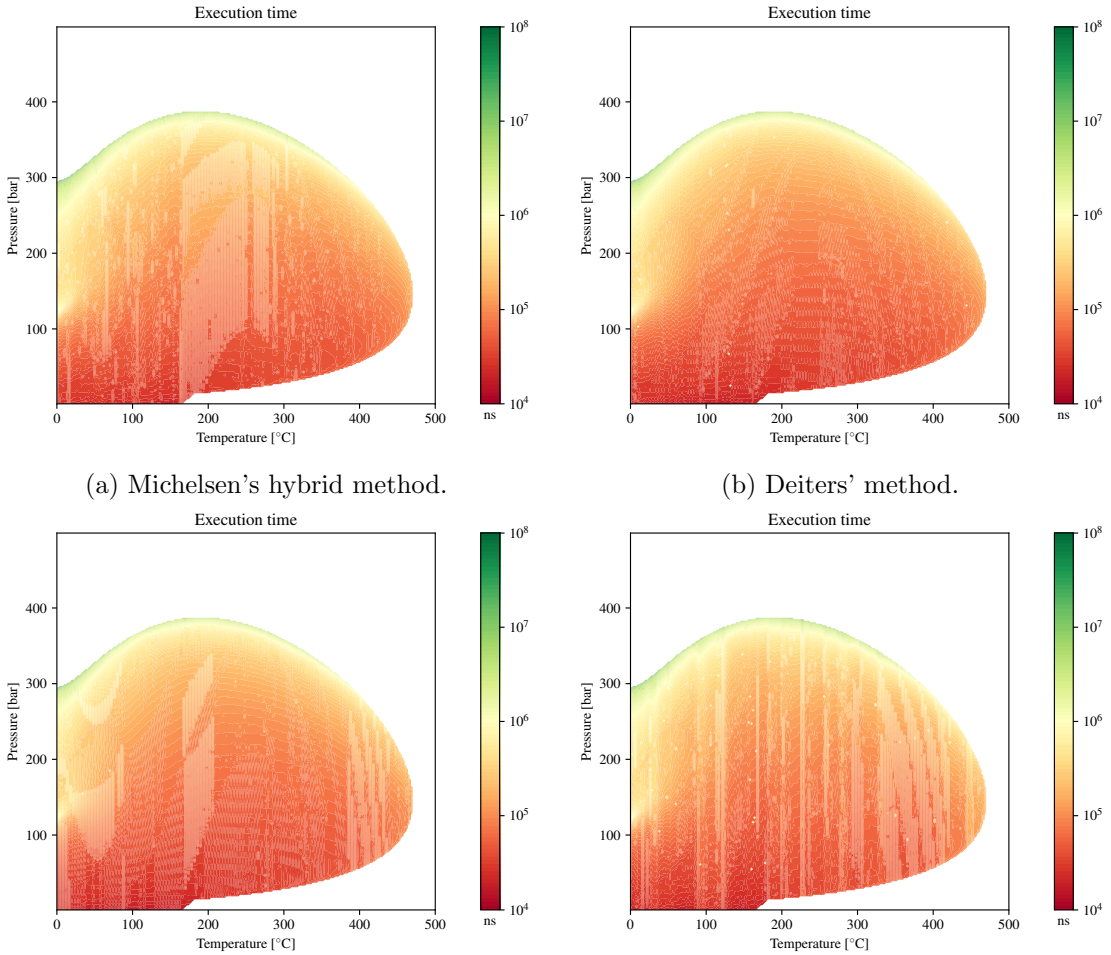
4.1 Cubic Solvers

Four different cubic equation solving methods were tested: Deiters' numerical method, Michelsen's hybrid method, and two different versions of the numerical method described in subsection 3.1.3 using the Newton-Raphson method and Halley's method. In Table 4.1 it can be seen that the initial guess-based Newton-Raphson method performed the best out of all methods. The difference in the results of the three other methods is not conclusive. This is in line with the conclusion described by Michelsen[20]. Figure 4.1 shows that the choice of a cubic equation solver does not impact the overall flash calculation performance behavior.

Table 4.1: Total execution time for non-trivial flashes using different cubic solvers.

	Execution Time (ms)				Flashes per second
Method	35C	14C	9C	Total	Total
Newton-Raphson	66103	19503	12322	97929	6259
Halley	76015	20314	11987	108318	5659
Deiters	71402	21234	14545	107182	5719
Michelsen	73138	23533	15283	111955	5475

Shading artifacts are seen in Figure 4.1. This is due to the uncertainty in the run time caused by external factors. External factors may be related to other programs running simultaneously, heat, etc. The total execution time reported in Table 4.1 will average these effects to some extent.



(a) Michelsen's hybrid method.

(b) Deiters' method.

(c) Initial-guess based Newton-Raphson method.

(d) Initial-guess based Halley's method.

Figure 4.1: Execution time of flash calculations vs temperature and pressure of composition 1 with the 35-component EOS model where different cubic equation solving methods are used.

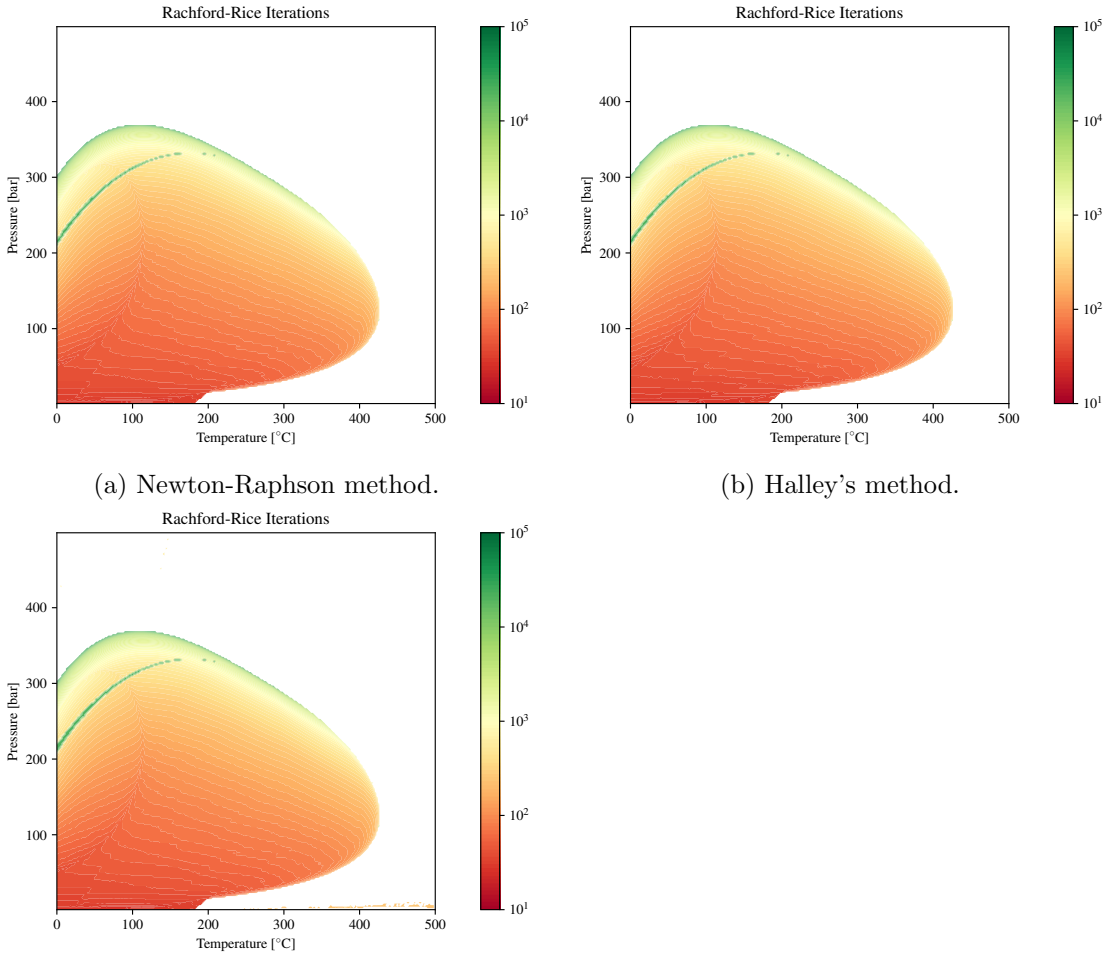
4.2 Rachford Rice Solvers

Three different methods for solving the Rachford-Rice equation were tested for performance: The Newton-Raphson method, Halley's method, and the novel recursive Newton-Raphson method. Table 4.2 shows that the recursive Newton-Raphson method is significantly faster than the Newton-Raphson method by approximately 6% and Halley's method by approximately 8% on the total runtime of the flash calculation for the 35-component EOS.

The difference in execution time is less significant with decreasing amount of components. This is expected because the number of operations in the Rachford-Rice equation linearly increases with the number of components.

Table 4.2: Total execution time for different Rachford-Rice solvers.

	Execution Time (ms)				Flashes per second
Method	35C	14C	9C	Total	Total
Newton-Raphson	75574	19499	12063	107137	5720
Halley	76584	22191	14842	113619	5394
Recursive Newton-Raphson	71162	18615	12095	101873	6017



(a) Newton-Raphson method.

(b) Halley's method.

(c) Recursive Newton-Raphson method.

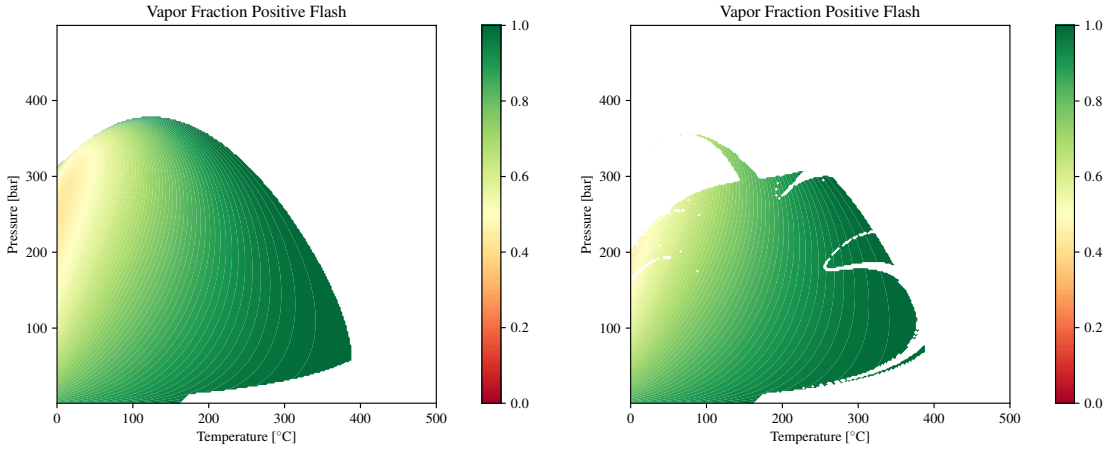
Figure 4.2: Number of Rachford-Rice iterations vs temperature and pressure of composition 4 with the 35-component EOS model where different Rachford-Rice solving methods are used.

The number of Rachford-Rice iterations increases near the phase boundary and the converge

pressure because the number of flash iterations increases near these boundaries.

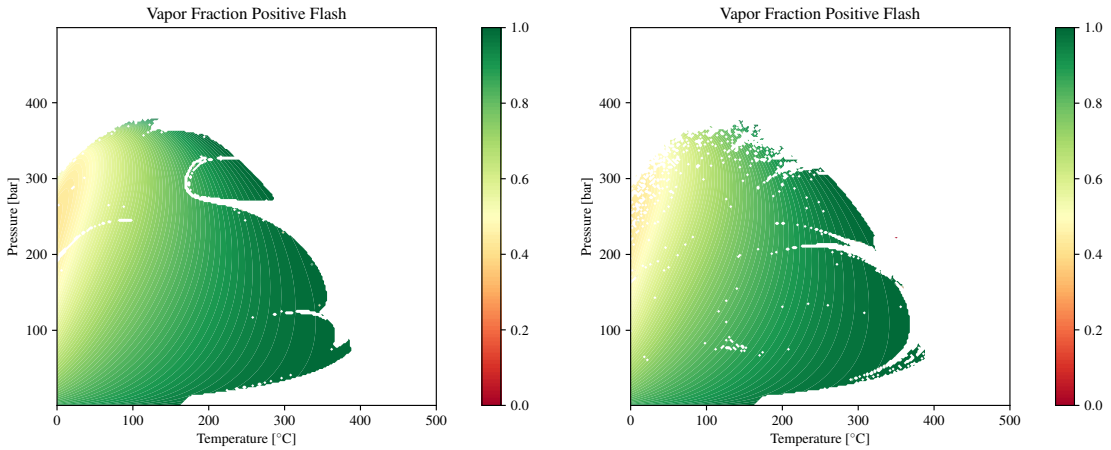
4.3 Accelerated Successive Substitution

Three different accelerated successive substitution methods were tested and compared to the standard successive substitution method. These acceleration methods are Wegstein's method, DEM and GDEM. Five successive substitution steps were used before performing the acceleration step. This procedure was used for all methods.



(a) Successive Substitution.

(b) DEM.



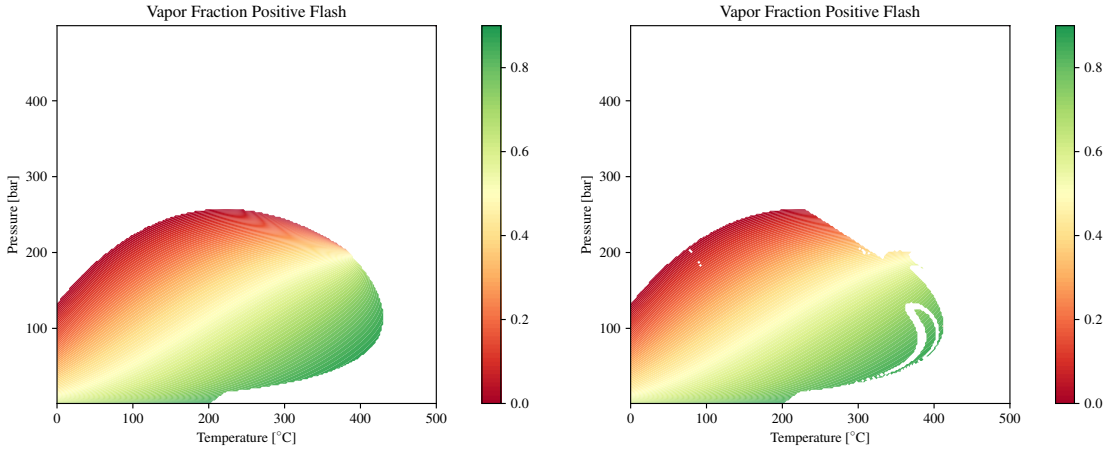
(c) GDEM.

(d) Wegstein's method

Figure 4.3: Calculated molar vapor fraction of composition 7 using 35-Component inside the two-phase region using different successive substitution methods. Note: Blank regions denote trivial solution or negative flash region.

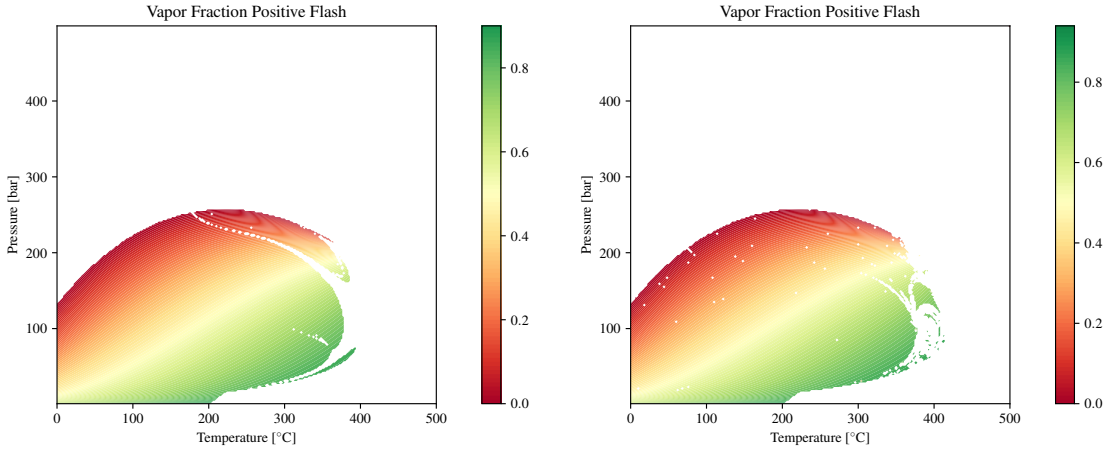
Figures 4.3 and 4.4 shows that the robustness of the successive substitution method is lost when applying acceleration methods. The DEM acceleration has the most severe loss of robustness. The GDEM and Wegstein's method have comparable *area of convergence*,

meaning that a similar amount of the total flash calculations performed converges to non-trivial solutions. However, the GDEM has a more continuous area of convergence and is therefore more predictable than the Wegstein's method.



(a) Successive Substitution.

(b) DEM.

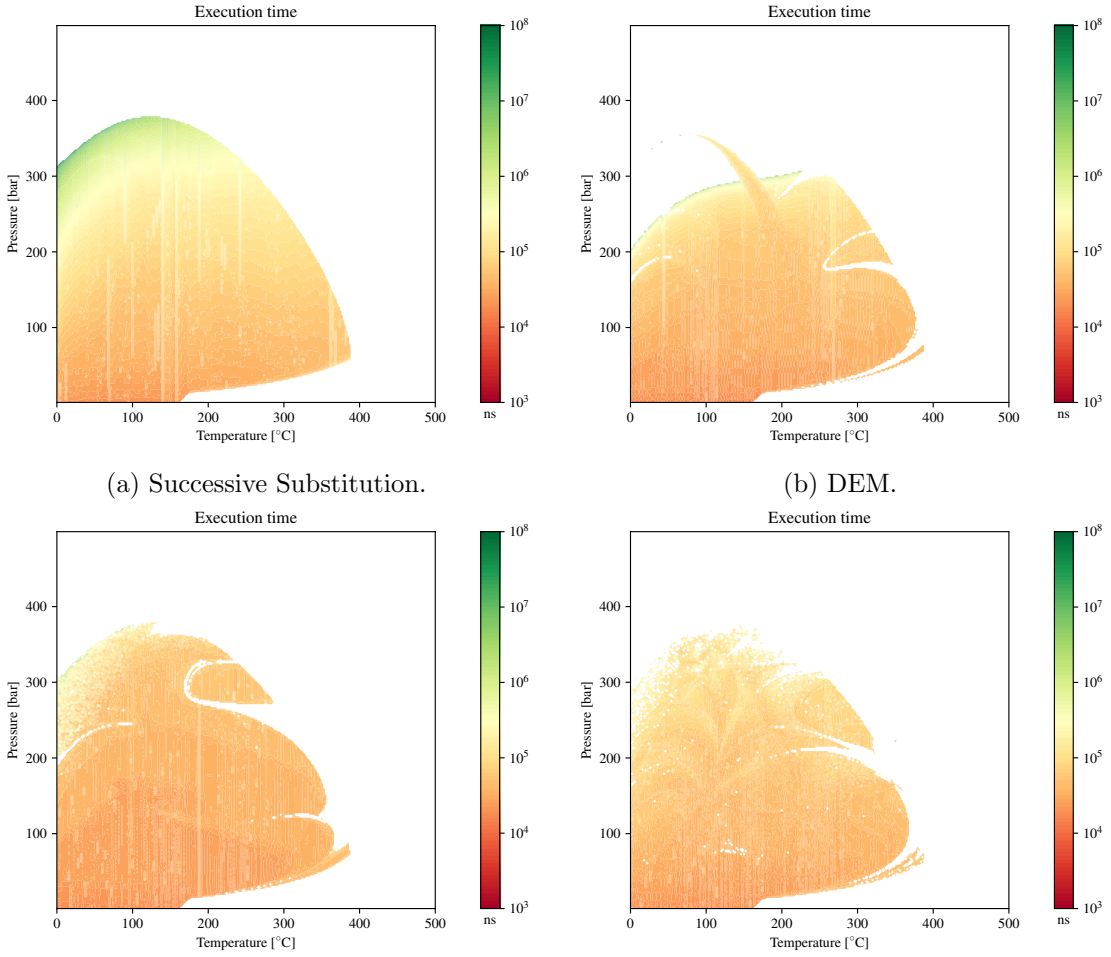


(c) GDEM.

(d) Wegstein's method

Figure 4.4: Calculated molar vapor fraction of composition 3 using 14-Component inside the two-phase region using different successive substitution methods. Note: Blank regions denote trivial solution or negative flash region.

As expected based on the extensive literature, the successive substitution method is slower than all the acceleration methods. The GDEM is the fastest when converging to a non-trivial solution, however, the DEM and Wegstein's method are also comparable.



(a) Successive Substitution.

(b) DEM.

(c) GDEM.

(d) Wegstein's method

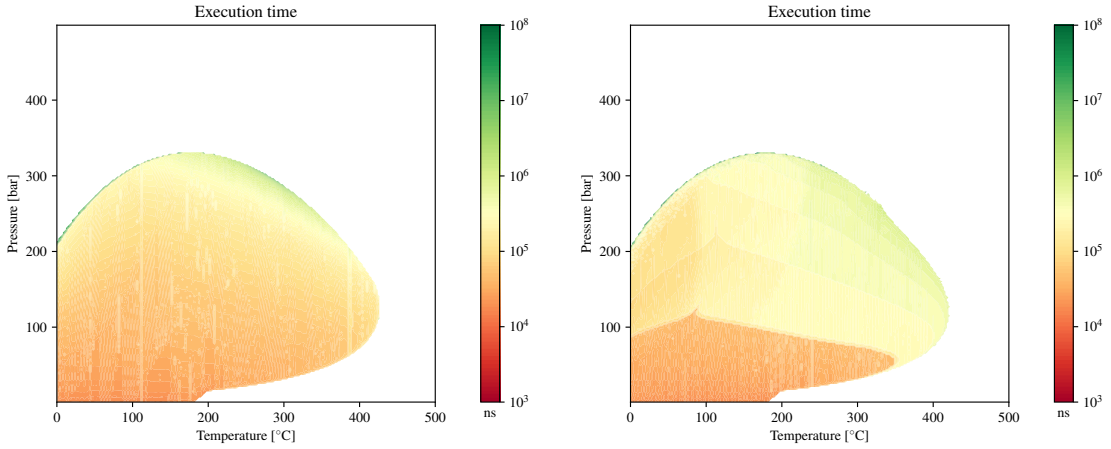
Figure 4.5: Execution time of flash calculations using composition 7 and the 35-Component inside the two-phase region using different successive substitution methods. Note: Blank regions denote trivial solution or negative flash region.

4.4 The Newton-Raphson Method

The Newton-Raphson K-value update method was tested and compared to the successive substitution method without acceleration. The different cases use an increasing amount of successive substitution steps before initiating the Newton-Raphson method. The number of successive substitution steps was set to 20, 30, 40, and 50.

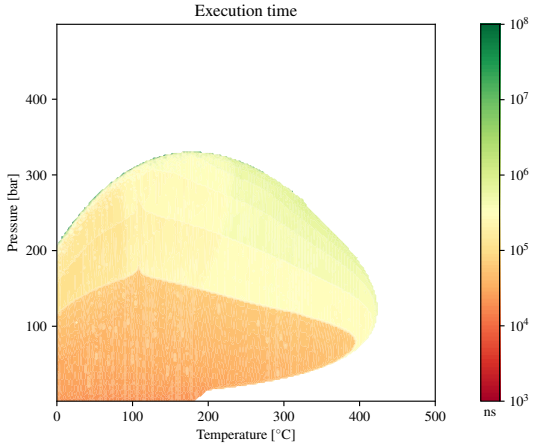
Table 4.3: Total execution time the Newton-Raphson method preceded by a number of successive substitutions iterations.

	Execution Time (ms)				Flashes per second
SS iterations	35C	14C	9C	Total	Total
Only SS	66103	19503	12322	97929	6259
20	61649	15424	5062	82137	7356
30	58619	14994	5531	79144	7568
40	54035	14696	5756	74488	8007
50	56601	15631	6971	79204	7511

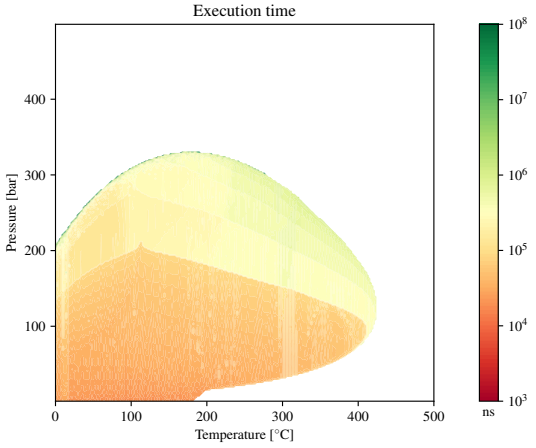


(a) Only successive substitution.

(b) 20 successive substitution steps.



(c) 30 successive substitution steps.



(d) 40 successive substitution steps.

Figure 4.6: Newton-Raphson K-value update method using different numbers of initial successive substitution steps.

Figure 4.6 shows that the Newton-Raphson method is as robust as the successive substitution method.

Table 4.3 shows that the Newton-Raphson method decreases execution time compared to the successive substitution method. Comparing Figures 4.5 and 4.6, the GDEM is faster than the Newton-Raphson method for most cases where the GDEM converges. Table 4.3 shows that there exist an optimal number of successive substitution steps between 30 and 50. Table 4.3 also shows that the reduction in execution time for the Newton-Raphson method significantly increases as the number of components decreases. This is as expected because the number of operations in the Newton-Raphson method is quadratically proportional to the number of components.

CHAPTER 5

CONCLUSION

1. A framework for developing a performant implementation of the two-phase isothermal flash calculation was proposed. Aspects of modern computer architecture and programming are discussed, such as cache locality, branching, compiled versus interpreted programming languages and profiling.
2. A novel method for solving for the roots of the cubic equation of state was developed and was shown to have a better performance than Deiters' numerical method and Michelsen's hybrid method.
3. A novel method for solving the Rachford-Rice equation without severe round-off errors was proposed and was shown to have a better performance than the conventional Newton-Raphson method and Halley's method.
4. The DEM, GDEM and Wegstein's method were implemented and was shown to increase performance, but was also shown to be less robust than the standard successive substitution method. Of the three acceleration methods, the GDEM was shown to have the best overall performance.
5. The Newton-Raphson K-value update method was implemented and shown to be as robust as the successive substitution method with a better performance.
6. No conclusive results were found to show that accelerated successive substitution or the Newton-Raphson method is superior in all cases. This indicates that a heuristic-based hybrid model will yield the best overall performance.

CHAPTER 6

FURTHER WORK

Based on the results in this thesis, a list of items for further investigation is proposed below.

1. Develop a heuristic-based hybrid algorithm utilizing the best aspects of accelerated successive substitution methods and the Newton-Raphson method.
2. Implement and test the stability analysis calculation and determine how the resulting K-value estimates will impact the performance of the two-phase isothermal flash calculation.
3. Implement and test the second-order Gibbs minimization method developed by Michelsen[20] and compare the results to the equation-based approach proposed in this work.

BIBLIOGRAPHY

[1] Émile Clapeyron. "Mémoire sur la puissance motrice de la chaleur". In: *Journal de l'École polytechnique* 14 (1834), pp. 153–190.

[2] Marshall B Standing and Donald L Katz. "Density of natural gases". In: *Transactions of the AIME* 146.01 (1942), pp. 140–149.

[3] C.H. Whitson and M.R. Brulé. *Phase Behavior*. Henry L. Doherty Series. Henry L. Doherty Memorial Fund of AIME, Society of Petroleum Engineers, 2000. ISBN: 9781555630874.

[4] K. R. Hall and L. Yarborough. "A new equation of state for Z-factor calculations". In: *Oil Gas J* 71 (1973), p. 82.

[5] G. Soave. "Equilibrium constants from a modified Redlich-Kwong equation of state". In: *Chemical engineering science* 27 (1972), pp. 1197–1203. DOI: https://doi.org/10.1016/0009-2509(72)80096-4.

[6] D. B. Robinson and D. Y. Peng. *The characterization of the heptanes and heavier fractions for the GPA Peng-Robinson programs*. Gas processors association, 1978.

[7] A. Péneloux, E. Rauzy, and R. Fréze. "A consistent correction for Redlich-Kwong-Soave volumes". In: *Fluid phase equilibria* 8 (1982), pp. 7–23. DOI: https://doi.org/10.1016/0378-3812(82)80002-2.

[8] M.L. Michelsen and J.M. Mollerup. *Thermodynamic Models: Fundamentals & Computational Aspects*. Tie-Line Publications, 2007. ISBN: 9788798996132. URL: https://books.google.no/books?id=qjmeOgAACAAJ.

[9] Keith H Coats. "Simulation of gas condensate reservoir performance". In: *Journal of Petroleum Technology* 37.10 (1985), pp. 1870–1886.

[10] Curtis H Whitson and Michael L Michelsen. "The negative flash". In: *Fluid phase equilibria* 53 (1989), pp. 51–71. DOI: https://doi.org/10.1016/0378-3812(89)80072-X.

[11] Dan Vladimir Nichita and Claude F Leibovici. "A rapid and robust method for solving the Rachford–Rice equation using convex transformations". In: *Fluid Phase*

Equilibria 353 (2013), pp. 38–49. DOI: https://doi.org/10.1016/j.fluid.2013.05.030.

[12] David Goldberg. "What every computer scientist should know about floating-point arithmetic". In: *ACM computing surveys (CSUR)* 23.1 (1991), pp. 5–48.

[13] Nicholas J Higham. *Accuracy and stability of numerical algorithms*. SIAM, 2002.

[14] Ulrich Drepper. "What every programmer should know about memory". In: *Red Hat, Inc* 11 (2007), p. 2007.

[15] Bjarne Stroustrup. *Bjarne Stroustrup: Why you should avoid linked lists*. July 2012. URL: https://www.youtube.com/watch?v=YQs6IC-vgmo&ab_channel=AlessandroStamatto.

[16] Techopedia. *What is a branch? - definition from Techopedia*. Sept. 2016. URL: https://www.techopedia.com/definition/18058/branch.

[17] Ali Danesh. *PVT and phase behaviour of petroleum reservoir fluids*. Elsevier, 1998.

[18] Saul A Teukolsky et al. "Numerical recipes in C". In: *SMR* 693.1 (1992), pp. 59–70.

[19] Ulrich K Deiters. "Calculation of densities from cubic equations of state". In: *AIChE Journal* 48.4 (2002), pp. 882–886.

[20] Michael L Michelsen. "Speeding up the two-phase PT-flash, with applications for calculation of miscible displacement". In: *Fluid Phase Equilibria* 143.1-2 (1998), pp. 1–12.

[21] Michael L Michelsen. "The isothermal flash problem. Part II. Phase-split calculation". In: *Fluid phase equilibria* 9.1 (1982), pp. 21–40.

[22] Rakesh K Mehra, Robert A Heidemann, and Khalid Aziz. "An accelerated successive substitution algorithm". In: *The Canadian Journal of Chemical Engineering* 61.4 (1983), pp. 590–596.

[23] MN Ammar and H Renon. "The isothermal flash problem: new methods for phase split calculations". In: *AIChE journal* 33.6 (1987), pp. 926–939.

[24] YS Teh and GP Rangaiah. "A study of equation-solving and Gibbs free energy minimization methods for phase equilibrium calculations". In: *Chemical Engineering Research and Design* 80.7 (2002), pp. 745–759.

[25] Michael L Michelsen, Wei Yan, and Erling H Stenby. "A comparative study of reduced-variables-based flash and conventional flash". In: *SPE Journal* 18.05 (2013), pp. 952–959.

[26] Grant M Wilson. "A modified Redlich-Kwong equation of state, application to general physical data calculations". In: *65th National AIChE Meeting, Cleveland, OH*. Vol. 15. 1969.

[27] Michael L Michelsen. "The isothermal flash problem. Part I. Stability". In: *Fluid phase equilibria* 9.1 (1982), pp. 1–19.

[28] Lee E Baker, Alan C Pierce, and Kraemer D Luks. "Gibbs energy analysis of phase equilibria". In: *Society of Petroleum Engineers Journal* 22.05 (1982), pp. 731–742.

[29] M Muskat and JM McDowell. "An electrical computer for solving phase equilibrium problems". In: *Journal of Petroleum Technology* 1.11 (1949), pp. 291–298. DOI: https://doi.org/10.2118/949291-G.

[30] Henry H Rachford and JD Rice. "Procedure for use of electronic digital computers in calculating flash vaporization hydrocarbon equilibrium". In: *Journal of Petroleum Technology* 4.10 (1952), pp. 19–3. DOI: https://doi.org/10.2118/952327-G.

[31] Curtis Whitson. *The Rachford-Rice Contest*. 1995. URL: http://www.ipt.ntnu.no/~curtis/courses/Rachford-Rice-Contest/RRcontest.pdf (visited on 12/07/2021).

[32] *A New Generation EOS Compositional Reservoir Simulator: Part I - Formulation and Discretization*. Vol. All Days. SPE Reservoir Simulation Conference. SPE-37979-MS. June 1997. DOI: 10.2118/37979-MS. eprint: https://onepetro.org/spersc/proceedings-pdf/97RSS/All-97RSS/SPE-37979-MS/1940265/spe-37979-ms.pdf. URL: https://doi.org/10.2118/37979-MS.

[33] Cameron M Crowe and Masatoshi Nishio. "Convergence promotion in the simulation of chemical processes—the general dominant eigenvalue method". In: *AIChE Journal* 21.3 (1975), pp. 528–533.

[34] Jorgen M Mollerup and Michael L Michelsen. "Calculation of thermodynamic equilibrium properties". In: *Fluid phase equilibria* 74 (1992), pp. 1–15.

[35] Aravindh Krishnamoorthy and Deepak Menon. "Matrix inversion using Cholesky decomposition". In: *2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*. IEEE. 2013, pp. 70–72.

[36] Bilal Younus. *Fluid definition module*. URL: https://manual.whitson.com/modules/fluid-definition/#initial-gor-and-stock-tank-api.

ACRONYMS

BIPs	Binary Interaction Parameters.
DEM	Dominant Eigenvalue Method.
EOS	Equation of State.
GDEM	General Dominant Eigenvalue Method.
PR	Peng-Robinson.
SRK	Soave-Redlich-Kwing.

Appendices

APPENDIX A

LOWEST GIBBS ENERGY CONDITION

When solving the Z-factor equation given by Equation (2.5), multiple solutions can exists. The correct solution is the Z-factor with the lowest Gibbs energy, which can be determined using the following equation

$$G(Z_1, \mathbf{u}, p) < G(Z_2, \mathbf{u}, p), \tag{A.1}$$

where Z_1 and Z_2 are two different solutions of Equation (2.5) and

$$G(Z, \mathbf{u}, p) = RT \sum_{i=1}^{N_c} u_i \ln(f_i(Z, \mathbf{u}, p)), \tag{A.2}$$

where $f_i(Z, \mathbf{u}, p)$ is the component fugacity given by

$$f_i(Z, \mathbf{u}, p) = \phi_i(Z, \mathbf{u}) \cdot u_i \cdot p. \tag{A.3}$$

Combining Equations A.2 and A.3 results in the following expression

$$G(Z, \mathbf{u}, p) = RT \sum_{i=1}^{N_c} u_i \ln \phi_i(Z, \mathbf{u}) + RT \sum_{i=1}^{N_c} u_i \ln u_i + RT \sum_{i=1}^{N_c} u_i \ln p, \tag{A.4}$$

where $\phi_i(Z, \mathbf{u})$ is calculated by the expression

$$\ln \phi_i(Z, \mathbf{u}) = \frac{B_i}{B}(Z-1) - \ln(Z-B) + \frac{1}{\delta_1 - \delta_2}\frac{A}{B}\left(\frac{B_i}{B} - \frac{2}{A}\sum_{j=1}^{N_c} u_j A_{ij}\right) \ln\left(\frac{Z+\delta_1 B}{Z+\delta_2 B}\right). \tag{A.5}$$

Because $\sum_{i=1}^{N_c} u_i = 1$, $\sum_{i=1}^{N_c} u_i \frac{B_i}{B} = 1$, and $\sum_{i=1}^{N_c} u_i \sum_{j=1}^{N_c} u_j \frac{A_{ij}}{A} = 1$ it can be shown that

$$\sum_{i=1}^{N_c} u_i \ln \phi_i(Z, \mathbf{u}) = Z - 1 - \ln(Z-B) - \frac{1}{\delta_1 - \delta_2}\frac{A}{B} \ln\left(\frac{Z+\delta_1 B}{Z+\delta_2 B}\right) = \ln \bar{\phi}(Z), \tag{A.6}$$

where $\bar{\phi}(Z)$ is the function for the pure-component fugacity, which means that Equation (A.4) can be expressed as

$$G(Z, \mathbf{u}, p) = RT \ln \bar{\phi}(Z) + RT \sum_{i=1}^{N_c} u_i \ln u_i + RT \sum_{i=1}^{N_c} u_i \ln p. \tag{A.7}$$

If a constant composition and pressure is given, Equation (A.7) can be expressed as

$$G(Z, \mathbf{u}, p) = RT \ln \bar{\phi}(Z) + RT \cdot C(\mathbf{u}, p). \tag{A.8}$$

Because the term $C(\mathbf{u}, p)$ only varies with composition and pressure and $RT > 0$, an equivalent condition for Equation (A.1) is therefore

$$\ln \bar{\phi}(Z_1) < \ln \bar{\phi}(Z_2). \tag{A.9}$$

APPENDIX B

EOS MODELS

The following sections give the component properties and binary interaction parameters for the 35-component, 14-component, and 9-component PR EOS models. The 35-component EOS is the parent model from which the 14- and 9-component EOS models were lumped from using the PhazeComp default lumping procedure.

B.1 35-component EOS model

Table B.1: Component properties for the 35-component PR EOS model.

Component	MW	p_c (bar)	T_c (K)	ω	s
N2	28.01400	33.98012	126.20000	0.037000	-0.167580
CO2	44.01000	73.74011	304.12222	0.225000	0.001910
H2S	34.08200	89.62977	373.40000	0.090000	-0.044700
C1	16.04300	45.99010	190.56111	0.011000	-0.149960
C2	30.07000	48.71973	305.32222	0.099000	-0.062800
C3	44.09700	42.47998	369.82778	0.152000	-0.063810
i-C4	58.12300	36.40018	407.85000	0.186000	-0.061970
n-C4	58.12300	37.95977	425.12222	0.200000	-0.053930
i-C5	72.15000	33.80982	460.38889	0.229000	-0.056460
n-C5	72.15000	33.70019	469.70000	0.252000	-0.029270
C6	84.02100	34.22764	514.62222	0.235610	-0.030240
C7	97.79100	31.74415	550.36667	0.270550	-0.017110
C8	111.72700	29.26962	581.42222	0.309140	-0.001000
C9	125.70600	27.05985	609.00000	0.348500	0.015340
C10	139.69500	25.12794	633.78889	0.387890	0.031050
C11	153.68400	23.44217	656.28333	0.427050	0.045820
C12	167.66900	21.96669	676.82778	0.465860	0.059490
C13	181.65100	20.67048	695.70556	0.500430	0.072000
C14	195.62800	19.52664	713.14444	0.536440	0.083320
C15	209.60100	18.51242	729.32778	0.571690	0.093460
C16	223.57000	17.60921	744.40000	0.606170	0.102460
C17	237.53700	16.80252	758.50000	0.639880	0.110370
C18	251.50000	16.07926	771.72778	0.672810	0.117230
C19	265.46100	15.42840	784.17222	0.704960	0.123110
C20	279.42000	14.84027	795.92222	0.736350	0.128080
C21	293.37700	14.30731	807.04444	0.766980	0.132190
C22	307.33300	13.82399	817.58889	0.796860	0.135510
C23	321.28800	13.38272	827.61667	0.826000	0.138100
C24	335.24200	12.98007	837.17222	0.854420	0.140010
C25	349.19600	12.61051	846.28889	0.882120	0.141290
C26	363.15000	12.27129	855.00556	0.909130	0.142000
C27	377.10300	11.95896	863.36111	0.935450	0.142180
C28	391.05600	11.67075	871.37222	0.961100	0.141870
C29	405.01000	11.40393	879.06667	0.986090	0.141120
C30+	550.00000	9.50235	944.87222	1.210620	0.114130

Table B.2: Component BIPs for the 35-component PR EOS model.

	N2	CO2	H2S	C1	C2	C3	i-C4	n-C4	i-C5	n-C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30+
N2	0.00	0.00	0.13	0.03	0.01	0.09	0.10	0.10	0.10	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11	0.11
CO2	0.00	0.00	0.14	0.11	0.13	0.13	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.12
H2S	0.13	0.14	0.00	0.07	0.09	0.08	0.08	0.08	0.07	0.07	0.06	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
C1	0.03	0.11	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.03	0.03	0.04	0.04	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.07	0.07	0.08	0.08	0.08	0.08	0.08	0.09	0.09	0.10
C2	0.01	0.13	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C3	0.09	0.13	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i-C4	0.10	0.12	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
n-C4	0.10	0.12	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i-C5	0.10	0.12	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
n-C5	0.11	0.12	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C6	0.11	0.12	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C7	0.11	0.12	0.05	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C8	0.11	0.12	0.05	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C9	0.11	0.12	0.05	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C10	0.11	0.12	0.05	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C11	0.11	0.12	0.05	0.04	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C12	0.11	0.12	0.05	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C13	0.11	0.12	0.05	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C14	0.11	0.12	0.05	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C15	0.11	0.12	0.05	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C16	0.11	0.12	0.05	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C17	0.11	0.12	0.05	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C18	0.11	0.12	0.05	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C19	0.11	0.12	0.05	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C20	0.11	0.12	0.05	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C21	0.11	0.12	0.05	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C22	0.11	0.12	0.05	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C23	0.11	0.12	0.05	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C24	0.11	0.12	0.05	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C25	0.11	0.12	0.05	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C26	0.11	0.12	0.05	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C27	0.11	0.12	0.05	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C28	0.11	0.12	0.05	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C29	0.11	0.12	0.05	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C30+	0.11	0.12	0.05	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

B.2 14-component EOS model

Table B.3: Component properties for the 14-component PR EOS model.

Component	MW	p_c (bar)	T_c (K)	ω	s
N2	28.014	33.9801	126.200	0.03700	-0.16758
CO2	44.010	73.7401	304.122	0.22500	0.00191
H2S	34.082	89.6298	373.400	0.09000	-0.04470
C1	16.043	45.9901	190.561	0.01100	-0.14996
C2	30.070	48.7197	305.322	0.09900	-0.06280
C3	44.097	42.4800	369.828	0.15200	-0.06381
C4	58.123	37.6661	421.872	0.19736	-0.05545
C5	72.150	33.7552	465.334	0.24112	-0.04201
C6-C7	88.943	33.2777	527.723	0.24831	-0.02511
C8-C10	122.388	27.5105	602.370	0.33974	0.01259
C11-C14	173.003	21.4311	683.873	0.48103	0.06507
C15-C19	234.953	16.9440	755.671	0.63385	0.10890
C20-C29	334.713	13.0230	836.308	0.85238	0.13793
C30+	550.000	9.5023	944.872	1.21062	0.11413

Table B.4: Component BIPs for the 14-component PR EOS model.

	N2	CO2	H2S	C1	C2	C3	C4	C5	C6-C7	C8-C10	C11-C14	C15-C19	C20-C29	C30+
N2	0.00000	0.00000	0.13000	0.03000	0.01000	0.09000	0.10000	0.10532	0.11000	0.11000	0.11000	0.11000	0.11000	0.11000
CO2	0.00000	0.00000	0.14000	0.11000	0.13000	0.13000	0.12000	0.12000	0.12000	0.12000	0.12000	0.12000	0.12000	0.12000
H2S	0.13000	0.14000	0.00000	0.07000	0.09000	0.08000	0.08000	0.07000	0.05612	0.05000	0.05000	0.05000	0.05000	0.05000
C1	0.03000	0.11000	0.07000	0.00000	0.00000	0.00000	0.00000	0.00000	0.01163	0.03265	0.04734	0.06388	0.07845	0.10000
C2	0.01000	0.13000	0.09000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C3	0.09000	0.13000	0.08000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C4	0.10000	0.12000	0.08000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C5	0.10532	0.12000	0.07000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C6-C7	0.11000	0.12000	0.05612	0.01163	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C8-C10	0.11000	0.12000	0.05000	0.03265	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C11-C14	0.11000	0.12000	0.05000	0.04734	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C15-C19	0.11000	0.12000	0.05000	0.06388	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C20-C29	0.11000	0.12000	0.05000	0.07845	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C30+	0.11000	0.12000	0.05000	0.10000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

B.3 9-component EOS model

Table B.5: Component properties for the 9-component PR EOS model.

Component	MW	p_c (bar)	T_c (K)	ω	s
H2S	34.082	89.6298	373.400	0.09000	-0.04470
N2-C1	16.045	45.9876	190.547	0.01100	-0.14996
C2-CO2	30.079	48.7254	305.289	0.09903	-0.06277
C3	44.097	42.4800	369.828	0.15200	-0.06381
C4	58.123	37.6661	421.872	0.19736	-0.05545
C5	72.150	33.7552	465.334	0.24112	-0.04201
C6-C9	99.086	31.2513	552.116	0.27667	-0.01197
C10-C29	229.682	17.2362	746.545	0.62281	0.10424
C30+	550.000	9.5023	944.872	1.21062	0.11413

Table B.6: Component BIPs for the 9-component PR EOS model.

	H2S	N2-C1	C2-CO2	C3	C4	C5	C6-C9	C10-C29	C30+
H2S	0.00000	0.07000	0.08996	0.08000	0.08000	0.07000	0.05355	0.05000	0.05000
N2-C1	0.07000	0.00000	-0.00001	0.00001	0.00001	0.00001	0.01934	0.06354	0.10000
C2-CO2	0.08996	-0.00001	0.00000	0.00000	-0.00001	-0.00001	-0.00001	-0.00001	-0.00001
C3	0.08000	0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C4	0.08000	0.00001	-0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C5	0.07000	0.00001	-0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C6-C9	0.05355	0.01934	-0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C10-C29	0.05000	0.06354	-0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
C30+	0.05000	0.10000	-0.00001	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

APPENDIX C

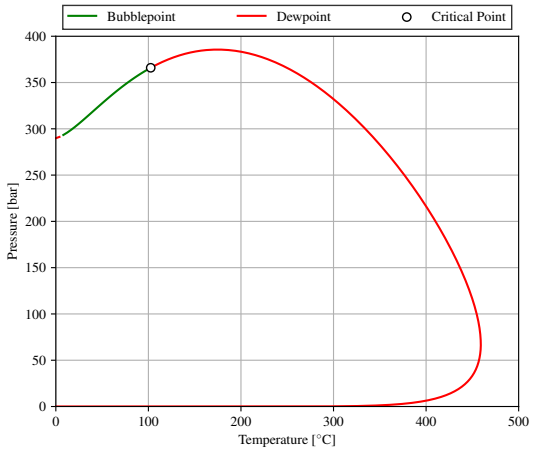
COMPOSITIONS

The following section contains the different molar compositions used in this work and their respective phase envelopes generated using the equations of state models given in Appendix B. The compositions were generated using a GOR recombination method in whitson+.

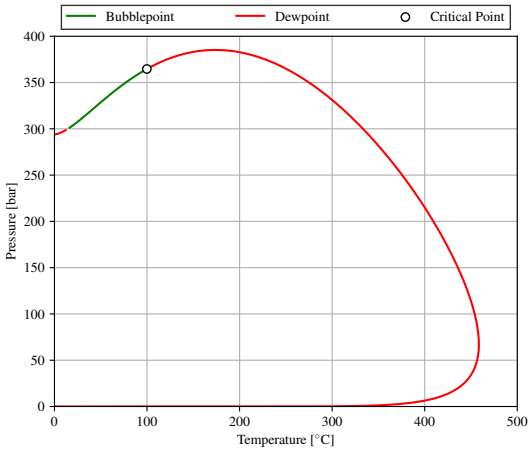
C.1 Composition 1

Table C.1: Composition 1

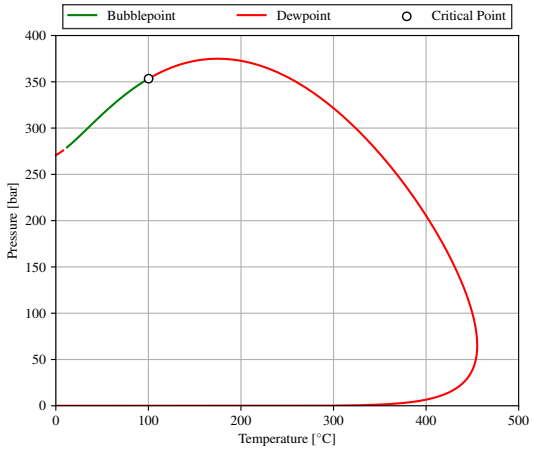
Component	z_i	Component	z_i
N2	0.000100	C14	0.002782
CO2	0.000100	C15	0.002635
H2S	0.000010	C16	0.002099
C1	0.527534	C17	0.001963
C2	0.151570	C18	0.001941
C3	0.115419	C19	0.001754
i-C4	0.013518	C20	0.001441
n-C4	0.058222	C21	0.001344
i-C5	0.016890	C22	0.001238
n-C5	0.018721	C23	0.001148
C6	0.018418	C24	0.001056
C7	0.010245	C25	0.001000
C8	0.009987	C26	0.000943
C9	0.006180	C27	0.000902
C10	0.004967	C28	0.000852
C11	0.003845	C29	0.000803
C12	0.003105	C30+	0.014043
C13	0.003226		



(a) 35-component EOS.



(b) 14-component EOS.



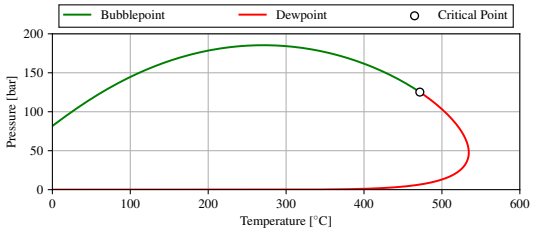
(c) 9-component EOS.

Figure C.1: Calculated phase envelope of composition 1.

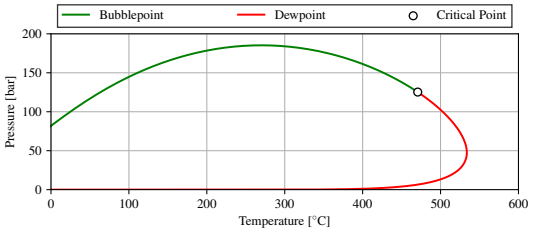
C.2 Composition 2

Table C.2: Composition 2

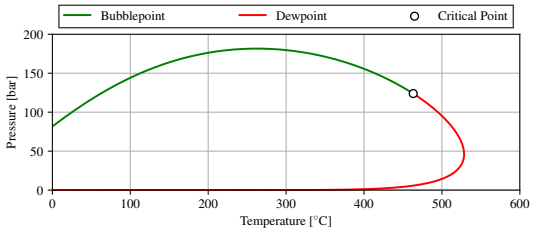
Component	z_i	Component	z_i
N2	0.000000	C14	0.014193
CO2	0.000000	C15	0.013463
H2S	0.000000	C16	0.010738
C1	0.302785	C17	0.010051
C2	0.090750	C18	0.009948
C3	0.075527	C19	0.009004
i-C4	0.010212	C20	0.007403
n-C4	0.049152	C21	0.006910
i-C5	0.020518	C22	0.006375
n-C5	0.026113	C23	0.005913
C6	0.041915	C24	0.005449
C7	0.035526	C25	0.005164
C8	0.043930	C26	0.004876
C9	0.029884	C27	0.004666
C10	0.024837	C28	0.004410
C11	0.019454	C29	0.004163
C12	0.015783	C30+	0.074457
C13	0.016434		



(a) 35-component EOS.



(b) 14-component EOS.



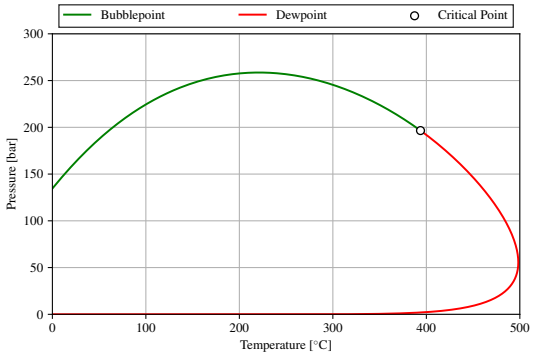
(c) 9-component EOS.

Figure C.2: Calculated phase envelope of composition 2.

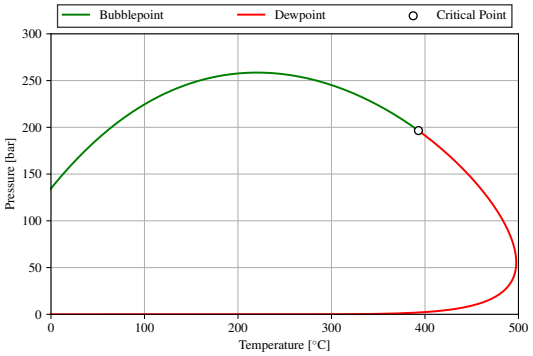
C.3 Composition 3

Table C.3: Composition 3

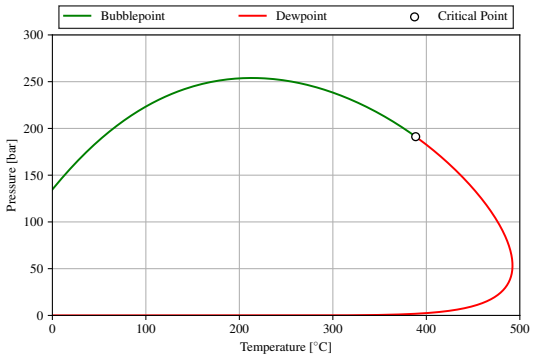
Component	z_i (%)	Component	z_i (%)
N2	0.000000	C14	0.011165
CO2	0.000000	C15	0.010414
H2S	0.000000	C16	0.008171
C1	0.426327	C17	0.007532
C2	0.104258	C18	0.007334
C3	0.075707	C19	0.006528
i-C4	0.010628	C20	0.005283
n-C4	0.040873	C21	0.004859
i-C5	0.015399	C22	0.004415
n-C5	0.019543	C23	0.004036
C6	0.027027	C24	0.003665
C7	0.031248	C25	0.003422
C8	0.038117	C26	0.003183
C9	0.025477	C27	0.003001
C10	0.020820	C28	0.002793
C11	0.016045	C29	0.002595
C12	0.012823	C30+	0.034167
C13	0.013145		



(a) 35-component EOS.



(b) 14-component EOS.



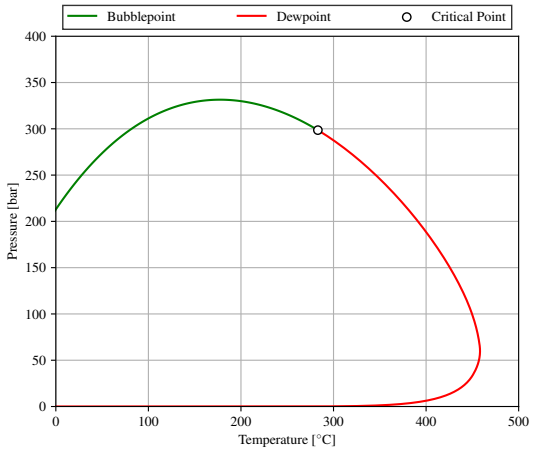
(c) 9-component EOS.

Figure C.3: Calculated phase envelope of composition 3.

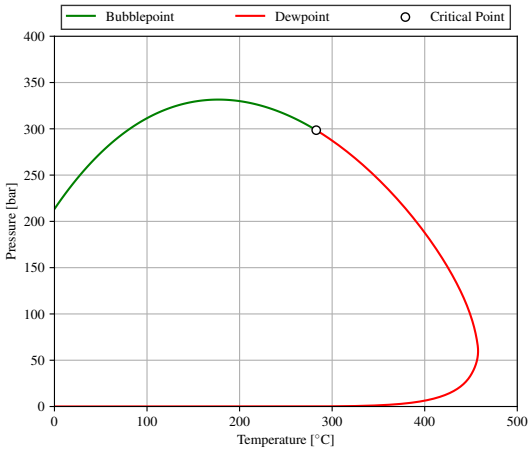
C.4 Composition 4

Table C.4: Composition 4

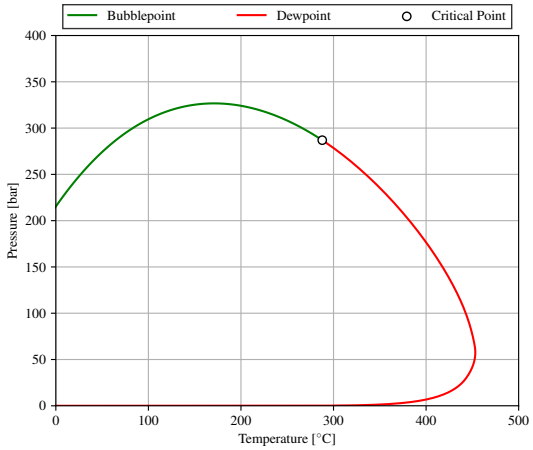
Component	z_i	Component	z_i
N2	0.000000	C14	0.007698
CO2	0.000000	C15	0.007035
H2S	0.000000	C16	0.005411
C1	0.542823	C17	0.004894
C2	0.110948	C18	0.004673
C3	0.071655	C19	0.004075
i-C4	0.010374	C20	0.003234
n-C4	0.032767	C21	0.002921
i-C5	0.011415	C22	0.002608
n-C5	0.014463	C23	0.002340
C6	0.017561	C24	0.002088
C7	0.024758	C25	0.001914
C8	0.029652	C26	0.001749
C9	0.019383	C27	0.001619
C10	0.015514	C28	0.001478
C11	0.011720	C29	0.001348
C12	0.009197	C30+	0.013434
C13	0.009249		



(a) 35-component EOS.



(b) 14-component EOS.



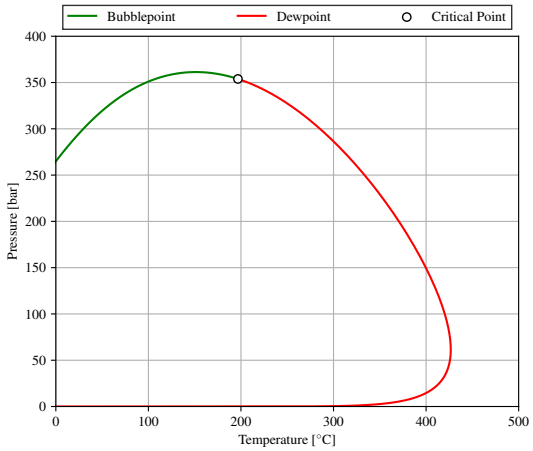
(c) 9-component EOS.

Figure C.4: Calculated phase envelope of composition 4.

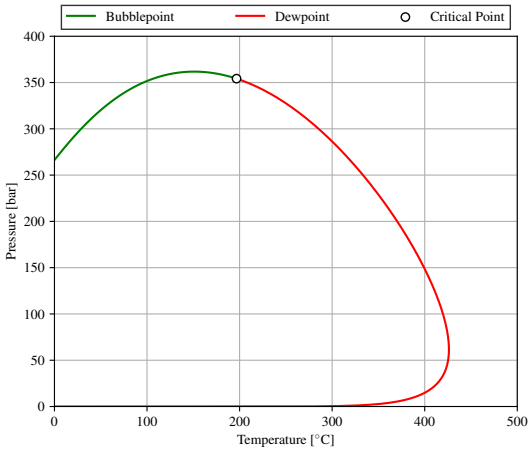
C.5 Composition 5

Table C.5: Composition 5

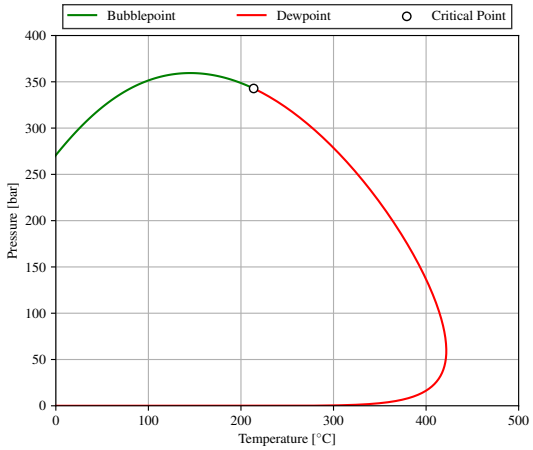
Component	z_i	Component	z_i
N2	0.000000	C14	0.006006
CO2	0.000000	C15	0.005380
H2S	0.000000	C16	0.004060
C1	0.606607	C17	0.003606
C2	0.109830	C18	0.003379
C3	0.065728	C19	0.002888
i-C4	0.009769	C20	0.002251
n-C4	0.026984	C21	0.001997
i-C5	0.009118	C22	0.001751
n-C5	0.011629	C23	0.001545
C6	0.013326	C24	0.001354
C7	0.021504	C25	0.001221
C8	0.025736	C26	0.001096
C9	0.016573	C27	0.000997
C10	0.013030	C28	0.000894
C11	0.009663	C29	0.000800
C12	0.007451	C30+	0.006468
C13	0.007358		



(a) 35-component EOS.



(b) 14-component EOS.



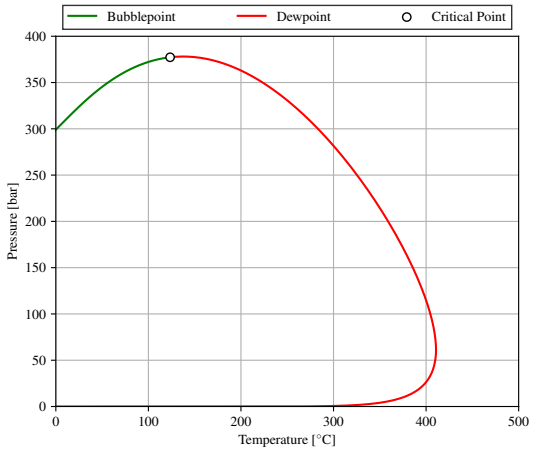
(c) 9-component EOS.

Figure C.5: Calculated phase envelope of composition 5.

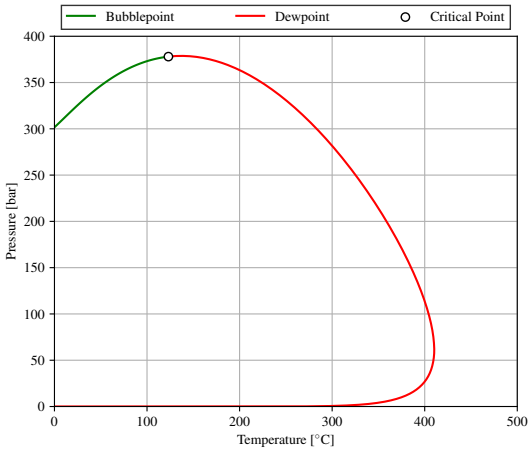
C.6 Composition 6

Table C.6: Composition 6

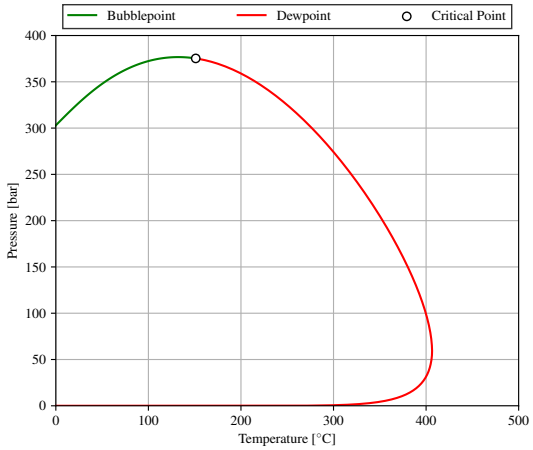
Component	z_i	Component	z_i
N2	0.000000	C14	0.004803
CO2	0.000000	C15	0.004269
H2S	0.000000	C16	0.003198
C1	0.638750	C17	0.002820
C2	0.111245	C18	0.002621
C3	0.064832	C19	0.002224
i-C4	0.009635	C20	0.001720
n-C4	0.025483	C21	0.001516
i-C5	0.008379	C22	0.001320
n-C5	0.010620	C23	0.001156
C6	0.011556	C24	0.001007
C7	0.018642	C25	0.000901
C8	0.021786	C26	0.000804
C9	0.013814	C27	0.000726
C10	0.010748	C28	0.000646
C11	0.007904	C29	0.000574
C12	0.006050	C30+	0.004320
C13	0.005931		



(a) 35-component EOS.



(b) 14-component EOS.



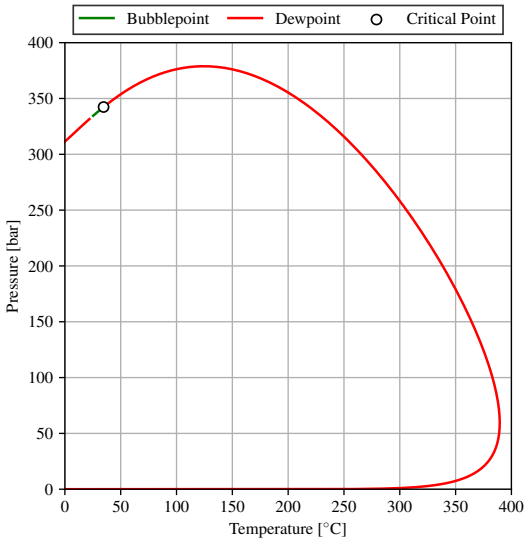
(c) 9-component EOS.

Figure C.6: Calculated phase envelope of composition 6.

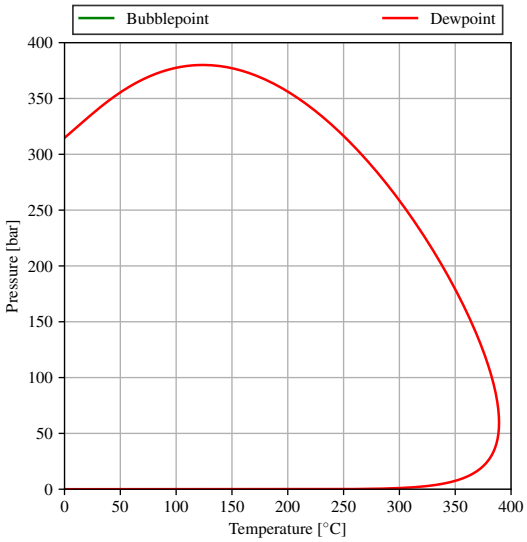
C.7 Composition 7

Table C.7: Composition 7

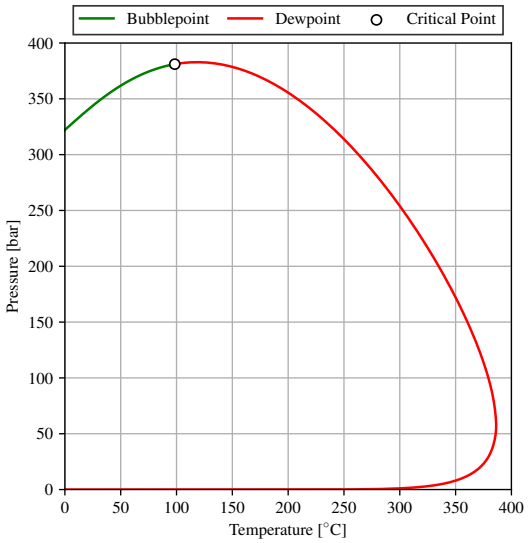
Component	z_i	Component	z_i
N2	0.000000	C14	0.004103
CO2	0.000000	C15	0.003581
H2S	0.000000	C16	0.002635
C1	0.669039	C17	0.002285
C2	0.108464	C18	0.002088
C3	0.060470	C19	0.001739
i-C4	0.009132	C20	0.001322
n-C4	0.022268	C21	0.001147
i-C5	0.007243	C22	0.000982
n-C5	0.009235	C23	0.000847
C6	0.009832	C24	0.000725
C7	0.017270	C25	0.000639
C8	0.020376	C26	0.000561
C9	0.012805	C27	0.000498
C10	0.009818	C28	0.000436
C11	0.007103	C29	0.000381
C12	0.005351	C30+	0.002470
C13	0.005158		



(a) 35-component EOS.

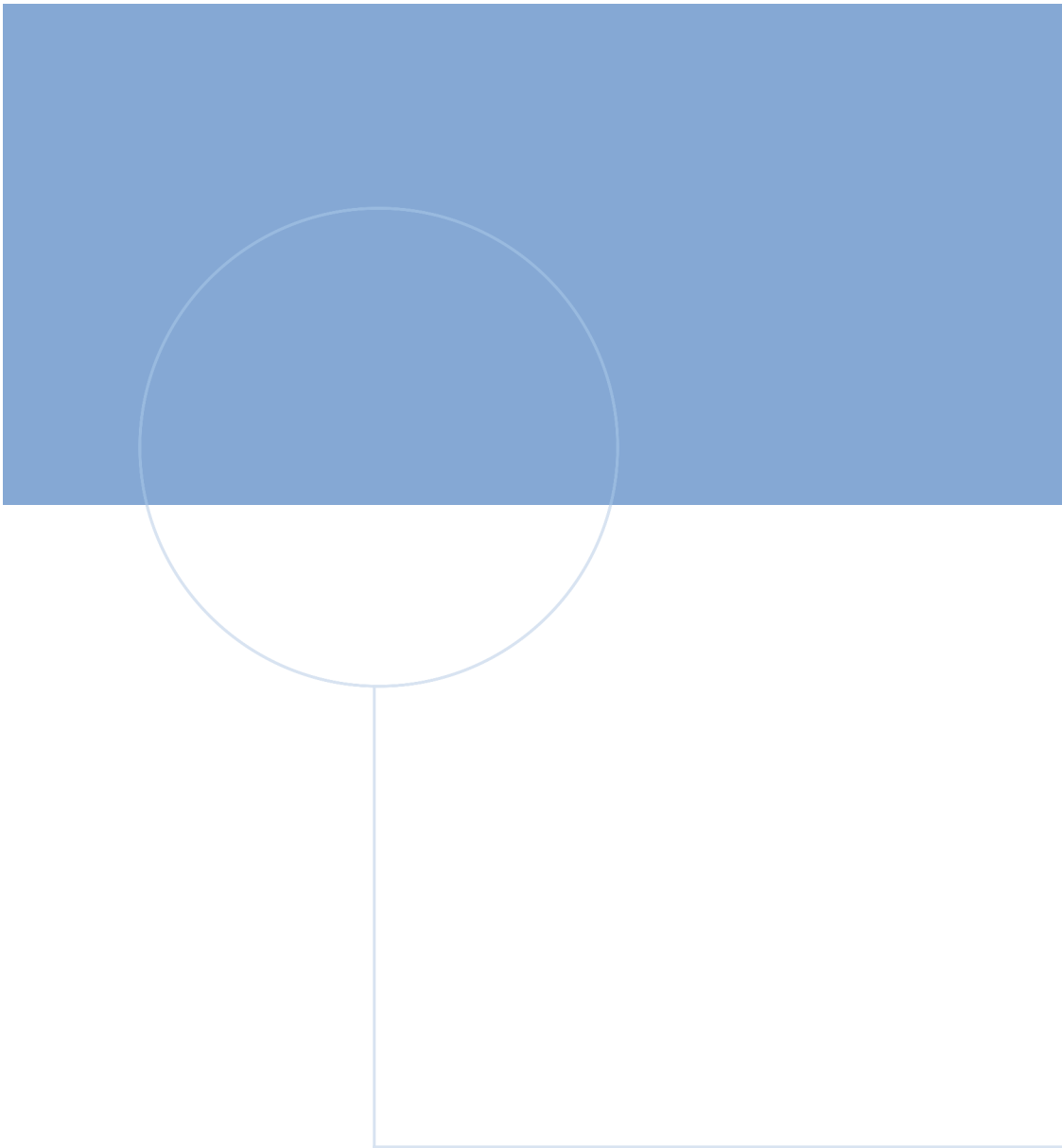


(b) 14-component EOS.



(c) 9-component EOS.

Figure C.7: Calculated phase envelope of composition 7.



NTNU

Norwegian University of
Science and Technology